MASTER EN SCIENCES INFORMATIQUES

Contributions aux réseaux locaux dans les systèmes temps réel

Vangeersdael, Joëlle

Award date:
1985

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR
Institut d'informatique

CONTRIBUTIONS AUX RESEAUX LOCAUX DANS LES SYSTEMES TEMPS REEL

Directeur du mémoire :

Ph. Van Bastelaer

Mémoire présenté par

Joëlle Vangeersdael

en vue de l'obtention
du grade de
licenciée et maître
en Informatique

Année 1984-85

Nous remercions pour leur aimable collaboration la section du CERN au sein de laquelle nous avons évolué pendant une période de six mois et plus particulièrement Monsieur Ghinet dont les conseils nous furent précieux.

Nous tenons également à témoigner notre gratitude à Monsieur Le Lann dont les travaux en matière de réseaux locaux nous ont inspiré et à notre directeur de mémoire, Monsieur Van Bastelaer, dont la lecture attentive du document nous a été d'un secours inestimable.

Enfin, nous remercions tous ceux qui, de près ou de loin, nous ont encouragé dans la poursuite de ce fameux défi que représente un mémoire...

TABLE DES MATIERES

INTRODUCTION : LE CERN ET SES RESEAUX LOCAUX TEMPS REEL

Le 29 septembre 1954 entrait en vigueur la convention qui a institué le Centre Européen de Recherches [sub]Nucléaires ([sub]Nucléaire veut dire à l'échelle plus petite que l'atome) appelé aussi Laboratoire pour la Physique des Particules. Ce centre le plus moderne d'Europe Occidentale a ses laboratoires installés sur 80 hectares à la frontière franco-suisse, une moitié dans le Pays de Gex, à Prévessin, l'autre moitié à Meyrin, à 8 km de Genève.

Cette convention scellait l'entente de 13 pays européens pour la construction d'accélérateurs : ces installations cyclopéennes pour étudier l'infiniment petit. Un accélérateur se présente, le plus souvent, sous la forme d'un énorme anneau, long de plusieurs km, placé au coeur de tunnels bétonnés. L'anneau est comme un immense circuit de vitesse à l'intérieur duquel, sous vide, les faisceaux de particules sont accélérés jusqu'à une vitesse proche de la lumière et guidés grâce à des aimants.

Le premier en date, l'accélérateur PS (Proton Synchrotron), avait une circonférence de 300 m. En 1976, il n'était déjà plus que l'appendice de l'accélérateur SPS (Super Synchrotron à Protons), de 7 km de circonférence, dont le premier rôle était d'accélérer les protons à une énergie de 400 GeV avant de les envoyer bombarder des cibles situées dans des zones expérimentales ; de ces collisions sortait un certain nombre de particules élémentaires. Depuis 1981, comme les collisionneurs de matière et d'antimatière se sont révélés de meilleurs outils dans la production de particules intéressantes, le SPS est utilisé comme centre de collisions proton-antiproton.

Comme les particules à rechercher sont de plus en plus infimes, on a besoin pour les "détacher" des objets bombardés de projectiles animés d'énergies de plus en plus élevées. C'est pourquoi on a songé à la construction, pour la période 1981-1986, d'un anneau de 27 km de circonférence, le LEP (Laboratoire Electrons Positons) pour lequel le SPS ferait office d'injecteur et qui se situerait pour 3/4 dans le Pays de Gex et pour 1/4 dans le Canton de Genève.

Pour gérer le fonctionnement de la myriade d'équipements nécessaires à la mise sur pied des accélérateurs, on a besoin de systèmes informatiques de contrôle, travaillant dans un environnement temps réel car la conduite d'un faisceau de particules nécessite une mécanique bien huilée ayant des temps de réaction de l'ordre de la milliseconde. Ces systèmes de contrôle

développent des architectures en réseau local c'est-à-dire en réseau de communication pour interconnecter une variété d'appareils s'échangeant des données sur une petite surface : le réseau en étoile pour permettre le dialogue entre les ordinateurs contrôlant l'accélérateur PS, le réseau interconnectant des sous-réseaux en étoile pour contrôler le SPS et le réseau en anneau "Token Ring" pour contrôler le LEP.

Pour assurer le fonctionnement de cette "station-service" de la physique nucléaire, les physiciens ont besoin de dizaines, voire de centaines de collaborateurs. Ils proviennent d'universités ou de laboratoires des 12 états membres. C'est dans cet esprit que nous avons été engagé pour une période de 6 mois au CERN. Cette prise de contact avec les réseaux locaux dans les systèmes temps réel a été le point de départ de notre étude.

Notre étude est une réflexion, sur base des réalisations du CERN, sur la manière dont **les réseaux locaux** peuvent rentrer dans les objectifs **des systèmes temps réel** (partie I) c'est-à-dire présenter **un temps de réponse** (partie II) et **un niveau de fiabilité** (partie III) satisfaisants.

Le travail se découpe, pour ces raisons, en trois parties.

La **PREMIERE PARTIE** s'intitule "**temps réel et ses illustrations distribuées**". Nous y débatons, dans un premier chapitre, des deux caractéristiques essentielles de tout système temps réel : le temps de réponse et le souci de fiabilité. Les deux chapitres suivants viennent en guise d'illustration : le système de contrôle du SPS (chapitre II) et celui du LEP (chapitre III). Nous y montrons, d'abord, en quoi une architecture distribuée répond aux besoins temps réel avant de passer à une description plus détaillée des systèmes mis en place : réseau local de communication, ordinateur-satellite, ... Certains lecteurs seront sans doute étonnés du niveau de détail avec lequel certains passages seront abordés. Cette attitude est volontaire et a pour but de les sensibiliser avec l'environnement dans lequel notre stage s'est effectué pour les amener à mieux comprendre le chapitre VI.

La **DEUXIEME PARTIE** "**temps de réponse et réseau local**" s'intéresse à l'étude des réseaux locaux sous l'angle du temps de réponse, première caractéristique générale des systèmes temps réel. Dans un premier temps (chapitre IV), nous précisons une série de concepts importants pour la

suite de l'exposé : notion de réseau local et ses critères d'analyse, notion de protocole,... et présentons notre modèle inspiré des processus stochastiques schématisant le fonctionnement des réseaux locaux. Dans un deuxième temps (chapitre V), nous étudions l'impact, dans un réseau local, de la gestion de l'accès concurrent à un moyen de communication unique par des utilisateurs dispersés (notion de protocole) sur le temps de réponse. Est-il borné supérieurement ou non ? L'existence de cette borne décide de l'utilisation possible d'un protocole dans un système temps réel où les messages doivent avoir été transmis au destinataire endéans un certain temps fixé a priori. Les différents protocoles y sont abordés selon une classification en arbre binaire. Cette idée est le fruit d'une rencontre au CERN avec Monsieur Le Lann, chercheur de l'INRIA, confronté dans ses recherches actuelles au problème du temps de réponse obtenu dans les réseaux locaux.

Dans la TROISIEME PARTIE "amélioration de la fiabilité du système de contrôle du SPS" qui se limite au seul chapitre VI, nous illustrons d'abord les moyens mis en oeuvre au SPS pour assurer un certain niveau de fiabilité du système informatique de contrôle, deuxième caractéristique des systèmes temps réel, avant de présenter notre contribution à son amélioration par la mise au point d'outils accélérant le diagnostic des pannes.

La lecture du travail peut s'effectuer selon deux axes :

- l'axe temps de réponse dans les réseaux locaux évoluant dans un environnement temps réel c'est-à-dire les chapitres I, IV et V ;
- l'axe amélioration de la fiabilité dans les réseaux locaux évoluant dans un environnement temps réel c'est-à-dire les chapitres I, II, III et VI.

PARTIE I : TEMPS REEL ET SES ILLUSTRATIONS DISTRIBUEES

Chapitre I : LES CONCEPTS DE TEMPS REEL ET D'ARCHITECTURE DISTRIBUEE

Dans ce chapitre, l'accent sera porté sur la notion de temps réel (voir point I.1) et les deux caractéristiques essentielles qui permettent de différencier un système temps réel d'autres systèmes :

- le temps de réponse (voir point I.2) ;
- le souci de fiabilité (voir point I.3).

Nous montrerons que ces conditions peuvent exercer un impact sur l'architecture adoptée par le système informatique (voir point I.4).

I.1. La notion de temps réel

Supposons un système en liaison directe c'est-à-dire un système dans lequel les entrées sont directement introduites dans l'ordinateur depuis leur point d'origine via des terminaux et les sorties directement transmises là où elles doivent être utilisées. Supposons des terminaux connectés à l'ordinateur par une ligne téléphonique ou par tout autre moyen de télécommunication. Supposons des messages ou des transactions qui arrivent à l'ordinateur en provenance des terminaux. Ceux-ci ne sont pas stockés les uns à la suite des autres sur des fichiers à bandes ou à cartes pour être triés et traités séparément plus tard comme sur un système en temps différé mais ils sont au contraire traités immédiatement, pour adresser une réponse dans un délai de l'ordre d'une seconde ou d'une fraction de seconde aux terminaux intéressés, qui peuvent se trouver à des kilomètres de l'ordinateur.

Voilà, nous nous trouvons en présence *d'un système temps réel* que J. Martin (Martin, 11, p5) définit comme : " ...étant un système qui gère une situation en recevant des données, en les traitant et en retournant les résultats suffisamment vite pour agir à temps sur l'évolution de la situation".

En voici quelques illustrations :

- le système de réservation de la Pan American réparti de par le monde : un message émis par une agence de voyage de Rome, par exemple, recevra une réponse de l'ordinateur installé à New York dans les 5 secondes ;
- le système SAGE ou Semi-Automatic Ground Environment pour traiter automatiquement les données fournies par les météorologues, les autorités contrôlant le trafic aérien et les bases militaires afin de faire face à la menace posée par des avions et des missiles de plus en plus rapides pouvant transporter des têtes nucléaires (Everett, R.R., Zraket, C.A., and Bennington, H.D. SAGE, 5) ;
- la possibilité de délivrer endéans la minute aux clients d'une banque le relevé de leur compte et le service qui leur est offert d'effectuer des retraits automatiques (comme dans les systèmes Mister Cash et Bancontact) ;
- l'amélioration de l'écoulement de la circulation dans une ville par le

fonctionnement, aux moments opportuns, des feux de signalisation compte tenu de tous les véhicules qui se trouvent dans les rues ; un agent de police pourrait certainement commander les feux d'un carrefour d'une façon optimum ; s'il disposait des informations nécessaires, il pourrait commander au mieux les feux de quatre carrefours ; mais un système basé sur un grand nombre d'agents répartis dans la ville serait inopérant ; en revanche, un ordinateur en liaison directe serait assez rapide pour prendre en charge tous les feux, essayer de grouper les véhicules par "rames" et les manoeuvrer à travers les rues avec le minimum d'interférence entre eux et le minimum d'arrêts et de démarrages ;

- la lecture d'instruments de mesure d'un atelier ou la commande des dispositifs de contrôle tels que : vannes, régulateurs de vitesse (voir les illustrations du SPS et du LEP dans les chapitres suivants).

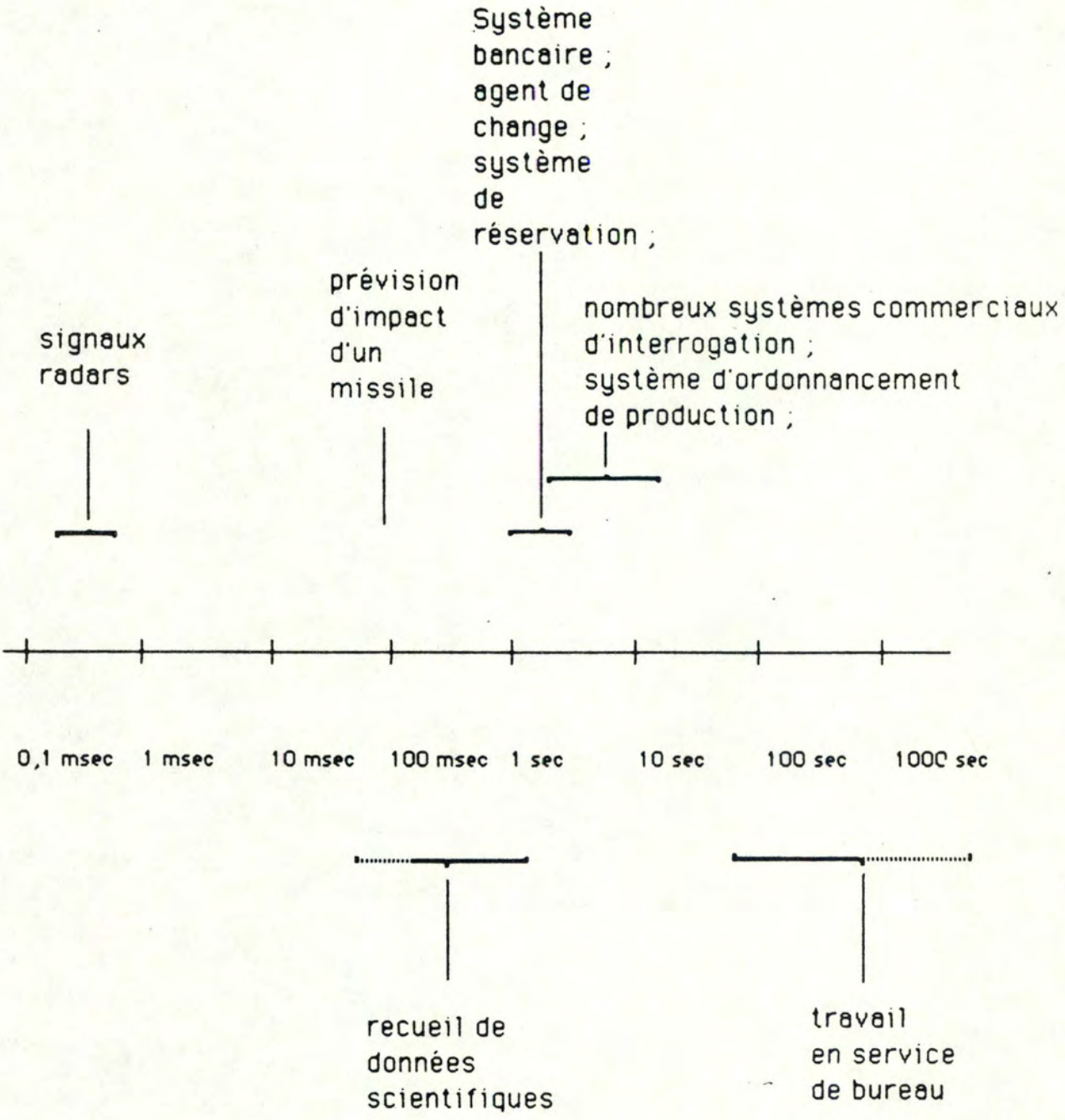
D'autres auteurs (Freedman et Lees, 7, p4) caractérisent un système temps réel comme étant *event-driven* car son but est de répondre à des événements extérieurs en un laps de temps suffisamment court pour pouvoir réagir à ces stimuli.

J. Martin souligne que le temps de réponse peut également intervenir dans la définition.

I.2. La notion de temps de réponse

Le *temps de réponse* est "le temps mis par le système pour réagir à une entrée déterminée de données" ou "le temps écoulé entre le moment où un événement se produit et celui où la réponse est connue" ou encore "le temps total de séjour d'une transaction dans le système de traitement, c'est-à-dire à partir du moment auquel elle est entièrement reçue jusqu'au moment où une réponse commence à être transmise", selon J. Martin (Martin, 10, p1-4).

Suivant le type d'application traitée, ce temps peut varier de la microseconde à une ou plusieurs secondes (voir figure I.1.). Il y a donc lieu d'effectuer la distinction entre les temps de réponse :



(tiré de MARTIN, 10, p 45)

Figure I.1.: Exemples de temps de réponse

- *immédiat* : des systèmes contrôlant un processus physique peuvent exiger un temps de réponse très rapide à certains événements de l'ordre de la microseconde ou de la milliseconde ; le processus d'accélération des particules dans les laboratoires de physique nucléaire comme le CERN exige une mécanique bien huilée dont le léger dérèglement peut avoir des conséquences désastreuses sur la poursuite d'une expérience ; de tels processus exigent des temps de réaction de l'ordre de la milliseconde ;
- *conversationnel* : un caissier de banque désire avoir un temps de réponse compatible avec les exigences de service aux clients c'est-à-dire de l'ordre de 2 à 3 sec ; ce temps est, en fait, influencé par le temps de réaction humaine pour éviter que l'opérateur humain au terminal ne devienne impatient ;
- *aussi-vite-que-possible* : certaines transactions peuvent être traitées endéans plusieurs minutes ou secondes mais ne sont pas contraintes par la vitesse de la conversation humaine ; un magasinier qui questionne le fichier des stocks peut se satisfaire d'une réponse qui lui parvient en 20 secondes ; pour assurer la commande d'une unité de raffinerie, 5 minutes peuvent convenir ;
- *en différé* : les transactions reçues des terminaux sont mises en file d'attente sur le site central et traitées suivant leur degré de priorité ; cependant, le système est conçu pour éviter des files trop longues et un temps d'attente supérieur à plus d'une heure pour les transactions ; dans cet esprit, pour envoyer des instructions à l'atelier d'une usine, une demi-heure peut suffire ;
- *endéans un jour* : aux commandes reçues le matin seront envoyées les réponses par le dernier courrier de la journée ;
- *plus d'un jour* : certaines fonctions ne sont pas urgentes et peuvent prendre une semaine ou plus ; de telles transactions peuvent être envoyées par la poste.

Certains auteurs défendent l'idée qu'un temps de réponse, dans le cadre d'un système temps réel, ne peut dépasser plusieurs millisecondes et que, de toutes façons, un système dont le temps de réponse dépasse une demi-heure et plus, n'est en aucun cas digne du titre de temps réel. Nous n'entrerons pas dans cette polémique mais indiquons qu'elle existe.

La conception de systèmes temps réel tourne autour de la nécessité d'assurer à tout moment un temps de réponse convenable même en cas de

période de pointe, ce qui peut signifier une sous-utilisation flagrante des capacités informatiques en dehors de ces périodes. Le choix du temps de réponse souhaité, tout en étant dicté par des contraintes strictes comme le temps de réponse à assurer pour un processus physique, résulte d'un compromis entre :

- son *coût* pour l'organisation entraîné par la pose et l'entretien de lignes à fortes capacités (la moitié du budget informatique de la BBL), par les extensions-mémoire pour contenir les programmes appelés aléatoirement, par un hardware plus coûteux pour accélérer le transfert des programmes des mémoires auxiliaires vers la mémoire centrale, ... ;
- et *un ensemble de critères subjectifs* tels que le service amélioré du client, la valeur inestimable pour la direction de disposer immédiatement de l'information voulue et les améliorations résultant d'une réduction du temps de réaction de l'organisation.

I.3. La notion de fiabilité

Les ordinateurs en temps réel ont généralement besoin d'être beaucoup plus fiables que les ordinateurs conventionnels.

Cette notion est cruciale vu le nombre croissant d'ordinateurs temps réel consacrés à des missions risquant de mettre en péril la société comme le contrôle des centrales nucléaires. La nature de leur fonction fait que tout arrêt de travail entraînerait de graves inconvénients, d'autant plus graves que les exigences de temps de réponse sont pressantes. Des systèmes à temps de réponse immédiat et conversationnel sont dérangés par une interruption de deux heures ; deux heures de panne ne seront pas catastrophiques pour les autres types de temps de réponse.

Les systèmes temps réel sont aussi plus automatiques. Les messages entrant dans le système sont bien souvent traités et renvoyés aux terminaux appropriés sans intervention humaine. Si une erreur survient pendant ce cycle, elle peut ne pas être décelée, à moins qu'elle ne saute aux yeux de

l'opérateur de terminal.

La fiabilité d'un système technique tant des points de vue hardware et software peut donc être définie, selon H.Kopetz (Kopetz, 9, p168), " comme la probabilité avec laquelle un système répond à ses spécifications fonctionnelles détaillées sous des conditions d'environnement spécifiées pour une période de temps donnée."

Une erreur est définie, selon le ANSI standard, "comme n'importe quelle divergence entre une quantité observée, calculée et la valeur vraie et correcte". Une faute qui à l'origine résultait des mauvais fonctionnements du hardware est définie maintenant, toujours selon le standard ANSI, "comme une construction mécanique ou algorithmique telle que sous certaines circonstances dans l'usage du système, cette construction amène le système dans un état erroné" (Kopetz, 9, p169).

Notons que la fiabilité n'est pas seulement liée au matériel ; elle est aussi un problème de programmation. En effet, dans un système qui possède un degré de multiprogrammation élevé, les procédures d'essai pour assurer la fiabilité des programmes sont particulièrement complexes, en raison du nombre presque infini de combinaisons de circonstances qu'il faudrait essayer.

Plusieurs techniques sont disponibles pour envisager le problème de la fiabilité :

- la *"Fault Avoidance Technique"* (voir point I.3.A) ;
- la *"Fault Tolerance Technique"* tant d'un point de vue hardware (voir point I.3.B) que software (voir point I.3.C).

Expliquons, avant toutes choses, quelques concepts utiles pour la suite :

- le *"Mean Time Between Failure"* (MTBF) définit le temps moyen entre deux pannes ;
- le *"Mean Time To Repair"* (MTTR) spécifie le temps moyen de réparation ;
- le rapport MTBF / (MTBF + MTTR) indique la disponibilité d'un composant.

1.3.A. La "Fault Avoidance Technique"

La "Fault Avoidance Technique" essaie d'éliminer les fautes avant que le système ne devienne opérationnel ; chaque fabricant construit des composants avec un certain taux de fiabilité qui peut être mesuré par sa disponibilité ; ce facteur peut être amélioré en augmentant le MTBF et en réduisant le MTTR.

Le **MTBF** peut être amélioré par la technique du vieillissement. En effet, étant prouvé statistiquement que les pièces les plus jeunes présentent le plus d'ennui technique, ces dernières subissent une période de rodage. Au terme de celle-ci, les pièces présentant des imperfections sont mises au rebut et les autres vendues, étant entendu que passées un certain âge, elles présentent des phénomènes d'usure et doivent être remplacées.

Quant à lui, le **MTTR** dépend entre autres :

- de l'ingénieur c'est-à-dire de sa compétence, de son habileté et de la distance le séparant du matériel en difficulté (s'il est dédicacé au contrôle exclusif de ce type d'équipement ou s'il a à parcourir de longues distances pour se rendre sur le lieu de la panne) ;
- ainsi que des méthodes mises à sa disposition pour effectuer la réparation ; ce critère joue un rôle prépondérant dans l'évaluation du MTTR et son amélioration fera l'objet de la troisième partie de ce mémoire.

Pour obtenir une mesure de la **disponibilité** du système global, il faut combiner la disponibilité de ses différents composants. Envisageons un système composé de deux unités de disponibilité respective a_1 et a_2. Si ces unités sont en série, la disponibilité globale sera donnée par $a = a_1 * a_2$; si les deux unités sont en parallèle dont l'une doit au moins fonctionner, la disponibilité globale sera donnée par $a = 1 - (1 - a_1)(1 - a_2)$.

Un ingénieur connaissant la disponibilité globale dont doit faire preuve le système arrangera les différentes unités suivant des combinaisons qui lui permettent d'atteindre le niveau de disponibilité désiré. Seulement la

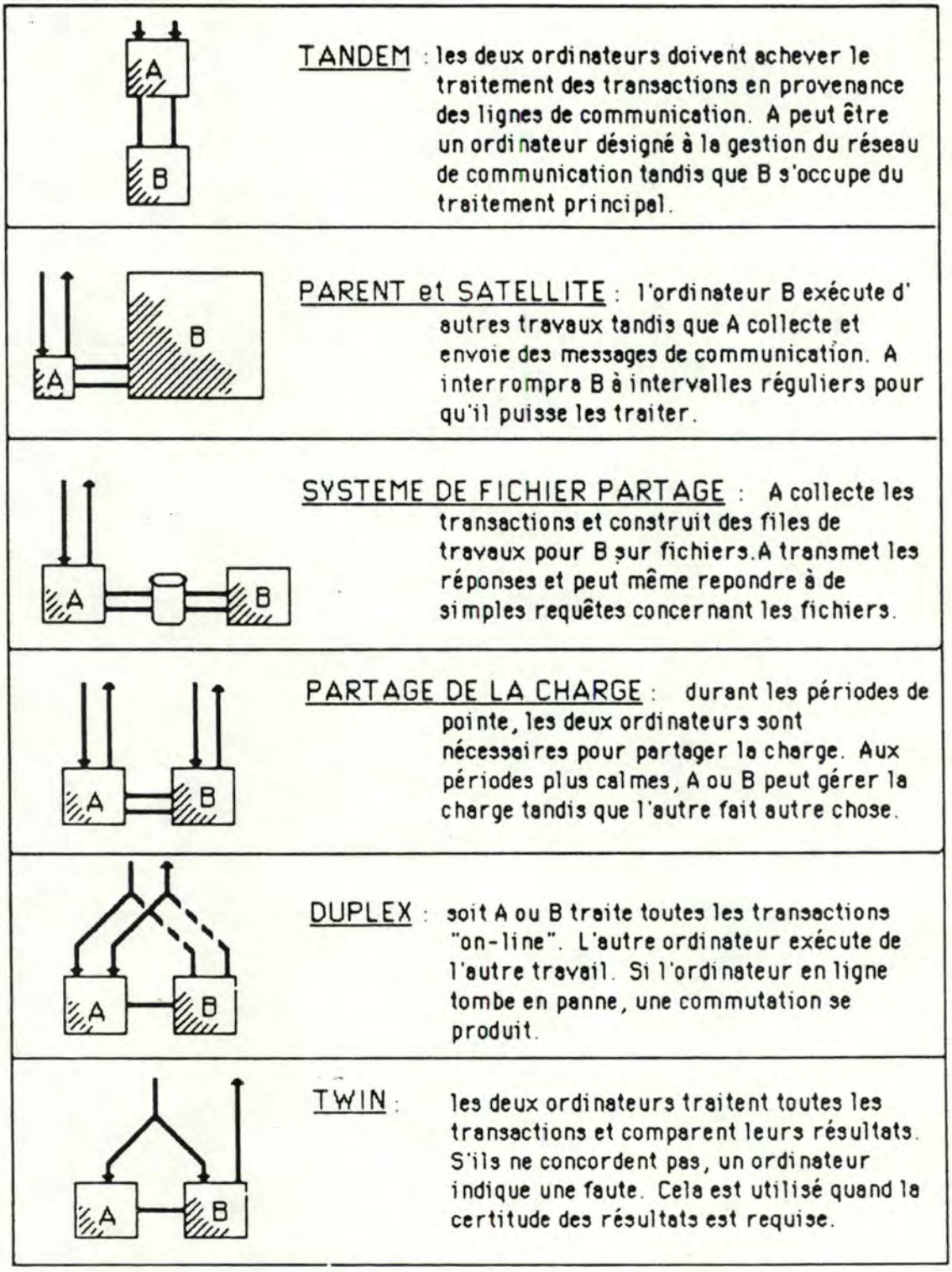
frontière entre un système fonctionnant et un système ne fonctionnant pas n'est pas si tranchée. Si une unité tombe en panne, la technique de la "dégradation amiable" consiste à tirer le maximum d'une mauvaise situation et à continuer d'exécuter une partie des tâches. Le système d'exploitation ordonne le "fonctionnement partiel" et le système peut continuer sa marche boîteuse jusqu'à ce que le composant défaillant ait été remis en état. La mesure la plus intéressante à calculer devient la "disponibilité fonctionnelle" c'est-à-dire la probabilité qu'une transaction soit traitée selon certains critères, si une unité tombe en panne .

I.3.B. La "Fault Tolerance Technique" dans le domaine hardware

---

La "Fault Tolerance Technique" est basée sur l'hypothèse qu'il est plus économique de construire des systèmes redondants avec des composants de qualité standard que de se braquer sur une fiabilité extrême des composants (Kopetz, 9). Dans le domaine hardware, nous envisagerons trois modes de prise en charge de la fiabilité.

Une première manière d'améliorer la fiabilité d'un système temps réel consiste à **dédoubler** tous ses composants critiques, de façon à ce que si un ordinateur tombe en panne, un ordinateur de secours le relaie immédiatement. Le doublement est naturellement coûteux et le choix de cette solution dépend de la fiabilité exigée. Le système d'exploitation doit déceler la nécessité d'effectuer une commutation de n'importe quel élément, y compris l'ordinateur lui-même. Il doit également ordonner la commutation, qu'elle soit exécutée manuellement par l'opérateur ou qu'elle soit automatique. Cette dernière pose de nombreux problèmes de programmation : elle peut survenir quand le système d'exploitation a à moitié terminé une opération en cours, quand un fichier a été lu mais pas complètement mis à jour... La mise à jour des fichiers doit être effectuée avec précaution pour éviter que le fichier soit mis à jour deux fois et ne devienne incorrect. De plus, le système obtenu n'est pas 100% fiable car l'équipement remplaçant peut également tomber en panne. Il serait plus fiable de recourir à un triplement des composants critiques mais cette solution est encore plus coûteuse à moins que les ordinateurs de réserve n'effectuent d'autres travaux lorsqu'ils sont en attente. (voir figure I.2.)



(tiré de MARTIN, 11, p 57)

Figure I.2. : Exemples de configurations redondantes

La présentation descendante des différentes configurations sur cette figure correspond à une fiabilité croissante :

- Dans la configuration *tandem* ou *parent-satellite,* si B tombe en panne, l'ordinateur A doit être capable, dans les limites de ses capacités, de se débrouiller : soit il enregistre les transactions sur ses fichiers attendant que B revienne à la rescousse, soit il essaie de traiter certains messages en se basant sur les fichiers tenus par B. Si A tombe en panne, les communications seront coupées à moins que certaines puissent être commutées sur B. Cette configuration n'est pas très satisfaisante du point de vue temps réel mais elle est meilleure qu'un simple ordinateur et moins chère que la configuration *duplex.*
- Dans la configuration *système de fichier partagé* ("shared file system"), A doit être capable de traiter la plupart des opérations temps réel. Si A tombe en panne, seules les transmissions *off-line* peuvent être utilisées à moins que des lignes soient commutées sur B.
- Dans la configuration *partage de la charge* ("parallel or load-sharing"), les ordinateurs A et B sont nécessaires pour venir à bout des moments de pointe. Sous charge moins forte, l'un des deux peut vaquer à d'autres occupations comme la maintenance préventive et les tests de programmes. Si un des deux tombe en panne aux heures de pointe, seules certaines fonctions temps réel urgentes peuvent être réalisées.
- Dans la configuration *duplex,* tout le matériel nécessaire aux travaux temps réel est dédoublé pour qu'une commutation puisse se produire si le système en ligne ("on-line") tombe en panne. Une illustration en est donnée à la figure I.3. par le Système SACHEM d'UBS. Tous les organes vitaux du système y sont dédoublés (CPU, disques, multiplexeurs, ...). La commutation s'y effectue selon le principe du "Hot Standby" c'est-à-dire que pour veiller à la parfaite synchronisation des deux systèmes, le disque du système de secours (appelé le passif) est constamment mis à jour pour toutes les modifications subies par le disque du système en ligne (appelé l'actif). Les deux systèmes se contrôlent mutuellement c'est-à-dire que si l'actif ne répond pas à un test de sa présence lancé par le passif, ce dernier procède au basculement.
- Dans la configuration *twin,* une haute fiabilité est assurée parce que certains des résultats sont vitaux.

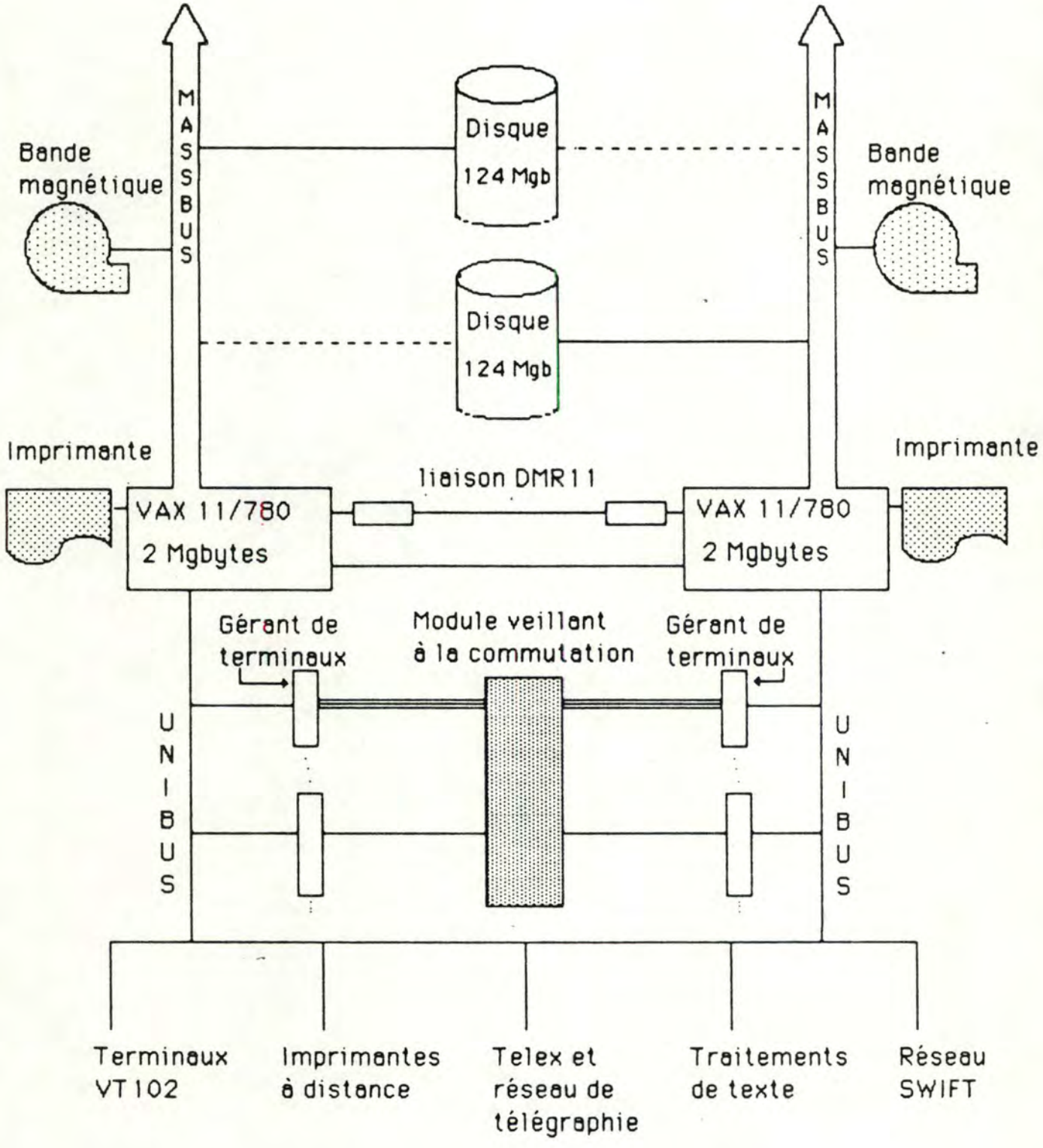


Figure I.3. : Architecture DUPLEX du système SACHEM d'UBS

La configuration la plus fiable s'appelle *polymorphic system*, elle utilise des unités relativement petites, les mémoires, les contrôleurs de canaux, les unités de traitement... toutes pouvant être remplacées par une unité similaire. Plus le nombre de chemins alternatifs pour mettre en communication deux unités est important, plus la fiabilité sera élevée. La difficulté se reporte sur le système d'exploitation qui doit gérer toutes ces possibilités de chemins.

Les "**fallback procedures**" constituent une deuxième solution dans laquelle le système recourt à un autre moyen de traitement pour enrayer la faute plutôt que de s'arrêter de fonctionner brutalement. Il travaille en service dégradé en mettant en attente les transactions ne nécessitant pas de temps de réponse draconnien.

Bien que ne relevant pas exclusivement du domaine hardware, nous pouvons néanmoins citer comme dernière solution les "**bypass procedures**" qui consistent en des moyens de détresse mis à la disposition des utilisateurs de terminaux pour qu'ils puissent malgré tout continuer à travailler. Par exemple, l'ordinateur central peut effectuer des impressions de ces fichiers de comptes de clients et envoyer dans les différentes agences bancaires durant la nuit le relevé de tous les comptes supérieurs à 10000 FB. Si le lendemain, les lignes de communication sont coupées, sur base de ces listings, l'employé de banque pourra déjà satisfaire les clients disposant de plus de 10000 FB.

I.3.C. La "Fault Tolerance Technique" dans le domaine software

---

La programmation de systèmes temps réel pose de nombreux problèmes : les différents messages qui atteignent l'ordinateur et ce de manière aléatoire, exigent des programmes différents la plupart du temps, une remise en cause continuelle de l'allocation de la mémoire, un ordonnancement de leurs traitements suivant un système de priorité. Le degré de multiprogrammation augmente aussi les difficultés d'écriture et d'essai des programmes. En effet, il est difficile de prévoir toutes les combinaisons d'événements pouvant se produire et de tester que le système répond correctement dans une situation donnée. Comme il est impossible d'obtenir

une certitude à priori de la capacité de "fault avoidance" du système, étant donné la complexité de celui-ci, le système d'exploitation temps réel et les programmes d'application doivent traiter du problème de la détection des erreurs de software et de leur récupération en concevant cependant des moyens de protection des fichiers et des mémoires. La plupart du temps, ce traitement est non systématique et des auteurs (Kopetz, 9) voudraient développer une méthodologie pour le traitement unifié des différents types d'erreurs.

Dans un système tolérant les fautes, il s'agit d'abord de détecter les erreurs : ceci suppose à chaque étape d'un processus, des contrôles de vraisemblance sur base d'un critère d'acceptation.

Pour ce qui est de la prise en charge de ces erreurs, une première méthode appelée "**forward error recovery**" corrige les symptômes en appliquant des sections de programme appropriées pour contrecarrer les effets néfastes et essaie de continuer le processus. Cette façon de procéder est souvent adoptée dans les systèmes temps réel car elle est rapide et permet de poursuivre le fonctionnement du système même dans un mode dégradé.

Cependant, cette méthode ne s'attaque pas à l'origine du problème et une autre méthode appelée "**backward error recovery**" se décompose en :

- un diagnostic de l'erreur c'est-à-dire trouver la faute ayant provoqué l'erreur ;
- une reconfiguration du système c'est-à-dire remplacer les modules logiciels fautifs par des modules correspondants conçus différemment ;
- et une restauration d'un état c'est-à-dire revenir au dernier point de reprise où l'état de l'information sauvée dans un vecteur était encore correct.

I.4. De l'architecture centralisée à l'architecture distribuée

La forme la plus simple que peut prendre un système en temps réel comporte un dispositif semblable à une machine à écrire qui envoie un message à l'ordinateur. Celui-ci interrompt son traitement en cours, prend en charge le message, envoie éventuellement une réponse, puis reprend son traitement. L'ordinateur peut être doté d'un fichier sur mémoire à accès aléatoire et la machine peut, via ses messages, mettre à jour ou consulter celui-ci.

Un système un peu plus compliqué comprendra plusieurs terminaux ou machines à écrire, qui sont reliés à une mémoire tampon ou à une ligne commune de transmission. Un seul terminal peut envoyer un message à la fois. (figure I.4.)

Le degré suivant de complexité consiste à avoir plusieurs lignes de transmission. (figure I.5) Avec certains types d'équipement, la ligne de transmission peut être soit directement connectée à l'ordinateur, soit à un multiplexeur ou à une unité de contrôle des lignes pour gérer les lignes, recevoir ou transmettre les messages en parallèle (figure I.6.).

Certains systèmes utilisent plusieurs ordinateurs lorsqu'un seul ne peut suffire par manque de capacité, de vitesse ou pour des raisons de fiabilité (voir, par exemple, la figure I.7. qui illustre la configuration *satellite et parent* du point I.3.B).

En résumé, ces configurations sont caractérisées par la liaison à un site *central* de terminaux d'accès ou d'organes de mesure disposant éventuellement de possibilités de traitement local. De telles structures existent depuis les débuts de l'informatique et encore actuellement dans le cadre de systèmes simples avec une configuration minimale d'ordinateurs pour traiter des applications géographiquement réparties.

A titre d'exemple, envisageons le cas d'une société de moyenne dimension disposant d'un centre de calcul important pour établir des plans de production et effectuer d'autres tâches à temps de réponse très court. Des usines filiales disposeraient de petits ordinateurs pour les travaux routiniers

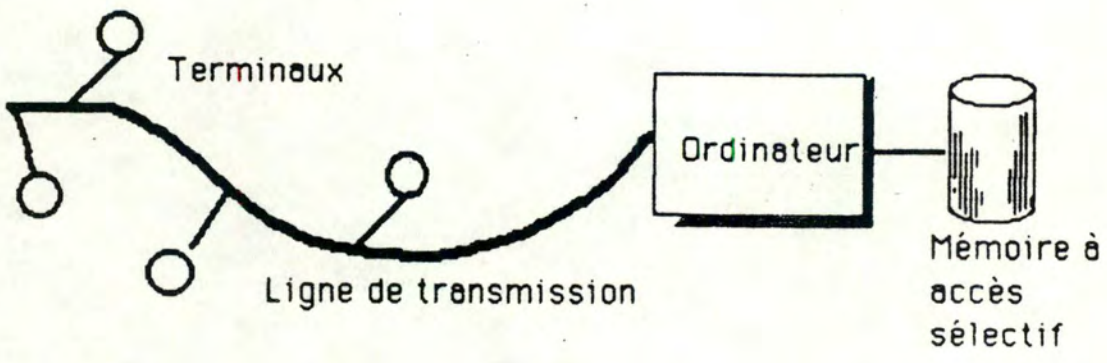


Figure I.4. : Système à une ligne de transmission

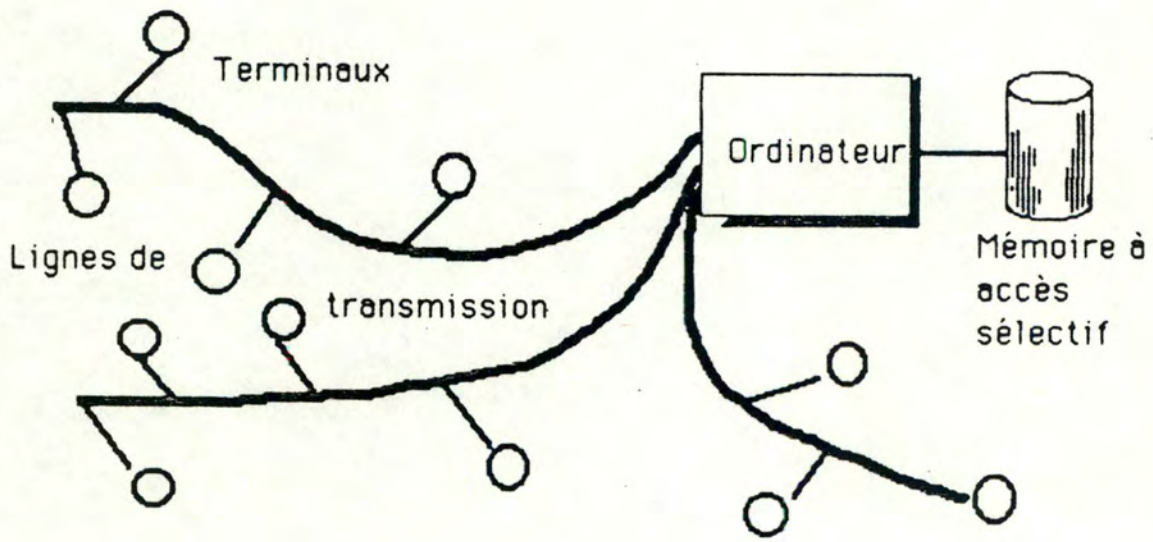


Figure I.5. : Système à plusieurs lignes de transmission

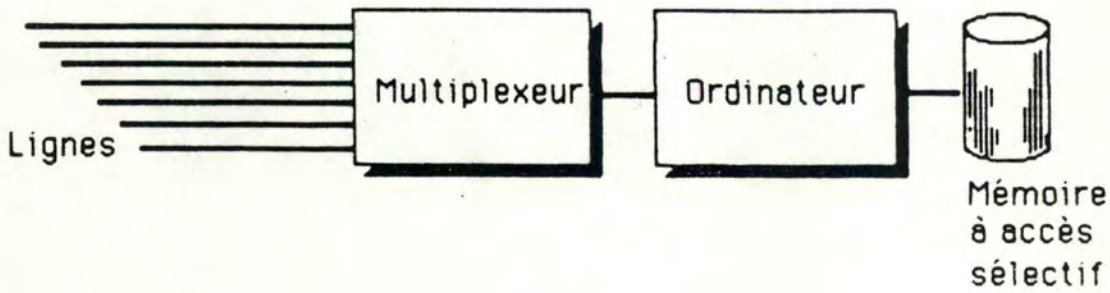


Figure I.6. : Système doté d'un multiplexeur pour la gestion des lignes

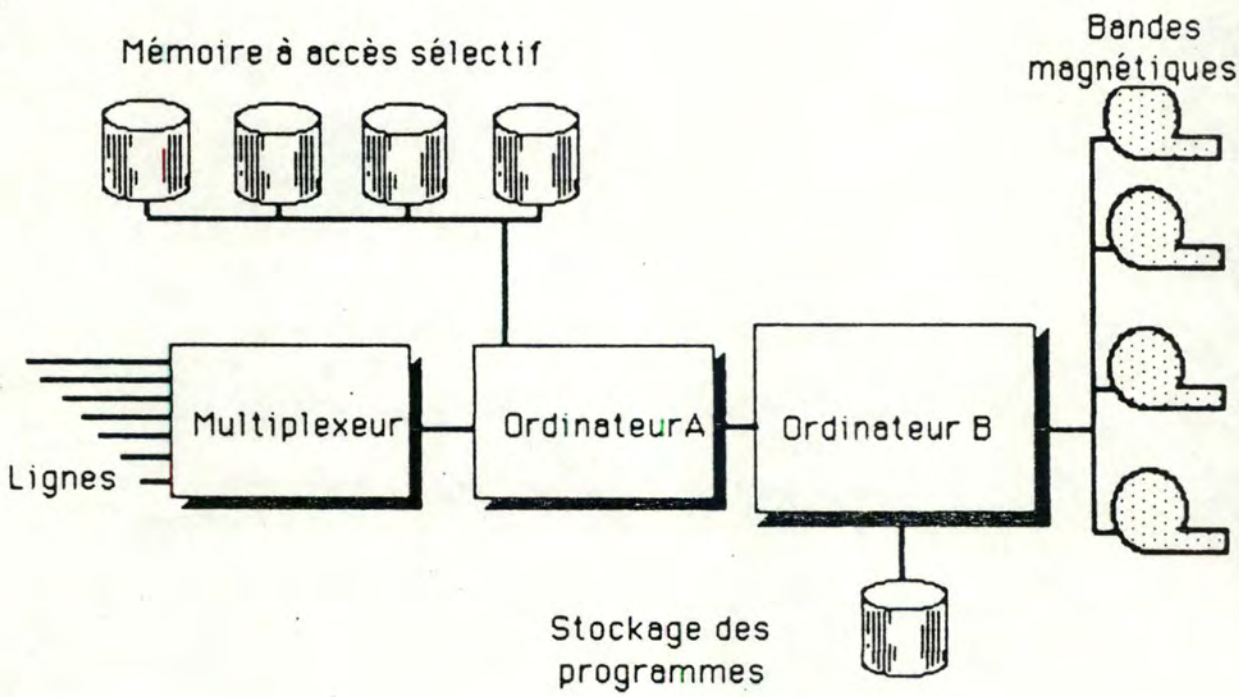


Figure I.7. : Système à plusieurs ordinateurs

comme la facturation et l'établissement des salaires. Cependant, dès qu'une tâche requiert de gros moyens de calcul, la filiale la fera exécuter au centre de calcul et en attendra l'impression des résultats sur une de ses imprimantes.

Même si au début, les configurations étaient principalement *centralisées*, les systèmes informatiques ont cependant évolué, sous l'influence de deux facteurs :

- la recherche d'une meilleure adéquation des architectures aux applications traitées (voir le point I.4.A) ;
- les progrès de la technologie qui ont rendu possible cette adéquation (voir le point I.4.B).

Et ainsi, dans une étape ultérieure, des soucis d'économie (mise en commun de ressources ou d'informations) amenèrent à connecter par des réseaux de communication des installations autonomes existantes capables de fournir des services généraux ou spécialisés pour obtenir des *systèmes informatiques répartis ou distribués*.

A cet égard, nous adhérons à la définition proposée par le groupe CORNAFION qui, contrairement aux théories anglo-saxonnes, considèrent équivalents les termes *répartition* et *distribution*. Voici la définition qu'ils proposent : "...ensembles informatiques autonomes constitués d'unités de traitement ou de stockage interconnectées par un système de communication." (Pour une identification de tous les problèmes rencontrés lors de la construction de tels systèmes, on consultera Cornafion, 3).

Il est courant de concevoir comme un tout un système informatique réparti, c'est-à-dire l'ensemble constitué par le matériel, le logiciel et le système de communication. Cependant, outre la dispersion géographique des équipements, certains aspects peuvent être centralisés sans que pour autant les organes sièges de ces tâches occupent une position privilégiée de maître dans la structure du système :

- soit le *contrôle* c'est-à-dire l'endroit où s'effectue la supervision globale du système ;
- soit les *fonctions* c'est-à-dire la position des activités ou des responsabilités au sein de la structure du système.

I.4.A. Adéquation des architectures aux applications traitées

Des applications fonctionnellement et géographiquement réparties s'adaptent très bien à ce genre de structure distribuée et extensible. Voici quelques exemples qui mettent en jeu un ensemble de services fournis par différents systèmes connectés :

- les systèmes de commande de processus industriels ;
- les systèmes de gestion administrative ou documentaire ;
- les bases de données réparties.

I.4.B. Evolution des microprocesseurs et influence sur l'architecture des systèmes

Le progrès des techniques de transmission de données qui permettent l'échange des informations sur une gamme étendue de distances et de débits et la baisse du coût des processeurs et mémoires ont également représenté un facteur décisif dans l'apparition de ce nouveau type de structure.

L'évolution de la technologie des circuits intégrés permet de multiplier chaque année, par un facteur 1.6, la complexité des composants digitaux (microprocesseurs, mémoires, contrôleurs d'entrée/sortie) (figure I.8.). L'état actuel de la technologie, en 1980, autorise l'intégration d'une unité centrale de miniordinateur 16 bits de milieu de gamme en un seul circuit. Les projections dans le futur montrent que, jusque dans les années 90, la complexité des circuits croîtra vers plusieurs dizaines de millions de transistors permettant la réalisation monolithique de la majorité des ordinateurs de moyenne et grande puissance. (Pour plus de détails, on consultera Anceau, 1)

L'évolution de la complexité découle de plusieurs faits :

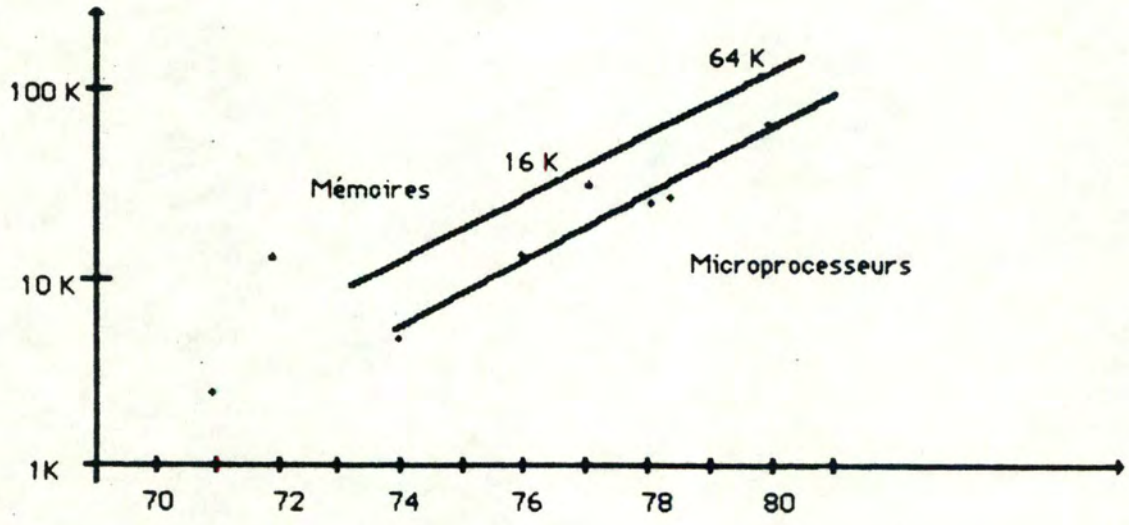


Figure I.8. : Evolution de la complexité des circuits intégrés

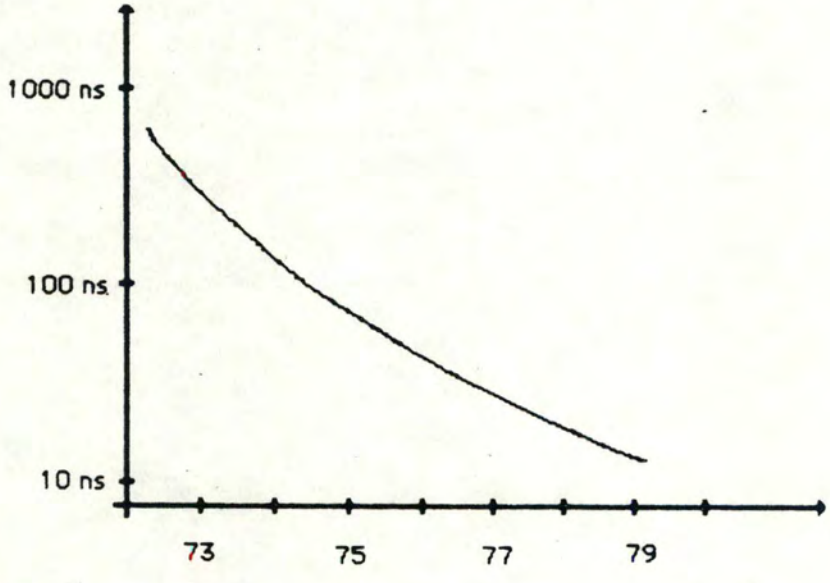


Figure I.9. : Evolution de la vitesse des mémoires MOS 1 K

(tiré de ANCEAU, 1, pp 228 et 229)

- la diminution de la taille de la géométrie des éléments du circuit intégré ;
- l'augmentation de la surface maximale des circuits réalisables à partir de laquelle le rendement chute en dessous de ce qui est économiquement viable ;
- la diminution du nombre de composants (transistors) pour réaliser une fonction déterminée ;
- la croissance continuelle de la vitesse de fonctionnement de la logique interne. De ce fait, les circuits MOS peuvent rivaliser en vitesse avec leur équivalent en circuits TTL de faible et moyenne intégration (figure I.9.)

L'offre de circuits de plus en plus complexes à des prix très bas, en regard de l'électronique traditionnelle qu'ils remplacent, amènent les concepteurs de systèmes informatiques à revoir leurs comportements. Les structures centralisées destinées à assurer le partage et la rentabilisation de la ressource chère, précieuse et fragile que constituait l'ordinateur, peuvent céder le pas à des structures distribuées, constituées par l'interconnexion de petits sous-systèmes simples plus ou moins dédiés à une partie du problème.

I.4.C. Le choix de l'architecture dans une organisation

Nous avons présenté un système informatique réparti comme un ensemble d'équipements intelligents sur le même pied structurel mais sur lequel certaines tâches peuvent se voir allouées exclusivement à un seul ordinateur. Pour mieux saisir les relations entre les notions de centralisation, de décentralisation et de répartition, intéressons-nous au cas d'une organisation pour qui se pose le choix d'une politique informatique sur base des avantages et inconvénients que représentent les différentes solutions.

Un système *centralisé* au sens strict est un système dans lequel chaque élément est centralisé. Il dispose d'un centre hardware, d'un seul groupe d'opérateurs, de programmeurs, de support d'administration, de diagnostic des problèmes du système et de maintenance du logiciel/du

hardware. Un système *décentralisé au sens strict* est, quant à lui, représenté par une multitude de systèmes centralisés autonomes. Un système *décentralisé au sens large* ou système *réparti* (*distribué*) est un système décentralisé au sens strict dont les éléments sont connectés par un réseau de communication.

Dans le cadre d'une organisation (King, 8), le choix dans le déploiement des ressources informatiques repose sur la structure organisationnelle sous-jacente. Si le contrôle et la plupart des fonctions sont centralisées, le système informatique a de grandes chances d'adopter la même structure. Le choix est également influencé par les besoins des utilisateurs ainsi que la volonté de la direction de contrôler les coûts et les utilisations du matériel.

Les **avantages d'un emplacement centralisé** des équipements, qui aboutit souvent à une centralisation du contrôle et des fonctions des ordinateurs, sont le respect de la cohérence des opérations qui y sont effectuées et les économies d'échelle résumées, entre autres, par la loi de Grosch, qui affirme que la puissance d'un ordinateur est proportionnelle au carré du coût de ce dernier. La centralisation permet un contrôle *"top-down"* des coûts des équipements, de l'usage de l'ordinateur, de sa croissance et parfois de l'information traitée.

La nécessité de fournir une réponse à des situations inusitées dans un laps de temps très court, la dispersion géographique des applications à soutenir qui, dans un système centralisé, entraîne des coûts importants d'apport de l'information aux différents centres intéressés de l'organisation ainsi que le volume croissant des opérations amenèrent les auteurs à penser décentralisation vers le milieu des années 70. Pourquoi des ordinateurs plus puissants n'ont-ils pas été construits pour rencontrer les besoins de l'installation ? A tout instant, vu la technologie disponible, il existe une capacité de processeur au-delà de laquelle une augmentation des performances n'est possible qu'au prix d'une augmentation inacceptable des coûts.

Parmi les **avantages de la décentralisation** peuvent être cités, entre autres :

- la facilité d'accès aux équipements ;
- l'implication de l'utilisateur dans la conception et la modification du

système pour que ses besoins soient mieux rencontrés ;
- la plus grande autonomie des différents départements d'utilisateurs ;
- la mise à profit par les utilisateurs de la technologie avec plus d'efficacité pour accroître le rendement de leur département ;
- l'adaptation de la puissance de calculs aux besoins des utilisateurs qui englobe aussi bien l'expansion que l'évolution de l'ensemble au fil du temps;
- l'amélioration de la fiabilité.

La décentralisation est orientée vers une amélioration *"bottom-up"* de la productivité. Elle peut conduire à la prolifération d'ordinateurs coûteux, à la difficulté pour la Direction d'obtenir des informations pertinentes sur les différents départements, à la prolifération de centres de calcul incapables de s'entendre sur une politique commune avec le risque de déboucher finalement sur l'anarchie la plus complète.

Dans le milieu des années 60-70, beaucoup d'informaticiens pensaient que la centralisation épargnait de l'argent et l'arrivée des minis ne changea en rien cette impression. Par la suite, on avait pu croire que le simple avénement des minis et des micros sur le marché aurait changé les règles du jeu et rendu la décentralisation à la fois plus faisable et plus économique. Il ne faut cependant pas oublier que les coûts de l'informatique dépassent, de loin, la simple acquisition des processeurs ; les autres frais à prendre en considération sont :

- l'achat de matériel périphérique : chaque ordinateur est entouré de disques et d'imprimantes ;
- la conception ou l'achat de logiciels : avec l'emploi accru des ordinateurs, le nombre d'applications opérationnelles croît également et l'offre aux utilisateurs de nouvelles possibilités les amènent à définir des besoins de plus en plus exigeants ;
- l'embauche de personnel d'exploitation et de conception : la croissance du nombre de logiciels provoque une demande en personnel informatique plus forte que l'offre sur le marché ; il en résulte une élévation des salaires ;
- la maintenance : elle exige un bonne documentation chère à produire ; Boehm affirme que pour 1985, 70% des dépenses de logiciel seront consacrées à ce poste (Boehm, 2) ; dans cette catégorie, on peut aussi évoquer la réécriture de beaucoup de systèmes développés sur d'anciennes architectures pour s'adapter aux nouveaux produits ;

- la formation des utilisateurs.

Toutefois, nous pouvons affirmer que l'économie représentée par la décentralisation sur la centralisation est fonction du degré de coordination avec lequel la nouvelle solution est introduite dans l'entreprise. (figure I.10.)

Sans plan global, la mise en oeuvre de systèmes tout-à-fait nouveaux étend le nombre d'applications que l'organisation peut supporter sans compter que ces nouveaux systèmes peuvent entrer en contradiction avec les autres départements. Pour éviter ce risque de chaos et améliorer d'autant plus les services aux utilisateurs, des directeurs d'entreprise ont pensé à la solution **répartie (distribuée) ou décentralisée au sens large** qui, pour rappel, est un système décentralisé au sens strict dont les éléments sont connectés par un réseau de communication.

Citons quelques **avantages** non négligeables :

- les utilisateurs pourront se partager les machines pour réaliser un meilleur emploi des ressources, travailler sur des bases de données et accéder à n'importe quel équipement du système ;
- les utilisateurs pourront coopérer à la réalisation d'une tâche commune et intégrer leur travail au sein de l'organisation ;
- l'emploi de la redondance assurera aux utilisateurs de terminaux une plus grande disponibilité du système ; en effet, supposons des systèmes locaux servant chacun une partie des terminaux et reliés à un système central par un réseau de télécommunication ; les transactions générées localement sont traitées sur le site central mais si le système central ou les lignes tombent en panne, les sites locaux peuvent prendre la relève sur base d'informations agrégées.

Immanquablement, ce réseau réduira la tendance des unités décentralisées à adopter des équipements incompatibles avec les autres, améliorera les relations entre les départements, facilitera les contrôles et la gestion des ressources informatiques par l'organe de direction. On peut observer que la notion de contrôle centralisé n'est pas en désaccord avec la notion de système distribué.

Cependant, d'autres problèmes se posent :

- le manque de standards dans les protocoles de communication

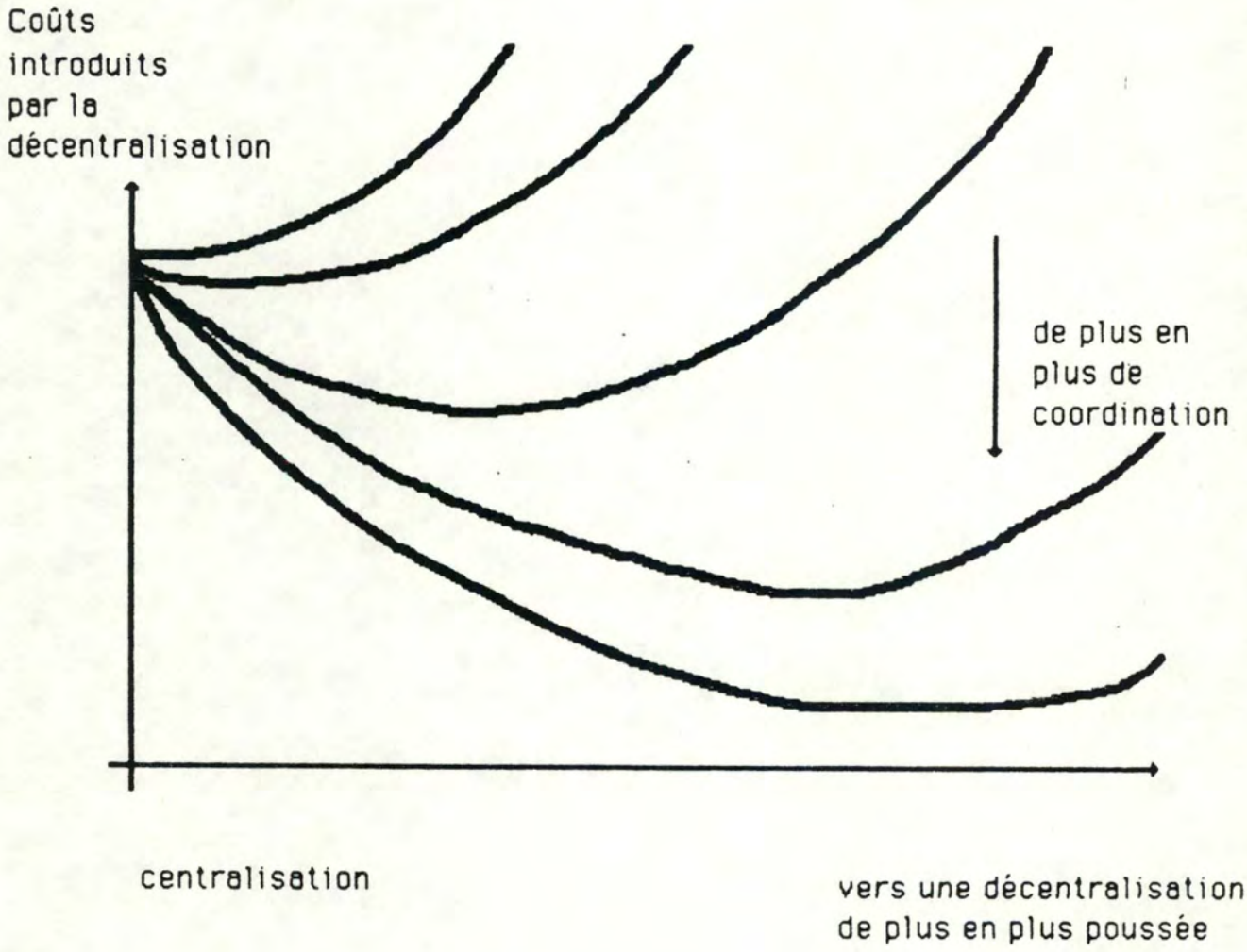


Figure I.10. : Impact de la coordination sur les coûts introduits par la décentralisation par rapport à la centralisation

développés par les constructeurs, les structures de fichier, les bases de données et les systèmes d'exploitation ;
- qui assurera le contrôle ? un organe central ou des organes décentralisés ;
- qui pourra s'y connecter ?- la difficulté pour les utilisateurs ayant leur propre système de stockage de divulguer leurs informations aux autres ;
- le temps d'apprentissage et d'adaptation des utilisateurs aux possibilités offertes par le réseau.

I.4.D. Le choix de l'architecture dans un système temps réel

Dégageons-nous maintenant de l'environnement spécifique d'une organisation pour retourner à notre problème du temps réel où les deux préoccupations sont le temps de réponse (voir point I.2) à respecter sinon à améliorer et la fiabilité du système (voir point I.3).

Les applications traitées par ordinateur peuvent généralement être découpées en sous-systèmes, relativement indépendants, tels que les interactions entre éléments d'un sous-ensemble soient beaucoup plus fréquentes que les interactions entre sous-ensembles distincts : c'est la propriété de *localité* (ou le principe de quasi-décomposabilité de Simon). Dans la mesure où la baisse du coût des processeurs et des mémoires le permet, il est tout naturel de calquer la structure d'un système informatique sur celle de l'application qu'il traite et de décentraliser les éléments d'un système informatique pour les rapprocher du lieu de production ou d'utilisation des informations. La localité des traitements entraîne des échanges peu fréquents entre sous-systèmes, ce qui permet de réduire les coûts de communication. Parfois, la communication peut être entièrement évitée et des lignes utilisées à temps plein peuvent être remplacées par des lignes utilisées à temps partiel ou tout autre moyen de communication.

De plus, si l'on analyse le **temps de réponse**, le traitement en local par un sous-système d'une opération est accéléré par rapport à un système centralisé où le traitement souffre des retards subis au niveau du réseau de communication et des files d'attente rencontrées au niveau central. D'autres

pourraient cependant argumenter qu'avec un accroissement des performances des processeurs et des mémoires résultant des progrès des composants intégrés (voir point I.4.B) et une utilisation de moyens de communication à plus large bande, la centralisation pourrait également assurer un temps de réponse acceptable. Cette solution ne mettrait, cependant, pas à profit le principe de localité présent dans de nombreuses applications mais permettrait des économies d'échelle (voir loi de Grosch dans le point I.4.C). Le tout est de voir, pour un niveau de performance donné, la note de frais à payer en retour pour chacune des solutions proposées.

Si l'on envisage l'aspect **fiabilité hardware**, la localité des traitements augmente l'indépendance mutuelle des sous-systèmes et tend donc à rendre chaque sous-système plus résistant aux défaillances des autres en supposant, bien sûr, que la fiabilité de chacun d'eux est suffisante. La décentralisation assure une plus grande disponibilité du système dans la mesure où un organe défaillant peut être remplacé plus commodément par un organe assurant une fonction équivalente. Un système centralisé est souvent complexe. Or, plus un système est complexe, plus difficile il est à réparer à cause de l'interdépendance des éléments qui le composent. En outre, la modularité d'un système décentralisé permet une modification et une expansion incrémentale du système plus faciles (par reconfiguration ou par addition de nouveaux processeurs) que dans un système centralisé qui peut montrer certaines limitations à l'adjonction de nouveaux éléments comme une baisse globale de la performance par ajout de nouveaux disques.

Enfin, si l'on considère la **fiabilité software**, la structure modulaire des sous-systèmes et l'emploi d'une série de petits et simples ordinateurs garantissent la simplification des différents logiciels. Dans la plupart des cas, les systèmes d'exploitation utilisés sur les minis et les micros sont de un à cent fois moins conséquents en terme de nombre de lignes de code nécessaires pour implémenter leurs fonctions que leurs contreparties sur les grosses architectures. En effet, une large part de ces derniers systèmes d'exploitation est consacrée à la gestion du partage de la machine et leur nécessaire optimisation rend leur conception particulièrement difficile. Là où réside la simplicité, les erreurs sont plus facilement dénichées et la fiabilité logicielle du système global accrue. Il s'agit en quelque sorte d'obtenir une baisse du coût de développement, la simplicité et la lisibilité par la multiplication du matériel.

Un problème soulevé par la distribution est cependant la tenue d'une

base de donnée répartie et la mise à jour de plusieurs de ses copies car on ne peut admettre qu'un sous-système réalisant une action autonome doive demander l'accès à une base de données centrale, accroissant ainsi le niveau de trafic sur le réseau et le ralentissant pour l'exécution de tâches plus prioritaires. Cette prolifération des copies entraîne des problèmes en cas de reprise suite à un accident. En effet, où se trouve l'information la plus à jour ? En outre, à quel moment les informations présentes dans le système étaient-elles encore cohérentes avec la réalité ?

Indépendamment des avantages qu'une solution centralisée pourrait offrir, l'accent sera porté dans la suite du mémoire sur la distribution (ou répartition) et nous montrerons en quoi elle rencontre les objectifs d'un système temps réel.

Il est intéressant de noter que cette optique est également celle suivie par le CERN. Nous illustrerons dans les chapitres II et III sa politique en matière de systèmes temps réel par la présentation de ses applications dans un environnement contrôlant un processus technique.

Chapitre II : LE SUPER SYNCHROTRON A PROTONS ET SON SYSTEME DE CONTROLE

Situé au CERN, le Super Synchrotron à Protons ou plus communément appelé SPS est aujourd'hui le plus grand accélérateur de particules du monde avec celui de FERMILAB aux Etats-Unis. Pour veiller à son parfait fonctionnement, il faut assurer un contrôle drastique de ses équipements. Ceux-ci sont répartis le long de l'anneau de 2,2 km de diamètre ou concentrés dans d'importants Bâtiments Auxiliaires situés à 1,1 km d'intervalle et dont chacun supervise un des six sextants dont est constitué l'anneau. Il ne faut pas oublier le contrôle des lignes d'injection et d'extraction de particules avant leur projection sur des cibles fixes pour les expériences physiques, dont les équipements sont regroupés dans des bâtiments placés le long des lignes de transfert. (figure II.1.)

Ce chapitre se propose, dans un premier temps (voir point II.1), de montrer comment une architecture distribuée répond, dans le chef des utilisateurs du SPS, aux besoins temps réel rencontrés par le système de contrôle de l'accélérateur. Dans un deuxième temps, une description plus détaillée du système mis en place pour conduire ce vaste processus industriel sera donnée. Nous parlerons d'abord des sous-réseaux en étoile (voir point II.2), puis de leur mise en communication par un système de transfert de messages mettant en cause les centres de ces étoiles (voir point II.3). Il nous restera enfin à évoquer les feuilles des étoiles ou plus précisément les satellites (voir point II.4).

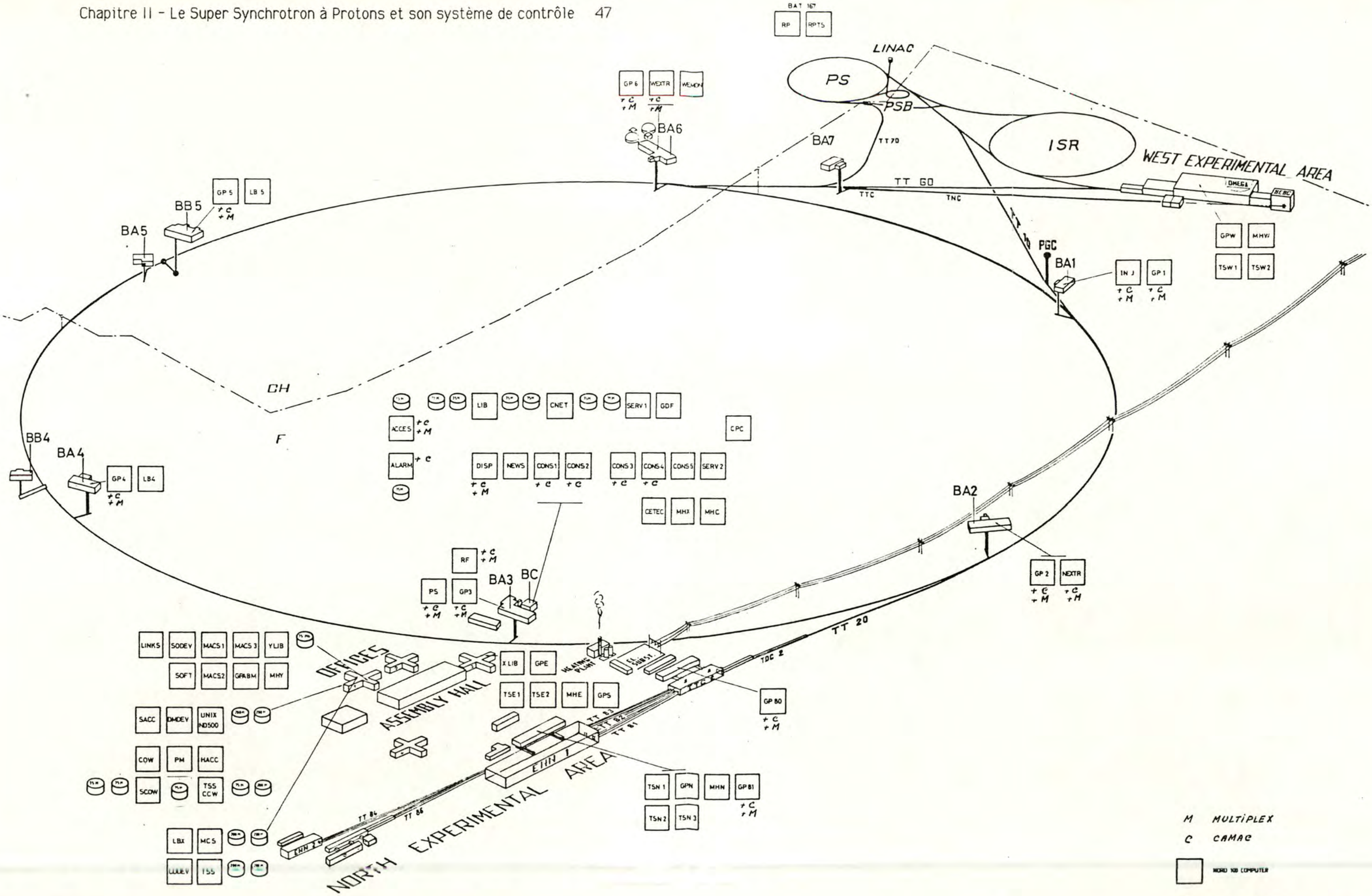


Figure II.1. : Répartition des ordinateurs autour de l'accélérateur SPS

II.1. Le système de contrôle centralisé en temps réel du SPS sur architecture entièrement distribuée

> To make a good aeroplane, one must "simplificate and add lightness"
>
> Barnes Wallis

Au SPS se pose le problème du contrôle en temps réel d'un vaste processus industriel. Le temps de communication constitue le point sensible dans la construction d'un tel système. Il s'agit en effet de saisir et de traiter plusieurs dizaines de milliers de valeurs analogiques et digitales informant de l'état des équipements en un laps de temps très court. On peut dire, en simplifiant, que le SPS est contrôlé par un réseau d'ordinateurs-satellites qui permet, de par sa nature distribuée, d'épargner le coût des connexions des équipements vers un point centralisateur ainsi que d'éviter au maximum les communications entre satellites. De plus, certaines fonctions sont centralisées : l'accélérateur SPS et ses équipements sont pilotés à partir d'une salle de contrôle unique par une petite équipe d'ingénieurs et techniciens, cette salle pouvant être déplacée et reliée à n'importe quel point du réseau.

Il ne s'agit pas d'un système de contrôle distribué où les procédures de pilotage sont regroupées dans un ordinateur Centre de Décision unique gérant des systèmes électroniques esclaves, non programmables et câblés en fonction de l'appareillage qui leur est associé.

La méthode de pilotage choisie repose, au contraire, sur des satellites intelligents qui sont à la fois Centre de décision, Esclave et Base de Données. Ce choix offre, en effet, des avantages non négligeables :

- l'usage extensif du **traitement en parallèle** par la définition d'un langage spécifique (voir point II.4.A) donnant la possibilité d'exprimer le parallélisme au sein des procédures de contrôle ; les moyens de calcul placés à proximité des sources d'information et des fonctions à

réaliser fournissent un temps de réponse aux interruptions plus court et en simultané dans plusieurs endroits ; dans ces conditions, si un algorithme multi-ordinateurs résidant dans un satellite temporairement maître de la salle de contrôle et impliquant certains équipements contrôlés par des satellites présents dans les Bâtiments Auxiliaires est lancé par un opérateur (pour mesurer la position du faisceau dans les différents sextants de l'anneau, par exemple), cet algorithme enverra pour exécution dans ces satellites temporairement esclaves les commandes nécessaires et en attendra les résultats retournés sous forme synthétique ;

- au niveau des satellites répartis le long de l'accélérateur, la scission du contrôle en **sous-ensembles cohérents** tels que la communication s'effectue dans un **groupe** et peu entre groupes ; chacun d'entre eux est assigné à un mini-ordinateur dont la tâche relève de la surveillance locale et ce sans relation nécessaire avec la salle de contrôle sauf pour rendre compte d'une erreur ; cette scission offre d'autres avantages :

  - chacun des sous-ensembles est plus petit que le système tout entier, donc plus facile à maîtriser ; chaque sous-système est aussi plus résistant aux défaillances des autres sous-systèmes (voir I.4.D.) ;
  - la mise en oeuvre peut être progressive, par sous-ensemble à la fois, jusqu'à l'intégration global ;

  ce souci de diminuer la quantité d'information circulant entre groupes amène à un pré- ou post-traitement des données localement chaque fois que possible et à leur transfert sous une représentation synthétique aux autres sous-groupes, quand nécessaire ; les données à utiliser localement ne traversent pas le réseau, elles sont réduites et gardées sur place dans des bases de données réparties qui reflètent la réalité à tout moment ; en effet, si les données étaient centralisées, entre deux mises à jour successives, la base de données pourrait se trouver dans un état incohérent ;

- l'**expansion** du SPS en évolution perpétuelle par l'ajout aisé de nouveaux modules sans perturbation du reste du système.

Le SPS développe une stratégie de contrôle utilisant un réseau d'ordinateurs répartis géographiquement et fonctionnellement.

Le découpage est **géographique** lorsque des groupes de mêmes types d'équipements sont affectés à un ordinateur parce qu'ils se trouvent à proximité de celui-ci et non en raison de leurs fonctions. C'est le cas au SPS des six ordinateurs GP* (ordinateurs GP1, GP2, GP3, GP4, GP5 et GP6 de la figure II.2.) qui contrôlent les équipements (par exemple, les pompes à vide) présents dans le sextant qu'ils supervisent.

Le découpage est **fonctionnel** lorsque des équipements de même type sont affectés à un ordinateur parce qu'en plus de se trouver dans sa proximité, ils concourent à l'exécution d'une tâche commune. Dans le cas du SPS, la fonction peut être le contrôle de sous-système de la machine comme l'injection des protons, la radio-fréquence ou l'extraction des protons ... associée à des types d'appareillage regroupés dans un même bâtiment ou bien, pour les fonctions qui ne sont pas directement liées au processus à piloter et qui peuvent être localisées au niveau de la salle de contrôle, la gestion d'une bibliothèque de fichiers, la production de schémas et graphiques donnant l'évolution de divers paramètres de fonctionnement, le développement de programmes d'application ou l'analyse des alarmes donnant une vue globale de l'état de l'accélérateur.

II.2. Les étoiles ou sous-réseaux

Nous pouvons maintenant affiner la structure du réseau contrôlant le SPS : il s'agit d'un réseau homogène local interconnectant une série de sous-réseaux en étoile de minis-satellites, chacun localisé dans une aire bien précise (figure II.2.). L'homogénéité vient de ce que tous les minis peuplant le SPS sont, à de rares exceptions près, des NORD-100 à processeur de 16 bits qui sont fournis par A/S NORSK DATA ELEKTRONIKK travaillant en étroite collaboration avec le CERN.

Pourquoi le choix de cette architecture?

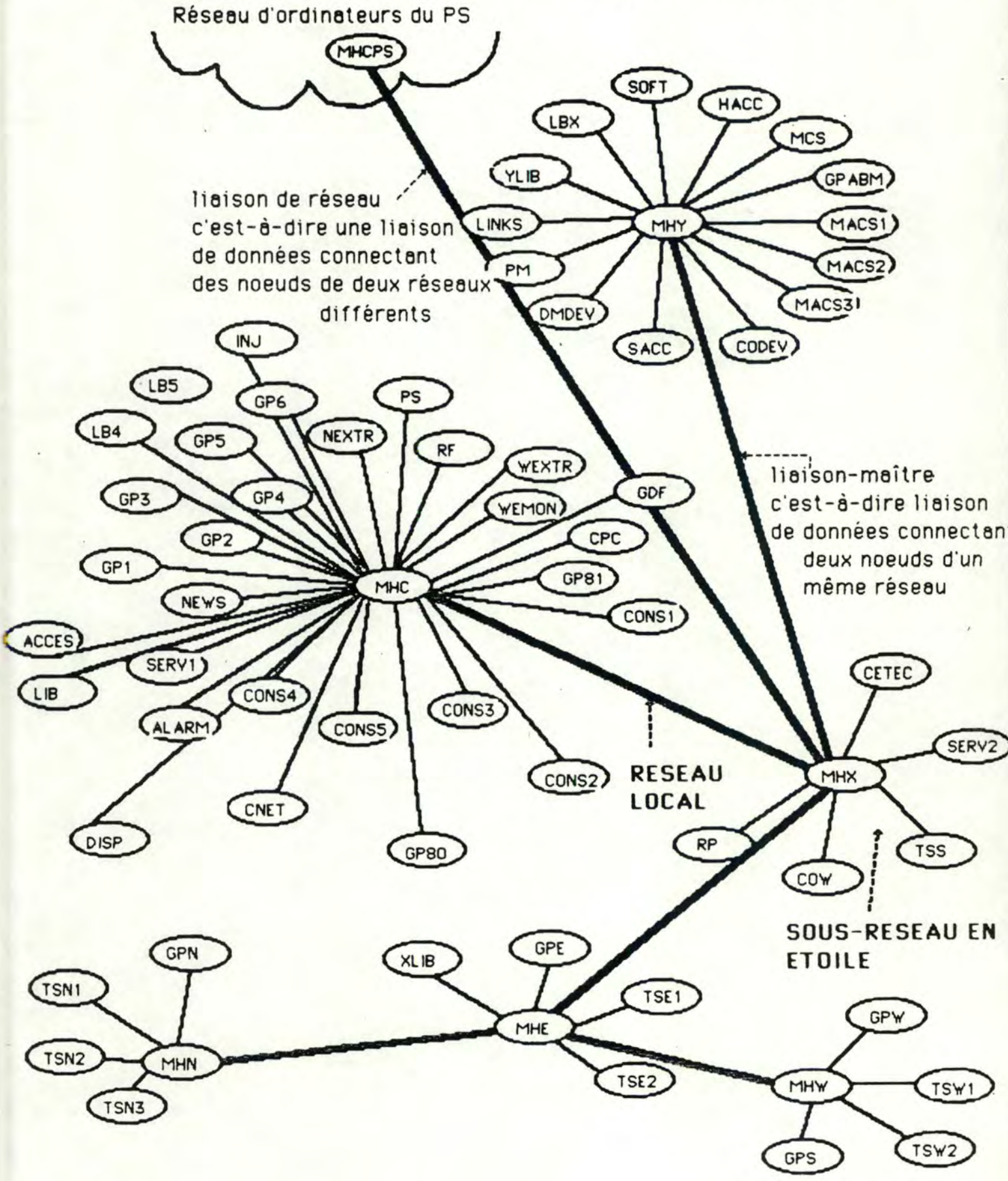


Figure II.2. : Réseau d'ordinateurs du SPS

Un réseau en étoile est un réseau où chaque noeud est relié à un noeud central qui travaille comme un commutateur de messages d'un réseau maillé par la technique du "Store and Forward". Par l'allocation statique d'une largeur de bande d'une ligne à un satellite, le réseau en étoile conduit à une **sous-utilisation des ressources**. A priori, cette topologie pose également des **problèmes de fiabilité** : en effet, comme tout passe par le noeud central, si celui-ci tombe en panne, tout le réseau est mort (nous verrons dans le chapitre VI la solution apportée à cette faiblesse). Enfin, les **coûts d'installation** sont très élevés pour la pose des cables surtout quand les stations sont loin du centre.

