NUM

MANUEL DE PROGRAMMATION DE LA FONCTION AUTOMATISME LANGAGE LADDER

0100938846/8

11 - 97 **fr-938846/8**

Table des matières

Tableau des mises à jour

Date	Indice	Pages modifiées	Pages ajoutées	Pages supprimées
11 - 97	8	Page de garde, 2, 3, 7, 10 Ch. 2 : 13 Ch. 3 : 1 à 4, 17, 27, 34, 37 à 78 Ch. 5 : 13 Ch. 8 : 3, 6 Ch. 9 : 3 Ch. 12 : 5 Ch. 15 : 8, 25 Ch. 17 : 1 Index : 1 à 4 Agences Questionnaire	79 à 82	

EVOLUTIONS DE LA DOCUMENTATION

Date	Indice	Nature des évolutions
07 - 92	0	Conforme au logiciel NUM 1060 - Indice D. Création du document.
10 - 92	1	Conforme au logiciel NUM 1060 - Indice D Corrections diverses
		La variable %R1.B devient %R0.W La variable %Rg1F.B devient %Rg1E.W La variable %Rg7E.W devient %Rg7C.L Adjonction des variables %R2.7, %R2.6, %W3.7, %W3.6 Supression des variables %W15.B et %W16.B Adjonction des fonctions call(), goto(), R_E42000(), W_E42000() Modification des tableaux de caractères en mode transparent Modes opératoires de l'utilitaire 7 Listes des fonctions en annexe.
04 - 93	2	Conforme au logiciel NUM1060 - Indice E. Corrections diverses. Adjonction des variables : - mots communs %S, - locales %Y, - %Qrc3B.1 autorisation accés CN, - %R2.5 E_INTERV Etat intervention, - %W5.6 INIB_E33 Autorisation d'écriture des cartes Entrées/Sorties par programme pièce

Date	Indice	Nature des évolutions
04 - 93	2	- %W4.4 PRESPUIS Présence puissance sur moteur, - %W15.B MSG1 Numéro de message à afficher ligne 1 - %W16.B MSG2 Numéro de message à afficher ligne 2, - %W2C.W Liste de bits - Incréments de JOG interdits, - %W30.L Liste de bits - Modes interdits, - %Rg01.0 E_RAZ1 à E_RAZ8 Raz en cours sur groupe N°g - %Rg01.4 E_DGURG1 à E_DGURG8 Dégagement d'urgence sur groupe N°g, - %Wg01.4 C_DGURG1 à C_DGURG8 Demande de dégagement d'urgence sur groupe N°g Contact de test à 1 d'une liste de bits. Contact de test à 0 d'une liste de bits. Contact de test sur front montant. Contact de test sur front descendant. Actions conditionnelles en zone test. Affectations numériques multiples sur bobines T et F. Appel d'un sous programme avec variables locales %Y - Fonction spy() Initialisation de la base associée aux variables %Y - Fonction y_init(). Initialisation graphique - Fonction inig(). Emission d'une requête vers un serveur distant - Fonction neto(). Lecture d'une requête venant d'un serveur distant - Fonction neti(). Configuration du service mots communs - Fonction setcomw(). Réponse à la requête STATUS - Fonction netst_ad() Fonction de programmation en C de la bibliothèque NUM (NUM.OBJ). Archivage du logiciel sous UT7.
02-94	3	Conforme au logiciel NUM1060 - Indice F. Corrections diverses et compléments d'informations Prise en compte du module UCSII (Temps CPU, Restrictions matérielles, .. etc ...) Mnémonique sur 12 caractères. Suppression de %R3.5, E_STOP. Adjonction des variables : - %R19.B ,ID_ICB_CN, Identificateur pupitre ou CN actif - %W2.0, KB_INIT, Initialisation clavier - %W4.6, INIBUTIL, Verrouillage des utilitaires - %W5.6, devient SK_DISPL, Affichage de la fenêtre cartouche - %W5.7, SC_SAVE, Mise en veille de l'écran CN - %Rg01.5, NO_POS1 à NO_POS8, Axe en attente de position - %Wg00.6, C_FAST1 à CFAST8, Commande de vitesse rapide en cours de cycle - %W900.0, INIB_E33, Autorisation d'écriture des cartes sorties par programmation pièce Adressage indirect ou par pointeur (Variable %Y) Compteur, CTU_n, CTD_n Temporisation, TOF_n, TON_n, TP_n Cartes 32-24 I/O et 64-48 I/O Suppression de la fonction message() Donnée non sollicitée : - $1, message non bloquant, - $11, message bloquant. Lignes séries - Prise en compte des standard RS232, RS485, RS422.

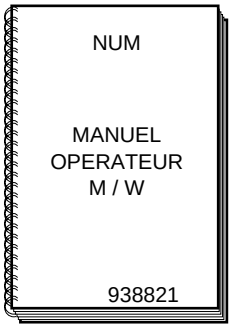
Date	Indice	Nature des évolutions
08 - 94	4	Conforme au logiciel NUM1060 - Indice G. Corrections diverses. Adjonction des variables : - %R2.1, E_MNAUTO, Fonctionnalité N/M AUTO - %W2.1, C_NMAUTO, Fonctionnalité N/M AUTO - %W34.0 à %W37.7, DISC_TRQx, Validation du couple sur l'axe x, - %W38.0, DISC_SDP, Validation référence vitesse des axes QVN, - %R950.B à %R976.W, Consommation moniteur et tâches %TS, - %W97a.L, Type et numéro de tâche en animation ladder, - %W97e.B, Numéro du composant à animer. Prise en compte des nouvelles gravures sur les cartes 32 entrées et 32 sorties. Consommation moniteur et tâches %TS sous UT7. Animation des grilles ladder sous UT7
04 - 95	5	Conforme au logiciel NUM1060 - Indice H Corrections diverses. Doublement du chien de garde Adjonction des variables: - %R2.4, STATETRACE, état recul / retour sur trajectoire, - %R14.0, SC_USED, validation écran en configuration PCNC, - %R22.0 à R22.3, STOPBR1 à STOPBR4, demande d'arrêt des broches 1 à 4, - %W39.0, BACKWARD, demande de recul sur trajectoire, - %W39.1, FORWARD, demande de retour sur trajectoire, - %W39.2, INITPOS, rappel automatique à la suite d'une intervention. Raz des variables sauvegardées.
11 - 95	6	Conforme au logiciel NUM 1020/1040/1060, indice J Corrections diverses Adjonctions de fonctions d'usage général: - transcodage BCD --> binaire -;transcodage binaire --> BCD Adjonction pupitre compact dans chapitre 3 Adjonction segment 235 Adjonction des variables : - %R12.4 à %R12.7, Bx_arr, Broche à l'arrêt - %R12.0 à %R12.3, Bx_ROT, Rotation broche correcte - %R24.L,AXBLKx, Axes blocables - %W3A.L, STOPAXx, Arrêt d'avance par axe
07 - 96	7	Conforme au logiciel NUM 1020/1040/1060, indice K Corrections diverses Adjonctions des variables : - %R14.1, E_BAT, Etat batteries - %W2.2, C_INDG, Commutation groupes communs/groupes indépendants - %W2.3, CHG_OPDC, Opérateurs dynamiques - %W22.4 à %W22.7, VERBRb, Puissance ou non sur broche b - %Rg00.0, E_PROG1 à E_PROG8, Programme en cours du groupe d'axes CN g - %Rg00.5, E_INTER1 à E_INTER8, Etat interventionsur groupe d'axes CN g - %Rg00.6, E_SLASH1 à E_SLASH8, Saut de bloc validé sur groupe d'axes CN g - %Rg00.7, E_M011 à E_M018, Arrêt optionnel validé sur groupes d'axes CN g - %Rg01.1, E_ARUS1 à E_ARUS8, Sortie arrêt usinage du groupe d'axes CN g

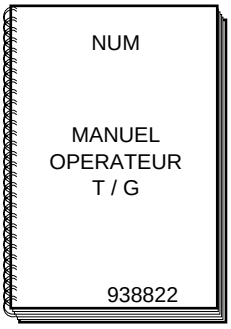
Date	Indice	Nature des évolutions
07 - 96	7	- %Rg01.3, E_RAX1 à E_RAX8, Rappel d'axes sur groupe d'axes CN g - %Rg01.7, E_OPER1 à E_OPER8, Signale un arrêt programmé provoqué par M00 ou M01 validé - %Rg06.B, MODCOUR1 à MODCOUR8, Mode en cours sur groupe d'axes CN g - %Wg01.1, C_ARUS1 à C_ARUS8, Demande d'arrêt usinage du groupe d'axes CN g - %Wg01.3, C_RAX1 à C_RAX8, Sélection du rappel d'axes sur groupe d'axes CN g - %Wg01.6, C_SLASH1 à C_SLASH8, Validation du saut de bloc sur groupe d'axes CN g - %Wg01.7, C_M011 à C_M018, Validation de l'arrêt programmé optionnel (M01) sur groupe d'axes g - %WE00.B à %WE1F.B, RDUC_TRQ0 à RDUC_TRQ31, Réduction de courant Adjonction dans la fonction dateur de DTGET Adjonction dans le chapitre «mise au point des programmes, commande de l'unité centrale», des messages au déassemblage du code client.
11 - 97	8	Corrections diverses - %Wg03.B, Mode groupe indépendant - Zone d'échange AP ↔ CN liée à la fonction AN96 "1050"

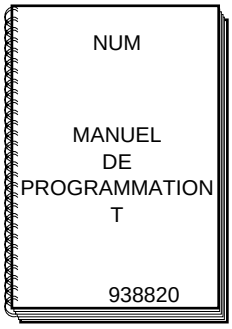
Structure de la documentation produits NUM 1020/1040/1060

Documents utilisateur

Ces documents sont destinés à l'exploitation de la commande numérique.

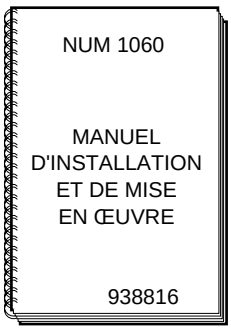


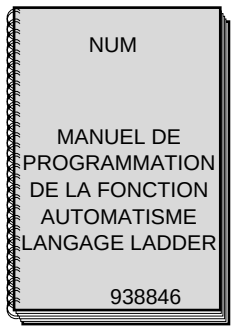




Documents intégrateur

Ces documents sont destinés à la mise en oeuvre de la commande numérique sur une machine.





Documents spécifiques de programmation

Répertoire des utilitaires des produits NUM 1020/1040/1060

Les produits NUM disposent d'une série d'utilitaires permettant l'intégration et l'exploitation du système.

Ces utilitaires peuvent être présents de base dans le système ou optionnels.

Suivant la fonction assurée par chaque utilitaire, sa mise en œuvre est décrite dans le manuel d'intégration ou d'exploitation approprié.

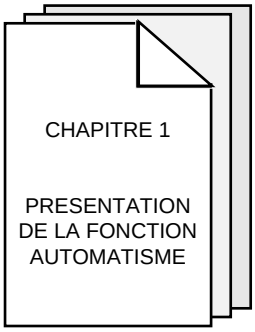
Le tableau ci-après fournit la liste des utilitaires et le chapitre de la documentation qui traite de leur utilisation :

Utilitaire	Intitulé	Manuel	Chapitre	Domaine d'application
UT0	gestion des utilitaires	manuels opérateur	8	NUM 1020/1040/1060
UT2	calibration d'axes	manuel d'installation et mise en œuvre (1020/1040 ou 1060)	10 11	NUM 1020/1040 NUM 1060
UT3	macros résidentes	manuels opérateur	8	NUM 1020/1040/1060
UT5	intégration des paramètres	manuel des paramètres	12	NUM 1020/1040/1060
UT7	mise au point de programmes	manuel de programmation de la fonction automatisme langage ladder	16	NUM 1020/1040/1060 programmation en langage ladder
UT12	verrouillage des options	manuels opérateur	8	NUM 1020/1040/1060
UT20	calibration inter axes	manuel d'installation et mise en œuvre (1020/1040 ou 1060)	11 12	NUM 1020/1040 NUM 1060
UT22	intégration des paramètres axes	manuel SET_TOOL	8	NUM 1060

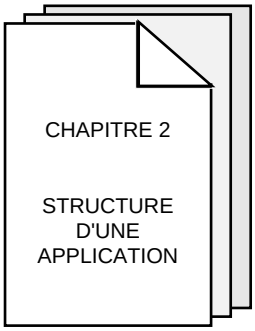
REMARQUE: L'utilitaire 22 n'est plus utilisé à partir du logiciel CN indice K et le logiciel SET_TOOL indice E.

Manuel de programmation de la fonction automatisme

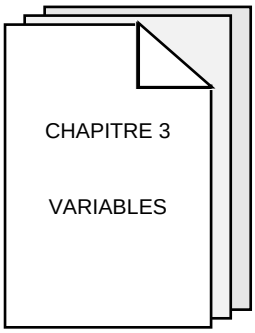
Programmation de la fonction automatisme en langage LADDER. Traitement des fonctions d'automatismes mettant en oeuvre capteurs et actionneurs implantés sur la machine. Traitement des informations d'interface avec la fonction CN.



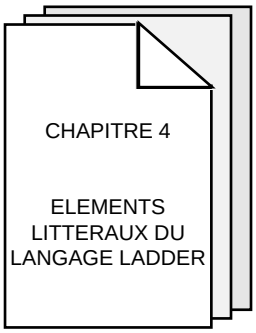
Présentation et caractéristiques de la fonction automatisme et de l'unité centrale .
- Synoptiques du système et des cartes mises en oeuvre.



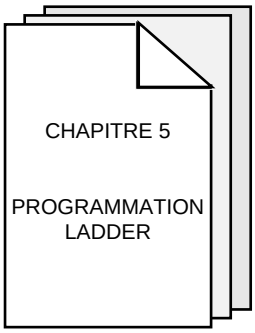
Principe de fonctionnement et d'organisation d'une application automate.
- Les tâches système.
- Les tâches utilisateur.
- La structure d'une application.
- Les modules.



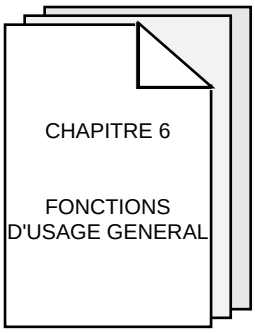
Détail des variables mises en oeuvre.
- Les variables internes.
- Les variables Entrées/sorties borniers.
- Les variables de configuration et de diagnostic.
- Les variables d'interface avec la CN.
- Les variables mots communs.
- Les variables locales.



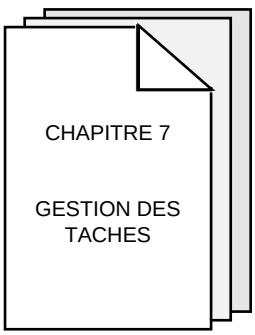
Information sur les éléments du langage ladder.
- Les éléments littéraux.
- Les opérateurs.
- Exemples de calculs.



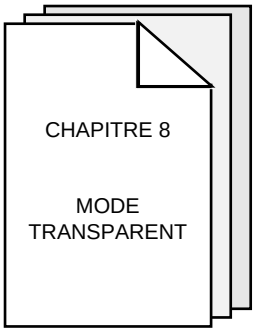
Information sur la programmation ladder.
- Les éléments communs.
- Etapes grafcet.
- La séquence réseau.
- Conseils de programmation.



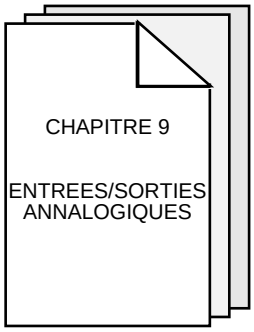
Fonctions d'usage général utilisées en langage ladder.
Syntaxe.
Fonctionnement.



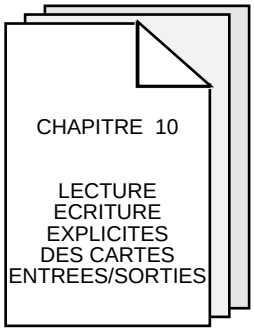
Principes et fonctions liés à la gestion des tâches.



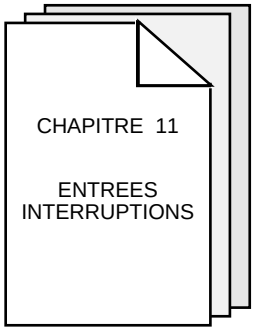
Principe et fonctions liés à la programmation du mode transparent.



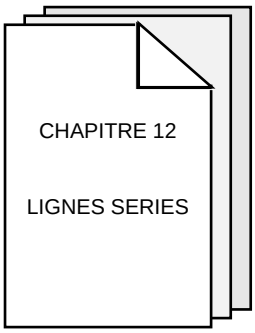
Principe et fonctions liés à la programmation des Entrées/Sorties analogiques.



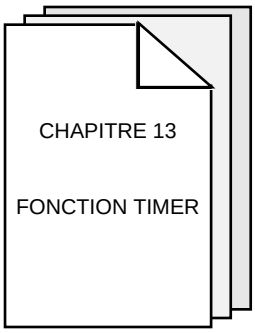
Principe et fonctions liés à la lecture et à l'écriture immédiate des cartes entrées sorties.



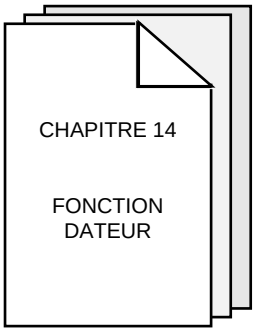
Principe et fonctions liés à la programmation des entrées interruptions.



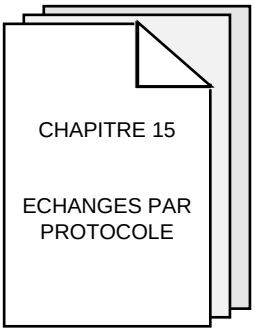
Principe et fonctions liés à la programmation des lignes séries.



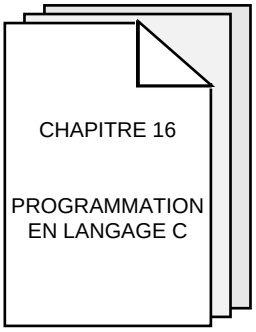
Principe et fonctions liés à la programmation de la fonction timer.



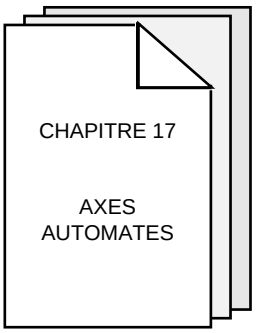
Principe et fonctions liés à la programmation du dateur.



Principes et fonctions liés à la programmation des échanges par protocole.

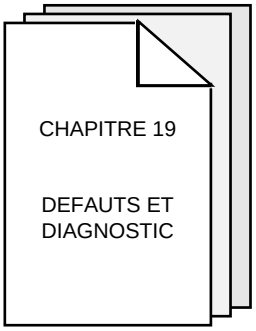


Fonctions traitant de l'appel de module en langage C.

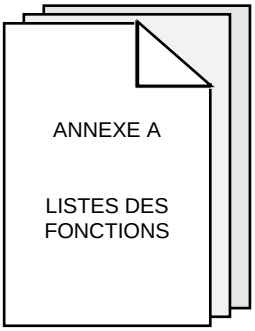


Principes et applications liés à la programmation des axes automates.

Outils de création et de mise au point des programmes.
- Modes opératoires.



Niveaux de contrôle de l'unité centrale et liste des défauts.



Liste des fonctions ladder.
- Classement par thème.
- Classement alphanumérique.

Utilisation du manuel de programmation de la fonction automatisme

Modes opératoires

Le manuel comporte des modes opératoires (en particulier dans le chapitre 18).

Les actions à réaliser sont présentées sous la forme suivante :

Réinitialiser le système.	

La partie droite indique les touches à actionner qui peuvent se présenter sous deux formes :

 Touches carrées : correspondent à des touches du pupitre.

Touches rectangulaires : correspondent à des touches logicielles qui apparaissent dans le cartouche en bas de l'écran et sont actionnées par les touches de fonction (F2 à F11) situées sous l'écran.

Index

L'index figure en fin de volume et permet d'accéder à des renseignements ponctuels par des mots clés.

Agences

La liste des agences NUM figure en fin de volume.

Questionnaire

Afin de nous aider à améliorer la qualité de notre documentation, nous vous demandons de bien vouloir nous retourner le questionnaire figurant en fin de volume.

1 Présentation de la fonction automatisme

1.1 Généralités

La CN NUM 1060 est un système multiprocesseur multimaître dans lequel la fonction automatisme assure la charnière avec la MOCN.

La fonction automatisme assure le traitement des fonctions qui mettent en oeuvre, tant les capteurs et actionneurs implantés sur la MOCN, que les informations booléennes ou numériques d'interface avec la fonction CN.

Ses possibilités, d'accès à l'écran de la CN et de simulation du pupitre, lui confèrent une grande souplesse d'utilisation permettant au constructeur de machines outils une personnalisation du système 1060 à son ergonomie.

La fonction automatisme est implantée dans l'unité centrale. L'unité centrale est composée d'une ou plusieur carte et assure les fonctions CN, graphique, automatisme ainsi que la mémoire.

L'unité centrale se compose :

Fonctions	CN	Graphique	Automatisme	Mémoire
NUM 1060 série I	Proc CN	Proc graphique	Proc machine	Carte mémoire
NUM 1060 série II multicarte	Proc graphique	Proc graphique	Proc machine	Carte mémoire
NUM 1060 série II monocarte	UCSII	UCSII	UCSII	UCSII

Le transfert d'information, sur bit ou sur octet, avec les cartes ENTREES/SORTIES, est assuré par le bus série.

Les échanges d'informations dans le système sont de deux types :
- communication par zone d'échanges,
- communication par protocole.

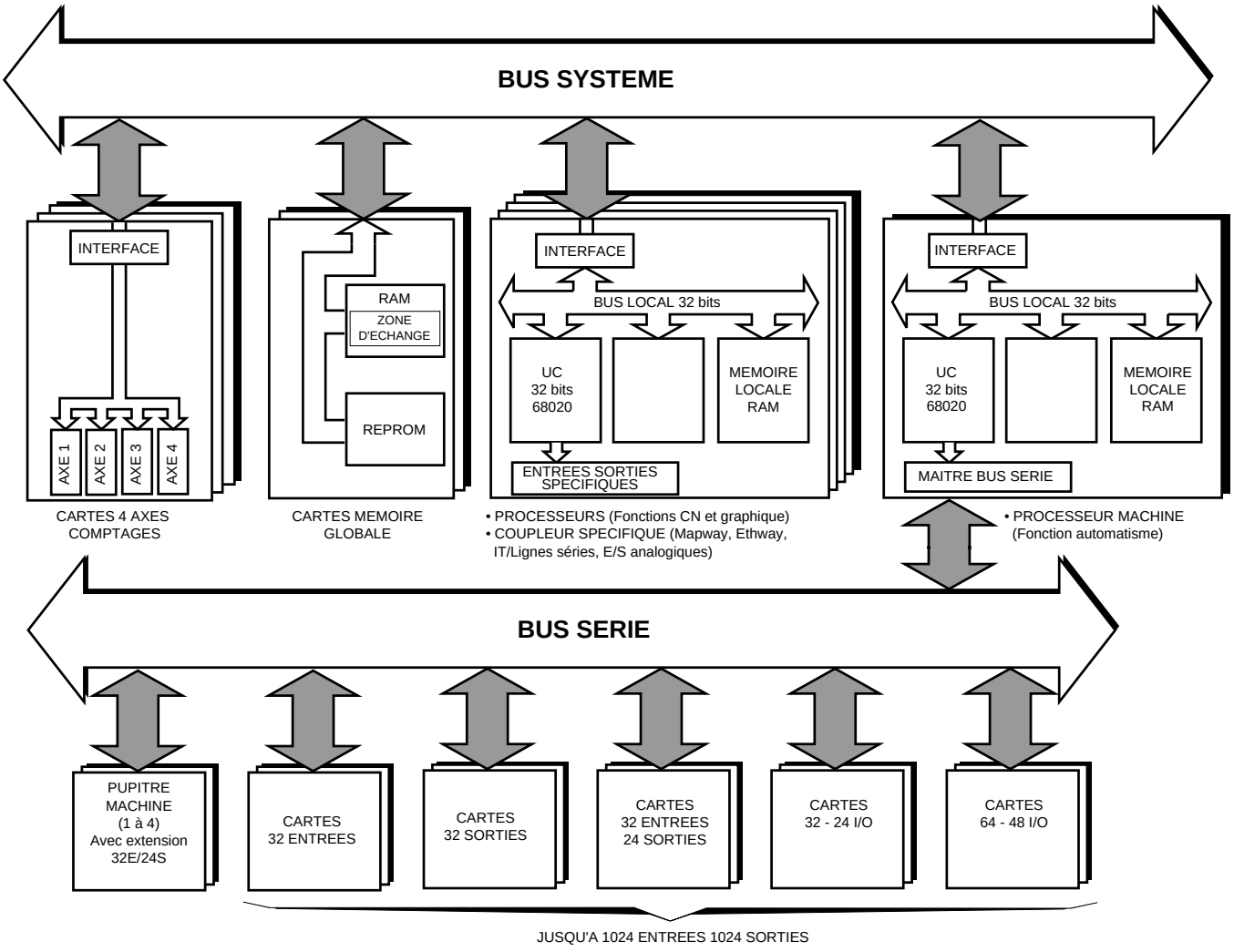


Figure 1.1 - Synoptique d’organisation générale d'une UC multiprocesseurs

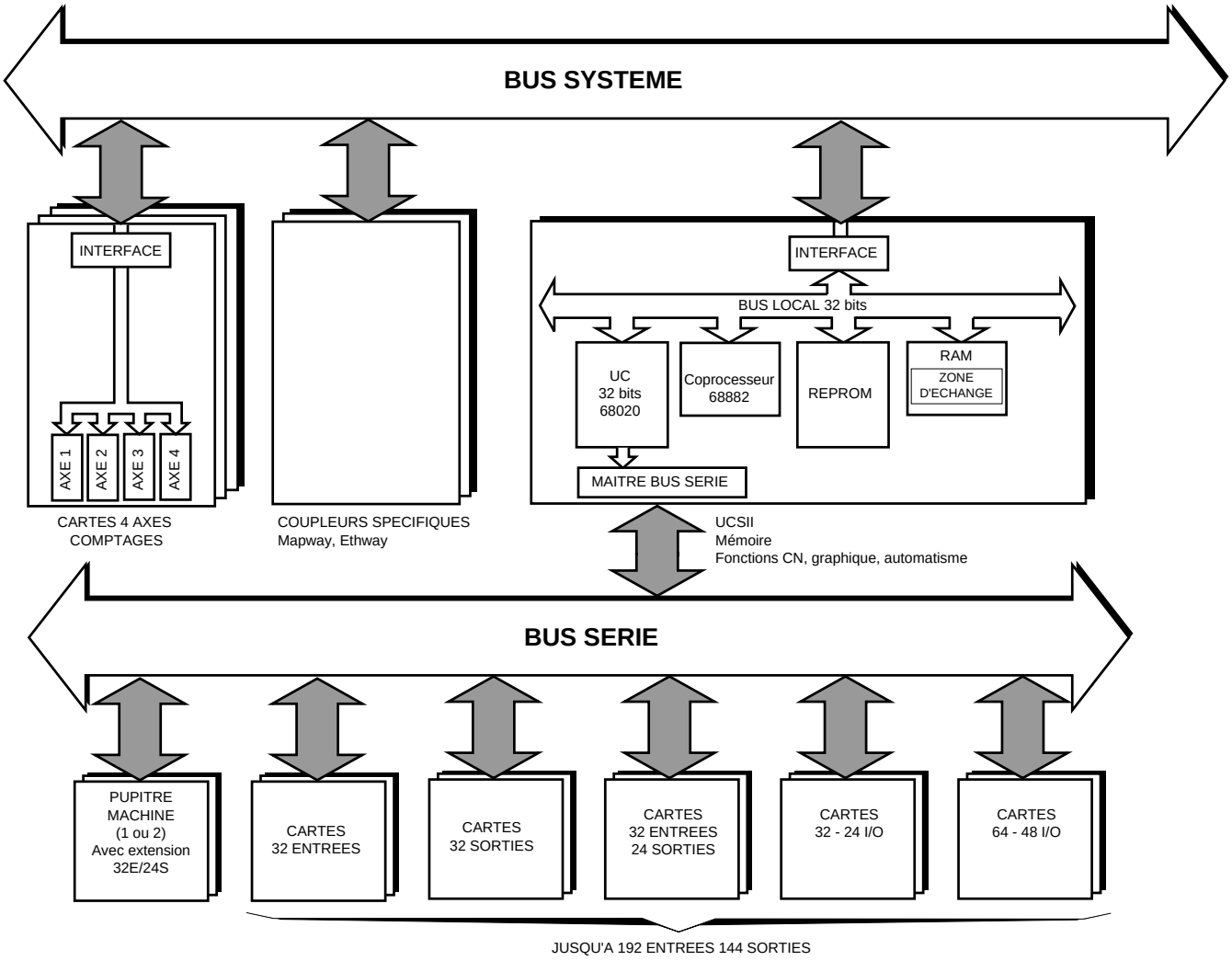


Figure 1.2 - Synoptique d’organisation générale d'une UC monocarte (UCSII)

1.2 Fonction automatisme

La gestion de la fonction automatisme est assuré par un moniteur chargé d'un certain nombre de tâches de base telles qu'initialisation, affectation des entrées/sorties sur les différents racks, échange des entrées/sorties, chien de garde ..etc..

A ce traitement systématique effectué par le moniteur, vient s'ajouter le traitement programme dit «Programme utilisateur».

Le déroulement du programme s'effectue sous le contrôle du moniteur de gestion. Le déroulement du programme est rythmé par l'horloge temps réel (HTR) dont la périodicité est de 20 ms.

La zone mémoire réservée à la programmation de la fonction automatisme est structurée comme suit:
- 30 Ko de mémoire RAM statique sauvegardée,
- 32 Ko de mémoire RAM dynamique initialisée à la mise sous tension,
- 180 Ko de mémoire RAM dynamique occupée par le programme utilisateur sur les processeurs machine V1 1Mo,
- 2,5 Mo de mémoire RAM dynamique occupée par le programme utilisateur sur les processeurs machine V1 4Mo,
- 3,5 Mo de mémoire RAM dynamique occupée par le programme utilisateur sur les processeurs machine V2 4Mo,
- 64 ko de mémoire RAM dynamique occupée par le programme utilisateur sur les modules UCSII.

La fonction automatisme permet:
- Un accès direct aux CNA,
- Un accès indirect en lecture et en écriture aux CAN et aux Entrées/Sorties. Cet accès se fait en espace mémoire virtuelle (toutes les 20 ms).

	Nb d'Entrées / Sorties	Nb de rack maximum
NUM 1060 série I	1024E / 1024S	1 principal 6 d'extension
NUM 1060 série II	192E / 144S	1 principal

La configuration des entrées / sorties est figée à la mise sous tension. Le temps de rafraîchissement demande 2 ms.

La mise en oeuvre de la fonction automatisme nécessite l'outil de programmation sur micro-ordinateur PLCTOOL. Il permet :
- la programmation en langage ladder,
- la compilation,
- le transfert des programmes vers l'unité centrale,
- la mise au point des programmes chargés.

Associé au compilateur MCC68K de MICROTEC RESEARCH, PLCTOOL permet également la saisie, la compilation, le transfert et la mise au point de programmes écrits en langage C.

Le chargement/déchargement des programmes se fait par l'intermédiaire d'une des lignes séries du système.

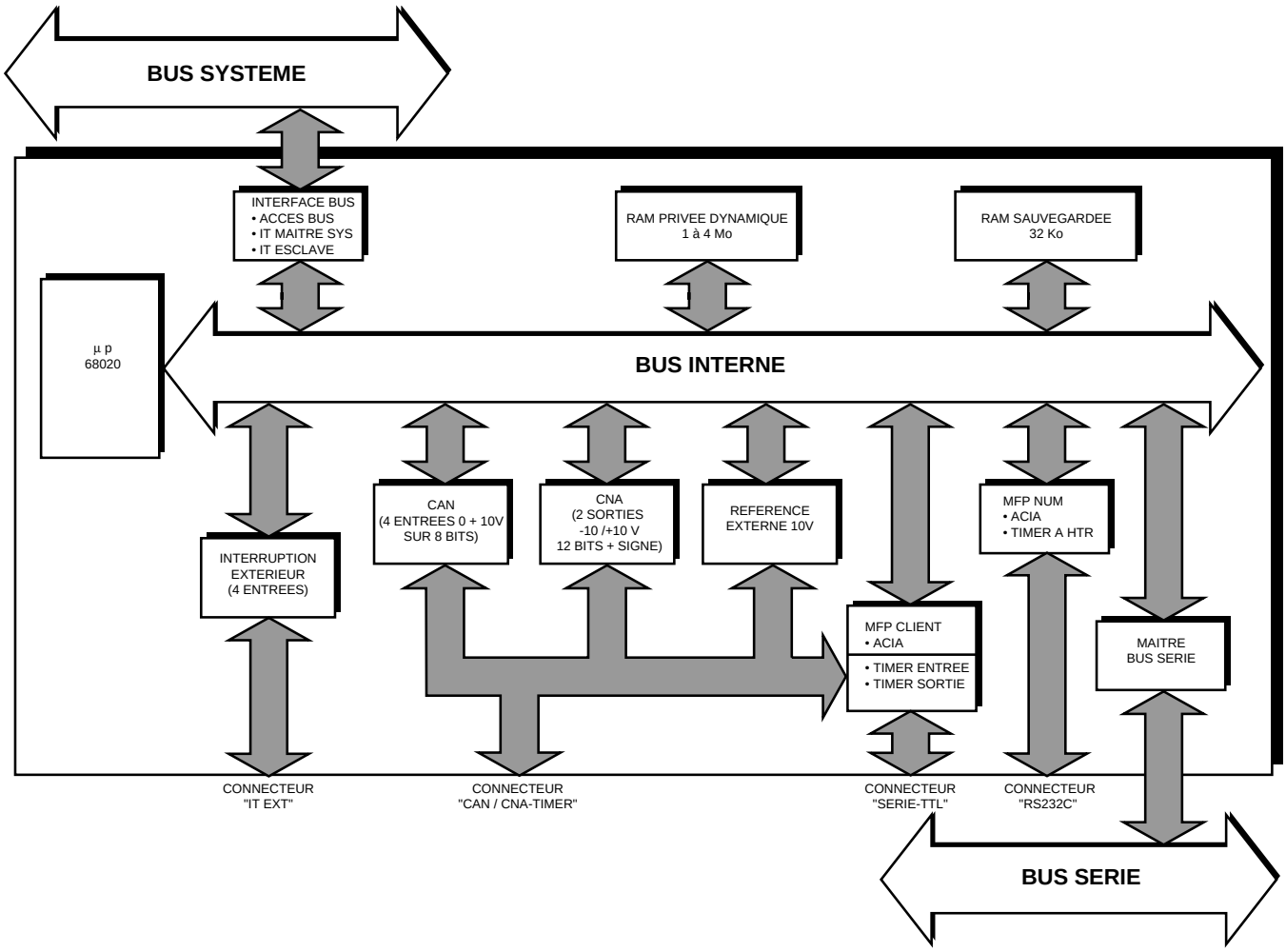


Figure 1.3 - Synoptique du processeur machine

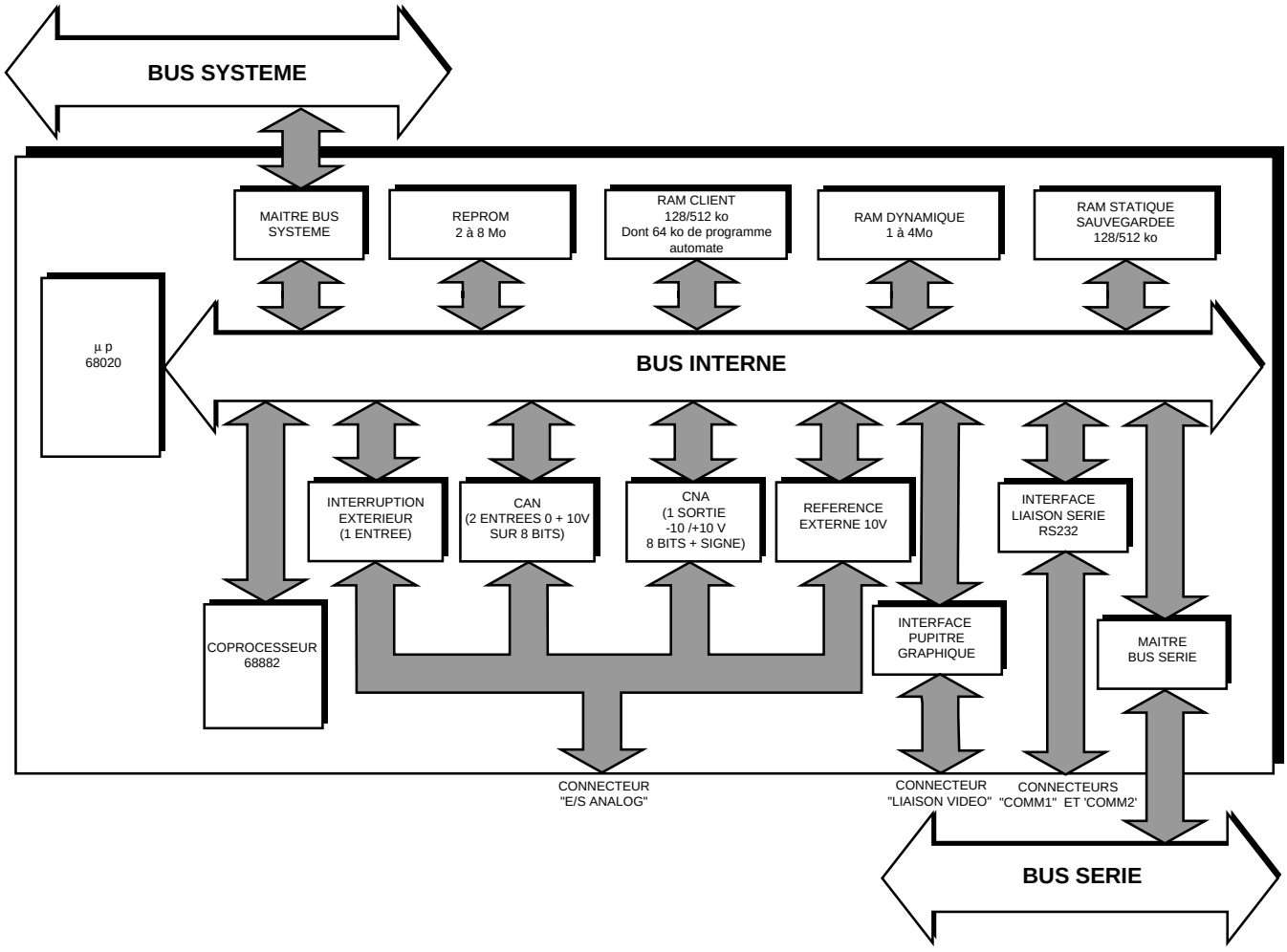


Figure 1.4 - Synoptique de la carte UCSII

2 Structure d'une application

2.1 Généralités

Deux types de tâches cohabitent dans la fonction automatisme :
- les tâches «système» déclenchées par le moniteur et non programmables par l'utilisateur,
- les tâches «utilisateur» programmables par l'utilisateur.

2.1.1 Tâches «système»

2.1.1.1 Tâche «système» initialisation

Traitement exécuté

Lors d'une initialisation le système exécute :
- auto-test des ressources de l'unité centrale,
- vérification de l'intégrité du code «système» en mémoire globale,
- transfert du code «système» de la mémoire globale vers la mémoire de travail,
- vérification de l'intégrité du code «utilisateur» en mémoire globale,
- transfert du code «utilisateur» de la mémoire globale vers la mémoire de travail,
- scrutation des cartes E/S présente sur le bus série :
  . mise à jour du status et de l'identificateur de chaque carte E/S,
  . lecture des entrées de chaque carte E/S et mise à jour de la zone image %I,
- lancement de la tâche «utilisateur» %INI.

Occurrence

L'initialisation de la fonction automatisme s'effectue :
- à la mise sous tension de la CN,
- lors de l'appui sur le bouton «RaZ» en face avant de la carte alimentation.

2.1.1.2 Tâche «système» rafraîchissement E/S CN

Traitement exécuté

Cette tâche traite les échanges systématiques avec la fonction CN :
- lecture des entrées CN (Variables %R. écrites par la fonction CN),
- écriture des sorties CN (Variables %W. lues par la fonction CN).

Occurrence

Cette tâche s'exécute à chaque cycle HTR (Horloge Temps Réel).

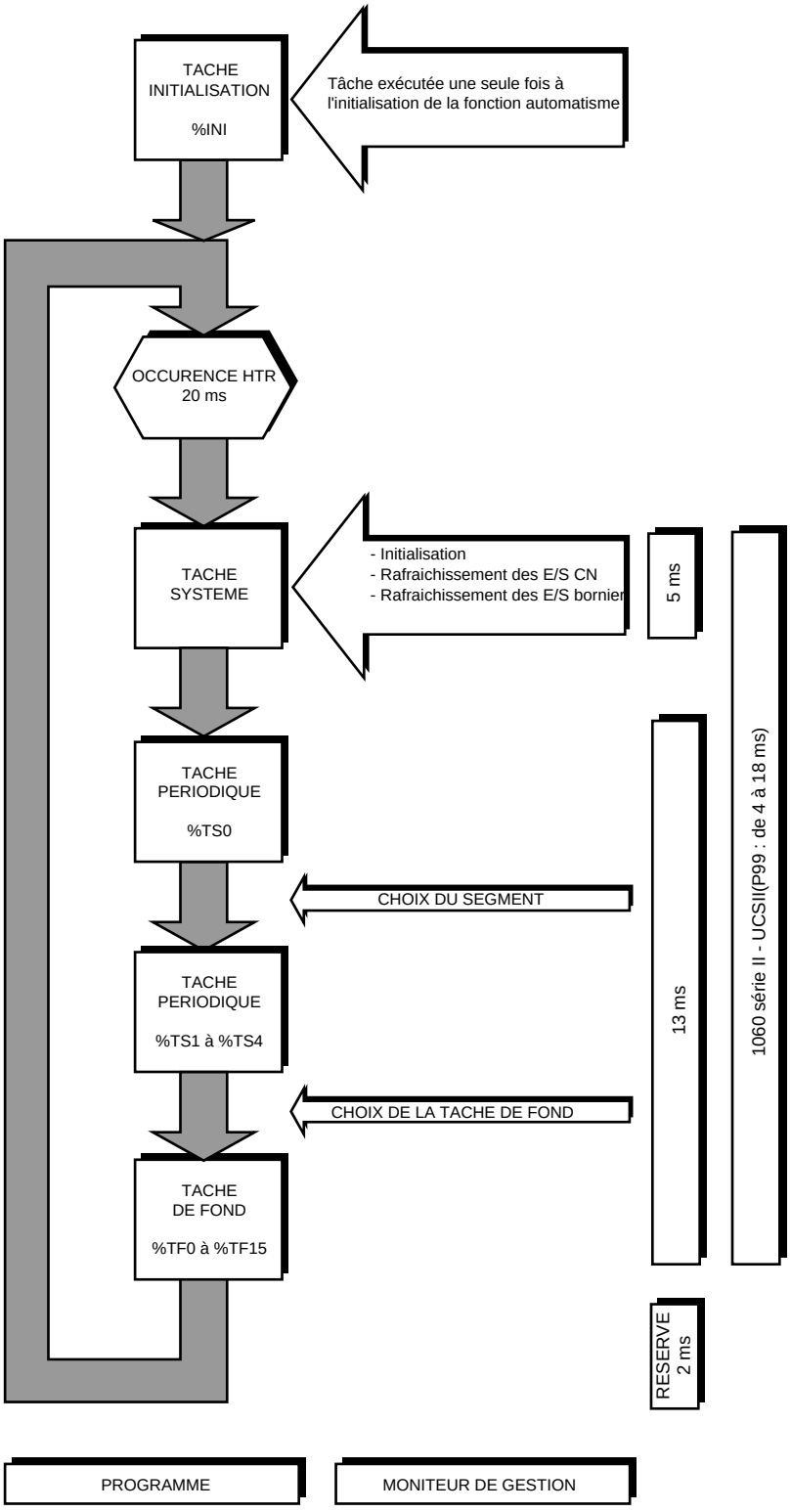


Figure 2.1 - Déroulement d'une application

2.1.1.3 Tâche «système» rafraîchissement E/S borniers

Traitement exécuté

Cette tâche effectue le rafraîchissement des E/S borniers :
- lecture des entrées borniers %I.,
- écriture des sorties borniers %Q.,
- mise à jour des variables de diagnostic cartes E/S borniers.

2.1.1.4 Tâche «système» serveur UNITE

Traitement exécuté

Cette tâche traite les requêtes UNITE destinées au serveur de la fonction automatisme.

Service rendu par le serveur UNITE

Le serveur de la fonction automatisme traite principalement les requêtes UNITE suivantes :
- lecture/Ecriture variables (%I, %Q, %R, %W, %M, %V),
- chargement et déchargement des fichiers de la fonction automatisme (%TS0, %SP30, .. etc ..),
- STOP de la fonction automatisme (arrêt des tâches utilisateur),
- INIT de la fonction automatisme (initialisation de l'unité centrale),
- RUN de la fonction automatisme (démarrage des tâches utilisateur).

2.1.2 Tâche «utilisateur»

2.1.2.1 Tâche à l'initialisation

La tâche %INI est appelée par le système à l'initialisation de la fonction automatisme avant toutes les autres tâches «utilisateur».

Cette tâche permet La configuration des cartes E/S sur borniers.

> ⚠ **ATTENTION**
>
> Le système prend en compte la configuration des cartes E/S au retour de %INI. Une modification ultérieure de la configuration n'est donc pas prise en compte.

2.1.2.2 Tâche périodique

Les tâches périodiques sont %TS0 à %TSn (Avec n tournant de 1 à 5 à chaque cycle HTR).

La période d'exécution de %TS0 est égale à un cycle HTR soit 20 ms.

La période d'exécution de %TS1, %TS2, %TS3, %TS4 est égale à cinq cycle HTR soit 100 ms (Le cinquième cycle HTR est utilisé par la tâche système %TS5).

Les tâches %TS ne sont pas interruptibles :
- à l'occurrence de la HTR (1060 série I et série II multicarte),
- sur l'IT fixée par le paramètre P99 (1060 série II - UCSII).

2.1.2.3 Tâche de fond

Ces tâches (%TF0 à %TF15) permettent d'effectuer des traitements non prioritaires sans pénaliser les tâches périodiques %TS. Elles permettent aussi l'utilisation de fonctions bloquantes.

Les tâches %TF ont une priorité inférieure à celle des tâches %TS et %TH.

Les tâches %TF ont les particularités suivantes :
- une tâche est exécutée sur une demande explicite par la fonction tfstart (..),
- une tâche ne sera exécutée que s'il reste du temps de cycle après l'exécution des tâches périodiques,
- une tâche n'est exécutée qu'une fois par cycle HTR,
- une tâche désarmée par la fonction tfstop (..) est exécutée en totalité.

Les tâches %TF sont interruptibles :
- à l'occurrence de la HTR (1060 série I et série II multicarte),
- sur l'IT fixée par le paramètre P99 (1060 série II - UCSII).

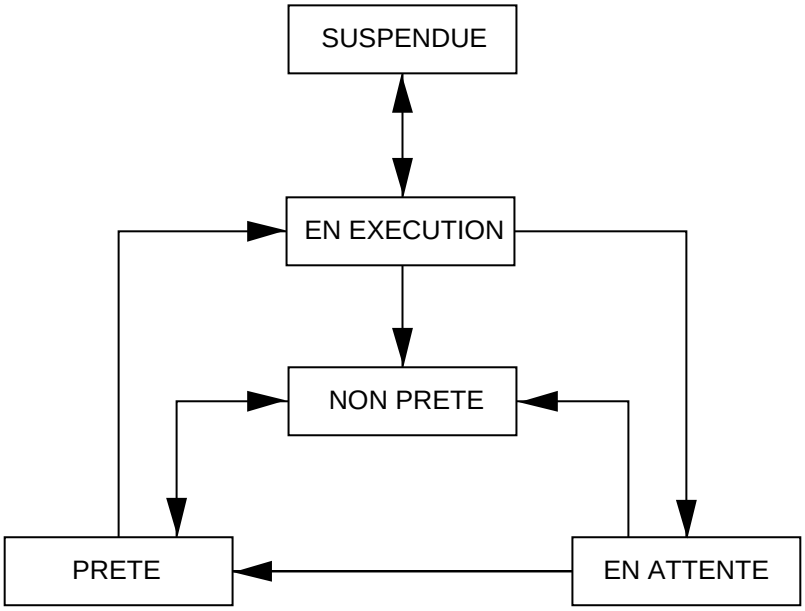


Figure 2.2 - Etats d'une tâche %TF

Fonctionnement des tâches de fond

NON PRETE —> PRETE

A l'initialisation du système les tâches %TF sont dans l'état NON PRETE. L'appel de la fonction tfstart(n) fait passer %TFn à l'état PRETE.

EN ATTENTE ou PRETE ou EN EXECUTION —> NON PRETE

L'appel de la fonction tfstop(n) fait passer la tâche %TFn dans l'état NON PRETE.

PRETE —> EN EXECUTION

Dans l'état PRETE une tâche %TF est exécutée dès qu'aucune tâche n'est EN EXECUTION et qu'aucune tâche %TF de priorité supérieure ne se trouve dans l'état PRETE .

A l'intérieur des tâches %TF la hiérarchie des priorités est fixée par le numéro :

priorité %TF0 > priorité %TF1 > .... > priorité %TF15

EN EXECUTION —> SUSPENDUE

La tâche %TF est suspendue pour permettre l'exécution d'une tâche %TS ou %TH. Les tâches %TF ne sont pas préemptibles entre elles.

SUSPENDUE —> EN EXECUTION

Aucune des tâches %TS ou %TH n'est en cours d'exécution: la tâche %TF suspendue est à nouveau exécutée.

EN EXECUTION —> EN ATTENTE

La tâche %TF a fait un appel à la fonction whtr(..) ou elle se termine (fin du code atteinte).

REMARQUE : La fonction whtr(..), programmée dans une tâche de fond, interrompt la tâche pendant son exécution et permet ainsi d'exécuter d'autres tâches %TF à l'état PRETE.

EN ATTENTE —> PRETE

La tâche était en attente depuis un certain nombre de cycle HTR par la fonction whtr(..) et le nombre de cycle HTR est écoulé.

La tâche était terminée, l'occurrence de la HTR la replace dans l'état PRETE.

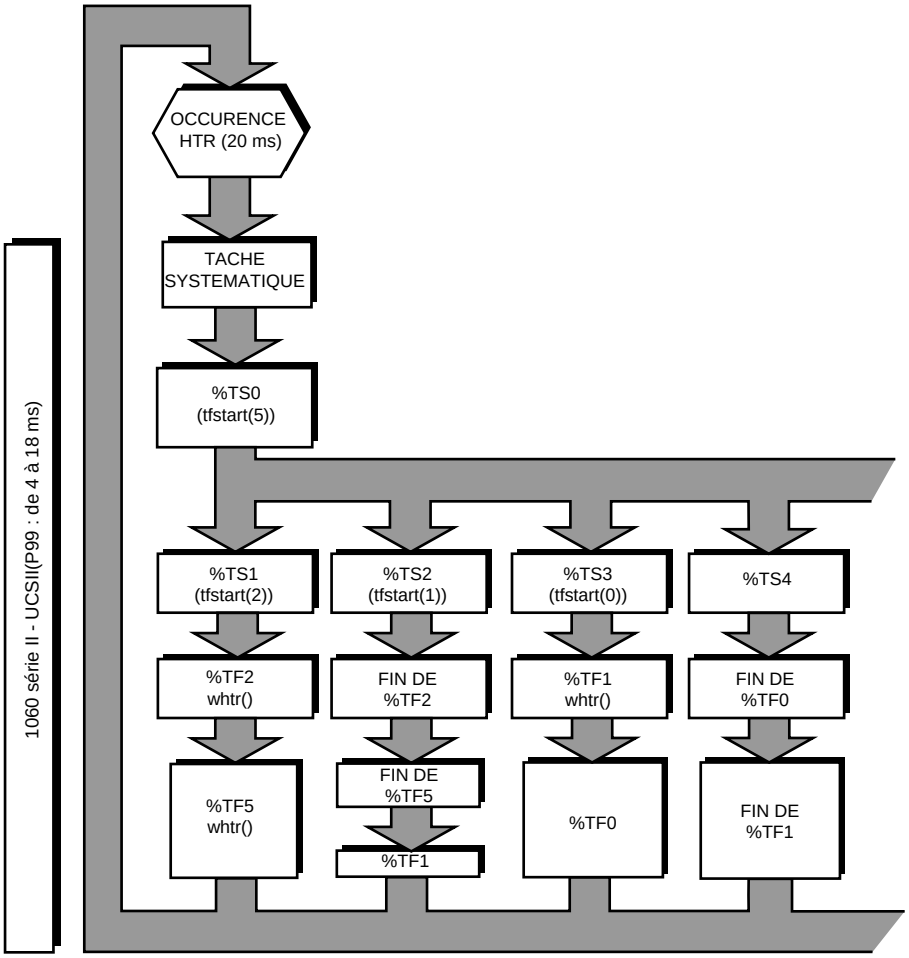


Figure 2.3 - Traitement des tâches %TS et %TF

2.1.2.4 Tâches temps réels

Les tâches (%TH0 à %TH15) permettent de prendre en compte des événements prioritaires dont le traitement ne peut attendre l'occurrence de la HTR.

Les tâches %TH ont une priorité supérieure à celle des tâches %TS et %TF.

Une tâche %TH activée ne peut interrompre une tâche %TH en cours.

A l'intérieur des tâches %TH la hiérarchie des priorités est fixée par le numéro :

priorité %TH0 > priorité %TH1 > .... > priorité %TH15.

Fonctionnement des tâche %TH

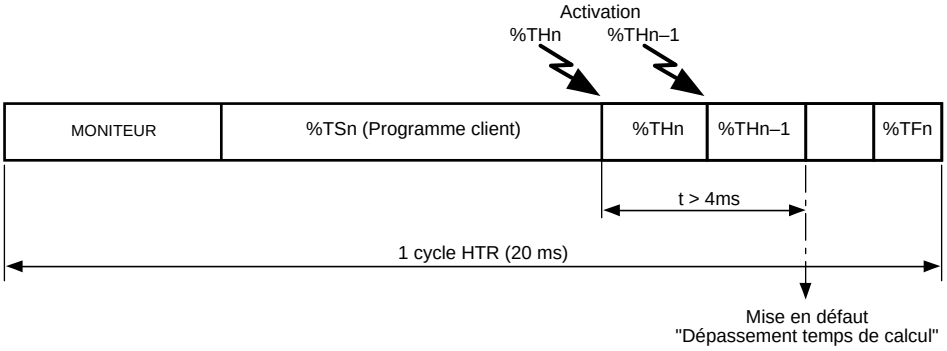
Le programmeur associe une tâche %TH à une interruption électronique grâce aux fonctions suivantes :

thiti(..) interruption palpeur.

thtimer(..) interruption timer.

A l'occurrence de l'interruption, le système lance l'exécution de la tâche %TH associée.

Si plusieurs tâches %TH sont activées au cours du même cycle HTR, le cumul de temps de traitement de chaque routine d'interruption ne doit pas excéder 4 ms. Si le temps de traitement est supérieur l'unité centrale sera mise en défaut «Dépassement temps de calcul».



2.1.3 Traitement des débordements

Les anomalies de fonctionnement du programme utilisateur sont signalées par :
- l'incrémentation du compteur de dépassement HTR %R97C.W,
- La mise en REPLI_SUR_DEFAUT de l'unité centrale dans les cas critiques.

La mise en évidence de ce type d'anomalies nécessite une reprise du programme utilisateur.

2.1.3.1 Systèmes 1060 série I et série II multicarte

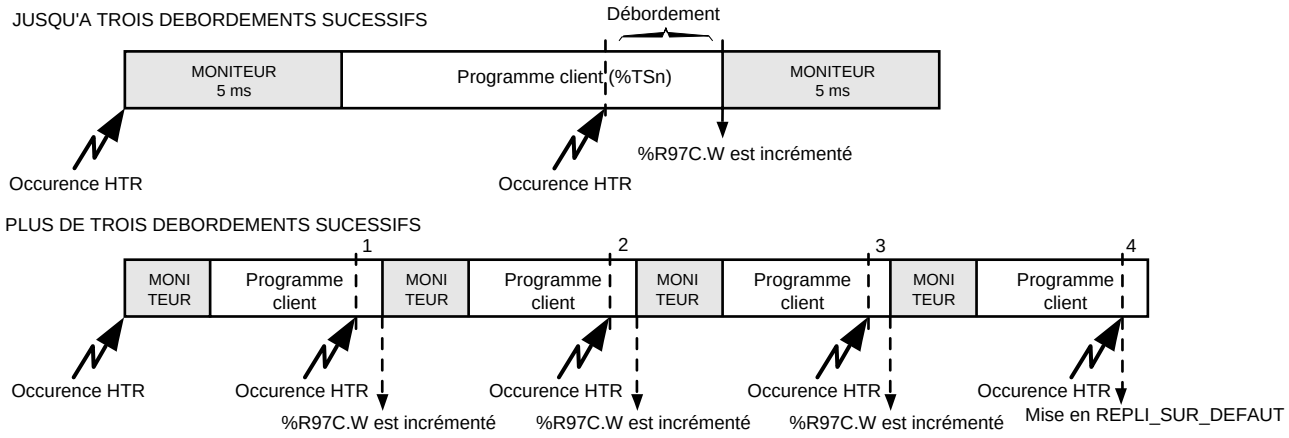
Etant rythmé par l'occurrence de la HTR toutes les 20 ms, le traitement des tâches %TS doit normalement être effectué avant l'apparition de celle-ci.

Débordements successifs

Un léger dépassement de l'occurrence de la HTR est toléré dans le traitement des tâches %TS. A chaque dépassement le système incrémente le compteur de dépassement HTR %R97C.W.

Le système autorise trois débordements successifs maximum.

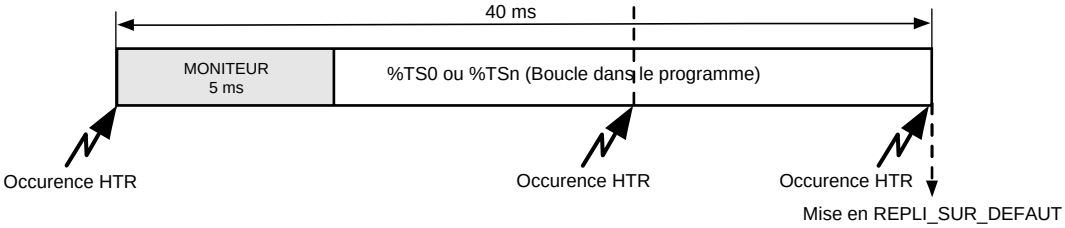
Le quatrième débordement provoque l'incrémentation du compteur de dépassement %R97C.W et la mise en REPLI_SUR_DEFAUT de l'unité centrale.



REMARQUE *Un programme utilisateur peut ne pas créer de débordement en fonctionnant à vide (par exemple sans usiner de piéce), mais en charge des débordements peuvent se produire par suite des taches hard CN (traitement des asservissements,....) ou des IT (lignes série....) qui se rajoutent dans l'intervalle entre deux HTR.*

Boucle dans un programme

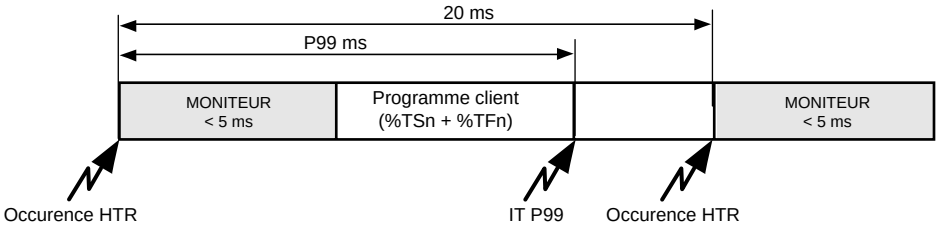
L'exécution ininterrompue de %TSn pendant plus de 40 ms provoque la mise en REPLI_SUR_DEFAUT avec l'erreur ERR_DEPASSEMENT_HTR.



2.1.3.2 Systèmes 1060 série II - UCSII

Fonctionnement normal

Etant rythmé par l'occurrence de la HTR toutes les 20 ms, le programme utilisateur est néanmoins limité à une durée fixée par le paramètre P99 (Voir manuel des paramètres). Le paramètre P99 est un multiple de 2 ms et doit être compris entre 4 et 18 ms. Le traitement des tâches %TS doit normalement être effectué avant l'apparition de l'IT P99.



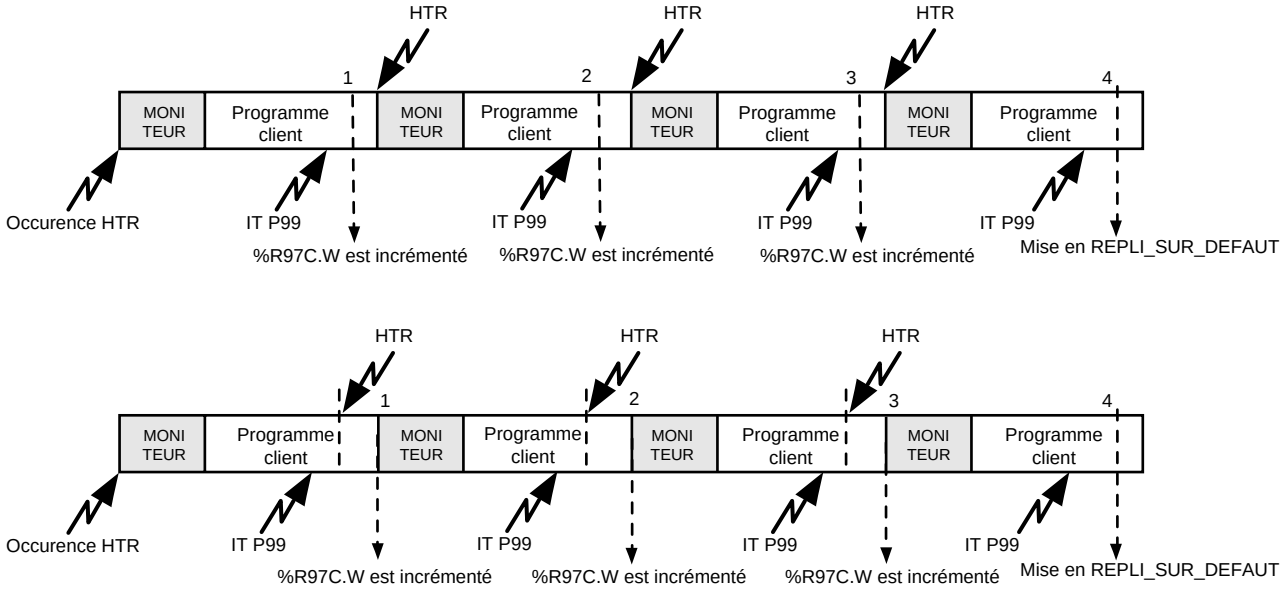
Débordements successifs

Un léger dépassement de l'IT P99 est toléré dans le traitement des tâches %TS. A chaque dépassement le système incrémente le compteur de dépassement HTR %R97C.W.

Si l'occurrence de la HTR arrive alors que les %TS ne sont pas achevées, le moniteur est relancé immédiatement.

Le système autorise trois débordements successifs maximum.

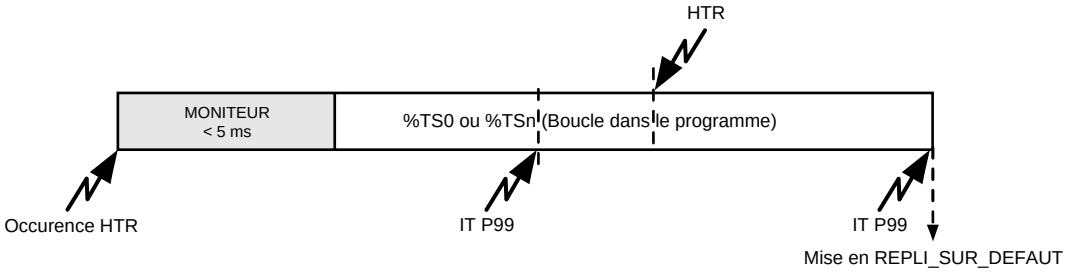
Le quatrième débordement provoque l'incrémentation du compteur de dépassement %R97C.W et la mise en REPLI_SUR_DEFAUT de l'unité centrale.



REMARQUE *Un programme utilisateur peut ne pas créer de débordement en fonctionnant à vide (par exemple sans usiner de piéce), mais en charge des débordements peuvent se produire par suite des taches hard CN (traitement des asservissements,....) ou des IT (lignes série....) qui se rajoutent dans l'intervalle entre deux HTR ou entre la HTR et l'IT P99.*

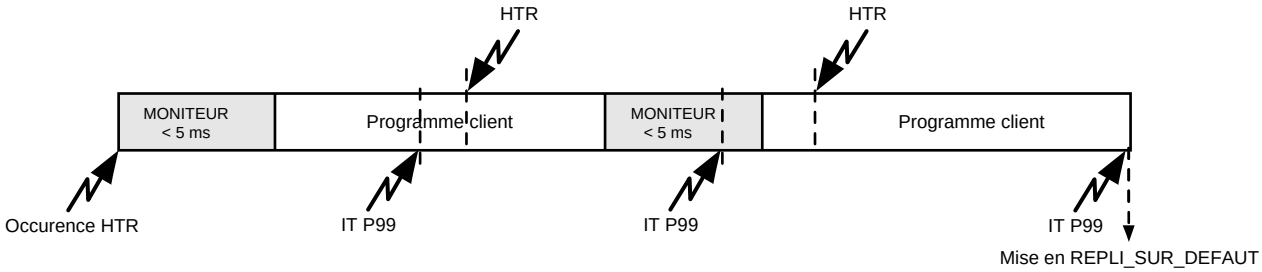
Boucle dans un programme

L'exécution ininterrompue de %TSn pendant 2 x P99 ms provoque la mise en REPLI_SUR_DEFAUT avec l'erreur ERR_DEPASSEMENT_HTR.



Non traitement du moniteur

La non exécution du moniteur pendant 2 x P99 ms provoque la mise en REPLI_SUR_DEFAUT avec l'erreur ERR_DEPASSEMENT_HTR.



2.2 Structure d'une application

Une application se compose d'un ensemble de modules, crées sous l'outil de programmation PLCTOOL, qui sont chargés sur la CN dans le but de piloter l'installation.

Détail des modules

Modules «tâche ladder»

Les modules «tâche ladder» sont associés :
- à la tâche %INI,
- aux tâches %TS0 à %TS4,
- aux tâches %TF0 à %TF15,
- aux tâches %TH0 à %TH15.

Ces modules sont appelés par le gestionnaire de tâches du système. Ils ne peuvent pas être appelés explicitement. L'utilisateur gère éventuellement ces modules «tâche» grâce aux fonctions de gestion des tâches (Voir chapitre 7).

Les modules «tâche ladder» sont des fichiers du type «*.XLA».

Modules «sous-programme ladder»

Les modules «sous-programme ladder» sont notés %SP0 à %SP255. Ils peuvent être appelés à l'intérieur d'un module «tâche» ou d'un autre module «sous-programme» grâce à la fonction sp(..) ou spy(..).

Un sous-programme automate écrit en ladder et appelé à partir d'un module C par la fonction SP, ne doit comporter AUCUN appel à une fonction (hormis les fonctions goto 0 et call 0).

Les modules «sous-programme ladder» sont des fichiers du type «*.XLA».

Les modules exécutables

Ces modules exécutables sont issus de la chaîne de compilation (MCC68K) en langage C. Ils peuvent être appelés à l'intérieur d'un module «tâche ladder» ou d'un module «sous-programme ladder» grâce à la fonction exec(..).

Toutes les tâche peuvent être programmées en langage C. Si une même tâche est écrite en langage C et en ladder, c'est le code C qui sera exécuté par la fonction automatisme.

Les modules exécutables sont des fichiers du type «*.XCX».

Particularité pour l'initialisation d'un module C (.XCX)

Implantation du code de la mémoire globale en mémoire locale AP,

Exécution dans le "main()" des directives suivantes:
- Import(),
- Export(),
- Initialisation possible de certains types de variables : variables globales C du module XCX, variables non initialisées (sauvegardées) du ladder.

Résolution des imports/exports (on ne peut pas utiliser des variables importées dans le "main()"),

Initialisation des variables ladder initialisées,

Exécution du module %INI,

Lancement du cycle normal de l'automate.

REMARQUE: Sur un Stop/Départ de l'automate (sans INIT), seules les trois dernières phases sont exécutées

Les variables sauvegardées sont %M, %C, %CQ; les variables initialisées sont %I, %Q, %R, %T, %TQ, %V. Les variables %W dites "impulsionnelles" sont remises à zéro par le bouton RAZ ou coupure secteur.

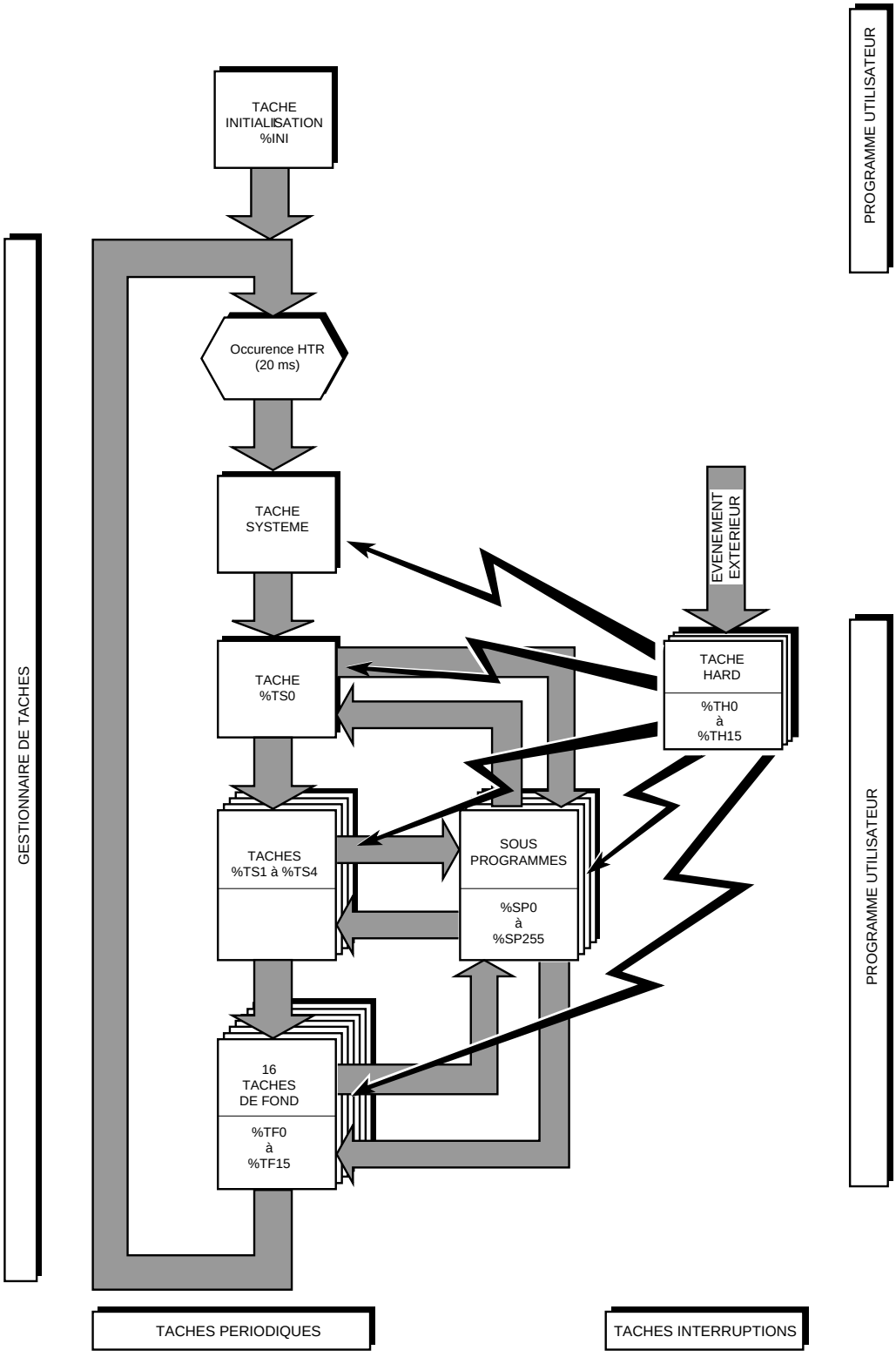


Figure 2.4 - Structure d'une application

2.3 Structure d'un module ladder - Séquences élémentaires

Un module ladder se compose d'une succession de séquences élémentaires. Le nombre de séquence dans un module est limité à 100 et la taille d'un module est limitée à 16 kO.

La séquence est l'unité de saisie et de compilation. Il existe trois types de séquences élémentaires :
- la séquence qui permet la saisie d'un tableau de constantes numériques,
- la séquence qui permet la saisie d'une ou plusieurs chaînes de caractères,
- la séquence qui permet la saisie d'un réseau de contacts et bobines.

2.4 Eléments communs à tous les types de séquence

Chaque type de séquence est composé en partie des éléments communs suivant :
- l'en-tête de séquence,
- l'étape grafcet.

2.5 La séquence tableau de constantes

2.5.1 Présentation

Ce type de séquence contient un tableau de données qui sera traité par une fonction qui exploite les buffers.

Cette séquence se compose :
- d'un label et d'un commentaire facultatif,
- d'une étape grafcet facultative,
- d'une variable %Vxx.L ou %Yxx.L associée qui va contenir l'adresse de début du tableau,
- d'une suite de valeurs numériques ou chaque valeur occupe un long mot.

Chaque tableau peut contenir jusqu'à 500 valeurs.

Le nombre de tableaux que peut contenir un module est limité par la taille maximum du module et le nombre de séquence doit être inférieur à 100.

2.5.2 Utilisation d'un tableau

L'accès se fait par l'intermédiaire de l'adresse de début du tableau contenue dans une variable %Vxx.L ou %Yxx.L et grâce aux fonctions du type cpyb(..), cpyw(..), cpyl(..), print(..), .. etc ...

On peut accéder directement aux valeurs d'un tableau par les pointeurs (Ex : %Yxx -> n.L)

2.5.3 Initialisation d'un tableau

L'initialisation d'un tableau est réalisée quand le moniteur charge la variable %Vxx.L ou %Yxx.L avec l'adresse de début de tableau.

Une séquence tableau doit donc avoir été exécutée une fois par le système avant de pouvoir être utilisée dans le programme par une séquence réseau.

2.6 La séquence chaîne de caractères

2.6.1 Présentation

Ce type de séquence contient des chaînes de caractères qui seront traitées principalement par les fonctions d'affichages à l'écran de la CN.

Cette séquence se compose :
- d'un label et d'un commentaire facultatif,
- d'une étape grafcet facultative,
- de la définition d'une à 32 chaînes de caractères.

La définition d'une chaîne de caractères se compose :
- d'une variable %Vxx.L ou %Yxx.L associée qui va contenir l'adresse de début de la chaîne,
- d'une suite de 120 caractères alphanumérique maximum.

Le nombre de séquences chaînes que peut contenir un module est limité par la taille maximum du module et le nombre de séquences doit être inférieur à 100.

Le compilateur ajoute automatiquement un octet nul à la fin d'une chaîne.

2.6.2 Utilisation d'une chaîne

On ne peut pas accéder directement aux caractères d'une chaîne.

L'accès se fait par l'intermédiaire de l'adresse de début de la chaîne contenu dansune variable %Vxx.L ou %Yxx.L et grâce aux fonctions du type printf(..), scano (..), scanu (..), .. etc ...

On peut accéder directement aux caractères d'une chaîne par les pointeurs (Ex : %Yxx -> n.B)

2.6.3 Initialisation d'une chaîne

L'initialisation d'une chaîne est réalisée quand le moniteur charge la variable %Vxx.L ou %Yxx.L avec l'adresse de début de la chaîne.

Une séquence chaîne doit donc avoir été exécutée une fois par le système avant de pouvoir être utilisée dans le programme par une séquence réseau.

On peut associer une même variable %Vxx.L ou %Yxx.L avec des chaînes différentes et situées dans des séquences différentes. Dans ce cas la variable % contient l'adresse de la chaîne située dans la séquence chaîne qui a été exécutée en dernier.

2.7 La séquence réseau

Ce type de séquence est l'entité de base du programme d'automatisme. La séquence réseau est composé de contacts, de dérivations et de bobines.

Un réseau de contacts est composé :
- d'un label et d'un commentaire facultatif,
- d'une étape grafcet facultative,
- d'une zone de test de six lignes de six contacts (36 cellules),
- d'une zone d'action de six lignes d'une bobine (Six cellules).

3 Variables

3.11 Zone d’échange

3.1 Principe des échanges

Les échanges entre la fonction automatisme et la fonction CN s'effectuent par l'intermédiaire d'une zone mémoire accessible au deux fonctions appelée «zone d'échange».

Les échanges avec les cartes Entrées/Sorties TOR sont traités directement par la fonction automatisme.

REMARQUE : Les termes Entrées/Sorties sont définis par rapport à la fonction automatisme. Une entrée est une variable lue par la fonction automatisme. Une sortie est une variable écrite par la fonction automatisme.

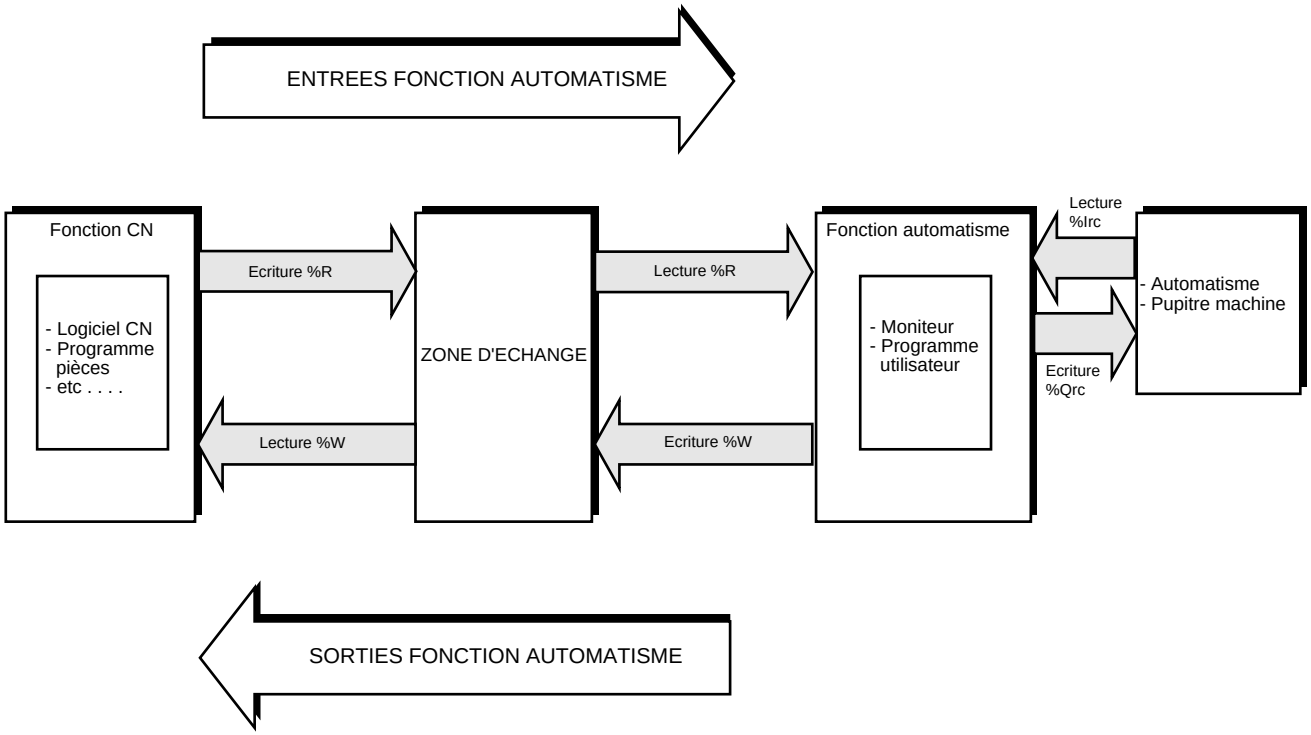


Figure 3.1 - Principe des échanges

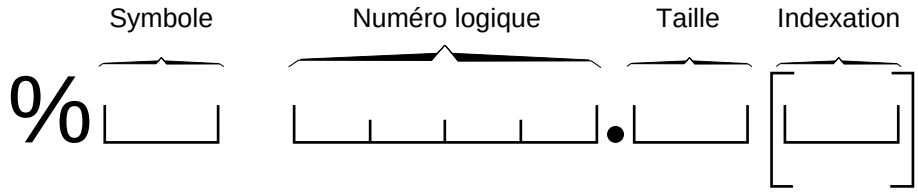
3.2 Variable % - Mnémonique

Une variable possède deux types de représentation :
- une représentation qui commence toujours par le caractère %. Cette représentation permet au compilateur de déterminer l'adresse physique de la variable,
- une représentation utilisateur facultative appelée aussi mnémonique. Cette représentation ne peut pas commencer par le caractère %.

L'utilisateur peut associer un mnémonique et une variable % dans une table de symboles (Voir manuel PLCTOOL - Outil de programmation).

3.3 Variable %

Ce type de variable commence toujours par le caractère % suivi des champs :
- Symbole,
- Numéro logique,
- Taille,
- Indexation.



3.3.1 Champ symbole

Ce champ est obligatoire.

Ce champ indique la famille de la variable

Valeur champ	Définition
%M	Pour les variables internes banalisées sauvegardées
%V	Pour les variables internes banalisées non sauvegardées
%I	Pour les variables en lecture de l'interface E/S borniers
%Q	Pour les variables en écriture de l'interface E/S borniers
%R	Pour les variables en lecture de l'interface E/S CN
%W	Pour les variables en écriture de l'interface E/S CN
%S	Pour les variables mots communs
%Y	Pour les variables locales

3.3.2 Champ numéro logique

Ce champ est obligatoire.

Ce champ permet de désigner un objet à l'intérieur d'une famille. Le numéro logique est codé en hexadécimal sur 4 chiffres maximum.

Le numéro logique représente l'adresse logique en OCTETS depuis le premier élément de la famille.

Exemples

%M9 pointe l'octet N° 9 dans la famille des variables internes %M.

%MA pointe l'octet N° 10 dans la famille des variables internes %M.

3.3.3 Champ taille

Ce champ commence par un point (.) suivi par un des caractères alphanumériques suivants :

Valeur champ	Définition
.n	Désigne le bit n (de 0 à 7) de l'octet (le bit 0 est le bit de poids faible, le bit 7 est le bit de poids fort)
.B	Désigne l'entier signé sur 8 bits
.W	Désigne l'entier signé sur 16 bits (poids fort à l'adresse n, poids faible à l'adresse n+1)
.L	Désigne l'entier signé sur 32 bits (poids fort à l'adresse n, poids faible à l'adresse n+3)
.&	Désigne l'adresse de la variable. Une adresse est codée sur 32 bits

3.3.4 Champ indexation

Ce champ est facultatif.

L'index est mis entre crochets [ ] après le champ taille.

L'index est une variable %M de taille .W (Ex : %M34.L[%M5.W]).

La valeur de l'index est ajoutée au numéro logique de la variable de base pour trouver l'adresse de la variable indexée.

Exemple

Si %M2.W = 4

Alors: %M8.L[%M2.W] désigne %MC.L .

⚠ ATTENTION

L'indexation est interdite avec une variable .&.

Exemple

%M34.&[%M4.W] est Interdit.

3.3.4.1 Indexation avec la variable Bit

L'indexation des variables bit agit sur l'adresse de l'octet mais ne modifie pas l'emplacement du bit à l'intérieur de l'octet.

Exemple

Si %M2.W = 4

Alors %M8.7[%M2.W] désigne %MC.7 .

3.4 Mnémonique

Un mnémonique est une combinaison de 12 caractères maximum choisis parmi :
- les 26 lettres majuscules (A, B, C .... Z),
- les 26 lettres minuscules (a, b, c, ... z),
- les 10 chiffres (0, 1, 2, ... 9),
- le souligné (_).

Un mnémonique doit commencer par une lettre (le souligné est exclu). Le compilateur ne fait pas la différence entre majuscule et minuscule. L'utilisateur doit associer un mnémonique avec une variable %.

Ces associations sont sauvegardées dans les fichiers de symboles (*.XSY) de PLCTOOL.

3.4.1 Champ de coercition

Lors de l'utilisation d'un mnémonique, il est possible de spécifier des variables de taille différente de celles indiquées lors de l'association mnémonique/variable %.

La coercition est indiquée après le mnémonique par un point (.) suivi du symbole de la nouvelle taille.

Exemple

Si le mnémomique «Mot_état» est associé à la variable %M3.B,

Alors : le mnémomique «Mot_état.0» représente %M3.0
le mnémomique «Mot_état.7» représente %M3.7
le mnémomique «Mot_état.W» représente %M3.W
le mnémomique «Mot_état.L» représente %M3.L

3.5 Variables internes banalisées sauvegardées

Ce sont les variables de %M0 à %M77FF (soit 30 koctets).

Les variables %M sont sauvegardées pendant une coupure secteur.

3.6 Variables internes banalisées non sauvegardées

Ce sont les variables %V0 à %V7FFF (soit 32 koctets).

Ces variables %V ne sont pas sauvegardées pendant une coupure secteur ou sur une INIT de l'unité centrale. Elle sont remises à zéro à l'initialisation de l'unité centrale.

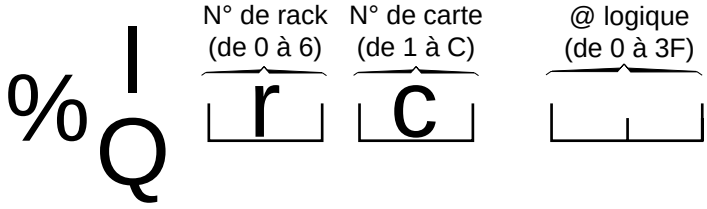
3.7 Variables E/S borniers %I et %Q

Ce type de variable est associé aux éléments suivant :
- cartes 32 entrées TOR,
- cartes 32 sorties TOR,
- cartes 32 entrées 24 sorties TOR,
- cartes 32-24 I/O,
- carte 64-48 I/O,
- pupitre machine,
- carte d'extension 32 entrées 24 sorties TOR du pupitre machine.

Chaque carte E/S TOR borniers se compose :
- d'un bloc de 64 octets de variables %I en lecture,
- d'un bloc de 64 octets de variables %Q en écriture.

Les cartes E/S sont adressées logiquement (Voir 3.7.4) sur 4 digits. Par défaut (sans configuration), on à :

@ logique = @ géographique



Le numéro de carte «c» et le numéro de rack «r» sont liés au type d'équipement. Se reporter au Manuel d'installation et de mise en oeuvre pour l'adressage des racks.

Type d'équipement	Numéro de rack	Numéro de carte
Rack principal 19"	0	5 à C
Rack principal 12"	0	5 à 8
Rack d'extension 12 cartes	1 à 6	1 à C
Rack d'extension 2 cartes	1 à 6	1 et 2
Pupitre machine	0	1 à 4

Exemple

%I3500 Représente l'octet 0 en lecture de la carte 5 située dans le rack 3.

%Q352F Représente l'octet 0x2F en écriture de la carte 5 située dans le rack 3.

REMARQUE : Un adressage logique est également possible (Voir 3.7.4).

3.7.1 Structure des variables en lecture %Irc

Le poste des variables en lecture %Irc (pour les carte 0 à C) est divisé en deux parties :
- la partie diagnostic de la carte,
- la partie image de la carte.

3.7.1.1 Partie diagnostic carte

Cette partie regroupe des variables de diagnostic qui sont lues par l'utilisateur.

Les informations sont situées aux adresses logiques hautes (%Irc3F,%Irc3E, .. etc ...).

La structure est identique pour tous les types de cartes.

3.7.1.2 Partie image des entrées

Cette partie regroupe les images des entrées de la carte. Les images des entrées sont situées aux adresses logiques basses (%Irc00, %Irc01, ... etc ...). La structure dépend du type de la carte.

3.7.2 Structure des variables en écriture %Qrc

Le poste des variables en écriture %Qrc (pour les cartes 0 à C) est divisé en deux parties :
- la partie configuration de la carte,
- la partie image de la carte.

3.7.2.1 Partie configuration carte

Cette partie regroupe des variables de configuration qui sont écrites par l'utilisateur.

Les informations sont situées aux adresses logiques hautes (%Qrc3F, %Qrc3E, .. etc ...).

La configuration des cartes E/S borniers doit être programmée dans la tâche initialisation %INI.

Le moniteur prend en compte la configuration à la fin de la tâche % INI, toute modification ultérieure de la configuration ne sera donc pas prise en compte par le moniteur.

La structure est la même pour tous les types de cartes.

3.7.2.2 Partie image des sorties

Cette partie regroupe les images des sorties de la carte.

Les images des sorties sont situées aux adresses logiques basses (%Qrc00 , %Qrc01, .. etc ...).

La structure dépend du type de la carte. Se reporter aux paragraphes suivants pour la structure de chaque carte.

3.7.3 Variables de diagnostic carte

3.7.3.1 Identificateur carte %Irc3E.W

Ce mot est écrit par le moniteur après interrogation de la carte.

%Irc3E.W == 0x700 indique une absence de carte.

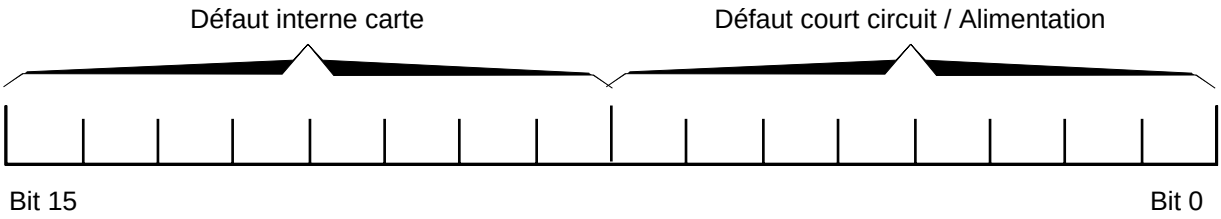
Exemple:

%I123E.W Contient l'identificateur de la carte 2 du rack 1.

3.7.3.2 Status carte %Irc3C.W

Ce mot renseigne l'utilisateur sur l'état interne de la carte. Cette fonctionnalité est disponible uniquement sur les cartes 32E / 24S, 32-24 I/O, 64-48 I/O et les cartes d'extensions 32E / 24S du pupitre machine.

Le registre %Irc3C.W se décompose comme suit :



Si aucun défaut n'est détecté, ce registre à pour valeur 0x00FF

Le contrôle du status interne de la carte est effectué périodiquement. Le programmeur peut fixer cette période par la fonction DIAGIQ().

En cas de détection de problème, ce registre ne sera plus rafraîchi. L'utilisateur doit forcer le registre à la valeur 0x00FF pour qu'il soit de nouveau rafraîchi.

REMARQUE : Si il y a détection d'un défaut interne carte, il y a montée du bit défaut général cartes E/S borniers %R97F.2 (DEFCARTE).

3.7.3.3 Compteur défaut dialogue %Irc3A.W

Ce mot s'incrémente chaque fois qu'un défaut liaison ou défaut carte est détecté lors de l'interrogation d'une carte. Ce compteur se bloque à 0x7FFF.

3.7.3.4 Status bus %Irc39.B

Cet octet renseigne l'utilisateur sur l'état de la liaison sur le bus E/S série :
- 0 fonctionnement correct,
- 1 pas de trame écho,
- 2 erreur check-sum sur trame écho,
- 3 pas de trame réponse,
- 4 erreur check-sum sur trame réponse,
- 5 fibre optique interrompue,
- 6 autres erreurs.

Les bits de défaut interne carte recouvrent les bits de défaut liaison entrée et les bits de défaut liaison sortie (voir détails pour chaque type de carte supportant cette fonctionnalité).

Si des bits de liaison entrée sont à 1, l'état des bits de défaut alimentation correspondant est non significatif.

Si des bits de liaison sortie sont à 1, l'état des bits de défaut court-circuit correspondant est non significatif.

REMARQUE : Si quatre défauts de transmission consécutifs sur la même carte se produisent, il y a montée du bit défaut général liaison sur le bus E/S série %R97F.0 (DEFBUS) et le chien de garde retombe.
Si les défauts de transmission se produisent à l'initialisation de l'unité centrale, il y a montée du bit défaut général liaison sur le bus E/S série %R97F.0 (DEFBUS) et le chien de garde n'est pas validé.

3.7.4 Variables de configuration carte

3.7.4.1 Identificateur carte %Qrc3E.W

Ce mot indique le type de carte que l'utilisateur s'attend à trouver à l'emplacement de numéro de rack r et de numéro de carte c.Il doit être programmé dans une tâche %INI.

Il permet de contrôler la conformité de la configuration carte/rack d'une application. Ce contrôle est effectué par comparaison avec les valeurs lues dans %Irc3E.W.

REMARQUE : Si il y a divergence entre la configuration prévue %Qrc3E.W et la configuration réelle %Irc3E.W, alors il y a montée du bit défaut général configuration cartes E/S borniers %R97F.1 (DEFCONF), les entrées/sorties ne sont plus rafraîchies et le chien de garde n'est pas validé.

%Qrc3E.W est initialisé à 0x700. Cette valeur indique une absence de configuration de la carte. Dans ce cas le moniteur traite la carte uniquement si elle est présente dans le rack.

