Étude d'un moniteur temps réel pour microprocesseur INTEL 8085 et utilisation dans une application de télétransmission

Michèle Matteï

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE AGREE
DE GRENOBLE (C.U.E.F.A)

MEMOIRE

présenté en vue d'obtenir

LE DIPLÔME D'INGÉNIEUR C.N.A.M.

en

INFORMATIQUE

par

Michèle MATTEÏ

Etude d'un moniteur temps réel pour microprocesseur INTEL 8085 et utilisation dans une application de télétransmission.

SOUTENU LE : 18 décembre 1984

JURY

Président : M. J.Y. RANCHIN

Membres : MM. L. BOLLIET
R. BOUTTAZ
J. GENIVET
P. BONNATERRE

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE AGREE
DE GRENOBLE (C.U.E.F.A)

MEMOIRE

présenté en vue d'obtenir

LE DIPLÔME D'INGÉNIEUR C.N.A.M.

en

INFORMATIQUE

par

Michèle MATTEÏ

Etude d'un moniteur temps réel pour microprocesseur INTEL 8085 et utilisation dans une application de télétransmission.

SOUTENU LE : 18 décembre 1984

JURY

Président : M. J.Y. RANCHIN

Membres : MM. L. BOLLIET
R. BOUTTAZ
J. GENIVET
P. BONNATERRE

Je tiens à remercier,

Monsieur J.Y. RANCHIN, Professeur au Conservatoire National des Arts et Métiers, qui m'a fait l'honneur d'accepter mon dossier de thèse et de présider ce jury.

Monsieur L. BOLLIET, Professeur à l'Université de Grenoble II, pour m'avoir incité et aidé à préparer cette thèse.

Monsieur R. BOUTTAZ, Ingénieur C.N.R.S. à l'Institut de Mathématiques Appliquées de Grenoble, d'avoir bien voulu participer au jury de cette thèse.

Monsieur J. GENIVET, Responsable du Service INDUSTRIE dans le Département d'Automatique de la Société MERLIN GERIN, qui m'a permis de réaliser ce travail.

Monsieur P. BONNATERRE, Responsable du Bureau d'Etudes de ce même Service, pour m'avoir conseillé et aidé tout au long de cette étude.

Toutes les personnes du Service INDUSTRIE qui m'ont prodigué leurs encouragements et leurs conseils.

Madame F. FAURE qui a su mettre en page, dactylographier et corriger le présent mémoire.

A mon mari, à mes parents, qui par leur compréhension, leur affection et leur patience, m'ont permis de mener à bien ce travail.

S O M M A I R E

INTRODUCTION

Qu'est-ce qu'un micro-ordinateur ?
En terme de définition, un micro-ordinateur est un ordinateur dont l'unité centrale est construite autour d'un microprocesseur réalisé en un seul boitier (tout au moins pour les systèmes 8 bits).

Cela signifie qu'il s'agit d'un ordinateur dont la constitution repose sur des composants LSI (Large Scale Integration) très denses.

Un micro-ordinateur, comme tout ordinateur, est une machine permettant d'effectuer des traitements (calculs ou manipulations de données) en exécutant un programme.

Pour entrer en dialogue avec le micro-ordinateur, l'utilisateur emploiera souvent un programme d'interface : Le Moniteur, dont la définition rigoureuse est AFNOR 01.04.07 : *"Programmerie destinée à commander l'exécution des programmes d'un calculateur et pouvant assurer l'enchainement des travaux, la mise au point des programmes, la commande des entrées et des sorties, la comptabilité d'exploitation, la compilation, l'attribution de la mémoire, la gestion des données et d'autres services"*.

Dans ses fonctions minimales, on retrouve alors le *moniteur de "base"* utilisé par la plupart des microprocesseurs. Cette notion de moniteur, appliquée à la micro-informatique, peut être généralisée en celle de *"Système d'exploitation"*.

Dans certaines applications, le micro-ordinateur doit disposer de la faculté de traiter les évènements dès qu'ils surviennent, selon leur ordre de priorité, et fournir des résultats en un temps négligeable par rapport à la durée du phénomène sur lequel il faut agir. On dit alors que le micro-ordinateur travaille en *temps réel*.

La gestion des programmes devra alors faire appel à un moniteur en temps réel RTOS (de "*Real Time Operating System*") connecté à une horloge "*en temps réel*" selon la terminologie propre à l'informatique. Le rôle du moniteur sera alors de gérer :

- les tâches à effectuer, en tenant compte des priorités
- le partage des ressources
- les entrées/sorties
- les interruptions
- le dialogue avec l'opérateur

En soi, l'étude d'un moniteur temps réel ne possède pas un caractère d'originalité particulier, étant donné que la plupart des mini-ordinateurs sont mis en oeuvre grâce à eux pour les applications en temps réel.

Par contre, au niveau des matériels à base de microprocesseurs, rares sont encore les systèmes dotés de cette possibilité.

Cette étude nous permettra, dans un premier temps, à partir des moniteurs existants, d'en définir un adapté à nos besoins.
Nous décrirons ensuite les possibilités d'utilisation du système ZS 500 piloté par le moniteur RTES85 pour des applications d'acquisition et de restitution de données.

Nous terminerons enfin par la génération du moniteur et du système d'acquisition.

1. ETUDE D'UN MONITEUR TEMPS REEL

Cette étude se déroule en deux étapes que l'on définit de la façon suivante :

- Intérêt du Moniteur Temps Réel et définition du besoin (buts recherchés, moyens)

- Structure générale d'un Moniteur Temps Réel adapté à un micro-ordinateur.

1.1. INTERET DU MONITEUR TEMPS REEL

1.1.1. Utilisation maximum du calculateur

L'évolution des matériels et l'accroissement des travaux à effectuer nécessitent d'appréhender différemment la façon de réaliser les traitements.
En effet, les contrôles de processus effectués par des ensembles de systèmes à logique câblée sont maintenant centralisés par des systèmes programmés.
D'autre part, l'apparition de ces systèmes avec leur puissance de calcul et leur souplesse d'utilisation a favorisé la prise en compte de paramètres de plus en plus nombreux qui étaient négligés jusqu'alors.
Chaque traitement est lié à un organe ou groupe d'organes d'entrée ou de sortie (cartes d'interface avec le processus). Les acquisitions et les restitutions correspondantes sont éventuellement périodiques et de périodes très variées, ou non périodiques. Ceci met en évidence la nécessité de lier un programme de traitement à un organe d'entrée/sortie.

A un instant donné, l'unité centrale ne peut exécuter qu'un programme. *Le rôle du moniteur temps réel est de donner le contrôle de l'unité centrale à l'un ou l'autre programme en fonction des priorités, du temps et de divers évènements pouvant se produire au niveau des organes d'entrées-sorties.*
(Ex : dépassement de seuil, action sur bouton-poussoir).
Par ce fait, un autre but recherché, qui était d'obtenir une réponse rapide à des sollicitations externes, pourra être atteint.
Nous verrons ultérieurement que ces programmes prennent alors le nom de *tâches*, les évènements extérieurs étant les *interruptions*.

1.1.2. Souplesse d'utilisation des entrées-sorties et interruptions

Ainsi que nous venons de l'évoquer, le maintien de l'environnement temps réel, c'est-à-dire le traitement *rapide* et *hiérarchisé* des informations, nécessite une gestion des interruptions. Celle-ci peut être faite par le programme d'application. L'inconvénient est que ceci alourdit sensiblement l'application et la rend fortement dépendante du matériel.
De même, la gestion des entrées-sorties peut être réalisée par le programme d'application.
Nous partirons du principe que le traitement des entrées-sorties se fait par interruption et non en mode programmé. En effet, en gérant l'interruption liée à l'organe d'entrée-sortie, pendant l'échange, il est possible de réaliser d'autres traitements en *pseudo-simultané*, alors qu'en mode programmé le système reste en attente de fin d'échange sans effectuer les autres traitements moins prioritaires.

Cette prise en compte des entrées-sorties par l'application pose le même problème que la gestion des interruptions car elle rend l'application étroitement dépendante du matériel et alourdit le travail de l'utilisateur.
Nous avons donc vu que la gestion des interruptions et des entrées-sorties par le moniteur décharge l'utilisateur d'un certain nombre de traitements généraux utilisables par n'importe quelle application et rend celle-ci indépendante de la configuration matérielle (adresses des périphériques).
Ces traitements d'entrées-sorties sont valables pour les périphériques classiques (imprimante, console de visualisation, ligne asynchrone).
La gestion des entrées-sorties acquises ou restituées par des cartes spécifiques sera développée dans le Chapitre 4 qui décrit le système d'acquisition et de restitution des informations.

1.1.3. Autres possibilités

Outre les points évoqués précédemment, un Moniteur Temps Réel met à la disposition de l'utilisateur un ensemble de moyens facilitant l'écriture de l'application et augmentant sa puissance de traitement.
Parmi ceux-ci, nous citerons brièvement :

- gestion des tâches logicielles (traitements simultanés, traitements différés,traitements périodiques)
- partage des ressources à l'aide de sémaphores
- gestion du temps et des délais
- système temps réel d'aide à la mise au point
- allocation dynamique de la mémoire.

1.1.4. Conclusion

Nous venons de montrer l'intérêt et les avantages que peut présenter un Moniteur Temps Réel en évoquant succinctement ses possibilités.

Nous pouvons les résumer ainsi :

- effectuer la gestion des ressources matérielles et logicielles du système afin d'en décharger l'application
- permettre d'utiliser au maximum la puissance de traitement du système
- rendre possible le découpage fonctionnel de l'application.

1.2. STRUCTURE D'UN MONITEUR TEMPS REEL

Après avoir défini l'élément de l'application traité par le Moniteur (La Tâche), nous décrirons les éléments constitutifs d'un Moniteur Temps Réel :

- le distributeur qui gère l'exécution des tâches en temps réel
- l'ensemble de tâches système immédiates
- les modules de traitement des requêtes moniteur
- le traitement des entrées-sorties

1.2.1. La tâche

1.2.1.1. Définition

Dans le paragraphe précédent (1.1.), nous avons parlé traitements ; la simultanéité de ces traitements nous a amené à évoquer la notion de tâche.
L'application est découpée en autant de tâches qu'il y a de traitements à réaliser en parallèle.
Ceci nous amène à définir ce qu'est la tâche.
Elle est *l'élement de programme connu du Moniteur.*
En effet, elle est l'association *d'un programme, d'un contexte d'exécution et d'un niveau de priorité.*

- le *programme est la suite d'instructions qui réalise le traitement.* A ce niveau apparait la notion de *modularité* qui permet à l'utilisateur d'effectuer un *découpage fonctionnel de son application.*
- *le contexte d'exécution est l'état de l'ensemble des registres du calculateur* (accumulateur, indicateurs, registres, pointeur de pile, pointeur d'instructions) caractérisant l'état du traitement à un instant donné.

Lors d'une suspension du traitement (interruption, attente), le contexte est sauvegardé en mémoire ; celui-ci est restitué pour permettre la poursuite de l'exécution de la tâche lorsque la suspension a été traitée.

- *Le niveau de priorité associé à une tâche est représentatif du degré d'urgence d'exécution de celle-ci.*

  Il existe deux types de priorité, la priorité *matérielle* et la priorité *logicielle* :

  . la priorité matérielle représente une hiérarchisation de signaux électriques (interruptions) les uns par rapport aux autres. Ce genre de priorité est traité par les circuits électroniques de l'unité centrale ; les tâches associées à des interruptions sont dites *tâches immédiates*.

  . la priorité logicielle représente une hiérarchisation par le moniteur des tâches auxquelles ne sont pas associés de niveaux d'interruption matériels. Cette hiérarchisation est traitée par une partie du moniteur appelée distributeur ou "scheduler" ;
  les tâches qui ne sont pas associées à des interruptions sont dites *tâches différées*.

1.2.1.2. Etats d'une tâche

Une tâche immédiate ou différée peut se trouver dans différents états. Le passage d'un état à un autre s'effectue à l'aide des primitives de synchronisation du moniteur, lors des opérations d'entrées-sorties ou de l'apparition d'évènements externes : interruptions.

Ces états sont les suivants :

- Dormant

  La tâche n'est pas connue du système.
  Dans le cas d'une tâche immédiate, le niveau d'interruption qui lui est associé n'est pas activé ou la tâche est terminée (désactivation du niveau d'interruption).
  Dans le cas d'une tâche différée, celle-ci n'a pas été activée ou bien elle est terminée.

- Actif ou éligible

  Une tâche immédiate est active lorsque le niveau d'interruption auquel elle est associée est démasqué et excité. La tâche reste dans cet état tant que son niveau d'interruption n'est pas désactivé.
  Une tâche différée est active lorsque le moniteur connait son existence ; ceci est réalisé grâce à une requête d'activation qui met le contexte de la tâche dans la file d'attente du distributeur.

- En cours

  Le programme sur lequel a été créé la tâche s'exécute.
  Une tâche immédiate ou différée est en cours s'il n'y a pas de tâche plus prioritaire active ou bien si la tâche plus prioritaire est suspendue.
  Dans le cas d'une tâche différée, le choix de la tâche à éxécuter est fait par le distributeur.
  Dans le cas d'une tâche immédiate, ce même choix est fait par la logique de traitement d'interruption (câblée).

- Suspendu

Une tâche immédiate ou différée a été créée avec des conditions d'attente ou a été suspendue au cours de son exécution. Les causes d'attente ou de suspension sont diverses :

. attente de libération de ressource
. attente d'activation d'un évènement
. attente de fin de délai
. attente de fin d'entrée-sortie

La suspension de la tâche en cours permet au distributeur de donner le contrôle de l'unité centrale à la tâche éligible la plus prioritaire.
La possibilité pour une tâche d'être suspendue et de laisser le contrôle à une tâche de niveau de priorité inférieur ou égal est une des caractéristiques essentielles d'un moniteur temps réel. Ceci permet en effet à l'application d'occuper l'unité centrale au maximum.

Les différents états d'une tâche et les transitions entre ces états sont résumés par le graphe représenté figure 1.1.

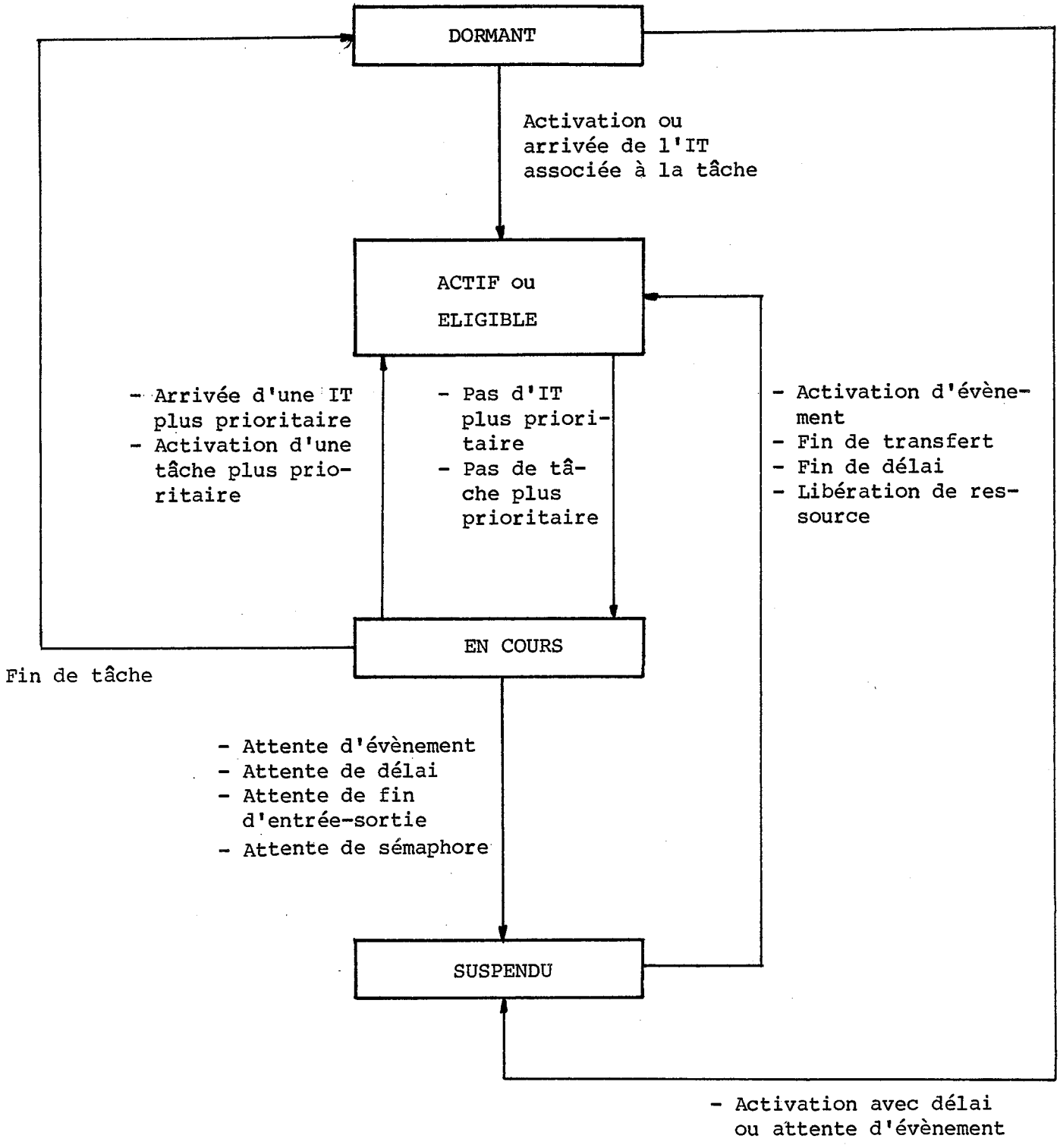


Figure 1.1. : Etats d'une tâche

1.2.2. Le distributeur

1.2.2.1. Définition et rôle

Le distributeur est un module de programme du moniteur qui gère les *tâches différées*. Il s'exécute en général au niveau d'interruption le plus bas. Il a pour but d'élire une tâche différée en consultant pour cela les files d'attente des tâches éligibles en mémoire. Son rôle est de permettre d'utiliser au maximum les possibilités du calculateur en gérant le temps où l'unité centrale n'est pas occupée (entrées-sorties, délais, ressources, évènements) et en le ré-affectant à d'autres tâches.

En effet, dans le contexte de multi programmation, plusieurs tâches sont, à un moment donné, susceptibles de s'exécuter.

Le distributeur devra *"ordonner"* les tâches prêtes à s'exécuter en fonction du niveau de priorité. Son but essentiel sera de minimiser le temps de commutation des tâches et de *réagir le plus rapidement possible à toute modification de l'état du système et ceci pour qu'à tout moment, ce soit la tâche disponible la plus prioritaire qui soit en exécution.*

Nous avons vu que les tâches évoluent entre divers états ; ceux-ci sont conservés dans la table des tâches gérée par le système.

Le distributeur travaille sur l'ensemble des tâches "éligibles" chainées généralement entre elles dans une file suivant leur priorité. Sur le plan strict du choix de la tâche prête la plus prioritaire, le rôle du distributeur est de prendre celle qui est en tête de file.

L'assignation des priorités aux tâches influe directement sur leur ordonnancement dans la file des tâches prêtes, donc sur le fonctionnement du distributeur. Lors de sa création, une tâche est placée dans la file des tâches "éligibles" en tête de celles de priorité inférieure, mais après celles de priorité supérieure ou égale.
Le distributeur est susceptible d'être appelé dans les cas suivants :

- lors de modifications de la file des tâches "éligibles" (activation d'une tâche, blocage d'une tâche, déblocage d'une tâche)
- à la fin de l'exécution d'une tâche
- lors des attentes et activation d'évènements de fin d'entrées-sorties et de délais
- lors des demandes et des libérations de ressources

1.2.2.2. Types de distributeurs

Le distributeur est l' *implémentation d'un algorithme de choix* qui doit être élaboré avec soin pour le système considéré. Cet algorithme peut être simple ou très complexe, limité seulement par les possibilités physiques du système. Il est conçu à partir d'un ensemble de régles qui déterminent à quel moment particulier une tâche peut être exécutée.
Un tel algorithme dépend entièrement des priorités assignées aux tâches et du principe de choix des tâches.

En ce qui concerne l'assignation des priorités, les types sont les suivants :

- statique

  dans ce type de distributeur, les priorités sont assignées aux tâches une fois pour toutes, soit à la génération du système, soit à la création de la tâche

- dynamique

  dans ce type de distributeur, les priorités des tâches peuvent être modifiées par une requête moniteur.

En ce qui concerne le principe de choix des tâches, les types sont les suivants :

- choix réduit ou non préemptif

  ce type de distributeur favorise le *débit global du système.*
  En effet, le distributeur n'est appelé que *lorsqu'une tâche différée cesse d'être dans l'état "en cours"* ;
  c'est-à-dire dans les cas suivants : fin d'une tâche, attente d'un évènement, attente de libération d'une ressource, attente d'un délai.
  L'avantage de cet algorithme est de réduire les temps de changement de contexte, son inconvénient est de perturber la gestion en temps réel des tâches, une tâche différée plus prioritaire pouvant ne pas être remise dans l'état "en cours" après avoir été "suspendue".

- choix préemptif

  ce type de distributeur privilégie *la rapidité de réponse* de la tâche la plus prioritaire du système à un moment donné.
  En effet, *à chaque changement d'état du système* (activation ou attente d'évènement, libération ou attente de ressources, etc...), le distributeur est activé, afin d'exécuter la tâche la plus prioritaire du système.

Dans la majorité des cas, le distributeur est du type à *choix préemptif* et à *priorité fixe*.

1.2.3. Tâches système immédiates

Ces tâches ont pour but de traiter les interruptions standard du calculateur. Elles sont connectées aux niveaux excités par les évènements suivants :

- coupure tension
- mise sous tension
- horloge

1.2.3.1. Tâche coupure tension

Le rôle de cette tâche est de sauvegarder l'état du système (registres, indicateurs, pointeur de pile, pointeur ordinal, etc..) lors de la disparition de la tension d'alimentation, afin de permettre, lors du retour de celle-ci, une reprise correcte de l'exécution de l'application là où celle-ci a été interrompue. Ceci ne s'applique que si le système est doté d'une mémoire vive non volatile (mémoire à tores de ferrite, mémoire à semi-conducteurs secourue par batterie).
Les entrées-sorties en cours au moment de la coupure sont perdues et devront éventuellement être relancées.

1.2.3.2. Tâches de mise sous tension

Le rôle de cette tâche est d'initialiser totalement le système lors de la mise sous tension (entrées-sorties, files d'attente, tables, etc..).
Dans le cas d'une reprise après coupure, le contrôle peut être rendu à la tâche interrompue.
Les programmes de traitement de ces interruptions (coupure et mise sous tension) sont en général connectés aux niveaux d'interruption les plus élevés.

1.2.3.3. Tâche horloge

Le rôle de cette tâche est de traiter les circuits de gestion du temps du calculateur. A l'arrivée de chaque interruption provoquée par l'horloge du système, il y a mise à jour de l'heure et des délais en cours. L'expiration de ces derniers entrainera l'activation du distributeur (choix préemptif).
De même, cette tâche gère le lancement des tâches périodiques par l'intermédiaire du distributeur.

1.2.4. Modules de traitement des requêtes moniteur

Un moniteur met à la disposition de l'utilisateur un ensemble de services. Ceux-ci lui permettent de mettre en oeuvre l'environnement temps réel du système et lui fournissent certaines facilités.
Ces services sont activés par des requêtes moniteur qui mettent l'application en rapport avec celui-ci.
Ces requêtes sont constituées de séquences d'instructions permettant de passer sous contrôle du moniteur en lui transmettant des paramètres précisant l'action que l'application veut réaliser.
Dans le chapitre suivant, nous préciserons quelles sont les requêtes mises à la disposition de l'utilisateur par chaque moniteur.
Nous citerons ici les plus courantes :

- initialisation de la date et de l'heure du système
- demande de la date et de l'heure
- lancement d'une tâche
- lancement d'une tâche périodique
- "meurtre" d'une tâche
- fin d'une tâche
- lancement d'un délai
- demande de mémoire dynamique

- libération de mémoire dynamique
- activation d'un évènement
- attente d'un évènement
- demande de sémaphores (privés et partagés)
- libération de sémaphores
- lancement d'une entrée-sortie
- actions sur les bascules d'interruptions (masquage, validation, excitation..)

Certaines des requêtes que nous venons d'évoquer permettent de travailler sur des éléments connus (initialisation ou prélèvement de la date et de l'heure) ou de réaliser des actions que nous avons évoquées précédemment (lancement ou arrêt d'une tâche, action sur les bascules d'interruptions, lancement d'une entrée-sortie). D'autres, par contre, servent à manipuler des concepts moins familiers que nous allons expliciter ci-après :

- Evènement

  C'est un élément qui peut prendre 2 états : *"actif"*, *"non actif"*.

  L'attente sur un évènement *"actif"* est transparente vis-à-vis de la tâche utilisateur (pas de suspension de celle-ci) ; par contre, si ce même évènement est *"non actif"*, la tâche est suspendue et ne repassera à l'état actif que lorsque l'évènement sera activé (fin de l'entrée-sortie ou du délai auquel est associé l'évènement, activation de l'évènement par une autre tâche).

- Sémaphore

  C'est un élément qui peut prendre 2 états : *"occupé"*, *"non occupé"*. La demande par une tâche utilisateur d'un sémaphore, dont au moins un point d'entrée est libre, entraine l'occupation d'un point d'entrée de ce sémaphore et la poursuite de la tâche (pas de suspension de celle-ci).

La demande par une tâche d'un sémaphore, dont tous les points d'entrée sont occupés, provoque la suspension de la tâche jusqu'à ce qu'un point d'entrée soit libéré par une autre tâche.

Un sémaphore peut comporter un ou plusieurs points d'entrée (n), ceci étant précisé lors de la description de celui-ci. Dans le premier cas, une seule tâche pourra l'occuper, dans le second cas, n-1 tâches pourront l'occuper.

Lors de l'attente d'un évènement ou lors de la demande d'un sémaphore, la suspension de la tâche (lorsque celle-ci intervient) où s'effectue la requête correspondante provoque le passage à l'état "en cours" de la tâche active de priorité immédiatement inférieure (cf § 1.2.2. Le distributeur).

- Mémoire dynamique

L'attribution de la mémoire de travail (variable) à une tâche est généralement statique (toute la durée de vie de la tâche). Cependant, afin d'optimiser l'utilisation de l'espace mémoire (celui-ci étant limité), on peut imaginer une attribution dynamique de la mémoire, toutes les informations en mémoire de travail n'ayant pas nécessairement la durée de vie de la tâche.
Dans ce cas, la tâche qui a un besoin de mémoire à un instant donné (acquisition d'un tableau et calcul sur les éléments de celui-ci) demandera un bloc de mémoire de la taille désirée. Le moniteur rendra à la tâche l'adresse de ce bloc afin qu'elle puisse travailler.
S'il n'y a pas de mémoire disponible, la tâche sera mise en attente jusqu'à ce qu'il y ait une zone disponible.
Après utilisation, la tâche rend ce bloc de mémoire au moniteur afin qu'une autre tâche puisse l'utiliser.