Cette architecture fut choisie pour sa **simplicité de mise en oeuvre et d'interfaçage de ligne**. Elle évite en effet le superflu des connexions rencontrées dans un réseau entièrement maillé, N*(N-1)/2 connexions pour un réseau de N stations. Par rapport à un réseau en anneau ou en bus, elle passe outre le problème de l'attribution du moyen de communication à une seule communication et un satellite quelconque peut utiliser le réseau en fonction de ses **besoins instantanés** sans attendre de recevoir son droit à l'émission comme le jeton. Comme un et un seul chemin n'est possible entre deux points du réseau, le service fourni peut être du **datagramme** et se libérer des protocoles d'initialisation et de conclusion des liaisons de données qui pénalisent lourdement les transferts classiques.

Décrivons brièvement les différentes étoiles et leur localisation. Nous suggérons au lecteur de se référer à la figure II.1. afin qu'il se rende compte de la proximité des différents sous-réseaux de leurs centres d'intérêt.

Le sous-réseau le plus important est celui développé autour du *Message Handling Computer* (MHC) car il est dédicacé au contrôle de l'accélérateur proprement dit autour duquel les satellites de cette étoile sont éparpillés.

Deux tiers des minis y sont chargés de recueillir les informations en provenance des différents équipements pilotant l'accélérateur et ses aimants ou en provenance de ceux suivant le faisceau des particules.

Le tiers restant des minis est localisé à proximité de la salle de contrôle pour piloter des consoles d'opération, des écrans de visualisation, pour analyser des alarmes repérées au niveau du réseau et pour présenter à l'équipe de contrôle des données recueillies dans une forme convenant à une assimilation facile.

Cette étoile répond aussi aux critères mis en exergue dans le point II.1 pour mieux rencontrer les impératifs d'un système temps réel : les communications sont maintenues au minimum, pour décharger la salle de contrôle et pour soulager le réseau. Tout satellite qui a besoin pour son traitement d'une information présente dans un autre mini peut disposer instantanément de l'entière largeur de bande du cable pour ses propres besoins.

Parmi les autres sous-réseaux, nous citons :

- l'étoile développée autour du *Message Handling Experiment* (MHE) dont le but est de tester les futurs équipements et ceux déjà présents dans les zones d'extraction Ouest et Nord des faisceaux de particules avant leur projection sur des cibles fixes pour expérience ;
- les étoiles développées autour du *Message Handling West* (MHW) et *Message Handling Nord* (MHN) dont le rôle est de contrôler les zones, soit Ouest, soit Nord d'extraction des faisceaux de particules ;
- les étoiles développées autour des MHX et MHY dont le sigle n'a d'intérêt que dans la différenciation et dont le dessein est d'assurer un support au développement de logiciels et de remplir des fonctions de surveillance du réseau.

Il est intéressant de remarquer que la configuration dépasse la simple interconnexion de sous-réseaux pour mettre en liaison des réseaux par la liaison MHCPS (Message Handling du CPS) -MHX. Le réseau en étoile dont le centre est le MHCPS contrôle l'accélérateur PS qui sert d'injecteur de protons dans le SPS et il a été mis en liaison avec le réseau du SPS pour permettre le transfert de messages de synchronisation dans le fonctionnement des deux systèmes. D'autres passerelles ont été prévues avec le réseau local CERNET du centre informatique du CERN (la Data Division) bien que ne figurant pas sur la figure II.2. car elles sortent du contexte de contrôle proprement dit de l'accélérateur SPS. Elles permettent l'emploi par des concepteurs de logiciels pour le SPS de gros moyens de calcul pour la simulation de leurs progiciels.

II.3. Le système de transfert de messages

On peut constater à la figure II.2. que le problème de la communication se pose à deux niveaux, d'une part à l'intérieur d'une étoile et d'autre part entre les étoiles. Le système de communication qui permet l'échange de messages entre deux satellites quelconques porte le nom de *Message Transfer System* (MTS). C'est un système à commutation de messages travaillant selon la technique du "store-and-forward" et fournissant un service datagramme (voir point II.2). Il est clair qu'un rôle essentiel est joué par les noeuds de ce réseau que sont les ordinateurs dont le sigle commence par MH, ordinateurs que désormais nous symboliseront par le nom MH*, ordinateurs de même type que les satellites et dont la fiabilité a été démontrée dans la pratique (leur MTBF s'élève à plus d'un mois).

Il existe trois paramètres pour définir les performances d'un système de transfert comme le MTS :

- le taux maximum de transfert de données entre deux ordinateurs, en l'absence de toute autre charge : il fut établi à 160 kbps pour éviter que l'opération des satellites ne soit ralentie par le fait que les fichiers sont stockés sur une bibliothèque centralisée ;
- le taux de transfert maximum quand beaucoup d'ordinateurs transmettent en même temps : il fut établi à 480 kbps pour un maximum de 32 ordinateurs dialoguant simultanément avec des messages de 512 bits ; d'autre part, il fut décidé que les alarmes détectées par les satellites, qui constituent le gros de leurs messages, seraient stockées localement et seraient envoyées à intervalles réguliers ou lorsque leur zone-tampon serait remplie vers le satellite du sous-réseau MHC de gestion des alarmes c'est-à-dire de toutes les erreurs et problèmes rencontrés sur le site du SPS (ordinateur appelé ALARM de la figure II.2.); il fut également tenu compte des estimations maximales de charge de pointe pour chaque satellite ;
- le temps de transfert d'un message depuis son émission par un processus d'un ordinateur à sa disponibilité dans le processus d'un second ordinateur : il fut établi à 5 msec pour le transfert d'un message de 512 bits et pour permettre la correction rapide d'un

paramètre d'un ordinateur surveillant des équipements à l'annonce d'un certain événement apparu dans la conduite de l'accélérateur.

II.3.A. La communication à l'intérieur du MTS

Au moment de la conception du MTS, le modèle "Interconnexion de Systèmes Ouverts" proposé par le Sous-Comité (SC 16) de l'ISO n'existait pas, ce qui explique que nous nous démarquerons de cette approche.

L'échange de messages entre un ordinateur dit source et un ordinateur dit destination se ramène à une série de **transfert de paquets** entre deux ordinateurs directement connectés ; soit

- entre l'émetteur-satellite et le centre de son étoile ;
- entre ce centre et le centre-destination si le récepteur appartient à une autre étoile ;
- entre le centre-destination et un satellite.

Toutes les transmissions se font à l'intérieur de paquets de 4 mots d'entête (un mot a une longueur de 16 bits) et de champ des données pouvant varier d'1 à 64 mots. Les messages plus longs que 64 mots sont découpés dans la station émettrice en paquet. Comme pour tout couple émetteur-récepteur, il n'existe jamais qu'un seul chemin pour les relier, le message est reformé dans l'ordinateur-destination sans problème sur base des informations contenues dans l'entête et de l'ordre dans lequel les paquets ont été reçus.

La transmission des données entre un satellite et son noeud se fait par DMA en série sur une paire torsadée vidéo à 490 Kbps et est synchronisée par l'échange de mots de contrôle en série sur une paire torsadée audio à 480 Kbps. Ces paires sont dédoublées pour permettre le dialogue en full-duplex.

Comment s'effectue un dialogue entre deux ordinateurs ?

- un émetteur ne peut envoyer un paquet vers un récepteur que si la liaison a été ouverte au préalable ;
- il n'y a pas d'interruption dans le récepteur lorsque le paquet a été

reçu ; c'est l'émetteur qui provoque une interruption par l'envoi, une fois son paquet transmis, du mot de contrôle "vous avez un paquet" sur le canal de contrôle ; ce mécanisme permet de transférer un paquet sans accord préalable du récepteur ;

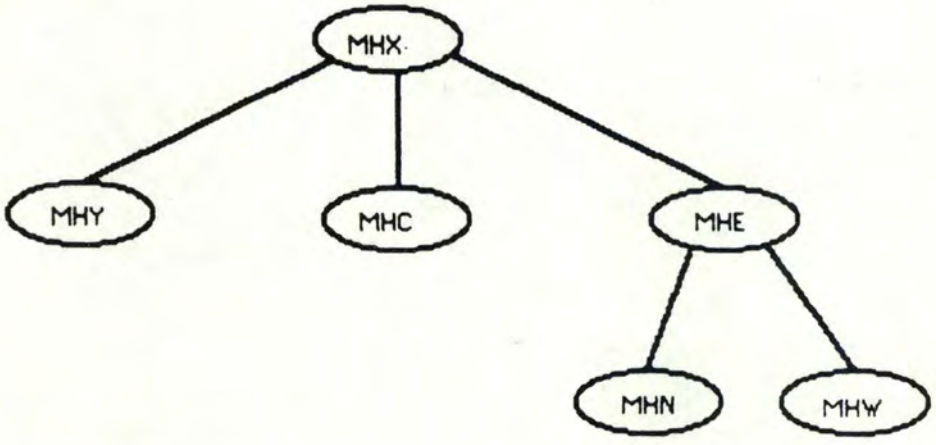
- le récepteur entre en phase de contrôle d'erreur ; chaque mot de données de 16 bits est entouré de deux start bits et d'un bit de parité transversale, le paquet, lui, est contrôlé par un mot de checksum constituant la parité longitudinale ; dans chaque mot de contrôle, 4 des 16 bits sont également utilisés pour la parité longitudinale ; si l'opération est concluante, l'émetteur subit une interruption à la réception du mot de contrôle "transmission activée" qui prépare l'envoi du second paquet ; si la transmission n'a pas été fructueuse, la source reçoit "répéter l'ancien paquet" ; il peut y avoir 7 essais avant le report d'une erreur vers l'opérateur ;
- lors de l'émission de son paquet, l'émetteur avait armé un time-out ; si avant que ce dernier ne tombe, il n'a pas vu de réaction de la part du récepteur, il lui envoie sur le canal de contrôle "demande de réponse" en ayant armé un deuxième time-out ; si, avant que ce dernier ne tombe, il reçoit "ok pour la demande de réponse", il se prépare à envoyer le second paquet sinon c'est le signe que le destinataire ne fonctionne plus, soit qu'il est mort ou soit que la ligne est coupée; un message d'erreur est émis et toutes les transactions avec le destinataire sont stoppées.

II.3.B. Le rôle du centre des étoiles

Les MH* ont pour premier rôle de **router les paquets incidents** vers la bonne file FIFO de sortie, une file étant constituée par ordinateur et par priorité. Le routage se fait sur base d'un "three linked tree" qui reprend pour chaque centre son noeud-centre père, le premier de ses noeuds-centre frères, le premier de ses noeuds-centre fils ainsi que sa table des satellites (figure II.3.).

Les MH* jouent un rôle prépondérant dans le **contrôle de flux des messages**. En effet, des messages nécessitant plusieurs paquets et qui ne sont pas attendus par le récepteur peuvent épuiser son pool des blocs libres car il faut que tous les paquets soient reçus avant de pouvoir analyser le

Le réseau d'ordinateurs du SPS a la topologie suivante :



Il peut être représenté par la structure "three linked tree" suivante
avec F, le pointeur vers le noeud-père,
avec S, le pointeur vers le noeud-fils de gauche,
avec B, le pointeur vers le noeud-frère de droite
et avec T, le pointeur vers la table des satellites :

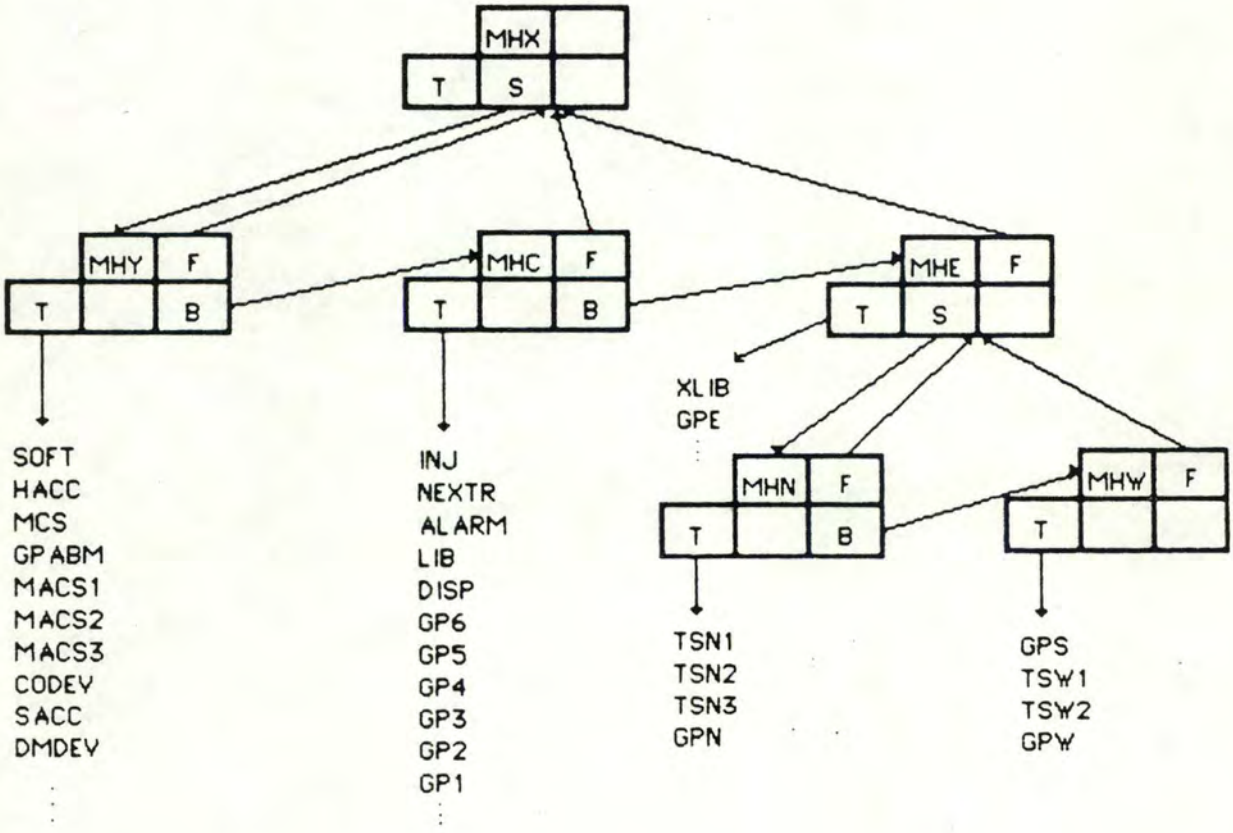


Figure II.3. : Les tables de routage

message et déclencher le processus auxquels ils sont destinés. Une situation de blocage peut se produire dans le récepteur où tous les blocs sont occupés par des paquets appartenant à des messages multi-paquets incomplets. Les MH* filtrent ces messages sur base des informations de type contenues dans l'entête des paquets pour qu'un seul à la fois n'atteigne un ordinateur-récepteur et pour qu'un seul message de ce type ne soit transmis à la fois par un émetteur. Si le récepteur ne peut plus suivre, il envoie un mot de contrôle au centre de son étoile pour qu'il suspende toute transmission ultérieure en mettant les paquets en file d'attente. Un autre mot de contrôle avertit le MH* que l'étranglement est passé et que la transmission peut reprendre. Si, par contre, le MH* tombe aussi à cours de blocs libres, il envoie un mot de contrôle à tous les satellites envoyant de longs messages sauf un (celui qui a commencé le premier). Les paquets sont mis en file d'attente au niveau des satellites dont le processus générant ces messages peut se voir également suspendu. Une fois la situation d'étranglement terminée, un mot de contrôle est envoyé par le centre de l'étoile pour permettre la poursuite des opérations.

Ce contrôle de flux est un exemple typique du mélange dans le mot de contrôle d'informations appartenant à plusieurs couches du modèle ISO. En effet, en plus de sa fonction liée au protocole de ligne, le mot de contrôle contient aussi de l'information lui permettant de jouer un rôle dans la régulation des débits des messages et dans l'initialisation du système. Certains affirment que pour adapter l'architecture du MTS au modèle ISO, la modification serait limitée au changement des couches support physique et liaison de données. Il suffirait d'emballer ces deux types d'information destinés au niveau réseau dans une trame d'information pour n'être traité qu'au niveau supérieur.

Un troisième rôle dévolu aux MH* consiste à **vérifier et contrôler les liaisons de données**. En effet, ils envoient des messages aux ordinateurs silencieux depuis un certain temps connu en armant un time-out. Ce time-out permet de vérifier que l'ordinateur est toujours en vie et permet de tester le bon état des liaisons en testant les configurations de bits émises et réceptionnées. Si une liaison est coupée, ils envoient des messages vers l'ALARM.

Bien que cela sera expliqué dans le point suivant, nous pouvons déjà souligner que le logiciel présent dans les MH* est, à quelques exceptions près, identique à celui rencontré sur les satellites.

II.4. Les satellites

Pour qu'un satellite s'intègre dans cet environnement de contrôle sur une grande échelle d'équipements, il doit contenir du logiciel pour assurer :

- l'exécution des processus ; pour remplir cette fonction, le logiciel sera constitué d'un interpréteur (le langage disponible sur le Nord-100 est le NODAL ou NOrD Accelerator Language, appelé ainsi pour également montrer sa parenté avec FOCAL développé par DEC) (voir point II.4.A) et d'un système d'exploitation multi-tâches en temps réel (SYNTRON) (voir point II.4.B) ;
- la connexion entre ordinateurs au travers de liaisons de données série (voir point II.4.C) ; le logiciel SYNTRON est, entre autres, dévolu à cette tâche ;
- la connexion de l'ordinateur aux équipements (voir point II.4.D) à contrôler ; du logiciel a été développé pour éviter de devoir adresser explicitement le hardware et pour fournir à l'utilisateur un interface agréable ; ce logiciel se traduit par la notion de Data Module (voir point II.4.D.3) et est le seul à ne pas se retrouver sur les MH*, ni sur les ordinateurs conduisant les consoles de la salle de contrôle.

II.4.A. L'interpréteur Nodal

En quoi un interpréteur répond-t'il aux exigences de contrôle du SPS ?

La réalité d'un accélérateur est que, même, lorsqu'il a atteint le stade de maturité pour les expériences en physique, il reste en développement (nouveaux équipements ajoutés). Cet environnement constamment changeant requiert la nécessité de modifier rapidement les programmes de contrôle suivant les besoins journaliers de la production et les études théoriques de la machine SPS. Un interpréteur a ainsi été fourni aux physiciens et aux ingénieurs responsables de la conception et de la construction des

équipements de contrôle :

- pour leur faciliter le développement et la mise au point de leurs procédures de pilotage qui, de plus, sont **relativement réduites**, sans nécessiter le recours à de hautes notions informatiques ;
- et pour leur permettre de se concentrer sur l'explication physique du phénomène observé et non sur des astuces de programmation.

Un langage interprété est facile d'apprentissage. De plus, les erreurs à l'exécution peuvent, après avoir été signalées, être directement corrigées au niveau du fichier source resté intact. Dans un programme compilé, il est parfois difficile de retrouver à partir d'une erreur d'exécution la ligne correspondante à corriger dans le programme source.

Le NODAL peut être vu comme un langage spécifique de la répartition c'est-à-dire comme un langage de mise en oeuvre qui permet à l'utilisateur de définir et de contrôler l'organisation de l'architecture (fermeture d'une liaison de données...) et comme un langage de commande pour réseau qui exprime le contrôle de l'exécution répartie. En effet, l'utilisateur a connaissance de l'architecture sur laquelle il travaille car l'équipement à surveiller est connecté à des minis bien spécifiques : le NODAL lui donne le choix du lieu d'exécution de ses programmes et du lieu de production de leurs résultats tout en veillant à ce que le protocole de ligne reste transparent à l'utilisateur qui n'a conscience que de la structure multi-ordinateurs.

Grâce à ses instructions-réseau dans lesquelles l'ordinateur impliqué sera donc explicitement spécifié et à sa structure interne qui permet de référencer facilement des sous-structures de programme telles qu'une variable, un groupe d'instructions ou une instruction, le NODAL permet tout naturellement le lancement d'exécution de parties de programmes en parallèle dans différents ordinateurs. Le tout se ramène à une manipulation de fichiers. L'interpréteur opère sur des fichiers contenant des commandes interprétatives ou des variables et si l'on veut soumettre des parties de programme à distance, il suffit d'envoyer le fichier contenant le programme et les données sur lequel l'interpréteur à distance devra travailler.

En voici des exemples :

- EXECUTE.(GP1) 2 A qui déclenche l'exécution dans l'ordinateur GP1 du groupe d'instructions 2 avec la variable A ; ceci montre comment une relation maître-esclave peut être programmée ; on voit que cette

relation n'est pas une conséquence statique de la configuration du matériel utilisé ;
- REMIT B C qui retourne à la tâche ayant soumis l'exécution à distance les résultats B et C ;
- WAIT (GP1) qui permet la synchronisation de la tâche ayant soumis une exécution à distance sur le retour des résultats.

Certains ont cependant critiqué la lenteur de l'interpréteur. Des solutions y ont été apportées :

- l'appel à des sous-routines écrites en assembleur ;
- l'emploi d'un pré-interpréteur pour certaines lignes de programme réalisant l'accès à distance ; il permet d'accélérer par un facteur deux l'exécution tout en doublant l'espace-mémoire nécessaire mais conserve la structure interprétative des lignes de programmes et des variables c'est-à-dire qu'il code les commandes, organise en pile des valeurs des paramètres d'une fonction et des variables et évalue des expressions constantes ;
- l'emploi d'un compilateur croisé pour l'exécution rapide d'une section locale de programme une fois qu'elle a été testée sous sa version interprétée.

II.4.B. Le système d'exploitation

Norsk Data fournit un système d'exploitation "Syntran" qui offre :

- des possibilités de temps réel c'est-à-dire qu'il peut réagir à des stimuli générés aléatoirement par l'environnement endéans une période suffisamment courte de sorte que la fonction à exécuter a toujours sa raison d'être (voir point I.1) ;
- des possibilités de temps partagé ;
- et des possibilités de traitement batch.

En son temps, la version numéro deux de "Syntran" a été augmentée d'un logiciel de transfert de messages pour assurer l'interface avec les liaisons de données, d'un interpréteur Nodal et d'un meilleur algorithme de répartition des tâches sur le CPU pour donner naissance à "Syntron" (équivalant à

Synchrotron Control) dont tout mini dispose actuellement d'une version taillée à sa mesure.

Les tâches sont répertoriables, de par leur nature, en différentes classes présentées ci-dessous par niveau de priorité croissante :

- la classe des tâches programmées de surveillance qui tourne en arrière-plan et sonde périodiquement les équipements connectés aux minis pour y déceler des cas pathologiques et envoyer, le cas échéant, un message vers l'Alarm pour lui signaler une faute ou un cas d'exception ; ces tâches sont réveillées à un instant bien précis ou suite à un événement ;
- la classe intéractive qui aide au développement de programmes ;
- la classe des programmes lancés par un opérateur pour contrôler des sous-systèmes de l'accélérateur comme la radio-fréquence ;
- la classe réseau qui permet le dialogue entre processus en exécution sur différents ordinateurs ; leur synchronisation est assurée par le système de timing du SPS dont l'unité de base est la msec et qui est lui-même resynchronisé au début de chaque cycle d'accélération des particules sur le système de timing du CPS (le CPS sert d'injecteur de protons accélérés à 10 Gev pour l'accélérateur SPS qui lui les accélère jusqu'à 400 Gev) ; chaque opération spécifique durant un cycle d'accélération est symbolisée univoquement par une configuration binaire appelée événement ; ces configurations sont émises par le système de timing et repérées par des équipements bien spécifiques qui génèrent une interruption dans les minis auxquels ils sont connectés ; cette interruption est provoquée dans un délai d'un nombre fixé d'impulsions d'horloge après la détection d'événement et déclenche des programmes ; cette solution a été adoptée pour éviter de devoir compter un certain nombre d'impulsions de 1 msec à partir du début de chaque cycle d'accélération avant de pouvoir déclencher un programme et pouvoir s'adapter à une longueur de cycle différent ;
- la classe des tâches de servitude du système ;
- la classe des tâches programmées de haute priorité.

II.4.C. Interface-réseau

Cet aspect a déjà été developpé dans les points II.2 sur les étoiles, II.3 sur le système de transfert de messages, II.4.B sur les systèmes d'exploitation.

II.4.D. Interface-équipements

Nous nous attaquerons d'abord à l'aspect hardware du problème en définissant les notions de CAMAC (voir point II.4.D.1) et de multiplexeur de données (voir point II.4.D.2) avant de nous attarder sur l'aspect logiciel par le concept de Data Module (voir point II.4.D.3).

II.4.D.1. Le CAMAC

Un interface est le nom donné aux équipements électro-mécaniques convertissant un signal d'une forme comprise par un équipement d'E/S en une autre forme comprise par l'ordinateur.

Si l'on dispose de n équipements et de m ordinateurs, on en a en principe besoin de n*m différents pour les interconnecter alors qu'il suffirait de définir un interface virtuel auquel s'adapteraient et l'interface-ordinateur et les interfaces moins communs des équipements. Dans ce cas, il ne resterait plus qu'à mettre au point n+m interfaces par la définition de ce dénominateur commun.

Les ordinateurs peuvent présenter une limite supérieure au nombre de ports d'E/S sur leur bus interne. La solution est de définir un interface-ordinateur occupant un seul port d'E/S, cet interface s'occupant de

l'interfaçage avec les n équipements.

L'*European Standards Of Nuclear Electronics group* (ESONE) où le Cern est représenté a défini le CAMAC ou standard d'interface IEEE-583 comme :

- un système d'interface par modules qui procure un moyen de communication standard entre des instruments de mesure et de contrôle et un ordinateur ou tout processeur non-programmable dédicacé à un tâche bien spécifique d'une façon indépendante des instruments et de l'ordinateur ;
- un chassis "CAMAC Crate" dans lequel on peut encastrer jusqu'à 25 modules d'interface-équipement et dont le 25ème est nécessairement le contrôleur et un bus parallèle sur l'arrière du chassis "CAMAC Dataway" qui supporte le transfert de données à 24 Mbps, des adresses, des signaux de contrôle et de temporisation ; ce taux de transfert est vital dans les applications qui nous concernent où de grandes quantités de données peuvent être transférées rapidement pendant chaque expérience ou prise de mesure. (figure II.4.)

Le nom originaire suggéré par le comité était Janus, le Dieu à deux visages regardant dans des directions opposées pour symboliser l'aspect bidirectionnel de l'interface. Par la suite, Janus a fait place à *Computer Automated Measurement And Control* (CAMAC).

Pour connecter à un ordinateur des périphériques spécifiques à une application comme un convertisseur analogique-digital (à ne pas confondre avec les périphériques propres à un système informatique fournis par le constructeur d'ordinateurs comme les disques), il suffit donc de connecter sur le bus interne de l'ordinateur un bus parallèle relié au panneau avant du contrôleur-CAMAC, le seul élément dépendant de l'ordinateur, et de connecter les interfaces-périphériques au CAMAC. (figure II.4.)

Le CAMAC peut aussi être vu sous l'angle "microprocesseur" en lui ajoutant des modules intelligents comme un microprocesseur. Les avantages en sont :

- un temps de réponse plus rapide aux stimuli de l'extérieur ;
- un taux de données E/S plus bas sur le bus reliant le contrôleur CAMAC à l'ordinateur ;

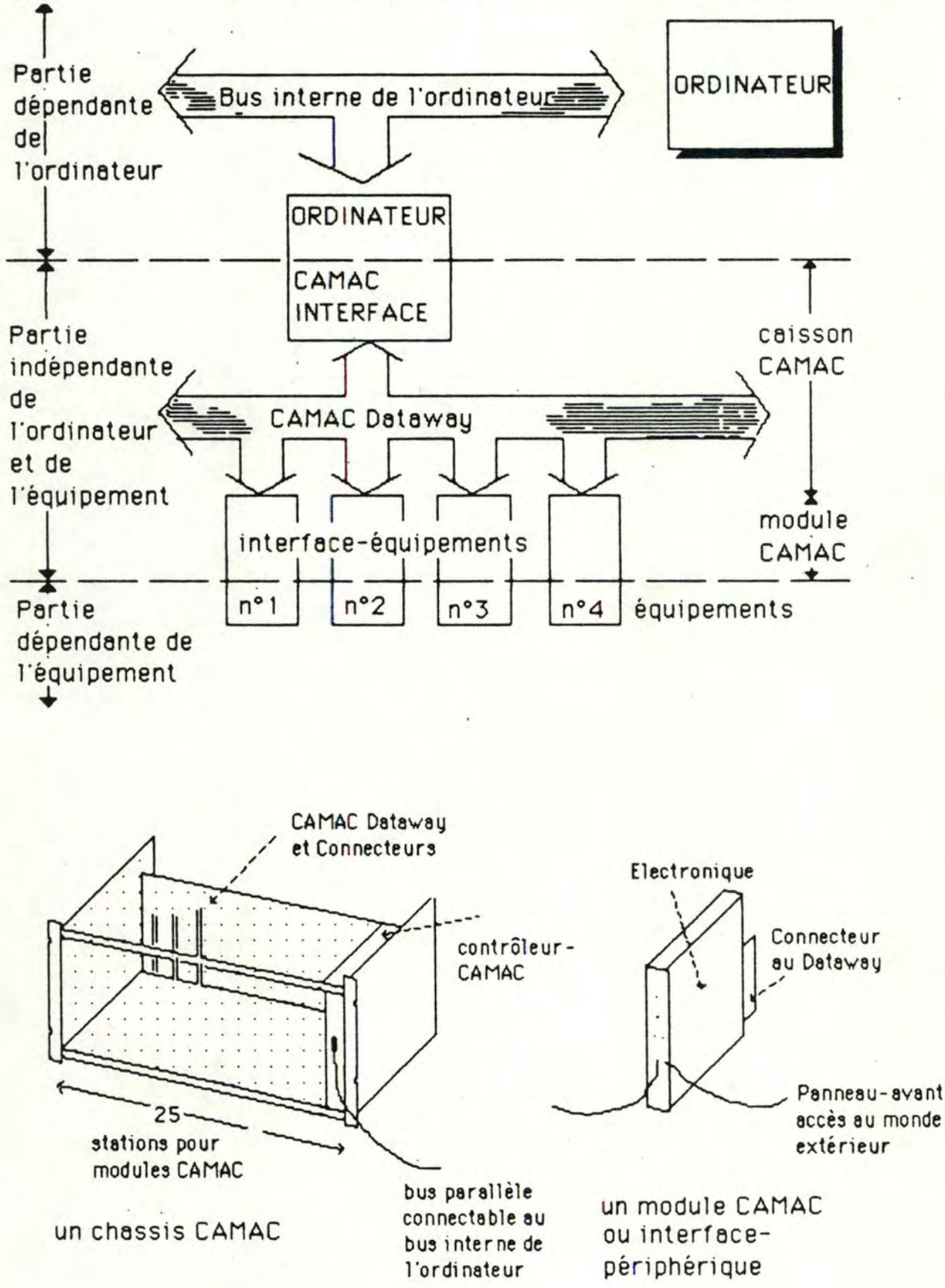


Figure II 4 : L'interface-ordinateur ou CAMAC

- la possibilité de mener des opérations en parallèle dans le CAMAC libérant, de ce fait, le CPU principal pour lui permettre la réalisation de tâches plus importantes de calcul, de prise de décision, de conversation avec les autres ordinateurs, d'interprétation d'instructions encodées par l'opérateur.

Le CAMAC présente les avantages suivants :

- les compagnies d'électronique peuvent développer en grande série des équipements présentant un interface standard CAMAC sans devoir les spécialiser pour un type d'ordinateur bien précis ;
- ces compagnies peuvent se dégager des constructeurs et de leurs intérêts pour développer à leur guise leurs équipements ;
- changer d'ordinateur revient uniquement à changer de contrôleur ;
- une fois que l'interface hardware CAMAC/ordinateur a été acheté, l'accès est ouvert à une multitude d'interfaces-équipements.....

Des limitations existent aussi car les appareils contrôlés par un mini sont nombreux et dispersés sur de grandes étendues, parfois à 1 km de l'ordinateur :

- il faudrait alors plusieurs CAMAC avec le risque de manquer de ports d'E/S sur le bus interne de l'ordinateur pour les connecter ;
- le CAMAC, à cause du bus parallèle le reliant à l'ordinateur, ne peut s'éloigner très fort de ce dernier ; il faudrait alors recourir à un bus série quand la vitesse demandée des transferts de données n'est pas trop élevée ; la structure obtenue porte le nom de CAMAC Serial Highway qui peut couvrir une distance d'une centaine de mètres, sur lequel on peut connecter jusqu'à 62 chassis CAMAC en utilisant des Serial Crate Controllers en guise de contrôleurs et qui nécessite l'introduction d'un interface supplémentaire avec l'ordinateur, le "Branch Driver" (figure II.5.) ; cette solution onéreuse n'est que rarement présente au CERN car le chassis n'est pas suffisamment isolé pour pouvoir être utilisé dans des salles à fortes perturbations électro-magnétiques ; on lui préfère la solution avec bus parallèle aménagé...

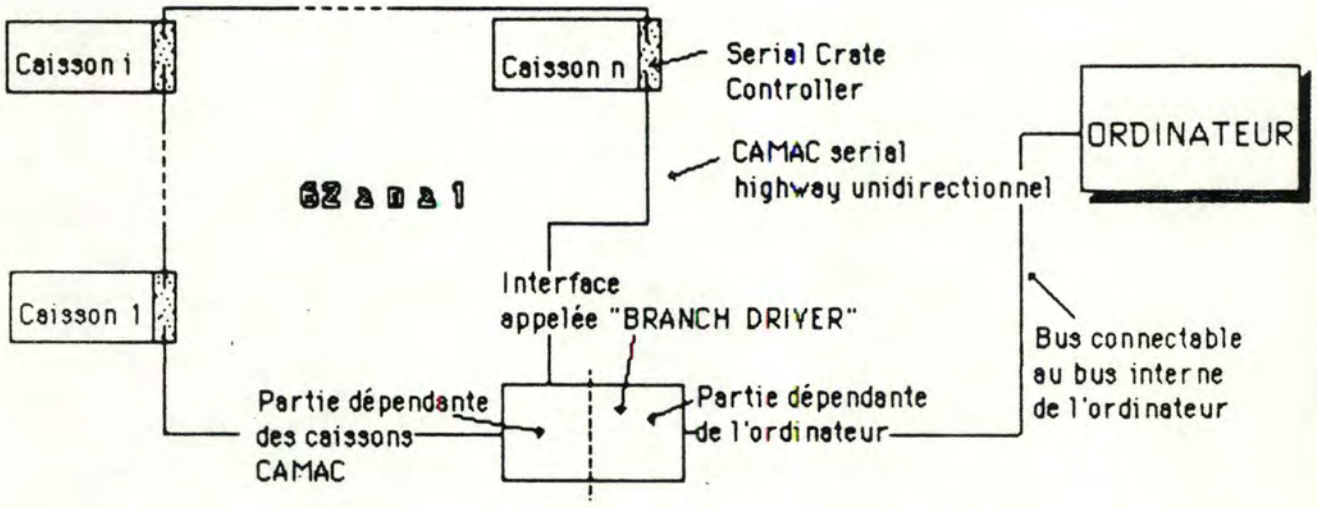


Figure II.5. : Le CAMAC "SERIAL HIGHWAY"

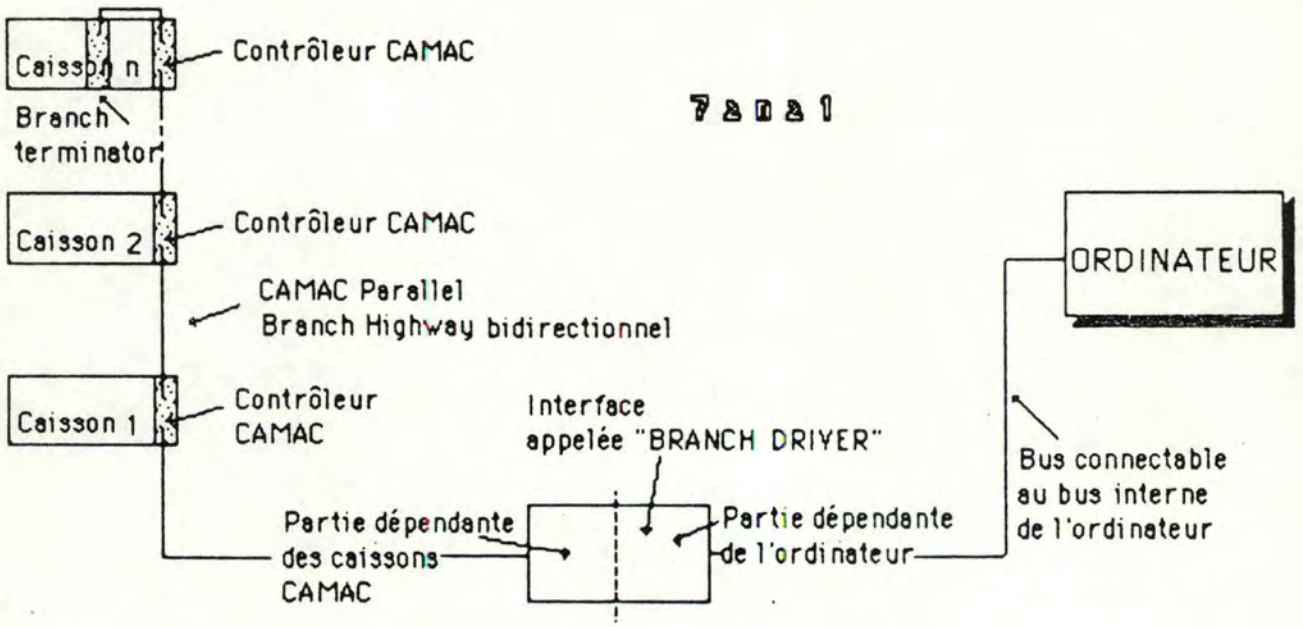


Figure II.6. : Le CAMAC "PARALLEL BRANCH HIGHWAY"

II.4.D.2. Le multiplexeur de données

Les ingénieurs ont décidé de recourir au CAMAC avec bus parallèle pour l'acquisition de données à très haute vitesse et quand les modules d'interface étaient disponibles. Pout tout ce qui est contrôle et acquisition qui ne nécessitent pas de grandes vitesses, ce qui représente 90% des cas, les ingénieurs se sont prononcés pour un système de multiplexage de données digitales et analogiques (figure II.7.). Cela signifie qu'au lieu d'avoir dans le chassis CAMAC les interfaces d'équipement, on y trouverait un module qui serait l'*Unité de Contrôle Série* (UCS) d'un bus à multiplexage temporel que nous désignerons par la suite de bus multiplex et sur lequel seraient connectées les interfaces-équipements. Cette Unité UCS contrôlerait l'accès au bus des différents équipements et permettrait l'envoi sur des distances plus longues de fonctions à exécuter et la surveillance avec plus de fiabilité d'équipements dans des cas exigeant une isolation importante entre le chassis CAMAC et les équipements à contrôler.

Le bus multiplex est série et est constitué d'un cable de 7 paires torsadées qui transporte les informations binaires et les signaux de synchronisation et d'un cable séparé de paire torsadée pour les mesures analogiques. Différentes configurations de ce bus sont possibles :

- configuration courte distance <= à 200 m ;
- configuration longue distance <= 1,5 km ;
- configuration très longue distance n*1,5 km où n est le nombre de répéteurs sur le bus et où n=5 pour pouvoir effectuer le tour de l'anneau du SPS.

Poursuivant notre analyse de la figure II.7., nous pouvons ajouter que l'UCS prend trois modules de large dans le chassis CAMAC. Par convention, les 4 premiers UCS se trouvent dans le premier CAMAC et si l'on utilise une configuration multi-chassis sur un bus parallèle, les 4 suivants sont dans le 2ème et ainsi de suite... (ce bus d'au plus 30 m porte alors le nom de CAMAC Parallel Branch Highway (figure II.6.) sur lequel on peut connecter jusqu'à 7 chassis CAMAC - sur la figure II.7., un seul chassis est représenté pour faciliter la présentation - et il est également connectable à l'ordinateur par l'intermédiaire de l'interface "Branch Driver"). Sur le bus multiplex, 32

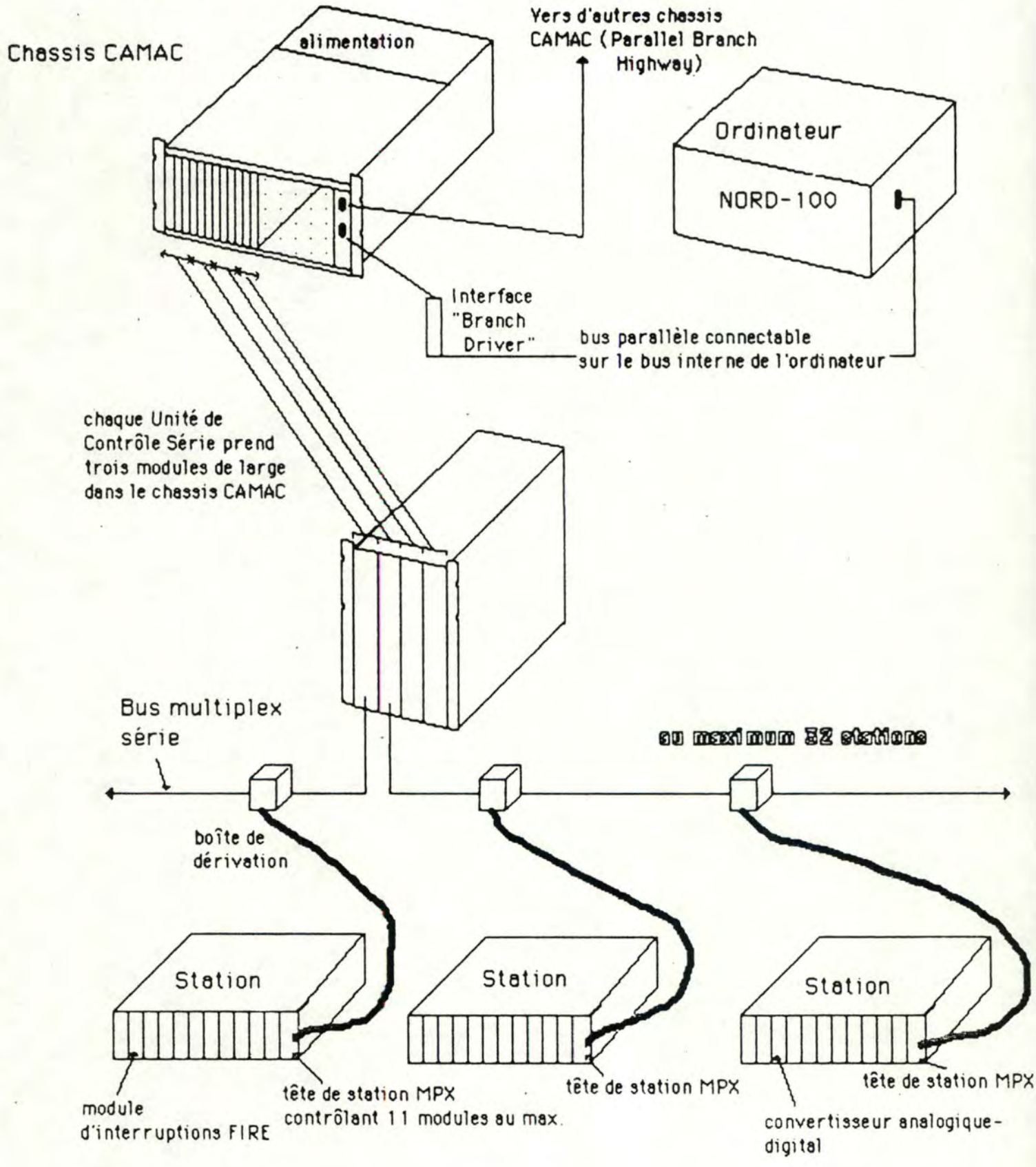


Figure II.7 : Le multiplexeur de données

stations peuvent être connectées avec dans chacune d'elles, un contrôleur (repris sous l'intitulé tête de station MPX sur la figure II.7.) à l'écoute de la ligne. Dans chaque station, on peut intégrer 11 modules d'interfaces-équipements dont par exemple, un module d'interruptions FIRE ou un convertisseur analogique-digital... (tous ces détails techniques ne sont pas destinés à assommer le lecteur mais à lui donner une idée du contexte dans lequel s'est placé notre travail au CERN, objet de la troisième partie du mémoire).

II.4.D.3. La notion de Data Module

Un Data Module est une sous-routine de gestion de l'interface d'accès à une famille d'équipements.

La notion de Data Module permet de regrouper l'accès aux équipements d'une même famille contrôlés par un mini dans un module réalisant toute la cuisine interne (CAMAC, bus multiplex, stations...) nécessaire à l'accès au hardware pour permettre la présentation d'un interface agréable à l'utilisateur dans les programmes de plus haut niveau. Elle permet de standardiser l'organisation des informations relatives à cette famille et de réduire le contrôle du hardware à l'appel d'une simple fonction software.

Ces routines d'E/S spécialisées constituent une part non négligeable du software installé sur les minis de contrôle d'équipements : il y a autant de Data Modules qu'il y a de familles différentes d'équipements connectés à un mini. Elles sont le plus souvent écrites par les constructeurs des sous-systèmes de l'accélérateur soit directement en langage machine MAC du NORD-100 en recourant à la sous-routine CAMPX donnant accès aux stations et modules connectés au NORD-100 via le CAMAC et le système Multiplex, soit en NODAL exigeant ainsi le passage par un compilateur croisé.

Il est intéressant de noter que le NODAL offre également des instructions d'E/S permettant dans un programme de plus haut niveau de court-circuiter le Data Module et de dialoguer directement soit avec un module du chassis CAMAC (avec la macro CAMAC), soit avec les équipements (avec la macro GPMPX qui revient à utiliser une fonction disponible de

l'arsenal CAMAC qui écrira dans l'UCS un mot de 16 bits ou "ordre" ; cette écriture aura pour effet de déclencher l'exécution d'une fonction donnée dans un module d'une station déterminée (soit écriture d'un mot, soit lecture...)).

Pour nous résumer, connecter une pompe ou tester l'état d'un générateur revient à l'appel d'un Data Module. L'interface consiste en l'appel à une sous-routine via deux paramètres :

- le nom de la sous-routine étant le nom générique de la famille ;
- le premier paramètre étant le numéro de séquence de l'appareil référencé (les éléments d'une famille sont numérotés séquentiellement);
- le deuxième paramètre étant la propriété particulière de cet appareil qui désigne l'action à exécuter sur l'équipement ou le renseignement à prendre comme dans les exemples cités.

Le Data Module utilise les informations contenues dans deux tables qui constituent une petite Base de Données, pour réaliser les actions demandées sur l'équipement :

- une entrée via le premier paramètre dans la Data Table où est définie l'adresse physique sur laquelle doit porter l'E/S c'est-à-dire le numéro d'UCS, la station et le module concernés par l'équipement identifié ainsi que les informations nécessaires pour tester le caractère licite de l'appel à ce Data Module ;
- une entrée via le deuxième paramètre dans la Property Table où est définie l'adresse d'emplacement du code réalisant la propriété.

A titre d'illustrations, MAGNET (6,CUR) désigne le niveau du courant (propriété CUR) dans l'aimant (famille MAGNET) numéro 6 (numéro d'équipement 6).

SET MAGNET (6,CUR) = 101.6 a pour effet de charger (instruction NODAL "SET ... = ...") le convertisseur analogique-digital contrôlant la source de courant du 6ème (numéro d'équipement 6) aimant (famille MAGNET) par une valeur provoquant du courant (propriété CUR) à 101,6 A dans l'aimant.

TYPE OCTUPL (4,CUR) imprime sur le terminal (instruction NODAL "TYPE") le niveau de courant (propriété CUR) dans l'octupole (famille OCTUPL) numéro 4 (numéro d'équipement 4).

La notion de Data Module revêt de nombreux avantages, elle permet une distribution des différentes tables d'informations dans des modules résidant dans les différents ordinateurs du réseau pour réaliser une pseudo Base de Données répartie ; l'addition d'un nouvel équipement à une famille revient à ajouter une nouvelle ligne à la Data Table ; le code du Data Module peut être descendu dans un microprocesseur contrôlant l'équipement pour décharger le mini de toute la cuisine interne de manipulation d'équipement (ce point revêtira toute son importance dans le chapitre VI de la partie III).

II.5. Exemple récapitulatif

Tous les concepts mis en évidence au cours de ce chapitre peuvent être illustrés par une procédure NODAL typique. Celle-ci est lancée de la salle de contrôle par un opérateur pour mesurer le profil du faisceau extrait de l'accélérateur dans la zone expérimentale Nord. Elle fait usage d'un miniscanner mû pas à pas dans le faisceau et dont le rôle est de prendre des informations sur la position exacte du faisceau.

```
1.1 ASK "INITIAL POSITION="IP "FINAL POSITION="FP
1.2 SET P=IP ; EXECUTE(NEXTR) 3.2 P
1.3 FOR P=IP+1,FP ; DO 2
1.4 END

2.1 WAIT-CYCLE 6
2.2 EXECUTE(NEXTR) 3 P
```

Possibilité pour l'opérateur d'exécuter une autre tâche en parallèle comme l'affichage d'un graphique, ... que nous représenterons symboliquement par l'instruction : 2.3 SET I=P-1;

```
2.4 TYPE "POSITION="I "CHARGE="
2.5 WAIT(NEXTR) ; TYPE A

3.1 SET A=MINSCN(2,CHRG)
3.2 SET MINSCN(2,PSN)=P
3.3 REMIT A
```

Ligne 1.1 : demande à l'opérateur l'intervalle de mesures à prendre de IP

à FP.

Ligne 1.2 : initialise le miniscanner en envoyant à l'ordinateur NEXTR la ligne 3.2 ainsi que la position initiale du miniscanner (P).

Ligne 1.3 : boucle jusqu'au recouvrement de l'intervalle de mesure. Le groupe 2 est exécuté durant chaque parcours de boucle.

Le groupe 2 commence par attendre l'événement "extraction du faisceau" (événement 6) envoyé par le système de timing du SPS ; la ligne 2.1 assure la synchronisation de l'exécution du programme avec le cycle de l'accélérateur.

Ligne 2.2 : envoie le groupe 3 pour exécution dans l'ordinateur NEXTR avec l'indication de la prochaine position du miniscanner (P).

Ligne 2.4 : quand la tâche à distance est lancée, cette ligne est exécutée immédiatement par affichage de la position sur le terminal.

Ligne 2.5 : synchronise l'exécution du programme avec l'arrivée des résultats de la tâche éloignée et affiche alors la charge mesurée (A).

Le groupe 3 qui est envoyé pour exécution dans l'ordinateur NEXTR et qui est exécuté en parallèle avec le programme principal, obtient la charge du miniscanner en ligne 3.1, le positionne pour la prochaine mesure et renvoie la charge au programme principal.

Chapitre III : L'ANNEAU A ELECTRONS-POSITONS ET SON SYSTEME DE CONTROLE

Le LEP ou Laboratoire Electrons-Positons, ce grand anneau de stockage en construction à la frontière franco-suisse, est un enfant du SPS en ce sens qu'il reprend à son compte un grand nombre de principes appliqués à l'accélérateur-père. Ses équipements, une fois la construction terminée, seront répartis dans des galeries parallèles à l'anneau ou dans des bâtiments regroupés selon les huit points d'accès au tunnel, chacun distant de 3,5 km.

Un système de contrôle de processus peut varier de simples systèmes avec une configuration minimale d'ordinateurs dont la tâche est d'enregistrer des données produites par l'opération du système à des systèmes hiérarchiques contrôlant des usines installées sur plusieurs centaines de km^2. Dans une première étape, nous montrerons en quoi le jeu de la décentralisation au travers d'une structure hiérarchique est poursuivi dans le cadre du LEP pour mieux rentrer dans les objectifs temps réel du système de contrôle. Ensuite, comme dans le cas du SPS, nous passerons à une description plus approfondie des différents constituants du mécanisme de contrôle.

III.1. Le système de contrôle en temps réel du LEP sur une architecture distribuée

Comme pour le SPS, la structure des tâches du LEP peut se répartir en deux niveaux hiérarchiques (figure III.1) :

- Le niveau supérieur consistant en un réseau d'ordinateurs-processus ou *Process Computer* . Chacun a un rôle bien défini :

  - au niveau de la salle de contrôle, l'analyse des alarmes ou la fonction de console ; le fait qu'un ordinateur-processus soit dévolu au système de contrôle central ne lui donne aucune position privilégiée ; ses standards d'interface et son logiciel sont compatibles avec le reste du système ;
  - au niveau de chaque octant du tunnel, la conduite d'un sous-système d'équipements comme la surveillance du niveau de radiation, de la position du faisceau et de l'accès aux zones dangereuses, la radio-fréquence et l'injection (distribution fonctionnelle) ;
  - au niveau de l'accélérateur tout entier, la conduite des équipements répartis le long des octants de l'anneau (distribution géographique).

- Le niveau inférieur consistant en un réseau d'équipements couplés avec des microprocesseurs et conduits par un ordinateur-processus. Un des premiers facteurs de la distribution est le coût des connexions. Si un grand nombre de périphériques sont à surveiller et s'ils sont relativement éloignés du centre de décision, il devient économique de relier ce centre à la zone des périphériques par une seule ligne multiplexée. Le pas suivant consiste à mettre en oeuvre, dans cette zone éloignée, un multiplexage "intelligent", celui-ci pouvant être effectué par un mini-ordinateur, le Process Computer, auquel on fera exécuter des tâches plus nobles que la conduite d'un simple multiplexage.

Si l'on s'arrête à la structure matérielle, les ordinateurs-processus du LEP ne seront pas des mini-ordinateurs comme dans le cas du SPS mais des multiprocesseurs ou collection de micro-ordinateurs partageant un espace d'adressage commun pour l'échange des messages : il s'agit des *Process*

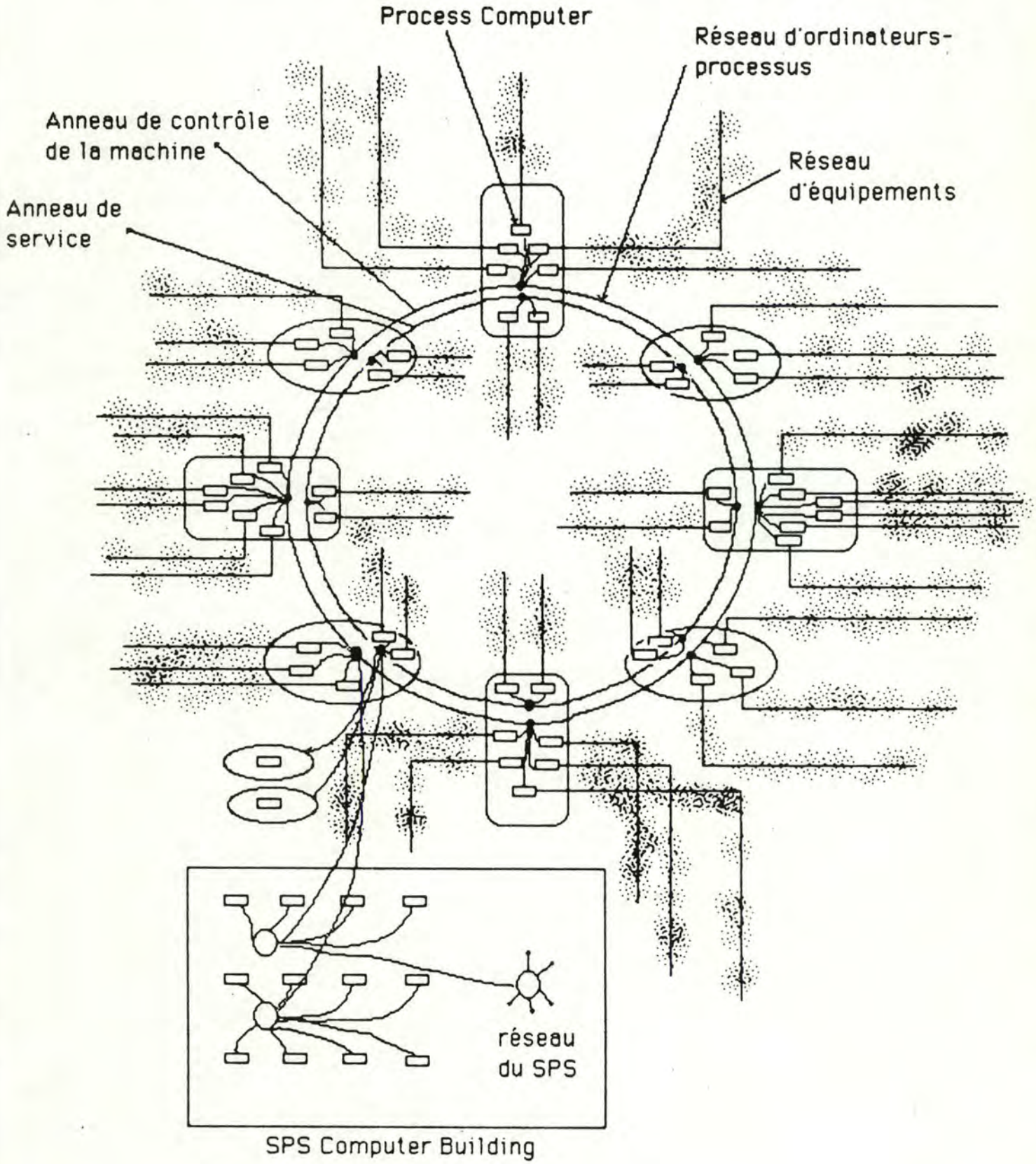


Figure III.1 : Structure des tâches du LEP en deux niveaux hiérarchiques

Control Assemblies ou *PCA'S,* moins chers que les minis et dont plusieurs fabricants peuvent fournir les éléments de base, aspect non négligeable quand on sait que le CERN se soucie de diversifier ses sources d'approvisonnement pour se soustraire du monopole de constructeurs tels que Norsk Data (figure III.2.).

Cette décentralisation supplémentaire permettra de pousser au maximum la notion de parallélisme et de pallier ainsi encore mieux à la lenteur du sous-système de communication pour améliorer le temps de réponse aux différents stimuli extérieurs :

- parallélisme entre les PCA'S dans le cas de lancement par l'opérateur, à partir de la salle de contrôle, de programmes dont les différentes parties sont exécutées dans différents PCA's, quand nécessaire. Cette idée est déjà exploitée au niveau du SPS.
- parallélisme entre le PCA et les équipements.
- parallélisme entre les modules microprocesseur du PCA où chaque tâche temps réel indépendante des autres tâches se verra attribuée la pleine capacité d'un processeur indépendamment des autres tâches de communication et d'E/S.

III.2. Le système de transfert de messages

Vu les distances couvertes par l'accélérateur LEP, pour permettre les communications entre les PCA'S, les responsables ont opté pour le Token Ring de la proposition 802.5 du comité de l'IEEE responsable de la standardisation des réseaux locaux.

En effet, un réseau local en anneau est constitué d'une série de liaisons uni-directionnelles, point-à-point et connectées par des répéteurs. Ces derniers regénèrent et retransmettent les bits qui les atteignent. Ils permettent au réseau de départ de s'étendre sans problème sur de plus longues distances.

Sur un réseau de topologie bus ou arbre, la voie est multipoint

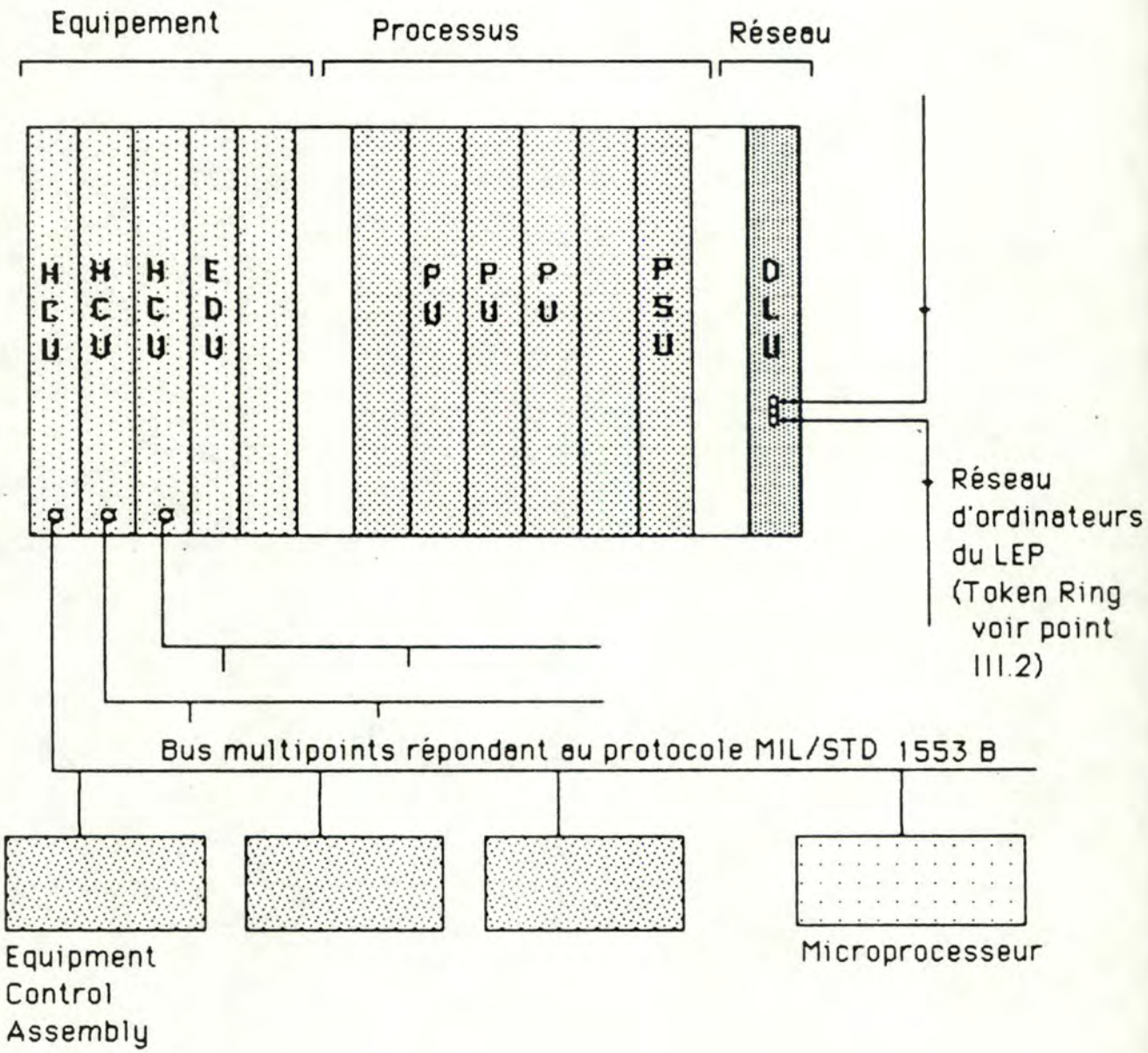


Figure III.2. : Les PROCESS CONTROL ASSEMBLIES

c'est-à-dire que plus de deux équipements peuvent être connectés en même temps sur le même support de communication et que toute station doit être à l'écoute de tous les autres transmetteurs possibles. Ce mode de configuration implique que, pour toute paire émetteur-récepteur quelle qu'elle soit, le signal émis, après atténuation au cours de sa traversée du moyen de communication, doit rencontrer les besoins de niveau minimal de signal pour être détecté par la logique du récepteur et doit rester suffisamment fort pour garder un ratio adéquat du signal sur le bruit. Supposons un réseau de N stations, N*(N - 1) combinaisons sont envisageables et le système doit être construit de manière que la combinaison émetteur-récepteur la plus défavorable délivre des performances raisonnables. La distance et le nombre de stations connectées sont donc des facteurs très limitatifs sur un réseau multipoint. Une solution pourrait être de diviser le moyen de communication en segments au sein desquels un bon niveau de signal peut être maintenu entre toute paire émetteur-récepteur, des amplificateurs ou répéteurs étant utilisés entre segments pour maintenir un niveau de signal adéquat. Cette solution contribue à réintroduire le caractère dépendant du réseau à la bonne marche des répéteurs et à retirer aux réseaux multipoints leur atout majeur par rapport aux réseaux en anneau c'est-à-dire leur passivité. Une autre caractéristique des répéteurs d'un réseau multipoint segmenté est leur aspect sélectif. En effet, une transmission peut être en cours sur chaque segment simultanément et tout répéteur doit connaître l'emplacement de toutes les stations pour savoir s'il doit répéter un paquet donné sur un segment donné. Cependant, sa tâche peut être simplifiée si le segment fait partie de l'adresse de la station.

D'autres avantages du Token Ring non négligeables pour des opérations en temps réel sont qu'il fournit un temps de réponse déterministe c'est-à-dire borné supérieurement, présente un bon comportement sous forte charge et offre suffisamment de possibilités de recouvrement pour être considéré comme un moyen sûr de transport d'informations sur la sécurité de l'accélérateur.

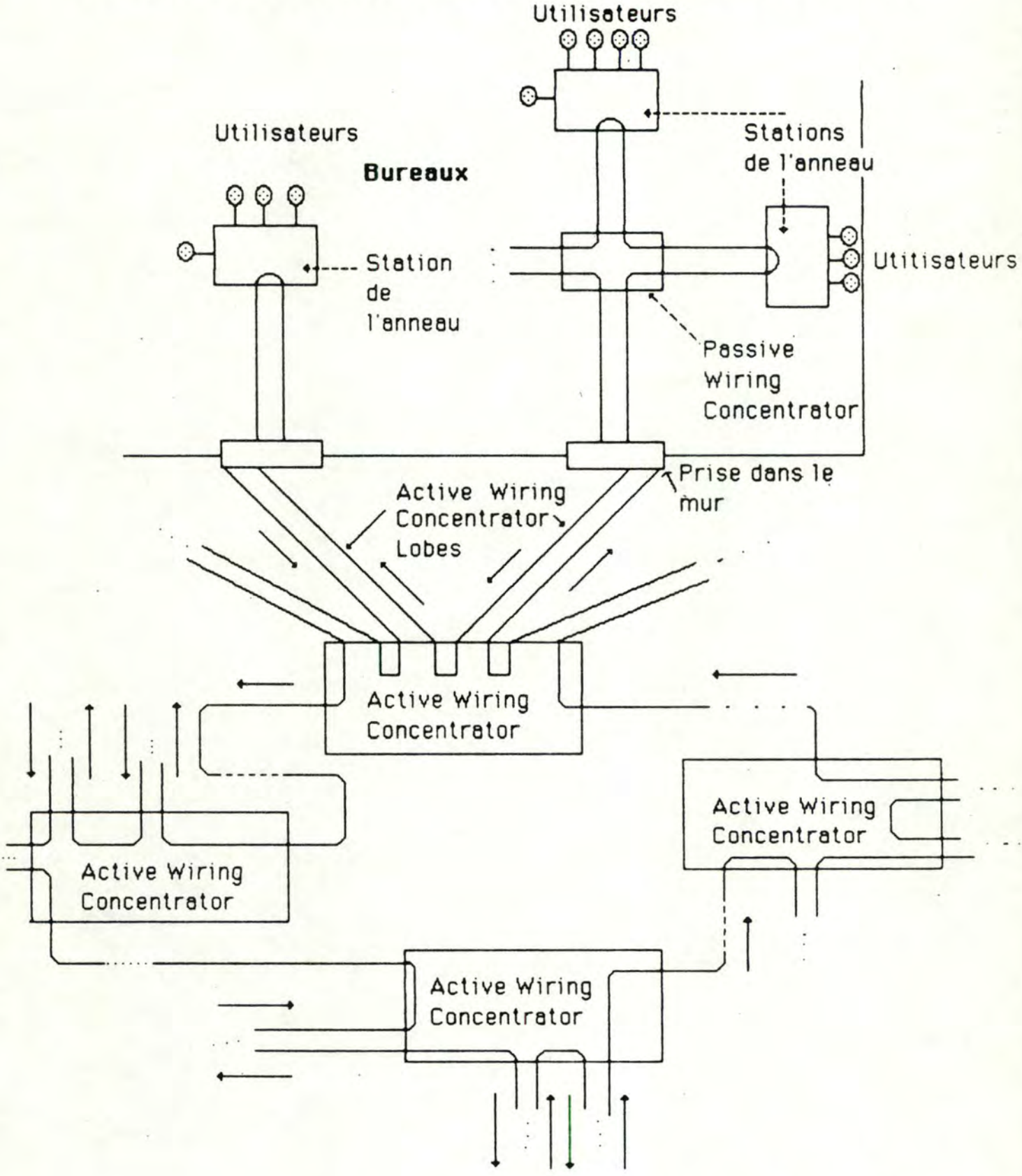
Cette dernière affirmation peut sembler optimiste car comme Saltzer et Pogram (Andrews et Schultz, 5) qui ont implémenté un anneau au MIT le soulignent :

- l'anneau simple dans son entièreté peut être dérangé par la panne d'un simple répéteur ou la coupure d'un câble ;
- la localisation de la faute nécessite une revue minutieuse et complète

de tous les câbles qui passent dans tous les bureaux ;
- l'installation de nouveaux câbles et nouvelles stations, l'adaptation de la longueur des câbles entre les stations après retrait d'anciennes et l'abandon des anciens câbles amènent à des complications pour la maintenance et la tenue à jour de la configuration de l'anneau.

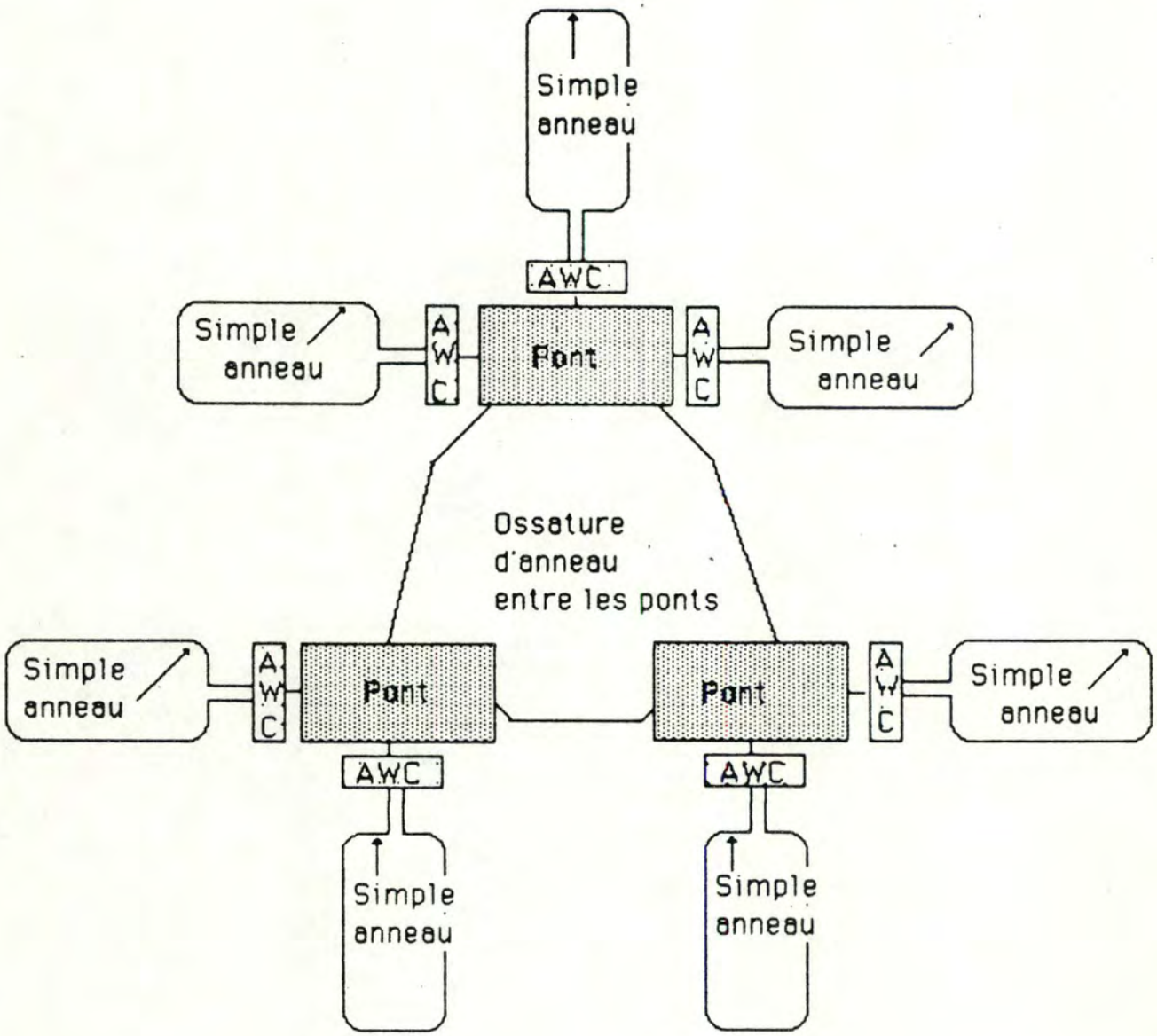
La solution a été un câblage qui épouse la forme d'une configuration hiérarchique radiale (figure III.3.). Dans cette configuration, les stations sont connectées via une "Ring Station" à l'anneau. Ces "Ring Stations" servent d'adaptateurs de réseau ou de répéteurs et sont connectées radialement via des "Active Wiring Concentrator Lobes" à des "Active Wiring Concentrators" qui constituent un point centralisé pour effectuer la maintenance, la reconfiguration ou le diagnostic des pannes et procurer une certaine fiabilité sans compromettre la nature distribuée du contrôle de l'anneau. Ces "Active Wiring Concentrators" sont également des "Ring Stations" sur l'anneau. Beaucoup d'"Active Wiring Concentrators" peuvent être connectés entre eux pour former l'anneau principal ou, dans le cas du LEP, former l'anneau de contrôle de la machine. Un anneau de secours est également prévu ou anneau de service dans le cas du LEP. Un "Passive Wiring Concentrator" peut connecter plusieurs "Ring Stations" pour les mettre en relation avec un "Active Wiring Concentrator Lobe", chaque liaison inter-"Ring Stations" passant par ce point central supplémentaire dont les relais sont alimentés par ces mêmes "Ring Stations". Un "Passive Wiring Concentrator" n'est pas à proprement parler une "Ring Station". Son rôle est de rendre compte de l'état connecté ou pas d'une "Ring Station" suivant qu'elle alimente ou pas ses relais (Pour plus de détails, on consultera Andrews et Schultz, 5).

L'architecture "Token Ring" permet l'interconnexion de plusieurs anneaux par des ponts, chaque pont étant connecté par un "Active Wiring Concentrator" à un anneau donné et jouant le rôle d'une "Ring Station" pour l'anneau sur lequel il est attaché (figure III.4.). Dans cet ordre d'esprit, ces "Active Wiring Concentrators" devront assurer les connexions avec les réseaux du PS et du SPS en effectuant les conversions de protocoles nécessaires. Il est à ajouter que l'anneau ne recourera pas aux fibres optiques malgré leurs grands avantages parce que les liaisons suivent les contours de l'accélérateur où le niveau de radiation est élevé et que les réactions des fibres optiques aux radiations sont mal maîtrisées à ce jour.



(tiré d'Andrews et Schultz, 5, 1982)

Figure III.3 : Configuration hiérarchique radiale du TOKEN RING



AWC correspond à ACTIVE WIRING CONCENTRATOR

(tiré d'Andrews et Schultz, 5,1982)

Figure III.4 : L'interconnexion d'anneaux par des ponts

III.3. Les PCA's ou Process Control Assemblies

Les PCA's sont des assemblages de briques hardware (figure III.2.) et pour définir leur structure, il faut identifier les fonctions pour lesquelles ils sont montés et auxquelles il sera donné suffisamment d'hardware pour qu'elles puissent être effectuées principalement de manière autonome. La distribution des tâches aux PCA'S s'effectuera également soit sur une base géographique, soit sur une base fonctionnelle comme nous l'avons déjà mentionné en parlant de la structure hiérarchique du système de contrôle.

La brique de base du PCA est le *General Purpose Unit* (GPU) c'est-à-dire un microordinateur avec une mémoire propre, un interface-bus pour communiquer avec d'autres GPU's et des mécanismes de réservation d'autres GPU's pour contribuer à la réalisation d'une tâche complexe. Pour une fonction qui implique la communication avec l'extérieur du PCA, le GPU peut se voir ajouter l'interface hardware nécessaire.

Pour la construction d'un PCA, il suffit d'un chassis ("crate") et d'un bus multi-maîtres (c'est-à-dire l'Eurobus de l'Esone Small System Standard) servant simplement à l'échange de messages entre GPU's plutôt que pour l'exécution de programmes, ce qui réduit fortement le flux des données entre unités. Le protocole suivi pour l'échange des messages est celui par valeur où l'on copie le message du module source vers le module destination. Un module peut réserver un autre module par une opération de "Test and Set Instruction" et cette réservation est sujette à un time-out pour éviter une réservation perpétuelle.

Comme pour les satellites du SPS (voir point II.4), les principales activités d'un PCA se résument à assurer :

- la connexion du réseau au Token Ring (voir point III.3.B) ;
- l'accès aux équipements (voir point III.3.C) ;
- l'exécution de procédures de contrôle de processus (voir point III.3.A).

III.3.A. L'interpréteur Nodal et le système d'exploitation

L'interpréteur Nodal (voir point II.4.A) employé par le LEP répond aux mêmes exigences que celles rencontrées par le SPS. Cependant, au lieu d'être exécuté sur un mini, il le sera sur un GPU appelé *Process Unit* (PU).

Dans un système multiprogrammé, le nombre de tâches corésidentes en attente du processeur central peut être trés conséquent à un moment donné ; des algorithmes de priorité sont utilisés pour déterminer qui utilisera le processeur prochainement. L'expérience avec le système de contrôle du SPS a montré que, pour contrôler un accélérateur, il est possible d'être maître du problème en organisant les tâches en un nombre limité de classes et en ne permettant que l'exécution concurrente d'une seule tâche par classe. La même idée a été reprise pour le LEP où un GPU appelé *Process Unit* (PU) sera dédicacé à chaque classe de tâches dont certaines exigeront la présence de l'interpréteur Nodal et l'ensemble des PU's permettra l'exécution en concurrence de tâches dans un environnement multiprocésseurs.

Quelles sont ces classes ? :

- les tâches soumises du réseau, soit normales, soit urgentes et courtes ;
- les tâches programmées sur des tops d'horloge ;
- les tâches conduites par événement qui se produit inlassablement au cours de chaque cycle d'injection des électrons-positons dans l'accélérateur ;
- l'intervention de l'opérateur à distance ou en local.

Pour l'exécution des procédures de contrôle, un système d'exploitation temps réel multiprogrammé sera émulé par une combinaison de plusieurs GPU's. Un composant du système d'exploitation est le noyau temps réel qui sera écrit dans un langage indépendant de l'ordinateur comme CORAL et BCPL pour en augmenter la portabilité et qui sera dévolu à trois types de tâches bien spécifiques :

- le lancement de tâches à la demande d'autres tâches, sur événements programmés (minuterie d'horloge) ou à l'arrivée d'un événement

externe ;
- le retrait du contrôle d'un *PU* ou Process Unit dédicacé à l'exécution d'une tâche quand cette dernière est achevée ;
- l'avortement de tâches enfreignant les règles du système.

Le nombre de modules (un module étant constitué d'une brique hardware de base, le GPU, habillé par du logiciel) de différents types utilisés pour réaliser un PCA varie selon l'application. L'exemple de l'ALARM computer dont le but serait de concentrer au niveau de la salle de contrôle toutes les erreurs qui se sont manifestées sur le réseau, en serait une bonne illustration. Il est constitué d'un module *Data Link Unit* (DLU) (voir point III.3.B) pour dialoguer avec le Token Ring, d'un module *Display Control Unit* (DCU) pour manipuler des écrans graphiques couleurs, d'un module *Mass Storage Unit* (MSU), d'un module *Process Unit* (PU) pour exécuter l'interpréteur Nodal et d'un module PSU pour coordonner le tout.

III.3.B. L'interface-réseau

Le dialogue avec le Token Ring s'effectue via le *Data Link Unit* (DLU) c'est-à-dire un GPU auquel on a ajouté le circuit intégré VLSI contenant tout le logiciel nécessaire à l'échange de messages entre PCA's.

III.3.C. L'interface-équipements

Les équipements sont organisés en famille géographico-fonctionnelle sous la supervision d'un PCA comme c'est le cas, par exemple, de tous les alimentateurs en courant de l'octant numéro 8 de l'anneau.

La connexion entre un PCA et les équipements ne se fera plus via le montage CAMAC (voir point II.4.D.1) et contrôleur de bus multiplex (voir point II.4.D.2) car cette solution du SPS coûte trop cher mais via un bus multipoints répondant au protocole MIL/STD 1553B développé pour le Département Americain de la Défense. Par rapport au bus multiplex qui travaillait à 500

Kbps sur une distance maximale de 200 m, le 1553 offre un débit de 1 Mbps sur 400 m de câble sans répéteurs. La communication bidirectionnelle par blocs ou messages se déroulera par invitation, sous l'initiative de la station primaire ou PCA, celle-ci invitant chacune à leur tour les différentes stations secondaires à émettre.

Au niveau du PCA, un module se charge de l'interface et de l'accès aux différents équipements : le *Highway Controller Unit* (HCU) c'est-à-dire un GPU augmenté du circuit intégré MIL 1553 contenant suffisamment de logiciel pour assurer la transmission en toute sécurité des messages dans les deux sens.

Les interfaces-équipements sont regroupés dans un chassis contenant un microprocesseur maître pour gérer les E/S ou encore appelé un *Equipment Control Assembly* (ECA) connecté à un bus multipoints dépendant d'un PCA. Les équipements peuvent aussi être directement connectés au bus par l'intermédiaire d'une intelligence locale ou microprocesseur (figure III.2.).

Ce microprocesseur, quelle que soit sa localisation, assurera l'usage maximum de l'autonomie des équipements :

- par l'exécution de programmes prévus pour être actifs à des moments bien précis sans intervention du PCA ;
- par l'exécution de Data Module (voir point II.4.D.3) dont le code, au lieu d'être résident dans le PCA, aura été descendu au niveau du microprocesseur de l'interface ; dans ce cas, la petite Base de Données dont il a été question au point II.4.D.3 servira à router l'appel Data Module vers le microprocesseur dédicacé à la conduite de l'équipement spécifié comme premier paramètre dans cet appel (voir chapitre VI pour plus de détails) ;
- par la surveillance régulière des appareils connectés et l'acquisition consommatrice de temps des paramètres de conversion analogique-digital.

Pour dialoguer avec les équipements réapparaît la notion traditionnelle de Data Module en y ajoutant la possibilité dans un seul appel d'invoquer plusieurs équipements d'une même famille pour accroître le niveau de parallélisme, les paramètres et les résultats étant regroupés dans un tableau. L'appel se fera par nom et non par adresse absolue pour faciliter l'insertion et le retrait d'équipements. Pour ce faire, on intègrera dans le PCA le module *Equipment Directory Unit* (EDU) c'est-à-dire un GPU permettant la mise

en correspondance du nom symbolique de l'équipement avec son adresse.

III.3.D. Le sous-réseau d'équipements

Pour connecter les interfaces-équipements au PCA, deux solutions ont été envisagées avant d'opter pour le MIL 1553.