Exemple

%Q123E.W == Contient l'identificateur de la carte que l'on s'attend à trouver à l'emplacement 2 du rack 1.

3.7.4.2 Option adresse logique géographique %Qrc3D.B

Permet de choisir la carte physique associée avec le poste %Irc et %Qrc. Cette option permet de gérer facilement les évolutions physiques du système sans modifier dans le programme les variables d'entrées/sorties.

Si l'octet %Qrc3D.B == r'c', alors la carte physique associée au poste %Irc et %Qrc est la carte c' du rack r'.

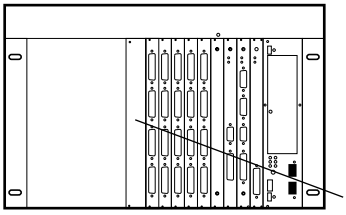
Le chargement de %Qrc3D.B avec r'c' doit se faire dans la tâche %INI, le système prend en compte %Qrc3D.B uniquement au retour de %INI.

Lorsque r' n'indique pas un rack configuré ou c' n'indique pas une carte configurée(Erreur «ERR_CONFIG_SBCE»), il y a montée du bit défaut général configuration cartes E/S borniers %R97F.1 (DEFCONF), les entrées/sorties ne sont plus rafraîchies et le chien de garde n'est pas validé.

Le système initialise par défaut l'octet %Qrc3D.B à la valeur rc (soit adresse logique = adresse géographique). La reconnaissance géographique du bus étant effectué avant la tâche %INI, l'utilisateur peut exploiter le mot %Irc3E.W (Identificateur carte) dans cette tâche. S'il y a utilisation de l'option adressage logique (%Qrc3D.B), l'identificateur lu sur le bus sera déplacé en conséquence dans la table d'entrées/sorties.

Exemple

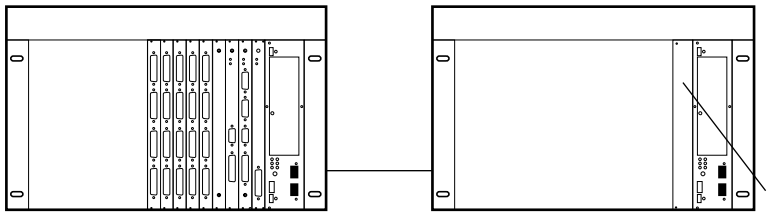
Dans la configuration de base, on a une carte 32E/24S à l'emplacement 7 dans le rack principal.



Carte 32E/24S rack 0 emplacement 7

Pour cette carte, le programme utilisateur est écrit avec les variables %I07xx.x et %Q07xx.x.

Une évolution de configuration oblige à déplacer la carte du rack 0 emplacement 7 vers le rack d'extension 1 emplacement 1.



Carte 32E/24S déplacée rack 1 emplacement 1

Pour conserver le programme utilisateur inchangé, il faut programmer dans %INI :
- %Q073D.B = 0x11
- %Q073E.W = 0x1500 (identificateur carte 32E/24S)

REMARQUE : La programmation de l'identificateur carte est obligatoire.

Dans ce cas, on a :
- 0x07xx qui est l'adresse logique,
- 0x11xx qui est l'adresse géographique (@ physique).

3.7.4.3 Priorité carte %Qrc3C.B

Doit être programmée dans la tâche %INI. Cet octet permet de fixer une priorité à la carte. Il permet d'associer une carte avec une tâche systématique %TS0, %TS1 à %TS4 ou %TS5.

Cette possibilité permet de diminuer le traitement systématique à chaque HTR.

Valeur de l'octet	Périodicité de traitement
0	La carte est traitée toutes les HTR.
1	La carte est traitée toutes les 5 HTR en phase avec %TS1.
2	La carte est traitée toutes les 5 HTR en phase avec %TS2.
3	La carte est traitée toutes les 5 HTR en phase avec %TS3.
4	La carte est traitée toutes les 5 HTR en phase avec %TS4.
5	La carte est traitée toutes les 5 HTR dans la tâche système %TS5.

Les entrées des cartes de priorité i = 1, 2, 3, 4 sont lues avant l'appel de %TSi.

Les sorties des cartes de priorité i = 1, 2, 3, 4 sont écrites en fin de %TSi.

Si l'octet priorité n'est pas compris entre 0 et 5, la carte n'est pas rafraîchie périodiquement par le moniteur. Son accès est cependant possible par les fonctions de lecture et écriture explicite (Voir 10.2 fonction read_i(...) et 10.3 fonction write_q(...)).

Le système initialise par défaut l'octet de priorité à la valeur 0.

3.7.4.4 Chien de garde %Qrc3B.0

Doit être programmé dans la tâche %INI. Lorsqu'il est à un, ce bit indique que la sortie %Qrc00.0 de cette carte est une sortie chien de garde.

Deux chiens de garde sont autorisés; le moniteur scrute l'ensemble des variables %Qrc3B.0 et sélectionne les deux premiers chiens de garde programmés dans l'ordre (r,c) croissant.

En cas de défaut d'initialisation des chiens de garde, il y a montée du bit défaut général configuration cartes E/S borniers %R97F.1 (DEFCONF), les entrées/sorties ne sont plus rafraîchies et les chiens de garde ne sont pas validés.

3.7.4.5 Autorisation accès CN %Qrc3B.1

Valide ou invalide l'accès aux cartes sorties (par les paramètres E33xxx) et aux cartes entrées (par les paramètres E43xxx) en programmation pièce.

La variable à 0 interdit l'accès à la carte par programmation pièce.

La variable à 1 autorise l'accès à la carte par programmation pièce.

Par défaut la variable %Qrc3B.1 est positionnée à 0 par le moniteur.

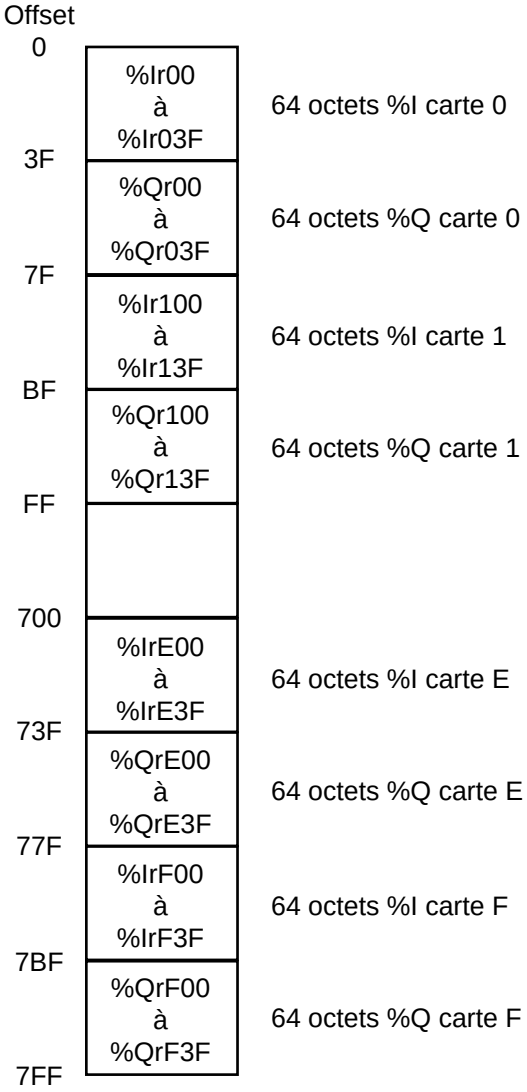
REMARQUE: %Qrc3B.1 doit être programmé dans le %INI. Suivant l'état de la variable %W900.0, l'accès aux sorties est possible ou non par E33xxx.

3.7.5 Organisation physique des variables %I et %Q

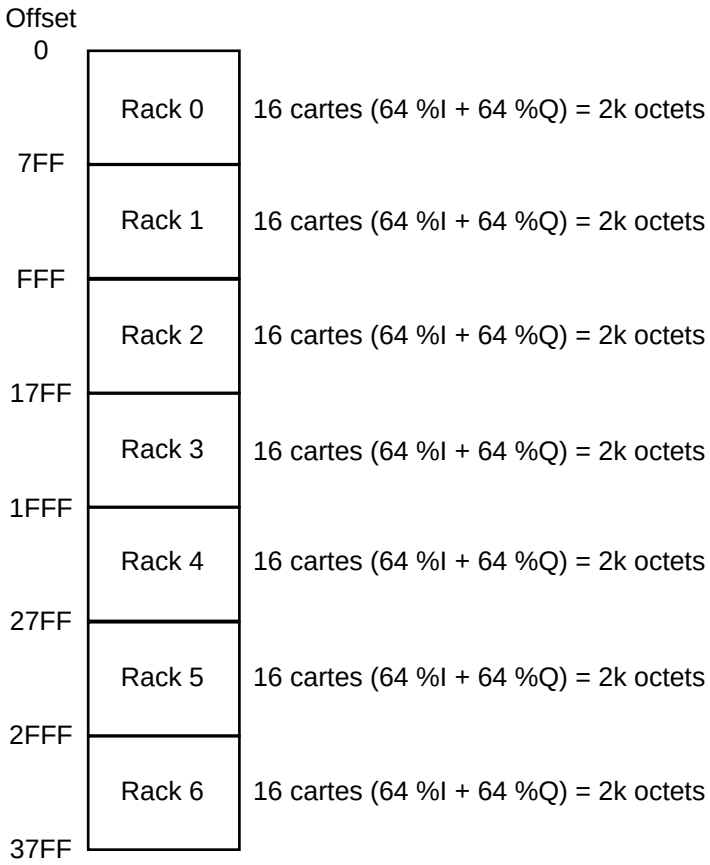
Les variables %I et %Q sont organisées en blocs mémoire de 64 octets %I suivis de 64 octets %Q correspondant à une carte et ainsi de suite jusqu'à la dernière carte du rack.

Les racks sont consécutifs et contigus (du rack 0 au rack 6).

3.7.5.1 Organisation physique des variables %I et %Q du rack N° r



3.7.5.2 Organisation physique des variables %I et %Q des différents racks



3.7.6 Identificateur des cartes et racks

3.7.6.1 Identificateurs des cartes

Cartes 1060

Type de carte	Valeur de %Irc3E.W et Qrc3E.W
Carte 32 Entrées	0x0A00
Carte 32 Entrées V2	0x0A10
Carte 32 Sorties	0x0100
Carte 32 Sorties V2	0x0110
Carte 32 Entrées 24 Sorties	0x1500
Carte 32-24 I/O	0x0F00
Carte 32-24 80 mA	0x0F10
Carte 64-48 I/O	0x0300
Carte 64-48 I/O 80 mA	0x0310
Pupitre machine	0x02C0
Pupitre machine avec extension	0x0200
Absence de carte	0x0700

Cartes 1020/1040/1050

Type de carte	Valeur de %Irc3E.W et Qrc3E.W
Carte 32 24 I/O 80mA	0x2100
Carte 64-48 I/O 80mA	0x2000

3.7.6.2 Identificateurs des racks

Rack 1060

REMARQUE : Les composants matériels racks (alimentation + tôlerie + bus) correspondent à la carte N° 0.

Type de rack	Nb de cartes	Alimentation	Fibre optique	Valeur de l'identificateur %Ir03E.W
Principal	8	130 W	Oui	0x0
Principal	8	130 W	Non	0x80
Principal	8	60 W	Oui	0x10
Principal	8	60 W	Non	0x90
Principal	4	130 W	Oui	0x3000
Principal	4	130 W	Non	0x3080
Principal	4	60 W	Oui	0x3010
Principal	4	60 W	Non	0x3090
Extension 12	12	130 W		0x1000
Extension 12	12	60 W		0x1010
Extension 2	2			0x2020

Rack 1020/1040

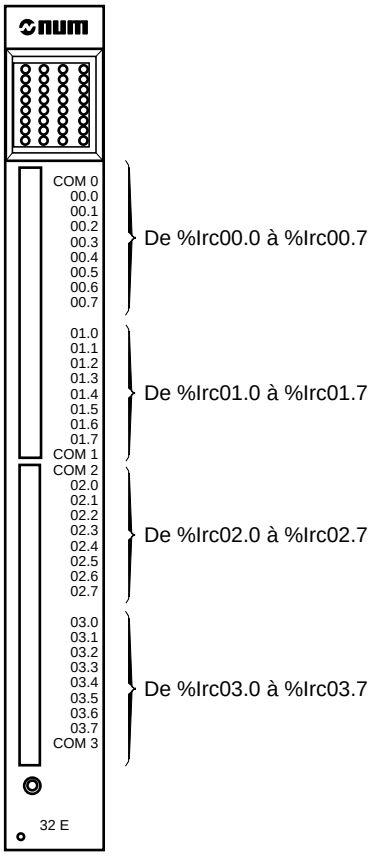
Fibre optique	Valeur de l'identificateur %Ir03E.W
Oui	0x40B0
Non	0x4030

3.7.7 Partie image de la carte 32 entrées TOR

Identificateur carte 32 entrées %Irc3E.W == 0x0A00.

Identificateur carte 32 entrées V2 %Irc3E.W == 0x0A10.

Type de variable	Type d'entrées	Variables
%Irc00	Entrées TOR de 0 à 7	%Irc00.0 (Entrée 00.0) à %Irc00.7 (Entrée 00.7)
%Irc01	Entrées TOR de 8 à 15	%Irc01.0 (Entrée 01.0) à %Irc01.7 (Entrée 01.7)
%Irc02	Entrées TOR de 16 à 23	%Irc02.0 (Entrée 02.0) à %Irc02.7 (Entrée 02.7)
%Irc03	Entrées TOR de 24 à 31	%Irc03.0 (Entrée 03.0) à %Irc03.7 (Entrée 03.7)

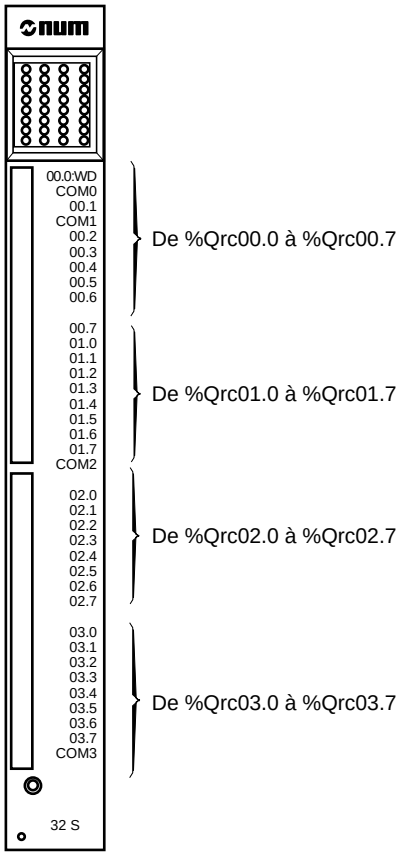


3.7.8 Partie image de la carte 32 sorties TOR

Identificateur carte 32 sorties %Irc3E.W == 0x0100.

Identificateur carte 32 sorties V2 %Irc3E.W == 0x0110.

Type de variable	Type de sorties	Variables
%Qrc00	Sorties TOR 0 à 7	%Qrc00.0 (Sortie 00.0) à %Qrc00.7 (Sortie 00.7)
%Qrc01	Sorties TOR 8 à 15	%Qrc01.0 (Sortie 01.0) à %Qrc01.7 (Sortie 01.7)
%Qrc02	Sorties TOR 16 à 23	%Qrc02.0 (Sortie 02.0) à %Qrc02.7 (Sortie 02.7)
%Qrc03	Sorties TOR 24 à 31	%Qrc03.0 (Sortie 03.0) à %Qrc03.7 (Sortie 03.7)



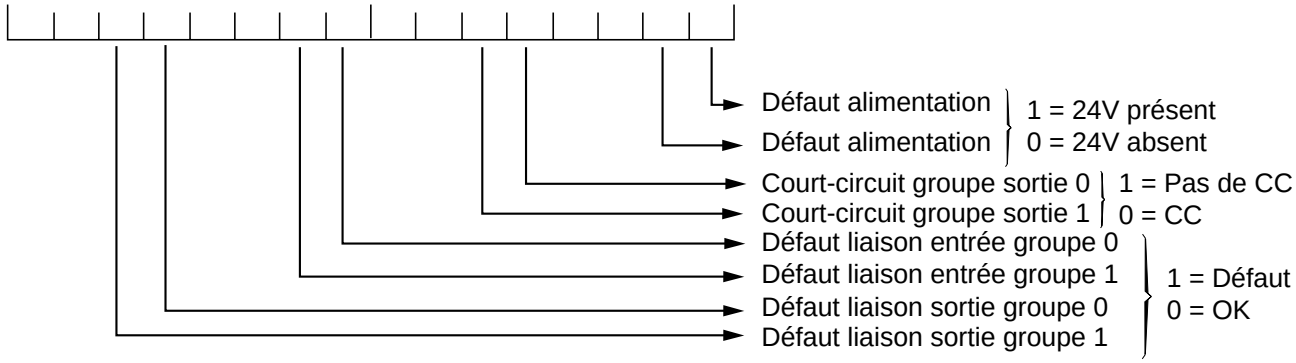
3.7.9 Partie image des cartes 32E 24S TOR et 32-24 I/O

Identificateur carte 32E 24S %Irc3E.W == 0x1500.

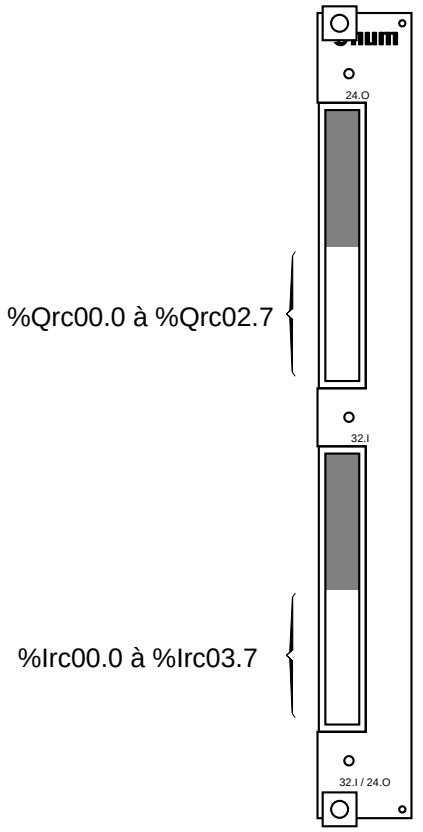
Identificateur carte 32-24 I/O %Irc3E.W == 0x0F00.

Identificateur carte 32-24 I/O 80 mA %Irc3E.W == 0x0F10.

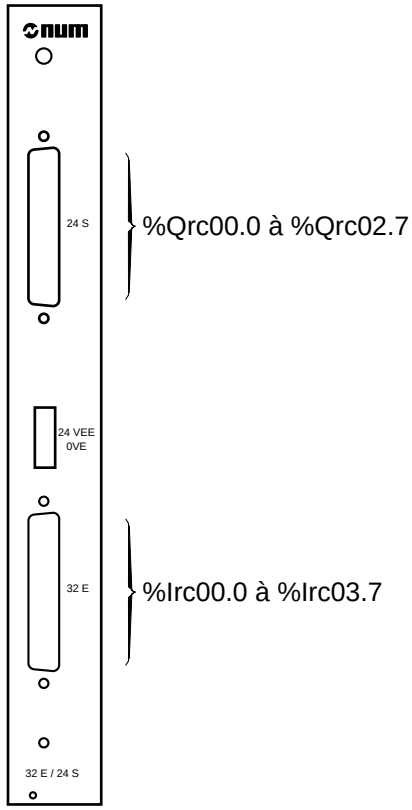
Détail du registre %Irc3C.W



Type de variable	Type d'entrées ou sorties
%Irc00	Entrées TOR 0 à 7
%Irc01	Entrées TOR 8 à 15
%Irc02	Entrées TOR 16 à 23
%Irc03	Entrées TOR 24 à 31
%Qrc00	Sorties TOR 0 à 7
%Qrc01	Sorties TOR 8 à 15
%Qrc02	Sorties TOR 16 à 23



Carte 32-24 I/O



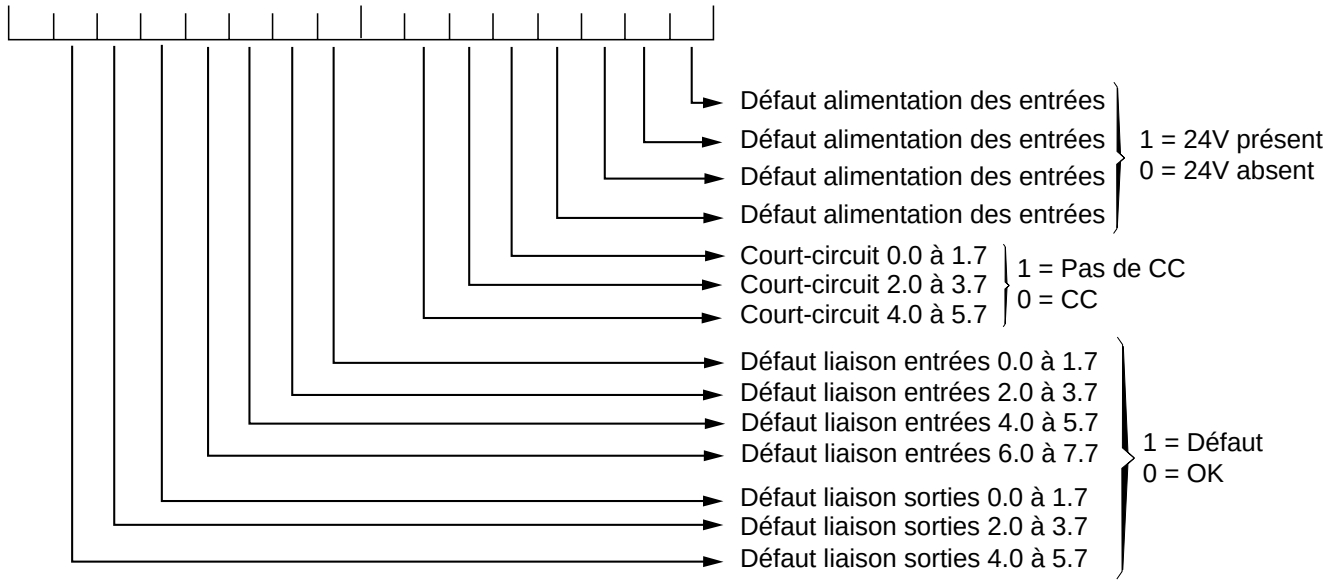
Carte 32E/24S

3.7.10 Partie image de la carte 64-48 I/O

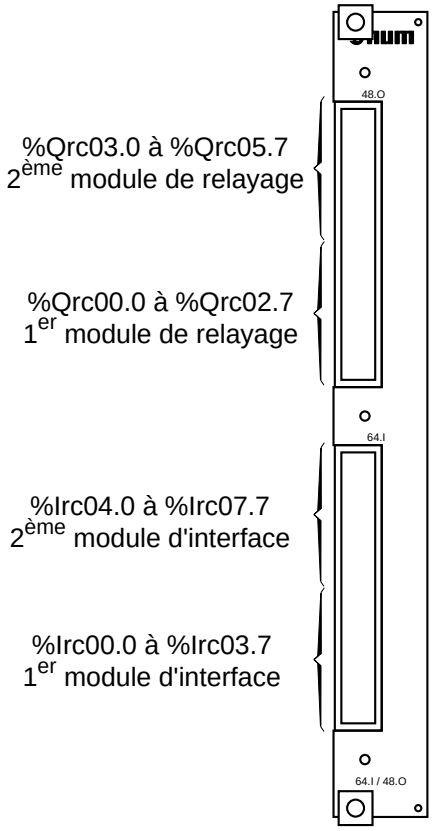
Identificateur carte 64-48 I/O %Irc3E.W == 0x0300.

Identificateur carte 64-48 I/O 80 mA %Irc3E.W == 0x0310.

Détail du registre %Irc3C.W



Type de variable	Type d'entrées ou sorties
%Irc00	Entrées TOR 0 à 7
%Irc01	Entrées TOR 8 à 15
%Irc02	Entrées TOR 16 à 23
%Irc03	Entrées TOR 24 à 31
%Irc04	Entrées TOR 32 à 39
%Irc05	Entrées TOR 40 à 47
%Irc06	Entrées TOR 48 à 55
%Irc07	Entrées TOR 56 à 63
%Qrc00	Sorties TOR 0 à 7
%Qrc01	Sorties TOR 8 à 15
%Qrc02	Sorties TOR 16 à 23
%Qrc03	Sorties TOR 24 à 31
%Qrc04	Sorties TOR 32 à 39
%Qrc05	Sorties TOR 40 à 47

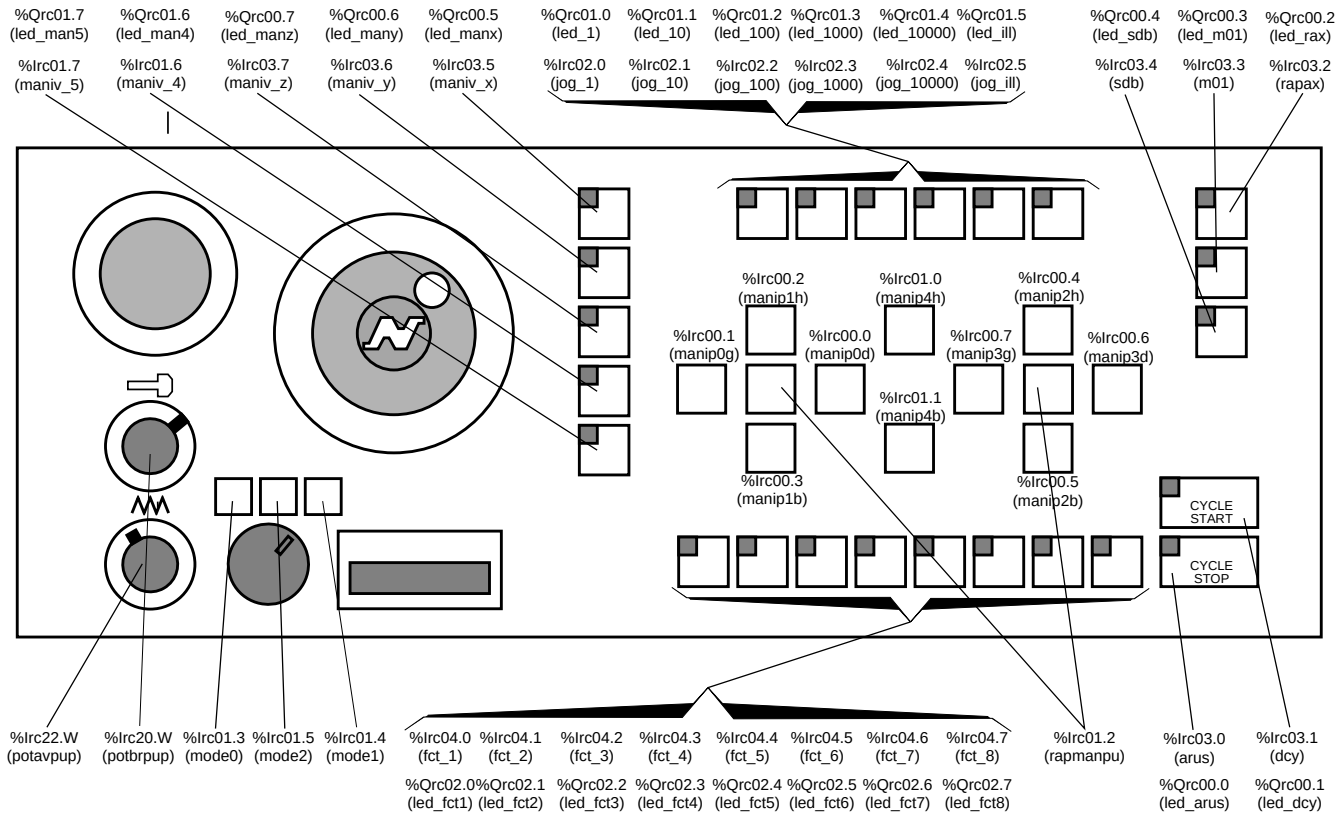


Carte 64-48 I/O

3.7.11 Partie image du pupitre machine

Identificateur carte %Irc3E.W == 0x2C0

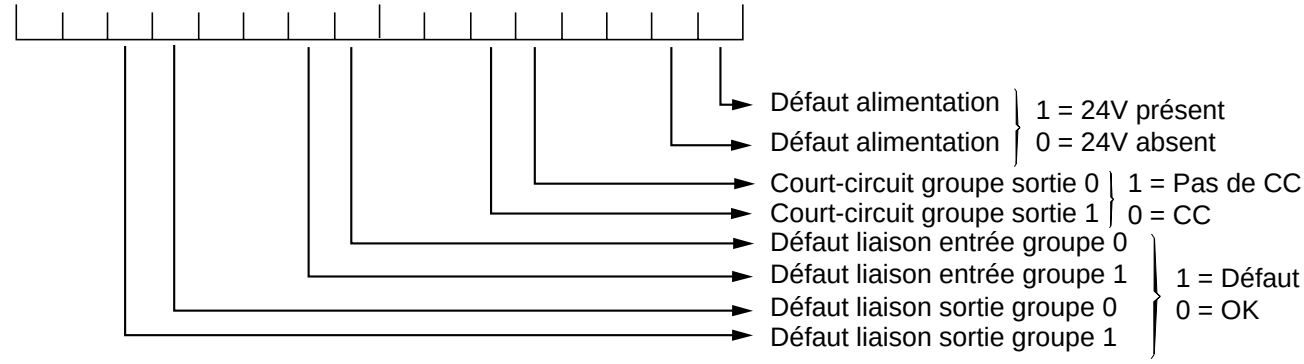
Type de variable	Type d'entrées ou sorties	Variables
%Irc00	Entrées TOR 0 à 7	%Irc00.0 (Entrée 0) à %Irc00.7 (Entrée 7)
%Irc01	Entrées TOR 8 à 15	%Irc01.0 (Entrée 8) à %Irc01.7 (Entrée 15)
%Irc02	Entrées TOR 16 à 23	%Irc02.0 (Entrée 16) à %Irc02.7 (Entrée 23)
%Irc03	Entrées TOR 24 à 31	%Irc03.0 (Entrée 24) à %Irc03.7 (Entrée 31)
%Irc04	Entrées TOR 32 à 39	%Irc04.0 (Entrée 32) à %Irc04.7 (Entrée 39)
%Irc20.W	Entrée analogique 0	
%Irc22.W	Entrée analogique 1	
%Qrc00	Sorties TOR 0 à 7	%Qrc00.0 (Sortie 0) à %Qrc00.7 (Sortie 7)
%Qrc01	Sorties TOR 8 à 15	%Qrc01.0 (Sortie 8) à %Qrc01.7 (Sortie 15)
%Qrc02	Sorties TOR 16 à 23	%Qrc02.0 (Sortie 16) à %Qrc02.7 (Sortie 23)



3.7.12 Partie image du pupitre machine avec carte d'extension

Identificateur carte %Irc3E.W == 0x200.

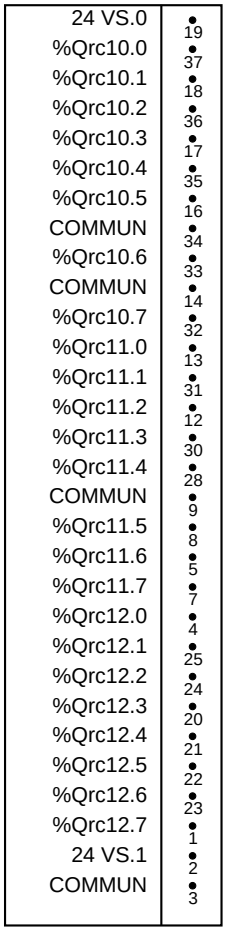
Détail du registre %Irc3C.W



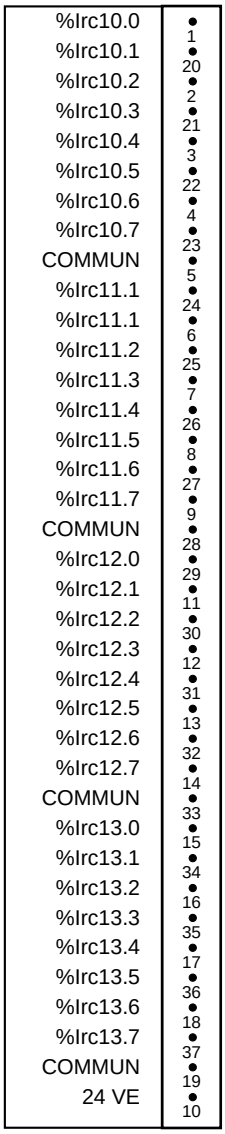
Type de variable	Type d'entrées ou sorties	Variables
%Irc00	Entrées TOR 0 à 7	%Irc00.0 (Entrée 0) à %Irc00.7 (Entrée 7)
%Irc01	Entrées TOR 8 à 15	%Irc01.0 (Entrée 8) à %Irc01.7 (Entrée 15)
%Irc02	Entrées TOR 16 à 23	%Irc02.0 (Entrée 16) à %Irc02.7 (Entrée 23)
%Irc03	Entrées TOR 24 à 31	%Irc03.0 (Entrée 24) à %Irc03.7 (Entrée 31)
%Irc04	Entrées TOR 32 à 39	%Irc04.0 (Entrée 32) à %Irc04.7 (Entrée 39)
%Irc10	Entrées TOR 40 à 47	%Irc10.0 (Entrée 40) à %Irc10.7 (Entrée 47)
%Irc11	Entrées TOR 48 à 55	%Irc11.0 (Entrée 48) à %Irc11.7 (Entrée 55)
%Irc12	Entrées TOR 56 à 63	%Irc12.0 (Entrée 56) à %Irc12.7 (Entrée 63)
%Irc13	Entrées TOR 64 à 71	%Irc13.0 (Entrée 64) à %Irc13.7 (Entrée 71)
%Irc20.W	Entrée analogique 0	
%Irc22.W	Entrée analogique 1	
%Qrc00	Sorties TOR 0 à 7	%Qrc00.0 (Sortie 0) à %Qrc00.7 (Sortie 7)
%Qrc01	Sorties TOR 8 à 15	%Qrc01.0 (Sortie 8) à %Qrc01.7 (Sortie 15)
%Qrc02	Sorties TOR 16 à 23	%Qrc02.0 (Sortie 16) à %Qrc02.7 (Sortie 23)
%Qrc10	Sorties TOR 24 à 31	%Qrc10.0 (Sortie 24) à %Qrc10.7 (Sortie 31)
%Qrc11	Sorties TOR 32 à 39	%Qrc11.0 (Sortie 32) à %Qrc11.7 (Sortie 39)
%Qrc12	Sorties TOR 40 à 47	%Qrc12.0 (Sortie 40) à %Qrc12.7 (Sortie 47)

REMARQUE *Pour réaliser le test des lampes de sorties du pupitre, il ne faut pas l'effectuer en une seule opération. Dans le programme automate, tester d'abord la moitié des lampes, puis ensuite la seconde moitié.*

Extension 24 sorties (Variables --> broches)



Extension 32 entrées (Variables --> broches)

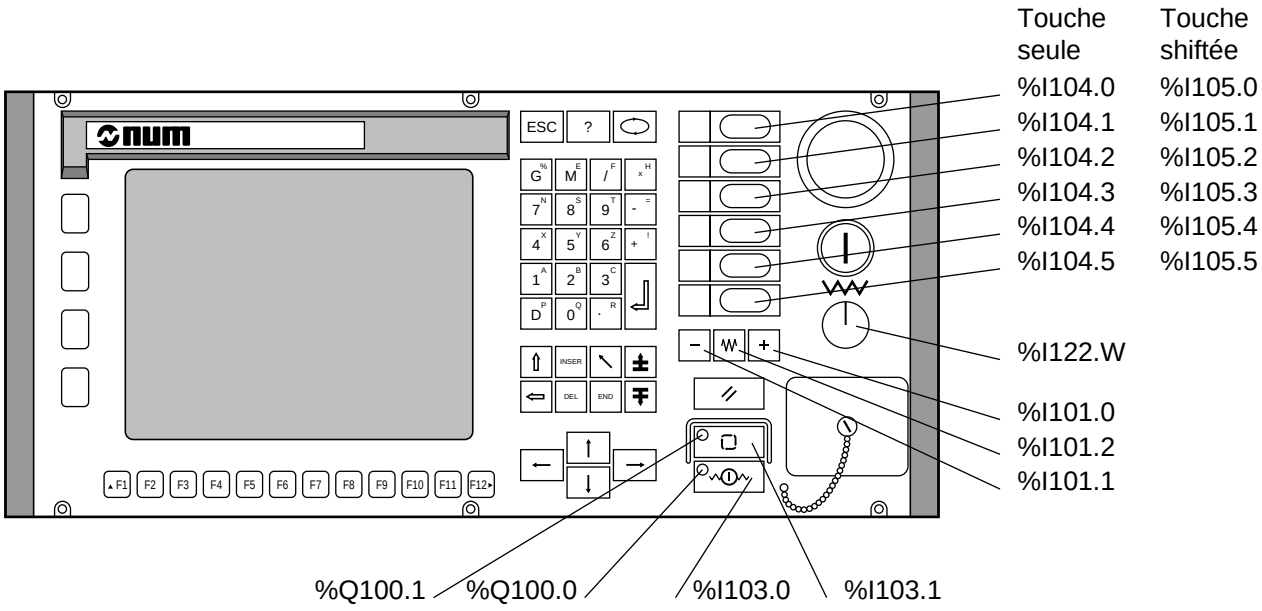


3.7.13 Partie image du pupitre compact

3.7.13.1 Image du pupitre compact dans la zone d'échange

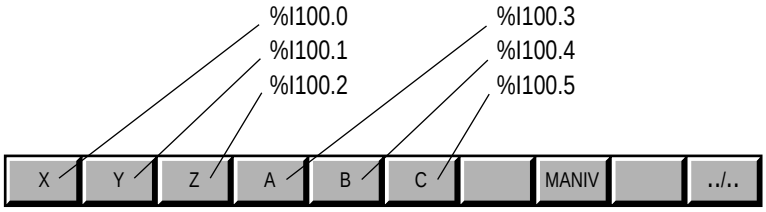
Type de variable	Type d'entrées ou sorties	Variables
%I100.B	Sélection des axes par le cartouche JOG	%I100.0 à %I100.5
%I101.B	Touches manipulateurs +, - et rapide	%I101.0 à %I101.2
%I103.B	Touches Arus et Cycle	%I103.0 (Arus) et %I103.1 (Cycle)
%I104.B	Touches personnalisables 1 à 6	%I104.0 (touche 1) à %I104.5 (touche 6)
%I105.B	Touches personnalisables shiftées 1 à 6	%I105.0 (touche 1) à %I105.5 (touche 6)
%I122.W	Entrée analogique potentiomètre	
%Q100.B	Voyants Arus et Cycle	%Q100.0 (Arus) et %Q100.1 (Cycle)
%Q102.B	Voyants des touches personnalisables 1 à 6	%Q102.0 (voyant 1) à %Q102.5 (voyant 6)
%Q103.B	Voyants des touches personnalisables shiftées 1 à 6	%Q103.0 (voyant 1) à %Q103.5 (voyant 6)

3.7.13.2 Image du pupitre compact



3.7.13.3 Image du cartouche JOG

Le pupitre compact dispose de cartouches spécifiques dont le nouveau cartouche JOG qui permet de sélectionner l'axe piloté par les manipulateurs :

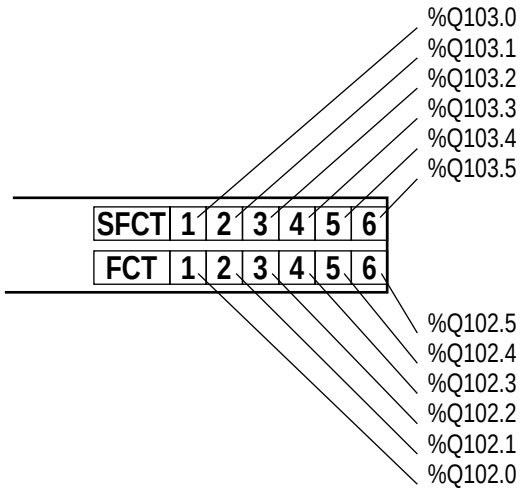


Ce cartouche est accessible par les touches : puis JOG (F7).

Les axes dont les noms figurent dans les touches du cartouche sont les axes définis par le paramètre machine P9 (Voir manuel des paramètres), ils apparaissent dans l'ordre de définition.

3.7.13.4 Image des voyants des touches personnalisables

Les fonctions activées par les touches personnalisables sont indiquées par un voyant de la fenêtre Status :



Les voyants du bas représentent les voyants des touches personnalisables seules, les voyants du haut représentent les voyants des touches personnalisables shiftées.

3.8 Famille interface E/S CN %R et %W

3.8.1 Entrées venant de la CN %R0 à %R7F.

3.8.1.1 Caractères clavier : %R0.W

Variable	Mnémonique	Description
%R0.W	CARCLAV	Reçoit le caractère frappé au clavier au rythme de %TS5 soit un caractère toutes les 5 HTR (Voir 8.1.2).

3.8.1.2 Etat Machine : %R2.W

Variable	Mnémonique	Description
%R2.7	E_M01	Arrêt programmé optionnel validé Image du champ «M01» de la fenêtre status CN. Signale la prise en compte des arrêts programmés optionnels dans un programme pièce. Inversion du bit par appui sur la touche «M01» du pupitre ou après lecture de C_M01 = 1.
%R2.6	E_SLASH	Saut de bloc validé Image du champ «/» de la fenêtre status CN. Signale la prise en compte des sauts de blocs dans un programme pièce. Inversion du bit par appui sur la touche «/» du pupitre ou après lecture de C_SLASH = 1.
%R2.5	E_INTERV	Etat intervention Après un ARUS, le passage à l'état INTERV se fait sur le premier appui sur la touche RAPPEL D'AXE du pupitre machine. Mise à 1 après lecture par la CN de C_RAX = 1. Mise à 0 après lecture par la CN de C_RAX = 0.
%R2.4	S_RECUL	Etat recul / retour sur trajectoire La mise à 1 met la CN en recul ou en retour La mise à 0 annule cet état.
%R2.1	E_NMAUTO	Fonctionnalité N/M AUTO La mise à 1 du bit signale que la fonctionnalité N/M (2/3, 3/5, .. etc ...) est active.
%R3.7	E_OPER	Image du voyant opérateur Signale un arrêt programme provoqué par M00 ou un M01 validé. Mise à 1 sur un M00 ou un M01. Mise à 0 par touche «CYCLE» du pupitre machine (C_CYCLE = 1).
%R3.6	E_DEFCN	CN en défaut Image du champ «CN??» de la fenêtre status CN. Signale un défaut machine ou un défaut de programmation pièce. Le numéro de défaut machine se lit dans ERRMACH. Mise à 1 sur apparition d'un défaut machine (Erreurs E30 à E33, E36, E40 à E71) ou un défaut de programmation pièce. Mise à 0 par touche «RAZ» du pupitre, C_RAZ = 1.
%R3.4	E_DGURG	Dégagement d'urgence général Signale l'exécution d'un programme de dégagement d'urgence. Mise à 1 après lecture par le CN de C_DGURG = 1 et si le programme de dégagement d'urgence est validé. Mise à 0 sur détection d'un M00 ou un M02.

Variable	Mnémonique	Description
%R3.3	E_RAX	Rappel d'axe général Dans le mode «INTERV» et en fin de RNS, signale que le rappel d'axe est validé. Mise à 1 après lecture par la CN de C_RAX = 1. Mise à 0 après lecture par la CN de C_RAX = 0.
%R3.2	E_CYCLE	Cycle en cours Mise à 1 par touche «CYCLE» du pupitre machine (C_CYCLE = 1). Mise à 0 par touche «RAZ» du pupitre, C_RAZ = 1 ou en fin d'exécution de programme (M02).
%R3.1	E_ARUS	Sortie arrêt d'usinage Signale l'état INTERV du système (arrêt du programme en cours d'exécution et validation des manipulateurs d'axes. Mise à 1 par touche «ARUS» du pupitre machine (C_ARUS = 1). Mise à 0 par touche «CYCLE» du pupitre machine (C_CYCLE = 0).
%R3.0	E_RAZ	Remise à zéro CN en cours Bit impulsionnnel d'une durée de 100 ms qui signale une initialisation du système. Pendant la durée de cette impulsion, les données venant de la fonction automatisme ne sont pas prises en compte. Mise à 1 par touche «RAZ» du pupitre, sur demande de RAZ de la fonction automatisme (C_RAZ = 1), en fin d'exécution d'un programme pièce (M02) ou à la mise sous tension de la CN. Cette variable est mise à 0 après 100 ms.

3.8.1.3 Etat CN : %R4.W

Variable	Mnémonique	Description
%R5.7	E_TRANSP	Mode transparent Permet à la fonction automatisme l'accès à l'écran de la CN pour y visualiser des informations (table de données, .. etc ...). La CN peut être en cours d'usinage. Mise à 1 par validation de la page écran «MODE TRANSPARENT». Mise à 0 par abandon de la page écran «MODE TRANSPARENT».
%R5.5	E_PPP	Mode passant prêt Indique que la CN est prête à fonctionner en mode passant, ou que l'usinage à effectuer commandé par la fonction automatisme, se fait en mode passant lecteur. Dans le second cas PROGDEM doit être chargé à la valeur -2 (0xFFFE). Mise à 1 après «CHOIX DU PROGRAMME COURANT» et programmation au clavier de PPR ou PPL suivi de «ENTER». Mise à 0 après «CHOIX DU PROGRAMME COURANT» et programmation au clavier de -PPR ou -PPL suivi de «ENTER».
%R5.1	E_PROG	Programme en cours Signale qu'un programme pièce est en cours d'exécution dans les modes «CONT», «SEQU», «IMD» et «RAP». Mise à 1 par une première impulsion sur la touche «CYCLE» du pupitre machine (C_CYCLE = 1). Mise à 0 par détection d'un M00 (Arrêt programmé), M01 (Arrêt programmé optionnel), M02 (Fin de programme), par touche «RAZ» du pupitre, C_RAZ = 1 ou à la mise sous tension.

Variable	Mnémonique	Description
%R5.0	E_CNPRET	CN prête Signale que la puissance peut être mise sur la machine. Mise à 1 à la mise sous tension et par touche «RAZ» du pupitre, C_RAZ = 1. Mise à 0 après détection d'une trop grande erreur de poursuite sur un axe ou défaut salissure ou de complémentarité des voies du générateur d'impulsions détecté sur un axe.

3.8.1.4 Axes en mouvements : %R6.L

Variable	Mnémonique	Description
%R6.7 à %R6.0	AXMVT31 à AXMVT24	axe N°31 en mouvement àaxe N°24 en mouvement Signale les axes 24 à 31 en mouvement pendant l'exécution d'un bloc dans un programme pièce ou en «IMD». Mise à 1 en début d'exécution du bloc. Mise à 0 en fin d'exécution du bloc s'il comporte un M00 ou un M01, en fin d'exécution du bloc en «IMD», avant d'effectuer un blocage d'axes, touche «RAZ» du pupitre, C_RAZ = 1.
%R7.7 à %R7.0	AXMVT23 à AXMVT16	axe N°23 en mouvement à axe N° 16 en mouvement Signale les axes 16 à 23 en mouvement pendant l'exécution d'un bloc dans un programme pièce ou en «IMD». Mise à 1 en début d'exécution du bloc. Mise à 0 en fin d'exécution du bloc s'il comporte un M00 ou un M01, en fin d'exécution du bloc en «IMD», avant d'effectuer un blocage d'axes, touche «RAZ» du pupitre, C_RAZ = 1.
%R8.7 à %R8.0	AXMVT15 à AXMVT8	axe N°15 en mouvement à axe N°8 en mouvement Signale les axes 8 à 15 en mouvement pendant l'exécution d'un bloc dans un programme pièce ou en «IMD». Mise à 1 en début d'exécution du bloc. Mise à 0 en fin d'exécution du bloc s'il comporte un M00 ou un M01, en fin d'exécution du bloc en «IMD», avant d'effectuer un blocage d'axes, touche «RAZ» du pupitre, C_RAZ = 1.
%R9.7 à %R9.0	AXMVT7 à AXMVT0	axe N°7 en mouvement à axe N°0 en mouvement Signale les axes 0 à 7 en mouvement pendant l'exécution d'un bloc dans un programme pièce ou en «IMD». Mise à 1 en début d'exécution du bloc. Mise à 0 en fin d'exécution du bloc s'il comporte un M00 ou un M01, en fin d'exécution du bloc en «IMD», avant d'effectuer un blocage d'axes, touche «RAZ» du pupitre, C_RAZ = 1.

3.8.1.5 Axes initialisés (POM faite) : %RA.L

Variable	Mnémonique	Description
%RA.7 à %RA.0	AXINI31 à AXINI24	axe N°31 initialisé à axe N°24 initialisé. Signale les axes dont la prise d'origine (POM) a été faite. Mise à 0 quand la prise d'origine est effectué sur l'axe correspondant Mise à 1 à l'initialisation du système (POM non faite)
%RB.7 à %RB.0	AXINI23 à AXINI16	axe N°23 initialisé à axe N°16 initialisé. Signale les axes dont la prise d'origine (POM) a été faite. Mise à 0 quand la prise d'origine est effectué sur l'axe correspondant Mise à 1 à l'initialisation du système (POM non faite)
%RC.7 %RC.0	AXINI15 AXINI8	axe N°15 initialisé à axe N°8 initialisé. Signale les axes dont la prise d'origine (POM) a été faite. Mise à 0 quand la prise d'origine est effectué sur l'axe correspondant Mise à 1 à l'initialisation du système (POM non faite)
%RD.7 à %RD.0	AXINI7 à AXINI0	axe N°7 initialisé à axe N°0 initialisé. Signale les axes dont la prise d'origine (POM) a été faite. Mise à 0 quand la prise d'origine est effectué sur l'axe correspondant Mise à 1 à l'initialisation du système (POM non faite)

3.8.1.6 Paramètres Externes E10000 à E10031 : %RE.L

Les paramètres externes E100xx sont lus par le programme utilisateur. La gestion de ces paramètres est assurée par le programme pièce qui peut les lire et les écrire.

Ils permettent d'échanger des informations booléennes entre les programmes pièce et le programme utilisateur.

Variable	Mnémonique	Variable	Mnémonique
%R11.0	E10000	%RF.0	E10016
%R11.1	E10001	%RF.1	E10017
%R11.2	E10002	%RF.2	E10018
%R11.3	E10003	%RF.3	E10019
%R11.4	E10004	%RF.4	E10020
%R11.5	E10005	%RF.5	E10021
%R11.6	E10006	%Rf.6	E10022
%R11.7	E10007	%RF.7	E10023
%R10.0	E10008	%RE.0	E10024
%R10.1	E10009	%RE.1	E10025
%R10.2	E10010	%RE.2	E10026
%R10.3	E10011	%RE.3	E10027
%R10.4	E10012	%RE.4	E10028
%R10.5	E10013	%RE.5	E10029
%R10.6	E10014	%RE.6	E10030
%R10.7	E10015	%RE.7	E10031

3.8.1.7 Etat des Broches : %R12.W

Variable	Mnémonique	Description
%R12.7	B4_ARR	Bit à 1 indique que la broche N°4 est à l'arrêt , c'est à dire que sa vitesse de rotation est inférieure au paramètre E90343 (Voir manuel de programmation)
%R12.6	B3_ARR	Bit à 1 indique que la broche N°3 est à l'arrêt , c'est à dire que sa vitesse de rotation est inférieure au paramètre E90342 (Voir manuel de programmation)
%R12.5	B2_ARR	Bit à 1 indique que la broche N°2 est à l'arrêt , c'est à dire que sa vitesse de rotation est inférieure au paramètre E90341 (Voir manuel de programmation)
%R12.4	B1_ARR	Bit à 1 indique que la broche N°1 est à l'arrêt , c'est à dire que sa vitesse de rotation est inférieure au paramètre E90340 (Voir manuel de programmation)
%R12.3	B4_ROT	Bit à 1 indique que la rotation de la broche N°4 est correcte , c'est à dire que sa vitesse de rotation est comprise dans la fourchettede tolérance de vitesse donnée par le paramètre E90353 (Voir manuel de programmation)
%R12.2	B3_ROT	Bit à 1 indique que la rotation de la broche N°3 est correcte , c'est à dire que sa vitesse de rotation est comprise dans la fourchettede tolérance de vitesse donnée par le paramètre E90352 (Voir manuel de programmation)
%R12.1	B2_ROT	Bit à 1 indique que la rotation de la broche N°2 est correcte , c'est à dire que sa vitesse de rotation est comprise dans la fourchettede tolérance de vitesse donnée par le paramètre E90351 (Voir manuel de programmation)
%R12.0	B1_ROT	Bit à 1 indique que la rotation de la broche N°1 est correcte , c'est à dire que sa vitesse de rotation est comprise dans la fourchettede tolérance de vitesse donnée par le paramètre E90350 (Voir manuel de programmation)
%R13.3	POSBR4	Broche N°4 en position Sur une demande d'indexation ou de synchronisation de broche, signale que la broche N°4 est en position ou synchronisée. Mise à 1 lorsque la position demandée est atteinte. Mise à 0 lorsque la position est quittée, sur des oscillations et fonction M19 révoquée.
%R13.2	POSBR3	Broche N°3 en position Sur une demande d'indexation ou de synchronisation de broche, signale que la broche N°3 est en position ou synchronisée. Mise à 1 lorsque la position demandée est atteinte. Mise à 0 lorsque la position est quittée, sur des oscillations et fonction M19 révoquée.
%R13.1	POSBR2	Broche N°2 en position Sur une demande d'indexation ou de synchronisation de broche, signale que la broche N°2 est en position ou synchronisée. Mise à 1 lorsque la position demandée est atteinte. Mise à 0 lorsque la position est quittée, sur des oscillations et fonction M19 révoquée.
%R13.0	POSBR1	Broche N°1 en position Sur une demande d'indexation ou de synchronisation de broche, signale que la broche N°1 est en position ou synchronisée. Mise à 1 lorsque la position demandée est atteinte. Mise à 0 lorsque la position est quittée, sur des oscillations et fonction M19 révoquée.

3.8.1.8 Type d'incrément de JOG : %R15.B

Variable	Mnémonique	Description
%R15.B	E_INCJOG	Incrément de JOG en cours La valeur de la variable est l'image de l'incrément de JOG en cours : 0x0A Déplacement manuel au pas de 10^{-6} pouce 0x09 Déplacement manuel au pas de 10^{-2} µm ou 10^{-5} pouce 0x00 Déplacement manuel au pas de 10^{-1} µm ou 10^{-4} pouce 0x01 Déplacement manuel au pas de 1µm ou 10^{-3} pouce 0x02 Déplacement manuel au pas de 10 µm ou 10^{-2} pouce 0x03 Déplacement manuel au pas de 100 µm ou 10^{-1} pouce 0x04 Déplacement manuel au pas de 1000 µm ou 1 pouce 0x05 Déplacement manuel au pas de 10000 µm ou 1 pouce 0x06 Déplacement manuel en continu 0x08 Déplacement manuel par manivelle Le choix de la valeur de l'incrément en µm ou en pouce dépend de la valeur de la variable C_UNIT.

3.8.1.9 Mode en cours : %R16.B

Variable	Mnémonique	Description
%R16.B	MODCOUR	Mode en cours La valeur de la variable est l'image du mode CN en cours : 0x00 Mode Continu «CONT» 0x01 Mode Séquentiel «SEQ» 0x02 Mode Immédiat «IMD» 0x03 Mode Rapide «RAP» 0x04 Mode Recherche de Numéro de Séquence «RNS» 0x05 Mode Modification «MODIF» 0x06 Mode Test «TEST» 0x07 Mode Manuel «MANU» 0x08 Mode Prise d'Origine Mesure «POM» 0x09 Mode Prise de Référence «PREF» 0x0A Mode Réglages Outils «REGOUT» 0x0B Absence de mode 0x0D Mode Chargement «CHARG» 0x0F Mode Déchargement «DECHG» 0x10 Mode spécifiant «groupes indépendnts»

3.8.1.10 Variables diverses

Variable	Mnémonique	Description
%R14.1	E_BAT	Etat des batteries E_BAT = 0 batteries OK E_BAT = 1 batteries à changer.
%R14.0	SC_USED	Validation écran en configuration PCNC La mise à 1 indique que l'écran est utilisé par une application utilisateur (mode transparent bloqué). La mise à 0 indique que l'écran est utilisé par l'application CN NUM (mode transparent possible)
%R17.B	PGVISU	Numéro de la page visualisée Cette variable est l'image de la page visualisée à l'écran de la CN : 0x01 Page liste «LISTE» 0x03 Page programme «PROG.» 0x04 Page informations «INFO» 0x05 Page variables programmes «L/@» 0x06 Pages point courant «AXES» 0x07 Page corrections d'outils «OUTILS» 0x08 Page programmation graphique «PROCAM» 0x19 Page chargement en cours d'usinage 0x1A Page déchargement en cours d'usinage 0x09 Page entrées/sorties»E/S» 0x0A Page des utilitaires «UTIL» 0x15 Page des décalages «PREF» 0x17 Page du mode modif 0x0E Page du mode chargement 0x11 Page du mode déchargement 0x1B Mode transparent appelé directement par PUTKEY
%R18.B	ERRMACH	Numéro d'erreur machine Cette variable permet la lecture en décimal du numéro d'erreur machine détectée par le système (Erreurs 18, 30 à 33, 35, 36, 39 à 71, 210 à 241, 245, 300 à 331). *REMARQUE Se reporter au manuel opérateur pour la liste des erreurs machine.*
%R19.B	ID_KB_CN	Identificateur pupitre actif ou CN active. En configuration multi pupitre, donne le numéro du pupitre actif (de 0 à 7). En configuration multi CN, donne le numéro de la CN active (de 0 à 4).
%R1A.W	PROGCOUR	Numéro du programme courant Permet la lecture du numéro de programme courant. La valeur 0XFFFF (-1) indique l'absence de programme courant. La valeur 0xFFFE (-2) indique la sélection du mode passant.

3.8.1.11 Vitesse de broche : %R1C.W à %R22.W

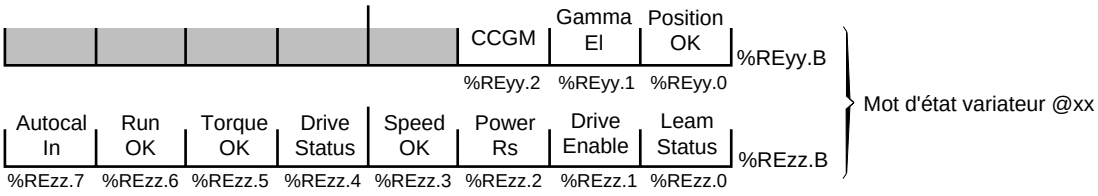
Variable	Mnémonique	Description
%R1C.W	VITBR1	Vitesse Broche 1 Permet la lecture de la valeur codée en hexadécimal de la référence du variateur de broche 1 dans la gamme de vitesse programmée. L'absence des fonctions M3 et M4 dans le programme pièce force la valeur de la variable à 0.
%R1E.W	VITBR2	Vitesse Broche 2 Permet la lecture de la valeur codée en hexadécimal de la référence du variateur de broche 2 dans la gamme de vitesse programmée. L'absence des fonctions M3 et M4 dans le programme pièce force la valeur de la variable à 0.
%R20.W	VITBR3	Vitesse Broche 3 Permet la lecture de la valeur codée en hexadécimal de la référence du variateur de broche 3 dans la gamme de vitesse programmée. L'absence des fonctions M3 et M4 dans le programme pièce force la valeur de la variable à 0.
%R22.W	VITBR4	Vitesse Broche 4 Permet la lecture de la valeur codée en hexadécimal de la référence du variateur de broche 4 dans la gamme de vitesse programmée. L'absence des fonctions M3 et M4 dans le programme pièce force la valeur de la variable à 0.

3.8.1.12 Axe blocable : %R24.L

Variable	Mnémonique	Description
%R24.7 à %R24.0	AXBLK31 à AXBLK24	Axe N° 31 à axe N° 24 La mise à 1 déclare l'axe blocable La mise à 0 déclare l'axe non blocable La RAZ remet les axes conformes au paramètre machine P8
%R25.7 à %R25.0	AXBLK23 à AXBLK16	Axe N° 23 à axe N° 16 La mise à 1 déclare l'axe blocable La mise à 0 déclare l'axe non blocable La RAZ remet les axes conformes au paramètre machine P8
%R26.7 à %R26.0	AXBLK15 à AXBLK8	Axe N° 15 à axe N° 8 La mise à 1 déclare l'axe blocable La mise à 0 déclare l'axe non blocable La RAZ remet les axes conformes au paramètre machine P8
%R27.7 à %R27.0	AXBLK7 à AXBLK0	Axe N° 7 à axe N° 0 La mise à 1 déclare l'axe blocable La mise à 0 déclare l'axe non blocable La RAZ remet les axes conformes au paramètre machine P8

3.8.1.13 Mot d'état variateur "1050"

Pour le variateur numérique d'adresse xx (xx compris entre 00 et 31), le mot d'état se présente sous la forme :



Bit	Signification	Valeurs
%REzz.0	Learn Status	Réservé
%REzz.1	Drive Enable	0 : variateur non validé 1 : variateur validé
%REzz.2	Power Rs	0 : tension bus non présente 1 : tension bus présente
%REzz.3	Speed OK	0 : vitesse non atteinte 1 : vitesse atteinte
%REzz.4	Drive Status	0 : arrêt variateur 1 : départ variateur
%REzz.5	Torque OK	0 : seuil de couple non atteint 1 : seuil de couple atteint
%REzz.6	Run OK	0 : moteur à l'arrêt 1 : moteur en mouvement
%REzz.7	Autocalibration In	0 : autocalibration achevée 1 : autocalibration en cours
%REyy.0	Position OK	0 : position non atteinte 1 : position atteinte
%REyy.1	Gamma El	0 : gamme de vitesse basse 1 : gamme de vitesse haute
%REyy.2	CCGM	0 : gamme de vitesse mécanique non demandée 1 : gamme de vitesse mécanique demandée

3.8.2 Sortie vers la CN %W0 à %W7F

3.8.2.1 Commandes Impulsionnelles : %W2.W

Variable	Mnémonique	Description
%W2.3	CHG_OPDC	Si CHG_OPDC est égal à 1, il y aura rechargement des opérateurs dynamiques en C sur une RAZ CN générale.
%W2.2	C_INDG	Commutations groupes communs/groupes indépendants C'est une information maintenue. La détection d'un changement d'état de C_INDG se fait uniquement sur une RAZ commune demandée par l'automate. C_INDG = 0 : groupes communs C_INDG = 1 : groupes indépendants.
%W2.1	C_NMAUTO	Fonctionnalité N/M AUTO La mise à 1 valide la fonctionnalité N/M (2/3, 3/5, ... etc ..) AUTO Cette commande est effective lorsque la commande C_CYCLE retombe.
%W2.0	KB_INIT	Initialisation clavier La mise à 1 autorise la reconnaissance de la configuration des claviers et des CN interconnectés.La reconnaissance doit être effectuée à chaque modification de la configuration. Après la reconnaissance, le clavier N°1 est affecté à la CN N°1.
%W3.7	C_M01	Validation de l'arrêt programmé optionnel (M01) Une impulsion valide ou invalide l'arrêt programmé optionnel suivant l'état précédent.
%W3.6	C_SLASH	Validation du saut de bloc Une impulsion valide ou invalide le saut de bloc suivant l'état précédent.
%W3.5	C_RAZER	Annulation de l'erreur de poursuite sans RAZ
%W3.4	C_DGURG	Demande dégagement d'urgence Cette demande est prise en compte dans les modes «CONT, SEQ, RAP». Le bloc en cours est interrompu et le système se branche sur le dernier programme de dégagement d'urgence déclaré dans le programme pièce par la fonction G75. Si aucun programme de dégagement d'urgence n'est défini, cette information est traitée de la même façon que C_ARUS.
%W3.3	C_RAX	Sélection du rappel d'axe. Cette demande est prise en compte lorsque E_ARUS = 1 et que tous les manipulateurs d'axes sont relâchés. C'est une commande de type bistable. Une première impulsion positionne E_INTERV à 1 et valide les manipulateurs d'axes dans les deux sens. Si au moins un axe a été déplacé dans le mode INTERV, une seconde impulsion positionne E_RAX à 1 et autorise un seul sens de déplacement des manipulateurs d'axes pour ramener le mobile dans la position initiale.
%W3.2	C_CYCLE	Demande d'exécution d'un «DEPART CYCLE». Permet l'exécution des modes «CONT, SEQ, IMD, RAP, RNS, TEST, CHARG, DECHARG». La commande de C_CYCLE doit être impulsionnelle pour éviter dans les modes «CONT et RAP» une reprise de l'usinage après la détection d'un M02 ou d'une RAZ.
%W3.1	C_ARUS	Demande arrêt d'usinage Cette demande est prise en compte dans les modes «CONT, SEQ, IMD, RAP et JOG incrémental». Une première impulsion provoque l'arrêt d'usinage. Relance de l'usinage par «CYCLE». Cette commande n'a pas d'action sur les groupes automate.

Variable	Mnémonique	Description
%W3.0	C_RAZ	Demande de remise à zéro. Provoque également une RAZ des axes automate en cas d'erreur machine. Prise en compte s'il n'y a pas de déplacement sur les axes.

REMARQUES Pour le traitement de C_ARUS, C_CYCLE et C_RAX, se reporter au manuel opérateur.
Pour le traitement de C_DGURG, se reporter au manuel de programmation.

3.8.2.2 Commandes Maintenues : %W4.W

Variable	Mnémonique	Description
%W4.7	VREDUIT	Demande de passage à vitesse réduite La mise à 1 force les vitesses de déplacements réduites paramétrées dans les mots N3 et N4 de P31 (Voir manuel des paramètres).
%W4.6	INIBUTIL	Vérouillage des utilitaires La mise à 1 interdit l'accès aux utilitaires. La mise à 0 autorise l'accès aux utilitaires.
%W4.5	C_UNIT	Unité des cotes affichées (métrique ou inch). La mise à 1 autorise l'introduction des cotes et la visualisation en pouce. La mise à 0 autorise l'introduction des cotes et la visualisation dans le système métrique.
%W4.4	PRESPUIS	Présence puissance sur moteur La mise à 0 indique à la CN une coupure de la puissance sur les moteurs des axes synchronisés (après un défaut de synchronisation). La mise à 1 indique à la CN que la puissance a été rétablie et autorise la synchronisation des axes.
%W4.3	NARFIB	Non arrêt en fin bloc Autorise le lancement d'un «CYCLE» dans les modes «CONT, SEQ, IMD, RAP» et autorise l'enchaînement des blocs dans les modes «CONT et RAP». La mise à 0 de cette variable provoque la retombée du cycle en fin d'exécution du bloc en cours.
%W4.2 %W4.1	VITMAN2 VITMAN1	Sélection de l'avance rapide en manuel 1 et 2 Autorisent la sélection des vitesses d'avance rapide dans les modes MANU et POM ou multiplient les incréments des manivelles. Les vitesses sont modulables par le potentiomètre des avances

VITMAN1	VITMAN2	VITESSE (Paramètre P31)	INCREMENT MANIVELLE
0	0	JOG normal	Ui x 1
0	1	JOG lent	Ui x 100
1	0	JOG rapide	Ui x 10
1	1	JOG rapide	Ui x 10

Avec Ui «Unité interne du système» fixée par paramètre machine.

%W4.0	AUTAV	Autorisation des avances général sur tous les groupes d'axes Autorise les mouvements dans tous les modes avec déplacement. Le champ «SECU» de la fenêtre status CN signale l'état 0 de cette information.
%W5.7	SC_SAVE	Mise en veille de l'écran CN La mise à 1 autorise la mise en veille de l'écran après 5 min d'inutilisation du clavier. La mise à 0 invalide la mise en veille de l'écran et réactive imédiatement l'écran.

Variable	Mnémonique	Description
%W5.6	SK_DISPL	Affichage de la fenêtre cartouche La mise à 1 invalide l'affichage de la fenêtre cartouche. La mise à 0 valide l'affichage de la fenêtre cartouche. *REMARQUE L'invalidation de l'affichage n'invalide pas l'utilisation des touches logicielles.*
%W5.5	INIBCLAV	Inhibition du clavier La mise à 1 inhibe le clavier alphanumérique et les touches de fonction pour le cartouche de base qui ne sont plus traités par la CN. Les codes des touches sont toutefois transmis à la fonction automatisme par CARCLAV.
%W5.4	IMPULS	Entrées impulsionnelles au pupitre Invalidation des touches «RAZ», «ARUS», «CYCLE», «M01», «/» et de la touche logiciel «JAUGE» La mise à 1 invalide les touches sur le pupitre CN et permet la sélection par la fonction automatisme.
%W5.3	CORDYN	Autorisation de chargement des correcteurs dynamique La mise à 1 autorise le chargement des corrections dynamiques par la fonction automatisme et invalide le chargement par le pupitre.
%W5.2	JOGPUP	Sélection des JOG au pupitre La mise à 1 invalide la sélection du type de JOG par le pupitre CN et autorise la sélection par la fonction automatisme.
%W5.1	MODPUP	Sélection des modes au pupitre La mise à 1 invalide la sélection des modes par le pupitre CN et autorise la sélection des modes par la fonction automatisme. Le mode est sélectionné par l'automate, le n° du mode étant codé dans %Wg03.b, les codes des modes étant identiques à ceux de %W14.b pour les modes communs. %W5.1=0. Hors IHM les modes peuvent être choisis au pupitre, le mode étant affecté au groupe sélectionné par %W17.b.
%W5.0	PUPABS	Pupitre CN absent La mise à 1 déclare le pupitre CN absent. Toutes les fonctions du pupitre CN sont invalidées et peuvent être simulées par la fonction automatisme.

3.8.2.3 Commandes JOG Positif : %W6.L

Variable	Mnémonique	Description
%W6.7 à %W6.0	JOGPOS31 à JOGPOS24	Jog positif sur l'axe N° 31 à Jog positif sur l'axe N° 24
%W7.7 à %W7.0	JOGPOS23 à JOGPOS16	Jog positif sur l'axe N° 23 à Jog positif sur l'axe N° 16
%W8.7 à %W8.0	JOGPOS15 à JOGPOS8	Jog positif sur l'axe N° 15 à Jog positif sur l'axe N° 8
%W9.7 à %W9.0	JOGPOS7 à JOGPOS0	Jog positif sur l'axe N° 7 à Jog positif sur l'axe N° 0

3.8.2.4 Commandes JOG Négatif : %WA.L

Variable	Mnémonique	Description
%WA.7 à %WA.0	JOGNEG31 à JOGNEG24	Jog négatif sur l'axe N° 31 à Jog négatif sur l'axe N° 24
%WB.7 à %WB.0	JOGNEG23 à JOGNEG16	Jog négatif sur l'axe N° 23 à Jog négatif sur l'axe N° 16
%WC.7 à %WC.0	JOGNEG15 à JOGNEG8	Jog négatif sur l'axe N° 15 à Jog négatif sur l'axe N° 8
%WD.7 à %WD.0	JOGNEG7 à JOGNEG0	Jog négatif sur l'axe N° 7 à Jog négatif sur l'axe N° 0

3.8.2.5 Paramètres Externes E20000 à E20031 : %WE.L

Les paramètres externes E200xx sont écrits par le programme utilisateur. La gestion de ces paramètres est assurée par le programme pièce qui peut les lire.

Ils permettent d'échanger des informations booléennes entre le programme utilisateur et le programme pièce.

Variable	Mnémonique	Variable	Mnémonique
%W11.0	E20000	%WF.0	E20016
%W11.1	E20001	%WF.1	E20017
%W11.2	E20002	%WF.2	E20018
%W11.3	E20003	%WF.3	E20019
%W11.4	E20004	%WF.4	E20020
%W11.5	E20005	%WF.5	E20021
%W11.6	E20006	%WF.6	E20022
%W11.7	E20007	%WF.7	E20023
%W10.0	E20008	%WE.0	E20024
%W10.1	E20009	%WE.1	E20025
%W10.2	E20010	%WE.2	E20026
%W10.3	E20011	%WE.3	E20027
%W10.4	E20012	%WE.4	E20028
%W10.5	E20013	%WE.5	E20029
%W10.6	E20014	%WE.6	E20030
%W10.7	E20015	%WE.7	E20031

3.8.2.6 Valeur de l'incrément de JOG : %W13.B

Variable	Mnémonique	Description
%W13.B	C_INCJOG	Commande de l'incrément du JOG La valeur de la variable correspond à l'incrément de JOG demandé : 0x0A Déplacement manuel au pas de 10^{-6} pouce 0x09 Déplacement manuel au pas de 10^{-2} µm ou 10^{-5} pouce 0x00 Déplacement manuel au pas de 10^{-1} µm ou 10^{-4} pouce 0x01 Déplacement manuel au pas de 1 µm ou 10^{-3} pouce 0x02 Déplacement manuel au pas de 10 µm ou 10^{-2} pouce 0x03 Déplacement manuel au pas de 100 µm ou 10^{-1} pouce 0x04 Déplacement manuel au pas de 1000 µm ou 1 pouce 0x05 Déplacement manuel au pas de 10000 µm ou 1 pouce 0x06 Déplacement manuel en continu 0x08 Déplacement manuel par manivelle Le choix de la valeur de l'incrément en µm ou en pouce dépend de la valeur de la variable C_UNIT.

3.8.2.7 Mode demandé : %W14.B

Variable	Mnémonique	Description
%W14.B	MODEDEM	Mode demandé La valeur de la variable correspond au mode CN demandé : 0x00 Mode Continu «CONT» 0x01 Mode Séquentiel «SEQ» 0x02 Mode Immédiat «IMD» 0x03 Mode Rapide «RAP» 0x04 Mode Recherche de Numéro de Séquence «RNS» 0x05 Mode Modification «MODIF» 0x06 Mode Test «TEST» 0x07 Mode Manuel «MANU» 0x08 Mode Prise d'Origine Mesure «POM» 0x09 Mode Prise de Référence «PREF» 0x0A Mode Réglages Outils «REGOUT» 0x0B Absence de mode 0x0D Mode Chargement «CHARG» 0x0F Mode Déchargement «DECHG»

3.8.2.8 Affichage de message : %W15.B et W16.B

Variable	Mnémonique	Description
%W15.B	MSG1	Numéro du message à afficher ligne 1. Le message est affiché sur la ligne 1 de la page «Messages de diagnostic». Le message correspondant au numéro doit figurer dans le programme pièce %9999.9.
%W16.B	MSG2	Numéro du message à afficher ligne 2. Le message est affiché sur la ligne 2 de la page «Messages de diagnostic». Le message correspondant au numéro doit figurer dans le programme pièce %9999.9.

Le programme %9999.9 doit être structuré de la façon suivante :

```
%9999.9
N0
N1 $ MESSAGE NUMERO 1
$    SUITE MESSAGE NUMERO 1
N2 $ MESSAGE NUMERO 2
$    SUITE MESSAGE NUMERO 2
$    SUITE MESSAGE NUMERO 2

Nx $ MESSAGE NUMERO X
```

Où :
- les numéro de bloc (N..) correspondent aux numéros des messages à afficher,
- le caractère $ doit précéder les messages,
- une ligne de message comporte au maximum 35 caractères,
- les blocs non numérotés sont affichés comme suite aux messages.

3.8.2.9 Sélection du groupe d'axes : %W17.B

Variable	Mnémonique	Description
%W17.B	SELECGR	Sélection du groupe d'axes Permet d'affecter à la visualisation, toutes les informations qui se rapportent à un groupe d'axes (programme pièce, variable programme, .. etc ...). Les données introduites au clavier CN en «IMD» sont affectées au groupe d'axes sélectionné. 0 Sélection du groupe d'axes 1 1 Sélection du groupe d'axes 2 2 Sélection du groupe d'axes 3 3 Sélection du groupe d'axes 4 4 Sélection du groupe d'axes 5 5 Sélection du groupe d'axes 6 6 Sélection du groupe d'axes 7 7 Sélection du groupe d'axes 8 *REMARQUE Utilisé uniquement pour les machine outils multi-groupes d'axes.*

3.8.2.10 Numéro de programme demandé : %W18.W

Variable	Mnémonique	Description
%W18.W	PROGDEM	Numéro du programme demandé Permet de charger le numéro de programme demandé comme programme courant ou de demander un usinage en mode passant lecteur. Le numéro de programme ou la demande d'usinage en mode passant est pris en compte par le système sur le front montant de l'information C_RAZ = 1 0 Pas de demande de numéro de programme par la fonction automatisme de 1 à 0x270F (9999) Numéro de programme spécifié par la fonction automatisme -2 (0xFFFE) Usinage en mode passant lecteur demandé par la fonction automatisme

REMARQUE *Le programme demandé doit être présent en mémoire CN pour être chargé comme programme courant. Si il n'est pas présent, le système invalide l'ancien programme courant et le message «PAS DE PROGRAMME COURANT» apparaît sur la page «PROG.».*

3.8.2.11 Affectation manivelle : %W1A.B à %W1D.B

Variable	Mnémonique	Description
%W1A.B	AFMAN1	Affectation manivelle N°1 Reçoit l'adresse physique d'un axe à déplacer. Se reporter au manuel d'installation et de mise en oeuvre pour l'affectation des adresses physiques des axes.
%W1B.B	AFMAN2	Affectation manivelle N°2 Identique à AFMAN1 pour la manivelle N°2.
%W1C.B	AFMAN3	Affectation manivelle N°3 Identique à AFMAN1 pour la manivelle N°3.
%W1D.B	AFMAN4	Affectation manivelle N°4 Identique à AFMAN1 pour la manivelle N°4.

⚠ ATTENTION

Les variables AFMAN1, AFMAN2, AFMAN3 et AFMAN4 doivent contenir l'adresse physique d'un axe mesuré.

L'affectation de la manivelle à un axe doit précéder le déplacement manuel par manivelle.

Les commandes de JOG, JOGPOSn et JOGNEGn (avec n de 0 à 31), doivent être validées pour l'axe concerné.

3.8.2.12 Potentiomètre de broche : %W1E.B à %W21.B

Variable	Mnémonique	Description
%W1E.B	POTBR1	Potentiomètre broche N°1 Valeur codée en hexadécimal correspondant à la valeur d'entrée du CAN. Valeur codée en hexadécimal : 0x0 ; 0xFF Valeur d'entrée du CAN (Fonction anai(..)) : 0 Volt ; 10 Volt Pourcentage de vitesse de broche : 50% ; 100%
%W1F.B	POTBR2	Potentiomètre broche N°2 Identique à POTBR1 pour la broche N°2.
%W20.B	POTBR3	Potentiomètre broche N°3 Identique à POTBR1 pour la broche N°3.
%W21.B	POTBR4	Potentiomètre broche N°4 Identique à POTBR1 pour la broche N°4.

3.8.2.13 Commandes Broches : %W22.W

Variable	Mnémonique	Description
%W22.7	VERBR4	Présence puissance sur la broche 4 VERBR4 = 0 : signale à la CN que la puissance de la broche 4 est mise, VERBR4 = 1 : signale à la CN le verrouillage manuel ou le blocage dela broche 4.
%W22.6	VERBR3	Présence puissance sur la broche 3 Identique à VERBR4 pour la broche 3
%W22.5	VERBR2	Présence puissance sur la broche 2 Identique à VERBR4 pour la broche 2
%W22.4	VERBR1	Présence puissance sur la broche 1 Identique à VERBR4 pour la broche 1
%W22.3	STOPBR4	Demande d'arrêt de la broche N° 4 par la fonction automate Commande maintenue, tant que le bit est à 1 la broche est arrêtée La mise à 0 de ce bit autorise de nouveau la rotation de la broche.
%W22.2	STOPBR3	Demande d'arrêt de la broche N° 3 par la fonction automate Identique à STOPBR4 pour la broche 3
%W22.1	STOPBR2	Demande d'arrêt de la broche N° 2 par la fonction automate Identique à STOPBR4 pour la broche 2
%W22.0	STOPBR1	Demande d'arrêt de la broche N° 1 par la fonction automate Identique à STOPBR4 pour la broche 1
%W23.3	COMBR4	Commande broche N˚4 La mise à 1 autorise le pilotage de la broche par la fonction automatisme. La consigne est transmise à la carte d'axes par C_VITBR4.
%W23.2	COMBR3	Commande broche N˚3 La mise à 1 autorise le pilotage de la broche par la fonction automatisme. La consigne est transmise à la carte d'axes par C_VITBR3.
%W23.1	COMBR2	Commande broche N˚2 La mise à 1 autorise le pilotage de la broche par la fonction automatisme. La consigne est transmise à la carte d'axes par C_VITBR2.
%W23.0	COMBR1	Commande broche N˚1 La mise à 1 autorise le pilotage de la broche par la fonction automatisme. La consigne est transmise à la carte d'axes par C_VITBR1.

3.8.2.14 Consigne de vitesse de broche : %W24.W à %W2A.W

Variable	Mnémonique	Description
%W24.W	C_VITBR1	Consigne vitesse broche N˚1 Permet d'envoyer la valeur codée de la référence du variateur de broche en binaire sur 14 bits avec signe. Le bit 15 de C_VITBR1 donne le signe de la consigne.
%W26.W	C_VITBR2	Consigne vitesse broche N˚2 Identique à C_VITBR1 pour la broche 2.
%W28.W	C_VITBR3	Consigne vitesse broche N˚3 Identique à C_VITBR1 pour la broche 3.
%W2A.W	C_VITBR4	Consigne vitesse broche N˚4 Identique à C_VITBR1 pour la broche 4.

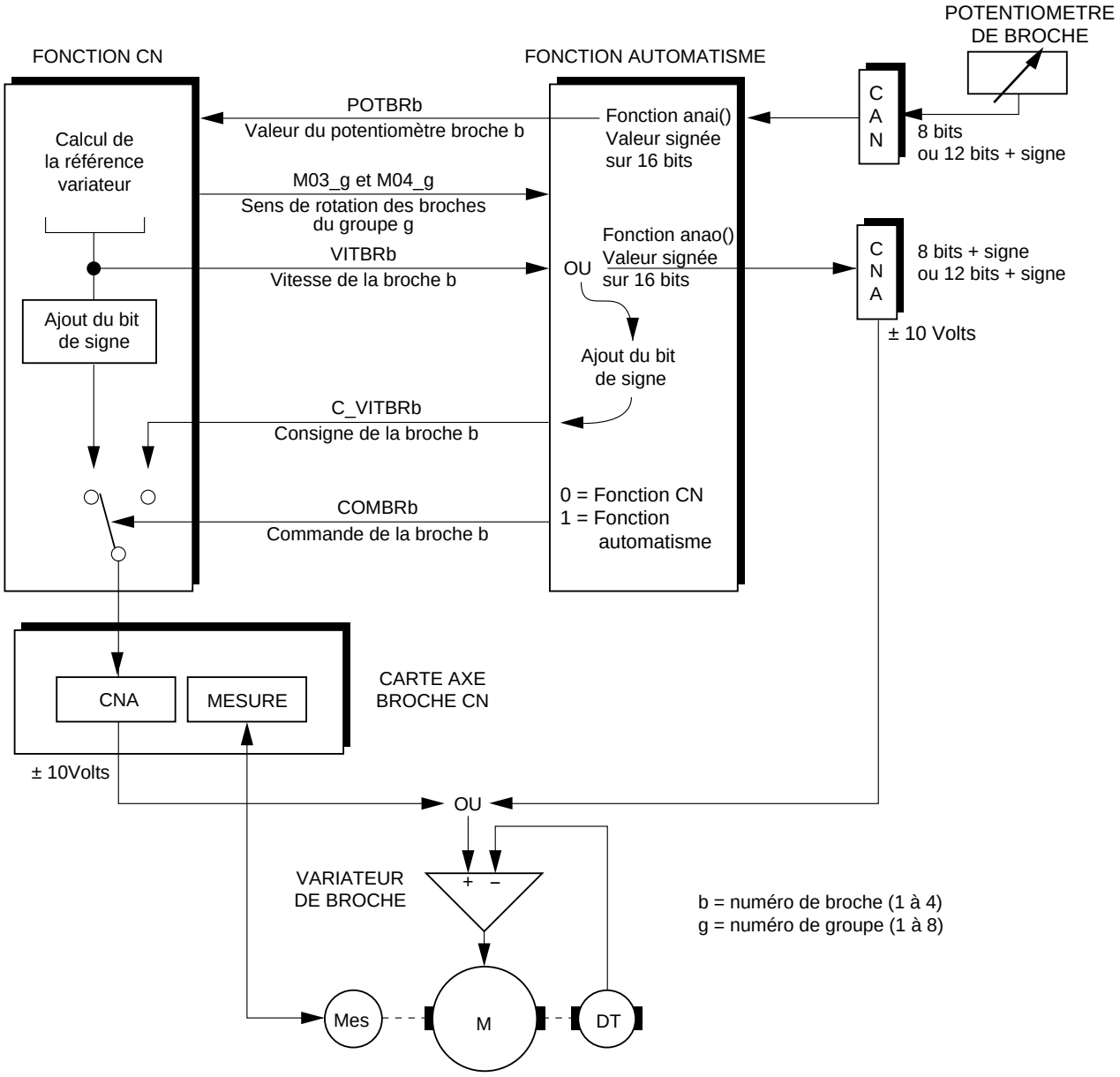


Figure 3.2 - Organisation d’une broche

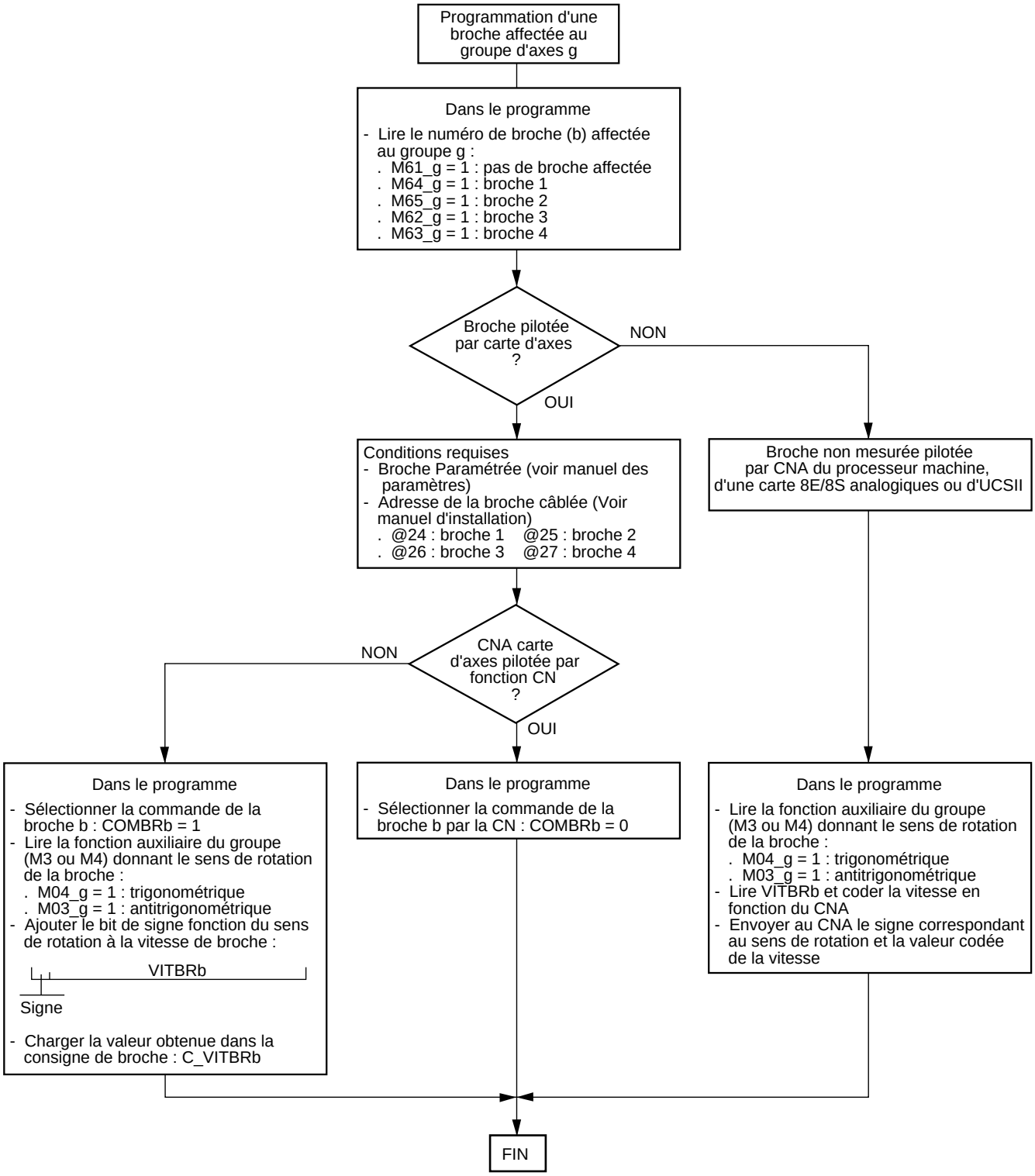


Figure 3.3 - Programmation d’une broche

3.8.2.15 Incréments de JOG interdits : %W2C.W

Variable	Mnémonique	Description
%W2C.1	NJGMANIV	Interdit la sélection manivelle Invalide la touche logiciel «MANIV» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W2C.0	NJG0001	Interdit la sélection de l'incrément à 0,001 mm Invalide la touche logiciel «.001» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W2D.7	NJG001	Interdit la sélection de l'incrément à 0,01 mm Invalide la touche logiciel «.01» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W2D.6	NJG01	Interdit la sélection de l'incrément à 0,1 mm Invalide la touche logiciel «.1» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W2D.5	NJG1	Interdit la sélection de l'incrément à 1 mm Invalide la touche logiciel «1» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W2D.4	NJG10	Interdit la sélection de l'incrément à 10 mm Invalide la touche logiciel «10» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W2D.3	NJG100	Interdit la sélection de l'incrément à 100 mm Invalide la touche logiciel «100» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W2D.2	NJG1000	Interdit la sélection de l'incrément à 1000 mm Invalide la touche logiciel «1000» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W2D.1	NJG10000	Interdit la sélection de l'incrément à 10000 mm Invalide la touche logiciel «10000» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W2D.0	NJGILLIM	Interdit la sélection du JOG illimité Invalide la touche logiciel «ILL» du cartouche JOG. La mise à 1 invalide la touche. La mise à 0 valide la touche.

3.8.2.16 Modes interdits : %W30.L

Variable	Mnémonique	Description
%W30.7	I_POM	Interdit la sélection du mode prise d'origine mesure Invalide la touche logiciel «POM» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W30.6	I_PREF	Interdit la sélection du mode prise de références Invalide la touche logiciel «PREF» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W30.5	I_REGOUT	Interdit la sélection du mode réglage automatique d'outils Invalide la touche logiciel «REGOUT» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W30.2	I_CHARG	Interdit la sélection du mode chargement Invalide la touche logiciel «CHARG» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W30.0	I_DCHG	Interdit la sélection du mode déchargement Invalide la touche logiciel «DCHG» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W31.7	I_CONT	Interdit la sélection du mode continu Invalide la touche logiciel «CONT» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W31.6	I_SEQ	Interdit la sélection du mode séquentiel Invalide la touche logiciel «SEQ» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W31.5	I_IMD	Interdit la sélection du mode introduction manuelle de données Invalide la touche logiciel «IMD» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W31.4	I_RAPID	Interdit la sélection du mode rapide Invalide la touche logiciel «RAP» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W31.3	I_RNS	Interdit la sélection du mode recherche de numéro de séquence Invalide la touche logiciel «RNS» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W31.2	I_MODIF	Interdit la sélection du mode modification Invalide la touche logiciel «MODIF» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.

Variable	Mnémonique	Description
%W31.1	I_TEST	Interdit la sélection du mode test Invalide la touche logiciel «TEST» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.
%W31.0	I_JOG	Interdit la sélection du mode Manuel Invalide la touche logiciel «MANU» du cartouche MODE. La mise à 1 invalide la touche. La mise à 0 valide la touche.

3.8.2.17 Validation du couple pour les axes QVN : %W34.L

Les bits de %W34.L sont initialisés à 0.

Variable	Mnémonique	Description
%W34.7 à %W34.0	DISC_TRQ31 à DISC_TRQ24	Validation du couple sur l'axe QVN N° 31 à Validation du couple sur l'axe QVN N°24 Mise à 1 valide le couple. Mise à 0 invalide le couple.
%W35.7 à %W35.0	DISC_TRQ23 à DISC_TRQ16	Validation du couple sur l'axe QVN N° 23 à Validation du couple sur l'axe QVN N° 16 Mise à 1 valide le couple. Mise à 0 invalide le couple.
%W36.7 à %W36.0	DISC_TRQ15 à DISC_TRQ8	Validation du couple sur l'axe QVN N° 15 à Validation du couple sur l'axe QVN N° 8 Mise à 1 valide le couple. Mise à 0 invalide le couple.
%W37.7 à %W37.0	DISC_TRQ7 à DISC_TRQ0	Validation du couple sur l'axe QVN N° 7 à Validation du couple sur l'axe QVN N° 0 Mise à 1 valide le couple. Mise à 0 invalide le couple.

3.8.2.18 Validation Référence vitesse pour les axes QVN : %W38.0

Variable	Mnémonique	Description
%W38.0	DISC_SDP	Validation référence vitesse des axes QVN La mise à 1 autorise le fonctionnement normal des axes QVN. La mise à 0 provoque une annulation brutale de la référence vitesse des axes QVN et donc un freinage au couple maximum.

Si les références vitesse sont invalidées, elles sont forcées à la valeur nulle.

A la mise sous tension, les références vitesse sont invalidées.

Dans le cas de détection d'une erreur CN provoquant la retombée de E_CNPRET, l'invalidation des référence vitesse est forcée pour les axes QVN. l'annulation de l'erreur sur une RAZ rend de nouveau effective la validation ou l'invalidation des références vitesse par la fonction automatisme.