1.2.5. Traitement des entrées-sorties

Nous avons indiqué précédemment qu'un des rôles du moniteur était de mettre à la disposition de l'utilisateur un ensemble de facilités en ce qui concerne les entrées-sorties, bien que ceci ne fasse pas à proprement parler partie de la fonction moniteur.
Ces modules sont généralement appelés *handlers* ou *drivers*. Leur rôle est d'effectuer la gestion des entrées-sorties sur les périphériques auxquels ils sont associés. En effet, chaque module est spécifique d'un périphérique.
Le rôle du module du moniteur traitant la requête d'entrée-sortie est de contrôler la faisabilité de celle-ci (vérification du bloc de contrôle, nombre d'octets, numéro de périphérique etc..) puis d'appeler le module traitant l'entrée-sortie proprement dite. Celui-ci est décomposé en plusieurs parties :

- une partie de lancement de l'entrée-sortie
- une partie d'entretien de l'entrée-sortie par relancement de celle-ci lors de l'apparition de l'interruption.

Dans ce module, sont réalisées les opérations de mise à jour de l'adresse du tampon et du compteur d'octets de l'entrée-sortie, ainsi que les éventuelles opérations de transcodage et de contrôle ou de génération de parité.
Lorsque l'entrée-sortie est terminée, le module active l'évènement associé à celle-ci et retourne un compte-rendu indiquant comment s'est effectuée l'opération, afin de rendre éventuellement le contrôle (si celui-ci avait été perdu par une attente de fin d'entrée-sortie) à la tâche ayant lancé l'entrée-sortie.

2. ANALYSE DES SYSTEMES EXISTANTS

Pour mettre en oeuvre leurs applications, la plupart des utilisateurs sont, à l'heure actuelle, obligés d'améliorer les moniteurs existants ou d'en développer d'autres ; on trouve ces programmes élaborés pour des applications bien particulières dans les laboratoires d'automatique de l'EDF et de sociétés privées. Peu de constructeurs ont développé, à ce jour, un tel produit, en particulier la société INTEL propose un moniteur temps réel, le RMX/80 susceptible de satisfaire une clientèle industrielle.

Si de nombreux et rapide progrès ont pu être enregistrés ces dix dernières années dans la réalisation matérielle des systèmes micro-ordinateurs, on ne peut pas constater, à l'heure actuelle, les mêmes progrès dans le domaine du logiciel. Très peu de logiciels associés aux ordinateurs ont été transférés sur les micro-ordinateurs. Seuls, quelques programmes, conçus pour le développement et la mise au point d'applications, existent et sont généralement vendus par les constructeurs. Il n'existe actuellement que peu de systèmes micro-ordinateurs équipés de la totalité des ressources logicielles nécessaires pour effectuer l'automatisation complète d'un processus industriel.

Ceci peut s'expliquer par le fait que dans beaucoup d'applications, les micro-ordinateurs sont utilisés comme sous-système ne traitant qu'une tâche unique, ce qui élimine les logiciels de temps partagé et entraine une conception différente du "temps réel", car chaque système demandeur voit sa demande satisfaite par une unité centrale qui lui est associée.

La notion d'interruption liée à l'unicité de l'unité centrale perd de son importance ; elle s'avère souvent superflue, entrainant ainsi une gestion plus simple des programmes.

Par cette explication, on peut comprendre pourquoi la plupart des constructeurs de micro-calculateurs ont développé des moniteurs qu'ils qualifient quelquefois de "Moniteurs Temps Réel", mais qui réalisent uniquement les fonctions :
- d'aide à la mise au point
- de traitement des entrées-sorties
- de traitement des interruptions

Notre étude des systèmes existants se décompose en deux parties :
- étude de moniteurs d'aide à la mise au point
- étude de moniteurs temps réel.

2.1. MONITEURS D'AIDE A LA MISE AU POINT

Les fonctions de ces systèmes seront mises en évidence dans les pages qui suivent par l'étude comparative de trois moniteurs réalisés pour des microprocesseurs différents par des constructeurs différents.
Les moniteurs choisis comme base de l'étude sont :

- Le MOZ 80 de ZILOG conçu pour le Z80 et utilisé sur la carte McBZ80
- Le MON85 de INTEL conçu pour l'utilisation du 8085
- Le MINIBUG 3E de MOTOROLA conçu pour l'utilisation du 6800

2.1.1. Caractéristiques et cadre des produits

Ces trois moniteurs sont des produits de bas de gamme ; ils utilisent comme support d'informations le ruban papier ou la cassette magnétique.
Ces moniteurs offrent à l'utilisateur les fonctions nécessaires au test et à la mise au point des applications utilisant les microprocesseurs 8 bits de leurs constructeurs respectifs. Ce sont des superviseurs d'opérations, contrôlés par commandes opérateurs.
Ils sont généralement implantés en mémoire PROM ou REPROM.

- Le moniteur MOZ80
  Il est étudié pour être implanté sur la carte McBZ80 fabriquée par ZILOG. Il utilise les circuits disponibles sur cette carte : l'USART (gestion de ligne série asynchrone) assurant l'interface imprimante et une section du CTC (temporisation-décompteur) pour en générer l'horloge. La vitesse de transmission et les temporisations nécessaires au fonctionnement de l'imprimante sont déterminées par des paramètres symboliques définis à l'assemblage du programme.

Différentes versions du moniteur MOZ80 sont donc possibles, spécifiques à un type d'imprimante :

. le MOZ80T prévu pour fonctionner avec le terminal SILENT 700 ASR TEXAS 1200 bauds

. le MOZ80N prévu pour fonctionner avec une télétype ASR33 110 bauds

- Le moniteur MON85

  Il est conçu pour l'utilisation des 8085. Il communique avec l'utilisateur par l'intermédiaire d'un équipement périphérique (téléimprimeur ou terminal de visualisation) en utilisant les entrées-sorties série du 8085.

  A la différence du MOZ80, le MON85 détermine la vitesse de transfert, en bauds, du terminal lors de l'initialisation du microprocesseur.

- Le moniteur MINIBUG 3E

  Il est conçu pour l'utilisation du 6800 de MOTOROLA. Comme les deux moniteurs précédents, il communique avec l'utilisateur par un terminal, la liaison également assurée par un circuit d'interface série asynchrone (ACIA).

  Le MINIBUG 3E est initialisé au départ pour une vitesse de périphérique de 110 bauds.

Ces moniteurs occupent peu de place en mémoire morte (1 Koctets pour MOZ80 et MINIBUG 3E, 2 Koctets pour MON85) ainsi qu'en mémoire vive (moins de 256 octets).

2.1.2. Description des principales fonctions remplies par les moniteurs étudiés

Le moniteur communique avec l'utilisateur par l'intermédiaire d'un équipement périphérique généralement appelé terminal (terminal imprimant ou de visualisation). Il permet d'introduire, de vérifier, de contrôler et d'exécuter des programmes.

Pour réaliser ceci, le moniteur comporte des modules permettant :

- de modifier la mémoire, d'afficher et de modifier les registres de l'unité centrale
- de programmer l'entrée et la sortie d'informations à partir de la console
- de reconnaitre des interruptions

2.1.2.1. Commandes moniteur

Le dialogue entre l'opérateur et le moniteur consiste en l'envoi de commandes par l'utilisateur, exprimées dans le langage de commande du moniteur. Celles-ci sont suivies de réponses du moniteur, soit sous forme de messages imprimés, soit sous forme d'exécution d'actions. La syntaxe des commandes est spécifique à chaque moniteur, mais on retrouve dans chacun d'eux un certain nombre de commandes remplissant des fonctions identiques. Le nombre de commandes mises à la disposition de l'utilisateur est environ d'une dizaine par moniteur (celles-ci sont souvent liées à la structure matérielle des microprocesseurs).

. Commandes communes aux moniteurs étudiés

- Chargement en mémoire vive d'un programme

  Cette commande effectue le chargement en mémoire vive d'un programme stocké sur cassette ou sur ruban papier suivant le terminal utilisé.

- Liste d'une zone héxadécimale

  Cette commande permet de lister en code héxadécimal une zone mémoire dont on aura défini l'adresse de début et l'adresse de fin.

- Sauvegarde mémoire

  Cette commande permet de recopier sur cassette ou sur ruban (suivant le terminal utilisé) une zone de mémoire comprise entre deux adresses.

- Visualisation et/ou modification de mémoire

  Cette commande permet d'examiner et/ou de modifier le contenu d'emplacements mémoire.

- Visualisation et/ou modifications des registres

  Cette commande permet de visualiser et/ou de modifier les registres de l'unité centrale.

- Commande d'exécution d'un programme

  Cette commande permet de transférer le contrôle de l'unité centrale du moniteur à un programme utilisateur.

- Points d'arrêts

  Ils sont gérés par plusieurs commandes : certaines de ces commandes permettent d'arrêter l'exécution d'un programme à une adresse donnée ; d'autres permettent de reprendre l'exécution du programme à partir de cette adresse.

Ces commandes, très utiles lors de la mise au point d'un programme, sont différemment élaborées suivant les constructeurs. En effet, sous MON85, il n'existe pas de commande de pose de point d'arrêt. Cette fonction peut cependant être réalisée par modification de l'instruction sur laquelle on désire s'arrêter.

. Commandes spécifiques à chaque moniteur

MOZ80

- Conversion d'un paramètre héxadécimal en décimal

  Cette commande permet de réaliser le transcodage et l'édition d'un paramètre frappé au clavier au format héxadécimal.

- Opérations arithmétiques

  Cette commande permet d'additionner deux nombres signés frappés au clavier et d'éditer le résultat.

- Initialisation de pile

  Le pointeur de la pile utilisateur est initialisé à la valeur spécifiée.

- Initialisation d'une zone mémoire

  Cette commande provoque l'écriture de la valeur spécifiée dans la zone mémoire délimitée par les adresses fournies dans la commande.

MON85

- Commande de déplacement mémoire

  Cette commande permet de transférer une zone mémoire dans une autre zone.

- Trace d'une instruction

  Cette commande provoque l'exécution de l'instruction pointée par le compteur de programme et l'édition de l'état des registres.

MINIBUG 3E

- Trace d'une instruction

  Traitement identique à celui de la commande correspondant au MON85.

- Trace de plusieurs instructions

  Cette commande provoque le même traitement que la commande ci-dessus pour le nombre d'instructions spécifié.

2.1.2.2. Traitement des entrées-sorties

Un ensemble de sous-programmes est mis à la disposition de l'utilisateur.
Comme pour les commandes moniteur, si on examine les sous-programmes d'entrées-sorties des trois moniteurs, on constate que certains d'entre eux remplissent les mêmes fonctions.

. sous-programmes communs aux moniteurs étudiés
  Ces sous programmes permettent de réaliser les actions suivantes :
  - écriture d'un caractère sur l'imprimante
  - écriture d'un caractère sur cassette ou sur ruban perforé
  - lecture d'un caractère envoyé par le clavier avec ou sans écho sur l'imprimante.

  Ces entrées-sorties se font exclusivement en mode programmé, ceci signifie qu'aucun autre traitement ne peut être réalisé pendant l'opération d'entrée-sortie, en dehors des programmes sous interruption.

. sous-programmes spécifiques à chaque moniteur

MON85

- Lecture d'un caractère

  Ce sous-programme effectue la lecture d'un caractère de 8 bits à partir d'un lecteur de ruban perforé.

MOZ80

- Sortie sur cassette ou ruban

  Ce sous-programme effectue l'écriture sur cassette ou ruban perforé du contenu d'une zone mémoire.

- Impression du contenu de registres

  Un ensemble de sous-programmes permet d'imprimer le contenu de divers registres.

MINIBUG 3E

- Lecture et transcodage de caractères

  Ce sous-programme convertit en binaire deux caractères ASCII héxadécimaux rentrés à partir du clavier.

- Impression de caractères

  Ce sous-programme provoque l'impression d'une chaine de caractères ASCII située en mémoire.

2.1.2.3. Traitement des interruptions

Alors que dans les systèmes à mini-calculateurs, les interruptions sont souvent utilisées, dans les systèmes à microprocesseurs, elles le sont encore peu.

L'interruption est le moyen câblé permettant, sur demande de l'extérieur, d'arrêter provisoirement le programme en cours pour faire passer en priorité un programme, associé à la demande, appelé programme d'interruption.
L'interruption est utilisée dans le cas où l'application n'a pas le temps de scruter en permanence les demandes extérieures. Elle permet donc de réagir rapidement à toute modification importante du processus.

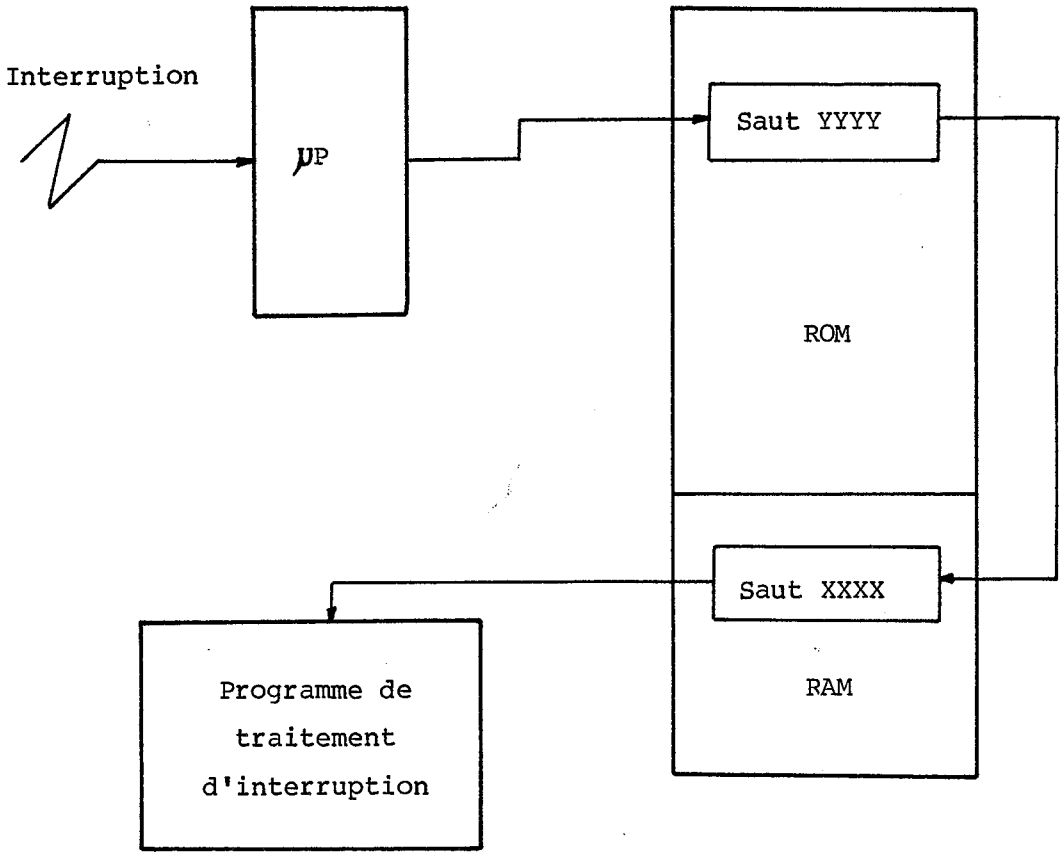
Elle peut être masquable ou non masquable.
Dans le premier cas, le programme peut contrôler et être maitre des séquences où il ne veut pas être interrompu. Celles-ci sont appelées séquences critiques.
Le second type d'interruption ne peut pas être inhibé.

Une interruption est générée une seule fois au lancement du système, c'est celle produite par la mise sous tension du matériel. Elle est bien entendu non masquable.
Une autre interruption également non masquable est excitée par programme. Cela signifie qu'elle est produite par l'exécution d'une instruction particulière. Elle a principalement pour rôle de réaliser la fonction "point d'arrêt".

- Récapitulatif des différents moyens d'interruptions de chaque microprocesseur

8085	6800	Z80	FONCTIONS
RESET	RESET	RESET	Interruptions non masquables. Ce signal permet de se brancher à des adresses déterminées du Moniteur. L'adresse du programme est perdue. On ne peut donc parler de véritable interruption. Ce processus est utilisé pour lancer l'exécution du programme d'initialisation de l'unité centrale à la mise sous tension.
TRAP	NMI	NMI	Interruptions non masquables. Ces signaux permettent à tout moment de se brancher à des adresses déterminées dans le moniteur. Le contexte est sauvegardé. Il s'agit bien ici d'une interruption. Elle est utilisée pour les systèmes de développement d'aide à la mise au point. Le fait d'être non masquables les rend incontrôlables par logiciel
(1)5,5 (2)6,6 (3)7,5	IRQ	IRQ	Interruptions masquables. Ces signaux permettent, à condition que le logiciel l'ait autorisé (démasquage), de se brancher à des adresses déterminées dans le moniteur. Le contexte est sauvegardé. Ce sont les interruptions à employer dans le cas d'un problème temps réel qui nécessite le recours à l'interruption. Le niveau d'interruption 7,5 est réservé au moniteur pour la fonction pas à pas.
RST	SWI	RST	Interruption logicielle (instruction). Ce signal d'interruption est généré lors de l'exécution de l'instruction correspondante. Il provoque le débranchement dans la séquence de traitement associée. Il permet, entre autres, de réaliser la fonction "point d'arrêt".

Les moniteurs étudiés permettent à l'utilisateur de disposer de toutes les possibilités de fonctionnement du mode interruption des microprocesseurs. D'une façon générale, nous pouvons dire que :
les interruptions masquables ou non provoquent un branchement à une adresse située dans la zone moniteur avec sauvegarde de tout ou partie du contexte sur la pile.
Pour laisser à l'utilisateur toute la souplesse d'emploi nécessaire, le moniteur comporte à ces adresses particulières, des instructions de branchement vers une zone RAM spéciale du moniteur où l'utilisateur a la possibilité de programmer de nouvelles instructions de branchement vers ses propres programmes de traitement de l'interruption.
Ce principe est illustré par le schéma ci-dessous.



- Le MOZ80 offre des commandes permettant de visualiser et de programmer le masque d'interruption ainsi que le registre de vecteur d'interruption. Il n'y a pas lieu de craindre des interférences entre le programme utilisateur et le moniteur, car ce dernier n'utilise pas les interruptions.

- Dans le MON85, les interruptions sont autorisées lorsque le moniteur rend le contrôle à l'application. Les interruptions sont rendues inopérantes pendant les interactions entre les instructions utilisateur/moniteur, de sorte que les interruptions en cours ou en attente n'interfèrent pas avec la vérification des programmes.

2.1.3. Conclusion

Les différentes fonctions des moniteurs décrites et partiellement analysées dans les pages précédentes, ne peuvent pas satisfaire pleinement un utilisateur industriel. En effet, les micro-ordinateurs utilisés en milieu industriel jouent le rôle de contrôleurs de plus en plus sophistiqués et sont destinés aux applications "temps réel" et plus particulièrement aux applications de contrôle de processus. Pour de telles applications, les tâches confiées aux microprocesseurs sont très variées ; on peut répartir l'ensemble de ces tâches en deux catégories :

- les tâches d'entrées-sorties
- les tâches de calcul des grandeurs de commande

Un micro-ordinateur industriel doit, en effet, acquérir ou restituer un très grand nombre de données qui se présentent sous les formats les plus divers et qui doivent être lues, écrites et traitées à des rythmes différents.

Ces moniteurs que nous venons d'examiner ne permettent pas de réaliser ce genre de traitements. Ceci nous conduit à nous intéresser à l'étude des moniteurs temps réel proprement dits.

2.2. MONITEURS TEMPS REEL

Les systèmes précédents n'ont pas répondu à nos besoins dans l'optique d'un contrôle de processus industriel. Nous allons étudier deux moniteurs temps réel dont les possibilités répondent de façon plus satisfaisante aux principes et aux besoins définis dans le chapitre 1.

Ces moniteurs sont le M6800 d'EDF et le RMX80 d'INTEL.

Ils ont été développés dans des buts différents.

En effet, le premier a été conçu par un utilisateur pour une application particulière (transmission de données).
Le second a été réalisé par un constructeur de microprocesseurs pour répondre à des besoins généraux.

2.2.1. Le Moniteur M6800

Ce moniteur, développé par EDF, est destiné à mettre en oeuvre des applications temps réel sur des systèmes à base de microprocesseurs MOTOROLA 6800.
Il permet en effet la gestion des évènements aléatoires par l'intermédiaire de tâches immédiates ou différées, ainsi que la synchronisation d'activités entre celles-ci.
Il dispose, en outre, d'un système de gestion dynamique de la mémoire et d'un système de communication entre tâches réalisé par l'intermédiaire de files de blocs.

2.2.1.1. Notion de tâche dans M6800

La tâche est le seul élément de traitement reconnu par M6800 ; comme dans tout autre système, une tâche est une séquence de code à laquelle est associé un niveau de priorité et une pile dans laquelle est sauvegardé le contexte.
Nous allons décrire les deux types de tâches traitées par M6800.

- Tâches immédiates
  Ce sont les tâches auxquelles M6800 donne le contrôle sur interruptions externes. La configuration matérielle associée est prévue pour que le niveau d'interruption IRQ (interruption masquable) du 6800 soit multiplexé sur 8 sources d'interruption distinctes.
  Sous ce moniteur une tâche immédiate ne peut pas se mettre en attente de sémaphore, le distributeur n'a, en effet, pas la possibilité de suspendre et de réactiver une tâche immédiate. Elle ne peut être interrompue que par d'autres tâches immédiates plus prioritaires.

- Tâches différées
  Ce sont les tâches auxquelles M6800 donne le contrôle par l'intermédiaire du distributeur. Le nombre de niveaux de priorités logicielles est limité à 16.
  Une tâche différée est interrompue par une tâche immédiate ou une tâche différée plus prioritaire ; elle peut se mettre en attente de sémaphore.

Il existe d'autre part une tâche logicielle de fond à laquelle M6800 donne le contrôle lorsqu'aucune autre tâche (immédiate ou différée) n'est "en cours".

2.2.1.2. Tâches systèmes

Il existe deux tâches système activées l'une par le RESET, l'autre par l'interruption horloge.
La première qui n'est pas à proprement parler une tâche est lancée par la mise sous tension. Elle effectue l'initialisation des variables du moniteur et active les tâches de l'application qui ont été déclarées lors de la génération du système. Elle initialise également certains dispositifs logiciels de l'utilisateur (sémaphores, évènements..) ; elle lance ensuite le moniteur proprement dit.
La tâche horloge décrémente à chaque interruption horloge le compte des sabliers armés déclarés par l'application. Au passage par zéro d'un sablier, elle lance sur son niveau d'interruption l'exécution d'un sous-programme associé à ce sablier par l'application.

2.2.1.3. Description générale des primitives moniteur

- Gestion des interruptions

L'action sur le système d'interruptions permet d'intervenir sur le déroulement des tâches immédiates, l'application ayant la possibilité de masquer et démasquer individuellement un niveau d'interruption.

- Gestion des tâches différées

  Des primitives permettent d'activer et d'arrêter une tâche à partir d'une autre tâche. Ceci correspond à réaliser les transitions entre l'état "dormant" et l'état "actif" ou "en cours" et l'état "dormant" (arrêt).

- Gestion des sémaphores

  Il existe deux types de sémaphores, les sémaphores de mutuelle exclusion (plusieurs points d'entrée) qui permettent de gérer des ressources réquisitionnables par plusieurs tâches, les sémaphores privés (un seul point d'entrée) qui permettent de gérer des ressources réquisitionnables par une seule tâche.
  Pour ces deux types de sémaphores, il existe deux actions possibles :
  . occupation d'un point d'entrée, s'il y en a un de disponible, ou attente de libération dans le cas contraire
  . libération d'un point d'entrée.

- Gestion de files de blocs

  Ce système permet de résoudre le problème de l'accumulation de messages à traiter.
  Une tâche quelconque est productrice dans une file lorsqu'elle introduit des blocs dans la file.
  Une tâche différée est consommatrice dans une file lorsqu'elle extrait des blocs de la file.
  Si la file est vide lors d'une demande de blocs, la tâche se met en attente sur le sémaphore associé à la file. Les actions possibles sur ces files de blocs sont les suivantes :

. Production d'un élément dans la file en mode premier entré, premier sorti (FIFO) ou dernier entré, premier sorti (LIFO), avec réveil des tâches consommatrices en attente sur la file.
. Consommation d'un élément dans la file avec ou sans attente lorsque la file est vide.

- Gestion dynamique de la mémoire

Ce système permet à une tâche de s'allouer des blocs d'espace mémoire. A l'aide de la gestion de files de blocs, les tâches peuvent échanger entre elles des zones de mémoire dynamique. Ces blocs peuvent également être utilisés par la tâche pour ses besoins en mémoire de travail, ils seront libérés après utilisation.
Les primitives de gestion de la mémoire dynamique sont les mêmes que celles réalisant la gestion des files de blocs. En effet, l'espace de mémoire dynamique est structuré en file de blocs de mémoire allouable.

- Gestion du temps

Le moniteur M6800 gère un ensemble de sabliers programmés pouvant être armés individuellement par les tâches de l'application. Une primitive permet d'armer un sablier en donnant sa valeur initiale et l'adresse du sous-programme à appeler au passage par zéro du sablier (cf tâche horloge § 2.2.1.2.).
Une autre primitive permet de désarmer un sablier. C'est-à-dire qu'elle interrompt le décomptage du temps pour le sablier. En conséquence, il n'y aura pas de nouvel appel au sous-programme de traitement associé.

L'exécution des sous-programmes de traitement sur le niveau d'interruption de l'horloge peut éventuellement entrainer des perturbations. En effet, si la durée du ou des sous-programmes à un instant déterminé est supérieure à la période d'interruption, les traitements pourraient ne pas s'exécuter dans leur ordre logique (dernier sous-programme appelé fini d'exécuté avant le précédent).
La solution, pour éviter ce phénomène consiste à lancer une tâche différée dans le sous-programme afin de libérer le plus tôt possible le niveau d'interruption horloge.

2.2.1.4. Traitement des entrées-sorties

Le moniteur M6800 n'offre pas de facilités particulières pour la gestion des entrées-sorties, en particulier pour les périphériques standard (imprimante à clavier ou console de visualisation sur ligne asynchrone).
Il appartient à l'utilisateur de gérer les transferts en traitant le niveau d'interruption lié à l'organe d'entrée-sortie.