La première idée avait été de recourir à **Ethernet** dont la méthode de partage du moyen de communication est le mode par compétition c'est-à-dire qu'un émetteur A utilise la voie, si elle est libre, dès qu'il est prêt à transmettre. A une certaine vitesse de propagation, si le message de A est trop court, il ne se rendra pas compte que son message a subi une collision suite à l'entrée en émission d'un autre émetteur B car l'écho ne lui en parviendra que quand il aura terminé d'émettre. Pour éviter le passage sous silence de paquets totalement démolis, la longueur minimale d'un message est imposée à 64 octets sur un Ethernet fonctionnant à 10 Mbps c'est-à-dire deux fois le nombre de bits en transit entre n'importe quelles stations ; le champ de données doit au moins être de 46 octets, ce qui correspond à un gaspillage important de la bande de base lorsque la plupart des messages transitant sur la voie de communication PCA-équipements sont très courts comme des mots d'état de 16 bits. L'importance du délai de propagation se manifeste aussi dans le nombre de collisions. Si, après l'entrée en émission d'une station A, une station B devient prête et que le signal de la station A n'a pas encore atteint la deuxième station, pour des raisons de délai de propagation, B verra le canal libre, transmettra et provoquera une collision. En toutes généralités, la méthode de partage d'Ethernet a ses performances dictées par le délai de propagation.

La deuxième idée avait été de choisir le protocole **HDLC synchrone** , basé sur l'élément binaire, permettant les communications de données suivant le mode bidirectionnel.

Toutes les transmissions se font à l'intérieur de trames et chaque trame est conforme au format de la figure III.5. :

- le fanion : la séquence de délimitation de trame;
- l'adresse : le champ d'adresse de la station secondaire;

Fanion	Adresse	Commande	Information	FCS	Fanion
01111110	8 éléments binaires	8 éléments binaires	*	16 éléments binaires	01111110

* Un nombre indéterminé d'éléments binaires qui peut dans certains cas être un multiple d'une dimension de caractère particulière (octet par exemple)

Figure III.5. : Structure de trame HDLC

	Bits	1	2	3	4	5	6	7	8
Format d'information (I)	(I)	0	N(S)			P/F	N(R)		
Format de supervision (S)	(S)	1	0	S		P/F	N(R)		

Figure III.6. : Format du champ de commande en HDLC

- la commande : le champ de commande (figure III.6.);
- l'information : le champ d'information qui peut être un nombre indéterminé d'éléments binaires ou qui peut être absent dans les trames de supervision de la liaison ;
- le Frame Check Sequence : la séquence de contrôle de trame.

Pour distinguer le transfert de l'information proprement dite du transfert de séquences de supervision de la liaison, trois formats ont été définis pour le champ de commande dont nous n'en présentons que deux sur la figure III.6. :

- N(s) désigne le compteur séquentiel émetteur modulo 8.
- N(r) désigne le compteur séquentiel récepteur modulo 8, qui est le numéro de la prochaine trame attendue. Il indique que les trames d'information numérotées jusqu'à N(r) - 1 ont été correctement reçues.
- P/F désigne par (P) une demande de réponse immédiate pour les transmissions primaires et par (F) une réponse au bit P mais, aussi, en mode normal, une trame finale pour les transmissions secondaires.

Trois types de fonctionnement sont possibles avec l'HDLC :

- le mode de réponse normal où à une station primaire correspond une ou plusieurs stations secondaires, où une station secondaire transmet sur invitation de la station primaire qui contrôle la liaison en positionnant le bit P du champ de commande à 1 et où la transmission d'une réponse peut consister en une ou plusieurs trames, la dernière trame de la transmission devant être indiquée de façon explicite par le secondaire en positionnant le bit F du champ de commande à 1. A la suite du positionnement de ce bit, le secondaire doit arrêter de transmettre jusqu'à ce qu'une permission explicite lui soit de nouveau envoyée du primaire. C'est ce mode qui avait été retenu.
- le mode de réponse asynchrone où le secondaire peut initier la transmission sans recevoir explicitement la permission du primaire.
- le mode équilibré asynchrone où toutes les stations peuvent être vues comme combinant les fonctions du primaire et du secondaire.

La décision pour le MIL 1553 s'est effectuée sur base de l'implémentation hardware qui n'est pas notre propos. Nous tenons cependant à donner les principes de base de ce protocole.

Le 1553 travaille également par commandes/réponses c'est-à-dire que les terminaux éloignés reçoivent et transmettent des données suite à l'invitation du contrôleur de bus dont la charge peut être attribuée à n'importe quel terminal. Même si deux terminaux veulent dialoguer entre eux, l'allocation du bus à leur intention doit passer par le feu vert du contrôleur. Il s'agit d'un protocole asynchrone pouvant gérer une liaison de données en mode bidirectionnel à l'alternat, par multiplexage temporel et discrétisant les signaux suivant la technique du "Pulse Code Modulation" avant de les coder en Manchester II-biphase. La voie peut être utilisée en mode point-à-point ou diffusion.

Un message entre un contrôleur et un terminal peut se composer, au minimum, d'un mot de commande et/ou d'un mot d'état de la part du terminal éloigné et/ou de données dont la taille maximale peut s'élever à 32 mots de 20 bits. Un mot de commande de 20 bits reprend, entre autres, l'adresse du terminal de dialogue, un bit indiquant que le contrôleur l'invite à émettre ou à recevoir et un bit de parité. Un message entre deux terminaux qui veulent dialoguer est constitué de deux mots de commande dont l'un est une invitation à émettre et l'autre une invitation à recevoir, deux mots d'état pour renseigner sur le bon déroulement des opérations et les mots de données.

Si, pour HDLC, le transfert d'informations est commandé par les champs N(s), N(r) et P/F du champ de commande ou par les trames de supervision, pour le 1553, le transfert est toujours validé par la valeur du mot d'état. Ce mot d'état est renvoyé soit par le terminal à la demande explicite du contrôleur de bus figurant dans son mot de commande, soit à la clôture d'une transaction pour renseigner sur son bon déroulement. S'il n'est pas émis, que l'on travaille en mode point-à-point et que le Time-out tombe, alors le contrôleur est au courant qu'un problème s'est produit dans l'envoi de données au terminal.

Les bits du mot d'état renseignent soit sur l'impossibilité pour le terminal de recevoir des données, soit sur une erreur dans la réception des messages ou soit sur la volonté d'un terminal d'émettre au cours du polling par le contrôleur des différents terminaux (Pour plus de renseignements, on consultera MIL-STD-1553B, 11).

Nous allons illustrer, par un bref exemple d'envoi de données de l'équipement vers le contrôleur pris en cours de dialogue, la différence d'approche entre l'HDLC et le MIL-STD-1553B.

Présentons d'abord l'HDLC qui fonctionne en mode de réponse normal. La station primaire, par envoi de la fonction de supervision RR, indique à la station secondaire qu'elle est prête à en recevoir de l'information ; le compteur N(R) de son champ de commande accuse réception des trames reçues, numérotées jusqu'à N(R) - 1 et le positionnement du bit P y invite la station secondaire à émettre les trames suivantes N(R), N(R) + 1, ...

La station secondaire émet alors ses trames d'information et conclue en positionnant son bit F à 1. Les trames transmises par cette dernière seront confirmées ou rejetées par la station primaire au prochain transfert de données ou suite à la demande d'entrée en phase de déconnexion.

Dans le cas du MIL-STD-1553B, le contrôleur demande à un terminal spécifié de lui communiquer son mot d'état ainsi que le contenu de son tampon, sur base d'un mot de commande spécifique. Si le bit de "Service Request" est positionné dans le mot d'état, le contrôleur regarde l'information complémentaire disponible dans le mot d'état pour connaître le nombre de mots qui doivent être lus. Si le contrôleur détecte une erreur en cours de lecture, il envoie un mot de commande redemandant la retransmission du contenu du tampon.

Bibliographie de la partie I

Chapitre 1

(1) F. Anceau, Laboratoire d'Informatique et de Mathématiques Appliquées de Grenoble, France, 1980, "Evolution des microprocesseurs et son influence sur l'architecture des systèmes", *Real-time data handling and process control,* North-Holland Publishing Company - Amsterdam, New York, Oxford.

(2) B.Boehm, 1979, "Software engineering: R & D trends and defense needs", *Research Directions in Software Technology,* P.Wagner, Ed. MIT Press, Cambridge, Mass.

(3) CORNAFION, 1981, *Systèmes Informatiques Répartis* , Dunod, Paris.

(4) M.C. Crowley-Milling, Septembre 1983, "Distributed digital control of accelerators", CERN LEP-DI/83-52.

(5) Everett, R.R., Zraket, C.A., and Bennington, 1957, "H.D. SAGE - A dataprocessing system for air defense", *Proc. Joint Eastern Computer Conf,* pp. 148-55.

(6) V. Frammery, 15 juin 1979, "Conception et réalisation d'un système temps réel réparti pour le pilotage d'une grande machine", CERN SPS/ACC/VF/Note Tech. 79-22 .

(7) A.L.Freedman et R.A.Lees, 1977, *Real-time computer systems*, Crane Russak, New York ou Edward Arnold, London.

(8) John Leslie King, December 1983, "Centralized versus Decentralized computing : organizational considerations and management options", *Computing Surveys, Vol. 15, No. 4.*

(9) H.Kopetz , Institut für Technische Informatik, Technische Universität Berlin 1980, "Future requirements and trends - reliability and error recovery", *Real-time data handling and process control,* North-Holland Publishing Company - Amsterdam, New York, Oxford.

(10) James Martin, 1965, *Programming real-time computer systems*, Printice Hall. Inc., Englewood Cliffs,N.J.

(11) James Martin, 1969, *Design of real-time computer systems*, Printice Hall. Inc., Englewood Cliffs,N.J.

(12) A. L. Scherr, 1982, "Distributed data processing", *Proceedings of IEEE Compcon 82 Fall.*

Chapitre II

(1) 10 mars 1977, "Opérations sur le réseau à partir des consoles", SPS/ACC/JPJ/Note/77-6.

(2) 24 April 1980, "What was novel about the SPS control system ? ", DG-DI/MCCM/tr.

(3) ACC Software Section, 01.09.81, "New features for the SPS message transfer system", SPS/ACC/SW-Sect. 81-21.

(4) J.Altaber, 25.4.1977, "Real-time network for the control of a very large machine", SPS/ACC/Reprint/77-10.

(5) J.Altaber, F.Beck, 21.04.80, "The distributed data-base for the Cern SPS control system", SPS/ACC/JA/Report 80-7.

(6) J.Altaber, C.Gareyte, V.Frammery, P.van der Stock, 20.01.82, "Environnement de programmation pour le contrôle en temps réel", SPS/ACC/JA/PVdS/Rapport 82-03.

(7) J.Altaber, V.Frammery, C.Gareyte, P.van der Stock, 20.01.82, "Principles of the operating system for the SPS real-time control network", CERN/SPS/ACC/79-13.

(8) J.Altaber, V.Frammery, C.Gareyte, P.van der Stock, décembre 1978,

"Expression et réalisation du parallélisme et de la distribution au moyen de système interprétatif", SPS/ACC/JA/Rapport 78-37.

(9) J.Altaber, V.Frammery, C.Gareyte, P.van der Stock, 30 July 1981, "Essential features of the CERN SPS distributed control system", SPS/ACC/JA/Report 81-19.

(10) J.Altaber, V.Frammery, C.Gareyte, J.P.Jeanneret, P.van der Stock, novembre 1977, "Contrôle en temps réel avec architecture distribuée", SPS/ACC/Reprint/77-32.

(11) G.Beetham, 20 August 1981, "Overview of the SPS timing system", CERN/SPS/81-17 (ACC).

(12) CORNAFION, 1981, *Systèmes Informatiques Répartis,* Dunod, Paris.

(13) M.C.Crowley-Milling, May 1975, "The control system for the SPS", Lab II-CO/75-3.

(14) C.Gareyte, 04.05.82, "MTS/HDLC", SPS/ACC/CG/Note Int. 82-20.

(15) C.Gareyte, le 8 mars 1984, "Les descriptions temps réel dans SYNTRON",SPS/ACC/Tech.Note/84-5.

(16) C.Macchi, J.-F.Guilbert, 1983, *Téléinformatique : transport et traitement de l'information dans les réseaux et systèmes téléinformatiques,* Dunod informatique, Paris.

(17) C.Michaud, R.Rausch, J.M.Sainson et R.Wilhelm, Octobre 1977, "Système de multiplexage de données du SPS, l'unité de contrôle série et son utilisation", SPS/ACC/JMS/Note tech. 77-15.

(18) E.M.RIMMER de la D.D.Division, 1979, "CAMAC, Notes for summer students, lecture 1".

(19) E.M.RIMMER de la D.D.Division, 1979, "CAMAC, Notes for summer students, lecture 2".

Chapitre III

(1) J.Altaber, F.Beck, R.Rausch, 21 September 1981, "Distributed management in a multiprocessing assembly for real-time control", SPS/ACC/Report 81-23.

(2) J.Altaber, F.Beck, M.C.Crowley-Milling, R.Rausch, 27 October 1981, "Truly distributed control, using one microprocessor per real-time task", SPS/ACC/FB/Report 81-28.

(3) J.Altaber, 11 November 1981, "HDLC procedures. A reader's Digest Approach", SPS/ACC/JA/Techn.Note 81-25.

(4) J.Altaber, R.Rausch, 3 December 1981, "The process/equipment interface for the LEP control system", LEP Controls Note No. 18.

(5) Don W.Andrews and Gary D.Schultz, 1982, "A token-ring architecture for local-area networks : an update", *Proceedings of IEEE Compcon 82 Fall*

(6) F.Beck, 2 May 1978, "Control of a Large Accelerator", SPS/ACC/FB/Note 78-16.

(7) F.Beck, 3 May 1981, "Some fundamental choices in the design of a control system for LEP", LEP Controls Note no.7 F.Beck.

(8) M.C.Crowley-Milling, 26 February 1981, "The Control System for LEP", Lep Controls Note No 1 MCCM/tr.

(9) M.C.Crowley-Milling, 12 January 1982, "The architecture of the LEP control system", Lep Controls Note No.20.

(10) C.Macchi, J.-F.Guilbert, 1983, *Téléinformatique : transport et traitement de l'information dans les réseaux et systèmes téléinformatiques,* Dunod informatique, Paris.

(11) MIL-STD-1553B, 21 September 1978, "Aircraft Internal Time Division Command/Response Multiplex Data Bus".

(12) Jerome H.Saltzer and David D.Clark, 1982, "WHY A RING ? " , *Proceedings of IEEE Compcon 82 Fall*

PARTIE II : TEMPS DE REPONSE ET RESEAU LOCAL

Dans la première partie du mémoire, notre propos a été de présenter les traits généraux des systèmes informatiques temps réel et leur impact sur le choix de l'architecture de tels systèmes (Chapitre I) avant de les appuyer par deux illustrations empruntées à la commande des expériences physiques dans deux accélérateurs du CERN : le SPS (Chapitre II) et le LEP en construction (Chapitre III).

Nous nous proposons, dans le cadre de la deuxième partie du mémoire, de nous attarder plus en détail sur le problème du **temps de réponse** qui constitue une contrainte fondamentale dans un système temps réel (voir point I.2). Les concepteurs de l'architecture du SPS et du LEP ont vu dans la **distribution** un moyen de l'améliorer. Pour rappel, le concept de distribution signifie pour Cornafion (Cornafion, 1) : "interconnexion par un système de communication efficace d'ensembles informatiques constitués d'unités de traitement ou de stockage" (voir aussi point I.4). Pour le SPS, le système de communication est constitué du Message Transfer System (voir point II.3) ; en ce qui concerne le LEP, il sera fait appel au Token Ring d'IBM annoncé pour 1988 (voir point III.2) mais non encore disponible sur le marché.

Cette deuxième partie du mémoire analysera plus en détails la classe générale des systèmes de communication que sont les réseaux locaux sous l'aspect temps de réponse et sous la question du bien-fondé de leur emploi dans un environnement temps réel.

Chapitre IV : LES CONCEPTS DE RESEAU LOCAL ET DE PROTOCOLE D'ACCES

Dans ce quatrième chapitre, nous préciserons la notion de **système de communication** représentée par les réseaux locaux (voir point IV.1) et les objectifs de l'investigation qui sera menée en détail dans le cinquième chapitre (voir point IV.5). Nous donnerons des **réseaux locaux** une représentation sous forme d'un modèle stochastique (voir point IV.2) ainsi qu'une liste exhaustive de **critères** fondamentaux d'analyse (voir point IV.3). Nous verrons que certains critères sont particulièrement liés à notre étude du temps de réponse ; ils ne sont pas indifférents aux **protocoles** définis (voir point IV.4) pour gérer l'accès au réseau par des utilisateurs dispersés.

IV.1. Le concept de réseau local

Pourquoi, comme au CERN, parler tant de réseau local ? L'utilisation d'un réseau local est bien souvent motivée par l'existence d'un grand ensemble de postes de travail qui doivent dialoguer entre eux et avec une base de données centralisée. Plutôt que de connecter chaque machine locale à la base de données, il suffit de les connecter à un réseau local et de mettre ce réseau en relation avec la base de données envisagée.

Deux domaines d'applications s'articulent particulièrement sur les réseaux locaux :

- celui du contrôle industriel de processus (manufacture, raffinerie...) où un grand nombre d'appareils conduits par microprocesseur doivent travailler en collaboration ;
- celui de la bureautique où des stations de travail se partagent l'information : voix, graphique, images, données... et des ressources communes comme une imprimante à laser...

William Stallings (Stallings, 5, p4) définit un réseau local comme : "un réseau de communication pour interconnecter une variété d'appareils (ordinateurs, terminaux, périphériques, capteurs, transmetteurs-récepteurs de télévision) s'échangeant des données sur une petite surface".

Un réseau local est généralement de propriété privée. Les caractéristiques qui permettent de le différencier d'un réseau à longue portée (réseau public) peuvent se résumer à :

- de hauts taux de transferts (de 0,1 à 100 Mbps) alors que les réseaux publics ne peuvent offrir que 48 à 64 Kbps à moins de disposer de lignes dédicacées ;
- de courtes distances couvertes (de 0,1 à 50 km) ;
- de faibles taux d'erreur (de 10^{-8} à 10^{-11}).

Par ailleurs, nous pouvons également le mettre en opposition avec un système multiprocesseur qui recourt à des taux de transfert encore plus élevés, à des distances encore plus courtes et à des taux d'erreurs encore

plus réduits.

William Stallings effectue la distinction entre trois types de réseaux locaux :

- le ***Local Area Network*** (LAN) : il s'agit d'un réseau local à objectif général qui peut interconnecter un large éventail d'appareils comme des mini-ordinateurs, de grosses configurations, des terminaux, des périphériques,... et qui peut véhiculer les données, la voix, la vidéo ou les graphiques à des vitesses de l'ordre de 1 à 10 Mbps ; *c'est sur ce type de réseau que portera notre étude ;*

- le ***High-Speed Local Network*** (HSLN) : il s'agit d'un réseau à grande vitesse (50 Mbps) à objectif particulier : ce type de réseau assure des débits importants entre les équipements d'une salle d'ordinateur, tels que de grosses configurations et des périphériques rapides ; le prix élevé des connexions sur son cable exclut son emploi pour raccorder des mini-ordinateurs comme ceux du LEP ou du SPS ; l'exemple le plus ancien et le plus populaire est celui de l'HYPERchannel (de Network Systems Corporation) ;

- le ***Computerized Branch Exchange*** (CBX) ou ***Private Autocommutation Branch Exchange*** (PABX) : il est surtout utilisé pour l'échange de conversation où la voix est de plus en plus souvent digitalisée mais où, de plus en plus, on ajoute l'échange de données ; il se rencontre très couramment dans les entreprises et est souvent relié au réseau téléphonique public pour mettre les membres de l'organisation en rapport avec le monde extérieur ; il s'agit d'un réseau à commutation de circuits alors que les autres sont, le plus souvent, à commutation de paquets ; il atteint de faibles taux de transfert et adopte la topologie en étoile.

Il faut remarquer que la distinction introduite par William Stallings entre les LAN et les HSLN n'est pas adoptée par tous les auteurs qui les classent tous les deux dans la catégorie des LAN mais nous trouvons cette nuance intéressante et nous l'adopterons.

IV.2. Le modèle stochastique de référence

Sous ce point, nous détaillerons la figure IV.1 schématisant sous forme d'un modèle stochastique le fonctionnement d'un réseau local LAN. Ce schéma nous servira de modèle de référence tout au long des lignes des chapitres IV et V.

Les utilisateurs des stations connectées au réseau engendrent des messages qui viennent gonfler les files d'attente auxiliaires de ces dernières. Ces messages sont habillés par la station avant leur passage dans la file auxiliaire c'est-à-dire que des bits de service sont ajoutés aux bits utiles porteurs d'information ; ces bits de service servent à la définition de l'adresse du destinataire du message, à la synchronisation des horloges de l'émetteur et du récepteur (fanion), au contrôle d'erreur,...

Chaque message qui a atteint la tête de la file d'attente de sa station est placé par un serveur intermédiaire dans une file centrale unique ; cette file symbolise le temps d'attente subi par les messages dû au partage de la voie de communication unique entre plusieurs stations concurrentes.

Ce temps dépend donc de la discipline de gestion de la file centrale, qui est étroitement fonction de la conception du réseau et de ses conditions de fonctionnement (voir point IV.4). En effet, parmi les messages présents dans la file centrale, certains seront effectivement transmis, sans incidents, au destinataire mais d'autres, en cours de transmission, subiront des collisions amenant à leur retour en file centrale et à une augmentation de leur temps d'attente. Des collisions peuvent se produire car plusieurs stations décident simultanément de transmettre leurs messages.

Les différentes notions de temps telles que le temps d'accès, le temps de propagation, ... reprises sur la figure seront expliquées au point suivant.

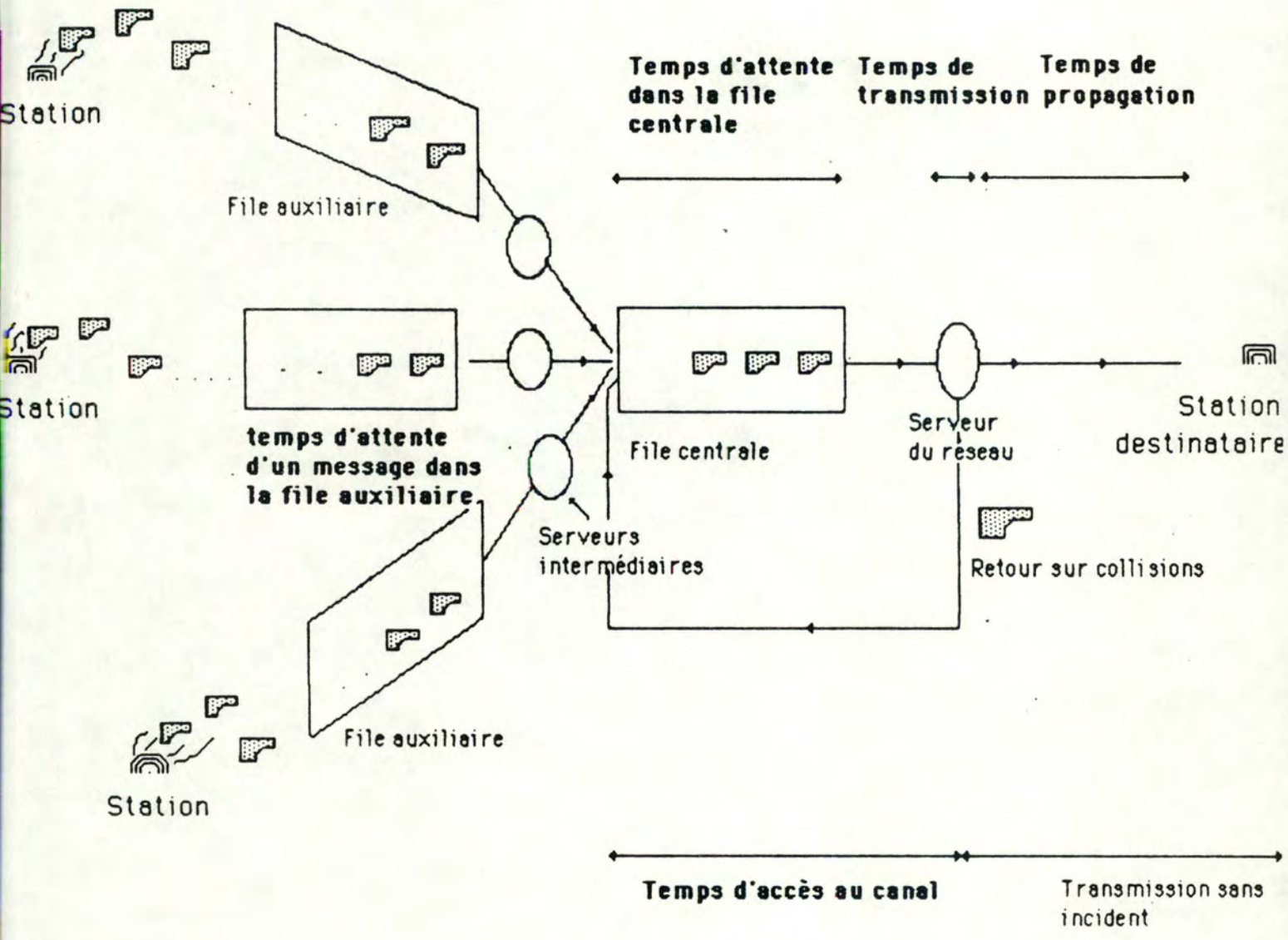


Figure IV.1. : Modélisation du temps de réponse d'un message dans un réseau local LAN

IV.3. Critères d'analyse d'un réseau local

Sur quels critères pouvons-nous analyser un réseau local LAN ?

1. Le **temps de réponse d'un message** ("message delay") : il s'agit du temps écoulé entre la génération complète d'un message dans une station-source jusqu'à son arrivée complète dans une station-destination ; notons que, contrairement à la définition classique du temps de réponse dans une application de gestion, nous n'incluons pas ici comme dans la définition proposée au point I.2. par J. Martin, le temps de génération et de retour de la réponse ; sur base de la figure IV.1., nous pouvons y repérer les composants suivants :

   - le temps d'attente ("queuing time") d'un message dans la file auxiliaire de la station où il a été reçu ; il s'agit du temps nécessaire à ce message pour atteindre la tête de cette file ; ce temps résulte évidemment du fait qu'il peut y avoir plusieurs messages à transmettre par station ;
   - **le temps d'accès au canal ("channel acquisition delay")** : il correspond au temps qui sépare l'arrivée d'un message en tête d'une file auxiliaire, prêt à être transmis, de sa réelle transmission ; en d'autres termes, si l'on se réfère à notre figure IV.1., il s'agit de la somme des temps d'attente subis pour chaque passage dans la file centrale des messages attendant d'être servis par le réseau, c'est-à-dire d'être transmis sans incident sur le canal ; être transmis sans incident signifie que le temps d'accès prend également en compte tous les retours d'un message dans la file d'attente parce qu'il a subi une collision en cours de service, par exemple (voir figure IV.1.) ; ce temps n'est pas à confondre avec le "*scan time*" qui équivaut à la somme du temps d'accès d'un message au canal et de son temps de transmission défini au point suivant ;
   - le temps de transmission du message : rapport entre la longueur totale du message et la vitesse du moyen de communication ;
   - le délai de propagation, c'est-à-dire le temps mis par un bit placé sur le réseau pour atteindre sa destination : ce temps intègre deux éléments :

- le délai de propagation inhérent au moyen de communication ;
- le retard éventuel introduit par chaque station qui lit le message présent sur le réseau pour voir s'il lui est destiné (il peut aller d'un bit à un maximum d'un message).

2. La robustesse ou l'insensibilité du réseau aux erreurs, au bruit sur le canal et à la mauvaise information (des horloges non synchronisées, pauvre estimation des paramètres du système comme l'utilisation du canal) ; cette robustesse est liée à :

   - la fiabilité (voir point I.3) : un réseau devrait pouvoir opérer en mode dégradé même en cas de pannes de station individuelle ou de mauvais fonctionnements ;
   - le taux d'erreurs généré.

3. Le degré de confidentialité c'est-à-dire la limitation de l'accès aux personnes autorisées.

4. La mesure de la flexibilité du réseau, selon différents points de vue :

   - fonctionnel (par la possibilité de faire évoluer les missions qui lui sont confiées : le réseau devrait pouvoir supporter des classes de trafic et des niveaux de priorité différents pour permettre le développement des réseaux de données à service intégré) ;
   - de réalisation (par la facilité de remplacement du software par du hardware) ;
   - géographique (par la possibilité de migration des équipements) ;
   - opérationnel (par la capacité de reconfiguration et d'adaptation dynamique à de nouvelles situations).

5. La **charge** et les **débits** d'un réseau :

   - la **charge** est le nombre moyen de bits à émettre par seconde par l'ensemble des stations ; ces bits se répartissent en bits utiles c'est-à-dire porteurs d'information et en bits de service c'est-à-dire ajoutés par la station aux messages qui y sont générés ; ce facteur peut s'exprimer par la somme des produits des taux moyens d'arrivée des messages par seconde dans chaque station (λ_i)

par la longueur moyenne de ces messages habillés (L_i) par la station ; arrivés en tête de la file d'attente auxiliaire, les messages des différentes stations entrent dans la file centrale pour accéder au réseau ; nous emploierons à l'occasion l'expression "sous forte charge" pour désigner un réseau tel que, en moyenne, la file centrale contienne un message issu de chaque station ; dans le cas contraire, nous parlerons de "faible charge" ;

- le **débit réel** est le nombre moyen de bits utiles et de service émis par unité de temps dans le réseau ; ils proviennent des messages nouvellement émis par les stations ;
- le **débit en ligne** est le nombre moyen de bits transmis sur le réseau par unité de temps ; il intègre le débit réel ainsi que le trafic supplémentaire engendré par les retransmissions des messages entrés en collision ; il faut ajouter que nous ne traitons pas des transmissions supplémentaires dues aux acquittements et aux retransmissions pour cause d'erreurs.

Citons, en guise d'illustration, le cas d'un Ethernet à 2,94 Mbps. A tout message généré qui contient entre 368 et 12.000 bits utiles, la station ajoute les bits de service suivants :

- un préambule de 64 bits ;
- une adresse de destination du message de 48 bits ;
- une adresse d'origine du message de 48 bits ;
- une information sur le type du message de 16 bits ;
- un champ de contrôle de 32 bits.

Aux bits utiles d'un message sont donc ajoutés 208 bits de service.

Si le taux moyen d'arrivées par station est de 50 message/sec, que chaque message est constitué de 368 bits utiles et que le nombre de stations s'élève à 100, la charge s'élève à 2 880 000 bps. Si, pour un débit en ligne de 95% c'est-à-dire de 2 793 000 bps, 10 % du trafic est dû aux retransmissions pour collisions, le débit réel s'élève à 2 513 700 bps c'est-à-dire aux 98,8% de la charge.

Ces critères analysent plutôt l'intéraction d'un réseau local avec son environnement mais ne constituent pas en eux-mêmes une caractéristique propre à un réseau local, c'est la raison pour laquelle ils ont été présentés en dernier point.

Dans une application temps réel, il est vital qu'un message généré à une station-source par un programme d'application soit reçu à la station-destination endéans un certain temps après sa génération par l'émetteur. *De ce fait, le critère le plus important sous lequel nous analyserons les réseaux locaux LAN sera celui de temps de réponse (critère 1).* Nous essaierons de faire ressortir les réseaux locaux LAN qui peuvent certifier la délivrance de tout message endéans un intervalle de temps spécifié. Ce critère doit bien entendu être analysé dans les conditions de fonctionnement les plus défavorables c'est-à-dire sous forte charge (critère 5). *Nous étudierons donc les intéractions entre ces deux critères pour rentrer dans les objectifs d'un système temps réel.*

Au lieu de ne choisir que les réseaux susceptibles de respecter la contrainte précitée, il aurait pu être intéressant d'analyser l'ensemble des LAN sous le critère du pourcentage de messages reçus par le récepteur avec un retard inférieur à un temps donné. En effet, dans un système temps réel, quand la réception dépasse le temps prescrit, le message doit être considéré comme perdu, qu'il soit reçu ou non par la station destinataire. Il existe cependant des cas où un retard exagéré peut être toléré comme, par exemple, dans une application distribuée de surveillance de construction d'automobiles (cela pourrait très bien s'adapter à la surveillance d'un accélérateur), les stations essaient de suivre la trace de l'objet en mouvement en recourant à leurs observations locales et aux observations communiquées par les autres stations. Etant donné que la voiture se déplace à une vitesse soit v, tout délai de transmission, soit t, amène une erreur de position de v*t. Ceci implique une limite supérieure sur t. Si, pour certaines mesures, t est trop grand, cette situation n'est acceptable que si elle se produit exceptionnellement car elle peut conduire à l'incertitude de la position calculée de l'objet.

Lors de notre explication de la notion de temps de réponse d'un message, nous indiquions qu'un de ses composants, le temps d'accès, dépendait des conditions de fonctionnement du réseau. Nous pouvons maintenant préciser notre pensée...

IV.4. Le concept de protocole d'accès et la standardisation

Outre la technologie d'un réseau et son moyen de transmission, le protocole de contrôle d'accès au moyen de communication constitue un élément-clé dans le coût et la performance d'un réseau. Le protocole vient de la nécessité de partager un simple canal de communication entre une communauté d'utilisateurs distribués pour empêcher que des messages en provenance de différentes sources ne soient transmis simultanément et ne se détériorent respectivement. Le protocole se présente sous la forme d'un algorithme distribué exécuté par toutes les stations ou d'un algorithme centralisé exécuté par une station centrale pour coordonner l'accès des différentes stations au canal unique, au prix, malheureusement, de retards accrus dans la transmission des messages.

James F. Kurose (Kurose, Schwartz et Yemini, 2) affirme que jusqu'à maintenant, très peu de recherches ont été consacrées à l'examen des problèmes de conception de protocoles d'accès contraints par le temps. L'*Institut National de Recherche en Informatique et en Automatique* (ou INRIA, Paris) en la personne de Monsieur Le Lann y travaille actuellement. Jetons cependant un regard sur l'état de la standardisation des protocoles d'accès pour les LAN dont l'expérience peut nous profiter.

IV.4.A. La standardisation

Le développement du marché des réseaux locaux LAN dépend de la disponibilité d'interfaces bon marché dont le prix doit être moins élevé que le coût de l'équipement en lui-même. Cette condition ainsi que la complexité des protocoles ordonne d'adopter la solution de l'intégration à grande échelle. Or, les constructeurs de puces se refuseront à engager les ressources nécessaires sans l'assurance d'un marché solide pour écouler leur marchandise. Un standard LAN assurerait le volume et permettrait la communication entre équipements d'origines diverses. L'*Institute of Electrical & Electronics Engineers Computer Society* décida alors de

fonder en février 1980 l'IEEE Project 802 pour établir un standard de LAN définissant un ensemble d'interfaces et de protocoles (voir figure IV.3.).

Un des impératifs de ce groupe fut d'intégrer le modèle des réseaux LAN dans le Modèle de Référence pour l'Interconnexion de Systèmes Ouverts mis au point par le SC-16 de l'ISO et qui est composé de sept couches (voir figure IV.2.) (pour plus de détails, on consultera Myers, 4). Ce système de couches permet de diviser une tâche très complexe en plus petites, chacune relativement indépendante des autres. Un message originaire d'une couche supérieure doit traverser toutes les couches inférieures avant de pouvoir franchir le moyen de communication et regravir les couches correspondantes opposées pour finalement arriver à la couche faisant vis-à-vis à la couche émettrice.

Les deux premières couches de l'ISO sont taillées sur la technologie du réseau utilisée dans certains types d'opérations. La *couche physique* (niveau 1) établit, maintient et libère une liaison physique entre deux types d'équipement. La couche suivante *"liaison de données"* s'occupe de l'envoi de paquets entre appareils. Si le taux d'erreur au niveau physique n'est pas acceptable, le niveau liaison peüt inclure un moyen de détecter les erreurs. Les trois niveaux supérieurs - *session*, *présentation* et *application* - s'occupent du transfert de données bout à bout. Ces niveaux et les deux intermédiaires - *réseau* et *transport* - ne sont pas du ressort des réseaux locaux LAN.

Le standard 802 repose sur trois couches qui procurent les fonctions des niveaux 1 et 2 de l'ISO (voir figure IV.2.) :

- La couche *"Logical Link Control"* (ou LLC) s'occupe de l'établissement, de la maintenance et de la libération d'une liaison logique fiable entre appareils.

- La couche *"Physical Signaling"* s'occupe de la nature du moyen de transmission, des détails de connexion d'un appareil et de signaux électriques.

- Entre les deux, la couche *"Media Access Control"* (ou MAC), qui constitue avec la couche LLC le pendant de la couche *liaison de données* de l'OSI, supporte les fonctions dépendantes du moyen de communication et utilise les services de la couche physique pour

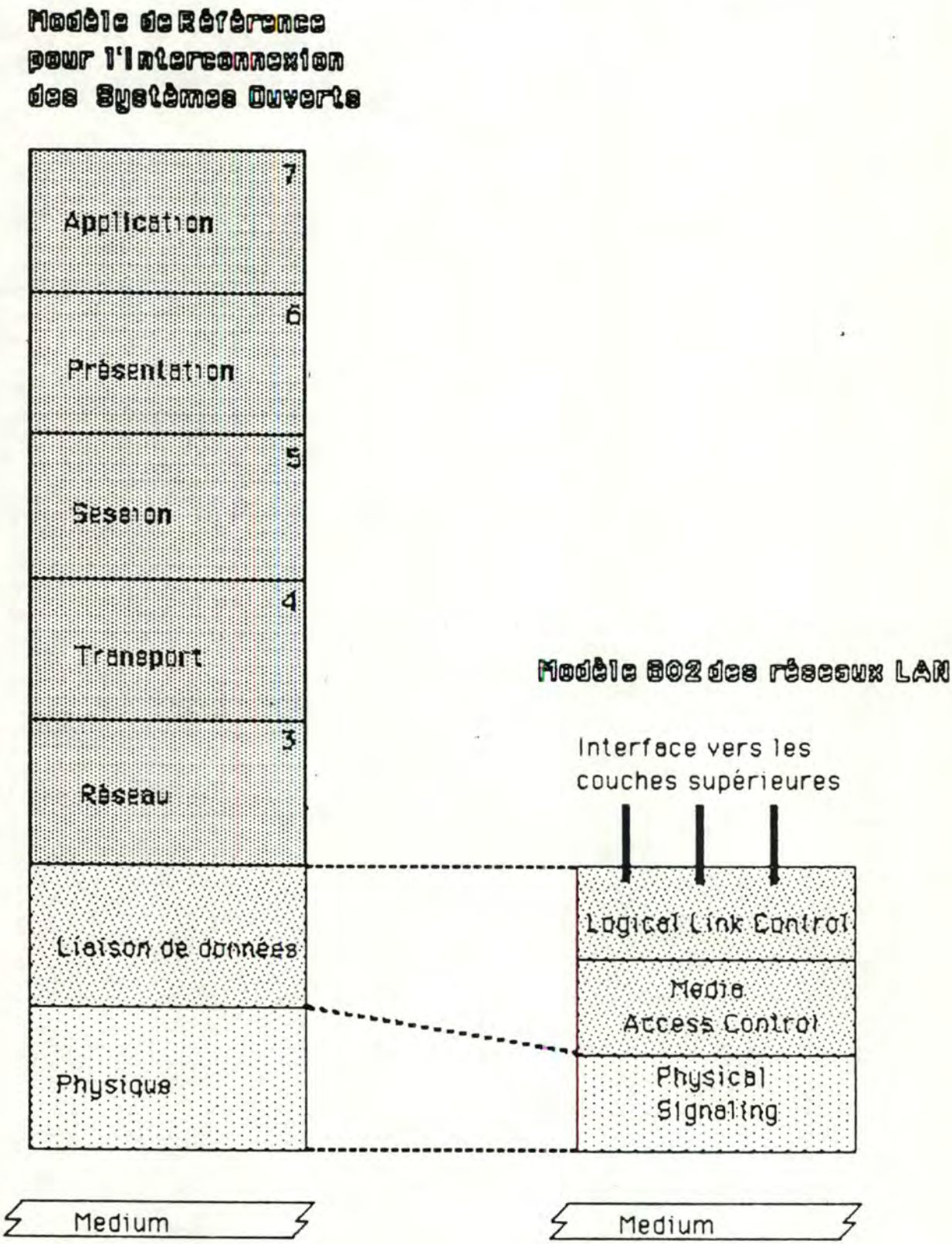
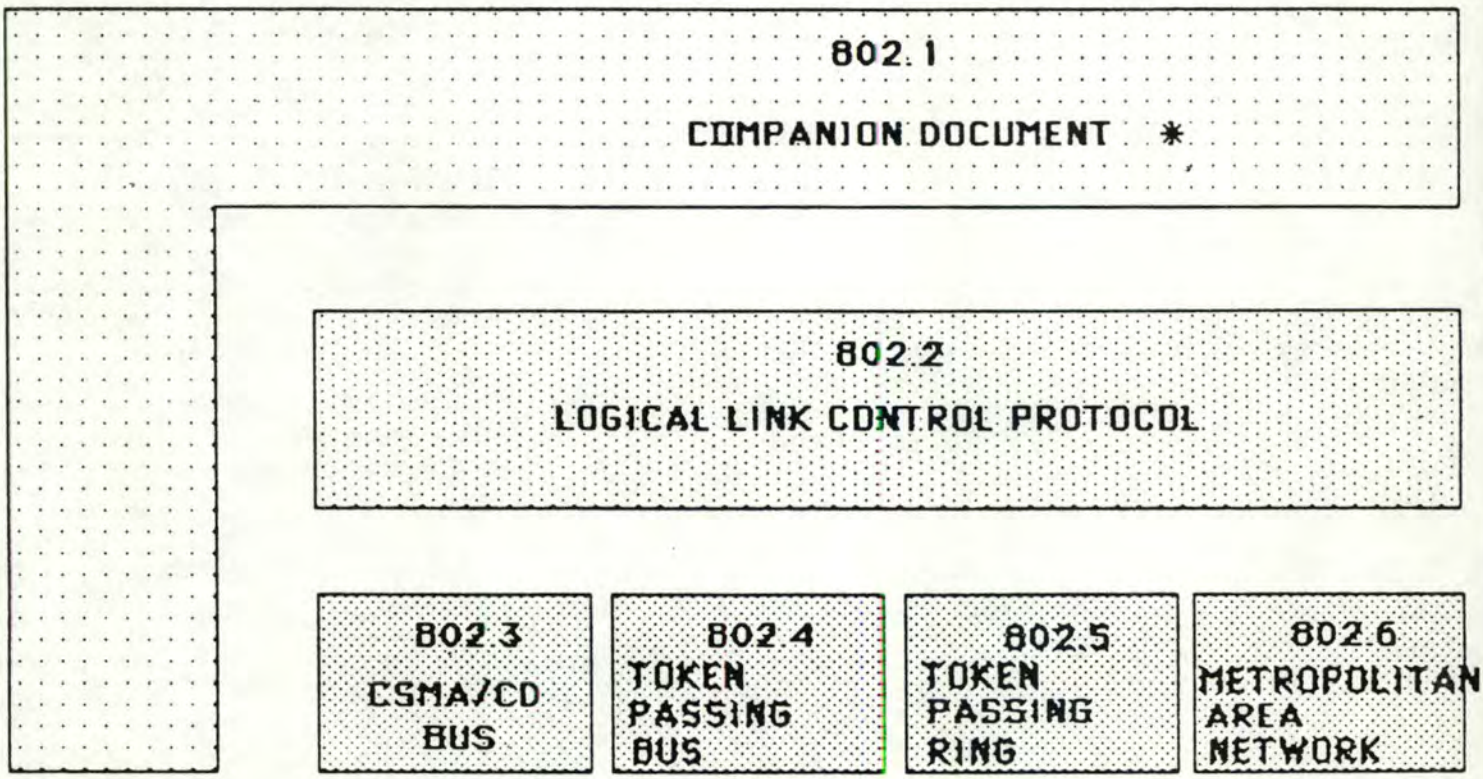


Figure IV.2. : Intégration du modèle des LAN dans le modèle de l'OSI



* Le "Companion Document" gère l'interface avec les couches supérieures
Il se consacre à d'autres questions qui y sont liées :

- l'intercommunication de réseaux
- la gestion de l'adressage et du réseau.

Figure IV.3. : Famille des standards de l'IEEE PROJECT 802

procurer ses services à la couche LLC.
Cette couche s'occupe de la gestion du partage entre utilisateurs d'un moyen unique de transmission de façon à ce qu'un seul équipement ne transmette à la fois : elle s'occupe donc de la mise en oeuvre des protocoles d'accès. Elle n'est pas explicitement présente dans le modèle OSI car ce dernier s'applique aux réseaux à longue portée, publics qui sont maillés et constitués de liaisons point-à-point où le problème ne se pose donc pas.
Un protocole unique ne pouvait convenir à toutes les topologies car le produit cartésien (architectures) X (protocoles d'accès) bien que possible amène à des combinaisons inefficaces.
Trois classes de protocoles ont été étudiées jusqu'à maintenant :

- le protocole *"Carrier-Sense Multiple Access with Collision Detection"* (CSMA-CD) (Norme IEEE 802.3) ;
- le protocole du *"Token Passing"* qui propose les techniques du *"Token Bus"* (Norme IEEE 802.4) et du *"Token Ring"* (Norme IEEE 802.5) ;
- le protocole "*Metropolitan Area Network*" (Norme IEEE 802.6) qui couvre plusieurs dizaines de km à haut débit, pour lequel peu de travail a été réalisé et sur lequel nous ne nous étendrons pas.

Tous ces standards (discutés dans Stallings, 5 et Myers, 4) à l'exception du "Token Ring" ont été approuvés par l'IEEE et sont maintenant proposés à titre de standard international auprès de l'Organisation Internationale de Normalisation.

Quel sera l'avenir de ces standards ?

Si l'expérience avec les méthodes d'accès "CSMA-CD", "Token Bus" et "Token Ring" révèle des différences de performances et de coûts, la méthode la moins chère et la plus performante deviendra prédominante dans les 5 à 10 années à venir. Elle pourrait alors être choisie comme **le standard**. Si les différences de performances et de coûts s'avèrent faibles et difficiles à quantifier et si chaque méthode d'accès est prouvée la meilleure pour un groupe particulier d'applications, on peut s'attendre à ce que les trois méthodes subsistent. Un troisième scénario est possible et c'est celui qui a notre faveur. Des chercheurs développent de nouvelles méthodes d'accès. Une d'entre elles pourrait l'emporter et faire l'objet d'une nouvelle standardisation. Ce résultat pourra élever le coût des réseaux mais une

meilleure adéquation de la nouvelle méthode à une application peut être à ce prix.

Dans cette optique, nous proposerons, dans le chapitre suivant, une classification des protocoles d'accès englobant ceux proposés comme standards par l'IEEE et d'autres développés par des groupes de recherche ou des constructeurs pour caractériser les différentes approches.

IV.4.B. Réflexions sur une orientation possible de la standardisation

---

Le risque de collisions dont la couche MAC doit éviter l'occurence pourrait être traité au niveau supérieur LLC s'il s'arrangeait pour présenter le message à la couche MAC au bon moment.

On pourrait prévoir, au niveau LLC, un tri des messages par station selon le temps limite pour lequel un message doit avoir été envoyé et un système de priorités temporelles comme le *"Deadline Driven Scheduling Program"* : dans ce cas, les priorités statiques assignées aux messages croîtraient en fonction du temps restant pour transmettre le message... En effet, dans la pratique (systèmes temps réel), plus l'âge d'un message non encore envoyé augmente, plus ce message tend à perdre sa valeur : il devient donc de plus en plus urgent de l'envoyer. Cela fonctionne évidemment pour un taux d'utilisation du canal inférieur à 1. Il est en outre possible de retirer sélectivement une partie des messages temps réel, lorsque leur âge augmente et que la perte de leur valeur est anticipée. Un exemple typique de trafic en temps réel est une conversation qui nécessite la délivrance de messages endéans les 50/100 msec, avec une perte tolérable inférieure à 1 % du trafic total.

Le protocole de la couche LLC pourrait aussi inclure dans ses messages des informations permettant aux autres stations d'être au courant de la station dont le message à transmettre est le plus urgent car proche de temps limite (seulement de tels champs ne sont pas prévus par la standardisation). Comme exemple, nous pouvons citer le cas d'une station A qui doit avoir émis son prochain message pour 12 h et d'une station B dont l'échéance du message

tombe à 12h 1 min, B s'inclinera devant A pour qu'ils accèdent dans l'ordre A,B au canal et non en même temps provoquant de ce fait, une collision...

IV.5. Conclusion : objectifs de notre étude des réseaux locaux

Nous disposons maintenant de tous les éléments pour affiner les objectifs de notre étude des systèmes de communication satisfaisant la contrainte du temps de réponse. Notre travail consistera à étudier l'impact des différents **protocoles d'accès** (voir point IV.4) utilisés par les **réseaux locaux LAN** (voir point IV.1), disponibles sur le marché ou en laboratoire, sur le **temps d'accès au canal** (voir point IV.3, critère numéro 1 : ce retard est un composant du temps de réponse d'un message) subi par un message dans les conditions les plus défavorables, c'est-à-dire **sous forte charge** (voir point IV.3, critère numéro 5).

Chaque protocole sera présenté dans ses grands traits. Puis, nous nous pencherons essentiellement sur l'analyse de son adéquation à présenter une borne supérieure au temps d'accès dans le cas le plus défavorable (pour rentrer dans les contraintes de temps de réponse d'un système temps réel) sur base, si possible, d'un modèle analytique et le cas échéant d'une illustration.

Chapitre V : CLASSIFICATION DES DIFFERENTS PROTOCOLES D'ACCES : DETERMINISME OU PROBABILISME ?

Dans ce chapitre, nous nous proposons de présenter sous le formalisme d'**arbre binaire** une classification des différents protocoles d'accès qui remplissent la fonction de la couche "Media Access Control" du standard 802 des réseaux LAN (voir point IV.4.A).

Cette découpe en arbre binaire s'inspire d'une idée présentée par Monsieur Le Lann dans le cadre d'un cours d'introduction aux réseaux locaux LAN donné au CERN en décembre 84 (Le Lann, 16). Nous avons fouillé et approfondi le sujet en nous appuyant sur la littérature actuelle pour pouvoir finalement présenter notre classification dans sa version définitive.

Cette découpe permettra de regrouper autour des feuilles les différents protocoles d'accès sur base de critères originaux (voir point V.1), par rapport à la décomposition classique de la littérature (voir point V.2). La multitude des protocoles résulte de la multiplicité des types de voie, de la variété des comportements des émetteurs et des divers objectifs recherchés. Il peut arriver que l'exploitation d'une voie s'appuie simultanément sur plusieurs protocoles, en particulier lorsqu'on a affaire à plusieurs familles d'émetteurs. Nous nous restreindrons aux méthodes élémentaires que l'on peut, bien entendu, combiner pour implanter une politique réelle de partage.

Nous effeuillerons progressivement notre arbre en étudiant, pour chaque feuille (voir points V.4 à V.10), les protocoles qu'elle rassemble. Après une présentation des principes de base d'un protocole (voir sous-point V.?.?.1), nous poursuivrons son étude par l'analyse de son temps d'accès au canal sous forte charge (voir point V.?.?.2), objectif principal assigné à cette deuxième partie du mémoire (voir point IV.5). Cette analyse, généralement étoffée d'un modèle analytique (dont les conventions sont présentées dans le point V.3), permettra d'établir le **caractère déterministe** (fondamental pour les systèmes temps réel) **ou probabiliste** du protocole, c'est-à-dire sa capacité de présenter avec certitude (déterministe) ou pas (probabiliste) à tout message en attente à une station, une borne supérieure dans son temps d'accès au moyen de communication. Dans certains cas, nous complèterons notre analyse par une illustration ou par une critique (voir sous-points V.?.?.3 et V.?.?.4).

Il est important de souligner que nous n'aborderons pas le problème des erreurs pour cause de parasites, de bruit... En effet, les erreurs de transmission présentent une probabilité finie d'occurence et rendent tous les réseaux non déterministes. Nous supposerons donc le taux d'erreurs

négligeable.

Nous conclurons ce chapitre par un certain nombre de commentaires résultant de nos investigations et par une discussion sur le bien-fondé du choix du Token Ring, contrôlé par le protocole de même nom, pour connecter les satellites du LEP (voir point V.11).

V.1. Classification des protocoles d'accès selon un arbre binaire

La classification par arbre binaire repose sur les trois éléments principaux constituant cet arbre :

- les noeuds non terminaux correspondent au point de départ de deux arcs vers des noeuds inférieurs, d'où le nom d'arbre binaire ; ils représentent les critères de classification des différents protocoles d'accès ;
- les arcs, partant d'un même noeud, donnent les valeurs alternatives du critère établi à ce noeud ;
- les noeuds terminaux ou feuilles reprennent les ensembles de protocoles homogènes du point de vue des valeurs des critères accumulés le long des arcs qui y aboutissent.

Avant d'aborder l'étude des différentes feuilles (voir points V.4 à V.10), nous vous proposons de parcourir les noeuds et arcs constituant notre arbre binaire (voir figure V.1.).

Partant de la racine, nous obtenons un premier critère, le **type de multiplexage des messages**. Il s'agit d'une technique assurant l'emploi d'une voie commune (physique) pour réaliser plusieurs voies de transmission (logiques). Une première technique, dite du **multiplexage dans l'espace**, divise la bande de fréquences transmises par cette voie commune en bandes moins larges, dont chacune sert à constituer une voie de transmission directe. Nous nous attarderons peu sur ces protocoles car ils éliminent purement et simplement le problème du partage d'une voie de transmission unique entre plusieurs stations en allouant, pour toute la durée d'un échange ou de façon permanente, à tout couple émetteur-récepteur, une voie propre. Une première feuille est ainsi définie et illustrée par les lignes dédicacées ou encore par le protocole "*FDMA*" (voir point V.4.A). La deuxième technique, moins onéreuse, assure un **multiplexage dans le temps** : la voie commune est utilisée par roulement dans le temps pour constituer différentes voies de transmission intermittentes.

Cette seconde méthode de multiplexage nous conduit à un nouveau noeud-critère, à savoir le mode de division du temps. Une division

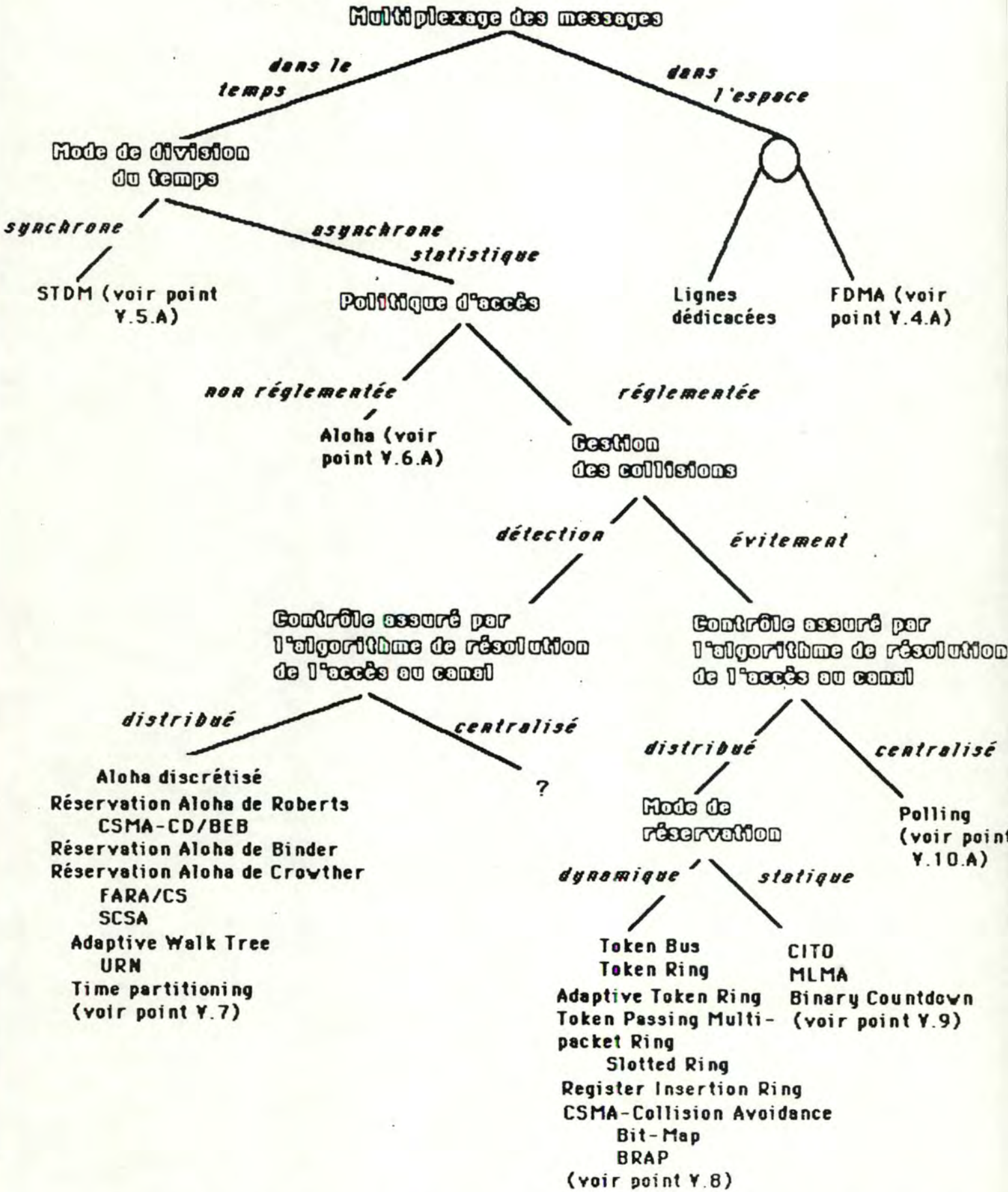


Figure V.1. : Classification des protocoles d'accès sous forme d'arbre binaire

élémentaire consiste à assigner des tranches de temps sur une base fixée, prédéterminée : on parle alors de **division synchrone** du temps, et nous aboutissons à une deuxième feuille (voir point V.5.A, le protocole "*STDM*"). Une deuxième méthode opposée dite **statistique ou asynchrone** alloue aux stations des tranches de temps, de façon dynamique, suivant leurs besoins.

Ce dernier mode de division du temps exige une politique d'accès appropriée, constituant un nouveau critère dans notre arbre. En l'absence de politique d'accès, les stations peuvent émettre aléatoirement et sans contrainte : la politique est dite **non réglementée** et nous obtenons une troisième feuille reposant sur le protocole "*Aloha*" (voir point V.6.A). Inversément, la présence d'une politique impliquera un algorithme gouvernant la séquence et le moment de l'accès des stations au canal : la politique d'accès est alors **réglementée**.

Dans le cadre des algorithmes réglementant l'accès au canal, le type de gestion des collisions correspond à un nouveau noeud de notre arbre. Une collision se produit quand plusieurs stations émettent simultanément un message, et que ces derniers se recouvrent. Les collisions sont soit **détectées** (gestion à posteriori), soit **évitées** (à priori).

Quel que soit le type de gestion des collisions, le contrôle assuré par l'algorithme de résolution de l'accès au canal est soit **centralisé** dans une station de contrôle, soit **réparti ou distribué** le long des stations du LAN. Dans le cas où les algorithmes de détection des collisions sont distribués, nous débouchons sur une quatrième feuille intégrant un nombre appréciable de protocoles (voir point V.7), par opposition à l'absence (à notre connaissance) de représentants pour ces mêmes algorithmes centralisés.

Les algorithmes distribués évitant les collisions sont encore subdivisés selon le mode de réservation du canal adopté dans l'algorithme. Ce mode de réservation caractérise l'ordre d'accès dans lequel les stations se voient allouer les droits de transmission. Lorsque cet ordre est **dynamique**, cela signifie qu'il est déterminé par les stations elles-mêmes, et nous rencontrons une cinquième feuille de notre arbre (voir point V.8) regroupant également de nombreux protocoles. Si par contre l'ordre est **statique**, il est alors déterminé à priori, ce qui nous conduit à une sixième feuille de l'arbre (voir point V.9).

La septième et dernière feuille regroupe les protocoles dans lesquels les

algorithmes évitent les collisions et sont centralisés (voir point V.10).

V.2. Classification courante des protocoles d'accès

Parallèlement à notre classification sous forme d'arbre binaire, il est courant dans la littérature de regrouper les protocoles d'accès selon deux grandes classes, que nous évoquerons rapidement (voir figure V.2.).

La première classe assure le partage de la voie de manière **statique** ("fixed assignment"). Dans ce cas, chaque émetteur connaît une fois pour toutes, sans avoir recours à des requêtes de contrôle, la manière selon laquelle il peut utiliser la voie sans provoquer de collision. Elle englobe le cas du partage d'une voie en bandes de fréquences (protocole "*FDMA*", voir point V.4.A) et le cas du partage de la voie sur base du temps synchrone (protocole "*STDM*", voir point V.5.A).

La seconde classe partage la voie de manière **dynamique** :

- soit par **compétition** ("contention, random-assignment ou random-access method") ;
- soit par **accès contrôlé** ("demand assignment method").

Lorsque la voie est partagée par compétition, tout émetteur l'utilise dès qu'il est prêt à émettre, sans demande préalable, donc en ignorant les autres, ce qui peut conduire à l'émission simultanée de plusieurs messages provoquant des collisions.

Une première série de méthodes est utilisée sur les voies radioélectriques terrestres ou avec les satellites (Aloha, voir point V.6.A et ses améliorations, voir point V.7.A, V.7.B, V.7.D, V.7.E et V.T.F).

Lorsque le délai de propagation est plus court que le temps de transmission d'un message, une station peut disposer de plus d'informations sur l'état du réseau : elle peut ainsi écouter le canal avant d'émettre, déterminer si le canal est occupé ou pas et ainsi réduire le risque de

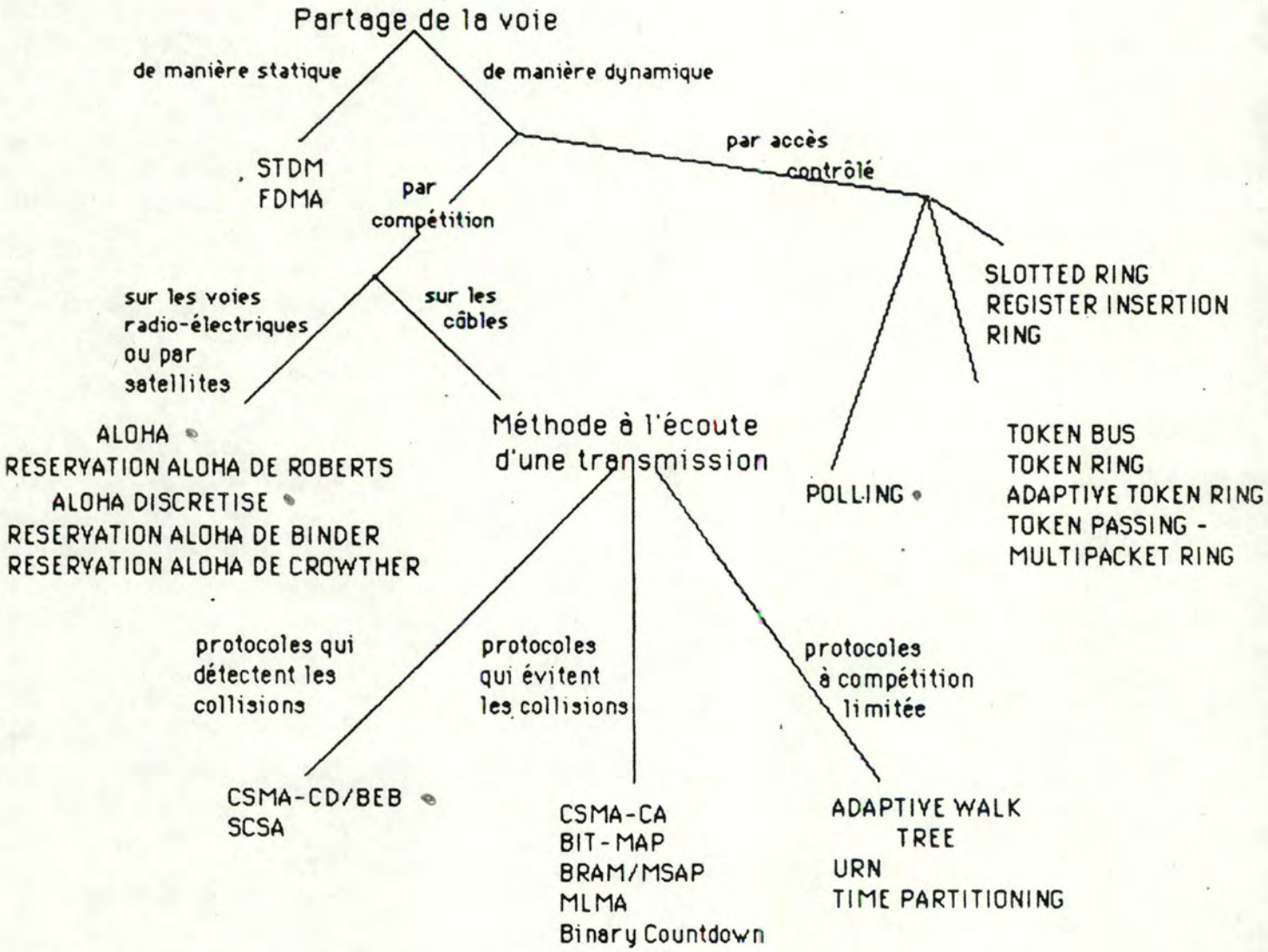


Figure V.2. : Classification courante

collisions (et par conséquent, améliorer les performances) : il s'agit des méthodes à l'écoute d'une transmission ("carrier sense method"). Dans cette catégorie, on peut encore discerner :

- les protocoles qui détectent les collisions (CSMA-CD-BEB, voir point V.7.C ; SCSA, voir point V.7.G) ;
- les protocoles qui évitent les collisions ("CSMA-CA", voir point V.8.G ; "Bit-Map", voir point V.8.H ; BRAM/MSAP, voir point V.8.I ; MLMA, voir point V.9.B ; "Binary Countdown", voir point V.9.C ; CITO, voir point V.9.A) ;
- les protocoles à compétition limitée ("Adaptative Walk Tree", voir point V.7.I ; URN, voir point V.7.J ; "Time partitioning", voir point. V.7.K) ; l'idée est de subdiviser dynamiquement les stations en groupe avec beaucoup de stations par groupe quand le débit réel est faible et peu (une seule station) par groupe pour éviter les collisions quand le débit réel est élevé.

Enfin, dans le cas où la voie est partagée par accès contrôlé, les stations sont coordonnées de manière à ce que deux ou plusieurs n'essaient jamais de transmettre simultanément ; cette coordination est obtenue en imposant un ordre dans l'allocation des droits d'accès au canal. Dans cette catégorie, on peut reconnaître :

- la classe où toutes les stations sont interrogées par une station centrale dans un ordre fixe, jusqu'à la rencontre d'une station souhaitant émettre ("polling method", voir point V.10.A) ;
- la classe où le canal est alloué dynamiquement par la possession d'un jeton : ce jeton, symbole d'autorité indique à la station qui le possède qu'elle a le contrôle du moyen de communication ("token passing", voir points V.8.A, V.8.B, V.8.C et V.8.D) ;
- ... ("slotted ring", voir point V.8.E ; "register insertion", voir point V.8.F).

Quant à nous, nous préférons recourir à notre classification selon un arbre binaire car elle met en évidence les critères à considérer quand on aborde l'application de protocoles d'accès dans des systèmes temps réel. En effet, les critères tels que gestion des collisions, algorithme de résolution de l'accès au canal,... ont un impact direct sur le temps d'accès et le temps de réponse rencontrés sur un réseau local LAN par l'emploi de tel ou tel protocole d'accès.

De plus, notre classification amène tout naturellement à une répartition ultérieure des protocoles rencontrés dans les feuilles de l'arbre selon le critère d'algorithme de résolution déterministe ou probabiliste de l'accès au canal, le but de notre travail...

V.3. Conventions adoptées pour la modélisation analytique des protocoles

Avant tout, dans un souci de simplification et d'uniformisation de la présentation des modèles et des formules qui en découlent, nous introduisons les **hypothèses de travail** suivantes :

- un message sera supposé comme impliquant un seul paquet à transmettre sur le canal ; s'il devait en être autrement, nous le ferions remarquer explicitement ;
- le processus d'arrivée des paquets dans les files auxiliaires suit une distribution de Poisson.

Lors de notre étude des différents protocoles dans les points suivants, nous aurons généralement recours à des modèles analytiques étudiant le temps d'accès au canal sous forte charge. Malgré la grande diversité des protocoles abordés, nous essaierons d'assurer une cohérence générale dans les notations que nous adopterons. Et notamment, nous tiendrons compte de notre modèle stochastique de référence introduit dans le chapitre précédent (voir point IV.2, figure IV.1.).

λ Taux moyen global d'arrivée ou nombre moyen de paquets arrivés par unité de temps dans les différentes files d'attente auxiliaires. Unité : paquet/seconde.

Tc Tranche-canal, c'est-à-dire temps de service constant pour des paquets de longueur standard. Unité : seconde.

Ts Temps de service ou de transmission maximal d'un paquet, c'est-à-dire le rapport entre la longueur maximale d'un paquet

supportée par le protocole considéré et la vitesse de transmission du moyen de communication. Unité : seconde.

Ta Temps d'accès maximal au canal, sous fort débit réel (voir point IV.3). Unité : seconde.

Tw Temps constant nécessaire pour un bit à un tour d'un anneau vide ("*walk time*") incluant le retard d'un bit introduit par station, le délai de propagation du signal ainsi que dans le cas des protocoles "Token Passing", le retard introduit pour traitement du jeton. Unité : seconde.

v_t Vitesse de transmission sur un canal de communication. Unité : bits par seconde.

S Nombre moyen de nouveaux paquets émis sur le réseau par tranche-canal dans la famille des protocoles Aloha. Unité : paquet/Tranche-canal.

E (x) Moyenne de la variable aléatoire x.

N Nombre de stations connectées au réseau local.

K Borne supérieure, dans le cadre des protocoles de la famille Aloha, dans la distribution aléatoire uniforme du nombre de tranches-canal à attendre par une station avant de pouvoir retransmettre son paquet.

θ Intervalle de propagation, temps de propagation de bout en bout d'un signal sur un réseau de topologie en bus ou en arbre. Unité : seconde.

$2 * \theta$ Intervalle de compétition, temps maximal nécessaire à une station, après émission de son paquet, pour qu'un éventuel signal de collision en provenance de la station qui en est la plus éloignée lui parvienne ; cet intervalle correspond à la période de vulnérabilité d'un paquet dans les réseaux à topologie en bus et en bande de base (voir figure V.8. présentée pour le CSMA-CD/BEB). Unité : seconde.

Dans de nombreux protocoles intervient la notion de trame que nous définirons une fois pour toutes comme étant une suite finie de paquets de bits de même taille issus des différentes stations.

Il est important de préciser que dans le cas où le temps d'accès au canal ne présenterait pas de borne supérieure, nous effectuerons une étude intuitive de ce dernier en en donnant un modèle de sa moyenne et en montrant intuitivement qu'il n'est pas déterministe.

V.4. Etude de la feuille numéro 1 de l'arbre binaire : multiplexage de la voie de communication dans l'espace

Cette feuille regroupe les protocoles recourant à la technique du multiplexage de la voie de communication dans l'espace.

V.4.A. Le protocole déterministe "Frequency Division Multiple Access"

V.4.A.1. Présentation générale

Dans le protocole "*Frequency-Division Multiple Access*" (FDMA), la largeur de bande est divisée en un nombre N de portions égales, chacune allouée à une station, avec une marge de sécurité entre elles. Etant donné que chaque station dispose de sa bande de fréquence privée, il n'y a pas d'interférence entre stations et si elle a un message présent en tête de sa file d'attente auxiliaire, ce dernier sera émis sans retard et donc de façon déterministe.

V.4.A.2. Etude analytique du temps d'accès

Comme les paquets présents dans la file centrale sont émis sans retard par chaque station dans la largeur de bande qui leur est allouée, le temps d'accès au canal de ces messages est donc nul.

$$Ta = 0 \qquad [V.1]$$

V.4.A.3. Critique du protocole.

Si le nombre de stations est important et varie continuellement et si le spectre reste subdivisé en un nombre fixe N de régions, alors des problèmes de gaspillage se posent rapidement, que ce soit lorsque moins de N stations sont connectées, ou quand au contraire plus de N stations veulent travailler simultanément. Dans ce dernier cas, la connexion est refusée à certaines stations (protocole non déterministe dans ce cas) pour manque de largeur de bande même si certaines stations connectées transmettent ou reçoivent très peu.

V.4.A.4. Illustration de ce protocole

Nous pouvons citer en guise d'illustration WANGNET (31).

V.5. Etude de la feuille numéro 2 de l'arbre binaire : multiplexage de la voie de communication dans le temps, selon un mode de division synchrone du temps

Cette feuille regroupe les protocoles recourant à la technique du multiplexage de la voie de communication dans le temps, selon un mode de division synchrone du temps.

V.5.A. Le protocole déterministe "Synchronous Time Division Multiplexing"

V.5.A.1. Présentation générale

Dans le protocole "*Synchronous Time Division Multiplexing*" (STDM), le temps est divisé en intervalles qui correspondent à la durée de transmission de trames de longueur fixe, chaque intervalle étant lui-même subdivisé en tranche-canal (voir figure V.3.). Dans la version la plus simple, le nombre de tranches-canal par intervalle équivaut au nombre de stations connectées au réseau. Chacune des N stations se voit alors allouer de manière statique une des N tranches-canal durant laquelle elle peut utiliser l'entière capacité de transmission du canal de communication, même si elle n'a rien à émettre...

V.5.A.2. Etude analytique du temps d'accès

La situation la plus défavorable se produit lorsqu'une station, après le

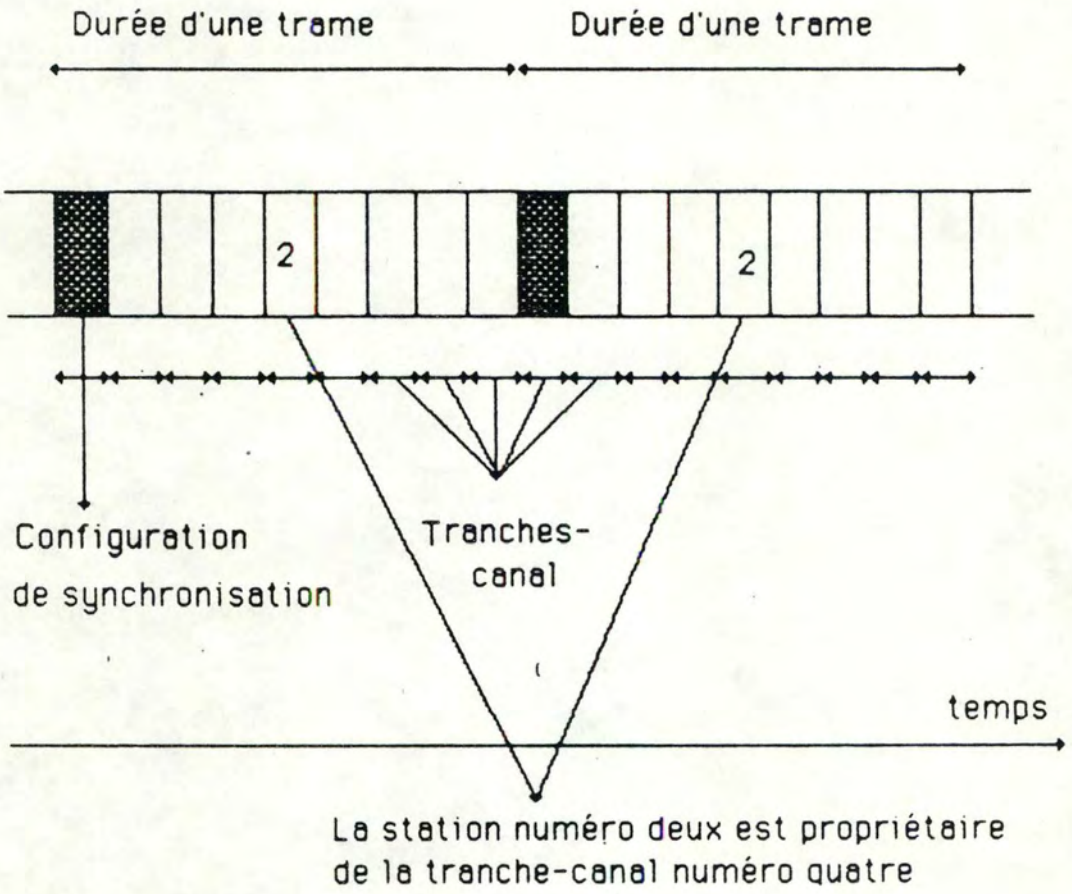


Figure V.3 : Protocole "SYNCHRONOUS TIME DIVISION MULTIPLEXING"

passage de sa tranche-canal, a un paquet présent dans la file centrale. A ce moment, elle doit attendre l'écoulement des N-1 autres tranches destinées aux autres stations. Nous obtenons la relation :

$$Ta = (N-1) * Tc \qquad [V.2]$$

V.5.A.3. Critique du protocole

Bien que facile à mettre en oeuvre, ce protocole est inefficace quand il y a beaucoup de stations et que les paquets arrivent par rafales, car les droits de transmission sont accordés à toutes les stations qu'elles aient ou non des paquets à transmettre.

Si le nombre de stations dépasse le nombre de tranches-canal d'un intervalle, on peut imaginer d'allouer temporairement une tranche-canal non utilisée à une des stations supplémentaires. Pour celles-ci, le temps d'accès ne peut plus être calculé de manière déterministe.

Si le nombre de stations reste constant, tout paquet en provenance d'une station quelconque peut être émis de façon déterministe. Cependant, si une station n'a rien à émettre sa tranche-canal pourrait être utilisée par d'autres stations déjà connectées qui ont un énorme besoin de transmission (voir, dans ce cas, l'implémentation de l'IBM 2790 loop par Steward et Hippert présentée dans Penney et Bagdadi, 22).

V.6. Etude de la feuille numéro 3 de l'arbre binaire : multiplexage de la voie de communication dans le temps, selon un mode de division asynchrone du temps et une politique d'accès non réglementée au canal

Cette feuille regroupe les protocoles recourant à la technique du multiplexage de la voie de communication dans le temps, selon un mode de

division asynchrone du temps et dont la politique d'accès au canal est non réglementée.

V.6.A. Le protocole probabiliste "Aloha"

V.6.A.1. Présentation générale

Le travail d'Abramson, en 1970, aboutit au protocole Aloha qui est le premier protocole distribué ayant permis à un nombre de stations terriennes réparties géographiquement de communiquer sur un simple canal de communication radio-électrique. Le réseau en étoile sur lequel ce protocole a été essayé permettait aux stations-terminal présentes dans les îles HAWAII de communiquer entre elles via un ordinateur centralisé. Certains auteurs (Schwartz, 24 et Tanenbaum, 29) ont appliqué le même raisonnement que celui adopté par Abramson à la transmission par satellite, bien que des différences soient à remarquer dans le temps de propagation. Dorénavant, nous nous placerons également dans le cas des satellites.

Comme dans tout protocole de la classe d'accès par compétition (voir point V.2), les stations transmettent chaque fois qu'elles disposent de données. Cependant, on peut ajouter que dans le genre d'application où est employé ce protocole, les paquets arrivent le plus souvent par rafales et sont très courts, entraînant un risque relativement faible pour plusieurs stations de transmettre simultanément. Si deux stations empruntent malgré tout le canal en même temps, leurs paquets seront détruits. Pour s'en rendre compte, les émetteurs réécoutent leur propre paquet après un certain retard dû à la propagation (270 msec ou "*one round-trip time*"). En effet, une station transmet vers le satellite dans une certaine bande de fréquences différente de celle sur laquelle le paquet est retransmis par le satellite. Cette retransmission s'effectue en mode diffusion c'est-à-dire que chaque paquet est retransmis simultanément à la totalité des stations (le récepteur comme l'émetteur). Si son paquet a subi une collision, l'émetteur attend alors une durée aléatoire avant de réémettre car autrement, les paquets entreraient

continuellement en collision. Ce protocole est simple mais cette simplicité provoque, comme nous le verrons, des dégradations de performances.

V.6.A.2. Etude intuitive du temps d'accès

Dans un souci de simplicité, nous prendrons comme unité de temps de référence, la tranche-canal Tc. Nous optons pour des paquets d'égale longueur car Abramson a montré que le débit réel des systèmes aloha est maximisé lorsque les paquets sont de dimension uniforme plutôt que de taille variable.

Nous ferons d'abord l'hypothèse de stationnarité c'est-à-dire que le flux moyen des entrées aux stations correspond au flux moyen des sorties ou encore que la charge égale le débit réel.

En plus de l'émission de nouveaux paquets, les stations effectuent des retransmissions de paquets qui ont précédemment subi des collisions. Supposons alors que la distribution, par tranche-canal, du nombre de paquets émis et réémis par suite de collision suit une loi de Poisson de taux G (des études montrent que dans le cas où la station attend une durée aléatoire avant de réémettre, l'hypothèse de Poisson ne devient valable que si l'intervalle d'attente avant réémission est important). G désigne donc la somme du nombre moyen de nouveaux paquets émis (ou reçus dans l'hypothèse de stationnarité) par les stations par tranche-canal et du nombre moyen de retransmissions durant cette même période. Dans ces conditions, si E désigne le nombre moyen d'essais de retransmission par paquet émis, G peut s'exprimer par la relation suivante :

$$G = S + (E * S) \text{ paquets/tc} \qquad [V.3]$$
$$= (\lambda * Tc) + (E * \lambda * Tc) \text{ paquets/tc}$$

Un paquet ne subira aucune collision si aucun autre paquet n'est également transmis durant son émission sur le canal. Sur base de la figure V.4., nous voyons que le paquet A entre en collision soit avec le paquet B ou soit avec le paquet C, si les premiers bits du paquet A se superposent aux derniers bits du paquet précédemment transmis B ou si les derniers bits du

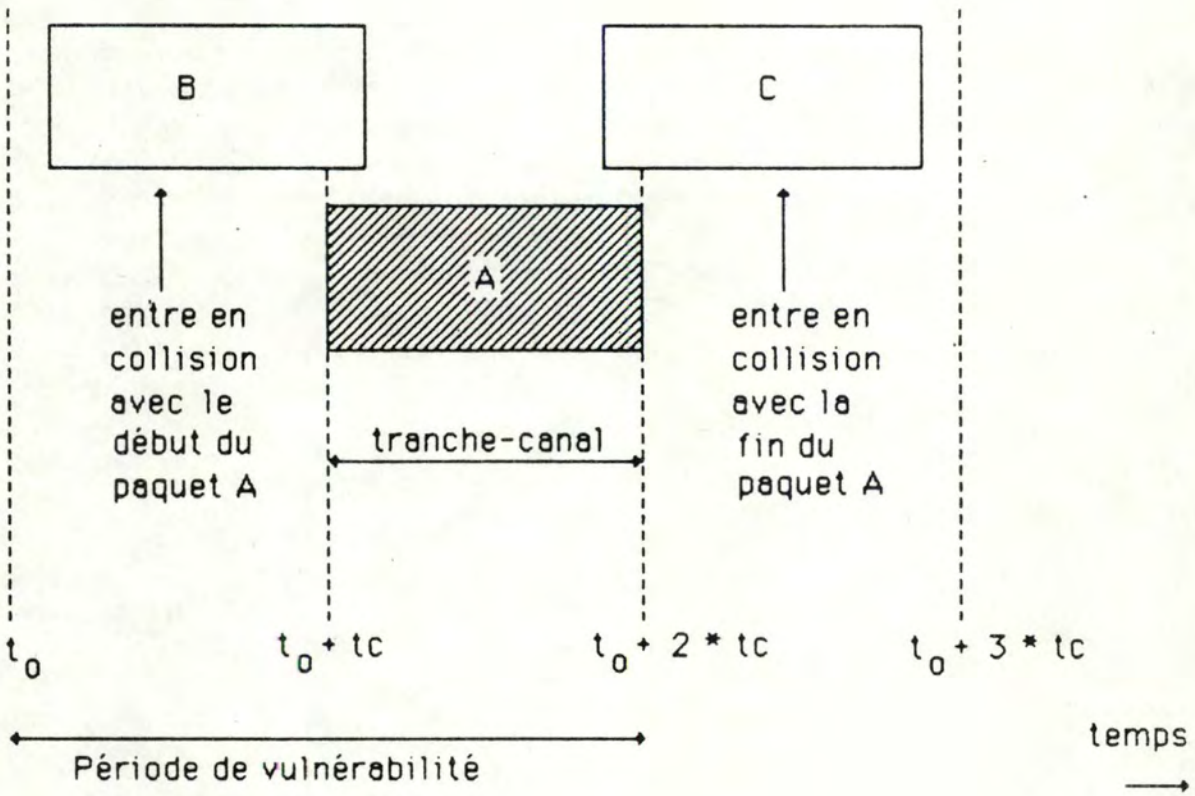


Figure V.4 : Période de vulnérabilité du paquet hachuré

paquet A sont recouverts par les premiers bits du paquet suivant C. La période durant laquelle un paquet est susceptible de subir une collision correspond donc à deux tranches-canal et porte le nom de période de vulnérabilité.