REMARQUE *Il est recommandé sur un arrêt d'urgence d'invalider DISC_SDP et d'activer un arrêt des avances pour ne pas générer une erreur de poursuite trop grande.*

3.8.2.19 Recul ou retour sur trajectoire

Variable	Mnémonique	Description
%W39.2	RAP_AUTO	Rappel automatique à la suite d'une intervention La mise à 1 active le rappel, La mise à 0 l'annule.
%W39.1	B_RETOUR	Demande de retour sur trajectoire à la position d'interruption La mise à 1 active la demande, La mise à 0 annule la demande de retour.
%W39.0	B_RECUL	Demande de recul sur trajectoire La mise à 1 active la demande, La mise à 0 annule la demande de recul.

3.8.2.20 Arrêt d'avance par axe (le rang du bit donne l'adresse physique de l'axe) : %W3A.L

Variable	Mnémonique	Description
%W3A.7 à %W3A.0	STOPAX31 à STOPAX24	axe N° 31 à axe N° 24 Dans le mode d'usinage ou en mode JOG, la mise à 1 d'un bit qui adresse un des axes en mouvement, provoque l'arrêt en vitesse des axes du groupe auquel il appartient. En mode d'usinage, si cet axe ne se déplace pas dans le bloc en cours d'exécution mais est programmé dans le bloc suivant, alors une demande d'arrêt en fin de bloc est provoquée et l'exécution des mouvements en début du bloc suivant reste suspendue tant qu'un axe programmé dans ce bloc a son arrêt maintenu à 1. En nmauto, l'action sur les manipulateurs ou la manivelle est ignorée tant que le bit correspondant à l'axe directement piloté est à 1.
%W3B.7 à %W3B.0	STOPAX23 à STOPAX16	axe N° 23 à axe N° 16 Idem à ci-dessus
%W3C.7 à %W3C.0	STOPAX15 à STOPAX8	axe N° 15 à axe N° 8 Idem à ci-dessus
%W3D.7 à %W3D.0	STOPAX7 à STOPAX0	axe N° 7 à axe N° 0 Idem à ci-dessus

3.8.2.21 Réduction de courant : %WE00.B à WE1F.B "D.I.S.C." et "1050"

La fonction réduction de courant permet de réduire le courant pour les axes et broches numériques en fonction de la valeur du byte respectif.

Variable	Mnémonique	Description
%WE1F.B à %WE00.B	RDUC_TRQ31 à RDUC_TRQ0	axe N° 31 à axe N° 0

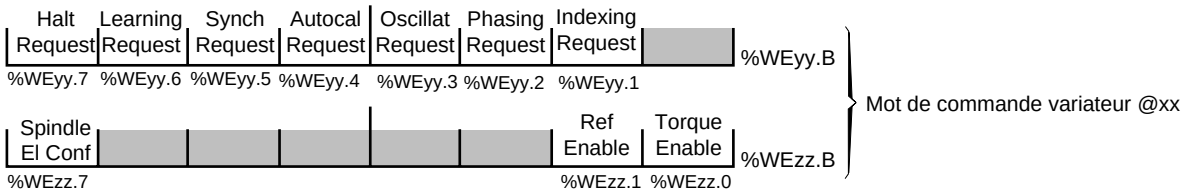
Soit $I_{maximal}$ le courant maximal compte tenu de la limitation statique et de α la valeur de l'octet :
- si α est négatif ou nul ($00, $80 à $FF), pas de réduction de courant,
- si α est positif ($01 à $7F), le courant maximal autorisé est : $I_{maximal} = I_{max\_stat} \times [(127 - \alpha) / 127]$.

La réduction dynamique de courant imposée à un variateur numérique mâitre est transmise aux variateurs numériques esclaves associés.

Dans le cas d'un fonctionnement en configuration anti-jeu, la réduction dynamique imposée à un variateur numérique maître est sans effet sur les courants de précharge du maître et de l'esclave.

3.8.2.22 Mot de commande variateur "1050"

Pour le variateur numérique d'adresse xx (xx compris entre 00 et 31), le mot de commande se présente sous la forme :



Bit	Signification	Valeurs
%WEzz.0	Torque Enable	0 : validation couple non demandée 1 : validation couple demandée
%WEzz.1	Reference Enable	0 : référence non validée 1 : référence validée
%WEzz.7	Spindle Electrical Configuration	0 : gamme basse 1 : gamme haute
%WEyy.1	Indexing Request	0 : indexation non demandée 1 : indexation demandée
%WEyy.2	Phasing Request	0 : calage capteur non demandé 1 : calage capteur demandé
%WEyy.3	Oscillation Request	0 : oscillation non demandée 1 : oscillation demandée
%WEyy.4	Autocalibration Request	0 : autocalibration non demandée 1 : autocalibration demandée
%WEyy.5	Synchronization Request	Réservé
%WEyy.6	Learning Request	Réservé
%WEyy.7	Halt Request	0 : arrêt non demandé 1 : arrêt demandé

3.8.3 Entrées venant des groupes d'axes

Les entrées venant des groupes d'axes sont groupées dans 8 postes de 128 octets ; ce sont les variables %Rg00 à %Rg7F où g vaut de 1 à 8 pour les groupes de 1 à 8.

3.8.3.1 Etat Groupe : %Rg00.W

REMARQUE *L'ensemble de ces variables s'applique aux groupes d'axes CN indépendants. Seules les variables E_RAZ1 à E_RAZ8, E_CYCL1 à E_CYCL8, E_DEGURG1 à E_DEGURG8, NO_POS1 à NO_POS8 et E_DEF1 à E_DEF8 s'appliquent aux groupes d'axes automates (Voir chapitre 17).*

Variable	Mnémonique (Groupe 1 à 8)	Description
%Rg00.7	E_M011	Arrêt programmé optionnel validé sur le groupe d'axes CN indépendants N°g.
	à E_M018	Signale la prise en compte des arrêts programmés optionnels dans un programme pièce.
%Rg00.6	E_SLASH1	Saut de bloc validé sur le groupe d'axes CN indépendants N°g.
	à E_SLASH8	Signale la prise en compte des sauts de blocs dans un programme pièce.
%Rg00.5	E_INTER1	Etat intervention sur le groupe d'axes CN indépendants N°g.
	à E_INTER8	
%Rg00.0	E_PROG1	Programme en cours du groupe d'axes CN indépendants N°g.
	à E_PROG8	Signale qu'un programme pièce est en cours d'exécution dans les modes «CONT, SEQ, RAP, RNS, TEST, IMD».
%Rg01.7	E_OPER1	Signale un arrêt programmé provoquépar un M00 ou un M01validé.
	à E_OPER8	
%Rg01.6	E_DEF1	Défaut sur groupe N˚g
	à E_DEF8	Indique l'ocurence d'une erreur de programmation ou l'absence de programme pièce sur le groupe. La mise à 1 indique que le groupe est en défaut.
%Rg01.5	NO_POS1	Axe en attente de position
	à NO_POS8	Lorsqu'un positionnement précis est demandé par programmation (Fonctions G09, M00, M02 ou M10), en IMD ou en mode JOG à chaque arrêt des mouvements, l'information NO_POSg est transmise pendant que l'axe est en attente de position. La mise à 1 indique que l'axe est en attente de position.
%Rg01.4	E_DGURG1	Dégagement d'urgence en cours sur groupe N˚g
	à E_DGURG8	Signale l'exécution d'un programme de dégagement d'urgence. Mise à 1 après lecture par le CN de C_DGURGg = 1 et si le programme de dégagement d'urgence est validé. Mise à 0 sur détection d'un M00 ou un M02.
%Rg01.3	E_RAX1	Rappel d'axes sur le groupe d'axes CN indépendants N°g.
	à E_RAX8	Signale que le rappel d'axe est validé.
%Rg01.2	E_CYCL1	Cycle en cours sur le groupe N˚g
	à E_CYCL8	Indique que le groupe est en train d'exécuter un bloc de programme pièce. Mise à 0 la CN attend l'information C_CYCLEg = 1 pour exécuter le programme pièce ou le bloc suivant. Mise à 1 indique qu'un bloc est en cours d'exécution.

Variable	Mnémonique (Groupe 1 à 8)	Description
%Rg01.1	E_ARUS1	Sortie d'arrêt usinage du groupe d'axes CN indépendants N°g.
	à E_ARUS8	Signale l'état intervention du système (arrêt du programme en cours d'exécution et validation des manipulateurs d'axes).
%Rg01.0	E_RAZ1	RAZ en cours sur groupe N˚g
	à E_RAZ8	Bit impulsionnnel d'une durée de 100 ms qui signale une RAZ sur le groupe. Pendant la durée de cette impulsion, les données venant de la fonction automatisme ne sont pas prises en compte. Mise à 1 par touche «RAZ» du pupitre, sur demande de RAZ de la fonction automatisme C_RAZg = 1, en fin d'exécution d'un programme pièce (M02) ou à la mise sous tension de la CN. Cette variable est mise à 0 après 100 ms.
%Rg06.B	MODCOUR1	Mode en cours sur le groupe d'axes CN indépendant N°g.
	à MODCOUR8	La valeur de la variable est à l'image du mode CN en cours sur le groupe d'axes CN indépendants N°g.

3.8.3.2 Numéro du cycle d'usinage en cours : %Rg02.B

Variable	Mnémonique (Groupe 1 à 8)	Description
%Rg02.B	NUMCYC1	Numéro du cycle d'usinage en cours sur groupe N˚g
	à NUMCYC8	Permet de lire le numéro de sous programme du cycle d'usinage de %10000 à %10255.(0 pour %10000 à 0xFF pour %10255).

3.8.3.3 Etat Fonction G : %Rg03.B

Variable	Mnémonique (Groupe 1 à 8)	Description
%Rg03.1	FILET1	Filetage sur groupe N˚g
	à FILET8	Signale l'exécution d'un cycle de filetage G31 (Filetage au grain) G33 (Filetage) ou G38 (Filetage enchaîné), G84K (Taraudage rigide). Mise à 1 par l'exécution de la fonction G31, G33 , G38 ou G84. Mise à 0 par révocation de la fonction.
%Rg03.0	RAPID1	Avance rapide (G00) sur groupe N˚g
	à RAPID8	Signale l'exécution de la fonction G0 dans le bloc en cours du programme pièce. Mise à 1 par l'exécution de la fonction G0. Mise à 0 par la révocation de la fonction G0.

3.8.3.4 Fonction M codée sans compte rendu : %Rg04.W

Variable	Mnémonique (Groupe 1 à 8)	Description
%Rg04.W	MSSCR1 à MSSCR8	Fonction M codée sans compte rendu venant du groupe N°g. Cette variable permet la lecture des fonctions auxiliaires M codées sans compte rendu « à la volée» de M200 à M899 (EX.M210 envoie à la fonction automatisme MSSCRg == 210). Ces fonctions sont considérées par le système comme des fonctions «Avant, Modale». La poursuite du programme pièce se fait sans attente d'acquittement. Exploitées dans les programmes pièce, elle sont accessibles en lecture par la fonction automatisme et nécessitent un décodage dans le programme utilisateur. Une seule fonction M codée «modale» est autorisée dans l'écriture d'un bloc en programmation pièce. Il est possible de programmer dans le même bloc en programmation pièce une fonction codée «modale» et une «non modale». Le décodage des fonctions M doit impérativement être effectué dans la tâche séquentielle TS0.

Figure 3.4 - Fonctions auxiliaires M codées «à la volée»

3.8.3.5 Fonction M codée avec compte rendu : %Rg1E.W

Variable	Mnémonique (Groupe 1 à 8)	Description
%Rg1E.W	MCODCR1 à MCODCR8	Fonction M codée avec compte rendu venant du groupe N°g Cette variable permet la lecture des fonctions auxiliaires M codées avec compte rendu jusqu'à M199 (Ex. M92 envoie à la fonction automatisme %MCODCRg == 92). Ces fonctions sont considérées par le système comme des fonction «Après, non modale». C'est la fonction automatisme qui doit gérer leur éventuelle modalité. Exploitées dans les programmes pièce, elle sont accessibles en lecture par la fonction automatisme et nécessitent un décodage dans le programme utilisateur. Une seule fonction M codée «Non modale» est autorisée dans l'écriture d'un bloc en programmation pièce. Il est possible de programmer dans le même bloc en programmation pièce une fonction codée «modale» et une «non modale». Le décodage des fonctions M doit impérativement être effectué dans la tâche séquentielle TS0.

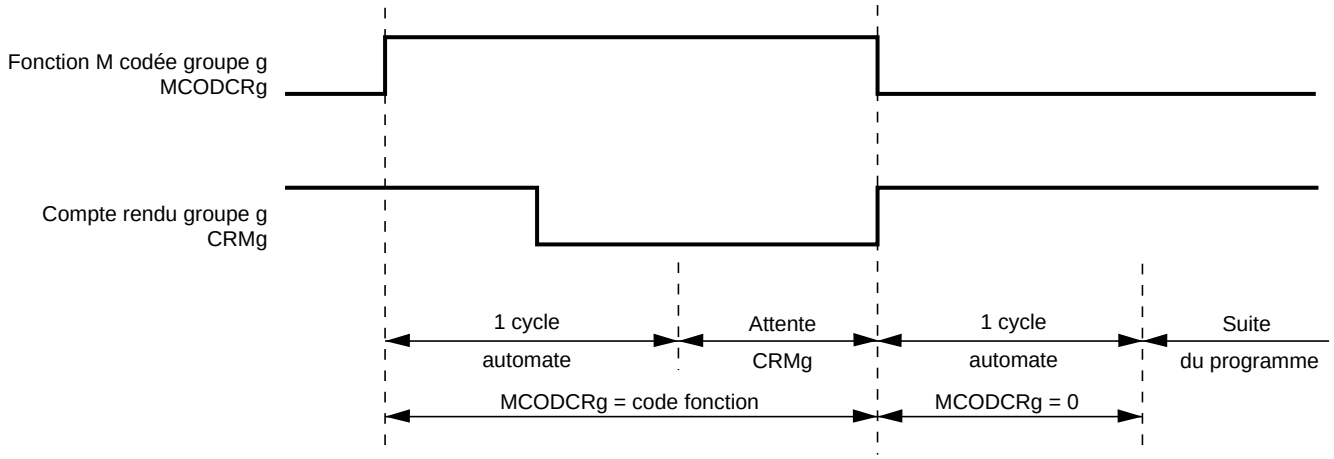


Figure 3.5 - Fonctions auxiliaires M codées avec compte rendu

REMARQUE *Si CRM1 à CRM8 est maintenu à 1, la suite du programme pièce est entreprise après un cycle automate.*

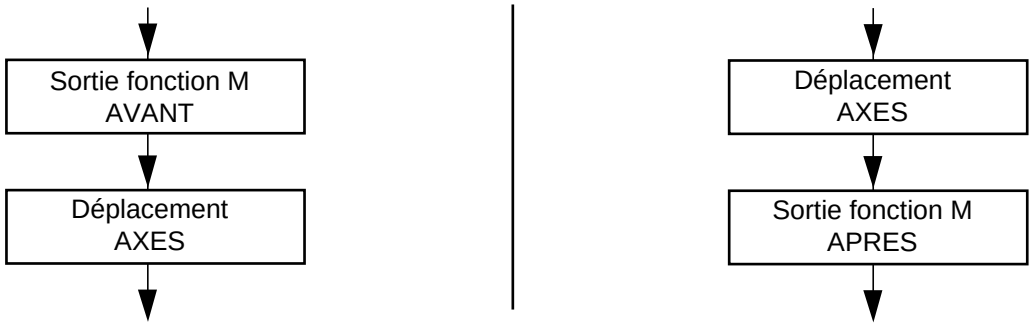
3.8.3.6 Fonctions M décodées : %Rg20.L

Ces fonctions, exploitées dans les programmes pièce sont accessibles en lecture par la fonction automatisme.

Ce sont des fonctions définies et connues par le système (par ex : blocage axes, gamme de broche, ... etc ..).

La fonction automatisme lit la fonction sur un bit (%Rg2n.i) affecté à une fonction M décodée.

Il faut distinguer les fonctions "avant" et "après" :



Les fonctions modales

Une fonction modale reste mémorisée et valide pendant l'exécution de plusieurs blocs de programme pièce jusqu'à la condition de révocation.

Exemple (sur le groupe 1)

N100 M3 M40 S1000	Sortie M3 et M40 vers la fonction automatisme soit %R122.0 =1 et %R121.0 =1.
N110 X100	Déplacement sur X. La fonction automatisme voit toujours %R122.0 =1 et %R121.0 =1.
N120 M5	Sortie M5 vers la fonction automatisme et révocation de M3 soit %R122.2 = 1 et %R122.0 = 0.

Les fonctions non modales

Une fonction non modale n'est valide que pendant l'exécution d'un bloc de programme pièce.

Exemple (sur le groupe 1)

N100 X100 Z200 M6	Sortie M6 vers la fonction automatisme soit %R122.3 = 1.
N110 X50	M6 est acquitté par le CRM1 au bloc précédent soit %R122.3 = 0.

⚠ ATTENTION

Toutes les fonctions auxiliaires décodées sont des fonctions avec compte rendu (CRM1 à CRM8)

L'état de CRM1 à CRM8 conditionne la poursuite ou l'attente de l'exécution du bloc de programme pièce

La fonction automatisme doit gérer CRM1 à CRM8, pour les fonctions programmées et pour les fonctions révoquées, ou initialisées (sur RAZ ou INIT).

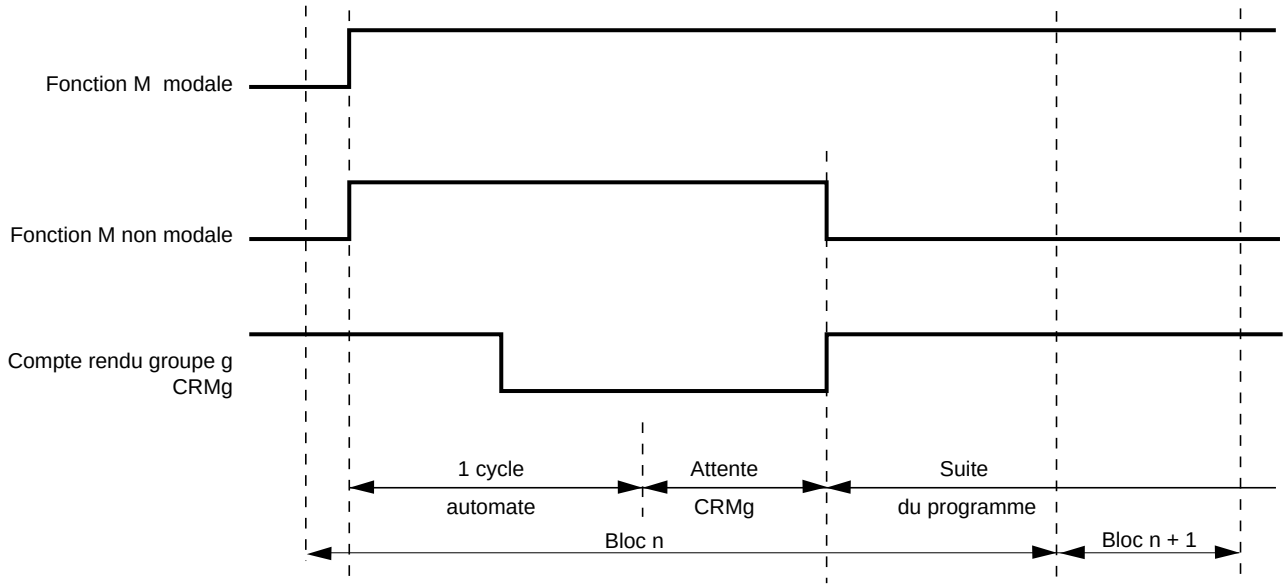


Figure 3.6 - Traitement des fonctions auxiliaires M décodées

REMARQUE *Si CRM1 à CRM8 est maintenu à 1, la suite du programme pièce est entreprise après un cycle automate.*

Variable	Mnémonique (Groupe 1 à 8)		Définition	Révocation par	type de fonction			
					Avant	Après	Modale	Non modale
%Rg20.7	M999_1	M999_8	Masquage par programmation des modes IMD, MODIF, et appel de sous-programmes par la fonction automatisme	M997, M998, M2	X		X	
%Rg20.6	M998_1	M998_8	Démasquage par programmation des modes IMD, MODIF, et appel de sous-programmes par la fonction automatisme	M999, M997	X		X	
%Rg20.5	M997_1	M997_8	Forçage de l'enchaînement des blocs	M998, M999, M2	X		X	
%Rg20.3	M49_1	M49_8	Potentiomètre d'avance et de broche forcés à 100%	M48, M2	X		X	
%Rg20.2	M48_1	M48_8*	Validation des potentiomètres de broche et d'avance	M49		X	X	
%Rg20.1	M11_1	M11_8	Déblocage d'axes	M10	X		X	
%Rg20.0	M10_1	M10_8	Blocage d'axes	M11		X	X	
%Rg21.7	M12_1	M12_8	Arrêt d'usinage programmé	C_CYCLE = 1		X	X	
%Rg21.5 %Rg21.4 %Rg21.3 %Rg21.2 %Rg21.1 %Rg21.0	M45_1 M44_1 M43_1 M42_1 M41_1 M40_1	M45_8 M44_8 M43_8 M42_8 M41_8 M40_8	Gammes de broches	Ces fonctions se révoquent entre elles, M2	X X X X X X		X X X X X X	
%Rg22.7	M19_1	M19_8	Arrêt broche indexée	M0,M2,M3,M4, ARUS		X	X	
%Rg22.6	M09_1	M09_8*	Arrêt des arrosages	M7, M8		X	X	
%Rg22.5	M08_1	M08_8	Arrosage N°1	M9, M2	X		X	
%Rg22.4	M07_1	M07_8	Arrosage N°2	M9, M2	X		X	
%Rg22.3	M06_1	M06_8	Changement d'outil	CRM1 à CRM8		X		X
%Rg22.2	M05_1	M05_8*	Arrêt broche	M3, M4		X	X	
%Rg22.1	M04_1	M04_8	Rotation broche sens trigonométri que	M3, M5, M19, M0, M2	X		X	
%Rg22.0	M03_1	M03_8	Rotation broche sens anti-trigo nométrique	M4, M5, M19, M0, M2	X		X	
%Rg23.7	M61_1	M61_8	Invalidation de la broche courante dans un groupe	M64, M65, M62, M63		X	X	
%Rg23.2	M02_1	M02_8	Fin de programme pièce	RAZ		X		X
%Rg23.1	M01_1	M01_8	Arrêt programmé optionnel	C_CYCLE = 1		X		
%Rg23.0	M00_1	M00_8	Arrêt programmé	C_CYCLE = 1		X		

(*) Fonction initialisée à la mise sous tension, par une RAZ ou par la fonction M02.

3.8.3.7 Fonctions M décodées (Etat des broches) : %Rg24.W

Variable	Mnémonique (Groupe 1 à 8)		Définition	Révocation par	type de fonction			
					Avant	Après	Modale	Non modale
%Rg24.3	M63_1	M63_8	Référence broche aiguillée sur broche 4.	M61, M62, M64, M65	X		X	
%Rg24.2	M62_1	M62_8	Référence broche aiguillée sur broche 3.	M61, M63, M64, M65	X		X	
%Rg24.1	M65_1	M65_8	Référence broche aiguillée sur broche 2.	M61, M62, M63, M64	X		X	
%Rg24.0	M64_1	M64_8	Référence broche aiguillée sur broche 1.	M61, M62, M63, M65	X		X	

Variable	Mnémonique (Groupe 1 à 8)		Définition	Révocation par	type de fonction			
					Avant	Après	Modale	Non modale
%Rg25.3	M69_1	M69_8	Mesure broche 4 exploitable	M66, M67, M68, M02	X		X	
%Rg25.2	M68_1	M68_8	Mesure broche 3 exploitable	M66, M67, M69, M02	X		X	
%Rg25.1	M67_1	M67_8	Mesure broche 2 exploitable	M66, M68, M69, M02	X		X	
%Rg25.0	M66_1	M66_8	Mesure broche 1 exploitable	M67, M68, M69, M02	X		X	

3.8.3.8 Blocage - déblocage d'axes

Les axes sont blocables par la fonction auxiliaire M10 et déblocable par la fonction auxiliaire M11. La liste des axes blocables est définie par le paramètre machine P8 (Voir manuel des paramètres).

Si la fonction M10 est présente (blocage d'axes s'il n'y à pas de mouvement) le système détecte le changement d'état des variables AXMVTaxe (avec axe de 0 à 31) sur les axes blocables.

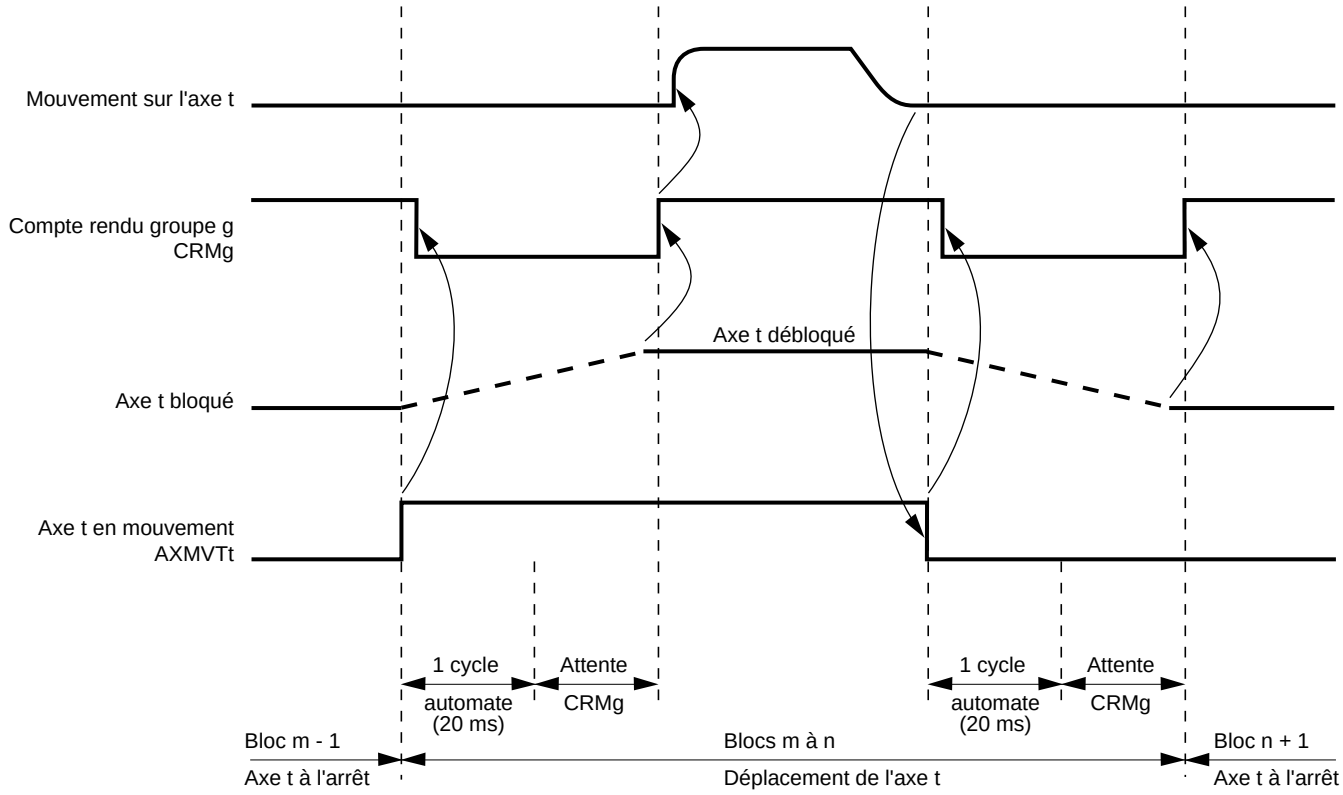


Figure 3.7 - Principe du blocage/déblocage d'axe

REMARQUE *Si au bloc n + 1 l'axe t est toujours en mouvement, la variable AXMVTt (avec axe de 0 à 31) reste monté et il y a enchaînement des blocs.*

3.8.3.9 Numéro d'outil : %Rg7C.L

Variable	Mnémonique (Groupe 1 à 8)	Description
%Rg7C.L	OUTIL1	Numéro d'outil demandé par le groupe N°g.
	à OUTIL8	Permet la lecture des numéro d'outils (valeur décimale de 0 à 65535). Les fonctions T sont considérées par le système comme des fonctions «Avant Modale» sans attente de compte rendu.

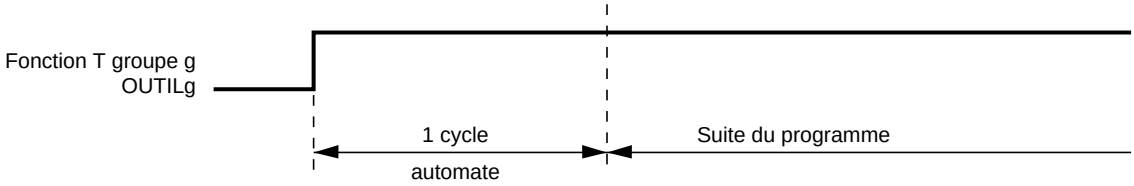


Figure 3.8 - Traitement des fonctions T

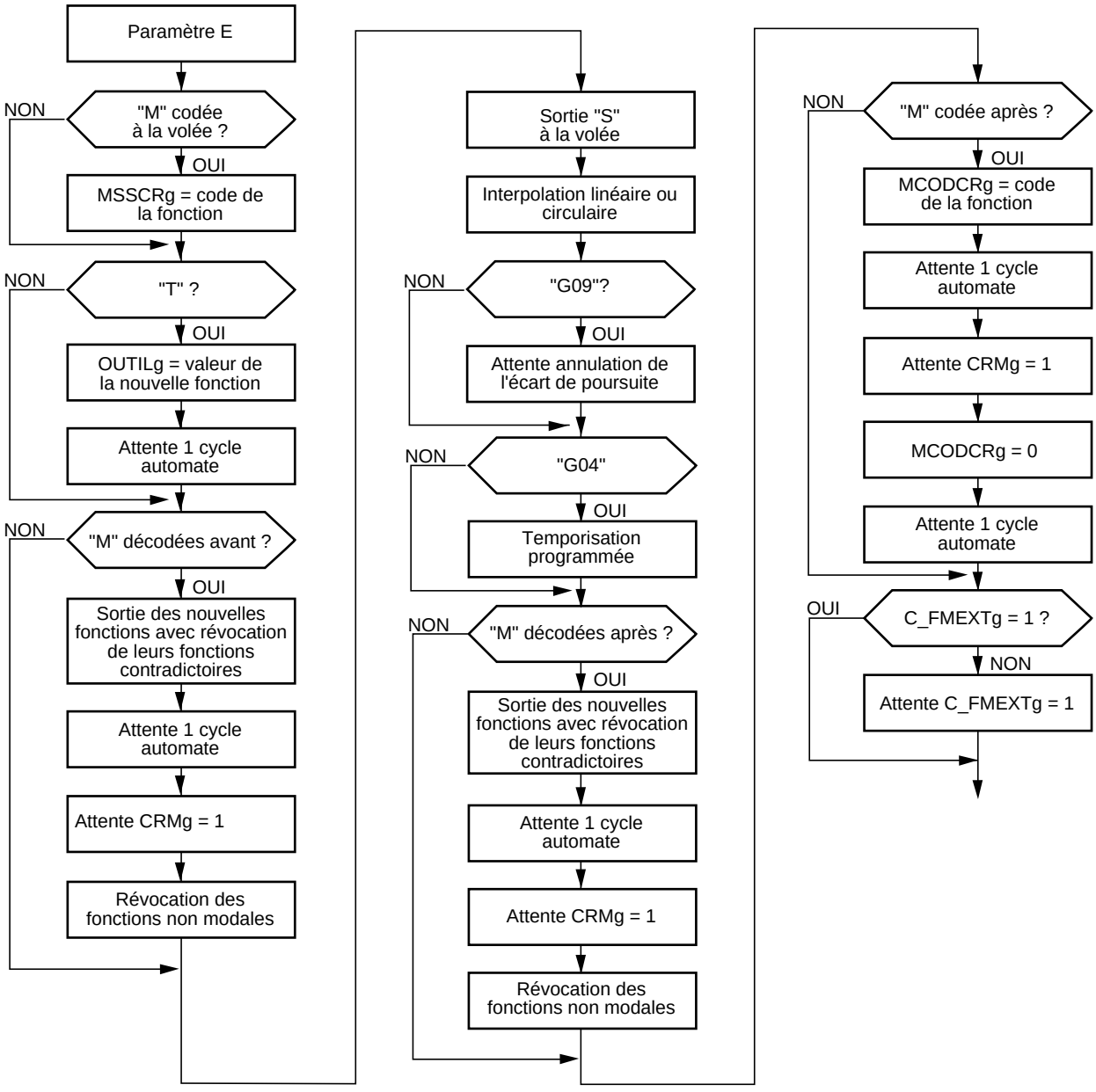


Figure 3.9 - Principe d'exécution des fonctions programmées dans un bloc de programme pièce

3.8.4 Sortie vers les groupes d'axes

Les sorties vers les groupes d'axes sont groupées dans 8 postes de 128 octets:

Concerne les variables %Wg00 à %Wg7F où g vaut de 1 à 8 pour les groupes de 1 à 8.

3.8.4.1 Commandes Groupe : %Wg00.W

REMARQUE *Les variables C_ MODE1 à C _MODE8 sont valides uniquement pour les groupes axes automates (Voir chapitre 17).*

Les variables C_ARUS1 à C_ARUS8, C_RAX1 à C_RAX8, C_SLASH1 à C_SLASH8 et C_M011 àC_M018 sont valides uniquement pour les groupes d'axes CN.

Variable	Mnémonique (Groupe 1 à 8)	Description
%Wg00.7	C_MODE1	Commande mode «CONT» et «SEQU» sur groupe d'axes automates N°g
	à C_MODE8	Mise à 0, le mode continu est validé à partir du bloc suivant. Mise à 1, le mode séquentiel est validé pour le bloc en cours d'exécution. Cette information n'a de sens que si le groupe est valide.
%Wg00.6	C_FAST1	Commande maintenue de vitesse rapide en cours de cycle
	à C_FAST8	Cette commande doit être utilisée en cours de cycle (C_CYCLEg = 1). La mise à 1 autorise un déplacement à vitesse la plus rapide possible. La mise à 0 entraîne un déplacement à vitesse de travail.
%Wg00.5	CRM1	Compte rendu des fonctions M du groupe N°g
	à CRM8	A l'état 0, entraîne une attente du système et le non traitement des fonctions suivantes dans le bloc en cours d'exécution. A l'état 1, autorise la poursuite des traitements.
%Wg00.4	APPSS1	Appel d'un sous-programme sur groupe N°g
	à APPSS8	En cours d'exécution d'un programme pièce, la mise à 1 entraîne le branchement à un sous-programme %9999.g (Avec g numéro de groupe). Le maintien de l'appel ou un nouvel appel de sous-programme est ignoré durant l'exécution du sous-programme. En fin d'exécution du sous-programme, aucun compte-rendu n'est émis par le système. C'est le sous-programme qui doit transmettre à la fonction automatisme un compte-rendu pour l'annulation de l'appel (Fonction M, paramètre externe,...) Si un seul groupe d'axes CN est déclaré, c'est le programme %9999 qui est appelé (soit %9999.0).
%Wg00.3	ARBUT1	Arrêt butée fin de bloc sur groupe N°g
	à ARBUT8	La mise à 1 provoque un arrêt des mouvements sur le groupe d'axes, l'enchaînement au bloc suivant ou un saut à un autre bloc. La fonction G10, associée à ses arguments, doit être présente dans le programme pièce.
%Wg00.2	VALID1	Validation du groupe N°g
	à VALID8	La mise à 1 valide l'utilisation du groupe d'axes La validation ou l'invalidation ne sont effectives que sur une «RAZ» ou un «M02».
%Wg00.1	C_FMEXT1	Commande fin de mouvement extérieur sur gr. N°g
	à C_FMEXT8	La mise à 0 interdit la retombée du «CYCLE» dans les modes «SEQU» et «IMD» ou l'enchaînement sur le bloc suivant dans les modes «CONT» et «RAP». La mise à 1 entraîne la poursuite normale dans l'exécution du mode. Cette variable est testée en fin d'exécution de chaque bloc.

Variable	Mnémonique (Groupe 1 à 8)	Description
%Wg00.0	C_AUTAV1 à C_AUTAV8	Autorisation des avances sur le groupe N°g Cette variable est active si l'autorisation d'avance générale AUTAV = 1..La mise à 0 provoque l'arrêt des mouvements sur le groupe d'axes dans tous les modes avec déplacement. La reprise des mouvements se fait lorsque C_AUTAVg = 1.
%Wg01.7	C_M011 à C_M018	Validation de l'arrêt programmé optionnel (M01) sur le groupe d'axes CN indépendants N°g. Une impulsion valide ou invalide l'arrêt programmé optionnel suivant l'état précédent.
%Wg01.6	C_SLASH1 à C_SLASH8	Validation du saut de bloc sur le groupe d'axes CN indépendants N°g. Une impulsion valide ou invalide le saut de bloc suivant l'état précédent.
%Wg01.4	C_DGURG1 à C_DGURG8	Demande de dégagement d'urgence sur le groupe N°g Cette demande est prise en compte dans les modes «CONT et SEQ». Le bloc en cours est interrompu et le système se branche sur le dernier programme de dégagement d'urgence déclaré dans le programme pièce par la fonction G75. Si aucun programme de dégagement d'urgence n'est défini, cette information est traitée de la même façon que C_ARUS.
%Wg01.3	C_RAX1 à C_RAX8	Sélection du rappel d'axes sur groupe d'axes CN indépendants N°g. Cette demande est prise en compte dans les modes «CONT, SEQ,RAP».
%Wg01.2	C_CYCL1 à C_CYCL8	Demande départ cycle sur groupe d'axes automate N°g ou groupe indépendant Permet l'exécution des modes «CONT» et «SEQ» pour les groupes d'axes automate.La commande de C_CYCLEg doit être impulsionnelle pour éviter dans les modes «CONT» une reprise de l'usinage après la détection d'un M02 ou d'une RAZ. Cette information n'est prise en compte que si le groupe est valide.
%Wg01.1	C_ARUS1 à C_ARUS8	Demande d'arrêt d'usinage du groupe d'axes CN indépendants N°g. Cette demande est prise en compte dans les modes «CONT, SEQ, RAP, RNS, TEST, IMD».
%Wg01.0	C_RAZ1 à C_RAZ8	Demande RAZ sur groupe d'axes automate N°g ou groupe indépendant Prise en compte s'il n'y a pas de déplacement sur les axes. C'est pendant la RAZ sur un groupe qu'est prise en compte l'information VALIDg et qu'est détecté la présence du programme pièce affecté au groupe automate.

3.8.4.2 Valeur du potentiomètre d'avance : %Wg02.B

Variable	Mnémonique (Groupe 1 à 8)	Description
%Wg02.B	POTAV1 à POTAV8	Potentiomètre d'avance sur groupe N°g Valeur codée en hexadécimal correspondant à la valeur d'entrée du CAN

Valeur codée en hexadécimal	Valeur d'entrée du CAN (Fonction anai(.))	Pourcentage vitesse d'avance
0x0	0 Volt	0%
0xFF	10 Volt	120%

3.8.4.3 Mode groupe indépendant : %Wg03.B

Variable	Mnémonique (Groupe 1 à 8)	Description
%Wg03.B	MOD-GR1 à MOD-GR8	Mode demandé sur le groupe indépendant

3.8.5 Défauts et diagnostic système

3.8.5.1 Défaut système ou de configuration

Les variables suivantes renseignent l'utilisateur sur les défauts système ou de configuration.

Variable	Mnémonique	Description
%R97C.W	DEFHTR	Compteur des défauts dépassement temps calcul (ou HTR) (*)
%R97F.2	DEFCARTE	Bit défaut général cartes E/S borniers (**)
%R97F.1	DEFCONF	Bit défaut général configuration cartes E/S borniers (**)
%R97F.0	DEFBUS	Bit défaut général liaison sur le bus E/S série (**)

(*) *Ce compteur est incrémenté par le système à chaque détection d'un dépassement, il est mis à ZERO par le programme utilisateur.*

(**) *Ces bits sont mis à UN par le système à chaque détection d'un défaut.*

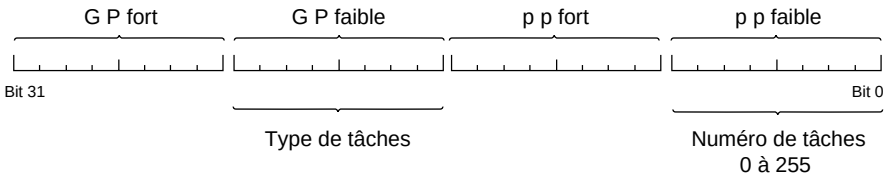
3.8.5.2 Diagnostic système

Les variables suivantes donnent la consommation (En % du temps) du moniteur et de chaque tâche automate.

Variable	Mnémonique	Description
%R950.B	Sys_avr1	Temps moyen d'occupation du moniteur sur le cycle %TS1
%R951.B	Sys_max1	Temps maximum d'occupation du moniteur sur le cycle %TS1
%R952.B	Ts0_avr1	Temps moyen d'occupation de la tâche %TS0 sur le cycle %TS1
%R953.B	Ts0_max1	Temps maximum d'occupation de la tâche %TS0 sur le cycle %TS1
%R954.B	Ts1_avr	Temps moyen d'occupation de la tâche %TS1
%R955.B	Ts1_max	Temps maximum d'occupation de la tâche %TS1
%R956.W	Overrun1	Dépassement temps de calcul sur le cycle %TS1
%R958.B	Sys_avr2	Temps moyen d'occupation du moniteur sur le cycle %TS2
%R959.B	Sys_max2	Temps maximum d'occupation du moniteur sur le cycle %TS2
%R95A.B	Ts0_avr2	Temps moyen d'occupation de la tâche %TS0 sur le cycle %TS2
%R95B.B	Ts0_max2	Temps maximum d'occupation de la tâche %TS0 sur le cycle %TS2
%R95C.B	Ts2_avr	Temps moyen d'occupation de la tâche %TS2
%R95D.B	Ts2_max	Temps maximum d'occupation de la tâche %TS2
%R95E.W	Overrun2	Dépassement temps de calcul sur le cycle %TS2
%R960.B	Sys_avr3	Temps moyen d'occupation du moniteur sur le cycle %TS3
%R961.B	Sys_max3	Temps maximum d'occupation du moniteur sur le cycle %TS3
%R962.B	Ts0_avr3	Temps moyen d'occupation de la tâche %TS0 sur le cycle %TS3
%R963.B	Ts0_max3	Temps maximum d'occupation de la tâche %TS0 sur le cycle %TS3
%R964.B	Ts3_avr	Temps moyen d'occupation de la tâche %TS3
%R965.B	Ts3_max	Temps maximum d'occupation de la tâche %TS3
%R966.W	Overrun3	Dépassement temps de calcul sur le cycle %TS3
%R968.B	Sys_avr4	Temps moyen d'occupation du moniteur sur le cycle %TS4
%R969.B	Sys_max4	Temps maximum d'occupation du moniteur sur le cycle %TS4
%R96A.B	Ts0_avr4	Temps moyen d'occupation de la tâche %TS0 sur le cycle %TS4

Variable	Mnémonique	Description
%R96B.B	Ts0_max4	Temps maximum d'occupation de la tâche %TS0 sur le cycle %TS4
%R96C.B	Ts4_avr	Temps moyen d'occupation de la tâche %TS4
%R96D.B	Ts4_max	Temps maximum d'occupation de la tâche %TS4
%R96E.W	Overrun4	Dépassement temps de calcul sur le cycle %TS4
%R970.B	Sys_avr5	Temps moyen d'occupation du moniteur sur le cycle %TS5
%R971.B	Sys_max5	Temps maximum d'occupation du moniteur sur le cycle %TS5
%R972.B	Ts0_avr5	Temps moyen d'occupation de la tâche %TS0 sur le cycle %TS5
%R973.B	Ts0_max5	Temps maximum d'occupation de la tâche %TS0 sur le cycle %TS5
%R974.B	Ts5_avr	Temps moyen d'occupation de la tâche %TS5
%R975.B	Ts5_max	Temps maximum d'occupation de la tâche %TS5
%R976.W	Overrun5	Dépassement temps de calcul sur le cycle %TS5

3.8.6 Choix du module à animer

Variable	Mnémonique	Description
%W97A.L		Type et Numéro de tâche %W97A.L donne le type et le numéro de tâche du module à animer.  Les valeurs codées pour le type de tâche sont : - 1 pour une tâche %TS, - 2 pour une tâche %TF, - 3 pour une tâche %SP - 4 pour une tâche %TH - 5 pour une tâche %INI
%W97E.B		Numéro du composant % W97E.B donne le numéro du composant à animer dans le module.

Si ces deux variables sont cohérentes, le composant du module spécifié est ouvert et animé.Dans le cas contraire, la liste de tous les modules chargés sur l'automate est proposée.

Exemple

%W97A.L = 0x00300F0

%W97E.B = 2

Le composant N° 2 du module SP240 sera ouvert et animé.

3.8.7 Autorisation d'écriture des cartes sorties %W900.0

Variable	Mnémonique	Description
%W900.0	INIB_E33	Autorisation d'écriture par programmation pièce des cartes sorties. Les variables %Qrc3B.1 doivent avoir été préalablement programmées dans %INI. La mise à 1 interdit l'écriture, par paramètres E33xxx, des variables %Qrc dans un programme pièce ou par opérateurs dynamiques. La mise à 0 autorise l'écriture.

3.8.8 Gestion défaut système

Ces variables permettent de contrôler les actions du moniteur lorsqu'il y a détection de défauts système ou de configuration.

Les variables de gestion des défauts systèmes seront traitées ultérieurement.

3.8.9 Paramètres externes E30xxx, E40xxx et E42xxx

⚠ ATTENTION

Les paramètres E30xxx et E40xxx ne sont pas sauvegardés. Ils sont initialisés à la mise sous tension.

Les paramètres E42xxx sont sauvegardés.

3.8.9.1 Paramètres externes E30xxx

128 mots de 32 bits sont adressés par E30000 à E30127.

Les paramètres E300xx sont lus et écrits par et pendant l'exécution du programme pièce. Ils transmettent des valeurs numériques signées significatives qui sont lues par le programme utilisateur.

Mnémonique	P.Fort			P.Faible
E30000	%RA00	%RA01	%RA02	%RA03
à				
E30031	%RA7C	%RA7D	%RA7E	%RA7F
E30032	%RB00	%RB01	%RB02	%RB03
à				
E30063	%RB7C	%RB7D	%RB7E	%RB7F
E30064	%RC00	%RC01	%RC02	%RC03
à				
E30095	%RC7C	%RC7D	%RC7E	%RC7F
E30096	%RD00	%RD01	%RD02	%RD03
à				
E30127	%RD7C	%RD7D	%RD7E	%RD7F

3.8.9.2 Paramètres externes E40xxx

128 mots de 32 bits sont adressés par E40000 à E40127.

Les paramètres E400xx sont écrits par le programme utilisateur. Il permettent d'introduire dans le programme pièce des valeurs numériques signées qui peuvent être des cotes à atteindre, des décalages, .. etc ....

Mnémonique	P.Fort			P.Faible
E40000	%WA00	%WA01	%WA02	%WA03
à				
E40031	%WA7C	%WA7D	%WA7E	%WA7F
E40032	%WB00	%WB01	%WB02	%WB03
à				
E40063	%WB7C	%WB7D	%WB7E	%WB7F
E40064	%WC00	%WC01	%WC02	%WC03
à				
E40095	%WC7C	%WC7D	%WC7E	%WC7F
E40096	%WD00	%WD01	%WD02	%WD03
à				
E40127	%WD7C	%WD7D	%WD7E	%WD7F

3.8.9.3 Paramètres E42xxx

128 mots adressés de E42000 à E42127. Ces paramètres peuvent être lus et écrits par le programme utilisateur (Fonctions R_E42000 (..) et W_E42000 (..) et par le programme pièce. Ils sont accessibles en lecture et écriture par opérateurs dynamiques.

REMARQUE : Il n'y a pas de garantie de cohérence des échanges au niveau du système (Une lecture par la fonction automatisme peut par exemple être interrompue par une écriture de la fonction CN) Il appartient donc à l'utilisateur de mettre en place un mécanisme de contrôle des échanges.

3.8.10 Organisation physique des variables %R et %W

Les variables %R et %W sont organisées en blocs de 128 octets %R suivi par 128 octets %W puis à nouveau 128 octets %R et ainsi de suite jusqu'à la fin de la famille.

Variables réservées non affectées

Les variables en entrée %RE00 à %RE7F et %RF00 à %RF7F sont réservées mais non affectées.

Les variables en sortie %WE20 à %WE7F et %WF00 à %WF7F sont réservées mais non affectées.

Tableau récapitulatif

Organisation physique des variables %R et %W (total 4 koctets)

Variables	Désignation
%R0 à %R7F	128 octets en entrée venant de la CN
%W0 à %W7F	128 octets en sortie vers la CN
%R100 à %R17F	128 octets en entrée venant du groupe d'axes 1
%W100 à %W17F	128 octets en sortie vers le groupe d'axes 1
%Rg00 à %Rg7F	6 groupes de 128 octets en entrée venant des groupes d'axes 2 à 7
%Wg00 à %Wg7F	6 groupes de 128 octets en sortie vers les groupes d'axes 2 à 7
%R800 à %R87F	128 octets en entrée venant du groupe d'axes 8
%W800 à %W87F	128 octets en sortie vers le groupe d'axes 8
%R900 à %R97F	128 octets en entrée (défauts internes)
%W900 à %W97F	128 octets en sorties (défauts internes)
%RA00 à %RA7F	128 octets en entrée paramètres E30000 à E30031
%WA00 à %WA7F	128 octets en sortie paramètres E40000 à E40031
%RB00 à %RB7F	128 octets en entrée paramètres E30032 à E30063
%WB00 à %WB7F	128 octets en sortie paramètres E40032 à E40063
%RC00 à %RC7F	128 octets en entrée paramètres E30064 à E30095
%WC00 à %WC7F	128 octets en sortie paramètres E40064 à E40095
%RD00 à %RD7F	128 octets en entrée paramètres E30096 à E30127
%WD00 à %WD7F	128 octets en sortie paramètres E40096 à E40127
%WE00 à %WE1F	32 octets en sortie vers la CN, réduction de courant.
%WF20 à %WF7F	réservées non affectées
%RF00 à %RF7F	réservées non affectées

3.9 Variables mots communs %S

Connecté aux réseaux MAPWAY ou ETHWAY, la commande numérique offre une ouverture au service mots communs des automates de la gamme TSX de Telemecanique. L'ensemble des mots communs constitue une base de données distribuée entre les stations d'un même réseau, chaque station pouvant être indifféremment un automate TSX ou une commande numérique.

Les stations participant au service mots communs se partagent une mémoire commune de 256 mots de 16 bits.

Chaque station dispose selon la configuration, de 4 à 64 mots communs (accessible en écriture) de la mémoire commune. Les mots affectés aux autres stations ne lui sont accessibles qu'en lecture.

3.9.1 Actualisation des variables

L'actualisation des variables %S est faite automatiquement par le système au rythme de la tâche séquentielle %TS0 et sans intervention du programme utilisateur.

En début de %TS0, la fonction automatisme va lire dans l'interface associée au processeur réseau, l'ensemble des mots communs ayant évolués dans les autres stations.

En fin de %TS0, la fonction automatisme écrit dans l'interface associée au processeur réseau, les mots communs de sa station.

Le coupleur réseau compare ces valeurs aux valeurs précédemment émises. Il n'émettra une trame que si une des valeurs au moins à évoluée ou après 30 cycle HTR s'il n'a pas émis depuis.

3.9.2 Configuration des mots communs

La configuration consiste :
- à définir le numéro de réseau et de station dans le paramètre machine P100 (Voir manuel des paramètres),
- à programmer dans la tâche %INI l'activité de la station et le nombre de mots communs par station par l'appel de la fonction setcomw(..).

3.9.3 Organisation des variables mots communs %S

Les variables %S sont organisées en 64 blocs de 128 octets indépendamment de la configuration des mots communs.

Le numéro d'une variable %S est codé sur quatre digits hexadécimaux. Les deux digits de poids faible indiquent le numéro de l'octet dans la station (de 0x0 à 0x7F) et les deux digits de poids fort indiquent le numéro de la station (de 0x0 à 0x3F). Ainsi %S21F.B représente l'octet 31 de la station 2.

Bloc	Variables	Taille
Station 0	De %S0 à %S7F	128 octets
Station 1	De %S100 à %S17F	128 octets
Stations 2 à 61		59 blocs de 128 octets
Station 62 (0x3E)	De %S3E00 à %S3E7F	128 octets
Station 63 (0x3F)	De %S3F00 à %S3F6F	112 octets
Diagnostic	De %S3F70 à %S3F7F	16 octets

Les variables %S3F70.B à %S3F77.B contiennent les bits indicateurs de rafraîchissement des stations :

Variables	Description
%S3F70.0 à %S3F70.7	Indicateurs de rafraîchissement des stations 0 à 7
%S3F71.0 à %S3F71.7	Indicateurs de rafraîchissement des stations 8 à 15
%S3F72.0 à %S3F72.7	Indicateurs de rafraîchissement des stations 16 à 23
%S3F73.0 à %S3F73.7	Indicateurs de rafraîchissement des stations 24 à 31
%S3F74.0 à %S3F74.7	Indicateurs de rafraîchissement des stations 32 à 39
%S3F75.0 à %S3F75.7	Indicateurs de rafraîchissement des stations 40 à 47
%S3F76.0 à %S3F76.7	Indicateurs de rafraîchissement des stations 48 à 55
%S3F77.0 à %S3F77.7	Indicateurs de rafraîchissement des stations 56 à 63

Ces bits sont mis à 1 par le système lors du rafraîchissement des variables %S de la station correspondante. Leurs mises à 0 pour contrôler le bon fonctionnement des échanges est à la charge du programmeur.

L'octet %S3F79.B contient lorsque le service mots communs est actif le numéro de sa propre station.

Le mot %S3F7E.W est réservé au service après vente NUM.

REMARQUE : Si le service mots communs n'est pas actif, les variables %S peuvent être utilisées comme des variables banalisées non sauvegardées.

3.10 Variables locales %Y - Pointeurs

3.10.1 Généralités

Le programmeur dispose d'une base du microprocesseur. Cette base est associée aux variables %Y.

Les variables %Y sont utilisées de deux façons :

- comme variables locales associées à un module %SP. Dans ce cas, la base est initialisée par le système lors de l'appel d'un module %SP par la fonction spy(..). Ces variables %Y sont créées dans la pile lors de l'appel du module %SP et sont détruites au retour à l'appelant. Leur nombre est de 128 octets (de %Y0.B à %Y7F.B). Leur utilisation permet d'écrire des modules portables et réentrants
- comme variable pouvant remplacer n'importe quelles variables globales (%M; %V, %I, %Q, %R et %W). Dans ce cas le programmeur doit faire pointer la base sur le début de la zone visée avec la fonction y_init(..). Les variables %Y permettent d'accéder à un champ De 32767 octets (de %Y0.B à %Y7FFF.B). Elles sont utiles par exemple, lorsqu'un même traitement doit être effectué sur des blocs de variables différents.

De plus, les variables %Y autorise l'adressage indirect ou adressage par pointeur.

REMARQUES *Les variables %Y ne sont pas indispensables à la programmation et leur utilisation est réservée aux programmeurs expérimentés.*
Les variables %Y ne sont pas visualisables sur l'écran de la CN et sur l'outil de programmation PLCTOOL.
Les variables %Y ne sont pas accessibles par requête UNITE.
Lorsque l'on utilise la fonction y_init(..), on perd la visibilité des éventuelles variables locales du modules.

3.10.2 Adressage indirect - Pointeurs

L'adressage indirect par pointeur est autorisé partout ou une variable simple peut être employée à l'exception des index.

⚠ ATTENTION

Avant d'utiliser un adressage par pointeur %Yi -> , il faut :
que les variables %Y soient définies, c'est à dire que l'on se trouve dans un %SP appelé avec la fonction spy( ) ou que le registre de base des variables %Y ait été défini par la fonction y_init( ),
que le pointeur %Yi.L soit chargé avec une adresse valide.

Une variable pointée peut être associée à un mnémonique (Voir Manuel PLCTOOL - Outil de programmation langage ladder).

Il est conseillé, afin d'optimiser la vitesse, d'utiliser des numéros multiples de 4 pour les pointeurs (Ex : %Y0 ->, %Y4 ->, %Y8 ->, %YC ->, .. etc ..).

Syntaxe

<pointeur> -> <post déplacement> . <taille>

Element du langage	Se compose de	Remarque
<pointeur>	%Y0 à %Y7C	Variable %Y de taille .L (la taille est omise)
<post déplacement>	0 à ff	Valeur imédiate (en hexadécimal)
<taille>	.0 à .7, .B, .W ou .L	Pour accéder à une variable sur bit, octet, mot ou long mot

Exemple

%Y4 -> 0.5	L'adresse de la variable pointée est égale à l'adresse contenue dans le pointeur + le post déplacement «0».
%Y7c -> ff.B	L'adresse de la variable pointée est égale à l'adresse contenue dans le pointeur + le post déplacement «0xff».

3.10.3 Exemples d'utilisation des pointeurs

Traitement d'une chaîne de caractères

```
%V500.L = "ABCDEF"      // %V500.L contient l'adresse de début de la chaîne «ABCDEF»
%Y8.L = %V500.L         // Initialisation du pointeur avec l'adresse début de la chaîne
%Y8 -> 0.B == "A"       // Accés au premier caractère de la chaîne
%Y8 -> 5.B == "F"       // Accés au sixième caractère de la chaîne
%Y8.L += 1              // Incrément du pointeur
%Y8 -> 0.B == "B"       // Accés au deuxième caractère de la chaîne
```

Gestion de quatre pupitre machine

REMARQUE : Le programme d'exemple PUPITREP disponible sous PLCTOOL illustre l'utilisation des pointeurs.

Dans un %TS

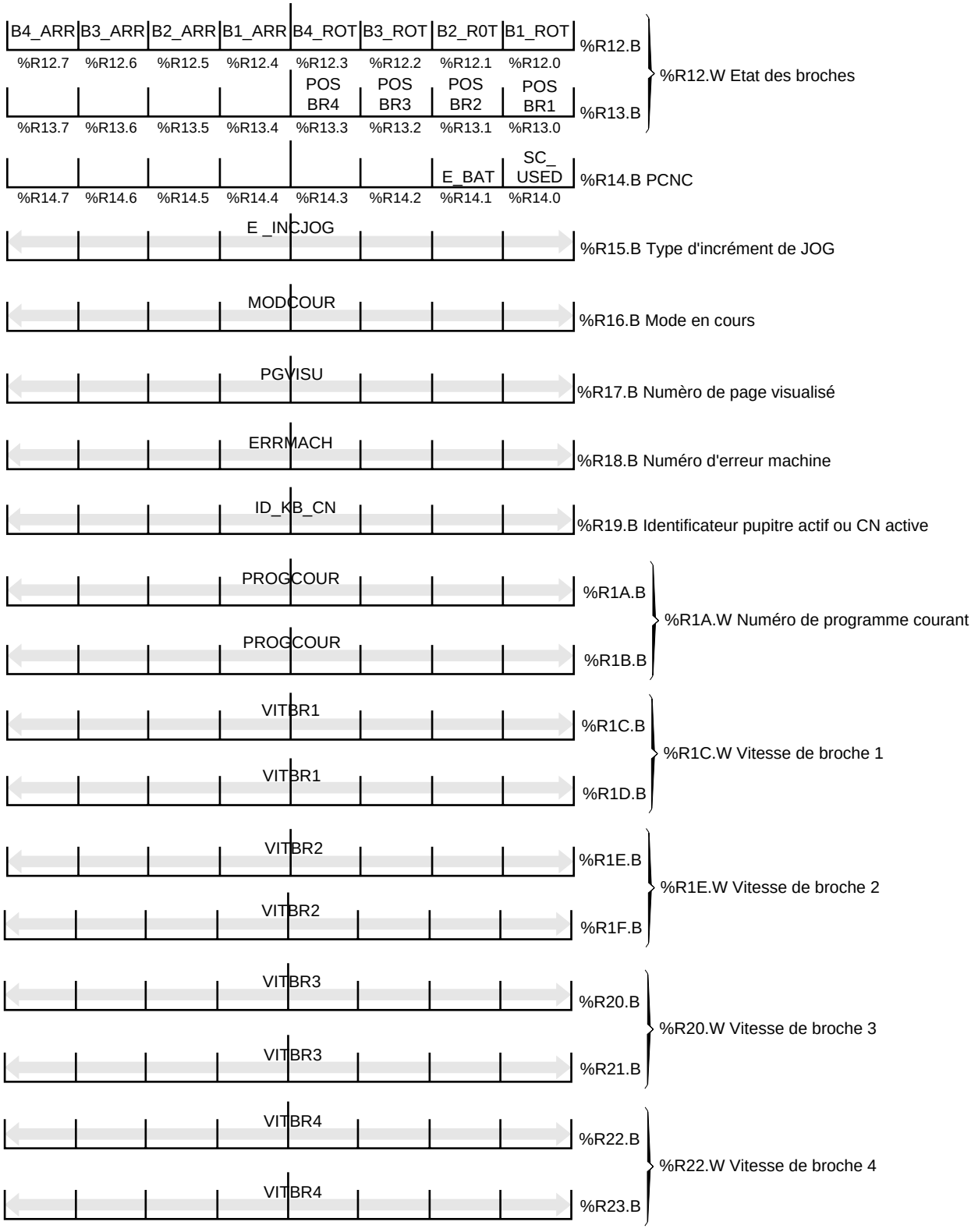
```
spy(0, %IrcOO.&, %Qrc00.&)   // Appel de %SP0 (Avec rc == numéro du pupitre de 1 à 4)
```

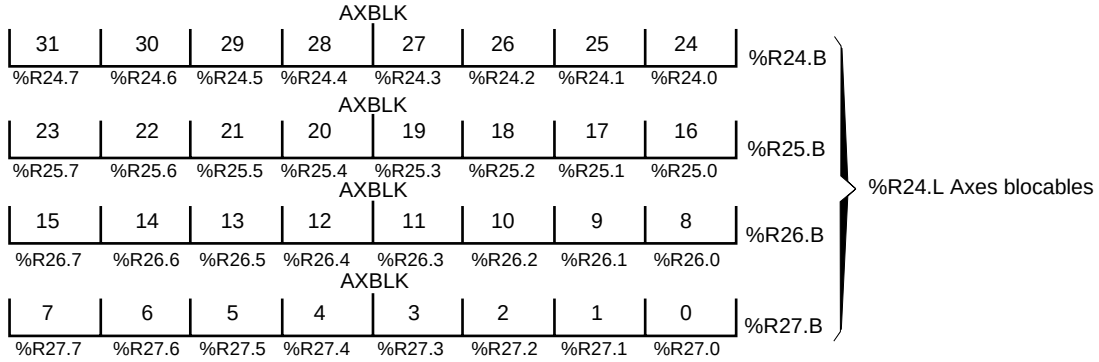
Dans %SP0

```
// %Y0.L contient l'adresse de la première entrée %Irc00
// %Y4.L contient l'adresse de la première sortie %Qrc00
%Y0 -> 2.0    <==> %Irc2.0     Jog_1
%Y0 -> 2.1    <==> %Irc2.1     Jog_10
%Y0 -> 2.2    <==> %Irc2.2     Jog_100
%Y0 -> 20.W   <==> %Irc20.W    Potentiomètre de broche
%Y0 -> 22.W   <==> %Irc22.W    Potentiomètre d'avance
%Y4 -> 0.0    <==> %Qrc0.0     Led_arus
%Y4 -> 0.1    <==> %Qrc0.1     Led_dcy
%Y4 -> 1.0    <==> %Qrc1.0     Led_1
%Y4 -> 1.1    <==> %Qrc1.1     Led_10
```

3.11 Zone d’échange

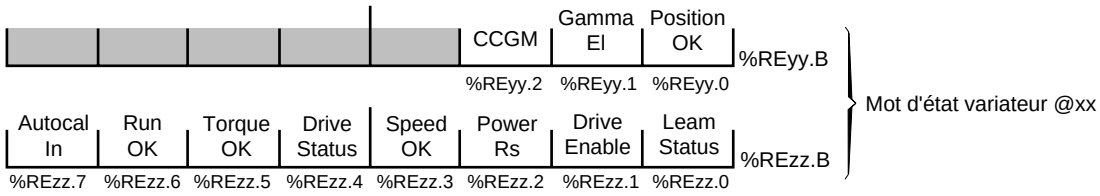
3.11.1 Entrées venant de la CN


B4_ARR B3_ARR B2_ARR B1_ARR B4_ROT B3_ROT B2_R0T B1_ROT
%R12.7 %R12.6 %R12.5 %R12.4 %R12.3 %R12.2 %R12.1 %R12.0
%R12.B
POS BR4 POS BR3 POS BR2 POS BR1
%R13.7 %R13.6 %R13.5 %R13.4 %R13.3 %R13.2 %R13.1 %R13.0
%R13.B
%R12.W Etat des broches
SC_ USED
E_BAT
%R14.7 %R14.6 %R14.5 %R14.4 %R14.3 %R14.2 %R14.1 %R14.0
%R14.B PCNC
E _INCJOG
%R15.B Type d'incrément de JOG
MODCOUR
%R16.B Mode en cours
PGVISU
%R17.B Numèro de page visualisé
ERRMACH
%R18.B Numéro d'erreur machine
ID_KB_CN
%R19.B Identificateur pupitre actif ou CN active
PROGCOUR
%R1A.B
PROGCOUR
%R1B.B
%R1A.W Numéro de programme courant
VITBR1
%R1C.B
VITBR1
%R1D.B
%R1C.W Vitesse de broche 1
VITBR2
%R1E.B
VITBR2
%R1F.B
%R1E.W Vitesse de broche 2
VITBR3
%R20.B
VITBR3
%R21.B
%R20.W Vitesse de broche 3
VITBR4
%R22.B
VITBR4
%R23.B
%R22.W Vitesse de broche 4



3.11.2 Zone d'échange CN - automate "1050"

Pour le variateur numérique d'adresse xx (xx compris entre 00 et 31), le mot d'état se présente sous la forme :



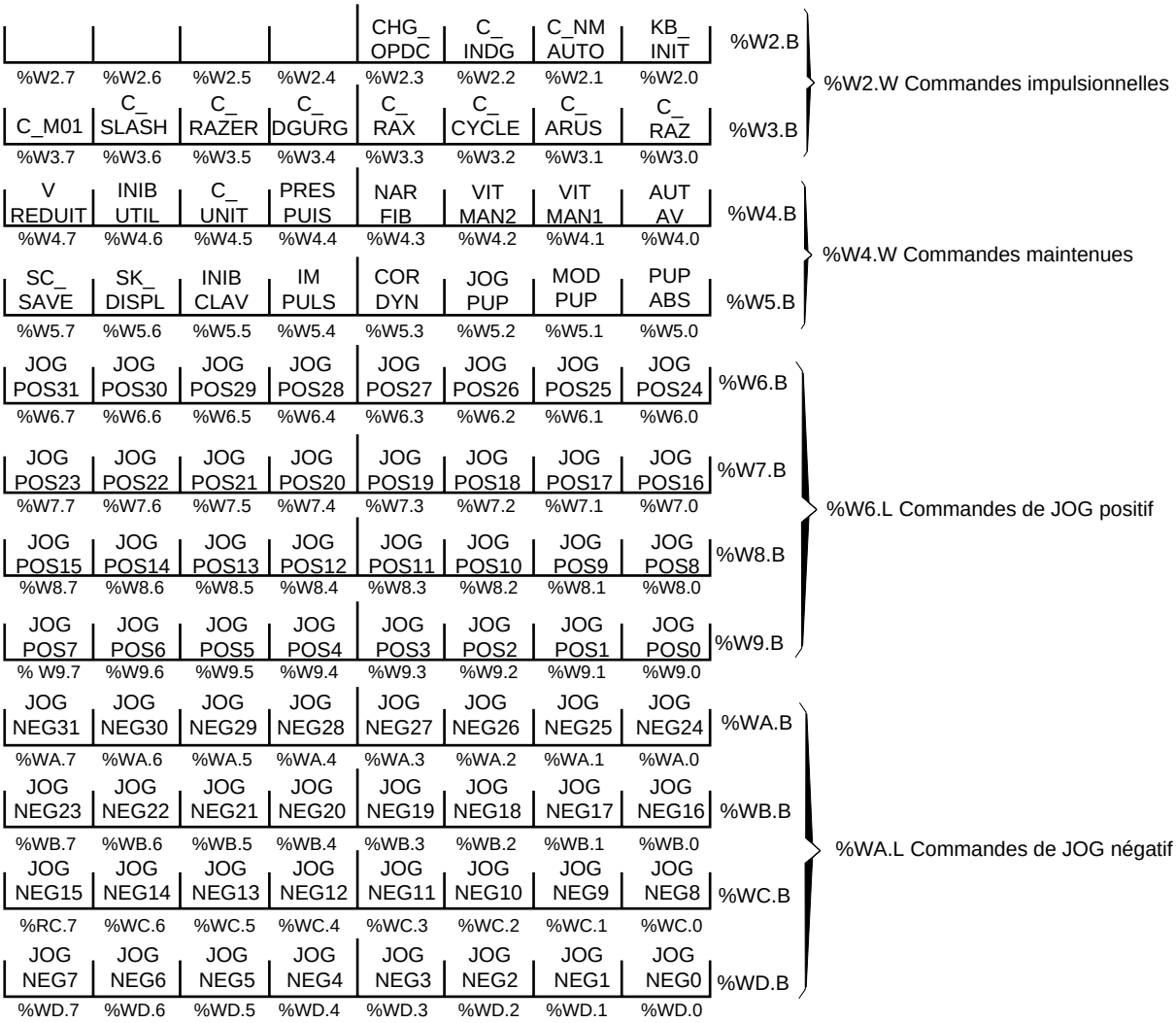
Valeurs de yy et zz en fonction de l'adresse du variateur xx :

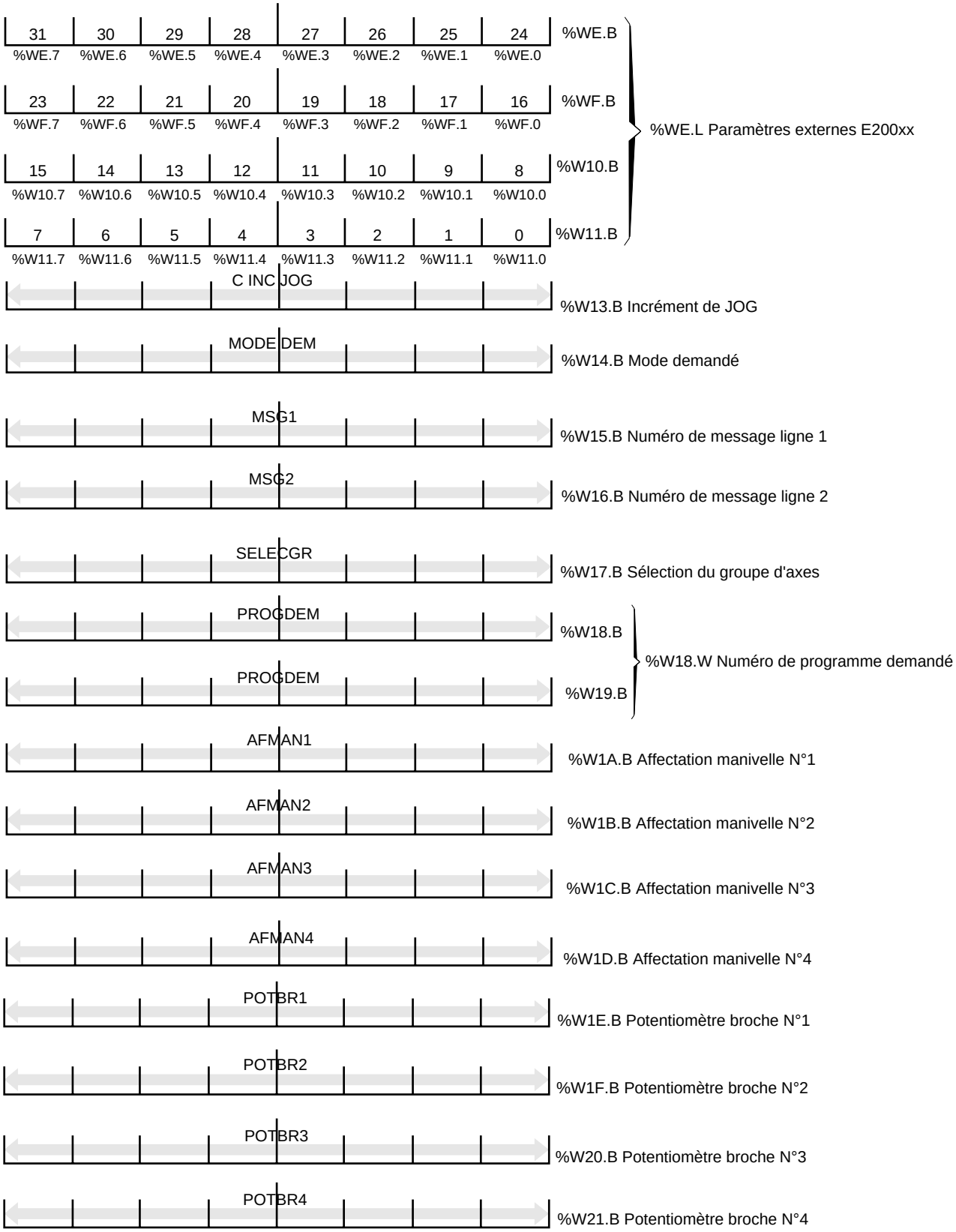
xx	00	01	02	03	04	05	06	07	08	09	10
yy	20	22	24	26	28	2A	2C	2E	30	32	34
zz	21	23	25	27	29	2B	2D	2F	31	33	35

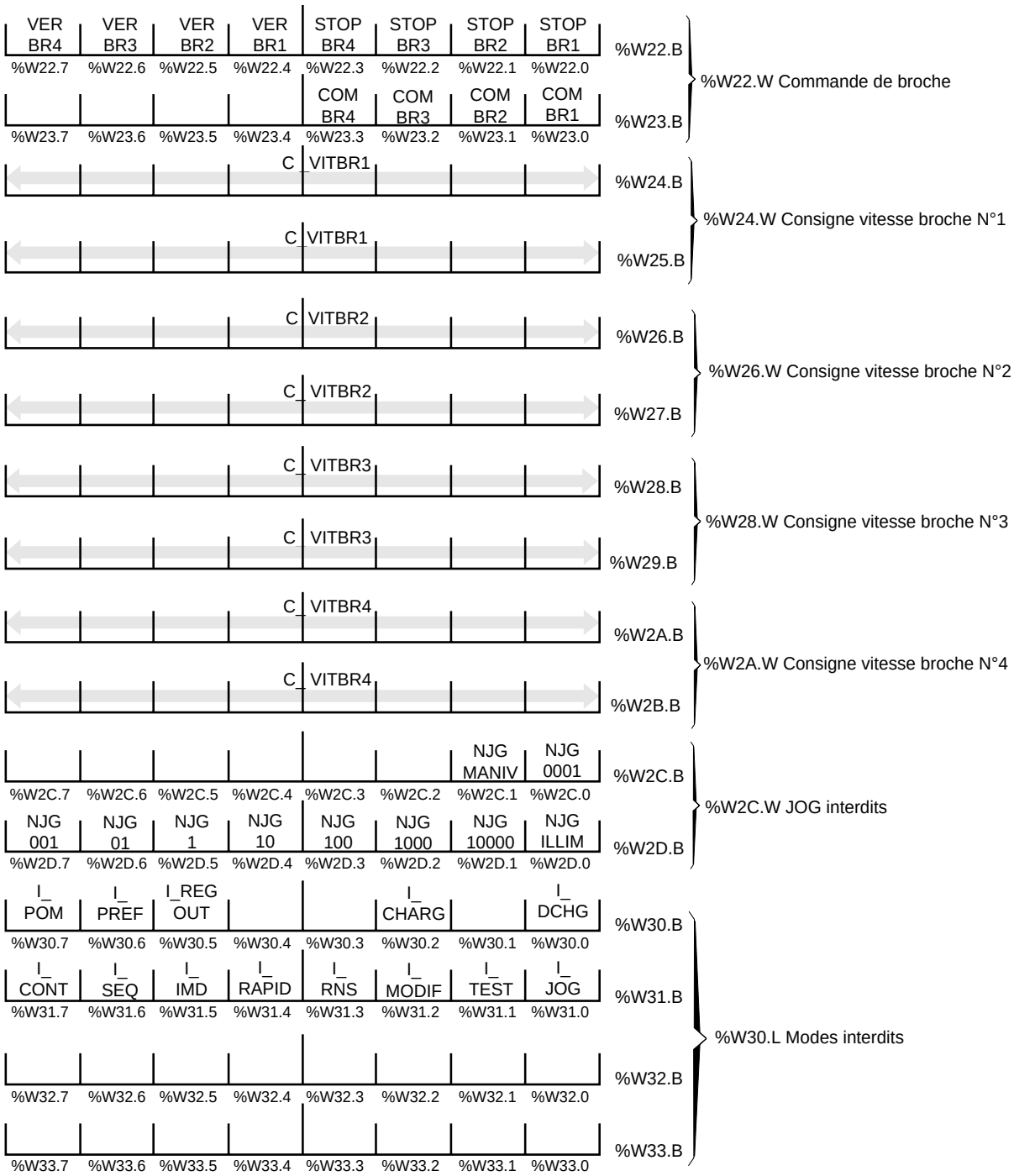
xx	11	12	13	14	15	16	17	18	19	20	21
yy	36	38	3A	3C	3E	40	42	44	46	48	4A
zz	37	39	3B	3D	3F	41	43	45	47	49	4B

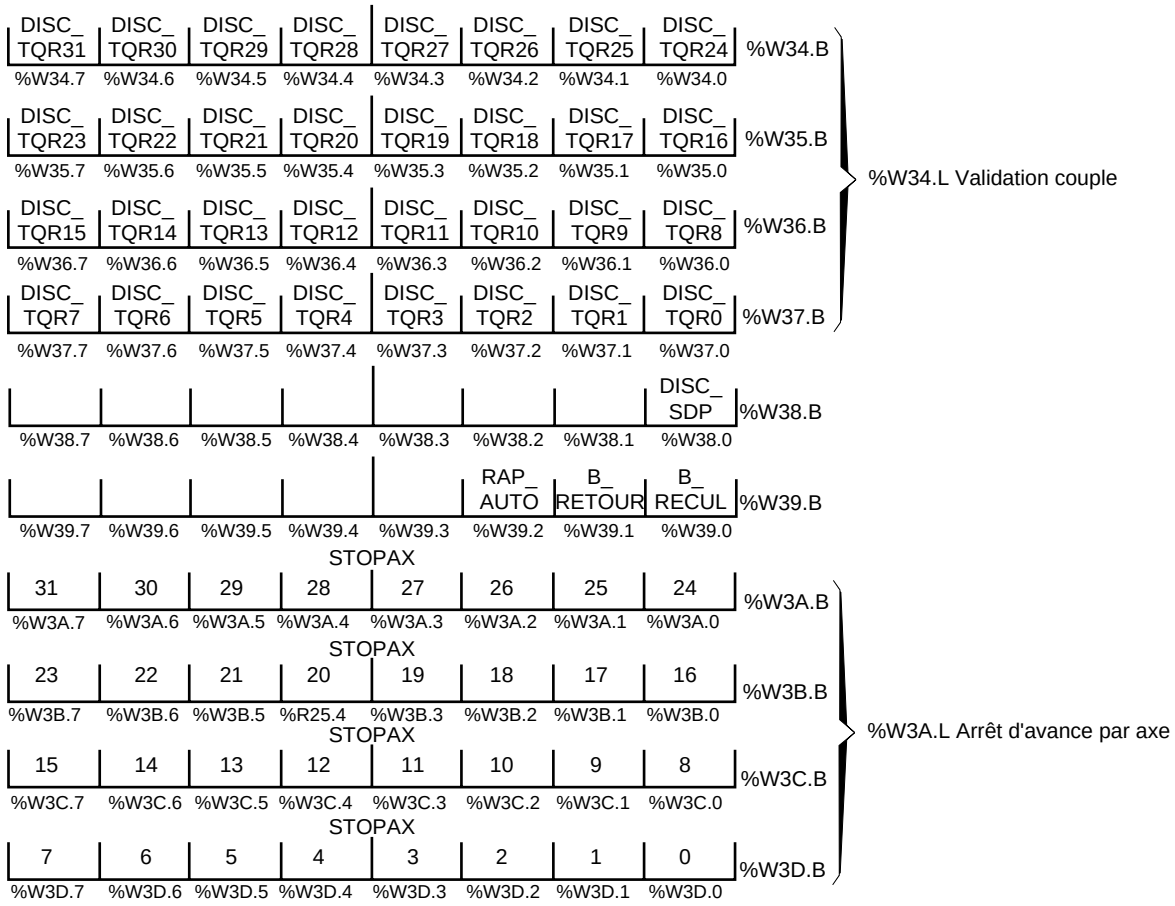
xx	22	23	24	25	26	27	28	29	30	31
yy	4C	4E	50	52	54	56	58	5A	5C	5E
zz	4D	4F	51	53	55	57	59	5B	5D	5F

3.11.3 Sorties vers la CN




31 30 29 28 27 26 25 24 %WE.B
%WE.7 %WE.6 %WE.5 %WE.4 %WE.3 %WE.2 %WE.1 %WE.0
23 22 21 20 19 18 17 16 %WF.B
%WF.7 %WF.6 %WF.5 %WF.4 %WF.3 %WF.2 %WF.1 %WF.0
15 14 13 12 11 10 9 8 %W10.B
%W10.7 %W10.6 %W10.5 %W10.4 %W10.3 %W10.2 %W10.1 %W10.0
7 6 5 4 3 2 1 0 %W11.B
%W11.7 %W11.6 %W11.5 %W11.4 %W11.3 %W11.2 %W11.1 %W11.0
%WE.L Paramètres externes E200xx
C INC JOG
%W13.B Incrément de JOG
MODE DEM
%W14.B Mode demandé
MSG1
%W15.B Numéro de message ligne 1
MSG2
%W16.B Numéro de message ligne 2
SELECGR
%W17.B Sélection du groupe d'axes
PROGDEM
%W18.B
PROGDEM
%W19.B
%W18.W Numéro de programme demandé
AFMAN1
%W1A.B Affectation manivelle N°1
AFMAN2
%W1B.B Affectation manivelle N°2
AFMAN3
%W1C.B Affectation manivelle N°3
AFMAN4
%W1D.B Affectation manivelle N°4
POTBR1
%W1E.B Potentiomètre broche N°1
POTBR2
%W1F.B Potentiomètre broche N°2
POTBR3
%W20.B Potentiomètre broche N°3
POTBR4
%W21.B Potentiomètre broche N°4

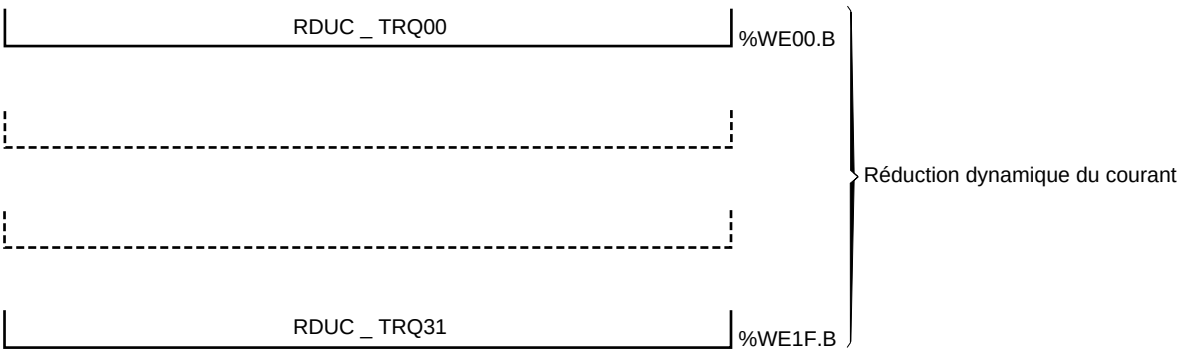

VER BR4 %W22.7
VER BR3 %W22.6
VER BR2 %W22.5
VER BR1 %W22.4
STOP BR4 %W22.3
STOP BR3 %W22.2
STOP BR2 %W22.1
STOP BR1 %W22.0
%W22.B
%W22.W Commande de broche
COM BR4 %W23.3
COM BR3 %W23.2
COM BR2 %W23.1
COM BR1 %W23.0
%W23.B
C_VITBR1
%W24.B
%W24.W Consigne vitesse broche N°1
C_VITBR1
%W25.B
C_VITBR2
%W26.B
%W26.W Consigne vitesse broche N°2
C_VITBR2
%W27.B
C_VITBR3
%W28.B
%W28.W Consigne vitesse broche N°3
C_VITBR3
%W29.B
C_VITBR4
%W2A.B
%W2A.W Consigne vitesse broche N°4
C_VITBR4
%W2B.B
NJG MANIV %W2C.1
NJG 0001 %W2C.0
%W2C.B
%W2C.W JOG interdits
NJG 001 %W2D.7
NJG 01 %W2D.6
NJG 1 %W2D.5
NJG 10 %W2D.4
NJG 100 %W2D.3
NJG 1000 %W2D.2
NJG 10000 %W2D.1
NJG ILLIM %W2D.0
%W2D.B
I_POM %W30.7
I_PREF %W30.6
I_REG OUT %W30.5
I_CHARG %W30.2
I_DCHG %W30.0
%W30.B
I_CONT %W31.7
I_SEQ %W31.6
I_IMD %W31.5
I_RAPID %W31.4
I_RNS %W31.3
I_MODIF %W31.2
I_TEST %W31.1
I_JOG %W31.0
%W31.B
%W32.B
%W33.B
%W30.L Modes interdits


DISC_ TQR31 DISC_ TQR30 DISC_ TQR29 DISC_ TQR28 DISC_ TQR27 DISC_ TQR26 DISC_ TQR25 DISC_ TQR24 %W34.B
%W34.7 %W34.6 %W34.5 %W34.4 %W34.3 %W34.2 %W34.1 %W34.0
DISC_ TQR23 DISC_ TQR22 DISC_ TQR21 DISC_ TQR20 DISC_ TQR19 DISC_ TQR18 DISC_ TQR17 DISC_ TQR16 %W35.B
%W35.7 %W35.6 %W35.5 %W35.4 %W35.3 %W35.2 %W35.1 %W35.0
DISC_ TQR15 DISC_ TQR14 DISC_ TQR13 DISC_ TQR12 DISC_ TQR11 DISC_ TQR10 DISC_ TQR9 DISC_ TQR8 %W36.B
%W36.7 %W36.6 %W36.5 %W36.4 %W36.3 %W36.2 %W36.1 %W36.0
DISC_ TQR7 DISC_ TQR6 DISC_ TQR5 DISC_ TQR4 DISC_ TQR3 DISC_ TQR2 DISC_ TQR1 DISC_ TQR0 %W37.B
%W37.7 %W37.6 %W37.5 %W37.4 %W37.3 %W37.2 %W37.1 %W37.0
%W34.L Validation couple
DISC_ SDP %W38.B
%W38.7 %W38.6 %W38.5 %W38.4 %W38.3 %W38.2 %W38.1 %W38.0
RAP_ AUTO B_ RETOUR B_ RECUL %W39.B
%W39.7 %W39.6 %W39.5 %W39.4 %W39.3 %W39.2 %W39.1 %W39.0
STOPAX
31 30 29 28 27 26 25 24 %W3A.B
%W3A.7 %W3A.6 %W3A.5 %W3A.4 %W3A.3 %W3A.2 %W3A.1 %W3A.0
STOPAX
23 22 21 20 19 18 17 16 %W3B.B
%W3B.7 %W3B.6 %W3B.5 %R25.4 %W3B.3 %W3B.2 %W3B.1 %W3B.0
STOPAX
15 14 13 12 11 10 9 8 %W3C.B
%W3C.7 %W3C.6 %W3C.5 %W3C.4 %W3C.3 %W3C.2 %W3C.1 %W3C.0
STOPAX
7 6 5 4 3 2 1 0 %W3D.B
%W3D.7 %W3D.6 %W3D.5 %W3D.4 %W3D.3 %W3D.2 %W3D.1 %W3D.0
%W3A.L Arrêt d'avance par axe

3.11.4 Zone d'échange automate - CN "1050"

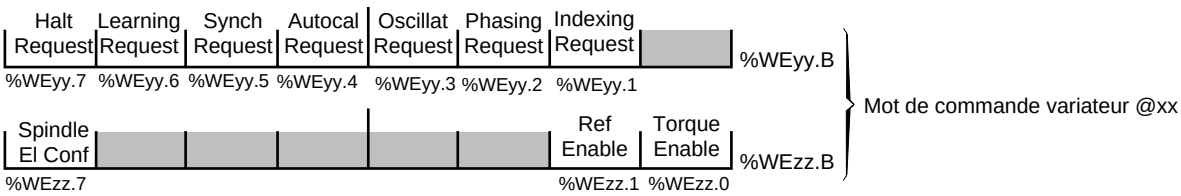
3.11.4.1 Modulation de couple

Il est possible de réduire dynamiquement le courant maximal par l'automate, sélectivement pour chaque variateur numérique.



3.11.4.2 Mot de commande variateur

Pour le variateur numérique d'adresse xx (xx compris entre 00 et 31), le mot de commande se présente sous la forme :



Valeurs de yy et zz en fonction de l'adresse du variateur xx :

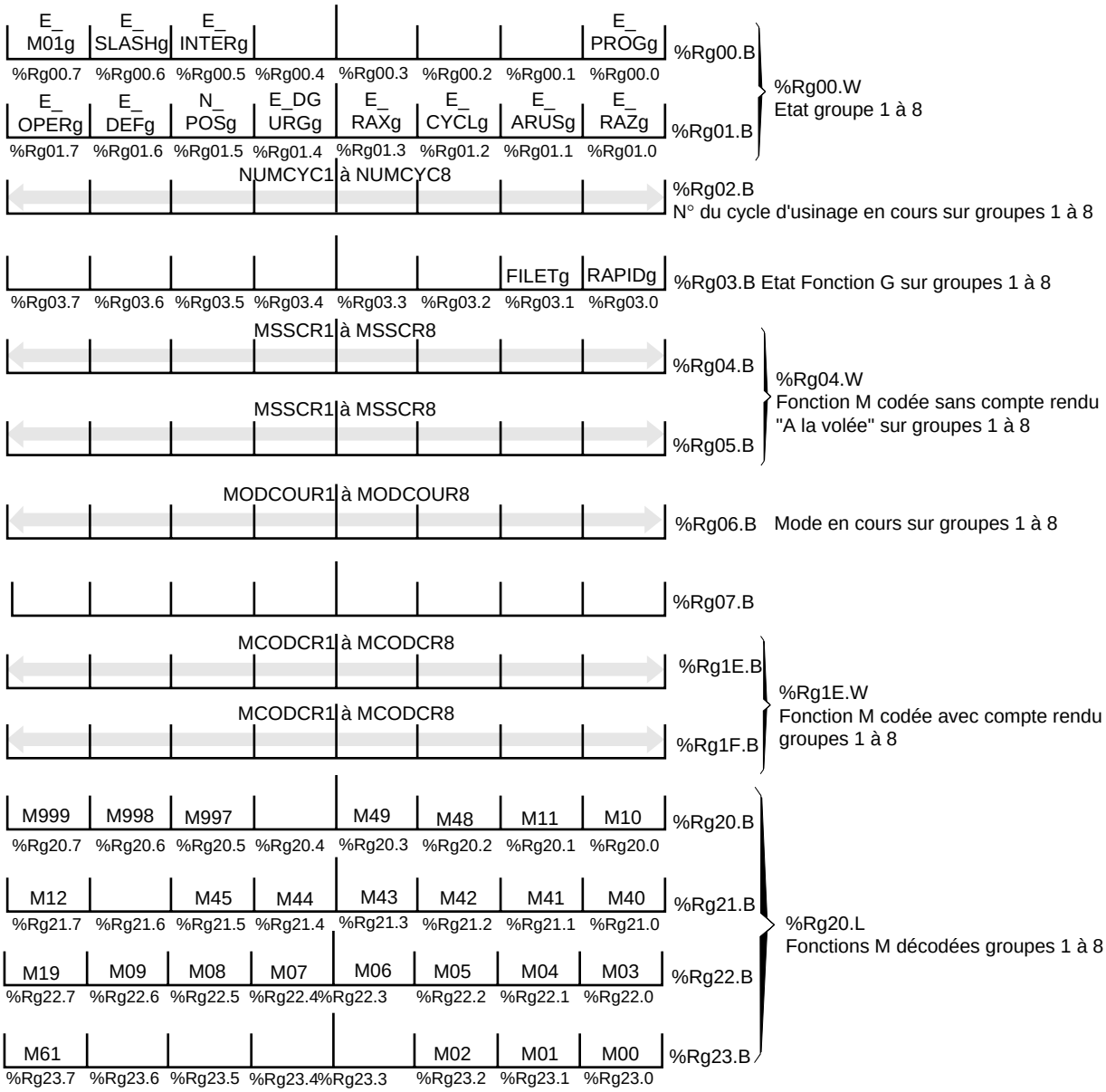
xx	00	01	02	03	04	05	06	07	08	09	10
yy	20	22	24	26	28	2A	2C	2E	30	32	34
zz	21	23	25	27	29	2B	2D	2F	31	33	35

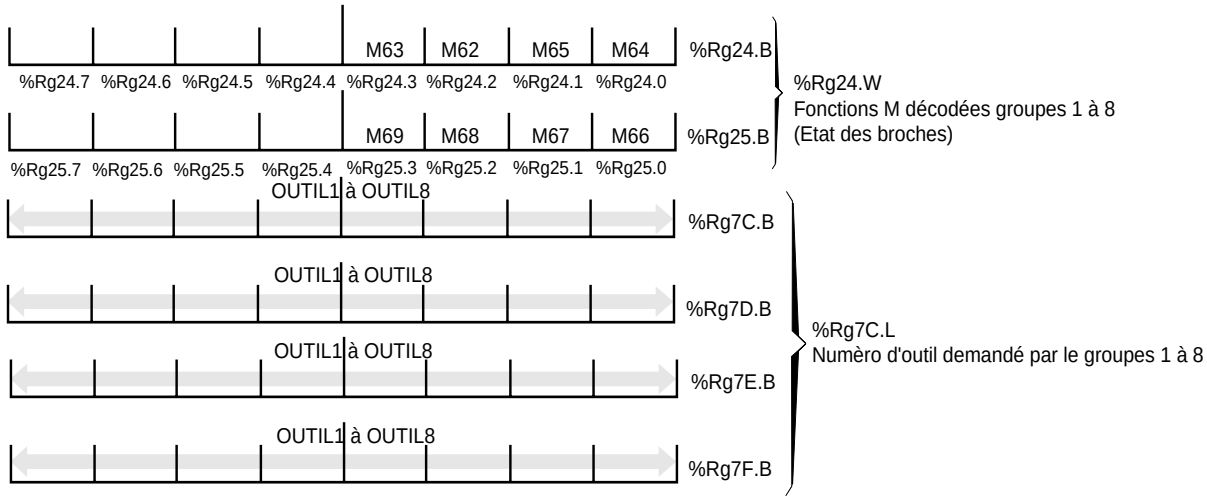
xx	11	12	13	14	15	16	17	18	19	20	21
yy	36	38	3A	3C	3E	40	42	44	46	48	4A
zz	37	39	3B	3D	3F	41	43	45	47	49	4B

xx	22	23	24	25	26	27	28	29	30	31	
yy	4C	4E	50	52	54	56	58	5A	5C	5E	
zz	4D	4F	51	53	55	57	59	5B	5D	5F	

3.11.5 Entrées venant des groupes d'axes

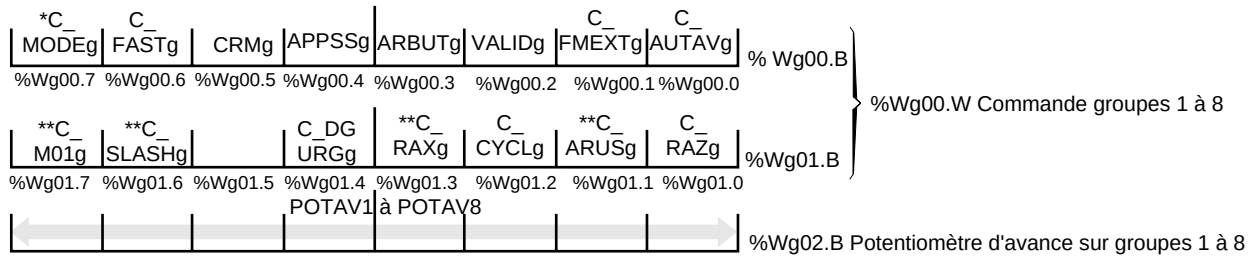
Dans cette grille, "g" prend la valeur du numéro de groupe (De 1 à 8)





3.11.6 Sorties vers les groupes d'axes

Dans cette grille, "g" prend la valeur du numéro de groupe (De 1 à 8)



* Valide uniquement pour les groupes d'axes automates
** Valide uniquement pour les groupes d'axes CN

4 Eléments littéraux du langage ladder

4.1 Notation utilisée

La notation utilisée pour décrire les éléments littéraux du langage est la suivante :

Caractères	Fonction
[ ]	Entre crochet signifie 0 ou 1 occurrence de ce qu'ils entourent
< >	Entourent les éléments non terminaux du langage
{}n	Les accolades signifient au plus n occurences de ce qu'ils entourent

REMARQUE : Un élément non entouré entre < et > est un symbole terminal, un mot clé ou un séparateur.