2.2.2. Le Moniteur RMX/80 D'INTEL

Ce moniteur a été développé pour un matériel précis (SBC 80/10, 8/20, 80/30). Il possède, outre le noyau temps réel, un système de gestion dynamique de la mémoire, un outil de mise au point, un programme de gestion d'entrées-sorties pour disquette, un programme de gestion d'entrées-sorties analogiques.

Ces différents produits permettent d'effectuer toutes les tâches essentielles d'un système temps réel.
Le RMX/80 peut, en effet, s'inscrire dans une approche très intéressante pour la structuration des systèmes temps réel.

- Il gère un ensemble d'opérations s'effectuant sur des structures de données générales et très complètes et, à ce titre, permet de résoudre les problèmes de conception d'un système multitâches pouvant être d'une grande complexité.
- Il permet, de façon très aisée, de faire des modifications dans un système temps réel, c'est-à-dire insérer ou supprimer une tâche, définir de nouvelles relations entre tâches. Cet aspect est essentiel pour des systèmes temps réel, en général jamais figés.

Une particularité de RMX/80 est de traiter certains dispositifs logiciels classiques (sémaphores, évènements, délais), par un principe général de communication de type boite aux lettres : "l'échange".
Des tâches produisent et consomment des messages.
L'échange est le concentrateur liant les messages et les tâches concernées par ces messages. Il se présente comme le noeud de communication entre tâches.

2.2.2.1. Notion de tâche dans RMX/80

De même que dans M6800, la tâche est l'élément de traitement disposant des ressources du système. Cependant le concept de tâche immédiate est moins apparent que dans d'autres systèmes. Seule subsiste la notion de tâche d'une façon générale.
Du point de vue de la structure, la tâche immédiate ne se distingue pas de la tâche différée. Nous décrirons plus loin les requêtes moniteur qui lui permettent de passer à l'état "en cours". Cependant, nous pouvons indiquer que le traitement de l'interruption est effectué par le moniteur (masquage de niveaux inférieurs, sauvegarde du contexte, etc..), celui-ci réveillant alors la tâche qui était en attente d'un évènement lié à l'interruption (échange d'interruption cf § 2.2.2.2.).
L'utilisateur a également la possibilité de se substituer entièrement au moniteur pour la gestion d'une interruption. Il lui appartient alors de traiter la sauvegarde du contexte, le masquage des interruptions, etc..
A chaque tâche est associée une priorité fixe ; il existe 256 niveaux de priorité utilisateur. Les priorités 1 à 128 correspondent aux niveaux d'interruption (0 à 7) (tâches immédiates).
Nous verrons ultérieurement que les tâches peuvent être créées dynamiquement et supprimées du système par des requêtes moniteur. Elles sont également créées de façon statique à l'aide de descripteurs en mémoire morte constitué lors de la génération du système.
Contrairement à M6800, il n'existe pas de tâche de fond. Elle est remplacée par une halte du processeur liée au niveau de priorité 255.

Avant de décrire les primitives du moniteur, nous développerons ce point particulier qu'est le système échange-message.

2.2.2.2. Echange-message

Pour réaliser les communications entre les tâches, RMX/80 fournit au concepteur un ensemble d'opérations qui permettent l'envoi et la réception de messages entre tâches.

Ces messages sont constitués par un ensemble d'informations banalisées. Ces paquets d'informations, destinés aux tâches, doivent passer préalablement par un concentrateur liant les messages et les tâches concernées par ces messages : "l'échange". L'échange se présente donc comme un noeud de communication entre tâches. Cette structure comporte, ainsi, des pointeurs pour accéder à deux files d'attente :
- une file de messages attendant des tâches
- une file de tâches attendant les messages.

Cette structure est créée à l'initialisation du système par une primitive spéciale. Il faut noter que la destination du message n'est pas la tâche mais l'échange.

L'information, quelquefois, peut ne pas être banalisée : c'est le cas d'un message signalant l'arrivée d'une interruption, ce message étant destiné à un échange particulier. Dans cette configuration, on utilise donc un échange d'interruption, qui est alors un noeud de communication n'acheminant qu'un certain type de message.

2.2.2.3. Description générale des primitives moniteur

- Gestion des interruptions

Un ensemble de requêtes permet d'agir sur le système d'interruption.

. Autorisation et mise hors service d'un niveau d'interruption
Lors du premier appel de la requête d'autorisation, la partie message du descripteur d'échange d'interruption associé au niveau d'interruption est initialisée.

. Initialisation d'un vecteur d'interruption
Cette requête permet de remplacer un programme de traitement d'interruption moniteur par un programme utilisateur dont l'adresse est fournie au système.

. Fin de traitement d'interruption utilisateur
Deux requêtes réalisent cette fonction ; elles signalent à RMX/80 que l'interruption est traitée. Le moniteur acquitte celle-ci. Une des deux requêtes permet en plus d'envoyer un message à l'"échange" d'interruption. Ceci a pour effet de réveiller toute tâche en attente sur l'"échange" d'interruption.

- Gestion des tâches

Des primitives permettent de créer et de supprimer dynamiquement des tâches. La création d'une tâche implique que le programme associé est déjà en mémoire ou peut être chargé en mémoire. On notera, d'autre part, que plusieurs tâches peuvent être créées sur le même programme, à condition bien entendu que celui-ci soit réentrant, c'est-à-dire qu'il ne modifie pas de variables statiques (les variables de travail sont sur la pile ou en mémoire dynamique).

La création d'une tâche entraine également son lancement (elle rentre dans la file d'attente des tâches prêtes).
Deux primitives permettent à une tâche d'en suspendre une autre ou de la remettre dans l'état "prêt".
Il faut cependant noter que dans un contexte temps réel, la suspension d'une tâche n'est pas essentielle, les relations entre tâches étant déterminées avant le lancement du système. Ce mécanisme reste utile dans la phase de mise au point.

- Gestion des sémaphores

Il n'existe pas de primitive spécifique pour la gestion des sémaphores. Cette fonction peut être réalisée à l'aide des requêtes d'envoi et d'attente des messages.
A l'initialisation, un message est envoyé à l'échange sur lequel est créé le sémaphore privé.
Toute tâche voulant utiliser le sémaphore se mettra en attente d'un message sur l'échange (consommation de message).
L'obtention d'un message bloquera alors les autres tâches qui se mettraient ultérieurement en attente sur l'échange.
La libération du sémaphore par la tâche qui l'a utilisé consiste à envoyer un message à l'échange, débloquant ainsi la tâche la plus prioritaire en attente.
On peut étendre ce principe aux sémaphores de mutuelle exclusion en envoyant plusieurs messages à l'échange au lieu d'un seul lors de l'initialisation du système (autant de messages que de points d'entrée), le reste du principe étant le même.

- Gestion des messages

Nous avons expliqué précédemment que le transfert des messages entre différentes tâches était réalisé par les "échanges".
Une primitive moniteur permet, de façon dynamique, de créer un échange, une autre permet d'en supprimer. Notons également que la création d'échanges peut être faite par le moniteur RMX/80 à l'aide de tables définies lors de la génération.

Trois primitives permettent de réaliser la gestion des messages au niveau des tâches.

- Envoi d'un message : cette requête provoque l'insertion de l'adresse du message dans la file d'attente de l'échange. On peut noter que l'envoi d'un message n'implique pas son transfert effectif. Il en résulte que la tâche émettrice ne doit pas altérer le message. Pour résoudre ce problème de conflit, la tâche réceptrice enverra à la tâche émettrice un message de réponse indiquant que le message a été traité. Ceci permet de réutiliser la zone mémoire pour construire un nouveau message.
- Attente d'un message : par cette requête, la tâche se met en attente sur un échange jusqu'à ce qu'un message soit reçu ou jusqu'à ce qu'un temps limité soit écoulé. Dans le premier cas, la tâche reçoit l'adresse d'un message. Dans le second cas, la tâche reçoit l'adresse d'un message système "temps écoulé".
- Réception d'un message : par cette requête, la tâche reçoit l'adresse d'un message, s'il y en a un, depuis l'échange spécifié. On s'aperçoit que contrairement à la requête précédente, il n'y a pas d'attente. Cependant, si la file d'attente de l'échange est vide, la tâche ne recevra pas d'adresse de message.

- Gestion dynamique de la mémoire

Le système d'allocation dynamique de la mémoire gère une zone de mémoire vive et fournit de l'espace mémoire aux tâches qui en font la demande. Ce système récupère aussi l'espace mémoire rendu par les tâches et le réintégre dans la zone disponible.
La mémoire vive est une des ressources principales d'un calculateur. Le système de gestion dynamique permet de réduire les besoins en mémoire vive du système. En effet, une tâche peut n'avoir besoin d'une zone mémoire qu'à un certain moment de son exécution et dans tous les cas uniquement jusqu'à sa fin. Une allocation statique de la mémoire définie lors de l'assemblage ou de la compilation ne permet pas d'arriver à ce résultat. L'allocation et la libération de la mémoire se font par envoi de messages entre la tâche et le système d'allocation de la mémoire.
Les deux principales techniques d'allocation sont celles dites du "premier choix" et du "meilleur choix". La première technique consiste à allouer le premier bloc de mémoire disponible de taille supérieure à la taille du bloc demandé.
La deuxième consiste à allouer le plus petit bloc de mémoire de taille supérieure à la taille du bloc demandé.
Dans le premier cas, il risque de se produire un gaspillage de la mémoire dynamique ; dans le second cas, le système prendra plus de temps pour rechercher le bloc de taille optimum. Afin d'obtenir le temps de réponse le plus court possible, le RMX/80 utilise le principe du "premier choix".

- Gestion du temps

Le traitement des délais est réalisé par l'utilisation de la primitive d'attente des messages.
En effet, un paramètre permet de limiter l'attente du message. Ce paramètre d'attente est le délai demandé.
Aucune autre tâche ne devra envoyer de message sur l'échange correspondant afin que le réveil de la tâche se fasse sur expiration du délai.

2.2.2.4. Traitement des entrées-sorties

Les entrées-sorties asynchrones s'effectuent en temps réel entre le terminal opérateur et les tâches par l'intermédiaire du "terminal handler" sous l'arbitrage de RMX/80.
Le "terminal handler" est conçu pour s'adapter à différents modèles de terminaux. Ces terminaux pouvant être des consoles de visualisation ou des télé-imprimeurs dont la vitesse est comprise entre 110 bauds et 9600 bauds.
Il est à remarquer que le "terminal handler" ne supporte qu'un terminal dans un système et un maximum de 132 caractères par ligne.
Le "terminal handler" réalise un certain nombre de fonctions qui permettent la simplification de la programmation d'un terminal.
Il soulage complètement les tâches utilisateurs des travaux nécessaires à la réalisation d'une entrée-sortie physique avec un terminal.
Il permet d'écrire une ligne construite par la tâche utilisatrice.

Il permet de la même façon à une tâche de recevoir une ligne frappée au clavier.
Les entrées et les sorties avec le terminal se font par envoi de message au "terminal handler" sur des échanges particuliers.

2.2.2.5. Programme d'aide à la mise au point (Debugger)

L'utilisateur dispose d'un système d'aide à la mise au point conçu pour être utilisé dans l'environnement temps réel de RMX/80. Ce dispositif permet de communiquer avec le RMX/80 à travers le "terminal handler". L'utilisateur dispose d'un jeu de commandes permettant d'effectuer certaines actions au cours de l'exécution de l'application :

- lecture et modification d'emplacements mémoire
- lecture de structures particulières du moniteur RMX/80 (liste des tâches, des échanges, des délais, etc..)
- exécution d'une procédure à une adresse déterminée
- positionnement et suppression de points d'arrêts
- reprise d'exécution après point d'arrêt.

Ce système est utilisé pendant la phase de mise au point de l'application. Il n'est plus nécessaire lorsque celle-ci est parfaitement au point et il sera supprimé en phase opérationnelle.
Cependant, on peut éventuellement le conserver si l'on veut intervenir en cas de problème sur le site.

3. ETUDE D'UN MONITEUR TEMPS REEL ADAPTE A LA CONFIGURATION DU ZS 500

3.1. RAISONS DU DEVELOPPEMENT D'UN MONITEUR

Nous avons étudié dans le chapitre précédent deux moniteurs temps réel et l'on peut s'interroger sur la nécessité d'en réaliser un autre.
Ces moniteurs sont conçus pour fonctionner sur des matériels particuliers, différents de celui sur lequel nous travaillons ; nous savons que la réalisation d'un système de contrôle d'un processus industriel nécessite l'utilisation de cartes d'interface (commande et acquisition) et que la configuration de l'unité de traitement (zone mémoire vive, zone mémoire morte, adresses d'entrées-sorties) est définie et adaptée par le constructeur pour le système de contrôle.
D'autre part, pour l'un de ces moniteurs, un certain nombre de fonctions essentielles sont absentes, pour l'autre, un trop grand nombre de possibilités entraine une occupation mémoire excessive.
En effet, le moniteur M6800 n'a pas de système de gestion des entrées-sorties et ne traite pas les délais ; en ce qui concerne le RMX/80, le système "échange-message" est puissant mais peu pratique à mettre en oeuvre.
Le moniteur que nous allons réaliser est spécialement conçu pour traiter des applications d'un volume compatible avec le type d'unité centrale utilisée et pour fournir un ensemble de requêtes nécessaire et suffisant pour le contrôle d'un processus.
Enfin, il permettra au programmeur une écriture modulaire des applications.
Avant de passer à l'étude proprement dite du moniteur, nous décrirons brièvement le matériel sur lequel il doit fonctionner.

3.2. PRESENTATION SUCCINCTE DU ZS 500

Un système ZS 500 se compose d'une configuration de base et des cartes périphériques nécessaires au pilotage du processus qu'il doit contrôler.
Dans sa configuration de base, il est constitué des éléments suivants :

- une carte d'alimentation qui génère les tensions nécessaires au fonctionnement du système. A la disparition ou à la réapparition de la tension d'entrée, elle fournit les signaux nécessaires à l'arrêt ou au redémarrage automatique du microprocesseur
- une carte décodage d'adresse permet l'adressage des cartes interfaces à l'aide de mémoires mortes programmées
- une carte unité centrale équipée d'un microprocesseur 8 bits INTEL 8085, de 1 Koctets de mémoire vive, de compteurs, de contrôleurs d'interruptions, d'un coupleur asynchrone, des supports pour mise en place de mémoires mortes reprogrammables (8 Koctets maximum)
  Si le volume du traitement le nécessite, il est possible d'augmenter la capacité mémoire par l'intermédiaire de cartes équipées de mémoire vive ou de mémoire morte reprogrammable dans la limite de 28 Koctets.

  Un ensemble de cartes d'interface permet de connecter l'unité de traitement au processus. Selon le type de chassis, le nombre maximum de ces cartes est de 10 ou de 16. Elles réalisent les fonctions suivantes :
  - interfaces d'entrées "tout ou rien". Elles permettent l'acquisition de 8 ou 16 entrées statiques par carte
  - interfaces de sorties "tout ou rien". Elles permettent la commande de 8 ou 16 sorties statiques par carte

- interfaces d'entrées analogiques. Elles permettent l'acquisition de 4 ou 8 voies
- interfacesde sorties analogiques. Elles permettent la commande de 8 voies analogiques
- interfaces de comptage. Elles permettent la prise en compte de fréquences élevées à raison de 4 ou 8 entrées par carte
- interface coupleur asynchrone. Une carte permet la connexion de 4 périphériques ou coupleurs asynchrones
- interface de pilotage clavier. Une carte permet de gérer de façon autonome le multiplexage de 12 afficheurs et l'acquisition de 64 entrées "tout ou rien" multiplexées.

Ces cartes d'interface, à l'exception des 3 derniers types, seront décrites avec plus de précision dans le chapitre suivant concernant le système d'acquisition et de restitution des informations.

Une vue du châssis ZS 500 équipé de cartes et une vue du même matériel en coffret sont représentées aux figures 3.1. et 3.2.

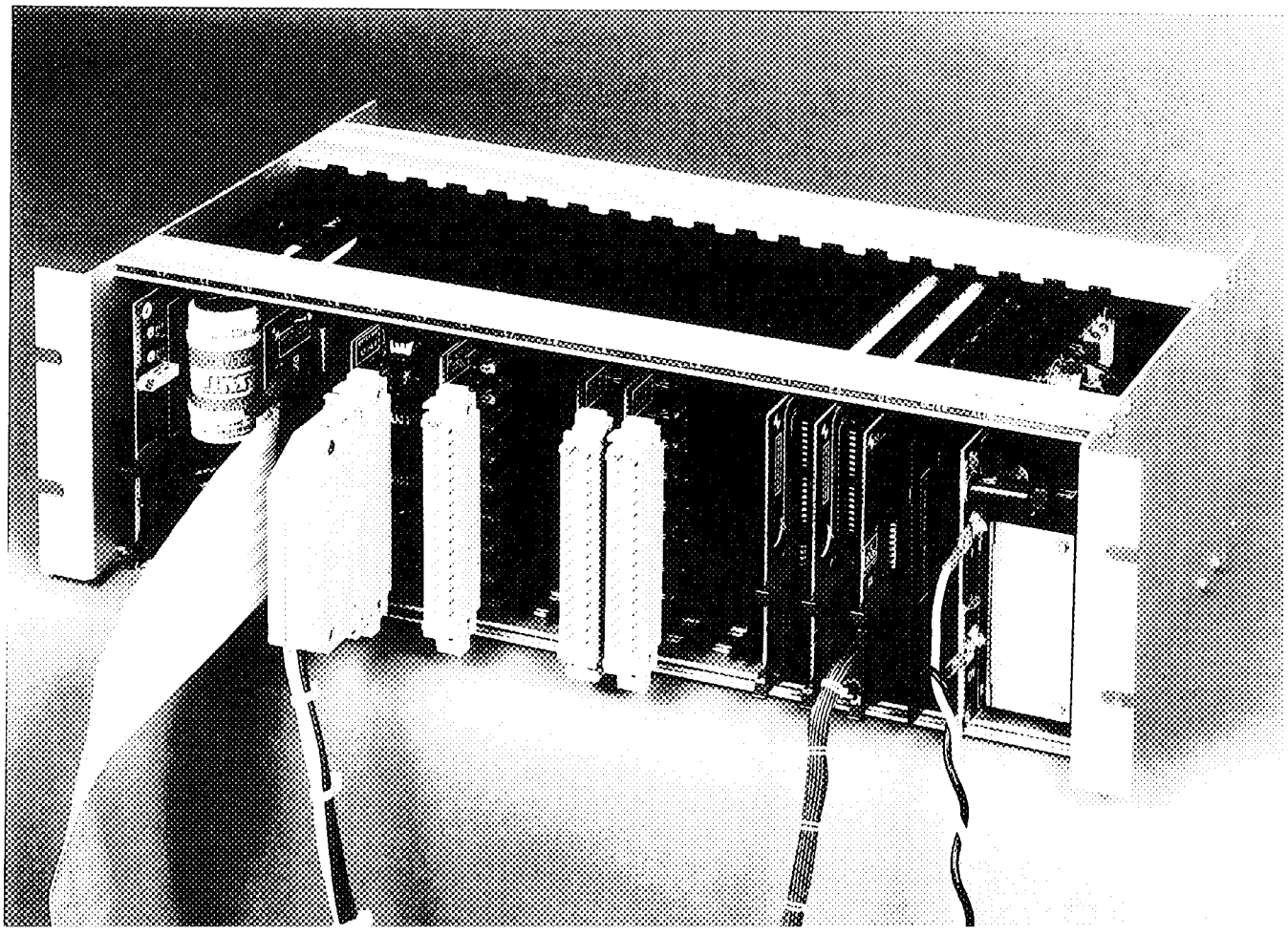


Figure 3.1. : Châssis ZS 500 équipé de cartes

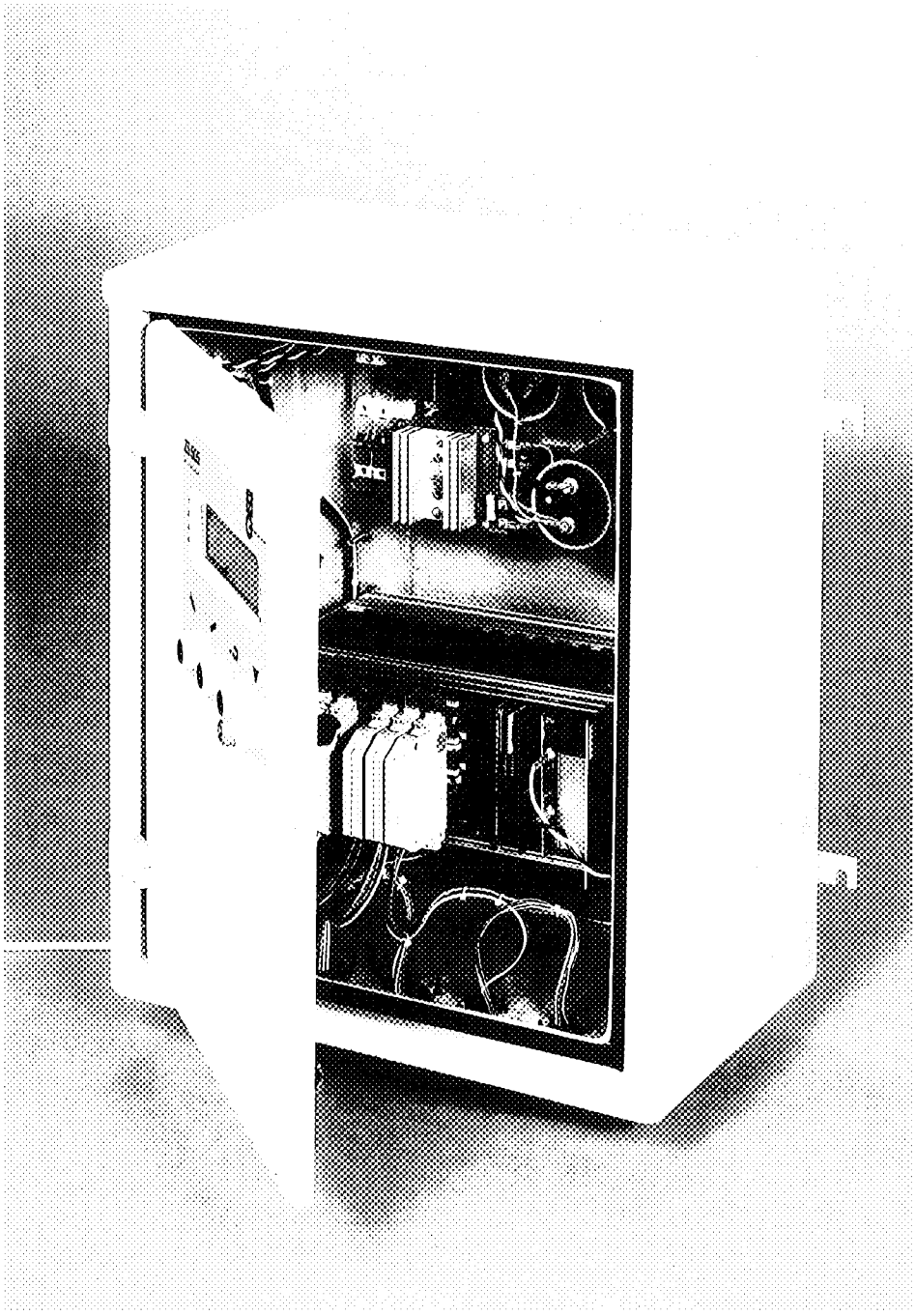


Figure 3.2. : Châssis ZS 500 en coffret

3.3. PRINCIPES ET RESTRICTIONS

Les caractéristiques du matériel et le type d'applications visé nous ont conduit à suivre certains principes et à nous fixer des limites pour la définition et l'étude de ce moniteur.
Le moniteur que nous voulons réaliser doit être mis en oeuvre sur microprocesseur. Nous devons donc limiter ses possibilités compte-tenu du niveau de puissance de calcul des microprocesseurs 8 bits. Il ne faut, en effet, pas perdre de vue que ceux-ci ne sont pas des mini-ordinateurs du point de vue de la vitesse de traitement et de l'efficacité du jeu d'instructions.
Dans ces conditions, la taille de ce moniteur sera volontairement limitée à 3 Koctets pour une version de base comprenant le moniteur proprement dit (gestion des tâches, traitement des requêtes) et le programme de gestion d'une ligne asynchrone.
Eventuellement, dans la phase de développement d'une application, on pourra adjoindre un système d'aide à la mise au point.
L'étude sera également menée de façon à ce que la mise en oeuvre soit la plus simple possible et que les concepts manipulés soient uniquement ceux nécessaires à la réalisation des applications de contrôle de processus.

3.4. ETUDE DU MONITEUR RTES85

Un système opérationnel utilisant le moniteur RTES85 est découpé de la façon suivante :
- programmes de contrôle
- programmes de traitement

La structure de ce système est symbolisée par une arborescence représentée sur le schéma de la figure 3.3.

Les programmes de traitement s'exécutent sous l'arbitrage des programmes de contrôle. Ils sont divisés en deux catégories, les tâches et les sous-programmes communs (sous-programmes partagés par plusieurs tâches).

Les programmes de contrôle qui forment le coeur du système opérationnel sont divisés en trois catégories :
- gestion des tâches
- gestion des requêtes
- gestion des entrées-sorties.

Ces programmes constituent le moniteur ou exécutif temps réel (RTE).
Le module de gestion des tâches est fréquemment appelé superviseur ; il réalise le traitement des interruptions, la gestion des tâches différées (distributeur) et la gestion du temps.
Le module de gestion des requêtes traite les commandes d'activation et de fin de tâches, de synchronisation entre tâches, les délais, le lancement des entrées-sorties.
Le module de gestion des entrées-sorties réalise les échanges avec les périphériques reliés au système.
L'organigramme fonctionnel de RTES85 est représenté par le schéma de la figure 3.4.