La probabilité que k paquets soient transmis pendant cet intervalle de vulnérabilité est donnée par la distribution de Poisson :

$$PR[k] = (2 * G)^K * e^{-2*G} / k! \text{ et donc,} \qquad [V.4]$$
$$PR[0] = e^{-2*G}$$

Le nombre moyen S de nouveaux paquets émis (ou le taux d'arrivée des paquets aux stations dans l'hypothèse de stationnarité) pendant une tranche-canal équivaut au produit du nombre moyen d'essais de transmission (de nouveaux paquets comme de paquets ayant subi des collisions) durant cette période par la probabilité qu'un paquet ne souffre pas de collision durant sa transmission. Nous trouvons donc l'égalité :

$$S = \lambda * Tc = G * PR[0] \qquad [V.5]$$
$$= G * e^{-2*G} \text{ , par [V.4]} \qquad [V.6]$$

Cette relation atteint sa valeur maximale pour G = 1/2 où S ou (λ * Tc) vaut 1/(2*e) paquet/Tc, ce qui correspond à un débit réel maximal possible s'élevant à 18% de la capacité du canal (voir figure V.5.).

D'autre part, nous pouvons exprimer la relation [V.3] en fonction du nombre moyen de retransmissions par paquet, c'est-à-dire E :

$$E = G / S - 1$$
$$= G / (\lambda * Tc) - 1$$
$$= e^{2*G} - 1 \text{ , par [V.6]} \qquad [V.7]$$

Sur base de la figure V.5. et de la formule [V.3], nous pouvons déjà comprendre que si S ou (λ * Tc) augmente pour atteindre la valeur de 0,18 quand G vaut 1/2, l'autre composant (E * S) ou (E * λ * Tc) de G augmente encore plus vite pour atteindre la valeur de 0,32. Cependant, comme le nombre de retransmissions est limité supérieurement, le temps d'accès au canal d'un message ne subit pas de retard exagéré.

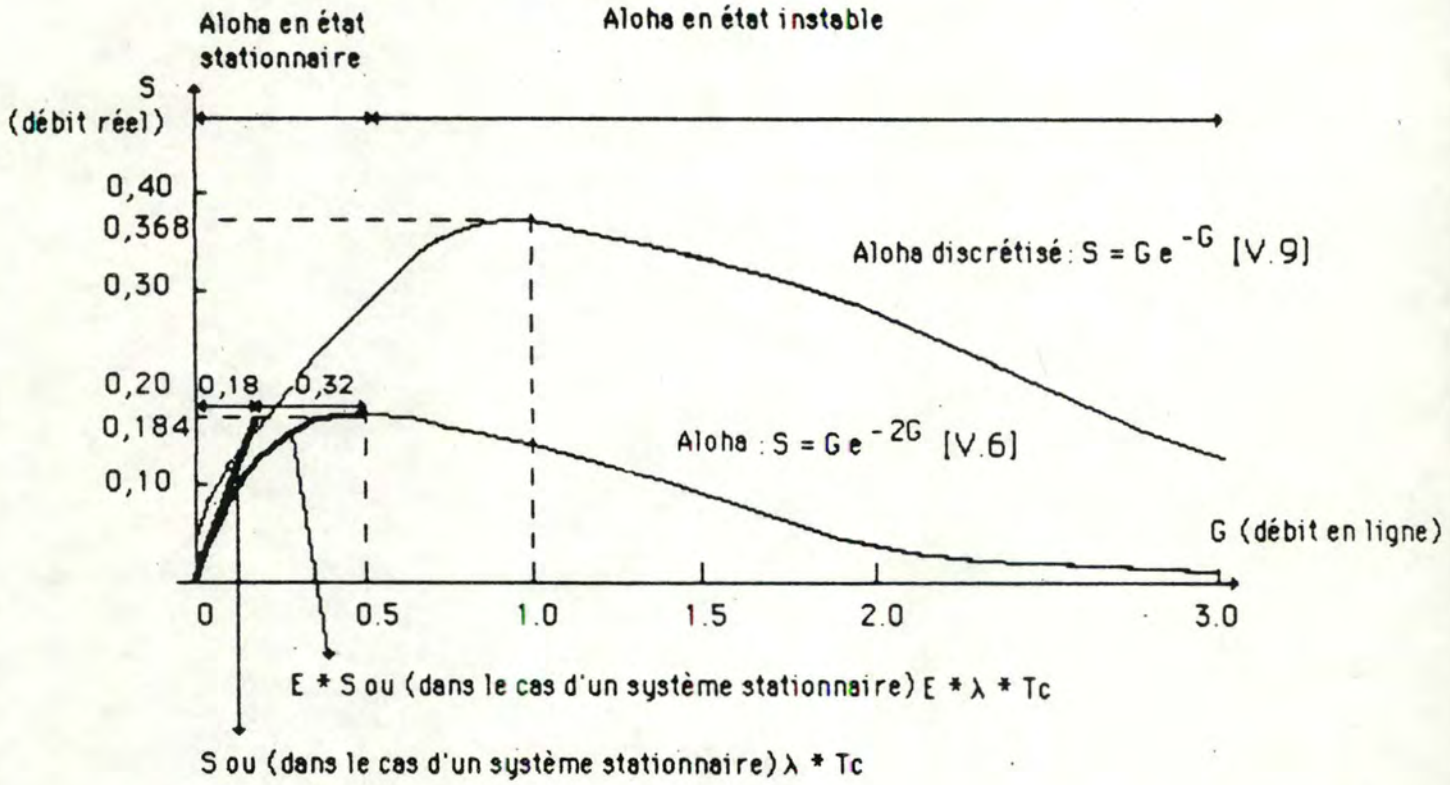


Figure V.5. : Expression du débit réel en fonction du débit en ligne

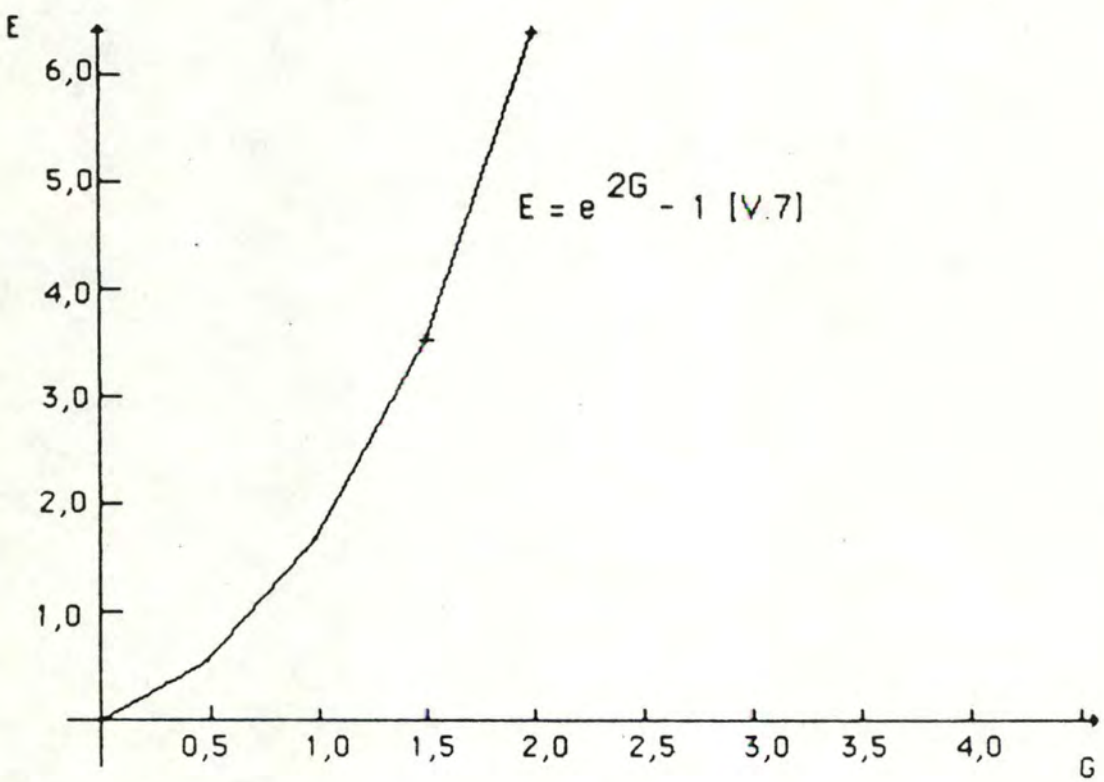


Figure V.6. : Expression du nombre moyen de retransmissions par paquet généré

Si la charge continue à croître ($\lambda * Tc > 0,18$ paquet/Tc), on assiste à une chute du débit réel (voir figure V.5.) mais à une explosion du débit en ligne dû au composant (E*S) de la formule [V.3]. On quitte alors l'hypothèse de stationnarité : la croissance de la charge est à l'origine d'un effet en chaîne, expliquant par là la formule [V.7]. En effet, la croissance de charge a augmenté les risques de collisions et les risques de retransmission E, ces retransmissions ont à leur tour provoqué de nouvelles collisions et un effet en chaîne s'est produit ; ceci explique la forme de la courbe représentant le nombre moyen de retransmissions par nouveau paquet émis (voir figure V.6) ainsi que le caractère non déterministe du temps d'accès.

Nous pouvons maintenant définir le temps d'accès moyen. Dans ce protocole, il correspond au produit du nombre moyen de retransmissions nécessaires avant l'envoi sans incident d'un paquet et de la durée moyenne d'une retransmission. De façon plus précise, nous obtenons la relation suivante :

$$E(Ta) = E * ((R + (1 + K)/2) * Tc) \qquad [V.8]$$

En effet, chaque retransmission implique un délai de propagation de R tranches-canal après un temps aléatoire d'attente qui suit une distribution uniforme entre 1 et K tranches-canal.

V.6.A.3. Variations sur un même thème

Des recherches ultérieures sur le protocole Aloha ont montré la nécessité d'une politique d'accès au canal qui soit réglementée, que ce soit dans le but d'améliorer le débit réel maximum autorisé (voir points V.7.A et V.7.B) ou encore d'obtenir un protocole déterministe (voir points V.7.D, V.7.E et V.7.F).

V.7. Etude de la feuille numéro 4 de l'arbre binaire : multiplexage de la voie de communication dans le temps, selon un mode de division asynchrone du temps et selon un algorithme de résolution d'accès au canal distribué et détectant les collisions.

Cette feuille regroupe les protocoles recourant à la technique du multiplexage de la voie de communication dans le temps, selon un mode de division asynchrone du temps et dont la politique d'accès au canal est réglementée : l'algorithme d'accès est distribué et détecte les collisions.

Nous aborderons d'abord les protocoles à résolution probabiliste des collisions (l'"Aloha discrétisé" (V.7.A), le "Reservation Aloha" de Roberts (V.7.B) et le "CSMA-CD/BEB" (V.7.C)) avant de passer à ceux à résolution déterministe. Dans ce dernier groupe, nous distinguerons trois ensembles homogènes de protocoles.

Nous isolerons d'abord les protocoles déterministes dérivés d'Aloha (le "Reservation Aloha" de Binder (V.7.D), le "Reservation Aloha" de Crowther (V.7.E), le "FARA/CS" (V.7.F)).

Vient ensuite la présentation d'un protocole déterministe à l'écoute du canal (voir point V.2) pour y détecter les collisions (le "SCSA" (V.7.G)).

Enfin, nous évoquerons les protocoles déterministes à l'écoute du canal et à collisions limitées (voir point V.2) (l'"Adaptive Walk Tree" (V.7.I), l'"URN" (V.7.J) et le "Time Partitioning" (V.7.K)).

V.7.A. Le protocole probabiliste "Aloha discrétisé"

Nous présentons donc d'abord les protocoles à résolution probabiliste des collisions.

V.7.A.1. Présentation générale

En 1972, pour doubler la capacité du système Aloha (voir point V.6.A), Roberts proposa de subdiviser le temps en intervalles égaux à une tranche-canal et de ne permettre l'émission que pendant ces tranches. Cependant, comme le satellite oscille autour de sa position orbitale, la tranche-canal est légèrement plus importante que le temps de transmission d'un paquet pour éviter des débordements du paquet dû à la variation du temps de propagation. Des collisions ne se produiront alors que si plusieurs paquets arrivent au satellite durant la même tranche-canal.

L'Aloha discrétisé présente une amélioration importante du débit réel sur la technique pure Aloha mais il est plus complexe car toutes les stations doivent être synchronisées sur une horloge commune localisée dans le satellite.

V.7.A.2. Etude intuitive du temps d'accès

Comme il ne peut y avoir de superposition de paquets sur des intervalles adjacents, la période de vulnérabilité d'un paquet se réduit à une tranche-canal. La relation [V.3] reste valable, et les relations [V.6] et [V.7] s'expriment alors :

$S = G * e^{-G}$, par [V.6] [V.9]

$E = e^{G} - 1$, par [V.7] [V.10]

La relation [V.9] atteint alors sa valeur maximale pour G = 1 où S vaut (1/e) paquet/Tc (voir figure V.5.), ce qui correspond à un débit réel maximal possible s'élevant à 36,8% de la capacité du canal, le double du système Aloha. Si (λ * Tc) dépasse la valeur de 1/e, le système devient instable comme dans le système "Aloha" et les relations [V.3] et [V.10] nous indiquent que le temps d'accès ne présente toujours pas de borne supérieure (voir discussion des formules [V.3] et [V.7] au point V.6.A.2).

Nous devons encore nous contenter d'une formulation moyenne du temps d'accès. Si nous suivons l'approche proposée par Lam et Kleinrock (Lam et Kleinrock, 15), nous pouvons reprendre la formule **[V.8]** en l'adaptant légèrement pour tenir compte de la découpe discrète du temps :

$$E(Ta) = E * (R + 1/2 + ((K + 1)/2) * Tc) \qquad [V.11]$$

La valeur 1/2 prend en compte le fait que si une collision se produit, le temps d'attente avant de réessayer d'émettre est un nombre entier de tranches-canal à sauter et que la retransmission doit commencer au début d'une tranche-canal.

Lam et Kleinrock conseillent de maintenir (λ * Tc) faible c'est-à-dire inférieur à 1/e pour obtenir un système stable, sinon il faut recourir à des mécanismes de contrôle qui cherchent à établir un compromis entre K et Ta. Si le nombre de retransmissions est faible, il faut définir un petite valeur de K pour se débarrasser au plus vite des retransmissions et obtenir un faible temps d'accès. Si, par contre, le nombre de retransmissions est élevé, pour éviter de nouvelles collisions, il faut définir un grande valeur de K aux dépens d'une dégradation du temps d'accès. En effet, si deux stations entrent en collision et que chacune attend zéro ou une tranche-canal avec la même probabilité, la chance d'une nouvelle collision la seconde fois est de 1/2. Si, par contre, les retransmissions sont étalées sur les 100 prochaines tranches-canal, le risque pour les mêmes paquets d'entrer en collision est de 1 % mais le temps d'accès moyen sera forcément plus important.

V.7.A.3. Critique de ce protocole

Ce protocole augmente le débit réel maximum possible du canal par rapport à celui de l'Aloha mais au prix d'un temps d'accès moyen plus élevé (voir figure V.5).

V.7.B. Le protocole probabiliste "Reservation Aloha" de Roberts

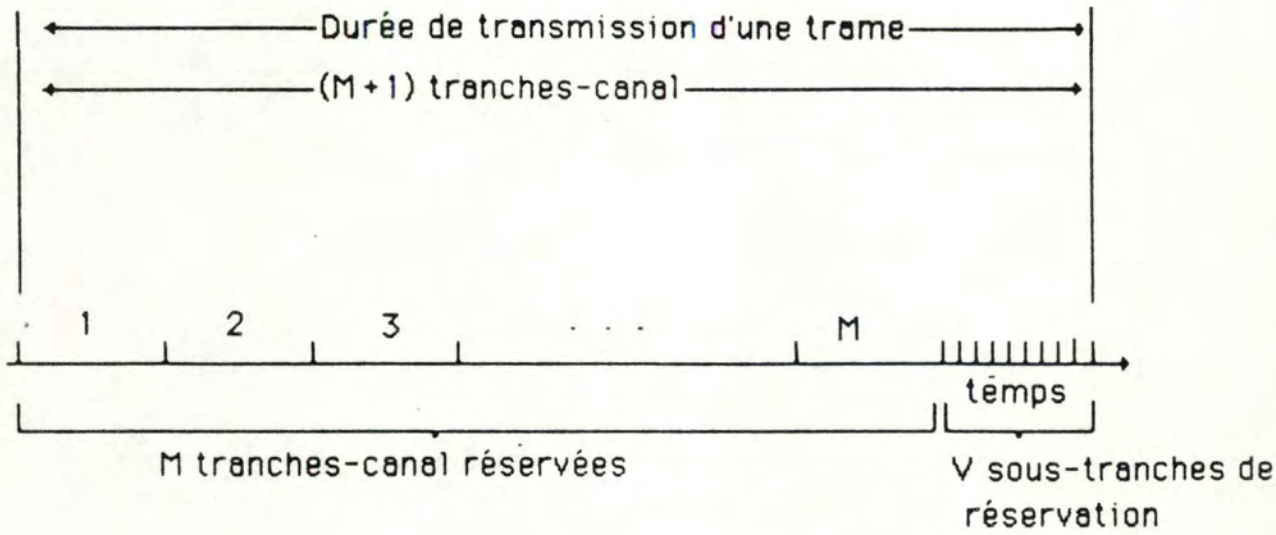
V.7.B.1. Présentation générale

Le protocole de réservation de Roberts présente une variante à l'Aloha discrétisé (voir point V.7.A) en ce sens que son fonctionnement se rapproche d'un système multiplexé dans le temps (voir point V.5.A) quand la charge augmente.

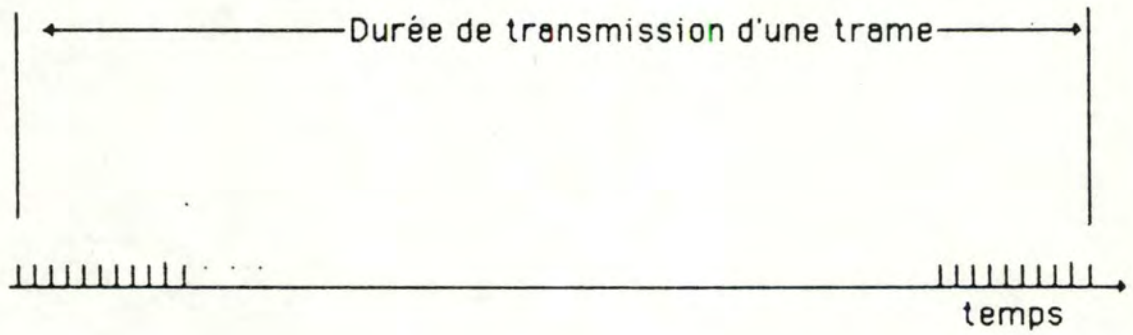
Il demande aux stations de faire connaître leurs intentions avant de transmettre dans une tranche-canal. Le temps est divisé par intervalles équivalant à la durée de transmission d'une trame et contenant chacun M+1 tranches-canal dont la dernière est subdivisée en V sous-tranches réservées à la transmission de courts paquets de réservation.

Quand une station veut envoyer des paquets, elle diffuse durant un des V sous-champs sa demande de tranche(s)-canal. Si la réservation réussit sur base du protocole mis en oeuvre dans l'Aloha discrétisé, des tranches-canal sont réservées. Comme toutes les stations sont à l'écoute en permanence, chacune retient la prochaine tranche-canal disponible pour que, quand une station effectue une réservation fructueuse, elles sachent le nombre de tranches à sauter avant de transmettre. Si le paquet de demande de réservation devait revenir pollué, c'est qu'il y a eu une collision et aucune réservation n'est effectuée. La station, après un certain temps aléatoire, doit retransmettre un autre paquet de demande pour éviter la possibilité d'une nouvelle collision.

Lorsque la longueur de la file centrale tombe à zéro, toutes les tranches-canal sont subdivisées en sous-tranches de réservation pour accélérer le processus. On dit alors que le système se trouve dans l'état Aloha sinon il se trouve dans l'état réservé (voir figure V.7.).



SYSTEME DANS L'ETAT RESERVE



SYSTEME DANS L'ETAT ALOHA : toutes les sous-tranches sont des sous-tranches de réservation

Figure V.7. : Schéma du "RESERVATION ALOHA" de ROBERTS

V.7.B.2. Etude intuitive du temps d'accès

Cette technique entraîne une amélioration du débit réel par rapport à l'Aloha discrétisé mais au prix d'un temps d'accès accru sous faible charge car toute transmission est précédée par deux délais de propagation (celui du paquet de requête et celui du paquet de données) non négligeables à mettre en balance avec un seul délai de propagation pour le protocole Aloha discrétisé.

Comme les réservations s'effectuent suivant la technique de l'Aloha discrétisé qui n'est pas déterministe lorsque (λ*Tc) dépasse 0,36 message/Tc, ce protocole ne l'est pas non plus. Nous présenterons dès lors le temps d'accès moyen qui tient compte du temps passé aux réservations ainsi que du temps à attendre que la bonne tranche-canal arrive.

$$E(Ta) = D_1 * (1 - S_1) + D_2 * S_1 + "M" \qquad [V.12]$$

avec D_1 :

le temps pour effectuer une réservation valable quand le système se trouve dans l'état Aloha ; comme la réservation suit la technique de l' Aloha discrétisé, D_1 s'obtient par la relation [V.11] :

$$D_1 = E_1 * (R + (0.5/V) + ((K + 1)/2V)) * Tc \qquad [V.13]$$

car chaque intervalle est subdivisé en V sous-intervalles ; E_1 représente le nombre moyen de retransmissions par paquet de réservation dans l'état Aloha.

avec $(1 - S_1)$:

la probabilité que le système soit dans l'état Aloha.

avec S_1 :

la probabilité que le système soit dans l'état réservé.

avec D_2 :

le temps pour effectuer une réservation valable quand le système se trouve dans l'état réservé ; comme la réservation suit la technique de l' Aloha discrétisé, D_2 s'obtient par la relation [V.11]:

$$D_2 = E_2 * (R + (M/2) + ((K + 1)/2V)) * Tc \qquad [V.14]$$

où il faut tenir compte du fait que les V sous-intervalles pour effectuer les demandes de tranches-canal se trouvent en fin d'intervalle seulement : cela ajoute donc un retard additionnel en moyenne de M/2 tranches-canal ; Roberts a prouvé la validité de la relation [V.14] dans le cas où $R > (M + 1)$ et $K \leq V$ (pour plus de détails, on consultera Schwartz, 24) ; pour mémoire, E_2 est le nombre moyen de retransmissions subies par un paquet de réservation généré dans l'état réservé.

et "M" :

le nombre de tranches-canal allouées aux stations ayant effectué une réservation fructueuse préalable à celle de la station envisagée.

V.7.B.3. Critique du protocole

Ce protocole améliore le débit réel maximum possible du canal par rapport à celui de l'Aloha mais au prix d'un temps d'accès moyen supérieur.

V.7.C. Le protocole probabiliste "Carrier-Sense Multiple-Access with Collision Detection"

V.7.C.1. Présentation générale

Défini par la norme 802.3 de l'IEEE, le protocole "*Carrier-Sense Multiple-Access with Collision-Detection*" (CSMA-CD) est le plus rencontré sur les topologies en bus et en arbre et convient pour le trafic en rafales qui rend le protocole "*STDM*" tout-à-fait inefficace.

Nous commencerons par rappeler les différentes stratégies de protocoles CSMA avant de définir le CSMA-CD et le CSMA-CD/BEB.

Le protocole **CSMA** permet l'accès concurrent au canal par plusieurs stations ("*Multiple Access*"). Chaque station désirant émettre écoute le moyen de communication pour y détecter la présence d'un message ("*Carrier Sense*"). Si le moyen est libre, la station peut transmettre. Sinon, la station attend une période de temps avant d'essayer à nouveau par application d'un des algorithmes expliqués ci-dessous.

L'algorithme non-persistent définit une première stratégie :

1. Si le canal est libre, la station émet.
2. Si le canal est occupé, la station doit attendre un certain temps tiré d'une distribution de probabilités avant de reprendre au point 1. Cette probabilité réduit le risque de collisions mais si plusieurs stations doivent retransmettre un paquet, du temps sera gaspillé après le premier essai.

Dans l'algorithme 1-persistent,

1. Si le canal est libre, la station émet. (Voir point 1, algorithme non-persistent).

2. Si le canal est occupé, la station doit l'écouter jusqu'à ce qu'il devienne libre et transmettre immédiatement, donc avec une probabilité de 1, ce qui lui vaut son nom de 1-persistent.

Dans l'algorithme p-persistent,

1. Si le canal est libre, la station doit transmettre avec la probabilité p et retarder d'un intervalle de compétition avec la probabilité 1-p avant de reprendre au point 1.
2. Si le canal est occupé, la station continue à écouter jusqu'à ce qu'il devienne libre avant de reprendre au point 1.

Ce dernier algorithme constitue un compromis qui essaie de réduire les collisions comme l'algorithme non-persistent et le temps gaspillé à attendre comme l'algorithme 1-persistent.

Dans le CSMA, quand deux paquets entrent en collision, le moyen de communication reste inutilisable pour toute la durée de la transmission des deux paquets endommagés. Le protocole CSMA-CD (Collision Detection) adopte une autre politique. Mais, avant toutes choses, attardons-nous un peu sur la notion de collision.

Une collision peut se produire dans deux cas :

- si plusieurs stations écoutent au même moment le canal et que, se rendant compte qu'il est libre, elles émettent ;
- si la présence d'un paquet émis sur le canal par une station A n'est pas encore révélée à une autre station B, dû au retard de propagation et que cette autre station B décide alors d'émettre.

S'il s'agit d'un système en bande de base et que les stations sont aux deux extrémités du bus, le temps nécessaire à la station A pour détecter la collision ou, encore appelé intervalle de compétition, équivaut à $2 * \theta$, avec θ le délai de propagation de bout en bout sur le bus (voir figure V.8.). Dans un système à large bande avec configuration en double cable c'est-à-dire où deux bus de données, l'un pour l'émission et l'autre pour la réception, sont reliés par un "Head End", si les deux stations sont celles qui sont les plus proches l'une de l'autre mais, en même temps, les plus éloignées du Head End, l'intervalle de compétition s'élève à $4 * \theta$ avec θ le délai de propagation de la

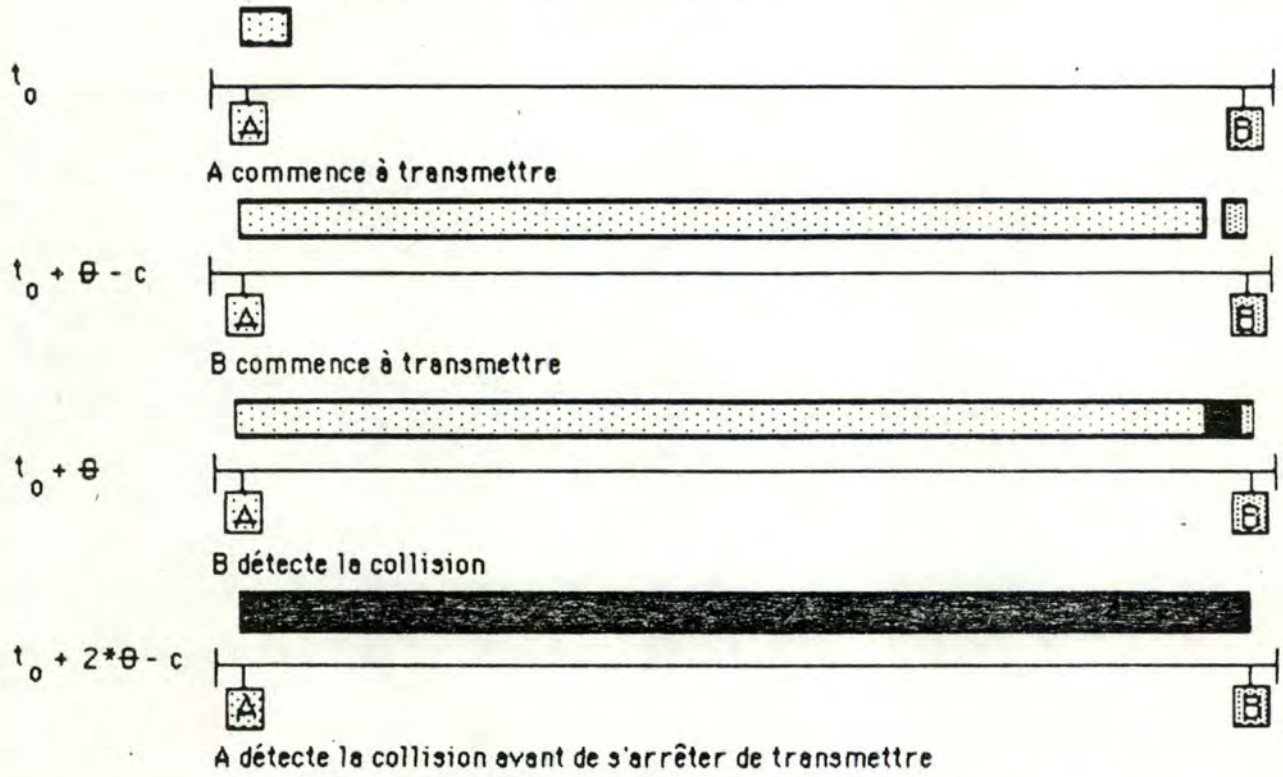


Figure V.8. : Période de détection d'une collision sur un bus en bande de base

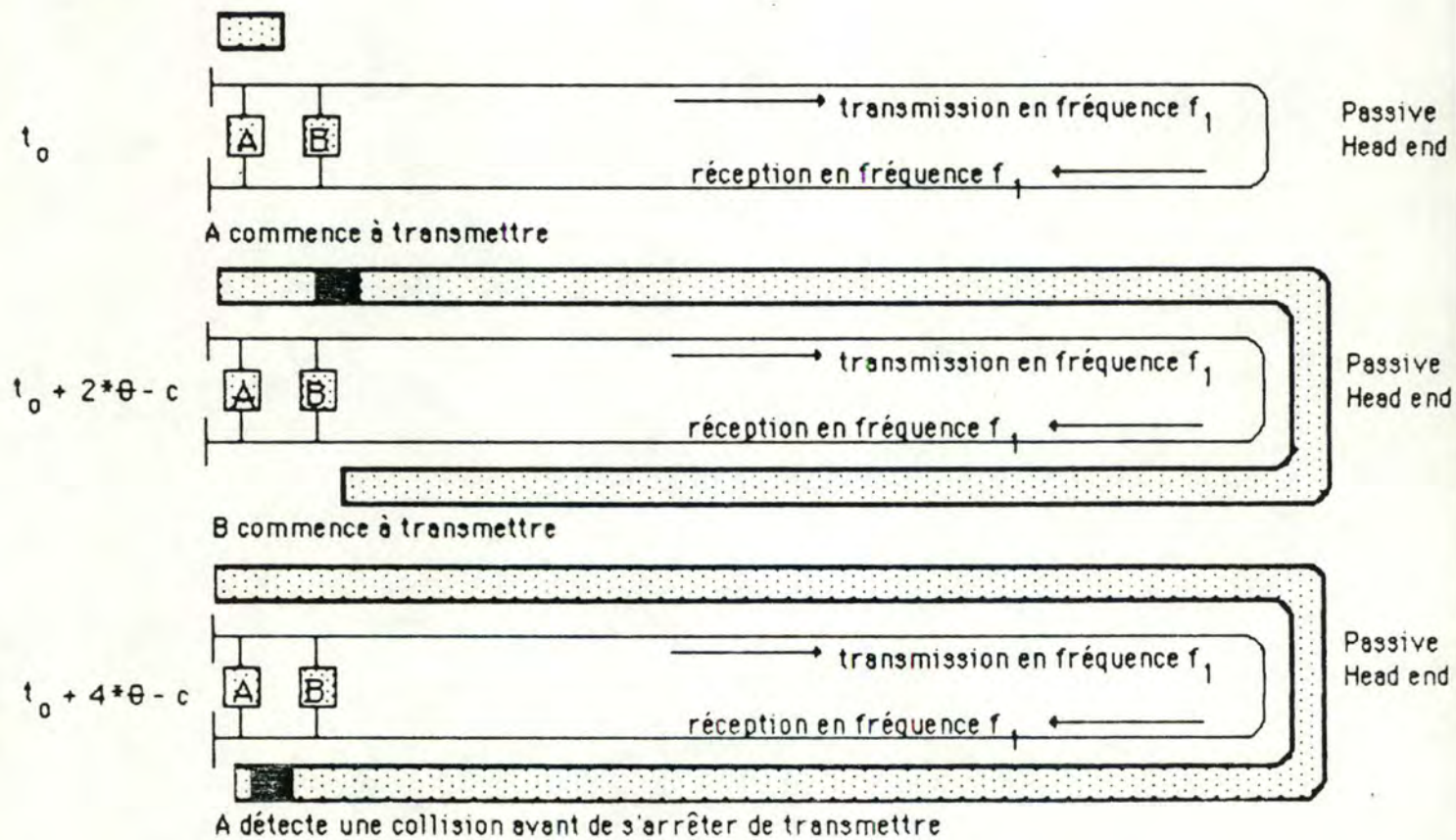


Figure V.9. : Période de détection d'une collision sur un bus à large bande

station la plus éloignée du Head End au Head End (voir figure V.9.).

A l'avenir, nous nous placerons toujours dans le cas d'une topologie en bus bande de base et prendrons 2 * θ pour valeur de l'intervalle de compétition.

Nous disions que le protocole CSMA-CD est plus efficace que le CSMA. En effet, dans ce protocole, une station, en cours d'émission, reste à l'écoute du moyen de communication pour y détecter une collision subie par son paquet. Si c'est le cas, elle s'arrête d'émettre immédiatement évitant par-là de gaspiller de la largeur de bande et transmet un signal de brouillage pour veiller à ce que toutes les stations soient au courant qu'une collision s'est produite. Elle attend alors un temps aléatoire avant d'essayer de retransmettre en recourant aux algorithmes du CSMA.

Comme le CSMA, le protocole CSMA-CD applique un des trois algorithmes mais le plus couramment employé est le 1-persistent (cas d'Ethernet et de Mitrenet). Cet algorithme garantit que sous faible charge, la station saisira le canal dès que possible. S'il introduit un plus grand risque de collisions, le temps perdu en collisions peut être réduit si les paquets sont longs relativement au délai 0 de propagation.

Les protocoles CSMA-CD étudiés par Tobagi et Hunt (Tobagi et Hunt, 30) présentent un temps de réponse moyen inférieur de moitié aux protocoles CSMA lorsque la charge est modérée ou forte. Ces derniers étudiés par Franta et Chlamtac (IEEE Project 802 Committee, 10) présentent également un temps de réponse moyen inférieur à celui de l'Aloha (voir point V.6.A).

V.7.C.2. Etude analytique du temps d'accès

Dans les protocoles CSMA-CD de type Ethernet, la gestion du temps aléatoire avant retransmission, suite à la détection de collisions, est assurée selon la stratégie du *"Binary Exponential Backoff"* (BEB) décrite par Metcalfe et Boggs (Metcalfe et Boggs, 19). Cette stratégie essaie de résoudre le problème de l'instabilité des protocoles CSMA-CD sous forte charge

(instabilité signifie que le débit réel est une fonction décroissante de la charge et que le temps d'accès n'admet pas de borne supérieure). Nous verrons que cette stratégie rend le CSMA-CD probabiliste.

Dans le protocole CSMA-CD/BEB, chaque fois qu'un même paquet émis par une station subit une collision, un compteur local à la station est multiplié par 2, avant de lui ajouter aléatoirement soit 0 soit 1. La station doit alors attendre un certain nombre d'intervalles de compétition, ce nombre étant défini aléatoirement entre 0 et la valeur courante du compteur de cette station. Si lorsque cette période d'attente se termine, le canal est occupé, la station reporte sa transmission jusqu'à ce que le canal devienne libre.

Si deux stations en collision se fixent une même période d'attente, leurs transmissions entreront de nouveau en collision et leurs compteurs respectifs seront multipliés par deux, réduisant ainsi la probabilité qu'elles tirent à nouveau un même intervalle d'attente avant de retransmettre. Après 16 collisions, une station abandonne et reporte vers l'utilisateur une erreur.

Cette stratégie BEB tente de maximiser le débit réel du bus au prix d'un retard croissant des messages et a un effet LIFO, c'est-à-dire que les stations ayant rencontré peu ou pas de collisions auront plus de chance de transmettre avant les stations qui ont subi davantage de collisions.

Le temps d'accès après i tentatives de transmission est donné par la formule suivante :

$$Ta^i = TFIN + (N^i * CST) + 2 * \Theta, \text{ pour } i \geq 1 \qquad [V.15]$$

$$\text{et } Ta^0 = TFIN$$

avec TFIN :

Temps nécessaire à la détection de la fin d'une transmission (*"end of carrier"*), d'où libération du canal.

avec N^i :

Le nombre aléatoire tiré dans l'intervalle maximum $[0 ; 2^i - 1]$.
$N^0 = 0$.

avec CST :

Une constante qui varie selon le débit en ligne du réseau.

avec $2 * \theta$:

Un intervalle de compétition permettant à la station émettrice de se rendre compte d'une collision.

Le temps d'accès global vaut donc :

$$\sum_{i=0}^{j} Ta^i \qquad [V.16]$$

avec j :

un entier prenant une des valeurs de l'intervalle [0..15].

A la 16ème collision, une erreur est reportée à l'utilisateur. Le protocole n'est donc pas déterministe. Dans les systèmes à temps réel, ce retard non contrôlable n'est pas supportable et le comité 802 a pensé à la méthode du "Token Passing" pour assurer l'accès de chaque station au moyen de communication endéans un temps spécifié (voir point V.8.A et V.8.B).

V.7.C.3. Illustration

Une illustration de ce protocole sur un réseau LAN en bande de base est donnée par ETHERNET (Metcalfe et Boggs, 19), et sur un réseau à large bande, par MITRENET (Hopkins, 9).

John F. Shoch et John A. Hupp (Shoch and Hupp, 25) montrèrent que, sous forte charge, le réseau transmet des portions honnêtes de la charge générée par chaque station. Ils ont testé un Ethernet qui tournait à 2,94 Mbps, s'étendant sur 550 m et connectant 10 stations, chacune générant une charge équivalent à 10 % de la capacité totale du réseau. Pour un débit en ligne s'élevant à 100 % de la capacité totale, ils constatèrent un débit réel avoisinant les 94 %. Seule une portion de 9,3 % à 9,6% de chaque charge fut donc transmise par station. Le message de certaines stations fut donc retardé pendant un temps déterminé. Et ce qui est le plus ennuyeux dans cette méthode d'accès, c'est qu'elle ne fournit aucune priorité aux stations dans l'accès au moyen de transmission.

V.7.D. Le protocole déterministe "Reservation Aloha" de Binder

---

Ce protocole est le premier représentant de l'ensemble traité des protocoles déterministes dérivés d'Aloha (voir point V.6.A).

En effet, pour contrecarrer le caractère probabiliste de l'Aloha, plusieurs chercheurs ont proposé des solutions qui se conduisent comme l'Aloha discrétisé (voir point V.7.A) sous faible charge (<< 1/e) et qui passent graduellement à un fonctionnement similaire à un multiplexage de la voie dans le temps (voir point V.5.A). Ces solutions augmentent le débit réel maximum du canal, au prix d'un temps d'accès certes déterministe mais en moyenne plus élevé.

==========

V.7.D.1 Présentation générale

==========

Dans la solution proposée par Binder, le temps est également subdivisé en intervalles équivalant à la durée de transmission d'une trame et regroupant un certain nombre de tranches-canal mais ici une tranche est assignée à chaque station. S'il y a plus de tranches-canal que de stations, les tranches supplémentaires sont mises à la disposition de la communauté selon la technique de l'Aloha discrétisé (voir point V.7.A). En outre, si le propriétaire d'une tranche-canal n'a pas l'intention de l'employer dans l'intervalle courant, il ne fait rien. Une tranche-canal vide est alors l'indication que son propriétaire n'a généré aucun trafic et que durant le prochain intervalle, cette tranche sera également disponible pour la communauté. Si le propriétaire veut récupérer sa tranche-canal, il y transmet un paquet, provoquant, le cas échéant, une collision si cette tranche-canal est utilisée par un autre émetteur. Une collision sera par conséquent interprétée comme une manifestation de la volonté du propriétaire légitime de la tranche-canal, de la récupérer ; tout autre émetteur devra donc s'en séparer.

V.7.D.2 Etude intuitive du temps d'accès

Ce protocole est déterministe et le temps d'accès, dans le cas le plus défavorable, équivaut à l'écoulement de deux trames. En effet, dans le cas où une station vient juste de laisser passer sa tranche-canal vide, et qu'au même moment elle transfère un paquet dans la file centrale, elle doit attendre la transmission de la prochaine trame avant de pouvoir émettre au risque de provoquer dans sa tranche-canal une collision avec le paquet d'une autre station ayant repéré cette tranche vide. Elle devra donc attendre le passage de la deuxième trame avant de pouvoir être sûre de transmettre sans incident son paquet.

V.7.D.3. Critique du protocole

Un inconvénient de ce protocole est que la seule façon pour le propriétaire d'une tranche-canal d'indiquer qu'il n'a plus rien à transmettre est de ne rien émettre durant la tranche de l'intervalle suivant qui lui est réservée. Pour éviter ce gaspillage de temps, il faudrait ajouter 1 bit dans l'entête de tout paquet pour annoncer que le propriétaire n'a rien à transmettre dans la trame suivante.

Un autre inconvénient est que le nombre d'utilisateurs doit être connu à l'avance et limité comme dans le "*STDM* (voir point V.5.A). Une solution serait d'assigner plusieurs stations à un même champ en espérant qu'ils ne transmettront pas trop souvent simultanément. Pour tenter un arbitrage, chaque station se verrait affecter une priorité statique connue de toutes, la plus basse devant céder à la plus forte en cas de conflit.

Une simulation de ce protocole a indiqué qu'il présentait un meilleur rapport temps de réponse/charge que pour le "*STDM*. En effet, les tranches-canal inutilisées sont disponibles à toutes les stations qui en ont besoin et une station peut donc disposer de plusieurs tranches-canal

c'est-à-dire transmettre plusieurs de ses paquets par trame.

V.7.E. Le protocole déterministe "Reservation Aloha" de Crowther

---

V.7.E.1. Présentation générale

Crowther et ses collègues ont proposé une variante de l'Aloha discrétisé (voir point V.7.A) qui est applicable même quand le nombre de stations est inconnu et varie dynamiquement. Ils subdivisent également le temps en intervalles correspondant à la durée de transmission d'une trame et constitué de R tranches-canal de façon à correspondre au délai de propagation du protocole Aloha (voir point V.6.A). Le nombre de tranches-canal allouées à chacun varie suivant la demande. Il s'agit d'un mixage entre l'Aloha discrétisé et le "*STDM*" qui a reçu le nom de *"slot-switched STDM"* .

Chaque fois qu'une station est seule à émettre durant une tranche-canal, elle la garde dans l'intervalle suivant et ce aussi longtemps qu'elle doit envoyer des données. Les tranches de l'intervalle précédent durant lesquelles aucun paquet n'a été transmis ainsi que celles durant lesquelles une collision s'est produite sont disponibles à toutes les stations selon la technique de l'Aloha discrétisé. En un mot, le canal est multiplexé dans le temps entre toutes les stations qui se sont vues assigner des tranches-canal. Pour cette raison, ce type d'approche porte le nom de circuit virtuel et le temps pendant lequel une station détient le canal est appelé connexion virtuelle.

La technique de l'Aloha est améliorée dans le cas où les stations ont de longs messages à transmettre ou transmettent continuellement. Vu qu'il est très peu probable que toutes les stations envoient simultanément de grands flots de données, cette méthode fonctionne même quand le nombre de tranches-canal par intervalle est inférieur au nombre de stations. Mais si ce cas se produisait, certaines stations se trouveraient dans l'état de privation et le temps d'accès ne serait pas déterministe.

V.7.F. Le protocole déterministe "Frame Adaptable Reservation Aloha with Carrier Sense"

V.7.F.1. Présentation générale

Le protocole *"Frame Adaptable Reservation Aloha with Carrier Sense"* (FARA/CS) propose une solution hybride du "CSMA" (voir point V.7.C), du "Reservation Aloha" (voir points V.7.D et V.7.E) et de la technique "URN" (voir point V.7.J).

Le temps est subdivisé en intervalles de N tranches-canal, ces intervalles servant à la transmission de trames de N paquets dont chaque station analyse le contenu. Durant une courte période précédant la transmission d'une nouvelle trame, chaque station détermine le nombre de tranches-canal dans l'intervalle écoulé où aucun paquet n'a été transmis (dénommée, dans la suite, tranche-canal vide), où une collision s'est produite (dénommée tranche-canal siège de collisions) et où un seul paquet a été transmis (dénommée tranche-canal réservée) pour estimer le trafic et adapter la stratégie de transmission. Etant donné que toutes les stations sont à l'écoute du canal, elles devraient toutes arriver à la même estimation du trafic. Afin de diminuer le risque de collisions dans un intervalle, chaque station recourt à une règle de probabilité pour déterminer le nombre et l'emplacement de ses futurs paquets.

Il s'agit d'un des seuls protocoles qui fasse la différence entre ancienne station, celle qui a transmis durant une tranche-canal de l'intervalle précédent, et nouvelle station. Il évite qu'une ancienne prenne le canal à ses propres fins alors que de nouvelles stations sont en attente. Il est supposé que chaque ancienne station laissera en compétition toute tranche-canal en excès si le nombre de paquets à transmettre dans la prochaine trame est inférieur au nombre de tranches-canal à sa disposition.

Analysons l'algorithme développé par une station pour adapter sa stratégie de transmission au trafic estimé.

Si le réseau se trouve dans l'état numéro 1 c'est-à-dire où le nombre de tranches-canal vides et siège de collisions est supérieur à N/3 **(niv1)**, l'algorithme adopté ressemble au "Reservation Aloha" de Crowther (voir point V.7.E) :

> Les nouvelles et anciennes stations, c'est-à-dire celles qui disposaient d'une ou de plusieurs tranches-canal lors de la transmission de la trame précédente entrent en concurrence, sur une base honnête, pour le partage des tranches-canal vides et siège de collisions de l'intervalle précédent.

Si le réseau se trouve dans l'état numéro 2 c'est-à-dire que le nombre de tranches-canal vides et siège de collisions est supérieur ou égal à N/8 **(niv2)** et inférieur ou égal à N/3 **(niv1)**, l'algorithme adopté ressemble encore au "Reservation Aloha" de Crowther (voir point V.7.E) :

> Les nouvelles stations entrent en concurrence avec les anciennes qui ne disposaient que d'une tranche-canal durant l'intervalle écoulé pour pouvoir émettre durant les tranches-canal vides et siège de collisions. Les anciennes stations disposant de plus d'une tranche-canal durant la transmission de la trame précédente ne peuvent plus prétendre à de nouvelles tranches.

Si le réseau se trouve dans l'état numéro 3 c'est-à-dire que le nombre de tranches-canal vides et siège de collisions est inférieur à N/8 **(niv2)**, le protocole se rapproche du "Reservation Aloha" de Binder (voir point V.7.D) :

> Les nouvelles stations ne transmettent que dans la tranche-canal qui leur est allouée ; elles n'entrent pas en concurrence pour les tranches-canal vides et siège de collisions ; elles peuvent entrer en collision avec une ancienne station occupant abusivement leur tranche-canal.
> Les anciennes stations disposant d'une tranche-canal dans l'intervalle précédent la gardent si elle n'a pas été le siège de collisions et ne prétendent à aucune autre ou la perdent si elle a été l'objet d'une collision et transmettent dans la tranche-canal qui leur est allouée comme une nouvelle station.
> Les anciennes stations disposant de plus d'une tranche-canal dans l'intervalle précédent perdent à l'exception de la tranche qui leur est

allouée, disons, un tiers de leurs tranches si une ou plusieurs collisions se sont produites durant la transmission de la trame précédente.
Au fur et à mesure que des collisions se produisent, les anciennes stations perdent de leurs tranches-canal et le nombre de tranches-canal libres ou siège de collisions augmente permettant au réseau de revenir dans l'état 2 ou 1.

V.7.F.2 Etude intuitive du temps d'accès

Ce protocole se conduit de manière déterministe car dans le cas le plus défavorable où les stations sont de plus en plus voraces (forte charge), plus aucune tranche-canal ne sera disponible et le système passera à un système de multiplexage de la voie dans le temps ("*STDM*") qui est déterministe (voir point V.5.A).

V.7.F.3. Illustration

Une simulation très bien décrite dans l'article de A.K. Elhakeem (Elhakeem, Goel et Dhawan, 7) a été tentée pour estimer la performance du protocole tant des points de vue temps d'accès que débit réel ou en ligne. L'idée a été de recourir à un compteur incrémenté chaque fois qu'un paquet est prêt à être transmis et qu'une tranche-canal d'un intervalle s'écoule sans qu'il le soit. Le comptage s'arrête dès qu'une transmission fructueuse s'est produite.

Par cette méthode, on a remarqué qu'avec λ proche de 1 paquet/sec et un nombre croissant de stations, le temps d'accès au canal des paquets reste déterministe. En effet, le réseau travaille en "*STDM*" et le retard maximal subi par un paquet ne dépassera jamais le temps de transmission de deux trames car pour qu'une station récupère sa tranche-canal détenue par une autre station, elle doit générer, durant sa tranche-canal de l'intervalle

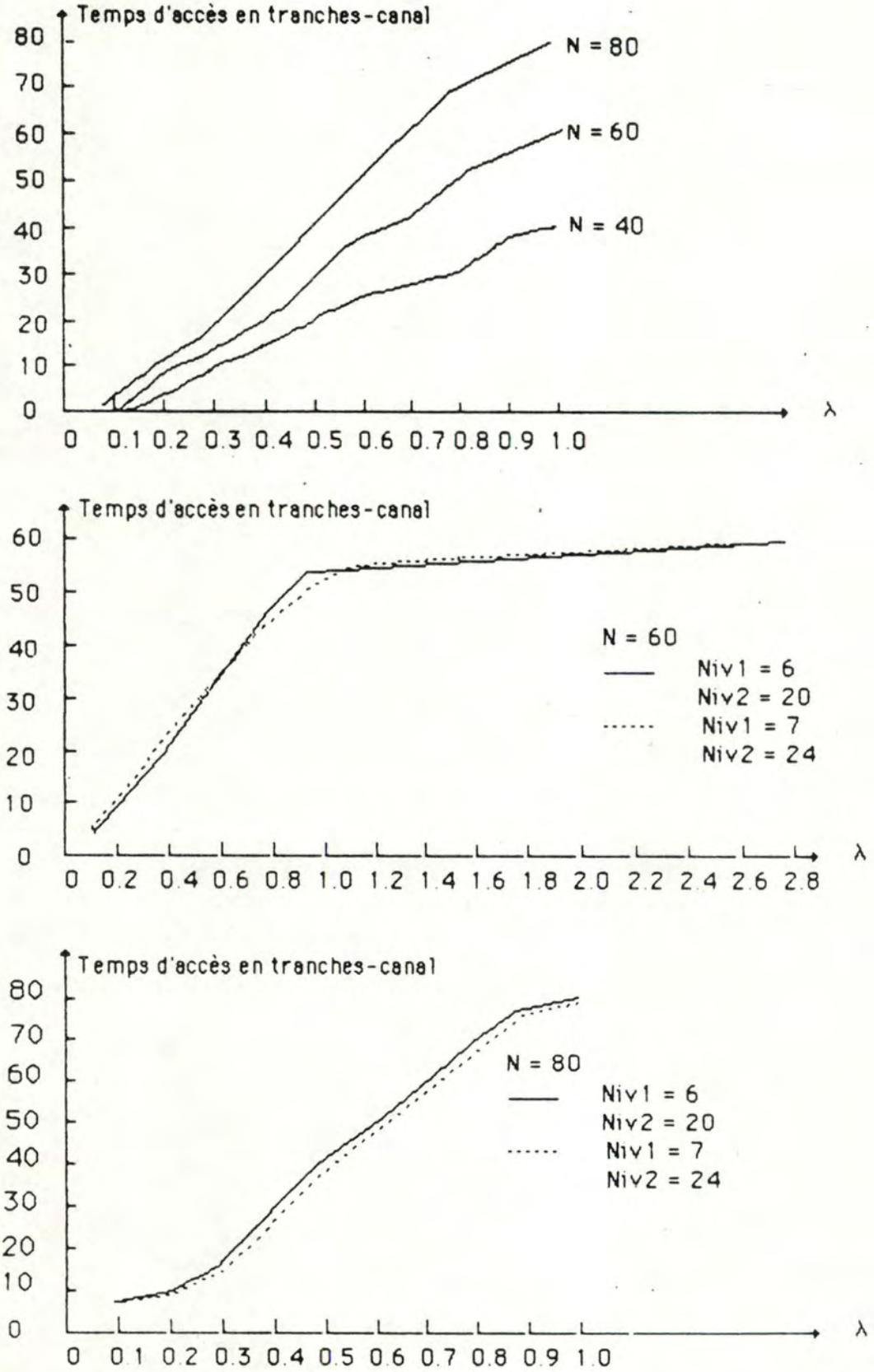


Figure V.10. : Temps d'accès

suivant, une collision, ce qui fait qu'elle pourra placer un paquet dans la trame suivante. D'autres essais montrent que pour améliorer la performance quand le nombre de stations augmente, il faut jouer sur les valeurs de **Niv1** et **Niv2** (voir figures V.10).

V.7.G. Le protocole déterministe "Simple Cooperative Symmetric Algorithm"

Nous présentons maintenant le seul protocole déterministe à écoute du canal pour y détecter les collisions.

==========

V.7.G.1 Présentation générale

==========

Le protocole "*Simple Cooperative Symmetric Algorithm*" modélisé par B.Mahbod et J.Howard (Mahbod et Howard, 18) (voir figure V.11.) est une technique de réservation suite à une collision. Il se conduit comme un protocole CSMA sous faible charge (voir points V.2 et V.7.D) et se rapproche de la politique du multiplexage de la voie dans le temps ("STDM", voir point V.5.A) quand la charge croît.

A tout instant, les stations sont à l'écoute du canal pour y détecter la présence d'une collision. Dès qu'une station a un paquet à transmettre, elle le fait. Si, durant l'intervalle de compétition entourant l'émission de son paquet, elle repère une collision, elle émet un signal de brouillage pour veiller à ce que toutes les autres stations s'en soient rendues compte et on entre dans une phase discrétisée où le temps est subdivisé par intervalle de propagation.

Durant les N premiers intervalles de θ sec, les stations impliquées dans la collision annoncent, selon leur séquence de numérotation et dans leur intervalle respectif, leur désir de transmettre. Cette période durera $N*\theta$ sec. Ce temps écoulé, l'information sera disponible à toutes les stations et les collisions seront évitées jusqu'à ce que tous les messages impliqués dans la

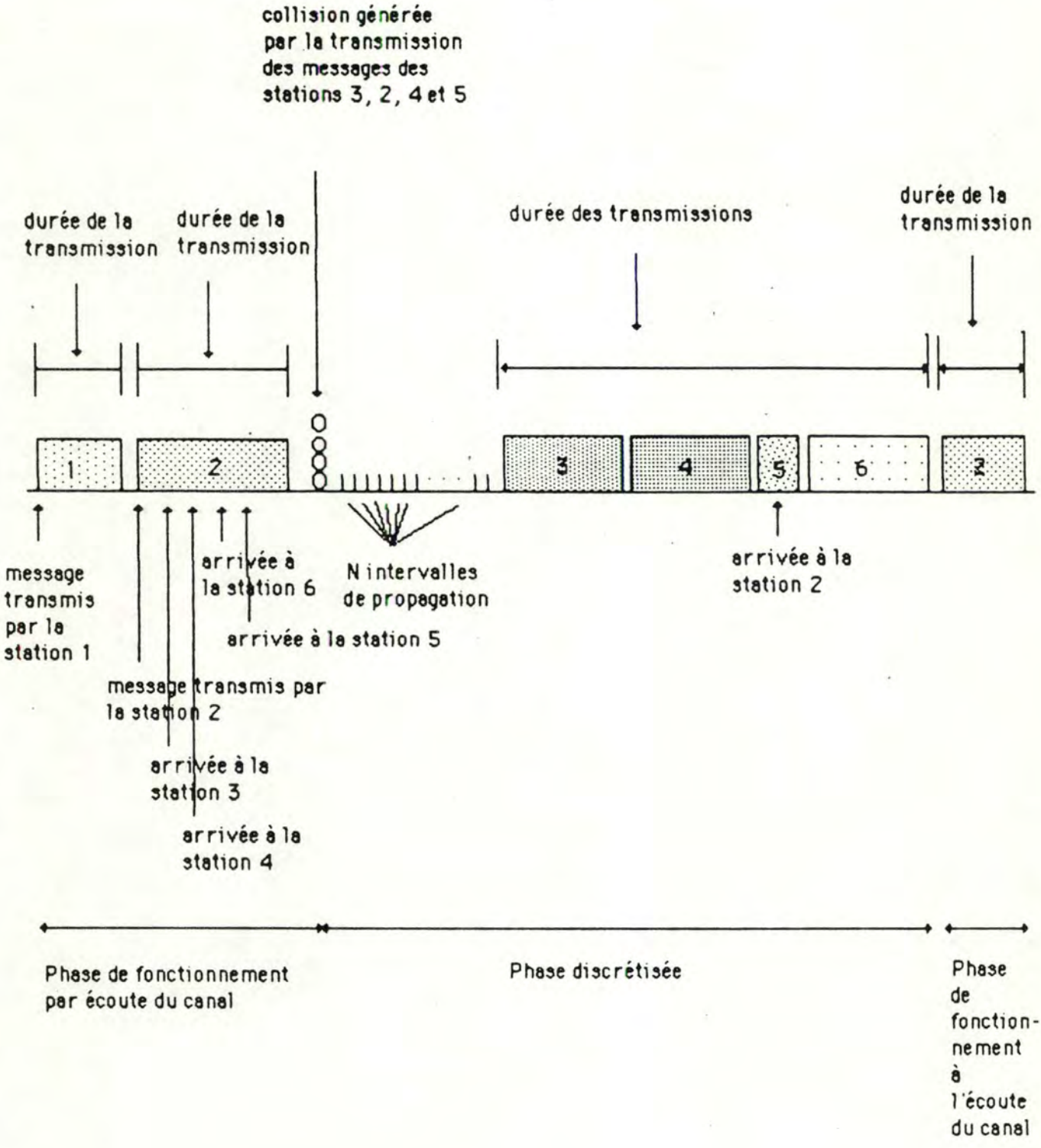


Figure V.11. : Fonctionnement du protocole "SCSA"

collision aient été transmis sans incident. Par opposition au CSMA-CD/BEB (voir point V.7.C), le SCSA garantit que les paquets plus anciens seront transmis avant les plus jeunes, c'est-à-dire dans l'ordre FIFO.

Par la suite, le système revient en mode de fonctionnement par écoute du canal.

V.7.G.2 Etude analytique du temps d'accès

Si, en phase discrétisée, un nouveau paquet arrive à une station, il devra attendre le retour en fonctionnement par écoute du canal. Ce paquet sera soit le seul à être transmis, soit il donnera lieu à une collision et le retour à la phase discrétisée. Le temps d'accès d'un paquet au canal s'interprète donc comme suit :

- dans le cas le plus favorable la station est la seule à vouloir émettre, et le temps d'accès porte alors sur la fin d'une phase discrétisée ou la fin de l'émission d'une autre station ;
- le cas le plus défavorable se produit pour la Nième station (priorité la plus basse), lorsqu'une collision résultant de paquets émis par toutes les autres stations, vient juste de se produire avant qu'elle ne puisse émettre son paquet (elle doit alors attendre toute une première phase discrétisée), puis lorsqu'une nouvelle collision se produit lors de l'émission de son paquet en concurrence avec toutes les autres stations (elle doit encore attendre la plus grande partie d'une deuxième phase discrétisée, avant de pouvoir finalement émettre son paquet).

Dans le cas le plus défavorable, le temps d'accès pour la station de plus basse priorité pourra donc s'exprimer par la relation :

$$Ta = (2* CRP) + (2* (N-1)* Ts) \qquad [V.17]$$

avec CRP :

la période de résolution de collision c'est-à-dire le temps de détection de la collision et le temps nécessaire aux stations du

système de se synchroniser (Θ * N) ; 2 * CRP signifie que l'on tient compte de la phase discrétisée en cours lors de l'arrivée du nouveau paquet ainsi que de la nouvelle phase pour résoudre la collision dans laquelle le nouveau paquet est impliqué.

avec (N-1) * Ts :

le temps de service des paquets de taille variable des N-1 stations précédant la station de plus basse priorité, dans une phase discrétisée.

Si le lecteur est intéressé par la déduction du temps d'attente moyen dans la file de l'émetteur sous l'hypothèse que les stations n'ont pas de tampon, c'est-à-dire qu'une station ne génère pas de nouveau paquet tant qu'elle en a à transmettre, ou sous l'hypothèse que les possibilités de tampon sont illimitées (cas plus réaliste où les paquets sont générés indépendamment du trafic sur le canal), nous lui conseillons la lecture de l'article écrit par Mahbod et Howard (18)).

V.7.H. Le protocole déterministe "Adaptive Tree Walk"

Nous présentons maintenant l'ensemble des protocoles déterministes à l'écoute du canal et à collisions limitées.

Lorsque le débit réel est faible, ces protocoles se conduisent comme les protocoles à écoute du canal avec détection de collisions car ils impliquent peu de retard. En effet, le calcul de la probabilité A qu'une station acquière fructueusement le canal durant un intervalle de compétition (voir point V.2), si k stations sont prêtes à transmettre et si chaque station transmet durant cet intervalle 2 * Θ avec la probabilité p, nous amène à la formule suivante :

$$A = k * p * (1 - p)^{k - 1} \qquad [V.18]$$

La valeur optimale de p est obtenue en différentiant A par rapport à p et en égalant à zéro pour donner p = 1/k. La figure V.12. nous montre que pour un petit nombre de stations en compétition, la chance de succès est bonne mais que dès que le nombre atteint 5, la probabilité A tombe à sa valeur

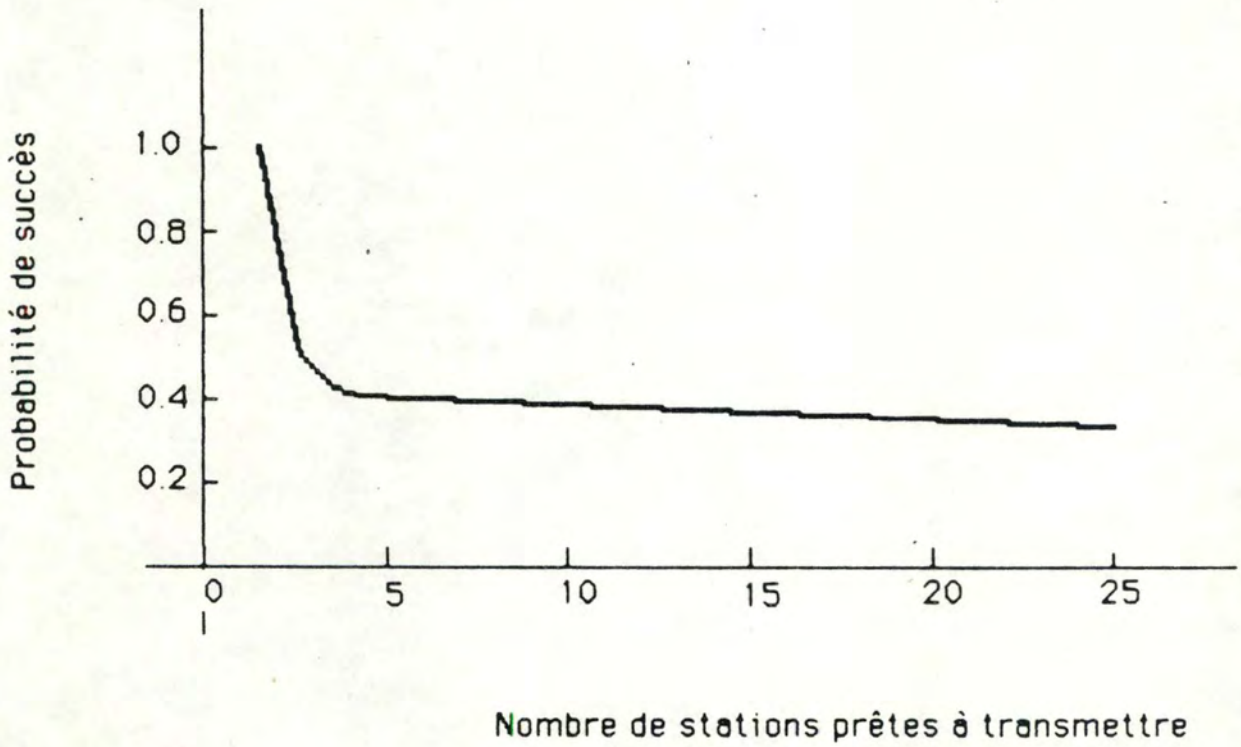


Figure V.12. : Probabilité d'acquisition du canal dans le cas où chaque station essaie de l'acquérir avec la même probabilité P

asymptotique de 1/e. Cette observation a pour conséquence que, sous forte charge, ces protocoles virent vers les protocoles à écoute du canal pour y éviter les collisions de façon à assurer un accès déterministe au canal.

Pour ce faire, les stations sont subdivisées en groupes dynamiquement, avec beaucoup de stations par groupe quand le débit réel est faible et peu (une seule station) pour éviter les collisions quand le débit réel est fort.

V.7.H.1. Présentation générale

Le protocole *"Adaptive Tree Walk"* a été proposé par Capetanakis et Hayes (Capetanakis, 5). L'idée est de considérer chacune des N stations comme feuille d'un arbre binaire distribué et de déterminer sa position en fonction de l'urgence des paquets qu'elle transmet (voir figure V.13.).

Le temps est discrétisé en intervalles de compétition. Si, dans le premier intervalle de compétition (numéro 0) suivant une transmission fructueuse de paquet, une seule station émet, alors il n'y a pas de problème.

Si, par contre, une collision se produit, l'ensemble des stations de l'arbre est partitionné en deux sous-arbres, de sorte que seules les stations dans le sous-arbre de gauche (sous le noeud B) peuvent transmettre durant l'intervalle de compétition suivant. Les stations dans le sous-arbre de droite (sous le noeud C) ne pourront transmettre tant que toutes les stations de gauche qui sont impliquées dans la collision (et elles seules) n'auront pas émis correctement leurs paquets.

Le problème de transmission concurrente de plusieurs stations de l'arbre se réduit ainsi à la gestion de la transmission des stations du sous-arbre de gauche, puis à celle des stations du sous-arbre de droite. Ce processus de découpe d'un arbre en deux sous-arbres se poursuit de manière récursive, en profondeur d'abord, puis de gauche à droite, tant que des collisions se produisent. A un certain moment, la découpe se terminera faute de collision : ou bien, il ne reste qu'une seule station qui doive émettre, ou bien, il n'y en a plus aucune (intervalle vide).

La figure V.13. illustre à l'étape numéro 1 le cas où toutes les stations qui ont un message à transmettre deviennent actives et entrent en collision. Nous appliquons la politique de profondeur d'abord à partir de la racine de l'arbre. Les stations 4 et 7 sont provisoirement écartées et il ne reste plus que deux stations (0 et 1) qui essaient de transmettre à l'étape numéro 2. Une nouvelle collision se produit et nous descendons d'un noeud à l'étape numéro 3, où une troisième collision se reproduit. A l'étape numéro 4, nous sommes aux feuilles de l'arbre. La station 0 transmet sans problème et nous donnons, à l'étape 5, parole à la station dans le sous-arbre droit c'est-à-dire le station numéro 1 qui transmet sans encombre. Nous poursuivons par le noeud contenant les stations 2 et 3 à l'étape 6. Elles ne sont pas actives et un intervalle de compétition vide est généré. Nous passons alors à l'étape 7 de gauche à droite au sous-arbre contenant les stations 4 et 7. Elles entrent en collision et par application du principe de profondeur d'abord, nous permettons à la station 4 de transmettre à l'étape numéro 8 puis à la station 7 ensuite (pour plus d'informations, on consultera Stuck, 28).

Des améliorations peuvent être introduites à ce protocole. Si le nombre de stations prêtes peut être estimé, l'arbre peut être partitionné de manière à maximiser la probabilité que l'ensemble des stations pouvant émettre contient une seule station prête. Sous forte charge, il vaut mieux commencer par les feuilles de l'arbre car il est certain qu'à commencer par la racine, on tombera sur une collision. Pour éviter que par le processus de partitionnement, ce soit toujours les mêmes stations qui puissent émettre avant d'autres, on peut alterner l'allocation des feuilles aux stations. Sous faible charge, on pourra commencer au sommet de l'arbre et laisser à toutes les stations la possibilité d'émettre.

V.7.H.2. Etude analytique du temps d'accès

Ce protocole est déterministe. En excluant l'introduction des améliorations, le temps d'accès dans le cas le plus défavorable, c'est-à-dire le temps d'accès pour la Nième station, lorsque toutes les stations ont un message à émettre, s'exprime par la formule suivante :

$$Ta = ((2 * \Theta) * ((2 * N) - 1)) + ((N-1) * Ts) \qquad [V.19]$$

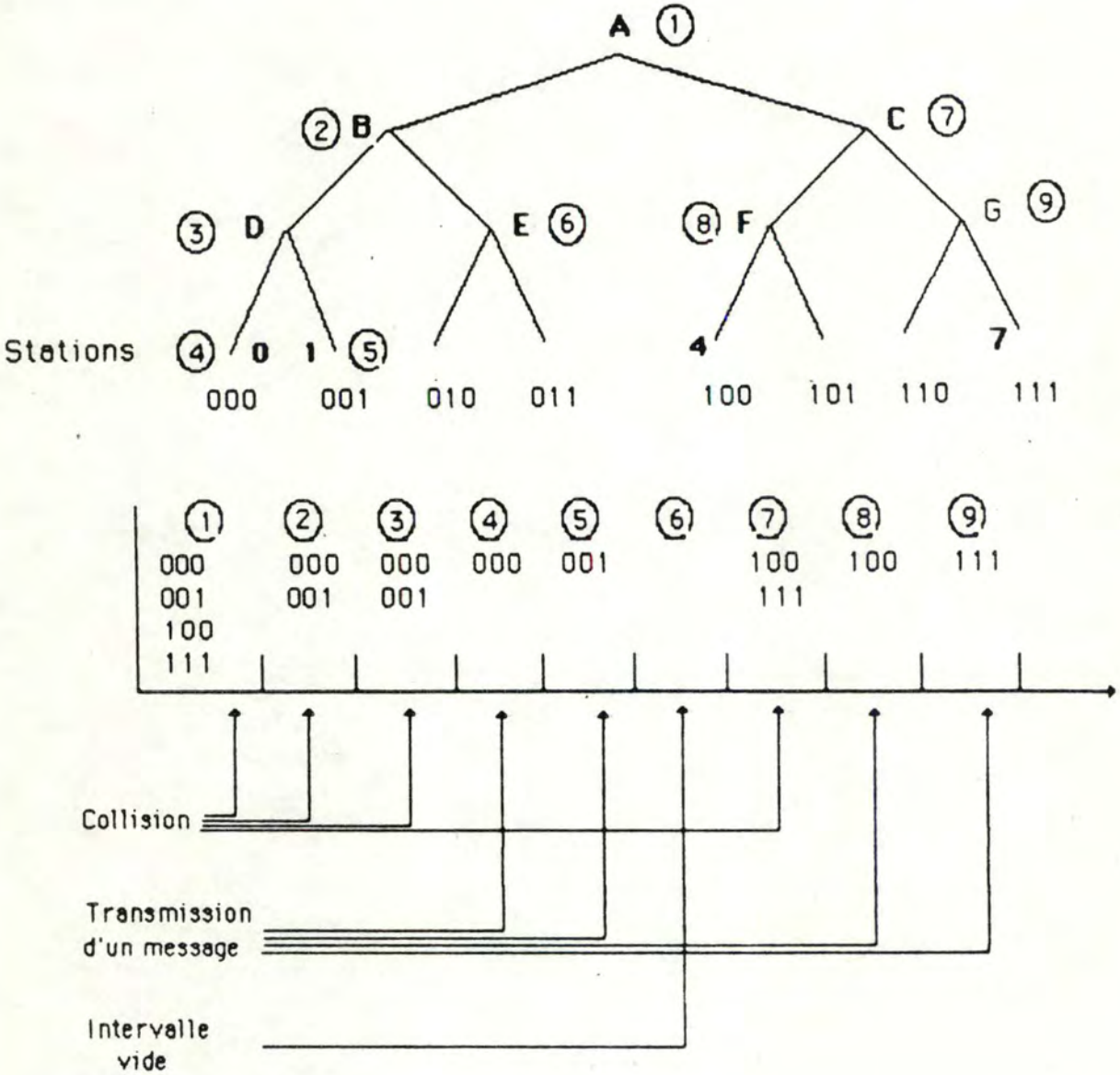


Figure V.13. : Illustration du protocole "ADAPTIVE TREE WALK"

avec (2 * N) - 1 :

Cet élément peut être décomposé en la somme de deux composants :

- **((2 * N) - 2)** qui est le nombre d'intervalles de compétition **(2 * Θ)** nécessaires pour départager les N stations impliquées dans la collision, avant de permettre à la Nième station d'émettre ;
- 1 qui est l'intervalle de compétition **(2 * Θ)** sous la racine durant lequel toutes les stations sont entrées en collision.

avec (N-1)*Ts :

Le temps de service des messages de longueur variable des N-1 stations précédant la dernière station.

Si l'on introduit maintenant les améliorations, ce temps peut être réduit en commençant la recherche au niveau judicieux de l'arbre. Supposons que chaque station estime à Q le nombre de stations prêtes uniformément distribuées. Le nombre de stations prêtes sous un noeud spécifique au niveau i est de $2^{-i} * Q$. On voudrait commencer au niveau où le nombre de stations en compétition par intervalle est de 1 c'est-à-dire :

$$2^{-i} * Q = 1 \quad \text{ou} \quad i = \log_2 Q \qquad [V.20]$$

V.7.1. Le protocole déterministe "URN"

V.7.1.1. Présentation générale

L'URN protocole dont la base est une urne a été décrit par Kleinrock et Yemini (Kleinrock et Yemini, 13). Il limite également le nombre de stations pouvant émettre durant un intervalle de compétition afin de maximiser la probabilité de n'obtenir qu'une seule station prête par intervalle.

Soit N stations dont n sont prêtes à émettre, représentons-les par des

billes vertes, les rouges étant celles qui n'ont rien à émettre. Le partitionnement en groupes consiste à sélectionner k stations d'une urne imaginaire contenant N stations. Si les k stations n'ont parmi elles qu'une seule station désirant émettre, le partitionnement s'avère fructueux. Cette opération revient à tirer une seule bille verte de l'urne quand nous tirons k billes sans replacement et la probabilité que cela se produise vaut :

$$\Pr[\text{une seule station doit émettre}] = (C_n^{\,1} * C_{N-n}^{\,k-1})/C_N^{\,n} \qquad [V.21]$$

Si les k stations contiennent une ou plusieurs stations prêtes, le processus de partitionnement doit être répété. La valeur de k qui maximise la probabilité qu'une seule station soit prête est donnée par la partie entière de N/n. Sous forte charge où chaque station a un message à émettre, n vaut N ; k est donc égal à 1 et cela même dès que $n \geq N/2$. Le protocole se comporte comme le "STDM" (voir point V.5.A). Sous faible charge, le nombre n de stations prêtes vaut un et le protocole se conduit alors comme l'Aloha discrétisé (voir point V.7.A).

Quelles stations doivent être choisies ? Les stations sont imaginées disposées en ordre numérique le long de la circonférence d'un cercle hypothétique. Une fenêtre encadrant k stations se promène le long du cercle et désigne les seules stations qui peuvent émettre. Si une transmission fructueuse a eu lieu ou qu'aucune ne s'est produite, la fenêtre est avancée pour donner la parole aux k stations suivantes. S'il y a une collision, la fenêtre est réduite de moitié et le processus est répété jusqu'à ce que les collisions cessent. Il faut que chaque station ait une estimation du nombre de stations désireuses de transmettre pour permettre une estimation de la largeur de la fenêtre.

V.7.I.2. Etude intuitive du temps d'accès

Ce protocole est déterministe, car sous forte charge, il se comporte comme un protocole de multiplexage de voie dans le temps qui est déterministe (voir point V.5.A).

V.7.J. Le protocole déterministe "Time partitioning"

V.7.J.1. Présentation générale

Le concept de base de ce protocole proposé par Gallager (Gallager, 8) en 1978 est celui de fenêtre : cette fenêtre est un intervalle de temps permettant de constituer un ensemble de stations qui peuvent émettre, parce qu'elles ont reçu des paquets durant cet intervalle de temps passé (voir figure V.14.).

Le protocole maintient une valeur t_0 telle que tous les messages générés avant ce temps ont été transmis. Le temps est discrétisé par intervalle de propagation (~~θ~~-sec). Si le canal qui est scruté par toutes les stations s'avère vide au début d'un intervalle de propagation, toutes les stations sélectionnent une dimension de fenêtre initiale W et transmettent si elles ont reçu un message durant l'intervalle passé $[t_0, t_0+W]$.

Quand une fenêtre initiale a été déterminée, trois possibilités peuvent surgir :

1. Si aucun paquet n'a été reçu sur l'intervalle de temps fixé par la fenêtre, le canal reste oisif ; la valeur de t_0 est mise à jour (t_0+W) et une nouvelle fenêtre initiale est sélectionnée.

2. Si une seule station avait reçu un paquet durant cet intervalle, sa transmission sera fructueuse ; une fois la transmission achevée, t_0 pourra être mis à jour (t_0+W) et une nouvelle fenêtre initiale déterminée.

3. Si deux ou plusieurs stations avaient reçu des paquets durant l'intervalle de temps fixé par la fenêtre, ces paquets entreront en

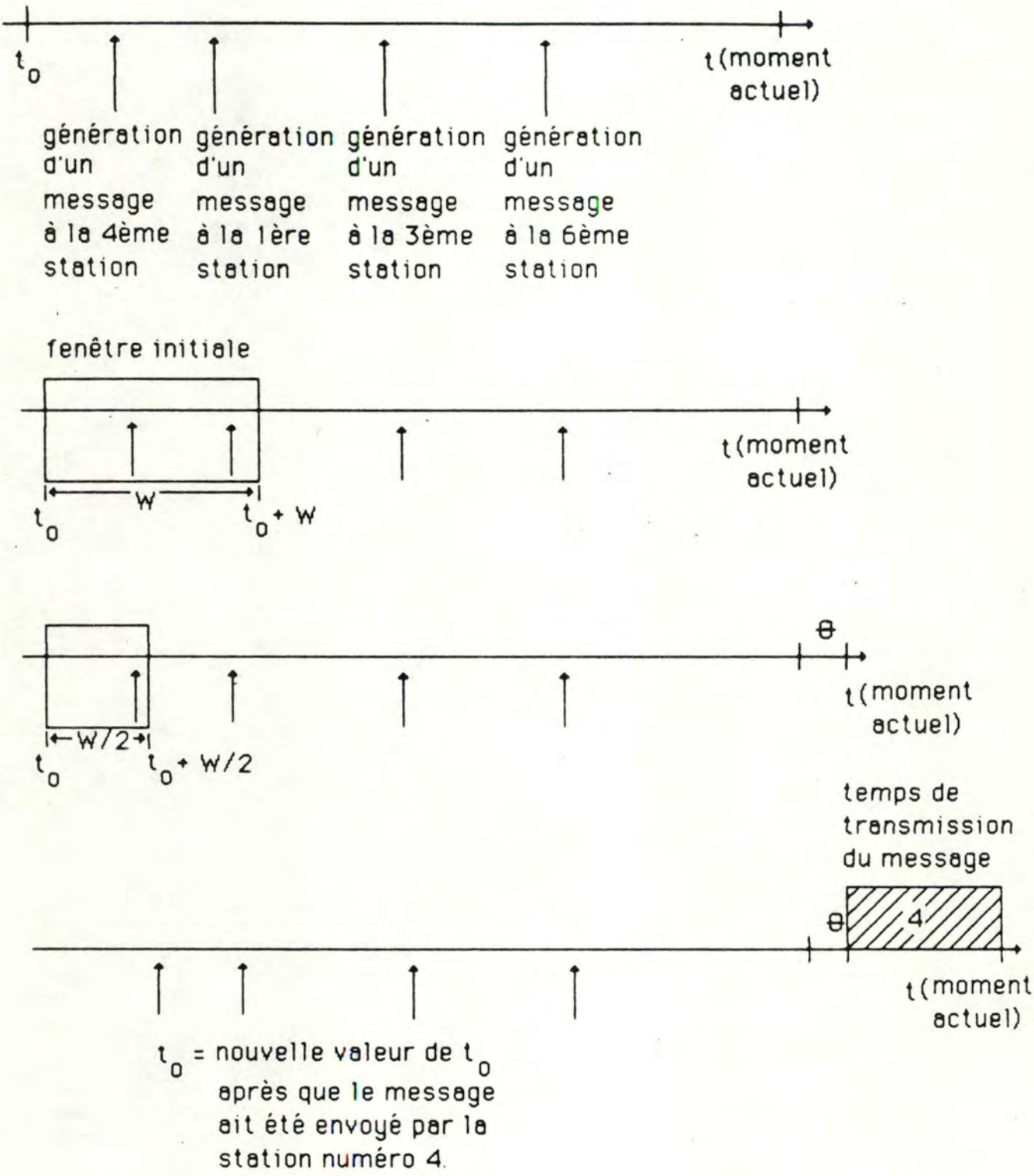


Figure V.14 : Illustration du protocole "TIME PARTITIONING"

collision et les stations le détectant arrêteront la transmission ; la taille de la fenêtre est alors réduite de moitié $[t_0, t_0+W/2]$. Si aucun message n'avait été reçu durant cette fenêtre, un intervalle vide se produit et le processus se poursuit pour la fenêtre $[t_0+W/2, t_0+W]$ et ainsi de suite...
Ainsi, dans la figure V.14., les stations 1 et 4 cessent de transmettre à la détection d'une collision (deuxième axe). Une demi-fenêtre est déterminée au début de l'intervalle de propagation suivant et la procédure est réappliquée (troisième axe). Quand une fenêtre $[t_0+W', t_0+W'']$, avec $W' < W''$, contenant la transmission fructueuse d'un paquet se sera produite, t_0 prendra la valeur t_0+W'' (quatrième axe).

V.8. Etude de la feuille numéro 5 de l'arbre binaire : multiplexage de la voie de communication dans le temps, selon un mode de division asynchrone du temps, selon un algorithme de résolution d'accès au canal distribué et évitant les collisions et selon un ordre d'accès au canal déterminé dynamiquement

---

Cette feuille regroupe les protocoles recourant à la technique du multiplexage de la voie de communication dans le temps, selon un mode de division asynchrone du temps et dont la politique d'accès au canal est réglementée : l'algorithme d'accès est distribué et évite les collisions et l'ordre d'accès au canal est déterminé dynamiquement.

Nous aborderons d'abord des protocoles qui se partagent la voie par accès contrôlé (le "Token Bus" (V.8.A), le "Token Ring" (V.8.B), l'"Adaptive Token Ring" (V.8.C), le "Token Passing Multipacket Ring" (V.8.D), le "Slotted Ring" (V.8.E) et le "Register Insertion Ring" (V.8.F)) avant de passer aux protocoles qui relèvent des protocoles à l'écoute d'une transmission sur le canal pour y éviter les collisions (le "CSMA-CA" (V.8.G), le "Bit-Map" (V.8.H), le "BRAP" (V.8.I)).