4.2 Label - commentaire

Elément du langage	Se compose de	Remarque
<label>	<lettre> ou <chiffre> ou _	Limité à 8 caractères
<commentaire>	<caractére> ou <blanc>	Limité à 64 caractères

4.3 Etape

Elément du langage	Se compose de	Remarque
<étape>	<variable_étape> <numéro_étape>	
	<variable_étape>	Variable %M, %V ou %Y de taille .W
	<numéro_étape>	Entier positif sur 16 bits

4.4 Eléments littéraux des séquences réseaux

4.4.1 Entités littérales autorisées en zone test d'un réseau

Elément du langage	Se compose de	Remarque
<variable_bit>	Variable % .0 à .7	Exemple : %V3.0
<comparaison>	<expression_numérique> <opérateur_comparaison> <expression_numérique>	
<affectation_numérique>	<variable_numérique> <opérateur_affectation> <expression_numérique>	
<appel_fonction>	[<variable_numérique> <opérateur_affectation>] <fonction>	

REMARQUE : L'évaluation de <variable_bit> et <comparaison> fournit un résultat booléen [1 ou 0].

4.4.2 Entités littérales autorisées en zone action d'un réseau

Elément du langage	Se compose de	Remarque
<variable_bit>	Variable % .0 à .7	Exemple :%V3.0
<affectation_numérique>	<variable_numérique> <opérateur_affectation> <expression_numérique>	
<appel_fonction>	[<variable_numérique> <opérateur_affectation>] <fonction>	
<goto_label>	goto(<label>)	Saut au label (interne au module) sans retour possible
<call_label>	call(<label>)	Saut au label (interne au module) avec retour
<return>	return([<expression_numérique>])	Retour au module appelant ou au call(<label>)

4.4.3 Grammaire des éléments littéraux

Elément du langage	Se compose de	Remarque
<fonction>	<nom_fonction> ( ) ou <nom_fonction> (<expression_numérique>) ou <nom_fonction> ({ <expression_numérique>, }6 <expression_numérique>)	
<nom_fonction>		Exemple :printf(...)
<expression_numérique>	<numérique_signé> { <opérateur_binaire> <numérique_signé> }n	Pour la détermination de n (Vo**ir 4.5).**
<numérique_signé>	[<opérateur_unaire>] <numérique_non_signé>	
<numérique_non_signé>	<variable_numérique> ou <entier_immédiat> ou (<expression_numérique>)	
<entier_immédiat>	<chiffre> { <chiffre> }9 ou 0x <chiffre_hexa> { <chiffre_hexa> }7	base dix base seize
<chiffre>	0, 1, 2, 3, 4, 5, 6, 7, 8 ou 9	
<chiffre_hexa>	<chiffre>, a, b, c, d, e, f, A, B, C, D, E ou F	
<variable_numérique>	Variable % .B ou .W ou .L ou .&	Exemple : %V3.L
<opérateur_comparaison>	== != >= <= > <	Egal Non égal Supérieur ou égal (comparaison signée) Inférieur ou égal (comparaison signée) Supérieur (comparaison signée) Inférieur (comparaison signée)
<opérateur_unaire>	- ~	Négation de l'opérande qui suit Inversion bit à bit de l'opérande qui suit

Elément du langage	Se compose de	Remarque
<opérateur_binaire>	*	Multiplication (signée)
	/	Division (signée)
	+	Addition
	-	Soustraction
	<<	Décalage arithmétique vers la gauche
	>>	Décalage arithmétique vers la droite
	&	ET bit à bit
	^	OU EXCLUSIF bit à bit
	|	OU bit à bit
<opérateur_affectation>	=	Affectation simple
	+=	Addition et affectation
	-=	Soustraction et affectation
	&=	ET bit à bit et affectation
	^=	OU EXCLUSIF bit à bit et affectation
	|=	OU bit à bit et affectation

4.5 Complément sur les éléments littéraux

4.5.1 Priorité des opérateurs

4.5.1.1 Priorité des opérateurs unaires

La priorité des opérateurs unaires est supérieure à celle des opérateurs binaires.

Priorité	Opérateur	Désignation
Plus prioritaire	[ ]	Indexation
	.&	Opérateur «adresse de «
	-	Négation
Moins prioritaire	~	Inversion bit à bit

4.5.1.2 Priorité des opérateurs binaires et comparaison

La priorité des opérateurs binaires et comparaison est supérieure à celle des opérateurs d'affectation.

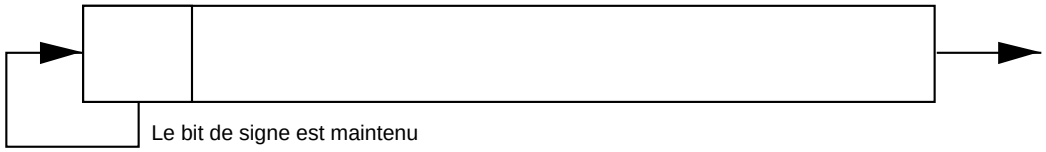
Priorité		Opérateur	Désignation
Plus prioritaire	P0	*	Multiplication
		/	Division
	P1	+	Addition
		-	Soustraction
	P2	>>	Décalage arithmétique à droite
		<<	Décalage arithmétique à gauche
	P3	&	ET bit à bit
	P4	^	OU EXCLUSIF bit à bit
	P5	|	OU bit à bit
Moins prioritaire	P6	== != >= <= > <	Opérateurs de comparaison

4.5.2 Opérateurs de comparaisons

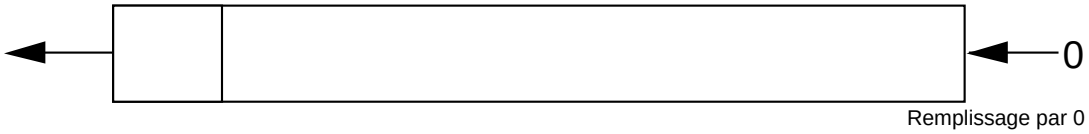
Les comparaisons sont des comparaisons signées c'est à dire que le bit de poids fort de la variable est considéré comme un bit de signe. (Les variables ayant le bit de signe à 1 sont inférieures aux variables ayant le bit de signe à 0).

4.5.3 Opérateurs >> et <<

>> Décalage arithmétique à droite de N modulo 64 bits



<< Décalage arithmétique à gauche de N modulo 64 bits



Remplissage par 0

REMARQUES: Permet de faire des divisions par des puissances de 2 plus rapidement que l'opérateur / (Var_1 / 2^n == Var_1 >> n).
*Permet de faire des multiplications par des puissances de 2 plus rapidement que l'opérateur * (Var_1 * 2^n == Var_1 << n).*

4.5.4 Opérateurs d'affectation

Les opérateurs d'affectation ont la plus faible priorité. L'affectation est donc effectuée en dernier.

4.5.4.1 Opérateurs =

L'affectation simple permet le chargement de la variable à gauche avec le résultat de l'expression numérique ou de la fonction à droite de l'opérateur d'affectation = .

4.5.4.2 Opérateurs combinés += -= &= ^= |=

Ces opérateurs combinent une opération entre la variable à gauche et le résultat de l'expression à droite suivie d'une affectation du résultat final dans la variable à gauche.

Exemples

Var_1 += <expression_numérique> est équivalent à : Var_1 = Var_1 + <expression_numérique>

Var_1 -= <expression_numérique> est équivalent à : Var_1 = Var_1 - <expression_numérique>

Var_1 &= <expression_numérique> est équivalent à : Var_1 = Var_1 & <expression_numérique>

Var_1 ^= <expression_numérique> est équivalent à : Var_1 = Var_1 ^ <expression_numérique>

Var_1 |= <expression_numérique> est équivalent à : Var_1 = Var_1 | <expression_numérique>

Les opérateurs combinés sont conseillés car ils permettent une génération de codes optimisée en vitesse et en taille.

4.5.5 Ordre d'évaluation des expressions

Dans une expression les opérations de plus forte priorité sont exécutées avant les opérations de priorité inférieure.

Les opérations de même priorité sont exécutées de gauche à droite.

Les parenthèses permettent de modifier l'ordre d'évaluation des expressions en forçant l'évaluation en premier de l'expression qu'elles entourent.

4.5.6 Entiers immédiats

Les entiers immédiats sont limités à 32 bits.

Le système considère les entiers comme signés , le bit 31 étant le bit de signe.

Un entier immédiat doit donc être compris entre:

Nature	Valeur
Entier négatif en décimal	de -2147483648 à -1
Entier négatif en hexadécimal	de 0x80000000 à 0xFFFFFFFF
Entier positif en décimal	de 0 à 2147483647
Entier positif en hexadécimal	de 0x0 à 0x7FFFFFFF

4.5.7 Promotion des variables - Format des calculs internes

Le système considère toutes les variables comme signées.

Variable sur octet

Le bit 7 indique le signe. $-128 \leq$ valeur d'un octet $\leq +127$

Variable sur mot

Le bit 15 indique le signe. $-32768 \leq$ valeur d'un mot $\leq +32767$.

Variable sur long mot

Le bit 31 indique le signe. -2147483648 $(-2^{31}) \leq$ valeur d'un long mot $\leq +2147483647$ $(2^{31} - 1)$.

Fonctionnement

Lorsque une variable est utilisée dans un calcul, elle est d'abord chargée dans un registre du microprocesseur.

Si la variable chargée était un octet , le système propage alors le bit 7 du registre sur les bits 8 à 31.

Si la variable chargée était un mot , le système propage alors le bit 15 du registre sur les bits 16 à 31.

Les calculs sont ensuite effectués avec les registres de 32 bits et génèrent un résultat sur 32 bits.

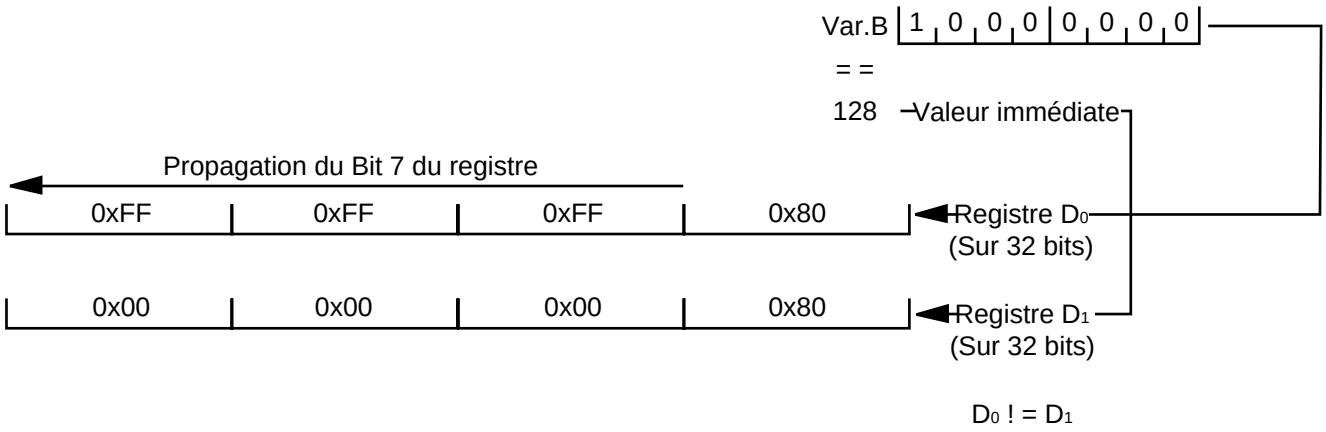
Ce résultat est alors chargé dans la variable destination :
- si la variable destination est un long mot alors les 32 bits du registre résultat lui sont affectés,
- si la variable destination est un mot alors les 16 bits de poids faible du registre résultat lui sont affectés,
- si la variable destination est un octet alors les 8 bits de poids faible du registre résultat lui sont affectés.

Piège à éviter

Les comparaisons entre variables (Octets et mots signés) et valeurs imédiates sont une source d'erreur fréquente.

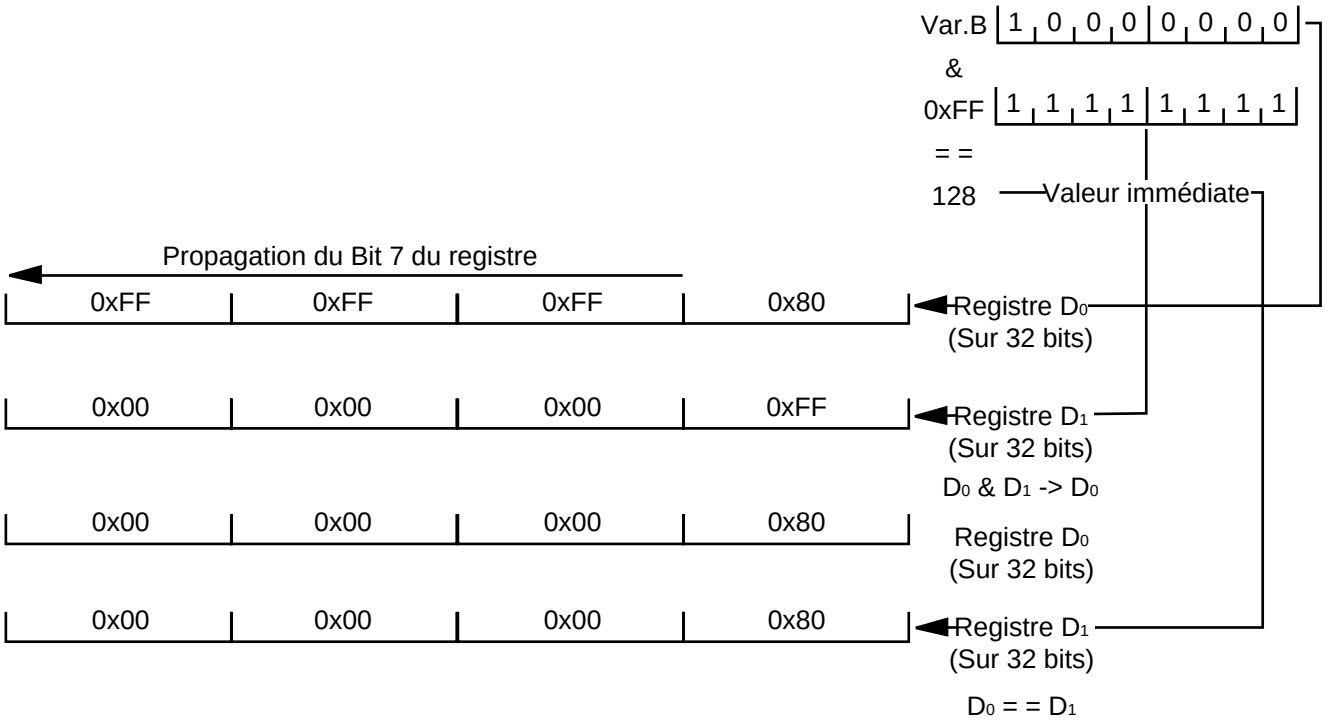
Exemple

Var.B == 128.



L'égalité entre la variable Var.B et la valeur imédiate 128 n'est jamais réalisée.

L'égalité peut être réalisée en utilisant un masque et en écrivant : Var.B & 0xFF == 128.



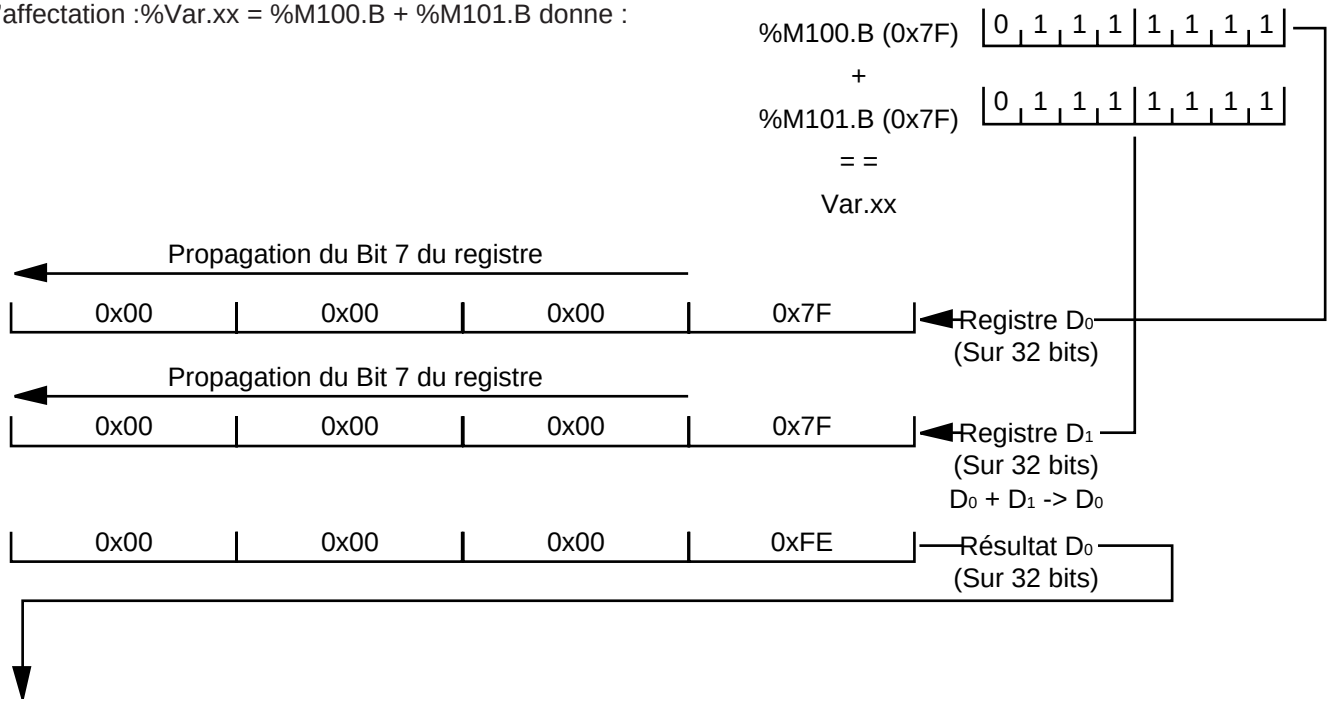
4.5.8 Débordement - Changement de signe

Le système n'effectue pas de contrôle de débordement. C'est donc au programmeur de prendre les précautions nécessaires.

Exemple de changement de signe

%M100.B et %M101.B sont deux variables sur octet qui valent toutes les deux 0x7F (soit +127).

L'affectation :%Var.xx = %M100.B + %M101.B donne :



Var.B est chargée avec 0xFE. Le bit 7 (Signe) étant à 1, Var.B== -2 (Résultat incorrect) 1 1 1 1 1 1 1 0

Var.W est chargée avec 0x00FE. Le bit 15 (Signe) étant à 0, Var.W == 254 (Résultat correct) 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0

Var.L est chargée avec 0x000000FE. Le bit 31 (Signe) étant à 0, Var.L == 254 (Résultat correct) 0x00 0x00 0x00 0xFE

4.5.9 Exemples d'entités littérales

Comparaisons

- %M5.B + %V33.L == %M10.W
- (Var_1 << 4 ^ 0x3) << %M10.B >= -((Var_6 & 0xF5) + (Var_3 & %M5.W))
- %I900.W & 1 << %V100.B != 0 // Test du bit N° %V100.B de %I900.W (Voir 5.2.7.4).

Affectations numériques

- %M5.B = %M33.L - %M10.W
- Var_1 = -%M10.B + ~(0xF5 - (Var_3 << %M5.W))
- Reste = Dividende - Dividende/Diviseur*Diviseur // calcul du reste d'une division entière

Appels fonction valides

- Var_1.L = printf(CHAINE_1, %M45.W << 4, %M100.L, (Var_4 & 0x33) + %M200.W);
- printf(%M250.L, %M45.W << 4, %M100.L, (Var_4 & 0x33) + %M200.W);

4.5.10 Longueur maximum d'une entité littérale

La longueur maximum d'une entité littérale est LGM_LITTERAL (soit 120 caractères).

4.5.11 Nombre maximum d'opérandes dans une expression numérique

Indépendamment de la longueur maximum de l'expression, le nombre maximum d'opérandes autorisés dans une expression numérique est limité par un autre critère: le nombre maximum de stockage NBM_DATA_REG (soit 5).

Ce dépassement est signalé lors de la compilation par le message «Erreur Nb maximum data register».

Exemple

L'expression numérique : Var_1 | Var_2 ^ Var_3 >> 8 + Var_4 * Var_5 qui génère la pile post fixée ci-dessous est refusée par le compilateur car le nombre maximum de stockage est dépassé.

Var_1	1° stockage
Var_2	2° stockage
Var_3	3° stockage
8	4° stockage
Var_4	5° stockage
Var_5	6° stockage * Erreur, plus de 5 niveaux de stockage
*	5° stockage
+	4° stockage
>>	3° stockage
^	2° stockage
|	1° stockage

Dans ce cas particulier une réorganisation de l'expression permet de réaliser le calcul, en effet l'expression équivalente : Var_3 >> Var_5 * Var_4 + 8 ^ Var_2 | Var_1 qui génère la pile post fixée ci-dessous est acceptée par le compilateur.

Var_3	1° stockage
Var_5	2° stockage
Var_4	3° stockage
*	2° stockage
8	3° stockage
+	2° stockage
>>	1° stockage
Var_2	2° stockage
^	1° stockage
Var_1	2° stockage
|	1° stockage

5 Programmation ladder

5.1 Eléments communs à tous les types de séquence

5.1.1 En-tête de séquence

Les séquences de type tableau de constantes, chaînes de caractères ou réseau possèdent une en-tête commune composée :
- d'un identificateur de séquence facultatif appelé label (Voir 4.2)
- d'un commentaire facultatif (Voir 4.2)
- d'une étape grafcet facultative (Voir 4.3).

5.1.2 Etape grafcet

Les étapes grafcet permettent d'augmenter la vitesse d'exécution d'un programme. En effet toutes les séquences non actives ne sont pas exécutées. Elles permettent de spécifier le logiciel suivant une méthodologie grafcet.

Si toutes les actions d'une étape grafcet ne peuvent être programmées dans une même séquence, le programmeur peut écrire autant de séquences qu'il le souhaite avec la même étape.

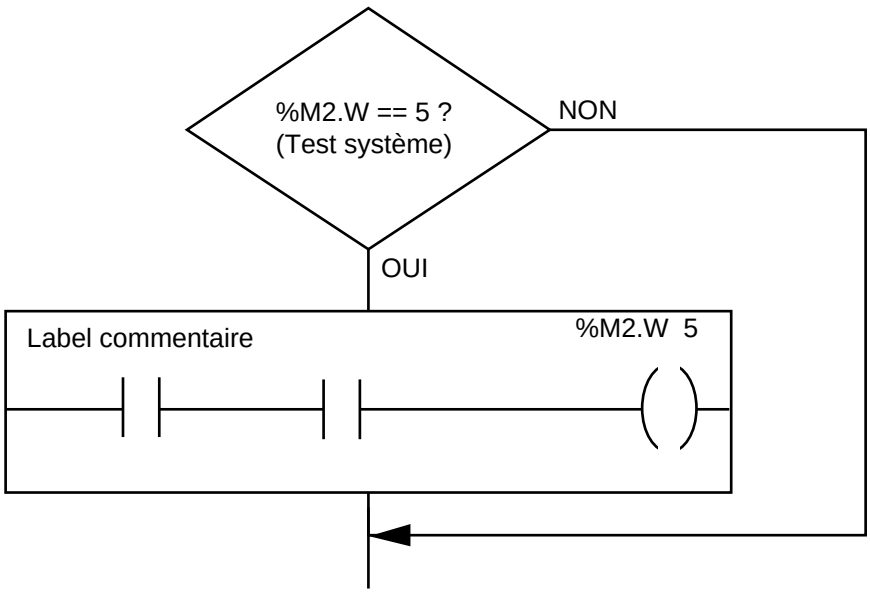
5.1.2.1 Présentation

Une séquence avec étape grafcet possède deux états :
- active lorsque <variable_étape> == <numéro_étape>,
- inactive lorsque <variable_étape> != <numéro_étape>.

5.1.2.2 Traitement des étapes grafcet par le système

Lorsqu'une séquence avec étape grafcet est active alors le système l'exécute comme une séquence sans étape.

Lorsqu'une séquence avec étape grafcet n'est pas active alors le système ne l'exécute pas.

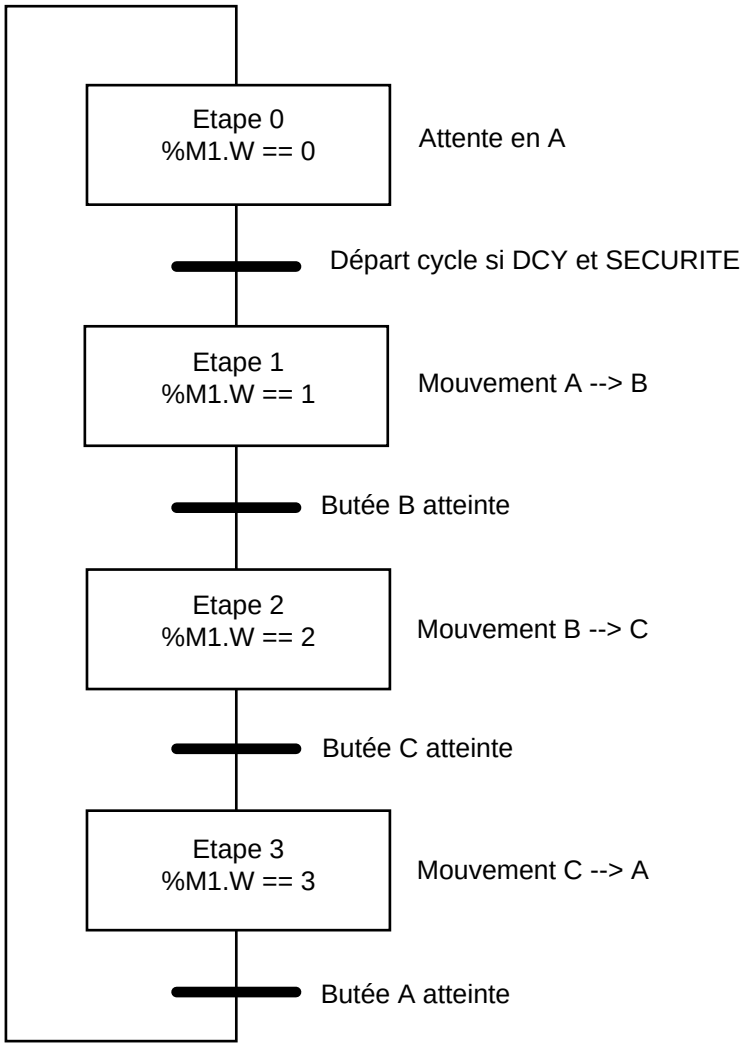


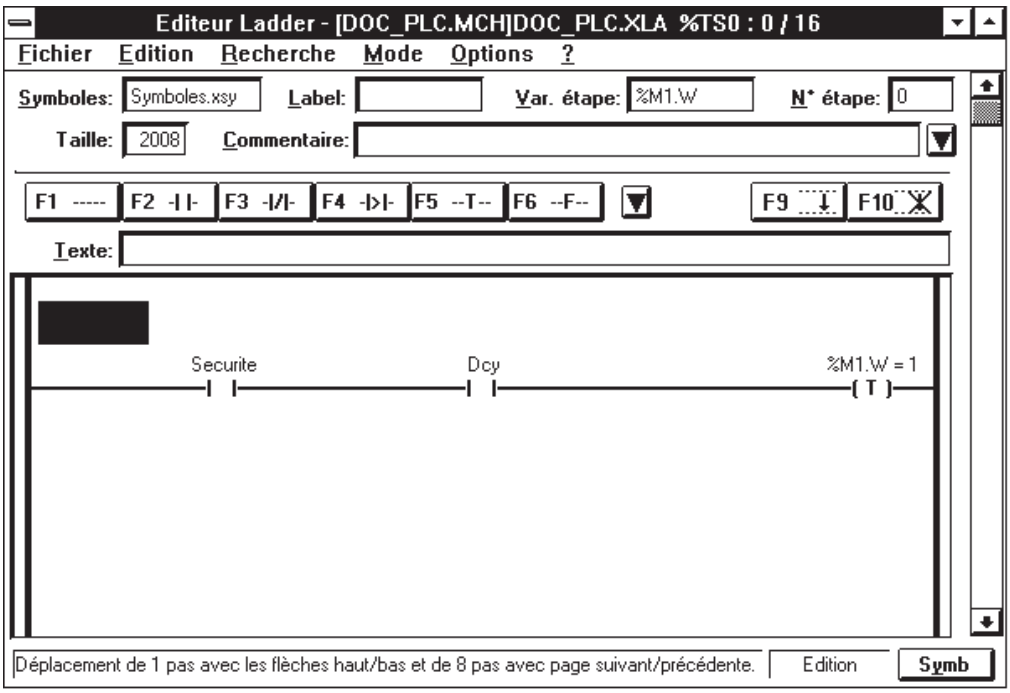
5.1.2.3 Activation/déactivation des étapes grafcet

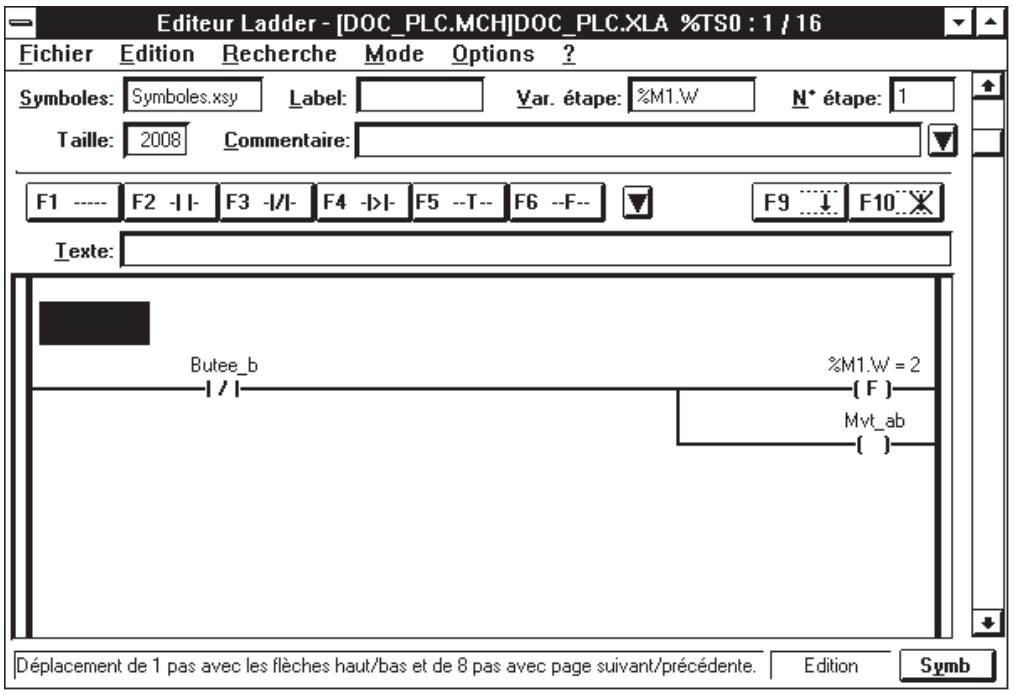
L'activation (désactivation) des séquences avec étapes se fait par programmation en chargeant la variable <variable_étape> avec l'entier correspondant à (aux) séquence(s) à activer.

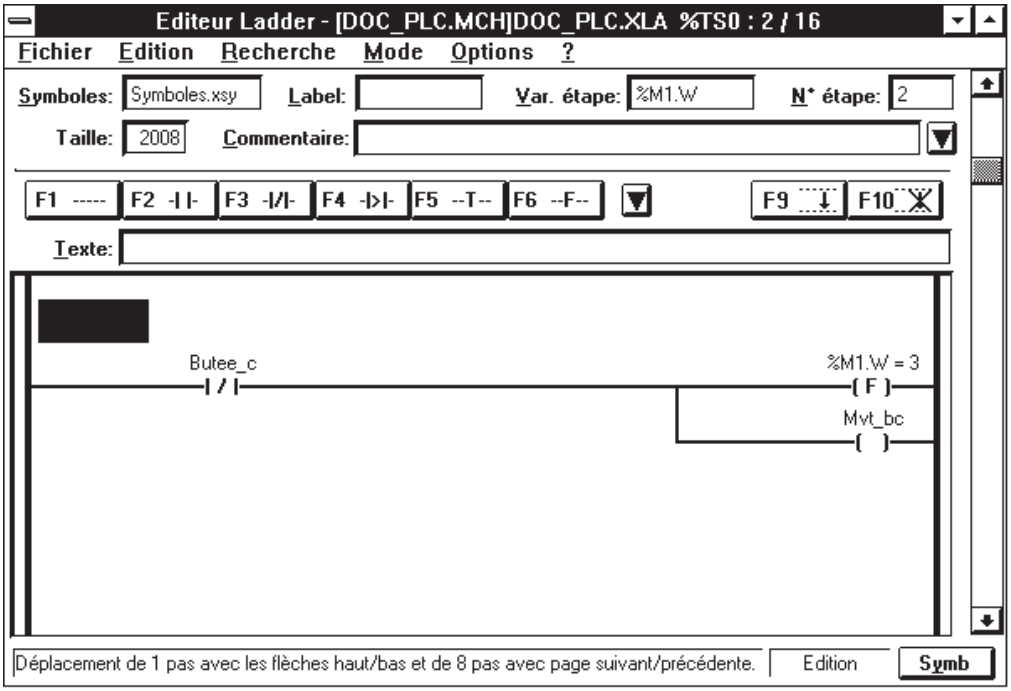
5.1.2.4 Exemples de programmation

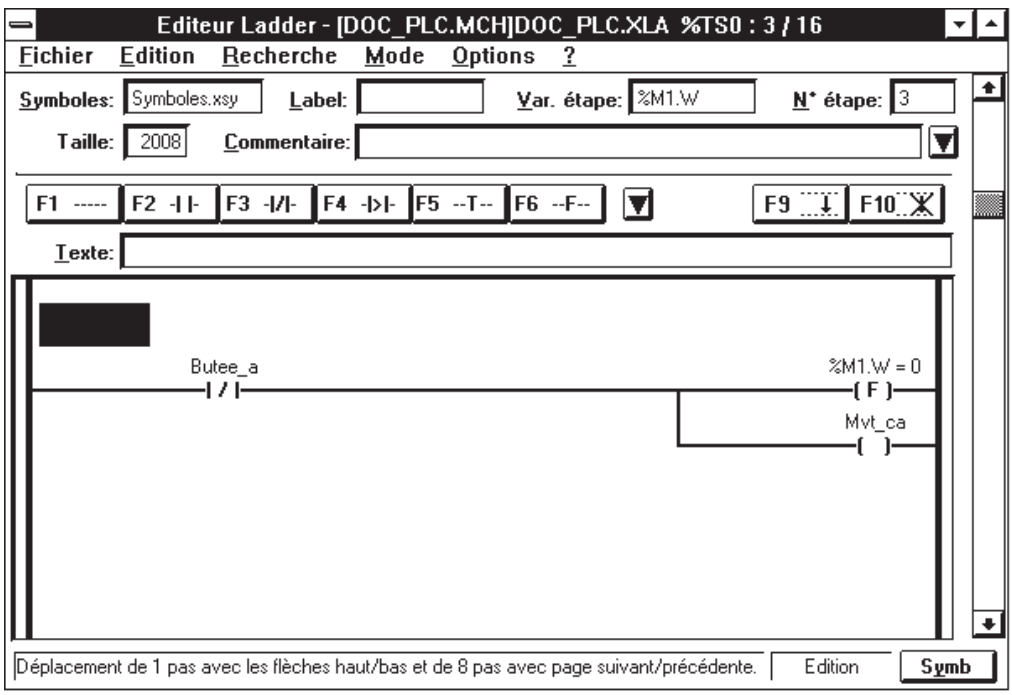
Spécification selon la méthodologie grafcet




Editeur Ladder - [DOC_PLC.MCH]DOC_PLC.XLA %TS0 : 0 / 16
Fichier Edition Recherche Mode Options ?
Symboles: Symboles.xsy
Label:
Var. étape: %M1.W
N° étape: 0
Taille: 2008
Commentaire:
F1 ----- F2 -| |- F3 -|/|- F4 -|>|- F5 --T-- F6 --F-- F9 F10
Texte:
Securite
Dcy
%M1.W = 1
(T)
Déplacement de 1 pas avec les flèches haut/bas et de 8 pas avec page suivant/précédente.
Edition
Symb


Editeur Ladder - [DOC_PLC.MCH]DOC_PLC.XLA %TS0 : 1 / 16
Fichier Edition Recherche Mode Options ?
Symboles: Symboles.xsy
Label:
Var. étape: %M1.W
N° étape: 1
Taille: 2008
Commentaire:
F1 ----- F2 -| |- F3 -|/|- F4 -|>|- F5 --T-- F6 --F-- F9 F10
Texte:
Butee_b
%M1.W = 2
(F)
Mvt_ab
( )
Déplacement de 1 pas avec les flèches haut/bas et de 8 pas avec page suivant/précédente.
Edition
Symb


Editeur Ladder - [DOC_PLC.MCH]DOC_PLC.XLA %TS0 : 2 / 16
Fichier Edition Recherche Mode Options ?
Symboles: Symboles.xsy
Label:
Var. étape: %M1.W
N° étape: 2
Taille: 2008
Commentaire:
F1 ----- F2 -| |- F3 -|/|- F4 -|>|- F5 --T-- F6 --F-- F9 F10
Texte:
Butee_c
%M1.W = 3
(F)
Mvt_bc
( )
Déplacement de 1 pas avec les flèches haut/bas et de 8 pas avec page suivant/précédente.
Edition
Symb


Editeur Ladder - [DOC_PLC.MCH]DOC_PLC.XLA %TS0 : 3 / 16
Fichier Edition Recherche Mode Options ?
Symboles: Symboles.xsy
Label:
Var. étape: %M1.W
N° étape: 3
Taille: 2008
Commentaire:
F1 ----- F2 -| |- F3 -|/|- F4 -|>|- F5 --T-- F6 --F-- F9 F10
Texte:
Butee_a
%M1.W = 0
(F)
Mvt_ca
( )
Déplacement de 1 pas avec les flèches haut/bas et de 8 pas avec page suivant/précédente.
Edition
Symb

5.2 La séquence réseau

5.2.1 Présentation

Un réseau de contacts est composé :
- d'un label facultatif et d'un commentaire facultatif,
- d'une étape grafcet facultative,
- d'une zone de test,
- d'une zone d'action.

5.2.2 Structure de la zone de test

5.2.2.1 Présentation

La zone test occupe la partie gauche du réseau.

La zone test permet de saisir des équations logiques.

Une équation logique est une combinaison de contacts en parallèle ou en série.

Un contact est un booléen résultat :
- du test d'une ou plusieurs variables sur bit,
- du test du front montant ou descendant du fil d'entrée,
- d'une comparaison entre deux expressions numériques,

La zone test est composée de 6 fils sur chacun desquels on peut brancher 6 contacts.

L'état du fil à la sortie d'un contact dépend de l'état du fil à l'entrée du contact et du résultat du test.

Si ce résultat est FAUX alors le fil correspondant est mis à ZERO. Sinon l'état du fil n'est pas changé.

Des actions conditionnelles sont autorisées dans la zone test. Ces actions sont conditionnées par l'état du fil d'entrée et ne modifie pas le fil de sortie.

Il est possible de mettre des fils en dérivation.

Une dérivation est symbolisée par une barre verticale.

5.2.2.2 Les contacts

Il y a cinq types de contacts :

Type de contact	Description
<variable bit> {,<variable bit>}7 —] [—	Testent l'état à UN d'une liste de variable sur bit. SI tous les bits sont à UN alors fil de sortie == fil d'entrée. SINON mise à ZERO du fil de sortie.
<variable bit> {,<variable bit>}7 —] / [—	Testent l'état à ZERO d'une liste de variable sur bit. SI tous les bits sont à ZERO alors fil de sortie == fil d'entrée. SINON mise à ZERO du fil de sortie.
<variable bit> — R_T —	Détecte le front montant du fil d'entrée (RISING TRIG) Permet de mémoriser l'état du fil d'entrée. SI le fil d'entrée est à UN et <variable bit> à ZERO alors mise à UN du fil de sortie. SINON mise à ZERO du fil de sortie. <variable bit> = fil d'entrée (Mémorisation du fil d'entrée).

Type de contact	Description
<variable bit> — F_T —	Détecte le front descendant du fil d'entrée (FALLING TRIG) Permet de mémoriser l'état du fil d'entrée. SI le fil d'entrée est à ZERO et <variable bit> à UN alors mise à UN du fil de sortie. SINON mise à ZERO du fil de sortie. <variable bit> = fil d'entrée (Mémorisation du fil d'entrée).
<comparaison_numérique> —[>]—	Permet la comparaison de deux expressions numériques. SI la comparaison numérique est VRAI alors fil de sortie == fil d'entrée. SINON mise à ZERO du fil de sortie.

Fonctionnement de la cellule R_T (Rising trig)

Fil entrée —— Mémoire R_T —— Fil sortie

Si (Entrée == 1 et Mémoire == 0) alors Sortie = 1

Sinon Sortie = 0

Mémoire = Entrée

Fonctionnement de la cellule F_T (Falling trig)

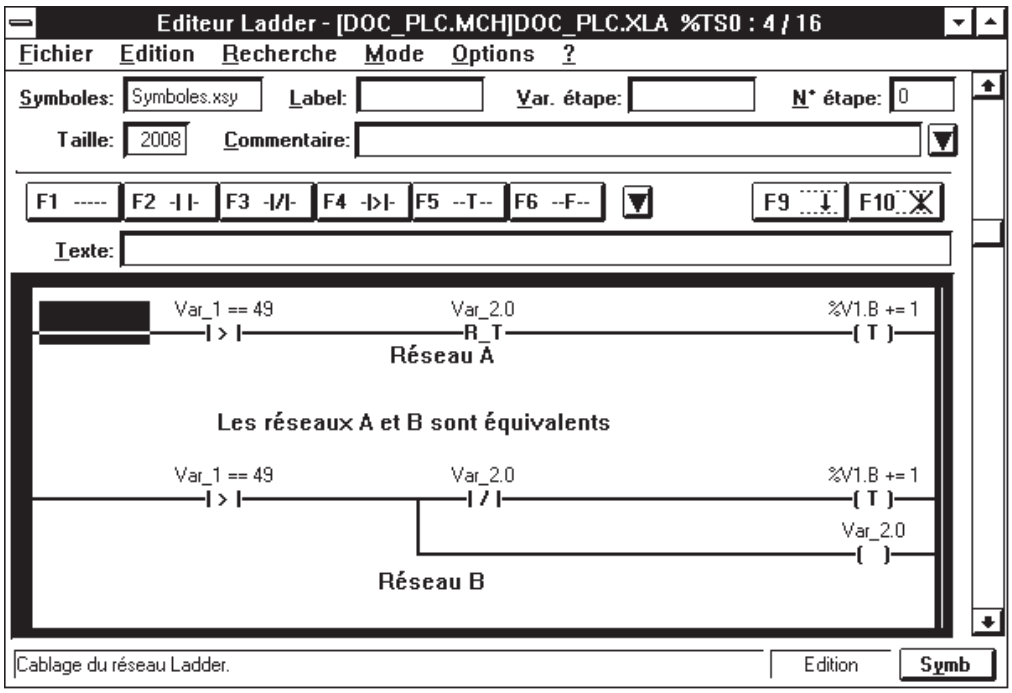
Fil entrée —— Mémoire F_T —— Fil sortie

Si (Entrée == 0 et Mémoire == 1) alors Sortie = 1

Sinon Sortie = 0

Mémoire = Entrée

Exemple

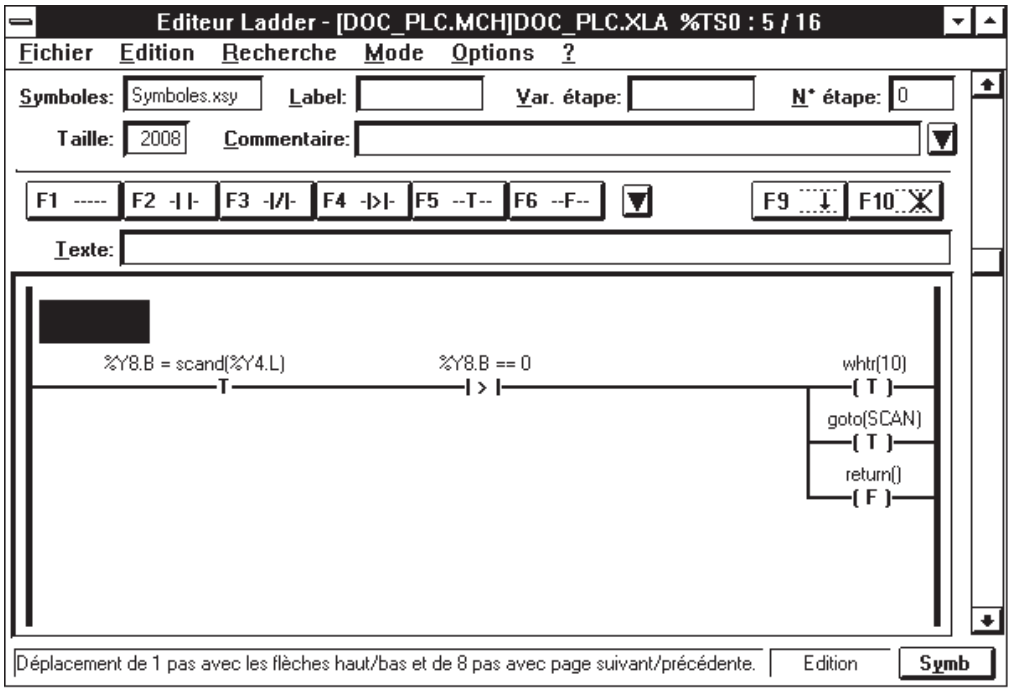


5.2.2.3 Les actions conditionnelles dans la zone test

Il y a deux types d'actions :

Type d'action	Description
<affectation_numérique> {;<affectation_numérique>}7 <appel_fonction> — T —	Action exécutée si le fil d'entrée est à UN. Les actions possible sont : - <affectation_numérique> Ex :%M10.B = %V34+3 - <appel_fonction> ex :setb(%M100.&,0,100)
<affectation_numérique> {;<affectation_numérique>}7 <appel_fonction> — F —	Action exécutée si le fil d'entrée est à ZERO. Les actions possible sont : - <affectation_numérique> Ex :%M10.B = %V34+3 - <appel_fonction> ex :setb(%M100.&,0,100)

Exemple



5.2.2.4 Temporisations

Trois types de blocs fonction temporisation sont disponibles :
- les temporisations de déclenchement TOF_n,
- les temporisations d'enclenchement TON_n,
- les temporisations d'impulsions TP_n.

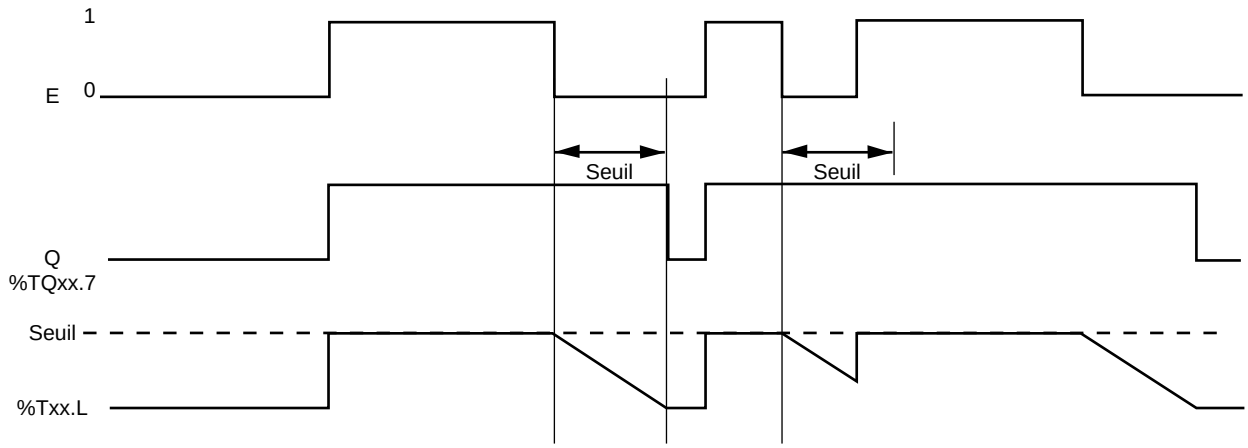
128 temporisations (Tous types confondus) sont disponibles.

Les variables %T0.L à %T7F.L contiennent la valeur courante de la temporisation en ms. Seule la taille .L est autorisée en programmation et en lecture par UNITE.

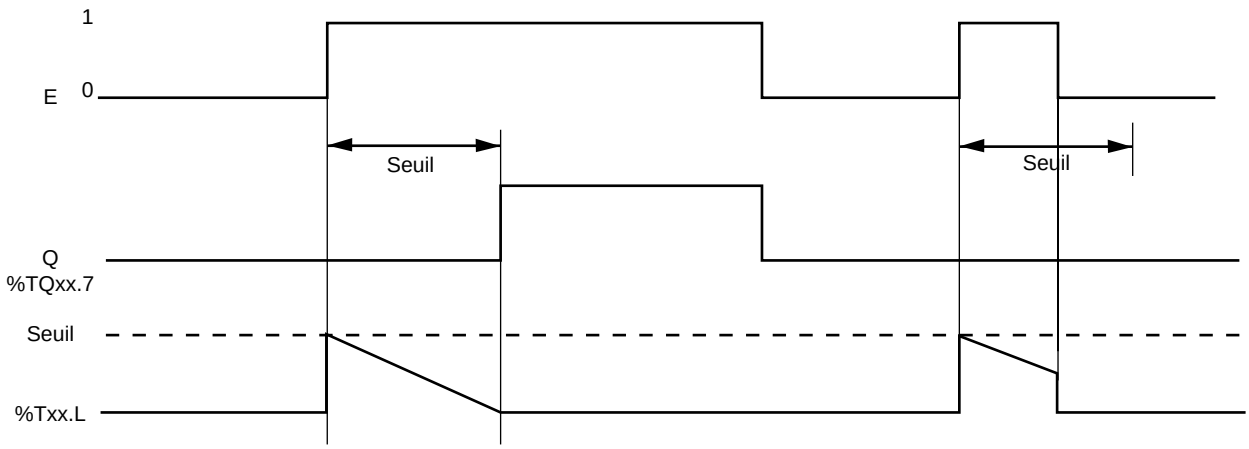
Les variables %TQ0.7 à %TQ7F.7 sont l'image de la sortie temporisation (Q). Seul le bit .7 est autorisé en programmation et en lecture par UNITE.

Type de temporisation	Description
TOF_n(<seuil>) E Q	Temporisation de déclenchement (Avec n de 00 à 7F) La mise à 1 de E positionne la sortie Q à 1 pour une durée indéterminé. La mise à 0 de E positionne la sortie Q à 0 après la temporisation. L'argument seuil est une expression numérique qui s'exprime en ms
TON_n(<seuil>) E Q	Temporisation d'enclenchement (Avec n de 00 à 7F) La mise à 1 de E positionne la sortie Q à 1 en fin de temporisation.Q retombe dès que E = 0. L'argument seuil est une expression numérique qui s'exprime en ms
TP_n(<seuil>) E Q	Temporisation d'impulsion (Avec n de 00 à 7F) La mise à 1 de E positionne la sortie Q à 1 pendant la temporisation.Q retombe après la temporisation. L'argument seuil est une expression numérique qui s'exprime en ms

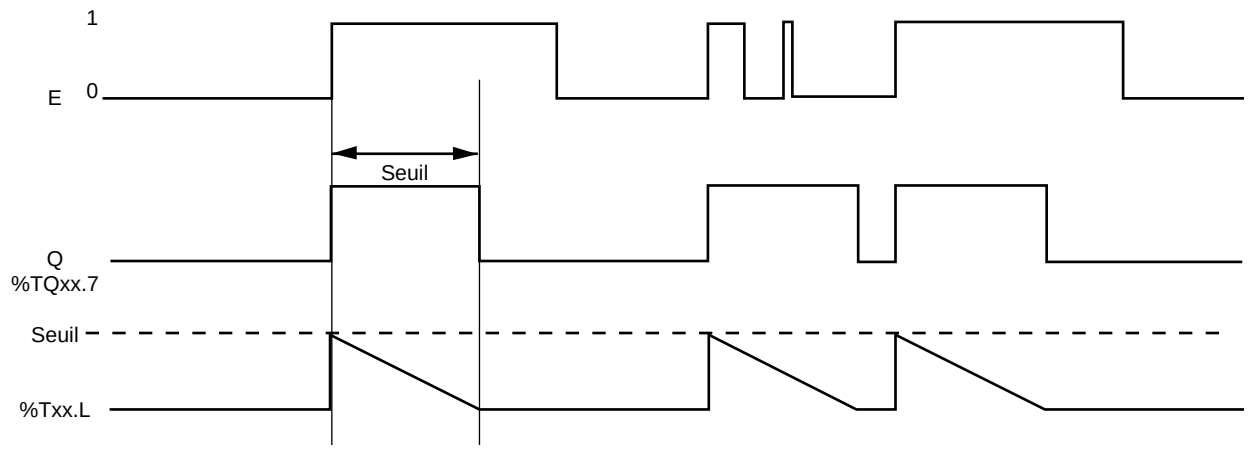
Temporisation de déclenchement «TOF_n»



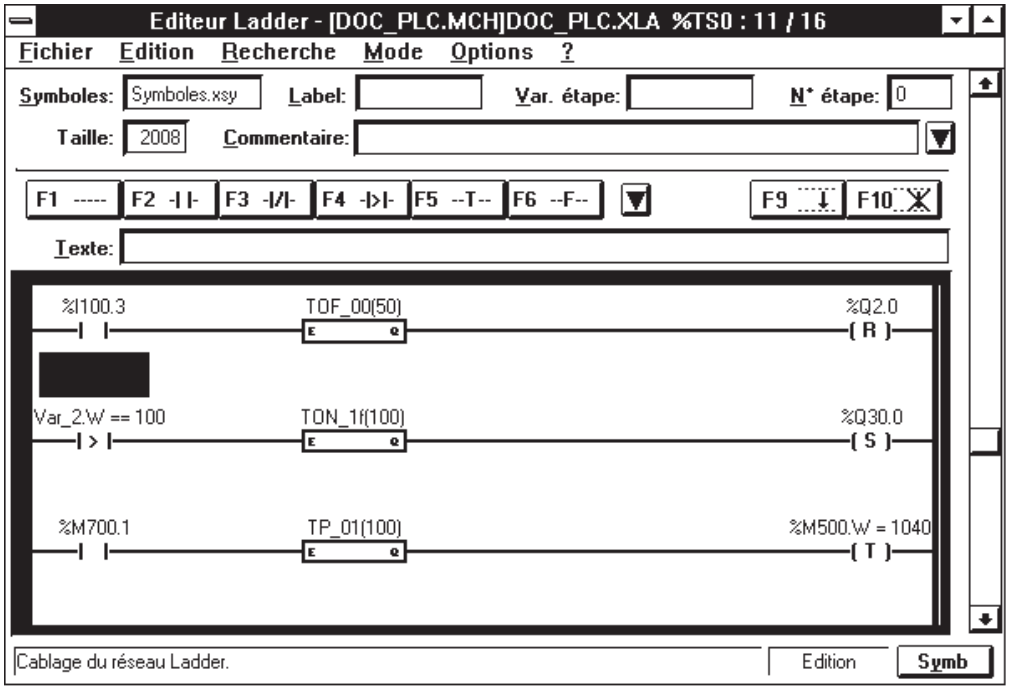
Temporisation d'enclenchement «TON_n»



Temporisation d'impulsion «TP_n»



Exemple



5.2.2.5 Compteurs/décompteurs

Deux types de blocs fonction compteur/décompteur sont disponibles :
- les compteurs CTU_n,
- les décompteur CTD_n.

128 compteurs/décompteurs sont disponibles.

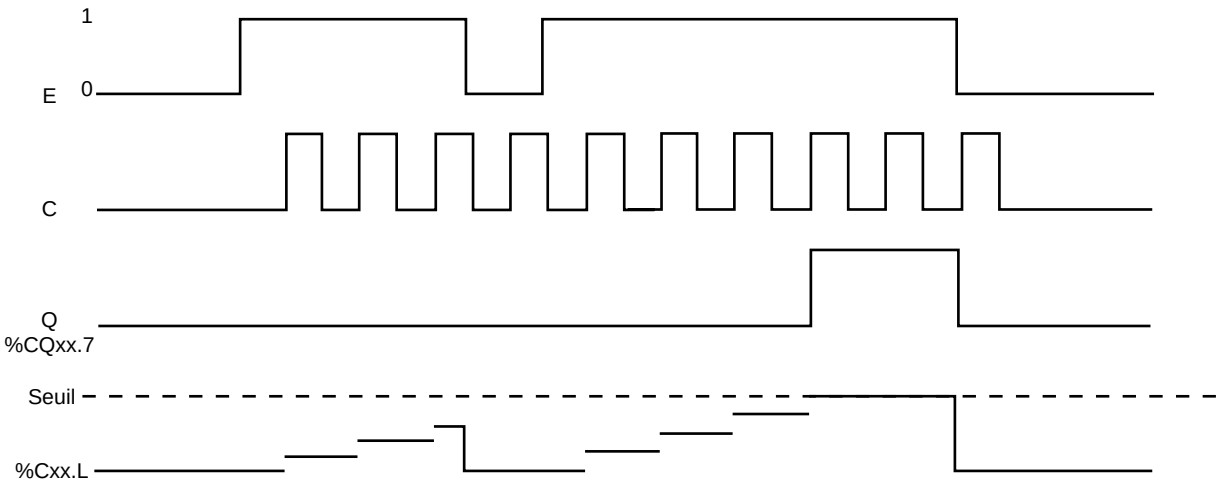
Les variables %C0.L à %C7F.L contiennent la valeur courante du compteur/décompteur. Seule la taille .L est autorisée en programmation et en lecture par UNITE.

Les variables %CQ0.7 à %CQ7F.7 sont l'image de la sortie compteur/décompteur (Q). Seul le bit .7 est autorisé en programmation et en lecture par UNITE.

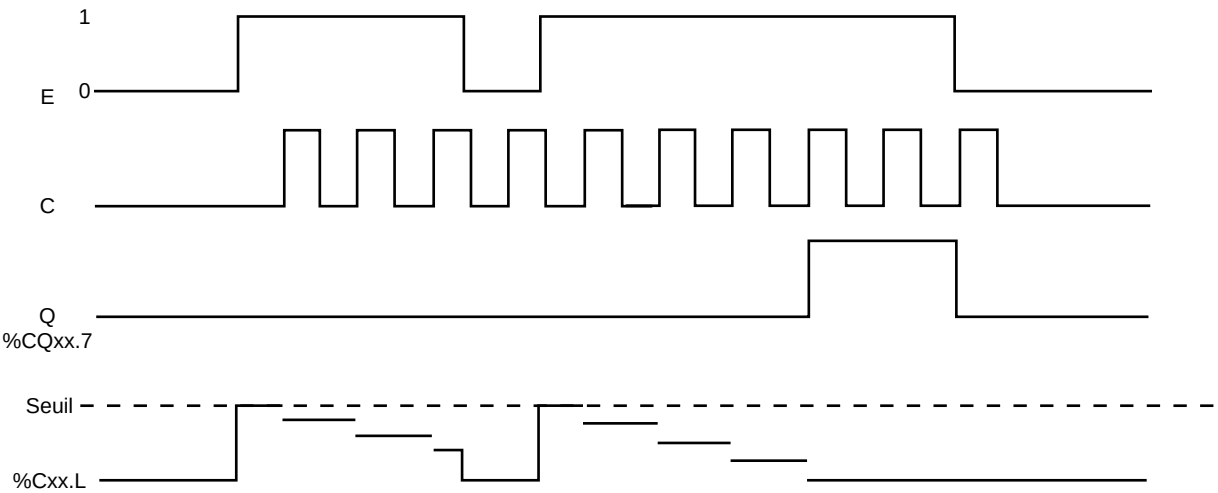
REMARQUE: Les compteurs/décompteurs sont réinitialisés uniquement sur une RAZ des variables sauvegardées.

Type de compteur/décompteur	Description
CTU_n(<seuil>) E Q C	Compteur (Avec n de 00 à 7F) La mise à 1 de E positionne la sortie Q à 1 dès que le seuil est atteint. La mise à 0 de E positionne la sortie Q à 0. C définit les éléments à compter. L'argument seuil est une expression numérique.
CTD_n(<seuil>) E Q C	Décompteur (Avec n de 00 à 7F) La mise à 1 de E positionne la sortie Q à 1 dès que le seuil est atteint. La mise à 0 de E positionne la sortie Q à 0. C définit les éléments à décompter. L'argument seuil est une expression numérique.

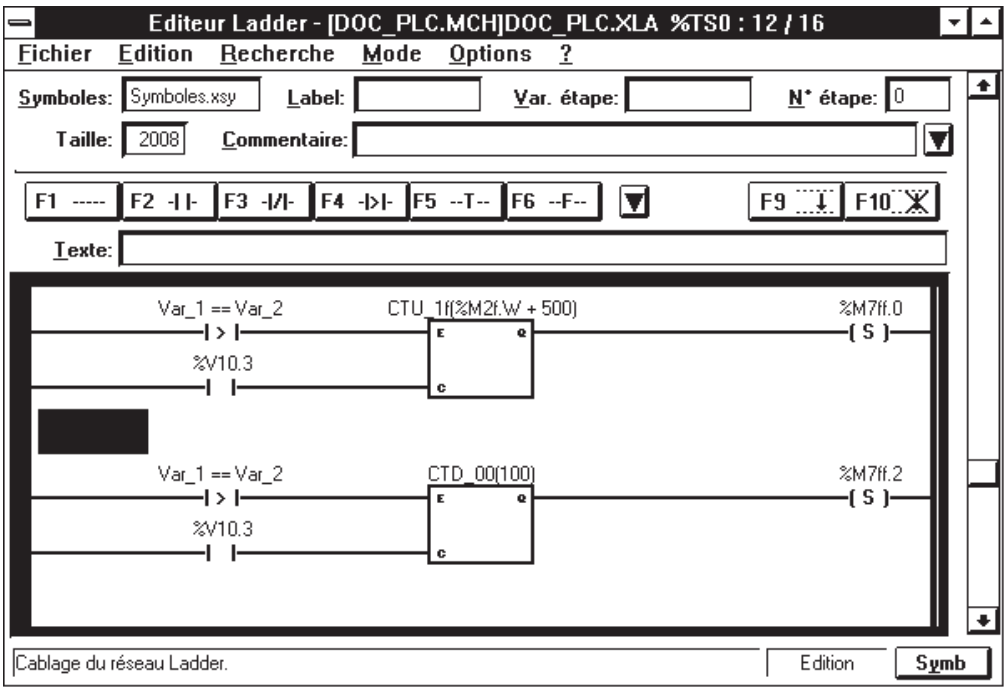
Compteurs



Décompteurs



Exemple



5.2.2.6 Les dérivations

Il est possible de mettre en dérivation des fils adjacents.

Une dérivation est matérialisée par une barre verticale après un contact ou un fil.

5.2.2.7 Exécution d'une zone test

A l'intérieur de la zone test la scrutation se fait de haut en bas et de gauche à droite.

Sur un fil la propagation du potentiel se fait de gauche à droite mais jamais de droite à gauche (contrairement à un schéma électrique où la propagation se fait dans les deux sens).

Sur une dérivation la propagation se fait de bas en haut et de haut en bas.

5.2.3 Structure de la zone action

5.2.3.1 Présentation

La zone action est située à droite du réseau.

Elle permet le déclenchement conditionnel d' actions suivants les résultats logiques de la zone test.

Six actions conditionnées par les six fils de la zone test peuvent être déclenchées dans une séquence.

Il y a six types d'actions possibles.

Les actions

Les six fils permettent de déclencher six actions parmi les suivantes:

Type de d'action	Description
<variable_bit> —( )—	Positionnement du bit à l'état logique du fil.
<variable_bit> —(/)—	Positionnement du bit à l'état logique inverse du fil.
<variable_bit> —(S)—	SI fil VRAI alors mise à un du bit. SINON passage action suivante.
<variable_bit> —(R)—	SI fil VRAI alors mise à zéro du bit. SINON passage action suivante.
<affectation_numérique> {;<affectation_numérique>}7 <appel_fonction> goto(<label>) call(<label>) return(...) —(T)—	SI fil VRAI alors exécution : - d'une ou plusieurs affectations numériques ex: %M10.B = %V34+3 - d'un appel fonction ex: setb (%M100.&,0,100) - d'un saut à un label interne au module ex: goto (FIN) - d'un appel à un label interne au module ex: call (COPIE) - d'un «return» au module ou au call appelant ex: return (%M10.B) SINON passage action suivante.
<affectation_numérique> {;<affectation_numérique>}7 <appel_fonction> goto(<label>) call(<label>) return(...) —(F)—	SI fil FAUX alors exécution: - d'une ou plusieurs affectations numériques ex: %M10.B = %V34+3 - d'un appel fonction ex: setb (%M100.&,0,100) - d'un saut à un label interne au module ex: goto (FIN) - d'un appel à un label interne au module ex: call (COPIE) - d'un «return» au module ou au call appelant ex: return (%M10.B) SINON passage action suivante.

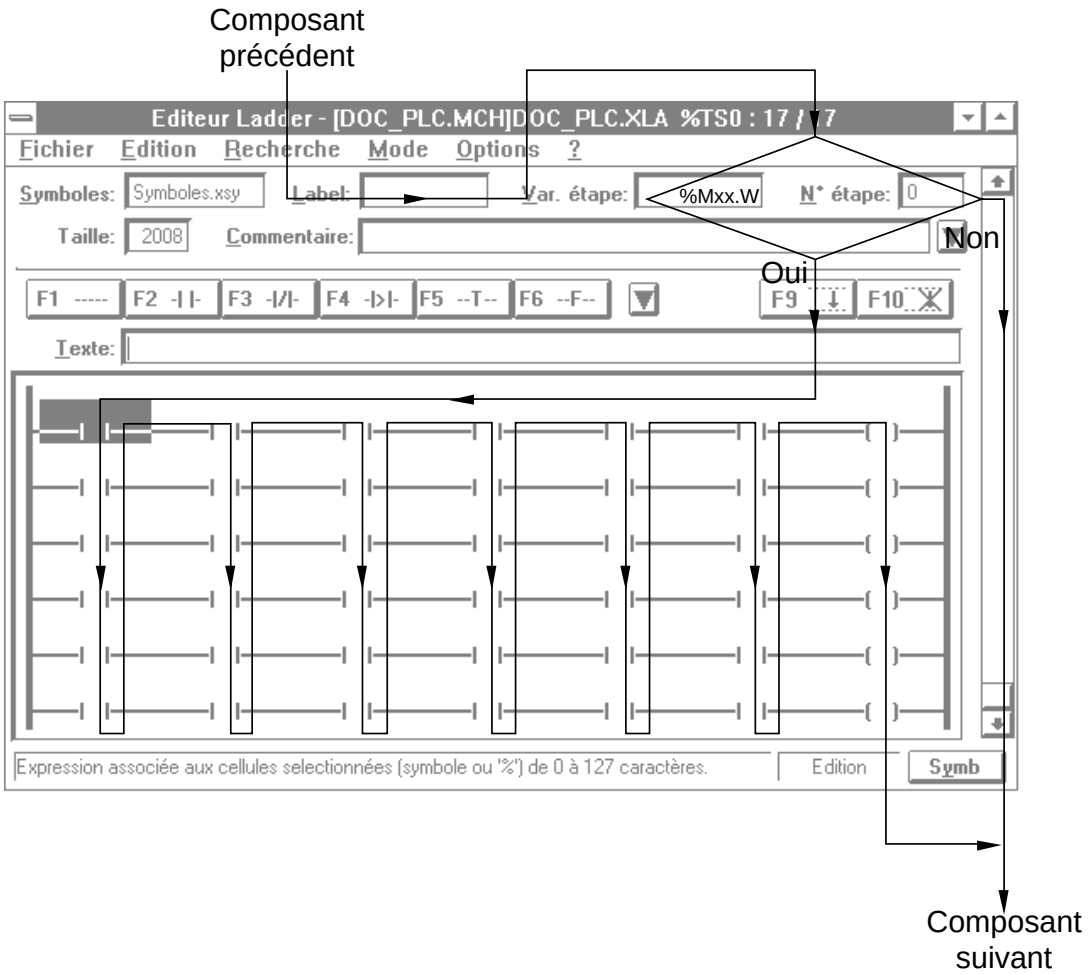
REMARQUE : l'appel à un sous programme externe au module (ex %SP30) se fait par la fonction sp(.....).

5.2.4 Exécution d'une zone action

L'exécution de la zone action se fait après l'exécution de la zone test et de haut en bas (du fil 0 au fil 5).

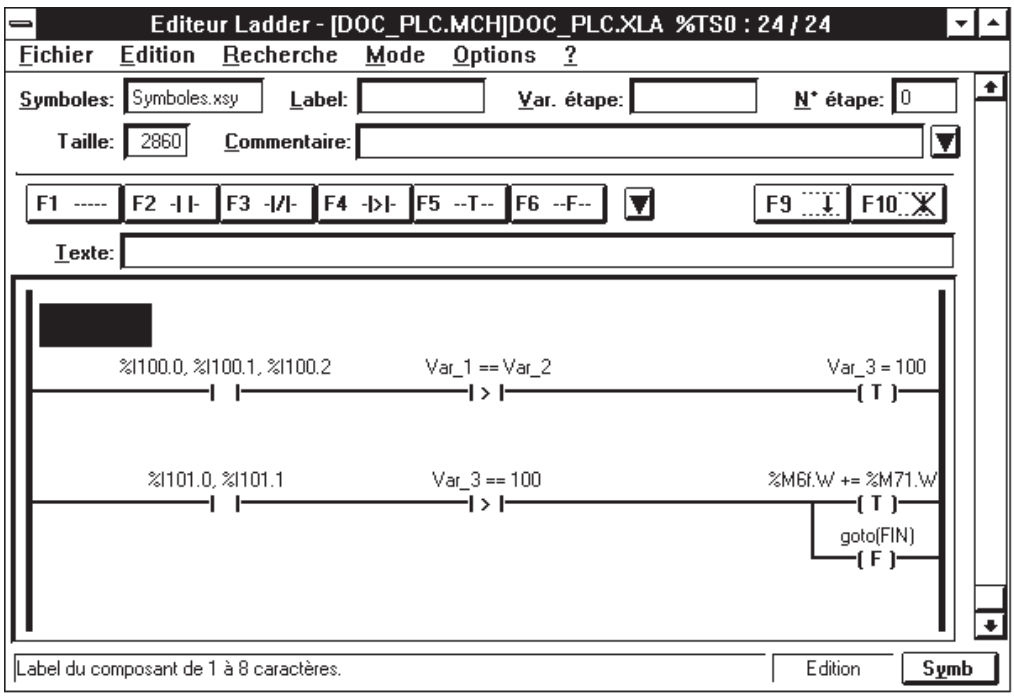
> **⚠ ATTENTION**
>
> Une action est toujours exécutée après la zone test. Le changement d'état d'une variable, dans la zone action, ne sera vu que dans la séquence suivante.

Ordre de scrutation d'un réseau LADDER



Piège lié à la scrutation

Dans l'exemple ci-dessous, le système effectue la lecture de la comparaison numérique «Var_3 == 100» avant l'écriture de «Var_3 = 100» en zone action et si les conditions de la première ligne de contacts sont réalisées. Il y aura donc un décalage d'un cycle automate entre l'écriture de «Var_3 = 100» et l'éventuelle réalisation de la deuxième ligne de contacts.



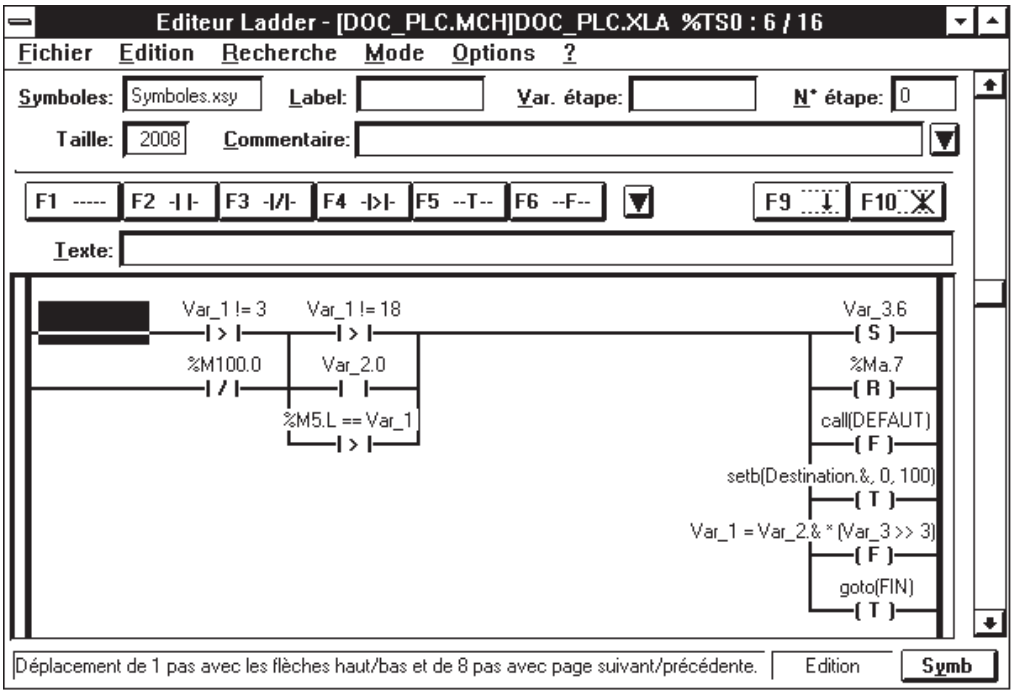
Il est donc important de contrôler que l'ordre de scrutation n'a pas d'incidence sur le déroulement d'un programme dont les traitements doivent être exécutés sur le même cycle automate.

5.2.5 Règle de construction d'un réseau

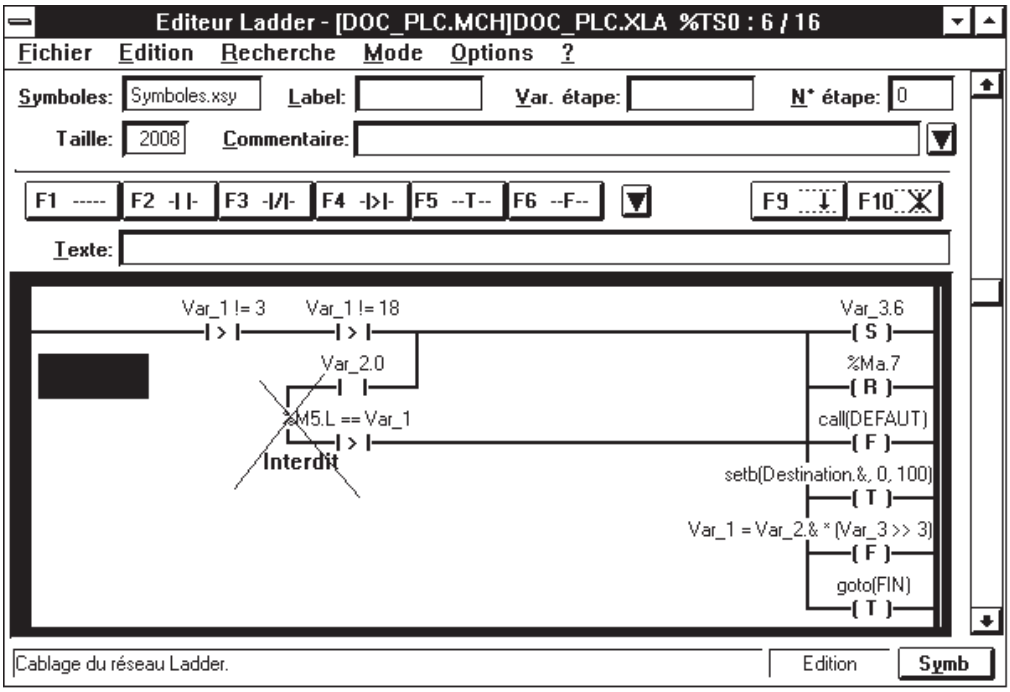
Pour être valide un réseau doit respecter les règles suivantes :
- la zone test d'un réseau ne doit pas être vide,
- un contact ou un fil doit être relié coté gauche et coté droit par un contact, un fil ou une dérivation,
- une dérivation ou un ensemble de dérivation contiguë doit être reliée en haut et en bas à au moins un contact ou un fil. De plus elle doit être connectée à au moins une alimentation en courant, c'est à dire un contact ou un fil venant de sa gauche et à au moins une sortie de courant c'est à dire un contact ou un fil partant vers la droite,
- la zone action d'un réseau ne doit pas être vide,
- une bobine doit être connectée sur sa gauche par un contact, un fil ou une dérivation,

5.2.6 Exemple de séquences réseau

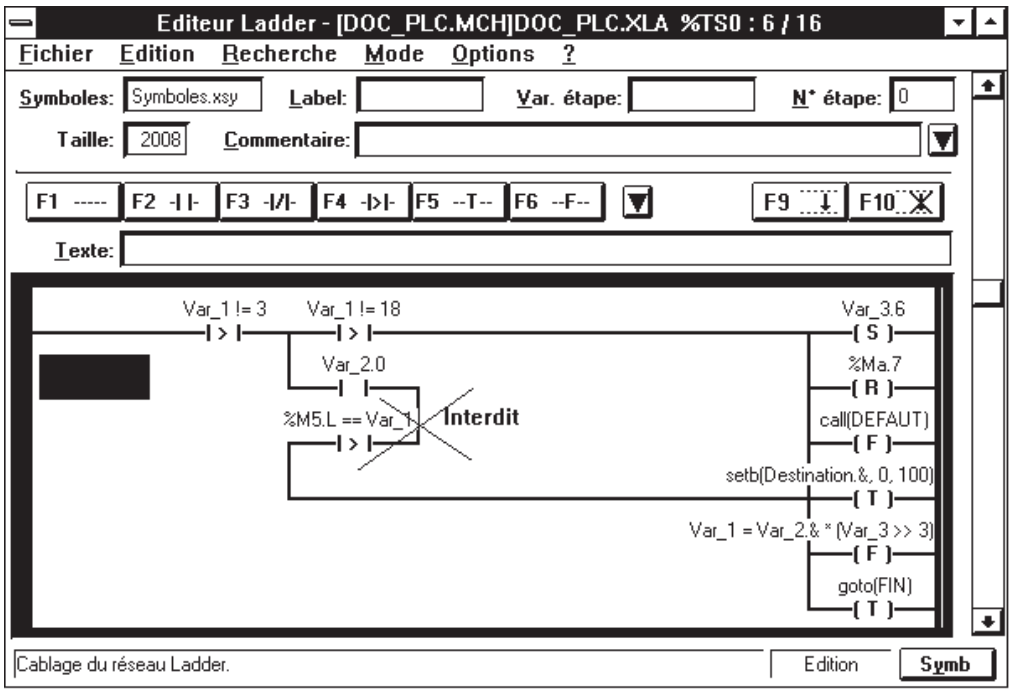
Réseau valide



Réseau invalide - Dérivation sans alimentation

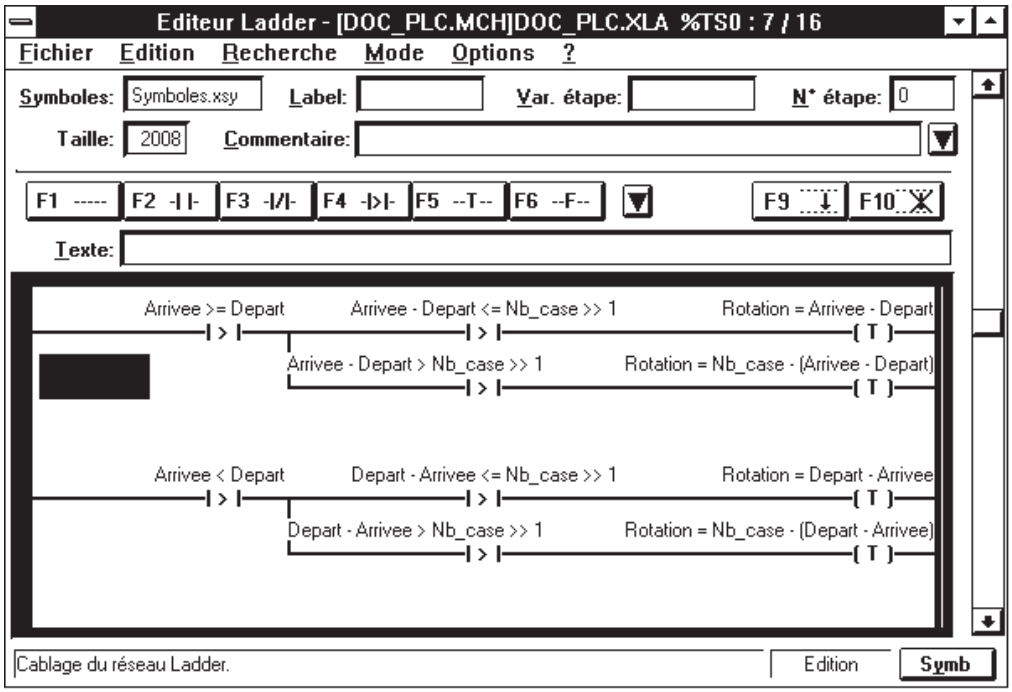


Réseau invalide - Dérivation sans sortie



5

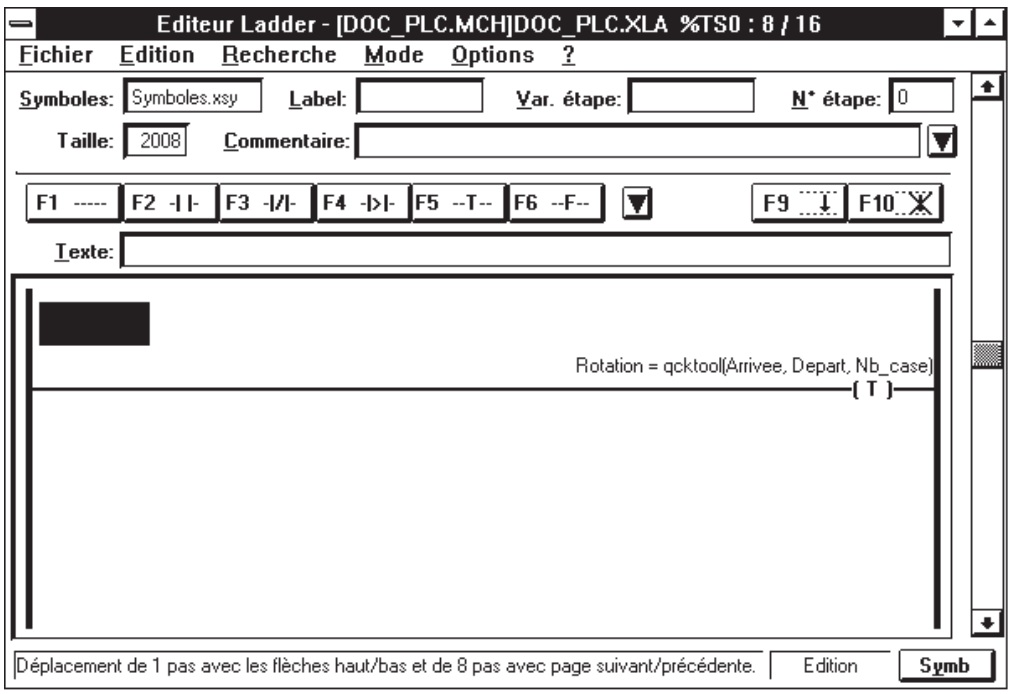
Recherche d'outils dans un magasin



La séquence ci-dessus détermine le sens de rotation et le nombre de pas pour aller chercher l'outil à l'emplacement «Arrivée» en partant de l'emplacement «Départ» dans un magasin d'outils rotatif comportant un nombre d'emplacement égal à «Nb_case».

La valeur absolue de «Rotation» indique le nombre de pas de la rotation, le signe de «Rotation» indique le sens.

La séquence ci-dessous utilise la fonction qcktool() pour résoudre ce problème.

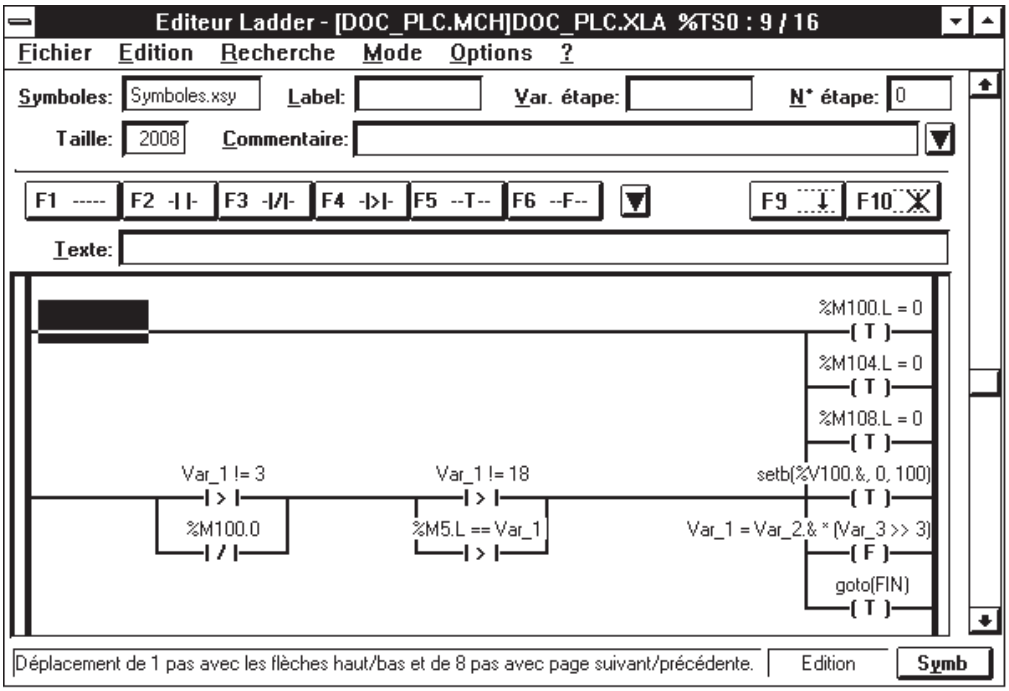


5.2.7 Conseils de programmation

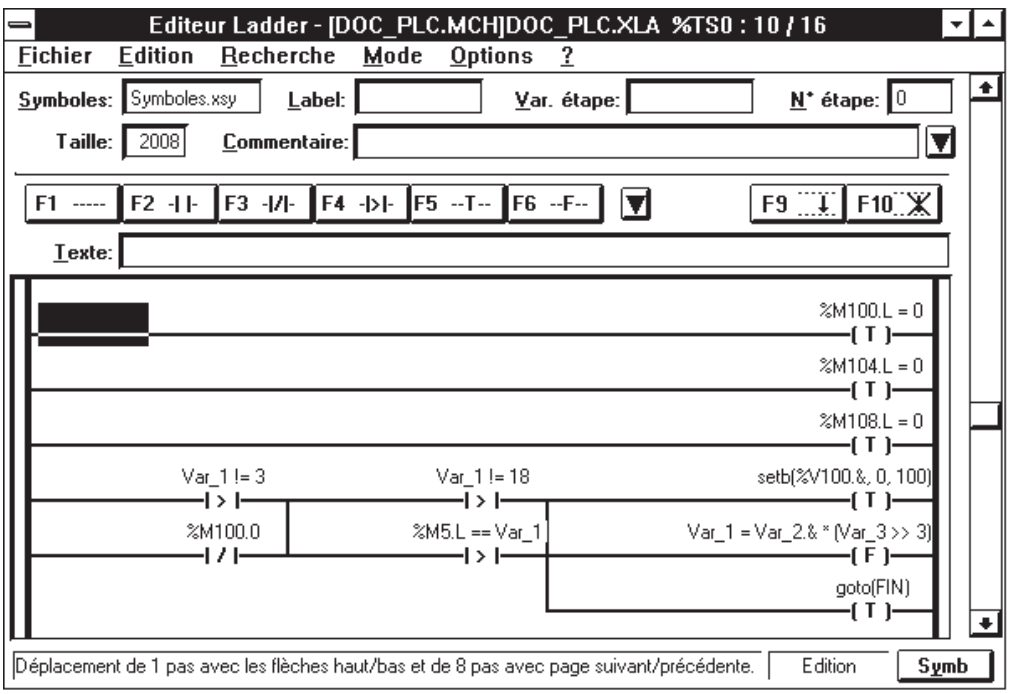
5.2.7.1 Optimisation des réseaux

Pour obtenir un réseau optimisé en taille de code et en vitesse il faut minimiser :
- le nombre de contacts,
- le nombre de dérivations (Barres verticales).