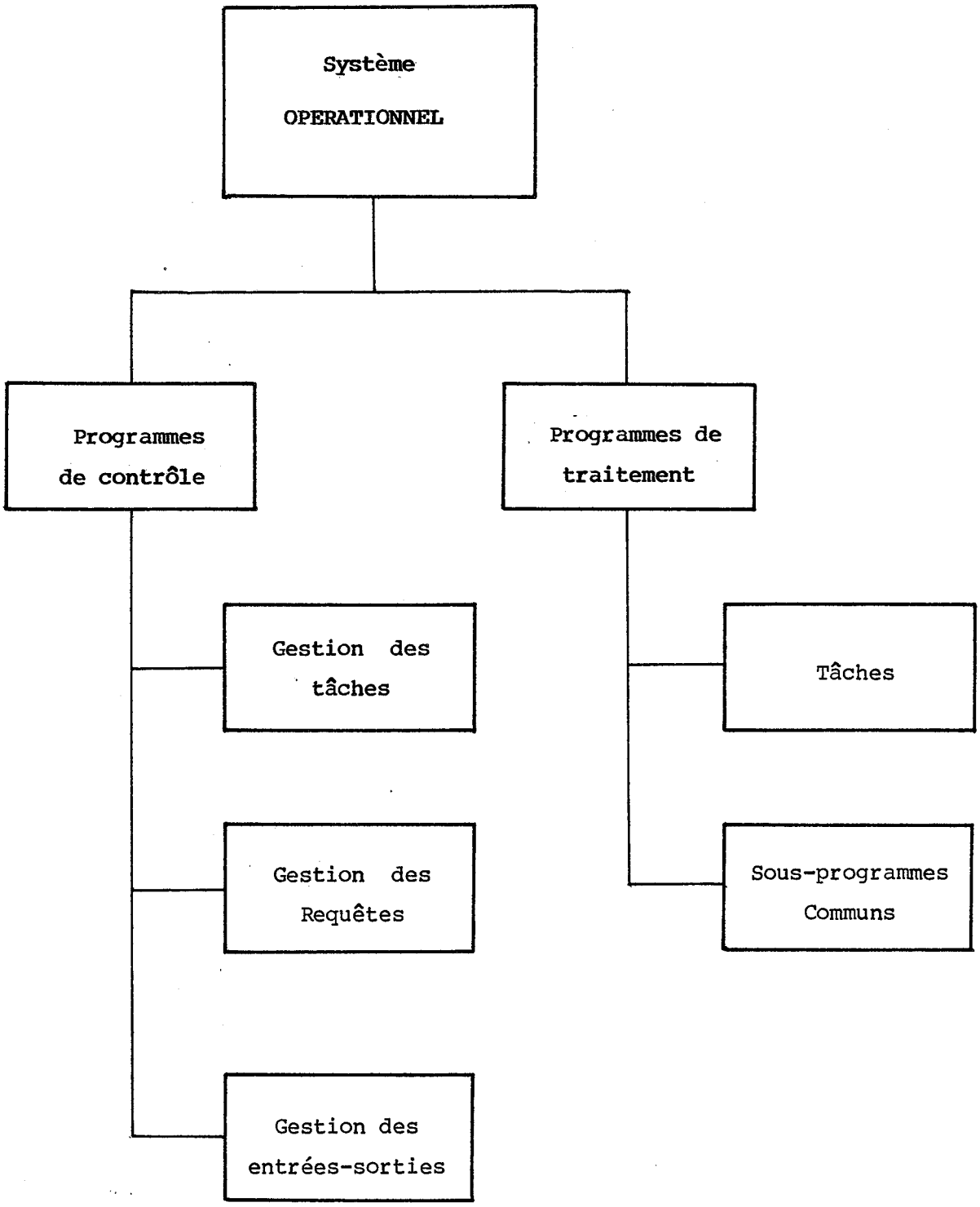


Figure 3.3. : Structure d'un logiciel Temps réel sous RTES85

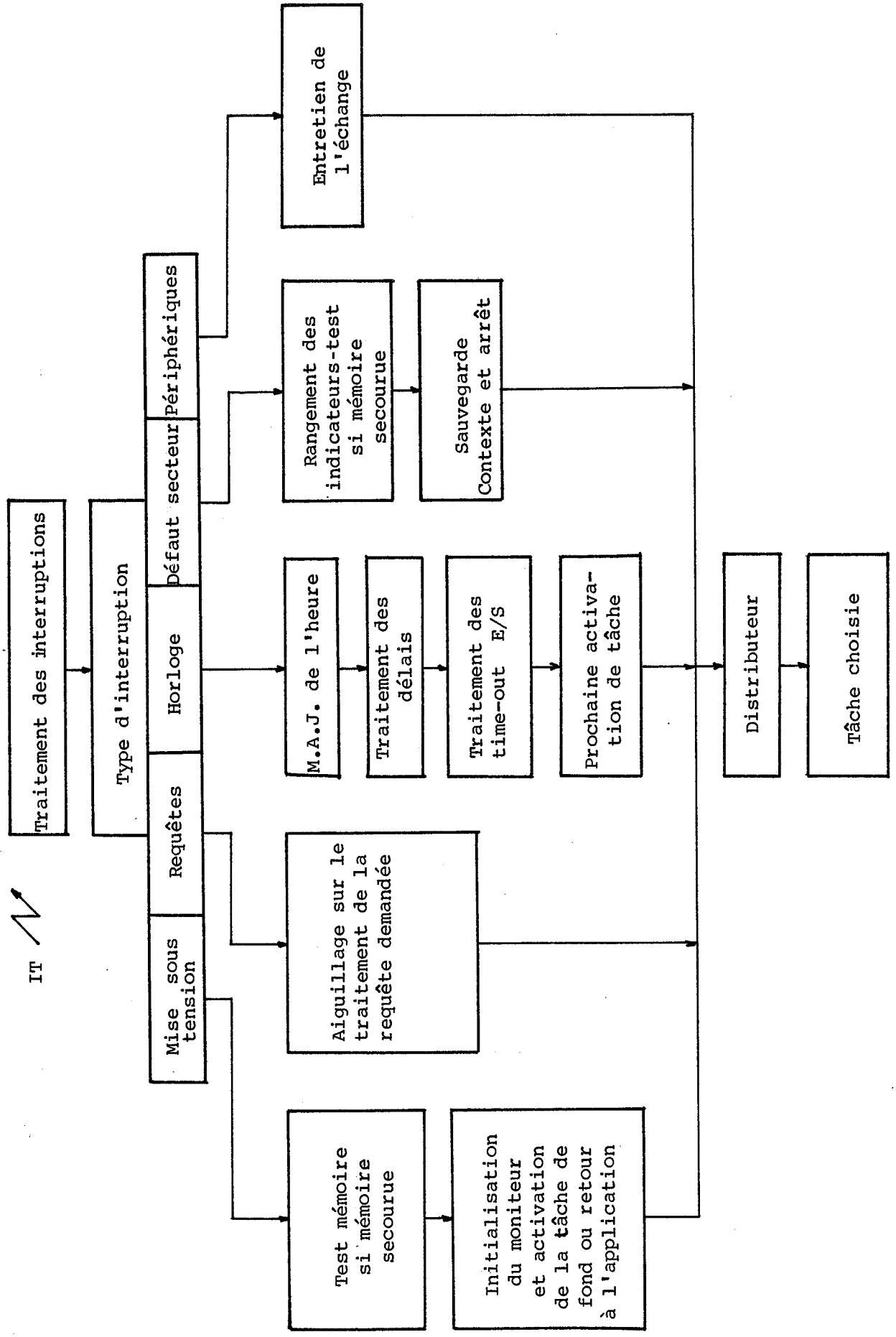


Figure 3.4. : Organigramme fonctionnel de RTES85

Cet algorithme fait apparaitre que chacun des modules du moniteur est lié à une interruption, que celle-ci soit externe (horloge, périphérique, alimentation) ou interne (requêtes). Chacun de ces modules se termine par l'activation du distributeur afin de choisir la tâche de l'application à laquelle il doit redonner le contrôle de l'unité centrale.

3.4.1. Notion de tâche dans RTES85

3.4.1.1. Tâches différées

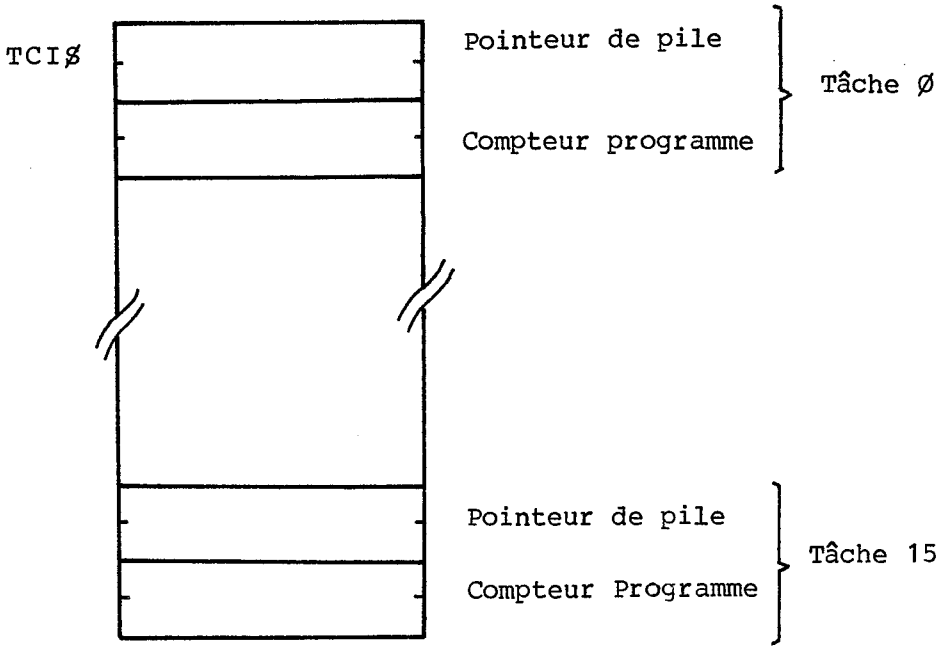
Le moniteur RTES85 gère, par l'intermédiaire du distributeur, un ensemble de tâches logicielles qui constituent l'application. Celles-ci sont au maximum au nombre de 16, de priorité décroissante de 0 à 15.
A chaque tâche est associée une priorité fixe définie lors de la génération de l'application.
Parmi ces tâches, la moins prioritaire (priorité 15) est réservée au système dans le cas où l'application est au repos, c'est-à-dire lorsque le distributeur n'a plus aucune tâche à traiter. Le processeur est alors mis en mode "HALT", ses bus étant alors libérés (aucun cycle de recherche d'instruction n'est exécuté.

Règles d'écriture des tâches sous RTES85

Une tâche est caractérisée par les éléments suivants :
- son code (instructions et données)
- son numéro de 0 à 14 (priorité)
- sa pile
- son contexte initial.

Le code sur lequel s'exécute la tâche est organisé en une ou plusieurs sections selon qu'il est implanté uniquement en mémoire vive ou bien en mémoire vive et en mémoire morte. La mémoire vive est, bien entendu, toujours nécessaire (variables, pile).

Le numéro de tâche permet au moniteur d'accéder, par l'intermédiaire d'une table, aux paramètres définissant la tâche : ceux-ci sont la valeur initiale du pointeur de pile et la valeur initiale du compteur de programme. Cette table, dont la structure est donnée ci-dessous, est implantée à une adresse fixe, connue du moniteur, dans une zone de mémoire vive ou morte utilisateur.



La pile est utilisée pour sauvegarder le contexte de la tâche lors des interruptions et lors des appels aux requêtes moniteur.
Ce contexte occupe 5 mots et est constitué des éléments suivants :
- registres B et C
- registres D et E
- registres H et L
- PSW (registre A et indicateur)
- compteur ordinal

Il appartient au programme interrompant de sauvegarder le pointeur de pile de la tâche interrompue s'il doit y avoir commutation de piles.
La pile sert également de zone de travail au moniteur pour traiter ses requêtes et à l'utilisateur pour ses besoins en zone de travail (réentrance).

Le contexte initial est défini, comme nous l'avons vu précédemment, dans la table TCI$ par le pointeur de pile (adresse début de pile) et par le compteur de programme (première adresse, de la tâche, à exécuter).

3.4.1.2. Tâches immédiates

Les tâches immédiates ne sont pas traitées par le moniteur RTES85 ; seule existe la possibilité de prendre en compte les interruptions, en spécifiant au niveau du vecteur d'interruption, l'adresse d'un programme traitant cette interruption. Ce traitement n'est pas considéré comme une tâche par le distributeur, puisqu'il est inconnu de celui-ci. Ceci peut altérer le déroulement normal de l'application lors de l'utilisation de certaines requêtes. En particulier, celles remettant en cause l'ordre d'exécution des tâches (sémaphores, entrées-sorties, délais..). Si l'on considère le cas des sémaphores, une tâche différée "en cours" peut en avoir occupé un ; si la tâche immédiate qui prend alors le contrôle, demande le même sémaphore (utilisation d'une ressource commune, par exemple une imprimante), on aboutit à un blocage de la tâche différée et du traitement d'interruption, car ceci revient à demander deux fois de suite le même sémaphore dans une tâche.

Pour pallier ces problèmes, nous transformerons le traitement de l'interruption en tâche différée. Ceci sera réalisé en activant une tâche de priorité maximum correspondant au traitement de l'interruption. Dans ce cas, la durée du traitement sous interruption sera minimum, ce qui permettra de prendre en compte une nouvelle interruption, à condition que la tâche différée soit réentrante.

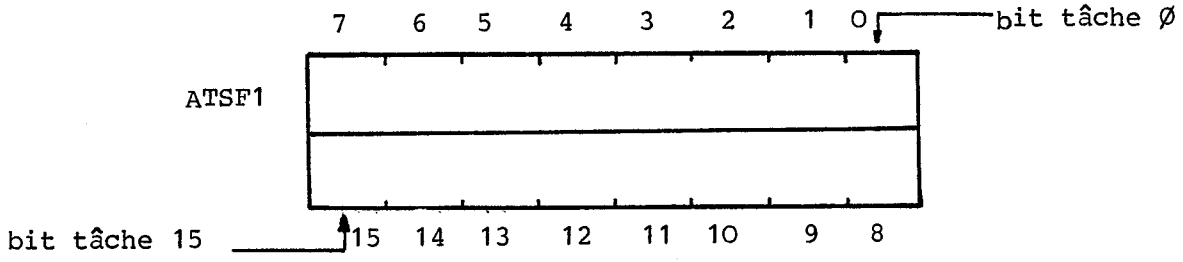
3.4.2. Le distributeur

Le souci majeur qui a guidé l'écriture du distributeur du moniteur RTES85 est la rapidité d'exécution, sans considération de volume mémoire (350 octets de programme et 50 octets de variables et tables).
L'algorithme correspondant à ce distributeur est du type à choix préemptif.
L'organigramme général de fonctionnement du distributeur est représenté figure 3.5.

Les paramètres essentiels qui sont surveillés par le distributeur sont l'état et la priorité des tâches existantes. Ces informations sont rangées dans une zone mémoire spécifiquement destinée à cet usage.

3.4.2.1. Organisation des files du distributeur

La zone mémoire servant à la gestion des tâches est organisée en deux files de bits (ATSF et ETSF).
Chacune de ces files est constituée de 2 octets, les numéros des bits (0 à 15) de ces files représentent les niveaux de priorité des tâches correspondantes.



Un bit à 1 dans la file ATSF signifie que l'exécution de la tâche correspondante est demandée ou en cours.

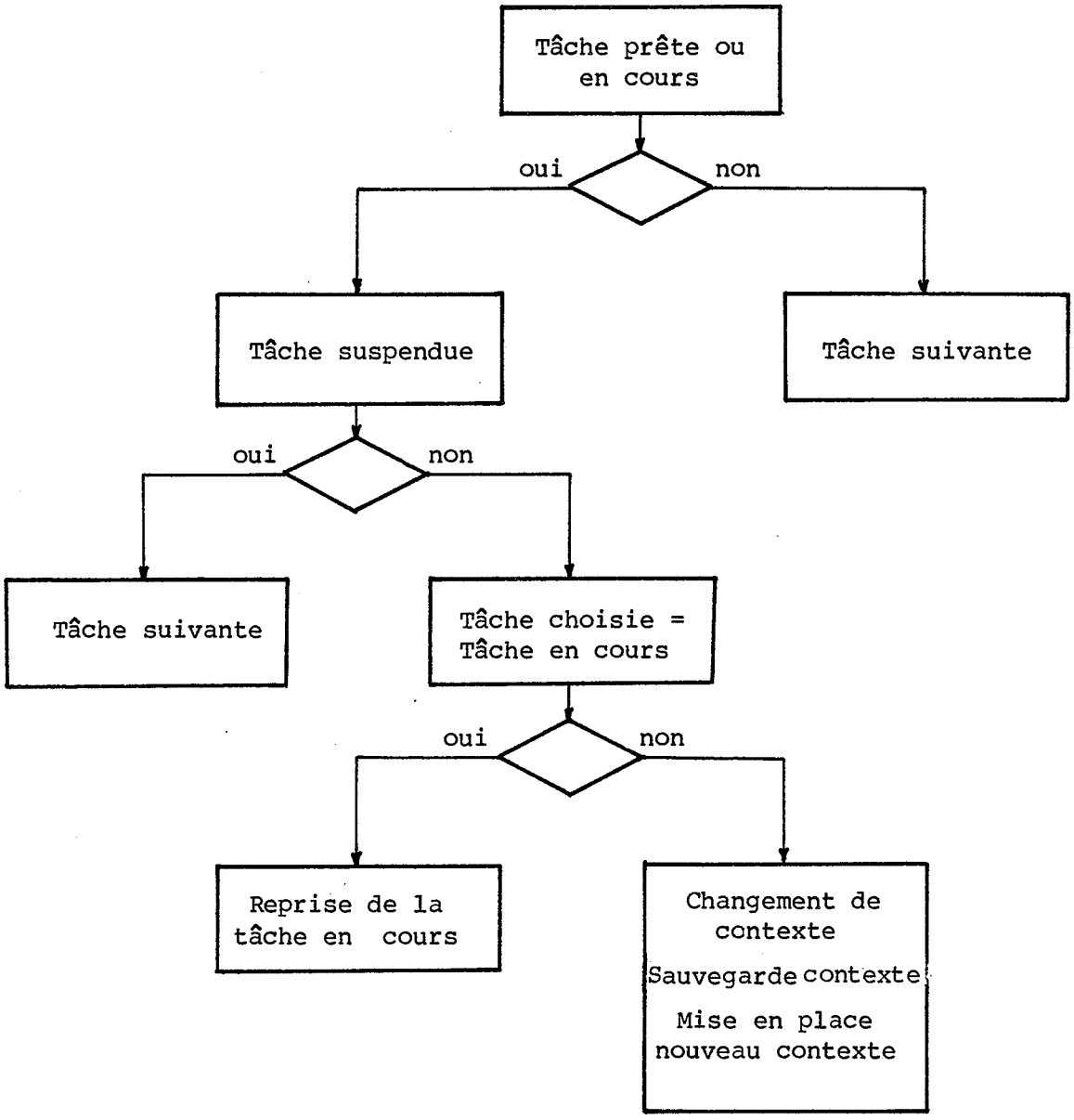


Figure 3.5. : Organigramme général du distributeur

La file ETSF est organisée sur le même modèle ; un bit à 1 signifie que la tâche correspondante est suspendue (attente de fin d'entrée-sortie, d'évènement, suspension temporisée).
Le rôle du distributeur consiste à examiner ces deux files de bits afin de déterminer la tâche éligible de plus forte priorité. Cette tâche est alors activée.
La mémoire TSK$ est mise à jour ; elle contient le numéro de la tâche active à l'instant considéré.
Le distributeur est activé chaque fois qu'un évènement peut entrainer un changement de tâche :
- requête moniteur
- fin d'entrée-sortie.

Le changement de tâche s'effectue par changement de contexte.

3.4.2.2. Changement de contexte

L'activation d'une tâche se réalise en deux temps :
- sauvegarde du contexte de la tâche suspendue
- mise en place du contexte de la tâche activée.

Le contexte de chaque tâche est sauvegardé par empilement dans sa propre pile. La valeur du pointeur de pile (haut de pile) alors obtenue est rangée dans la table des pointeurs de piles (TOS$$), au rang correspondant à la priorité de la tâche.
La mise en place du contexte s'effectue en sens inverse des opérations décrites ci-dessus : le pointeur de pile est initialisé à la valeur contenue dans la table TOS$$ au rang correspondant à la priorité de la tâche. Le contexte est ensuite dépilé et la tâche lancée par l'instruction RET (chargement du compteur ordinal).

Les programmes d'interruptions nécessitant un traitement long (horloge, fin d'échanges) utilisent la pile système.
Chaque traitement d'interruption ou de requête moniteur se terminant par un branchement au distributeur (cf figure 3.4.), celui-ci doit savoir si le contexte à changer se trouve dans la pile système ou dans une pile utilisateur.
Un indicateur (SYSTAK) permet de faire ce choix et d'autoriser l'activation d'une tâche utilisateur (aucun programme d'interruption en cours).
D'autre part, ces traitements d'interruption (horloge et entrées-sorties) peuvent provoquer le réveil d'une tâche en attente (fin de délai , fin d'entrée-sortie). Ceci sera mémorisé dans un indicateur (AFSCH$). Cet indicateur, testé dans la séquence de restauration du contexte du programme d'interruption, autorisera le distributeur à lancer la tâche d'application la plus prioritaire.
Dans tous les cas, l'indicateur SYSTAK peut interdire le fonctionnement du distributeur. En effet, il est modifié par les programmes d'interruption. Ceci permet, dans le cas de 2 traitements d'interruption imbriqués, et en cas d'activation d'une tâche (fin de délai), le retour au programme d'interruption non terminé (et non à la tâche à activer).
Un dernier indicateur (ENSCH$) permet à une requête d'interdire le fonctionnement du distributeur si celui-ci est sollicité, lors d'un retour d'un programme d'interruption (l'interruption étant intervenue au cours du traitement de la requête). Ceci évite que le distributeur travaille sur des éléments en cours d'élaboration.
Un organigramme détaillé du distributeur est représenté figure 3.6.

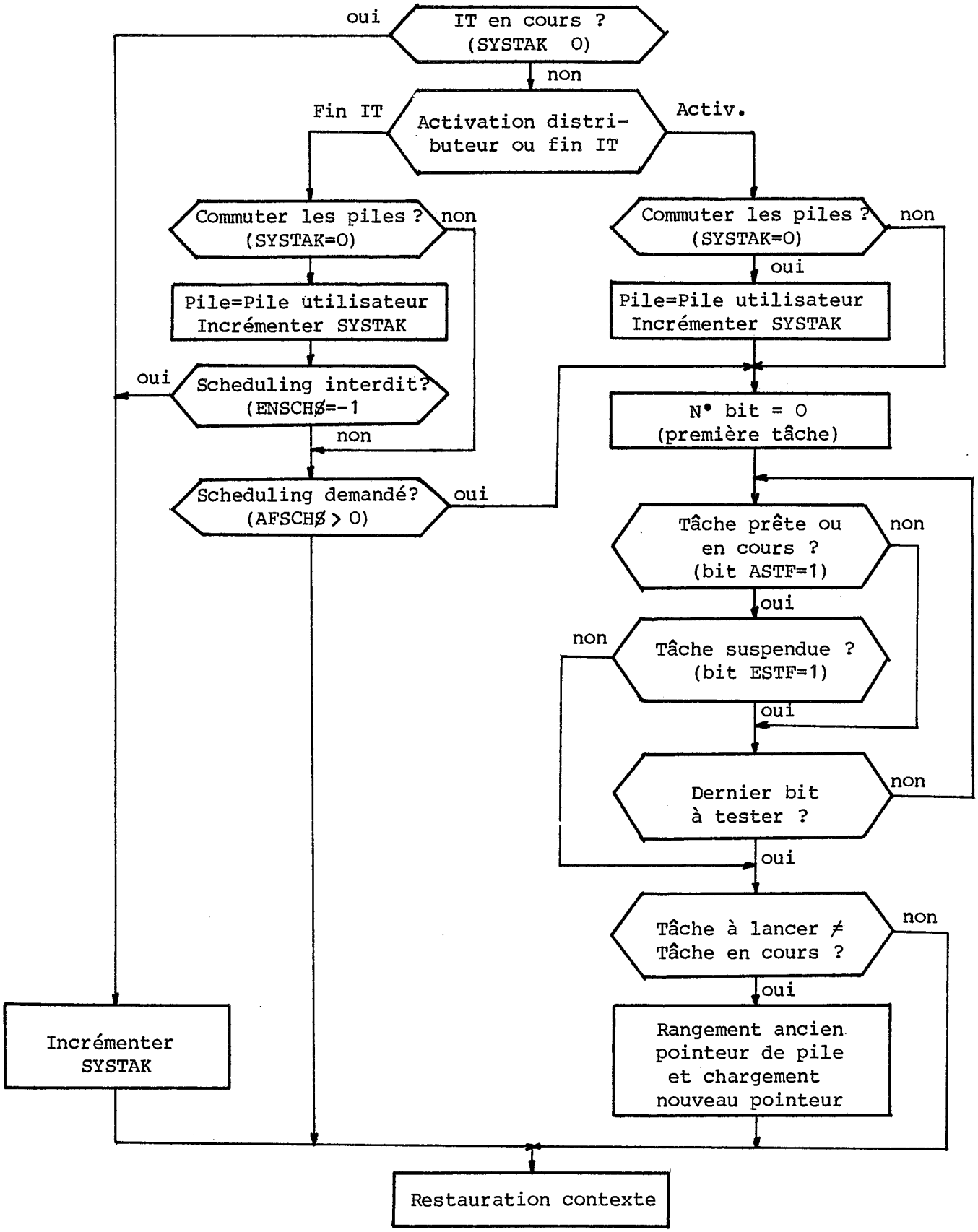


Figure 3.6. : Organigramme détaillé du distributeur

3.4.3. Tâches systèmes

On désigne sous ce terme les traitements d'interruptions pris en compte par le système, à l'exception des entrées-sorties et des requêtes moniteur. Ces deux derniers points seront développés séparément.
Nous étudierons dans ce paragraphe les traitements liés à la mise sous tension, à la coupure secteur et à l'horloge.

3.4.3.1. Mise sous tension

La séquence du moniteur liée à l'interruption de mise sous tension réalise un ensemble d'actions destinées à lancer le moniteur et l'application.
Si la configuration du ZS 500 est équipée de l'option "mémoire secourue" et si le moniteur est généré en conséquence, la tâche contrôle l'état d'un certain nombre de mémoires témoin. Ceci permet de s'assurer que la mémoire secourue n'a pas été altérée. Si le test est satisfaisant, le pointeur de pile est restauré à partir de sa valeur sauvegardée lors de la coupure secteur et l'activité est reprise par dépilement du contexte.
Si le test n'est pas concluant, les mémoires témoin sont réinitialisées et le moniteur reprend la phase d'initialisation.
Si l'option secours n'est pas présente, le moniteur est réinitialisé à chaque mise sous tension.
La séquence d'initialisation réalise les fonctions suivantes :

- Initialisations logicielles
  - initialisation des divers compteurs, indicateurs et files du distributeur
  - remise à zéro des "time-out" des périphériques
  - remise à zéro des tables des échanges (entrées-sorties)
  - initialisation des sémaphores

- Initialisations matérielles
  - initialisation des contrôleurs d'interruptions (seuls les niveaux d'interruption des contrôleurs déclarés lors de la génération du moniteur sont démasqués)
  - initialisation de la période de l'interruption horloge (50 ms)
  - démasquage des niveaux d'interruption de l'unité centrale utilisés
  - initialisation des périphériques du système définis lors de la génération .
- Simulation du lancement de la tâche de fond (constitution d'un contexte de cette tâche)
- Demande de lancement de la première tâche de l'application (choix déterminé lors de la définition du système)
- Activation du distributeur.