V.8.A. Le protocole déterministe "Token Bus"

Présentons d'abord la famille des protocoles qui se partagent la voie de communication par accès contrôlé.

V.8.A.1 Présentation générale

Dans le protocole *"Token Bus"* défini par la norme 802.4 (voir point IV.3.A) du modèle 802 des réseaux LAN, les stations constituent un anneau logique (voir figure V.15.) sur une voie à diffusion physique sur laquelle elles sont interconnectées via un interface passif. Chaque station se voit assigner une position dans une séquence logique et connaît l'identité de son prédécesseur et son successeur.

Lorsqu'elle a fini de transmettre, une station origine envoie une configuration-jeton, qui n'est rien d'autre qu'un paquet de contrôle, à son successeur. Si ce dernier n'a rien à émettre, il passe immédiatement le jeton au suivant dans la séquence logique ; ceci a comme conséquence que l'émission d'un paquet par un successeur rassure le noeud origine sur le bon fonctionnement de son successeur. De même, dès qu'une station a fini de transmettre ou que son temps de possession du jeton a expiré, elle transmet le jeton à son successeur. Les stations ne possédant pas le jeton ne peuvent répondre qu'aux demandes de confirmation. En coordonnant de cette manière la possibilité de transmettre des stations, cette méthode empêche les paquets de se gêner.

Une gestion dynamique de l'anneau est nécessaire car à tout moment, des stations peuvent désirer s'insérer dans l'anneau (périodiquement, la mission incombe au détenteur du jeton de donner la possibilité à de nouvelles stations de s'intégrer) ou se retirer (la station désireuse de se retirer doit prévenir son successeur et prédécesseur) (pour plus de détails, on consultera Stallings, 26).

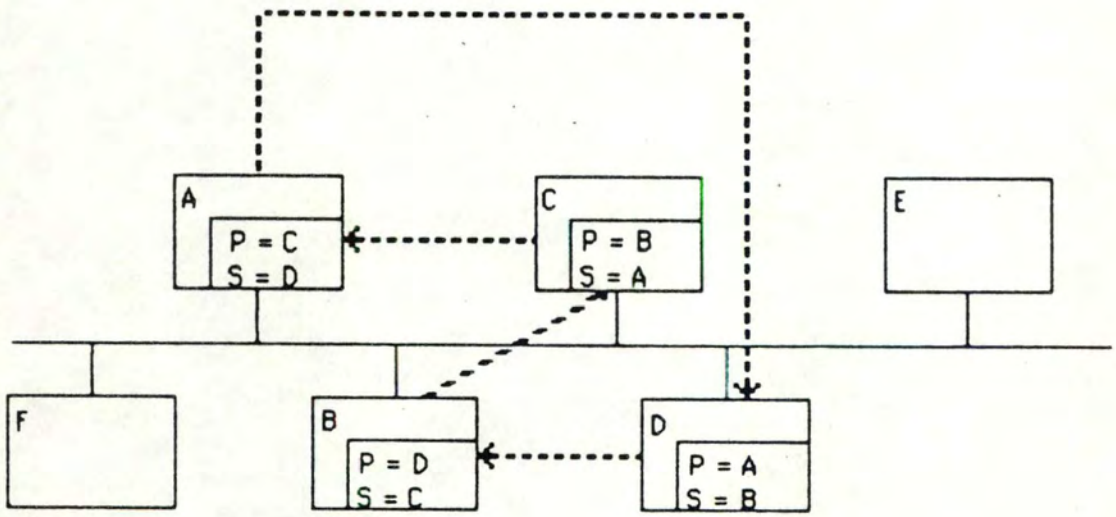


Figure V.15 : Anneau logique du protocole "TOKEN BUS"

V.8.A.2. Etude intuitive du temps d'accès

Comme le temps de possession du jeton par une station est limité et que chaque station reçoit le jeton en séquence, ce protocole est déterministe. Dans le cas le plus défavorable où toutes les stations ont des paquets à transmettre, le temps d'accès d'une station revient à la somme des temps de propagation du jeton de station à station dans leur séquence logique et du temps d'émission de tous les paquets par les N-1 autres stations.

Pour les applications temps réel, le nombre de stations et la longueur maximale des paquets peuvent être fixés pour procurer l'accès endéans une période fixée. Comme option, le système peut inclure des classes de service qui procurent un mécanisme d'accès par priorité à la voie de diffusion : la classe 6 désigne le trafic synchrone, la classe 4, le trafic asynchrone et urgent et la classe 2, le trafic asynchrone et normal. On peut décider qu'une station réalisant des fonctions moins critiques place un paquet sur la voie de diffusion après chaque seconde ou après le nième passage du jeton. Dans cet ordre d'idées, si une station entretient une charge de 9,6 kbps de données et de 64 kbps de voix, des paquets de voix pourraient être transmis 7 fois plus souvent que des paquets de données. On peut aussi permettre aux stations à haute priorité d'envoyer de plus longs messages que les stations moins importantes qui ne pourraient transmettre que sous faible charge.

V.8.A.3 Illustration

Une illustration de ce protocole est le réseau Gixinet mis sur le marché par Concord Data Systems.

V.8.B. Le protocole déterministe "Token Ring"

V.8.B.1 Présentation générale

Défini par la norme 802.5 du modèle des réseaux locaux LAN mais non encore approuvé par l'IEEE (voir point IV.3.A), le protocole "*Token Ring*" diffère du "*Token Bus*" (voir point V.8.A) sur un seul point : il s'agit d'un anneau physique unidirectionnel reliant les interfaces actives (répéteurs) de stations par des liaisons point-à-point. Son ancêtre est la boucle de Farmer et Newhall introduite en 1969 (Pour plus détails, on consultera Mahbod et Howard, 18).

L'analyse de ce protocole est d'autant plus intéressante qu'elle pourra nous permettre de mieux comprendre ce qui a guidé les concepteurs du LEP (voir point III.2) a le choisir pour gérer le dialogue des différents satellites contrôlant cet accélérateur.

Le fonctionnement correct du système est dicté par l'unicité du jeton : une station moniteur s'assure qu'il n'y en a qu'un. Supposons que le jeton est défini par une configuration de huit bits mis à un. Si une station désireuse de transmettre le reçoit, elle doit d'abord mettre à zéro le dernier bit pour indiquer que le jeton est occupé avant d'insérer le paquet de données qui peut être de taille arbitrairement large (voir figure V.16.). La station ayant lu tous les bits sauf le dernier doit avoir le temps de le lire et de l'inverser, le cas échéant, avant de le remettre sur la boucle. En effet, la configuration peut être celle d'un jeton libre si le dernier bit est à un ou celle d'une donnée ou d'un jeton occupé si le dernier bit est à zéro. Chaque bit arrivant à une station doit donc y être stocké dans son entièreté et un nouveau bit généré à la place, créant un retard d'un bit dans chaque répéteur. Si le nombre de stations sur l'anneau est important, l'effet cumulatif de ces retards d'un bit peut avoir un impact important sur la performance globale du réseau. Si la station n'a rien à émettre, elle répète le jeton appelé aussi champ de contrôle de l'accès sans le modifier.

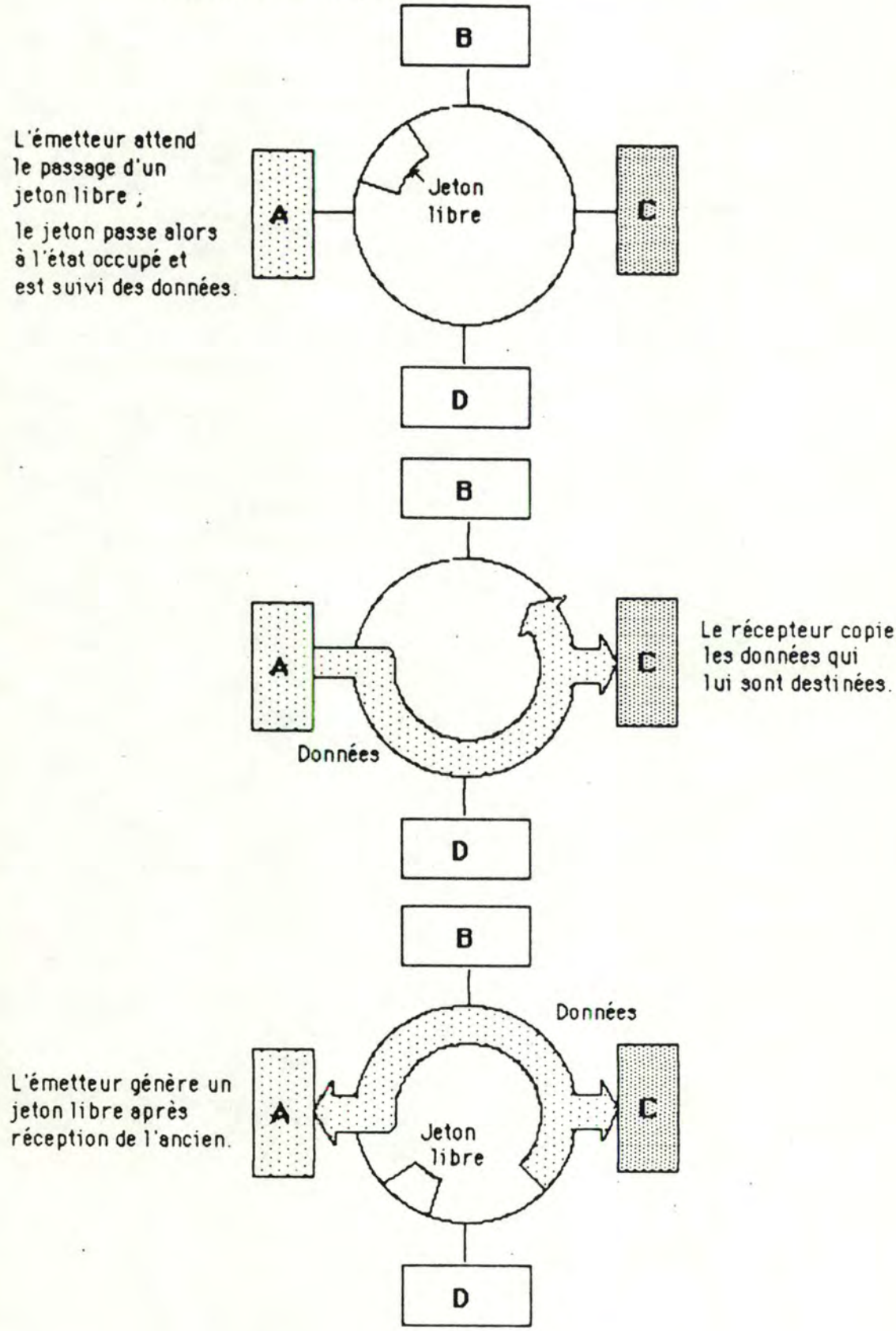


Figure V.16. : Illustration du protocole "TOKEN RING"

Encore faut-il, bien sûr, que le réseau soit suffisamment long pour contenir le jeton lorsque toutes les stations sont désactivées. En effet, si la vitesse de transmission est de 1 Mbps, un bit est émis toutes les 1 µsec et si la vitesse de propagation est de 200 m/µsec, chaque bit occupe 200 m sur l'anneau. Un anneau de 1000 mètres de circonférence ne peut donc contenir que 5 bits, qui porte le nom de longueur en bits ("*bit length*"). Il faut en tenir compte car parfois, il faut agrandir artificiellement un anneau pour qu'il puisse contenir le jeton.

Le paquet circule sur tout l'anneau avant d'être retiré par la station émettrice qui insère un nouveau jeton libre sur l'anneau si elle n'a plus rien à transmettre ou que son temps de détention du jeton a expiré (voir figure V.16.). Si la longueur en bits du réseau est moindre que la longueur du paquet, quand la transmission sera terminée, le jeton occupé sera revenu à la station et il ne circulera qu'un seul jeton sur l'anneau ; cette version du "*Token Ring*" appelée "*Single-Token*" est préconisée par la norme 802.5 où un nouveau jeton ne peut être explicitement émis qu'après récupération de l'ancien ; cette situation est préférable pour des raisons de fiabilité. Dans la version "*Single-Packet*", un émetteur ne peut transmettre un nouveau jeton qu'après complète réception de son paquet. Si l'inverse se produit, un nouveau jeton sera généré à la fin de la transmission, avant récupération de l'ancien ; dans cette version "*Multiple Token*", deux jetons dont un seul se trouve à l'état libre, seront présents sur l'anneau.

La méthode du jeton peut fonctionner sur une base non-prioritaire ou à plusieurs niveaux de priorités pour s'accommoder à la transmission de données synchrones et asynchrones (pour plus détails, on consultera Stallings, 26).

V.8.B.2 Etude analytique du temps d'accès

Ce protocole est déterministe. Sous l'hypothèse que la longueur en bits est inférieure à la longueur d'un paquet, le cas le plus défavorable du temps d'accès se produit pour une station qui vient de passer le jeton à son voisin et qui, au même moment, a transféré un paquet de sa file auxiliaire à la file centrale. Le temps d'accès peut s'exprimer dans le cas où toutes les autres

stations ont un paquet à transmettre, par la relation :

$$Ta = Tw + ((N-1) * TO) \qquad [V.22]$$

avec TO :
Le time-out ou temps maximal de détention du jeton accordé à une station pour lui permettre la transmission de ses messages.

avec Tw :
Le "walk time" (voir point V.2).

V.8.B.3 Critique du protocole

Des collisions sont des événements rares sur le "Token Ring" ; elles peuvent cependant résulter de l'initialisation de l'anneau, de la perte du jeton ou de la génération de multiples jetons. Cette situation est du ressort du traitement des erreurs que, dès le début, nous avons écarté de notre étude.

Certains auteurs comme Ware Myers (Myers, 21) disent que la conception des protocoles "Token Ring" et "CSMA-CD" (voir point V.7.C.) implique beaucoup de facteurs non déterministes et que la probabilité d'accès au réseau endéans un certain temps sous protocole CSMA-CD peut aussi être calculée sous certaines circonstances. Selon eux, le concepteur ne peut que réduire, selon le type d'application envisagé, la probabilité pour une station de se voir refuser l'accès au canal et cette probabilité ne peut être de zéro.

V.8.C Le protocole déterministe "Adaptive Token Ring"

V.8.C.1 Présentation générale

On reproche au "*Token Ring*" d'être inefficace quand il y a beaucoup de stations et de paquets à transmettre par rafales car une caractéristique du trafic temps réel est sa perte de valeur avec l'âge. Or, dans la détermination du temps d'accès, le temps de détention du jeton attribué à une station est un facteur déterminant par son aspect cumulatif. Pour cette raison, Byung G. Kim a proposé le protocole "*Adaptive Token Ring*" comme variante du "*Token Ring*" (voir point V.8.B) qui ajuste dynamiquement le temps de détention du jeton par une station sur base du temps d'attente des paquets dans les files auxiliaires ou de l'estimation des activités des stations.

Une stratégie exhaustive c'est-à-dire où une station garde le jeton le temps nécessaire au vidage de sa file auxiliaire semble intéressante car la station possédant le jeton peut émettre ses messages sans risque de pertes. Cependant, cette stratégie n'est pas honnête car les files des autres stations vont s'engorger de sorte que le système entrera dans un cercle vicieux de longues périodes de détention du jeton suivies par la surcharge des files auxiliaires des autres stations.

L'"*Adaptive Token Ring*" propose deux modèles non-exhaustifs.

Dans le premier modèle, appelé "*inverse-adaptive*", la ième station détermine son prochain temps de détention du jeton H_i par la relation :

$$H_i = C - Ta_i \qquad [V.23]$$

avec C :
une constante indépendante de la station.

avec Ta_i :

le temps d'accès courant de la ième station c'est-à-dire que si le temps d'accès au canal du dernier message émis par la ième station est élevé, les autres stations ont donc beaucoup de messages à émettre et le temps de détention du jeton par la station i est ajusté en conséquence c'est-à-dire réduit. Si, par contre, le temps d'accès est faible, la charge du réseau est faible et le temps de détention du jeton peut être augmenté sans problème.

A la fin de la période de détention du jeton, la station rejette tous les paquets non encore transmis en attente dans sa file auxiliaire.

Dans le deuxième modèle, appelé "*delay-adaptive*", la station rejette les paquets dont le temps d'attente dépasse une valeur prédéterminée D sur base de l'observation que le contenu d'un message retardé exagérément perd sa valeur pour le récepteur. Ce modèle n'est valable que si le taux de rejet est minime. Si l'on prend pour exemple le cas d'une conversation, les messages doivent être envoyés endéans les 50-100 ms qui suivent leur génération et le pourcentage de perte pour être tolérable ne peut s'élever qu'à moins de 1 %.

V.8.C.2 Illustration

Des simulations ont montré que, même pour un anneau fortement utilisé, il est possible d'améliorer les performances du "*scan time*" (temps d'accès au réseau d'une station augmenté de son temps de transmission) avec un taux relativement bas de perte de paquets. Dans l'article de Kim (11), des réseaux en anneau basés sur l'"Adaptive Token Ring" sont simulés avec 8 stations, une distribution poissonnienne des arrivées de paquets de taux λ, des multiples du nombre de stations de l'anneau [8,16,24] pour valeur de C dans le modèle "*inverse-adaptive*", les valeurs [10, 20, 50,100, 200] pour le paramètre D du modèle "*delay-adaptive*" et 1/8 d'une tranche-canal pour le temps de propagation du jeton d'une station à une autre.

Si D vaut 100 ou 200, l'expérience montre qu'aucun des paquets n'est perdu et que le comportement du modèle "*delay-adaptive*" est

presqu'identique à celui du modèle exhaustif.

Rappelons d'abord la définition de l'intensité du trafic (soit ρ) : il s'agit du rapport entre le temps de service moyen par message et le temps moyen d'arrivées entre messages ; cette dernière mesure est un indicateur de congestion du réseau et exprime le temps de service total estimé par unité de temps ; elle peut être identique sur tout le réseau si tout message parcourt l'entièreté du réseau avant de revenir à son émetteur ou varier suivant la portion envisagée du réseau s'il s'agit d'un réseau en étoile où toute branche connaît un niveau de trafic différent.

La figure V.17. indique que lorsque l'intensite ρ du trafic est de 0.4, toutes les observations sont presqu'identiques. A intensité plus élevée, les différences entre les deux modèles pour différentes valeurs de paramètres deviennent plus prononcées. Pour ρ valant 0.96, le modèle "*inverse-adaptive*" présente un "scan time" deux fois moindre que la stratégie exhaustive au prix d'une perte de seulement 5,2% des paquets. Le compromis entre le "scan time" et la perte des messages est plus prononcé quand C = 8 avec une perte de 17,5%.

V.8.D Le protocole déterministe "Token Passing Multipacket Ring"

V.8.D.1 Présentation générale

Il est possible d'améliorer la composante Tw du temps d'accès du "*Token Ring*" (voir point V.8.B) ainsi que son débit réel en recourant à une de ses variantes, le protocole "*Token Passing Multipacket Ring*". Le Tw peut être réduit car durant un parcours le long de l'anneau du jeton dont le dernier bit est à zéro, le jeton peut être utilisé plusieurs fois par différentes stations qui n'ont pas à attendre le retour d'un jeton libre pour pouvoir émettre.

Pour cela, il faut recourir à un jeton spécial renseignant sur les

		Temps d'accès	% de pertes
Stratégie exhaustive		25.0	0.0
Stratégie Inverse-adaptive	C = 8	4.7	17.5
	16	7.7	8.7
	24	10.3	5.2
Stratégie Delay-adaptive	D = 10	7.3	9.5
	20	11.2	4.4
	50	17.9	0.9

Statistiques exprimées en tranches-canal pour une intensité du trafic de 0.96

		Temps d'accès	% de pertes
Stratégie exhaustive		1.7	0.0
Stratégie Inverse-adaptive	C = 8	1.7	0.3
	16	1.7	0.0
	24	1.7	0.0
Stratégie Delay-adaptive	D = 10	1.7	0.0
	20	1.7	0.0
	50	1.7	0.0

Statistiques exprimées en tranches-canal pour une intensité du trafic de 0.4

Figure V.17. : Simulation du protocole "ADAPTIVE TOKEN RING"

portions disponibles de l'anneau : un champ (champ numéro un) renseigne sur l'adresse de la station émettrice détenteur du jeton et un autre champ (champ numéro deux) sur l'adresse du destinataire du paquet. Supposons, sur base de la figure V.18., que la station K doive transmettre un paquet de données à la station K + N. Dès qu'elle reçoit le jeton, elle indique son adresse dans le champ numéro un et celle du destinataire dans le champ numéro deux avant de transmettre son paquet. Quand la station K+N reçoit le paquet, elle reconnaît son adresse, remet le champ numéro deux à zéro avant de remettre le jeton sur l'anneau et retire son paquet. Une station P située entre K+N et K repère, sur base du champ numéro un, la fin de la portion disponible de l'anneau et si elle doit transmettre à P+M (P+M < K), elle indique l'adresse du destinataire des données dans le champ numéro deux. Dès que le jeton atteint la station K, il est arrêté et cette station, si elle a terminé sa transmission de données, remet un jeton libre sur l'anneau.

Plusieurs paquets peuvent circuler sur l'anneau mais chaque paquet occupe seulement la portion de l'anneau située entre l'émetteur et le récepteur, ce qui explique que l'intensité du trafic ρ (voir point V.8.C.3) varie sur l'anneau (pour plus de renseignements, on consultera Mirabella et Salvo, 20).

V.8.D.2 Illustration du protocole

L'article propose une simulation de ce protocole en se basant sur un modèle de réseau de Pétri. Il a montré un très bon comportement sous forte charge. Seulement, dans certaines situations, certaines stations ont souffert de temps d'accès plus longs car l'acquisition du jeton n'est plus strictement séquentielle. Dans ce cas, un mécanisme de priorités peut être habituellement introduit pour résoudre ce problème.

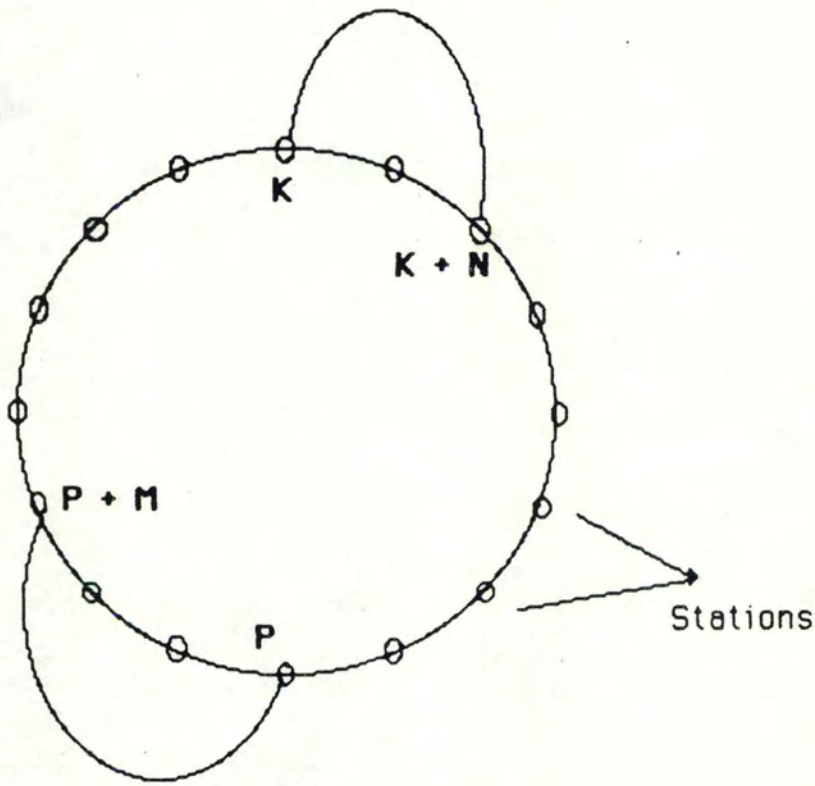


Figure V.18. : Fonctionnement du protocole "TOKEN PASSING MULTIPACKET RING"

V.8.E. Le protocole déterministe "Slotted Ring"

V.8.E.1 Présentation générale

Le protocole "*Slotted Ring*" a pour ancêtre le "*Pierce Loop*" (Mahbod et Howard, 18) conseillé en 1971 par les laboratoires Bell.

Dans ce protocole, l'anneau est subdivisé en un nombre de trains de bits de taille fixe. A moins que la distance couverte par l'anneau soit importante ou qu'il y ait beaucoup de stations (le répéteur par lequel chaque station est connectée à l'anneau, introduit un retard d'un ou deux bit(s) pour lui permettre de décoder l'adresse d'un train par exemple...), il n'y a généralement pas suffisamment de place sur l'anneau pour contenir plusieurs trains ; une première solution est l'introduction artificielle de bits de retard par la station-moniteur ; celle-ci ajuste leur totalité si un brusque changement se produit dans la configuration par la panne, par exemple, d'un répéteur ; une deuxième solution est la mise en oeuvre de registres à décalage présents dans les répéteurs qui introduisent alors des retards de plusieurs bits.

Le premier bit de chaque train indique s'il est vide ou occupé. Quand une station veut émettre, elle attend le passage d'un train vide avant de le marquer occupé et de le remplir de données façonnées à sa taille contrairement au "Token Ring" (voir point V.8.B) dont les paquets sont de taille arbitrairement large. Si le message est plus petit que le train, ce dernier n'est pas utilisé efficacement. Si, par contre, il est plus long, il doit être segmenté en différents paquets transmis individuellement. Les trains de bits seront souvent de petite taille (avec le risque que le % de bytes de contrôle soit élevé par rapport aux bytes utiles) bien que des trains plus conséquents réduiraient la fragmentation d'un message en paquets, améliorant de ce fait le temps d'accès au canal d'un message.

Ce protocole permet à plus d'une station de transmettre simultanément des paquets. Chaque répéteur reconnaîtra son paquet en utilisant un compteur initialisé par le moniteur et après qu'autant de trains soient passés, le

répéteur saura qu'il s'agit du sien et automatiquement modifiera le 1ier bit du train.

V.8.E.2 Illustration du protocole

Cependant, pour des considérations pratiques et d'efficacité, la plupart de ces protocoles ne sont prévus qu'avec un seul train (cas du Cambridge Ring commercialisé par la firme Logica) car la simplicité accroît la fiabilité.

Généralement, ces protocoles, comme dans le cas du Cambridge Ring, présentent un temps d'accès déterministe car ils ne permettent pas à une station de réutiliser le train de bits qu'elle vient de libérer. Le Pierce Loop, quant à lui, n'imposait pas cette interdiction et n'était donc pas déterministe.

V.8.E.3 Etude analytique du temps d'accès

Pour modéliser un "*Slotted Ring*" à un seul train de bits, empruntons une idée à Werner Bux (Bux, 4). Quand une station récupère son train de bits, elle en modifie le premier bit avant de le passer à la station suivante et éviter ainsi de le monopoliser. C'est le cas dans la figure V.19. où les stations 1 et 2 ont un message à transmettre et le découpent en petits morceaux. Si nous représentons différemment la situation et que nous mettons l'un à côté de l'autre les différents temps de propagation du train à l'état vide d'une station à une autre, nous pouvons montrer que le train de bits effectue un tour complet de l'anneau dans l'état vide. Un "*slotted ring*" est donc constitué de N stations et d'une station fictive qui génère des paquets vides.

Si l'on fait l'hypothèse que le "*walk time*" est égal à la durée de service d'un train de bits, le temps d'accès d'une station qui vient de faire circuler le train vide et dont un paquet est introduit dans la file centrale peut s'exprimer par la relation :

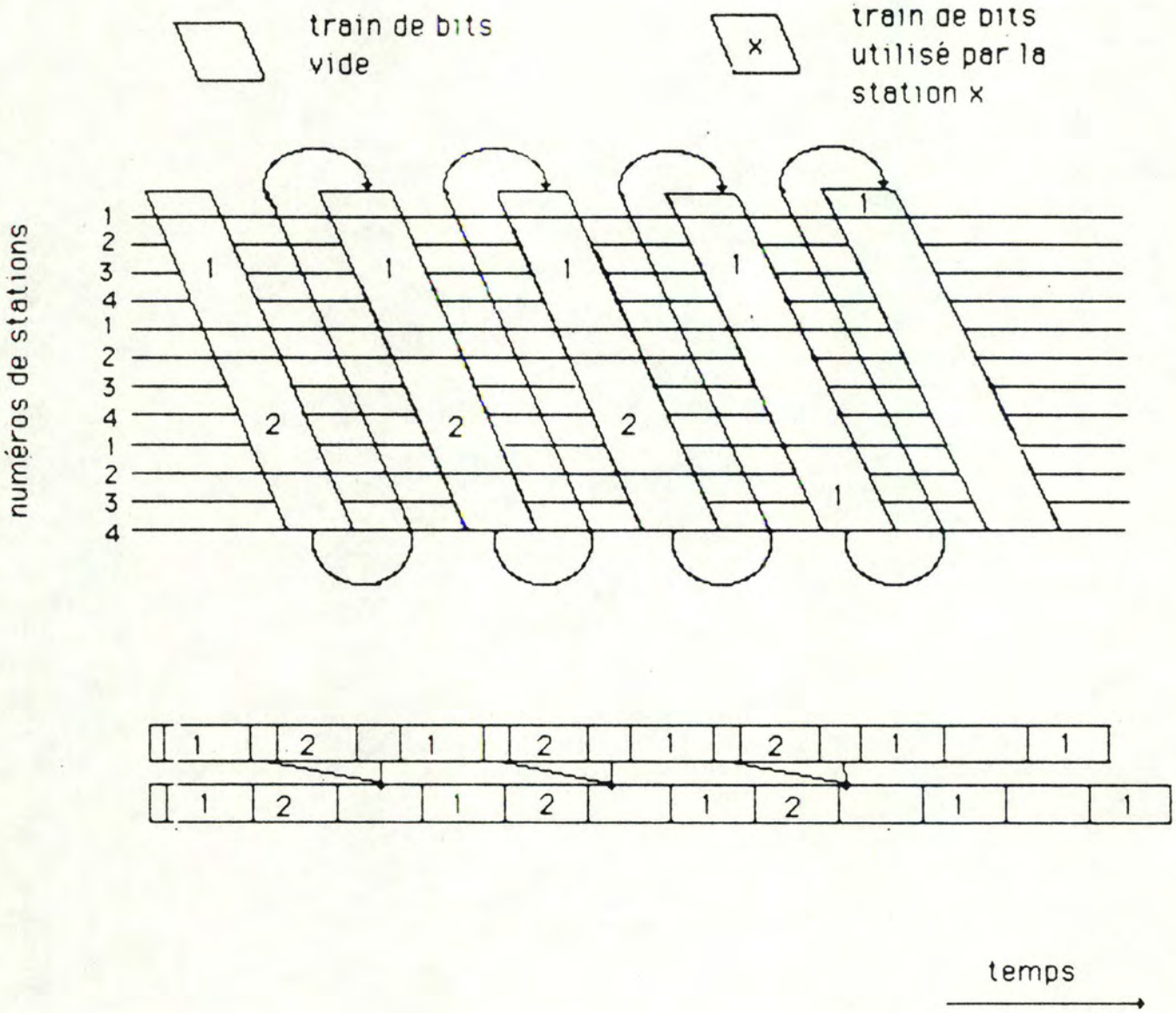


Figure V 19 Modélisation du protocole "SLOTTED RING"

$$Ta = (N-1)*Tc + Tc \qquad [V.24]$$

avec Tc :
le temps de service constant d'un paquet de la station fictive. En effet, les paquets ont une taille de 38 bits sur le "Cambrigde Ring".

avec (N-1)*Tc :
le temps de service constant des N-1 autres stations ayant un paquet à émettre.

Werner Bux (Bux, 4) développe un modèle pour calculer le temps de réponse moyen d'un paquet depuis sa génération à la source jusqu'à sa réception à la destination en tenant compte du temps d'attente, du temps d'accès au canal de l'émetteur, du temps de transmission d'un paquet et du délai de propagation.

V.8.F. Le protocole probabiliste "Register Insertion Ring"

V.8.F.1 Présentation générale

Le protocole "Register Insertion Ring" est le résultat des travaux de LIU sur les "Distributed Loop Computer Networks" (DCLN) (Babic, Liu et Pardo, 2) à l'université de l'état de l'Ohio.

Sur la figure V.20. présentant l'interface d'une station de l'anneau, nous constatons qu'elle contient deux registres, un registre à décalage pour contenir l'information entrante et un tampon de sortie pour stocker les données à émettre. Quand une station a un paquet à émettre, de longueur inférieure ou égale à la taille du tampon de sortie, elle le charge dans son tampon. Initialement, le pointeur du registre de décalage pointe vers le bit de position extrême droite signifiant que toutes les autres positions sont

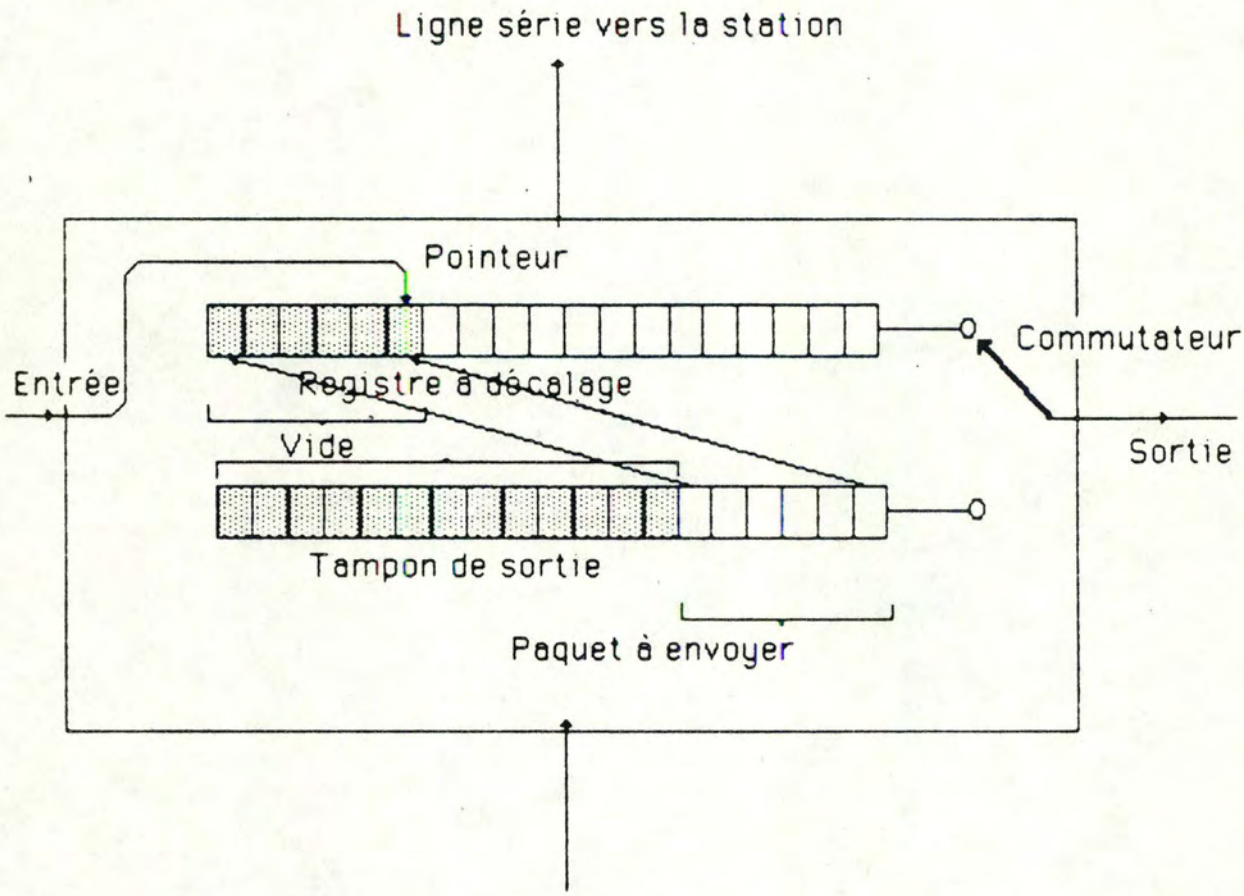


Figure V.20. : Interface d'une station du protocole "REGISTER INSERTION RING"

vides. Quand un bit arrive de l'anneau, il est mis à la position pointée par le pointeur et ce dernier est déplacé d'une position vers la gauche. Dès que le champ d'adresse du paquet est arrivé, l'interface peut déterminer si le paquet qui va suivre lui est destiné. Si oui, le paquet est retiré bit par bit de l'anneau avant de repositionner le pointeur à l'extrême droite. Si non, au fur et à mesure qu'un nouveau bit arrive du réseau, le contenu entier du registre à décalage est déplacé à droite d'une position, mettant le bit le plus à droite sur l'anneau sans bouger le pointeur. Si aucun bit ne provient de l'anneau, dû à un trou entre paquets, le contenu de registre à décalage est réduit d'un bit et le pointeur est déplacé d'une position vers la droite.

Quand le registre à décalage a mis le dernier bit sur l'anneau, il regarde si la station doit émettre un paquet et s'il a accumulé autant de positions vides dans le registre à décalage que la longueur du paquet à émettre. Si ces conditions sont remplies, le commutateur de sortie est actionné et le tampon de sortie est mis sur l'anneau, un bit à la fois, en synchronisation avec le registre à décalage qui fait fonction d'extension du réseau. Dès que la sortie du message est terminée, le commutateur est de nouveau activé et le registre à décalage vidé... .

V.8.F.2 Etude intuitive du temps d'accès

Ce protocole présente d'excellentes performances sous charge faible et modérée. Il permet une allocation équitable de la largeur de bande du canal aux stations et permet la transmission simultanée de paquets de longueur variable par plus d'une station. Sous faible charge, le temps d'accès au canal est faible. Cependant, sous forte charge, il n'est pas déterministe car la probabilité, pour une station émettrice, de trouver suffisamment de positions libres dans son registre à décalage est faible étant donné que l'émission à volonté possible de toute station peut amener à une situation de congestion. Si le paquet accède au canal, il peut encore rencontrer sur son chemin des registres à décalage remplis et immobilisés par l'émission d'un paquet contenu dans le tampon de sortie des stations traversées et subir un temps de propagation détestable l'amenant à sa station destinatrice au-delà des temps demandés. Cependant, le calcul du temps de réponse n'est pas notre propos et nous arrêterons là la discussion.

Afin d'étudier le temps de réponse moyen rencontré dans un DCLN (généralisation du "Register Insertion Ring") sur boucle symétrique et asymétrique avec tampons de sortie infinis, Liu dans son article (2) développe un modèle de simple serveur M/M/1 pour exprimer un réseau général d'ordinateurs. La boucle est vue comme une juxtaposition d'hôtes en intéraction avec les autres hôtes au travers des flux de paquets A_i (demandes à distance devant être satisfaites par l'hôte i) et B_i (les réponses destinées à l'hôte i de ses demandes à distance) et en intéraction avec le réseau de communication via le flux C_i des paquets (réponses aux demandes présentes dans le flux A_i et demandes addressées par le l'hôte i aux autres hôtes).

Il est bon d'ajouter qu'en moyenne, le temps de réponse de ce protocole qui permet la transmission simultanée de paquets de longueur variable est meilleur que celui fourni par le "Slotted Ring" (voir point V.8.E) qui admet plusieurs paquets, lui-même meilleur que le "Token Ring" (voir point V.8.B) qui ne permet la transmission que d'une seule station à la fois.

V.8.F.3 Illustration

Il a été utilisé dans la gamme des produits de la Série 1 d'IBM et dans un produit suisse appelé SILK.

V.8.G. Le protocole déterministe "CSMA with Collision Avoidance"

Nous abordons maintenant l'ensemble des protocoles à écoute du canal pour y éviter les collisions.

V.8.G.1. Présentation générale

Sous ce point, nous effectuerons une légère diversion en présentant un protocole très rencontré sur les réseaux locaux HSLN (voir point IV.1). Le protocole *"CSMA with Collision Avoidance"* ou *"Prioritized CSMA"* est décrit par le standard ANSI X3T9.5 pour une topologie de bus à large bande avec simple canal (*Single-Channel broadband*) et est illustré par Hyperchannel pour une topologie de bus en bande de base.

Ce protocole est basé sur le protocole à l'écoute du canal (voir point V.2) c'est-à-dire qu'une station désireuse de transmettre écoute le canal et diffère sa transmission si ce dernier est occupé. En plus, un algorithme détaillé plus loin est utilisé pour éviter les collisions lorque le canal est trouvé libre par plusieurs stations. Pour ce faire, les stations sont réparties en une séquence ordonnée qui ne doit pas nécessairement correspondre à leurs positions physiques sur le bus.

Dans la version initiale, après transmission par une station quelle qu'elle soit, l'initiative est redonnée à la station numéro 1. Si elle n'a aucun paquet à émettre après un certain temps appelé "Arbitrated Access Opportunity" qui correspond à la durée de transmission de 16 bits, la chance est laissée à la station numéro deux et ainsi de suite... En bref, une station de numéro i + 1 ne peut émettre que si aucune station précédente n'avait de message à transmettre.

Une première amélioration au schéma de base est d'admettre le dialogue multi-paquets c'est-à-dire qu'après une transmission quelle qu'elle soit, la station destinataire peut prendre la parole et si elle n'a aucun paquet à transmettre, le tour est rendu à la station numéro 1.

Une deuxième amélioration traite du cas où personne n'a de paquet à transmettre. La solution proposée par l'ANSI est très élégante. Si, après son "Arbitrated Access Opportunity", la station de numéro N n'a rien transmis, elle émet un paquet bidon après lequel, si l'on se réfère à la version initiale, l'initiative est donnée à la station numéro 1 et le processus est réinitialisé.

On aboutit ainsi au schéma suivant :

1. Le moyen de communication est en activité
2. Le moyen de communication devient libre
   alors si le destinataire transmet alors se rendre en 1
   sinon se rendre en 3
3. Si la station numéro un transmet alors se rendre en 1
   sinon se rendre en 4

i+2. Si la station numéro i transmet alors se rendre en 1
   sinon se rendre en i+3

N+3. Si aucune station ne transmet, retour au point 1.

La norme ANSI propose une troisième amélioration afin de rendre le schéma loyal. Il s'agirait d'une part d'éviter que les stations numérotées par des chiffres bas aient toujours davantage de chance d'émettre et d'autre part de faire en sorte qu'une station qui vient d'émettre cède la place jusqu'à ce que toutes les autres stations aient eu leur chance. Hyperchannel n'inclut pas cette amélioration.

V.8.G.2 Etude intuitive du temps d'accès

Selon William Stallings, un réseau HSLN n'est constitué que d'un petit nombre de stations (une dizaine) et comme les vitesses atteintes sur son support sont très élevées, le protocole ne provoque pas de retards illimités.

Selon nous, le protocole n'est déterministe que par l'introduction par l'ANSI de la troisième amélioration. Il nous semble douteux qu'Hyperchannel présente un temps d'accès déterministe car la version du "*CSMA-CA*" qu'il utilise avantage les stations de faible numéro.

V.8.H. Le protocole déterministe "Bit-Map"

V.8.H.1 Présentation générale

Le protocole "*Bit-Map*" est une légère variante d'un protocole datant de 1980, dû à Kleinrock et Scholl (Kleinrock et Scholl, 12) et non présenté pour des raisons de similitude avec le protocole "*Bit-Map*".

Les états du moyen de transmission alternent entre période de réservation subdivisée en champs équivalant à un intervalle de propagation de Θ sec et période de transmission de paquets (voir figure V.21.).

Les stations disposent d'une adresse entre 1 et N. Pour indiquer aux autres stations qu'elle va transmettre un paquet au cours de la période de transmission suivante, chaque station, dans l'ordre croissant des adresses, transmet un "1" durant son champ de réservation. Une fois le processus de réservation terminé, chaque station connaît les identités de toutes les stations prêtes à émettre avant la reprise d'un nouveau cycle réservation/transmission.

V.8.H.2 Etude analytique du temps d'accès

Ce protocole est déterministe. Le cas le plus défavorable se produit pour la station d'adresse N quand son champ de réservation vient de passer et qu'elle a, à ce moment, un paquet présent dans la file centrale. Si chaque station d'adresse inférieure à N a un paquet présent dans la file centrale, le temps d'accès peut être donné par la relation suivante :

$$Ta = ((N - 1)*Ts) + N * \Theta + ((N - 1)*Ts) \qquad [V.25]$$

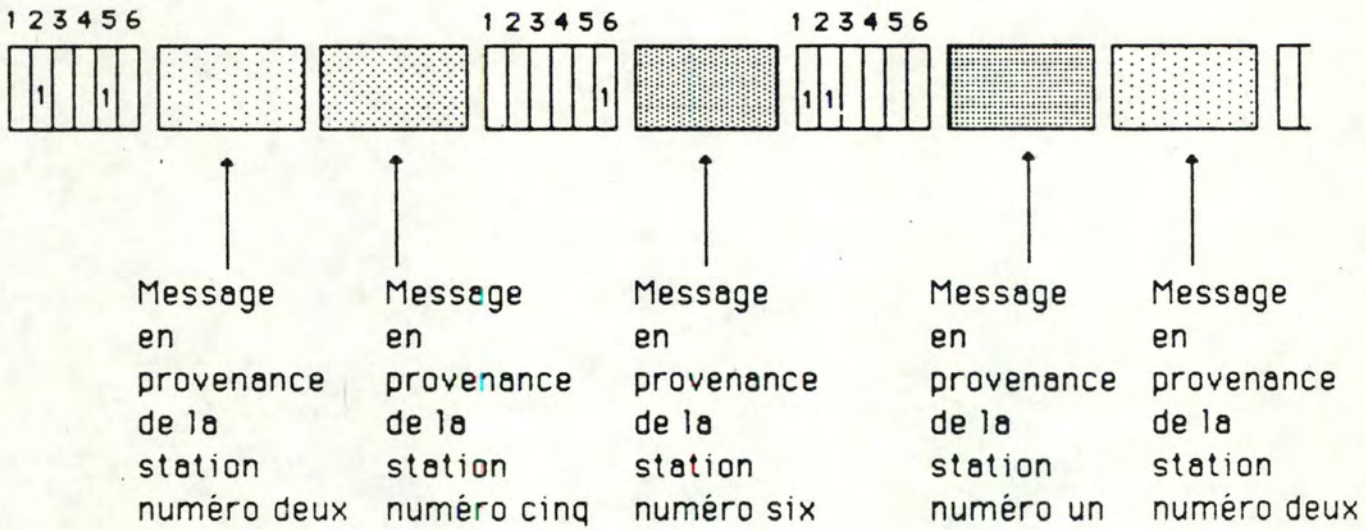


Figure V.21. : Illustration du protocole "BIT-MAP"

avec (N-1)*Ts :

le temps de service des paquets de taille variable des N-1 stations suivant la période de réservation où la station N reçoit son paquet ainsi que la période de réservation où la station N a positionné son bit.

avec N * Θ :

la durée de la période de réservation où chaque station positionne son bit.

Nous n'avons pas tenu compte ci-dessus de l'argument présenté par certains auteurs comme Tanenbaum (Tanenbaum, 29) selon lequel les stations à adresse élevée ont un temps d'accès au canal moins important que celles à adresse moindre parce que, typiquement, une station devient prête quand la période de réservation s'est écoulée à moitié. Dans ce cas, les stations à adresse plus élevée ont encore le temps de positionner leur bit et de pouvoir émettre avant le début de la prochaine période de réservation.

Pour pallier à cet inconvénient ainsi qu'au fait qu'à faible charge, une station doive attendre la fin de la période de réservation avant d'émettre, Tanenbaum dit que le protocole suivant a été proposé.

V.8.1. Le protocole déterministe "Broadcast Recognition with Alternating Priorities"

V.8.1.1 Présentation générale

Le protocole "*Broadcast Recognition with Alternating Priorities*" se présente sous le nom de "*Broadcast Recognition Access Method*" dans la version de Chlamtac ou sous celui de "*Mini Slotted Alternating Priorities*" dans la version de Scholl.

L'ordre dans lequel les stations acquièrent le droit de transmettre est

déterminé par l'ordre numérique de leurs adresses. La parole est donc donnée initialement à la station numéro un ; si elle a un paquet, elle transmet ; sinon, elle reste silencieuse. La présence ou l'absence d'une transmission après un intervalle de propagation de Θ sec renseigne les autres stations sur l'intention de la station numéro un d'émettre un paquet. Si cette station ne réagit pas, les droits passent implicitement à la station suivante dans la séquence logique c'est-à-dire la station numéro deux, qui réitère le même processus (voir figure V.22.) tandis que les autres stations restent à l'écoute de ce qui se passe. Si, par contre, la station numéro un désire émettre les droits passent à "deux" dès que "un" a terminé sa transmission.

V.8.H.2. Etude analytique du temps d'accès

Ce protocole est déterministe. Le temps d'accès au canal dans le cas le plus défavorable où la station d'adresse N veut émettre après la transmission d'un paquet par chacune des N-1 premières stations est donné par la relation :

$$Ta = (N - 1) * Ts \qquad [V.26]$$

V.8.H.3. Illustration du protocole

Dans Express-net et Fasnet d'ATT (voir figure V.22.), l'emploi du même protocole entraîne l'attente de moins de Θ sec. Chaque station dispose d'une sortie et d'une entrée sur un canal unidirectionnel plié en trois parties. L'idée est que la station, en tête du canal, émet un train de paquets qui se propage le long du bus. Une station lit les paquets du train quand ils atteignent son entrée mais ce n'est que lorsque sa sortie détecte la fin du train et qu'elle a un paquet à émettre, qu'elle ajoute son paquet au train. L'intervalle entre paquets sur le train est le temps nécessaire pour une station de détecter la fin du train passant et de commencer l'émission de son paquet, ce qui est moins de Θ sec...

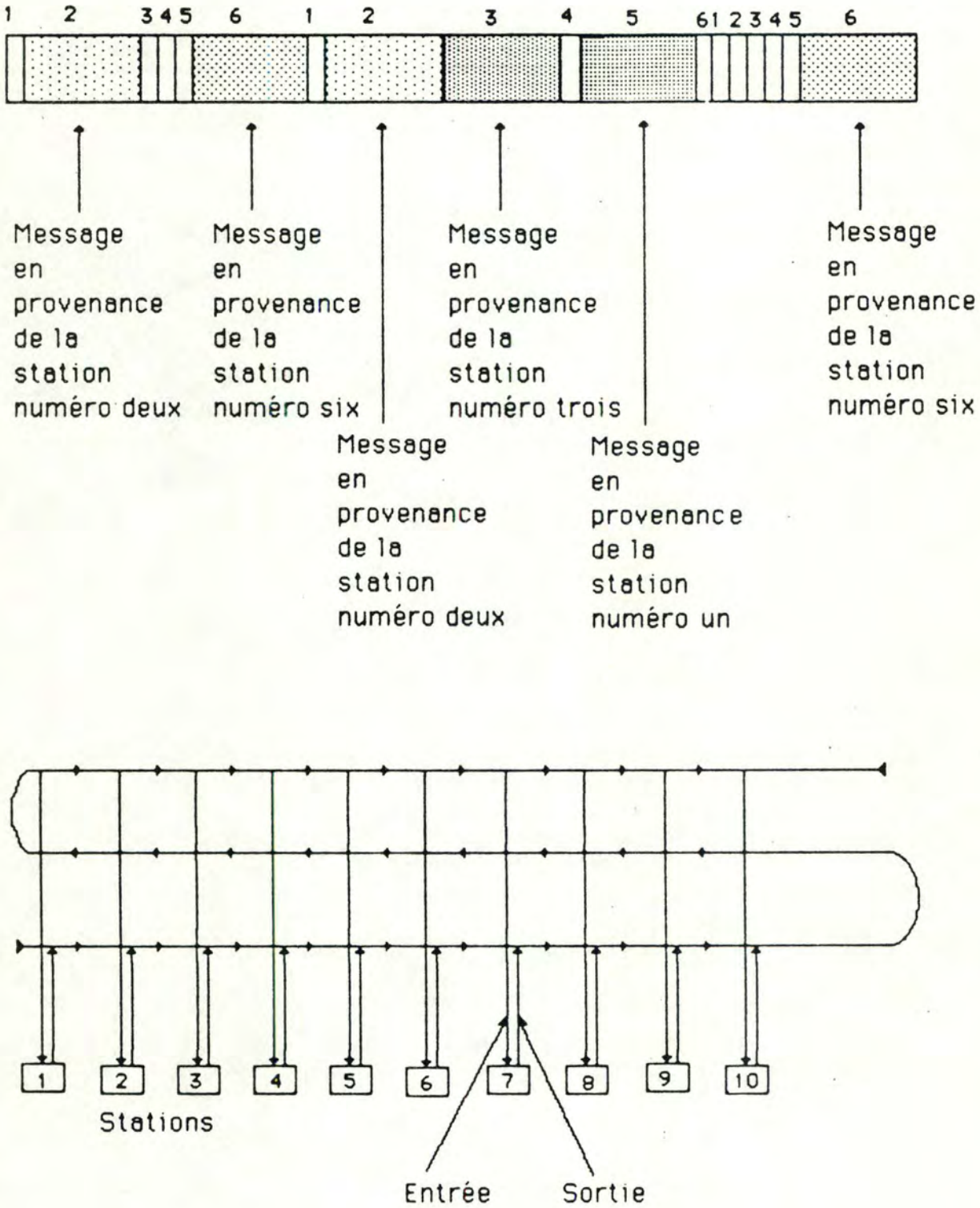


Figure V.22. : Le protocole "MSAP/BRAM" et son illustration (EXPRESS-NET)

V.9. Etude de la feuille numéro 6 de l'arbre binaire : multiplexage de la voie de communication dans le temps, selon un mode de division asynchrone du temps, selon un algorithme de résolution d'accès au canal distribué et évitant les collisions et selon un ordre d'accès déterminé statiquement.

Cette feuille regroupe les protocoles recourant à la technique du multiplexage de la voie de communication dans le temps, selon un mode de division asynchrone du temps et dont la politique d'accès au canal est réglementée : l'algorithme d'accès est distribué et évite les collisions et l'ordre d'accès au canal est déterminé statiquement.

Nous aborderons dans ce point les protocoles qui relèvent de la classe des protocoles à l'écoute d'une transmission sur le canal pour y éviter des collisions (le "CITO" (V.9.A), le "MLMA" (V.9.B) et le "Binary Countdown" (V.9.C)).

V.9.A. Le protocole déterministe "Content Induced Transaction Overlap"

V.9.A.1 Présentation générale

Le protocole "*Content Induced Transaction Overlap*" (CITO) met sur pied des règles de comportement coopératif entre les émetteurs basé sur le contenu des paquets qui sont des mots de r bits et évite les retransmissions. Les transmissions sont effectuées par groupe c'est-à-dire un ensemble de mots transmis par toutes les stations émettrices.

Chaque appareil-émetteur dispose des registres de base suivants pour organiser sa transmission :

- un registre de mot de r bits contenant l'information à transmettre ;
- un registre de position de bit de $\log_2 r$ bits pointant vers le bit en cours de transmission ;
- un registre de limite de mots de $\log_2 r$ bits pour la synchronisation du canal.

Tout message CITO est constitué de deux informations : la position du bit nécessaire pour la reconstruction du mot par le récepteur et des fragments de bits. Nous supposons, de plus, que le mot de données renseigne sur l'identité de l'émetteur et du récepteur. L'opération de base consiste en l'envoi en concurrence des bits de données en commençant par le bit de poids le plus fort. Comme toutes les stations transmettent simultanément, trois situations peuvent se produire à l'envoi de chaque bit :

- signal '0' sur le canal si toutes transmettent des bits '0' ;
- signal '1' sur le canal si toutes transmettent des bits '1' ;

  En même temps qu'elles transmettent, les stations en comparant la valeur sur le canal (bit canal) avec leur propre émission se rendent compte que tout le monde est d'accord et transmettent le prochain bit en décrémentant leurs registres de position de bit et de limite de mot ;

- signal 'X' sur le canal si certaines transmettent des bits '1' et d'autres des bits '0' mais qui est interprété comme valant '0' pour le récepteur; celui qui transmettait un '1' s'arrête d'émettre, décrémente son registre de limite de mot et ce tant qu'un bit est transmis ; par contre, celui qui transmettait un '0' continue...

Cette procédure sera répétée jusqu'à ce qu'un appareil détecte l'envoi de ses r bits de données et le passage à zéro de son registre de limite de mot (comme c'est le cas pour toutes les stations). La transmission d'un mot est alors achevée et l'on passe en phase de "bit competition". Dans cette nouvelle phase, toutes les stations ayant encore à transmettre la fin de leur mot de données, envoient les bits de leur registre de position de bit en commençant par celui de poids le plus fort selon le même protocole que pour la transmission de données et font passer le bit qu'elles entendent sur le canal dans leur registre de limite de mot. Ce manège se poursuit jusqu'à ce que les $\log_2 r$ bits du registre de position de bit de la station ayant le plus petit

nombre de bits restant à transmettre aient été envoyés et passés dans le registre de limite de mot de toutes les autres stations.

Celui dont les registres de position de bit et de limite de mot sont identiques commence à transmettre le prochain bit de son registre des données. La phase de transmission recommence comme auparavant, se termine de nouveau quand le registre de limite de mot a atteint zéro et se poursuit par l'étape de "bit competition". La succession de transmission de données et de "bit competition" se termine quand tous les appareils ont transmis leur mot de données et que leur registre de position de bit vaut zéro (pour plus de renseignement, on consultera Berkovitch, Wilson et Walter, 3).

V.9.2. Etude intuitive du temps d'accès

Le concept de groupe de transmission rend ce protocole déterministe. Toute station essayant de transmettre se trouve soit dans le groupe de transmission courant, soit dans le suivant. Le temps d'accès le plus défavorable revient donc à la transmission d'un groupe complet.

V.8.A.3 Critique du protocole

Vu que CITO envoie les données en concurrence, un effet de compression se produira quand des données similaires seront transmises. Si le débit réel est élevé, la probabilité de données semblables est accrue et montre que CITO est plus efficace dans ce cas.

Dans les applications temps réel, CITO permet l'interruption d'un groupe de transmission pour expédier des messages d'alarme. Supposons un émetteur de données urgentes, il attendra la prochaine phase de "bit competition" et interviendra avec un registre de position de bit de valeur zéro pour remporter l'accès. Le récepteur courant et les autres émetteurs reconnaîtront cela comme un cas spécial et attendront que l'émetteur ait transmis ses r bits.

Quand la transmission sera terminée, la "bit competition" reprendra et la transmission interrompue continuera à partir de l'endroit où elle s'était arrêtée.

V.9.B. Le protocole déterministe "Multi-Level Multi-Access"

V.9.B.1 Présentation générale

Défini par Rothauser et Wild, le protocole "*Multi-Level Multi-Access*" propose un accès ordonné au bus.

Un contrôleur signale à intervalle de temps approprié le début d'un intervalle de réservation-transmission qui est subdivisé en deux parties, une pour la réservation et une pour la transmission d'un nombre arbitraire de paquets. Au terme du cycle de réservation où toutes les stations désireuses d'émettre ont diffusé leur adresse dans un format particulier, toutes savent qui utilisera l'intervalle suivant et la séquence de transmission est donnée selon une assignation de priorités à toutes les stations. Bien qu'ils soient aussi efficaces que les protocoles à réservation statique sous forte charge, les protocoles à réservation dynamique comme le "BRAP" (voir point V.8.G) imposent, sous faible charge, à une station d'attendre un certain nombre de Θ sec avant de pouvoir émettre. Dans ce cadre-ci, seules les stations qui désirent émettre diffusent leur adresse.

Supposons que les stations soient au nombre de 1000 et que nous travaillons en modulo 10. Chaque adresse dans le système consiste en trois chiffres décimaux, chacun représenté par un groupe de 10 bits ou décade. Ainsi, 472 correspond au positionnement du bit 4 de la 1ère décade, du bit 7 de la 2ième décade et du bit 2 de la dernière.

Si une station veut émettre, elle utilise une entête de 30 bits dans la partie réservation pour s'annoncer avant d'envoyer son paquet. Le problème se pose quand plusieurs stations veulent insérer leur adresse dans la même

entête.

Voici l'algorithme utilisé pour le départage des stations :

La première décade correspond à la position des centaines dans le numéro des stations. Après la première décade, toutes les stations qui n'ont pas positionné un bit doivent rester silencieuses jusqu'à ce que toutes les autres stations aient transmis leurs données.

Tant que les différents chiffres positionnés dans la décade des centaines n'ont pas été épuisés :

- prendre, parmi les chiffres non encore retenus, le plus élevé (soit x) ;
- dans la décade suivante, toutes les stations d'adresse commençant par x annoncent le chiffre des dizaines :

  - Tant que les différents chiffres positionnés dans la décade des dizaines n'ont pas été épuisés :

    - prendre, parmi les chiffres non encore retenus, le plus élevé (soit y) ;
    - toutes les stations dont l'adresse commence par xy positionne dans la décade suivante le bit correspondant aux unités ; il y en a au plus 10.

V.9.B.2. Etude analytique du temps d'accès

Il faudrait que le choix du modulo (soit r) et du nombre de niveaux (soit l) minimise le nombre de bits de l'entête ou la partie réservation. En sachant que le nombre de niveaux pour un r donné est la plus petite valeur de l telle que $r^l \geq N$, 1000 stations impliquent le choix de 2 pour r et de 10 pour l.

Intuitivement, si N vaut 1000, r 10 et l 3, le cas le plus défavorable se produit pour la station 1000 quand la première décade vient de passer sans qu'elle y ait positionné son bit et qu'elle vient de recevoir un paquet à émettre et lorsque toutes les stations veulent émettre. Elle doit donc

attendre la fin d'un intervalle réservation-transmission avant de pouvoir positionner son bit dans une décade de l'intervalle de réservation suivant. Le temps d'accès déterministe s'exprime alors par la relation :

$$Ta = 1110 * \Theta + ((N - 1)*Ts) + 1110 * \Theta + ((N - 1)*Ts) \quad [V.27]$$

avec 1110 :
une décade pour les centaines, 10 pour les dizaines et 100 pour les unités reviennent à un total de 1110 bits ;

avec Θ :
l'intervalle de propagation pour que toutes les stations prennent connaissance de la valeur du bit courant ;

avec ((N - 1)*Ts) :
le temps de service des paquets de taille variable des N - 1 autres stations c'est-à-dire des 999 autres.

Werner Bux a développé un modèle à serveur unique (Bux, 4) pour formaliser le "MLMA". Il a étudié le temps de réponse moyen sans présenter la variance.

V.9.C. Le protocole déterministe "Binary Countdown"

V.9.C.1 Présentation générale

Le protocole "*Binary Countdown*" est une légère variante du "*MLMA*" (voir point V.9.C) où au lieu d'avoir 10 bits à chaque niveau, on en a un seul et où le nombre de niveaux nécessaires pour départager les stations est de $\log_2 N$.

Chaque station effectue un 'ou' exclusif des signaux présents sur le canal. En effet, pour signaler qu'elle veut émettre, une station écrit son

adresse dans l'entête comme un nombre binaire et dès qu'une station s'aperçoit qu'un bit positionné à 0 de son adresse a été surécrit par un 1, elle abandonne. Quand la station gagnante a transmis son adresse et son paquet, il n'y a plus d'information disponible sur le nombre de stations qui veulent émettre de sorte que l'algorithme recommence pour l'émission d'un nouveau paquet.

Mok et Ward ont proposé une variation où les stations ont des numéros virtuels et où les numéros de 0 à celui incluant la station fructueuse sont augmentés de un après chaque transmission pour donner la priorité aux stations qui sont restées silencieuses pendant longtemps. Si les stations A, B, C, D, E, F, G et H ont les numéros 4, 2, 7, 5, 1, 0, 3 et 6, une transmission fructueuse de la station D assignera aux 8 stations les numéros 5, 3, 7, 0, 2, 1, 4 et 6. La station D ne pourra acquérir le canal que si aucune autre ne le veut.

V.9.C.2. Etude analytique du temps d'accès

Le temps d'accès au canal dans le cas le plus défavorable se présente quand celui qui vient d'émettre a un nouveau paquet prêt et que toutes les autres stations ont un message à émettre. Il peut s'exprimer par la relation suivante :

$$Ta = Log_2N * \Theta * N + ((N - 1) * Ts) \qquad [V.28]$$

avec Θ :
l'intervalle de propagation ;

avec log_2N :
le délai de propagation de l'entête ou de l'intervalle de réservation ;

avec $((N - 1) * Ts)$:
le temps de service des paquets de taille variable des N - 1 autres stations.

V.9.C.3. Illustration du protocole

Le Datakit d'ATT propose une solution de pipeline sur les bus courts en décomposant le problème en deux bus, l'un pour réaliser l'arbitrage par la méthode du "Binary Countdown" sur base d'une entête de 8 bits durant 80 µsec et l'autre pour le transfert des paquets d'une longueur moyenne de 80 à 1600 bits qui peuvent se succéder au rythme d'un paquet toutes les 80 µsec c'est-à-dire une largeur de bande de l'ordre de 1 à 20 Mbps.

V.10. Etude de la feuille numéro 7 de l'arbre binaire : multiplexage de la voie de communication dans le temps, selon un mode de division asynchrone du temps et selon un algorithme de résolution d'accès au canal centralisé et évitant les collisions

Cette feuille regroupe les protocoles recourant à la technique du multiplexage de la voie de communication dans le temps, selon un mode de division asynchrone du temps et dont la politique d'accès au canal est réglementée : l'algorithme d'accès est centralisé et évite les collisions.

V.10.A Le protocole déterministe "polling"

V.10.A.1 Présentation générale

Le plus rencontré dans les structures centralisées, le protocole "*polling*" constitue le schéma d'accès historique sur la topologie en bus ou en arbre.

Il peut se manifester selon deux disciplines :

- la discipline du "roll-call" ou "polling" sur ligne multipoints ;
- la discipline "hub" ou "polling" à contrôle distribué, courante sur les topologies en boucle, sur celles où les stations sont peu espacées et sur celles où les stations sont reliées par des liaisons point-à-point.

Quelle que soit la discipline, chaque source de messages est interrogée à son tour par la station contrôleur central. A l'arrivée d'un paquet d'élection, la station sollicitée transmet tous ses paquets en attente vers la station centrale. Une fois la transmission terminée, l'élection de la station suivante est initiée. Dans la discipline du "roll-call", cette opération est effectuée par la source centrale. Dans la discipline du "hub", l'initiation de la période d'élection se fait par l'interrogation par la station centrale d'une station située à une extrémité de la boucle ; après transmission de ses données en attente, cette station signale à la source prochaine sur la ligne de commencer à transmettre ; à la fin du cycle, quand toutes les stations connectées à la boucle ont été interrogées, le contrôle revient à la station centrale.

Le contrôleur peut aussi envoyer des données à l'une d'entre elles.

V.10.A.2. Etude analytique du temps d'accès

Ce protocole est déterministe car un time-out empêche toute station d'utiliser la ligne trop longtemps.

Le temps d'accès représente le temps qui s'écoule entre la fin d'une émission par une station et son élection suivante. Intuitivement, nous pressentons qu'il sera moindre dans la "hub" discipline car le paquet d'élection n'y effectue qu'une fois le tour de toutes les stations tandis que dans la "roll-call" discipline, le paquet d'élection nécessite l'envoi d'un paquet de confirmation vers la station centrale.

Un des paramètres critiques dans la détermination du temps d'accès est le "walk time" qui reçoit, dans ce cadre, une définition différente de celle présentée au point V.3. Il s'agit de la portion fixe de l'intervalle d'élection constituée du décodage du paquet d'élection, du délai de propagation, du temps de synchronisation du modem et autres,... Supposons qu'il soit identique pour toutes les stations. Le "walk time" global vaut :

$$L = N * T_w \qquad [V.29]$$

Le temps d'accès dans le cas le plus défavorable est alors représenté par la relation :

$$Ta = N * Tw + (N - 1) * TO \qquad [V.30]$$

avec TO :
le temps de service des demandes mises en file d'attente par chacune des N-1 autres stations et transmises vers la station centrale endéans le time-out TO.

Analysons d'abord le "walk time" global dans la discipline "Hub".

$$L = N * Tw, \text{ d'après [V.29]}$$
$$= N * (C + y)$$
$$= N * C + Y \qquad [V.31]$$

avec C :
le retard introduit par chaque source pour synchroniser les modems à la réception d'une invitation à émettre reçue de la station précédente et pour envoyer au terme de l'émission de ses messages une invitation à émettre à la station suivante ;

avec y :
le temps de propagation du message d'invitation à émettre de station à station si ces dernières sont uniformément distribuées autour d'une boucle.

avec Y :
le temps de propagation global sur la boucle sur lesquelles les stations sont connectées.

Analysons ensuite le "walk time" global dans la discipline "Roll Call".

$L = N * Tw$, d'après [V.29]

$= N * (C + (P/v_t)) + Y'$ [V.32]

avec C :
le retard introduit par chaque source pour synchroniser les modems et pour ajouter, afin d'indiquer la fin de la transmission de ses messages en attente, des caractères au dernier message destiné à la station centrale ;

avec P :
le retard introduit par l'émission par le contrôleur central, à destination de chaque station, d'un paquet d'élection de P bits ;

avec Y' :
le temps de propagation global des messages d'invitation à émettre qui dépend de la configuration du système et qui peut être amélioré par le choix judicieux de l'emplacement du site central par rapport aux stations ;
Y' est moins important si le contrôleur est au centre d'un bus bidirectionnel où les terminaux répartis en deux moitiés de part et d'autre de la station centrale, sont séparés par la même distance l'un de l'autre et vaut :

$$Y' = 2 * (Y/N * (1 + 2 + 3 + ... + N/2)) = Y/2 (1 + N/2) ; \quad [V.33]$$

rappelons qu'Y correspond au temps de propagation global si toutes les stations étaient connectées sur une boucle ;
dans le cas où les terminaux sont raccordés à un bus multipoints commandé d'un point central situé à l'une des extrémités et où chaque station est aussi identiquement espacée de son voisin, Y' vaut :

$$Y' = Y/N * (1 + 2 + ... + N) = (Y / 2) (1 + N); \quad [V.34]$$

Konheim et Meister ont analysé ces systèmes en tenant compte de l'interdépendance de l'état des files aux différentes sources pour en calculer le temps de réponse moyen (voir chapitre 12 de Schwartz, 24).

V.10.A.3 Critique du protocole

Bien que ce protocole soit déterministe, certains se refusent à l'utiliser car le temps passé à l'élection des stations est trop important, car le temps d'accès d'une station est élevé même si la charge est faible.

Ce protocole a un point névralgique qui est son contrôleur ; si le contrôleur tombe en panne, tout le système est en panne.

Certains lui reprochent son manque de flexibilité car l'ordre d'interrogation des stations est prédéfini. Enfin, cette méthode nie certains des avantages d'un système distribué et rend difficile la communication entre deux stations autres que le contrôleur.

V.10.A.4 Illustration

Comme illustration, on peut citer le système IBM PARS développé à partir de l'original American Airlines Sabre System de 1961 (Schwartz, 24) où un concentrateur effectue le "polling" d'une ligne multipoints connectant un groupe de terminaux.

V.11. Conclusion

Une première remarque est que la majeure partie de la littérature raisonne en termes moyens pour calculer le temps de réponse et qu'elle consacre peu d'efforts à la détermination du temps d'accès. Or, même si, sous faible charge, le temps d'accès moyen d'un protocole probabiliste est inférieur au temps d'accès d'un protocole déterministe, ce dernier sera le seul à présenter une borne supérieure sous forte charge.

Une deuxième remarque est qu'il est difficile de comparer les études réalisées par différents auteurs car les hypothèses de départ varient. Certains se concentrent sur le temps de réponse subi par paquet, d'autres par message. Nous avons résolu le problème en supposant à priori qu'un message est constitué d'un seul paquet. La définition du temps de réponse d'un message inclut pour certains le temps pris pour la confirmation de la réception du message par le destinataire. Certaines études négligent le délai de propagation, d'autres pas... Liu (dans Liu, Hilal et Groomes, 17) est un des rares auteurs à être parvenu à comparer des modèles analytiques développés par d'autres sur le "*Token Ring*" (voir point V.8.B), le "*Slotted Ring*" (voir point V.8.E), le "*CSMA-CD*" (voir point V.7.C) et le DLCN (généralisation du "*Register Insertion Ring*" voir point V.8.F) et comme Bux, il en a conclu que le "Token Ring" présentait des retards plus importants que le "CSMA-CD" sous faible charge mais un état plus stable sous forte charge.

Une troisième remarque est que d'autres facteurs influencent le temps

d'accès comme la topologie et certains auteurs comme Sudhir R. Alufa des laboratoires Bell à Holmdel affirment que les réseaux en étoile présentent les meilleurs caractéristiques pour les applications temps réel. En effet, contrairement aux autres topologies, plusieurs stations peuvent émettre simultanément des messages sur le réseau sans que ces derniers ne subissent de retard de transmission c'est-à-dire d'attente en file centrale. Le réseau présente donc un temps d'accès plus favorable que dans le cas où une seule station à la fois ne pourrait émettre un message sur le canal. Si l'on prend le cas du SPS (voir point II.3), le nombre moyen d'émissions simultanées de messages qui peuvent être effectuées sur le réseau de l'étoile du MHC s'exprime par le rapport de la capacité maximale de traitement des messages du MHC en Kmots/sec sur le débit réel moyen d'une liaison. Si les liaisons soutiennent un débit réel maximum, deux messages peuvent être transmis sans subir d'attente en file centrale. La figure V.23. représente les variations du nombre de transmissions simultanées en fonction du débit réel moyen d'une liaison par une partie d'hyperbole équilatère dont les asymptotes sont les axes de coordonnées (pour plus de détails sur les simulations du réseau du SPS, on consultera Altaber et Jeanneret, 1).

De même, sur une topologie en bus, les protocoles comme les protocoles à l'écoute du canal comme le "CSMA-CD" (voir point V.7.C), le "SCSA" (voir point V.7.D),... sont sensibles à la longueur du bus qui influence le délai Θ de propagation et donc, le calcul du temps d'accès. En particulier, pour un protocole "CSMA-CD", plus le délai de propagation est important, plus le risque de collisions est élevé car deux stations distantes mettront un certain temps avant d'être conscientes que le canal est occupé par l'une ou par l'autre.

Une quatrième remarque est que le temps d'accès des protocoles à accès statique et à accès contrôlé (voir points V.2) est influencé par le nombre de stations connectées au réseau. En effet, dans le "*Token Passing*", si le nombre de stations augmente, de plus en plus de temps sera consacré au passage du jeton de station à station et relativement moins au temps de transmission des données provoquant un temps d'accès supérieur. Sa performance dépend donc du temps nécessaire pour traiter le jeton, de sa taille,...

En guise de cinquième remarque, nous allons discuté de l'opportunité du choix du protocole d'accès "Token Ring" (voir point V.8.B) par les concepteurs du LEP (voir point III.2) pour régler l'accès concurrent des satellites au moyen

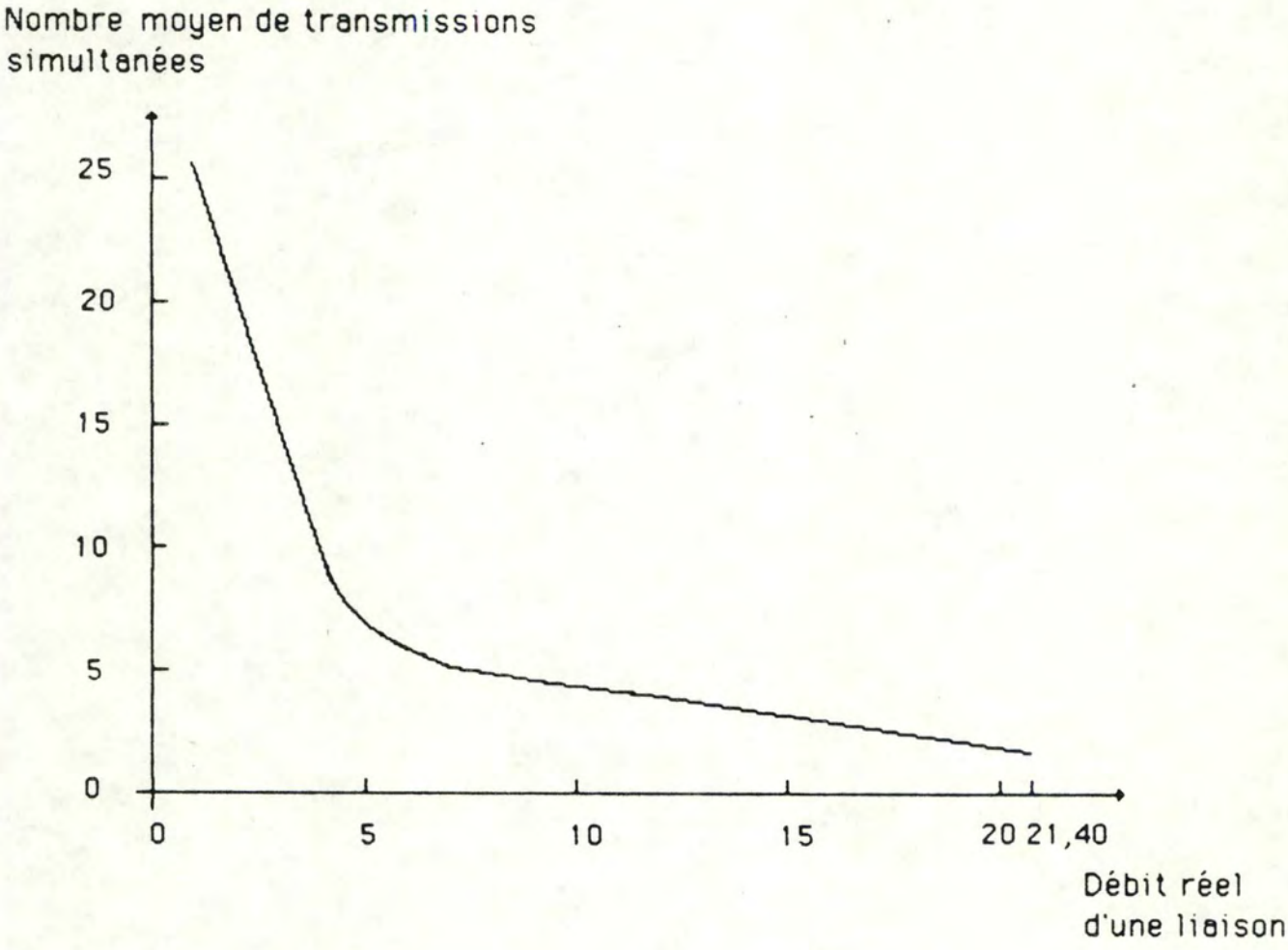


Figure V.23. : Nombre moyen de transmissions simultanées de messages

de communication.

Selon nous, des réseaux autres que le Token Ring régis par d'autres protocoles auraient pu convenir : le réseau Ethernet, par exemple. Certes, tel quel c'est-à-dire conduit par le protocole CSMA-CD/BEB (voir point V.7.C) il aurait été un choix regrettable par son caractère probabiliste. Cependant, si on lui adjoignait un protocole déterministe à l'écoute du canal qui détecte les collisions comme le "SCSA", qui évite les collisions comme le "CSMA-CA" ou qui limite les collisions comme l'"Adaptive Tree Walk", on pourrait le préférer au Token Ring. De plus, Ethernet est plus fiable, ce qui n'est pas sans importance dans un système temps réel. Il utilise un moyen de communication passif et n'introduit des répéteurs pour connecter ses différents segments que dans le cas où les distances couvertes sont trop importantes pour qu'un bon niveau de signal soit maintenu entre tout couple émetteur-récepteur (voir point III.2).

Si l'on compare les modèles mis en évidence pour les différents protocoles déterministes à l'écoute du canal c'est-à-dire pour :

- le "SCSA" : $Ta = (2 * CRP) + (2 * (N-1)*Ts)$ [V.17]
- l'"Adaptive Tree Walk" : $Ta = (2 * \Theta)*((2*N)-1) + (N-1)*Ts$ [V.19]
- le "Bit-Map" : $Ta = (N-1)*Ts + N*\Theta + (N-1)*Ts$ [V.25]
- le "BRAP" : $Ta = (N-1)*Ts$ [V.26]
- le "MLMA" : $Ta = 1110*\Theta + (N-1)*Ts + 1110*\Theta + (N-1)*Ts$ [V.27]
- le "Binary Countdown" : $Ta = Log_2N*\Theta*N + (N-1)*Ts$ [V.28],

nous remarquons que le protocole "Broadcast Recognition with Alternating Priorities" présente le temps d'accès le plus favorable. Si Ethernet était régi par un tel protocole, son temps d'accès pourrait même être inférieur à celui du Token Ring qui s'exprime par la formule [V.22], $Ta = Tw + (N-1)*TO$, suivant les valeurs de Tw et de TO.

Nous n'avons pas envisagé comme concurrent le protocole "Polling" car le temps passé à l'élection des différentes stations est trop important. De même, les protocoles "FDMA" et "STDM" ne sont pas à recommander car ils allouent statiquement la largeur de bande, soit dans le temps, soit dans l'espace aux différentes stations même si elles n'ont rien à émettre. Les protocoles déterministes de la famille "Aloha" sont particuliers aux voies

radioélectriques terrestres ou aériennes et ne s'appliquent donc pas à notre cas où les distances couvertes ne nécessitent pas l'emploi de satellite ou de liaisons hertziennes. Parmi les autres protocoles à accès contrôlé déterministes, le "Slotted Ring" n'est pas non plus à conseiller car il impose, dans le cas du "Cambridge Ring", un découpage de l'information utile en tranches de 16 bits ; ceci amène à un temps d'accès dégradé d'un message au canal quand d'importants flots de données sont échangés entre les satellites.

Peut-être que suite aux difficultés rencontrées par la firme IBM pour mettre sur le marché son réseau Token Ring, les concepteurs du LEP opteront pour un Ethernet à protocole d'accès "BRAP"...

Bibliographie de la partie II

Chapitre IV

(1) CORNAFION, 1981, *Systèmes Informatiques Répartis,* Dunod, Paris.

(2) James F.Kurose, Mischa Schwartz and Yechiam Yemini, "Multiple-Access Protocols and Time-Constrained Communication", *Computing Surveys, Vol.16, No.1, March 1984.*

(3) G. Le Lann, décembre 1984, Cours d'introduction aux réseaux locaux donné au CERN.

(4) Ware Myers, Août 1982, "Toward a Local Network Standard", *IEEE Micro*, by The Institute of Electrical and Electronics Engineers, Inc., pp 28-45.

(5) William Stallings, 13 Février 1984, "IEEE Project 802 Setting standards for local-area netwoks", *Computerworld.*

Chapitre V

(1) J. Altaber et J-P. Jeanneret, février 1978, "Mesures du réseau de contrôle du SPS", CERN-SPS/ACC/78-1.

(2) Gojko A. Babic, Ming T. Liu, and Roberto Pardo, 1977, "A Performance Study of Distributed Loop Computer Network (DLCN)", *Proc. 1977 International Conference on Parallel Processing.*

(3) Simon Ya. Berkovitch, Colleen R. Wilson & Chris Walter, 1983, "CITO - A New Technique for Computer Communication", *Proceedings of Computer Networking Symposium,* IEEE Computer Society.

(4) Werner Bux, October 1981, "Local-Area Subnetworks : A Performance Comparison", *IEEE Transactions on Communications* , Vol. Com. 29, N° 10.

(5) J.I. Capetanakis, 10 October 1979, "Generalized TDMA : The multi-accessing tree protocol", *IEEE Trans. Commun. COM-23,* pp 1476-1484.

(6) Cornafion, 1981, *Systèmes Informatiques Répartis*, Dunod, Paris.

(7) A.K. Elhakeem, R.K. Goel, and A.P. Dhawan, December 13 1983, "Simulation of FARA/CS, a New Access Protocol", *Proceeedings òf Computer Networking Symposium,* IEEE Computer Society.

(8) R.G. Gallager, 17-20 September 1978, "Conflict resolution in random access broadcast networks", *Proceedings of the AFOSR Workshop in Communication Theory and Apllications*, pp 74-76.

(9). G.T. Hopkins, 1979, "Multimode communications on the Mitrenet", *Proceedings of The Local Area Communications Network Symposium (Boston, May),* Mitre Corp., McLean, Va., pp 167-178.

(10) IEEE Project 802 Committee, October 19 1981, "A Status Report on Local Network Standards Committee", Draft B.

(11) Byng G. Kim, December 13 1983, "Two Adaptive Token Ring Strategies for Real-Time Traffic", *Proceeedings of Computer Networking Symposium,* IEEE Computer Society.