Réseau non optimisé



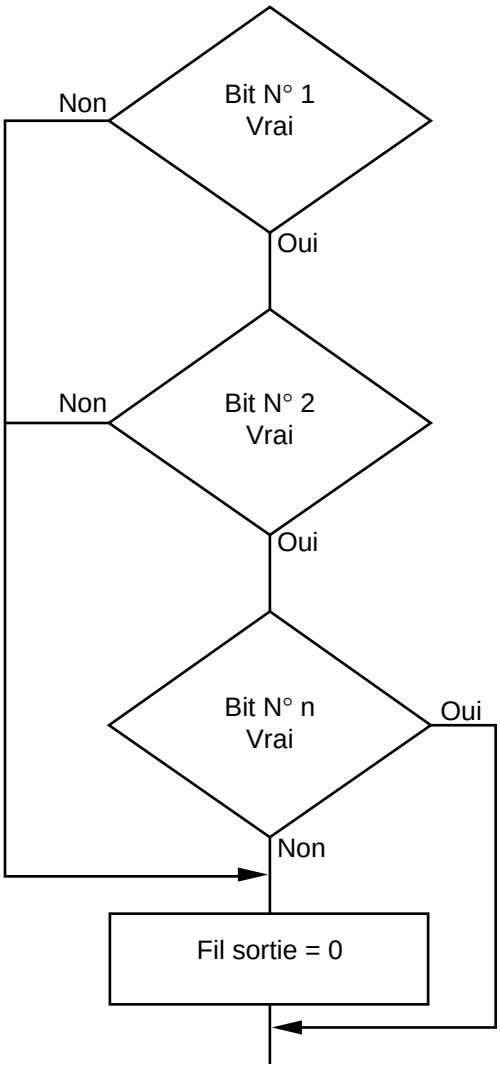
Réseau optimisé - 48 octets de moins que le réseau non optimisé



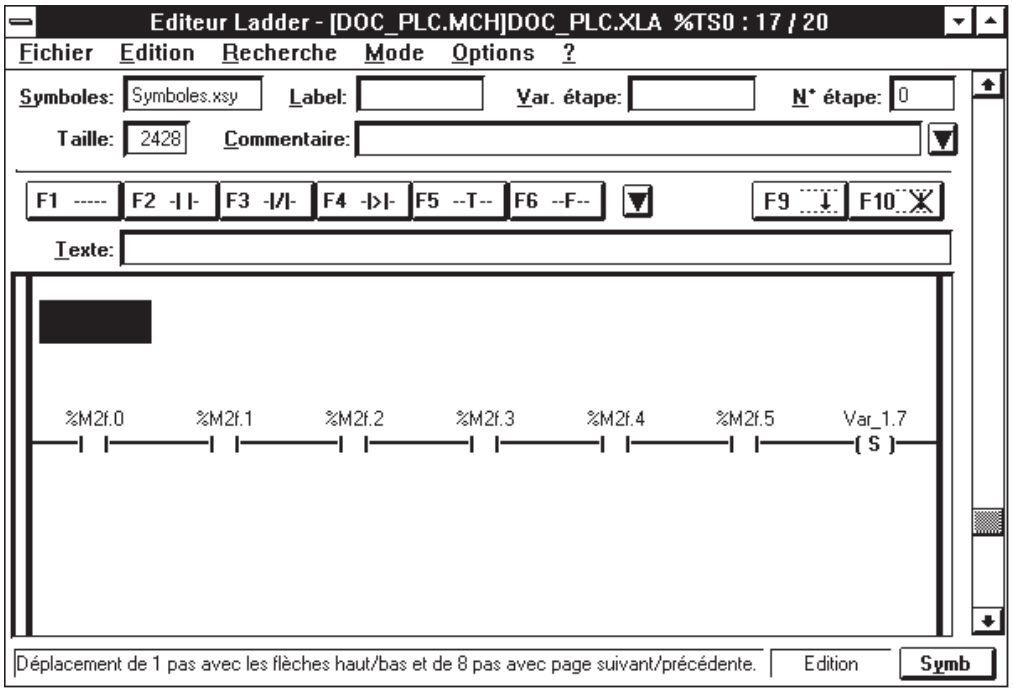
5.2.7.2 Liste de bits en zone test

Les listes de bits permettent d'optimiser les réseaux en taille et en vitesse.

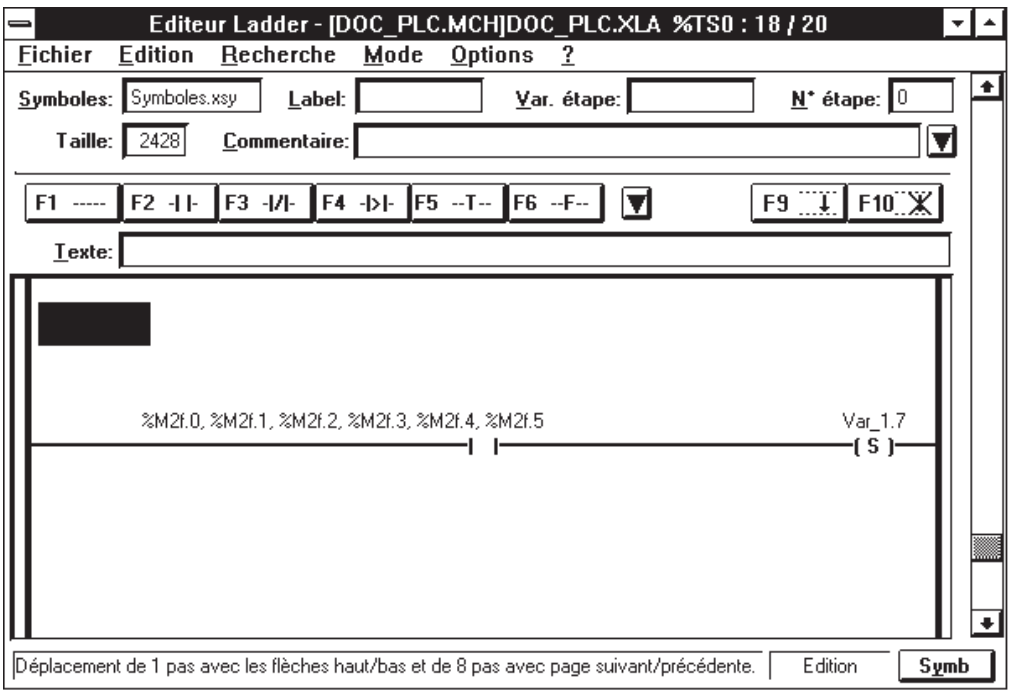
L'organigrame ci-dessous donne le principe de traitement d'une liste de bits par le système. Dès qu'un bit n'est pas vrai, le système saute les tests des bits suivants.



Réseau non optimisé



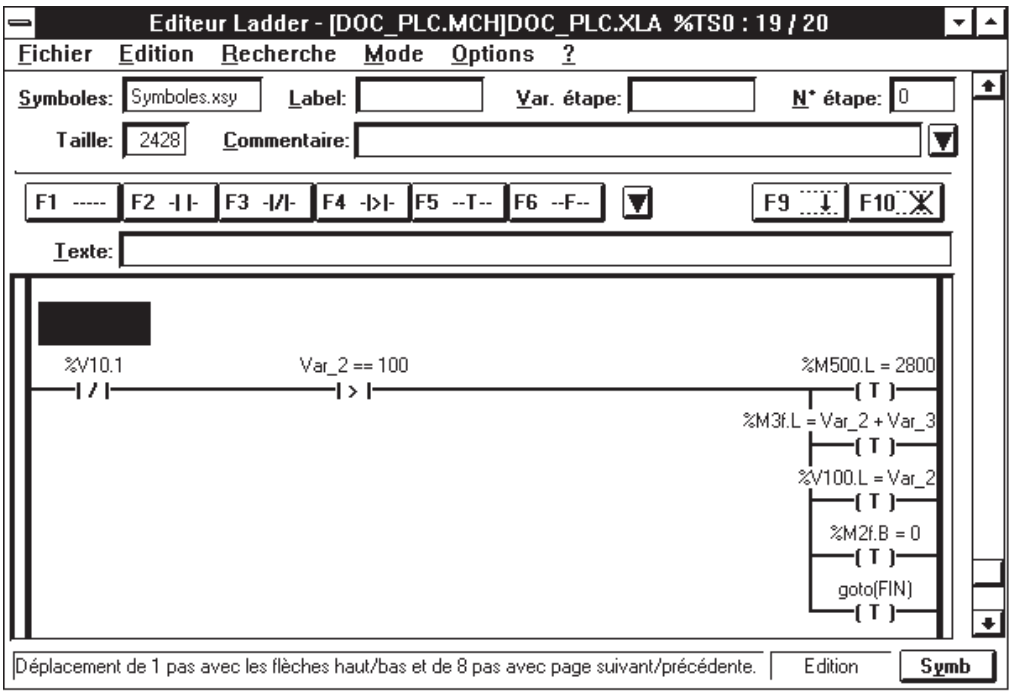
Réseau optimisé - 20 octets de moins que le réseau non optimisé



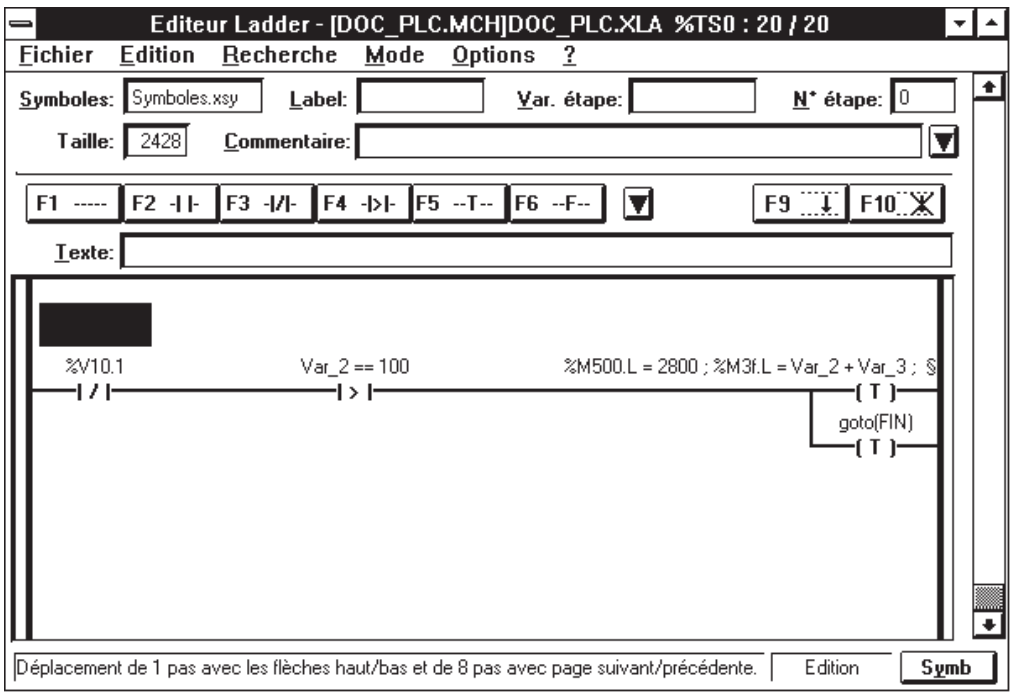
5.2.7.3 Affectations numériques multiples

Les affectations numériques multiples permettent d'optimiser les réseaux en taille et en vitesse.

Réseau non optimisé



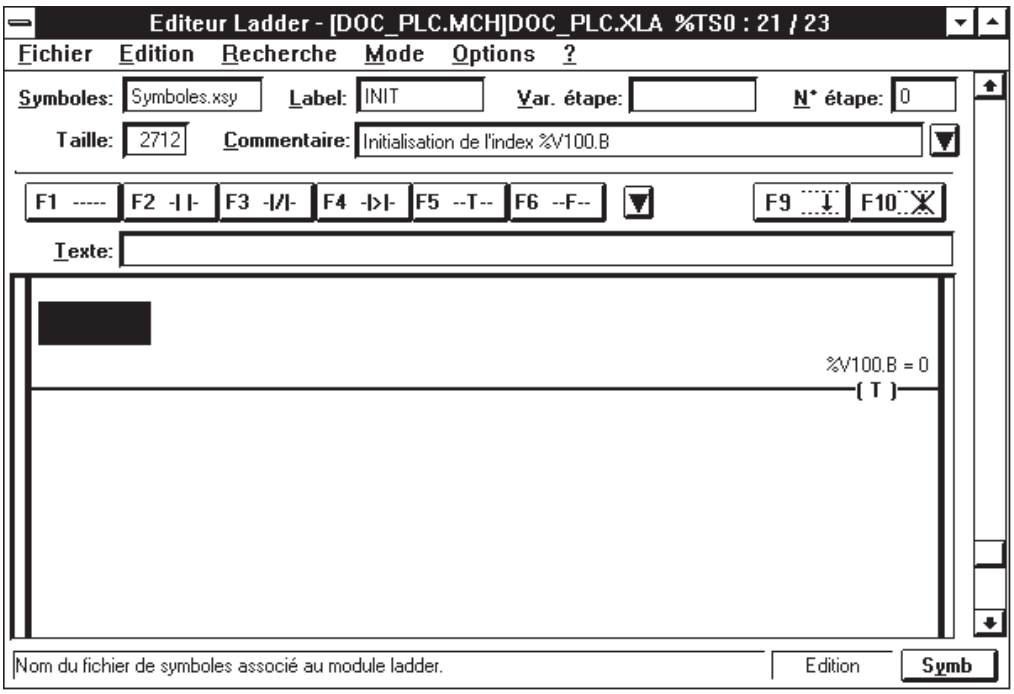
Réseau optimisé - 12 octets de moins que le réseau non optimisé



5.2.7.4 Test des bits d'un octet, mot ou long mot

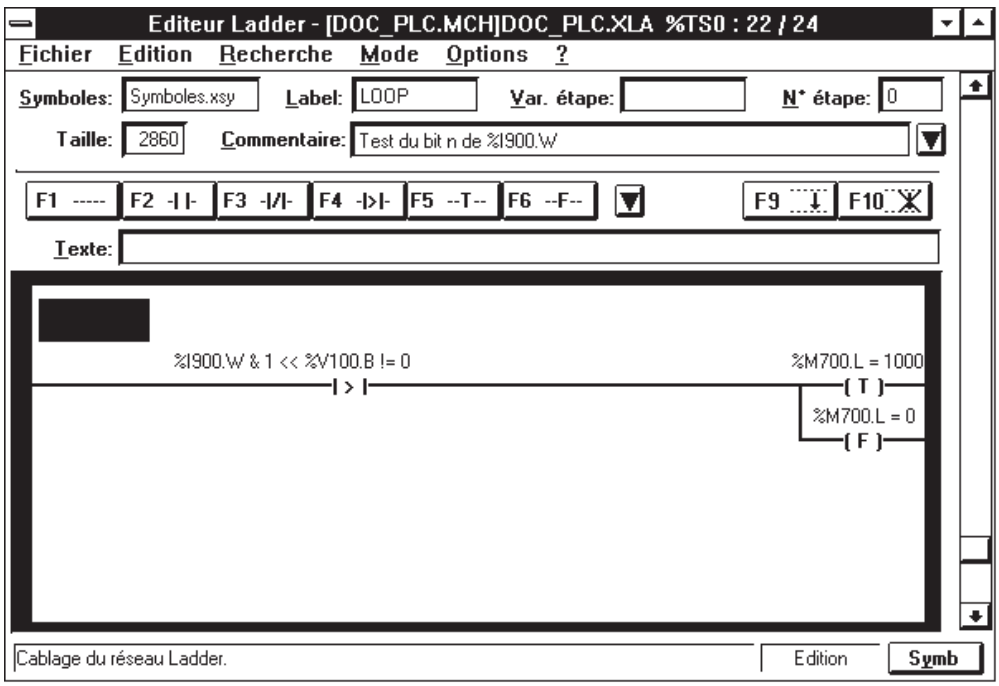
Ces séquences permettent de tester tous les bits de la variable %I900.W.

Séquence 1 - Initialisation de l'index %V100.B

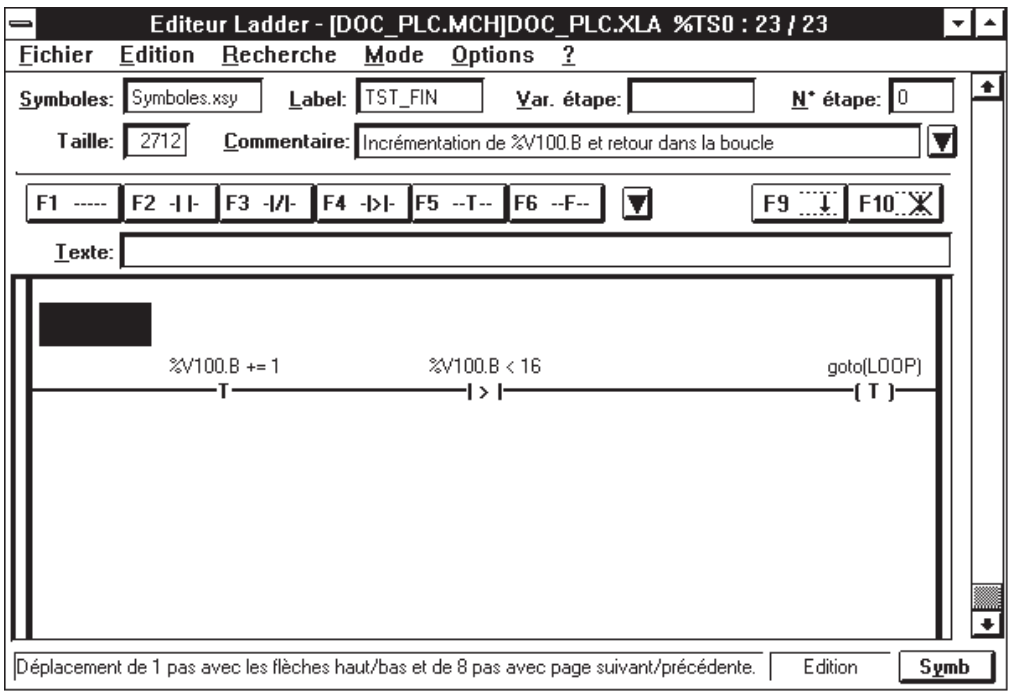


Séquence 2 - Test de chaque bit de %I900.W

Décalage arithmétique de la valeur 1 vers la gauche d'une valeur contenue dans %V100.B puis ET logique avec %I900.W (Si %V100.B == 0, test du bit 0 de %I900.W, si %V100.B == 1, test du bit 1 de %I900.W, .. etc ..).Résultat différent de 0. La bobine TRUE est activée si le bit testé est égal à 1. La bobine FASE est activée si le bit testé est égal à 0.



Séquence 3 - Incrémentation de l'index %V100.B et retour dans la boucle si %V100.B < 16



5.3 Appel d'une fonction

Le langage ladder permet l'appel de fonctions.

La syntaxe est la suivante :

[<variable_numérique> <opérateur_affectation>] <nom_fonction> ([<expression_numérique>] { ,<expression_numérique>}6)

L'affectation numérique à gauche du nom de la fonction est facultative. Elle permet de récupérer le code retourné par la fonction lorsque le programmeur le juge utile.

Exemples d'appels de fonctions :

%M100.L = atoi(%M20.L) // %M100.L reçoit le résultat de la conversion

bit(%M20.&, M30.&, 8); // Ici le code retourné est perdu

cpyb(%V100.&, %V100.& + %M10.B, %V110.W + 10) // Ici le code retourné est perdu

5.4 Contrôle des paramètres

Le nombre de paramètres passés est contrôlé à la compilation.

La valeur des paramètres passés ne peut être contrôlée à la compilation. Le moniteur effectue certains contrôles lors de l'appel de la fonction, avant son exécution.

6 Fonctions d'usage général

6.1 Conversion d'une chaîne ASCII en entier signé sur 32 bits atoi

Syntaxe de l'instruction

atoi(&source)

&source : Adresse de la chaîne ASCII à convertir.

Retourne un entier signé sur 32 bits résultat de la conversion de la chaîne ASCII.

Fonctionnement

La fonction atoi() prend les chiffres décimaux en partant de la gauche.

Les blancs et les caractères de tabulation en tête sont ignorés.

Un signe éventuel (+ ou -) peut être placé , permettant d'obtenir un résultat signé.

La conversion s'arrête à la détection d'un octet NUL ou d'un caractère autre qu'un chiffre décimal.

En cas de débordement, la fonction atoi retourne la valeur positive maximum d'un entier signé sur 32 bits soit 0x7FFFFFFF.

Code retourné

Si OK

Entier signé sur 32 bits résultat de la conversion.

Si défaut

0x7FFFFFFF : La conversion a débordé du champ d'un entier signé sur 32 bits.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- fin de chaîne hors zone autorisée.

6.2 Conversion d’une chaîne ASCII en entier signé sur 32 bits **atoj**

Syntaxe de l’instruction

atoj(&&fin, &source)

&&fin : Adresse du long mot (%M ou %V) qui va recevoir l’adresse du caractère sur lequel s’est arrêté la conversion.

&source : Adresse de la chaîne ASCII à convertir.

Retourne un entier signé sur 32 bits résultat de la conversion de la chaîne ASCII.

Fonctionnement

La conversion s’arrête à la détection d’un octet NUL ou d’un caractère autre qu’un chiffre décimal.

Fonctionnement identique à la fonction atoi(). La fonction atoj() écrit dans le long mot à l’adresse &&fin l’adresse du caractère sur lequel s’est arrêtée la conversion ou zéro si fin de la chaîne atteinte.

En cas de débordement, atoj () retourne la valeur positive maximum d’un entier signé sur 32 bits soit 0x7FFFFFFF.

Le long mot d’adresse &&fin reçoit :
- 0 si la conversion s’est arrêtée sur un octet NUL de fin de chaîne,
- l’adresse du caractère (non NUL) sur lequel s’est arrêtée la conversion,
- -1 si débordement.

Code retourné

Si OK

Entier signé sur 32 bits résultat de la conversion.

Si défaut

0x7FFFFFFF : La conversion a débordé du champ d’un entier signé sur 32 bits.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- paramètre "&&fin" incorrect,
- fin de chaîne hors zone autorisée.

6.3 Transcodage BCD —> binaire

bcd_bin

Syntaxe de l'instruction

bcd_bin (codage_BCD)

Codage : Opérande ou expression numérique codé en BCD.

Fonctionnement

L'opérande , considéré comme signé , est étendu sur 32 bits avant d'être placé sur la pile. Le transcodage ne peut s'effectuer que sur un opérande dont chaque quartet ne dépasse pas la valeur 9 (codage en BCD). Si une erreur est détectée , la fonction retourne - 1.

Exemple:

%V0.L=bcd_bin(%V4.L)

%V4.L contient la valeur 12345678 codée en BCD ,

Représentation mémoire de %V4.L : 0001-0010-0011-0100-0101-0110-0111-1000

1 2 3 4 5 6 7 8

12345678 = = 0xBC614E

Représentation mémoire de %V0.L : 0000-0000-1011-1100-0110-0001-0100-1110

0 0 B C 6 1 4 E

⚠ ATTENTION

Lorsque l'opérande BCD est sur 8 ou 16 bits , afin de ne pas propager le bit de signe si le dernier quartet est > à 8 , il faut masquer le paramètre avec la valeur 0xFF ou 0xFFFF.
ex : bcd_bin(%V0.B & 0xFF) ; bcd_bin(%V0.W & 0xFFFF)

Code retourné

Si OK

Résultat du transcodage

Si défaut

- 1 : opérande non codé en BCD - un des quartets > 9

6.4 Transcodage binaire —> BCD

bin_bcd

Syntaxe de l'instruction

bin_bcd(codage_binaire)

Codage_binaire : Opérande ou expression numérique codé en binaire.

Fonctionnement

L'opérande considéré comme signé peut être de taille 8 , 16 ou 32 bits. Il est étendu sur 32 bits avant d'être placé sur la pile. Le transcodage ne peut s'effectuer que sur un opérande compris entre 0 et 99999999. Dans le cas contraire , le transcodage est erroné et la fonction retourne -1.

Exemples :

%V0.W=bin_bcd(1234)

1234==0x4D2	représentation en mémoire :	0000-0100-1101-0010
		0 4 D 2
%V0.W	représentation mémoire	0001-0010-0011-0100
		1 2 3 4

%V0.L=bin_bcd(12345678)

12345678==0xBC614E	représentation en mémoire	0000-0000-1011-1100-0110-0001-0100-1110
		0 0 B C 6 1 4 E
%V0.L	représentation en mémoire	0001-0010-0011-0100-0101-0110-0111-1000
		1 2 3 4 5 6 7 8

Code retourné

Si OK

Résultat du transcodage

Si défaut

- 1 : opérande non compris entre 0 et 99999999

6.5 Eclatement BIT —> octet

bit

Syntaxe de l'instruction

bit(&dest, &source, n)

&dest : Adresse du premier octet destination.

&source : Adresse du premier octet à éclater.

n : Nombre d'octets à éclater.

Eclatement de n octets en partant du bit 0 de l'adresse &source dans les bits de poids fort de 8xn octets débutant à l'adresse &dest.

Fonctionnement

Le bit 0 de l'octet à l'adresse &source est recopié dans le bit 7 de l'octet à l'adresse &dest; les 7 autres bits sont mis à 0.

Le bit 1 de l'octet à l'adresse &source est recopié dans le bit 7 de l'octet à l'adresse &dest + 1; les 7 autres bits sont mis à 0.

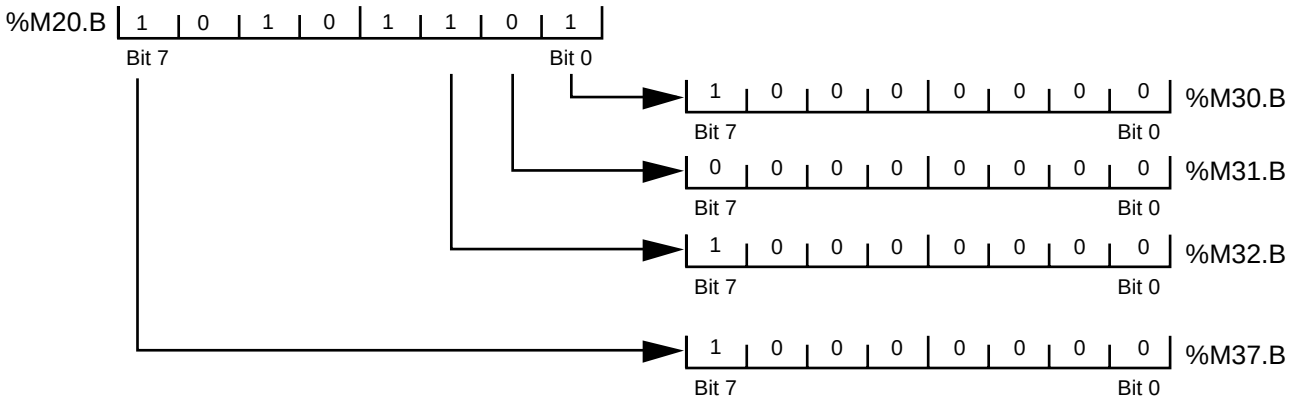
Le bit 0 de l'octet à l'adresse &source + 1 est recopié dans le bit 7 de l'octet à l'adresse &dest + 8; les 7 autres bits sont mis à 0.

Jusqu'à l'éclatement de n octets

REMARQUE : La fonction oct() effectue l'opération inverse (Voir 6.13)

Exemple

bit(%M30.&,%M20.&, 1)



Code retourné

Si OK

Non significatif

Si défaut

-1 : n négatif

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- paramètre "&dest" incorrect,
- "&source+n" hors zone autorisée
- "&dest+n" hors zone autorisée.

6.6 Lecture des paramètres stockés dans la pile cpyarg

Syntaxe de l'instruction

cpyarg(&dest, n)

&dest : Adresse d'un bloc mémoire dans lequel le système recopie les arguments.

n : Nombre d'arguments à recopier (6 maximum).

Copie à partir de l'adresse &dest n arguments stockés au sommet de la pile lors de l'appel du module par sp().

Fonctionnement

Chaque argument occupe 32 bits.

La fonction cpyarg() doit être appelé au début du module %SP avant que la pile ne soit modifiée par un appel à un label interne au module (call(<label>)).

Si le nombre d'arguments n demandé est supérieur au nombre d'arguments m passés lors de l'appel, le système ne génère pas d'erreur mais bien évidemment seuls les m premiers arguments seront significatifs.

Code retourné

Si OK

non significatif.

Si défaut

-1 : n négatif, nul ou supérieur au nombre maximum autorisé.

Erreur de programmation provoquant la mise endéfaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- "&dest+n" hors zone autorisée.

6.7 Copie d'un ou plusieurs octets

cpyb

syntaxe de l'instruction

cpyb(&dest, &source, n)

&dest : Adresse de la destination.

&source : adresse de la source.

n : Nombre d'octets à copier.

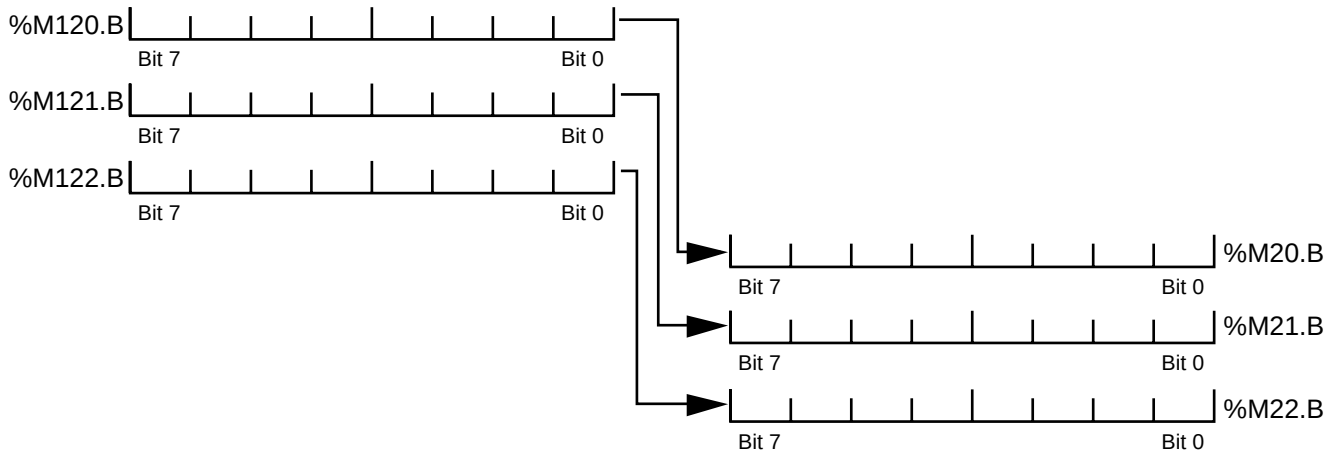
Copie n octets de la source vers la destination.

Sens du transfert

Pour permettre le transfert de zone mémoire ayant une intersection commune, le sens de la copie est fonction des adresses &dest et &source :
- Si &dest < &source alors la copie est faite du début vers la fin (adresses croissantes),
- Si &dest > &source alors la copie est faite de la fin vers le début (adresses décroissantes).

Exemple

cpyb(%M20.&, %M120.&, 3)



Code retourné

Si OK

0

Si défaut

-1 : n négatif ou nul.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- paramètre "&dest" incorrect,
- "&source+n" hors zone autorisée,
- "&dest+n" hors zone autorisée.

6.8 Copie d'un ou plusieurs mots — cpyw

Syntaxe de l'instruction

cpyw(&dest, &source, n)

&dest :	Adresse de la destination.
&source :	Adresse de la source.
n :	Nombre de mots à copier.

Copie n mots de la source vers la destination.

Sens du transfert

Se reporter au paragraphe *6.5.*

Code retourné

Si OK

0

Si défaut

-1 : n négatif ou nul.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- paramètre "&dest" incorrect,
- "&source+n" hors zone autorisée,
- "&dest+n" hors zone autotisée.

6.9 Copie d'un ou plusieurs long mots — cpyl

Syntaxe de l'instruction

cpyl(&dest, &source, n)

&dest : Adresse de la destination.

&source : Adresse de la source.

n: Nombre de longs mots à copier.

Copie n longs mots de la source vers la destination.

Sens du transfert

Se reporter au paragraphe 6.5.

Code retourné

Si OK

0

Si défaut

-1 : n négatif ou nul

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- paramètre "&dest" incorrect,
- "&source+n" hors zone autorisée,
- "&dest+n" hors zone autotisée.

6.10 Fixe la période de l'auto-test — diagiq

Syntaxe de l'instruction

diagiq(période)

période : Période de l'auto-test (en dixième de seconde).

Le système lit périodiquement le statu interne des cartes connectées sur le bus série (%I, %Q) et rafraîchit le mot de diagnostic %Irc3C.W.

Par défaut la période est de 400 millisecondes.

La fonction diagiq() permet de supprimer l'auto-test ou modifier la période par défaut. Le paramètre période doit être ZERO pour supprimer l'auto-test ou compris entre 1 (0,1 seconde) et 10 (1 seconde). On notera qu'une période petite est une charge supplémentaire pour le CPU.

diagiq() doit être appelé dans la tâche %INI.

Code retourné

Si OK

0

Si défaut

-1 : Période non valide (non compris entre 0 et 10) (la période par défaut reste valide).

6.11 Conversion d'une valeur entière signée en chaîne ASCII itoa

Syntaxe de l'instruction

itoa(i, &dest)

i : Valeur entière à convertir (la valeur est considérée signée).

&dest : Adresse de la chaîne ASCII_ZERO qui recevra les caractères ASCII.

Conversion d'une valeur entière signée dans la base 10. Les caractères ASCII résultat sont placés dans la chaîne d'adresse &dest. La chaîne est terminée par un octet NUL.

Code retourné

Si OK

Nombre de caractères placés dans la chaîne sans compter l'octet terminal NUL.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- fin de chaîne hors zone autorisée.

6.12 Conversion d'une valeur entière non signée en chaîne ASCII itostr

Syntaxe de l'instruction

itostr(u, &dest, base)

u : Valeur entière à convertir (la valeur est considérée non signée).

&dest : Adresse de la chaîne qui recevra les caractères ASCII.

base : Base de la conversion.

Conversion d'une valeur entière non signée dans la base spécifiée. Les caractères ASCII résultats sont placés dans la chaîne d'adresse &dest. La chaîne est terminée par un octet NUL.

La base doit être comprise entre 2 et 36 sinon la base 10 est prise.

Code retourné

Si OK

Nombre de caractères placés dans la chaîne sans compter l'octet terminal NUL.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- fin de chaîne hors zone autorisée.

6.13 Concaténation OCTet —> bit **oct**

Syntaxe de l'instruction

oct(&dest, &source, n)

&dest :	Adresse du premier octet destination.
&source:	Adresse du premier octet à concaténer.
n :	Nombre d'octets destination à concaténer.

Concaténation des bits de poids fort de 8 x n octets depuis &source sur les n octets débutant à &dest.

Fonctionnement

Le bit 7 de l'octet à l'adresse &source est recopié dans le bit 0 de l'octet à l'adresse &dest.

Le bit 7 de l'octet à l'adresse &source + 1 est recopié dans le bit 1 de l'octet à l'adresse &dest.

.....

Le bit 7 de l'octet à l'adresse &source + 8 est recopié dans le bit 0 de l'octet à l'adresse &dest + 1.

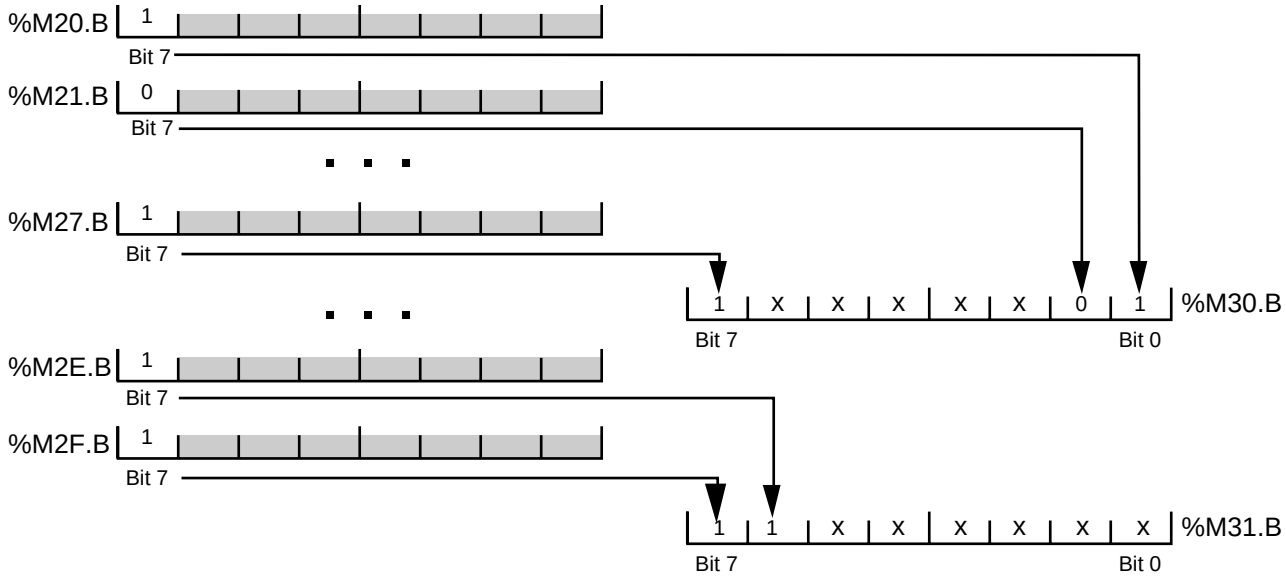
.....

Le bit 7 de l'octet à l'adresse &source + (n - 1) x 8 est recopié dans le bit 0 de l'octet à l'adresse &dest +(n - 1).

.....

Le bit 7 de l'octet à l'adresse &source + (n - 1) x 8 + 7 est recopié dans le bit 7 de l'octet à l'adresse &dest + (n - 1).

Exemple : oct(%M30.&, %M20.&, 2)



Code retourné

Si OK

0

Si défaut

-1 : n négatif ou nul

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- paramètre "&dest" incorrect,
- "&source+n" hors zone autorisée,
- "&dest+n" hors zone autorisée.

6.14 Simulation du clavier du pupitre — putkey

Syntaxe de l'instruction

putkey(code_touche)

Code_touche : Code ASCII d'une touche pupitre.

Simulation du clavier du pupitre par la fonction automatisme.

Fonctionnement

La fonction putkey() est valide si le pupitre est absent (%W5.0 = 1).

Pour s'assurer qu'un code simulé est pris en compte par la CN, après émission du code, attendre que le compte rendu repasse à 0. Le code a été pris en compte par la CN mais il n'y a pas moyen de s'assurer que le code va être traité. Aussi, il est recommandé de temporiser un nouvel appel de putkey() de 100 ms au minimum.

REMARQUE : La valeur 0xAF dans l'argument "Code_touche" permet un appel direct du mode transparent.

Code retourné

Si OK

0

Si défaut

-1 : Clavier pupitre non invalidé.

1 : Buffer saturé, réitérer l'appel de putkey(..)

6.15 Recherche circulaire optimale — qcktool

Syntaxe de l'instruction

qcktool(origine, destination, n)

origine: Numéro de la case origine (Voir Remarque).

destination: Numéro de la case destination (Voir Remarque).

n: Nombre de cases du magasin d'outils.

La fonction qcktool() détermine le nombre de cases et le sens de rotation permettant d'aller le plus rapidement de la case origine à la case destination dans un magasin d'outils circulaire.

REMARQUE : les cases sont numérotées en partant du numéro zéro (de 0 à n-1).

Code retourné

Si OK

Si > 0:	Le sens positif (numéros croissants) est le plus court. Indique le nombre de pas.
Si < 0:	Le sens négatif (numéros décroissants) est le plus court. La valeur absolue indique le nombre de pas.
Si = 0:	Aucun déplacement à faire le magasin est déjà sur la position destination.
Si = n	Hors magasin

6.16 Recherche de la valeur d'un octet **rchb**

Syntaxe de l'instruction

rchb(&source, b, pas, n)

&source :	Adresse de début de la recherche.
b :	Valeur de l'octet à chercher.
pas :	Valeur du pas de la recherche en octets.
n :	Nombre maximum de pas de la recherche.

Recherche, avec un pas, la première occurrence de l'octet b, en partant de l'adresse &source.

Le pas peut être positif ou négatif :

Pas positif:	Dans ce cas la recherche s'effectue vers les adresses croissantes.
Pas négatif:	Dans ce cas la recherche s'effectue vers les adresses décroissantes.

Code retourné

Si valeur trouvée

Nombre positif égal au nombre de pas effectués jusqu'à la première occurrence.

Pas positif:	Code retour = (adresse occurrence - &source) / pas
Pas négatif:	Code retour = (&source - adresse occurrence) / (-pas)

Si valeur non trouvée

-1 :	Valeur non trouvée.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre"&source" incorrect,
- "&source+pas*n"hors zone autorisée.

6.17 Recherche de la valeur d'un mot — rchw

Syntaxe de l'instruction

rchw(&source, w, pas, n)

&source: Adresse de début de la recherche.

w : Valeur du mot à chercher.

pas : Valeur du pas de la recherche en octets.

n : Nombre maximum de pas de la recherche.

Recherche, avec un pas, la première occurrence du mot w, en partant de l'adresse &source.

Le pas peut être positif ou négatif :

Pas positif: Dans ce cas la recherche s'effectue vers les adresses croissantes.

Pas négatif: Dans ce cas la recherche s'effectue vers les adresses décroissantes.

Code retourné

Si valeur trouvée

Nombre positif égal au nombre de pas effectués jusqu'à la première occurrence .

Pas positif: Code retour = (adresse occurrence - &source) / pas

Pas négatif: Code retour = (&source - adresse occurrence) / (-pas)

Si valeur non trouvée

-1 : valeur non trouvée.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- "&source+pas*n" hors zone autorisée.

6.18 Recherche de la valeur d'un long mot — rchl

Syntaxe de l'instruction

rchl(&source, l, pas, n)

&source: Adresse de début de la recherche.

l : Valeur du long mot à chercher.

pas : Valeur du pas de la recherche en octets.

n : Nombre maximum de pas de la recherche.

Recherche, avec un pas, la première occurrence du long mot l, en partant de l'adresse &source.

Le pas peut être positif ou négatif :

Pas positif: Dans ce cas la recherche s'effectue vers les adresses croissantes.

pas négatif: Dans ce cas la recherche s'effectue vers les adresses décroissantes.

Code retourné

Si valeur trouvée

Nombre positif égal au nombre de pas effectués jusqu'à la première occurrence .

Pas positif: Code retour = (adresse occurrence - &source) / pas

Pas négatif: Code retour = (&source - adresse occurrence) / (-pas)

Si valeur non trouvée

-1 : valeur non trouvée.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- "&souce+pas*n" hors zone autorisée.

6.19 Retour au module ou au réseau appelant return

Syntaxe de l'instruction

return ([expression_numérique])

expression_numérique: Valeur retournée vers le module appelant.

Fonctionnement

Effectue le retour :
- au module appelant dans le cas d'un appel inter-module de la forme <variable> = sp(.....). Dans ce cas <variable> permet de récupérer la valeur de l'expression numérique,
- au réseau de contacts appelant dans le cas d'un appel intra-module de le forme call(<label>). Dans ce cas l'éventuelle valeur retournée ne peut être récupérée.

⚠ ATTENTION

Cette fonction ne peut pas être appelée dans la zone test.

Code retourné

Aucun code n'est retourné dans le module lui même.

REMARQUE : Une affectation de la forme %M20.B = return(Var_1+3) n'a pas de sens.

6.20 Saut à un label du module sans retour goto

Syntaxe de l'instruction

goto (<label>)

Label: Label de la séquence appelée.

Fonctionnement

Saut à une séquence sans retour.

> ⚠ **ATTENTION**
>
> Cette fonction ne peut pas être appelée dans la zone test.

Code retourné

Aucun code n'est retourné.

6.21 Saut à un label du module avec retour call

Syntaxe de l'instruction

call (<label>)

Label: Label de la séquence appelée.

Fonctionnement

Saut à une séquence avec retour à la bobine suivant le Call(), sur le premier return () rencontré.

> ⚠ **ATTENTION**
>
> Cette fonction ne peut pas être appelée dans la zone test.

Code retourné

Aucun code n'est retourné.

6.22 Sémaphore sema

Syntaxe de l'instruction

sema (&sémaphore)

&sémaphore: Adresse de l'octet de sémaphore.

Utilise une instruction non interruptible du type Test and Set pour mettre à 0x80 (-128) l'octet à l'adresse &sémaphore.

Cette fonction doit être utilisée lorsque des tâches différentes doivent partager une même ressource (par exemple : le clavier, l'écran, .. etc ..).

Code retourné

Etat du sémaphore

0 : Le sémaphore était libre.

1 : Le sémaphore est déjà pris.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&sémaphore" incorrect.

6.23 Ecriture d'un ou plusieurs octets setb

Syntaxe de l'instruction

setb(&dest, b, n)

&dest: Adresse de la destination.

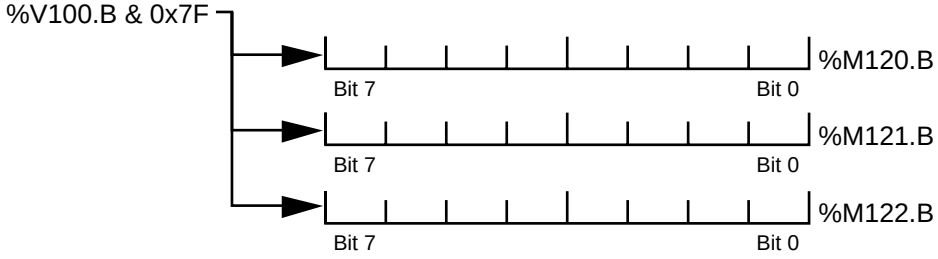
b : Valeur de l'octet à écrire.

n : Nombre d'octets à écrire.

Ecrit n octets à la valeur b depuis l'adresse &dest.

Exemple d'utilisation

setb(%M120.&, %V100.B & 0x7f, 3)



Code retourné

Si OK

Non significatif

Si défaut

-1 : n négatif ou nul

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- "&dest+n" hors zone autorisée.

6.24 Ecriture d'un ou plusieurs mots — setw

Syntaxe de l'instruction

setw(&dest, w, n)

&dest :	Adresse de la destination.
w :	Valeur du mot à écrire.
n :	Nombre de mots à écrire

Ecrit n mots à la valeur w depuis l'adresse &dest.

Code retourné

Si OK

Non significatif

Si défaut

-1 : n négatif ou nul

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- "&dest+n" hors zone autorisée.

6.25 Ecriture d'un ou plusieurs long mots setl

Syntaxe de l'instruction

setl(&dest, l, n)

&dest: Adresse de la destination.

l : Valeur du mot à écrire.

n : Nombre de mots à écrire

Ecrit n longs mots à la valeur l depuis l'adresse &dest.

Code retourné

Si OK

Non significatif

Si défaut

-1 : n négatif ou nul

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- "&dest+n" hors zone autorisée.

6.26 Appel de modules %SP

6.26.1 Appel d'un module %SP sp

Syntaxe de l'instruction

sp(n_module {, argn }6 ...)

n_module : Numéro du module %SP à appeler .

argn : Argument éventuel.

Appel de module %SP (%SP0 ... %SP255) avec passage éventuel d'arguments dans la pile.

Fonctionnement

Le numéro du module doit être compris entre 0 (appel de %SP0) et 255 (appel de %SP255).

Les arguments sont étendus sur 32 bits et placés sur la pile. L'appel est alors effectué.

Le nombre total d'arguments (n_module inclus) ne doit pas dépasser NBM_PARAM (soit 7).

L'appel de la fonction cpyarg() au début du module appelé permet de récupérer les arguments passés dans la pile.

Code retourné

Si OK

Valeur retournée par le module %SP appelé grâce à la fonction return(<expression_numérique>).

Non significatif si le module appelé n'a pas retourné de valeur.

Exemple d'utilisation de sp(), cpyarg() , return()

Echange d'arguments lors de l'appel d'un module %SP.

Module appelant (%TS , %TF ou %SP) :

%M100.W = sp(33, 10, %M20.B + %M30.B);Les arguments 10 et (%M20.B + %M30.B) sont étendus sur 32 bits et placés sur la pile. L'appel de %SP33 est alors effectué.

Module appelé %SP33 :

cpyarg(M200.&, 2); Recopie de deux paramètres de l'appel à partir de %M200.%M200.L reçoit 10 et %M204.L reçoit le résultat de l'expression (%M20.B + %M30.B).

return(%V100.W+25); Retour à l'appelant. %M100.W reçoit la valeur de l'expression %V100.W + 25.

6

Recommandation

Le passage d'argument permet d'éviter un couplage des modules par des variables communes.

C'est une règle de programmation à suivre car elle permet l'écriture de modules indépendants facilitant ainsi leur réutilisation dans une autre application.

6.26.2 Appel d'un module %SP avec variables locales %Y — spy

Syntaxe de l'instruction

spy (n_module {, argn }6 ...)

n_module : Numéro du module %SP à appeler .

argn : Argument éventuel.

Appel de module %SP (%SP0 à %SP255) avec création de 128 variables locales %Y et passage éventuel d'arguments dans la pile.

Fonctionnement

Le numéro du module doit être compris entre 0 (appel de %SP0) et 255 (appel de %SP255).

Il y a création dans la pile, de 128 variables locales %Y. Ces variables sont détruites lors du retour à l'appelant.

Les arguments, sauf n_module qui n'est pas empilé, sont étendus sur 32 bits et placés sur la pile.

Le nombre total d'arguments (n_module inclus) ne doit pas dépasser NBM_PARAM (soit 7).

L'utilisation de spy(..) et des variables %Y permet d'écrire des modules %SP portables et réentrants.

REMARQUE : Le programme d'exemple L_E_VAR.MCH disponible sous PLCTOOL illustre l'instruction spy().

Organisation des variables %Y disponibles dans le module %SP appelé :

%Y0.L	Contient le premier argument s'il existe sinon la valeur est indéterminée.
%Y4.L	Contient le deuxième argument s'il existe sinon la valeur est indéterminée.
%Y14.L	Contient le dernier argument s'il existe sinon la valeur est indéterminée.
%Y18.B	Suite des variables locale.
%Y7f.B	Dernière variable locale.

Code retourné

Si OK

Valeur retournée par le module %SP appelé grâce à la fonction return(<expression_numérique>).

Non significatif si le module appelé n'a pas retourné de valeur.

Exemple d'utilisation de spy(..) et return(..)

Echange d'arguments lors de l'appel d'un module %SP.

Module appelant (%TS , %TF ou %SP) :

%M100.W = spy(33, 10, %M20.B + %M30.B);	Création de 128 variables locales %Y dans la pile. Les arguments 10 et (%M20.B + %M30.B) sont étendus sur 32 bits et placés sur la pile. L'appel de %SP33 est alors effectué.
Module appelé %SP33 :	%Y0.L contient 10 %Y4.L contient le résultat de l'expression (%M20.B + %M30.B).
return(%Y10.W + 25)	Retour à l'appelant, les variables locales sont détruites, %M100.W reçoit le résultat de l'expression (%Y10.W + 25)

6.27 Formatage d'une chaîne de caractères — sprintf

Syntaxe de l'instruction

sprintf(&dest, &format {, argn }5)

&dest :	Adresse de la chaîne destination.
&format :	Adresse de la chaîne format.
argn :	Argument éventuel.

Formate la chaîne à l'adresse &format et recopie à l'adresse &dest. Un octet NUL est ajouté en fin de &dest.

La fonction sprintf() supporte les spécifications de conversion du langage C norme ANSI.

Fonctionnement

La fonction sprintf() est équivalent à printf() mais la chaîne formatée, au lieu d'être transmise à l'écran, est copiée à partir de l'adresse &dest.

Spécification des formats de conversion se reporter à la fonction printf().

Code retourné

Si OK

Nombre de caractères écrits dans &dest sans compter l'octet terminal NUL.

Si défaut

-1 : Chaîne format contenant des formats non valides.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- paramètre "&formt" incorrect,
- fin de chaîne hors zone autorisée.

6.28 Racine carrée entière — sqrt

Syntaxe de l'instruction

sqrt(n)

n : Entier positif.

Retourne la racine carrée entière de n.

Le temps de calcul est inférieur à 60 microseconde.

Code retourné

Si OK

Nombre entier positif le plus proche de la racine carrée de n.

6.29 Analyse d'une chaîne ASCII — sscanf

syntaxe de l'instruction

sscanf(&chaînesource, &chaîneformat, {, &argn }5)

&chaînesource : Adresse de la chaîne source.

&chaîneformat : Adresse de la chaîne format.

&argn : Adresse des variables à renseigner.

Analyse une chaîne ASCII (terminée par NUL) à l'adresse &chaînesource et renseigne les paramètres suivant les spécifications de conversion de la chaîne format.

La fonction sscanf() supporte les spécifications de conversion du langage C norme ANSI.

Fonctionnement

Chaque argument &argn doit être une adresse d'une variable %M, %V, %Q, %W.

Spécification des formats de conversion se reporter à la fonction printf() (Voir 8.2.5).

Code retourné

Si OK

Nombre de paramètres qui ont été effectivement renseignés.

Si défaut

0 :	L'analyse de chaîne source infructueuse, chaîne format contenant des formats non valides.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&chaînesource" incorrect,
- paramètre "&chaîneformat" incorrect,
- paramètre "&argn" d'un %d, %E, %C, %f, %G, %g, %i, %n, %o, %P, %u, %X ou %x incorrect,
- paramètre "&argn" d'un %s incorrect.
- fin de chaîne hors zone autorisée,

6.30 Comparaison d'une chaîne de caractères **strcmp**

Syntaxe de l'instruction

strcmp(&chaîne1, &chaîne2)

&chaîne1 :	Adresse chaîne 1.
&chaîne2 :	Adresse chaîne 2.

Compare deux chaînes terminées par un octet NUL.

Code retourné

Si OK

n == 0	Si chaîne1 == chaîne2.
n > 0	Si chaîne1 > chaîne2 (octet N°i de chaîne1 > octet N°i de chaîne2).
n < 0	Si chaîne1 < chaîne2 (octet N°i de chaîne1 < octet N°i de chaîne2).

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&chaîne1" incorrect,
- paramètre "&chaîne2" incorrect,
- fin de chaîne hors zone autorisée.

6.31 Copie d'une chaîne de caractères — strcpy

Syntaxe de l'instruction

strcpy(&dest, &source)

&dest : Adresse destination.

&source : Adresse source.

Copie les octets de la chaîne débutant à l'adresse &source dans &dest.

La copie s'arrête au premier octet NUL de la chaîne source. Un octet NUL est copié à la fin de &dest.

Code retourné

Si OK

Retour du pointeur sur la destination.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- paramètre "&dest" incorrect,
- fin de chaîne hors zone autorisée.

6.32 Calcul de la longueur d'une chaîne — strlen

Syntaxe de l'instruction

strlen(&chaîne)

&chaîne : Adresse de début de la chaîne.

Calcul la longueur d'une chaîne (nombre d'octets avant le premier octet NUL).

Code retourné

Si OK

Longueur de la chaîne.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&chaîne" incorrect,
- fin de chaîne hors zone autorisée.

6.33 Echange des octets d'un mot

swapw

Syntaxe de l'instruction

swapw(&dest, &source, n)

&dest : Adresse de la destination.

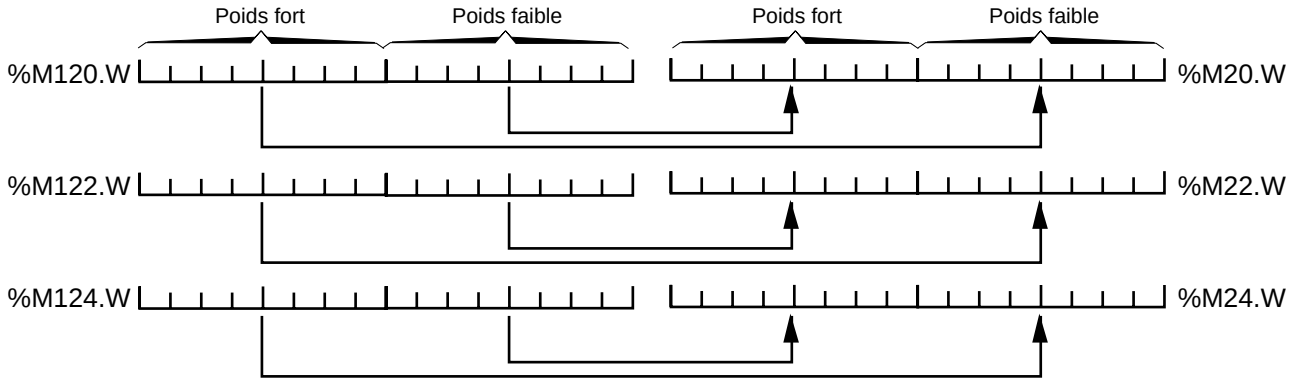
&source : Adresse de la source.

n : Nombre de mots à copier.

Copie n mots de &source dans &dest en inversant les octets de poids faible et de poids fort de chaque mot.

Exemple

swapw(%M20.&, %M120.&, 3)



Code retourné

Si OK

Non significatif.

Si défaut

-1 : n négatif ou nul

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- paramètre "&source" incorrect,
- "&dest+n" hors zone autorisée,
- "&source+n" hors zone autorisée.

6.34 Echange des quatre octets d'un long mot — swapl

Syntaxe de l'instruction

swapl(&dest, &source, n)

&dest : Adresse de la destination.

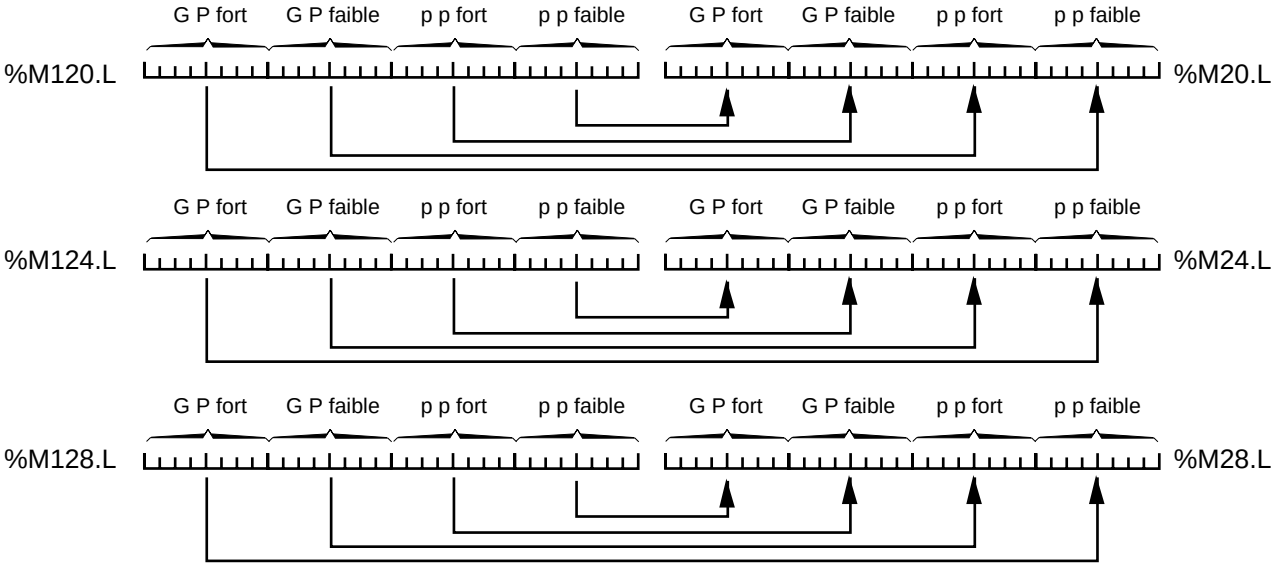
&source : Adresse de la source.

n : Nombre de longs mots à copier.

Copie n longs mots de &source dans &dest en inversant les 4 octets de chaque mot.

Exemple

swapl(%M20.&, %M120.&, 3)



Code retourné

Si OK

Non significatif.

Si défaut

-1 : n négatif ou nul

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- paramètre "&source" incorrect,
- "&dest+n" hors zone autorisée,
- "&source+n" hors zone autorisée.

6.35 Correction dynamique d'un outil

tooldyn

Syntaxe de l'instruction

tooldyn(correction, axe, n_outil)

correction: Valeur de la correction dynamique (entier signé sur 16 bits fonction de l'unité interne du système (Voir manuel des paramètres)).

axe : Type de correction.

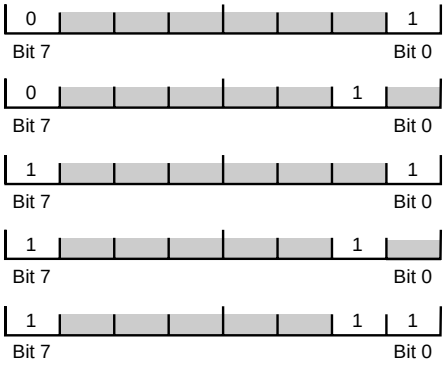
n_outil : Numéro de l'outil (de 0 à 255).

Ecriture d'une correction dynamique d'un outil (les corrections dynamiques sont cumulées par la CN).

Fonctionnement

Il est recommandé d'espacer le traitement de deux fonctions tooldyn(..) d'au moins une HTR.

Axe



0x1 : Incrémentation correction en X (Tour) ou L (Fraiseuse)

0x2 : Incrémentation correction en Z (Tour) ou R (Fraiseuse)

0x81 : RAZ du cumul des corrections en X (Tour) ou L (Fraiseuse)

0x82 : RAZ du cumul des corrections en Z (Tour) ou R (Fraiseuse)

0x83 : RAZ du cumul des corrections en X et Z (Tour) ou L et R (Fraiseuse)

Code retourné

Si OK

0

1 Fonction refusée - File saturée par une fonction tooldyn(..) émise précédement et encore en traitement.

6.36 Lecture de n variables E42000

R_E42000

Syntaxe de l'instruction

R_E42000(&dest, numéro, n)

&dest : Adresse de la destination.

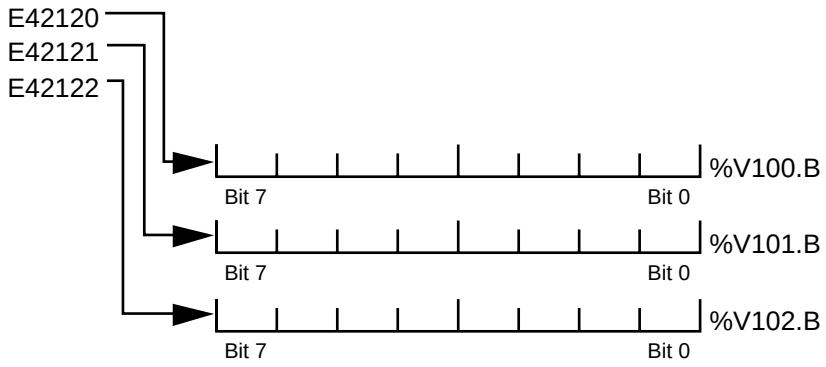
Numéro : Numéro de la première variable E42000 à lire (0 ... 127).

n : Nombre d'octets à lire (1 à 128).

Permet de lire n octets à partir de la variable E42000 + numéro dans la zone pointée par &dest.

Exemple

R_E42000(%V100.&, 120, 3)



Code retourné

Si OK

0

Si défaut

-1 : numéro > 127

numéro+n > 128

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- "&dest+n" hors zone autorisée.

6.37 Ecriture de n variables E42000

W_E42000

Syntaxe de l'instruction

W_E42000(&source, numéro, n)

&source : Adresse de la source.

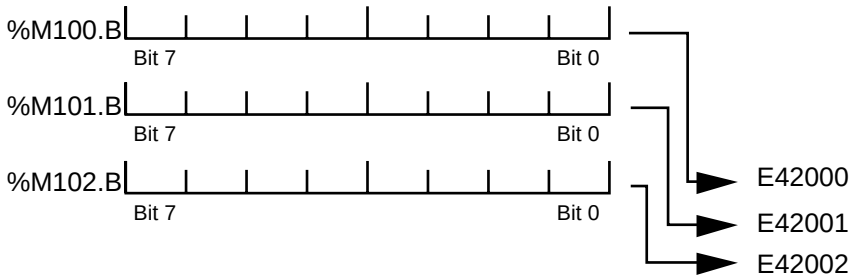
Numéro : Numéro de la première variable E42000 à écrire (0 ... 127).

n : Nombre d'octets à écrire (1 à 128).

Copie n octets de &source vers les variables E42000 + numéro.

Exemple

W_E42000(%M100.&, 0, 3)



Code retourné

Si OK

0

Si défaut

-1 : numéro > 127

numéro+n > 128

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- "&source+n" hors zone autorisée.

6.38 Initialisation de la base associée aux variables %Y — y_init

Syntaxe de l'instruction

y_init(&adresse_debut_y)

adresse_debut_y : Adresse chargée dans la base associée aux variables %Y.

Charge la base associée aux variables %Y avec l'adresse passée en paramètre.

Fonctionnement

Les variables %Y peuvent remplacer n'importe quelles variables globales %M, %V, %I, %Q, %R, %W. Le programmeur doit initialiser la base avec la fonction y_init(..) avant d'utiliser les variables %Y.

L'utilisation de la fonction y_init(..) fait perdre la visibilité des éventuelles variables locales du module %SP. Pour récupèrer la visibilité il faut procèder comme suit :

Var_1 = %Y0.&	Sauvegarde de la base dans Var_1 (Ex : %V100.L)
y_init(Var_2 + 100)	La base pointe sur une nouvelle zone de variables
.....	Utilisation des nouvelles variables %Y
y_init(Var_1)	Restitution de la base
.....	Utilisation des variables %Y locales

REMARQUE: Le chargement d'une base associée aux variables %Y par la fonction y_init est opérant seulement pendant l'exécution de la tâche en cours(%TS, %TF, %TH ou %INI).

Par exemple:

- si les variables %Y sont utilisées dans la tâche %TS0, il faudra appeler la fonction y_init à chaque exécution de la tâche %TS0.

- si les variables %Y sont utilisées dans une tâche de fond sans fin (tâche de fond qui boucle), il suffira de l'appeler une fois en début de tâche.

Exemple d'utilisation de y_init(..) et %Y

Traitement des 8 groupes d'axes avec %SP0 unique en utilisant les variables %Y.

y_init(%R100.&)	%Y0.B remplace %R100.B	%Y80.B remplace %W100.B
sp(0)	Traitement du groupe d'axes N°1	
y_init(%R200.&)	%Y0.B remplace %R200.B	%Y80.B remplace %W200.B
sp(0)	Traitement du groupe d'axes N°2	
y_init(%R800.&)	%Y0.B remplace %R800.B	%Y80.B remplace %W800.B
sp(0)	Traitement du groupe d'axes N°8	

Traitement d’une chaîne.

%V100.L = «ABCDEF»	%V100.L contient l’adresse de début de chaîne.
y_init(%V100.L)	La base y pointe sur le début de la chaîne.
%Y0.B == A	%Y0.B correspond au premier caractère de la chaîne.
%Y1.B == B	%Y1.B correspond au deuxième caractère de la chaîne.
%Y5.B == F	

Code retourné

Le code retourné est non significatif.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&adresse_debut_y" incorrect.

7 Gestion des tâches

7.1 Introduction

Pour plus d'informations concernant le traitement des tâches de fond, se reporter au paragraphe **2.1.2.3**.

7.2 Début d'une section critique — csbegin

Syntaxe de l'instruction

csbegin()

Description

Interdit la préemption de la tâche appelante par une autre tâche %TS, %TH %TF.

Code retourné

Toujours OK

0

7.3 Fin d'une section critique — csend

Syntaxe de l'instruction

csend()

Description

Autorise la préemption de la tâche appelante par une tâche de priorité supérieure. Cette fonction annule les effets de la fonction csbegin().

Code retourné

Toujours OK

0

7.4 Mise en sommeil temporaire d'une tâche %TF — whtr

Syntaxe de l'instruction

whtr(n)

n : Nombre de HTR pendant lesquels la tâche %TF est en ATTENTE.

Description

Fait passer la tâche %TF appelante de l'état EN EXECUTION à l'état EN ATTENTE pendant n HTR. A la fin de ce délai la tâche %TF passera à l'état PRETE. n doit être compris entre 0 et 255.

Code retourné

Si OK

0

7.5 Départ d'une tâche %TF **tfstart**

Syntaxe de l'instruction

tfstart(numero_tf)

numéro_tf : Numéro de la tache %TF.

Description

Fait passer la tâche %TF dans l'état PRETE.

Code retourné

Si OK

0

7.6 Arrêt d'une tâche %TF **tfstop**

Syntaxe de l'instruction

tfstop(numero_tf)

numéro_tf : Numéro de la tache %TF.

Description

Fait passer la tâche %TF dans l'état NON PRETE.

Code retourné

Si OK

0

8 Mode transparent

8.1 Introduction

Les programmes en mode transparent doivent être exécutés lorsque la variable %R5.7 est à 1. Cette variable doit être utilisée dans les conditions d'armements du programme.

Pour accéder à la page «MODE TRANSPARENT» se reporter au «MANUEL OPERATEUR».

8.1.1 Gestion de la visu

Le mode transparent libère la fonction CN de la gestion de l'écran, et permet à la fonction automatisme de disposer de l'écran du pupitre pour afficher des caractères alphanumériques ou effectuer des tracés graphiques.

Les commandes de gestion du curseur, les caractères alphanumériques et les instructions graphiques sont associés à des codes hexadécimaux.

Les fonctions de gestion écran/clavier sont valides uniquement dans le mode transparent.

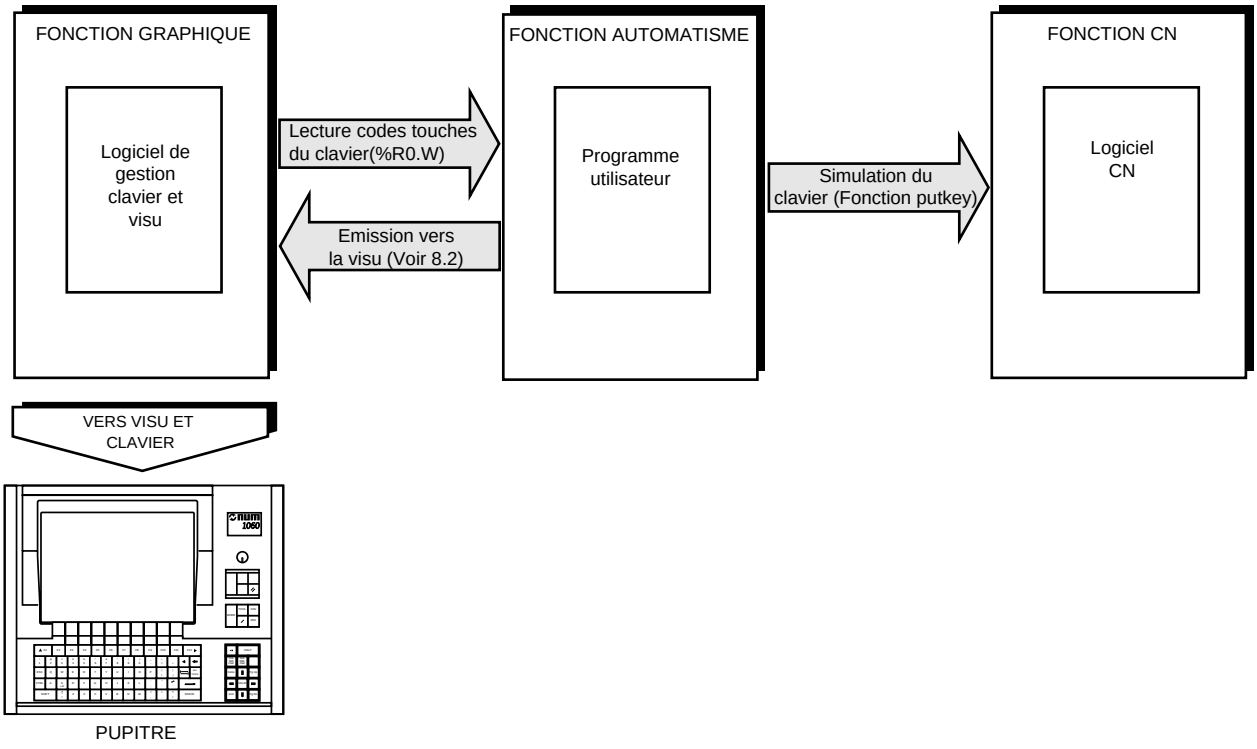


Figure 8.1 - Principe du mode transparent

⚠ ATTENTION

Le forçage du mode transparent ne peut-être réalisé qu'avec les pages du graphique comprenant le cartouche de base. Pour être sur ce cartouche il faut ajouter l'envoi du code $8D par putkey, ce qui correspond à la touche << (que l'on soit en MODE, TOOL ou JOG).

8.1.2 Variable d'échange

La variable %R0.W «CARCLAV» permet la lecture du code des touches émis par le clavier du pupitre au rythme de 1 caractère toutes les 5 HTR et de les exploiter au travers du programme utilisateur.

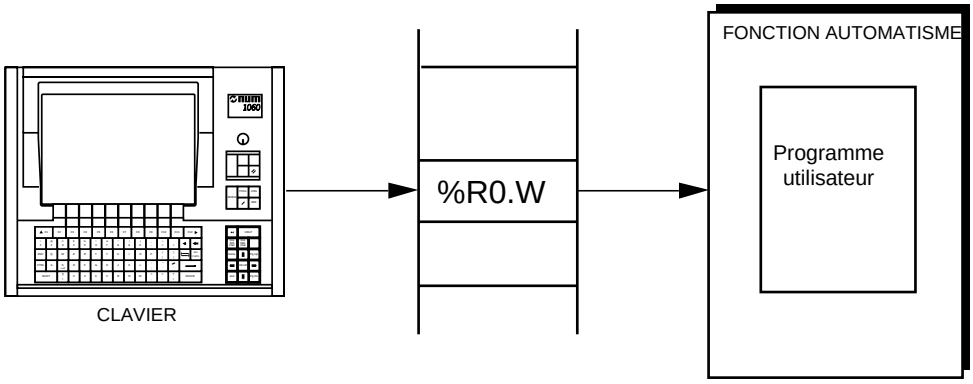


Figure 8.2 - Variable d'échange %R0.W

La fonction putkey() permet, si le clavier du pupitre est invalidé (Variable %W5.0 à 1), de simuler le clavier du pupitre au travers du programme utilisateur.

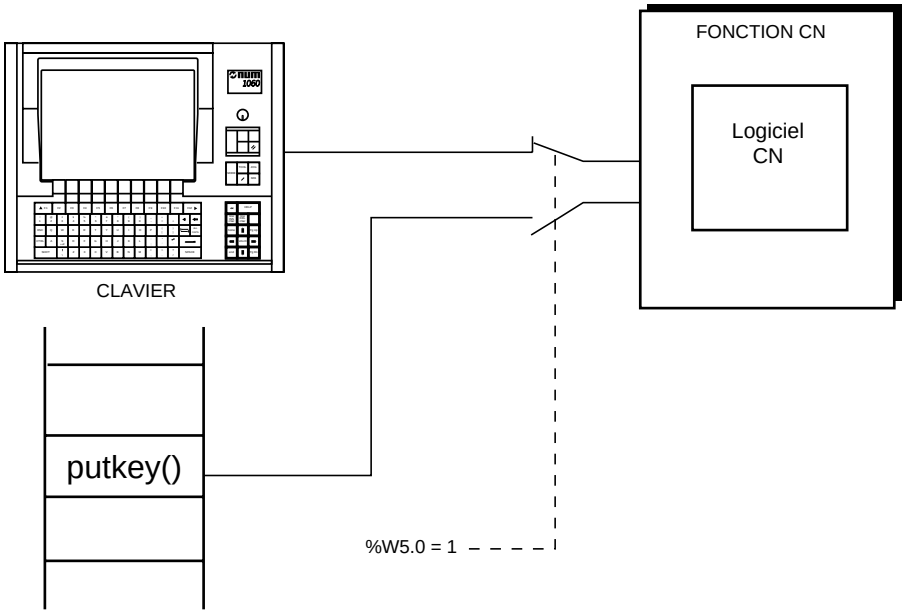


Figure 8.3 - Fonction putkey()

8.1.3 Envoi de caractères codés vers l'écran

Les fonctions putchar(), puts(), print(), printf() (Voir 8.2) permettent d'envoyer les commandes de gestion du curseur et les caractères alphanumériques vers l'écran du pupitre.

8.1.4 Caractères codés exploités par %R0.W et putkey()

Tous les caractères sont lus dans la variable %R0.W. La fonction putkey() simule la partie dialogue uniquement.

CODE HEXA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		Ctrl P (DLE)	SP	0	@	P	’	p							0,1	CONT
1	Ctrl A	Ctrl Q (Xon)	!	1	A	Q	a	q	F2	↑	MODE	Shift F2	Shift ↑		1	SEQ
2	Ctrl B	Ctrl R	”	2	B	R	b	r	F3	↓		Shift F3	Shift ↓	Jauge	10	IMD
3	Ctrl C	Ctrl S (Xoff)	#	3	C	S	c	s	F4	←	TOOL	Shift F4	Shift ←		100	RAP
4	Ctrl D	Ctrl T	$	4	D	T	d	t	F5	→	/ (Saut bloc)	Shift F5	Shift →		1000	RNS
5	Ctrl E	Ctrl U	%	5	E	U	e	u	F6	HOME	JOG	Shift F6	Shift HOME		10 000	MODIF
6	Ctrl F	Ctrl V	&	6	F	V	f	v	F7	END	M01 (Aropt)	Shift F7	Shift END	INCOR	ILL	TEST
7	Ctrl G	Ctrl W	‘	7	G	W	g	w	F8	Pg Up	// (Raz)	Shift F8	Shift Pg Up	L ou X		MANU
8	Ctrl H ⇦	Ctrl X	(	8	H	X	h	x	F9	Pg Dn	NU_CN	Shift F9	Shift Pg Dn	R ou Z	MANIV.	POM
9	Ctrl I	Ctrl Y	)	9	I	Y	i	y	F10	Ins/Over (] [)		Shift F10	Shift Ins/Over	RAZCOR	0.01	PREF
A	Ctrl J LF	Ctrl Z	*	:	J	Z	j	z	F11	Del car		Shift F11	Del line		0.001	REGOUT
B	Ctrl K	ESC Ctrl [	+	;	K	[	k	{	F1 s	NU_EDT	MACHINING		Shift NU_EDT			
C	Ctrl L	Ctrl \	,	<	L	\	l	|	F12 ▶	VALID LF	PRESET		Shift VALID			
D	Ctrl M ↵ CR	Ctrl]	-	=	M	]	m	}	◀◀	HELP	PROGRAM EDIT					CHARG
E	Ctrl N →|		.	>	N	^	n	~			SPLIT					
F	Ctrl O →|		/	?	O	_	o	⇦			Appel Mode transparent					DECHARG

Ne correspondent pas à des touches du clavier. Les codes sont émis par le gestionnaire de menus.

8

Pupitre compact

CODE HEXA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		Ctrl P														
1	Ctrl A	Ctrl Q														
2	Ctrl B	Ctrl R														
3	Ctrl C	Ctrl S														
4	Ctrl D	Ctrl T														
5	Ctrl E	Ctrl U														
6	Ctrl F	Ctrl V														
7	Ctrl G	Ctrl W														
8	Ctrl H ⇦	Ctrl X														
9	Ctrl I	Ctrl Y														
A	Ctrl J LF	Ctrl Z														
B	Ctrl K	Ctrl [														
C	Ctrl L	Ctrl \											F13	●		
D	Ctrl M ↵ CR	Ctrl]											F14	●		
E	Ctrl N												F15	●	●	
F	Ctrl O												F16	●	●	

REMARQUE *Ces codes touches sont accessibles par l'espion automate, mais la simulation de ces codes n'est pas prise en compte.*

8.2 Fonctions affectées au mode transparent

> ⚠ **ATTENTION**
>
> Ces fonctions sont valides uniquement en mode transparent.
>
> Elles sont programmables uniquement dans une tâche %TF.

8.2.1 Positionnement du curseur **pcur**

Syntaxe de l'instruction

pcur(ligne, colonne)

ligne : Numéro de la ligne.

colonne : Numéro de la colonne.

Description

Positionne le curseur sur la ligne et la colonne

Code retourné

Si OK

0

Si défaut

-1 : Pas en mode transparent, la tâche appelante n'est pas une %TF.

-2 : Tentative de positionnement hors écran.

8.2.2 Affichage d'un caractère **putchar**

Syntaxe de l'instruction

putchar(caractère)

caractère : Code ASCII d'un caractère.

Description

Envoi d'un caractère sur l'écran du système.

REMARQUE: Si l'économiseur d'écran est en fonctionnement, la commande putchar est en attente. Pour reprendre l'affichage en mode transparent, il est nécessaire de désactiver préalablement l'économiseur par la variable %W5.7

Code retourné

Si OK

Retourne le caractère écrit.

Si défaut

-1 : Pas en mode transparent, la tâche appelante n'est pas une %TF.

8.2.3 Affichage d'une chaîne sans formatage **puts**

Syntaxe de l'instruction

puts(&chaîne)

&chaîne : Adresse de la chaîne ASCII (terminée par NUL) à afficher.

Description

Emission d'une chaîne sur l'écran du système. Le système ajoute un 0x0D9C (\n) à la fin de la chaîne.

Code retourné

Si OK

Retourne le nombre de caractères transmis.

Si défaut

-1 : Pas en mode transparent, la tâche appelante n'est pas une %TF.

-2 : Dépassement de la taille maxi du tampon (512 octets).

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&chaîne" incorrect,
- fin de chaîne hors zone autorisée.

8.2.4 Affichage d'un tampon **print**

Syntaxe de l'instruction

print(&source, n)

&source: Adresse du tampon à émettre.

n: Nombre d'octets à émettre.

Description

Emission d'un tampon d'octets sur l'écran du système (le tampon peut contenir des commandes graphiques).

L'affichage s'arrête suivant la valeur de n.

Si n == 0:	L'affichage s'arrête sur le premier octet NUL (NUL non affiché).
Si n > 0:	L'affichage s'arrête au bout de n octets.

Code retourné

Si OK

Retourne le nombre de caractères transmis.

Si défaut

-1 :	Pas en mode transparent, la tâche appelante n'est pas une %TF.
-2 :	dépassement de la taille maxi du tampon (512 octets).

Erreur de programmation provoquant la mise en défaut de l'unité centrale:

Accès à une adresse interdite :
- paramètre "&source" incorrect,
- "&source+n" hors zone autorisée.

8.2.5 Affichage d'une chaîne avec formatage — printf

Syntaxe de l'instruction

printf(&format {, argn}6)

&format :	Adresse de la chaîne format.
argn :	Argument éventuel.

Affichage d'une chaîne avec conversion des éventuels arguments (la chaîne ne doit pas contenir de commande graphique).

La fonction printf() supporte les spécifications de conversion du langage C norme ANSI.

Fonctionnement

La chaîne format contient des caractères affichables et éventuellement des spécifications pour la conversion des arguments.

La fonction printf() analyse les caractères de la chaîne format.

Si le caractère est un caractère affichable, printf() le recopie dans un tampon de travail.

Lorsque printf() détecte le caractère %, elle analyse les caractères suivants qui indiquent la conversion à effectuer sur l'argument correspondant. Les caractères affichables, résultat de la conversion de l'argument, sont placés dans le tampon de travail.

Lorsque printf() détecte la fin de la chaîne format (octet NUL), elle transmet le tampon à la tâche chargée de l'affichage sur l'écran CN.

Format des spécifications de conversion

% [flags] [chiffres] [. [chiffres]] [l] lettre_conversion

% :	Indique le début d'une spécification de conversion.
[flags] :	Caractères facultatifs suivants :
- :	Indique que le résultat de la conversion doit être cadré à gauche dans le champ réservé.
+ :	Indique que le résultat d'une conversion signée doit débuter par un signe + ou un signe -.
«espace» :	Indique que le résultat d'une conversion signée doit débuter par un «espace». Ce flag est ignoré si le flag + est présent.
# :	Indique que le résultat de la conversion doit être modifié de la façon suivante:
o Conversion :	Le résultat doit débuter par un 0.
x ou X conversion :	Le résultat doit débuter par 0x ou 0X.
0 :	Indique que le zéro de tête du résultat doivent être affichés.
[chiffres] :	Caractères décimal ASCII facultatifs indiquant la taille minimum du champ utilisé pour l'affichage du résultat de la conversion.
[. [chiffres]] :	Caractères décimal ASCII facultatifs indiquant le nombre minimum de chiffres à afficher dans le cas d'une conversion d, o, u, x ou X ou le nombre maximum de caractère de la chaîne à afficher dans le cas d'une conversion s.
lettre_conversion :	Lettre obligatoire indiquant la conversion à effectuer sur l'argument :
d :	L'argument est affiché en décimal signé.
o :	L'argument est affiché en octal.
x :	L'argument est affiché en hexadécimal en utilisant les lettres «abcdef».
X :	L'argument est affiché en hexadécimal en utilisant les lettres «ABCDEF».
u :	L'argument est affiché en décimal non signé.
c :	L'argument est pris comme le code d'un caractère ASCII et affiché sans conversion.
s :	L'argument est un pointeur sur une chaîne affichée sans conversion.
% :	%% permet d'afficher le caractère %.

Exemple 1

Soit les variables suivantes.

%V100.L = « Outil numéro:%5d Type:%2c%2c Temps d'utilisation: %2d heures %2d minutes»

%M50.W = 255

%M52.B = 0x55 (0x55 code ASCII de t , 0x57 code ASCII de v)

%M54.B = 2

%M55.B = 57

L'instruction printf(%V100.L, %M50.W, %M52.B, 0x57, %M54.B, %M55.B) affichera :

Outil numéro: 255 Type: t v Temps d'utilisation: 2 heures 57 minutes

Exemple 2

Affichage d'une chaîne simple

%V200.L = «Voulez-vous connaître l'heure ? (O/N)»

printf(%V200.L) Affichera: Voulez-vous connaître l'heure ? (O/N)

Affichage d'une chaîne avec format d'affichage pour les arguments

Si %M10.B = 3;

Si %M11.B = 15;

%V200.L = «Il est %2d heure(s) et %2d minute(s)»

printf(%V200.L, %M10.B, %M11.B) affichera: Il est 3 heure(s) et 15 minute(s)

Les deux caractères \n provoque un saut à la ligne suivante lors de l'affichage de la chaîne (Le compilateur remplace les deux caractères \n par les deux octets 0xd 0xa).

Exemple 3

%V200.L = «1 - Lecture \n 2 - Ecriture»

printf (%V200.L) affichera

1 - Lecture

2 - Ecriture

Code retourné

Si OK

Nombre de caractères transmis pour affichage.

Si défaut

-1 : Pas en mode transparent, la tâche appelante n'est pas une %TF.

-2 : Dépassement de la taille maxi du tampon de formatage (255 octets).

-3 : Erreur de format dans la chaîne format.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&format" incorrect,
- fin de chaîne hors zone autorisée.

8.2.6 Ouverture d'une acquisition clavier

scano

Syntaxe de l'instruction

scano(&question, largeur)

&question : Adresse d'une chaîne de caractères (terminée par NUL).

largeur : Largeur maximum du champ de saisie.

Ouverture d'une acquisition clavier.

Fonctionnement

Le système affiche la chaîne pointée par &question en bas de l'écran et engage le dialogue à la suite de la chaîne.

Si le paramètre &question == 0 aucune chaîne n'est affiché

La saisie des caractères est sous contrôle de l'éditeur ligne du système.

L'éditeur contrôle que le nombre de caractères saisis est inférieur à largeur.

Les commandes de l'éditeur ligne sont celles, classiques, de l'éditeur des programmes pièces :
- déplacement du curseur avant et arrière, début et fin de ligne,
- insertion et effacement de caractères,
- la touche Line Feed clôt la saisie.

ATTENTION

Les fonctions d'affichages putchar() et printf() sont interdites pendant une acquisition clavier.

Code retourné

Si OK

0

Si défaut:

-1 : Pas en mode transparent, la tâche appelante n'est pas une %TF.

-2 : Ressource déjà prise (une acquisition clavier est déjà en cours).

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&question" incorrect,
- fin de chaîne hors zone autorisée.

8.2.7 Ouverture d'une acquisition clavier numérique — scanu

Syntaxe de l'instruction

scanu(&question, largeur)

&question : Adresse d'une chaîne de caractères (terminée par NUL).

largeur : Largeur maximum du champ de saisie.

Ouverture d'une acquisition numérique au clavier.

Fonctionnement

Le fonctionnement de scanu() est identique à scano(), sauf en ce qui concerne l'éditeur ligne qui interdit la saisie de caractères qui ne sont pas des chiffres décimaux (0, 1 .... 9).

> ⚠ **ATTENTION**
>
> Cette fonction ne peut-être utilisée qu'avec la police 12 lignes x 40 colonnes. La saisie s'effectue sur la 11ème ligne avec effacement de la ligne immédiatement supérieure.

Code retourné

Si OK

0

Si défaut

-1 : Pas en mode transparent, la tâche appelante n'est pas une %TF.

-2 : Ressource déjà prise (une acquisition clavier est déjà en cours).

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&question" incorrect,
- fin de chaîne hors zone autorisée.

8.2.8 Acquisition d'une chaîne — scans

Syntaxe de l'instruction

scans(&dest)

&dest : Adresse d'une zone mémoire (%M ou %V) qui va recevoir les caractères frappés au clavier.

Lecture d'une acquisition clavier. Cette fonction doit être appelée après une fonction d'ouverture de dialogue scano() ou scanu().

Fonctionnement

Cette fonction permet de réceptionner la chaîne frappée à la fin d'un dialogue opérateur.

Le système termine la chaîne par un octet NUL.

Si le dialogue est en cours (la touche Line Feed n'a pas été frappée) le code 0 est retourné; il faut donc appeler cycliquement scans() jusqu'à la fin du dialogue.

Code retourné

Si OK

0 :	Dialogue en cours
n > 0 :	Nombre de caractères transférés dans &dest (le dialogue est terminé).

Si défaut

-1 :	Pas en mode transparent, la tâche appelante n'est pas une %TF.
-2 :	Pas de dialogue en cours.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&dest" incorrect,
- fin du champ acquisition hors zone autorisée.

8.2.9 Acquisition et conversion d'un nombre décimal — scand

Syntaxe de l'instruction

scand(&lvariable)

&lvariable :	Adresse d'une variable .L (ex : %V100.L) qui va recevoir le résultat de la conversion ASCII -> entier signé de la chaîne entrée au clavier.

Lecture et conversion d'une valeur décimale acquise au clavier. Cette fonction doit être appelée après une fonction d'ouverture de dialogue scano() ou scanu().

La conversion s'arrête sur le premier caractère non décimal. Si aucun caractères décimal n'est détecté alors &lvariable reçoit 0.

Fonctionnement

Cette fonction permet de réceptionner la valeur d'un nombre décimal à la fin d'un dialogue opérateur.

Si le dialogue est en cours (la touche Line Feed n'a pas été frappée) le code 0 est retourné; il faut donc appeler cycliquement scand() jusqu'à la fin du dialogue.

Code retourné

Si OK

0 : Dialogue en cours

1 : Acquisition et conversion terminées avec succès. Le résultat est transféré dans la variable . L pointée par &Ivariable (le dialogue est terminé). La conversion s'arrête sur le premier caractère non décimal.

Si défaut

-1 : Pas en mode transparent, la tâche appelante n'est pas une %TF.

-2 : Pas de dialogue en cours.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&Ivariable" incorrect.

8.2.10 Acquisition et conversion d'un nombre hexadécimal **scanx**

Syntaxe de l'instruction

scanx(&Ivariable)

&Ivariable : Adresse d'une variable .L (ex: %V100.L) qui va recevoir le résultat de la conversion ASCII -> entier signé de la chaîne entrée au clavier.

Lecture et conversion d'une valeur hexadécimale acquise au clavier. Cette fonction doit être appelée après une fonction d'ouverture de dialogue scano() ou scanu().

La conversion s'arrête sur le premier caractère non hexadécimal. Si aucun caractères hexadécimal n'est détecté alors &Ivariable reçoit 0.

Fonctionnement

Cette fonction permet de réceptionner la valeur d'un nombre hexadécimal à la fin d'un dialogue opérateur.

Si le dialogue est en cours (la touche Line Feed n'a pas été frappée) le code 0 est retourné; il faut donc appeler cycliquement scanx() jusqu'à la fin du dialogue.

Code retourné

Si OK

0 : Dialogue en cours.

1 : Acquisition et conversion terminées avec succès. Le résultat est transféré dans la variable .L pointée par &Ivariable (le dialogue est terminé). La conversion s'arrête sur le premier caractère non hexadécimal.

Si défaut

-1 : Pas en mode transparent, la tâche appelante n'est pas une %TF.

-2 : Pas de dialogue en cours.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&Ivariable" incorrect.

8.2.11 Fermeture d'une acquisition clavier **scanc**

Syntaxe de l'instruction

scanc( )

Description

Cette fonction annule un dialogue en cours (engagé par la fonction scano() ou scanu()).

Code retourné

Si OK

0

Si défaut

-1 : Pas en mode transparent, la tâche appelante n'est pas une %TF.

-2 : Pas de dialogue en cours.

8.2.12 Positionnement et affichage d'une image **putimage**

Syntaxe de l'instruction

putimage(x, y, &image, n)

x : Abscisse de départ.

y : Ordonnée de départ.

&image: Adresse d'un tampon de commandes graphiques (0x9b...).

n: Nombre d'octets à émettre.

Emission d'un tampon contenant des commandes graphiques avec positionnement préalable du curseur à l'abscisse x, ordonnée y. L'affichage s'arrête suivant la valeur de n.

Fonctionnement

putimage() fonctionne comme print() mais avec positionnement préalable en (x, y).

putimage() permet de dupliquer avec des (x,y) différents une même image.

Si n == 0 :	L'affichage s'arrête sur le premier octet NUL (NUL non affiché).
Si n > 0 :	L'affichage s'arrête au bout de n octets.

Code retourné

Si OK

Retourne le nombre de caractères transmis.

Si défaut

-1 :	Pas en mode transparent, la tâche appelante n'est pas une %TF.
-2 :	Dépassement de la taille maxi du tampon (512 octets), tentative de positionnement hors écran.

Erreur de programmation provoquant la mise endéfaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&image" incorrect,
- "&image+n" hors zone autorisée.

8.2.13 Init graphique

inig

8

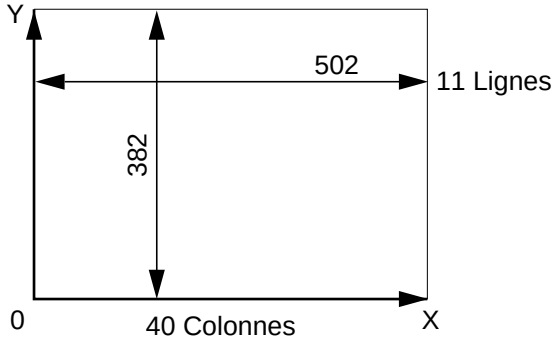
Syntaxe de l'instruction

inig(..)