3.4.3.2. Coupure secteur

Cette tâche n'est définie que si le moniteur a été généré avec l'option secours mémoire. Dans ce cas, l'information défaut secteur provoque l'activation de l'interruption TRAP. Il y a alors empilement du contexte courant sur la pile en cours et sauvegarde du pointeur de pile. Le processeur est ensuite mis en HALT.

3.4.3.3. Horloge

Un compteur programmable de la carte unité centrale est initialisé pour délivrer des interruptions toutes les 50 ms.
La tâche de traitement de l'horloge est activée lors de l'apparition de chacune de ces interruptions.

Si le traitement interrompu est une tâche utilisateur (tâche différée ou programme d'interruption utilisateur), il y a commutation des piles. Le programme d'interruption horloge utilise la pile système (SYSTAØ) et le signale en décrémentant l'indicateur SYSTAK (voir le distributeur § 3.4.2.2.).

- Mise à jour de l'heure et de la date
  A chaque interruption, le compteur de temps est incrémenté. Lorsque le compteur atteint minuit, il est remis à zéro et la date est mise à jour.
- Traitement des tâches
  Ce traitement est réalisé toutes les 50 ms. Il consiste à examiner la file des tâches en attente de fin de délai ou d'activation périodique.
  Dans le cas d'un délai, si le temps arrive à expiration, la tâche en attente est réveillée par la mise à zéro du bit associé de la file ESTF (voir le distributeur § 3.4.2.1.).
  Dans le cas d'une activation périodique (cf requête activation de tâche), le bit de la file ASTF (voir § 3.4.2.1.) associé à la tâche est positionné et la tâche est activée.
- Traitement des "time-out"
  Ce traitement est également réalisé toutes les 50 ms. Il consiste à décrémenter les "time-out" en cours. Si l'un d'eux arrive à zéro, le traitement se poursuit dans le programme de gestion du périphérique auquel est associé le "time-out".

3.4.4. Traitement des requêtes moniteur

Comme dans les moniteurs déjà étudiés, les requêtes sont destinées à assurer divers services aux utilisateurs.
Le mode d'appel de ces requêtes est le suivant :
-Chargement dans les registres H et L de l'adresse d'une table
-Excitation du niveau d'interruption 7.
La structure de la table évoquée ci-dessus varie suivant la requête, mais son premier octet contient toujours le numéro de requête ; les octets suivants contiennent les informations nécessaires à l'exécution de la requête.
Les conditions de retour au programme demandeur peuvent différer selon les requêtes, mais sauf cas particuliers signalés, le contrôle est donné au distributeur.

3.4.4.1. Gestion des interruptions

Il n'existe pas de requête facilitant la mise en oeuvre du système d'interruption. Cependant, ainsi que nous l'avons déjà mentionné, l'utilisateur a la possibilité d'écrire des programmes d'interruption. Il dispose d'un jeu de macro-instructions permettant d'associer un niveau d'interruption à un programme de traitement. Il appartient à l'utilisateur de gérer directement le contrôleur d'interruptions (masquage, démasquage).
Dans la mesure où le programme est court et ne fait pas appel à des requêtes moniteur, il fonctionnera en mode ininterruptible. S'il est long, ou s'il fait appel à des requêtes moniteur, il y aura lieu de le transformer en tâche différée, comme nous l'avons vu précédemment.

3.4.4.2. Gestion des tâches

Trois requêtes servent à lancer et à terminer une tâche.

- Lancement immédiat d'une tâche
  Cette requête permet de demander au système l'activation immédiate d'une tâche. Le numéro de la tâche est rangé dans la table transmise en paramètre lors de l'appel de la requête.
  Si la tâche est déjà en exécution, le nouvel appel est ignoré.
- Lancement périodique d'une tâche
  Cette requête réalise l'activation immédiate d'une tâche et périodiquement son activation ultérieure.
  Le numéro de la tâche ainsi que sa période d'activation (en tops de 50 ms) sont rangés dans la table transmise en paramètre lors de l'appel de la requête.
  L'utilisateur devra s'assurer que la tâche n'a pas déjà été activée. Sinon les activations s'ajoutent.
- Fin d'une tâche
  Cette requête indique au système que la tâche est terminée. Celle-ci est retirée des files du distributeur et retourne à l'état "dormant".
  Seul le numéro de requête est transmis en paramètre lors de l'appel de la requête.
  Le contexte associé à la tâche qui utilise cette requête est perdu.

3.4.4.3. Gestion des sémaphores

Les sémaphores gérés par le RTES85 sont à un seul point d'entrée, cela signifie qu'une seule tâche peut en prendre possession à la fois.

- Test et occupation d'un sémaphore
  Cette requête teste l'état d'un sémaphore. Si celui-ci est libre, la tâche l'occupe et continue son exécution ; dans le cas contraire, elle se met en attente de la libération du sémaphore.
  Le numéro du sémaphore est rangé dans la table transmise en paramètre lors de l'appel de la requête.
- Libération d'un sémaphore
  Cette requête permet de libérer un sémaphore et de réveiller la tâche la plus prioritaire en attente.
  Le numéro du sémaphore à libérer est rangé dans la table transmise en paramètre lors de l'appel de la requête.

3.4.4.4. Gestion du temps

Le moniteur RTES85 met à la disposition de l'utilisateur deux requêtes liées à la gestion du temps. L'une permet d'initialiser l'heure du système, l'autre traite les demandes de délais :

- Initialisation de l'heure du système
  Cette requête effectue la mise à l'heure du système à l'aide d'une valeur transmise en tops de 50 ms.
  Cette valeur est rangée dans la table fournie en paramètre lors de l'appel de la requête.
  Le module de traitement de la requête modifie l'heure de réactivation des tâches périodiques et des tâches en attente de délais, si elles existent.

Il ajoute aux heures de réactivation, la différence entre l'heure avant la requête et l'heure envoyée par la requête.
En fin d'exécution de cette requête, le distributeur n'est pas activé.

- L'utilisateur a la possibilité d'acquérir l'heure et la date en faisant appel à deux sous-programmes inclus dans l'application lors de l'édition de liens.
- Demande de délai
  Cette requête permet à une tâche de suspendre son exécution pendant une durée déterminée. La durée de la suspension s'exprime en tops de 50 ms.
  Cette valeur est rangée sur 3 octets dans la table fournie en paramètre lors de l'appel de la requête.
  La suspension de la tâche ayant demandée le délai est réalisée par la mise à un du bit associé de la file ESTF (voir le distributeur § 3.4.2.1.).

3.4.4.5. Gestion des entrées-sorties

Le moniteur RTES85 met à la disposition des tâches utilisateur une requête permettant de réaliser des opérations d'entrées-sorties sur les périphériques du système.
L'opération est décrite dans la table fournie en paramètre lors de l'appel de la requête.
Cette table contient les éléments suivants :

- numéro du périphérique sur lequel se fait l'entrée-sortie (l'association numéro du périphérique/périphérique est réalisée lors de la génération du système)
- fonction : entrée ou sortie
- longueur de l'échange en octets
- adresse du message à envoyer ou recevoir.

La tâche ayant demandé l'opération d'entrée-sortie est suspendue pendant celle-ci. Le contrôle est rendu à la tâche en fin d'échange, ou en cas d'erreur (paramètre de retour de la requête).
Un "time-out" peut être associé à la fonction d'entrée ; celui-ci est défini lors de la génération du système. Il permet d'arrêter l'entrée-sortie si celle-ci dépasse le temps maximum autorisé par le time-out.

3.4.5. Traitement des entrées-sorties

Le ZS 500 possède, dans sa version de base, un coupleur asynchrone permettant de dialoguer avec un périphérique ou avec un calculateur.
Les possibilités d'échanges peuvent être augmentées en adjoignant une carte coupleur asynchrone. Celle-ci est équipée de 4 canaux indépendants.

Le système de gestion des entrées-sorties est constitué d'un module réentrant qui traite toutes les demandes d'échanges, et d'un programme de gestion de l'échange, également réentrant, par type de périphérique.
Le module de traitement des demandes d'échanges, appelé par la requête d'entrée-sortie, réalise la gestion des accès à chaque périphérique, par l'intermédiaire du sémaphore implicitement associé à ce périphérique. Ce mécanisme assure la mise en attente et le réveil des tâches demandant des accès simultanés au même périphérique.
Lorsqu'une demande d'échange est prise en compte, le module de traitement des échanges effectue la mise à jour des tables descriptives de l'échange, le lancement éventuel du "time-out" et l'appel du module d'initialisation du programme de gestion des échanges sur le périphérique concerné.

La tâche ayant demandé l'échange est suspendue pendant toute la durée de celui-ci.
Dans le cas qui nous intéresse, il y a un programme de gestion des lignes asynchrones pour l'ensemble des voies traitées (ligne U.C., coupleur supplémentaire).
Ce programme se décompose en trois parties :
- un module d'initialisation de l'échange
- un module d'entretien de l'échange
- un module "time-out"

- Initialisation de l'échange
  Ce module décode la fonction (entrée ou sortie) et initialise l'échange en chargeant ses variables de travail (compteur de caractères, adresse du message à lire ou à écrire) et, dans le cas d'une sortie, en envoyant le premier caractère.
- Entretien de l'échange
  Ce module est activé lors de l'arrivée de l'interruption liée à l'organe de couplage (1 caractère disponible en entrée ou bien un caractère écrit).
  Il réalise l'entretien de l'échange en lisant les caractères disponibles si c'est une entrée, ou en émettant un caractère si c'est une sortie. Il assure la gestion des comptes d'octets et des adresses des messages.
  Il détecte la fin d'échange et les erreurs au cours des transferts ; il rend alors le contrôle au module de traitement de la demande d'échange qui active le distributeur.
  Si l'échange n'est pas terminé, il restaure le contexte et rend le contrôle à la tâche interrompue.

- Module "time-out"
  Ce module élabore le compte-rendu d'échange, en cas de dépassement du temps maximum imparti et rend le contrôle au module de traitement des demandes d'échanges.
- Ligne console
  Le programme de gestion des échanges sur ligne asynchrone est conçu pour activer une tâche particulière dite "tâche dialogue", à la réception d'un caractère déterminé. Tous les autres caractères reçus hors échanges sur la ligne console sont ignorés.
  Le numéro de périphérique de la ligne console est défini lors de la génération du moniteur.

3.4.6. Aide à la mise au point

En phase de développement, il est possible d'intégrer au système, le moniteur d'aide à la mise au point, MON85, décrit dans le chapitre 2 de la présente étude.
Dans ces conditions d'utilisation, RTES85 est considéré par MON85 comme une partie de l'application et doit être généré avec les options voulues (emplacement en mémoire, traitement des interruptions).

4. UTILISATION DU MONITEUR RTES85 DANS UNE APPLICATION DE TELETRANSMISSION

4.1. FONCTION DE L'APPLICATION DE TELETRANSMISSION

Nous allons dans un premier temps situer le système dans le contexte de l'application qui nous intéresse. En fait, nous ne nous préoccuperons pas de l'application par elle-même, mais plutôt de sa structure matérielle et logicielle.

Le rôle du système à base de ZS 500 est d'effectuer l'acquisition et la restitution d'informations depuis ou vers un processus. Il consiste également à recevoir ou transmettre ces mêmes informations depuis ou vers un organe central de traitement.

L'architecture générale d'une telle configuration est schématisée par la figure 4.1.

Le système ZS 500 ainsi conçu décharge totalement l'organe central des opérations d'entrées-sorties avec le processus. Il joue alors réellement le rôle d'un processeur frontal vis-à-vis du processus. Il réalise l'acquisition des entrées tout ou rien (TOR) et analogiques qui sont transmises au poste central par l'intermédiaire d'une ligne asynchrone. Ce même central envoie au ZS 500, par ligne asynchrone, des informations destinées à commander les sorties tout ou rien et analogiques.

Les systèmes ZS 500 reliés au poste central (P.C.) sont appelés postes auxiliaires (P.A.).

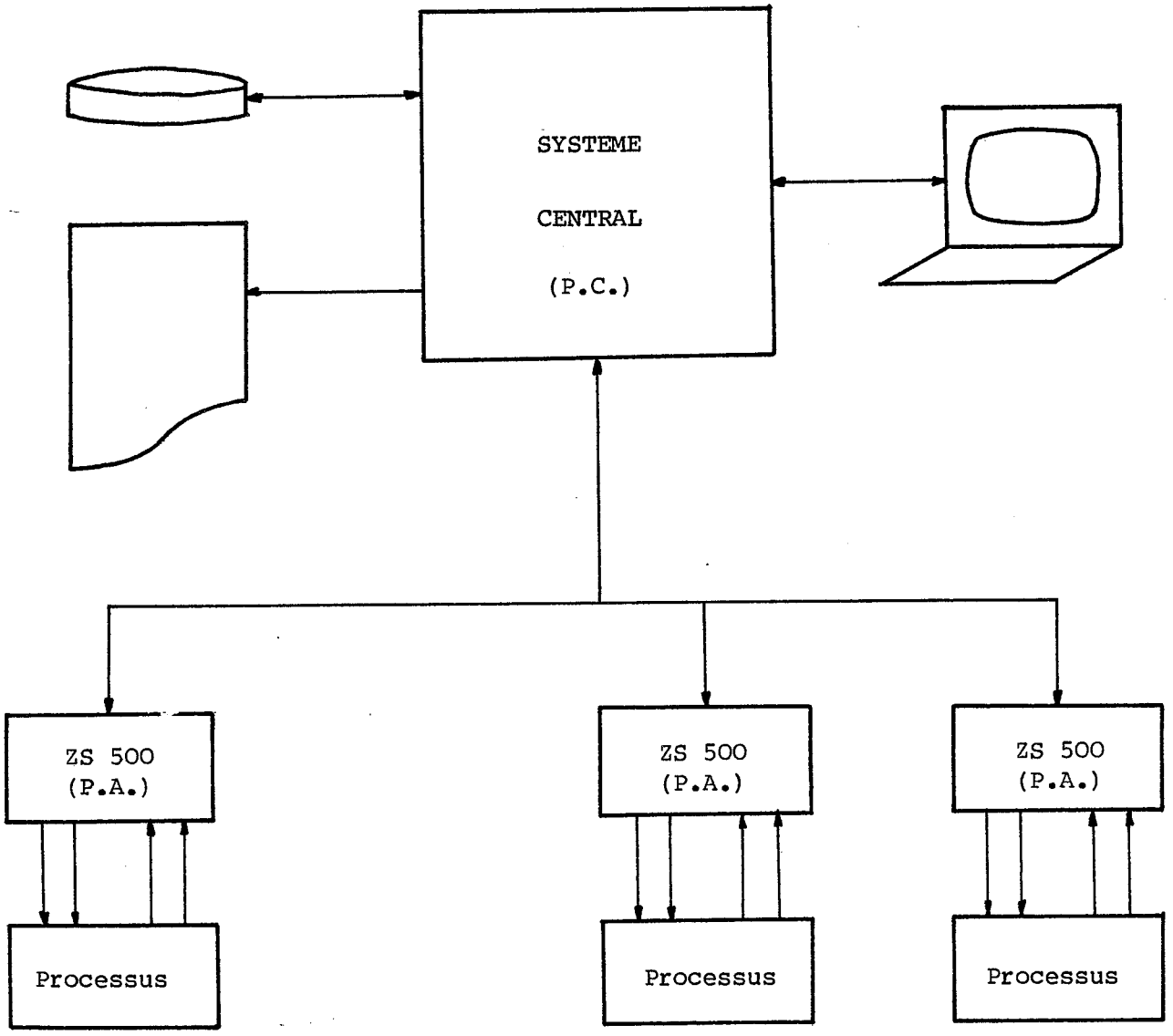


Figure 4.1. : Architecture générale du système de Télétransmission

4.2. STRUCTURE DE L'APPLICATION

L'utilisation maximale des possibilités de l'unité de traitement entraine une forte corrélation entre la structure des interfaces d'entrées-sorties et la structure du logiciel d'acquisition-restitution. En effet, les diverses interfaces n'ont pas toutes le même temps de réponse. D'autre part, le rythme des échanges avec le processus est lié aux besoins du système central.
Les problèmes que pose le traitement en parallèle de processus asynchrones ont été résolus dans le chapitre précédent.
Nous y avons en effet défini un produit logiciel, le moniteur temps réel (RTES85), auquel sont reliées les tâches de l'application.
Nous adopterons donc cette structure en découpant le système en autant de tâches qu'il y a de fonctions "d'acquisition-restitution". Chaque tâche traite un ensemble d'informations de même type.
Les fonctions à réaliser sont les suivantes :

- Fonctions d'acquisition :
  - . acquisition d'états tout ou rien par simple ou double scrutation
  - . acquisition d'états tout ou rien pour comptage
  - . acquisition d'états tout ou rien multiplexés
  - . acquisition de valeurs analogiques
- Fonctions de restitution :
  - . sortie d'états tout ou rien
  - . sortie de valeurs analogiques

Deux autres fonctions, bien que ne participant pas directement aux opérations d'acquisition et restitution, sont cependant essentielles pour le bon fonctionnement de l'application ; ce sont les fonctions d'initialisation et de transmission.

- Fonction d'initialisation
  Cette fonction n'est pas liée à un type de carte, mais à l'ensemble des cartes du système.
- Fonction de transmission
  . Emission d'informations du ZS 500 vers le poste central
  . Réception par le ZS 500 d'informations venant du poste central.

Ce découpage en fonctions est schématisé par la figure 4.2.

Les tâches réalisant ces différentes fonctions sont des modules standard. L'utilisateur devra constituer des tables de configuration (en mémoire REPROM) dans le but d'adapter le traitement de ces tâches au problème à résoudre. La structure de ces tables sera décrite au fur et à mesure de l'étude des diverses fonctions.

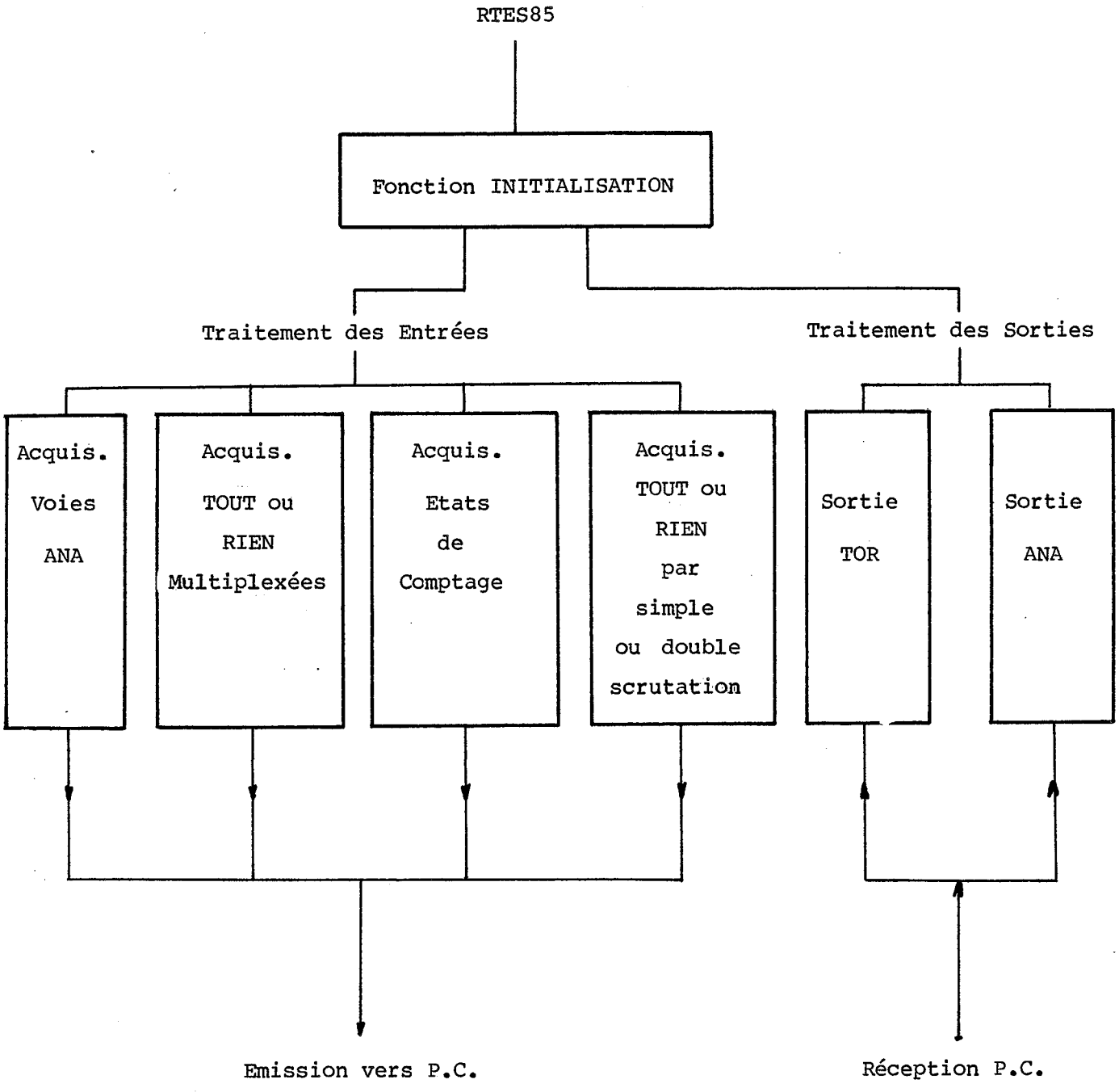


Figure 4.2. : Découpage de l'application en tâches

4.3. ANALYSE DES TACHES DE L'APPLICATION

4.3.1. Tâches d'acquisition

4.3.1.1. Acquisition d'états tout ou rien

1. Objet

Le rôle de cette tâche est de réaliser l'acquisition cyclique des informations TOR présentées par les cartes décrites dans le paragraphe suivant.
La période d'acquisition de chaque carte est définie par l'utilisateur dans la table de configuration, la période de base étant de 50 ms.
Cette acquisition peut être réalisée par simple ou double scrutation ; ce dernier procédé permettant d'augmenter la capacité de filtrage des changements d'états. Ce choix est également défini dans les tables de configuration (une pour la simple scrutation et une pour la double scrutation) dont la structure, pour cette tâche, est représentée figure 4.3.

2. Matériel utilisé

- Description

L'utilisateur dispose de deux types de cartes pour réaliser l'acquisition des états tout ou rien du processus :

. cartes 8 entrées 110 V alternatif
. cartes 16 entrées 24 V continu

Les cartes 8 et 16 entrées sont équipées d'un dispositif optoélectronique réalisant le découplage galvanique entre la logique d'acquisition et le processus.

Ces cartes sont constituées d'une interface avec le ZS 500 et de 8 ou 16 circuits découplés permettant l'acquisition des états tout ou rien.
L'état de chaque entrée est visualisé par diode électro-luminescente.
Chaque carte est dotée d'un dispositif de filtrage permettant de ne prendre en compte les informations que si elles sont stables à l'entrée pendant un temps suffisant.
Carte 8 entrées :
La constante de temps de filtrage est de 30 ms.
Le niveau des entrées TOR est 110 V alternatif ; cette alimentation est extérieure à la carte.
Carte 16 entrées :
La constante de temps de filtrage est de 1 ou 10 ms (option sur la carte).
Le niveau des entrées TOR est de 24 V continu ; cette alimentation est extérieure à la carte.

- Utilisation

Pour ces deux types de cartes, les informations d'entrée sont lues comme des octets aux adresses fixées par la position des cartes dans le chassis.
Une carte 16 entrées délivre les informations sur deux octets, une carte 8 entrées, sur un octet.
Ces deux types de cartes peuvent s'insérer dans n'importe quel emplacement banalisé du chassis ZS 500.

3. Principe

La tâche est découpée en deux grandes parties, la simple scrutation et la double scrutation.

- Simple scrutation :
  La tâche examine l'un après l'autre les compteurs de périodes associés à chaque carte, recopiés en mémoire vive lors de l'initialisation. Elle les décrémente d'une unité ; lorsqu'un compteur passe par zéro, il est rechargé avec sa valeur initiale et la tâche effectue l'acquisition des informations TOR présentées par la carte associée, à l'adresse inscrite dans la table de configuration.
  Si les informations lues dans le cycle courant sont différentes des informations lues au cours du cycle précédent, elles sont rangées dans la table des états.
  Si cela a été demandé par le poste central, le rang de la carte correspondante et la valeur du ou des octets associés sont rangés dans la table des changements d'états.

- Double scrutation :
  Elle fonctionne de façon similaire à la simple scrutation, mais permet de confirmer l'état d'une carte après deux scrutations.
  Elle travaille sur une table d'états précédents (E.P.), une table d'états confirmés (E.CF.) et sur l'état courant (E.C.).
  Si l'on symbolise une information de chaque table par le nom de la table, le changement d'état d'une information est donné par la formule :
  $CHGT = (E.C. \oplus E.CF.).(E.P. \oplus E.CF.)$
  Le nouvel état confirmé est alors :
  $nouvel\ E.CF. = CHGT \oplus E.CF.$

Celui-ci est rangé dans la table des états confirmés. L'information lue dans le cycle courant est rangée dans la table des états précédents.
Si cela a été demandé par le poste central, le rang de la carte correspondante et la valeur du ou des octets confirmés associés sont rangés dans la table des changements d'états.

L'algorithme succinct de la tâche est fourni figure 4.4.