(12) L. Kleinrock and M.O Scholl, 7 July 1980, "Packet switching in radio channels : New conflict-free multiple access schemes", *IEEE Trans. Commun. COM-28,* pp 1015-1029.

(13) L. Kleinrock and Y. Yemini, 1978, "An optimal Adaptive Scheme for Multiple Access Broadcast Communication", *Proc. ICC,* pp 7.2.1. - 7.2.5.

(14) James F.Kurose, Mischa Schwartz and Yechiam Yemini, "Multiple-Access Protocols and Time-Constrained Communication", *Computing Surveys, Vol.16, No.1, March 1984.*

(15) Lam et Kleinrock, 1973, "Packet-Switching in a Slotted Satellite Channel", *AFIPS Conference Proceedings*, National Computer

Conference, 42, pp. 703-710.

(16) G.Le Lann, décembre 1984, Cours d'introduction aux réseaux locaux donné au CERN.

(17) Ming T. Liu, Wael Hilal and Bernard H. Groomes, 1982, "Performance evaluation of channel access protocols for local computer networks", *Proceedings of the COMPCON Fall 82 Conference,* IEEE, Inc.

(18) B. Mahbod and J. Howard, December 13 1983, "Performance Analysis of a Cooperative Symmetric Algorithm for Real-Time Local Area Computer Communication Networks", *Proceedings of Computer Networking Symposium,* IEEE Computer society.

(19) Robert M. Metcalfe and David R. Boggs of Xerox Palo Alto Research Center, July 1976, "Ethernet : Distributed Packet Switching for Local Computer Networks", *Commun. Ass. Comput. Match.*, Volume 19, Number 7, pp 395-404.

(20) O. Mirabella and A. Salvo, December 13 1983, "Throughput Improvement in a Ring Lan", *Proceedings of Computer Networking Symposium,* IEEE Computer society.

(21) Ware Myers, Août 1982, "Toward a Local Network Standard", *IEEE Micro*, by The Institute of Electrical and Electronics Engineers, Inc., pp 28-45.

(22) B.K. Penney and A.A. Bagdadi, August 1979, "Survey of computer communications loop networks : part 1 and 2, vol. 2, n° 4 et 5, *IPC Business Press.*

(23) Louis Pouzin, Centre National d'Etudes des Télécommunications, Issy-les-moulineaux, France, 1982, "Local Area Networks".

(24) Mischa Schwartz, *Computer-Communication Network Design & Analysis*, Prentice-Hall, Inc., Englewood Cliffs, N.J.07632.

(25) John F. Shoch and Jon A. Hupp, "Performance of an Ethernet Local Network - A Preliminary Report", *Digest of Papers - Compcon Spring 82,* IEEE Computer Society, Los Alamitos, CA, pp. 115-120.

(26) William Stallings, 13 Février 1984, "IEEE Project 802 Setting standards for local-area netwoks", *Computerworld.*

(27) William Stallings, March 1984, "Local Networks", *Computing Surveys,* Vol. 16, N° 1.

(28) Bart Stuck from Bell Laboratories, January 1983, "Which Local net bus access is most sensitive to traffic congestion ? ", *Data Communications* , McGraw-Hill, Inc., pp 107-120.

(29) A.S. Tanenbaum, 1981, *Computer Networks,* Prentice-Hall, Englewood Cliffs, New Jersey.

(30) F.A. Tobagi and V.B. Hunt, October 1980, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection", *Computer Networks,* Vol. 4, No. 5, pp. 245-259.

(31) Mars 1985, "Le Wangnet sur la sellette", *Data News n°5,* Dossier n°54.

PARTIE III : CONTRIBUTION A LA FIABILITE DANS LES SYSTEMES TEMPS REEL

Nous nous sommes longuement étendu dans la deuxième partie du mémoire sur le problème du temps de réponse dans les systèmes temps réel. Or, notre étude de ces systèmes ne saurait être complète sans aborder l'aspect fiabilité que nous allons illustrer dans cette dernière partie. Cette deuxième caractéristique fondamentale des systèmes temps réel a déjà été introduite dans la première partie du mémoire (voir chapitre I, point I.3). Nous y avons vu que la fiabilité était un élément crucial dans les systèmes temps réel car la nature de leurs fonctions fait que tout arrêt de travail entraînerait de graves inconvénients, d'autant plus graves que les exigences de temps de réponse sont pressantes.

Chapitre VI : AMELIORATION DE LA FIABILITE DU SYSTEME DE CONTROLE DU SPS

Nous proposons maintenant une illustration pratique de la fiabilité, basée sur les réalisations du CERN, et plus particulièrement de l'accélérateur SPS (voir chapitre II).

Dans ce cadre d'idées, nous appliquerons les concepts-clés (voir point I.3) de la fiabilité au système de contrôle du SPS (voir point VI.1). Nous évoquerons ensuite les moyens initialement mis en oeuvre pour effectuer des diagnostics rapides (et ainsi réduire le temps moyen de réparation ou MTTR) (voir point VI.2).

Malheureusement, ces diagnostics nécessitent des interventions sur le site même de la panne (dialogue exclusivement local avec un ordinateur incriminé). Pour réduire cette perte de temps en déplacements, la section responsable de la maintenance des ordinateurs du SPS proposa de mettre au point une liaison de secours, qui fut l'objet de notre stage de 6 mois au CERN. Cette solution exigea la définition d'une architecture matérielle adéquate (voir point VI.3) et la mise au point d'une architecture logicielle cohérente dont nous donnerons les spécifications fonctionnelles (voir point VI.4). Le résultat du stage sera illustré par un exemple de diagnostic à distance (voir point VI.5), et, par les possibilités d'évolution qu'il offre pour l'avenir (voir point VI.6)...

VI.1. La fiabilité et le système de contrôle du SPS

Si l'on en croit les statistiques, l'ensemble de l'accélérateur SPS fonctionne à 95 % du temps d'opération prévu. Parmi les 5% du temps de panne, le système informatique de contrôle constitué d'un réseau local interconnectant une série de sous-réseaux en étoile de minis n'y contribue que pour moins d'1 % c'est-à-dire que le système ne connaît des arrêts qu'une à deux fois par mois et ce pour une durée maximale de 3 heures.

Comme annoncé au chapitre I, les problèmes peuvent se situer tant au niveau logiciel (voir point I.3.C) que matériel (voir points I.3.A et I.3.B).

Pour faire face aux pannes logicielles, la méthode radicale utilisée est le rechargement du système d'exploitation commandé soit localement, soit à distance d'un autre point du réseau. Cependant, par le recours à l'architecture distribuée, on essaie de garder les différents logiciels aussi simples que possible (voir point I.4.D).

Une illustration peut en être donnée par les pannes qui, bien que peu fréquentes, se produisent dans les minis-satellites. Elles se manifestent le plus souvent sous la forme d'une fermeture de la ligne assurant la liaison d'un satellite avec le réseau : la raison en est reportée sur l'ordinateur ALARM de la salle de contrôle. La remise en service nécessite l'intervention de l'opérateur afin de recharger à distance le satellite en faute. Dans la plupart des cas, ces manoeuvres sont suffisantes pour remettre le satellite en route. Le problème qui se pose est que ces manoeuvres sont extrêmement faciles à exécuter et que les rares fautes restantes du système d'exploitation qui sont dues essentiellement à des blocages sur ressources lors de situations exceptionnelles sont, de ce fait, masquées. Cependant, un enregistrement de ces diverses manoeuvres est gardé et il est examiné régulièrement. Il est possible de cette façon de détecter les satellites instables qui seront mis sous surveillance particulière.

Pour faire face aux pannes matérielles, on recourt soit à la technique de "Fault Tolerance" (voir point I.3.A), soit à la technique de "Fault Avoidance" (voir point I.3.B).

Dans la technique de "Fault Tolerance", la méthode la plus usitée est celle du dédoublement.

Un premier exemple se situe au niveau des centres des sous-réseaux en étoile (voir points II.2 et II.3) qui sont les seuls à être dédoublés par des ordinateurs de secours car la topologie *en étoile* est, par réputation, la moins fiable. La topologie la plus fiable est, en effet, celle *en bus* car il s'agit d'un moyen de communication passif. Ensuite vient la topologie *en anneau* qui dépend fortement de l'état des répéteurs actifs à chaque noeud. Dans cette topologie, une coupure de liaison entre noeuds peut amener à une panne générale du réseau. Cet état de fait peut cependant être résolu par l'emploi d'un commutateur à chaque répéteur qui court-circuitera automatiquement chaque noeud fautif ou par le recours à une configuration en double boucle (on consultera, pour plus de détails, l'article 7). On a donc prévu un ordinateur de remplacement dont la commutation en opération est effectuée manuellement pour chaque centre d'étoile. Cependant, statistiquement, les centres restent fréquemment plus d'un mois sans nécessiter d'intervention.

Un deuxième exemple est donné dans le sous-réseau développé autour du MHC (figure II.2. du chapitre II). Les ordinateurs *LIB* sur lequel se trouve la librairie des programmes, *ALARM* qui concentre tous les messages d'erreur rencontrés sur le site du SPS et *DISP* qui sert à l'affichage de graphiques y forment une configuration "*duplex*" (voir point I.3.B) avec l'ordinateur *SERV1*. Cependant, que la commutation soit automatique ou manuelle, elle introduit des complications. Certains, par expérience, affirment même que les ordinateurs de secours dans les systèmes à commutation automatique donnent plus de problèmes que les ordinateurs originaux (Crowley-Milling, 2).

Ces exemples renseignent sur les seuls satellites qui ont été duplifiés. En effet, l'hypothèse de départ pour cette décision a été qu'un ordinateur était ni plus ni moins fiable que tout autre appareillage intervenant dans l'accélérateur : prévoir une duplication à ce niveau aurait dû logiquement entraîner une duplication de tout équipement de l'accélérateur.

Dans la technique de "Fault Avoidance", on essaie de réduire le MTTR en recourant à une architecture distribuée (voir point I.4.D) et en procurant des moyens pour effectuer des diagnostics rapides. Notre stage au CERN dans la section responsable de la maintenance des ordinateurs destinés au contrôle de l'accélérateur et des zones expérimentales du SPS a eu justement pour but

d'augmenter la panoplie de ces moyens.

VI.2. Moyens mis en oeuvre au SPS, avant le stage, pour effectuer de rapides diagnostics des ordinateurs

Quelles sont les seules méthodes mises, avant le stage, à la disposition de l'ingénieur pour accélérer les réparations dans le cas des ordinateurs du SPS (voir point I.3.A) ?

D'abord, il faut être certain que le problème vient d'un ordinateur. Une première étape consiste alors à inspecter sur l'ordinateur ALARM la liste des erreurs et problèmes rencontrés sur le site du SPS. Ce premier examen permet d'orienter les recherches soit vers un CAMAC (voir point II.4.D.1), soit vers un multiplexeur de données (voir point II.4.D.2), soit vers une liaison de données (voir point II.3.A) ou soit vers un mini satellite (voir point II.4) ou centre d'une étoile (voir point II.3.B). Suivant la cause du problème, on fait appel à l'équipe spécialisée de son dépannage.

S'il s'agit d'un satellite ou du centre d'une étoile, plusieurs expédients sont disponibles à l'équipe responsable. Prenons le cas de la panne d'un satellite.

Il faut d'abord veiller à ce qu'il ne s'agisse pas d'une panne logicielle car, dans ce cas, il faut entamer les procédures de redémarrage citées plus haut. Si elles ne donnent rien, il s'agit de fautes dues au mauvais fonctionnement du matériel d'interface de ligne de l'ordinateur ou de l'ordinateur lui-même.

Lorsque l'ordinateur n'est pas totalement en panne, une deuxième étape consiste alors à commander, de n'importe quel point du réseau et par des instructions-réseau du type de celles présentées au point II.4.A, le chargement, via le MTS (voir II.3), de programmes NODAL de test stockés en librairie (ordinateur LIB dans le sous-réseau du MHC, ordinateur YLIB dans le sous-réseau MHY,...). Ce moyen permet de réaliser un diagnostic grossier et de détecter qu'un bit d'une adresse-mémoire est en panne, qu'un registre CPU

ne fonctionne plus ou qu'une horloge temps réel est bloquée...

Le plus souvent, la liaison de données entre le centre de l'étoile et le satellite en problème est elle-même malade, ce qui rend impossible la procédure précédente, d'où la nécessité d'une troisième étape. Celle-ci nécessite le déplacement sur place de l'ingénieur pour effectuer localement des tests de programmes binaires. Par la console opérateur, il est possible de dialoguer avec le CPU même si l'ordinateur est en panne mais pour autant qu'on ait sélectionné, sur le panneau avant du mini incriminé, le bouton OPCOM qui active le programme MOPC (Microprogrammed OPerator's Communication) (figure VI.1.). Ce programme détaillé dans l'annexe A-1 : "section IV du NORD-100 Reference Manual : Operator's interaction with NORD-100" assure :

- l'interface avec le terminal opérateur connecté au port d'entrée/sortie de la carte CPU ;
- l'exécution de fonctions comme la lecture/écriture du contenu d'adresses-mémoire et de registres, le chargement de système d'exploitation, les dumps-mémoire...

Quand on sait que le site du SPS s'étend sur plusieurs km^2, on comprend que l'idée ait surgi, pour éviter les déplacements en voiture, de mettre au point une liaison de secours permettant à l'opérateur de transporter à distance sa console de dialogue avec le MOPC, c'est-à-dire de pouvoir effectuer, à partir de n'importe quel point du réseau, un télé-diagnostic des ordinateurs. C'est dans ce but qu'un stage de six mois fut mis sur pied.

Le projet a été limité aux ordinateurs *General Purpose,* que nous symboliserons désormais par GP*, plus particulièrement destinés au contrôle de l'accélérateur SPS et distribués autour du centre de l'étoile MHC dans les différents Bâtiments Auxiliaires (voir présentation du SPS dans l'introduction au chapitre II). Il ne s'agit donc pas des GP* présents dans les zones expérimentales.

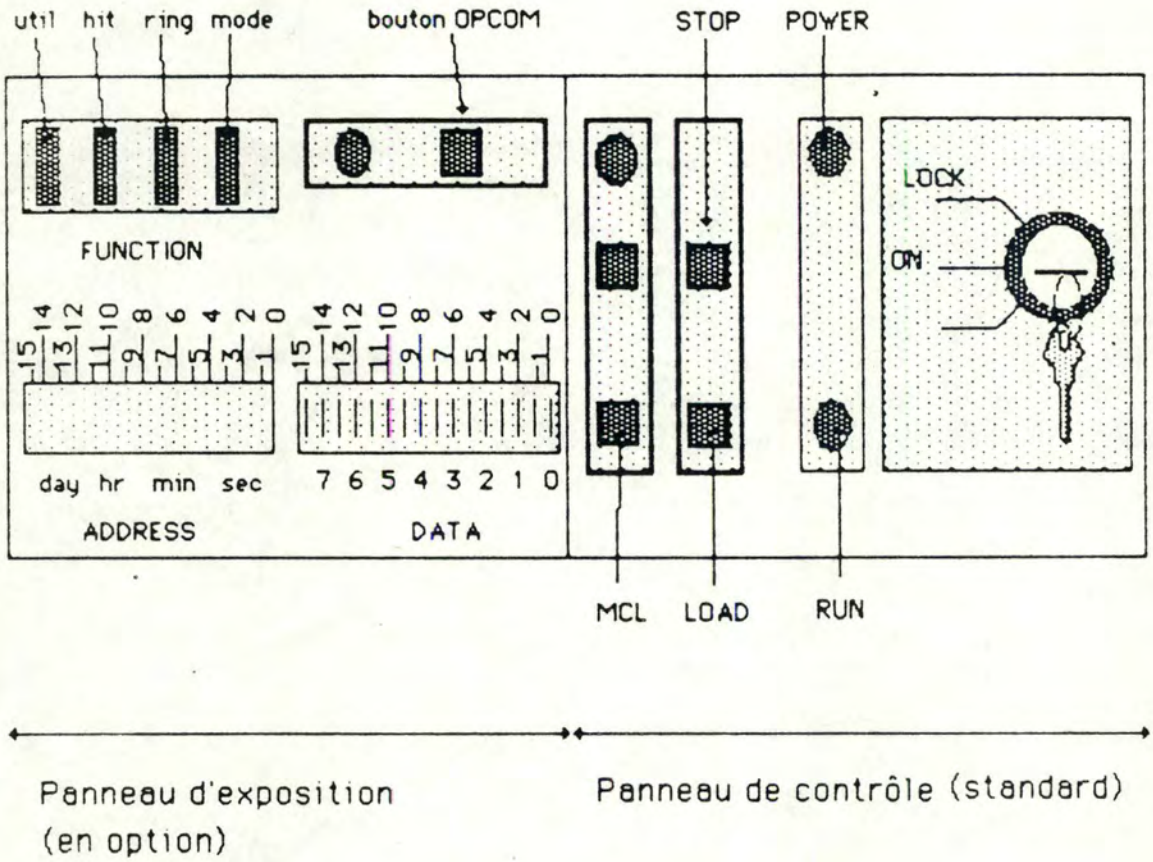


Figure VI.1. Le panneau avant du NORD-100 avec le bouton OPCOM

VI.3. Configuration matérielle mise sur pied

L'idée de base du projet a été de brancher sur l'architecture CAMAC (voir point II.4.D.1) et multiplexeur de données (voir point II.4.D.2) existante de l'ordinateur Acces une série de stations particulières ou micro-ordinateurs, un par Bâtiment Auxiliaire et tel que chacun serait dédicacé au contrôle, via le MOPC, des différents GP* présents dans ce Bâtiment (figure VI.3.). Il s'agit de stations particulières car normalement, les stations connectées au bus sont des chassis dans lesquels on peut intégrer des modules d'interface-équipements. Ainsi, dans le BA6, un microordinateur doit assurer le contrôle des GP6, GP12 et GP23.

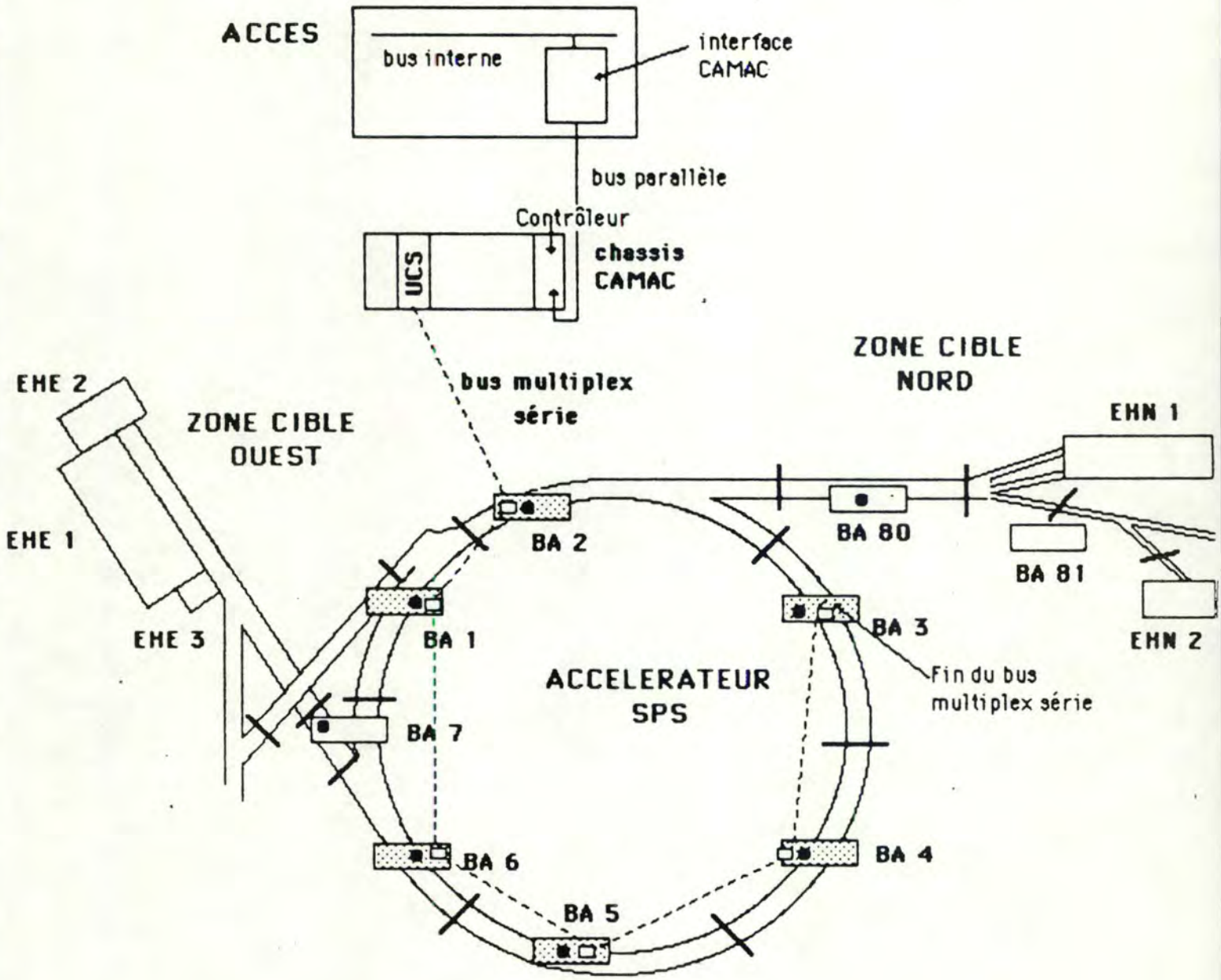
VI.3.A. Le micro-ordinateur

Le micro-ordinateur est un petit système construit autour du bus à 8 bits de données G64. Ce bus défini à son origine par Gespac est devenu un standard CERN pour des projets d'ordre général qui ne nécessitent pas l'emploi d'un microprocesseur de plus de 8 bits. Dans le cas qui nous occupe, il s'agissait d'un Motorola MC 6809.

Ce recours aux micro-ordinateurs rentre dans l'objectif général du SPS de distribuer les centres de décision pour accroître la vitesse d'intervention et pour introduire plus de souplesse dans l'architecture générale.

VI.3.B. L'ordinateur Acces

Le choix de l'ordinateur Acces comme support du projet ne s'est pas fait par hasard. L'Acces (figure VI.2.), satellite du sous-réseau MHC, contrôle l'accès du personnel aux zones éventuellement dangereuses du SPS comme le



- Point d'accès au tunnel de l'accélérateur ou aux tunnels de transfert
- | Porte séparant les différents sextants du tunnel ou scindant les tunnels de transfert en plusieurs zones.
- ▭ Station connectée au bus multiplex contenant les interfaces des équipements de surveillance de l'accès au tunnel

Figure VI.2. : L'Accès et son bus multiplex série très longue distance

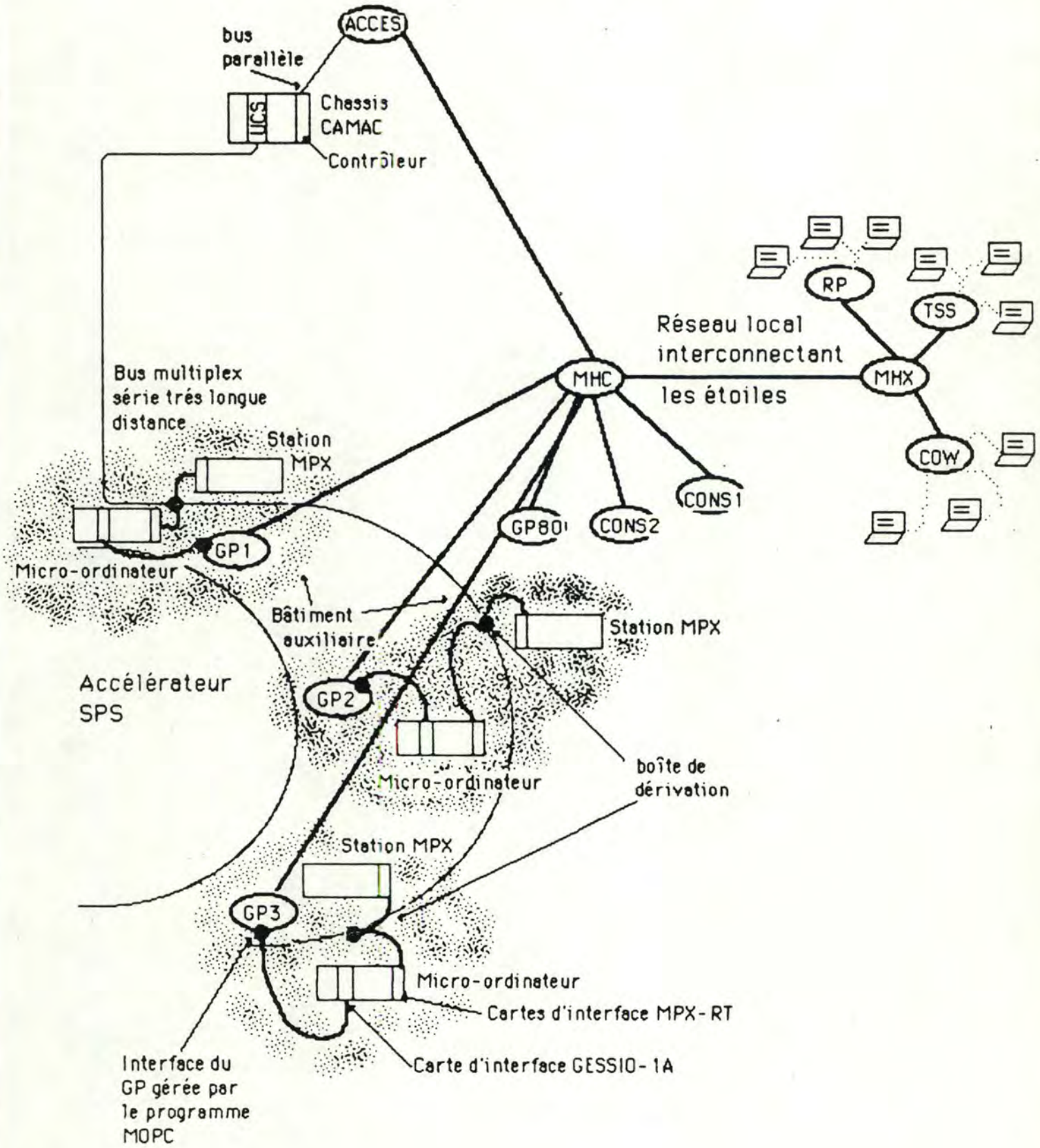


Figure VI.3. : Configuration matérielle et liaison de secours

tunnel de l'anneau et les puits. Il est donc d'office relié par ligne aux Bâtiments Auxiliaires où se trouvent les ordinateurs GP* à contrôler.

Rappelons que le tunnel de l'accélérateur est subdivisé en six sextants, l'accès à chacun d'entre eux étant surveillé depuis l'intérieur d'un des 6 Bâtiments Auxiliaires cantonnés autour de l'accélérateur dans lesquels se trouvent également les GP*. Le même dispositif est prévu pour les tunnels de transfert.

Pour pouvoir contrôler des équipements sur de si longues distances, on a adjoint au satellite Acces une interface classique d'équipements (voir point II.4.D) c'est-à-dire un CAMAC et un bus multiplex série très longue distance qui effectue le tour de l'accélérateur et passe par les Bâtiments Auxiliaires figure VI.2.). Sur ce bus, comme sur tout bus multiplex, on peut connecter des stations pouvant contenir des modules d'interface-équipements (dans ce cas, interface aux équipements de surveillance). Ces stations seront, dans ce projet, nos micro-ordinateurs chargés, d'une part, de l'interface avec les équipements particuliers que sont les GP* et, d'autre part, de l'interface avec le bus multiplex et donc le reste du réseau.

VI.3.C. La liaison de secours

L'idée de ce projet était de pouvoir interroger, à vitesse respectable, un GP* d'un des terminaux ou d'un des domiciles d'un ingénieur de la section c'est-à-dire de son téléphone privé, si on adapte un modem aux ordinateurs de la section, satellites du MHX (autre centre d'étoile).

Pour cela, il faut emprunter la liaison MHX-MHC dessinée sur la figure VI.3., puis la liaison MHC-Acces dans lequel aura été défini un Data Module (voir point II.4.D.3) qui s'appelle MOPC et dont le rôle sera d'assurer l'acheminement des caractères via le CAMAC et le multiplexeur de données vers le micro-ordinateur auquel est connecté le GP* litigieux.

Nous avons donc écrit un Data Module et programmé le micro-ordinateur.

VI.4. Spécifications fonctionnelles du logiciel réalisé

Le logiciel développé peut être schématisé par une architecture de télécommunication en trois couches (figure VI.4.).

Au premier niveau, sont définis les interfaces de dialogue avec le niveau physique : le *module 1* réalise l'interface du micro-ordinateur avec l'interface du GP* (*module 5*), interface géré par le programme MOPC (voir point VI.2) et le *module 2* réalise l'interface du micro-ordinateur avec le bus multiplex.

Ces interfaces établissent donc **la couche physique** et permettent le dialogue entre les équipements constitutifs de la configuration matérielle c'est-à-dire d'une part entre le micro-ordinateur et le GP* et d'autre part entre le micro-ordinateur et le multiplexeur de données (figure VI.3.).

Au deuxième niveau est défini un Data Module MOPC qui présente à tout utilisateur de l'Acces un interface agréable utilisable dans les programmes NODAL pour commander un équipement c'est-à-dire un GP* (pour lui ordonner, par exemple, la lecture de 1000 adresses-mémoire...). Il ne faut pas se méprendre sur le nom de ce Data Module : il ne s'agit pas du programme MOPC activé dès que le bouton OPCOM sur le panneau avant d'un GP* est enclenché. Désormais, quand nous parlerons de MOPC, il s'agira de ce programme. Sinon, nous citerons explicitement le terme de Data Module. Celui-ci diverge légèrement de la notion classique (voir point II.4.D.3) et se scinde en deux parties :

- le *module 3A* est la partie présente dans l'Acces ; il aiguille, sur base d'une pseudo-data table, vers le micro-ordinateur adéquat auquel est connecté le GP* litigieux tous les paramètres présents dans l'appel au Data Module ; ce module dialogue avec le *module 2* sur base d'un protocole établi entre l'Acces et le micro-ordinateur ;
- le *module 3B* est la partie présente dans le micro-ordinateur ; il analyse, sous interruption, les paramètres transmis au *module 2* par le *module 3A* ; parmi les paramètres, se trouve la propriété c'est-à-dire l'action à exécuter par le micro-ordinateur ; celle-ci concerne soit uniquement le micro-ordinateur, soit directement le

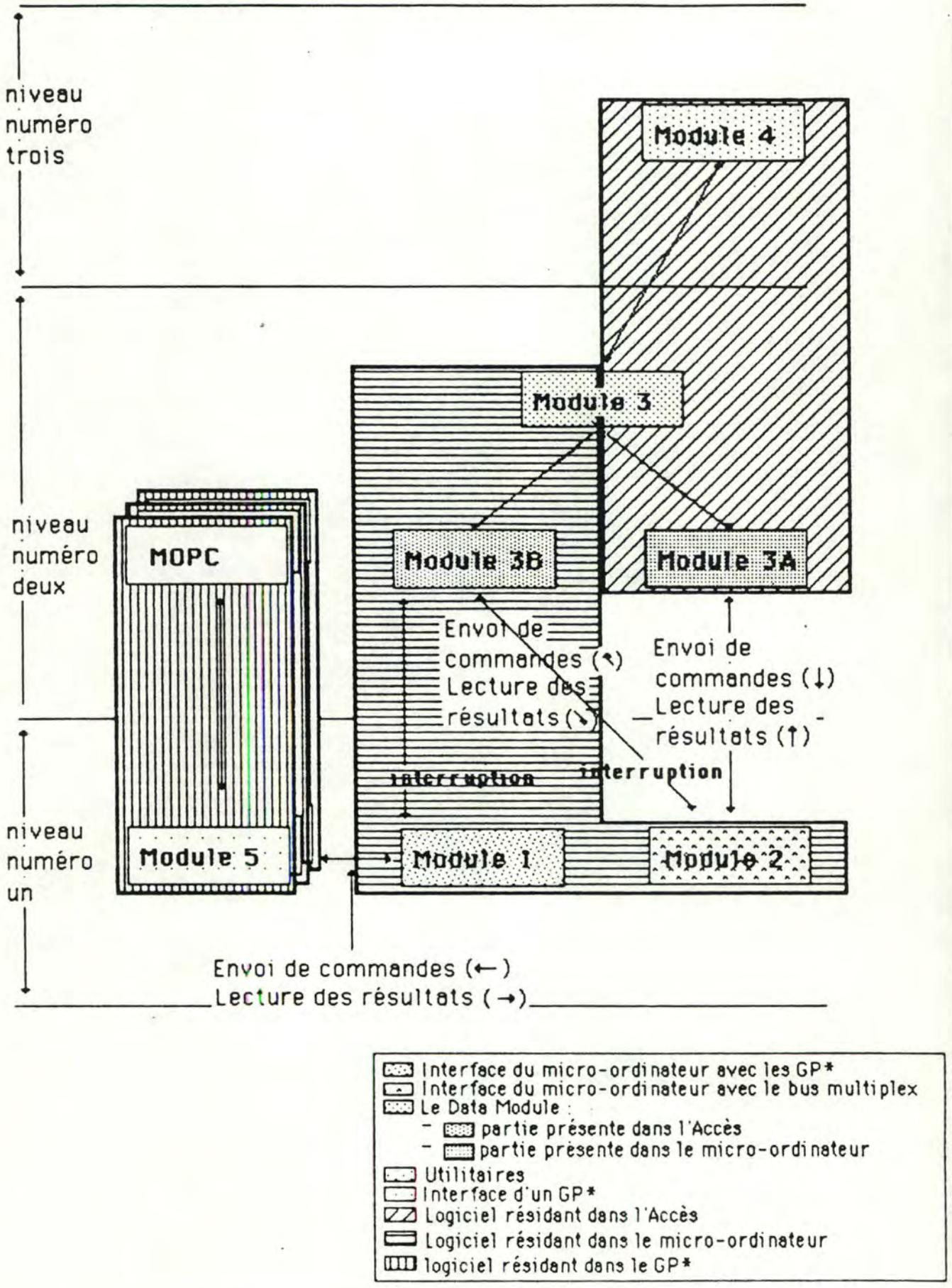


Figure VI.4. : Architecture logicielle en trois couches

MOPC ou soit le chargement de programmes de test binaires dans le GP* à contrôler ;

sur base de la table des propriétés, le micro-ordinateur se branche sur le code traitant de la propriété ; si cette propriété demande l'envoi d'instructions au MOPC selon la syntaxe correcte (définie en annexe au point A-1), le micro-ordinateur se charge de coder les informations en provenance de l'Acces pour leur compréhension par le GP* ; via le *module 1*, il conduit ensuite le dialogue, sous interruption prioritaire par rapport à celle en provenance de l'interface avec le bus multiplex, avec le MOPC ;

l'Acces teste, toutes les 20 msec, la présence de résultats disponibles dans le micro-ordinateur via le *module 2* et renvoie une erreur vers l'opérateur si le traitement de l'interruption n'est pas achevée après 100 msec.

Il est important d'ajouter que le dialogue ne s'effectuera jamais qu'avec un seul GP* à la fois. Ce niveau réalise **la couche logique** qui permet de l'Acces de dialoguer avec les équipements c'est-à-dire les GP* (figure II.3.).

Au troisième niveau, au niveau de l'Acces, sont définis des utilitaires en NODAL (voir point II.4.A) pour effectuer le chargement dans le GP* en litige d'une série de programmes de test binaires résidant en librairie (ordinateur LIB du réseau défini autour du MHC). Ces utilitaires constituent **la troisième couche** et le module 3 de la figure VI.4. ; leur exécution à distance peut être demandée par des instructions-réseau de n'importe quel point du réseau.

Les modules 2, 3A et 5 avaient déjà été implémentés. Le travail a donc consisté en la programmation de logiciels (modules 1 et 3B) résidant dans le micro-ordinateur dans les langages Pascal Omegasoft 6809 (référence 5) et Omegasoft Assembleur 6809 (référence 6) pour les parties critiques et en la programmation d'utilitaires en NODAL (voir point II.4.A.). La première phase s'est effectuée, d'une part, sous système de développement FLEX (référence 3) : elle revenait à la programmation des modules 1 et 3B et à leur intégration avec le module 2 dans le micro-ordinateur, et, d'autre part, sous système d'exploitation SYNTRON (voir point II.4.B) pour l'écriture des utilitaires. La deuxième phase a consisté au brûlage du logiciel dans l'EPROM du micro-ordinateur car il n'existe pas encore de système d'exploitation temps réel disponible pour un micro-ordinateur développé autour du bus G64. Pour plus de détails, il est conseillé de se référer dans l'annexe au document A-2

"remote diagnosis : rapport final" écrit en fin de stage.

VI.4.A. Le module numéro 1

La réalisation de l'interface du micro-ordinateur avec l'interface d'un GP* est revenue à la programmation du gérant d'une carte GESSIO-1A de Gespac. Cette carte procure deux canaux de communication série, permettant donc le dialogue avec deux GP*, en mode asynchrone et au taux de 9600 bps (voir figure VI.3.).

Un micro-ordinateur contiendra autant de cartes GESSIO-1A que de paires de GP* présents dans un Bâtiment Auxiliaire. Il a donc fallu définir une table de branchement au sein du micro-ordinateur pour que suivant le numéro de séquence du GP* référencé dans le Data Module, on connaisse le numéro de canal c'est-à-dire les adresses des registres de l'interface par lesquels dialoguer avec le GP*.

VI.4.B. Le module numéro 2

Ce module déjà implémenté réalise l'interface du micro-ordinateur avec le bus multiplex. Il revient à la programmation de deux cartes MPX-RT dont l'une traite des signaux de communication et l'autre des tampons de données. L'avantage de ces cartes est que si l'on passe d'un bus multiplex tel que défini au SPS à un bus multipoints conforme au protocole MIL-STD-1553B (voir point III.3.D) tel que défini pour le LEP, il suffit de remplacer la première carte de décodage des signaux en utilisant la même carte mémoire-tampon.

Cette remarque n'est pas sans importance quand on sait que le projet sera adapté pour être utilisable dans l'environnement LEP (voir chapitre III).

VI.4.C. Le module numéro 3A

L'appel à un Data Module peut s'effectuer de deux manières :

- soit par le transfert d'un simple mot de données, comme présenté au point II.4.D.3 par l'exemple SET MAGNET (6, CUR) = 101.6 ;
- soit par le transfert de tableaux d'informations ; l'appel est alors du type (AR, "R/W", N-VALUE, *property)
  avec AR :
  un tableau d'entiers ;
  avec R/W :
  le type de l'opération, R pour lecture et W pour écriture ;
  avec N-VALUE :
  le numéro d'équipement que l'opérateur veut contrôler ;
  avec *property :
  une chaîne de trois caractères ASCII donnant le nom de la propriété que l'opérateur veut voir exécuter.

Dans l'acception classique, il aurait fallu coder dans le tableau AR la commande à envoyer vers le GP* ainsi que le numéro d'ordinateur à contrôler (car le micro-ordinateur doit connaître le destinataire de la commande pour calculer le canal avec lequel dialoguer) car seul ce tableau aurait été transmis par le Data Module au micro-ordinateur. De plus, le code des propriétés aurait dû aussi être résident dans l'Acces, impliquant une surcharge pour ce dernier.

Dans le cas actuel, sur base de la pseudo-data table présente dans l'Acces et du numéro d'ordinateur N-VALUE spécifié dans l'appel au Data Module MOPC, qui n'est rien d'autre que le GP* à contrôler, le module 3A aiguillera les différents paramètres, AR, "R/W", *property, N-VALUE vers le bon micro-ordinateur. Nous parlons de pseudo-data table car il s'agit d'un sous-ensemble des informations disponibles dans une Data Table classique comme l'adresse physique sur laquelle doit porter l'E/S ou les données pour tester le caractère licite de l'appel. Le micro contient le véritable gérant aux appels Data Module (c'est-à-dire le module 3B) car c'est lui qui traite des propriétés. Il n'y a pas non plus de Property Table définie dans l'Acces car celle-ci

est résidente dans le micro-ordinateur avec le code des propriétés. Nous rappelons que la Property Table associe à chaque entrée les informations concernant une propriété comme l'adresse d'emplacement du code la réalisant.

VI.4.D. Le module numéro 3B

Suite à l'interruption générée en son sein par l'appel au Data Module MOPC par l'opérateur connecté à l'Acces, le micro analyse le type de la propriété demandée et, sur base de la table des propriétés, se branche à l'adresse du code correspondant. Spécifions maintenant ces différentes propriétés définies au niveau d'un appel Data Module.

==========

VI.4.D.1. Les propriétés relatives au micro-ordinateur

==========

Une première propriété ***RSV** permet d'effectuer la réservation manuelle du micro-ordinateur, ce qui implique qu'il est dédicacé à l'usage exclusif de l'opérateur qui l'a réservé. Si une autre personne tente de s'immiscer, sa demande sera rejetée avec indication de la personne actuellement en session de travail avec le micro-ordinateur. Cette précaution permet, au cas où un micro serait abusivement utilisé par un opérateur, d'annuler sa session en indiquant son nom et éviter ainsi des réservations infinies.

Pourquoi cette réservation ?

- Certaines commandes envoyées au MOPC du GP* contrôlé comme la "Memory Dump Instruction" (définie à l'annexe I-A) peuvent être interrompues par le simple envoi d'un caractère alphanumérique. Il faut éviter que, par mégarde, deux opérateurs non conscients de leur existence se gênent.
- Tout appel au Data Module MOPC réserve la liaison CAMAC -

multiplexeur de données - micro-ordinateur (figure VI.3.) pour une durée de 100 msec. Malheureusement, aucun dialogue avec le MOPC ne peut se résoudre dans ces temps vu sa lenteur de réaction. Toute opération avec le MOPC se découpe donc en une partie envoi de la commande et une partie lecture des résultats qui, chacune, doive s'effectuer en 100 msec. Il faut éviter, à tout pris, qu'une personne étrangère ne vienne s'immiscer et le seul moyen est d'imposer une réservation manuelle du micro-ordinateur. On aurait pu réaliser implicitement la réservation lors de l'envoi d'une commande et la relâcher à la fin de la lecture des résultats. Cependant, certaines opérations comme le "Memory Dump Instruction" et la lecture des résultats d'un test émettent des résultats en plusieurs étapes. On ne peut donc savoir, à priori, quand libérer la réservation.

Une deuxième propriété *RLS permet d'annuler la réservation manuelle du micro-ordinateur mais il faut lui indiquer le nom de l'opérateur ayant effectué auparavant la réservation. Cette précaution a été prise pour éviter qu'une personne étrangère n'éjecte un opérateur sans avoir pris connaissance de son existence.

Cette annulation ne peut être rendue effective que s'il n'y a aucun dialogue en cours avec le MOPC du GP* contrôlé c'est-à-dire aucune "Memory Dump Instruction", aucune opération de test ou aucune opération élémentaire d'envoi de commande/lecture des résultats.

Une troisième propriété *TBR permet de définir la table de branchement (voir point VI.4.A) où chaque entrée représentant un canal d'une carte GESSIO-1A est associée à un numéro d'ordinateur ou de GP* présent dans un Bâtiment Auxiliaire.

En écriture, cette propriété permet d'allouer un numéro d'ordinateur à un canal et en lecture, de se remémorer le numéro de canal associé à un numéro d'ordinateur.

Une quatrième propriété *NOTIMP avertit l'opérateur de l'appel à une propriété non implémentée que ce soit dans le sens lecture ou dans le sens écriture.

VI.4.D.2. Propriétés relatives au MOPC

Tout dialogue avec le MOPC se décompose en trois parties :

- l'envoi de la commande au MOPC telle qu'elle aurait pu lui être envoyée en local et suivant la syntaxe définie dans l'annexe A-1 (seule la "Breakpoint Instruction" n'est pas permise) ; il est traité par la propriété *ODR ;
- un temps d'attente ;
- la lecture des résultats par l'Acces et ce par la propriété *TPR.

La première propriété *ODR consiste donc en l'envoi de la commande au MOPC. Cependant, vu la contrainte des 100 msec, nous nous bornons, sous interruption en provenance du bus multiplex, à initialiser le dialogue avec le MOPC avant de rendre la main à l'Acces. Une fois revenu du niveau d'interruption, le micro envoie la commande habillée au MOPC (sorte de travail en arrière-plan) et attend d'en recevoir sous interruption les résultats.

Tant que l'envoi de la commande par le micro n'est pas achevé, il est inutile de lancer des appels Data Module pour lire les résultats car ceux-ci sont traités sous niveau d'interruption et ralentissent le travail d'envoi de la commande par le micro. C'est la raison de la définition d'un temps d'attente.

Une deuxième propriété *TPR consiste à communiquer au module numéro 2 le nombre de bytes utiles à lire et les bytes qui séjournent en attendant dans un tampon cyclique du micro-ordinateur ; ce tampon intermédiaire a été défini car en cours de réception des résultats du MOPC, le micro peut dialoguer, via messages d'erreur et le module 2, avec l'opérateur ; en l'absence de tampon, ces messages viendraient se superposer aux résultats dans le module 2 ; il ne faut donc communiquer les valeurs du tampon au module 2 qu'à la demande explicite de l'opérateur, par la propriété *TPR ; si l'ordre de lecture lancé par l'Acces n'est pas assez rapide, un message "overflow du buffer cyclique" est adressé à l'opérateur. Ce transfert peut s'effectuer sans intervention du micro-ordinateur c'est-à-dire en mode transparent où les caractères émis par le MOPC sont remontés sans distinction vers l'opérateur au niveau de l'Acces. Une autre propriété serait

de communiquer les résultats avec filtration des caractères à effectivement renvoyer vers l'opérateur. Cette propriété non implémentée, faute de temps, reviendrait à un travail en mode non transparent.

Une troisième propriété *STP permet d'arrêter tout dialogue en cours avec le MOPC. Cette opération permet le démarrage d'un nouveau dialogue avec le MOPC et, dans le cas du test, constitue l'unique moyen de remettre le micro-ordinateur dans un état cohérent si l'on veut mettre un terme à l'opération de test. Cette opération ne s'accompagne pas d'une libération automatique de la réservation manuelle du micro-ordinateur car on peut vouloir stopper l'opération en cours sans interrompre sa session d'opérateur.

VI.4.D.3. Propriétés liées au chargement de programmes de test binaires

C'est dans la réalisation de ces propriétés que le logiciel développé au niveau du micro-ordinateur acquiert toute sa valeur car elles permettent, sans devoir surcharger l'Acces, le chargement de programmes de test dans la mémoire du GP* contrôlé ainsi que la lecture à distance des résultats.

Ces propriétés se déroulent en quatre phases.

La première phase ou propriété *INT consiste en la communication au micro-ordinateur des paramètres de test c'est-à-dire le nombre de mots dont est constitué le fichier binaire à charger en mémoire, l'adresse de départ de chargement du programme et l'adresse de lancement d'exécution du test. Ces différentes informations sont nécessaires car pour pouvoir changer le contenu d'une cellule-mémoire par une instruction "Memory Deposit", il faut en avoir lu au préalable le contenu par l'instruction "Examine memory". L'adresse de départ de chargement sert à effectuer l'"Examine Memory" préliminaire. Par la suite, le nombre de mots dont est constitué le fichier binaire communique au module 3B le nombre de "Memory Deposit" qui doivent être effectués, chacun d'entre eux effectuant par défaut la lecture du contenu de la cellule d'adresse suivante. L'adresse de chargement sert au module 3B à envoyer au MOPC l'instruction "Start a program" quand tout le fichier binaire a été chargé en mémoire.

La deuxième phase ou propriété *WRT consiste au chargement proprement dit dans la mémoire du GP* contrôlé des programmes de test stockés en librairie LIB et chargés pour la cause sur l'Acces. Cette propriété qui reçoit en paramètre des tranches du fichier binaire n'initialise, sous interruption, que les opérations à cause de la contrainte des 100 msec. Une fois la main rendue à l'Acces et donc revenu du niveau d'interruption, le micro charge, byte par byte, le contenu des cellules-mémoire du GP* avec les valeurs définies dans le fichier binaire. Cette propriété, ainsi que la précédente, limite la charge de l'Acces car, si elles n'existaient pas, il faudrait effectuer le chargement, byte par byte, à partir de l'Acces par appels successifs à la propriété *ODR et passage dans AR de la valeur d'une cellule-mémoire.

La troisième phase ou phase de lecture des résultats dans le tampon cyclique s'effectue comme dans le cas d'un simple dialogue avec le MOPC par appels consécutifs à la propriété *TPR (voir point VI.4.D.2).

La quatrième phase ou phase d'arrêt s'effectue par appel à la propriété *STP (voir point VI.4.D.2). En effet, le micro-ordinateur ne dispose d'aucun moyen de détecter la fin d'une opération de test. Si le MOPC n'émet plus de résultats :

- soit il attend l'intervention de l'opérateur suite à une question posée dans le programme de test ; pour y répondre, il suffit par appel à la propriété *ODR d'envoyer la réponse soit RUN, soit EXIT, soit HELP, ...
- soit le test est terminé ; pour y réagir, il suffit d'envoyer la propriété *STP qui remet le système dans un état cohérent.

VI.4.E. Le module numéro 4

Les utilitaires écrits en NODAL se résument à un ensemble d'appels au Data Module MOPC avec les propriétés appropriées.

VI.5. Exemple de déroulement d'une session avec le micro-ordinateur à partir de l'Acces

Débutons par une série d'opérations indispensables en début de session :

- MOPC ('(VGD)',"W",---,*RSV) ; cet appel, dans le sens écriture, sert à la réservation manuelle du micro-ordinateur par communication de l'identifiant de l'opérateur dans le tableau AR, soit la chaîne de caractères (VGD) convertie sous forme d'entiers ;
- MOPC ('(VGD)1',"W",N-VALUE,*TBR) ; il est d'abord important de noter que dans tout appel au Data Module et c'est ce qui se passera dans la suite, il faut respécifier, dans les premières positions du tableau AR, son identifiant d'opérateur, pour le cas (VGD), pour permettre la vérification par le micro-ordinateur du caractère licite de l'appel ; cet appel, dans le sens écriture, sert à la constitution de la table de branchement ; dans le cas présent, on associe au canal numéro 1 (numéro défini après l'identifiant dans le tableau AR) le numéro d'équipement spécifié par le N-VALUE ;

Poursuivons par une opération d'envoi de commande/lecture des résultats élémentaire :

- MOPC ('(VGD)1/',"W",N-VALUE, *ODR) ; cet appel, en écriture, a pour effet d'envoyer au MOPC du GP* N-VALUE la commande "Examine Memory" de format xxxxxx/ qui donne le contenu d'une adresse-mémoire dont la valeur en octal, xxxxxx, est spécifiée avant le caractère '/' ; il s'agit ici de la commande '1/' spécifiée dans AR après l'identifiant de l'opérateur ;
- MOPC ('(VGD)',"R",N-VALUE, *TPR) ; cet appel, en lecture, a pour effet l'envoi par le micro-ordinateur dans le tableau d'entiers AR de la valeur de la cellule d'adresse 1 ; le contenu d'AR peut être affiché sur le terminal de l'opérateur connecté à l'Acces par exécution de l'instruction NODAL "TYPE AR" ;

Poursuivons par une opération de test :

- MOPC ('(VGD)' "80" "0" "0","W",N-VALUE,*INT) ; cet appel est nécessaire pour initialiser l'opération de test ; après indication de l'identifiant '(VGD)' converti en entiers dans AR, l'entier "80" désigne le nombre de mots à charger dans la mémoire du GP*, l'entier "0" l'adresse de début du chargement et l'entier "0" l'adresse de lancement de l'exécution ;
- MOPC ('(VGD)' "1" "2" "3" "4" "5" "6" "7" ...,"W",---,*WRT) ; cet appel communique la tranche représentée sous forme d'entiers "1" "2" "3" "4" "5" "6" "7" ... du fichier binaire à charger en mémoire ;

- MOPC ('(VGD)',"R",N-VALUE,*TPR) ; cet appel sert à la lecture des résultats fournis par le MOPC dans le tableau AR ;
- MOPC ('(VGD)'...','W",---,*ODR) ; cet appel sert à l'envoi dans le tableau AR de réponses au programme de test du type RUN ou EXIT ;

- MOPC ('(VGD)',"W",---,*STP) ; cet appel permet de mettre un terme à l'opération de test ;

Concluons par une opération indispensable en fin de session :

- MOPC ('(VGD)',"W",---,*RLS) ; cet appel met fin à la session de dialogue avec le micro-ordinateur.

VI.6. Evolution

Le module 3B doit d'abord être repris par un ingénieur de la section pour y ajouter d'autres propriétés comme le mode non transparent de la propriété *TPR et achever celles qui n'ont pu être complètement implémentées, faute de temps : il s'agit des propriétés *INT et *WRT. En effet, ces propriétés tournent dans leur version primitive c'est-à-dire qu'actuellement, des programmes de test sont chargés par appels successifs, depuis l'Acces, au Data Module MOPC avec la propriété *ODR. Une fois, le fichier totalement communiqué, l'appel à la propriété *INT à laquelle on donne uniquement

l'adresse de lancement de l'exécution du programme de test provoque l'envoi par le module 3B de l'instruction "Start a program" au MOPC. Les versions telles que spécifiées plus haut ont été écrites mais n'ont pu être testées. Il faudrait aussi songer à la construction d'une bibliothèque d'utilitaires divers.

Dans un avenir plus lointain, la section compte coupler au micro-ordinateur un disque Winchester sur lequel seraient stockés les différents programmes de test binaires ; cela éviterait de devoir emprunter la liaison librairie LIB - Acces - CAMAC - multiplexeur de données - micro-ordinateur pour charger des programmes de test.

La section a été très heureuse des résultats obtenus après six mois de travail. Après discussions avec d'autres groupes, il est envisagé d'employer cette liaison soutenue par logiciel, moyennant quelques modifications, dans l'environnement LEP. En effet, comme l'accélérateur couvrira une circonférence de 27 km, la nécessité du télé-diagnostic se fait encore plus sentir que pour le SPS qui n'en couvrait que 7.

Peut-être s'agit-il là de l'occasion d'un nouveau stage ?

Bibliographie de la partie III

Chapitre VI

(1) J. Altaber, F. Beck, M. Collins, M.C. Crowley-Milling, 18 avril 1978, "Expérience avec un système de multi-ordinateurs pour le pilotage en temps réel d'un processus distribué", SPS/ACC/JA/Rapport 78-11.

(2) M.C. Crowley-Milling, 22 December 1975, "The Design of the Control System for the SPS", CERN 75-20 Laboratory II Control Group.

(3) *6809 FLEX Operating System*, Technical Systems Consultants, Inc.

(4) *NORD-100 Reference Manual.*

(5) *Omegasoft 6809 Pascal Language Handbook*, Certified Software Corporation.

(6) *Omegasoft 6809 Relocatable Assembler and Linking Loader*, Certified Software Corporation.

(7) *The Selection of Local Area Computer Networks*, November 1982, NBS Special Publication 500-96, pages 10-17, U.S. Government work.

EN GUISE DE CONCLUSION ...

Notre étude nous a permis de nous sensibiliser, d'une part, aux exigences des systèmes temps réel et, d'autre part, aux moyens de rencontrer ces exigences dans les réseaux locaux. En effet, ce mémoire avait pour but, après présentation des caractéristiques des systèmes temps réel, d'analyser l'impact des protocoles d'accès mis en oeuvre dans les réseaux locaux sur le temps de réponse et de présenter notre contribution à l'amélioration de la fiabilité d'un réseau local, pour le cas le réseau de contrôle de l'accélérateur SPS.

Nous ne pouvons dans ces lignes que paraphraser les différentes conclusions qui ont déjà été tirées aux termes des parties II et III.

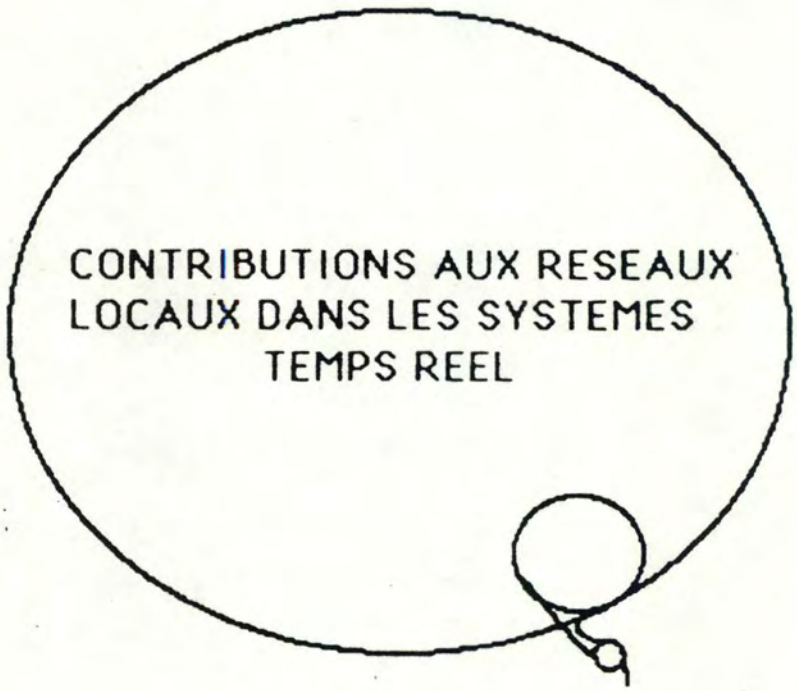
Pour ce qui est de la partie II, il est frappant de constater les balbutiements de la standardisation dans le domaine de l'application des réseaux locaux aux systèmes temps réel et la pauvreté de l'état de la littérature sur la question : nous assistons à l'éveil d'une question scientifique.... Notre étude poursuivie dans la partie II peut se vanter d'être une des seules menées dans le domaine. En bref, notre travail est la première pierre posée à la construction d'un édifice. Les modèles proposés pour le temps d'accès pourraient être élargis pour rendre compte des autres composants du temps de réponse comme le temps d'attente, le temps de transmission et le temps de propagation. De même, les protocoles pourraient être étudiés sous l'aspect du traitement des erreurs Comment l'anneau logique du "Token Bus" est-il géré ? Quel en est l'impact sur le temps de réponse ? Comment la perte ou la duplication du jeton sur le "Token Ring" est-elle gérée ? Quel en est également l'impact sur le temps de réponse ?

Pour ce qui est de la partie III, notre contribution à l'amélioration de la fiabilité du système informatique contrôlant le SPS a amené les concepteurs du LEP à la prise de conscience que de telles démarches pouvaient également réussir et être appliquées dans l'environnement du LEP. Il ne faut pas non plus oublier que les modules réalisés ne sont pas une fin en soi et que de nombreuses extensions doivent encore y être apportées pour obtenir un système de télé-diagnostic complet. Cependant, la question de leur application dans un système de télé-diagnostic des ordinateurs du LEP est actuellement sérieusement en discussion car, vu les distances couvertes, un gain de temps considérable serait obtenu si de tels mécanismes pouvaient être mis en place. Il faudrait alors sérieusement envisager le problème de l'adaptation des logiciels pour leur fonctionnement dans une nouvelle

infrastructure. Peut-être serait-ce l'occasion de donner à un(e) autre étudiant(e) la chance de faire connaissance avec les fameuses installations du CERN ?

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR
Institut d'informatique

CONTRIBUTIONS AUX RESEAUX LOCAUX DANS LES SYSTEMES TEMPS REEL



ANNEXES

Directeur du mémoire :

Ph. Van Bastelaer

Mémoire présenté par

Joëlle Vangeersdael

en vue de l'obtention
du grade de
licenciée et maître
en Informatique

Année 1984-85

ANNEXES

Pour être plus précis, ces annexes se rapportent à la partie III du mémoire et sont données en guise d'informations complémentaires pour aider à la compréhension de l'objet de notre stage.

L'annexe I donne les spécifications du programme assurant la communication de l'opérateur avec un mini-ordinateur NORD-100 de NORSK DATA : le Microprogrammed Operator's Communication.

L'annexe II donne le manuel de maintenance matérielle et logicielle réalisé par nos soins et destiné à l'ingénieur désigné pour la poursuite de la construction du système de télé-diagnostic. Un exemplaire des programmes dont il y fait mention seront disponibles lors de la défense publique du mémoire. Ils incluent la version tournant sous système FLEX, la version brûlée en EPROM et la version contenant le code complet des propriétés *INT et *WRT conformes aux spécifications du point VI.4.D.3.

Annexe I

4 OPERATOR'S INTERACTION

4.1 *CONTROL PANEL PUSH BUTTONS*

When the panel key is unlocked, the panel push buttons are active and have the following effect:

MCL This is the MASTER CLEAR button used to force the computer system into a defined initialized state. First, the red and green indicator lamps on the CPU board will light up. Then the microprogram is forced to execute the master clear routine. This will also be executed when the MACL command is given to MOPC (refer to Section 4.2.2.1.1), when the CPU goes through the power up sequence, or when the bus line called MCL is activated by an interface.

The master clear routine turns off the green indicator lamp, then the PIE register is cleared. The paging and interrupt systems are turned off. The paging system is set in REX mode. Subsequent memory examine functions with MOPC are set to 24 bit physical examine mode. The CPU self test microprogram is executed. If no errors are found, the green indicator lamp is lit, and the terminal interface on the CPU board (the MOPC terminal) is initialized to receive and transmit 7 bits and even parity. Parity is not checked by MOPC on input. An interrupt level change to level 0 is then executed. After this the CPU will be in stop mode.

STOP This push button has the same effect as giving the STOP command to MOPC. The CPU will enter stop mode and MOPC will be active.

LOAD This push button has the same effect as writing **$** or & to MOPC. Its exact effect is determined by the setting of the ALD thumb-wheel switch on the CPU board.

OPCOM OPCOM is always operative in stop mode. When the machine is running, pressing this button will allow the operator to use the CPU board terminal for operator communication. When the CPU is running, it will enable MOPC to read input from the terminal interface located on the CPU board. It will also inhibit input interrupts from this terminal, and disable the transfer of data from the terminal interface to any macro program (main memory program). The terminal interface will be in this state until the escape character is typed, or the CPU is stopped and restarted.

When MOPC is entered a # is printed at the beginning of each line.

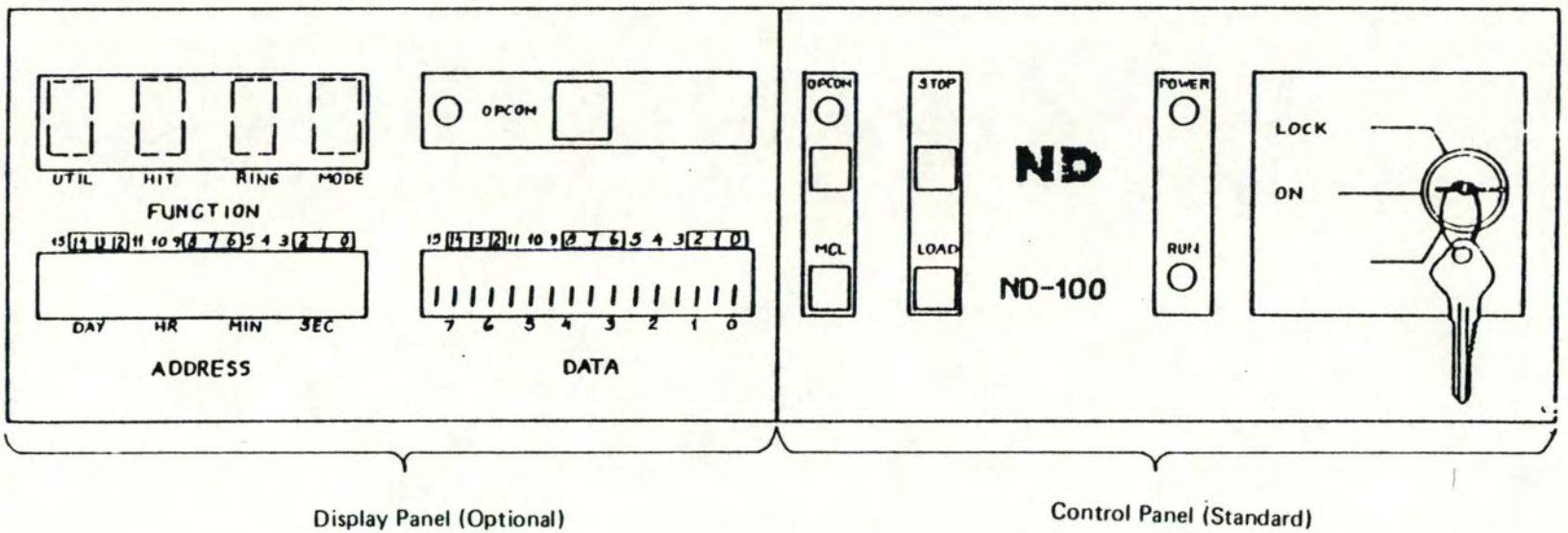


Figure 4.1: ND-100 Front Panel

4.1.1 *The Panel Lock Key*

The Panel Lock Key has three positions:

1. LOCK

   When placed in this position, the operator's panel control switches are disabled. This is the normal position for an operating machine. Main power is applied to the computer.

   Note: Automatic restart may be initiated after power failure only if the lock key is switched in this position.

2. ON

   In this position the panel switches can be operated. Main power is applied to the computer.

3. STAND-BY

   In this position the main power is disabled. Stand-by voltage is applied to memory and display. This position will not be present (or valid) on machines delivered from January 1980.

4.1.2 *Status Indicators*

POWER ON

Indicates that +5V is present in the rack.

RUN

Indicates that the CPU is running.

OPCOM — Operator Communication.

Indicates that the operators communication microprogram is running. This light may also be lit in RUN mode by pressing the OPCOM button. (OPCOM and RUN are lit at the same time). The OPCOM light will always be lit when the computer is not running.

Note: When OPCOM and RUN are lit at the same time, input from the console terminal will only interact with the OPCOM microprogram. Output to console may come from OPCOM or the active program.

4.2 *MICROPROGRAMMED OPERATOR'S COMMUNICATION*

4.2.1 *General Considerations*

The ND-100 has a microprogram in the read only memory for communication between the operator and the machine. This program is called MOPC (Microprogrammed Operator's Communication) and is used for operational control of the ND-100. It includes such functions as memory and register examine and deposit, breakpoint control, bootstrap loading, etc.

Whenever entered, MOPC will perform the necessary communication with the terminal connected to the current loop interface on the CPU printed circuit board. This terminal will be shared as output device between MOPC and other possible programs. As input device MOPC will receive input from the terminal as long as the OPCOM lamp on the operator's panel is lit.

MOPC will never wait if the terminal is not ready for the transmission of characters. Instead, it will start executing the STOP routine or the running program. MOPC will then be dormant until next time it is entered, and continue with the tasks it had to postpone. The maximum time spent in MOPC is 20 μs. If MOPC does not have any activity to sustain on the terminal, it will use 6 μs every time it is entered.

The ND-100 operator's communication includes bootstrap programs and automatic hardware load from both character oriented devices and mass storage devices.

When communicating with the MOPC program, the following characters are legal input characters:

Characters legal in STOP or RUN:

Character:	*Use:*
0 - 7	Octal digits used to specify addresses and data.
A - Y	Letters used to specify commands and register names. Letters typed in succession are acted upon when CR (carriage return) or / is typed. Different letter combinations may have the same effect because of a scrambling algorithm used to pack the letters.

@ or (space)	All characters written before this character are ignored (break character).
<	Used to separate lower and upper bounds in dump commands.
/	Specifies memory or register examine.
↵ (carriage return)	Ends a line. Used to terminate commands or to perform a register or memory deposit function.
*	This character will cause the address of the last examined memory address to be printed.
"escape"	Terminates the communcation between the CPU board terminal and MOPC. This character has no effect if the CPU is in STOP mode.

Characters only legal in STOP:

Character:	*Use:*
!	Start program in main memory command.
Z	Single instruction command.
& or $	Bootstrap load command.
.	Breakpoint command.
"	Manual instruction command.
#	Start microprogrammed memory test.

All other characters are answered with a ?, and characters written before the erroneous character will be forgotten (as if "space" had been typed).

4.2.2 *Control Functions (Does not affect display)*

4.2.2.1 System Control

4.2.2.1.1 MASTER CLEAR

When MACL ↓ is written to MOPC, the CPU microprogram will execute the master clear routine. The effect of this routine is described in the section on Panel Pushbuttons - 4.1.

4.2.2.1.2 STOP

When STOP ↓ is written to MOPC, the CPU will stop execution of the program in main memory. No level change will be performed and program execution can be continued by typing the exclamation mark character.

4.2.2.1.3 ALD LOAD

In the following table the different columns signify:

ALD — Setting of the ALD thumbwheel switch on the CPU module.

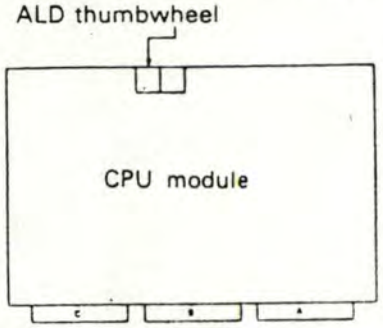
I12 — Corresponding value of the internal register number 12.

POW OK — Indicates the action performed when the panel key is locked and power comes on (or hardware master clear is finished), and standby power has been on all the time since power last went off.

POW NOK — Indicates the action performed when the panel key is locked and power comes on (or hardware master clear is finished), and standby power has been missing for some time since power last went off.

LOAD — Indicates the action performed if the load button is pressed, or $ & is written to MOPC.

ALD	I12	STB POW OK	STB POW NOK	LOAD
15	0	Start in address 20	Stop	Nothing
14	1560	Start in address 20	Binary load from 1560	Binary load from 1560
13	20500	Start in address 20	Mass storage load from 500	Mass storage load from 50
12	21540	Start in address 20	Mass storage load from 1540	Mass storage load from 15
11	400	Start in address 20	Binary load from 400	Binary load from 400
10	1600	Start in address 20	Binary load from 1600	Binary load from 1600
9		Start in address 20		
8		Start in address 20		
7	100000	Stop	Stop	Nothing
6	101560	Binary load from 1560	Binary load from 1560	Binary load from 1560
5	120500	Mass storage from 500	Mass storage load from 500	Mass storage load from 50
4	121540	Mass storage from 1540	Mass storage load from 1540	Mass storage load from 15
3	100400	Binary load from 400	Binary load from 400	Binary load from 400
2	101600	Binary load from 1600	Binary load from 1600	Binary load from 1600



position of the ALD thumberwheel on the CPU module

4.2.2.1.4 GENERAL LOAD

Binary load is started by typing:

<physical device address> & or <physical device address> $

Loading will take place from the specified device. This device must conform with the programming specifications of either Teletype or tape reader. The device address is the lowest address associated with the device. Binary load will be performed if & or $ is written (or the LOAD button is pressed) and the switch selected ALD has bit 13 equal to "0".

4.2.2.1.5 LEAVE MOPC

ESCAPE

If the ESCAPE key is pressed and the CPU is running, MOPC will be left, and subsequent input from the terminal will be routed to main memory programs. MOPC will be entered again by pushing the OPCOM button on the panel or by executing the instruction 150400 (OPCOM).

4.2.2.2 Program Execution

4.2.2.2.1 START PROGRAM

Format:

xxxxxx !

The machine is started in the address given by the octal number. The address will be physical or virtual depending on whether the paging system is on or off.

4.2.2.2.2 CONTINUE PROGRAM

!

If the octal number is omitted, the P register is used as start address, i.e., this is a "continue function". The program level will be the same as when the computer was stopped (if Master Clear has not been pushed or the MACL command typed).

4.2.2.2.3 SINGLE INSTRUCTION

xxxxxxZ

A single Z character will cause one main memory instruction (or one interrupt level change) to be executed. If an octal argument is specified, the specified number of instructions are executed, after which stop mode is entered again. Page faults, protect violations and interrupt level changes are executed correctly, but are counted as extra instructions. An extra overhead of approximately 3 μs is introduced between each instruction when the CPU is in this semi-RUN mode.

4.2.2.2.4 INSTRUCTION BREAKPOINT

xxxxxx.

This command starts execution in the same semi-RUN mode as described in Section 4.2.2.2.3. When the program address xxxxxx is reached, execution stops before that address is executed, and a "." is printed. If the specific address is never reached, the semi-RUN mode continues until a character other than 0-7 or A-Y is typed.

4.2.2.2.5 MANUAL INSTRUCTION

xxxxxx"

This command starts continuous execution of the instruction specified as argument. The execution stops when a character other than 0-7 or A-Y is typed.

Example:

150410" is an easy way to turn on the paging system.

4.2.2.2.6 SINGLE I/O INSTRUCTION FUNCTION

xxxxxxIO/

This function exeuctes an IOX instruction with xxxxxx as device number. The output data is taken from the operator's register OPR (see Section 4.2.3.2.5). Returned data is printed after the slash and not stored anywhere. No working registers are affected.

4.2.2.3 Miscellaneous Functions

4.2.2.3.1 INTERNAL MEMORY TEST

xxx#

When the # character is typed, memory test of the addresses between the B register (lower limit) and the X register (upper limit) is performed in segment xxx. If the test is successful, # is typed when finished. If the test is unsuccessful, ? is typed and the test stops at the failing address. The registers then contain the following information:

T: Failing bits
P: Failing address
D: Error pattern
L: Test pattern
B: Start address
X: Stop address

4.2.2.3.2 DELETE ENTRY

When @ or (space) is typed, all characters written before this character are ignored.

4.2.2.3.3 CURRENT LOCATION COUNTER

*

When * is typed, an octal number is printed indicating the current physical or virtual address on which a memory examine or memory deposit will take place. The current location counter is set by the examine command /, and it is incremented for each time carriage return is typed afterwards.

4.2.3 *Monitor Functions (Also shown on Display)*

4.2.3.1 Memory Functions

4.2.3.1.1 PHYSICAL EXAMINE MODE

E↵

Subsequent examine will be in physical memory with a 24 bit address. Default mode after master clear.

4.2.3.1.2 VIRTUAL EXAMINE MODE

nE↵

This command will change the xamine mode for subsequent memory examine functions. n is in the range 0-3 and specifies the page table via which the examine address shall be mapped. Page fault and memory protect violation are ignored and physical page 0 used instead.

4.2.3.1.3 MEMORY EXAMINE

Format:

xxxxxx /

The octal number before the character "/" specifies the memory address.

When the "/" is typed, the contents of the specified memory cell are printed out as an octal number.

If a ↓ (carriage return) is given, the contents of the next memory cell are printed out.

If the paging system is used, examine mode may be selected by an E command (see Section 4.2.3.1.1 and 4.2.3.1.2). If virtual examine is specified page faults and protect violations are ignored. In this case, <octal number> specifies a virtual address. If physical examine is specified, <octal number> may contain up to 24 bits of physical address.

Example:

```
717/003456                  % Examine address 717

717/003456 ↓                % Examine address 717
003450 ↓                    % and 720
000013                      % and 721
```

4.2.3.1.4 MEMORY DEPOSIT

Format:

xxxxxx ↓

After a memory examine, the contents of the memory cell may be changed by typing an octal number terminated by CR. If the CPU is running, "DEP" must be written between the number and CR.

Example:

```
717/003456   3475 ↓      % The contents of
003450   1700 ↓          % address 717 is changed
000123 ↓                 % From 3456 to 3475 and 720
123456                   % is changed from 3450 to 1700.
                         % 721 contains 123 and remains
                         % unchanged.
```

4.2.3.1.5 DEPOSIT RULES

Content is only changed by zzzzzz↙ in STOP mode and by zzzzzzDEP↙ in STOP or RUN mode.

Content is unchanged by ↙ in STOP or RUN mode and zzzzzz↙ in RUN mode (? is answered).

4.2.3.1.6 MEMORY DUMP

xxxxxx < yyyyyy↙

The contents of the memory addresses between xxxxxx and yyyyyy are printed out, with 8 addresses per line. The dump is taken from the 64K area last addressed by a preceding memory examine function. A memory examine function should always be done before a memory dump. The dumping will stop if any key is pressed.

4.2.3.2 Register Functions

4.2.3.2.1 REGISTER EXAMINE

Format:

xx Ry/

The first octal (xx) number specifies the program level (0-17). If this number is omitted, program level zero is assumed.

The second octal number (y) specifies which register to examine on that level. The following codes apply:

0 Status register, bits 0-7
1 D register
2 P register
3 B register
4 L register
5 A register
6 T register
7 X register

After the "/" is typed, the contents of the register are printed out.

Example:

R5/	A register level 0
7R2/	P register level 7

Instead of the notation Ry, it is possible to address registers by their names. The names are single letter names, namely: S, D, P, B, L, A, T, X corresponding to R0-R7 respectively.

4.2.3.2.2 REGISTER DEPOSIT

Format:

xxxxxx↵

After a register examine, the contents of the register may be changed by typing an octal number terminated by CR. If the CPU is running, "DEP" must be written between the number and CR.

Examples:

A/ 123456 54321↵ % Contents of A register on level 0
% is changed to 054321

7P/ 000044 55↵ % Contents of P register on level 7
% is changed to 000055

4.2.3.2.3 REGISTER DUMP – RD

xx < yy RD ↵

The contents of the working registers in register blocks xx to yy are printed out, with one register block per line. The registers are printed in the following order: STS, D, P, B, L, A, T, X.

If only one reegister block should be printed, xx must be equal to yy.

Note the case: <RD↵ dump register block on level 0.

4.2.3.2.4 USER REGISTER – U

U/

The last value written by TRR LMP, is selected as display source.

4.2.3.2.5 OPERATOR PANEL SWITCH REGISTER — OPR

OPR/

This selects a scratch register where a code to be read by TRA OPR can be deposited. Content of OPR can be read and changed from the console.

4.2.3.3 Internal Register Functions

4.2.3.3.1 INTERNAL REGISTER EXAMINE

Format:

I xx /

The octal number (xx) specifies which internal register is examined. The following codes apply:

0	PANS	Operator's Panel Status, used by operator's panel microprogram.
1	STS	Status register.
2	OPR	Operator's panel switch register, simulated by a scratch register.
3	PGS	Paging status register
4	PVL	Previous program level
5	IIC	Internal interrupt code
6	PID	Priority interrupt detect
7	PIE	Priority interrupt enable
10	CSR	Cache status register, for maintenance only.
11	ACTL	Current level, decoded.
12	ALD	Automatic load descriptor
13	PES	Memory error status
14	PGC	Paging control register. The examined register belongs to the program level controlled by bits 3-6 of the A register.
15	PEA	Memory error address
16	Spare	Do not use.
17	Spare	Do not use.

4.2.3.3.2 INTERNAL REGISTER DEPOSIT

Format:

xxxxx /

After an internal register examine the contents of the internal register with the same internal register code may be changed by typing an octal number terminated by CR. If the CPU is running, "DEP" must be written between the number and CR. For deposit, the following internal register codes apply:

0	PANC	Operator's panel control, used by operator's panel microprogram.
1	STS	Status register. Only bits 0-7 will be changed.
2	LMP	Writes into a scratch register that may be displayed by writing U/ to MOPC.
3	PCR	Paging control register.
4	Spare	Do not use.
5	IIE	Internal interrupt enable.
6	PID	Priority interrupt detect.
7	PIE	Priority interrupt enable.
10	CCL	Cache Clear.
11	LCIL	Lower cache inhibit limit register.
12	UCILR	Upper cache inhibit limit register.
13	Spare	Do not use.
14	Spare	Do not use.
15	ECCR	Error correction control register.
16	Spare	Do not use.
17	Spare	Do not use.

Examples:

```
I7/ 030013  0/          % Examine PIE and change to 000000
I12/ 021540  20044/     % Examine ALD and change UCILR
                        % to 020044
```

4.2.3.3.3 INTERNAL REGISTER DUMP — IRD

IRD↵

The 16 internal registers are printed out. This function is only allowed when the CPU is in STOP mode. This restriction avoids the unintentional unlocking of PEA, PES and IIC when the CPU is running.

4.2.3.3.4 SCRATCH REGISTER DUMP — RDE

xx < yy RDE↵

The contents of the 8 scratch registers (only microprogram accessible) in the register blocks xx to yy are printed out, with one register block per line. This function is useful for microprogram debugging only.

4.2.4 *Display Functions (Affects only display)*

4.2.4.1 Displayed Format

uuzzyx F↙

This command will define the display format when the optional display unit is included in the system. uuzzyx are octal digits and define the chosen format. F, without argument, (or with argument equal to zero) will set the default display format which is octal format. The parts of the argument have the following effect:

x — Number representation code.

- x = 0 — Displayed data is in octal representation. zz have no effect.
- x = 1 — Displayed data is in unary representation, i.e., 4 of the bits in the displayed data are used to light one out of 16 indicators. zz indicates which 4 bits to decode.
- x = 2 — Displayed data is in binary representation. zz has no effect.

y — Afterglow code.

- y = 0 — No afterglow in display.
- y = 1 — Zeros are stretched.
- y = 2 — Ones are stretched.
- y = 3 — Zeros and ones are stretched.

zz — Lower start bit for binary display.

- zz = $0\text{-}24_8$ — Position of lowest bit position to be represented in binary representation.

uu — Display processor maintenance codes (4 bits).

- uu = 1 — Display year and month.
- uu = 2 — Inhibit message.
- uu = 4 — Initialize panel processor.
- uu = 10 — Abort message.

Example:

1421F↙
After this format specification, bits 14_8 - 17_8 will be shown in unary representation with afterglow on ones.

4.2.4.2 Display Memory Bus

xy BUS/

This command is only useful when the optional display is included in the system. The memory bus is displayed, and depending on the argument xy, various types of bus information can be sampled and displayed. Read from cache is not displayed.

x = 0 = CD	CPU Data is displayed
x = 1 = DD	DMA Data is displayed
x = 2 = CA	CPU Address is displayed
x = 3 = DA	DMA Address is displayed

y = 0	nothing is displayed
y = 1 = R	only read accesses are displayed
y = 2 = W	only write accesses are displayed
y = 3 = WR	both read and write accesses are displayed

Example:

23 BUS/

All addresses sent from the CPU to memory will be displayed in the DATA field and "CAWR" is shown in the FUNCTION field.

4.2.4.3 Display Activity

ACT/

With this display mode active levels (ACT), clock and indicator functions are displayed.

4.2.5 *Bootstrap Loaders*

The ND-100 has bootstrap loaders for both mass storage and character oriented devices. There are two different load formats:

- Binary format load.
- Mass storage load.

Octal load is not implemented in ND-100.

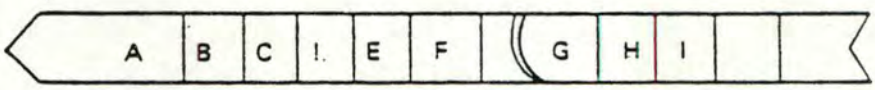
4.2.5.1 Binary Format Load

Binary load is started by typing:

<physical device address> & or <physical device address> **$**

Loading will take place from the specified device. This device must conform with the programming specifications of either Teletype or tape reader. The device address is the lowest address associated with the device. Binary load will be performed if & or **$** is written (or the LOAD button is pressed) and the switch selected ALD has bit 13 equal to "0".

The binary information must obey the following format:



A Any characters not including ! (ASCII 41_8).

B (Optional) octal number (any number of digits) terminated with a CR (line feed is ignored).

C (Optional) octal number terminated with the character ! (see below).

! Indicates start of binary information (ASCII 41_8).

E Block start address. Presented as two bytes (16 bits), most significant byte first.

F Word count. Presented as two bytes (16 bits), most significant byte first (E, F and H are not included in F).

G Binary information. Each word (16 bits) presented as two bytes, most significant byte first.

H Checksum. Presented as two bytes (16 bits), most significant byte first. The checksum is the 16 bit arithmetic sum of all words in G.

I Action code. If I is a blank (zero), then the program is started in the address previously found in the octal number (see above). If I is not a blank, then control is returned to the operator's communication. (The number B will be found in the P register.)

If no device address precedes the & command, then the & is equivalent to pushing the LOAD button on the operator's panel.

If a checksum error is detected, "?" is typed on the console and control is returned to the operator's communication.

Note that the binary loader does not require any of the main memory.

The binary load will change the registers on level 0.

The binary load format is compatible with the format dumped by the)BPUN command in the MAC assembler.

4.2.5.2 Mass Storage Load

Mass storage load is started in the same way as binary format load, except that bit 13 in the device address should be a "1".

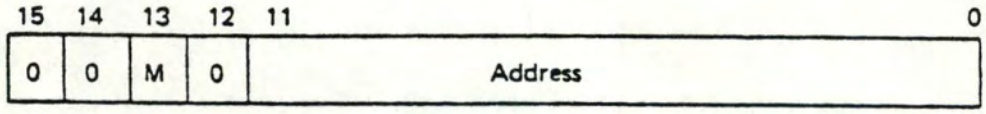
When loading from mass storage, 1K words will be read from mass storage address 0 into main memory starting in address 0. After a successful load, the CPU is started in main memory address 0.