Description

Permet d'initialiserle graphique et de définir le référentiel. L'unité de programmation est le pixel.

Le référentiel est définit comme suit :
- l'axe X des abscisses de 0 à 502 pixels,
- l'axe Y des ordonnées de 0 à 382 pixels.



8.3 Mode transparent pupitre

8.3.1 Exploitation de l'écran du pupitre

L'écran, de définition 640 x 480 pixels, est divisé en quatre fenêtres. A chaque fenêtre correspond un canal de communication et un contexte. Le gestionnaire d'affichage exploite tous les canaux et assure la sauvegarde des contextes.

REMARQUE : Chaque fenêtre doit être considérée comme un écran.

8.3.1.1 Définition d'une fenêtre

Une fenêtre est définie par sa taille et sa position dans l'écran.

Chaque fenêtre dispose d'un espace alphanumérique et d'un espace graphique qui ont leur propre contexte (couleur, police, .. etc ...). Ces espaces se positionnent dans la zone visualisable de la fenêtre.

REMARQUES : Les fenêtres se chevauchent.
Les espaces se superposent.

8.3.1.2 Définition de l'espace alphanumérique

Il permet l'affichage des caractères ASCII codés (au pixel près) en lignes et colonnes et la gestion du curseur

REMARQUE : L'affichage d'un caractère alphanumérique écrase l'élément préalablement affiché quelque soit l'espace de celui-ci.

8.3.1.3 Définition de l'espace graphique

Il permet l'affichage des textes (au pixel près) et des éléments graphiques.

Deux référentiels (écran et utilisateur) sont disponibles. Le mixage des référentiels dans le même espace est possible.

REMARQUE : L'affichage d'un élément dans l'espace graphique se superpose à l'élément préalablement affiché quelque soit l'espace de celui-ci.

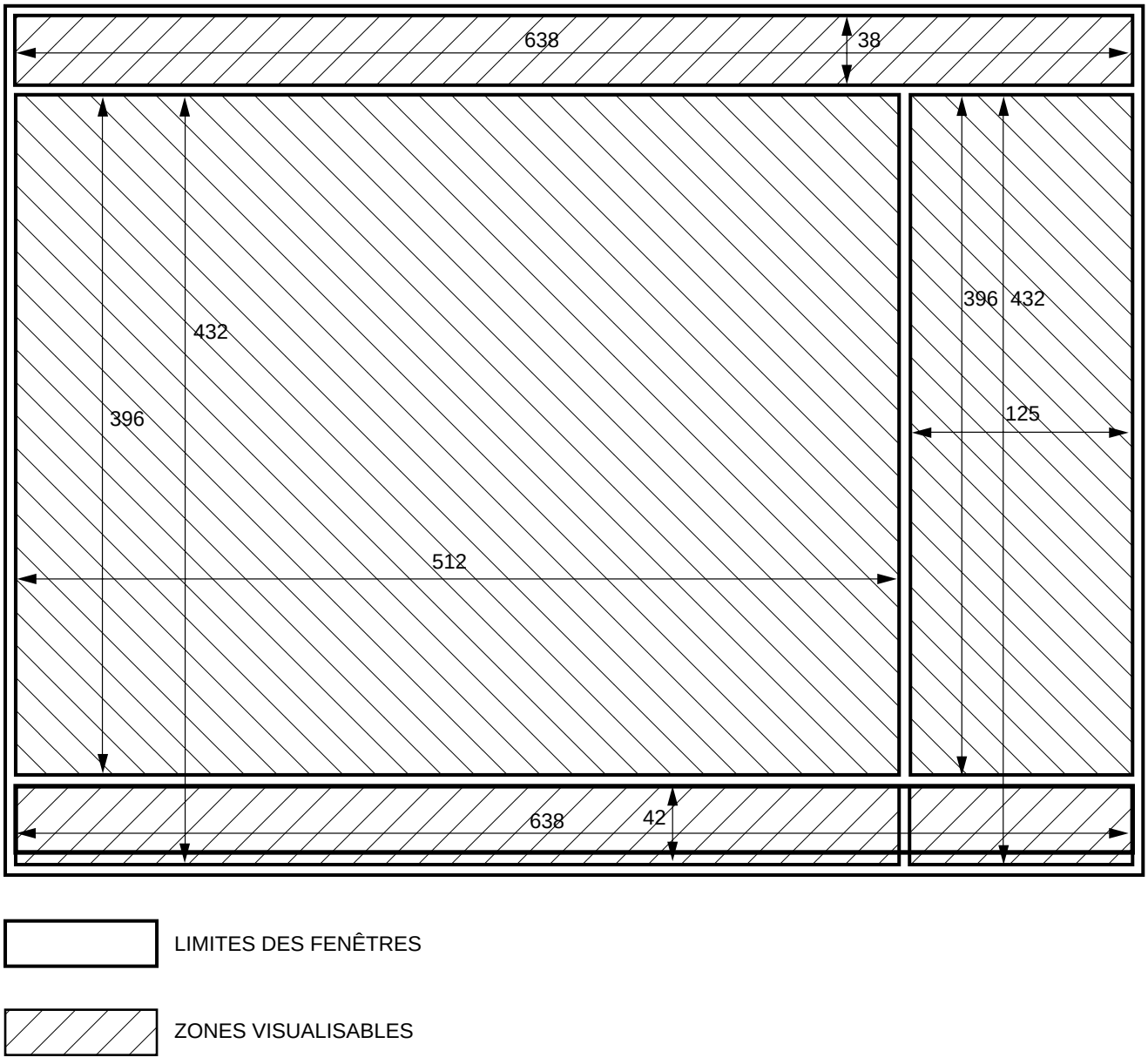
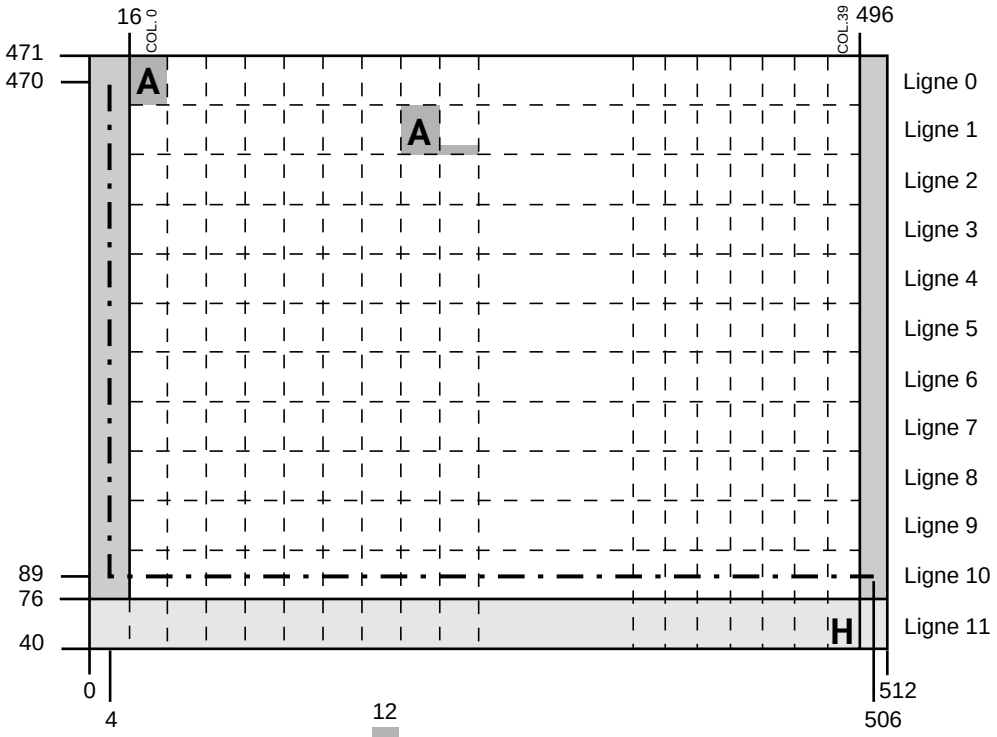
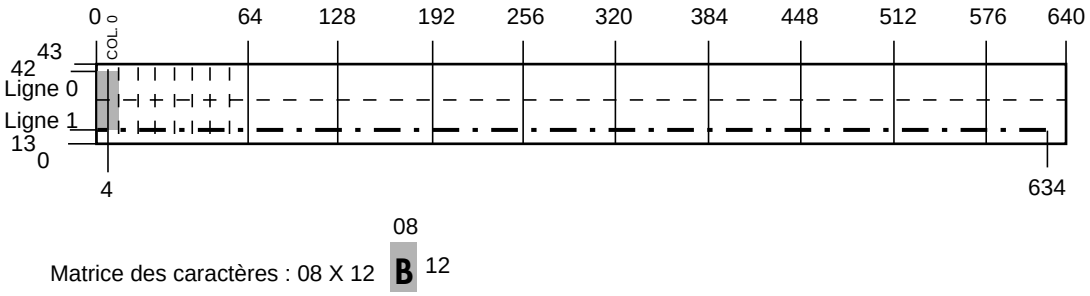


Figure 8.4 - Dimensions des fenêtres



FENETRE PRINCIPALE



FENETRE CARTOUCHE

Figure 8.5 - Positionnement des espaces «Fenêtres principale et cartouche»

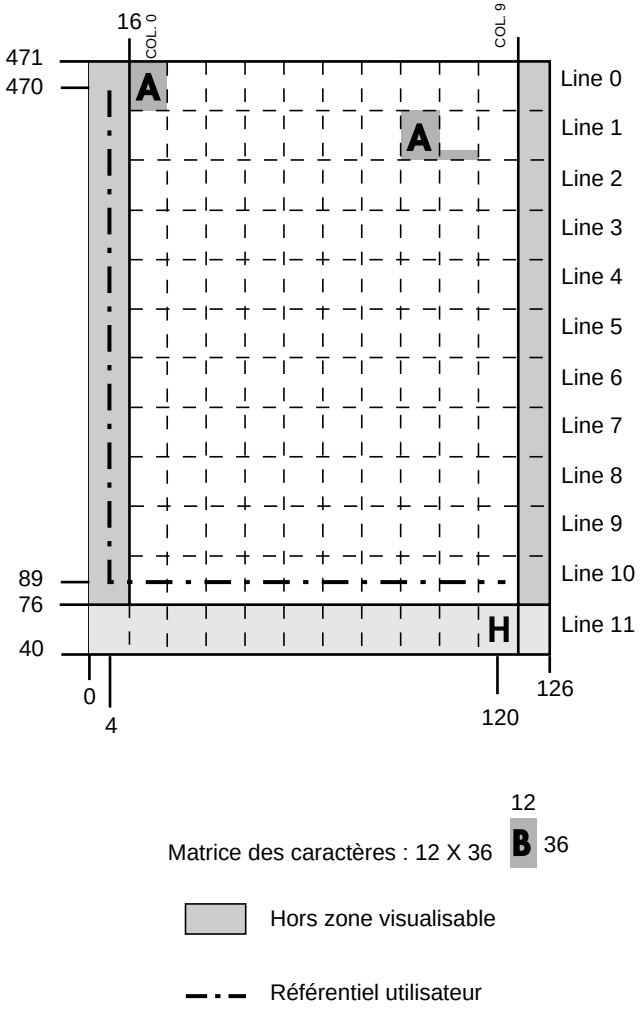


Figure 8.6 - Positionnement des espaces «Fenêtre paramètrable»

8.3.2 Définition des instructions

> **⚠ ATTENTION**
>
> Toutes les instructions sont exploitables dans la fenêtre principale, dans la fenêtre cartouche et dans la fenêtre paramètrable.
>
> L'envoi des instruction doit obligatoirement être programmé dans une tâche %TF.

8.3.2.1 Composition d'une instruction

Une instruction se compose du code de la commande suivie ou non d'expressions.

Une expression se compose d'une suite d'expressions ou d'arguments.

Un argument est un caractère ASCII.

Le caractère «LF» indique la fin des instructions.

8.3.2.2 Principe de notation

Les expressions sont représentées en majuscule, et les arguments entre guillemets.

Notation	Définition
XX YY ZZ	Ordre des expressions obligatoire
{XX YY ZZ}	Ordre des expressions quelconque
|XX YY ZZ|	Une seule des expressions est nécessaire
(XX) ...	Expression pouvant être répétée plusieurs fois
[YY]	Expression optionnelle
Valeur par défaut	Pour les expressions optionnelles, valeur prise par défaut
LF	Caractère de fin de commande (code hexadécimal 0x8A)
'0'	Caractère ASCII 0 (code hexadécimal 0xB0)

8.3.2.3 Liste des instructions

Description de l'instruction	Instruction	Voir
Initialisation soft	0x9BDD	8.3.3.1
Sélection d'une couleur	0x9BBD	8.3.3.2
Sélection de la fenêtre	0x9B2D	8.3.3.3
Caractère normal	0x9BC8	8.3.4.3
Caractère en surbrillance	0x9BC9	8.3.4.3
Caractère non souligné	0x9BCA	8.3.4.3
Caractère souligné	0x9BCB	8.3.4.3
Sélection de la couleur	0x9BBD	8.3.4.3
Caractère élargi	0x9BDB	8.3.4.3
Positionnement quelconque du curseur	0x9BBF	8.3.4.5
Définition du référentiel utilisateur	0x9BB0	8.3.5.1
Tracé référentiel utilisateur	0x9BD8	8.3.5.2

Description de l'instruction	Instruction	Voir
Tracé Utilisateur	0x9BB2	8.3.5.3
Définition d'outil	0x9BB1	8.3.5.4
Animation	0x9BDF 0x9BE7	8.3.5.5
Non animation	0x9BDE	8.3.5.6
Tracé écran	0x9BB6	8.3.5.7
Décalage origine écran	0x9BB7	8.3.5.8
Transfert point courant	0x9BE4	8.3.5.9
Icônes	0x9BB4	8.3.5.10
Chaîne de caractères référentiel écran	0x9BA8	8.3.5.11
Chaîne de caractères référentiel utilisateur	0x9B98	8.3.5.12
Remplissage zone utilisateur	0x9BA9 0x9BAA	8.3.5.13
Remplissage zone écran	0x9BAB 0x9BAC	8.3.5.14
Tracé de cartouche	0x9BBC	8.3.5.15

8.3.3 Instructions d'usage général

8.3.3.1 Initialisation soft

L'instruction 0x9BDD réalise une initialisation rapide de la visu.

Syntaxe de l'instruction

0x9BDD

Cette instruction réalise une initialisation dans les espaces alphanumérique et graphique et provoque la suppression du référentiel utilisateur.

Elle réalise dans l'espace graphique :
- effacement de l'écran,
- effacement des zones de sauvegarde,
- coordonnée décimale par défaut,
- couleur blanche par défaut.

Elle réalise dans l'espace alphanumérique :
- sélection du format A,
- couleur par défaut,
- curseur non visible,
- curseur sur la première ligne et première colonne,
- vidéo normale,
- non souligné.

8.3.3.2 Sélection d'une couleur

L'instruction 0x9BBD sélectionne une couleur parmi les 16 disponibles.

Syntaxe de l'instruction

0x9BBD COULEUR

COULEUR Code couleur (Voir tableau ci-après)

Code couleur	Code HEXA	Couleur	Pourcentage R.V.B		
			%R	%V	%B
0	0xB0	Bleu foncé	0	0	50
1	0xB1	Rouge	100	0	0
2	0xB2	Bleu	24	75	100
3	0xB3	Rose	100	50	100
4	0xB4	Vert	0	100	0
5	0xB5	Jaune	100	100	0
6	0xB6	Cyan	0	100	100
7	0xB7	Noir	0	0	0
8	0xB8	Blanc	100	100	100
9	0xB9	Brun	75	24	0
10	0xBA	Bleu clair	50	75	100
11	0xBB	Gris clair	75	75	75
12	0xBC	Gris foncé	33	33	33
13	0xBD	Orange	100	75	0
14	0xBE	Rouge/Blanc	100/100	24/100	0/100
15	0xBF	Gris clair/Blanc	75/100	75/100	75/100

8.3.3.3 Sélection de la fenêtre

L'instruction 0x9B2D permet de sélectionner la fenêtre accessible en programmation. Cette instruction est modale.

Syntaxe de l'instruction

0x9B2D NUMERO

NUMERO : | «0x1», «0x3», «0x4» | Valeur par défaut : «0x1» (Fenêtre principale).

Code HEXA	Type de fenêtre
«0x1»	Fenêtre principale
«0x4»	Fenêtre cartouche
«0x3»	Fenêtre paramètrable

8.3.4 Caractères et instructions alphanumériques

8.3.4.1 Caractères alphanumériques

Code hexa	0	1	2	3	4	5	6	7
0		├	ESP	0	@	P	`	p
1		┤	!	1	A	Q	a	q
2	CARACTERE NON CLIGNOTANT	┬	"	2	B	R	b	r
3		┴	#	3	C	S	c	s
4	CURSEUR CLIGNOTANT	┼	$	4	D	T	d	t
5	CURSEUR FIXE	┘	%	5	E	U	e	u
6	CURSEUR NON VISIBLE	┐	&	6	F	V	f	v
7	CARACTERE CLIGNOTANT	┌	'	7	G	W	g	w
8	CURSEUR VERS LA DROITE	└	(	8	H	X	h	x
9	CURSEUR VERS LA GAUCHE	→	)	9	I	Y	i	y
A	CURSEUR VERS LE BAS (LF)	←	*	:	J	Z	j	z
B	CURSEUR VERS LE HAUT		+	;	K	[	k	{
C	HOME	EFFACEMENT FENETRE	,	<	L	\	l	|
D	CURSEUR EN DE-BUT DE LIGNE (CR)	FORMAT A	-	=	M	]	m	}
E	EFFACEMENT FIN LIGNE	FORMAT B	.	>	N	^	n	~
F	EFFACEMENT FIN PAGE	FORMAT C	/	?	O	_	o	

REMARQUE : Les caractères codés 0x10 à 0x18 sont exploitables dans les fenêtres principale et paramètrable uniquement en format A. Les caractères codés 0x19 et 0x1A sont exploitables dans les fenêtres principale et paramètrable uniquement en format D.

8.3.4.2 Choix du format de police

La sélection d'un nouveau format entraîne :
- un effacement du curseur précédent,
- un affichage du nouveau curseur avec ses précédents attributs (fixe, clignotant, non visible).

Format A

0x9D		
	Taille de la police	Affichage maxi autorisé
Fenêtre principale	12x36	12 lignes de 40 caractères (dernière ligne hors zone visualisable)
Fenêtre cartouche	16x24	1 ligne de 40 caractères
Fenêtre paramétrable	12x36	12 lignes de 10 caractères (dernière ligne hors zone visualisable)

Format B

0x9E		
	Taille de la police	Affichage maxi autorisé
Fenêtre principale	06x18	24 lignes de 80 caractères (deux dernières lignes hors zone visualisable)
Fenêtre cartouche	08x12	2 lignes de 80 caractères
Fenêtre paramétrable	06x18	24 lignes de 20 caractères (deux dernières lignes hors zone visualisable)

Format C

0x9F		
	Taille de la police	Affichage maxi autorisé
Fenêtre principale	24x56	7 lignes de 20 caractères (dernières lignes hors zone visualisable)
Fenêtre cartouche	09x12	2 lignes de 71 caractères
Fenêtre paramètrable	24x56	7 lignes de 5 caractères (dernières lignes hors zone visualisable)

Format D (caractère élargi)

0x9B DB

	Taille de la police	Affichage maxi autorisé
Fenêtre principale	12x18	24 lignes de 40 caractères (deux dernières lignes hors zone visualisable)
Fenêtre cartouche	16x12	2 lignes de 40 caractères
Fenêtre paramètrable	12x18	24 lignes de 10 caractères (deux dernières lignes hors zone visualisable)

8.3.4.3 Visualisation des caractères

Ces instructions sont modales et valables quelque soit le format sélectionné.

Caractère normal

0x9BC8

Caractère en surbrillance

0x9BC9

Caractère non souligné

0x9BCA

Caractère souligné

0x9BCB

Sélection de la couleur

0x9BBD COULEUR

COULEUR Code couleur (Voir 8.3.3)

Le clignotement des caractères est assuré par les codes couleurs 14 et 15.

8.3.4.4 Visualisation du curseur

Sélection curseur fixe

0x85

Sélection curseur non visible

0x86

8.3.4.5 Déplacement du curseur

Déplacement sur le caractère suivant

0x88

Déplacement sur le caractère précédent

0x89

Déplacement sur le caractère du dessous

0x8A

Déplacement sur le caractère du dessus

0x8B

Déplacement en début d'écran

0x8C

Déplacement en début de ligne

0x8D

Positionnement quelconque du curseur

0x9BBF LIGNE COLONNE

Description

La «LIGNE» et la «COLONNE» sont définies par deux codes hexadécimaux

LIGNE Code position réelle + 0x20 = Valeur à programmer

COLONNE Code position réelle + 0x20 = Valeur à programmer

Exemple

Positionnement du curseur LIGNE 2 COLONNE 34

LIGNE 2 (3 ème ligne) : 0x2 + 0x20 = 0x22

COLONNE 34 (35 ème colonne) : 0x22 + 0x20 = 0x42

La commande à programmer est : 0x9BBF 0x22 0x42

8.3.4.6 Effacement

Les instructions d'effacement sont valables quelque soit l'espace et le format.

Effacement fin de ligne

0x8E

Effacement fin de page

0x8F

Effacement fenêtre

0x9C

8.3.5 Instructions graphiques

8.3.5.1 Définition du référentiel utilisateur

L'instruction 0x9BB0 permet à l'utilisateur de définir son propre référentiel ainsi que les caractéristiques de visualisation de celui-ci (couleur, légende .. etc...).

REMARQUES : Les bornes sont recalculées pour obtenir un facteur de conversion identique sur les deux axes.

Syntaxe de l'instruction

0x9BB0 AXE_1 AXE_2 AXE_3 AXE_4 { [FORMAT] [TRAIT] [COULEUR] } LF

AXE_1 : NOM [SIGNE] VALEUR	Nom de l'axe horizontal et valeur de la borne gauche.
NOM	Nom de l'axe
	Définit par les caractères de «A» à «Z» en majuscule ou minuscule (en général X et Y).
SIGNE	Signe de la valeur de la borne.
	Signe algébrique «+» ou «-».
	Valeur par défaut : «+».
VALEUR	Valeur de la borne de l'axe (Valeur décimale en pixels).

AXE_2 Nom de l'axe horizontal et valeur de la borne droite.

REMARQUE : De syntaxe identique à AXE_1, le nom d'AXE_2 doit être le même que celui de la borne gauche déclarée.

AXE_3 Nom de l'axe vertical et valeur de la borne basse.

REMARQUE : La syntaxe est identique à celle de AXE_1.

AXE_4 Nom de l'axe vertical et valeur de la borne haute.

REMARQUE : De syntaxe identique à AXE_1, le nom d'AXE_4 doit être le même que celui de la borne basse déclarée.

FORMAT : «.» VALEUR format des nombres décimaux, à l'affichage des bornes du référentiel. (ex : si format .3 la valeur 10000 devient 10.000 à l'affichage).

VALEUR Nombre de décimales. Cette argument s'exprime en décimal.

Valeur par défaut : 0.

TRAIT : «M» | «1», «2», «3», «4», «5» | Caractéristique du trait utilisé pour le tracé des axes. Ne modifie pas le type de trait courant.

Valeur par défaut : «1» (trait continu).

Caractère	Type de trait	Code hexa
«1»	continu	0xB1
«2»	pointillé	0xB2
«3»	tireté	0xB3
«4»	mixte	0xB4
«5»	plume levée (sans trait)	0xB5

COULEUR : «C» VALEUR Couleur des axes. Ne modifie pas la couleur courante.

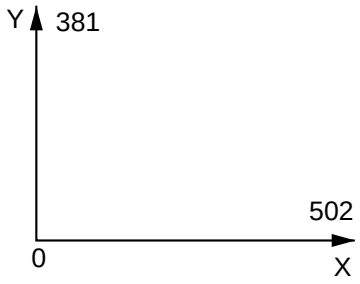
VALEUR Code couleur (Voir 8.3.3). S'exprime en décimal ou en hexadécimal.

Valeur par défaut : couleur courante au moment du tracé.

Exemples

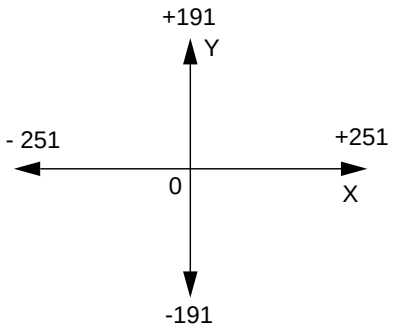
Définition d'un référentiel choisi par la fonction inig(..).

0x9BB0 X0 X502 Y0 Y381 (LF)



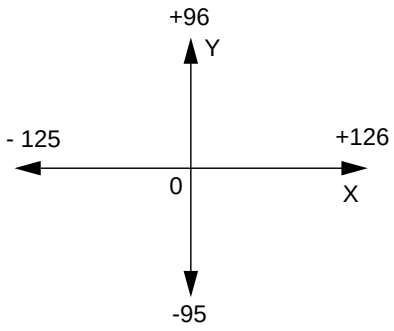
Définition d'un référentiel centré avec échelle 1.

0x9BB0 X-251 X251 Y-191 Y191 (LF)



Définition d'un référentiel centré avec échelle 2.

0x9BB0 X-125 X126 Y-95 Y96 (LF)



8

8.3.5.2 Tracé référentiel utilisateur

L'instruction 0x9BD8 permet de visualiser le référentiel utilisateur. Les indications de coordonnées sont définies par les arguments de l'instruction 0x9BB0.

Syntaxe de l'instruction

0x9BD8

8.3.5.3 Tracé Utilisateur

L'instruction 0x9BB2 permet de tracer une droite ou un arc de cercle dans le référentiel utilisateur. Le tracé se fait avec ou sans visualisation de l'outil (Voir instruction 0x9BB1, 0x9BDE, 0x9BDF).

Syntaxe de l'instruction

0x9BB2 { [LINCIR] [DECIHEXA] } { [TRAIT] [PLUME] [X] [Y] [I] [J] } LF

LINCIR : «G» | «1» «2» «3» | Définie le type de tracé (instruction non modale).

Valeur par défaut : Tracé linéaire.

Caractère	Type de tracé	Code hexa
«1»	Linéaire	0xB1
«2»	Circulaire sens anti-trigonométrique	0xB2
«3»	Circulaire sens trigonométrique.	0xB3

DECIHEXA : «G» | «4» «5» | Type de coordonnées courantes (instruction modale).

Caractère	Type de coordonnée	Code hexa
«4»	Décimale	0xB4
«5»	Hexadécimale	0xB5

REMARQUE : Cette commande est valable pour le tracé référentiel écran (9BB6).

TRAIT : «M» | «1», «2», «3», «4», «5» | Caractéristique du trait utilisé pour le tracé (Voir 8.3.5.1).

PLUME : «M» | «6» «7» «10»| Type de plume courante (instruction modale).

Caractère	Type de plume courante	Code hexa
«6»	Plume (les couleurs s'ajoutent)	0xB6
«7»	Gomme (la couleur 0 est forcée)	0xB7
«10»	Plume (la couleur demandée est forcée)	0xB1B0

X : «X» [VALEUR] Déplacement sur l'axe horizontal.

Valeur par défaut : Pas de déplacement suivant cet axe.

VALEUR Abscisse du point d'arrivée.

Valeur par défaut : «0».

Y : «Y» [VALEUR] Déplacement sur l'axe vertical.

Valeur par défaut : Pas de déplacement suivant cet axe.

VALEUR Ordonnée du point d'arrivée.

Valeur par défaut : «0».

I : «I» [VALEUR] Abscisse courante du centre (Instruction modale).

VALEUR Abscisse du centre.

Valeur par défaut : «0».

J : «J» [VALEUR] Ordonnée courante du centre (instruction modale).

VALEUR Ordonnée du centre.

Valeur par défaut : «0».

8.3.5.4 Définition d'outil

L'instruction 0x9B B1 permet de définir l'outil utilisé en animation.

Syntaxe de l'instruction

0x9BB1 | RETICULE PASTILLE FRAISE OUTIL | LF

RETICULE : «R» VALEUR [COULEUR] Définition d'un outil de forme réticule.

VALEUR Dimension d'une branche dans le référentiel utilisateur. S'exprime en décimale.

COULEUR : «C» VALEUR Couleur de l'outil (Voir 8.3.3.2).

Valeur par défaut : 8.

PASTILLE : «P» VALEUR [COULEUR] Définition d'un outil de forme pastille.

VALEUR Rayon de la pastille dans le référentiel utilisateur. S'exprime en décimal.

COULEUR : «C» VALEUR Couleur de l'outil (Voir 8.3.3.2).

Valeur par défaut : 8.

FRAISE : «F» VALEUR SENS [VALEUR] {[HAUTEUR] [COULEUR]} Définition d'un outil fraise.

VALEUR Rayon de la fraise dans le référentiel utilisateur. S'exprime en décimal.

SENS : «P» , «Q» , «R» , «S» Sens d'orientation de la fraise

Caractère	Sens d'orientation	Code hexa
«P»	X croissant	0xD0
«Q»	Y croissant	0xD1
«R»	X décroissant	0xD2
«S»	Y décroissant	0xD3

VALEUR Rayon du tore dans le référentiel utilisateur. S'exprime en décimal.

Valeur par défaut : «0».

HAUTEUR : «H» VALEUR Hauteur de la fraise dans le référentiel utilisateur. S'exprime en décimale.

Valeur par défaut : 4 fois le rayon de la fraise.

COULEUR : «C» VALEUR Couleur de l'outil (Voir 8.3.3.2).

Valeur par défaut : 8.

OUTIL : (DEPLACEMENT (XY)...) **... [COULEUR]**	Définition d'un outil type quelconque.
DEPLACEMENT : [LEVER XY] BAISSER	Déplacement sans tracé.
LEVER : «M5»	Lever de plume.
XY : { X Y }	Coordonnées du premier point de l'outil dans le référentiel utilisateur. S'expriment en décimale.
X	Abscisse du point.
Y	Ordonnée du point.
BAISSER : «M1»	Baisser de plume.
XY	Coordonnées d'un point de l'outil en décimale dans le référentiel utilisateur.
COULEUR : «C» VALEUR	Couleur de l'outil (Voir 8.3.3.2). Valeur par défaut : 0xB8.

8.3.5.5 Animation

Les instructions 0x9BDF ou 0x9BE7 sélectionnent le tracé utilisateur avec visualisation de l'outil. Les caractéristiques de visualisation sont définies dans la commande 0x9BB1 (Voir 8.3.5.4).

Syntaxe de l'instruction

0x9BDF ou 0x9BE7

8.3.5.6 Non animation

L'instruction 0x9BDE sélectionne le tracé utilisateur sans visualisation de l'outil.

Syntaxe de l'instruction

0x9BDE

8.3.5.7 Tracé écran

L'instruction 0x9BB6 permet le tracé d'une droite ou d'un arc de cercle dans le référentiel écran.

Syntaxe de l'instruction

0x9BB6 {[LINCIR] [DECIHEXA] } { [TRAIT] [PLUME] [X] [Y] [I] [J] } LF

REMARQUE : La syntaxe est rigoureusement identique à la commande 0x9BB2 (Voir 8.3.5.3).

8.3.5.8 Décalage origine écran

L'instruction 0x9BB7 permet de décaler l'origine du référentiel écran.

REMARQUE : L'autre borne est recalculée de façon à ne pas modifier la taille du référentiel.

Syntaxe de l'instruction

0x9BB7 [DECIHEXA] { [X] [Y] } LF

DECIHEXA	Sélection coordonnées en décimal ou hexadécimal. Ne modifie pas le type de coordonnées courant.
	Valeur par défaut : décimal.
X	Valeur du décalage horizontal du référentiel écran.
	Valeur par défaut : Conserve le précédent décalage horizontal.
Y	Valeur du décalage vertical du référentiel écran.
	Valeur par défaut : Conserve le précédent décalage vertical. Transfert point courant.

Exemple

Décalage du référentiel de 100 pixels sur l'axe X et de 200 pixels sur l'axe Y.

0x9BB7 X100 Y200 (LF)

8.3.5.9 Transfert point courant

L'instruction 0x9BE4 permet de faire coïncider le point courant du référentiel écran avec le point courant du référentiel utilisateur.

Syntaxe de l'instruction

0x9B E4

8.3.5.10 Icônes

L'instruction 0x9BB4 permet le tracé d'une icône (symbole) de dimension constante ou paramétrable, prise dans un repère orienté comme le référentiel utilisateur et dont l'origine correspond à la position courante du tracé.

Syntaxe de l'instruction

0x9BB4 NUMERO [SUITE_PARAM] { [TRAIT] [PLUME] [COULEUR] } LF

NUMERO : «N» VALEUR	Numéro de l'icône.
REMARQUE : Toutes les valeurs ne sont pas significatives (Voir Figure 8-7).	
SUITE_PARAM : (PARAM)...	Paramètrage de l'icône. Le nombre de paramètres est variable et dépend du numéro d'icône. L'ordre d'écriture des paramètres est important (P0, P1, P2, ..., Pn).
	Valeur par défaut : Table des paramètres en coordonnées écran.
PARAM chaîne : «P» [VALEUR]	Chaîne de paramètre (de P0 à Pn).
VALEUR	Valeur du paramètre en décimal dans le référentiel utilisateur.
	Valeur par défaut : «0».
TRAIT : «M» | «1», «2», «3», «4», «5» |	Caractéristique du trait utilisé pour le tracé (Voir 8.3.5.1).
PLUME : «M» | «6» «7» «10» |	Type de plume courante (Voir 8.3.5.3).
COULEUR : «C» VALEUR	Couleur de l'icône (Voir 8.3.3.2).
	Valeur par défaut : Couleur courante.

Exemple

Tracé d'un cercle de rayon 20 et de couleur rouge.

0x9BB4 N38 P20 M1 M6 C1 (LF)

⚠ ATTENTION

Le tracé d'icones nécessite d'avoir défini un référentiel utilisateur (Commande 0x9BB0 ou fonction inig(..)).

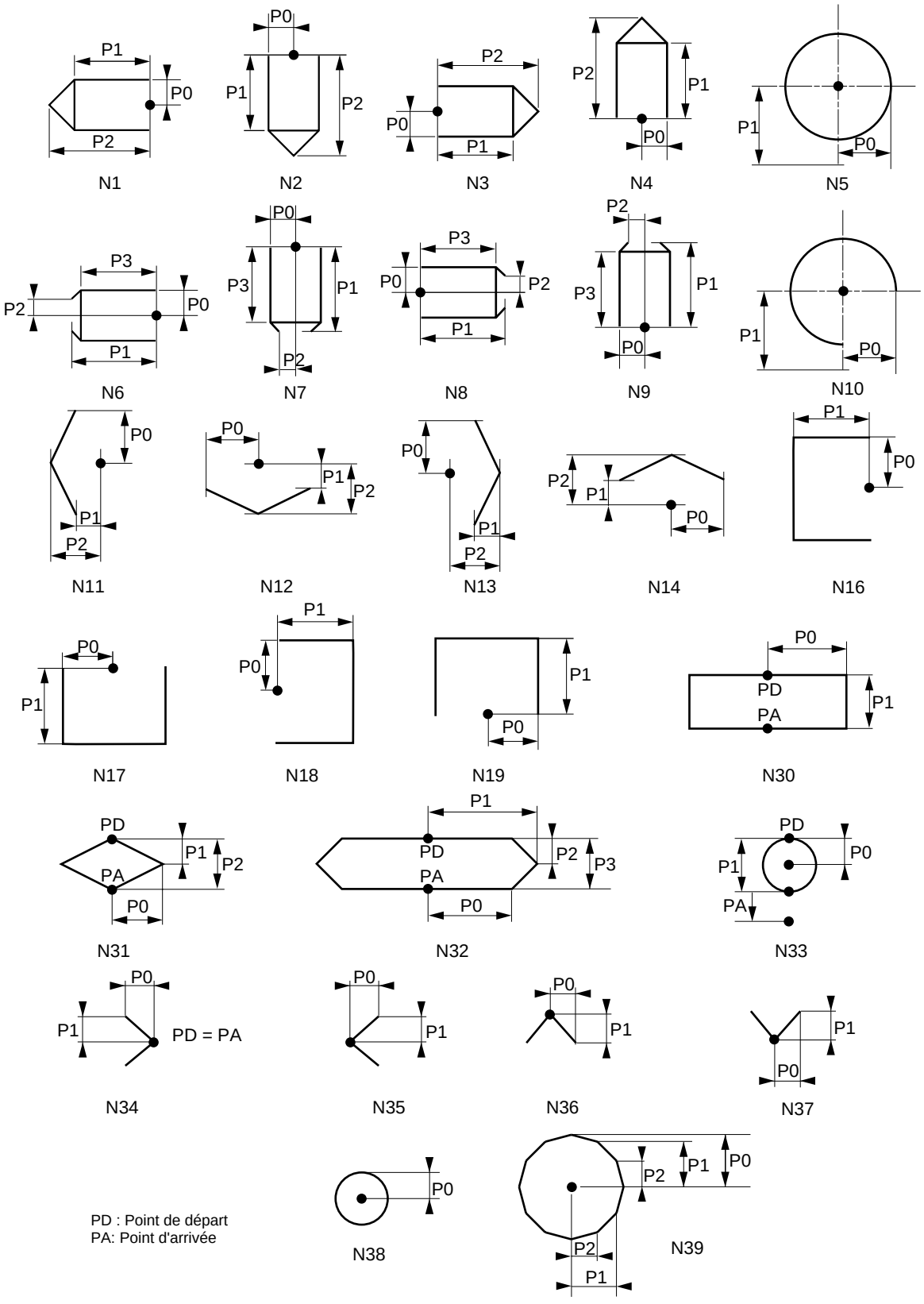


Figure 8.7 - Définition des icônes

8.3.5.11 Chaîne de caractères référentiel écran

L'instruction 0x9BA8 permet d'afficher une chaîne de caractères alphanumériques au point courant écran.

Syntaxe de l'instruction

0x9BA8 POLICE CHAINE LF

POLICE Numéro de police.

Caractère	Type de police	Code hexa
«0»	Police 6 x 18	0xB0
«1»	Police 12 x 18	0xB1
«2»	Police 12 x 36	0xB2
«3»	Police 24 x 56	0xB3
«4»	Police 8 x 12	0xB4
«5»	Police 9 x 12	0xB5
«6»	Police 6 x 12	0xB6
«7»	Police 16 x 24	0xB7

CHAINE Tous les caractères alphanumériques autorisés dans la police.

8.3.5.12 Chaîne de caractères référentiel utilisateur

L'instruction 0x9B98 permet d'afficher une chaîne de caractères alphanumériques au point courant utilisateur.

Syntaxe de l'instruction

0x9B98 POLICE CHAINE LF

POLICE Numéro de police (Voir).

CHAINE Tous les caractères alphanumériques autorisés dans la police.

8.3.5.13 Remplissage zone utilisateur

Les instructions 0x9BA9 et 0x9BAA permettent de remplir une zone du référentiel utilisateur.

Le remplissage s'arrête si :
- la limite de la fenêtre est atteinte,
- la borne du clipping (coupure) est atteinte,
- la couleur de remplissage est rencontrée.

Syntaxe de l'instruction

0x9BA9 { [CLIP] [CLIP] [CLIP] [CLIP] [X] [Y] [COUL-CONT] } LF
0x9BAA { [CLIP] [CLIP] [CLIP] [CLIP] [X] [Y] [COUL-CONT] } LF

⚠ ATTENTION

Les arguments [CLIP], [X] et [Y] sont affectés des signes «+» et «-».
Le signe + est codé par le caractère ASCII «0».
Le signe - est codé par le caractère ASCII «?».

CLIP : | «G» «D» «H» «B» | [VALEUR] Sélection des bornes de «clipping»

Valeur par défaut : pas de clipping.

Caractère	Sélection des bornes	Code hexa
«G»	Gauche	0xC7
«B»	Bas	0xC2
«H»	Haut	0xC8
«D»	Droit	0xC4

VALEUR — Valeur de la borne en hexadécimal dans le référentiel utilisateur.

Valeur par défaut : «0».

X — Abscisse d'un point compris dans la zone en hexadécimal dans le référentiel utilisateur.

Valeur par défaut : abscisse du point courant.

Y — Ordonnée d'un point compris dans la zone en hexadécimal dans le référentiel utilisateur.

Valeur par défaut : ordonnée du point courant.

COUL-CONT | "C" "c"| [VALEUR] Couleur du contour

Caractère	Définition	Code hexa
«C»	Recherche du contour dans les quatre plans (Arrêt sur couleur exacte)	0xC3
«c»	Recherche du contour dans les plans relatifs à la couleur (Arrêt sur une composante de la couleur)	0xE3

Valeur par défaut : couleur courante et «C» sélectionné.

REMARQUE : Les couleurs sont codées sur 4 bits. Une composante de la couleur choisie est une couleur avec les même bits à 1 que la couleur choisie (ex : Si la couleur choisie est Jaune «c5» (soit 0101 en binaire), les composantes de la couleur sont noir (soit 0111 en binaire), orange(soit 1101 en binaire) et gris clair/blanc (soit 1111 en binaire)).

Exemple

Remplissage d'un rectangle rouge

0x9BA9 G09 D0100 B0120 H0120 X050 Y0110 C1

8.3.5.14 Remplissage zone écran

Les instructions 0x9BAB et 0x9BAC permettent de remplir une zone du référentiel écran.

Syntaxe de l'instruction

0x9BAB { [CLIP] [CLIP] [CLIP] [CLIP] [X] [Y] [COUL-CONT] } LF **0x9BAC { [CLIP] [CLIP] [CLIP] [CLIP] [X] [Y] [COUL-CONT] } LF**

REMARQUE : la syntaxe est rigoureusement identique à celle de la commande 9BA9 (Voir 8.3.5.13). Les coordonnées sont dans le référentiel écran.

8.3.5.15 Tracé de cartouche

L'instruction 0x9BBC permet de séparer verticalement la fenêtre en 10 zones.

Syntaxe de l'instruction

0x9BBC LF

REMARQUE : Bien qu'étant utilisable dans toutes les fenêtres, cette instruction n'a de sens que dans la fenêtre cartouche.

9 Entrées/sorties analogiques

9.1 Généralités

Un maximum de 18 CNA et 20 CAN sont disponibles sur les système NUM 1060. Les entrées/sorties analogiques sont accessibles par le programme utilisateur, par le programme de pièce ou les opérateurs dynamiques. Elle sont réparties comme suit :

	Carte processeur machine	Carte 8E/8S analogique (2 cartes maxi)	Carte UCSII
1060 série I	4 CAN - 2 CNA	8CAN - 8CNA	
1060 série II	4 CAN - 2 CNA	8CAN - 8CNA	
1060 série II			2CAN - 1CNA

Les E/S analogiques sont repérées géographiquement par le N° d'emplacement de la carte et le N° de voie dans la carte. Le codage se fait sur un octet.

Les bits 0 à 3 de l'octet codent le N° de voie (0 à 7). Les bits 4 à 7 codent le N° de carte.

L'unité centrale est numérotée 1.

Loi d'évolution

Les entrées et sorties analogiques sont des valeurs signées sur 16 bits. La loi d'évolution de ces valeurs est :
- pour les valeurs positives :
  de 0 --> 0x7FFF
  pour 0^+ --> n Volts (Avec n : valeur de la pleine échelle)
- pour les valeurs négatives :
  de 0xFFFF --> 0x8000
  pour 0^- --> -n Volts (Avec n : valeur de la pleine échelle)

Ceci est vrai quelquesoit la résolution du CAN ou CNA (8 ou 12 bits).

La pleine échelle dépend des caractéristiques de la carte utilisée (Voir Manuel d'installation et de mise en oeuvre).

La précision dépend du format du CNA ou CAN utilisé (8 bits, 8 bits + signe ou 12 bits + signe).

9

9.2 Configuration des cartes E/S analogiques — anas

Syntaxe de l'instruction

anas(cv, wconfig)

cv : Octet désignant la carte (la voie est non significative).

config : Configuration codée sur 16 bits.

Description

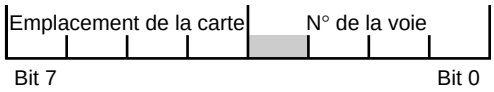
Cette fonction permet la configuration du nombre d'entrées analogiques utiles (1 ou 8) et du gain de chaque entrée (1 ou 10).

La période de rafraîchissement interne de chaque entrée analogique est de 1,36 ms lorsque les 8 entrées sont configurées et de 0,170 ms lorsque une seule entrée est configurée (l'entrée 0).

Cette fonction est facultative. A l'initialisation du système les cartes sont configurées en 8 entrées avec gain de 1.

REMARQUE : Cette fonction concerne uniquement les Entrées/Sorties analogiques des cartes additionnelles.

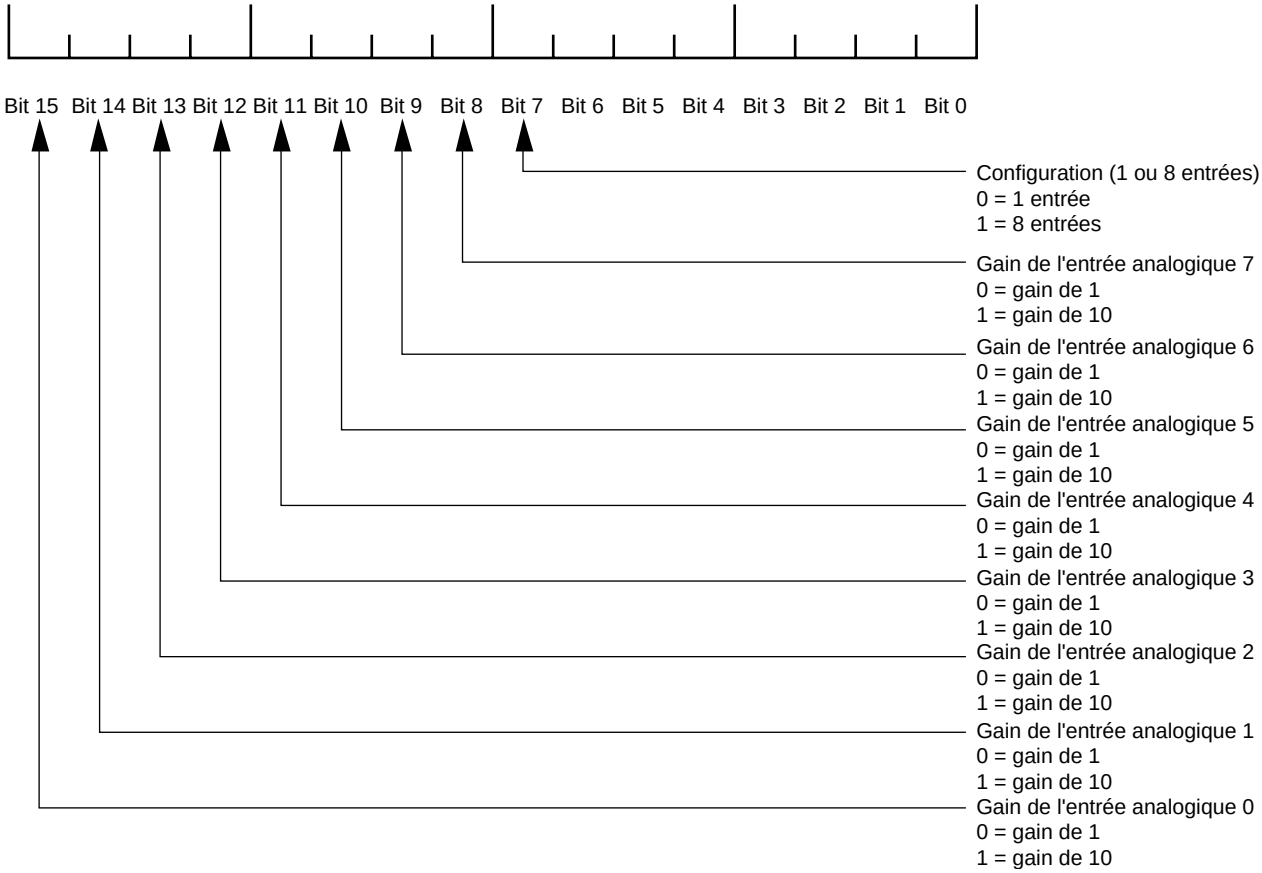
Détail de l'argument «cv»



Se reporter au «Manuel d'installation et de mise en oeuvre» pour définir l'emplacement de la carte sur le bus système.

Les valeurs 0 à 7 correspondent aux numéros de voie des huit registres d'entrées ou des huit registres de sorties.

Format du mot de configuration



Code retourné

Si OK

0

Si défaut

1 :	Carte absente.
2 :	Paramètre carte érroné.

9.3 Ecriture d'une sortie analogique — anao

Syntaxe de l'instruction

anao(cv, woutput)

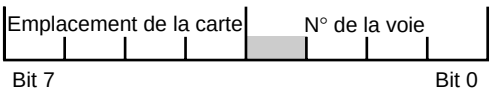
cv : Adresse de la sortie codée sur un octet.

woutput : Valeur entière signée sur 16 bits.

Description

Ecriture du CNA N°v de la carte N° c.

Détail de l'argument «cv»



Se reporter au «Manuel d'installation et de mise en oeuvre» pour définir l'emplacement de la carte sur le bus système.

Les valeurs 0 à 7 corespondent aux numéros de voie des huit registres d'entrées ou des huit registres de sorties.

Exemple : Programmation de la vitesse de broche pilotée par automate

Lire la fonction auxiliaire du groupe (M3 ou M4) donnant le sens de rotation de la broche:

- M03_g = 1 : antitrigonométrique
- M04_g = 1 : trigonométrique

Lire le module de la vitesse de broche (VITBRb); la valeur du module varie de 0 (vitesse nulle) à 0x7FFF (vitesse maximum).

Envoyer au CNA la valeur codée signée sur 16 bits, le signe dépendant du cablage du variateur de broche et de la fonction auxiliaire du groupe:

- si la valeur est positive ou nulle, ANAO(cv,VITBRb),
- si la valeur est strictement négative, ANAO(cv,~VITBRb).

Code retourné

Si OK

0

Si défaut

1 : Carte absente.

2 : Paramètre carte érroné.

3 : Paramètre voie erroné.

9.4 Lecture d'une entrée analogique anai

Syntaxe de l'instruction

anai(cv, &winput)

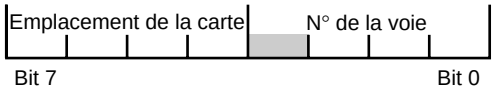
cv : Adresse de l'entrée (codée sur un octet).

&winput : Adresse d'une variable (%Mxxx. W ou %Vxxx. W) qui va recevoir la valeur de l'entrée sur 16 bits signées.

Description

Lecture d'une entrée analogique.

Détail de l'argument «cv»



Se reporter au «Manuel d'installation et de mise en oeuvre» pour définir l'emplacement de la carte sur le bus système.

Les valeurs 0 à 7 corespondent aux numéros de voie des huit registres d'entrées ou des huit registres de sorties.

Exemple

anai(0x37, %V100.&) Lecture de l'entrée N°7 de la carte N°3. Le résultat est transféré dans %V100.W.

Code retourné

Si OK

0

Si défaut

1 : Carte absente.

2 : Paramètre carte érroné.

3 : Paramètre voie erroné.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&winput" incorrect.

9.5 Redirection d'une carte analogique — anaa

Syntaxe de l'instruction

anaa(cv_initial, cv_futur)

cv_initial : Octet codant le N° de carte (la voie est non significative).

cv_futur : Octet codant le N° de carte (la voie est non significative).

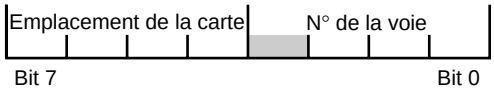
Description

Redirection d'une carte analogique.

Cette fonction, facultative, permet de rediriger les fonctions anas(cv_initial, ..), anao(cv_initial, ...), anai(cv_initial, ..) vers la carte cv_futur.

> ⚠ **ATTENTION**
>
> Cette fonction est autorisée uniquement dans la tâche %INI.

Détail des arguments «cv-initial» et «cv-futur»



Se reporter au «Manuel d'installation et de mise en oeuvre» pour définir l'emplacement de la carte sur le bus système.

Les valeurs 0 à 7 corespondent aux numéros de voie des huit registres d'entrées ou des huit registres de sorties.

Code retourné

Si OK

0

Si défaut

1 : Carte finale absente

2 : Paramètre carte incorrect

4 : Fonction appelée dans une tâche autre qu'un %INI

10 Lecture/Ecriture explicites des cartes Entrées/Sorties

10.1 Généralitées

Le programmeur à la possibilité d'accéder immédiatement aux entrées/sorties sur le bus série (SB), sans attendre leur rafraîchissement par le moniteur.

REMARQUE : Cette fonctionnalité doit être réservée aux cas prioritaires car elle est coûteuse en temps CPU.

10.2 Lecture explicite d'une carte entrée — read_i

Syntaxe de l'instruction

read_i(rcmv, n)

rcmv : Mot codant, le rack , la carte , le module et la voie.

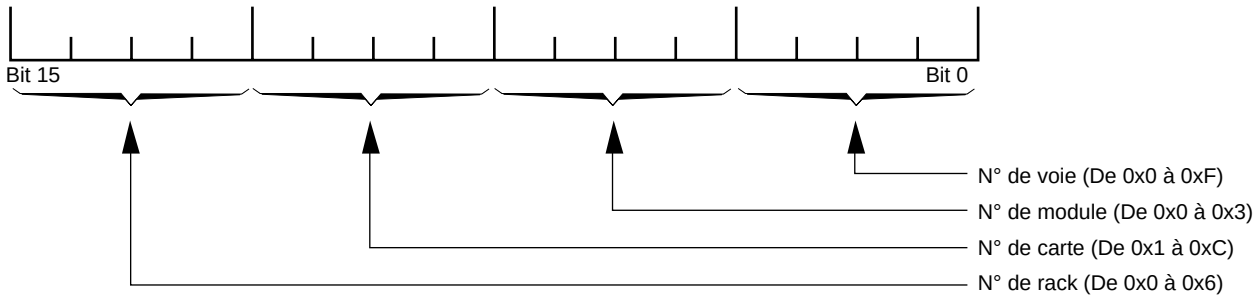
n : Nombre d'octets à lire.

Fonctionnement

Le système vérifie la cohérence des paramètres rcmv et n.

Le système lit immédiatement la carte rc et met à jour la zone image %Ircmv à %Ircmv+n.

Détail de l'argument «rcmv»



Exemples

read_i(0x6b10, 1) Provoque le rafraîchissement de %I6b10.B

read_i(0x6b10, 2) Provoque le rafraîchissement de %I6b10.W

read_i(0x6b10, 4) Provoque le rafraîchissement de %I6b10.L

Code retourné

Si OK

0 : Lecture OK.

Si défaut

1 : Les variables demandées dépassent les limites de la carte. Il y a néammoins échange après troncature aux limites autorisées dans la carte.

2 : Demande d'accès à une carte absente.

3 : Paramètre rcvm en dehors des limites

-1 : Défaut dans l'échange sur le bus.

10.3 Ecriture explicite d'une carte sortie — write_q

Syntaxe de l'instruction

write_q(rcmv, n)

rcmv : Mot codant, le rack, la carte, le module-et la voie.

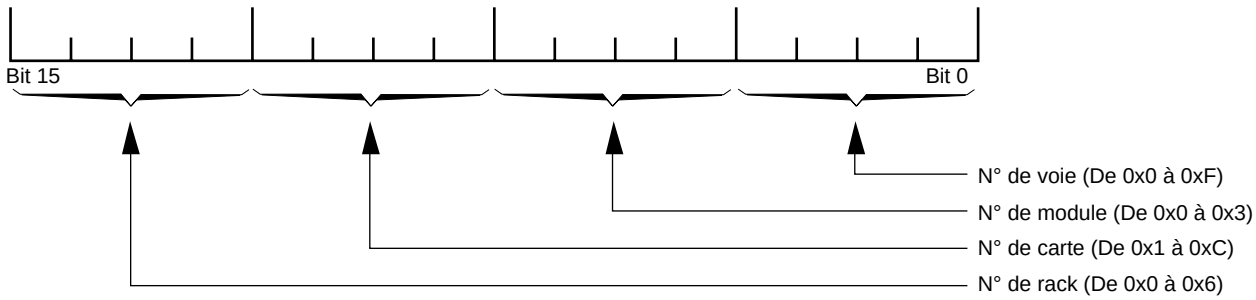
n : Nombre d'octets à écrire.

Fonctionnement

Le système vérifie la cohérence des paramètres rcmv et n.

Le système écrit immédiatement la zone image %Qrcmv à %Qrcmv+n dans la carte rc.

Détail de l'argument «rcmv»



⚠ ATTENTION

L'instruction provoque une écriture rapide des sorties choisies à la condition que celles-ci aient été écrites précédemment.

Exemples:

%QB04.B= 0xAA	
write_q(0xb04, 1)	provoque l'écriture immédiate de %Qb04.B avec 0xAA
%QB04.W= 0xAAFC	
write_q(0xb04, 2)	Provoque l'écriture immédiate de %Qb04.W avec 0xAAFC
%QB04.L= 0xAAFC0000	
write_q(0xb04, 8)	Provoque l'écriture immédiate de %Qb04.L et %Qb08.L avec 0xAAFC0000

Code retourné

Si OK

0 :	Ecriture OK.

Si défaut

1 :	Les variables demandées dépassent les limites de la carte. Il y a néammoins échange après troncature aux limites autorisées dans la carte.
2 :	Demande d'accès à une carte absente.
3 :	Paramètre rcvm en dehors des limites
-1 :	Défaut dans l'échange sur le bus.

11 Entrées interruptions

11.1 Généralités

La fonction automatisme traite les interruptions prioritaires sur les lignes :

	Carte processeur machine	Carte IT/Ligne série (2 cartes maxi)	Carte UCSII
1060 série I	iti0 à iti3	iti4 à itiB	
1060 série II	iti0 à iti3	iti4 à itiB	
1060 série II			iti0

Les interruptions prioritaires sont associée :
- à la fonction iti_gr() pour prise de cote à la volée,
- à la fonction thiti() pour dérouter les tâches périodiques du programme utilisateur et exécuter une routine d'interruption programmée dans une tâche hard %TH.

11.1.1 Prise de cote à la volée

Ces interruptions sont prises en compte par la fonction G10 dans le programme pièces (Voir «Manuel de programmation»).

Une interruption émise sur une des lignes iti0 à itiB est traitée par la fonction automatisme. Dés l'émission de l' IT, le moniteur informe la fonction CN de rafraîchir les paramètres externes E70001 à E78001 (référence de position d'un axe d'un groupe sur prise de cote au vol).

Ces interruptions externes, dédiées au palpage, sont paramétrables par la fonction iti-gr().

La gestion des interruptions doit être programmée dans %TS0.

REMARQUES : En multigroupe d'axes, si deux interruptions arrivent en même temps, l'interruption affectée à la ligne iti0 est la plus prioritaire, l'interruption affectée à la ligne itiB est la moins prioritaire.
Sur une même ligne, si une interruption affectée au groupe d'axes 1 arrive en même temps qu'une interruption affectée au groupe d'axes 8, c'est l'interruption affectée au groupe 1 qui sera traitée en priorité par le moniteur.

11.1.2 Interruptions affectées à une tâche %TH

L'affectation d'une ligne iti0 à itiB à une tâche hard provoque l'exécution de la routine d'interruption programmée dans la tâche %TH.

⚠ ATTENTION

Dans le cas ou une tâche hard et un palpage sont affectés et programmés sur la même ligne, le programme utilisateur est dérouté après la prise de cote effective par le moniteur.

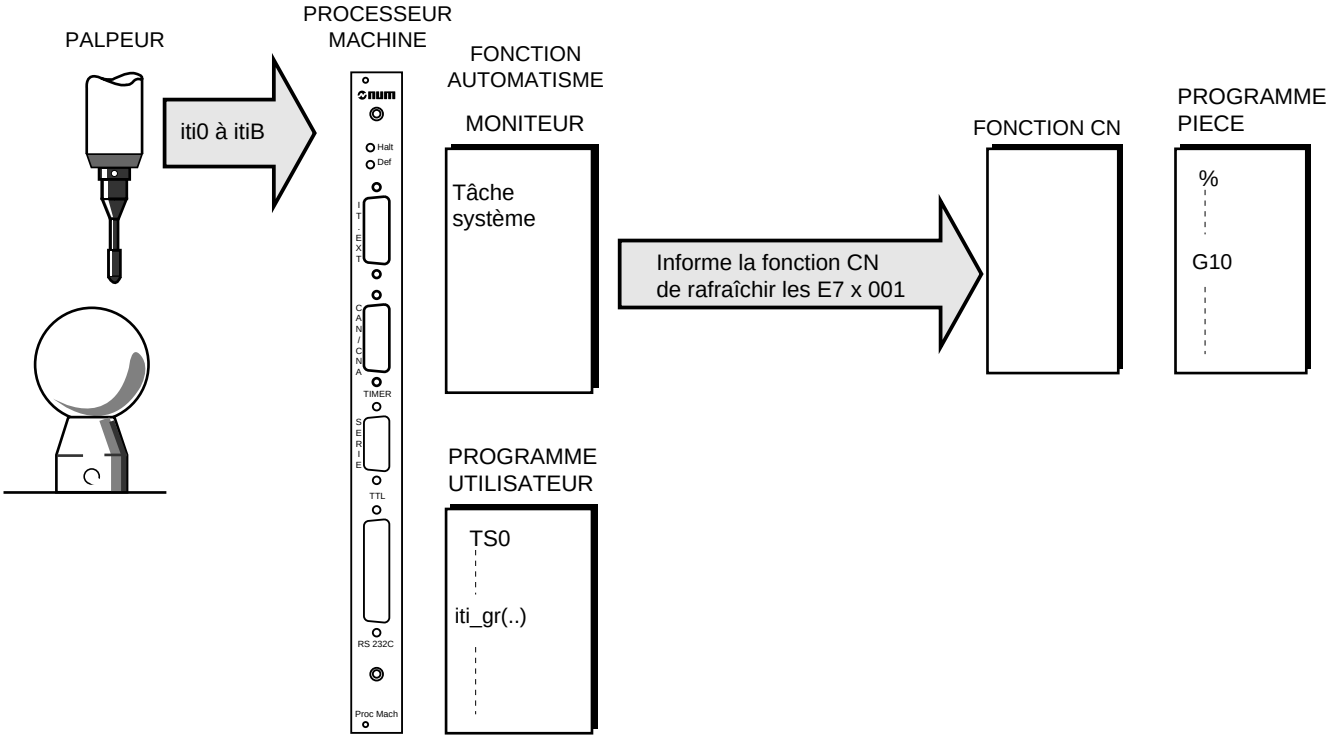


Figure 11.1 - Traitement pour prise de cote à la volée

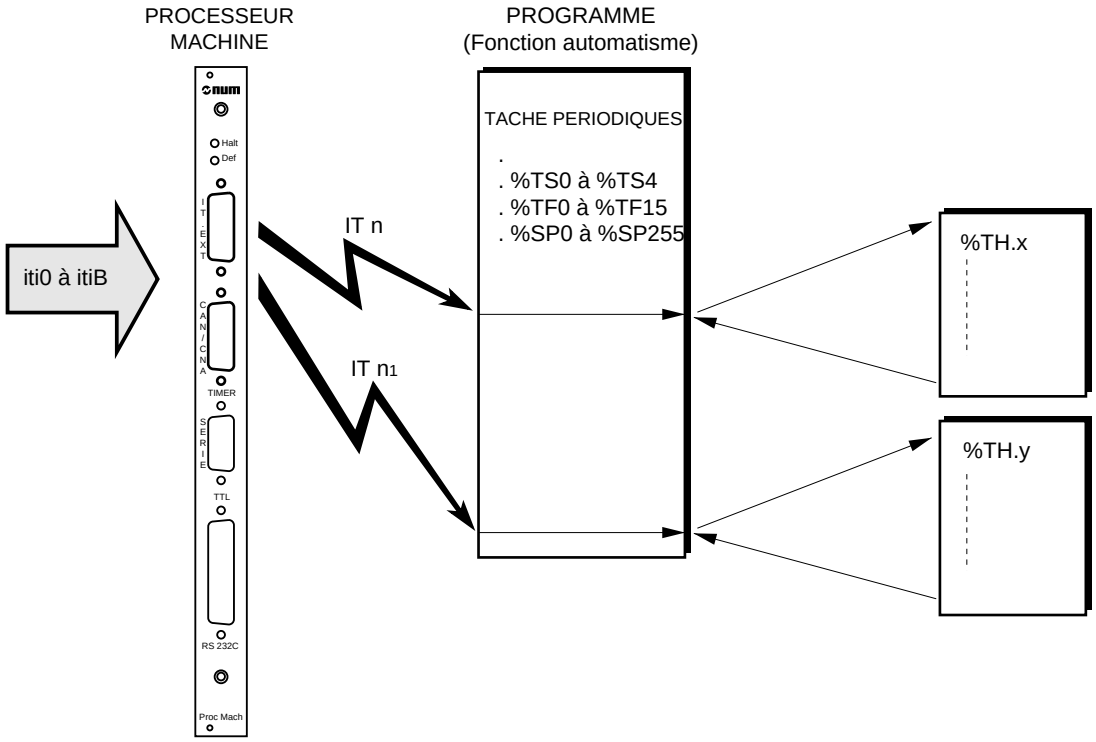


Figure 11.2 - Interruptions affectées à une TH

11.2 Principe d'affectation des lignes

NUM 1060 série I et NUM 1060 série II (multicarte)

Les numéro d'entrées interruptions sont affectées par ordre croissant :
- sur la carte processeur machine,
- sur la première carte IT/Lignes séries rencontrée dans le rack,
- sur la seconde carte IT/Lignes séries rencontrée dans le rack.

N° broches			N° entrée interruption	Type de carte
5V	24V	Commun		
1	2	9	Ligne 0	Carte processeur machine
10	11	3	Ligne 1	Carte processeur machine
4	5	12	Ligne 2	Carte processeur machine
13	14	6	Ligne 3	Carte processeur machine
1	2	5	Ligne 4	Première carte IT/Lignes séries
3	4	5	Ligne 5	Première carte IT/Lignes séries
6	7	5	Ligne 6	Première carte IT/Lignes séries
8	9	5	Ligne 7	Première carte IT/Lignes séries
1	2	5	Ligne 8	Seconde carte IT/Lignes séries
3	4	5	Ligne 9	Seconde carte IT/Lignes séries
6	7	5	Ligne A	Seconde carte IT/Lignes séries
8	9	5	Ligne B	Seconde carte IT/Lignes séries

NUM 1060 série II (UCSII)

Une seule ligne est disponible :

N° broches			N° entrée interruption	Type de carte
5V	24V	Commun		
6	1	2	Ligne 0	Carte UCSII

11.3 Association entrées interruptions/ groupes d'axes iti_gr

Syntaxe de l'instruction

iti_gr(n_iti, groupe)

n_iti : Numéro de l'entrée interruption (de 0 à 0xB).

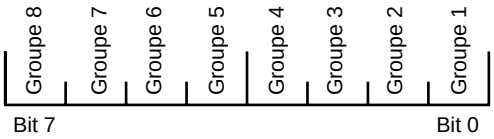
groupe : Liste de bits indiquant les groupes d'axes impliqués dans cette interruption.

Cette fonction permet d'associer une entrée IT avec un (des) groupe(s) d'axes.

Fonctionnement

Une interruption sur l'entrée provoquera la lecture par le moniteur de tous les coupleurs des axes constitutifs des groupes déclarés. Le moniteur signale ensuite l'occurrence d'une lecture d'axeà la fonction CN et ceci groupe d'axe par groupe d'axe.

Détail de l'argument «groupe»



Bit à 0 = groupe invalidé
Bit à 1 = groupe validé

Code retourné

Si OK

0

Si défaut

-1 : n_iti non compris entre 0 et 0xB

11.4 Configuration d'une entrée interruption — itictl

Syntaxe de l'instruction

itictl (n_iti, iti_config)

n_iti : Numéro d'une entrée interruption (De 0 à 0xB).

iti_config : Valeur codée de configuration du composant.

Permet la configuration d'une entrée interruption.

Fonctionnement

La configuration du composant, codée sur 8 bits, est transmise dans le paramètre iti_config.

Après détection du changement d'état, la fonction automatisme attend la stabilité du signal avant de prendre en compte le changement d'état.

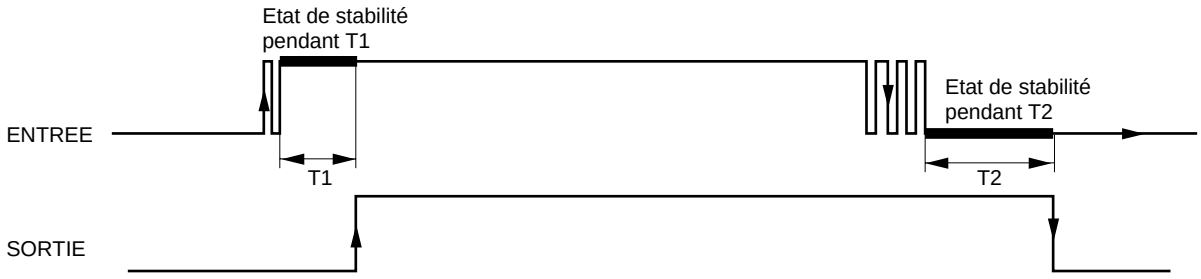
Les possibilités de choix de filtrage se font indépendamment du front actif.

Le temps de filtrage correspond au temps de stabilité avant prise en compte.

La durée du filtrage est paramétrable avec les valeurs suivantes :
- 0,5 ms,
- 1 ms,
- 4 ms,
- 8 ms,
- 1 à 3 ms en cycle rapide (sans filtrage).

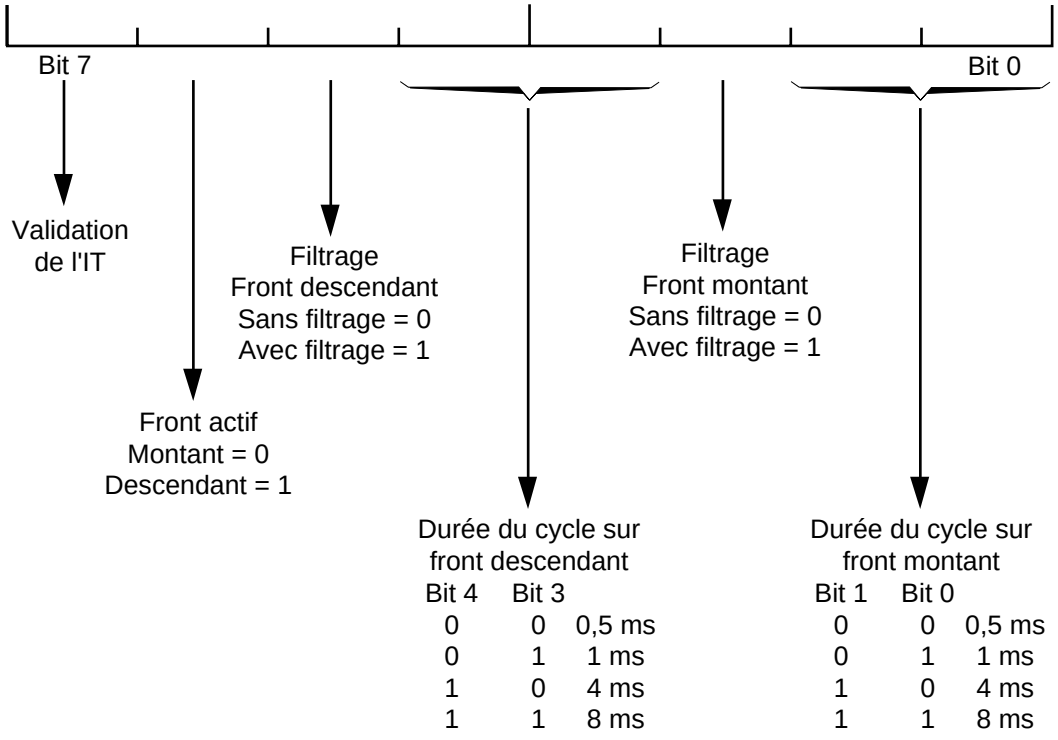
Exemple

Filtrages sur front montant T1 et sur front descendant T2.



ENTREE : signal émis par le palpeur
SORTIE : signal envoyé pour traitement aprés filtrage

Détail du paramètre iti_config



Code retourné

Si OK

0

11.5 Lecture d’une entrée interruption

itiget

Syntaxe de l’instruction

itiget(n_iti)

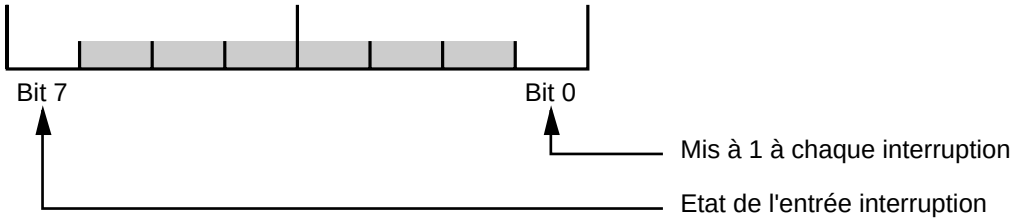
n_iti : Numéro d’une entrée interruption (De 0 à 0xB).

Description

itiget() retourne le registre d’état de l’entrée interruption.

Ce paramètre est mis à jour à chaque cycle HTR par le moniteur. Le retard avec l’état réel de la ligne est au maximum de 20 ms

Détail du registre



L’appel de itiget() provoque la mise à 0 du bit 0 du registre (mémorisation de l’occurrence d’une IT).

Code retourné

Si OK

Registre d’état de l’entrée

Si défaut

0 : n_iti non compris entre 0 et 0xB

11.6 Association tâche %TH avec une entrée IT

thiti

Syntaxe de l'instruction

thiti(numéro_th, n_iti)

numéro_th : Numéro de la tache %TH.

n_iti : Numéro de l'entrée interruption (De 0 à 0xB).

Permet d'associer la tâche %TH avec une entrée interruption.

Fonctionnement

Lorsque l'entrée IT provoque une interruption électronique, le système appelle la tâche %TH associée.

Code retourné

Si OK

0

12 Lignes séries

12.1 Généralitées

La fonction automatisme pilote 12 lignes séries réparties comme suit :

	Carte processeur CN	Carte processeur machine	Carte IT/Ligne série (2 cartes maxi)	Carte UCSII
1060 série I	DNC - PERIPH	RS232C - TTL	Ligne 1 à Ligne 4	
1060 série II		RS232C - TTL	Ligne 1 à Ligne 4	
1060 série II				COMM 1 - COMM 2

Le programme utilisateur peut gérer le chargement et le déchargement d'informations avec un périphérique dans le cadre d'application spécifique.

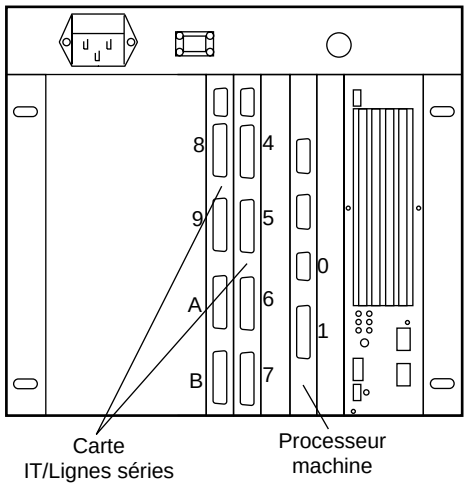
Pour plus de précision sur l'installation de ces lignes, se reporter au «Manuel d'installation et de mise en oeuvre».

Principe de numérotation des lignes

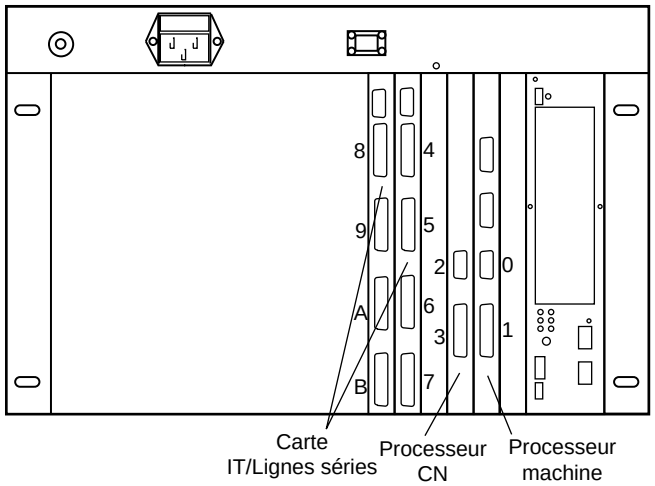
NUM 1060 série I et NUM 1060 série II (Multicarte)

Le numéro des lignes est figé sur les cartes processeur CN et processeur machine :
- N°0 pour la ligne «TTL» du processeur machine,
- N°1 pour la ligne «RS232C» du processeur machine,
- N°2 pour la ligne «DNC» du processeur CN.
- N°3 pour la ligne «PERIPH.» du processeur CN;

Ensuite les numéros de 4 à 7 sont affectés à la première carte IT/LIGNES SERIES rencontré dans le rack et les numéros de 8 à 0xB sont affectés à la seconde carte IT/LIGNES SERIES rencontré (Balayage de droite à gauche).



NUM 1060 série II



NUM 1060 série I

NUM 1060 série II (UCSII)

Le numéro des lignes est figé sur la carte UCSII :
- N°0 pour la ligne «COMM 1»,
- N°1 pour la ligne «COMM 2»,

12.2 Initialisation d'une ligne

comf

Syntaxe de l'instruction

comf(n_port, vitemi, vitrec, format)

n_port:	Numéro du port de communication (0 à 0xB).
vitemi:	Vitesse d'émission.
vitrec:	Vitesse de réception.
format:	Codage du format de données et contrôle de flux.

Fonctionnement

La fonction comf() alloue la ligne à la fonction automatisme et configure le port. Une fois initialisée, la ligne ne peut plus être allouée à un autre utilisateur (Fonction CN, ... etc ..).

L'appel de la fonction comf(n_port, vitemi, vitrec, 0) a pour effet de libérer la ligne et de la rendre disponible pour un autre utilisateur.

ATTENTION

Dans tous les cas, les vitesses d'émission et de réception doivent être identiques.

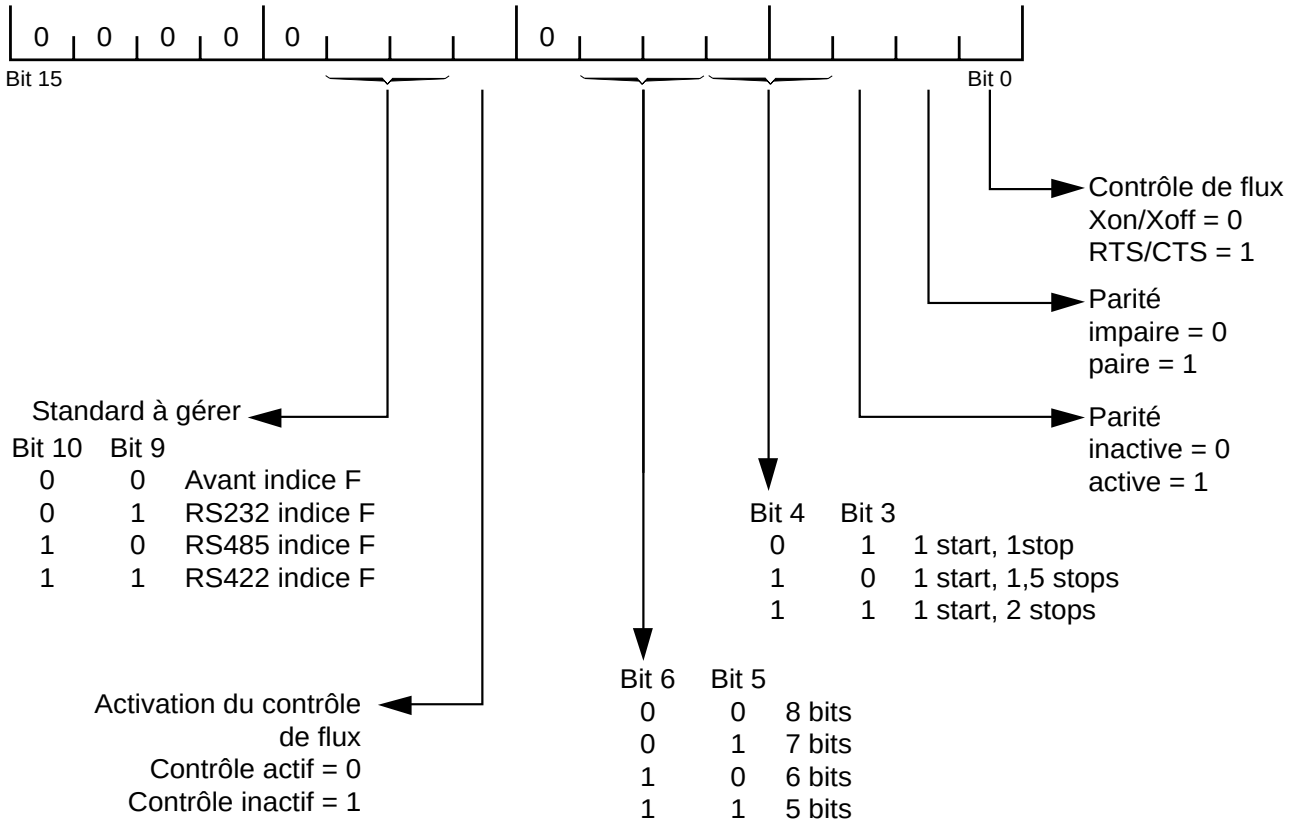
Dans le cas de l'initialisation d'une ligne de la carte processeur CN (Ligne 2 ou 3), il est nécessaire d'appeler la fonction comf() dans une tâche %TF.

Valeur des arguments «vitémi» et «vitrec»

Les valeurs correspondent à des vitesses en bauds.

Valeur de vitémi et vitrec	Vitesse (en bauds)
300	300
600	600
1200	1200
2400	2400
4800	4800
9600	9600
19200	19200

Détail de l'argument «format»



REMARQUE : Dans le cas d'une évolution future, il est recommandé de mettre les bits non significatifs à 0.

Code retourné

Si OK

0

Si défaut

-1	Argument «format» incohérent. Ligne déjà allouée à un autre utilisateur que la fonction automatisme.

12.3 Emission d’un tampon

comout

Syntaxe de l’instruction

comout(n_port, &buffer, nb)

n_port : Numéro du port série.

&buffer : Adresse du tampon à émettre.

nb : Nombre d’octets à émettre ($1 \leq nb \leq 255$).
nb est codé sur un octet non signé.

Permet l’émission d’un tampon sur une ligne de communication série n_port.

Fonctionnement

A l’appel de cette fonction, le système recopie le tampon «&buffer», lance l'émission et retourne à l’appelant. Cette fonction est non bloquante et l’émission se poursuit sous IT jusqu’à la fin du tampon. La fonction comreg() permet de connaître l’état de la transmission en cours.

L’appel de comout(n_port, &buffer, 0) provoque l’abandon d’une éventuelle émission en cours.

Code retourné

Si OK

0

Si défaut

-1 : n_port non valide
Ligne non initialisée
Emission en cours
Argument «nb» supérieur à 255
Pas de full duplex avec contrôle de flux Xon/Xoff

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&buffer" incorrect,
- "&buffer+nb" hors zone autorisée.

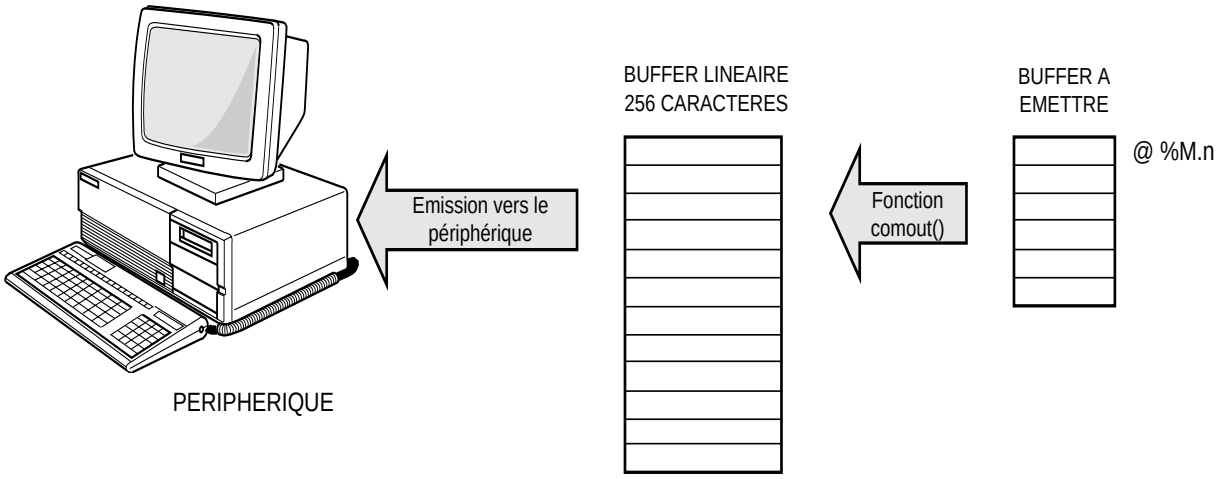


Figure 12.1 - Emission d'un tampon

12.4 Réception d'un tampon

comin

Syntaxe de l'instruction

comin(n_port, &buffer, nb)

n_port : Numéro du port série.

&buffer : adresse du tampon où stocker les caractères reçus.

nb : Nombre maximum de caractères à lire.

Permet la lecture du tampon de réception de la ligne série n_port.

Fonctionnement

Le système gère un tampon de réception en anneau. La fonction comin() permet de lire tout ou partie de ce tampon. Le nombre de caractères recopiés est égal au minimum du nombre de caractères demandés (nb) et du nombre de caractères présents dans le buffer en anneau.

Si la ligne n'avait pas encore été mise à l'écoute avec la fonction comctl(), le premier appel de comin() effectue une mise à l'écoute automatique.

L'appel de comin(n_port, &buffer, 0) provoque l'arrêt et l'initialisation de la réception.

Code retourné

Si OK

n >= 0 Nombre de caractères recopiés dans &buffer.

Si défauts

-1 : n_port non valide
Ligne non initialisée.
Pas de full duplex avec contrôle de flux Xon/Xoff.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&buffer" incorrect,
- "&buffer+nb" hors zone autorisée.

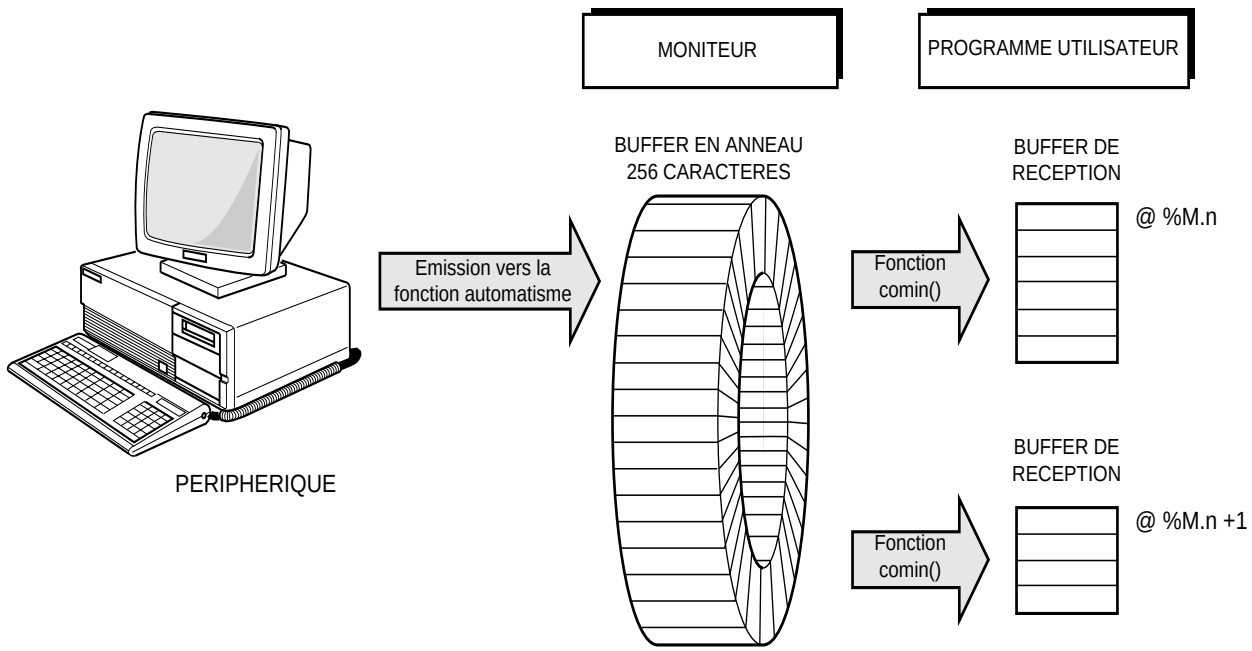


Figure 12.2 - Réception d'un tampon

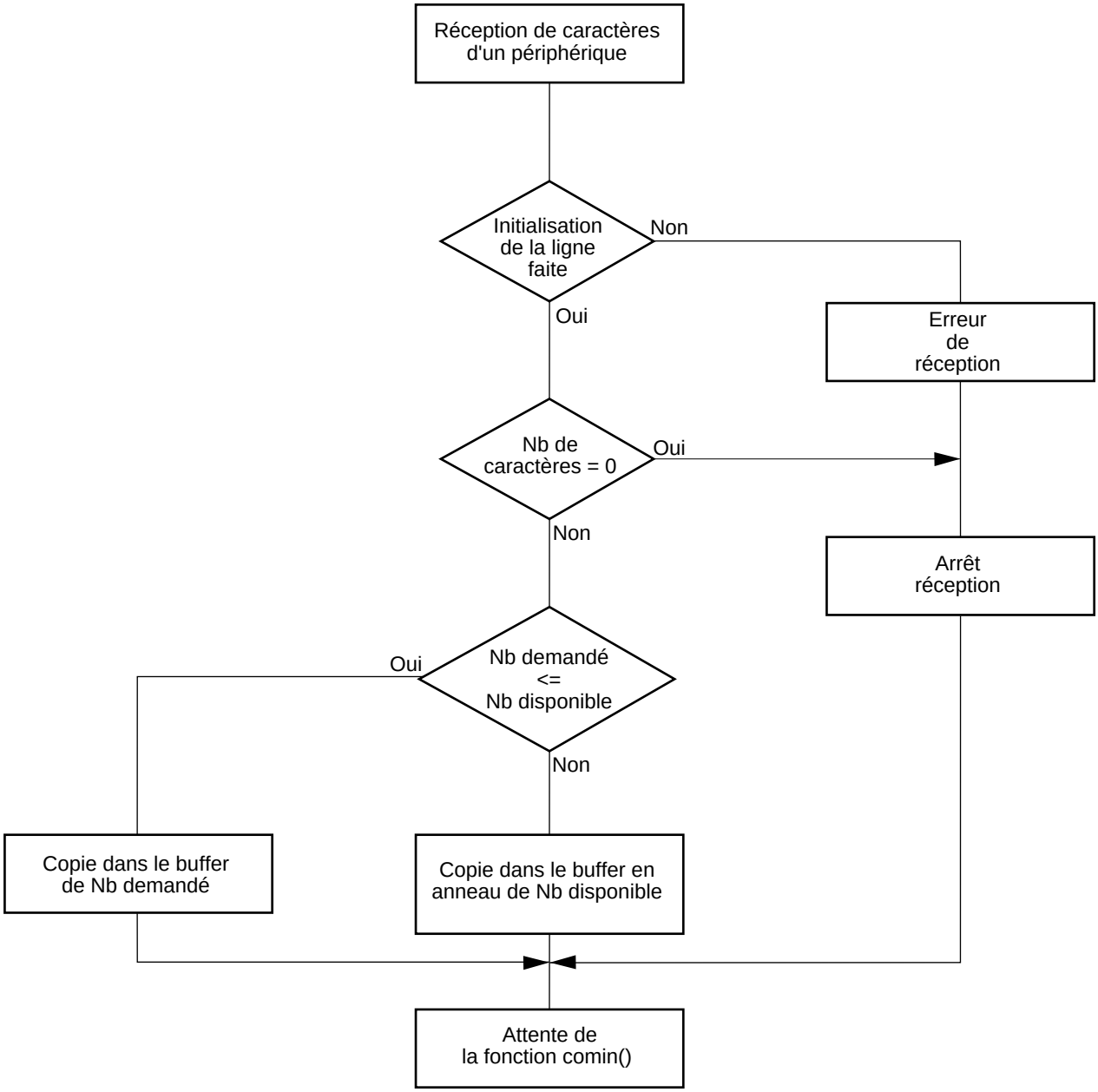


Figure 12.3 - Traitement d'une réception par le moniteur

12.5 Lecture de l'état d'une ligne série

comreg

Syntaxe de l'instruction

comreg(n_port)

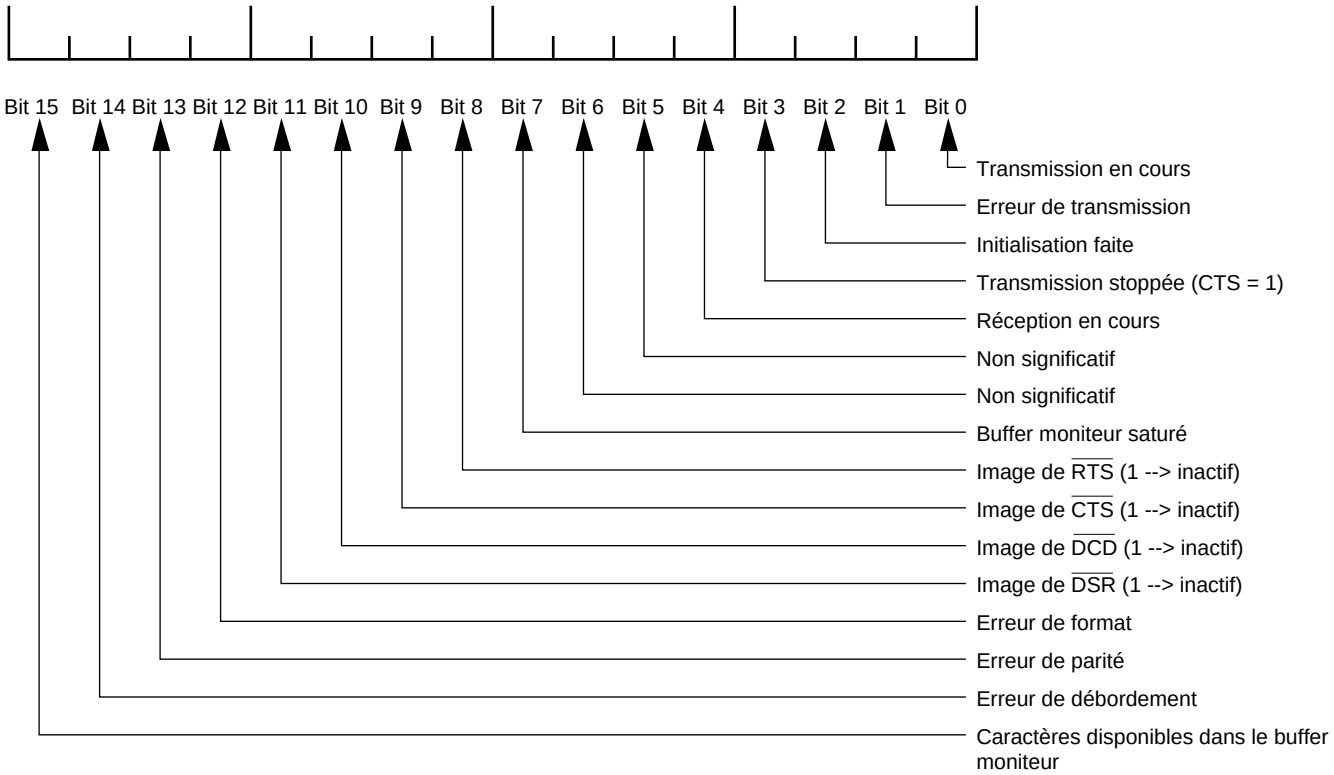
n_port : Numéro du port série.