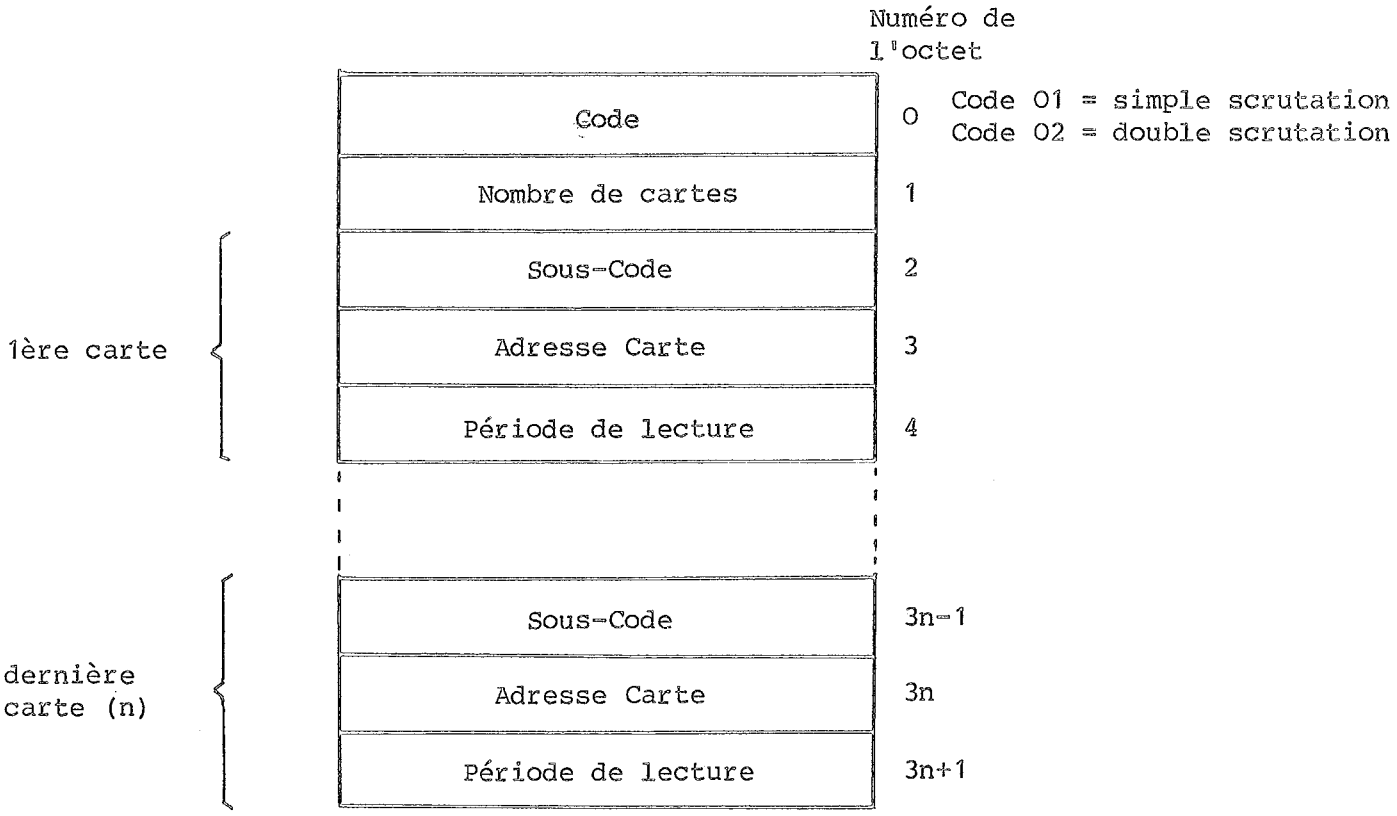
4. Résultats en sortie

La tâche met à la disposition du poste central (par l'intermédiaire du module d'émission) un ensemble de tables :

- Table d'états en simple et double scrutation (partagée avec la tâche d'acquisition des TOR multiplexées) TBTOR
- Table de changements d'états en double scrutation (partagée avec la tâche d'acquisition des TOR multiplexées) TBGTL
- Table de changements d'états en simple scrutation TBGTR

5. Ressources moniteur utilisées

Le rôle de la tâche est de remplir des tables d'états utilisables par d'autres tâches (tâche de transmission et d'acquisition d'état TOR multiplexés). Un accès simultané à ces tables par deux tâches peut nuire à leur intégrité. Pour éviter ce phénomène, avant toute utilisation de ces tables, la tâche testera et occupera un sémaphore associé à celles-ci.



Chaque case est codée sur 1 octet

Octet 0	Code significatif du type de traitement effectué
Octet 1	Nombre de cartes d'acquisition traitées
Octet 3n-1	Type de carte d'acquisition 1 = 8 entrées 2 = 16 entrées
Octet 3n	Adresse de la carte d'acquisition n (octet poids faible). L'octet poids fort de l'adresse, commun à toutes les cartes, n'est pas défini dans la table
Octet 3n+1	Période d'acquisition de la carte n, exprimée en nombre de périodes de 50 ms

Figure 4.3. : Structure de la table de configuration de la tâche d'acquisition Tout ou Rien

```
DEBUT Tâche acquisition TOR
   SI NCASSC  ≠ 0     ALORS     /Nb cartes simple scrutation ≠ 0
      POUR NOCART = 1  JUSQU'A    NCASSC   FAIRE
         SI CPTS(NOCART) = 0  ALORS /Période de lecture écoulée
            CPTS(NOCART) = valeur référence / Recharger le compteur
            Lire la carte (8 bits ou 16 bits) dans VALLUE
            SI VALLUE ≠ TBETOR (rang carte)  ALORS
               Test et positionnement sémaphore SETOR
               TBETOR (rang carte) = VALLUE
               SI Changements d'états demandés  ALORS
                  Ranger le rang de la carte et VALLUE dans la table TBGTR
               FINSI
               Libération sémaphore SETOR
            FINSI
         FINSI
      FINPOUR
   FINSI
   SI NCADSC  ≠ 0     ALORS     /Nb cartes double scrutation ≠ 0
      POUR NOCART = 1  JUSQU'A    NCADSC   FAIRE
         SI CPTD(NOCART) = 0  ALORS /Période de lecture écoulée
            CPTD(NOCART) = valeur référence /Recharger le compteur
            Lire la carte (8 bits ou 16 bits) dans VALLUE
            CHANGE =(VALLUE ⊕ TBETOR(rang carte)).(VALANT(rang carte) ⊕ TBETOR(rang carte))
            VALANT (rang carte) = VALLUE
            SI CHANGE ≠ 0   ALORS
               TBETOR (rang carte) = TBETOR (rang carte) ⊕ CHANGE
               Test et positionnement sémaphore SETOR
               SI Changements d'états demandés ALORS
                  Ranger le rang de la carte et TBETOR (rang carte) dans la table TBGTL
               FINSI
               Libération sémaphore SETOR
            FINSI
         FINSI
      FINPOUR
   FINSI
FINTACHE
```

Figure 4.4. : Algorithme succinct de la tâche d'entrée TOR

4.3.1.2. Acquisition d'états tout ou rien pour comptage

1. Objet

Le rôle de cette tâche est de comptabiliser le nombre de changements d'états intervenus sur un ensemble d'informations spécifiées par l'utilisateur.
En l'absence de cartes de comptage, cette fonction est réalisée à l'aide de cartes d'acquisition tout ou rien. Le type de matériel et la synchronisation des acquisitions, identiques dans les deux cas, auraient pu conduire à regrouper la fonction de comptage avec la tâche d'acquisition d'états tout ou rien. Cependant, le traitement étant notablement différent, la création d'une tâche spécifique améliore la modularité et la lisibilité.
La période d'acquisition de chaque carte est définie par l'utilisateur dans la table de configuration, la période de base étant de 50 ms.
Chaque carte permet, selon son type, de définir 16 ou 8 voies de comptage.

La structure de la table de configuration est représentée figure 4.5.

2. Matériel utilisé

Le matériel utilisé est du même type que celui décrit dans le paragraphe concernant la tâche d'acquisition des états tout ou rien en § 4.3.1.1.
Les informations d'entrée sont acquises de façon semblable.

3. Principe

A chaque carte d'entrée est associé un mot mémoire contenant son état antérieur ; à chaque voie de comptage est associé un compteur (un mot mémoire).
A chaque activation de la tâche, les compteurs de périodes d'acquisition de chaque carte sont décrémentés. Au passage par zéro d'un compteur, l'état de la carte associée est lu.
Les changements d'états de chaque voie sont détectés, et pour chaque voie de comptage définie, le compteur correspondant est incrémenté. En cas de dépassement de capacité d'un compteur, celui-ci est maintenu à sa valeur maximum.
Tous les changements d'états sont pris en compte (passage de 1 à 0 ou passage de 0 à 1). Il en ressort que le résultat du comptage du nombre d'occurences des phénomènes observés est égal au contenu des compteurs divisé par deux.
La capacité maximale d'un compteur est de 15 bits (mot de 16 bits divisé par 2) soit 32767.
L'utilisateur spécifie les voies de comptage qu'il utilise sur une carte à l'aide d'un masque (8 ou 16 bits selon le type de cartes) fourni dans la table de configuration.

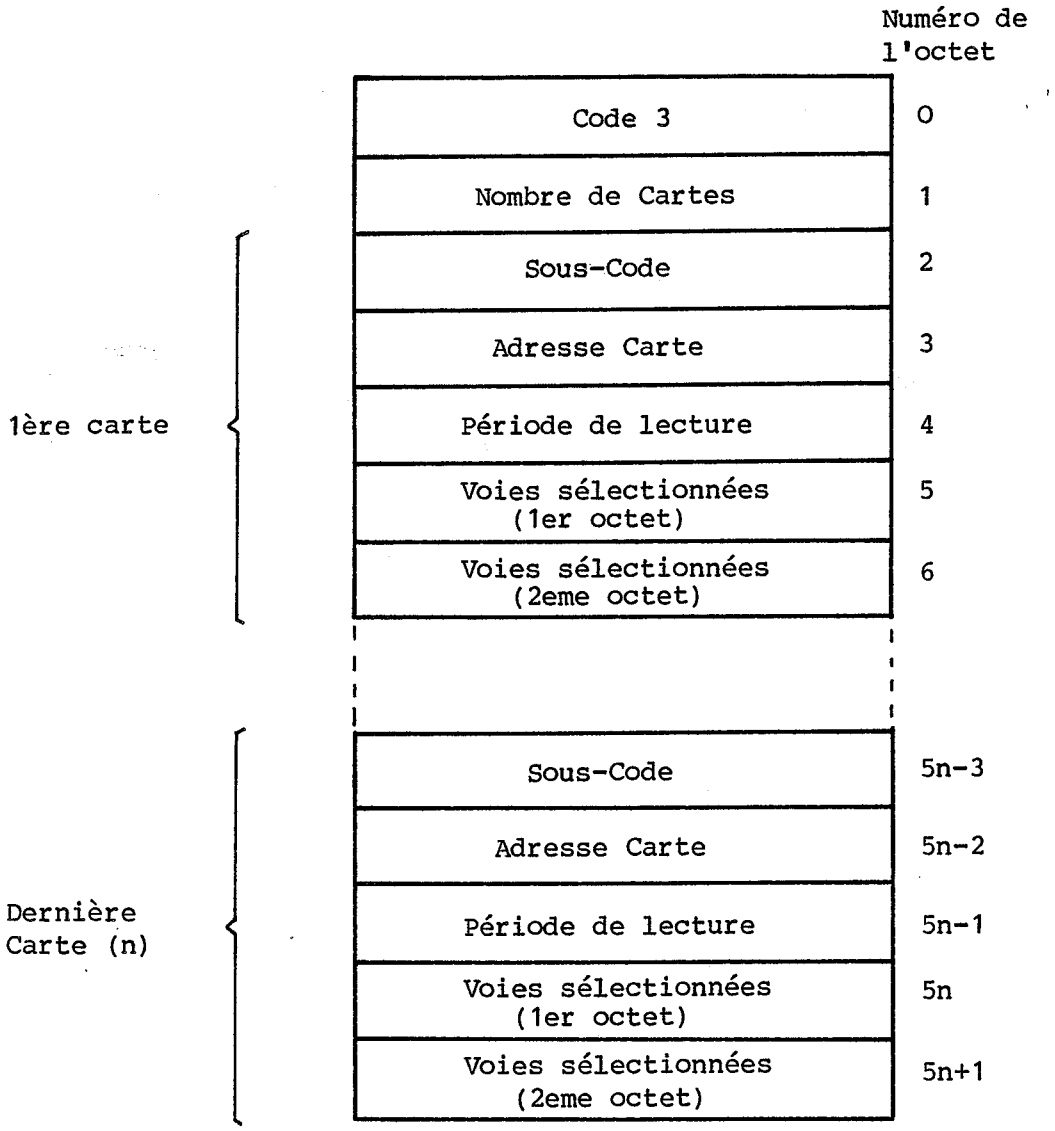
Un algorithme succinct de la tâche de comptage est fourni figure 4.6.

4. Résultats en sortie

La tâche met à la disposition du poste central (par l'intermédiaire du module d'émission) la table TBCPT contenant les valeurs des compteurs associés à chaque voie de comptage.

5. Ressources moniteur utilisées

Afin d'éviter tout conflit avec le module d'émission lors des accès à la table des compteurs, la tâche demande l'occupation d'un sémaphore associé à cette table pendant les opérations sur les compteurs.



Octet 0 — Code significatif du type de traitement effectué

Octet 1 — Nombre de cartes d'acquisition traitées

Octet 5n-3 — Type de carte d'acquisition 1 = 8 entrées
2 = 16 entrées

Octet 5n-2 — Adresse de la carte d'acquisition n (cf. fig. 4.3.)

Octet 5n-1 — Période d'acquisition des voies de la carte n (nombre de période de 50 ms)

Octets 5n et 5n+1 — Indicateurs des voies de comptage utilisées (1 bit par compteur). Il y a 2 octets quel que soit le type de carte (8 ou 16 bits)

Figure 4.5. : Table de configuration de la tâche de comptage

```
DEBUT Tâche Comptage
|  POUR NOCART = 1  JUSQU'A NCACPT  FAIRE  /Nbre de cartes
|  |  SI CPTC(NOCART) = 0  ALORS  /Période de lecture écoulée
|  |  |  CPTC(NOCART) = Valeur référence  /Recharger le compteur
|  |  |  Lire la carte (8 bits ou 16 bits) dans VALLUE
|  |  |  CHANGE = VALLUE ⊕ VALANT (NOCART)
|  |  |  VALANT(NOCART) = VALLUE
|  |  |  K = 8              /si carte 8 entrées
|  |  |  SI carte 16 entrées  ALORS
|  |  |  |  K = 16
|  |  |  FINSI
|  |  |  POUR J = 1  JUSQU'A  K  FAIRE
|  |  |  |  SI bit n° J de CHANGE ≠ 0  ALORS
|  |  |  |  |  Incrémenter le compteur correspondant dans la table TBCPT
|  |  |  |  FINSI
|  |  |  FINPOUR
|  |  FINSI
|  FINPOUR
FINTACHE
```

Figure 4.6. : Algorithme succinct de la tâche de comptage

4.3.1.3. Acquisition d'états tout ou rien multiplexés

1. Objet

Cette tâche effectue l'acquisition des états tout ou rien multiplexés.
La période d'activation de cette tâche est de 4 secondes. Au cours de cette période, les informations sont acquises en totalité.
Les informations sont acquises par simple scrutation avec détection de changements d'états.

La structure de la table de configuration est représentée figure 4.8.

2. Matériel utilisé

- Description

  Le système d'acquisition d'états multiplexés est constitué de 2 cartes à 16 sorties tout ou rien (24 V continu) reliées à 1 ou 2 cartes 16 entrées tout ou rien (24 V continu) selon le schéma représenté à la figure 4.7. (1 seule carte d'entrée).

  L'interface entre le processus et les cartes d'entrée est réalisée par un ensemble d'amplificateurs 3 états implantés sur des cartes dans un chassis séparé.
  Les liaisons entre ce châssis et celui du ZS 500 sont les 32 fils des sorties TOR et les 16 ou 32 fils du bus d'entrée.

  Cette configuration permet d'acquérir l'état de 512 ou 1024 états tout ou rien.

- Utilisation

Contrairement aux cas précédents (acquisition tout ou rien et comptage), les informations ne sont pas lues comme des octets à des adresses fixées par la position des cartes. En se plaçant dans le cas de 2 cartes de sortie et 1 carte d'entrée, les 2 cartes de sortie servent à multiplexer l'acquisition des 512 états (par groupe de 16) tout ou rien sur la carte d'entrée. La procédure d'acquisition de 16 ou 32 états consiste tout d'abord à valider une sortie parmi les 32 (2 cartes de sortie) puis à lire 1 ou 2 cartes d'entrée (selon le cas).

3. Principe

L'acquisition des états tout ou rien est réalisée conformément à la procédure indiquée au paragraphe précédent :

. Mise à "1" d'une sortie parmi les 32 que comportent les deux cartes
. Attente de 100 ms
. Lecture de la ou des deux cartes d'entrée et rangement du résultat dans la table (TBETOR) des états tout ou rien après comparaison à l'état antérieur. En cas de différence, le rang de la voie multiplexée (considérée comme une carte d'acquisition) et la valeur lue sont rangés dans la table des changements d'états (TBGTL) si le poste central l'a demandé.

Cette séquence est exécutée 32 fois, chaque cycle correspondant à l'activation d'une sortie et à la lecture de 16 ou 32 états (1 ou 2 cartes d'entrée).

L'attente de 100 ms est rendue nécessaire par le temps que mettent les sorties à se stabiliser. Il en résulte un temps d'exécution de la tâche de l'ordre de 3,2 secondes, d'où une période d'activation de la tâche de 4 secondes.

L'algorithme succinct de la tâche est fourni figure 4.9.

4. Résultats en sortie

La tâche met à la disposition du poste central (par l'intermédiaire du module d'émission) deux tables :
- table d'états tout ou rien TBETOR
- table de changements d'états TBGTL

Ces deux tables sont partagées avec la tâche d'acquisition des états tout ou rien (pour leur remplissage).

5. Ressources moniteur utilisées

La tâche demande l'occupation d'un sémaphore lors des accès aux tables d'états et de changements d'états, pour éviter tout conflit avec le module d'émission et avec la tâche d'acquisition des états tout ou rien.
D'autre part, l'attente de 100 ms est générée par appel à la requête de demande de délai permettant à la tâche de suspendre son exécution pendant le temps spécifié.

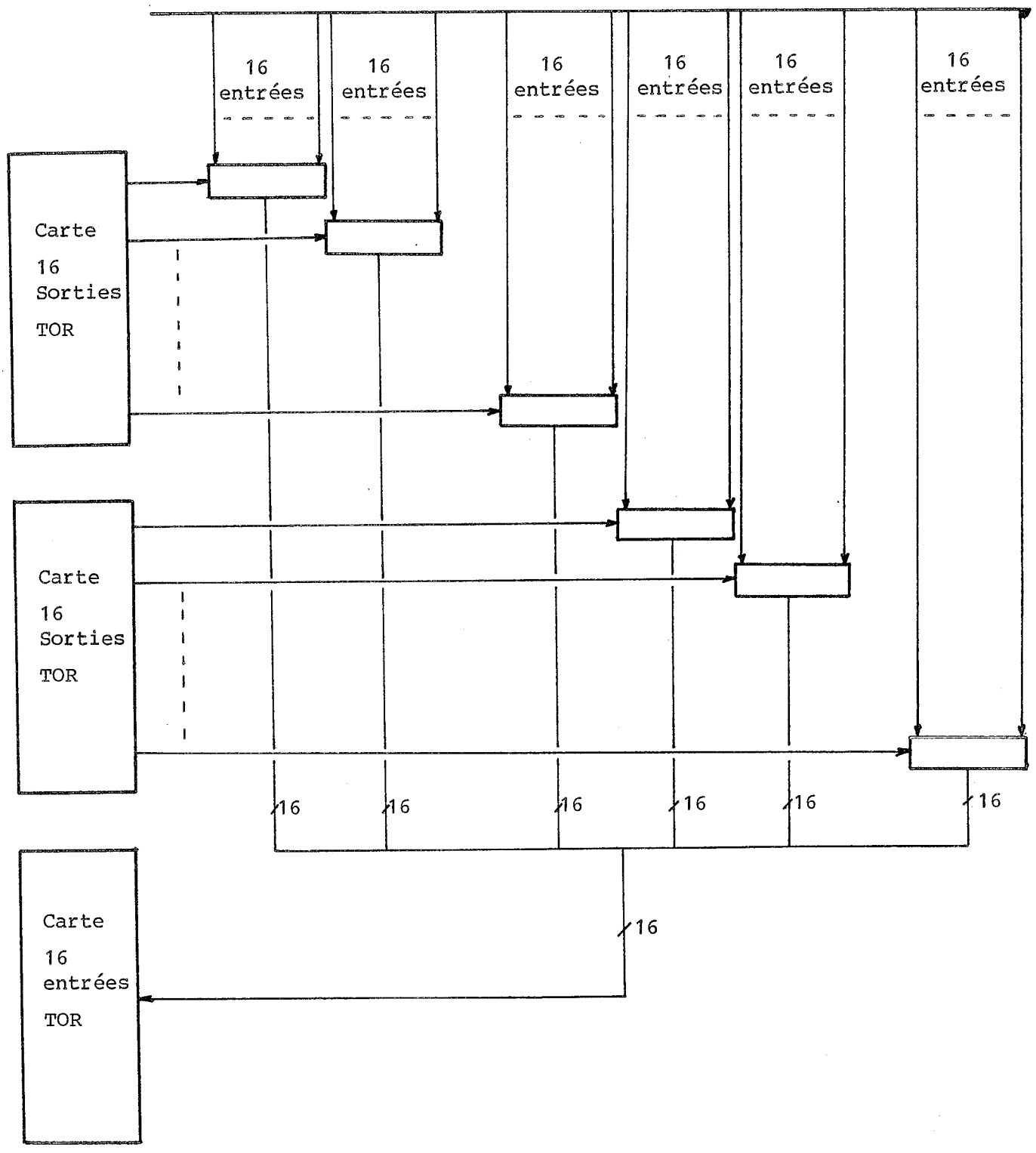


Figure 4.7. : Schéma de l'interface d'acquisition des états TOR multiplexés

Code 4	0
Nombre de cartes d'entrée	1
Adresse carte d'entrée 1	2
Adresse carte de sortie 1	3
Adresse carte de sortie 2	4
Adresse carte d'entrée 2	5

Octet 0 Code significatif du type de traitement effectué

Octet 1 Prend pour valeur 1 ou 2 selon le nombre de cartes d'entrée

Octet 2 Adresse de la carte d'entrée TOR n° 1 (cf. fig. 4.3.)

Octets 3 et 4 Adresses des cartes de sortie TOR 1 et 2 (cf. fig. 4.3.)

Octet 5 Adresse de la carte d'entrée TOR n° 2 si l'octet 1 a pour valeur 2 (cf. fig. 4.3.)

Figure 4.8. : Structure de la table de configuration de la tâche d'acquisition Tout ou Rien multiplexés

```
DEBUT Tâche entrées TOR multiplexées
|  POUR CARSOR = 1  JUSQU'A  2 FAIRE /Cartes de sortie
|  |  POUR VOISOR = 1  JUSQU'A  16 FAIRE /16 bits par carte
|  |  |  Ecrire le bit n° VOISOR sur la carte de sortie n° CARSOR
|  |  |  Attente 100 ms
|  |  |  POUR I = 1  JUSQU'A NBCARE  FAIRE  /Nbre de cartes d'entrée
|  |  |  |  Lire la carte d'entrée TOR n° 1 dans VALLUE
|  |  |  |  SI VALLUE ≠ TBETOR (rang carte)  ALORS
|  |  |  |  |  Test et positionnement sémaphore SETOR
|  |  |  |  |  TBETOR (rang carte) = VALLUE
|  |  |  |  |  SI changements d'états demandés  ALORS
|  |  |  |  |  |  Ranger le rang de la carte et VALLUE dans la table TBGTL
|  |  |  |  |  FINSI
|  |  |  |  |  Libération sémaphore SETOR
|  |  |  |  FINSI
|  |  |  FINPOUR
|  |  FINPOUR
|  FINPOUR
FINTACHE
```

Figure 4.9. : Algorithme succinct de la tâche d'acquisition des TOR multiplexées

4.3.1.4. Acquisition de valeurs analogiques

1. Objet

Le rôle de cette tâche est de réaliser l'acquisition cyclique des informations analogiques présentées par les cartes décrites ci-dessous.
Elle effectue les corrections de dérive et de température, ainsi que les éventuels traitements sur les valeurs binaires (linéarisation, racine carrée..) et les filtrages (non définis).
Ces divers traitements ainsi que les types de cartes sont spécifiés dans la table de configuration dont la structure est représentée figure 4.10.

2. Matériel utilisé

- Description

L'utilisateur dispose de deux types de cartes pour réaliser l'acquisition des valeurs analogiques fournies par le processus :
. cartes 8 voies haut niveau
. cartes 8 voies bas niveau
Sur ces deux types de cartes, un dispositif opto-électronique réalise le découplage galvanique entre la logique d'acquisition et le processus. Les 8 voies de mesure de chaque carte ont un point commun.

. Option haut niveau :
Le niveau maximum d'entrée est de $\pm$ 10,24 V (gain = 1). L'utilisateur a la possibilité de mesurer la dérive.

. Option bas niveau :
Le niveau maximum d'entrée est de ± 20,48 mV (gain = 500). L'utilisateur a la possibilité de mesurer la dérive de la chaine d'acquisition et la température au niveau du bornier (compensation de soudure froide).

Dans les deux cas, la carte fournit la tension de référence pour l'alimentation des 8 ponts de mesure.
Le temps de stabilisation d'une entrée après sélection est de 2 millisecondes.
Le temps de connexion est de 40 microsecondes.
La constante de temps de filtrage est de 20 millisecondes.

- Utilisation

La procédure d'acquisition d'une voie analogique est la suivante :
. Mesure du zéro (dérive)
. Mesure de la température au niveau du bornier (option bas niveau)
. Remise à zéro sélection précédente
. Sélection de la voie
. Demande de connexion
. Acquisition de la valeur sur 12 bits

Ces deux types de cartes peuvent s'implanter à n'importe quel emplacement banalisé du ZS 500.

3. Principe

La tâche est activée toutes les 100 millisecondes.
Pour chaque carte décrite dans la table de configuration, elle lit la correction de dérive et la température du bornier (bas niveau).
Pour chaque voie d'une carte, elle réalise alors les opérations suivantes :

- Lancement de la conversion pour la voie
- Lecture de la valeur numérisée
- Correction en dérive
- Correction en température (si bas niveau)
- Traitement spécifique sur la valeur corrigée
- Sur demande du poste central, la tâche calcule pour chaque voie analogique la différence entre la valeur courante et la valeur acquise lors du cycle d'acquisition précédent. La valeur absolue du résultat est comparée à l'écart spécifié dans la table de configuration. En cas de dépassement, un indicateur de changement d'état est rangé dans la table TBIANA, exploitée par le module de transmission.
- Rangement dans la table des valeurs analogiques (TBEANA)

Les temporisations après la sélection d'une voie (2 ms) et après le lancement de conversion (40 µs) sont réalisés en mode programmé. Elles sont en effet trop courtes pour être gérées par le moniteur.
Les temps maximum sont cependant contrôlés pour éviter tout blocage, en particulier dans le cas du test de fin de conversion.