The mass storage device must conform with either drum or disk programming specifications.

4.2.5.3 Automatic Load Descriptor

The ND-100 has a thumbwheel switch called the Automatic Load Descriptor (ALD) (CPU card). This switch selects a 16 bit value to use when the LOAD button is pushed or when a single **$** or & is typed.

The 16 bit value has the following meaning:



M Mass Storage Load

If this bit (bit 13) is 1, mass storage load is taken from the device whose (lowest) address is found in bits 0-10 (unit 0).

If bit 13 is 0, binary load is taken from the device whose (lowest) address is found in bits 0-10.

4.3 *THE DISPLAY*

4.3.1 *General*

The optional display part of the panel is present if the machine has the memory management module installed. This module contains, in addition to the memory management system and cache memory, a display processor. The display processor controls the activity on the display.

There is one button on the display part, the "OPCOM" button. This button allows the operator to use the CPU board terminal for operator communication. This button has the same function as the "OPCOM" button on the operator's panel. The display part of the panel may be placed outside the cabinet (in another room, etc.). Therefore, it is practical to have an "OPCOM" button on this part of the panel.

4.3.2 *The Different Display Functions*

Figure 4.1 shows the normal activity on the display when the machine is running.

The DATA field displays information in binary or octal format (see Section 4.2.4.1). The possible contents are:

- *Active levels (only binary)*

  The active levels in the computer will be shown. There are 16 positions (0-15), one for each level. A one ( ) is set in one of these positions, indicating the active level. The display is provided with afterglow so that it is possible to observe a single instruction on a program level.

- *Register contents.*

  If a register examine is done, the content of the register is shown here.

- *Memory contents.*

  When a memory examine is done, the content of the examined cell will be shown here.

- *Bus information.*

  If the BUS command is given to display memory accesses on the ND-100 bus, the data present on the bus will be shown here and updated continually. When binary format is selected, the address field is used as extension for bit 16-23.

The ADDRESS Field:

— *Calendar clock.*

A clock that tracks the operating system clock is shown here displaying day, hour, minute and second. This clock is adjusted by the "UPDATE" command under SINTRAN III. Under the load procedure this clock will be read by the operating system and taken as system clock. The clock is also connected to the stand-by power and will stay correct even in case of a power failure.

— *Year and month.*

Year and month from the system clock is also shown here by giving the specific F command to MOPC (see Section 4.2.4.1). For example, 1979:10 means October 1979.

— *Current program counter.*

During a register examine, the current program counter is shown here. For example, PC:10153.

— *Memory address.*

If a memory examine is done, the address of the memory location examined is shown here.

The FUNCTION Field:

— *Indicator functions.*

UTIL, utility of the machine, is shown here. That is, how much time the machine spends on level 0 (idle). The more utility, the less the time spent on level 0 and more segments on the display are lit up.

Example:

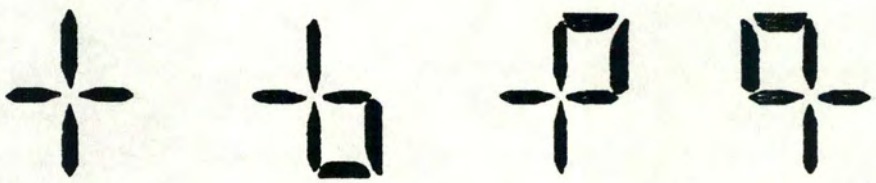
— *No activity.*

HIT, tells the hits rate in cache memory. The higher the cache hit rate , the more segments are lit up on the display.

Example:

— RING, indicates the user ring taken from the PCR.

Example:



Paging off Ring 1 Ring 2 Ring 3

— MODE, tells if the interrupt system and/or the paging system is turned on.
Example:

Both the interrupt system and the paging system is on.

Only the interrupt system is on.

— *Register name.*

If a register examine is done, the name of the register, eventually also the level for the register, is shown.

Example:

5A, OPR, etc.
5A = A register on level 5
OPR = Operator's Register

— *Memory examine mode.*

When a memory examine is done, the examine mode; virtual or physical, will be shown.

Example:

PEXM — physical examine
2EXM — virtual examine mapped through page table 2.

— *Bus examine type.*

What kind of bus information to be sampled and displayed by the BUS command is displayed here.

Example:

DC R — data under a CPU read from memory operation.

Annexe II

REMOTE DIAGNOSIS : RAPPORT FINAL

Par Joëlle Vangeersdael
Technical Student
Periode du 1/8/84 au
31/1/85

1. Bref rappel de l'objet du package

Le but du projet qui m'a amené en ces lieux était de mettre au point une liaison de secours permettant à l'opérateur de transporter à distance sa console de diagnostic d'ordinateur et donc, de pouvoir effectuer à partir de n'importe quel point du réseau SPS un télé-diagnostic des ordinateurs GP* répartis le long de l'accélérateur.

Actuellement, si le système tombe en panne ou qu'il y a un problème avec un GP*, il est possible, dans le meilleur des cas, d'orienter les recherches par l'envoi, de n'importe quel point du réseau, de programmes Nodal stockés en librairie dans le GP supposé malade par des instructions-réseau du type EXECUTE, REMIT et WAIT et d'en attendre les résultats. Ce moyen permet, en général, de réaliser un diagnostic grossier, de se faire une idée du problème, de détecter une corruption ou de contrôler certains indicateurs de la bonne santé du système.

Si le GP est réellement malade ou suspecté l'être ou si la liaison de données empêche tout accès au GP, une liaison de secours devient intéressante par laquelle des programmes de test binaires pourraient être chargés dans le GP en contentieux, des Dumps ou d'autres opérations pourraient être effectuées. Cette liaison permettrait de tester non seulement le GP mais aussi ses interfaces Camac et Liaison de donnees.

Cet autre moyen d'accès aux GPs est l'OPCOM, ce bouton présent sur le panneau avant de chaque NORD-100, et dont l'enclenchement active le programme MOPC (Microprogrammed Operator's Communication) qui :

- assure la communication avec le terminal connecté au port d'E/S Current Loop de la carte CPU
- et accepte des fonctions comme la lecture/écriture de locations-memoire, de registres et le chargement de système d'exploitation.

Mais le problème ne reste qu'à moitié resolu car une possibilité supplémentaire de dialogue en local n'évite pas les ballades en voiture pour se rendre au Batiment Auxiliaire ou se trouve le GP supposé malade et y exécuter en local des programmes de test binaires. Il faut donc déplacer l'OPCOM et mettre au point un moyen de s'y adresser de n'importe quel point du réseau. Pour faire face aux nécessites, au niveau de chaque BA, un microordinateur à microprocesseur 6809 construit autour du bus G64 a dû être programmé pour assurer le dialogue, via le MOPC, avec les différents GP* présents dans le bâtiment. En plus de l'interface-équipement, ce micro doit aussi être interfacé avec le reste du réseau SPS. Il a suffit, pour cela, de trouver une liaison Multiplex suivant les contours de l'accélérateur et dont le G64 serait une station. Quand on sait que l'Acces contrôlant les appareils d'accès aux différents sextants du tunnel a un bus Multiplex d'un tel gabarit et que, de plus, il est accessible de n'importe quel ordinateur du réseau, le

tour est joué.

2. Manuels de référence

- MC6809 - MC6809E Microprocessor Programming Manual, Innovative Systems through silicon MOTOROLA INC.
- NORD-100 Reference Manual
- Sreditor 111 version 1.2 Alford & Associates
- Omegasoft 6809 Pascal Language Handbook Certified Software Corporation
- Omegasoft 6809 Relocatable Assembler and Linking Loader Certified Software Corporation
- 6809 Flex Operating System Technical Systems Consultants, Inc

Conseil : La lecture de l'Omegasoft 6809 Pascal Language Handbook est recommandée pour avoir des connaissances sur le language Pascal.

3. Le hardware

3.1 Configuration générale du système G64

3.1.1 Cartes utilisées avec le système de développement FLEX

Une carte CPU GESSBS-4 de GESPAC
Deux cartes Mémoire Dynamique 32K MCE GGMD2
Une carte Driver du floppy disk P-ISR 7502
Une carte Coupleur Imprimante CENTRONICS GGCI2
Une carte interface GESSIO-1A de GESPAC
Deux cartes interface MPX-RT

Il ne faut pas oublier un boitier d'alimentation de secours vu la forte consommation des cartes MPX-RT.

3.1.2 Cartes utilisées sans système de développement FLEX

Une carte CPU GESSBS-4 de GESPAC
Une carte interface GESSIO-1A de GESPAC
Deux cartes interface MPX-RT

Il est possible que la présence des deux cartes Mémoire Dynamique soit nécessaire dans l'avenir si le package d'application occupait une zone de données de plus de 8KBytes.

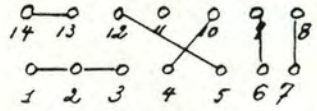
3.2 Carte GESSBS-4 de GESPAC avec processeur 6809

3.2.1 Description de la carte

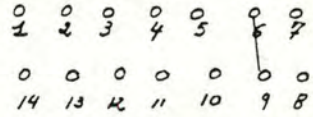
Lire la documentation technique fournie par GESPAC.

3.2.2 Configuration de la carte dans le cas général

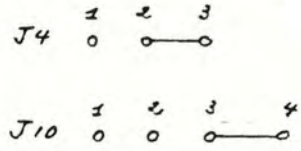
Le positionnement des connecteurs doit s'effectuer comme suit :
- Connecteur RESET P2 : aucun cavalier n'y est positionné;
- Connecteur J2 pour sélectionner la configuration modem (un terminal peut donc être interfacé avec la carte CPU);



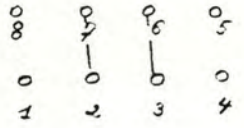
- Connecteur J1 pour sélectionner l'horloge d'émission/ de réception comme étant d'origine interne; le Page signal n'est pas activé;



- Connecteur J3 pour activer le Watch Dog : aucun cavalier n'y est positionné;
- Connecteur J5 pour le sélecteur d'interruption : aucun cavalier n'y est positionné;
- Connecteurs J4 et J10 doivent supprimer la Battery Backup;



- Connecteur J11 pour sélectionner le type des signaux 'E' comme 3-state et 'SYCLK' comme TTL;



3.2.3 Configuration de la carte pour accepter le système de développement FLEX

La configuration-mémoire étant spécifique à FLEX c'est-à-dire qu'il ne s'agit ni de la version standard, ni de la version pour configuration OS9, ni de la version pour configuration GESDOS-09 de GESPAC (Voir figure nr 1), la PAL20L10 U20 étiquetée S8S4 FLEX est donc spécialement programmée pour une découpe de la mémoire spécifique à ce système de développement;
Attention : Cette puce est une 24-broches à mettre convenablement dans un socket à 28 broches.

SINGLE BOARD SYSTEM 6809

The GESSBS-4 Processor module is one of the most complete single euroboard designed around the 6809 for industrial applications. The module can be used alone in a simple project or can take advantage of the G-64 standard bus for further extensions.

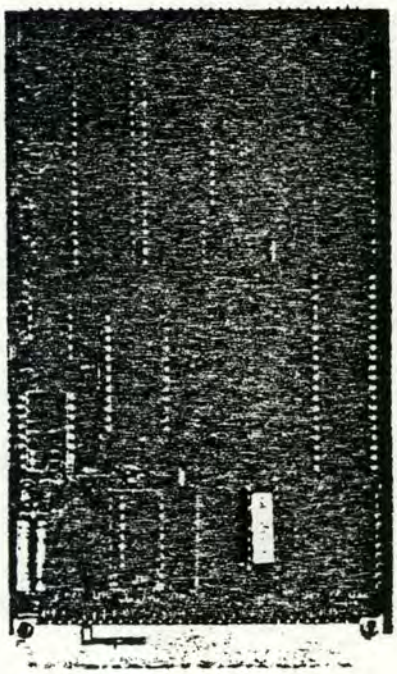
The 6809 Processor can address directly on the board up to 32 Kbytes of EPROM and up to 16 Kbytes of RAM, or any other combination with a large choice of options selectable with jumpers for configurations. In addition the RAM area can take advantage of the CMOS logic with a power fail detect circuitry and standby power supply through the bus or an on-board connector.

For peripheral function the GESSBS-4 offers up to 40 I/O lines at TTL level, 4 16-bit timers and a complete RS 232-C compatible serial interface with programmable Baud Rate clock. Finally a "Watch dog" function can protect the module against lost of program control.

The GESSBS-4 module can address up to 128 Kbytes of memory and is fully compatible with all boards designed for the G-64 standard bus.

Technical features

- 6809 Microprocessor
- 4 MHz quartz controlled oscillator
- Sockets for up to 32 Kbytes EPROM (2732–27256)
- Sockets for up to 16 Kbytes CMOS RAM (with 32 Kbytes EPROM)
- Power fail detect logic and battery option
- RS 232-C compatible serial interface
- Programmable Baud Rate generator
- Up to 40 TTL Bidirectional I/O lines
- 4 interval timers 16-bit resolution
- "Watch dog" circuitry for better program control
- Multiple initialization mode
- Selection of alternate page (128 K addressable)
- Fully compatible with all G-64 Bus modules
- Standard power supply +5V, ± 12V



References

GESSBS-4: Single Board System 6809 with 2 K RAM
GESSBS-4C08: idem, with 8 K RAM
GESSBS-4C16: idem, with 16 K RAM
Other configurations on request

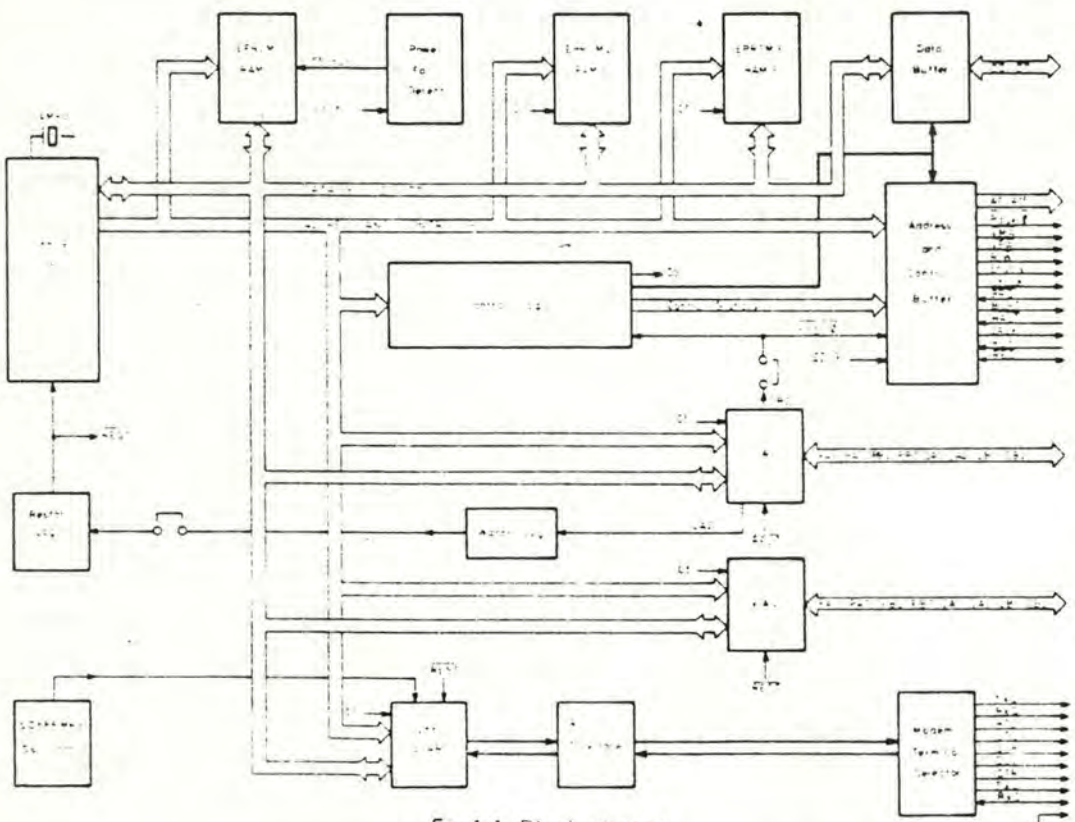


Fig 1.1 Block diagram

1. GENERAL INFORMATIONS

1.1 DESCRIPTION

The GESSBS-4 Euroboard is built around the 6809 microprocessor. The board provides the user with RAM, EPROM, parallel and serial I/O ports.

The board includes 3 x 28-pin sockets that can be supplied with 2K x 8 up to 32K x 8 EPROM or RAM devices. This allows a configuration with up to 96 Kbytes of on board memory. In the standard configuration the GESSBS-4 module has 2 selectable memory maps for the use of either 4-Kbyte EPROM/2-Kbyte RAM or 8-Kbyte EPROM/RAM devices.

Two 6522 VIA peripherals offer 40 parallel I/O lines, 8 control lines and 4 x 16-bit timers. Two of the control lines are optionally used for internal control like "Watch Dog" trigger and page selector (128 Kbyte addressing). A Watch Dog circuit which is selectable on the module allows survey of correct program execution and generates a RESET function in case of error.

1.2 SPECIFICATIONS

Addressing capability	2 pages of 65536 bytes (128K)
Internal memory MAP1	EPROM : 8 Kbytes (2 x 2732) RAM : 2 Kbytes I/O : 192 bytes
External addresses MAP1	Memory : 2 pages of 54272 bytes (106K) I/O : 832 bytes
Internal memory MAP2	EPROM : 16 Kbytes* RAM : 7 Kbytes* I/O : 192 bytes *Other possible configurations: -EPROM 8Kbytes/RAM 15 Kbytes -EPROM 23 Kbytes
External addresses MAP2	Memory : 2 pages of 40960 bytes (80 K) I/O : 832 bytes
Common memory to each page	Internal EPROM/RAM Internal and external I/O
Parallel I/O interface	2 x 6522 VIA (40 I/O lines, 8 control lines, 4x16-bit timers)
Serial communication Interface	Asynchronous RS 232-C compatible, crystal controlled baud rate, programmable from 50 to 19200 bauds
Processor clock	Crystal controlled 4 MHz
Bus interface	-Address and data bus: 3-state TTL compatible -Other signals: TTL compatible Bus drivers: 48 mA type
Power requirement	+12 Vdc = 20 mA typ. -12 Vdc = 15 mA typ. + 5 Vdc = 680 mA typ.* *without memory on the module
Operating temperature	5°C to 55°C
PCB dimensions	100 x 160 mm

Table 1.1 Specifications

An electronic switch protects the content of CMOS memory located in a dedicated 28-pin socket in case of power failure. When memory protection is required it is necessary to connect a battery either on the bus or on the connector preview on the module.

This board includes also an RS 232-C compatible serial communication interface with programmable baud rate.

The GESSBS-4 module is fully compatible with the standard G-64 bus. The block diagram on fig 1.1 illustrates the different parts of the module and their interconnections.

1.3 MEMORY MAP, I/O MAP - STANDARD VERSION

The GESSBS-4 module provides the user with generation of 2 standard memory maps which are jumper selectable. Each memory map is available in the two 64 Kbyte pages which are selectable by the CB2 line of the VIA1 parallel interface device. The 2 standard memory maps are shown on fig 1.2.

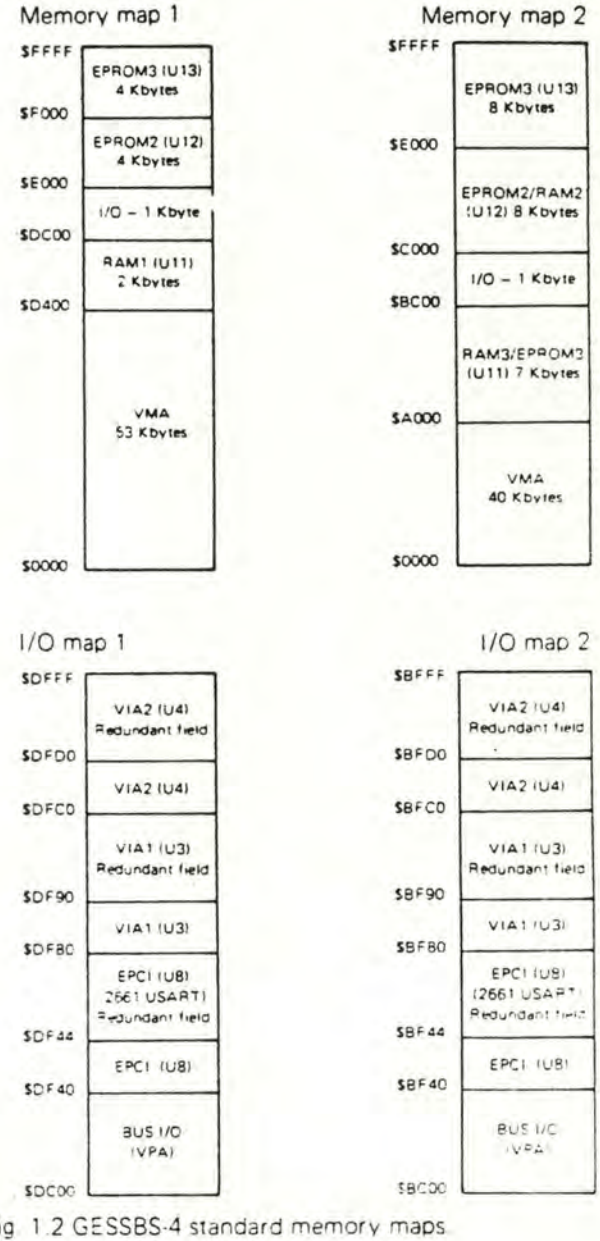


Fig. 1.2 GESSBS-4 standard memory maps

1.4 OPTIONAL MEMORY MAPS – VERSION FOR 0S9® CONFIGURATIONS

The 2 memory maps proposed here are available as an option (special PAL) for dedicated 0S9 configurations. The memory map 3 is adapted for typical development system and memory map 4 is suitable for typical ROM based 0S9 system.
The devices considered in these mappings are 8 Kbyte RAM and 8/16 Kbyte EPROM. If smaller capacity devices are used, they will be redundant in the correspondant decoded field. The 2 memory maps are shown on fig 1.3.

0S9® is a Trademark of Microware System Corporation.

*Redundant field when using 8 Kbyte EPROM devices instead of 16K.

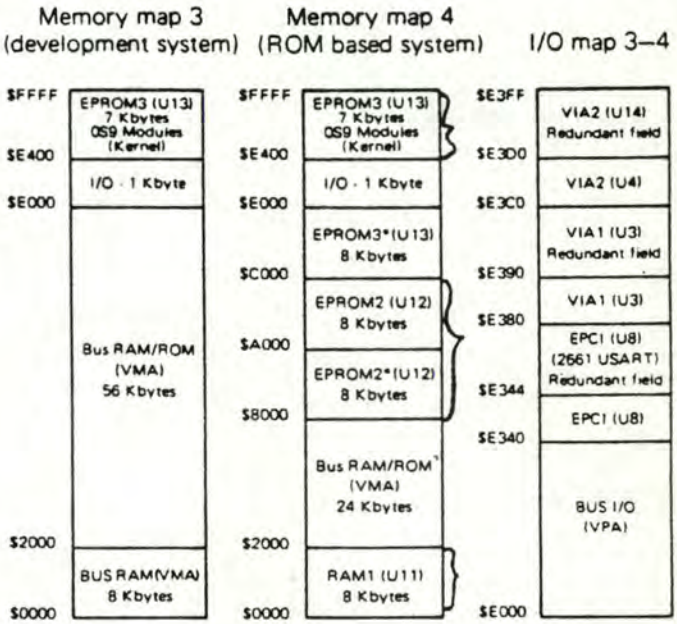


Fig 1.3 GESSBS-4 optional 0S9 configuration memory maps.

1.5 OPTIONAL MEMORY MAP – VERSION FOR GESDOS-09 CONFIGURATION

The memory map proposed here is compatible with the one of the GESMPU-2A module. It is available as an option (Special PAL) and provides only 1 memory map without regards to the jumper selection.
Only 2 Kbyte RAM and EPROM devices can be used in this application as it shown on fig 1.4.

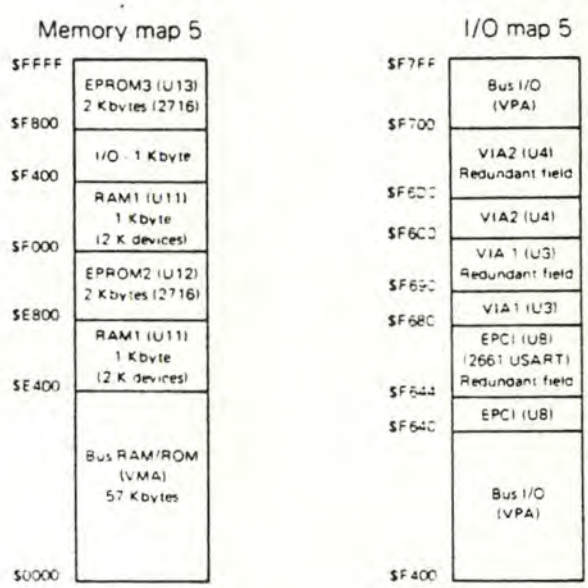


Fig 1.4 GESSBS-4 optional GESDOS-09 configuration memory map

1.6 I/O ADDRESSES

The peripheral devices implemented on the GESSBS-4 have a decoded field which is redundant. A recommanded address field for each device is proposed here and shown for the different memory maps in the table 1.2 below.

I/O map	I/O devices address field			Bus address field-VPA
	USART	VIA1	VIA2	
Map 1 – standard	$DF40 – $DF43	$DF80 – $DF8F	$DFC0 – $DFCF	$DC00 – $DF3F
Map 2 – standard	$BF40 – $BF43	$BF80 – $BF8F	$BFC0 – $BFCF	$BC00 – $BF3F
Map 3/4 – opt. 0S9	$E340 – $E343	$E380 – $E38F	$E3C0 – $E3CF	$E000 – $E33F
Map 5 – opt. GESDOS-09	$F640 – $F643	$F680 – $F68F	$F6C0 – $F6CF	$F400 – $F63F

Table 1.2 Peripheral addresses

1.7 USART – 2661 REGISTERS

The USART registers are described in the table 1.3. Refer to table 1.2 for the device address field.

Address	Write	Read
Base address	Transmit holding register	Receive holding register
Base address +1	SYN1/SYN2/DLE registers	Status register
Base address +2	Mode register 1 and 2	Mode register 1 and 2
Base address +3	Command register	Command register

Table 1.3 USART–2661 registers

1.8 VIA – 6522 REGISTERS

The VIA registers are described in the table 1.4. Refer to table 1.2 for the device address field.

Module address	VIA Ø Register	Description	
		Write	Read
Base address	ORB/IRB	Output Register "B"	Input Register "B"
Base address + 1	ORA/IRA	Output Register "A"	Input Register "A"
Base address + 2	DDRB	Data Direction Register "B"	
Base address + 3	DDRA	Data Direction Register "A"	
Base address + 4	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
Base address + 5	T1C-H	T1 High-Order Counter	
Base address + 6	T1L-L	T1 Low-Order Latches	
Base address + 7	T1L-H	T1 High-Order Latches	–
Base address + 8	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
Base address + 9	T2C-H	T2 High-Order Counter	
Base address + 10	SR	Shift Register	
Base address + 11	ACR	Auxiliary Control Register	
Base address + 12	PCR	Peripheral Control Register	
Base address + 13	IFR	Interrupt Flag Register	
Base address + 14	IER	Interrupt Enable Register	
Base address + 15	ORA/IRA	Same as Reg 1 Except No "Handshake"	

Table 1.4 VIA–6522 Registers

2. PREPARATION FOR USE, INTERCONNECTIONS

2.1 CONNECTORS AND JUMPERS IDENTIFICATION

Table 2.1 identifies the jumpers and connectors of the GESSBS-4 module. Fig 2.1 shows their location on the printed circuit.

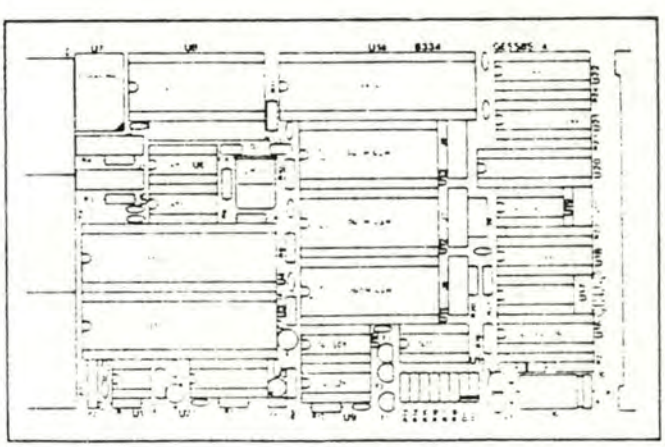


Fig 2.1 Implementation

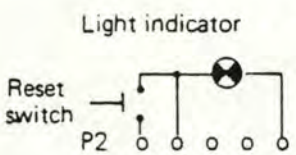
Designation	Function
P1	G-64 Bus interface connector
P2	External Reset connector
P3	VIA1–6522 interface connector
P4	VIA2–6522 interface connector
P5	USART–2661 interface connector
J1	Page signal/clock for sync. transmission selector
J2	Modem/Terminal selector
J3	"Watch-Dog" selector
J4	Vcc/Battery voltage selector for U11 only
J5	VIA and USART interrupt selector
J6	Device type selector for U11
J7	Device type selector for U12
J8	Device type selector for U13
J9	Memory map selector
J10	Vcc/Bus battery selector
J11	Enable/Syclk bus signal, TTL/three-state selector

Table 2.1 Connectors and jumpers identification

2.2 RESET CONNECTOR

Power-on reset is automatically generated on the module. The reset signal on the G-64 bus can be used to reinitialize other modules.

An external switch for reset can be connected to P2 as illustrated on fig 2.2.



P2 pin #	Function
1	Reset
2	Ground
3	Reset
4	+ 5 Vdc
5	+ 5 Vdc

P2 external connection P2-pin assignment

Fig 2.2 External switch connection

2.3 SERIAL COMMUNICATION INTERFACE

2.3.1 GENERAL INFORMATION

The GESSBS-4 module includes a compatible RS 232-C serial interface.

The baud rate is programmed directly in the 2661-USART registers. An oscillator of 5.0688 MHz provides the input clock to the 2661-USART

Interconnections on J2 allow to configure the interface in the modem or terminal mode.

The pinning of P5 corresponds to an RS 232-C 25-pin delta connector type. In this way, connection to a peripheral device can be made with flat ribbon cable.

Standard signals for an RS 232-C serial interface are defined in the table 2.2 and P5 pinning is shown in table 2.3.

Pin No	Signal name	Signal description
1	Protective Ground	Common for the 12 Vdc source
2	Transmit Data (TxD)	Direction: TO modem (DCE) This line transfers data from a terminal to a modem
3	Receive Data (RxD)	Direction: FROM modem This line transfers data from a modem to a terminal
4	Request to Send (RTS)	Direction: TO modem When active (high level) this signal requires a data transfer on TxD from the terminal to the modem
5	Clear to Send (CTS)	Direction: FROM modem When active (high level) this signal indicates that the modem is ready to receive a data transfer on TxD. CTS is the modem answer to RTS
6	Data Set Ready (DSR)	Direction: FROM modem When active (high level) this signal indicates that the modem is connected and ready to receive a command from the terminal
7	Signal Ground	Signal used as the reference potential for unbalanced interchange circuits
8	Data Carrier Detect (DCD)	Direction: FROM modem When active (high level) this signal indicates that the modem is receiving a telephone modulation in the modem appropriate limits
9–14	Not used	
15	RxC	Direction: FROM modem Terminal clock input
16	Not used	
17	TxC	Direction: TO modem Terminal clock output
18,19	Not used	
20	Data Terminal Ready (DTR)	Direction TO modem When active (high level) this signal indicates the modem that the terminal is connected and operational
21–25	Not used	

Note: DCE = Data Communication Equipment
DTE = Data Terminal Equipment

Table 2.2 RS 232-C 25 pin delta connector

RS 232-C pin #	P5 pin # (flat cable)	RS 232-C signal
1	1	GND
2	3	TxD
3	5	RxD
4	7	RTS
5	9	CTS
6	11	DSR
7	13	GND
8	15	DCD
15	4	RxC
17	8	TxC
20	14	DTR

Table 2.3 P5 pinning and RS-232-C equivalence

2.3.2 TERMINAL/MODEM CONFIGURATIONS

Signals defined in the table 2.2 are connected to P5 through the interconnections of J2 illustrated on fig 2.3.

Fig 2.4 illustrates the interconnection of J2 for a standard RS-232-C Terminal or Modem connection to P5.

For special configurations refer to table 2.2 for signal definition and fig 2.3 for their interconnection.

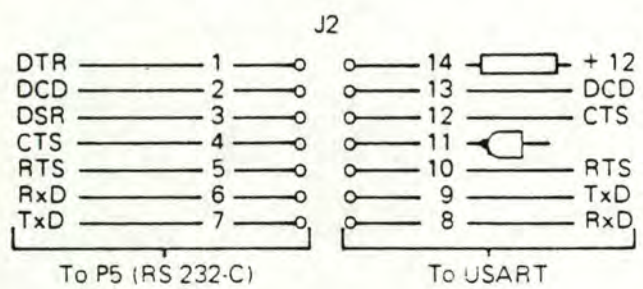


Fig 2.3 J2 connector (modem/terminal)

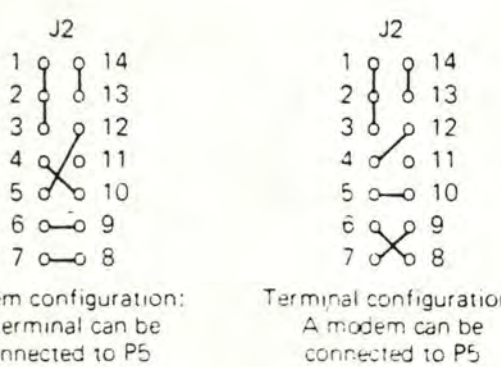


Fig 2.4 Typical configurations

2.3.3 Tx AND Rx CLOCK ORIGIN

Transmit and receive clock can be selected on J1 to be internal or external (synchronous operations)

When internal, the baud rate clock is derived from a 5.0688 MHz oscillator and is programmable on the 2661 USART

The baud rate origin selection arrangement is shown on fig 2.5.

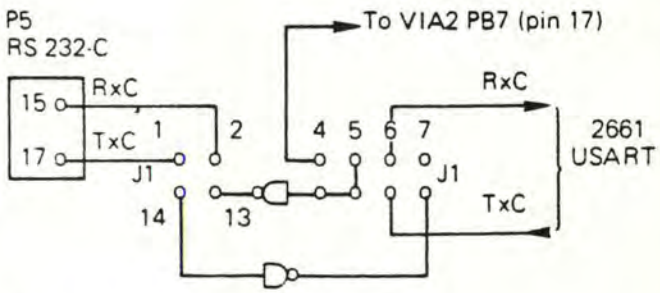


Fig 2.5 Clock selection arrangement

2.4 PAGE SELECTION

The Page signal, if used, is controlled by the CB2 signal of VIA1. The selection of this signal is made by a jumper on J1 as it is shown on fig 2.6. The reason of this selection is to not load the CB2 signal when the Page signal is not used.

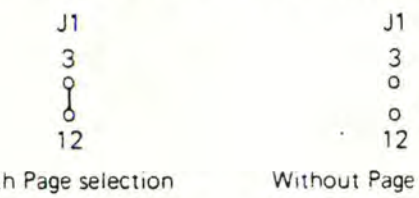


Fig 2.6 Page selection by CB2 of VIA1

2.5 WATCH DOG

The CA2 control line of VIA1 can be connected to the "Watch Dog" circuit by placing a jumper on J3. The "Watch Dog" will be inactive as far as a pulse on CA2 is generated every 200 ms, otherwise the "Watch Dog" output generates a reset on the CPU and on the bus.

Fig 2.7 shows how to connect the "Watch Dog"

Fig 2.7 Watch Dog selection

2.6 INTERRUPT SELECTION

The different interrupts generated by VIA1, VIA2 and the USART are jumper selectable on J5. The interrupts issued from the 2661 USART can be selected individually but they will be wired-or to generate a common interrupt.

The fig 2.8 shows how interrupt selection is made on J5.

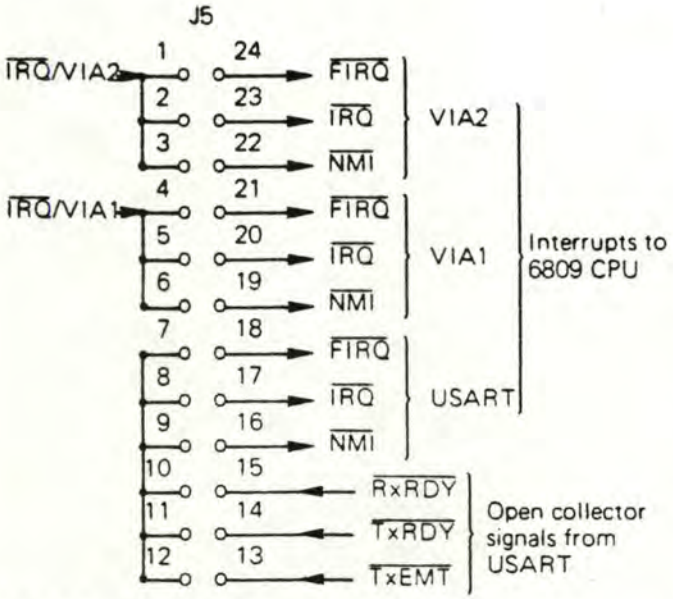


Fig 2.8 Interrupt selection

2.7 MEMORY TYPE SELECTION

The GESSBS-4 module can be supplied with three 24 or 28-pin devices (U11 – U13) which can be individually selected to be RAM or EPROM.

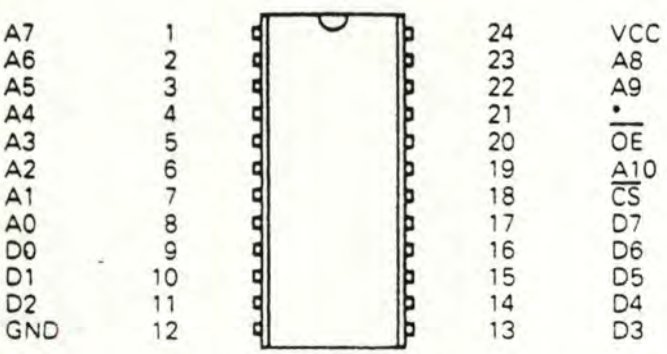
The RAM/EPROM selection exists also on U13 but must be configurated for EPROM since for the proposed memory maps (see 1.3 – 1.5) it is the top address device.

Battery backup is selectable for a CMOS RAM only on the device U11 if required. In this case, a battery must be connected on the battery bus line or on J10 with a connector. U12 can be also selected to be EPROM or RAM but battery-backup will not be supported on this device.

2.7.1 USE OF 24-PIN DEVICES

A 24-pin device must be plugged at the bottom of the 28-pin sockets provided on the GESSBS-4 module.

The pinning of 24-pin compatible devices is shown on fig 2.9 and device type selection on table 2.4. Fig 2.11 illustrates the selected signals (shown for 28-pin socket).



*Pin 21 can be connected to the write signal or to address A11

Fig 2.9 Pinning of 24-pin compatible circuits

2.7.2 USE OF 28-PIN DEVICES

The pinning for 28-pin compatible devices is illustrated on fig 2.10 and device type selection in table 2.4 and fig 2.11 which illustrates the selected signals. Compatible 4 K to 32 Kbytes 28-pin EPROM or RAM can be used with the GESSBS-4 module.

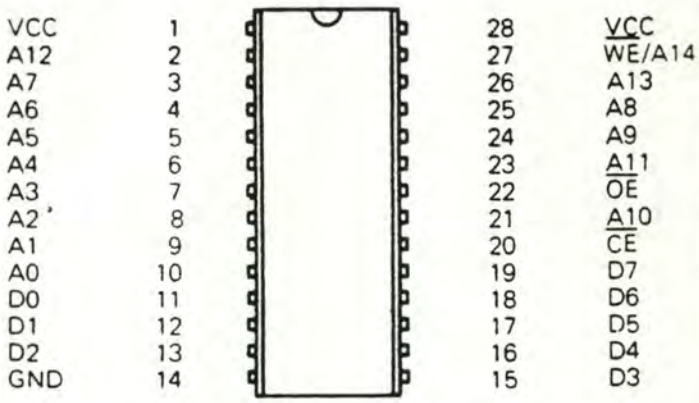


Fig 2.10 Pinning of 28-pin compatible circuits

Device type		Jumper selection
24-pin	2 Kbytes EPROM/RAM	12 11 10 9 8 7 J6 → U11 o o o o o o J7 → U12 o—o o o o o J8 → U13 1 2 3 4 5 6
24-pin	4 Kbytes EPROM (2732)	12 11 10 9 8 7 J6 → U11 o o o o o o J7 → U12 o—o o o o o J8 → U13 1 2 3 4 5 6
24-pin	8 Kbytes EPROM/RAM	12 11 10 9 8 7 J6 → U11 o o—o o o o J7 → U12 o—o o o o o J8 → U13 1 2 3 4 5 6
28-pin	16 Kbytes EPROM/RAM	12 11 10 9 8 7 J6 → U11 o o—o o o o J7 → U12 o o—o o o o J8 → U13 1 2 3 4 5 6
28-pin	32 Kbytes EPROM	12 11 10 9 8 7 J6 → U11 o—o o o o o J7 → U12 o o—o o o o J8 → U13 1 2 3 4 5 6

Note: The selected device size should correspond to the memory map in use (see 1.3 – 1.5).

Table 2.4 Device type selection

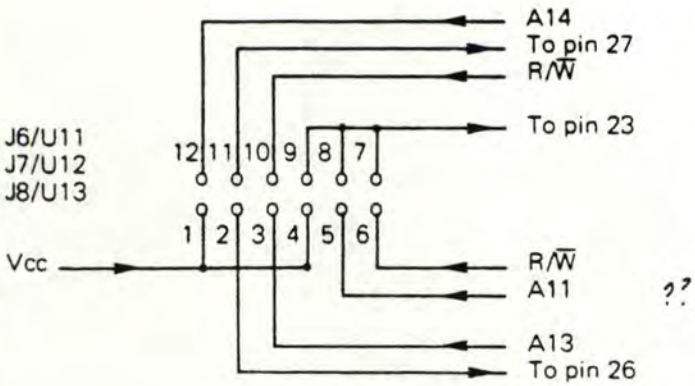


Fig 2.11 Memory signal selection

Memory map	Configuration	J9 selection
1	Standard	8 7 6 5
3	OS9-Development system-Option	1 2 3 4
5	GESDOS-09-Option	
2	Standard	8 7 6 5
4	OS9-ROM based system-Option	1 2 3 4

⚠ Placing simultaneously jumpers in position 1–8 and 2–7 on J9 will cause ***important damage*** to the GESSBS-4 module.

Table 2.5 Memory map selection

2.8 BATTERY BACKUP

There is a selection on J4 and J10 which allows U11 to be powered by a battery during power-fail. Fig 2.12 shows the configuration with and without battery.

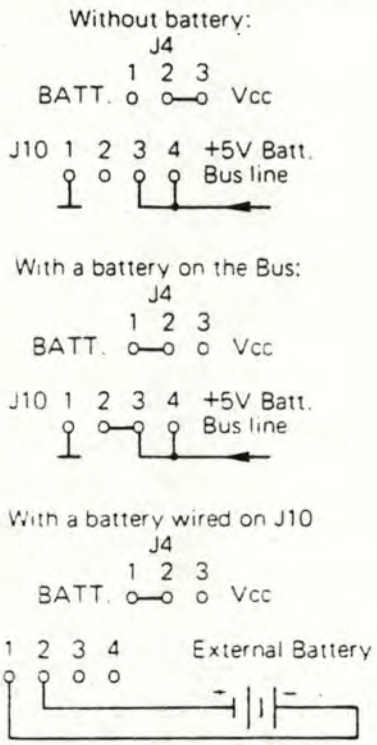


Fig 2.12 Battery backup selection for U11

2.9 MEMORY MAP SELECTION

In the standard version the PAL on GESSBS-4 module generates 2 memory maps (see 1.3) selected by J9. In the optional OS9 configuration there is also 2 memory maps (see 1.4) selected by J9.

In the optional GESDOS-09 there is only 1 memory map.

The table 2.5 shows how to select the desired mapping for the different versions.

2.10 BUS CLOCK

The Enable "E" and SYCLK bus signals can be selected to be TTL or 3-state. In TTL mode the clock will continue to run on the bus during DMA operations while it will be in high impedance in 3-state mode.

The SYCLK has the same frequency and phase relation as the Enable signal.

Mode selection is made by J11 and shown in the table 2.6.

Configuration	J11 selection
E → TTL SYCLK → TTL	8 7 6 5 1 2 3 4
E → TTL SYCLK → 3-state	8 7 6 5 * 1 2 3 4
E → 3-state SYCLK → TTL	8 7 6 5 * 1 2 3 4
E → 3-state SYCLK → 3-state	8 7 6 5 1 2 3 4

* These configurations allow 48 mA operation for each clock on the bus, the other configurations allow only 24 mA for each clock signal

Table 2.6 Bus clock selection

2.11 INPUT/OUTPUT CONNECTORS

Each VIA is connected to a 20-pin connector for flat cable. The configuration of both connectors is identical and described in the table 2.7.

VIA1 is connected to P3 and VIA2 to P4.

P3/P4 pin #	Signal	P3/P4 pin #	Signal
7	PA0	8	PB0
9	PA1	10	PB1
4	PA2	12	PB2
3	PA3	14	PB3
2	PA4	16	PB4
1	PA5	18	PB5
5	PA6	17	PB6
6	PA7	15	PB7
20	CA1	13	CB1
19	CA2	11	CB2

Table 2.7 P3/P4 signal identification

2.12 INTERFACE WITH THE G-64 BUS

The GESSBS-4 module interconnects directly on the G-64 bus. Signals used by the module are identified in the table 2.8. For more information on the bus, refer to the G-64 specification note AN0001E.

ROW B		ROW A		
GND	1	GND		Power (2)
A8	2	A0		Address lines A0 to A15 (16)
A9	3	A1		
A10	4	A2		
A11	5	A3		
A12	6	A4		
A13	7	A5		
A14	8	A6		
A15	9	A7		
$\overline{BRQ}$	10	BGRT		Control lines (18)
$\overline{DS1}$ *	11	$\overline{DS0}$	*	
$\overline{BGACK}$ *	12	$\overline{HALT}$		
Enable	13	SYCLK		
$\overline{RES}$	14	$\overline{VPA}$		
$\overline{NMI}$	15	RDY ($\overline{DTACK}$)	*	
$\overline{IRQ}$	16	$\overline{VMA}$ ($\overline{AS}$)		
$\overline{FIRQ}$	17	R/$\overline{W}$		
IACK	18	Halt Ack		
$\overline{D12}$ *	19	$\overline{D8}$	*	DATA 2nd Byte (8)
$\overline{D13}$ *	20	$\overline{D9}$	*	
$\overline{D14}$ *	21	$\overline{D10}$	*	
$\overline{D15}$ *	22	$\overline{D11}$	*	
$\overline{D4}$	23	$\overline{D0}$		DATA 1st Byte (8)
$\overline{D5}$	24	$\overline{D1}$		
$\overline{D6}$	25	$\overline{D2}$		
$\overline{D7}$	26	$\overline{D3}$		
$\overline{\text{Parity error}}$				Miscellaneous (4)
($\overline{BERR}$) *	27	$\overline{Page}$		
Chain In *	28	Chain Out		
+ 5V Battery	29	− 5V	*	Power (8)
− 12V	30	+ 12V		
+ 5V	31	+ 5V		
GND	32	GND		

*Not used on the GESSBS-4 module

Table 2.8 P1 connector, G-64 bus.

3. SWITCHING CHARACTERISTICS

3.1 GENERAL INFORMATION

Dynamic characteristics that will be defined correspond to the G-64 bus signals. Symbols used are shown on fig 3.1.

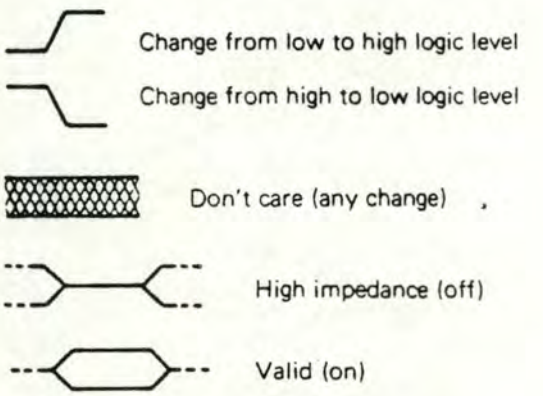


Fig 3.1 Signal symbols

Signal name	Description	Value (ns)		
		Min	Typ.	Max
tE	Cycle time		1000	
tEL	Pulse width low	450		
tEH	Pulse width high	450		
tEAL	Address lead time for memory	200		
tEML	VMA command lead time	170		
tEPL	VPA command lead time	170		
tEWML	Write lead time for memory	200		
tEWPL	Write lead time for peripheral	200		
tEWDS	Data set-up time for write cycle			50
tERDS	Data set-up time for read cycle			350
tH	Hold time	10		

Table 3.1 Read-Write timing values

Note: "System Clock" – SYCLK signal has the same timing as "Enable"

3.2 READ-WRITE OPERATION

All transfers are synchronous with ENABLE signal and controlled by R/W, VMA (or VPA) signals.

Fig. 3.2 illustrates a memory read or write cycle and the timing is given in the table 3.1.

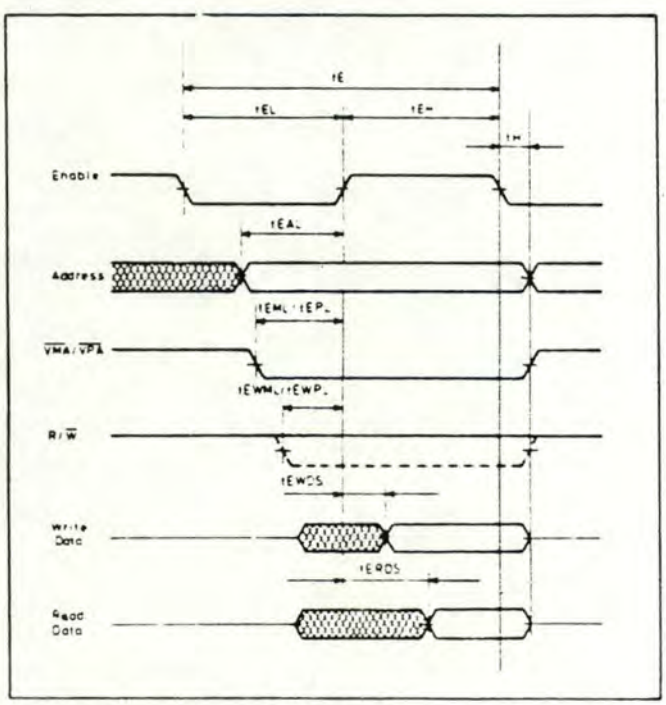


Fig. 3.2 Read-Write operation

3.3 SLOW MEMORY

The "Ready" signal allows slow memory or peripheral to communicate with the GESSBS-4 module on the G-64 Bus. When ready is low, the Enable pulse high will be stretched by increments of 250 ns as shown on fig 3.3.

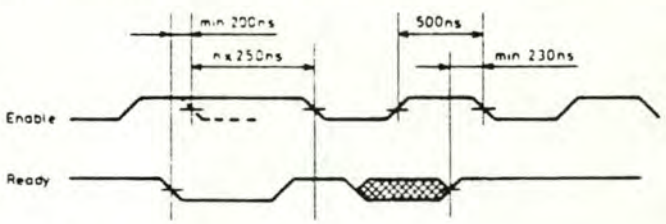


Fig. 3.3 "Ready" timing

3.4 DIRECT MEMORY ACCESS – DMA

The GESSBS-4 module provides the user with DMA capability. The processor needs control to refresh itself every 16 cycles, this allows the requesting device to operate up to 14 DMA cycles. DMA characteristics are illustrated on fig 3.4 and 3.5.

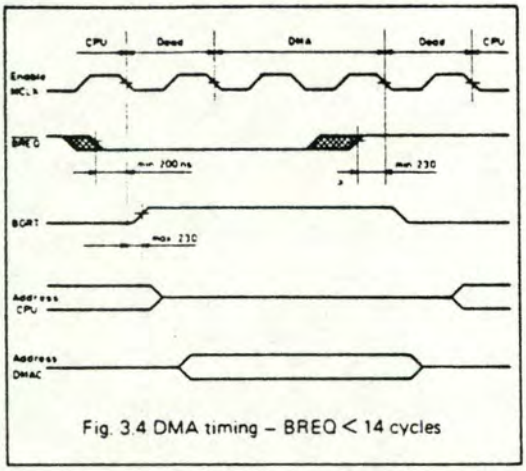


Fig. 3.4 DMA timing – BREQ < 14 cycles

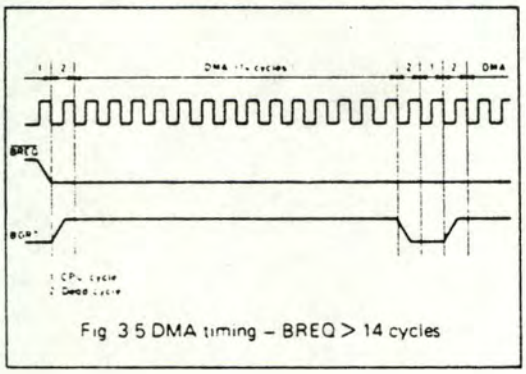
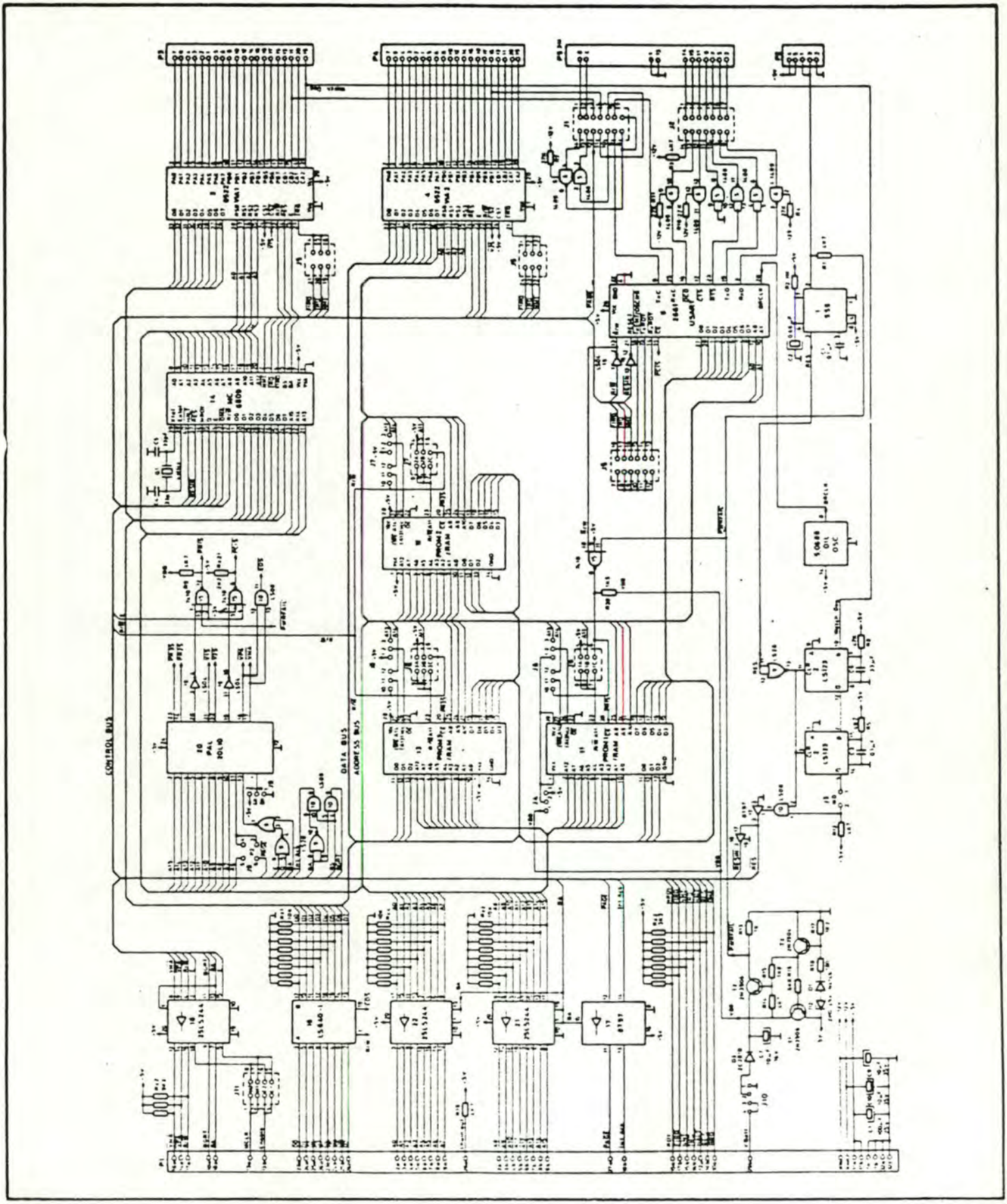


Fig. 3.5 DMA timing – BREQ > 14 cycles

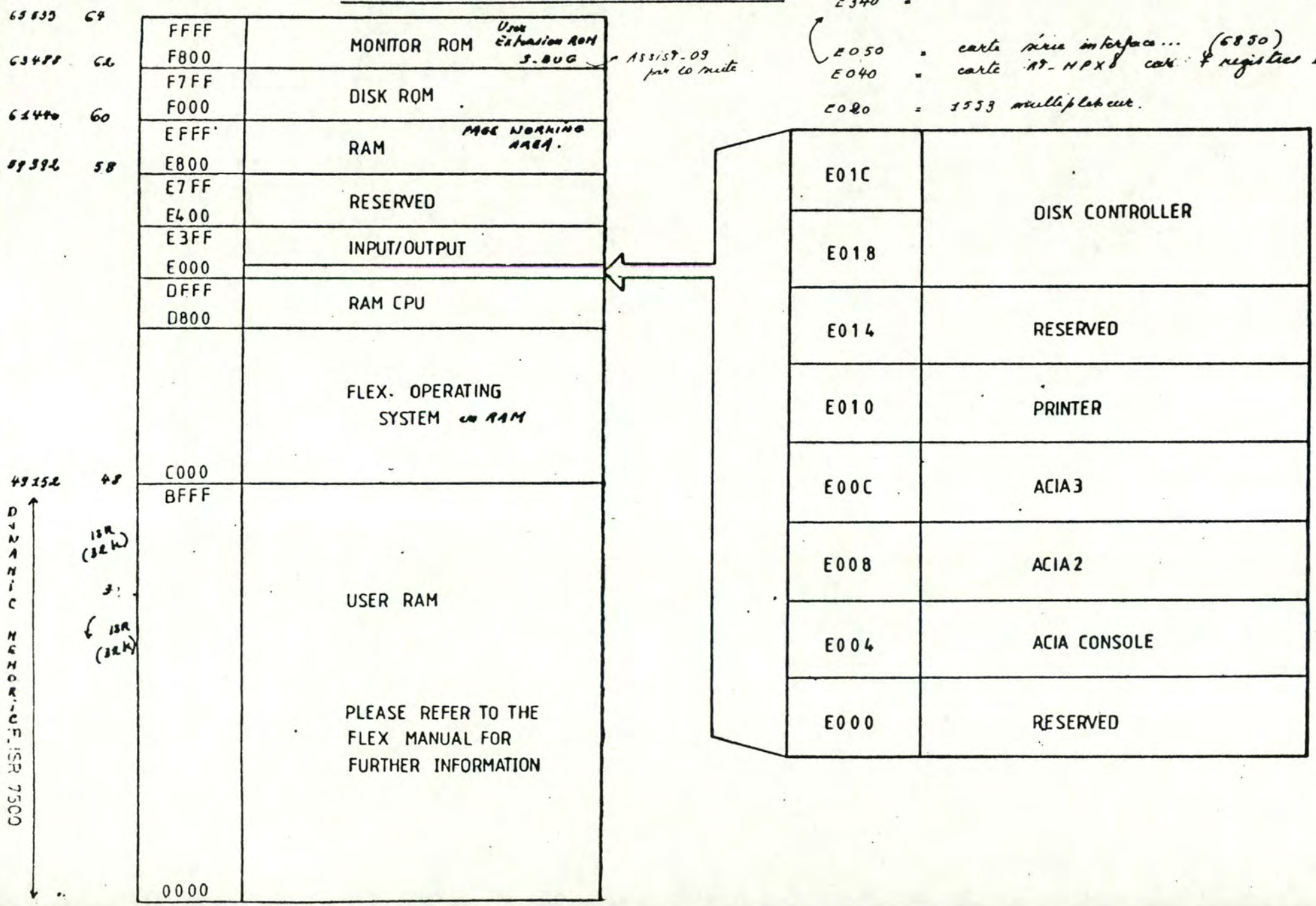


DEVELOPED BY:

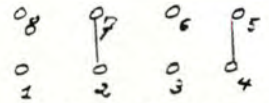
Lespar s a
3, ch. des Aulx
CH-1228 Genève/Plan-les-Ouates
Tél. 022/713 400
Télex 429 989

DISTRIBUTED BY:

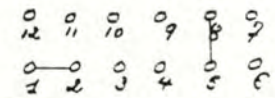
FIG 1 SYSTEM MEMORY ALLOCATION



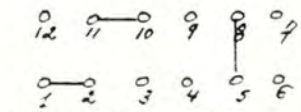
La sélection du type de configuration-mémoire se faisant en positionnant des cavaliers sur le connecteur J9, la figure suivante montre comment effectuer le choix:



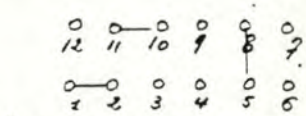
L'espace ROM compris entre les adresses F000 et FFFF se trouve sur la puce U13 PROM3/RAM qui pour être sélectionée correctement en tant qu'EPROM de 4Kbytes doit voir le connecteur J8 dans l'état suivant:



L'espace RAM compris entre E400 et EFFF se trouve sur la puce U12 PROM2/RAM qui pour être sélectionnée correctement en tant que RAM de 8Kbytes doit voir le connecteur J7 dans l'état suivant:



L'espace RAM compris entre C000 et DFFF se trouve sur la puce U11 PROM1/RAM qui pour être sélectionnée correctement en tant que RAM de 8 Kbytes doit voir le connecteur J6 dans l'état suivant:



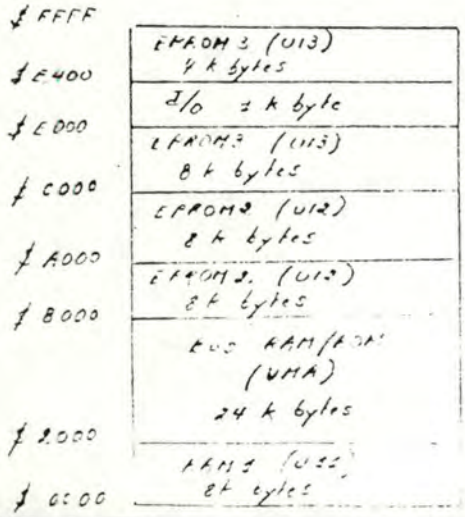
Le reste est de la mémoire dynamique, soit 48Kbytes, fournie par des cartes-mémoire de 32Kbytes chacune. Les EPROM sont des 27128 Kbits, les RAM des 6232 Kbits.

3.2.4 <u>Configuration de la carte sans système FLEX, sans monitor S-BUG</u>

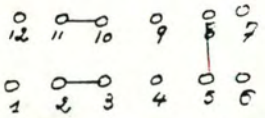
Prenons l'exemple de mon code contenu dans le fichier 1.MOPCIF.BIN. Comme le code est composé de deux parties :

- celle contenant le programme principal et devant occuper 12 Kbytes et 273 bytes en EPROM,
- celle permettant l'initialisation correcte des vecteurs d'interruption et de Reset et devant occuper 10 bytes en EPROM

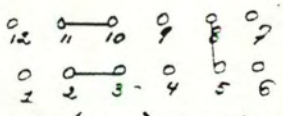
et occupe une zone de données de 8Kbytes, la meilleure configuration-mémoire était la nr 4 proposée par GESPAC. A ces fins, il suffit de trouver la puce U20 PAL20L10 étiquetée MAP3/4 donnant cette configuration et de l'interchanger avec celle configurant la mémoire pour FLEX.



Les vecteurs d'interruption et de Reset seront sur la puce U13 EPROM3/RAM à partir de l'adresse FFF6, qui pour être sélectionnée en tant qu'EPROM de 16 Kbytes doit voir le connecteur J8 dans l'état suivant:



Le programme en code binaire se trouvera à partir de l'adresse 8000 dans la puce U12 EPROM2/RAM qui pour être sélectionnée en tant qu'EPROM de 16 Kbytes doit voir le connecteur J7 dans l'état suivant:



Les variables seront logées à partir de l'adresse 0000 dans la puce U11 EPROM1/RAM dont le connecteur associé reste inchangé.

En résumé, par rapport à la configuration précédente, les EPROM U13 et U12 doivent changées respectivement de 2732 en 27128 et de 6264 en 27128.

3.3 Carte GESSIO-1A de GESPAC

3.3.1 Emploi de la carte

Le but de cette carte est d'assurer l'interface entre le G64 et les ordinateurs GP* à contrôler. Comme chaque carte est pourvue de deux canaux permettant la connexion de deux GPs, il faudra prévoir autant de canaux qu'il n'y a de GPs mais deux fois moins de cartes. Pour le moment, le nombre de canaux admissibles est fixé par software à 20 (constante NCANAUX dans le fichier MOPCIF.TXT sans système FLEX ou BMOPCIF.TXT avec système FLEX). Ces cartes seront placées en série sur le bus G64 à partir de l'adresse E050, chacune ayant par rapport à la précédente une adresse de base valant l'adresse de la précédente plus 8 car huit registres, quatre pour chaque canal, sont associés à chaque carte sur l'espace I/O-memoire.

3.3.2 Description de la carte

Lire la documentation technique fournie par GESPAC.

3.3.3 Changement d'adresse

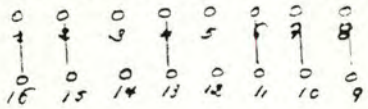
Si, pour une raison ou une autre, l'adresse de base venait à être changée, deux manipulations sont nécessaires:

- changer dans le fichier PDEV6850.TXT sans système FLEX ou BDEV6850.TXT avec système FLEX, la valeur de la constante ADDR-6850 et recompiler, réassembler le tout;
- modifier le connecteur J14 qui sélectionne par hardware l'adresse de base du module.

3.3.4 Configuration de la carte

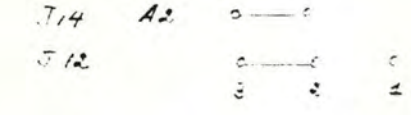
Cette carte fournit deux canaux de communication série, en mode asynchrone, sur le bus G64.

- Connecteur J14 pour sélectionner l'adresse de base du module E050:



En effet, E050 vaut en binaire 1110 0000 0101 0000 et les valeurs A0-A1 et A10-A15 sont connues au niveau hardware suite à la puce U20 PAL 20L10 de la carte GESSBS-4 configurant la mémoire pour travailler avec FLEX. Cette disposition des cavaliers est à changer au cas où l'adresse de base du module serait modifiée, pour passage à une autre configuration-mémoire où l'espace I/O se situerait dans un autre intervalle d'adresses.

- Connecteurs J14 (A2) et J12 doivent se présenter comme suit pour indiquer que la carte GESSIO-1A occupe 8 adresses de l'espace d'adressage I/O:



DOUBLE SERIAL INTERFACE MODULE

The GESSIO-1A module provides two serial communication channels on the G-64 bus. This module can support both synchronous and asynchronous mode with respectively 6852 (SSDA) and 6850 (ACIA) type of interface. Sockets for the two different devices are provided on board and the mode can be changed in field just by plugging the suitable device in the right socket.

Two on board programmable baud rate generators can be programmed either by software or by jumpers with the choice of different transmission speeds for each channel as well as for transmission and reception betwen 50 to 19200 bauds.

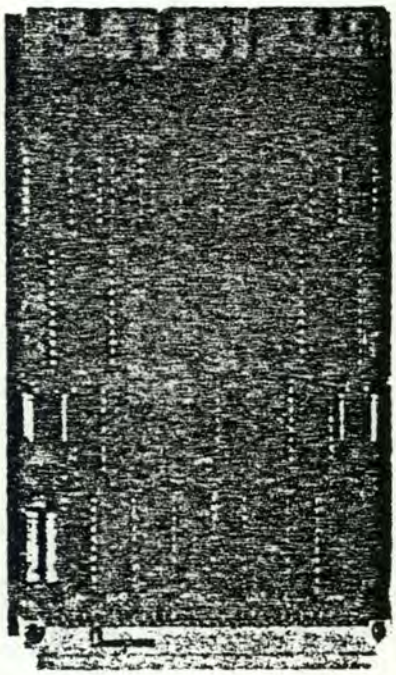
Several interface drivers will offer great flexibility in connection either in RS-232C or RS-422 as well as in TTY isolated current loop for industrial applications.

Two rows of straps are provided to adapt the board to different types of connection including the capability to work in terminal or modem configuration.

The GESSIO-1A is fully compatible with the G-64 bus and is upward compatible with the GESSIO-1.

Technical features

- Two identical serial channels
- Synchronous (6852) and asynchronous mode (6850)
- Separate programmable baud rate generators for each channel
- Clock programmable separately for transmission and reception
- Baud rate from 50 bauds to 19200 bauds
- Choice of interface RS-232C, RS-422, TTY isolated current loop
- Active and passive current loop
- Automatic generator of interrupt vector
- Standard power supply: +5V, ± 12V



References

GESSIO-1A: Double asynchronous Serial Interface
GESSIO-1S: Double synchronous Serial Interface
GESSIO-1C: Combined sync./ async. Serial Interface

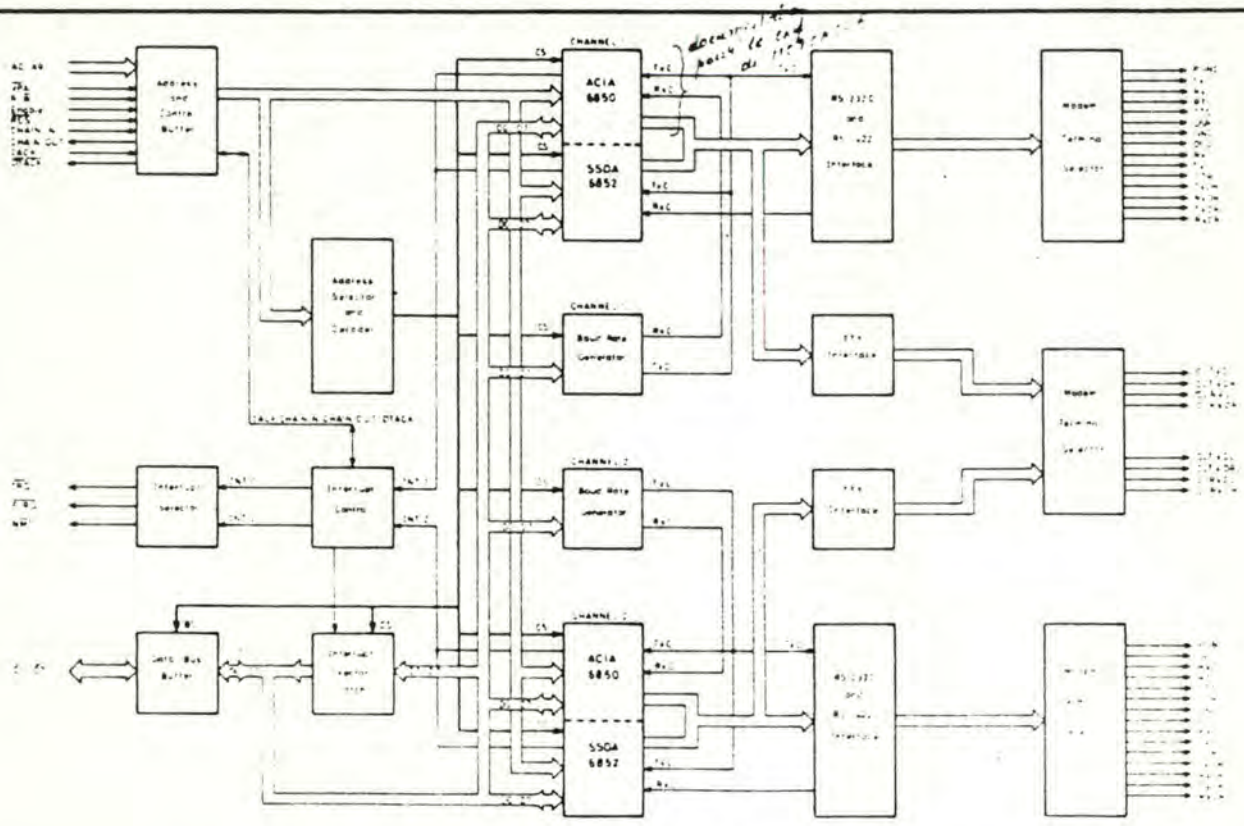


Fig 1.1 Block diagram

1. GENERAL INFORMATION

1.1 DESCRIPTION

The GESSIO-1A module provides two serial communication channels fully independent of each other. These are selectable separately either in RS-232C, RS-422 or TTY current loop interface. The channels can support synchronous and asynchronous modes with respectively 6852 (SSDA) and 6850 (ACIA).

Two programmable baud rate generators can be programmed either by software or by jumpers to provide different transmission speeds on each channel for transmission as well as reception.

The EIA RS-232C (or the CCITT V 24) electrical interface is a single-ended, bipolar - voltage, unterminated circuit. It allows communication over short distances (up to 15 meters at 19.2 Kilobaud according to EIA specifications).

The EIA RS-422 is a differential, terminated, balanced voltage interface capable of significantly higher data rates over longer distances (up to 1200 meters at 100 Kilobaud). The standard RS-422 provides a better immunity to electromagnetic interferences.

The TTY interface circuit allows full duplex, 20 mA current loop operation. This interface is isolated by optocouplers.

Each channel has its own 20-pin connector with a pinning that correspond to a RS-232C 25-pin delta connector type. In this way, connection to a peripheral device can be made with a flat ribbon cable.

The RS-422 signals are combined with the RS-232C lines on the same connector and use four extra lines not connected in the standard RS-232C interface. The RS-422 receiver and transmitter must be connected with a twisted pair cable of approximately 120 Ω impedance terminated on the GESSIO-1A with a 100 Ω resistor.

The isolated TTY interfaces of the two channels are linked together on a 10-pin connector.

Two rows of straps allow to adapt each channel to different types of communication (either RS-232C, RS-422 or TTY). Furthermore these straps allow to modify the pinning of interface connectors in a modem or terminal connector.

Each channel can generate an interrupt level selectable by jumpers. The module provides an interrupt vector which can be used with asynchronous processor module like GESMPU-4A.

Warning: the 6850 and 6852 must not be plugged simultaneously on the same channel.

The GESSIO-1A module is fully compatible with the standard G-64 bus. The block diagram of fig 1.1 illustrates the different parts of the module and their interconnections.

1.2 SPECIFICATIONS

Transmission mode	Asynchronous with 6850 (ACIA) Synchronous with 6852 (SSDA)
Serial interface	Selectable on each channel: EIA RS-232C/V 24 EIA RS-422 TTY 20 mA current loop
Baud rate generator	4 baud rate clock programmable by software or selectable by jumpers
Reference frequency	5.068 MHz
Baud rate frequency	16 selectable frequencies 50 to 19200 baud (16 x clock)
Asynchronous mode	8 or 9-bit transmission Optional Even or Odd parity One or Two-stop bit 1x, 16x, 64x clock Programmable interrupts
Synchronous mode	7, 8 or 9-bit transmission Optional Even or Odd parity One or Two-sync codes Programmable interrupts
Bus interface	–Address and Data bus: 3-state TTL compatible –Other signals: TTL compatible
Bus drivers	48 mA device type
Power requirements	+ 5 Vdc: 650 mA typ. +12 Vdc: 35 mA typ. –12 Vdc: 30 mA typ.
Operating temperature	+5°C to +55°C
PCB dimensions	100 x 160 mm

Table 1.1 Specifications

2. PREPARATION FOR USE, INTERCONNECTIONS

2.1 CONNECTORS AND JUMPERS IDENTIFICATION

Table 2.1 identifies the jumpers and connectors of the GESSIO-1A module. Fig.2.1 shows their location on the printed circuit.

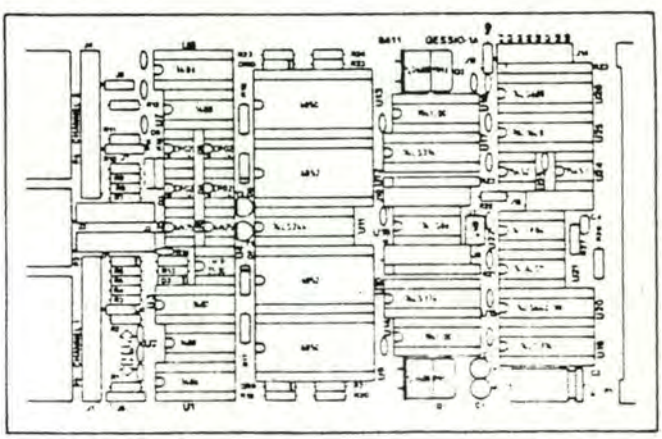


Fig 2.1 Implementation

Designation	Function
P1	G-64 Bus interface connector
P2	Channel 1 RS-232C or RS-422 interface connector
P3	Channel 1 and channel 2 TTY current loop interface
P4	Channel 2 RS-232C or RS-422 interface connector
J1	Channel 1, RS-232C/RS-422 selector
J2	Channel 1 and channel 2 TTY current loop selector
J3	Channel 1 and channel 2 TTY source current selector
J4	Channel 2, RS-232C/RS-422 selector
J5,J6	Channel 1, RS-232C/RS-422 selectors
J7,J8	Channel 2, RS-232C/RS-422 selectors
J9	Channel 1, baud rate selection by jumpers or by software
J10	Channel 2, baud rate selection by jumpers or by software
J11 (–*)	TTY TxD active level selector
J12 (J13*)	Field address option selector
J13 (J11*)	Interrupt level selector / DTACK selector
J14 (J12*)	Module address selector.

*Jumper identification on previous GESSIO-1A version before boards with date code 8403 and up.

Table 2.1 Connectors and jumpers identification

2.2 ADDRESS SELECTION

The module base address is jumper selectable on J14 in the field reserved for peripherals which is characterized by the VPA signal. Table 2.3 shows how to select 1 of 128 base addresses in the 1 K VPA field.

The 8 consecutive addresses required by the module are defined in the table 2.2.

Module Base address	Read operations	Write operations
	Channel 1[1]	
+0	Status register	Control register
+1	Receive Data	Transmit Data
	Channel 2	
+2	Status register	Control register
+3	Receive Data	Transmit Data
+4[3]	Not used	Interrupt vector[2]
+5	Not used	Not used
	Channel 1	
+6	Not used	Baud rate register
	Channel 2	
+7	Not used	Baud rate register

[1] The G-64 RESET signal is applied to the ACIA/SSDA as a master reset

[2] The board provides the same interrupt vector for the two channels

[3] Refer to the paragraph 2.3.

Table 2.2 Module address registers

J14	OFF	ON
1 o o 16	A2*	$\overline{A2}$
2 o o 15	A3	$\overline{A3}$
3 o o 14	A4	$\overline{A4}$
4 o o 13	A5	$\overline{A5}$
5 o o 12	A6	$\overline{A6}$
6 o o 11	A7	$\overline{A7}$
7 o o 10	A8	$\overline{A8}$
8 o o 9	A9	$\overline{A9}$

*A2 only on new version, see paragraph 2.3

Table 2.3 Module address selection

2.3 FIELD ADDRESS OPTION

This new feature is offered to the GESSIO-1A user's on the boards with date code 8403 and up.

This possibillity allows to adapt the mapping of the GESSIO-1A for compatibility with the GESSIO-1 field address.

In this case the module only uses 4 consecutive addresses in VPA field. The module base address +4 to +7 are suppressed and the baud rate must be selected by jumpers (J9 for channel 1, J10 for channel 2)

Table 2.4 shows how to select the field address option.

8 addresses Standard GESSIO-1A mapping		4 addresses GESSIO-1 compatible mapping	
J14 (A2)	J12	J14 (A2)	J12
o—o A2	3 o 2 o 1 o	A2 or $\overline{A2}$ module base address selection	3 o 2 o 1 o

Table 2.4 Field address selection

2.4 SERIAL COMMUNICATION RS-232C

The RS-232C (or V 24) specifications define a standard transmission mode between the two standard equipments:

DTE Data Terminal Equipment (Terminal or Computer)

DCE Data Communication Equipment (Modem)

The RS-232C signal names are always specified in conjunction with the DTE. For instance:

TxD Transmit Data (from DTE to DCE)

RxD Receive Date (from DCE to DTE)

Fig 2.2 illustrates a typical RS-232C connection between a computer and a terminal.

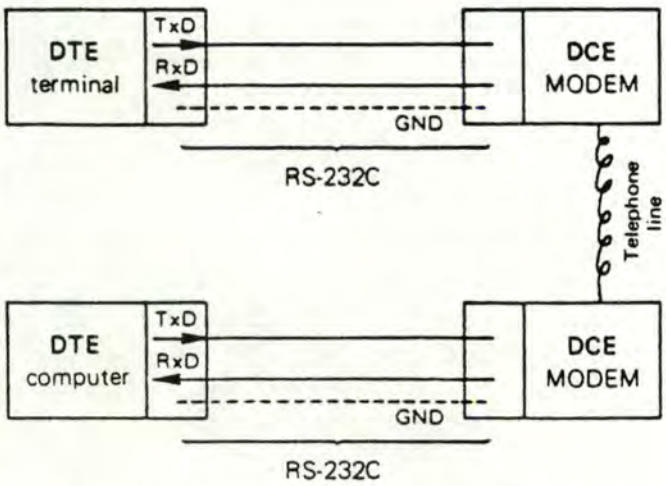


Fig 2.2 Typical RS-232C connection

The standard RS-232C allows direct connection between two DTE with a Null-Modem box as shown on fig 2.3.

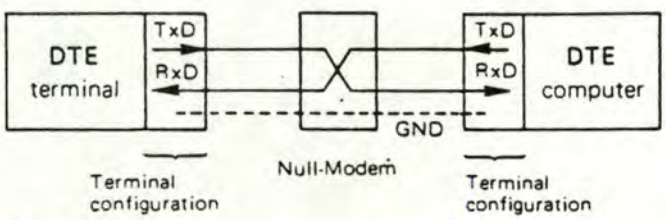


Fig 2 3 DTE-DTE connection with Null-Modem box

The RS-232C pinning can be changed directly by jumpers on the GESSIO-1A module that allows a connection DTE-DTE without Null-Modem box. See fig 2.4

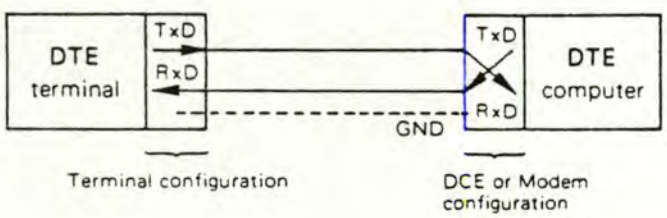


Fig 2.4 DTE-DTE connection without Null-Modem box

2.5 RS-232C AND RS-422 CONNECTORS

Each channel has its own 20-pin connector (P2 and P4) with a pinning that correspond to a RS-232C 25-pin delta connector type as shown in fig 2.5. In this way, connection to a peripheral device can be made with a flat ribbon cable

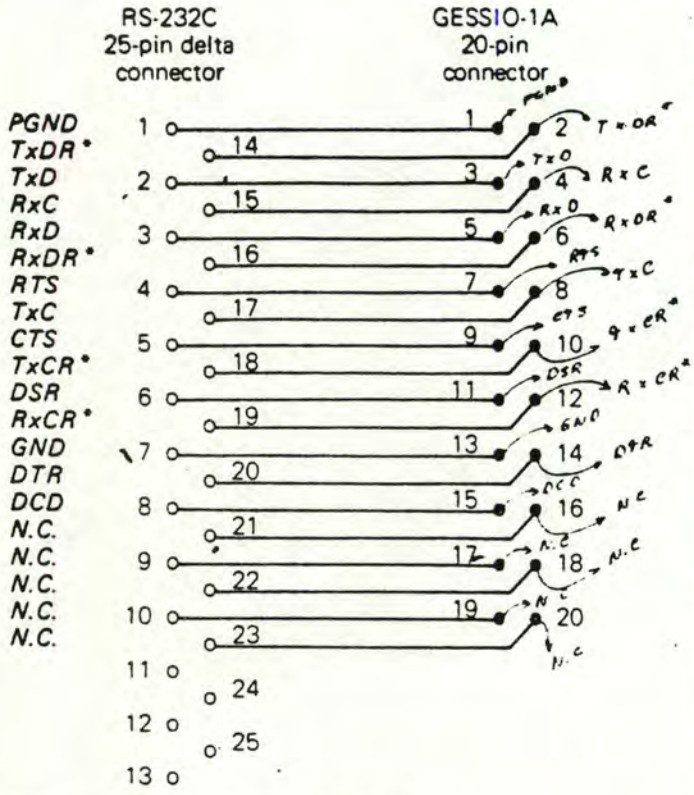
The RS-422 signals are combined with the RS-232C lines on the same connector and use four extra lines not connected in standard RS-232C interface.