Description

Permet de retourner l'état général de la ligne série «n_port».

> **ATTENTION**
>
> Dans le cas de l'initialisation d'une ligne de la carte processeur CN (Ligne 2 ou 3), il est nécessaire d'appeler la fonction comreg() dans une tâche %TF.

Code retourné



12.6 Contrôle du pilote de ligne série — comctl

Syntaxe de l'instruction

comctl(n_port, config)

n_port : Numéro du port série.

config : Valeur codée de configuration.

La fonction comctl( ) permet le contrôle du pilote de ligne série n_port.

Fonctionnement

L'action dépend du contrôle de flux utilisé et de l'état de ligne au moment de la demande.

Valeur de «config»	Sans contrôle de flux	Contrôle de flux RTS/CTS	Contrôle de flux Xon/Xoff
Pas de réception en cours et config == 0	Mise à l'écoute de la ligne	Mise à l'écoute de la ligne et activation du signal RTS	Mise à l'écoute de la ligne et émission du caractère Xon
Réception en cours et config == 1	Aucun effet	Déactivation du signal RTS	Emission du caractère Xoff

Code retourné

Si OK

0

Si défauts

-1 : Ligne non initialisée.
Pas de full duplex avec contrôle de flux Xon/Xoff.
Demande (config) incohérente avec l'état en cours.

12.7 Standards de transmission

Le standard de transmission est défini dans l'argument «format» de la fonction comf() (Voir 12.2).

12.7.1 Avant logiciel indice F

12.7.1.1 Sans contrôle de flux

Aucun signal hardware ou software n'est géré en émission comme en réception.

Un fonctionnement bidirectionnel simultané (full duplex) est possible.

12.7.1.2 Contrôle de flux RTS/CTS

Le signal RTS est géré lors d'une réception afin de stopper ou de relancer les transferts. En émission, RTS reste activé durant toute la transmission du buffer.

Un fonctionnement bidirectionnel simultané (full duplex) est impossible.

REMARQUE : Le fait de ne pas tenir compte des signaux de contrôle RTS et CTS (Avec un câble rebouclé RTS sur CTS) permet un fonctionnement bidirectionnel simultané sans contrôle de flux.

12.7.1.3 Contrôle de flux Xon/Xoff

En réception, l'échange est contrôlé par l'envoi de caractères de contrôle sur le canal émission.

Dès l'émission du caractère DC1 (Xon) par la réception, l'entité émettrice est autorisée à émettre. A l'émission du caractère DC3 (Xoff) par la réception, l'entité émettrice dispose d'un délai équivalent au temps de transmission de 20 caractères pour suspendre son émission.

12.7.2 Standard RS232

A partir du logiciel indice F.

12.7.2.1 Sans contrôle de flux

Aucun signal hardware ou software n'est géré en émission comme en réception.

Un fonctionnement bidirectionnel simultané (full duplex) est possible.

12.7.2.2 Contrôle de flux RTS/CTS

En réception, le signal RTS est géré afin de contrôler la ligne. L'émission ne positionne pas ce signal.

Dès l'invalidation de ce signal, l'émetteur doit suspendre l'émission de ces données. Seul un caractère supplémentaire pourra être pris en compte après invalidation de la ligne RTS.

Vu du coté de l'émetteur, à l'invalidation du signal CTS, l'émission doit être suspendue.

Un fonctionnement bidirectionnel simultané (full duplex) est possible.

12.7.2.3 Contrôle de flux Xon/Xoff

En réception, l'échange est contrôlé par l'envoi de caractères de contrôle sur le canal émission.

Dès l'émission du caractère DC1 (Xon) par la réception, l'entité émettrice est autorisée à émettre. A l'émission du caractère DC3 (Xoff) par la réception, l'entité émettrice dispose d'un délai équivalent au temps de transmission de 20 caractères pour suspendre son émission.

Un fonctionnement bidirectionnel simultané (full duplex) est impossible.

12.7.3 Standard RS485

A partir du logiciel indice F.

Dans le standard RS485, il est impossible de réaliser un contrôle de flux. Si le standard RS485 est validé dans la fonction comf(), la valeur du bit 0 de l'argument «format» est non significative.

Le signal RTS est actif durant l'émission d'un buffer et inactif lors d'une réception, ceci afin de piloter les boîtiers d'adaptation RS232/RS485 en émission/réception.

⚠ ATTENTION

La gestion du standard RS485 demande un câblage adapté des boîtiers d'adaptation RS232/RS485.

12.7.4 Standard RS422

A partir du logiciel indice F.

Dans ce standard, le signal RTS est activé durant toute l'utilisation de la ligne.

12.7.4.1 Sans contrôle de flux

Aucun signal hardware ou software n'est géré en émission comme en réception.

Un fonctionnement bidirectionnel simultané (full duplex) est possible.

12.7.4.2 Contrôle de flux Xon/Xoff

En réception, l'échange est contrôlé par l'envoi de caractères de contrôle sur le canal émission.

Dès l'émission du caractère DC1 (Xon) par la réception, l'entité émettrice est autorisée à émettre. A l'émission du caractère DC3 (Xoff) par la réception, l'entité émettrice dispose d'un délai équivalent au temps de transmission de 20 caractères pour suspendre son émission.

13 Fonction timer

13.1 Présentation de la fonction timer

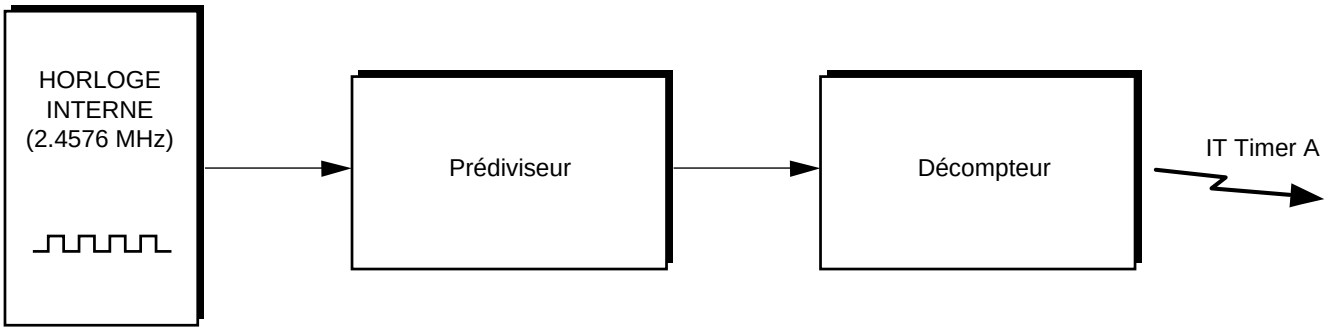
La fonction timer est disponible uniquement sur les système NUM 1060 série I et NUM 1060 série II multicarte.

La fonction automatisme met à disposition de l'utilisateur deux timers A et B. Les timers A et B sont constitués d'un décompteur 8 bits capable de générer une impulsion à chaque changement de valeur de son contenu. Le décompteur est alors immédiatement rechargé avec une valeur programmée et conservée dans le registre de donnée du timer.

13.2 Mode de fonctionnement

13.2.1 Mode délai

En mode délai, la fréquence de l'horloge interne (2.4576 Mhz) est divisée par le prédiviseur programmable qui fournit des impulsions au décompteur.



13.3 Association tâche %TH avec un timer

thtimer

Syntaxe de l'instruction

thtimer(numero_th, n_timer, n_milliseconde)

numéro_th : Numéro de la tâche %TH.

n_timer : Numéro du timer (0 ou 1).

n_milliseconde : Délai en millisecondes.

Permet d'associer une tâche %TH avec une interruption générée par le timer A ou B

Fonctionnement

L'appel de thtimer() provoque l'armement du timer dont le numéro est passé dans n_timer avec une valeur égale à n_milliseconde. Quand ce délai est écoulé, le timer génère une IT qui est exploitée par le système pour appeler la tâche %TH de numéro numéro_th.

Le délai n_milliseconde est compris entre 0 et 2.147.483.647 millisecondes.

La fonction thtimer() est «modale» c'est à dire qu'une fois l'appel de thtimer() effectué, la tâche %TH sera appelée à la période fixée par n_milliseconde.

Pour annuler une fonction thtimer() il faut appeler thtimer() avec le paramètre n_milliseconde égal à ZERO.

L'appel de thtimer() alors que le timer est en cours provoque une RAZ du timer et son chargement avec la nouvelle valeur de n_milliseconde.

Code retourné

Si OK

0

Si défaut

-1 : n_milliseconde < 0 ou > 2.147.483.647

14 Fonction dateur

14.1 Présentation de la fonction dateur

La lecture de la date courante est réalisé par la fonction tmget().

La date du système est gérée par un dateur sauvegardé en mémoire globale.

Une page CN permet la mise à jour de ce dateur par l'opérateur.

14.2 Lecture de la date courante — tmget

Syntaxe de l'instruction

tmget(&date)

&date : Adresse du bloc mémoire (11 octets) qui va recevoir la structure date.

Description

Permet de lire la date courante.

Structure du bloc date

N° octet	Type de donnée	Valeur
octets 0 - 1	Année	0 à 65535
octet 2	Mois	1 à 12
octet 3	Jour	1 à 31
octet 4	Heure	0 à 23
octet 5	Minutes	0 à 59
octet 6	Secondes	0 à 59
octets 7-8	Millisecondes	0 à 999 (Précision de l'ordre de 50 ms)

Code retourné

Si OK

0

Si défaut

-1 : La date n'a pas été mise à jour.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&date" incorrect.

14.3 Lecture de la date courante avec jour de la semaine dtget

Syntaxe de l'instruction

dtget(&date)

&date : Adresse du bloc mémoire (11 octets) qui va recevoir la structure date.

Description

Permet de lire la date courante avec le jour de la semaine spécifié.

Structure du bloc date

N° octet	Type de donnée	Valeur
octet 0	Jour de la semaine	0 à 6
octet 1	Jour	1 à 31
octet 2	Mois	1 à 12
octet 3	Année	0 à 99
octet 4	Heure	0 à 23
octet 5	Minutes	0 à 59
octets 6	Secondes	0 à 59

Code retourné

Si OK

0

Si défaut

-1 : La date n'a pas été mise à jour.

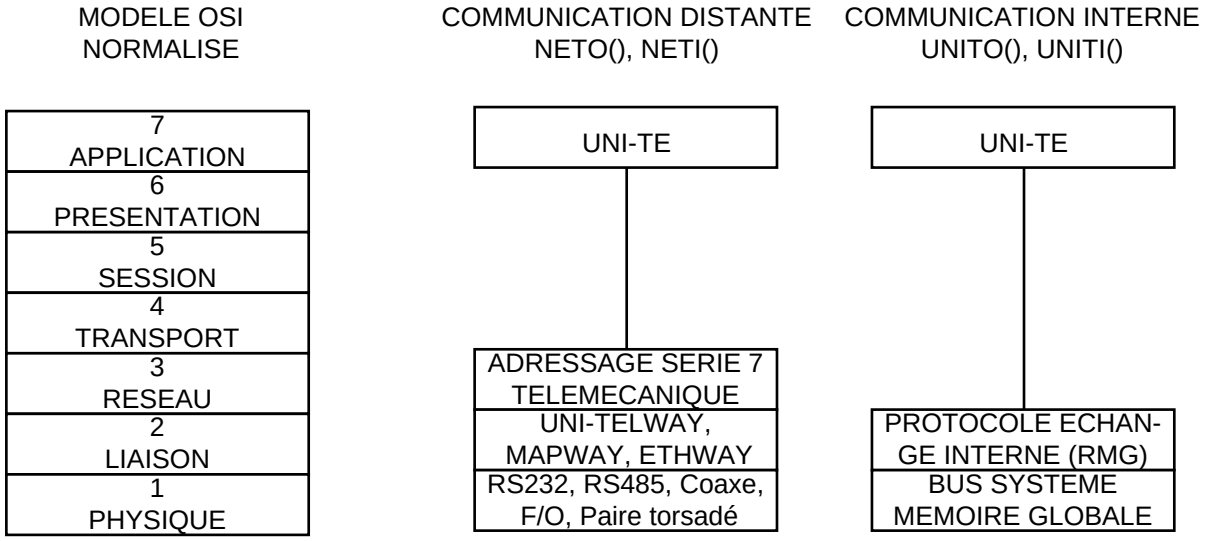
Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&date" incorrect.

15 Echanges par protocole

15.1 Présentation des échanges

Les échanges par protocoles permettent la communication :
- entre les fonctions automatisme et CN du système (communication locale DNC1000),
- entre la CN NUM1060 et les stations distantes connectées sur les réseaux MAPWAY, ETHWAY et UNI-TELWAY (communication distante).



REMARQUE : Seul la communication locale DNC1000, les requêtes et les fonctions qui lui sont liées sont traités dans ce chapitre. Se reporter au manuel «Protocole UNITE» pour la communication avec des stations distantes et en fin de ce chapitre pour les fonctions de communication.

15.1.1 Présentation du protocole DNC1000

DNC1000 est une procédure de communication locale entre la fonction automatisme et les autres fonctions du système. Elle permet la transmission d'informations inaccessibles par la zone d'échange.

La communication s'effectue entre un demandeur (ou client) et un serveur par requettes UNITE. En général la fonction automatisme est le demandeur et la fonction CN le serveur.

Le programme pièce peut également initier un échange à destination de la fonction automatisme (données non solicitées

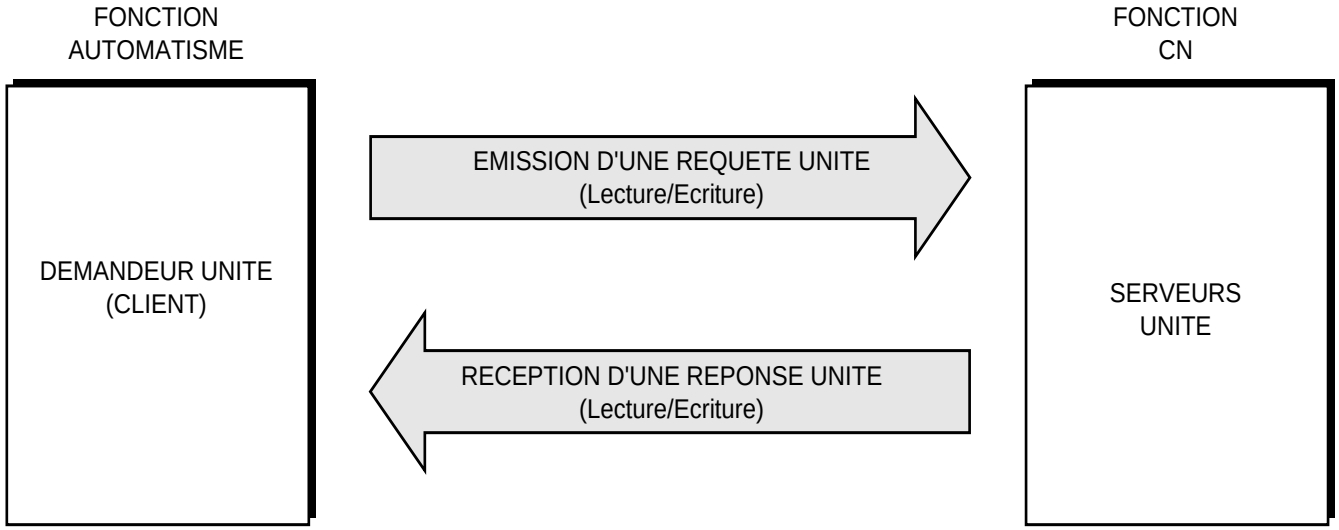


Figure 15.1 - Echange par protocole

15.1.2 Mécanisme des échanges DNC1000

15.1.2.1 Déroulement du traitement d'une requête

La fonction automatisme (demandeur) émet une requête (Lecture/Ecriture) vers le serveur. Cette requête, placée dans un tampon, est stockée en file d'attente. Elle est traitée ensuite par le serveur qui émet un code réponse avec d'éventuelles données. La réponse et les données sont récupérées par le programme utilisateur.

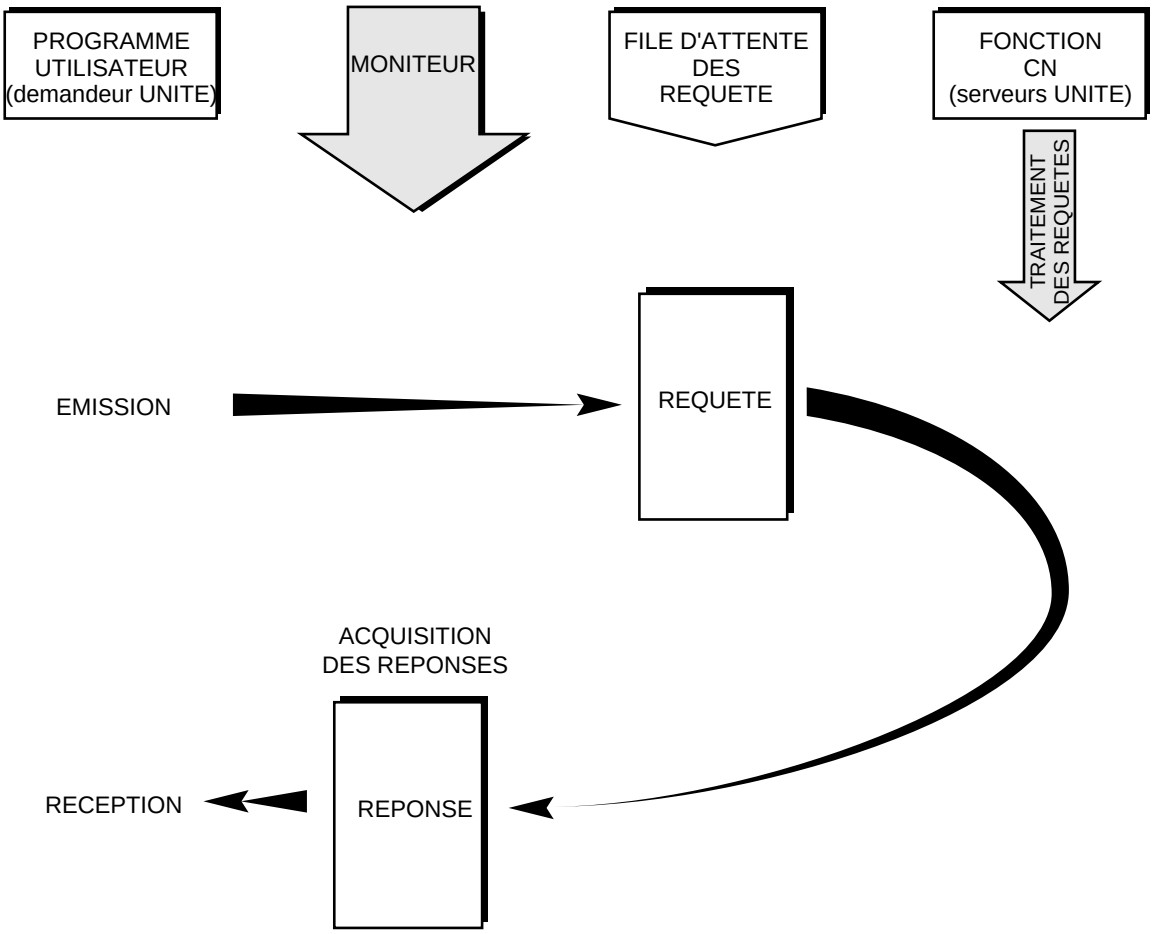


Figure 15.2 - Principe de traitement d'une requête

15.1.2.2 Notion de porte

Le demandeur doit associer chaque requête avec une porte.

L'émission d'une requête et la réception de la réponse se font sur la même porte.

Deux types de portes sont accessibles :
- 16 portes adressées de 0x30 à 0x3F permettent au demandeur d'émettre plusieurs requêtes en parallèle,
- 8 portes adressées de 0x10 à 0x17, associées aux groupes d'axes 1 à 8, qui permettent de recevoir des données non sollicités en provenance du programme pièce ($1 et $11 dans le programme pièce).

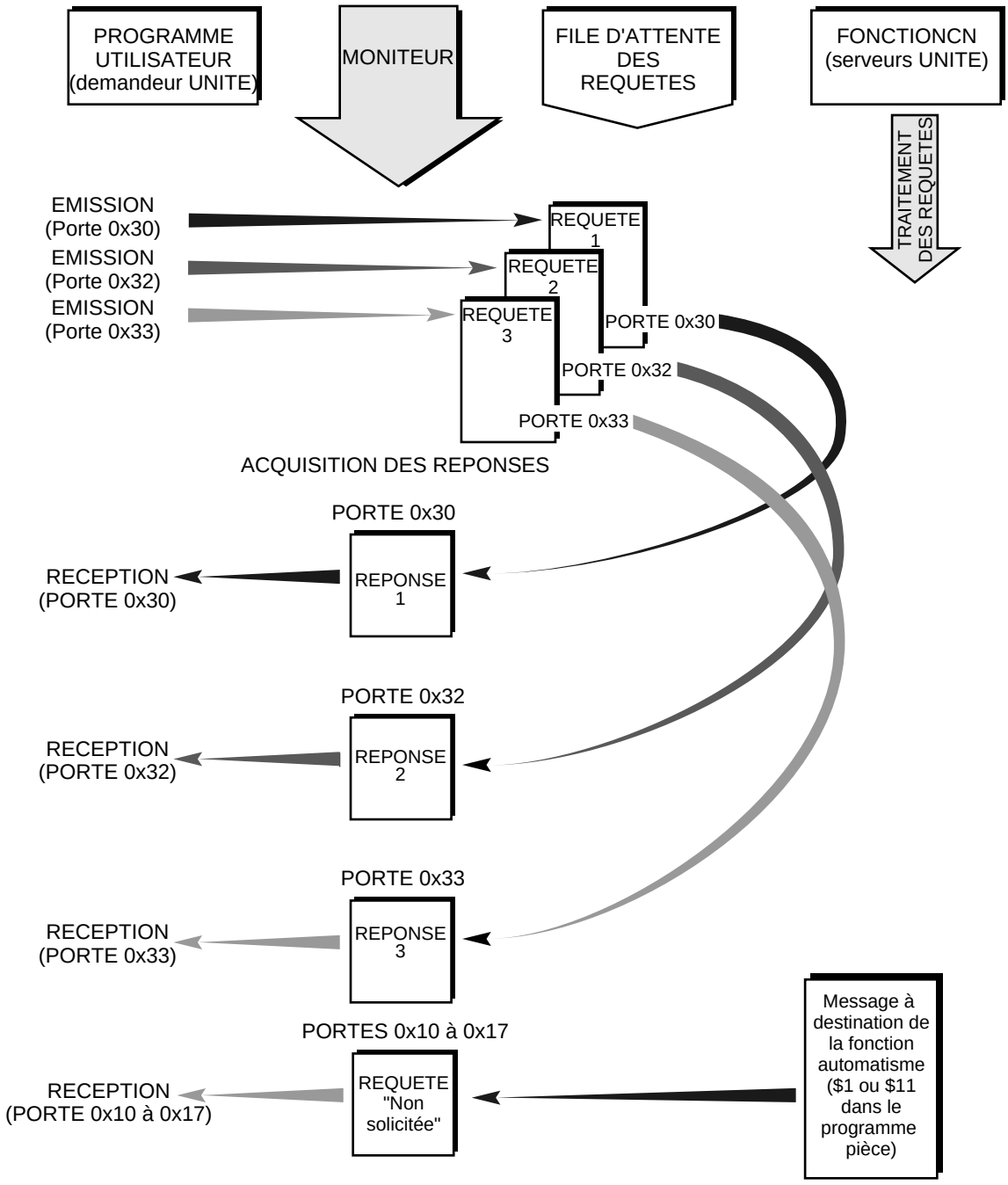


Figure 15.3 - Utilisation des portes

15.2 Objets accessibles par requête UNITE

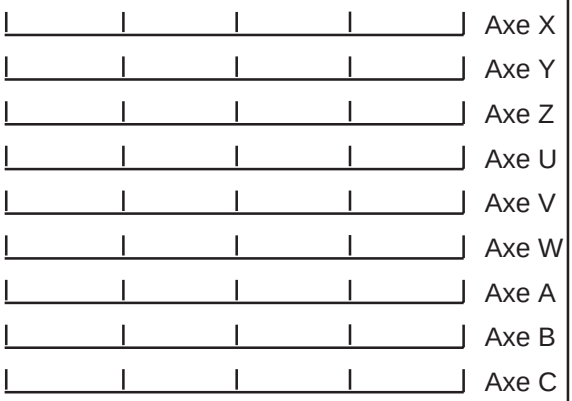
15.2.1 Liste des requêtes de type «Objet» traitées par la fonction CN

Nom de la famille	N° de segment	Accessible en		Définition de l'objet		
		Lecture	Ecriture	Nom	Taille	Nombre maximum d'objet dans la famille
Référence de position des axes	128	X		Axes programmes	9 longs mots	8 (1 par groupe d'axes)
Mesure des axes	129	X		Axes programmes	9 longs mots	8 (1 par groupe d'axes)
PREF	130	X	X	Axes programmes	9 longs mots	8 (1 par groupe d'axes)
DEC1	131	X	X	Axes programmes	9 longs mots	8 (1 par groupe d'axes)
DEC3	132	X	X	Axes programmes	9 longs mots	8 (1 par groupe d'axes)
Limite minimale de l'usinage	133	X	X	Axes programmes	9 longs mots	8 (1 par groupe d'axes)
Limite maximale de l'usinage	134	X	X	Axes programmes	9 longs mots	8 (1 par groupe d'axes)
Inclinaison des axes	135	X	X		1 long mot	32 (1 par axe)
Origine machine	136	X	X	Axes physiques	1 long mot	32 (1 par axe)
Courses machines mini	137	X	X	Axes physiques	1 long mot	32 (1 par axe)
Courses machines maxi	138	X	X	Axes physiques	1 long mot	32 (1 par axe)
Correction de références des axes	139	X		Axes physiques	1 long mot	32 (1 par axe)
Référence de position des axes	140	X		Axes physiques	1 long mot	32 (1 par axe)
Position mesurée des axes	141	X		Axes physiques	1 long mot	32 (1 par axe)
Axes asservis	143	X		Présence d'axes	1 long mot	1 (1 bit par axe)
Vitesse de broche mesurée	144	X		Broches	1 long mot	4 (1 par broche)

Nom de la famille	N° de segment	Accessible en		Définition de l'objet		
		Lecture	Ecriture	Nom	Taille	Nombre maximum d'objet dans la famille
Position mesurée des broches	145	X		Broches	1 long mot	4 (1 par broche)
Correcteur d'outils	146	X	X	Outils	7 longs mots	255
Variable H	147	X	X	Temps utilisation outils	1 long mot	255
Etat interpolation	148	X		Etat interpolation	4 longs mots	8 (1 par groupe d'axes)
Axes initialisés	149	X	X	Présence d'axes	1 long mot	1 (1 bit par axe)
Paramètres E80000	150	X	X		1 long mot	51
Paramètres E81000	151	X	X		1 long mot	Nombre déclaré dans le paramè tre machine P58
Paramètres E82000	152	X	X		1 long mot	Nombre déclaré dans le paramè tre machine P58
Status programme	153	X			22 octets	8 (1 par groupe d'axes)
Cotes de fin de bloc	157	X			11 longs mots	8 (1 par groupe d'axes)
Sélection du mode	180	X	X		1 mot	1
Sélection du programme pièce courant	181	X	X		1 mot	1
Données transmises au programme en cours d'exécution	224	X	X		1 long mot	8 (1 par groupe d'axes)
Acquittement de messages	226	X	X		1 octet	8 (1 par groupe d'axes)
Configuration ligne IT	227	X	X		1 octet	8 (1 par ligne)
Validation/révo-cation synchro des axes	235	X	X		1 long mot	1 (1 bit par axe)

15.2.2 Eléments constitutifs des objets

L'unité Ui correspond à l'unité interne du système définie par paramètre machine.

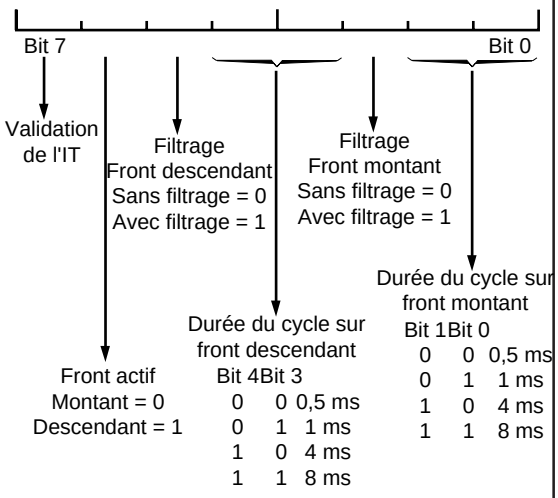
Numéro de segment (Valeur Hexa)	Accessible en	Description	Valeur ou unité	Paramètres correspondants
128 (0x80)	Lecture	Référence de position des axes Taille de l'objet : 9 longs mots Adresse du premier objet dans la famille : 0 	Ui ou 1/10000°	E70000 à E78000
129 (0x81)	Lecture	Mesure des axes Taille de l'objet : 9 longs mots Adresse du premier objet dans la famille : 0	-99999999 à 99999999 Ui	E90000 à E90031
130 (0x82)	Lecture/Ecriture	PREF Taille de l'objet : 9 longs mots Adresse du premier objet dans la famille : 0	-99999999 à 99999999 Ui	E60000 à E68000
131 (0x83)	Lecture/Ecriture	DEC1 Taille de l'objet : 9 longs mots Adresse du premier objet dans la famille : 0	-99999999 à 99999999 Ui	E60001 à E68001
132 (0x84)	Lecture/Ecriture	DEC3 Taille de l'objet : 9 longs mots Adresse du premier objet dans la famille : 0	-99999999 à 99999999 Ui	E60004 à E68004
133 (0x85)	Lecture/Ecriture	Course dynamique minimum Taille de l'objet : 9 longs mots Adresse du premier objet dans la famille : 0	-99999999 à 99999999 Ui	E60002 à E68002
134 (0x86)	Lecture/Ecriture	Course dynamique maximum Taille de l'objet : 9 longs mots Adresse du premier objet dans la famille : 0	-99999999 à 99999999 Ui	E60003 à E68003

Numéro de segment (Valeur Hexa)	Accessible en	Description	Valeur ou unité	Paramètres correspondants
135 (0x87)	Lecture/Ecriture	valeur de l'angle pour les axes inclinés Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	1/10000°	E69001
136 (0x88)	Lecture/Ecriture	Origine machine Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	Ui Ou 1/10000°	Paramètre P16
137 (0x89)	Lecture/Ecriture	Course statique minimum Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	Ui	Paramètre P17
138 (0x8A)	Lecture/Ecriture	Course statique maximum Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	Ui	Paramètre P17
139 (0x8B)	Lecture	Corrections courantes d'un axe esclave Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	-99999999 à 99999999 Ui	E95000 E95031
140 (0x8C)	Lecture	Référence de position d'un axe Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	Ui	E70000 à E78000
141 (0x8D)	Lecture	Position mesurée d'un axe Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	Ui	E90000 à E90031
143 (0x8F)	Lecture	Axes asservis Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	0 ou 1	E91000 à E91031
144 (0x90)	Lecture	Vitesse de broche mesurée Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	Ui	
145 (0x91)	Lecture	Référence de position des broches mesurées Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	0 à 3599999 °/10000	E90101 à E90104

Numéro de segment (Valeur Hexa)	Accessible en	Description	Valeur ou unité	Paramètres correspondants
146 (0x92)	Lecture/Ecriture	Corrections d'outils en tournage Taille de l'objet : 7 longs mots Adresse du premier objet dans la famille :		
		1er long mot - Longueur en X	Ui	E50001 à E50255
		2ème long mot - Longueur en Z	Ui	E51001 à E51255
		3ème long mot - Rayon de pastille	Ui	E52001 à E52255
		4ème long mot - Correction dynamique en X	Ui	E53001 à E53255
		5ème long mot - Correction dynamique en Z	Ui	E54001 à E54255
		6ème long mot - Direction de nez d'outil	De 0 à 8	E55001 à E55255
		7ème long mot - Type d'outil	De 1 ou 2	E57001 à E57255
146 (0x92)	Lecture/Ecriture	Corrections d'outils en fraisage Taille de l'objet : 7 longs mots Adresse du premier objet dans la famille : 1		
		1er long mot - Longueur de l'outil	Ui	E50001 à E50255
		2ème long mot - Rayon de bout de fraise	Ui	E51001 à E51255
		3ème long mot - Rayon d'outil	Ui	E52001 à E52255
		4ème long mot - Correction dynamique de longueur		UiE53001 à E53255
		5ème long mot - Correction dynamique de rayon	Ui	E54001 à E54255
		6ème long mot - non significatif		
		7ème long mot - Type d'outil	0	E57001 à E57255
147 (0x93)	Lecture/Ecriture	Paramètres disponibles (H de la table des corrections dynamiques) Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 1	-99999999 à 99999999	E56001 à E56255

Numéro de segment (Valeur Hexa)	Accessible en	Description	Valeur ou unité	Paramètres correspondants
148 (0x94)	Lecture	Etat interpolation Taille de l'objet : 4 longs mots Adresse du premier objet dans la famille : 0 1er long mot - Vitesse courante 2ème long mot - Distance restant à parcourir sur le bloc en cours (Sur trajectoire) 3ème long mot - Vitesse programmée 4ème long mot - Coefficient de modulation de vitesse	 mm/Ech mm mm/mn mm/tr, V/L 2^{-16}	
149 (0x95)	Lecture/Ecriture	POM non faite sur un axe Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	1 ou 0	E91100 à E91131
150 (0x96)	Lecture/Ecriture	Paramètre de données locales Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	-99999999 à 99999999	E80000 E80050
151 (0x97)	Lecture/Ecriture	Position de référence des axes maîtres (Calibration inter axes) Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	-99999999 à 99999999 Ui	E81000 E81999
152 (0x98)	Lecture/Ecriture	Correction des axes esclaves (Calibration inter axes) Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0	-99999999 à 99999999 Ui	E82000 E82999
153 (0x99)	Lecture	Status programme (Voir 15.2.3) Taille de l'objet : 22 octets Adresse du premier objet dans la famille : 0 1 long mot : Liste des fonctions G présentes 1 long mot : Numéro du programme en cours d'exécution 1 mot : Numéro du bloc en cours d'exécution 1 mot : Numéro d'erreur programme 1 mot : Numéro de bloc en erreur 1 mot : Numéro d'outil 1 mot : Direction d'outil 1 mot : Numéro du correcteur d'outil 1 mot : Liste des traitements restant à exécuter		

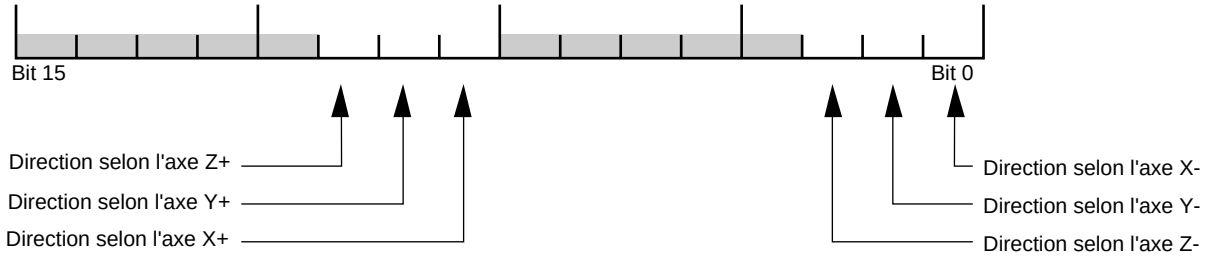
Numéro de segment (Valeur Hexa)	Accessible en	Description	Valeur ou unité	Paramètres correspondants
157 (0x9D)	Lecture	Cote de fin du bloc Taille de l'objet : 11 longs mots Adresse du premier objet dans la famille : 0 Les 36 premier octets donnent les cotes de fin de bloc (4 octet par axes) pour les axes X, Y, Z, U, V, W, A, B, C Le long mot suivant donne l'abscisse du centre en interpolation circulaire. Le dernier long mot donne l'ordonnée du centre en interpolation circulaire.		
180 (0xB4)	Lecture/Ecriture	Sélection du mode Taille de l'objet : 1 mot Adresse du premier objet dans la famille : 0 Mode Continu Mode Séquentiel Mode immédiat Mode rapide Mode «RNS» Mode Modification Mode Test Mode Manuel Mode Prise d'Origine Mesure Mode Prise de Référence Mode Réglage Automatique d'Outils Mode Chargement Mode Déchargement Si le bit de poids fort (bit 15) est positionné à 1, la demande de changement de mode reste maintenue.	 0x0000 0x0001 0x0002 0x0003 0x0004 0x0005 0x0006 0x0007 0x0008 0x0009 0x000A 0x000D 0x000F	E41000
181 (0xB5)	Lecture/Ecriture	Sélection du programme courant Taille de l'objet : 1 mot Adresse du premier objet dans la famille : 0	1 à 99999	
224 (0xE0)	Lecture/Ecriture	Données transmises au programme pièce en cours d'exécution (Voir 15.3.11) Taille de l'objet : 1 long mot Adresse du premier objet dans la famille : 0		
226 (0xE2)	Lecture/Ecriture	Acquittement de messages bloquant transmis par le programme pièce »$11» (Voir 15.3.11) Taille de l'objet : 1 octet Adresse du premier objet dans la famille : 0		

Numéro de segment (Valeur Hexa)	Accessible en	Description	Valeur ou unité	Paramètres correspondants
227 (0xE3)	Lecture/Ecriture	Configuration des lignes d'IT des cartes IT/lignes séries 		
235 (0xEB)	Lecture/Ecriture	Validation ou révocation de la synchronisation des axes Taille de l'objet : 1 long mot (1 bit par axe)		

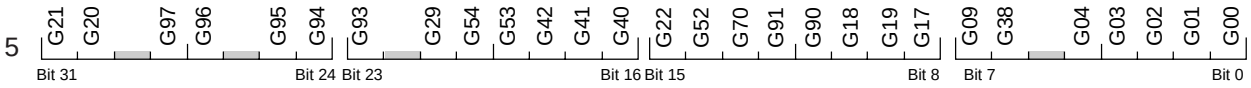
15.2.3 Segment status programme

Direction d'outil

La valeur de la direction d'outil est positionnée dans l'octet de poids faible si elle est négative, ou dans l'octet de poids fort si elle est positive.

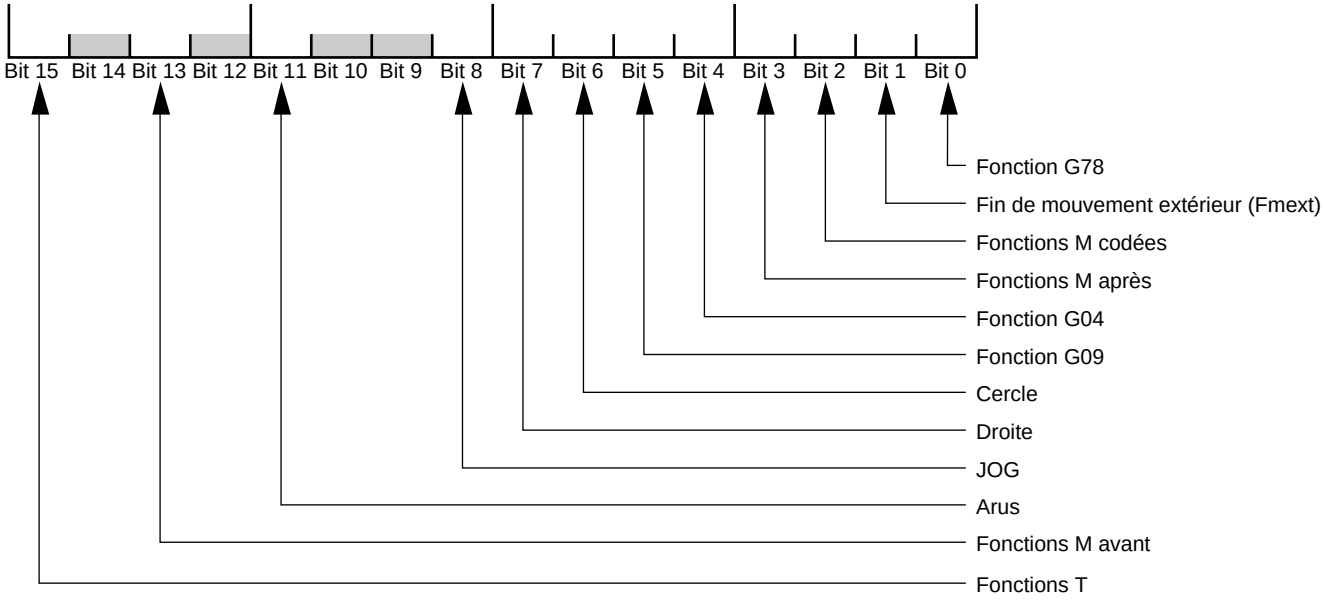


Détail de la liste des fonctions G présentes



Liste des traitements restant à exécuter

Le bit de rang le plus élevé désigne la fonction en cours d'exécution.



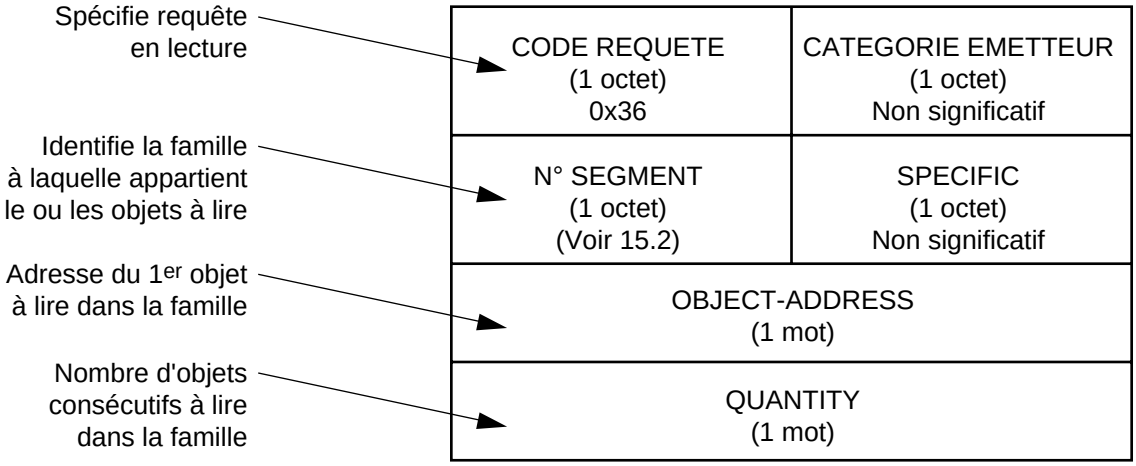
15.3 Requêtes UNITE traitées par la fonction CN

15.3.1 Requête «READ-OBJECT»

Description

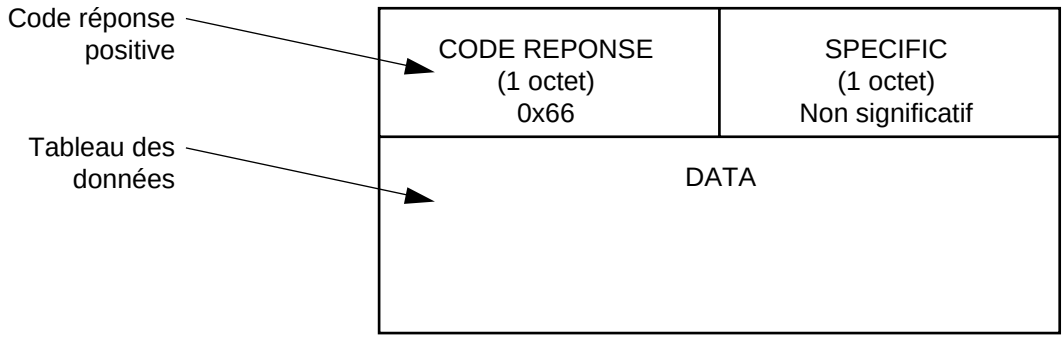
La requête «READ OBJECT» permet de lire les objets accessibles en lecture du serveur CN (Voir 15.2.2).

Format de la requête

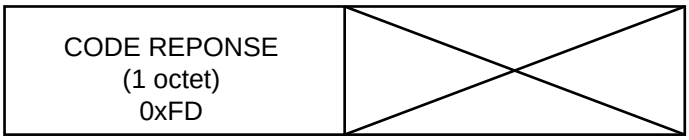


Format de la réponse

Réponse positive



Réponse négative



REMARQUE : Si la quantité précisée est telle que la réponse pourrait contenir plus de 128 octets, la requête est refusée (code réponse négatif).

Exemple de lecture du numéro de programme courant

Requête émise

CODE REQUETE 0x36	CATEGORIE EMETTEUR 0x00
SEGMENT 0xB5	SPECIFIC 0x00
OBJECT-ADDRESS 0x0000	
QUANTITY 0x0001	

Réponse positive avec données

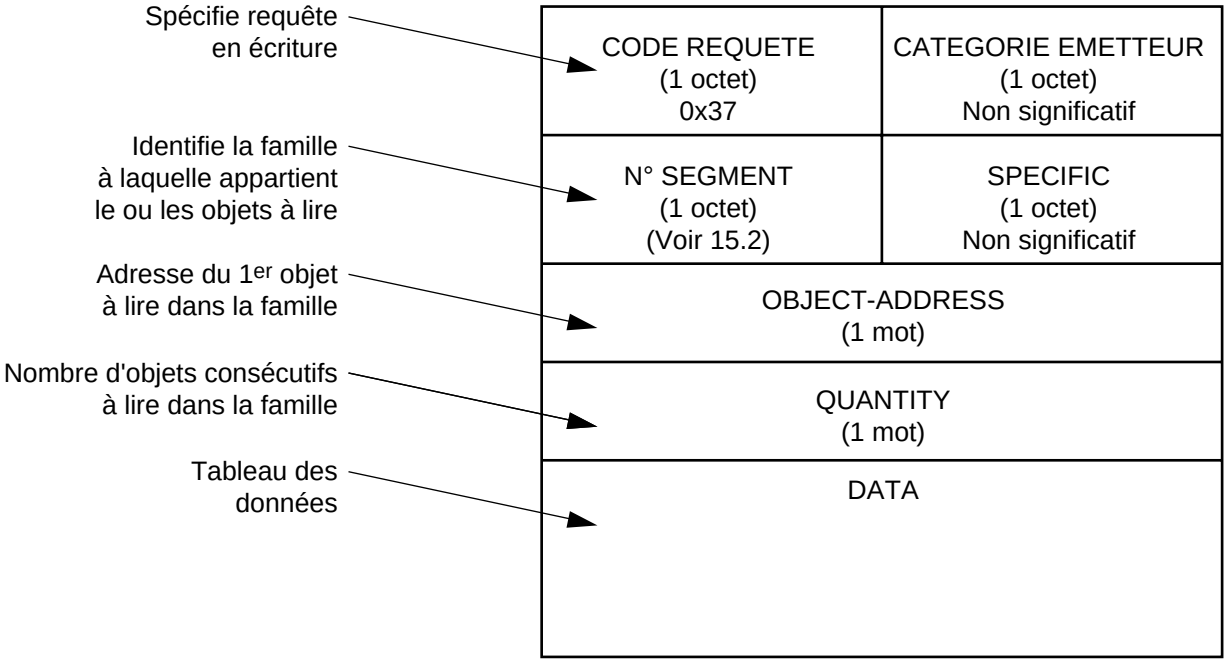
CODE REPONSE 0x66	SPECIFIC 0x00
DATA 0x0053 (Programme %83.)	

15.3.2 Requête «WRITE-OBJECT»

Description

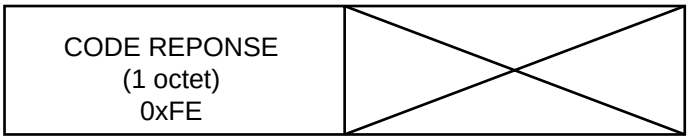
La requête «WRITE OBJECT» permet d'écrire les valeurs des objets du logiciel CN (Voir 15.2.2).

Format de la requête

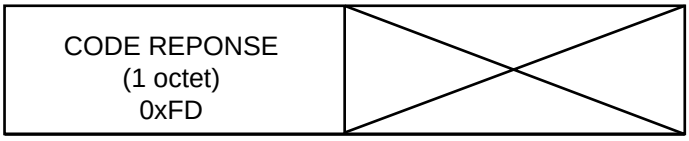


format de la réponse

Réponse positive



Réponse négative



REMARQUE : Si la quantité précisée est telle que la requête contient plus de 128 octets, la requête est refusée (code réponse négatif).

15.3.3 Requête «DELETE-FILE»

Permet de détruire un programme pièce stocké en mémoire RAM de la CN.

Format de la requête

CODE REQUETE (1 octet) 0xF5	CATEGORIE EMETTEUR (1 octet) Non significatif
COMPLEMENT DE CODE REQUETE (1 octet) 0x46	NOM DE FICHIER (1 long mot) 1er octet
2ème octet	3ème octet
4ème octet	

Détail du champ «NOM FICHIER»

Le champ «NOM FICHIER» donne le numéro du programme pièce indexé par le groupe d'axes (N° du programme pièce x 10 + N° du groupe d'axes).

Format de la réponse

Réponse positive

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x76
STATUS (1 octet) 0x00	

Réponse négative

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x76
STATUS (1 octet) Voir tableau ci après	

Code status	Définition
0x02	Manipulation dans zone programme
0x05	Fichier inexistant
0x0A	CN pas dans l'état RAZ

15.3.4 Requête «READ-MEMORY-FREE»

Permet de connaître le nombre d'octets disponible en mémoire RAM de la CN.

Format de la requête

CODE REQUETE (1 octet) 0xF5	CATEGORIE EMETTEUR (1 octet) Non significatif
COMPLEMENT DE CODE REQUETE (1 octet) 0x47	

Format de la réponse

Réponse positive

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x77
STATUS (1 octet) 0x00	VALUE (1 long mot) 1er octet
2ème octet	3ème octet
4ème octet	

Réponse négative

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x77
STATUS (1 octet) Voir tableau ci-après	

Code status	Définition
0x02	Manipulation dans zone programme

15.3.5 Requête «OPEN-DIRECTORY»

Permet de connaître la liste des programme pièce présent en mémoire RAM de la CN.

Si la liste est trop longue pour figurer totalement dans la réponse à cette requête (Status = 0x00), le complément sera donné par la réponse à la requête «DIRECTORY» (Voir 15.3.6). Si toutefois ce complément ne devait pas être demandé, la requête «CLOSE DIRECTORY» (Voir 15.3.7) doit être émise pour clore l'opération.

Si la liste peut figurer totalement dans la réponse à cette requête (Status = 0x0C), l'opération est close automatiquement et la requête «CLOSE DIRECTORY» n'est pas nécessaire.

Format de la requête

CODE REQUETE (1 octet) 0xF5	CATEGORIE EMETTEUR (1 Octet) Non significatif
COMPLEMENT DE CODE REQUETE (1 octet) 0x48	
NOM FICHIER (1 long mot) (Voir REMARQUE)	

REMARQUES : Le champ «NOM FICHIER» donne un numéro du premier programme pièce indexé par le groupe d'axes (N° du programme pièce x 10 + N° du groupe d'axes), que l'on veut voir figurer dans la réponse.

Si ce programme n'est pas présent en mémoire, la liste est donnée à partir du programme suivant.

Si la valeur du champ est 0, la liste est donnée à partir du premier programme présent en mémoire.

Format de la réponse

Réponse positive

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x78
STATUS (1 octet) Voir tableau ci-après	
DATA (Voir REMARQUE)	

Code status	Définition
0x00	OK Il reste des informations à transmettre
0x0F	OK Fin de directory (Fermeture automatique)

REMARQUE : Dans le champ «DATA», chaque programme pièce est décrit par 2 longs mots :

- *le premier donne le numéro du programme pièce indexé par le groupe d'axes (N° du programme pièce x 10 + N° du groupe d'axes),*
- *Le second donne la longueur en octet de ce programme pièce.*

Réponse négative

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x78
STATUS (1 octet) Voir tableau ci-après	

Code status	Définition
0x02	Manipulation dans zone programme
0x09	Taille du buffer insuffisante pour la réponse

15.3.6 Requête «DIRECTORY»

Permet de connaître le complément de la liste des programmes pièce présent dans la mémoire de la CN à la suite d'une requête «OPEN DIRECTORY».

Si la liste est trop longue pour figurer totalement dans la réponse à cette requête (Status =0x00), le complément sera donné par la réponse à une autre requête «DIRECTORY». Si toutefois ce complément ne devait pas être demandé, la requête «CLOSE DIRECTORY» (Voir 15.3.7) doit être émise pour clore l'opération.

Si la liste peut figurer totalement dans la réponse à cette requête (Status = 0x0F), l'opération est close automatiquement et la requête «CLOSE DIRECTORY» n'est pas nécessaire.

Format de la requête

CODE REQUETE (1 octet) 0xF5	CATEGORIE EMETTEUR (1 octet) Non significatif
COMPLEMENT DE CODE REQUETE (1 octet) 0x49	

Format de la réponse

Réponse positive

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x79
STATUS (1 octet) Voir tableau ci-après	
DATA (Voir REMARQUE)	

Code status	Définition
0x00	OK Il reste des informations à transmettre
0x0F	OK Fin de directory (Fermeture automatique)

REMARQUE : Dans le champ «DATA», chaque programme pièce est décrit par 2 longs mots :

- *le premier donne le numéro du programme pièce indexé par le groupe d'axes (N° du programme pièce x 10 + N° du groupe d'axes),*
- *Le second donne la longueur en octet de ce programme pièce.*

Réponse négative

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x79
STATUS (1 octet) Voir tableau ci-après	

Code status	Définition
0x02	Manipulation dans zone programme
0x09	Taille du buffer insuffisante pour la réponse

15.3.7 Requête «CLOSE-DIRECTORY»

Permet de clore une opération de directory.

Format de la requête

CODE REQUETE (1 octet) 0xF5	CATEGORIE EMETTEUR (1 octet) Non significatif
COMPLEMENT DE CODE REQUETE (1 octet) 0x4A	

Format de la réponse

Réponse positive

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x7A
STATUS (1 octet) 0x00	

Réponse négative

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x7A
STATUS (1 octet) Voir tableau ci-après	

Code status	Définition
0x04	Directory déjà fermé

15.3.8 Requête «READ-BLOCK»

Permet de lire un bloc de programme pièce.

Format de la requête

CODE REQUETE (1 octet) 0xF5	COMPLEMENT DE CODE REQUETE (1 octet) 0x50
NUMERO PROGRAMME (1 long mot) (Voir REMARQUE)	
NUMERO BLOC (1 mot)	
OFFSET BLOC (1 mot)	

REMARQUE : Le champ «NUMERO PROGRAMME» donne le numéro du programme pièce indexé par le groupe d'axes (N° du programme pièce x 10 + N° du groupe d'axes).

Format de la réponse

Réponse positive

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x80
LONGUEUR BLOC (1 mot)	
DATA (Voir REMARQUE)	

REMARQUE : Le champ DATA peut contenir jusqu'à 119 octets et se termine par (LF).

Réponse négative

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x80

Pour cette requête, le code réponse et son complément ne sont pas significatif. C'est le code retourné dans la fonction uniti() (Voir 15.4.2) qui informe sur l'aboutissement de la requête.

15.3.9 Requête «WRITE-BLOCK»

Permet d'insérer, de modifier, ou de supprimer un bloc de programme pièce. La réservation préalable d'un espace mémoire supérieur à l'espace utile pour ce programme (Requête «RESERVE MEMORY» Voir 15.3.10) permet d'utiliser une requête hors état RAZ.

REMARQUE : La requête peut contenir jusqu'à 132 octets.

Format de la requête

CODE REQUETE (1 octet) 0xF5	COMPLEMENT DE CODE REQUETE (1 octet) 0x51
NUMERO PROGRAMME (1 long mot) Voir REMARQUE 1	
NUMERO BLOC (1 mot)	
OFFSET BLOC (1 mot)	
LONGUEUR BLOC (1 mot)	
DATA Voir REMARQUE 2	

REMARQUE 1 : Le champ «NUMERO PROGRAMME» donne le numéro du programme pièce indexé par le groupe d'axes ((N° du programme pièce x 10) + N° du groupe d'axes).

REMARQUE 2 : Le premier caractère doit être :
- *«+» pour une insertion à la suite du bloc donné par numéro et offset,*
- *«#» pour une modification,*
- *«-» pour une Suppression.*

Le dernier caractère doit être «LF» et le champ doit comporter au plus 119 caractères.

Format des réponses positive ou négative

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x81

Pour cette requête, le code réponse et son complément ne sont pas significatif. C'est le code retourné dans la fonction uniti() (Voir 15.4.2) qui informe sur l'aboutissement de la requête.

15.3.10 Requête «RESERVE-MEMORY»

Permet de réserver l'espace mémoire d'un programme pièce déjà existant, afin de pouvoir effectuer les modification de ce programme hors état RAZ.

Format de la requête

CODE REQUETE (1 octet) 0xF5	COMPLEMENT DE CODE REQUETE (1 octet) 0x52
NUMERO PROGRAMME (1 long mot) Voir REMARQUE 1	
TAILLE MEMOIRE (1 long mot) Voir REMARQUE 2	

REMARQUE 1 :Le champ «NUMERO PROGRAMME» donne le numéro du programme pièce indexé par le groupe d'axes ((N° du programme pièce x 10) + N° du groupe d'axes).

REMARQUE 2 :Le champ «TAILLE MEMOIRE» donne la taille que l'on veut réserver pour le programme. Une valeur nulle redonne au programme une taille égale à la taille utile.

Format des réponses positive ou négative

CODE REPONSE (1 octet) 0xF5	COMPLEMENT CODE REPONSE 0x82

Pour cette requête, le code réponse et son complément ne sont pas significatif. C'est le code retourné dans la fonction uniti() (Voir 15.4.2) qui informe sur l'aboutissement de la requête.

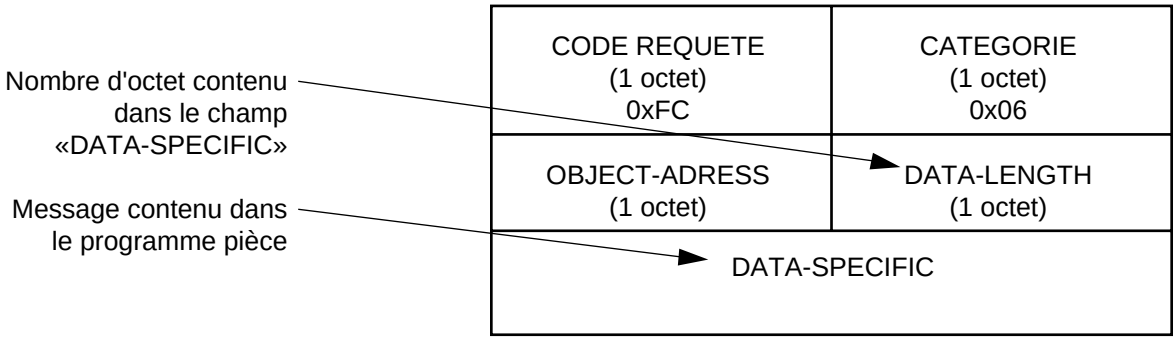
15.3.11 Requête «LECTURE DE MESSAGES»

La fonction CN émet, de sa propre initiative, cette requête, vers la fonction automatisme, suite à l'instruction «$1» ou «$11» dans le programme pièce en cours d'exécution.

Deux type de messages sont transmis par le serveur CN :
- les messages non bloquants «$1 "Message" LF»,
- les messages bloquants «$11 "Message" LF».

La fonction automatisme doit réceptionner le message qui lui est transmis par surveillance sur les portes 0x10 à 0x17 (Groupes d'axes 1 à 8) au moyen de la fonction uniti(..).

Format de la requête



Emission d'un message non bloquant

Cette requête n'attend pas de réponse de la fonction automatisme. Elle peut par contre être associée à une autre requête à l'initiative de la fonction automatisme, requête qui constituera un acquittement de la donnée non sollicitée. Le mécanisme permettant d'attendre cet acquittement est décrit au paragraphe suivant

L'instruction «Ln = $1» programmée dans le programme pièce, permet de récupérer une réponse éventuelle de la fonction automatisme.

Mécanisme de l'échange

Le message «$1 "Message" LF» est émis une seule fois à destination de la fonction automatisme et l'exécution du programme pièce se poursuit sans attente de compte rendu.

La réponse est transmise par une requête en écriture «WRITE OBJECT» avec le segment 224. La valeur de l'objet est mémorisée par la fonction CN en attendant son acquisition par le programme pièce.

Le programme pièce récupère cette valeur par la fonction $1 programmée dans une expression paramètrée du type «Ln = $1».

Si aucune valeur n'à été transmise ou si la dernière valeur transmise à déjà été acquittée, le programme pièce se met en attente d'une nouvelle écriture du segment 224.

En lecture, la CN renvoie la dernière valeur qui lui a été transmise, si elle est toujours mémorisée, c'est à dire si le programme pièce ne l'a pas encore récupérée. Dans le cas contraire, elle renvoie cette valeur complémentée à 1.

REMARQUE : Toutes tentatives de transfert de message, d'un programme pièce vers la fonction automatisme, annule la mémorisation sur le groupe d'axes considéré de la précédente écriture du segment 224.

Emission d'un message bloquant

Après l'émission d'un message bloquant «$11», la fonction CN se met en attente d'un compte rendu d'acquittement de la fonction automatisme. Tant que cette requête en écriture ne lui parvient pas, la fonction CN réémet le message toutes les 10 s et le programme pièce est mis en attente.

Mécanisme de l'échange

Le message «$11 "Message" LF» est émis à destination de la fonction automatisme et le programme pièce est en attente.

Le message est ensuite émis toutes les 10 s à destination de la fonction automatisme jusqu'à envoi d'une requête en écriture «WRITE OBJECT» avec le segment 226 = 1 (acquittement sans libération de la CN).

La réponse est transmise par une requête en écriture «WRITE OBJECT» avec le segment 224.

La fonction automatisme acquitte le message par l'envoi d'une requête en écriture «WRITE OBJECT» avec le segment 226 = 2 (acquittement et libération de la CN).

La libération de la CN permet le passage au bloc suivant et la récupération de la réponse par la fonction $11 programmée dans une expression paramètrée du type «Ln = $11».

15.4 Programmation de la fonction demandeur

15.4.1 Emission d'une requête

unito

Syntaxe de l'instruction

unito(porte_source, &datagramme)

porte_source : Numéro de la porte source.

&datagramme : Adresse du buffer à émettre.

Description

Permet d'émettre une requête vers un serveur sur les 16 portes source adressées de 0x30 à 0x3F.

Code retourné

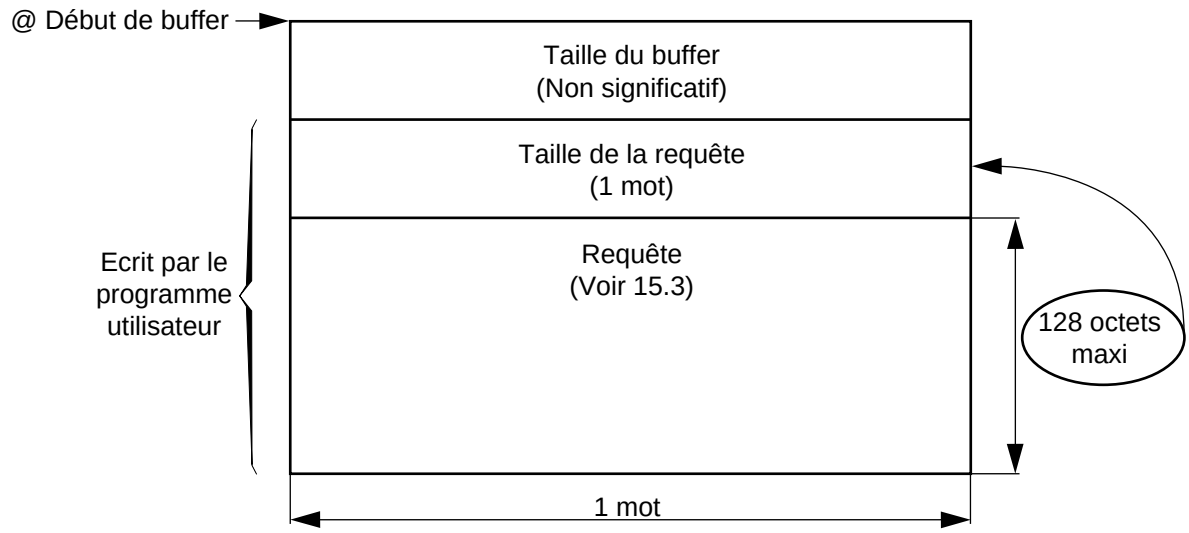
Si OK

Code	Message de compte rendu
0x00	Emission correcte

Si défaut

Code	Message de compte rendu
0x01	Longueur du buffer trop grande
0x02	Longueur du buffer nulle
0x03	File saturée - Les 16 buffers sont occupés
0x04	Mauvais numéro de porte
0x05	Option pour cette requête non valide
0xFF	Pas dans une tâche de fond

Structure du buffer d'émission



Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&datagramme" incorrect,
- "&datagramme+taille" hors zone autorisée.

15.4.2 Lecture d'une réponse

uniti

Syntaxe de l'instruction

uniti (porte_source, &datagramme)

porte_source : Numéro de la porte source.

&datagramme : Adresse du bloc mémoire qui va recevoir la requête.

Description

La fonction uniti fonctionne sur les portes 0x30 à 0x3F et 0x10 à 0x17.

Elle permet :
- de recevoir la réponse à une requête précédemment émise par unito() sur la même porte source,
- de recevoir une requête «non sollicitée» émise par le programmes pièce d'un groupe d'axe. Dans ce cas la porte source indique le groupe d'axe dont on veut recevoir un message (0x10 à 0x17).

REMARQUES : Si le code retourné est 0x06 , la fonction uniti() doit être appelée périodiquement jusqu'à la réception de la requête.

Code retourné

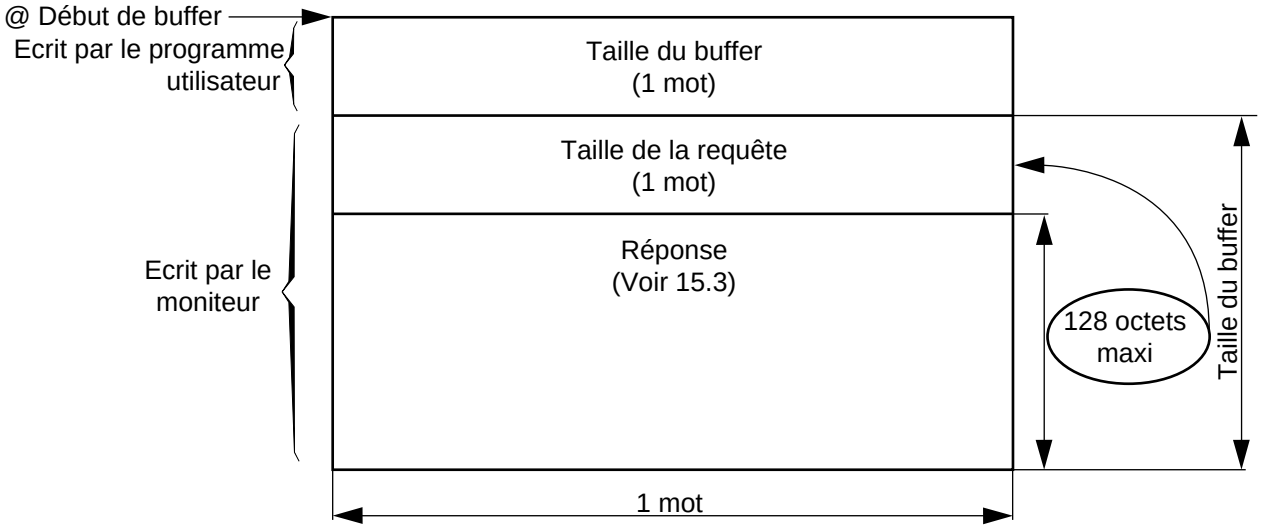
Si OK

Code	Message de compte rendu
0x00	Lecture correcte

Si défaut

Code	Message de compte rendu
0x04	Mauvais numéro de porte
0x06	Pas de message à ce numéro de porte
0x07	Buffer trop petit pour stocker la réponse
0x81	Numéro de programme existant
0x82	Edition programme en cours
0x83	Zone programme pièce saturée
0x84	Fichier fermé
0x85	Numéro de programme inexistant
0x86	Fichier ouvert
0x87	Saturation du buffer PPP
0x88	Défaut de l'en-tête Segment non reconnu Requête écriture interdite «Quantity» nulle ou négative «Objet-adress» négatif «Quantity» + «Objet-adress» supérieur au nombre de poste maxi
0x89	Taille buffer insuffisante
0x8A	Etat CN incompatible avec échange
0x8B	Données échangées incohérentes
0x8F	Indicateur de close automatique
0xFF	Pas dans une tâche de fond

Structure du buffer de réception



Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&datagramme" incorrect,
- "&datagramme+taille" hors zone autorisée.

15.4.3 Règles de programmation

⚠ ATTENTION

Les fonction unito(), uniti(), neto() et neti() doivent être programmées dans une tâche de fond (%TF0 à %TF15).

La taille d'une requête ne doit être ni nulle ni supérieure à 128 Octets (Sauf pour les requêtes READ_BLOCK et WRITE_BLOCK).

REMARQUE : 16 portes sont accessibles au programme utilisateur ce qui permet à chaque cycle HTR d'émettre un maximum de 16 requêtes.

L'émission successive de plusieurs requêtes sur la même porte, sans acquisition de la réponse, engendre la perte d'un ou plusieurs codes réponses.

Il est important :
- de programmer, sur la même porte, l'émission vers le serveur suivie de la réception de la réponse avant toutes autres émissions,
- ou d'utiliser des portes différentes pour chaque commande d'émission réception.

Coté serveurs, la notion de file d'attente peut entraîner un décalage dans la chronologie de traitement de requêtes.En effet, la charge sur les différents serveurs peut faire que les requêtes ne sont pas traitées par celui-ci dans l'ordre d'émission.

Si on veut assurer une chronologie dans le traitement des requêtes, il faut s'assurer de la bonne réception de la réponse avant d'émettre la requête suivante sur une même porte.

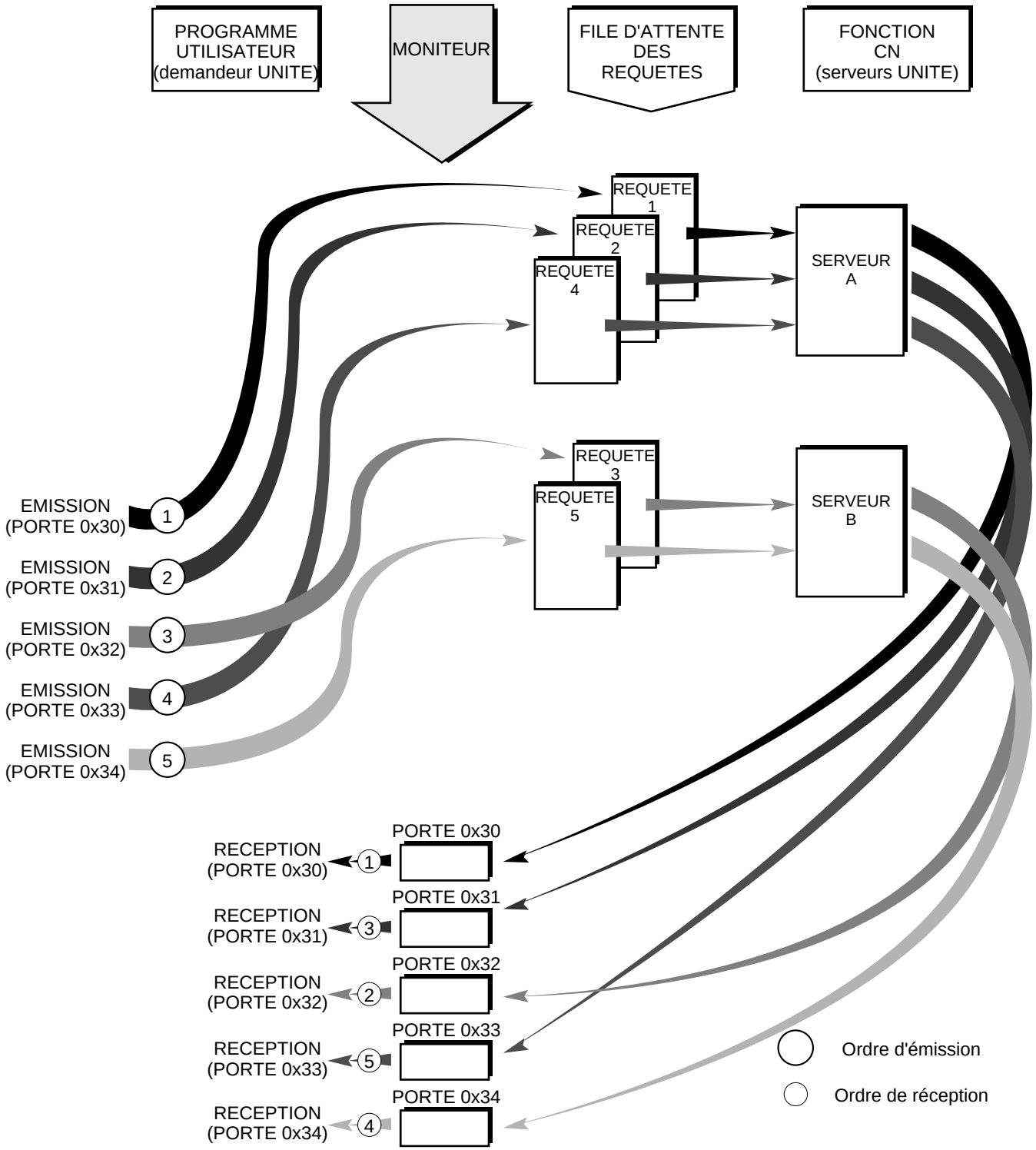


Figure 15.4 - Traitement des requêtes par les serveurs

15.5 Echanges avec une station distante

15.5.1 Emission d'une requête

neto

Syntaxe de l'instruction

neto (porte_source, &datagramme)

porte_source : Numéro de la porte source.

&datagramme : Adresse du buffer à émettre.

Description

Permet d'émettre une requête vers une station distante. La requête est émise sur une des 16 portes sources adressées 0x50 à 0x5F.

Fonctionnement

Lors de l'appel de la fonction neto(..), si la porte source est valide (Comprise entre 0x50 et 0x5F) et si l'adressage série 7 (Réseau, station, porte, module, voie) est valide, le moniteur exécute :
- l'envoi de la requête au serveur destinataire et le retour à l'appelant (Code retour OK) si la voie d'émission est libre,
- le retour à l'appelant (Code retour SATURATION) si la voie d'émission est saturée.

Si une erreur de programmation est détectée, retour à l'appelant avec code retour indiquant l'erreur détectée.

Il est possible d'échanger simultanément autant de requêtes que de portes source disponible.

Code retourné

Si OK

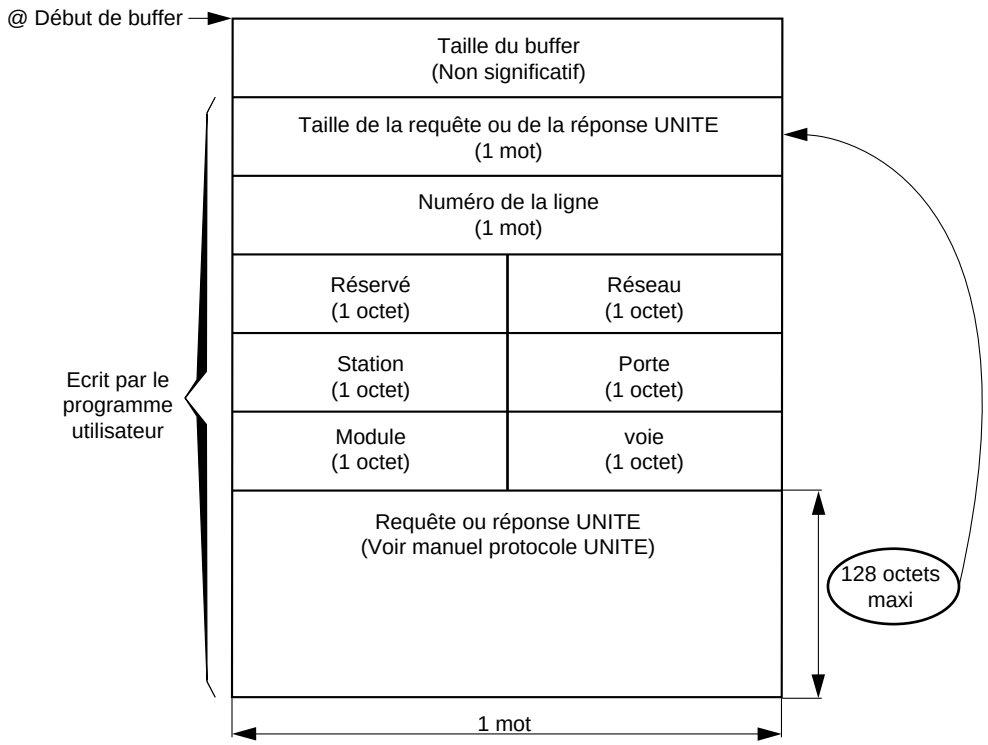
Code	Message de compte rendu
0x00	Emission correcte

Si défaut

Code	Message de compte rendu
0x01	Longueur du buffer trop grande
0x02	Longueur du buffer nulle
0x03	File saturée - Les 16 buffers sont occupés
0x04	Mauvais numéro de porte
0x05	Option pour cette requête non valide
0x08	Numéro de ligne non valide
0xFF	Pas dans une tâche de fond

REMARQUE : Les champs Réseau, Station, Porte, Module et Voie correspondent à l'adressage Série 7 Telemecanique qui désigne le destinataire de la requête. Se reporter au manuel réseau correspondant.

Structure du buffer d'émission



N° ligne	Carte processeur machine	1 ère carte IT/lignes séries	2 ème carte IT/lignes séries	Coupleur spécifique
UNI-TELWAY	0x20 et 0x21	0x24 à 0x27	0x28 à 0x2B	
MAPWAY - ETHWAY				0x30
ETHERNET				0x40

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&datagramme" incorrect,
- "&datagramme+taille" hors zone autorisée.

15.5.2 Lecture d'une réponse

neti

Syntaxe de l'instruction

neti (porte_source, &datagramme)

porte_source : Numéro de la porte source.

&datagramme : Adresse du bloc mémoire qui va recevoir la réponse.

Description

La fonction neti() fonctionne sur les portes 0x50 à 0x5F.

Elle permet :

- de recevoir la réponse à une requête précédemment émise, sur le réseau, par neto(..),
- de recevoir une requête «non sollicitée» émise par une station distante.

Fonctionnement

Lors de l'appel de la fonction neti(..), s'il n'existe pas dans la file des requêtes reçues, de requête ayant une porte source identique à la porte paramétrée à l'appel de neti(..), le moniteur exécute un retour à l'appelant avec code retour «0x6»,

Si la taille réservée pour la réception du datagramme est suffisante il y a transfert de la requête à l'adresse &datagramme et retour à l'appelant avec code retour OK «0x0»

Si la taille du buffer est insuffisante, retour à l'appelant avec code retour «0x7».

Il est possible d'attendre simultanément autant de requêtes que de portes source (16 réponses à des requêtes émise par neto(..) ou «non sollicitée»).

Code retourné

Si OK

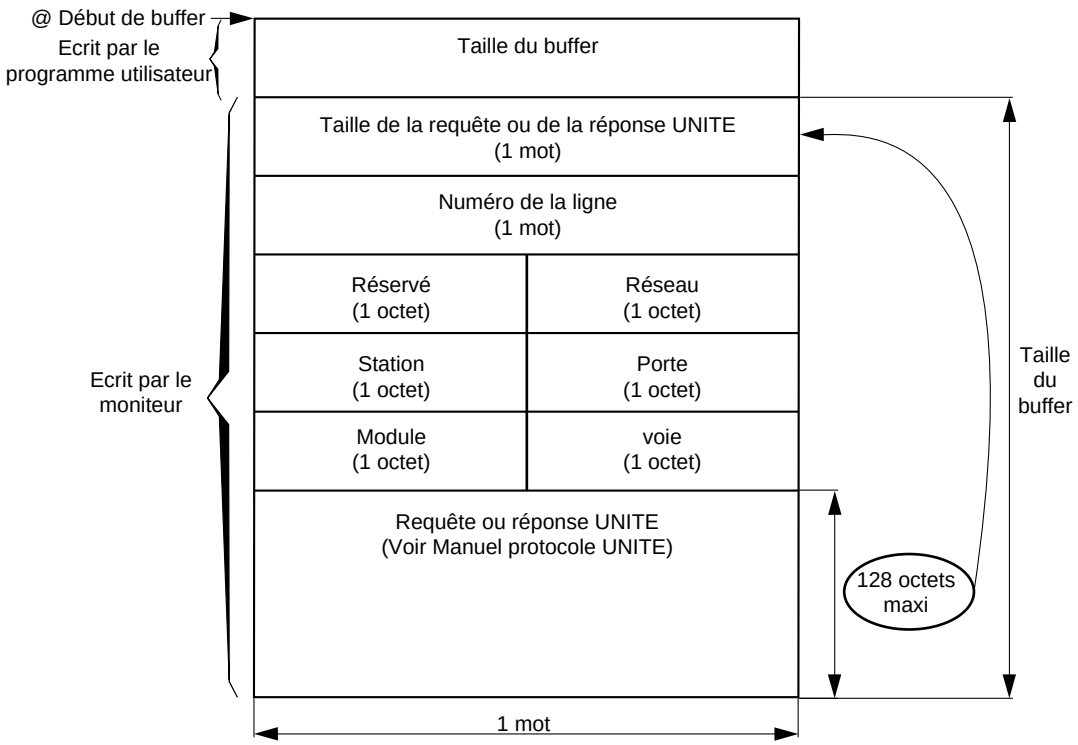
Code	Message de compte rendu
0x00	OK - Lecture correcte

Si défaut

Code	Message de compte rendu
0x04	Mauvais numéro de porte
0x06	Pas de requête reçue pour cette porte
0x07	Buffer trop petit pour stocker la réponse
0x08	Numéro de ligne non valide
0xFF	Pas dans une tâche de fond

REMARQUE : Si le code retourné est 0x06, la fonction neti(..) doit être appelée périodiquement jusqu'à réception de la requête.

Structure du buffer de réception



REMARQUE : Les champs Réseau, Station, Porte, Module et Voie correspondent à l'adressage Série 7 Telemecanique qui désigne le destinataire de la requête. Se reporter au manuel réseau correspondant.

N° ligne	Carte processeur machine	1 ère carte IT/lignes séries	2 ème carte IT/lignes séries	Coupleur spécifique
UNI-TELWAY	0x20 et 0x21	0x24 à 0x27	0x28 à 0x2B	
MAPWAY - ETHWAY				0x30
ETHERNET				0x40

REMARQUE Avec UNI-TELWAY, si la réponseest de 2 octets et le code retour 0xFF, les valeurs du code réponse peuvent être :

- 0x03 : destinataire inaccessible,

- 0x04 : NACK buffer du destinataire saturé,

- 0x0A : Time out.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&datagramme" incorrect,
- "&datagramme+taille" hors zone autorisée.

15.5.3 Exemples d'adressage série 7

SOURCE Demandeur	DESTINATION Serveur	N° ligne	Réserv.	Réseau	Station	Porte	Module	Voie
UNI-TELWAY Maitre Ligne N° L (16)	UNI-TELWAY Esclave N° S	0x20+L	0	0	0xFE	5	0xFE	0x64 +S
UNI-TELWAY Esclave Ligne N° L (1)	UNI-TELWAY Maitre	0x20+L	0	0	0xFE	0	0	0
UNI-TELWAY Esclave Ligne N° L (1)	UNI-TELWAY Maitre Porte P programme applicatif	0x20+L	0	0	0xFE	P	0	0
UNI-TELWAY Esclave Ligne N° L (1)	UNI-TELWAY Esclave N° S	0x20+L	0	0	0xFE	5	0xFE	0x64 +S
MAPWAY-ETHWAY (16)	MAPWAY-ETHWAY Station N°S de mon réseau	0x30	0	0	S	0	0	0
MAPWAY-ETHWAY (16)	MAPWAY-ETHWAY Station N°S du réseau R	0x30	0	R	S	0	0	0
MAPWAY-ETHWAY (1)	MAPWAY-ETHWAY Porte P programme applicatif Station N°S du réseau R	0x30	0	R	S	P	0	0

(1) 1 porte source (0x50 à 0x5F) autorisée par destinataire permettant 1 seul échange avec ce destinataire.

(16) 16 portes sources (0x50 à 0x5F) autorisées par destinataire permettant 16 échanges simultanés avec le même destinataire.

REMARQUE : Seule la fonction serveur pour ETHERNET-MMS est disponible.

15.5.4 Configuration du service mots communs

setcomw

Syntaxe de l'instruction

setcomw(taille, activité)

taille : Nombre d'octets attribués à chaque station.

activité : Activité de la station par rapport au mots communs (0 pour inactive, 1 pour lecture-écriture ou 2 pour lecture seule).

Description

Permet de configurer le service mots communs.

Fonctionnement

La fonction setcomw(..) doit être appelée dans la tâche %INI. Si cette fonction n'est pas appelée, le service mots communs n'est pas actif.

La taille de mots communs doit être identique sur toutes les stations du réseau. Si la valeur configurée sur une station est erronée, celle-ci sera ignorée par les autres.

Les stations devant se partager 256 mots au maximum, le choix de la taille détermine le nombre maximum de stations pouvant participer au service mots communs.

Taille par station	Nombre maxi de stations	Adresse des stations
8 octets	64	De 0 à 0x3F
16 octets	32	De 0 à 0x1F
32 octets	16	De 0 à 0xF
64 octets	8	De 0 à 7
128 octets	4	De 0 à 3

Une station inactive n'émet pas de mots communs et ne peut pas lire ceux émis par les autres stations.

Une station active en lecture n'émet pas de mots communs mais peut lire ceux émis par les autres stations.

Une station active en lecture/écriture émet ses mots communs et peut lire ceux émis par les autres stations.

Les stations déclarées en lecture/écriture doivent avoir les adresses les plus basse sur le réseau.

Il est possible de configurer un nombre de mots communs inférieur au maximum autorisé par station. Cette possibilité doit être utilisée dans les cas ou il y a peu d'informations à transmettre. Le traitement des mots communs par la fonction automatisme s'en trouvera amélioré.

Code retourné

Si OK

Code	Message de compte rendu
0x00	Configuration OK

Si défaut

Code	Message de compte rendu
0x01	Paramètre taille incorrect
0x02	Paramètre activité incorrect
0x03	Taille incompatible avec l'adresse de station (Si activité == 1)
0x04	Processeur réseau en défaut
0x05	Processeur réseau en mode test

15.5.5 Réponse à la requête STATUS

netst_ad

Syntaxe de l'instruction

netst_ad(&adresse_status)

&adresse_status Adresse du premier octet de la zone status utilisateur.

Description

Permet de définir l'adresse de la zone utilisateur ou se trouve les valeurs spécifique sur l'état de la commande numérique accessible par la requête STATUS (Code 0x31) (Voir manuel du protocole UNITE).

Fonctionnement

Cette fonction doit être appelée dans la tâche %INI.

Elle permet de définir l'adresse à partir de laquelle sont mémorisés les 16 octets du champ USER_SPECIFIC.

REMARQUE : Si cette fonction n'est pas appelée, le champ USER_SPECIFIC n'est pas significatif.

Exemple

netst_ad(%M100.&)

La requête STATUS trouvera les valeurs du champ USER_SPECIFIC à l'adresse %M100.&.

Code retourné

Si OK

Code	Message de compte rendu
0x00	Configuration OK

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&adresse_status" incorrect,
- fin du champ status hors zone autorisée.

16 Programmation en langage C

16.1 Généralités

L'utilisation du langage C pour la programmation de la fonction automatisme présente les avantages suivants :
- écriture de programmes structurés, (Emploie de noms de variables explicites, syntaxe du langage, structure de données, chaînes de caractères, .. etc ...),
- utilisation de bibliothèques de fonctions (Gestion de chaînes de caractères, calcul mathématique, .. etc ...),
- écriture de ses propres librairies de fonctions qui peuvent être utilisées pour plusieurs applications.

Toutes les tâches peuvent être écrites en langage C.

Toutes les variables de la zone d'échanges sont accessibles en programmation C.

Les fonctions exec(..) et exechdl(..) permettent la gestion d'exécutables C, issus de la chaîne de compilation C, à partir d'un module ladder.

⚠ ATTENTION

Lors de l'utilisation d'un pointeur, il faut garantir l'initialement correct de celui-ci. Si ce contrôle n'est pas effectué, il y a risque de générer un défaut "adresse interdite" Par exemple, il faut tester le compte rendu de la fonction "MALLOC" avant d'utiliser l'adresse renvoyée par celle-ci.

16.2 Appel d'un module exécutable — exec

Syntaxe de l'instruction

exec(whexec, {arg }6)

whexec : Identificateur logique du module exécutable à appeler.

arg : Eventuels arguments (étendus sur 32 bits et passés sur la pile).

Permet l'appel d'un exécutable issu d'une chaîne de compilation pour langage C.

Fonctionnement

Les arguments (signés) arg sont étendus sur 32 bits et empilés suivant la convention du langage C (le premier argument au sommet de la pile). whexec n'est pas empilé.

L'identificateur logique whexec est fourni par la fonction exechdl().

Le système analyse whexec et appelle le module exécutable associé.

Code retourné

Si OK

La valeur retournée par l'exécutable C.

16.3 Identification d’un module exécutable — exechdl

Syntaxe de l’instruction

exechdl(&chaîne)

&chaîne : Adresse d’une chaîne terminée par un octet NUL.

Permet la lecture de l’identificateur logique d’une fonction d’un module exécutable issu d’une chaîne de compilation pour langage C.

Fonctionnement

La chaîne pointée par &chaîne contient le nom d’une fonction en C.

Pour être reconnu par le moniteur, le nom d’une fonction doit être communiqué depuis un module C par la fonction EXPORT(..) (Voir 16.4.9).

La fonction exechdl() doit être appelée dans la tâche %INI.

Code retourné

Si OK

whexec > 0 : Identificateur logique du module exécutable (valeur sur 16 bits). Cet identificateur est utilisé pour l’appel de l’exécutable par la fonction exec().

Si défaut

whexec == 0: Le système ne connaît pas le nom pointé par &chaîne.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "&chaîne" incorrect,
- fin de chaîne hors zone autorisée.

16.4 Programmation en langage C

16.4.1 Notion de module

Un module est une entité exécutable indépendante. Tous les objets définis dans un module (Données ou fonctions) sont internes à celui-ci et peuvent être mis à la disposition d'une entité extérieure (Modules C, modules Ladder ou moniteur).

Pour être valide, un module doit posséder une seule fonction «main()».

Un module est issu de la compilation et de l'édition de lien d'un ou de plusieurs fichiers sources C. Il est visible et accessible sous PLCTOOL comme un fichier de type «*.XCX».

Une application peut être constituée d'un ou plusieurs modules C. Le découpage des grosses applications en plusieurs modules est fortement conseillé.

16.4.2 Interface entre les modules C

Les applications écrites en C peuvent atteindre des tailles très importantes (Plusieurs centaines de Koctets).