L'utilisateur a la possibilité de corriger la valeur numérisée de quatre façons différentes, selon le type de capteurs :
- pas de correction
- linéarisation
- linéarisation par segments
- racine carrée

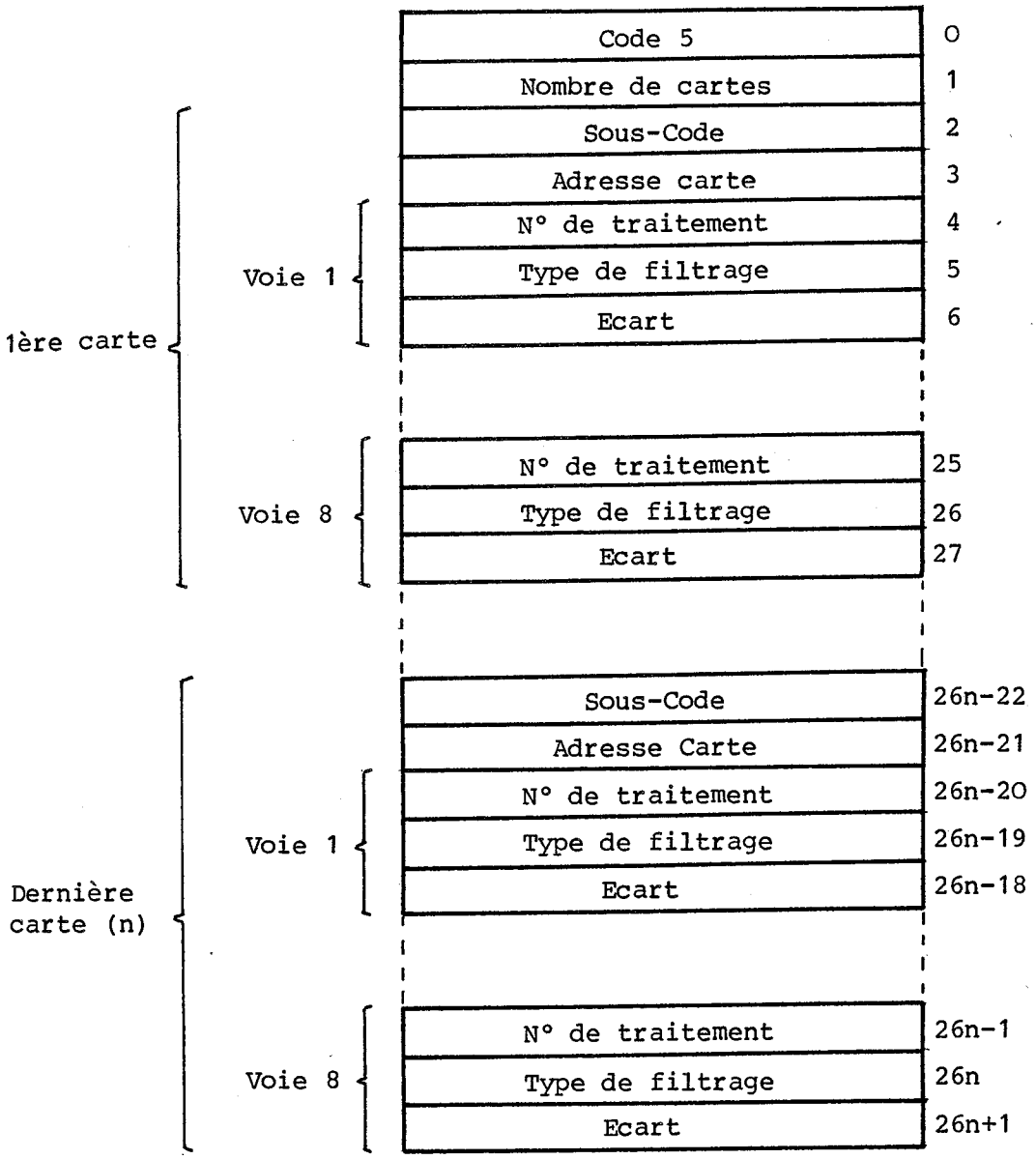
Un algorithme succinct de la tâche est fourni figure 4.11.

4. Résultats en sortie

La tâche met à la disposition du poste central, par l'intermédiaire du module d'émission, une table contenant les valeurs analogiques (TBEANA) numérisées et sur lesquelles ont été effectuées les corrections liées au système d'acquisition et aux capteurs.
Elle fournit également une table d'indicateurs (TBIANA) permettant de générer la table de changements d'états des valeurs analogiques.

5. Ressources moniteur utilisées

La tâche utilise un sémaphore pour se prémunir contre les accès simultanés à la table des valeurs analogiques lors de son utilisation par le module de transmission.



Octet 0 Code du traitement à effectuer

Octet 1 Nombre de cartes analogiques traitées

Octet 26n-22 Type de carte analogique 1 = carte haut niveau
2 = carte bas niveau

Octet 26n-21 Adresse de la carte analogique (cf. fig. 4.3.)

Octet 26n-20 Traitement à effectuer sur la valeur numérique :
0 pas de traitement
1 linéarisation
2 linéarisation par segment
3 racine carrée
4 non défini
.
.
F

Octet 26n-19 Type de filtrage : non défini

Octet 26n-18 Ecart de la valeur analogique corrigée pour détection de changement d'état

Figure 4.10. : Table de configuration de la tâche d'acquisition de valeurs analogiques

```
DEBUT Tâche acquisition valeurs analogiques
|  POUR NOCART = 1  JUSQU'A  NCEANA  FAIRE   /Nbre de cartes analogiques
|  |  Lecture valeur de la dérive
|  |  SI Carte de type "bas niveau"  ALORS
|  |  |  Lecture de la température de jonction
|  |  FINSI
|  |  POUR NOVOIE = 1  JUSQU'A 8  ALORS
|  |  |  SI Voie NOVOIE utilisée  ALORS
|  |  |  |  Sélectionner la voie NOVOIE sur la carte NOCART
|  |  |  |  Attendre 2 millisecondes
|  |  |  |  Lancer la conversion sur la carte NOCART
|  |  |  |  Attendre 40 microsecondes
|  |  |  |  Lire la carte NOCART
|  |  |  |  Corriger en température (si bas niveau)
|  |  |  |  Corriger en dérive
|  |  |  |  Lancer le traitement approprié au type de capteur
|  |  |  |  Test et positionnement sémaphore SEANA
|  |  |  |  SI Changements d'états demandés  ALORS
|  |  |  |  |  SI  Valeur précédente - Valeur courante > Ecart ALORS
|  |  |  |  |  |  Ranger indicateur dans TBIANA
|  |  |  |  |  FINSI
|  |  |  |  FINSI
|  |  |  |  Ranger la valeur analogique dans TBEANA
|  |  |  |  Libération sémaphore SEANA
|  |  |  FINSI
|  |  FINPOUR
|  FINPOUR
FINTACHE
```

Figure 4.11. : Algorithme succinct de la tâche d'acquisition des valeurs analogiques

4.3.2. Tâches de restitution (sortie)

4.3.2.1. Sortie d'états tout ou rien

1. Objet

Le rôle de cette tâche est de positionner les sorties à destination des organes de commande du processus. Les états de commande des sorties sont transmis à la tâche par l'organe central via la liaison série asynchrone.
Ces états sont affectés à chaque carte de sortie à l'aide de la table de configuration définie par l'utilisateur. Celle-ci est décrite à la figure 4.12.

2. Matériel utilisé

- Description

L'utilisateur dispose de deux types de cartes pour transmettre les ordres de commande :
. cartes 8 sorties 110 ou 220 V alternatif
. cartes 16 sorties 24 V continu

. La carte 8 sorties est constituée d'une interface avec le bus ZS 500 et de 8 circuits découplés permettant la commande d'organes de puissance en alternatif. La tension de commande des organes est de 110 V ou 220 V selon l'alimentation externe fournie. Le courant maximum est de 1 ampère par sortie.
Les ordres donnés par le microprocesseur sont visualisés sur 8 diodes électroluminescentes situées à l'avant de la carte.
Un système de protection remet à zéro automatiquement les ordres toutes les 100 millisecondes. Ceci protège le processus contre un maintien d'ordre intempestif en cas d'arrêt du système.

. La carte 16 sorties est constituée d'une interface avec le bus ZS 500 et de 16 circuits découplés permettant la commande d'organe de puissance en continu.
L'alimentation 24 V de puissance est externe à la carte ; toutes les sorties ont pour point commun le zéro de cette tension. Le courant maximum de commande est de 1 ampère par sortie.
Les ordres donnés par le microprocesseur sont visualisés sur 16 diodes électroluminescentes situées à l'avant de la carte.

- Utilisation

. Pour les deux types de cartes, la commande des sorties s'effectue par une opération d'écriture à l'adresse de la carte. Dans le cas d'une carte 8 sorties, une opération de lecture à la même adresse permet de connaitre l'état des commandes.
. Ces deux types de cartes peuvent s'insérer dans n'importe quel emplacement banalisé du chassis ZS 500.

3. Principe

La tâche traite les deux types de cartes de sortie : les unes (8 sorties) nécessitent un rafraichissement périodique, les autres (16 sorties) n'en ont pas besoin. Afin de simplifier le traitement, toutes les cartes seront commandées de la même façon ; la tâche est activée toutes les 100 millisecondes. Elle écrit sur toutes les cartes de sortie les informations provenant du poste central.

Les cartes 8 sorties sont considérées, pour les opérations d'écriture, comme les poids faibles des cartes 16 sorties.

La fonction de relecture des cartes 8 sorties n'est pas traitée.

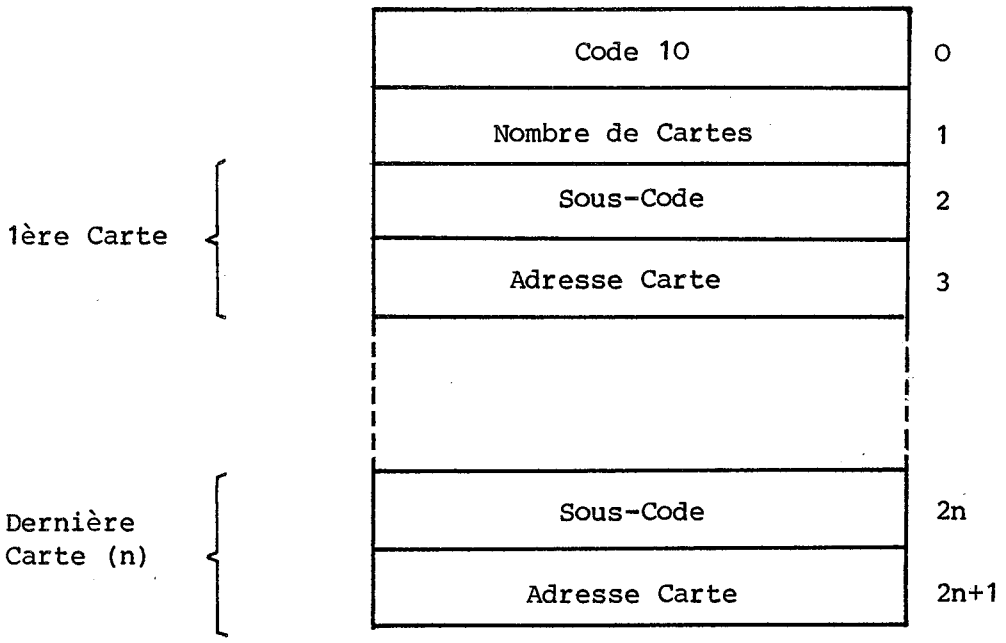
L'algorithme succinct de cette tâche est fourni figure 4.13.

4. Données en entrée

La tâche reçoit du poste central par l'intermédiaire du module émission-réception la table d'états des sorties TBVTOR.

5. Ressources moniteur utilisées

Pour éviter les accès simultanés à la table d'états des sorties, un sémaphore est utilisé par les tâches qui y accèdent.



Octet 0	Code significatif du type de traitement effectué
Octet 1	Nombre de cartes de sortie TOR traitées
Octet 2n	Type de carte de sortie TOR 1 = carte 8 sorties 2 = carte 16 sorties
Octet 2n+1	Adresse de la carte de sortie (cf. fig. 4.3.)

Figure 4.12. : Table de configuration de la tâche de sortie d'états Tout ou Rien

```
DEBUT Tâche sortie TOR
|  POUR NOCART = 1  JUSQU'A NCASTO  FAIRE  /Nbre de cartes de sortie
|  |  Test et positionnement sémaphore SESOR
|  |  ADCART = adresse de la carte NOCART
|  |  Prélever la valeur à écrire dans TBVTOR (2 * NOCART)
|  |  Ecrire la valeur à l'adresse ADCART
|  |  SI carte 16 sorties  ALORS
|  |  |  Prélever la valeur à écrire dans TBUTOR (2 * NOCART+1)
|  |  |  Ecrire la valeur à l'adresse ADCART+1
|  |  FINSI
|  |  Libération sémaphore SESOR
|  FINPOUR
FINTACHE
```

Figure 4.13. : Algorithme succinct de la tâche de sortie d'états Tout ou Rien

4.3.2.2. Sortie de valeurs analogiques

1. Objet

Le rôle de cette tâche est de commander les sorties des cartes analogiques reliées au processus à contrôler. Les valeurs analogiques à transmettre sont envoyées à la tâche par l'organe central via la liaison série asynchrone. Elles doivent avoir le format requis par les cartes analogiques (voir description du matériel utilisé). L'utilisation de chaque voie de chaque carte est définie dans la table de configuration dont la structure est décrite figure 4.14.

2. Matériel utilisé

- Description

Un seul type de carte est actuellement prévu :

. Carte 8 sorties + 10 V, - 10 V

Cette carte est constituée d'une interface avec le bus ZS 500 permettant la sélection des voies et l'indication de la valeur de la tension et d'une interface avec le processus. Celle-ci est constituée d'un convertisseur numérique analogique, dont la sortie est aiguillée par programme vers une des 8 mémoires analogiques.

Les deux interfaces sont isolées entre elles par photocoupleurs.

La tension de sortie varie entre - 10,24 V et + 10,24 V pour une valeur numérique variant de 000 à FFF (héxadécimal).

Le courant maximum est de 10 mA par voie.

- Utilisation

A la mise sous tension, il est nécessaire d'effectuer une mise à zéro de la carte pour éviter que les sorties se trouvent à des potentiels indésirables pour l'utilisateur.
Après cette phase, la procédure d'utilisation est la suivante :

- Désélection de toutes les voies
- Temporisation 50 microsecondes
- Envoi de la valeur au convertisseur
- Temporisation de 1 milliseconde
- Sélection de la voie
- Temporisation 2 millisecondes

Les sorties doivent être rafraichies périodiquement toutes les 100 millisecondes maximum.

Ce type de carte peut s'implanter à n'importe quel emplacement banalisé du ZS 500.

3. Principe

Nous avons vu, dans le paragraphe concernant le matériel, que le temps de traitement d'une voie est de 3 ms.
En conséquence, le temps nécessaire pour traiter une carte est de l'ordre de 24 ms.
Afin de ne pas multiplier ce temps par le nombre de cartes de sorties analogiques du système, nous travaillerons en parallèle sur les voies de même numéro de chaque carte.
Les temporisations sont traitées en mode programmé.

Les cartes devant être rafraichies, la tâche sera activée toutes les 100 ms. Au cours de cette période, les 8 voies de chaque carte auront donc pu être traitées de la façon indiquée précédemment.

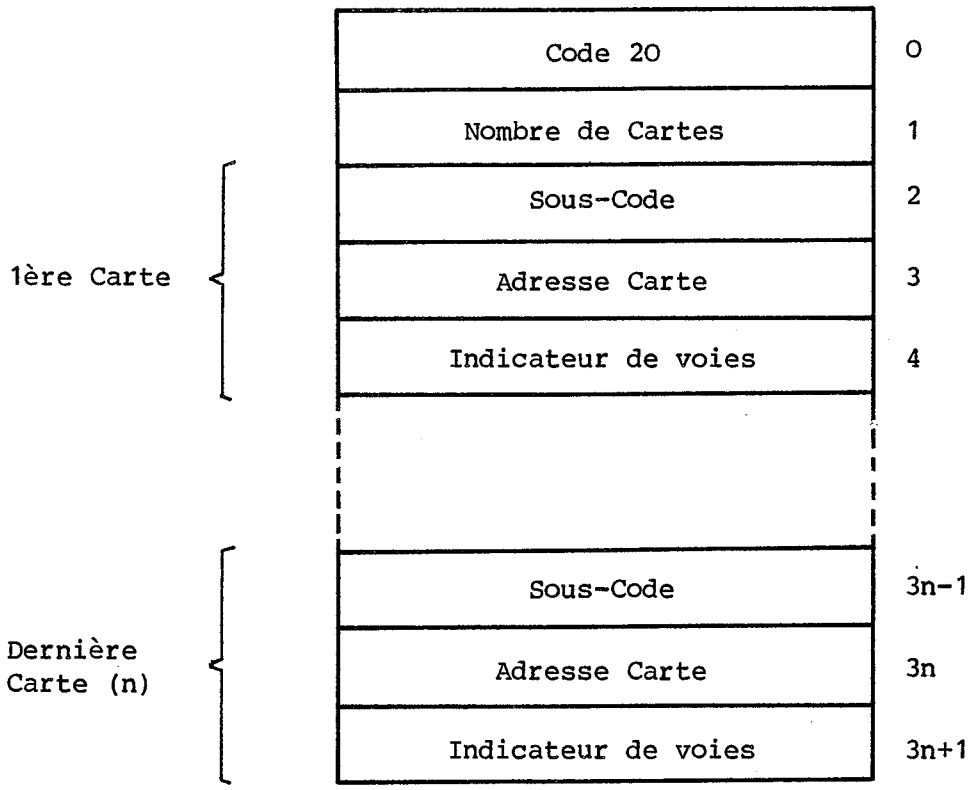
Un algorithme succinct de cette tâche est fourni figure 4.15.

4. Données en entrée

La tâche reçoit du poste central, par l'intermédiaire du module de transmission les valeurs analogiques destinées à commander le processus. Celles-ci sont fournies dans la table TBVANA.

5. Ressources moniteur utilisées

La tâche utilise un sémaphore pour se prémunir contre les accès simultanés à la table des valeurs analogiques.



Octet 0	Code significatif du type de traitement effectué
Octet 1	Nombre de cartes de sortie analogique
Octet 3n-1	Type de carte de sortie analogique 1 = carte 8 voies 2 = carte 2 voies
Octet 3n	Adresse de la carte de sortie analogique (cf. fig. 4.3.)
Octet 3n+1	Octet indiquant les voies utilisées (1 bit par voie)

Figure 4.14. : Table de configuration de la tâche de sortie analogique

```
DEBUT Tâche sortie analogique
|  POUR NOVOIE = 1  JUSQU'A  8  FAIRE
|  |  POUR  NOCART = 1  JUSQU'A NCSANA  FAIRE
|  |  |  Désélectionner toutes les voies de la carte NOCART
|  |  FINPOUR
|  |  Attente 50 microsecondes
|  |  POUR NOCART = 1  JUSQU'A NCSANA  FAIRE  /Conversion voies
|  |  |  SI voie NOVOIE de la carte NOCART utilisée  ALORS
|  |  |  |  Test et positionnement sémaphore SESAN
|  |  |  |  Prélever la valeur dans TBVANA
|  |  |  |  Libération sémaphore SESAN
|  |  |  |  Sortir la valeur sur la carte NOCART
|  |  |  FINSI
|  |  FINPOUR
|  |  Attente 1 milliseconde
|  |  POUR NOCART = 1  JUSQU'A NCSANA  FAIRE /Sélection voies
|  |  |  SI voie NOVOIE de la carte NOCART utilisée  ALORS
|  |  |  |  Sélectionner la voie NOVOIE de la carte NOCART
|  |  |  FINSI
|  |  FINPOUR
|  |  Attente 2 millisecondes
|  FINPOUR
FINTACHE
```

Figure 4.15. : Algorithme succinct de la tâche de sortie analogique

4.3.3. Tâches d'initialisation

1. Objet

Le rôle de cette tâche est d'initialiser tous les éléments nécessaires au bon déroulement de l'application.
Ceci concerne essentiellement les ressources du système, telles que :
- la mémoire morte
- la mémoire vive
- les cartes de sortie
- les tâches de l'application

2. Matériel utilisé

Cette tâche met en oeuvre les cartes de sortie analogique. Celles-ci ont été décrites dans le paragraphe 4.3.2.2.

3. Principe

Le signal RESET généré à la mise sous tension du ZS 500 lance le moniteur RTES85. Celui-ci, après initialisation de ses propres tables et de la liaison série asynchrone, active la première tâche de l'application.

Cette tâche réalise les actions suivantes :
- Mémoire morte
  . Contrôle des tables de configuration définies par l'utilisateur.
  La tâche vérifie la validité des codes des traitements demandés, du nombre de cartes, des sous-codes (type de carte).
  Elle contrôle également que les adresses des cartes sont dans la plage autorisée.

- Mémoire vive

  . Remise à zéro des tables d'acquisition et restitution TOR, comptage et analogiques
  . Remise à zéro des tables d'émission et réception utilisées par le module de transmission
  . Remise à zéro des diverses variables de l'application
  . Elaboration des tables de périodes d'acquisition, à partir des tables de configuration, pour les tâches d'acquisition tout ou rien et comptage
  . Elaboration des tables de rangs pour les tâches d'acquisition tout ou rien, tout ou rien multiplexées, comptage et analogiques et pour la tâche de sortie analogique.

  Les tables de rangs permettent d'établir la correspondance entre une information à acquérir ou à restituer et sa position dans les tables où elle est rangée ou prélevée. Elles permettent également au poste central et au poste auxiliaire (ZS 500) de référencer les informations de façon identique.

- Cartes de sortie

  Les cartes de sortie tout ou rien ne sont pas traitées dans cette tâche ; en effet, lors de la remise sous tension, elles reçoivent le signal RESET qui remet à zéro les sorties.
  Par contre, les mémoires analogiques des cartes de sortie analogique ne sont pas affectées par ce signal. En conséquence, pour éviter tout niveau indésirable, la tâche effectue une mise à zéro de toutes les voies des cartes analogiques.

- Tâches de l'application

  En fonction des codes qui ont été rencontrés dans les tables de configuration lors du contrôle de celles-ci, les tâches concernées par ces codes sont activées. Elles le sont en fonction d'un calcul de priorité effectué dynamiquement dans cette tâche.

Après avoir réalisé l'ensemble de ces actions, la tâche se poursuit dans le module de transmission et se met en attente d'un message en provenance du poste central.
Ce module sera analysé dans le paragraphe suivant.

4. Tables traitées

Ensemble des tables traitées dans les diverses tâches d'acquisition et de restitution (tables d'états et de valeurs), ainsi que les tables de rangs et de périodes.

5. Ressources moniteur utilisées

Les tâches sont lancées périodiquement à l'aide de la requête d'activation synchrone.

4.3.4. Module de transmission

Cette partie de l'application ne constitue pas elle-même une tâche ; elle est le dernier module de la tâche d'initialisation. Il en résulte que cette dernière n'est jamais désactivée ; cette façon de procéder permet d'économiser une tâche.

1. Objet

Ce module assure les échanges entre le poste central (P.C.) et le système ZS 500 qui joue le rôle de poste auxiliaire (P.A.). Il traite les messages reçus du P.C. et renvoie à celui-ci les informations acquises par les tâches de l'application.

2. Matériel utilisé

La liaison P.C. - P.A. est réalisée en mode asynchrone par deux paires téléphoniques. Elle est prévue pour pouvoir fonctionner en "multipoint" (plusieurs postes auxiliaires sur la même ligne) ou "point à point" (un seul poste auxiliaire par ligne).
La vitesse de transmission et les caractéristiques des informations (format) sont à définir pour chaque application lors de la génération du moniteur.

3. Principe

Les échanges s'effectuent en mode *maître-esclave*, le poste central étant maitre en permanence. Ils se font par interrogation ou envoi d'ordre par le poste central au poste auxiliaire.

Un échange se compose d'un message d'émission (P.C. vers P.A.) et d'un message de réponse du P.A. vers le P.C. Ces messages sont décrits au paragraphe 4.

Le module analyse le message reçu et effectue les actions demandées :
- transmission d'une table d'états ou de valeurs au P.C.
- écriture d'un bit dans une table
- écriture d'un mot dans une table
- écriture d'une table
- initialisation des tables d'acquisition
- transmission d'une table de changements d'états

. Transmission d'une table d'états ou de valeurs au P.C.
Le contenu de la table d'états tout ou rien (TBETOR), de comptage (TBCPT), ou de valeurs analogiques (PBANA), selon le cas, est envoyé au poste central sans que la table elle-même soit modifiée à l'exception de celle de comptage qui est remise à zéro.

. Ecriture dans une table
Cette action ne concerne que les tables traitées par les tâches de sortie d'états tout ou rien (TBVTOR) ou de valeurs analogiques (TBVANA). Elle permet, selon le cas, de modifier un bit (table de sortie TOR) ou un mot d'une table, ou la totalité de la table.

. Initialisation des tables d'acquisition
Cette action provoque la mise à zéro des tables d'acquisition. Ceci permet au P.C., après coupure secteur, ligne hors service, arrêt du P.A., etc.., de pouvoir réactualiser ses tables sans risquer de saturer les tables de changements d'états.

. Transmission d'une table de changements d'états
Le contenu d'une des tables de changement d'états tout ou rien (TBGTR ou TBGTL) ou analogiques (TBIANA) est envoyé au poste central. La table est ensuite remise à zéro.

Dans le cas où le P.C. veut rappeler le dernier message reçu du P.A., il renvoie à celui-ci le même message en positionnant en plus le bit de répétition (voir paragraphe 4). Le P.A. retransmet alors la table précédemment émise sans modifications de son contenu. Cette possibilité est surtout utilisée pour la retransmission d'une table de changements d'états.

4. Structure des messages

Un message est constitué de deux parties principales, une structure de contrôle des informations et les informations proprement dites.

- Structure de contrôle
L'ensemble des éléments qui la constitue permettent de définir la taille du message et de contrôler la validité de celui-ci. Chacun de ces éléments est codé sur un octet.
  . Taille du message : cette valeur comprise entre 3 et 255 représente le nombre d'octets, transmis sur la ligne moins un.
  . Numéro de poste auxiliaire : cette information permet d'identifier l'origine (P.A. vers P.C.) ou la destination (P.C. vers P.A.) du message transmis sur la ligne.
  Si le numéro de P.A. a pour valeur zéro, le message transmis s'adresse à tous les P.A.

. Nature du message : l'action demandée au P.A. et/ou le type d'information transmise sont caractérisés par un code (4 bits) et un sous-code (3 bits). Le tableau figure 4.18 récapitule les types de messages standard. Le dernier bit de l'octet permet au poste central de demander un nouvel envoi du message.

. Contrôle de validité du message : un octet égal au OU exclusif de tous les octets précédents du message, permet de vérifier la validité de celui-ci après transmission.

- Informations

Cas d'un échange P.C.-P.A. : si le poste central demande au poste auxiliaire une table, cette partie est vide. Si le poste central veut transmettre des ordres au processus, cette partie contient la table d'informations ; celle-ci peut être réduite à l'envoi d'un ou plusieurs mots ou à l'envoi d'un ou plusieurs bits. Cas d'un échange P.A.-P.C. : en réponse à une demande du poste central, cette zone contient les informations demandées. En réponse à la réception d'ordres venant du poste central, le poste auxiliaire envoie un acquit pour lequel cette zone est vide.