Une modification de l'application entraîne la compilation, l'édition des liens et le chargement de tout le module. Le temps nécessaire à toutes ces opérations peut rapidement devenir très long.

Pour optimiser les temps de traitement, il est nécessaire de séparer les gros modules en plusieurs petits modules indépendants qui peuvent s'échanger des informations de tout type (Fonctions, tableaux, structures, variables).

Ainsi, dans un module donné, on travail sur les pointeurs des objets à transférer, et on initialise ces pointeurs au moment de la «résolution des liens».

Dans les paragraphes suivants, nous utiliserons les termes :
- «objet importé» pour les objets utilisés dans un module et défini dans un autre,
- «objet exporté» pour un objet défini dans un module et mis à la disposition de tout autre module.

Le terme objet regroupe toutes les types de données globales :
- Structures,
- Fonctions,
- Variables globales,
- tableaux,
- .. etc ...

Deux fonctions, IMPORT() et EXPORT(), sont disponibles pour traiter les objets importés et exportés.

Fonctions

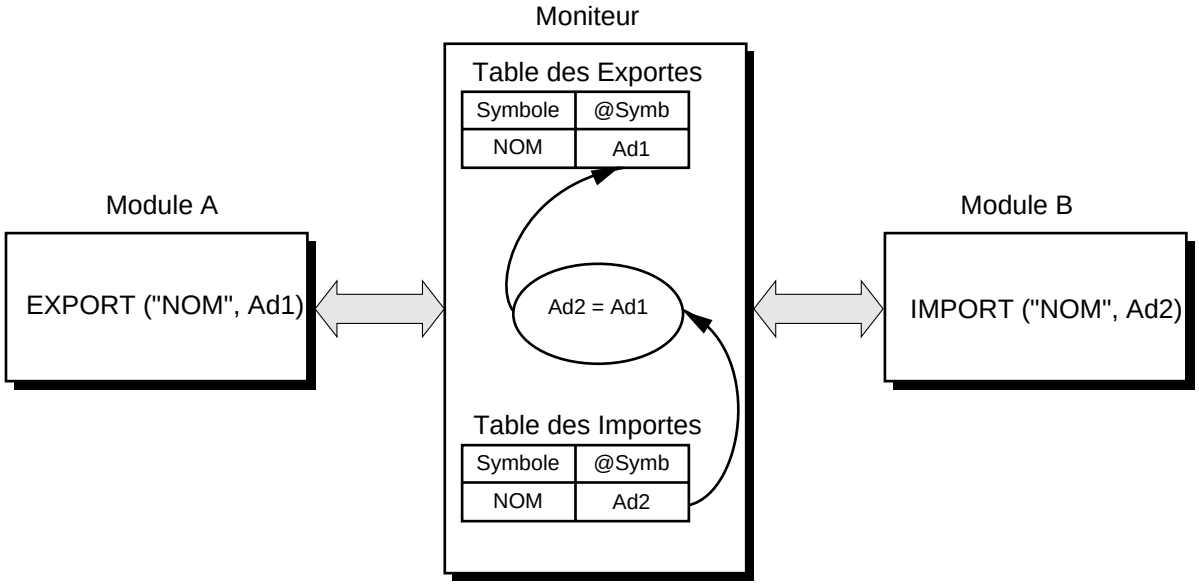
La fonction IMPORT() permet d'exploiter dans un module un objet extérieur.

La fonction EXPORT() permet de mettre à la disposition des autres modules un objet en vue de son exploitation.

> **ATTENTION**
>
> Un objet ne peut être importé que si un autre module l'a exporté.
> L'utilisateur doit définir les objets importés et exportés dans la fonction main() de ses modules.

Lorsque le traducteur est appelé, le moniteur met à jour une table dite des «exportés» et une table dite des «importés». Ces deux tables contiennent la liste des symboles et leur adresse respective. Une fois que toutes les fonctions main() de tous les modules ont été appelées, le moniteur peut faire les liens entre objets exportés et objets importés.

Le nom du symbole sert au moniteur pour lier les objets, on ne pourra, par conséquent, utiliser un même nom pour exporter deux objets différents.



NOM : Symbole (Chaîne de caractères) servant lors de l'importe

Ad1 : Adresse de l'objet à exporter (Défini dans le module A)

Ad2 : Adresse de l'objet qui sera utilisé dans le module B

16.4.3 Zone d'échanges

Toutes les variables de la zone d'échanges (Voir chapitre 3) sont accessibles en programmation C. La zone d'échanges est définie dans le fichier en-tête NUM.H.

Il est nécessaire d'inclure en en-tête des fichiers sources qui doivent utiliser la zone d'échanges le fichier NUM.H.

Le fichier NUM.H est divisé en quatre zones qui sont validées par les définitions suivantes :
- la zone d'échanges est validée par «#define VariablesLAD»,
- la zone fonctions du mode transparent est validée par «#define EcranCN»,
- la zone du formalisme grafcet est validée par «#define Grafcet»,
- la zone de gestion de fichiers est validée par «#define Gestion_Fichier».

Exemple

Pour utiliser la zone d'échanges

#define VariablesLAD

#include <NUM.H>

Pour utiliser la zone d'échanges et les fonctions du mode transparent

#define VariablesLAD

#define EcranCN

#include <NUM.H>

16.4.4 Accès aux variables internes banalisées sauvegardées

Les variables par octet, mot ou long mot sont accessibles par les mots clés suivants.

Mot clé	Valeurs	Définition
_MB(a)	0 < a < 77FF	Octet signé
_MW(a)	0 < a < 77FE	Mot signé
_ML(a)	0 < a < 77FD	Long mot signé

On peut aussi accéder à l'adresse d'une donnée (Equivalent à l'opérateur .& du ladder).

Mot clé	Valeurs	Définition
_pM(a)	0 < a < 77FF	

16.4.5 Accès aux variables internes banalisées non sauvegardées

Les variables par octet, mot ou long mot sont accessibles par les mots clés suivants.

Mot clé	Valeurs	Définition
_VB(a)	0 < a < 7FFF	Octet signé
_VW(a)	0 < a < 7FFE	Mot signé
_VL(a)	0 < a < 7FFD	Long mot signé

On peut aussi accéder à l'adresse d'une donnée (Equivalent à l'opérateur .& du ladder).

Mot clé	Valeurs	Définition
_pV(a)	0 < a < 7FFF	

16.4.6 Accès aux entrées borniers

Les variables ne sont accessibles qu'en lecture et par octet, mot ou long mot.

Mot clé	Valeurs	Définition
_IB(a,b,c)		Octet
_IW(a,b,c)		Mot
_IL(a,b,c)		Long mot

Pour tous ces mots clés, on a :
- a : N° du rack de 0 < a < 6
- b : N° de carte dans le rack de 0 < b < F
- c : Adresse logique à l'intérieur du poste de 0 < c < 3F

REMARQUE : Pour accéder à une entrée particulière, il faut masquer l'octet correspondant

16.4.7 Accès aux sorties borniers

Les variables ne sont accessibles qu'en écriture et par octet, mot ou long mot.

Mot clé	Valeurs	Définition
_QB(a,b,c)		Octet
_QW(a,b,c)		Mot
_QL(a,b,c)		Long mot

Pour tous ces mots clés, on a :
- a : N° du rack de 0 < a < 6
- b : N° de carte dans le rack de 0 < b < F
- c : Adresse logique à l'intérieur du poste de 0 < c < 3F

REMARQUE : Pour accéder à une sortie particulière, il faut masquer l'octet correspondant

16.4.8 Type des données standards

Pour plus de clarté, le type des données standards C a été redéfini.

Donnée standard	Définition
UINT32	Variable non signée sur 4 octets
UINT16	Variable non signée sur 2 octets
UINT8	Variable non signée sur 1 octet
SINT32	Variable signée sur 4 octets
SINT16	Variable signée sur 2 octets
SINT8	Variable signée sur 1 octet (caractère)

16.4.9 Les fonctions de la librairie

Une application écrite en C est générée sur système autonome. La librairie de fonctions «NUM.OBJ» permet d'avoir accès aux primitives du moniteur. Cette librairie est utilisée au moment de l'édition des liens.

Toutes ces fonctions sont prototypées dans le fichier entête «NUM.H»

16.4.9.1 Fonctions systèmes

Exportation d'un objet

EXPORT

Syntaxe

SINT32 EXPORT(SINT8 *symbole, void *ad_symbole)

symbole : Chaîne de caractère

ad_symbole : Adresse du symbole

Cette fonction rend un objet C visible par tous les autres modules (visibilité globale) ou associe une fonction à une tâche automate.

Compte rendu:
- 0 = OK
- -1 = tâche déjà définie ou trop de symboles d'exportation.

Exemple 1

Cet EXPORT a pour effet d'associer ts01_en_c à la tâche TS01.

```
main()
{
                EXPORT(«TS01»,ts01_en_c);
}
void ts01_en_c()
{
                corps de la fonction
}
```

Exemple 2

Cet EXPORT aura pour effet de donner à «tableau» une visibilité globale.

```
SINT16 tableau[100];
main()
{
                EXPORT(«ETIQUETTE»,tableau);
}
```

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*symbole" incorrect,
- paramètre "*ad_symbole" incorrect,
- fin de chaîne "symbole" hors zone autorisée.

IMPORT

Importation d'un objet

Syntaxe

SINT32 IMPORT(SINT8 *symbole, void **ad_symbole)

symbole : Chaîne de caractère

ad_symbole : Pointeur de pointeur du symbole

Cette fonction permet d'utiliser un objet défini dans un autre module. Un objet importé doit être exporté préalablement dans un autre module.

Compte rendu:
- 0 = OK
- -1 = appel en dehors de la tâche d'initialisation ou trop de symboles d'importation.

Exemple

```
void (* fonction_IMPORT)();
main()
{
        IMPORT(«ETIQUETTE», &fonction_IMPORT);
}
void essai()
{
        fonction_IMPORT();
        corps de la fonction
}
```

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*symbole" incorrect,
- paramètre "**ad_symbole" incorrect,
- fin de chaîne "symbole" hors zone autorisée.

16.4.9.2 Exploitation des fonctions systèmes

Exemple «EXPORT d'un tableau»

Dans le fichier «EXP.C»

```
#include <NUM.H>
SINT16 tableau [100];
main()
{
                EXPORT(«ETIQUETTE» , tableau);
}
```

Dans le fichier IMP.C

```
#include <NUM.H>
SINT16 *ptab;
main()
{
                IMPORT(«ETIQUETTE» , &ptab);
}
```

Exemple «EXPORT d'une fonction»

Dans le module 1

```
#include <NUM.H>
SINT16 Affiche(UINT8 quoi , SINT16 combien)
{
                SINT16 i ;
                for (i = 0 ; i < combien ; i++);
                {
                        EMIV(quoi) ;
                }
                return(i);
}
main()
{
                EXPORT(«FONCT1» , Affiche);
}
```

Dans le module 2

```
#include <NUM.H>
SINT16 (* FoncImp) (UINT8 , SINT16);
/* FoncImp : pointeur sur une fonction nécessitant deux paramètres et qui retourne un SINT16 */
void fonct2()
{
                    PCUR(5 , 2);
                    FoncImp(«_» , 10);
                    /* Exécution de la fonction importée */
}
main()
{
                    IMPORT(«FONCT1» , &FoncImp);
                    /* Initialisation du pointeur sur la fonction extérieure */
}
```

16.4.9.3 Fonctions d'échanges par protocole

Le fonctionnement et les paramètres de ces fonctions sont identiques à ceux des fonctions ladder (Voir chapitre 15).

Lecture d'une réponse d'un serveur distant **NETI**

Syntaxe

UINT8 NETI(UINT8 porte, UINT8 *ad_buffer)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*datagramme" incorrect,
- "*datagramme+taille" hors zone autorisée.

Envoie d'une requête UNITE vers un serveur distant **NETO**

Syntaxe

UINT8 NETO(UINT8 porte, UINT8 *ad_buffer)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*datagramme" incorrect,
- "*datagramme+taille" hors zone autorisée.

Lecture d'une réponse en interne

UNITI

Syntaxe

UINT8 UNITI(UINT8 porte_source, UINT8 *datagramme)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*datagramme" incorrect,
- "*datagramme+taille" hors zone autorisée.

Envoie d'une requête UNITE en interne

UNITO

Syntaxe

UINT8 UNITO(UINT8 porte_source, UINT8 *datagramme)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*datagramme" incorrect,
- "*datagramme+taille" hors zone autorisée.

16.4.9.4 Gestion des lignes séries

Le fonctionnement et les paramètres de ces fonctions sont identiques à ceux des fonctions ladder (Voir chapitre 12).

Contrôle du pilote de ligne série

COMCTL

Syntaxe

SINT8 COMCTL(UINT8 n_port, UINT8 config)

Sélection des vitesses et formats

COMF

Syntaxe

SINT32 COMF(UINT8 n_port, UINT16 vitemi, UINT16 vitrec, UINT16 format)

Lecture du tampon de réception

COMIN

Syntaxe

SINT16 COMIN(UINT8 n_port, UINT8 *buffer, UINT16 nb)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*buffer" incorrect,
- "*buffer+nb" hors zone autorisée.

Emission d'un tampon

COMOUT

Syntaxe

SINT8 COMOUT(UINT8 n_port, UINT8 *buffer, UINT16 nb)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*buffer" incorrect,
- "*buffer+nb" hors zone autorisée.

Lecture de l'état d'une ligne série

COMREG

Syntaxe

UINT16 COMREG(UINT8 n_port)

16.4.9.5 Gestion du mode transparent

Le fonctionnement et les paramètres de ces fonctions sont identiques à ceux des fonctions ladder (Voir chapitre 8).

Emission d'un caractère vers la visu

EMIV

Syntaxe

void EMIV(UINT8 char)

char : Caractère ou code du caractère (Voir 8.3.4.1).

INIG

Initialisation des caractéristiques de l'écran graphique

Syntaxe

```
void INIG()
```

MAIN_WINDOW

Sélection de la fenêtre principale

Syntaxe

```
void MAIN_WINDOW()
```

STATUS_WINDOW

Sélection de la fenêtre paramétrable

Syntaxe

```
void STATUS_WINDOW()
```

KEY_WINDOW

Sélection de la fenêtre cartouche

Syntaxe

```
void KEY_WINDOW()
```

PCUR

Positionnement du curseur

Syntaxe

```
void PCUR(UINT8 ligne, UINT8 col)
```

PICO

Tracé d'un trait

Syntaxe

```
void PICO(UINT8 Type_Trait, UINT16 X, UINT16Y)
```

Type_Trait : Type de trait pour le tracé

Type_trait	Type de trait
0	Trait continu
1	Trait pointillé
2	Trait tireté
3	Trait mixte
4	Absence de trait

X,Y : Position du point d'arrivé (En pixels).

Tracé d'une flèche

FLEC

Syntaxe

void FLEC(UINT8 Code, UINT16 Longueur, UINT16 Largeur)

Code : Orientation de la flèche

Code	Type de flèche
1	Pointe vers la droite
2	Pointe vers la gauche
3	Pointe vers le haut
4	Pointe vers le bas

Longueur : Longueur de la pointe (En pixels).

Largeur : Largeur de la pointe (En pixels).

Tracé d'un rectangle

RECT

Syntaxe

void RECT(UINT16 Largeur, UINT16 Longueur)

Largeur : Largeur du rectangle (En pixels).

Longueur : Longueur du rectangle (En pixels).

CERC

Tracé d'un cercle

Syntaxe

void CERC(UINT16 Rayon)

Rayon : Rayon du cercle (En pixels).

LOSA

Tracé d'un losange

Syntaxe

void LOSA(UINT16 Largueur, UINT16 DemiHauteur, UINT16 HauteurTotale)

ARCA

Tracé d'un arc de cercle dans le sens antitrigonométrique

Syntaxe

void ARCA(UINT8 Type_Trait, UINT16 Xarr, UINT16 Yarr, UINT16 Xcentre, UINT16 Ycentre)

TypeTrait : Type de trait pour le tracé
Xarr, Yarr : Coordonnées du point d'arrivé (En pixels).
Xcentre, Ycentre : Coordonnées du centre (En pixels).

ARCT

Tracé d'un arc de cercle dans le sens trigonométrique

Syntaxe

void ARCT(UINT8 Type_Trait, UINT16 Xarr, UINT16 Yarr, UINT16 Xcentre, UINT16 Ycentre)

TypeTrait : Type de trait pour le tracé
Xarr, Yarr : Coordonnées du point d'arrivé (En pixels).
Xcentre, Ycentre : Coordonnées du centre (En pixels).

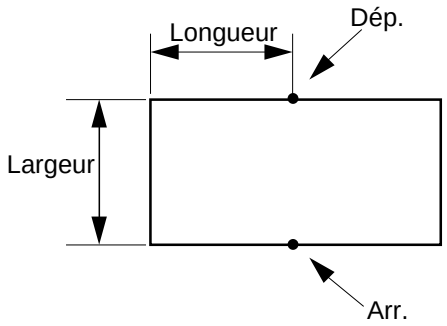
TEST

Tracé d'un icône de test.

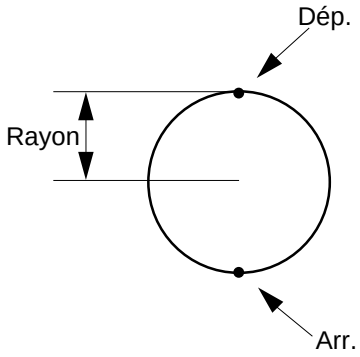
Syntaxe

void TEST(UINT16 DemiBase, UINT16 DemiLargeur, UINT16 DemiHauteur, UINT16 Hauteur)

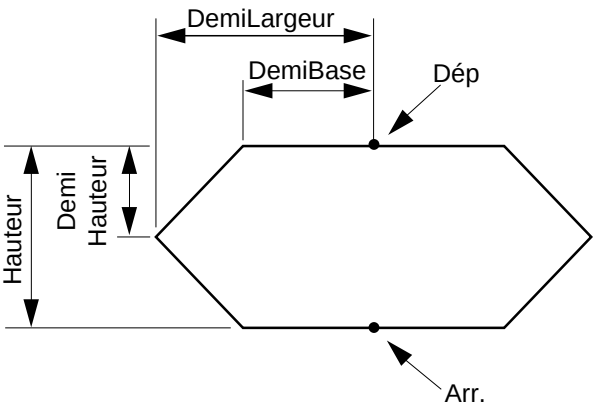
RECT : Rectangle



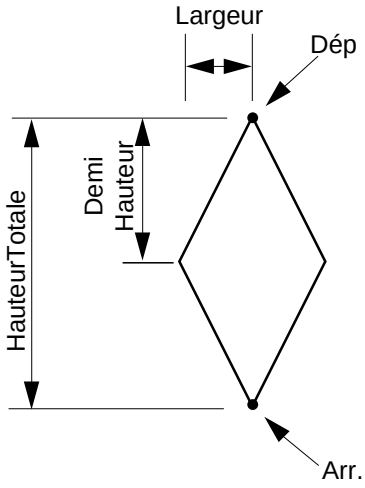
CERC : Cercle



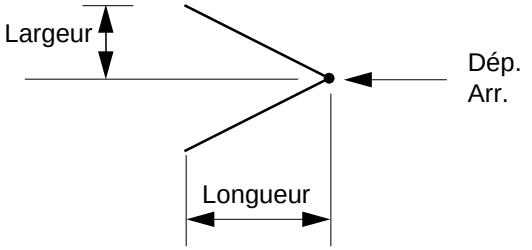
TEST : Test



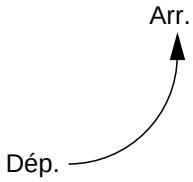
LOSA : Losange / Triangle



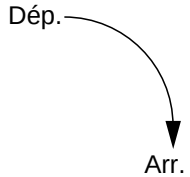
FLEC : Flèche

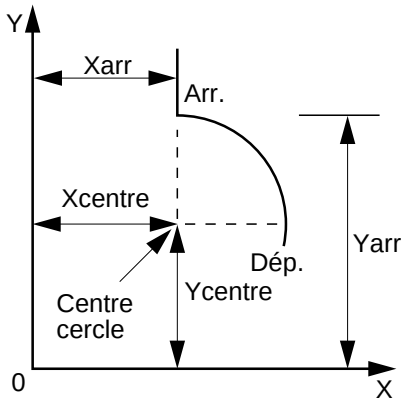


ARCT



ARCA





COLOR

Coloriage avec arrêt sur contour

Syntaxe

void COLOR(UINT8 Couleur, UINT16 X, UINT16 Y)

Couleur : Code de couleur du contour, de 0 à 15 (Voir 8.3.3.2).

X, Y : Position de début de coloriage (En pixels).

SELCOL

Sélection d'une couleur

Syntaxe

void SELCOL(UINT8 Couleur)

PUTKEY

Simulation du clavier pupitre

Syntaxe

SINT32 PUTKEY(UINT8 Code_touche)

SCANO

Ouverture d'une acquisition clavier

Syntaxe

SINT32 SCANO(UINT8 *Question, UINT16 Largeur)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*question" incorrect,
- fin de chaîne hors zone autorisée.

SCANU

Ouverture d'une acquisition clavier numérique

Syntaxe

SINT32 SCANU(UINT8 *Question, UINT16 Largeur)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*question" incorrect,
- fin de chaîne hors zone autorisée.

Acquisition d’une chaîne de caractères **SCANS**

Syntaxe

SINT32 SCANS(UINT8 *Dest)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*dest" incorrect,
- fin du champ acquisition hors zone autorisée.

Acquisition et conversion d’un nombre en décimal **SCAND**

Syntaxe

SINT32 SCAND(UINT32 *Lvariable)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*Ivariable" incorrect.

Acquisition et conversion d’un nombre en hexadécimal **SCANX**

Syntaxe

SINT32 SCANX(UINT32 *Lvariable)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*Ivariable" incorrect.

Fermeture d’une acquisition clavier **SCANC**

Syntaxe

SINT32 SCANC()

16.4.9.6 Programmation des entrées/sorties analogiques

Le fonctionnement et les paramètres de ces fonctions sont identiques à ceux des fonctions ladder (Voir chapitre 9)

Redirection d’une carte analogique **ANAA**

Syntaxe

SINT32 ANAA(UINT8 Cv_Initial, UINT8 Cv_Final)

Lecture d'une entrée analogique **ANAI**

Syntaxe

SINT32 ANAI(UINT8 cv, SINT16 *winput)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*winput" incorrect.

Ecriture d'une sortie analogique **ANAO**

Syntaxe

SINT32 ANAO(UINT8 cv, SINT16 woutput)

Configuration d'une carte E/S analogique **ANAS**

Syntaxe

SINT32 ANAS(UINT8 cv, SINT16 wconfig)

16.4.9.7 Lectures/Ecritures explicites

Le fonctionnement et les paramètres de ces fonctions sont identiques à ceux des fonctions ladder (Voir chapitre 10).

Lecture explicite d'une carte entrée **READ_I**

Syntaxe

SINT8 READ_I(UINT16 rcmv, UINT8 n)

Ecriture explicite d'une carte sortie **WRITE_Q**

Syntaxe

SINT8 WRITE_Q(UINT16 rcmv, UINT8 n)

16.4.9.8 Programmation des entrées interruptions

Le fonctionnement et les paramètres de ces fonctions sont identiques à ceux des fonctions ladder (Voir chapitre 11).

Configuration d'une entrée interruption **ITICTL**

Syntaxe

```
SINT32 ITICTL(UINT32 Numéro_iti, UINT8 iti_config)
```

Lecture d'une entrée interruption **ITIGET**

Syntaxe

```
UINT8 ITIGET(UINT8 Numéro_iti)
```

Association une entrée interruption avec un groupe d'axes **ITI_GR**

Syntaxe

```
SINT32 ITI_GR( UINT32 Numéro_iti, UINT32 Groupe)
```

Association d'une tâche hard avec une entrée interruption **THITI**

Syntaxe

```
SINT32 THITI(UINT32 Numéro_Th, UINT32 Numéro_iti)
```

16.4.9.9 Gestion des tâches de fond

Le fonctionnement et les paramètres de ces fonctions sont identiques à ceux des fonctions ladder (Voir chapitre 7).

Début d'une section critique **CSBEGIN**

Syntaxe

```
void CSBEGIN(void)
```

Fin d'une section critique **CSEND**

Syntaxe

```
void CSEND(void)
```


Départ d'une tâche de fond **TFSTART**

Syntaxe

```
SINT32 TFSTART(UINT16 Numero_tf)
```

Arrêt d'une tâche de fond **TFSTOP**

Syntaxe

```
SINT32 TFSTOP(UINT16 Numero_tf)
```

Mise en sommeil d'un tâche de fond pendant n cycle automate **WHTR**

Syntaxe

```
void WHTR(UINT16 n)
```

16.4.9.10 Fonctions d'usage général

Le fonctionnement et les paramètres de ces fonctions sont identiques à ceux des fonctions ladder correspondantes (Voir chapitre 6).

Mise à jour de la sortie analogique N°0 **CNA0**

Syntaxe

```
void CNA0(SINT16 valeur)
```

Mise à jour de la sortie analogique N°1 **CNA1**

Syntaxe

```
void CNA1(SINT16 valeur)
```

Recherche circulaire optimale — QCKTOOL

Syntaxe

SINT32 QCKTOOL(SINT32 origine, SINT32 destination, SINT32 n)

Correction dynamique d'un outil — TOOLDYN

Syntaxe

SINT32 TOOLDYN(SINT16 correction, UINT8 axe, UINT8 n_outil)

Temporisation de type enclenchement — TEMPO_ENCLENCHEMENT

Syntaxe

SINT32 TEMPO_ENCLENCHEMENT(SINT8 Instance, UINT8 Entrée, SINT32 Seuil)

Temporisation de type déclenchement — TEMPO_DECLENCHEMENT

Syntaxe

SINT32 TEMPO_DECLENCHEMENT(SINT8 Instance, UINT8 Entrée, SINT32 Seuil)

Temporisation de type impulsion — TEMPO_IMPULSION

Syntaxe

SINT32 TEMPO_IMPULSION(SINT8 Instance, UINT8 Entrée, SINT 32 Seuil)

Appel d'un sous programme — SP

Syntaxe

void SP(UINT8 n_module, argn)

Semaphore

SEMA

Syntaxe

SINT8 SEMA(SINT8*semaphore)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*semaphore" incorrect,

Lecture de n variables E42000

R_E42000

Syntaxe

SINT32 R_E42000(SINT8 *dest, UINT32 numero, UINT32 n)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*dest" incorrect,
- "*dest+n" hors zone autorisée.

Ecriture de n variables E42000

W_E42000

Syntaxe

SINT32 W_E42000(SINT8 *source, UINT32 numero, UINT32 n)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*source" incorrect,
- "*source+n" hors zone autorisée.

Lecture de la date courante

TMGET

Syntaxe

TMGET(*date)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*date" incorrect.

Lecture de la date courante et jour de la semaine

DTGET

Syntaxe

DTGET(*date)

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*date" incorrect.

16.4.9.11 Gestion de fichiers

La mémoire globale est gérée comme une unité de disque. Les données y sont stockées sous forme de fichiers rangés dans des répertoires.

Il existe trois répertoires destinés à recevoir les différents types de fichiers :
- les fichiers applicatifs «LADDER» ayant une extension .XLA,
- les fichiers applicatifs «C» ayant une extension .XCX,
- les fichiers utilisateur.

La programmation en C offre la possibilité de créer ses propres fichiers. Un certain nombre de primitives sont à sa disposition pour la gestion de ces fichiers.

Les codes d'erreurs retournés par les primitives (fichiers et répertoire) sont :

Code d'erreur	Description
UF_SYSFAIL	Erreur système
UF_DSKFULL	Disque plein
UF_ERRNAME	Nom de fichier incorrect
UF_NEXIST	Fichier inexistant
UF_OPEN	Fichier ouvert
UF_NOPEN	Fichier non ouvert

Création d'un fichier dans le répertoire utilisateur

USER_CREATE_F

Syntaxe

SINT32 USER_CREATE_F(SINT8 *pNom_Fichier)

Paramètres

Entrées :	pNom_Fichier : Le répertoire ne doit pas être spécifié car les fichiers utilisateurs sont forcément dans un répertoire figé.
Sortie :	Aucune

*REMARQUES : Si le nom du fichier existe déjà, la fonction retourne le code d'erreur «UF_ERRNAME». Un fichier existant doit préalablement être effacé par la fonction «*USER_REMOVE_F*» avant que le nom soit réutilisé pour un autre fichier.*
*Après exécution de la fonction «*USER_CREATE_F*», le nouveau fichier est vide.*
*La fonction «*USER_CREATE_F*» n'est utilisée que pour les fichiers de données.*
*La fonction «*USER_CREATE_F*» n'ouvre pas automatiquement le fichier. Il faut utiliser la fonction «*USER_OPEN_F*» pour réaliser cette ouverture.*

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*pNom_fichier" incorrect.

Suppression d'un fichier

USER_DELETE_F

Syntaxe

SINT32 USER_DELETE_F(SINT8 *pNom_Fichier)

Paramètres

Entrées : pNom_Fichier : Le répertoire ne doit pas être spécifié car les fichiers utilisateurs sont forcément dans un répertoire figé.

Sortie : Aucune

REMARQUE : Si un fichier est toujours ouvert, il ne sera pas effacé.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*pNom_fichier" incorrect.

Ouverture d'un fichier

USER_OPEN_F

Syntaxe

SINT32 USER_OPEN_F(UINT32 *pF_Id, SINT8 *pNom_Fichier)

Description

Ouvre un fichier identifié par pNom_Fichier. Ce fichier est accessible en lecture et écriture. Si l'opération se déroule normalement, le gestionnaire de fichiers retourne un identificateur pF_Id utilisé par les fonctions «USER_CLOSE_F», «USER_READ_F», «USER_WRITE_F» et «user_seek_f».

Paramètres

Entrées : pNom-Fichier : Le répertoire ne doit pas être spécifié car les fichiers utilisateurs sont forcément dans un répertoire figé.

Sortie : pF_Id : Identificateur de fichier si l'opération s'est bien déroulé.

*REMARQUES : La fonction «*USER_OPEN_F*» ne vérifie pas le type du fichiers.*
*La fonction «*USER_OPEN_F*» positionne le pointeur de fichier sur le 1er octet du fichier.*

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*pF_id" incorrect,
- paramètre "*pNom_Fichier" incorrect.

Fermeture d'un fichier

USER_CLOSE_F

Syntaxe

SINT32 USER_CLOSE_F(UINT32 F_Id)

Description

Ferme un fichier préalablement ouvert par la fonction «USER_OPEN_F».

Paramètres

Entrées : pF_Id : Identificateur de fichier

Sorties : Aucune

*REMARQUE : Le nombre de fichier ouvert simultanément étant limité, la fonction «*USER_CLOSE_F*» doit être utilisée dés que l'ouverture d'un fichier n'est plus nécessaire.*

Lecture des données d'un fichier

USER_READ_F

Syntaxe

SINT32 USER_READ_F(UINT32 pF_Id, UINT8 *pBuf, UINT32 Nb_Demande, UINT32 *Nb_lus)

Paramètres

Entrées : pF_Id : Identificateur de fichier retourné par user_open_f.
pBuf : Buffer de réception des données.
Nb_Demande : Nombre d'octets à lire.

Sorties : Nb_Lus : Nombre d'octets effectivement lus. Si ce nombre est inférieur au nombre demandé, cela signifie que la fin de fichier à été atteinte.

REMARQUES : Les données sont lues à partir de la position courante du pointeur de fichier.
Le pointeur de fichier est automatiquement repositionné après la lecture.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*pBuf" incorrect,
- "*pBuf+nb_Demandes" hors zone autorisée,
- paramètre "*nb_lus" incorrect.

Ecriture d'un fichier

USER_WRITE_F

Syntaxe

SINT32 USER_WRITE_F(UINT32 pF_Id, UINT8 *pBuf, UINT32 Taille_Buf)

Description

Ecrit des données dans le fichier spécifié par «pF_Id». L'écriture commence toujours au niveau du pointeur de fichier. Après l'écriture, le pointeur est positionné sur le dernier octet du fichier.

Paramètres

Entrées : pF_Id : Identificateur de fichier retourné par la fonction «USER_OPEN_F».
pBuf : Buffer des données à écrire.
Taille_Buf : Taille du buffer.

Sorties : Aucune

REMARQUES : Les données sont écrites à partir de la position courante du pointeur de fichier. Le pointeur de fichier est automatiquement repositionné après l'écriture.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*pBuf" incorrect,
- "*pBuf+nb_demandes" hors zone autorisée.

Repositionnement du pointeur

USER_SEEK_F

Syntaxe

SINT32 USER_SEEK_F(UINT32 pF_Id, UINT32 mode, SINT32 offset, UINT32 *Old_Ptr)

Description

Repositionne pour le pointeur pour lecture écriture dans un fichier spécifié par «pF-Id».

Paramètres

Entrées : pF_Id : Identificateur de fichier retourné par la fonction «USER_OPEN_F».
mode : Position à partir de laquelle on fait le déplacement 0 déplacement à partir du début du fichier.
1 déplacement à partir de la position courante.
2 déplacement à partir de la fin du fichier.
offset : Déplacement signé relatif au mode choisi.

Sortie : Old_Ptr : Valeur initiale du pointeur.

REMARQUES : Le pointeur est différent pour chaque fichier.
Le pointeur est une variable non signé. Un déplacement en dehors de limites du fichier, génère une erreur.

Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*pOld_Ptr" incorrect,

16.4.9.12 Gestion de répertoire

Seul le répertoire utilisateur est accessible. Trois primitives permettent de connaître le contenu de ce répertoire.

Ouverture du répertoire utilisateur **USER_OPEN_DIR**

Syntaxe

SINT32 USER_OPEN_DIR()

Paramètres

Entrées : Aucune

Sortie : Aucune

REMARQUE : Un répertoire est ouvert en lecture seulement.

Fermeture du répertoire utilisateur **USER_CLOSE_DIR**

Syntaxe

SINT32 USER_CLOSE_DIR()

Description

Ferme le répertoire utilisateur préalablement ouvert avec la primitive «USER_OPEN_DIR».

Paramètres

Entrées : Aucune

Sorties : Aucune

Lecture des données du répertoire utilisateur **USER_READ_DIR**

Syntaxe

SINT32 USER_READ_DIR(UINT8 *pBuf, UINT32*Nb_Demande, UINT32*Nb_lus)

Paramètres

Entrées : pBuf : Buffer de réception des données.
Nb_Demande : Nombre d'octets à lire.

Sorties : Nb_Lus : Nombre d'octets effectivement lus. Si ce nombre est inférieur au nombre demandé, cela signifie que la fin du répertoire à été atteinte.

Description d'un fichier du répertoire

Un fichier du répertoire est décrit par 32 octets organisés comme suit :

Nombre d'octet	Description
8 octets	Nom du fichier Si Nom_fichier[0] prend les valeurs suivante, alors : 0x00 signifie fin de répertoire 0x2E signifie Fichier système 0xE5 signifie Fichier détruit 0x05 signifie Le nom commence par 0x05
3 octets	Extension du fichier
1 octet	Attribut du fichier Bit 0 = 1 : Lecture seule Bit 1 = 1 : Fichier caché Bit 2 = 1 : Fichier système Bit 3 = 1 : Nom du volume (ROOT) Bit 4 = 1 : Fichier répertoire Bit 5 = 1 : Bit d'archive Bit 6 = 1 : Réservé Bit 7 = 1 : Réservé
Nombre d'octet	Description
10 octets	Réservé
2 octets	Heure au format INTEL
2 octets	Date au format INTEL
2 octets	Cluster de départ au format INTEL
4 octets	Taille du fichier au format INTEL

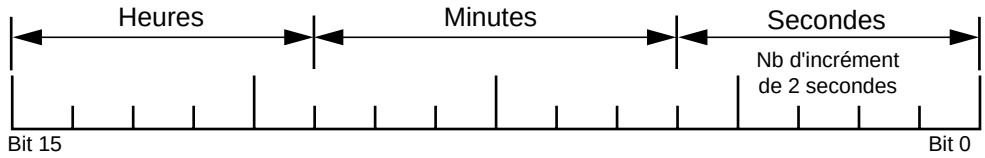
Erreur de programmation provoquant la mise en défaut de l'unité centrale

Accès à une adresse interdite :
- paramètre "*pBuf" incorrect,
- "*pBuf+Nb_Demande" hors zone autorisée,
- paramètre "Nb_lus" incorrect.

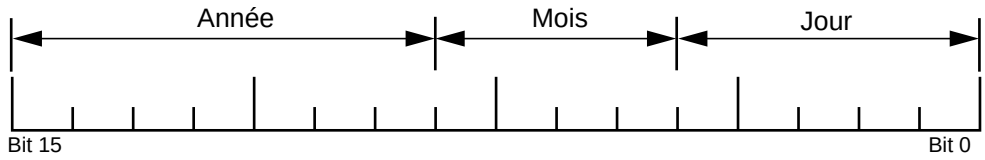
L'heure, la date et la taille sont codées au format INTEL c'est à dire que les octets de poids fort et les octets de poids faible sont inversés par rapport au format MOTOROLA.

Au format MOTOROLA, les bits ont la signification suivante :

Pour l'heure



Pour la date



Pour faciliter le traitement de ces données, on peut utiliser les structures suivantes (définies dans le fichier d'en-tête NUM.H) :

```
struct S_HEURE_FICHIER
{
                UINT8 Sec   :5;     /* Seconde */
                UINT8 Min   :6;     /* Minutes */
                UINT8 Heure         :5;     /* Heures */
};
struct S_DATE_FICHIER
{
                UINT8 Jour  :5;     /* jour */
                UINT8 Mois :4;      /* Mois */
                UINT8 Annee         :7;     /* Années */
};
struct S_ELEMENT_REPERTOIRE
{
                UINT8 Nom_Fichier[8];
                UINT8 Extension_Nom[3];
                UINT8 Attribut;
                UINT8 Reserve[10];
                UINT16 Heure_Intel;         /* Mot au format INTEL */
                UINT16 Date_Intel /* Mot au format INTEL */
                UINT16 Cluster_Depart;
                UINT32 Taille_Fichier_Intel;      /* Long mot au format INTEL */
};
```

17 Axes automates

17.1 Présentation

La fonction axes automates permet à la fonction automatisme de commander des axes.

Ces axes dit «axes automates» sont pilotés par la fonction CN.

En fonction du système 1060, la fonction CN pilote un maximum de :

	Maximum axe	Maximum groupe
NUM1060 série I	32	8 (9 axes par groupe)
NUM 1060 série II	8	3 (9 axes par groupe)

Les axes automates sont rangés dans les groupes automates au sein desquels ils sont interpolables.

Le nombre de groupes d'axes (CN et automate) composant le système est défini par le paramètre machine P97 (Voir manuel des paramètres).

Les fonctions de programmation pièce assurées par les groupes automates sont identiques à celles assurées par les groupes d'axes CN sauf pour les fonctions M01, M12 et G75 (voir manuel de programmation pièce).

Les principaux modes de fonctionnement des groupes automates sont les modes «Continu» et «Séquentiel» (Voir manuel opérateur). Un groupe automate fonctionne dans un de ces modes indépendamment du mode de fonctionnement des autres groupes (CN ou automate).

Fonctionnement en mode "JOG" :
- lorsqu'un groupe automate est validé par %Wg00.2 puis RAZ (impulsion %Wg01.0) ce groupe peut être piloté en mode JOG. Il utilise alors son propre potentiomètre %Wg02.b pour moduler les avances.

17.2 Principe de programmation

Les commandes et compte-rendus échangés entre la fonction automatisme et la fonction CN transitent par la zone d'échange. Ces informations doivent être traitées dans le programme utilisateur.

17.2.1 Informations échangées de la fonction automatisme vers la fonction CN

Les informations concernant les groupes d'axes automates transmises par la fonction automatisme vers la fonction CN sont :

Fonction	Variable	Mnémonique
Demande de dégagement d'urgence	%Wg01.4	C_DGURG1 à C_DGURG8
Demande de «Départ cycle»	%Wg01.2	C_CYCLE1 à C_CYCLE8
Demande de RAZ sur le groupe	%Wg01.0	C_RAZ1 à C_RAZ8
Choix du mode (continu ou séquentiel)	%Wg00.7	C-MODE1 à C_MODE8
Commande maintenue de vitesse rapide sur groupe	%Wg00.6	C_FAST1 à C_FAST8
Compte rendu des fonctions M sur le groupe	%Wg00.5	CRM1 à CRM8
Appel de sous programme par le processeur machine	%Wg00.4	APPSS1 à APPSS8
Signal d'arrêt sur butée	%Wg00.3	ARBUT1 à ARBUT8
Validation du groupe	%Wg00.2	VA1LID1 à VALID8
Signal de fin de mouvement extérieur	%Wg00.1	C_FMEXT1 à C_FMEXT8
Autorisation des avances sur le groupe	%Wg00.0	C_AUTAV1 à C_AUTAV8

17.2.2 Informations échangées de la fonction CN vers la fonction automatisme

Les informations concernant les groupes d'axes automates transmises par la fonction CN vers la fonction automatisme sont :

Fonction	Variable	Mnémonique
Groupe en défaut	%Rg01.6	E_DEF1 à E_DEF8
Axe en attente de position	%Rg01.5	NO_POS1 à NO_POS8
Dégagement d'urgence en cours	%Rg01.4	E_DGURG1 à E_DGURG8
Cycle en cours	%Rg01.2	E_CYCL1 à E_CYCL8
RAZ en cours	%Rg01.0	E_RAZ1 à E_RAZ8

18 Mise au point des programmes

18.1 Programmation et mise au point avec PLCTOOL

Se reporter à la documentation «PLCTOOL - Outil de programmation langage ladder».

18.2 Mise au point sur la CN

L'utilitaire 7 résident en mémoire, permet la gestion de l'application automate et la liaison avec l'atelier logiciel PLCTOOL pour le chargement / déchargement des fichiers.

18.2.1 Procédure d'Accès à l'utilitaire

Conditions requises

L'utilitaire 7 ne nécessite pas de condition particulière d'accès.

Actions

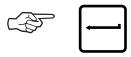
Sélectionner le menu des utilitaires	

Le menu «UTILITAIRES CN» est affiché à l'écran.

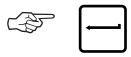
Sélectionner le menu «PROGRAMMES UTILITAIRES PRESENTS»	

Un menu listant les utilitaires présent en mémoire CN est affiché à l'écran.

Choisir éventuellement la langue dans laquelle les programmes utilitaires seront édités.

Frapper «A» pour Anglais.	

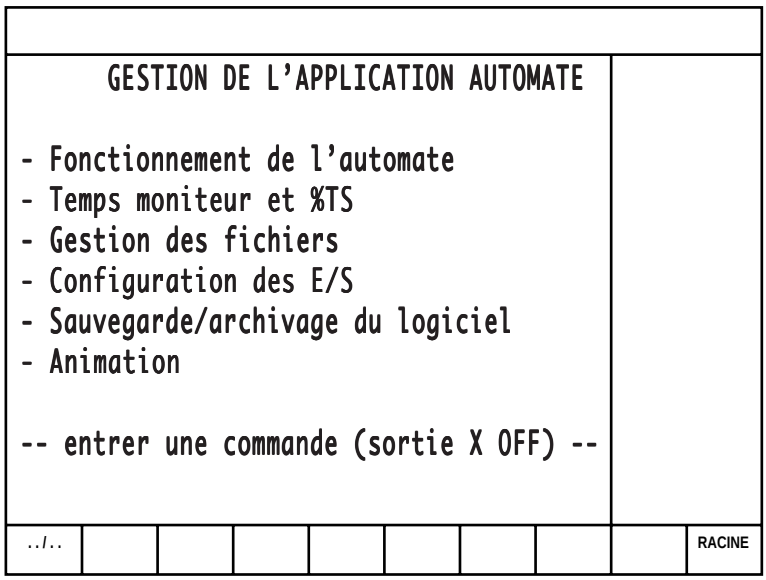
Ou

Frapper «F» pour Français.	

Le menu est édité dans la langue choisie.

Sélectionner l'utilitaire 7.	

Affichage du menu principal «GESTION DE L'APPLICATION AUTOMATE».



Abandon de la procédure

Frapper au clavier la commande.  

Retour à la page «AXES».

18.2.2 Contrôle du fonctionnement de l'unité centrale

Cette fonction permet de visualiser des informations concernant le fonctionnement de l'unité centrale.

Conditions requises

Menu «GESTION DE L'APPLICATION AUTOMATE» à l'écran.

Actions

Frapper au clavier «F» pour «Fonctionnement de l'automate».

Visualisation du menu «FONCTIONNEMENT DE L'AUTOMATE».

```
        FONCTIONNEMENT DE L'AUTOMATE
Etat automate : MARCHE
(Pas de défaut)
Activite TF :
( 0-7 )   0   0   0   0   0   0   0   0
( 8-15 )  0   0   0   0   0   0   0   0
Activite TH :
( 0-7 )   0   0   0   0   0   0   0   0
( 8-15 )  0   0   0   0   0   0   0   0
Commandes : Depart / Stop / Init
Raz variables sauvegardees
```

../..									RACINE

Abandon de la procédure

Frapper sur la touche «F11».

Retour au menu «GESTION DE L'APPLICATION AUTOMATE».

18.2.2.1 Etat automate

L'»Etat automate» renseigne en permanence sur l'état de fonctionnement de l'unité centrale.

Les messages visualisés dans cette zone sont spécifié dans le tableau ci après.

Etat automate «MARCHE»

Messages	Commentaire
Pas de défaut	Fonction automatisme en fonctionnement correct
Défaut configuration bus E/S	**Cause :** Présence d'une carte non reconnue. Absence d'une carte sur le bus. Divergence entre configuration programmée et configuration réelle. Programmation du chien de garde incorrecte. **Action corrective** Vérifier la configuration des entrées/sorties (Voir 18.2.5). Controler l'état de %R97F.B (Voir 3.8.5). Vérifier le %INI dans le programme client (Variables de configuration carte et chien de garde).
Fonctionnement bus E/S incorrect	**Cause :** Défaut de liaison sur le bus. **Action corrective** Controler l'état du status bus %Rrc39.B (Voir 3.7.3). Vérifier la continuité de l'anneau fibre optique. Contacter le SAV NUM.

Etat automate «ARRET»

Messages	Commentaire
Défaut interne moniteur	**Cause :** Défaut interne grave. **Action corrective** Contacter le SAV NUM.
Défaut dépassement temps calcul	**Cause :** Dépassement de la HTR (boucle dans programme) **Action corrective** Reprendre le programme client.
Défaut automate non répertorié	**Cause :** Défaut interne grave. **Action corrective** Contacter le SAV NUM.

Messages	Commentaire
Défaut configuration bus E/S	**Cause :** Présence d'une carte non reconnue. Absence d'une carte sur le bus. Divergence entre configuration programmée et configuration réelle. Programmation du chien de garde incorrecte. **Action corrective** Vérifier la configuration des entrées/sorties (Voir 18.2.5). Controler l'état de %R97F.B (Voir 3.8.5). Vérifier le %INI dans le programme client (Variables de configuration carte et chien de garde).
Fonctionnement bus E/S incorrect	**Cause :** Défaut de liaison sur le bus. **Action corrective** Controler l'état du status bus %Rrc39.B (Voir 3.7.3). Vérifier la continuité de l'anneau fibre optique. Contacter le SAV NUM.
Excès de carte sur bus E/S	**Cause :** Nombre d'entrées/sorties supérieure aux limites du système. **Action corrective** Réduire le nombre de carte entrées/sorties dans les limites autorisées.

A l'installation du code client

Messages	Commentaire
Lecture application impossible	**Cause :** Défaut interne grave. **Action corrective** Contacter le SAV NUM.
Erreurs dans module TSi / TFi / THi / SPi / code C / ???	**Cause :** Problèmes au chargement (module trop grand, pas assez de place en mémoire locale pour charger le module). **Action corrective** Réduire la taille du module. Augmenter la taille de la mémoire locale. Si «???», contacter le SAV NUM.
Module C : double exportation de symboles	**Cause :** Double exportation de symboles dans un module C. **Action corrective** Vérifier et modifier le module C.
Double définition du module TSi / TFi / THi / SPi / code C / ???	**Cause :** Un module de même nom est présent deux fois dans l'application. **Action corrective** L'application client doit être composée de module de nom différent. Mettre en conformité l'application client. Si «???», contacter le SAV NUM.

A l'exécution du code client

Un message sur trois lignes est visualisé.

Ligne 1 : Libellé du message.

Ligne 2 : Donne la tâche mise en cause «TSi / THi / TFi / INI»

Ligne 3 : Donne le module mis en cause «Module : *.*[@ relative par rapport au début du module]» ou »Module : ??? [@ relative par rapport au début du mapping moniteur AP]».

REMARQUE : Les adresses relatives par rapport au début du module ne sont exploitables que pour les modules C (.XCX). Le fichier *.MAP de l'application C donne ces adresses.*

Messages	Commentaire
Code client : Débordement sur division	**Cause :** Débordement signalé sur une division **Action corrective** Vérifier et modifier le module mis en cause. Si «???», contacter le SAV NUM.
Code client : @ relative au moniteur = Adresse interdite	**Cause :** Opération sur une adresse interdite. **Action corrective** Vérifier et modifier le module mis en cause. Si «???», contacter le SAV NUM.
Code client incohérent	**Cause :** Utilisation de fonctions ou de symboles incohérent dans le programme client. Programme client incohérent **Action corrective** Vérifier et modifier le module mis en cause. Si «???», contacter le SAV NUM.

18.2.2.2 Activité des tâches de fond

L'activité des tâches de fond est visualisée par 16 compteurs, associés aux tâches %TF0 à %TF15.

A chaque traitement total ou partiel d'une tâche de fond, pendant un cycle HTR, le compteur de la tâche est incrémenté de un. Cette fonction permet de visualiser les tâches en sommeil, en cours d'exécution, le nombre de cycle HTR nécessaire à l'exécution d'une tâche, .. etc ...

18.2.2.3 Activité des tâches hard

L'activité des tâches hard est visualisée par 16 compteurs, associés aux tâches %TH0 à %TH15. A chaque traitement d'une tâche hard, son compteur est incrémenté de un.

18.2.2.4 Commande de l'unité centrale

Les commandes «DEPART», «STOP» et «INIT» permettent d'intervenir sur le fonctionnement de l'unité centrale pendant la mise au point du programme utilisateur.

Conditions requises

Menu «FONCTIONNEMENT DE L'AUTOMATE» à l'écran.

Actions

Frapper au clavier la commande choisie. (Voir tableau ci-après) ☞ ⟵

Opération à réaliser	Commande	Remarque
Mise en marche de l'unité centrale	Frapper «D» pour Départ	Armement du chien de garde. Déroulement du programme utilisateur.
Arrêt de l'unité centrale	Frapper «S» pour Stop	Retombée du chien de garde. Arrêt du programme utilisateur. L'état «ARRET» est spécifié dans la page d'écran.
Initialisation du système	Frapper «I» pour Init	Nécessite un arrêt de l'unité centrale. Réalise : - l'effacement de tous les défauts, - l'initialisation des Entrées/Sorties.

Au déassemblage du code client (contrôle des appels fonctions)

Messages	Commentaire
Nom de module inconnu	**Cause :** Un pointeur mal initialisé dans un modole C détruit une zone de code **Action corrective :** Localiser le module C et apporter les modifications nécessaires.
Erreur dans module INI, TSi, TFi, THi, Code C	**Cause :** Un pointeur mal initialisé dans un module C détruit une zone de code **Action corrective :** Localiser le module C et apporter les modifications nécessaires.

A l'activation du contrôle des appels fonctions (PLCTOOL)

Messages	Commentaire
Accès à une zone interdite	**Cause :** Le paramètre d'adresse d'une fonction ladder ou C pointe dans une zone autre qu'une zone de données **Action corrective :** Editer le module mis en cause et modifier la fonction en défaut
Plus de 512 zones autorisées	**Cause :** L'application chargée contient plus de 512 zones de données non contiguë **Action corrective :** Regrouper les composants chaînes et constantes les uns à la suite des autres.

18.2.2.5 Raz des variables sauvegardées

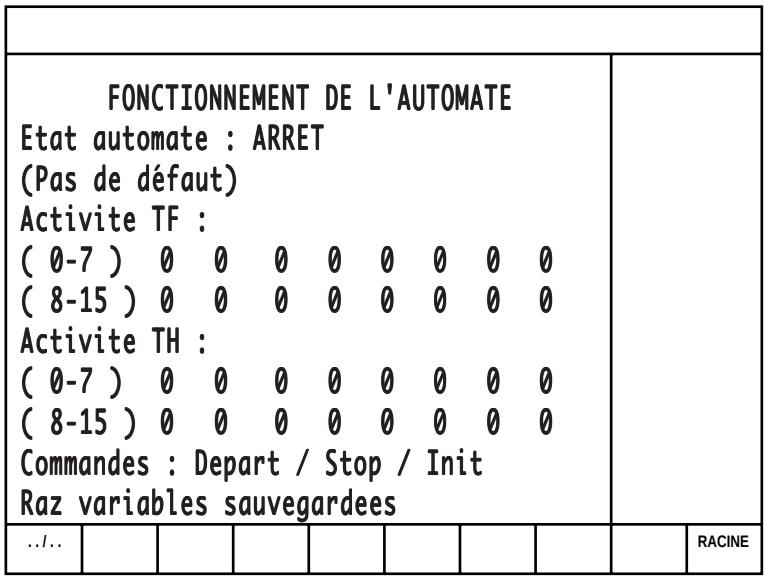
Cette fonctionnalité permet une remise à zéro des variables sauvegardées (%M).

Conditions requises

Menu «FONCTIONNEMENT DE L'AUTOMATE» affiché à l'écran.

Actions

Frapper au clavier la commande «S».



Frapper au clavier la commande «R».

Le message «Etes-vous sur ? (O/N)» est affiché à l'écran.

Valider la Raz par la commande «O».

ou

Annuler la Raz par la commande «N».

Relancer l'automate per la commande «D».

Abandon de la procédure

Frapper sur la touche «F11».

Retour au menu «GESTION DE L'APPLICATION AUTOMATE».

18.2.3 Temps moniteur et tâches %TS

Cette rubrique permet de visualiser le pourcentage du temps occupé par le moniteur et les tâches %TS à chaque cycle automate.

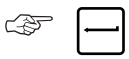
On y observe :
- le temps moyen occupé par le moniteur à chaque cycle,
- le temps maximum occupé par le moniteur ,
- le temps moyen occupé par chaque tâche %TS,
- le temps maximum occupé par chaque tâche %TS,
- les dépassements de temps de calcul à chaque cycle.

Conditions requises

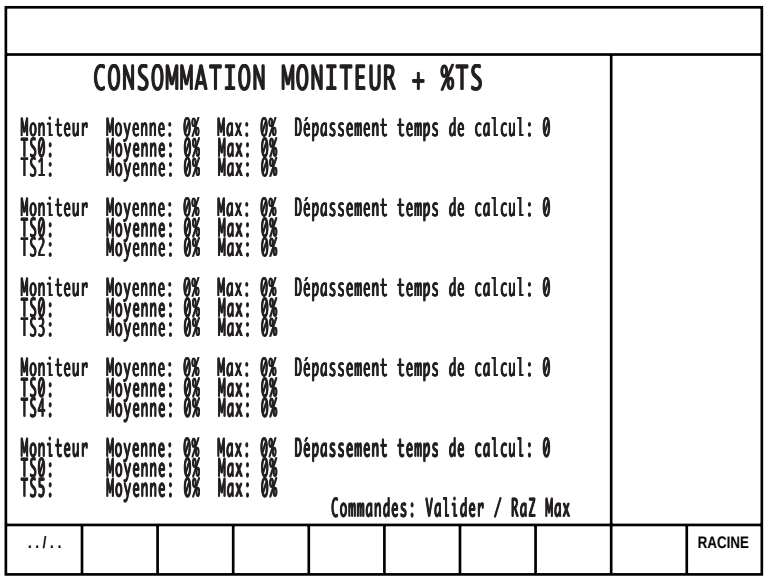
Menu «GESTION DE L'APPLICATION AUTOMATE» à l'écran.

Actions

Frapper au clavier «T» pour «Temps moniteur et %TS».



Visualisation du menu «CONSOMMATION MONITEUR + %TS».



Le pourcentage est calculé par rapport au temps alloué au moniteur et au programme client, soit :
- 18 ms en série I et série II biprocesseur, bien que la HTR soit de 20 ms car 2 ms sont réservées par le système (Voir 2.1)
- Valeur de P99 en ms en série II UCSII.

Acquisition des mesures

Frapper au clavier «V» pour valider l'acquisition des mesures.

Les valeurs de consommations du moniteur et des tâches %TS sont mises à jour.

Arrêter les mesures

Frapper au clavier «I» pour inhiber l'acquisition des mesures.

Les valeurs de consommations du moniteur et des tâches %TS sont remises à zéro.

RAZ des maxima

Frapper au clavier «R» pour une remise à zéro des maxima

Les consommations maximales du moniteur et des tâches %TS sont remises à zéro.

Abandon de la procédure

Frapper sur la touche «F11».  RACINE

Retour au menu «GESTION DE L'APPLICATION AUTOMATE».

18.2.4 Gestion des fichiers

Permet la gestion des informations concernant les fichiers chargés dans la fonction automatisme.

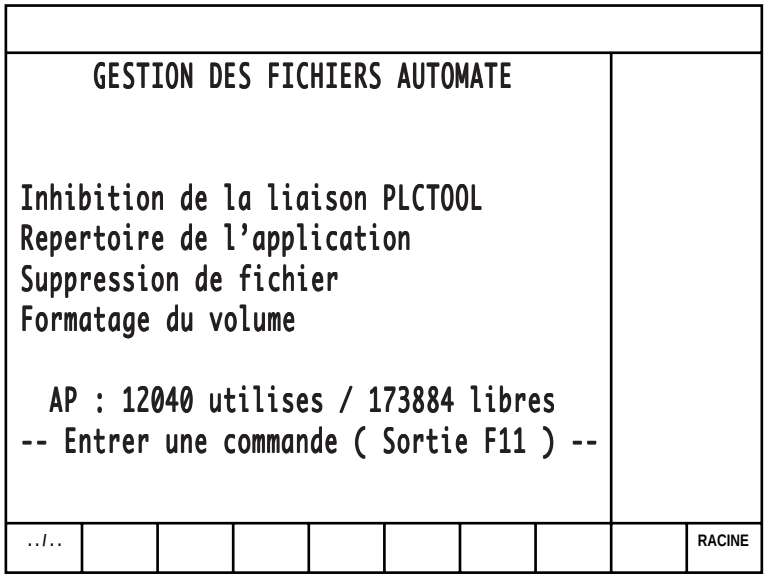
Conditions requises

Menu «GESTION DE L'APPLICATION AUTOMATE» à l'écran.

Actions

Frapper au clavier «G» pour «Gestion des fichiers».

Visualisation du menu «GESTION DES FICHIERS AUTOMATE».



L'information «AP» renseigne sur les tailles mémoires utilisées et libres (En octets).

Abandon de la procédure

Frapper sur la touche «F11».

Retour au menu «GESTION DE L'APPLICATION AUTOMATE».

18.2.4.1 Formatage du volume

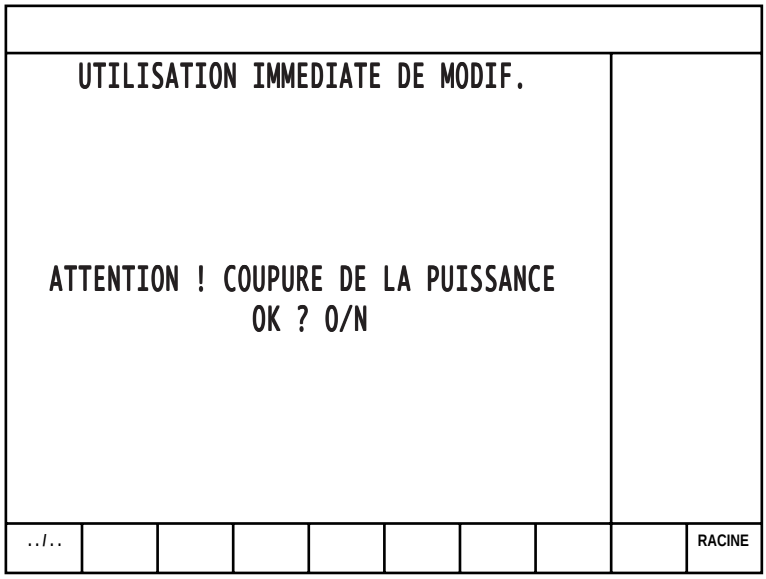
Cette commande permet d'initialiser la mémoire automate et de supprimer tous les fichiers en mémoire.

Actions

Frapper au clavier «F» pour «Formatage du volume». ☞ ⟵

Le message «Redemarrage système applic detruite, confirmer : (O)» est affiché.

Confirmer la Suppression de tous les fichiers en mémoire ☞ O ⟵



Relancer le système en acquittant les messages successifs.

Abandon de la procédure

Frapper sur la touche «F11». ☞ RACINE

Retour au menu «GESTION DE L'APPLICATION AUTOMATE».

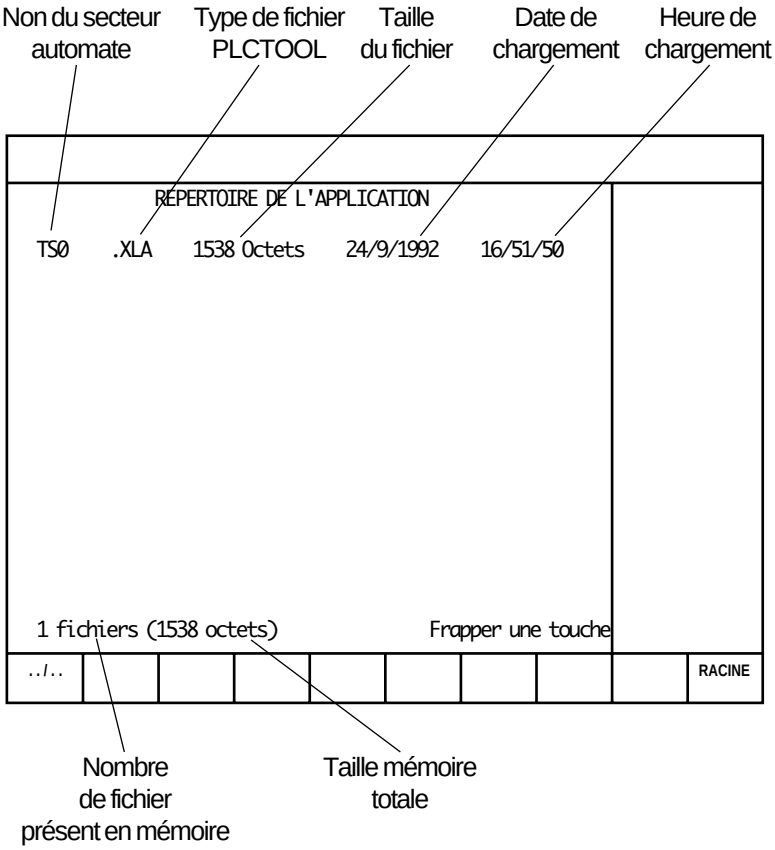
18.2.4.2 Répertoire de l’application

Cette page d’information permet de visualiser tous les fichiers chargés dans les secteurs de l'unité centrale.

Actions

Frapper au clavier «R» pour «Répertoire de l’application».

Visualisation du menu «REPERTOIRE DE L’APPLICATION».



Abandon de la procédure

Frapper une touche quelconque.

Retour au menu «GESTION DES FICHIERS AUTOMATE».

ou

Frapper sur la touche «F11».

Retour au menu «GESTION DE L’APPLICATION AUTOMATE».

18.2.4.3 Suppression de fichier

Permet la suppression de fichiers présent en mémoire automate.

Actions

Frapper au clavier «S» pour «Suppression de fichier». ☞ ⟵

Le message «Nom de fichier ?» est affiché.

Frapper le nom du fichier à supprimer [Nom du secteur].[Type de fichier]. (Exemple : TS0.XLA) ☞ ⟵

Le message «fichier supprimé» est affiché.

Abandon de la procédure

Frapper sur la touche «F11». ☞ RACINE

Retour au menu «GESTION DES FICHIERS AUTOMATE».

18.2.4.4 Validation - invalidation de la liaison PLCTOOL

Cette fonction permet de valider ou d'invalider la liaison série avec le logiciel PLCTOOL pour le chargement/ déchargement de fichier et pour la fonction DEBUG «ON LINE».

Validation de la ligne

La ligne de menu «Validation de la liaison PLCTOOL» est affichée.

Frapper au clavier «V» pour valider la ligne. ☞ ⟵

La ligne de menu devient «Invalidation de la liaison PLCTOOL».

Invalidation de la ligne

La ligne de menu «Invalidation de la liaison PLCTOOL» est affichée.

Frapper au clavier «I» pour invalider la ligne. ☞ ⟵

La ligne de menu devient «Validation de la liaison PLCTOOL».

18.2.5 Configuration des Entrées/Sorties

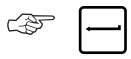
Le menu de configuration des Entrées/Sorties permet de visualiser :
- le type de rack présent dans le système,
- le type de carte présente dans chaque rack.

Conditions requises

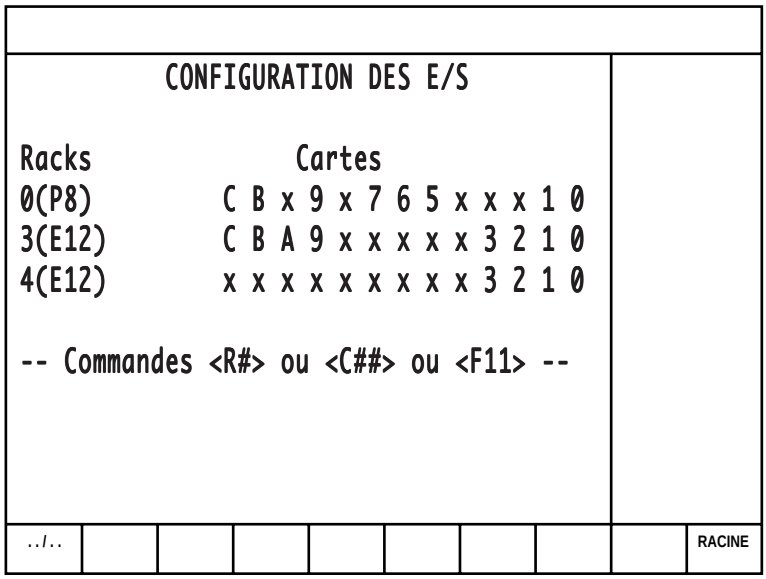
Menu «GESTION DE L'APPLICATION AUTOMATE» à l'écran.

Actions

Frapper au clavier «C» pour «Configuration des E/S».



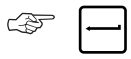
Visualisation du menu «CONFIGURATION DES E/S».



REMARQUE : Le caractère «x» indique l'absence de carte.

Identification du rack

Frapper au clavier la commande «R [N° du rack]» (Numéro de 0 à 7).

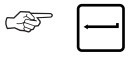


visualisation, en ligne de dialogue, des messages :

N°du rack	Identification	Messages
0	P8 (Principal 8 emplacements) P4 (Principal 4 emplacements)	«R0 : Rack principal 8 emplacements» «R0 : Rack principal 4 emplacements» Si le rack n'existe pas : «Rack absent !»
De 1 à 6	E12 (Extension 12 emplacements) M2 (Module 2 emplacements)	«Rx : Rack d'extension 12 emplacements» «Mx : Rack module 2 emplacements» Si le rack n'existe pas : «Rack absent !» (avec «x» N° du rack)

Identification des cartes

Frapper au clavier la commande «C [N° du rack] [N° de la carte]».



visualisation en ligne de dialogue, des messages :

Type de rack	N° de carte	Messages
P8, P4 et E12	0	Alim. 130 W avec fibre optique Alim. 130 W sans fibre optique Alim. 60 W avec fibre optique Alim. 60 W sans fibre optique
P8 et P4	De 1 à 4	Carte pupitre de base Carte pupitre extension
P8 P4 E12 M2	De 5 à 0xC De 5 à 8 De 1 à 0xC 1 et 2	Cartes 32 sorties à relais Carte 32 entrées continue Carte 32 entrées 24 sorties Carte 64 entrées - 48 sorties Carte 32 entrées - 24 sorties#
P8 P4 E12 M2	De 1 à 0xC De 1 à 8 De 1 à 0xC 1 et 2	Carte absente !

Abandon de la procédure

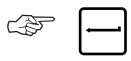
Frapper sur la touche «F11».

Retour au menu «GESTION DE L'APPLICATION AUTOMATE».

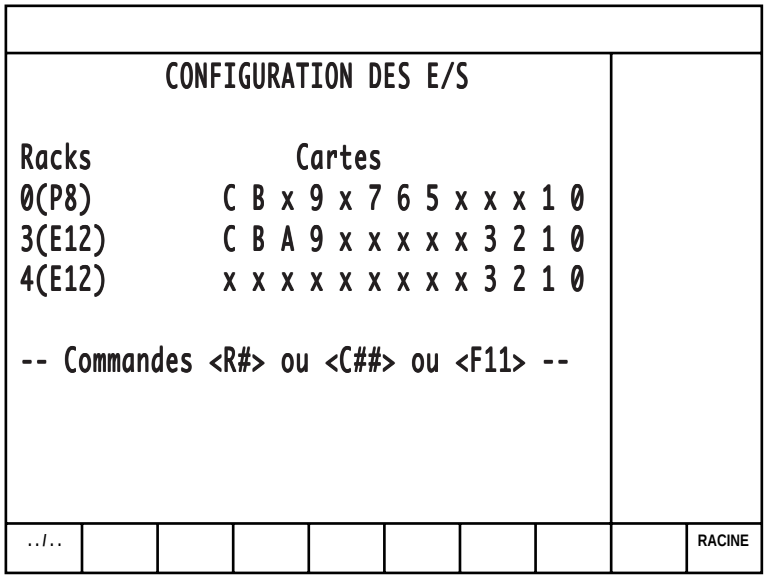
Exemple

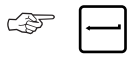
Identification du rack 0 et des cartes adressées 5, 8 et B qui l'équipent

Menu «GESTION DE L'APPLICATION AUTOMATE» à l'écran.

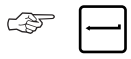
Frapper au clavier «C» pour «Configuration des E/S». 

Visualisation du menu «CONFIGURATION DES E/S».

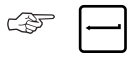


Frapper au clavier la commande «R 0». 

Visualisation, en ligne de dialogue, du message «R0 : Rack principal 8 emplacements».

Frapper au clavier la commande «C05». 

Visualisation en ligne de dialogue, du message «Cartes 32 sorties à relais».

Frapper au clavier la commande «C08». 

Visualisation en ligne de dialogue, du message «Carte absente !».

Frapper au clavier la commande «C0B». 

Visualisation en ligne de dialogue, du message «Carte 32 entrées continue».

18.2.6 Sauvegarde et Archivage du logiciel

Ce module permet via une ligne série de l'unité centrale :
- d'archiver le programme utilisateur sur un périphérique (lecteur de disquettes ou un lecteur/perforateur de bandes),
- de vérifier le programme archivé par rapport à la source,
- de restituer le programme archivé.

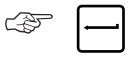
Conditions requises

Vitesse de transmission conforme et paramètres de communications correct sur le périphérique.

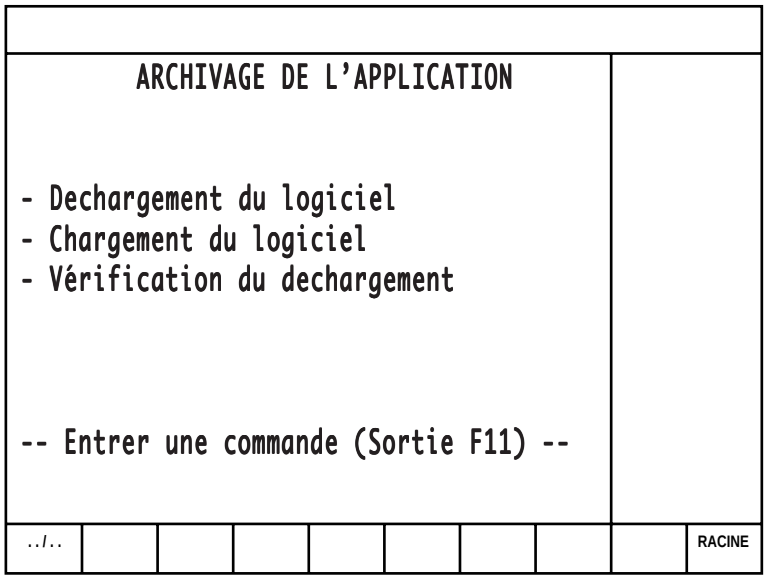
CN connectée au périphérique (sur une ligne série de l'unité centrale)

Menu «GESTION DE L'APPLICATION AUTOMATE» à l'écran.

Actions

Frapper au clavier «A» pour «Archivage du logiciel». 

Visualisation du menu «ARCHIVAGE DE L'APPLICATION».



Abandon de la procédure

Frapper sur la touche «F11». 

Retour au menu «GESTION DE L'APPLICATION AUTOMATE».

18.2.6.1 Déchargement du logiciel

Actions

Mettre le périphérique en mode déchargement.
Frapper au clavier «D» pour «Déchargement du logiciel». 

Le message «Déchargement en cours» est affiché.

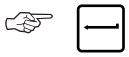
Disparition du message en fin de déchargement.

Abandon de la procédure

Frapper sur la touche «F11». 

18.2.6.2 Vérification du déchargement

Actions

Frapper au clavier «V» pour «Vérification du déchargement». 

Le message «Attente vérification» est affiché.

Mettre le périphérique en mode chargement.

Le message «Chargement en cours» est affiché.

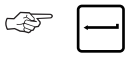
Disparition du message en fin de chargement.

Abandon de la procédure

Frapper sur la touche «F11». 

18.2.6.3 Chargement du logiciel

Actions

Frapper au clavier «C» pour «Chargement du logiciel». 

Le message «Attente chargement» est affiché.

Mettre le périphérique dans le mode approprié à la vérification de la sauvegarde.

Le message «Vérification en cours» est affiché.

Disparition du message en fin de vérification.

Abandon de la procédure

Frapper sur la touche «F11». 

18.2.7 Chargement/déchargement de fichier vers PLCTOOL

Pour plus de précision, se reporter au manuel «PLCTOOL - Outil de programmation langage ladder».

Vérifier l'état du paramètre P112 (Voir «Manuel des paramètres»).

Vérifier que la liaison PLCTOOL est validée (Voir 18.2.4.4).

Harmoniser les vitesses de transmission entre la CN et le micro-ordinateur.

Installer un câble de liaison entre la ligne série paramétrée et le micro-ordinateur.

Lancer la procédure de chargement ou de déchargement à partir du logiciel PLCTOOL.

REMARQUE : La procédure de chargement actuelle ne nécessite aucune intervention du coté de la CN.

18.2.8 Animation ladder

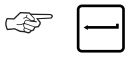
L'animation ladder permet la visualisation dynamique des contacts ladder d'un programme automate. On a ainsi une visualisation graphique de l'évolution d'un réseau de contact.

Conditions requises

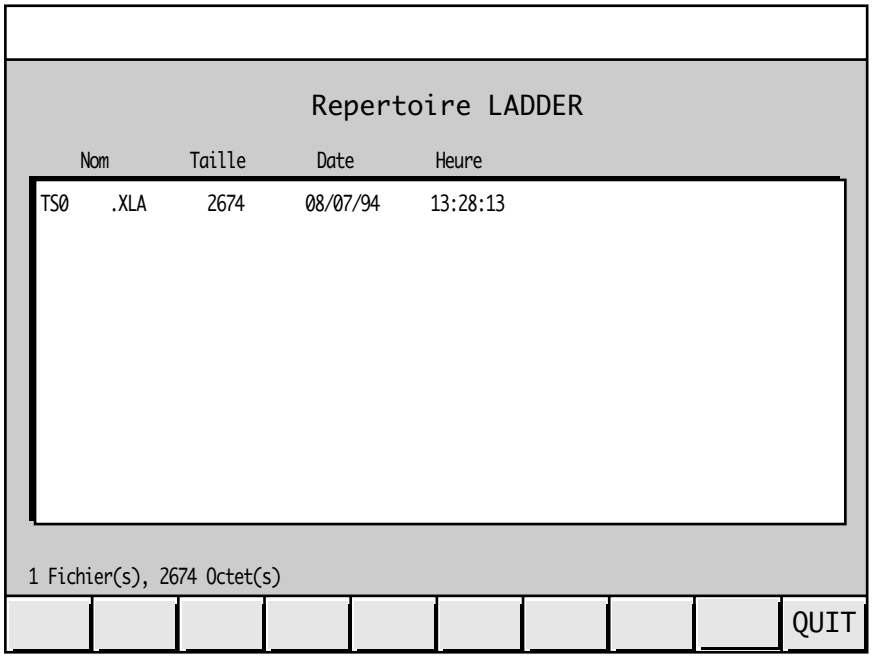
Menu «GESTION DE L'APPLICATION AUTOMATE» à l'écran.

Actions

Frapper au clavier «A» pour «Animation».

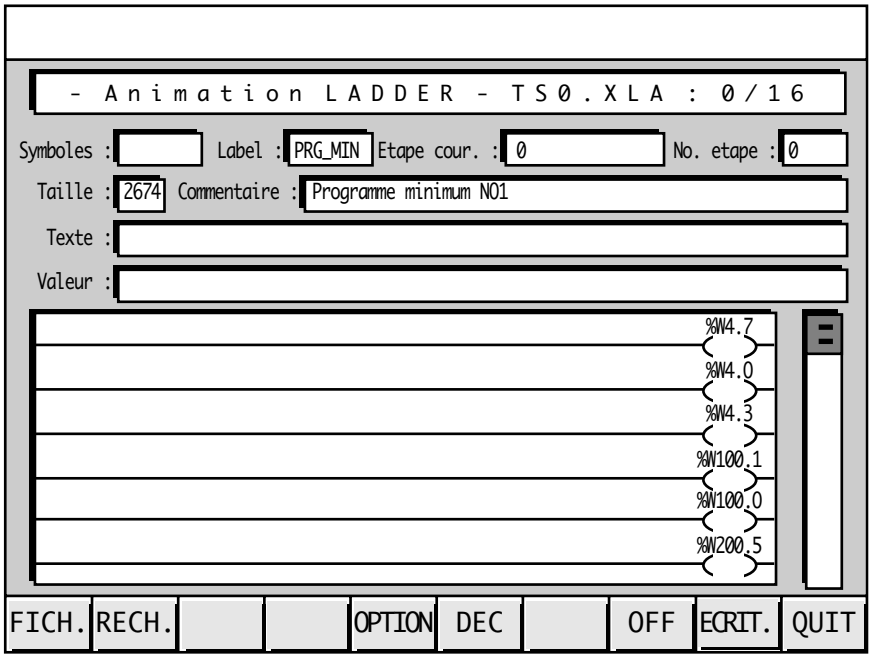


Visualisation du «Repertoire LADDER».



Sélectionner le module à animer à l'aide des touches de direction puis valider.

Visualisation de la grille ladder à animer.



A l'ouverture, le composant visualisé est animé.

En animation, le champ «Var.etap» devient «Etape cour» et permet de visualiser la valeur courante de la variable d'étape.

Si cette valeur est égale au N° d'étape (composant passant) ou si aucune variable d'étape n'à été définie, le fond du champ prend la couleur active.

Interprétation des couleurs

Etat	Moniteur couleur	Moniteur monochrome
Actif	Rouge	Blanc
Inactif	Noir	Noir
Indéterminé	Clignotant	Clignotant

Naviguer dans l'application

Opération à réaliser	Commandes
Déplacer le focus sur l'objet suivant	⇥ ou SHIFT ⇥ ou SHIFT ← ou SHIFT →
Se déplacer dans la grille ladder	Focus sur la grille ladder puis ↑ ou ↓ ou ← ou →
Visualiser le composant ladder suivant	Focus sur l'ascenseur puis Pg Dn
Visualiser le composant ladder précédent	Focus sur l'ascenseur puis Pg Up
Visualiser le dernier composant ladder	Focus sur l'ascenseur puis End
Visualiser le premier composant ladder	Focus sur l'ascenseur puis Home

Arrêt de l'animation

Frapper sur la touche «F9». ☞ OFF

L'animation ladder est arrêté. Le cartouche permet l'accès à une nouvelle touche «EFFACER».

Initialiser la grille ladder

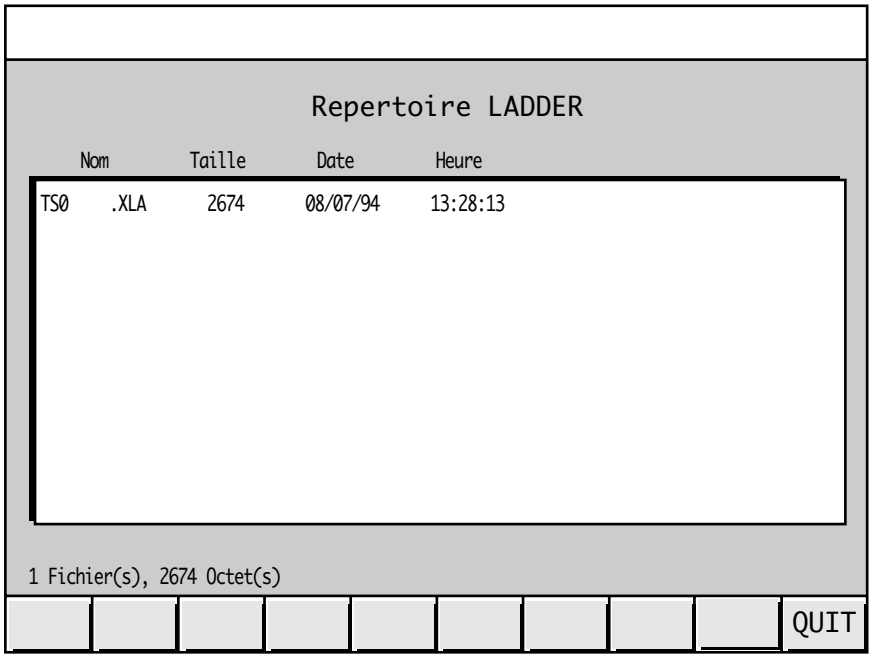
Frapper sur la touche «F8». ☞ EFFACER

Tous les éléments de la grille ladder sont forcés à l'état inactif.

Charger un nouveau module ladder

Frapper sur la touche «F2».

Visualisation du «Repertoire LADDER».



Sélectionner le module à animer à l'aide des touches de direction puis valider.

REMARQUE : La touche «QUIT» permet un retour au module ladder précédent

Visualisation de la grille ladder à animer.

Animer le module

Frapper sur la touche «F9».

Le module ladder est animé.

Définir les options

Affichage décimal/hexadécimal

Frapper sur la touche «F7».

Ou

Frapper sur la touche «F7».

Visualisation des valeurs numérique en décimal ou en hexadécimal.

Quadrillage des cellules

Frapper sur la touche «F6».

OPTION

Visualisation d'une nouvelle barre de menu

Frapper sur la touche «F2».

QUADRI. ON

Ou

Frapper sur la touche «F2».

QUADRI. OFF

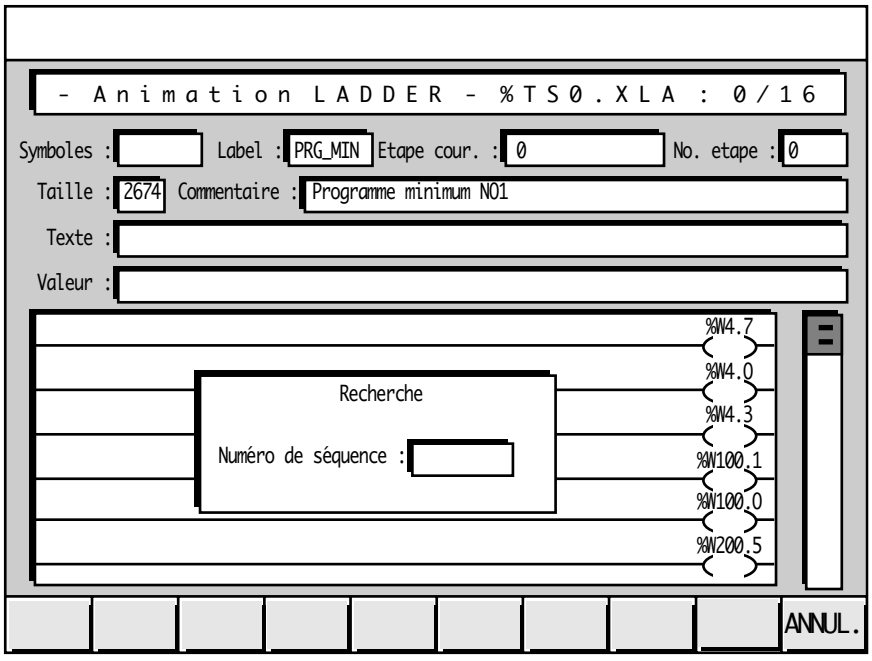
Validation ou invalidation d'un quadrillage au dimension des cellules

Rechercher un composant

Frapper sur la touche «F3».

RECH.

Ouverture de la boîte de dialogue «Recherche»



Saisir le numéro du composant à atteindre

Le composant spécifié est visualisé.

Annuler la procédure

Frapper sur la touche «F11».

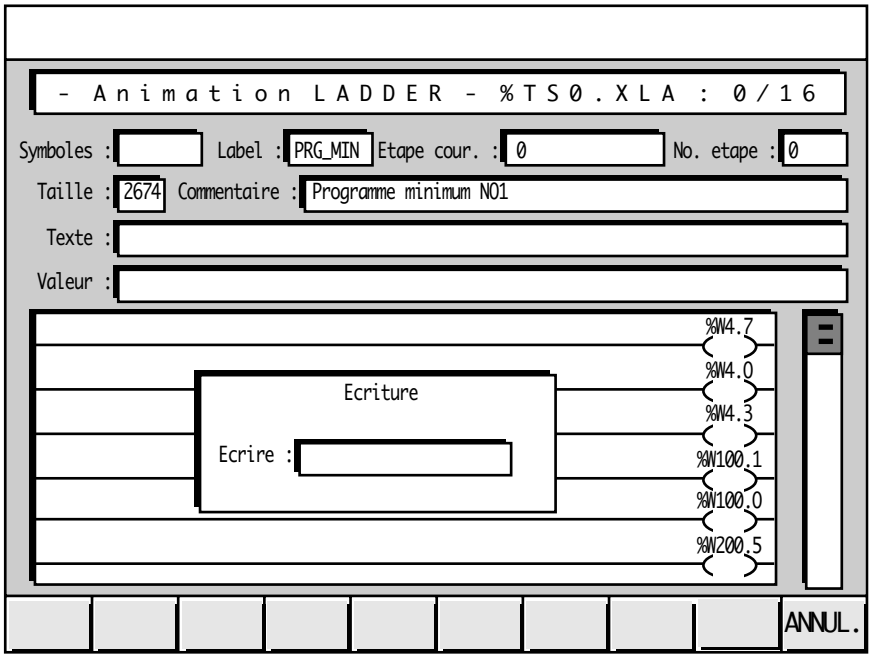
ANNUL.

Ecrire une variable

Frapper sur la touche «F10».

ECRIT.

Ouverture de la boîte de dialogue «Ecriture» ?



Saisir le nom de la variable et sa valeur.

REMARQUE : Toutes les variables sont accessibles en écriture sauf les variables d'entrées %Ixx.x.

Annuler la procédure

Frapper sur la touche «F11».

ANNUL.

Abandon de la procédure

Frapper sur la touche «F11».

QUIT

Retour au menu «GESTION DE L'APPLICATION AUTOMATE».

19 Défauts et diagnostic

19.1 Liste des défauts matériel

ERR_BUS_SBCE : Dysfonctionnement du bus série.

19.2 Liste des défauts de configuration

ERR_CONFIG_SBCE : Carte E/S mal placée ou erreur de configuration.

19.3 Liste des défauts de programmation

ERR_HTR : Dépassement de la période HTR.

ERR_ACCESS_VIOLATION : Tentative de lecture ou d'écriture dans une zone interdite.

A Listes des fonctions

A.1 Liste par thèmes

A.1.1 Fonctions d'usages générales

Fonction	Désignation	Page
W_E42000	Ecriture de n variables E42000	6-32
y_init()	Initialisation de la base associée aux variables %Y	6-33

A.1.2 Gestion des tâches

Fonction	Désignation	Page
csbegin()	Début d'une section critique	7-3
csend()	Fin d'une section critique	7-3
tfstart()	Départ d'une tâche %TF	7-4
tfstop()	Arrêt d'une tâche %TF	7-4
whtr()	Mise en sommeil temporaire d'une tâche %TF	7-3

A.1.3 Mode transparent

Fonction	Désignation	Page
inig()	Init graphique	8-17
pcur()	Positionnement du curseur	8-7
print()	Affichage d'un tampon	8-8
printf()	Affichage d'une chaîne avec formatage	8-9
putchar()	Affichage d'un caractère	8-7
putimage()	Positionnement et affichage d'une image	8-16
puts()	Affichage d'une chaîne sans formatage	8-8
scanc()	Fermeture d'une acquisition clavier	8-16
scand()	Acquisition et conversion d'un nombre décimal	8-14
scano()	Ouverture d'une acquisition clavier	8-12
scans()	Acquisition d'une chaîne	8-13
scanu()	Ouverture d'une acquisition clavier numérique	8-13
scanx()	Acquisition et conversion d'un nombre hexadécimal	8-15

A.1.4 Gestion des Entrées/Sorties analogiques

Fonction	Désignation	Page
anaa()	Redirection d'une carte analogique	9-7
anai()	Lecture d'une entrée analogique	9-6
anao()	Ecriture d'une sortie analogique	9-5
anas()	Configuration d'une carte E/S analogiques	9-3

A.1.5 Lecture/écriture explicite d'une carte

Fonction	Désignation	Page
read_i()	Lecture explicite d'une carte entrée	10-3
write_q()	Ecriture explicite d'une carte sortie	10-4

A.1.6 Gestion des entrées interruptions

Fonction	Désignation	Page
iti_gr()	Association entrée interruption avec des groupes d'axes	11-5
itictl()	Configuration d'une entrée interruption	11-6
itiget()	Lecture d'une entrée interruption	11-8
thiti()	Association tâche %TH avec une entrée IT	11-9

A.1.7 Gestion des lignes séries

Fonction	Désignation	Page
comctl()	Contrôle du pilote de ligne série	12-11
comf()	Sélection des vitesses et formats	12-4
comin()	Lecture du tampon de réception	12-7
comout()	Emission d'un tampon	12-6
comreg()	Lecture de l'état d'une ligne série	12-10

A.1.8 Gestion des timers

Fonction	Désignation	Page
thtimer()	Association tâche %TH avec un timer	13-1

A.1.9 Gestion du dateur

Fonction	Désignation	Page
tmget()	Lecture de la date courante	14-1
dtget()	Lecture de la date courante avec jour de la semaine	14-2

A.1.10 Echanges par protocole

Fonction	Désignation	Page
neti()	Lecture d'une requête d'un serveur distant	15-36
neto()	Emission d'une requête vers un serveur distant	15-34
netst_ad	Réponse à la requête STATUS	15-40
uniti()	Lecture d'une réponse en interne	15-30
unito()	Emission d'une requête en interne	15-29
setcomw	Configuration du service mots communs	15-39

A.1.11 Appel de modules en langage C

Fonction	Désignation	Page
exec()	Appel d'un module exécutable	16-3
exechdl()	Identification d'un module exécutable	16-4

A.2 Liste par classement alphanumérique

Fonction	Désignation	Page
putimage()	Positionnement et affichage d'une image	8-16
putkey()	Simulation du clavier du pupitre	6-15
puts()	Affichage d'une chaîne sans formatage	8-8
qcktool()	Recherche circulaire optimale	6-15
rchb()	Recherche de la valeur d'un octet	6-16
rchl()	Recherche de la valeur d'un long mot	6-17
rchw()	Recherche de la valeur d'un mot	6-16
read_i()	Lecture explicite d'une carte entrée	10-3
return()	Retour au module ou au réseau appelant	6-18
R_E42000	Lecture de n variables E42000	6-31
scanc()	Fermeture d'une acquisition clavier	8-16
scand()	Acquisition et conversion d'un nombre décimal	8-16
scano()	Ouverture d'une acquisition clavier	8-12
scans()	Acquisition d'une chaîne	8-13
scanu()	Ouverture d'une acquisition clavier numérique	8-13
scanx()	Acquisition et conversion d'un nombre hexadécimal	8-15
sema()	Sémaphore	6-20
setb()	Ecriture d'un ou plusieurs octets	6-20
setcomw	Configuration du service mots communs	15-39
setl()	Ecriture d'un ou plusieurs long mots	6-22
setw()	Ecriture d'un ou plusieurs mots	6-21
sp()	Appel d'un module %SP	6-22
sprintf()	Formatage d'une chaîne de caractères	6-24
spy()	Appel d'un module %SP avec variables locales %Y	6-23
sqrt()	Racine carrée entière	6-25
sscanf()	Analyse d'une chaîne ASCII	6-25
strcmp()	Comparaison d'une chaîne de caractères	6-26
strcpy()	Copie d'une chaîne de caractères	6-27
strlen()	Calcul de la longueur d'une chaîne	6-27
swapl()	Echange les quatre octets d'un long mot	6-29
swapw()	Echange les octets pair et impair d'un mot	6-28
tfstart()	Départ d'une tâche %TF	7-4
tfstop()	Arrêt d'une tâche %TF	7-4
thiti()	Association tâche %TH avec une entrée IT	11-9
thtimer()	Association tâche %TH avec un timer	13-1
tmget()	Lecture de la date courante	14-1
tooldyn()	Correction dynamique d'un outil	6-30
uniti()	Lecture d'une réponse en interne	15-30
unito()	Emission d'une requête en interne	15-29
whtr()	Mise en sommeil temporaire d'une tâche %TF	7-3

Index

Symboles

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z