La structure de la partie informations, pour les différents volumes d'échanges (tables, mots, bits) est représentée par la figure 4 .16.

Le déroulement d'un échange est schématisé à la figure 4.17.

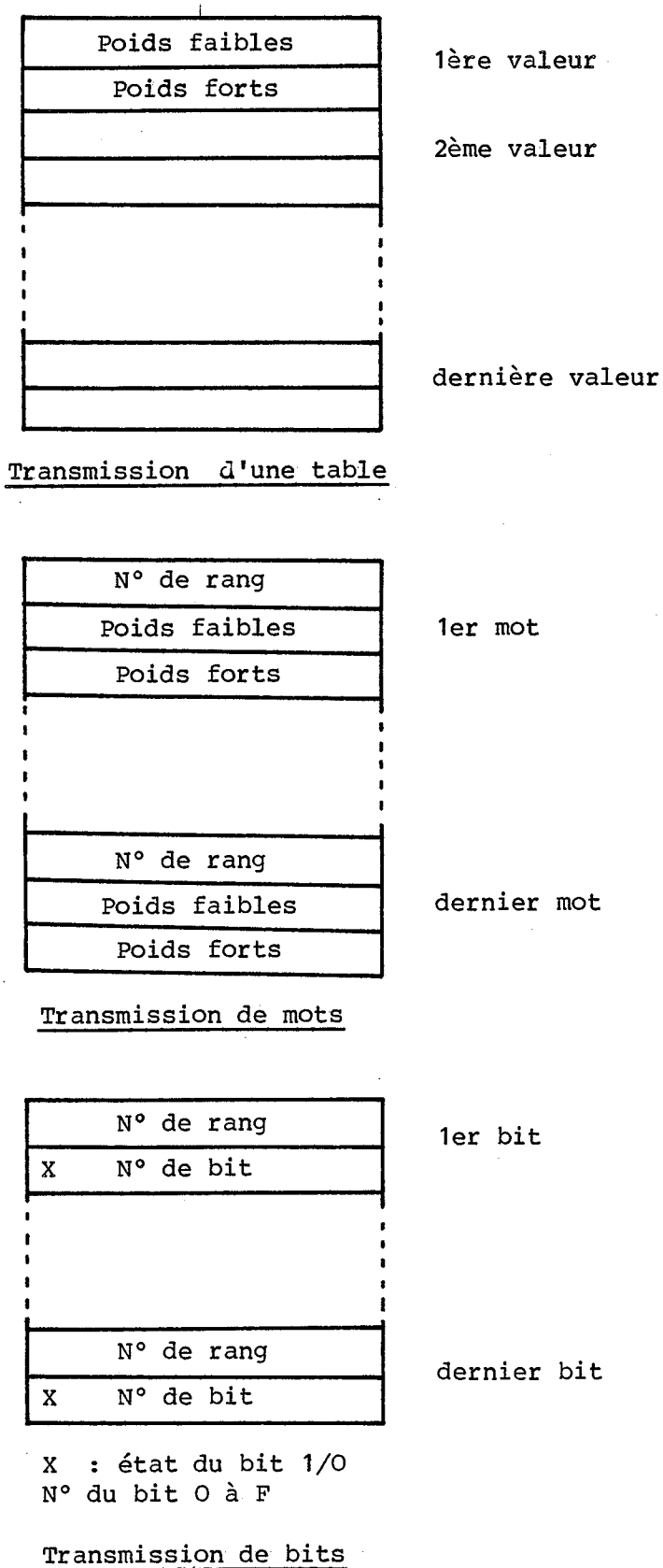


Figure 4.16. : Structure de la partie informations

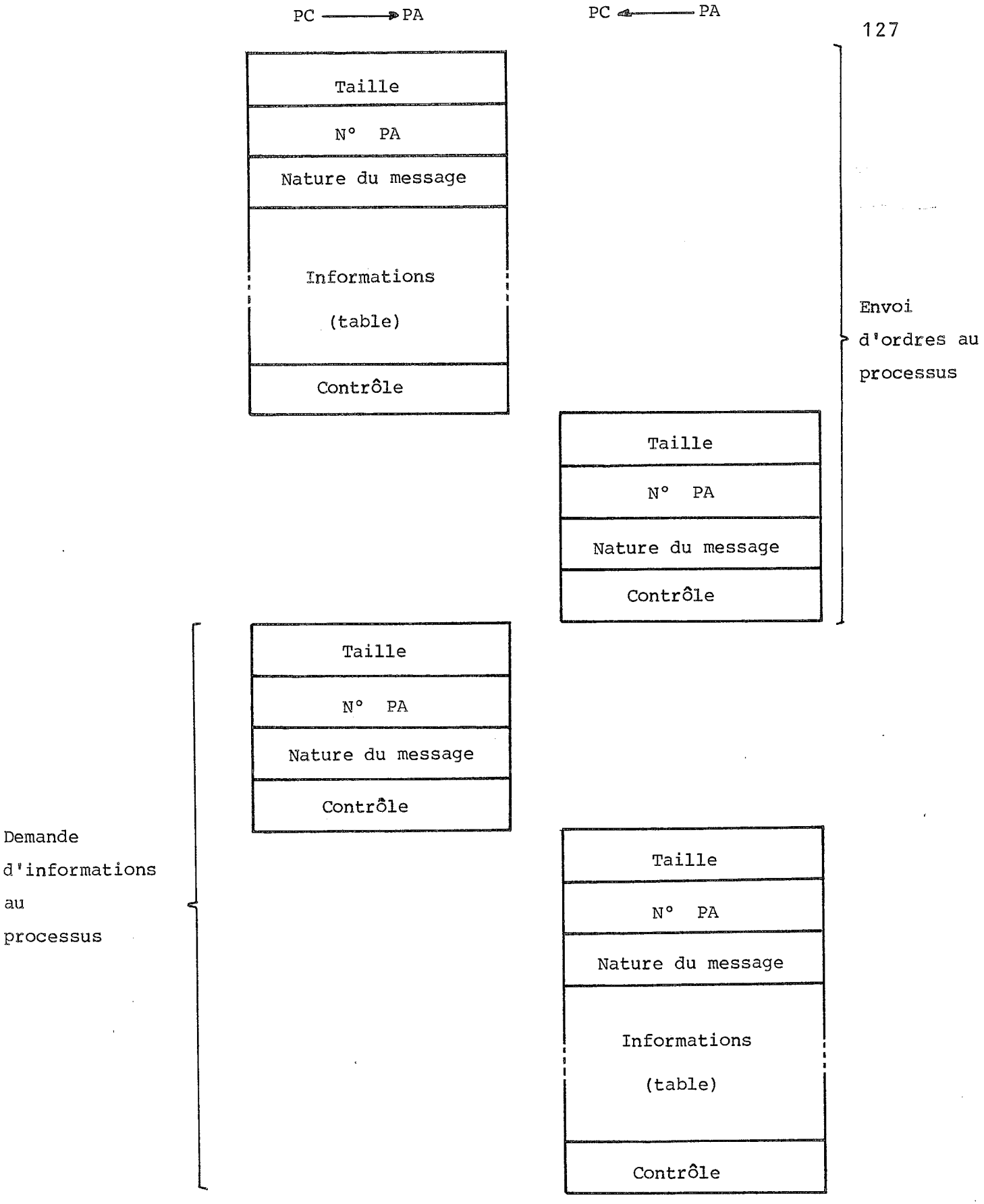


Figure 4.17. : Déroulement d'un échange

	CODE	SOUS CODE	BIT REPETER
MESSAGES SUR TABLES			
LECTURE DE LA TABLE	0001		
ECRITURE DE TOUTE LA TABLE	0100		
ECRITURE MOT DANS TABLE	0011		
ECRITURE BIT DANS TABLE	0010		
Repère de Table			
Table 0		000	
Table 1		001	
Table 2		010	
Table 3		011	
Table 4		100	
Table 5		101	
Table 6		110	
Table 7		111	
COMMANDES DIVERSES			
INITIALISATION	0101		
Demande d'initialisation		001	
Validation scrutation après initialisation		010	
LECTURE CHANGEMENT D'ETAT	0110		0-1
Table 0 :			
- Tout ou rien (TBGTR)		000	
- Tout ou rien (TBGTL)		001	
Table 1 :			
- Entrées analogiques		010	

Figure 4.18. : Récapitulatif des types de messages standard

5. GENERATION DU MONITEUR ET DU SYSTEME D'ACQUISITION; RESTITUTION

Nous avons étudié au cours des deux chapitres précédents un système constitué d'un moniteur temps réel et de tâches de traitement (acquisition, restitution, transmission). Ces logiciels ont été conçus pour satisfaire un ensemble de besoins standard que l'on rencontre dans de nombreuses applications de contrôle de processus.
Ils seront particularisés et adaptés à l'application de l'utilisateur en définissant les tables et les paramètres spécifiques, au cours d'une phase de génération que nous décrirons ci-dessous.

5.1. GENERATION DU MONITEUR

La phase de génération proprement dite est précédée de l'adaptation des différentes tables et particularités du moniteur au niveau des fichiers sources. Celles-ci sont regroupées dans trois fichiers où sont définies les adresses d'implantation du moniteur, ses caractéristiques et les tables des périphériques.

- Implantation

Dans le fichier CLEMON seront définies les adresses d'implantation du code et des tables du moniteur en mémoire morte, de la mémoire vive qu'il utilise, ainsi que de la zone où sont rangées les valeurs initiales du pointeur de pile et du pointeur de programme des différentes tâches. Cette dernière zone est définie en tête des tâches d'application.

- Caractèristiques

Dans le fichier DEFMON seront définis un ensemble d'éléments, certains par leur présence, d'autres par leur valeur :

. Nombre d'éléments de la file d'attente traités par l'horloge (cette valeur est fonction du nombre de délais et d'activations périodiques de tâches)
. Nombre de périphériques gérés par le moniteur
. Nombre de sémaphores réservés à l'utilisateur
. Numéro de la tâche à lancer lors de l'initialisation
. Numéro du périphérique "console"
. Code (caractère) d'activation de la tâche console
. Présence du moniteur d'aide à la mise au point
. Présence de l'option secours mémoire.

- Tables des périphériques

Dans le fichier TABPER seront définies les caractèristiques matérielles et logicielles de chaque périphérique. Les seuls périphériques pouvant être pris en compte par le moniteur actuel sont les lignes asynchrones.

. Définition de périphérique
Nom du périphérique, adresse du point d'entrée du "handler", (initialisation), adresse du point d'entrée sur time-out, adresse physique du périphérique, numéro du contrôleur d'interruption, valeur initiale du time-out.

. Définition de vecteur d'interruption
  Numéro de l'interruption sur le contrôleur, nom du périphérique (défini précédemment), adresse du point d'entrée sur interruption dans le "handler".
. Initialisation de ligne asynchrone
  Adresse physique du périphérique, adresse physique du temporisateur programmable associé, numéro du temporisateur utilisé, vitesse de transmission.
. Déclaration de la ligne asynchrone de l'U.C.
  Nom du périphérique, valeur du time-out en entrée, vitesse de transmission.
  Cette dernière description remplace les trois précédentes pour la ligne U.C.

Après avoir défini cet ensemble de fichiers, la génération s'effectue automatiquement en lançant l'enchainement de travaux GENMON.
Les listings du moniteur seront obtenus en lançant l'enchainement LSTMON.

5.2. GENERATION DU SYSTEME D'ACQUISITION-RESTITUTION

La phase de génération de ce système consiste, de la même façon que pour la génération du moniteur, à définir un ensemble de paramètres et de tables caractérisant la configuration matérielle et logicielle (tâches à activer, correction analogique, filtrage, etc...).
Cette opération sera facilitée par l'utilisation d'un logiciel conversationnel permettant de constituer le fichier de définition : DZS"NAM".
La zone "NAM" est remplacée au cours du dialogue opérateur par un nom associé à l'application.

- Choix des tâches

Dans un premier temps seront définies les tâches d'acquisition et de restitution qui seront activées par la tâche d'initialisation. En effet, si l'utilisateur prévoit de ne jamais mettre en oeuvre un type de carte, le traitement associé n'a aucune raison d'être sollicité ; ceci permettant d'économiser du temps d'Unité Centrale.
Cependant, si initialement ce même utilisateur n'a pas intégré un type de carte, mais s'il pense en avoir l'utilité ultérieurement, il demandera que la tâche correspondante soit générée.
La table de configuration associée permettra de spécifier à la tâche le nombre de cartes à traiter (0 ou n).

Le dialogue qui a permis d'établir la liste des tâches à intégrer dans l'application affecte des valeurs aux variables d'assemblage conditionnel correspondant à ces tâches. En fonction de ces valeurs, au cours de la phase d'assemblage, le code des tâches associées sera généré ou non.

- Tables en mémoire vive

L'application utilise pour ses besoins en stockage de données, des tables en mémoire vive. Une partie de celles-ci est directement liée à la structure et au volume de la configuration (tables d'acquisition et de restitution, tables de périodes, tables de rangs...).
Afin d'utiliser au mieux la zone mémoire vive, laquelle est de taille limitée à 800 octets, la longueur de ces tables sera définie à l'aide de paramètres liés au nombre (16 maximum) et aux types de cartes d'interface avec le processus.
La table présentée à la figure 5.1. définit pour chaque carte le volume de mémoire vive utilisée.

	Type de carte	Traitement	Occupation mémoire vive (octets)	
			Table de stockage pour 1 carte	Table de travail par tâche
ENTREES	TOR 16 E	Simple scrutation	2	20
		Double scrutation	4	
		(Si changement d'état)	3	
	TOR 8 E	Simple scrutation	1	
		Double scrutation	2	
		(Si changement d'état)	2	
	TOR 16 E	Comptage	37	20
	TOR 8 E	Comptage	21	
	TOR Multiplexées 2 cartes 16s + 1 carte 16e	Simple scrutation	64	20
		(Si changement d'état)	96	
	ANA 8 voies		16	14
SORTIES	TOR 16 S		4	5
	TOR 8 S		2	
	ANA 8 voies		6	10

Figure 5.1. : Occupation mémoire vive

- Tables de configuration

  Au cours de l'étude des tâches de traitement, dans le chapitre précédent, nous avons décrit des tables de configuration. Celles-ci permettent de définir avec précision la structure matérielle d'une application, ainsi que certains traitements logiciels. Une partie du contenu de ces tables (types et nombres de cartes) a été acquis au cours de la phase d'élaboration des tables en mémoire vive. Les adresses des cartes, les périodes d'acquisition, les traitements... sont définis à l'aide d'un complément de dialogue opérateur.

Après avoir constitué le fichier DZS"NAM", la génération de l'application s'effectue automatiquement en lançant l'enchainement de travaux CZS"NAM".

CONCLUSION

Nous nous étions fixés pour but d'étudier et de réaliser un logiciel d'application de télétransmission mettant en oeuvre un Moniteur Temps Réel, sur microprocesseur.

Nous en sommes arrivés à définir un produit logiciel aisément configurable, en fonction des besoins des utilisateurs.
Celui-ci a déjà eu l'occasion d'être testé sur diverses applications de contrôle de processus (régulation de cuve d'aluminium) et de consignation d'états (aéroport de Bagdad).

Cette étude a été commencée il y a maintenant 3 ans et l'on peut s'interroger sur l'intérêt qu'elle présente, appliquée aux microprocesseurs 8 bits, alors que les modèles 16 bits sont de plus en plus utilisés. En fait, les coûts de développement des cartes d'électronique utilisant ces derniers sont nettement supérieurs à ceux nécessaires au développement et à la réalisation de systèmes 8 bits.
Dans l'industrie, cette différence de coût est toujours un critère de choix important, dans la mesure où la différence de caractèristiques qui est en faveur des 16 bits, ne se justifie pas par le niveau de puissance de calcul demandé.
Le type d'application qui a guidé notre étude fait appel à de nombreux systèmes répartis (voir figure 4.1.) réalisant essentiellement les fonctions d'interface, la fonction traitement des données étant réalisée par le poste central.

L'importance de la réduction des coûts des systèmes décentralisés apparait donc d'autant plus clairement ; de plus, le logiciel étant défini comme un produit, son impact sur le coût final est considérablement réduit par rapport à des logiciels spécifiques.

Malgrè cela, il est certain que dans d'autres types d'application (nombreuses tâches en parallèles, calculs mathématiques) que celles auxquelles nous nous sommes intéressés (acquisition-restitution, traitements mathématiques limités), les microprocesseurs 16 bits, voire 32 bits, seront les seuls à être à même de réaliser des traitements de façon suffisamment rapide.

Le logiciel que nous avons décrit dans ce rapport est destiné à fonctionner sur un seul type de microprocesseur.
Cependant, sa structure et son organisation sont directement applicables à des matériels plus performants et les concepts de modularité et de facilité d'adaptation qui ont guidé notre étude pourront être conservés et rendront plus aisée une éventuelle transposition.

ANNEXE 1

EXEMPLE DE PROGRAMMATION D'UNE TACHE

```
:S      592     CR8085/11 version 4.45.75       15-Jul-81  8:21:45      Page 1
TACHE DE SORTIES TOR   SORTOR.SRC

                        ;
                        ; FICHIER DE DEFINITION DE L'APPLICATION
                        ;
                                RSECT   ROM
                        ;
                        ; DEFINITION DES TACHES
                        ;
0000    INECPT  EQU     0       ; INDICATEUR TACHE ENTREES DE COMPTAGE
0000    INETOR  EQU     0       ;     "          "      "         TOR
0000    INEMUX  EQU     0       ;     "          "      "         TOR MULTIPLEXEES
0001    INEANA  EQU     1       ;     "          "      "         ANA
0000    INSTOR  EQU     0       ;     "          "   SORTIES TOR
0001    INSANA  EQU     1       ;     "          "   SORTIES ANA


                        ;
                        ; DEFINITION DES PARAMETRES DE LA TACHE DE COMPTAGE
                        ;
0002    CPTNBC  EQU     2       ; NBRE CARTES DE COMPTAGE POUR TABLE ETATS ANTERIEURS
0020    CPTRNB  EQU     32.     ; NBRE DE CPTRS

                        ;
                        ; DEFINITION DE PARAMETRES DES TACHES ENTREES TOR ET MULTIPLEXEES
                        ;
0002    TSSNBC  EQU     2       ; NBRE DE CARTES TOR SIMPLE SCRUTATION
0002    TDSNBC  EQU     2       ; NBRE DE CARTES TOR DOUBLE SCRUTATION

0004    TVALNB  EQU     4       ; NBRE DE VALEURS TOR (S.SCRU+DBLE.SCRU+TOR.MUX)
0002    TQLNBC  EQU     2       ; NBRE  DE CARTES TOR LENTES
0002    TQRNBC  EQU     2       ; NBRE  DE CARTES TOR RAPIDES
0000    TMXNBC  EQU     0       ; NBRE DE CARTES D'ENTREES MULTIPLEXEES


                        ;
                        ; DEFINITION DES PARAMETRES DE LA TACHE ENTREES ANA
0001    ANANBC  EQU     1       ; NBRE DE CARTES ENTREES ANALOGIQUES
0008    ANEVAL  EQU     8       ; NBRE DE VALEURS ANALOGIQUES MAX


                        ;
                        ; DEFINITION DES PARAMETRES DE LA TACHE SORTIES ANALOGIQUES
                        ;
0002    ANSNBC  EQU     2       ; NBRE DE CARTES SORTIES ANALOGIQUES
0010    ANSVAL  EQU     16.     ; NBRE DE VALEURS ANALOGIQUES MAX

                        ;
                        ; DEFINITION DES PARAMETRES DE LA TACHE SORTIES TOR
```

```
ZS      592     CR8035/11 version 4.45.75       15-Jul-81  8:21:45      Page 1-1
TACHE DE SORTIES TOR   SORTOR.SRC

                        ;
                        ;
0002                    STRNBC  EQU     2       ; NBRE DE CARTES DE SORTIE TOR
                                TITLE   ZS      592
                        ;
                        ;
                                SUBTTL  TACHE DE SORTIES TOR
                        ;
                        ;
                        ;*****************************************************
                        ;*****************************************************
                        ;**                                                 **
                        ;**  CETTE TACHE EFFECTUE LA RESTITUTION DES INFOR-**
                        ;**  MATIONS TOR EN PROVENANCE DU POSTE CENTRAL AUX**
                        ;**  CARTES DE SORTIES TOR:                         **
                        ;**                          -ZS 551  (16S)         **
                        ;**                          -ZS 552  ( 8S)         **
                        ;**                                                 **
                        ;**                                                 **
                        ;**  LES INFORMATIONS SONT RAFRAICHIES TOUTES LES   **
                        ;**  100MS                                          **
                        ;**                                                 **
                        ;*****************************************************
                        ;*****************************************************
                        ;
                        ;
                        ;  VERSION 01 05-05-1981
                        ;
                        ;  M.FAISSE
                        ;
                        ;


                        ;
                        ;
                        ;  REFERENCES EXTERNES
                        ;
                        ;
                                EXTB    ADBUS,CD16ST
                                EXTERN  TBSTOR
                                EXTERN  TBVTOR
                                EXTERN  PKTERM,TSESOR,RSESOR

                        ;
                        ;
                        ;  FICHIER  DEFINITION MACROS
                        ;
                        ;
                                INCLUDE [210,200]MAC35.SRC
```

```
:S      592     CR3085/11 version 4.45.75       15-Jul-81  8:21:50      Page 4
TACHE DE SORTIES TOR   SORTOR.SRC

                        ;       RESERVATION STACK TACHE
                        ;
                        ;
                                RSECT   STCI
0000" 00 00                     DW      STACK
0002" 00 00                     DW      DEBUT
                        ;
                                RSECT   STACK
          0023                  DS      40.
0028"                   STACK:
                        ;
                        ;
                        ;
                                RSECT   ROM
                        ;
                        ;*****************************************
                        ;*                                       *
                        ;* SORTIE DES ETATS SUR LES CARTES 16S OU *
                        ;*                                   8S  *
                        ;*                                       *
                        ;*****************************************
                        ;
0000" 3A 00 00$ DEBUT:  LDA     TBSTOR+1        ; NBRE DE CARTES DE SORTIES TOR
0003" 32 00 00          STA     NBCAR
                        ;
0006" 21 00 00$         LXI     H,TBSTOR+3      ; @ TABLE DE CONFIG.SORTIES TOR
0009" 22 00 00          SHLD    PTCTOR          ; (@ POIDS FAIBLES DE LA CARTE X0)
                        ;
000C" 21 00 00$         LXI     H,TBVTOR        ; @ TABLE VALEURS ETATS TOR A SORTIR
000F" 22 00 00          SHLD    PTVTOR
                        ;
0012" 21 00 00$         LXI     H,TSESOR
0015" FF                RST     7
                        ;
0016" 2A 00 00  1$:     LHLD    PTCTOR
0019" 06 FF$            MVI     B,ADBUS         ; ELABORATION @ CARTE: POIDS FORTS
001B" 4E                MOV     C,M             ;                      POIDS FAIBLES
001C" EB                XCHG                    ; SAUVEGARDE DE H ET L DANS D ET E
                        ;
001D" 2A 00 00          LHLD    PTVTOR
0020" 7E                MOV     A,M
0021" 02                STAX    B               ; SORTIE DES POIDS FAIBLES DE LA VALEUR
                        ;
0022" 1B                DCX     D               ; POINTE SUR TYPE DE CARTE
0023" 1A                LDAX    D
0024" FE FF$            CPI     CD16ST          ; CARTE 16S
0026" C2 2D 00"         JNZ     2$              ; NON: CARTE 8S
                        ;
0029" 0C                INR     C               ; @ CARTE X1
002A" 23                INX     H
002B" 7E                MOV     A,M
```

```
ZS      592     CR8035/11 version 4.45.75       15-Jul-81  8:21:50      Page 4-1
TACHE DE SORTIES TOR   SORTOR.SRC

        729  002C" 02                    STAX    B                       ; SORTIE DES POIDS FORTS DE LA VALEUR
        730
        731                      ;
        732                      ; MISE A JOUR DES POINTEURS
        733                      ;
        734
        735  002D" 21 00 00$   2$:      LXI     H,RSESOR                ; LIBERATION DU SEMAPHORE
        736  0030" FF                    RST     7
        737
        738  0031" 2A 00 00              LHLD    PTCTOR
        739  0034" 23                    INX     H
        740  0035" 23                    INX     H
        741  0036" 22 00 00              SHLD    PTCTOR
        742
        743  0039" 2A 00 00              LHLD    PTVTOR
        744  003C" 23                    INX     H
        745  003D" 23                    INX     H
        746  003E" 22 00 00              SHLD    PTVTOR
        747
        748  0041"                       DECR    NBCAR
        749  0047" C2 15 00"             JNZ     1$
        750
        751
        752  004A" 21 00 00$   FIN:      LXI     H,PKTERM
        753  004D" FF                    RST     7
        754
```

```
ZS      592     CR8085/11 version 4.45.75       15-Jul-81  8:21:50      Page 5
TACHE DE SORTIES TOR   SORTOR.SRC

                                        RSECT   RAM
                                ;
                                ;
                                ;  PARAMETRES UTILISES PAR LA TACHE SORTIE TOR
                                ;
                                ;
0000" 00                NBCAR:  DB      0       ; NBRE DE CARTES
0001" 00 00             PTCTOR: DW      0       ; PTR TABLE DE CONFIG.
0003" 00 00             PTVTOR: DW      0       ; PTR TABLE VALEURS A SORTIR
                                END
```

ANNEXE 2

BIBLIOGRAPHIE

GIROD D. & DUBOIS R., *Au coeur des microprocesseurs*, Paris, Eyrolles, 1979.

LILEN H., *Du microprocesseur au micro-ordinateur*, Paris, Radio, 1980.

TOULOTTE J.M., *Dispositif de commande en temps réel*, Paris, Dunod Université.

ROLANDER Thomas, *RMX/80 real time multitasking executive*, Santa Clara U.S.A., Intel Corporation, 1978.

RMX/80 user's guide, Santa Clara U.S.A., Intel Corporation, 1978.

Minibug 3E user's guide, Genève, Motorola semiconductor products Inc.

MARION B., *Liaison controbloc TCI - Moniteur multitâches M6800*, Electricité de France, juin 1979.

Moniteur MOP2 Manuel de référence, Velizy, CIMSA, juillet 1979.

Moniteur 8085 - MON85, traduit de INTEL.

Moniteur Z80 - ZILOG.

Cours logiciel temps réel - Support de cours - Concepts fondamentaux, Velizy, STERIA.

Cours logiciel temps réel - Support de cours - Moniteurs temps réel, Velizy, STERIA.

BASS Charlie, A perspective on microcomputer software, *Proceeding of the IEEE*, VOL.64, N°6, june 1976.

PARRISH E.A., A scheduler for real time task control in micro-computers, *IEEE transaction on industrial electronic and control instrumentation*, IECI.25, Feb.78, pp. 21-25.