Standard signals for an RS-232C serial interface are defined in the table 2.5.

Pin No	Signal name	Signal description
1	Protective Ground	Common for the 12 Vdc source
2	Transmit Data (TxD)	Direction: TO modem (DCE) This line transfers data from a terminal to a modem
3	Receive Data (RxD)	Direction: FROM modem This line transfers data from a modem to a terminal
4	Request to Send (RTS)	Direction: TO modem When active (high level) this signal requires a data transfer on TXD from the terminal to the modem
5	Clear to Send (CTS)	Direction: FROM modem When active (high level) this signal indicates that the modem is ready to receive a data transfert on TxD. CTS is the modem answer to RTS
6	Data Set Ready (DSR)	Direction: FROM modem When active (high level) this signal indicates that the modem is connected and ready to receive terminal command
7	Signal Ground	Signal uses as the reference potential for unbalanced interchange circuits
8	Data Carrier Detect (DCD)	Direction: FROM modem When active (high level) this signal indicates that the modem is receiving a telephone modulation in the modem appropriate limits
15	RxC	Direction: FROM modem Terminal clock input
17	TxC	Direction: TO modem Terminal clock output
20	Data Terminal Ready (DTR)	Direction: TO modem When active (high level) this signal indicates the modem that the terminal is connected and operational
9–13	Not used	
14*	RS-422 Transmit Data Return(TxDR)	Direction: TO modem Complementary signal of TXD for differential data transmission
16*	RS-422 Receive Data Return(RxDR)	Direction: FROM modem Complementary signal of RxD for differential data reception
18*	RS-422 TxCR	Direction: TO modem Complementary signal of TxC for differential clock transmission
19*	RS-422 RxCR	Direction: FROM modem Complementary signal of RxC for differential Clock reception
21–25	Not used	

*This lines are not connected with RS-232C configuration

Table 2.5 RS-232C 25-pin delta connector



*This lines are not connected with RS-232C configuration.

Fig 2.5 GESSIO-1A and RS-232C direct connection

2.6 RS-232C AND RS-422 CONFIGURATION

The RS-232C or RS-422 mode may be selected by jumpers J1 and J4 for Channels 1 and 2 respectively. Futhermore these straps allow to modify the pinning of interface connectors in a modem connector or terminal connector.

Signals defined in the table 2.5 are connected to P2 (channel 1) or to P4 (channel 2) through the interconnections of J1 or J4 as illustrated on fig 2.6.

	J1 or J4	
DCD	16 o o 15	+12V
DCD	17 o o 14	***DSR***
DTR	18 o o 13	***DSR***
TxCR*	19 o o 12	RxCR*
TxCR*	20 o o 11	***RxCR****
CTS	21 o o 10	***RTS***
CTS	22 o o 9	RTS
TxC	23 o o 8	***TxC***
RxC	24 o o 7	TxC*
RxC	25 o o 6	***TxC***
RxD	26 o o 5	TxD
RxD	27 o o 4	***TxD***
RxD	28 o o 3	TxD*
TxDR*	29 o o 2	***RxDR****
TxDR*	30 o o 1	RxDR*

Printed in italic: signal to P2 or P4 connectors

Normal characters: signal to ACIA/SSDA

*Lines with RS-422 levels

Fig 2.6 Jumpers J1 or J4 pin assignment

The pinning of J1/J5/J6 or J4/J7/J8 allows direct jumper interconnection for standard applications as shown in table 2.6.

Jumpers	RS-232C/V 24		RS-422	
	DCE Modem	DTE Terminal	DCE Modem	DTE Terminal
J1 (channel 1) / J4 (channel 4)	16 o—o 15 17 o—o 14 18 o—o 13 19 o o 12 20 o o 11 21 o—o 10 22 o—o 9 23 o o 8 24 o o 7 25 o o 6 26 o—o 5 27 o—o 4 28 o o 3 29 o o 2 30 o o 1	16 o—o 15 17 o—o 14 18 o—o 13 19 o o 12 20 o o 11 21 o o 10 22 o o 9 23 o—o 8 24 o—o 7 25 o—o 6 26 o o 5 27 o o 4 28 o o 3 29 o o 2 30 o o 1	16 o—o 15 17 o—o 14 18 o—o 13 19 o o 12 20 o—o 11 21 o—o 10 22 o—o 9 23 o o 8 24 o—o 7 25 o—o 6 26 o o 5 27 o—o 4 28 o—o 3 29 o—o 2 30 o—o 1	16 o—o 15 17 o—o 14 18 o—o 13 19 o o 12 20 o o 11 21 o o 10 22 o o 9 23 o o 8 24 o o 7 25 o o 6 26 o o 5 27 o o 4 28 o o 3 29 o o 2 30 o o 1
J5 (channel 1) / J7 (channel 2)	(1 2 3)* 1 2 3 4 o o o—o	(1 2 3)* 1 2 3 4 o o o—o	(1 2 3)* 1 2 3 4 o o—o o	(1 2 3)* 1 2 3 4 o o—o o
J6 (channel 1) / J7 (channel 2)	1 2 3 o o—o	1 2 3 o o—o	1 2 3 o—o o	1 2 3 o—o o

*Pin assignment on previous GESSIO-1A version (before date code 8403).

Table 2.6 RS-232C and RS-422 standard applications

2.7 BAUD RATE SELECT

The baud rate generators can be programmed either by software or by jumpers with the choice of different transmission and reception speeds on each channel

Table 2.7 illustrates how to program the baud rate generators and table 2.8 shows the baud rate selection.

In asynchronous mode (with ACIA), the transmit and receive clock are provided by the on board baud rate generators. In synchronous mode (with SSDA), the receive clock must be provided externally.

Baud rate	Jumpers selection	Software* selection
	J9 (channel 1) J10(channel 2) 1 o o 18 N.C.	Ch1: base address +6 Ch2: base address +7
Transmit Clock	2 o o 17 A 3 o o 16 B 4 o o 15 C 5 o o 14 D	D0 = A D1 = B D2 = C D3 = D
Receive Clock	6 o o 13 A 7 o o 12 B 8 o o 11 C 9 o o 10 D	D4 = A D5 = B D6 = C D7 = D

*Software selection: only connect the jumper 1–18 of J9/J10

Table 2.7 Baud rate generator programming mode

Transmit/Receive Selector					Baud rate
D	C	B	A	HEXA	16x clock
0	0	0	0	0	50
0	0	0	1	1	75
0	0	1	0	2	110
0	0	1	1	3	134,5
0	1	0	0	4	150
0	1	0	1	5	300
0	1	1	0	6	600
0	1	1	1	7	1200
1	0	0	0	8	1800
1	0	0	1	9	2000
1	0	1	0	A	2400
1	0	1	1	B	3600
1	1	0	0	C	4800
1	1	0	1	D	7200
1	1	1	0	E	9600
1	1	1	1	F	19200

Table 2.8 Baud rate selection

2.8 TTY INTERFACE

The TTY interface circuit allows full-duplex, asynchronous, 20 mA current loop operation. The interface of the two channels are linked together on P3.

The signals are defined in table 2.9 and are connected to P3 through the interconnections of J2 as illustrated on fig 2.7.

TTY standard interconnections are shown in table 2.10.

Pin No	Signal name	Signal description
2	CLTxD1	Transmit Data (channel 1)
3	CLRxD1	Receive Data (channel 1)
4	CLTxDR1	Transmit Data Return (channel 1)
5	CLRxDR1	Receive Data Return (channel 1)
6	CLTxD2	Transmit Data (channel 2)
7	CLRxD2	Receive Data (channel 2)
8	CLTxDR2	Transmit Data Return (channel 2)
9	CLRxDR2	Receive Data Return (channel 2)
1,10	GND	GESSIO-1A Ground (Not isolated)

Table 2.9 P3 TTY interface connector

J2

CLRxD1	1 o o 16	***CLTxD1***
CLRxD1	2 o o 15	CLTxD1
CLRxDR1	3 o o 14	***CLTxDR1***
CLRxDR1	4 o o 13	CLTxDR1
CLRxD2	5 o o 12	***CLTxD2***
CLRxD2	6 o o 11	CLTxD2
CLRxDR2	7 o o 10	***CLTxDR2***
CLRxDR2	8 o o 9	CLRxDR2

Printed in italic: signal to P3 connector

Normal characters: signal to ACIA/SSDA

Fig 2.7 Jumper J2 pins assignment

TTY interface Selection		TTY mode selector	
		DCE Modem	DTE Terminal
J1(Ch 1)	J5(Ch1)	J2 (Ch1)	J2 (Ch1)
J4(Ch2)	J7(Ch2)	1 o—o 16	1 o o 16
16 o—o 15	1 o	2 o—o 15	2 o o 15
17 o o 14	(1)* 2 o	3 o—o 14	3 o o 14
18 o o 13	(2) 3 o	4 o—o 13	4 o o 13
19 o o 12	(3) 4 o		
20 o o 11			
21 o o 10			
22 o o 9			
23 o o 8	J6(Ch1)		
24 o o 7	J8(Ch2)	J2 (Ch2)	J2 (Ch2)
25 o o 6	1 o	5 o—o 12	5 o o 12
26 o o 5	2 o	6 o—o 11	6 o o 11
27 o o 4	3 o	7 o—o 10	7 o o 10
28 o o 3		8 o—o 9	8 o o 9
29 o o 2			
30 o o 1			

*With previous GESSIO 1A version: connect (2)–(3).

Table 2.10 TTY interface standard applications

The TTY current loop can be connected with an external power source (that allows an isolated interface) or with the bus (+12V and GND) power supplies.

Current loop power selection on J3 is illustrated in table 2.11.

A new feature is offered to the GESSIO-1A user's on boards with date code 8403 and up. In this case J11 allows to select the active level of CLTxD for parallel or serial current source interfaces. See table 2.12.

J3	Current loop power supplies		
	external	internal*	
	X	Transmission lines	Reception lines
Channel 1	1 o o 16 2 o o 15 3 o o 14 4 o o 13	1 o o 16 2 o—o 15 3 o o 14 4 o—o 13	1 o—o 16 2 o o 15 3 o—o 14 4 o o 13
Channel 2	5 o o 12 6 o o 11 7 o o 10 8 o o 9	5 o o 12 6 o—o 11 7 o o 10 8 o—o 9	5 o—o 12 6 o o 11 7 o—o 10 8 o o 9

*Internal power supplies may be used only if the current loop lines are not externally supplies.

Table 2.11 Current loop power supplies selection

Parallel power source	Serial current source
General purpose interface between DTE – DCE	Industrial current loop interface
+12V	X
J11 Ch2 { 1 o o 6 2 o o 5 3 o o 4 } Ch1	J11 Ch2 { 1 o o 6 2 o o 5 3 o o 4 } Ch1

Table 2.12 Parallel or serial current source selection

Note Only parallel current source interface is compatible with previous version GESSIO-1A (before 8403).

2.9 INTERRUPTS

The interrupt signal of each channel can be routed either on the bus $\overline{IRQ}$, $\overline{FIRQ}$, or $\overline{NMI}$ lines.

The GESSIO-1A module provides the daisy chain signals as well as an interrupt vector that can be written at the module base address +4.

to allow a longer data access time

The asynchronous $\overline{DTACK}$ signal is generated on the module for the interrupt and can only be used with asynchronous processor module like the GESMPU-4A.

Interrupt and $\overline{DTACK}$ selection is shown in the table 2.13.

J11	Signal
1 o o 14	$\overline{DTACK}$
2 o o 13	$\overline{NMI}$ (Channel 1)
3 o—o 12	$\overline{IRQ}$ (Channel 1)
4 o o 11	$\overline{FIRQ}$ (Channel 1)
5 o o 10	$\overline{NMI}$ (Channel 2)
6 o—o 9	$\overline{IRQ}$ (Channel 2)
7 o o 8	$\overline{FIRQ}$ (Channel 2)

Table 2.13 Interrupt and $\overline{DTACK}$ selection

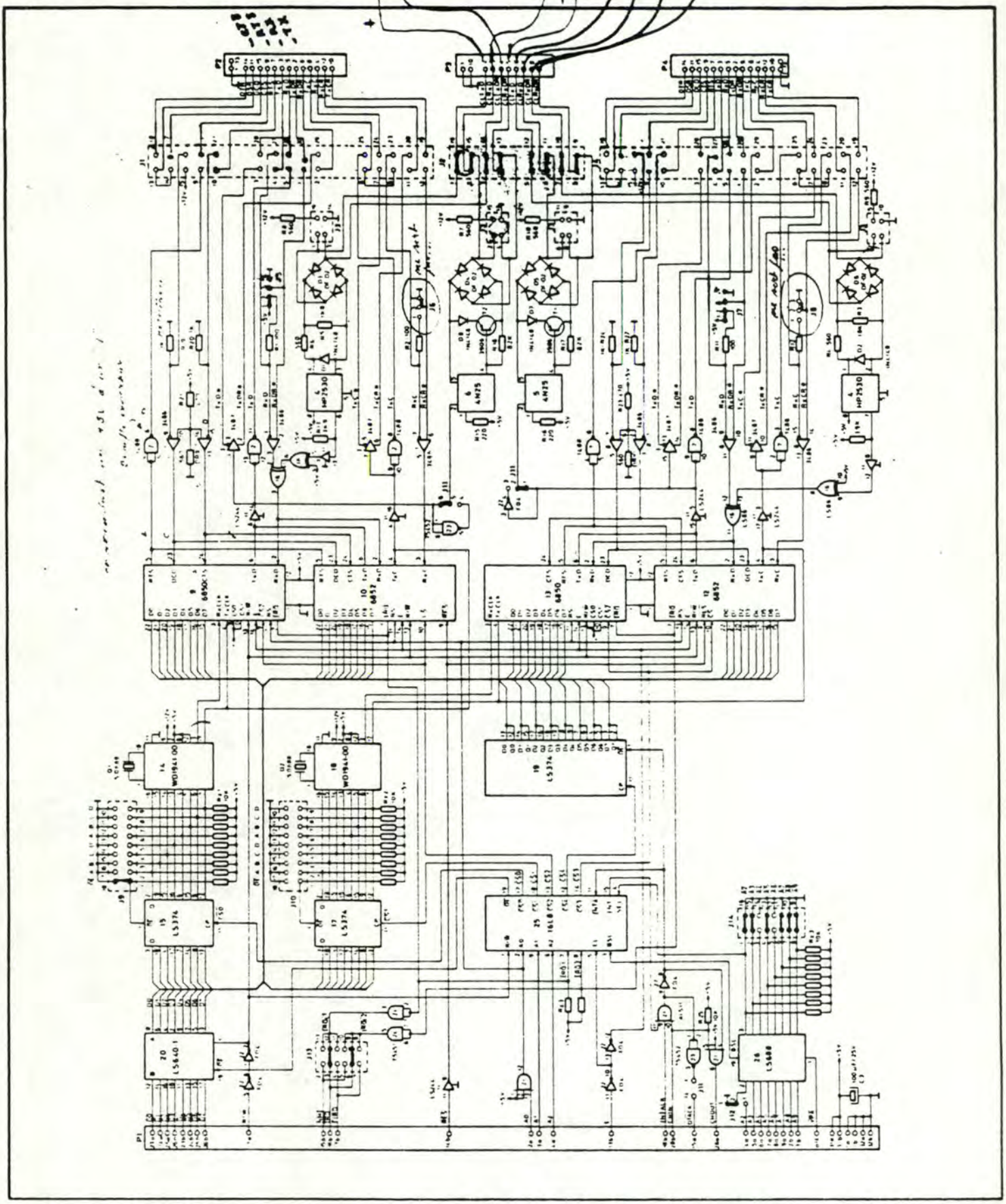
2.10 INTERFACE WITH THE G-64 BUS

The GESSIO-1A module interconnects directly on the G-64 Bus. Signals used by the module are identified in the table 2.14. For more information on the bus, refer to the G-64 Bus specification.

ROW B			ROW A	
GND		1	GND	Power (2)
A8		2	A0	Address lines A0 to A15 (16)
A9		3	A1	
A10	•	4	A2	
A11	•	5	A3	
A12	•	6	A4	
A13	•	7	A5	
A14	•	8	A6	
A15	•	9	A7	
$\overline{BRQ}$	•	10	BGRT	•
$\overline{DS1}$ ($\overline{PRQ}$)	•	11	$\overline{DS0}$ (PSAT)	•
$\overline{BGACK}$	•	12	$\overline{HALT}$	•
Enable		13	SYCLK (MCLK)	•
$\overline{RES}$		14	$\overline{VPA}$	Control lines (18)
$\overline{NMI}$ ($\overline{IN-3}$)		15	RDY ($\overline{DTACK}$)	
$\overline{IRQ}$ ($\overline{IN-2}$)		16	$\overline{VMA}$ ($\overline{AS}$)	•
$\overline{FIRQ}$ ($\overline{IN-1}$)		17	R/$\overline{W}$	
IACK		18	Halt Ack	•
$\overline{D12}$	•	19	$\overline{D8}$	•
$\overline{D13}$	•	20	$\overline{D9}$	•
$\overline{D14}$	•	21	$\overline{D10}$	• DATA 2nd Byte (8)
$\overline{D15}$	•	22	$\overline{D11}$	•
$\overline{D4}$		23	$\overline{D0}$	DATA 1st Byte (8)
$\overline{D5}$		24	$\overline{D1}$	
$\overline{D6}$		25	$\overline{D2}$	
$\overline{D7}$		26	$\overline{D3}$	
$\overline{Parity\ error}$ ($\overline{BERR}$)	•	27	$\overline{Page}$	• Miscellaneous (4)
Chain In		28	Chain Out	
+ 5V Battery	•	29	– 5V	• Power (8)
– 12V		30	+ 12V	
+ 5V		31	+ 5V	
GND		32	GND	

*Not used on GESSIO-1A module

Table 2.14 J1 connector, G-64 Bus



DEVELOPED BY:

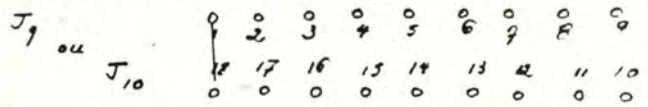
Gespac s a
3, ch. des Aulx
CH-1228 Genève/Plan-les-Ouates
Tel 022/713 400
Telex 429 989

DISTRIBUTED BY:

Mk. BUCB

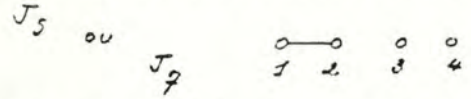
Printed in Switzerland

- Connecteur J9 pour le canal 1, J10 pour le 2 ont la forme suivante pour indiquer que la sélection du Baud Rate s'effectue par software:

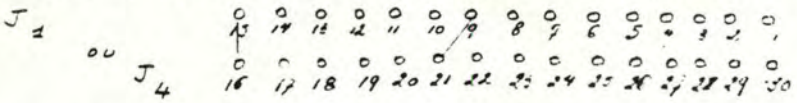


- Connecteur J2 configure la carte comme modem auquel on peut connecter un terminal:

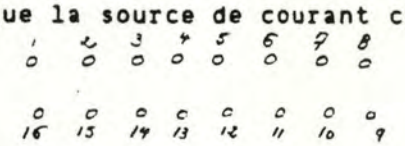
- Connecteurs J5 (canal 1) et J7 (canal 2) se présentent comme suit:



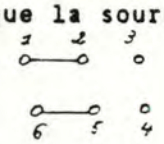
- Connecteurs J1 (canal 1) et J4 (canal 2) se présentent comme suit:



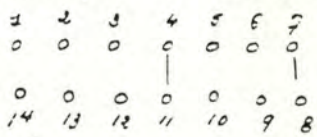
- Connecteur J3 indique la source de courant comme externe:



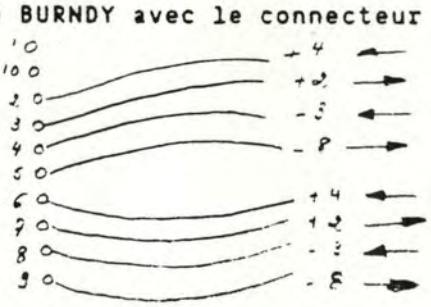
- Connecteur J11 indique que la source de courant est sérielle:



- Les signaux d'interruption générés soit par le canal 1, soit par le canal 2 sont sélectionnés par le connecteur J11 comme étant du niveau FIRQ:



- Connexions de la prise BURNDY avec le connecteur P3:



Rappel : La carte utilisée pour les essais est une carte prêtée par GESPAC car celle fournie par Mr DIEPERING était le premier prototype faux.

3.3.5 Programmation de la puce ACIA 6850

Le driver de cette puce se trouve dans le fichier PDEV6850.TXT sans FLEX et BDEV6850.TXT avec FLEX. Il s'agit d'un module constitué d'un ensemble de procédures déclarées ENTRY et appelables de n'importe quel autre programme pour autant qu'elles y aient été déclaré EXTERNAL (Voir Modular

Compilation dans l' Omegasoft Pascal Handbook).

Les canaux sont programmés pour émettre au taux de 9600 bauds (procédure init-baud). Le caractère envoyé est formé de 7 bits, d'une parité paire et d'un seul stop bit. De plus, il est possible d'activer soit l'interruption en émission, soit l'interruption en réception, soit les deux de l'ACIA 6850 ou de la faire travailler en mode polling (procédures rint-ctrl-reg, tint-ctrl-reg,wint-ctrl-reg...).

Pour éviter la réception inconditionnée de caractères ASCII NUL, un test est effectué au niveau de la lecture par le G64 des caractères reçus pour ne pas en tenir compte. Cette précaution est surtout intéressante dans le cadre de chargement de programmes de test binaires dans un GP contrôlé. En effet, ces programmes émettent des tas de caractères NUL, non intéressants à répercuter vers l'opérateur. De plus, toujours dans le cas d'une lecture, les erreurs de parité et d'overrun pour une lecture non assez rapide des caractères sont detectées et reportées sur la console-operateur (s'il y en a une) et si le code utilisé est la version ayant gardé les commentaires. (procédure read-char)

3.3.6 Protocole de dialogue avec le MOPC de l'ordinateur contrôlé

Le MOPC ne répond que lorsqu'on lui a envoyé une commande. A priori, aucun mécanisme sur la carte GESSIO-1A ne permet de détecter qu'il n'émettra plus . Il faut donc recourir à la notion de Time-Out et il y a tout lieu de croire que si après 40 msec, le MOPC n'a rien émis, il n'émettra plus. En effet, en lancant une opération de DUMP, il est apparu que , bien que travaillant à 9600 bps, les caractères n'étaient reçus par le canal de la carte GESSIO-1A qu'après un intervalle de 2,8 msec.

Ce time-out est réalisé par software, par une boucle Pascal du type

```
I:= 0;
REPEAT
  I:= I+1;
UNTIL (I=NB-LOOP);
```

Comme cette boucle, présente dans la procédure DIALOG-OPCOM du fichier MOPCIF.TXT ou BMOPCIF.TXT, prend 40 microsec. en un passage normal et 42 microsec pour le passage final, pour obtenir un time-out de 40 msec., NB-LOOP doit valoir 1000. (Variable N8-LOOP dans le fichier DECVARE.TXT sans FLEX et BDECVARE.TXT avec FLEX)

Voila comment la boucle est traduite en Assembleur :

```
LB2D EQU *
 LDD -$8C,Y     --> 6 cycles CPU
 ADDD #1        --> 4 cycles CPU
 STD -$8C,Y     --> 6 cycles CPU
 LDD -$8C,Y     --> 6 cycles CPU
 SUBD -$94,Y    --> 7 cycles CPU
 BNE LB2F       --> (3)
 LDB #1         --> 2 cycles CPU
 BRA LB30       --> (3)
LB2F EQU *
 CLRB           --> 2 cycles CPU
LB30 EQU *
 LBEQ LB2D      --> 5(6)
LB2E EQU *
```

3.3.7 Tests de la carte

Un programme de test de l'initialisation correcte de la carte se trouve dans le fichier 1.ESSAI.BIN. Un programme de test de la carte et de sa connexion à l'OPCOM sous niveau FIRQ d'interruption se trouve dans le fichier 1.OPCOMF.BIN

3.4 Cartes d'interface MPX-G64 (ST-G64)

3.4.1 Emploi de ces cartes

Ces cartes-interface permettent de relier les systèmes standards G64 avec le bus MPX existant dans les installations du SPS. La connexion sur le bus est réalisée par les traditionnelles boîtes de jonction comme pour une station MPX normale. Ainsi, chaque interface ou station G64 (ST.G64) a une adresse parmi les 31 disponibles sur un bus MPX. Dans le cadre de l'installation prototype, cette adresse valait 29. La longueur maximale du cable reliant une boite de jonction du bus MPX aux cartes d'interface dans le chassis G64 est de 1,5 mètres.

Sur la carte traitant des signaux de communication MPX, seul le cavalier IRQ est positionné. De plus, le connecteur de selection de l'adresse de base du module indique E040 (à changer si l'adresse de base venait à être modifiée).

Pour une description plus détaillée de la carte, se référer à la note SPS/ACC/Tech.Note/83.23.

3.4.2 Remarques concernant la carte

- La carte des buffers d'E/S a le facheux inconvenient de permuter les bytes au moment de l'écriture ou de la lecture dans/du Buffer à partir du G64. Tout mon code a dû en tenir compte.
- Le registre de contrôle et d'état doit être pourvu du bit Power Fail indiquant au Nord-100 que le G64 a été redémarré.

Cependant, employant le Data Module de Mr D'AMICO, grand utilisateur des cartes RT et le software de Me Michel, une partie du logiciel constituant le package d'application devra être automatiquement adaptée en fonction de l'évolution des cartes MPX-RT. Il ne faut pas non plus oublier l'application Pascal qui devra aussi être modifiée en conséquence.

Une autre remarque est que les cartes actuelles entrent parfois en conflit avec la carte CPU GESSBS-4 de GESPAC utilisée.

- Une lecture trop rapide du Buffer de lecture du G64 augmente la probabilité d'un blocage du système.
- Parfois des bits sont perdus, au niveau de la lecture par le G64 du Buffer d'écriture. Ainsi l'envoi de VGD s'est traduit par la réception de VGP. Cela s'explique par la valeur binaire de D = 100100 et P = 101000.

Il est conseillé d'utiliser à l'avenir une carte GESMPU-2. La raison en est que les cycles du G64 et des cartes MPX-rt sont actuellement différents.

Le driver de ces cartes a été écrit par Me Michel sous GESDOS, puis adapté pour être relu par un système FLEX et se trouve dans le fichier BYCYINT.TXT. Des modifications y ont été apportées :

- Au niveau du désassemblage du driver pour qu'il fonctionne avec un système FLEX, des erreurs y ont été introduites au niveau de la constitution des messages ASCII au moyen de FCC, FCB, FDB. (Voir remarque nr 1 dans le listing du code fourni en annexe);
- Suppression de certains passages redondants. (Voir remarques nr 2, 3, 4 dans le listing du code fourni en annexe).

SPS/ACC/Tech.Note/83-23 le 2 décembre 1983

Interface MPX-G64 (ST-G64)

D. Francart, H. Hauer et R. Wilhelm

1. Introduction

Cette interface permet de relier les systèmes standards G64 avec le bus MPX existant dans les installations du SPS. La connexion sur le bus MPX se fait directement sur les boîtes de jonctions existantes comme pour une station MPX normale. Ainsi chaque interface ou station G64 (ST.G64) a une adresse parmi les 31 stations d'un link MPX.

La longueur maximum du câble reliant une boîte de jonction du bus MPX à la carte d'interface dans le châssis G64 est de 1,5 mètres.

2. Organisation (voir Fig. 1)

Dans le châssis utilisateur l'interfaçage sur le bus G64 est réalisé par deux cartes (100x160) enfichées sur le bus interne.

L'une de ces cartes décode l'adresse de la station, traîte les signaux de communication MPX et fournit à la seconde carte les signaux de synchronisation nécessaires.

Cette deuxième carte comporte deux tampons de données de 2 K octets en entrée et 2 K octets en sortie. Ces tampons sont des RAM organisées en FIFO. Ainsi le tampon d'écriture "Write Buffer" peut être chargé avec des commandes MPX et lu par le G64 et le tampon de lecture "Read Buffer" sera chargé avec les données venant du G64 et lu par des commandes MPX.

Les compteurs de mots en écriture "Write word counter" et en lecture "Read word counter" permettent de connaître le nombre de mots dans chaque tampon.

Un registre de contrôle et d'état (CSR) permet d'échanger des bits d'information entre le MPX et le G64 sur l'état de l'interface.

La deuxième carte comporte également l'interfaçage avec le bus G64 par 8 lignes de données, 5 lignes de contrôle et trois fils d'adresse.

3. Compatibilité entre le MPX et le 1553

Compte tenu de la décision d'adopter pour le contrôle du LEP un bus multi-points conforme au protocole MIL1553B, il est souhaitable de rendre compatible les interfaçages MPX et MIL1553 au niveau de la carte mémoire tampon insérée dans le bus G-64. Ainsi un projet développé en standard G64 pour le SPS et connecté sur le bus MPX pourra être ultérieurement transféré sur le bus MIL1553 si nécessaire. Il suffirait alors de remplacer la carte de décodage MPX par une carte 1553, l'utilisateur dialoguant avec la même carte mémoire.

4. Commandes MPX de lecture et d'écriture

Dans le système MPX chaque transfert d'un mot de 16 bits de données est précédé par une commande qui s'identifie de la façon suivante :

15 – 13	12 – 8	7 – 3	2 – 0
Fonction multiplex	STATION	MODULE	Sous adresse

Pour l'interface MPX G-64, le champ MODULE sera toujours à "Ø".

La fonction multiplex FM2 permet d'écrire un mot de 16 bits dans l'interface. La fonction FM6 permet de lire un mot de 16 bits dans l'interface. Ces deux fonctions sont associées avec des sous-adresses (SA) pour l'adressage des diverses composantes de l'interface.

Les autres fonctions MPX ne sont pas utilisées dans cette interface. Un logiciel spécifique sera disponible pour l'utilisateur au SPS, voir SPS/ACC/Note/83-17.

Le tableau de la figure 2 donne les fonctions MPX opérationnelles dans l'interface ST G-64. Le temps de transfert d'un mot 16 bits de données entre l'unité de contrôle dans le CAMAC et la mémoire tampon de l'interface est de l'ordre de 100 µs.

5. Registre de contrôle et d'état - CSR (voir Fig. 3)

Ce registre de 16 bits peut être lu et écrit par des commandes MPX. FM6 SA1 pour la lecture et FM2 SA1 pour l'écriture sélective des 16 bits du CSR. L'octet de faible poids du CSR (Handshaker) peut également être lu et écrit par le bus G64 à l'adresse 1. Ce registre permet aux deux systèmes, MPX et G-64, d'échanger des bits d'informations sur l'état de l'interface.

L'écriture est toujours sélective par un "1" dans le bit considéré pour la mise à "Ø" ou la mise à "1" excepté pour le bit BU qui est positionné à zéro ou à un par un cycle d'écriture du G-64.

Les bits de ce registre sont mis à zéro par une commande de remise à zéro de l'interface, excepté les bits REM et LOC qui gardent l'état précédent.

Lors d'une remise sous tension seul le bit REM est mis à "1".

5.1 Identification des bits du registre CSR (Fig. 3)

RF Read Buffer Full : Ce bit doit être mis à "1" par le G64 pour signaler au système que des données ont été chargées dans le tampon de lecture et remettre à zéro le pointeur d'adresse de la mémoire tampon de lecture. Ceci permet au MPX de lire et d'acquérir les données.

Il sera remis à "Ø" par une écriture sélective d'un "1" issue d'une commande MPX (FM2 SA1) après l'acquisition des données par le système. La mise à zéro de RF remettra le pointeur d'adresse de la mémoire de lecture à zéro.

WF Write Buffer Full : Ce bit doit être mis à "1" par le MPX pour signaler au G64 que des données ont été chargées dans le tampon d'écriture. Après la lecture des données, le G64 doit remettre ce bit à "Ø" pour signifier au système que les données ont été acquises par le microprocesseur local. Chaque transition de WF remet à zéro le pointeur d'adresse de la mémoire d'écriture.

SR Service Request : Il est positionné à "1" par le G64 et remis à zéro par une commande MPX sélective. Il sera traité par le système dans le sens d'une "interruption" générée par l'équipement utilisateur.

SF Subsystem Flag : Positionné à "1" par le G64 et remis à zéro par une commande MPX. Il peut signaler un défaut dans l'équipement utilisateur. (A définir avec plus de précision.).

BU Busy : Il peut être positionné à "Ø" ou à "1" par le G64 pour signifier au système que les échanges de données ne sont pas souhaités ou que le microprocesseur est occupé.

Ce bit sera également mis à zéro lors d'une remise à zéro de l'interface.

IT Interrupt : Ce bit est positionné par l'écriture sélective d'un "1" issu d'une commande MPX.

Sa remise à zéro est faite soit par l'écriture sélective d'un "1" par le G-64, soit suite à l'écriture du bit RES, soit par une remise sous tension.

RRP Reset Read Pointer : Par l'écriture sélective d'un "1" à cette position, le pointeur d'adresse mémoire du tampon de lecture est remis à zéro. Ceci permet au MPX ou au G64 de relire les données dans le tampon de lecture pour une vérification ou un test.

RWP Reset Write Pointer : L'écriture sélective d'un "1" à cette position, fait la remise à zéro du pointeur d'adresse du tampon d'écriture. Ceci permet au G64 ou au MPX de relire les données dans le tampon d'écriture pour une vérification ou un test.

RES Reset interface : L'écriture d'un "1" à cette position fait la remise à zéro des deux pointeurs de mémoires, des deux compteurs de mots, ainsi que des bits du CSR excepté les bits REM et LOC qui gardent l'état précédent. Si l'interface est en mode de fonctionnement à distance (REM=1), une impulsion de 20 µs est transmise sur la ligne "Reset G-64" vers l'équipement utilisateur.

Un cavalier, sur la carte, permet de débrancher cette ligne.

LRR Local/Remote Request : Ce bit est positionné à "1" par une demande de fonctionnement en local ou de remise en fonctionnement à distance par le bouton poussoir de la face avant de l'interface. Il est remis à zéro par l'écriture sélective d'un "1" issu d'une commande MPX - FM2SA1.

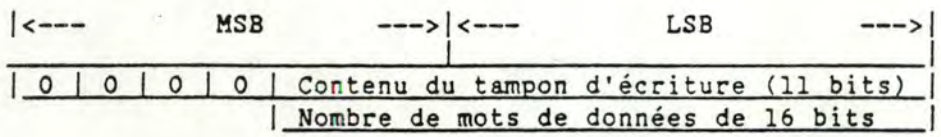
REM Remote : Par l'écriture d'un "1" à cette position, l'interface est mise en fonctionnement à distance. Ce bit est également à "1" après une mise sous tension.

LOC Local : par l'écriture d'un "1" à cette position, l'interface est mise en fonctionnement "local". Le bus d'interfaçage sur le G64 est mis en haute impédance interdisant toute communication entre celui-ci et la carte d'interface ST-G64 .

6. Les compteurs de mots

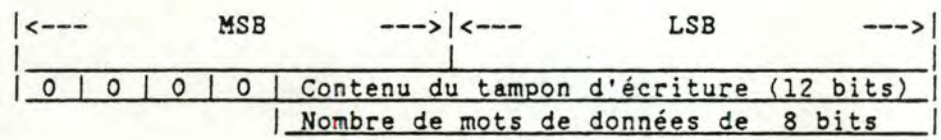
La fonction de ces compteurs est de faire connaître le nombre de mots de données chargé dans les tampons. Ainsi le compteur d'écriture "Write word counter" de 11 bits donne le nombre de mots de 16 bits chargés par le MPX dans le tampon d'écriture et le compteur de lecture "Read word counter" de 12 bits donne le nombre d'octets chargés dans le tampon de lecture par le G64.

- Compteur d'écriture "Write word counter" FM6 SA4/G64 Adr. 4.6



0 tampon vide
1024 tampon plein.

- Compteur de lecture "Read word counter", FM6 SA5/G64 Adr. 5.7



0 tampon vide
2048 tampon plein

7. Fonctionnement à distance ou en local

Un bouton poussoir situé sur la face avant de la station ST.G64 permet à l'utilisateur de formuler une demande de mise en local ou de remise en fonctionnement à distance sous le contrôle de l'ordinateur. Dans les deux cas, local ou distance, la décision est prise par l'opérateur.

- Le bit LRR à "1" dans le registre CSR signifie qu'une demande de mise en local ou à distance a été formulée. En fonctionnement "Local" aucun signal n'est délivré vers le G64, le bus d'interfaçage est mis en haute impédance.

- La fonction Reset (bit RES=1) ne modifie pas le mode de fonctionnement Local ou Distance.

- Une remise sous tension, met la ST.G64 en fonctionnement à distance.

- Le bit REM à "1" dans le CSR signifie que la station est en fonctionnement à distance et le bit LOC à "1" que la station est en fonctionnement local.

8. Interfaçage avec le bus G64

8.1 La ligne d'interruption vers le bus G64

Cette ligne est associée avec le bit IT du CSR elle sera donc activée si le bit passe à "1". Au niveau du connecteur du bus G64 elle n'est pas activée si l'interface est en fonctionnement local. Son utilisation est facultative, un pont câblé permet de la débrancher.

8.2 La ligne reset

Une impulsion de 20 µs est délivrée sur cette ligne lors de l'écriture sélective du bit RES du CSR. Cette ligne n'est pas activée si la station est en fonctionnement local. L'utilisation de la ligne est optionnelle par un pont câblé sur la carte d'interface.

8.3 Les cinq lignes de contrôle

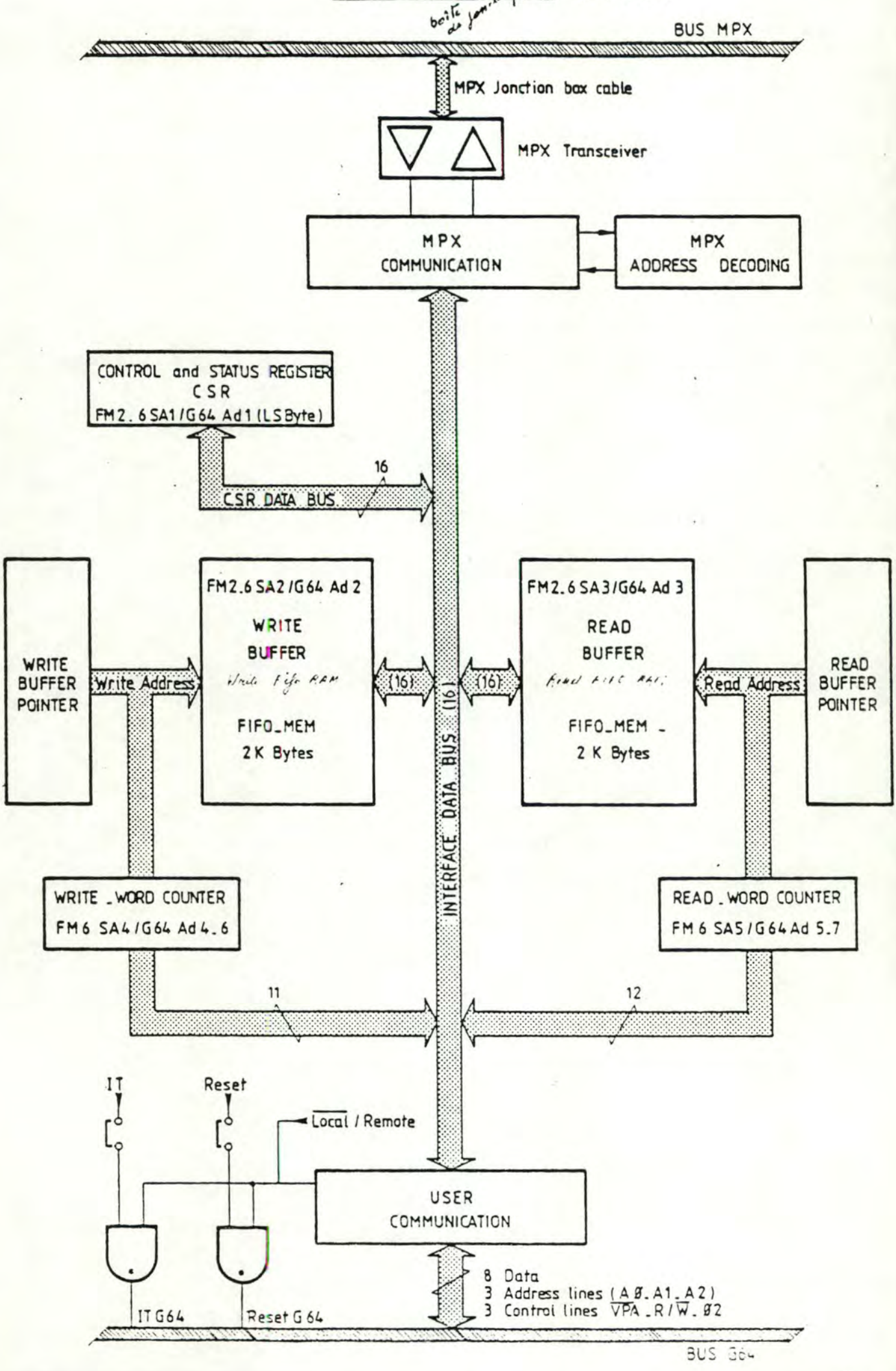
		Bus G64
Valid Address	$\overline{\text{VPA}}$	14a
Valid Data	ENABLE (Ø2)	13b
Read/Write	R/$\overline{\text{W}}$	17a
Reset (option)	$\overline{\text{RES}}$	14b
Interrupt (option)	($\overline{\text{IRQ}}$	16b
	($\overline{\text{FIRQ}}$	17b
	(NMI	15b

Les lignes Reset et Interrupt sont optionnelles par câblage d'un pont sur la carte d'interface.

Distr. :

"SPS Controls Meeting List"

MPX.G64 INTERFACE (ST.G64)

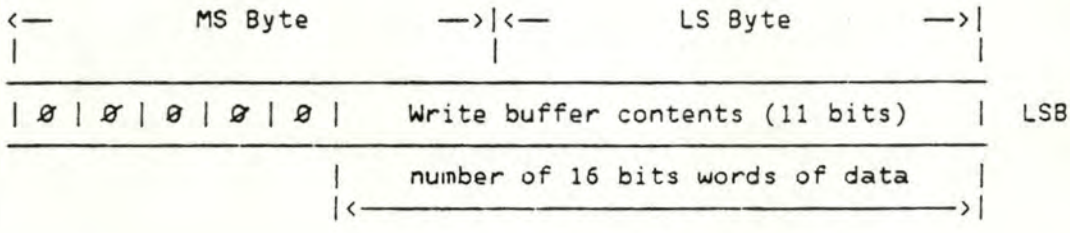


FONCTIONS OPERATIONNELLES dans l'interface ST G64

OPERATION	FONTIONS MPX	ADRES. G64	REMARQUES
Ecriture/lecture du registre CSR	FM2/FM6-SA1	1	G64 n'accède qu'au LSB=HANDSHAKER
Ecriture/lect. du tampon d'écrit.	FM2/FM6-SA2	2	G64 R/W 8 bits - MPX R/W 16 bits
Ecriture/lect. du tampon de lect.	FM2/FM6-SA3	3	G64 R/W 8 bits - MPX R/W 16 bits
Lecture LSB du compteur d'écriture	FM6 - SA4	4	G 64 R 8 bits - MPX R 16 bits (de Ø à 1024)
Lecture MSB du compteur d'écriture		6	
Lecture LSB du compteur de lecture	FM6 - SA5	5	G 64 R 8 bits - MPX R 16 bi (de Ø à 2048)
Lecture MSB du compteur de lecture		7	

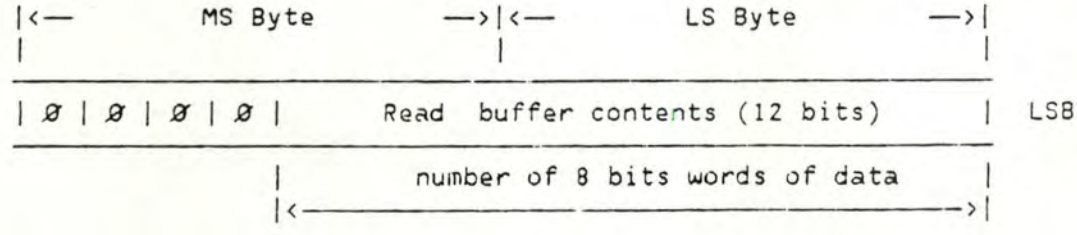
Les compteurs de mots de données en entrée et en sortie des tampons

Write word counter : FM6 SA4 / G64 4.6



Ø Buffer empty
1024 Buffer full

Read word counter : FM6 SA5 / G64 5.7

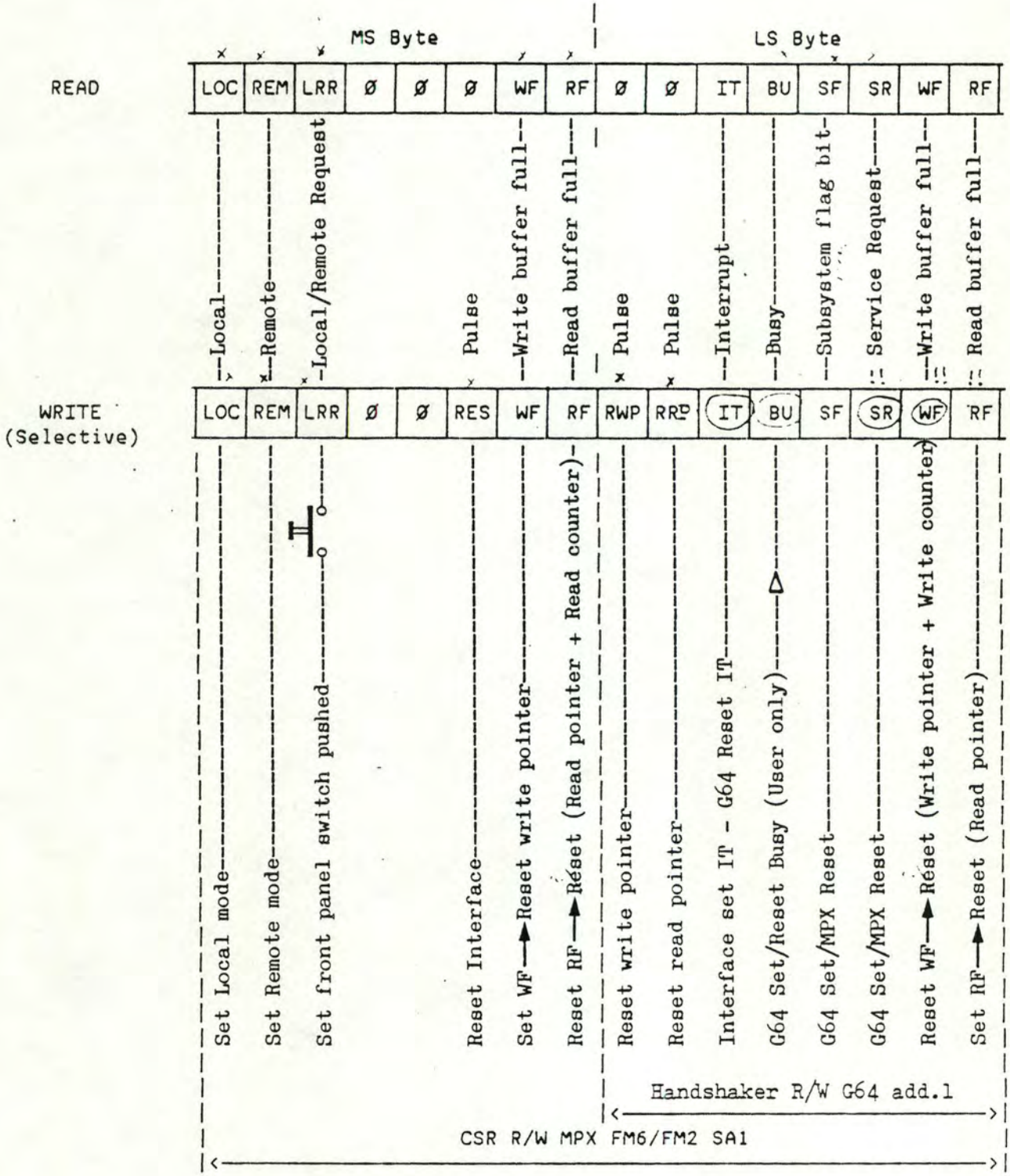


Ø Buffer empty
2048 Buffer full

Fig. 2

CONTROL AND STATUS REGISTER CSR - ST G64

Pourquoi g. a-t-il tant de WF, RF ?



Notes :

- READ CSR
No change

- WRITE CSR

"1" selective write - Ø hold except BU
BU SET/Reset by G 64 only
RES - RWP - RRP - Pulse deliver

Fig. 3

```
 NAM MIX6809

* IRQ - Communications protocol.
* Software Interface in the G64 with the UCS of the NORD-100 which contains
* the M6809, common root in the N100 of all the DMS resident in the G64
* crates equipped with M6809 microprocesseurs and using the MPX-G64 interfaces
* itself.  Dans un souci de garder ce code intact, si on l'integre avec du code
* Pascal, des adresses absolues doivent etre definies pour l'adresse du buffer
* MPX-RT et du buffer de 32 bytes.

 XDEF ENTIRQ
 XREF MPXAD,COMBFV

* ASCII CODE EQUATES

NUL EQU $00
CR EQU $0D
DLE EQU $10
DC4 EQU $14
ETB EQU $17

* EXTERNAL LABEL EQUATES

!0000 EQU $0000
.  FF EQU $FFFF

ADIX FDB MPXAD
ADCB FDB COMBFV
ER1 FCC NUL,DLE,"WRONG WRITE FLAG",CR
ER3 FCC NUL,'"',"UNKNOWN PROPERTY"
 FCC " IN MICROPROCESSOR",CR
* ER4 FCC NUL,ETB,"MICROPROCESSOR I"
* FCC "N LOCAL",CR
ER5 FCC NUL,DC4,"WRONG RESET POIN"
 FCC "TERS",CR
ER6 FCC NUL,ETB,"USER WORD COUNT "
 FCC "MISSING",CR

*************************************************
*   START IRQ PROGRAM
*************************************************

ENTIRQ PSHS A,B,DP,X,Y,U    Laisse pour des raisons de compatibilite avec
                            l'emploi du niveau d'interruption FIRQ
 LEAY [ADIX,PCR]
 LDU [ADCB,PCR]
 LEAX 3,Y *ADD. OF READ FIFO.
 STX 26,U *PUT IN COMM.BUFF
 LDA 1,Y
 STA 29,U
* CMPA #$36          Test si micro est en local mais ce test est redondant
* BEQ L80F2          car l'appel a cette routine prouve que le microprocesseur
* ASLA               ne travaille pas en local
* BCS L80C8
* ASLA
* BCC L80D7
* L80C8 LDA 1,Y
* STA 29,U
* CMPA #$36
* BEQ L80F2
 ASLA
 BCS L80F9
 ASLA
 BCS L80F9
L80D7 LDA 29,U
 ASRA
 ASRA
 BCC L80EB
```

ER1 FCB NUL,DLE
FCC "WRONG WRITE FLAG"
FCB CR

ER3 FCB NUL,$22
FCC "UNKNOWN PROPERTY IN "
FCC "MICROPROCESSOR""
FCB CR

ER5 FCB NUL,DC4
FCC "WRONG RESET POINTERS""
FCB CR

ER6 FCB NUL,ETB
FCC "USER WORD COUNT MISSING "
FCB CR.

```
 STA 29,U
 BRA L80FF
L80EB LEAX ER1,PCR
 LBRA ERROR
* L80F2 LEAX ER4,PCR
* LBRA ERROR
L80F9 LEAX ER5,PCR
 LBRA ERROR


*
*
*   FILL DATA FIELD
*
*


L80FF LDA 2,Y
 LDB 2,Y
 STD 2,U
 LDA 2,Y
 LDB 2,Y
 STD 4,U
 LDA 2,Y
 STA 9,U
  JA 2,Y
 STA 6,U
 LDB 2,Y
 LDA 2,Y
 STD 7,U
 LDA 9,U
 CMPA #$FE
 BNE L8123
 LDA 2,Y
 LDA 2,Y
L8123 LDD 4,U
 CMPD #$5453
 BEQ L8133
 CMPD #$4152
 BEQ L813B
 BRA L8198
L8133 LDA 6,U
 CMPA #$54
 BEQ L8143
  RA L8198
L813B LDA 6,U
 CMPA #$52
 BEQ L8159
 BRA L8198
L8143 LDA 9,U
 BMI L8150
 LDD 18,U
 STB 3,Y
 STA 3,Y
 BRA L816D


L8150 LDB 2,Y
 LDA 2,Y
 STD 18,U
 BRA L818D
L8159 LDA 9,U
 BMI L8178
 LEAX 21,U
 LDB 0,X+
L8162 LDA 1,X
 STA 3,Y
 LDA 0,X++
 STA 3,Y
```

```
 DECB
 BNE L8182
L8160 LDA 29,U
 ORA #3
 STA 29,U
 LBRA RETFIN
L8178 LDB 2,Y
 LDA 2,Y
 STD 20,U
 LEAX 22,U
L8182 LDA 2,Y
 STA 1,X
 LDA 2,Y
 STA 0,X++
 DECB
 BNE L8182
L818D LDA 29,U
 ORA #2
 STA 29,U
 LBRA RETFIN

*************************************************
*
*   SEARCH FOR PROPERTY, AND BRANCH ADDRESS
*
*************************************************

L8198 PSHS Y
 LEAY [0,U] *Y=PROP.TAB.ADDRESS
EXPLOR CLRB
 LEAX 4,U *X=POINTS TO PROP.IN.COMM.BUFF
 STY 20,U *STORE IN COMM.BUFF
 LDA 0,Y
 CMPA #$FF
 BEQ L81F5
 CMPA #$20
 BEQ L81C3
 CMPA 0,X
 BEQ L8182
NEXTY LEAY 17,Y
 BRA EXPLOR


L..82 INCB
 CMPB #3
 BEQ L81C5
 LEAX 1,X
 LEAY 1,Y
 LDA 0,Y
 CMPA 0,X
 BEQ L8182
 LDY 20,U
 BRA NEXTY
L81C3 LEAX 2,X
L81C5 LEAX 1,X
 LDA 9,U
 BMI L81E0
 LEAX [3,Y]
 CMPX #L0000
 BEQ L8219
 STX 30,U
 LDA 29,U
 ORA #3
 STA 29,U
 LBRA L81FE
L81E0 LEAX [1,Y]
 CMPX #L0000
 BEQ WINTER
```

```
 LDRA ERROR


*
*
*    INTERNAL ARRAY CALL
*
*


L8219 LEAX [11,Y]
 CMPX #L0000
 BEQ L81F5
 LDD 13,Y
 CMPD #$0000
 BEQ ERROR6
 BRA ADRX
WINTER LEAX [7,Y]
 CMPX #0
 BEQ L81F5
 LDD 9,Y
 CMPD #0
 BEQ ERROR6
ADRX STD 20,U
 LEAX [0,X]
 LDD 7,U
 CMPD #$0001
 BEQ L8259
 SUBD #$0001
 STD 22,U
 STX 24,U
L823F LDD 24,U
 ADDD 20,U
 ADDD 20,U
 STD 24,U
 LDD 22,U
 SUBD #$0001
 STD 22,U
 BNE L823F
 LDX 24,U
L8259 NOP
 TST 9,U
 BMI L828B
  BRA L826D
ERROR6 PULS Y *NO USER WORD COUNT
 LEAX ER6,PCR
 BRA ERROR
L826D LDY 20,U
L826E LDD ,X++
 STB [26,U]
 STA [26,U]
 LEAY -1,Y
 BNE L826E
 PULS Y *RESTORE POINTER INT.
 LDA 29,U
 ORA #3
 STA 29,U
 BRA RETFIN


L828B PULS Y
 LDB 2,Y
 LDA 2,Y
 CMPD 20,U
 BGT L8298
L8295 STD 20,U
L8298 LDA 2,Y
 STA 1,X
```

```
 STA 0,X++
 LDD 20,U
 SUBD #$0001
 BNE L8295
 LDA 29,U
 ORA #2
 STA 29,U
 BRA RETFIN


*
*
*   FILL DATA FIELD
*
*


L81FE PULS Y
 LEAX 2,Y
 STX 10,U
 LEAX 3,Y
 STX 12,U
 CLRA
 CLRB
   D 16,U


*
*
*   DMS USER BRANCH
*
*


 TFR PC,D
 ADDD #$000A
 STD 14,U
 LEAX [30,U]
 EXG PC,X


*
*
    RETURN TO NORD 100
*
*


RETOUR LEAY [ADIX,PCR]
 LDU [ADCB,PCR]
 LDD 16,U
 BEQ RETFIN
 EXG D,X
ERROR NOP
L82C2 LDB 0,X+
 CMPB #$0D
 BEQ L82CC
 STB 3,Y
 BRA L82C2
L82CC LDA 29,U
 ORA #7
 STA 29,U
* RETFIN LDA 9,S GET OLD CC          Supprime car le RTI active par defaut le
*  ANDA #$EF ENABLE IRQ              niveau IRQ d'interruption.
*  STA 9,S  SAVE ON STACK
RETFIN LDA 29,U *BUSY SET TO 0
 ANDA #$EF
 STA 1,Y *WRITE TO HANDSHAKER
```

4. Le logiciel : package d'application MOPC proprement dit

4.1 L'interfacage avec les logiciels utilisés de Mr D'AMICO et Me MICHEL

4.1.1 Justification de l'emploi de ces logiciels

Pour éviter des pertes de temps inutiles, il fut décidé de recourir à du logiciel existant, un Data Module permettant d'un programme Nodal l'envoi à l'equipement, qui, dans ce cas, serait le MOPC de l'ordinateur GP contrôlé, de commandes via la technique des propriétés. Pour répondre également aux nécessités de décentralisation qui se font de plus en plus sentir avec les équipements de plus en plus compliqués à manipuler, le Data Module M6809B de Mr D'AMICO et sa partie interface-G64 MIX6809 de Me MICHEL furent choisis.

4.1.2 Le M6809B de Mr D'AMICO

4.1.2.1 Intérêt du code

L'avantage incontestable de ce code est qu'il permet de descendre directement au niveau de la station G64 tous les paramêtres présents dans l'appel au Data Module et d'y localiser le coding proprement dit des propriétés ainsi que la procédure de branchement à ces dernieres . Dans l'acceptation classique d'un Data Module, il aurait fallu coder dans le tableau AR la commande à envoyer vers l'équipement et le nr d'ordinateur contrôlé car seul ce tableau était transmis au G64 et que ces informations lui sont indispensables. De plus, le coding des proprietes aurait dû être résident dans l'Acces, ce qui impliquait une surcharge pour ce dernier.

Soit l'appel au Data Module MOPC (AR,"R/W",N-VALUE,#PROPERTY) : sur base d'une pseudo data table qui sera présente dans l'Acces et du nr d'ordinateur N-VALUE dans l'appel au Data Module, le message sera aiguillé vers le bon G64 et se présenteront à lui le tableau AR Nodal contenant la commande à envoyer au MOPC soit "1/", le type de l'opération à effectuer soit "R" pour lecture, soit "W" pour écriture, le type de propriété et le nr d'équipement qui n'est rien d'autre que le GP à contrôler.

Il suffit, pour obtenir ce résultat, de définir dans le G64 une table de branchement qui indiquera pour le nr d'ordinateur stipulé, quel canal de quelle carte GESSIO-1A utilisé. Le tour est alors joué et au G64 à réagir en conséquence. (Voir figure nr 2)

Remarque : La notion de pseudo data-table de Mr D'AMICO est un peu tournée car sa table contient normalement autant de lignes qu'il n'y a de microprocesseurs différents accessibles par son Data Module et les first N-value et last N-value désignent, dans son esprit, le premier et le dernier en ordre croissant des équipements connectés à un micro. Comme dans le cas qui nous préoccupe, les ordinateurs présents dans un Bâtiment Auxiliaire n'ont pas de numéro d'ordre successif, les lignes, le first et le last N-value désignent le nr d'équipement ou le GP.

Soit MOPC (AR, "W", (1), # Property)

	μP status	μP MPX-adress	μP logical number	first N. value processed by the μP	last N. value processed by the μP	μP pseudo N. value
1		G64 n°1		1	1	1
2				2	2	2
3						
4						
.						
.						
.						
125						
126						
127						
128						

Pseudo data-table dans L'Accès

bien que pour le moment, il n'y ait que 64 GP#

G64 n°1

avec sa table de branchement

GP1 (canal 2) GP4 (canal 2) GP10 (canal 3)

OPCOM

GP1

FIGURE n° 2

4.1.2.2 Remarques concernant le M68098

Le problème tourne autour de l'emploi du string ABMERR pour le renvoi à l'utilisateur de messages d'erreur taillés sur mesure, ayant une adresse-mémoire bien précise, ce qui implique un coding non portable tel quel sur d'autres configurations.

- ABMERR n'est jamais réinitialisé après un appel Data Module MOPC
- Ce manque de portabilité pourrait être résolu par l'emploi d'un paramêtre optionel après la spécification de la propriété dans l'appel au Data Module. Il suffirait alors d'étendre la liste possible des erreurs et quand l'interpréteur en rencontre une, il charge le paramêtre de son numéro et passe à la prochaine commande. Dans le programme d'application, si le code retourné est différent de zéro, il faudrait recourir à un fichier d'équivalence pour sortir le libellé de l'erreur au niveau du terminal.

Voici la liste des messages renvoyés par ABMERR:

- Propriété non implémentée à ce jour avec, entre parenthèses, la commande définie dans le tableau AR
- Opération en cours à interrompre
- Seuls 20 canaux sont attribuables
- Ancien canal dédicacé à l'ordinateur
- Ordinateur non connecté à un canal
- La carte de l"OPCOM n"est pas connectée
- Label or Checksum (loading) error or Incompatible mode
- L"OPCOM n"est pas enclenché
- TEST...Buffer empty
- Overflow du buffer cyclique
- Ordinateur actuel de dialogue
- Cle d"accès mal/non définie..."(---)"--> G64 réservé ?
- G64 sous réservation manuelle
- Opération de test non initialisée
- La BREAKPOINT INSTRUCTION n"est pas disponible
- G64 busy
- Aucun canal initialisé par un dialogue

A l'état de projet:

- DUMP terminé
- Découpe trop large du fichier de test à charger en mémoire

4.1.2.3 Un peu de rigueur

Comment se présentent les paramêtres au G64 ? Comme le Write Buffer de la carte RT est de 2Kbytes, des 1024 mots transmissibles,

- 3 bytes sont employés pour la propriété
- 2 bytes pour la N-Value ou le nR d'équipement
- 1 byte pour le Flag renseignant sur le type d'opération à effectuer
- 2 bytes pour chaîner les commandes aux données
- 2 bytes pour renseigner sur le nombre de mots descendus de l'Acces vers le G64 (Word Count)

et 1019 mots sont disponibles pour la descente des données.

Dans le sens G64-Acces, la definition du Word Count n'est pas nécessaire car la dimension du tableau AR définit le nombre de mots à lire dans le Read Buffer du G64. En théorie, 1023 mots pourraient être lus mais le Time-Out de 100 ms de Mr D'AMICO et le temps de transfert d'un mot de 16 bits de données entre l'unité de contrôle dans le Camac et le Read Buffer de l'ordre de 100 microsec.font qu'au grand maximum 1001 mots peuvent être lus sans risque.

4.1.3 Le MIX6809 de Me MICHEL

4.1.3.1 Emploi du code

Ce code sert d'interface avec le Data Module de Mr D'AMICO, de driver des 2 cartes MPX-RT et d'interface avec l'application constituée, en fait, du code des propriétés appelables du Data Module MOPC.

4.1.3.2 Interfacage avec l'application Pascal proprement dite

Pour plus de détail, consultez la note SPS/ABM/Note/84-06.

Le code MIX6809 a besoin de deux zones :

- la zone du Communication Buffer (variable BUFCOM dans les fichiers DECVARE.TXT et DECVARL.TXT sans système FLEX, dans les fichiers BDECVARE.TXT et BDECVARL.TXT avec système FLEX). Cette zone est remplie par le MIX6809 avant de passer la main à l'application Pascal et contient :
  - le libellé de la propriété ASCII
  - le nr d'équipement dans l'appel au Data Module
  - le type de l'opération (Simple Read, Read d'un tableau, Simple Write ou Write d'un tableau)
  - l'adresse du Write Buffer de la carte MPX
  - l'adresse du Read Buffer de la carte MPX
  - l'adresse de retour dans le code MIX6809 après traitement de la propriété demandée par l'opérateur.

  Deux informations doivent cependant lui être communiquées :
  - l'adresse de la table des propriétés. Cela est effectué à l'exécution par l'instruction BUFCOM.PTR-PROPERTY:= ADDR PROPRIETE; présente dans le fichier MOPCIF.TXT sans système FLEX et BMOPCIF.TXT avec système FLEX (pour comprendre le sens de ADDR, se référer à l'Omegasoft Pascal Language Handbook).
  - l'adresse du message d'erreur mais seulement dans le cas où l'utilisateur veut envoyer un message taillé sur mesure au niveau de l'opérateur. Cela est effectué à l'exécution par l'instruction BUFCOM.PTR-ERREUR:= ADDR(ERREUR); de la procédure Wmessage du fichier PRIMIT.TXT.

- la zone des propriétés terminée par le byte $FF. Chaque propriété nécessite 17 bytes pour sa description avec, entre autres, son nom, les points d'entrée au coding exécutant une requête "write"/"read". Cette table est utilisée par MIX6809 pour permettre le branchement sur le coding de la propriété réclamée par l'opérateur après avoir analysé le type de cette dernière. Cette table peut être initialisée par la procédure CREATE-PTY présente dans le fichier PROP.TXT sans système FLEX et BPROP.TXT avec système FLEX, le caractère de fin $FF étant ajouté par la procèdure END-PTY présente dans les mêmes fichiers. Cette table est initialisée actuellement dans le programme principal contenu dans le fichier MOPCIF.TXT sans système FLEX ou BMOPCIF.TXT avec système FLEX et elle contient, pour le moment, huit propriétés.

  Ex. d'appel : CREATE-PTY (1,'TPR',ADDR(NOTIMP),ADDR(RTRANS)); qui crée dans la table le descriptif de la propriété $TPR, NOTIMP et

SPS/ABM/Note/84-06 8/8/84

INTERFACE SOFTWARE 6809 POUR LA LIAISON MPX-G64

H. Michel

I. Structure des tables utilisées

Le programme d'interface MIX 6809 a besoin d'une location de mémoire dans laquelle l'utilisateur doit placer l'adresse du communication buffer.

Structure du communication buffer (32 bytes)

(a) 2 bytes : pointeur à la table des propriétés ←
2 bytes : DMS status (à définir)
3 bytes : PTY (en ASCii)
2 bytes : N-value
1 byte : Flag (+1 = single RD, -1 = single WR)
(+2 = array RD, -2 = array WR)
2 bytes : adresse du "Write Buffer" (EC42) } sur l'interface
← 2 bytes : adresse du "Read Buffer" (EC43) } MPX
2 Bytes : adresse de retour 11BA
(b) 2 bytes : pointeur au message d'erreur ←
← 14 bytes : working area

adresse de début du communication buffer + 2E

Write + 29 :

30' : handshake
32' : adresse de [illegible]

(a): Information fournie par l'utilisateur à l'initialisation.
(b): Pointeur mis à Ø et chargé par la DMS de l'utilisateur en cas d'erreur.

Les tables des propriétés et des erreurs sont sous la responsabilité de l'utilisateur.

Structure de la table des propriétés

	3 bytes	: PTY (en ASCii)
(A)	2 bytes	: point d'entrée au coding exécutant une requête "Write"
(B)	2 bytes	: point d'entrée au coding exécutant une requête "Read"
	2 bytes	: status (à définir)
(C)	2 bytes	: adresse du vecteur contenant l'adresse du User Buffer "Write"
(D)	2 bytes	: word count du User Buffer "Write"
(E)	2 bytes	: adresse du vecteur contenant l'adresse du User Buffer "Read"
(F)	2 bytes	: word count du User Buffer "Read"

Ce bloc de 17 bytes est à répéter pour chaque propriété.
La table est terminée par 1 byte : $ FF.

Lorsque le traitement d'une propriété consiste simplement en un transfert de donnée, cela peut être fait directement dans l'interface. Il suffit pour cela de mettre à Ø la zone (A) et/ou (B) et de remplir les zones (C), (D) et/ou (E), (F) suivant que la propriété s'exécute en "Write" et/ou en "Read". (Le word count est donné en words, il sert également d'incrément dans le buffer en fonction de la N-value. Pour un simple call, le word count est à 1.)

Structure du message d'erreur

2 bytes	: nombre de bytes du libellé
1 à 78 bytes	: libellé de l'erreur (en ASCii)
1 byte	: $ØD (carriage return en ASCii) pour la fin du message d'erreur.

II. Configuration du stack avant le branchement à la DMS utilisateur

Le programme MIX 6809 est lancé par FIRQ. Le stack contient alors le PC (Program Counter) et le CC (Condition Code Register). Avant de lancer la DMS, MIX 6809 sauve les registres du niveau interrompu dans la stack.
La configuration du stack à l'entrée de la DMS est donc la suivante :

stack ↑	1 byte	: A ← hardware stack pointer
	1 byte	: B
	1 byte	: DP
	2 bytes	: X
	2 bytes	: Y
	2 bytes	: U
	1 byte	: CC
	2 bytes	: PC

Au retour de la DMS dans MIX 6809 (via l'adresse du retour du communication buffer), les registres du niveau interrompu sont restitués.

III. Obligations de l'utilisateur

Le MIX 6809 peut être mis sur EPROM et chargé à n'importe quelle adresse choisie par l'utilisateur.

L'utilisateur doit :

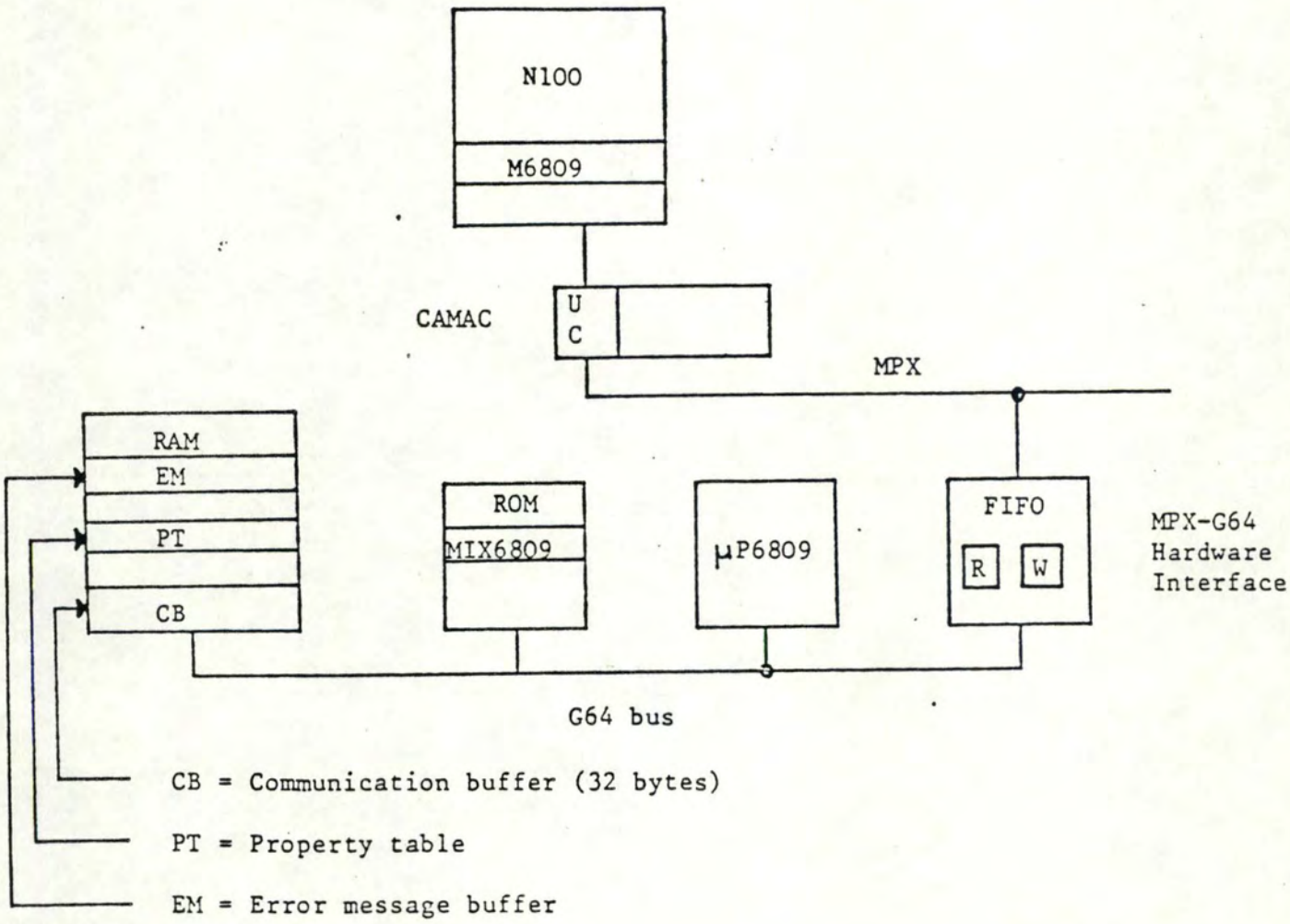
a) Fournir une zone de mémoire de 32 bytes pour le communication buffer qu'il peut placer à son gré.

b) Charger le point d'entrée du MIX 6809 dans la location associée à l'interrupt FIRQ (FFF6-7). faut aussi le sigle de l'IRQ...

c) Charger le pointeur au communication buffer ~~dans~~ avec la location de mémoire qu'il a définie.

d) Charger à l'initialisation le pointeur à la table des propriétés dans les 2 premiers bytes du communication buffer.

e) Construire les tables de propriétés et de messages d'erreur.



LIAISON N100 - µP 6809

RTRANS sont des labels définissant le point d'entrée aux procédures de traitement de la propriété en Write/Read call. Ces labels sont définis par XDEF dans les fichiers MOPCIF.PS sans FLEX ou BMOPCIF.PS avec FLEX, fichiers contenant le traitement des propriétés et déclarés PCR dans les fichiers DECVARE.TXT sans FLEX ou BDECVARE.TXT avec FLEX reprenant les déclarations globales au programme principal contenu dans MOPCIF.TXT/BMOPCIF.TXT. (Consultez l'Omegasoft Pascal Language Handbook pour de plus amples renseignements sur ADDR, PCR, XDEF)

4.1.3.3 Niveau de l'application

La première chose à effectuer pour l'application, s'il s'agit d'une propriété en écriture, est de lire le Write Buffer de la carte MPX-RT; cela est effectué par la procédure READBWS du fichier PRIMIT.TXT pour la lecture de 20 bytes ou par la procédure READBWC dans le même fichier dans le cas de 2000 bytes.

S'il s'agit d'une propriété en lecture, l'écriture des résultats de la part du G64 dans le Read Buffer s'effectue par la procedure TRANSREAD présente dans le fichier PROP2.TXT sans système FLEX ou BPROP2.TXT avec système FLEX. Cette procédure, suite à un accord avec l'opérateur, envoie dans les deux premiers bytes du Read Buffer le nombre de bytes utiles qui y sont présents pour éviter au niveau du programme d'utilisation de l'opérateur l'affichage à l'ecran de caractères non significatifs.

4.1.3.4 Passage du code Assembleur au code Pascal

Le passage du MIX6809 écrit en Assembleur au programme Pascal nécessite un certain nombre de précautions dont vous trouverez l'explication dans l'Omegasoft Pascal Language Handbook au chapitre Assembly Language Interface et à l'appendice C Runtime Environment. Ce passage est effectué au niveau de l'appel aux differentes propriétés dans les fichiers MOPCIF.PS sans FLEX et BMOPCIF.PS sous FLEX.

4.1.3.5 Problèmes rencontrés pour intégrer le MIX6809 dans l'application Pasca

1) Suite à la volonté de Mr GHINET de garder le MIX6809 intact, il a fallu user de subterfuges pour que les instructions

```
ADIX FDB MPXAD
ADCB FDB COMBFV
```

du MIX6809 mettent réellement aux adresses ADIX, l'adresse de base des cartes MPX-RT (à changer si l'adresse de base venait à être modifiée) et ADCB, l'adresse d'une position-mémoire contenant l'adresse de début du Communication Buffer BUFCOM (technique d'indirection). Cela a été realisé dans les fichiers MOPCIF.PS/BMOPCIF.PS par les instructions

```
MPXAD EQU $E040
BUFCOM EQU DSEND-31   adresse de début du Communication Buffer
COMBFV    FDB   BUFCOM
```

utilisées au moment du link et du calcul des adresses absolues. Cette adresse de debut devait être connue au moment de l'assemblage du code car ce dernier se trouve en EPROM et elle ne pouvait donc pas être communiquée au moment de l'exécution. Il a donc fallu définir dans le Data Stack Pascal une zone fixe attribuée par défaut au Communication Buffer et c'est le rôle de l'instruction BUFCOM EQU DSEND-31. Au niveau Pascal cette zone est définie en la déclarant comme PCR (voir variable BUFCOM dans les fichiers

DECVARE.TXT, DECVARL.TXT ou BDECVARE.TXT, BDECVARL.TXT et l'Omegasoft Pascal Language Manual pour l'explication de la déclaration en PCR) et au niveau Assembleur, MPXAD, COMBFV sont déclarés XREF et BUFCOM XDEF. Il est intéressant de se rappeler que le byte le plus significatif d'une variable se trouve toujours à l'adresse la plus haute. Ainsi, l'adresse de départ d'une variable de 32 bytes occupera les positions X-31 à X.

2) Pour le retour du programme Pascal à l'Assembleur, il a fallu tenir compte du fait que le registre Y est utilisé comme Global Stack Mark (voir Appendice C de l'Omegasoft Pascal Language Handbook) par le Pascal et qu'il est utilisé dans le MIX6809 comme pointeur vers le Communication Buffer. Voila le pourquoi de l'instruction LDY #BUFCOM dans le fichier MOPCIF.PS/BMOPCIF.PS dans les routines de traitement des propriétés.

Mes programmes étant auto-documentés, consultez, si nécessaire, les listings des disketttes sans FLEX/avec FLEX.

4.1.3.6 Test de l'intégration correcte du MIX6809 dans le code Pascal

Tests Nodal disponibles pour l'opérateur vérifier la bonne intégration des codes étrangers utilisés :

```
1) SE MOPC (1,#TST) = 8
   T MOPC (1,#TST)
2) DI-I A(2); SE A(1) = 7; SE A(2) = 9;
   MOPC (A,"W",1,#ARR)
   DI-I B(2)
   MOPC (B,"R",1,#ARR)
   F I=1,2; T B(I)!
```

4.2 Logiciel proprement dit réalisant le Data Module au niveau G64

4.2.1 Ossature de l'application

Il s'agit d'un package conduit par interruption. Le programme principal se trouvant dans le fichier MOPCIF.TXT/BMOPCIF.TXT, après avoir initialisé les variables du problème, boucle sans fin en attente d'interruption. Sur le Motorola 6809, six niveaux d'interruption sont utilisables mais deux suffisent pour nos besoins : l'interruption FIRQ en provenance de la carte GESSIO-1A indiquant soit la possibilité d'émettre un caractère vers le MOPC d'un NORD-100, soit l'invitation à lire un caractère en provenance du MOPC ou les deux suivant la maniere dont le registre de contrôle du canal de la carte aura été programmé; celle-ci est prioritaire sur l'interruption IRQ en provenance des cartes MPX-RT.

Avant d'aller plus loin, il est conseillé de lire la brochure "How to use interrupts in Omegasoft Pascal (V2.1)" d'Alistair Bland car elle servira d'article de référence.

Les différences à noter sont que :

1) EXAMPL.TXT et EXAMPL.PS doivent être remplacés par MOPCIF.TXT/BMOPCIF.TXT et MOPCIF.PS/BMOPCIF.PS. Il est intéressant de prendre les fichiers jumeaux et de les comparer.
2) Deux niveaux d'interruption FIRQ et IRQ avec deux stacks indépendants sont utilisés.
3) Que le programme principal tourne avec FLEX ou non, il vaut mieux définir un petit fichier annexe en code machine soit VECTORS sous FLEX, soit BVECTORS sans FLEX, qui sera utilisé au moment du linkage pour charger les vecteurs d'interruption avec les adresses des routines de service. Il est vrai que si le système tourne sans FLEX et moniteur, FFF6 et FFF7 sont les adresses du vecteur d'interruption FIRQ, FFF8 et FFF9 celles du vecteur IRQ, sinon ces adresses sont occupées par le moniteur en ROM et pointent par indirection vers les adresses en RAM DFC6 et DFC7 pour le vecteur FIRQ, DFC8 et DFC9 pour le vecteur IRQ. Ce chargement, effectue au linkage, est conduit par les fichiers MOPCIF.CF/BMOPCIF.CF.
4) Les routines d'activation, de désactivation des niveaux d'interruption sont utilisées dans le programme Pascal par appel à des procédures Assembleur définies comme EXTERNAL : firqon, firqof, irq-on, irqoff dont le code se trouve dans MOPCIF.PS/BMOPCIF.PS où elles sont declarées comme XDEF. Pour comprendre les notions de procédures declarées comme ENTRY, EXTERNAL, consultez le chapitre PROCEDURE de l'Omegasoft Pascal Language Handbook.

La routine servant l'interruption IRQ n'est rien d'autre que le MIX6809 en code machine de Me MICHEL dont le point d'entrée est étiqueté par le label ENTIRQ dans BYCYINT.TXT. Ainsi, dans le fichier VECTORS ou BVECTORS on trouve à l'adresse DFC8 ou FFF8 FD8 ENTIRQ, ENTIRQ ayant été déclaré au préalable par XREF ENTIRQ signifiant que l'on utilise dans le module une variable déclarée extérieurement par XDEF ENTIRQ et ce dans le fichier BYCYINT.TXT contenant le MIX6809.

Dans le fichier MOPCIF.PS/BMOPCIF.PS, on a trace de la routine FIRQ servant l'interruption FIRQ et préparant le contexte Pascal avant de sauter dans la routine de service de l'interruption FIRQ proprement dite écrite en Pascal et portant le nom FQINT. Cette procédure se trouve dans le fichier

HOW TO USE INTERRUPTS IN OMEGASOFT PASCAL (V2.1)

WARNING - do not use the linkage creator LC on program EXAMPLE or you will delete EXAMPLE.PS. which has been modified for interrupts.

The explanation of interrupts in the omegasoft pascal manual is not sufficient to allow their use. What follows is an example to show how they may be used in practice.

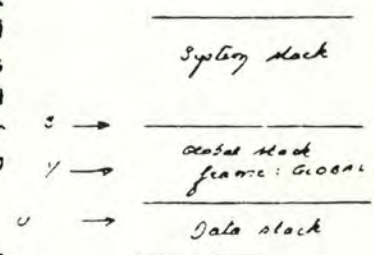
The example assumes the following:

1. you are developing software for a standalone G-64 system ie no flex, no monitor but that you are testing it on a flex system for the moment.

2. you have some hardware to generate interrupts. I have used an isr 7530 double via card.the example program may be modified for any other hardware.

The example program consists of 2 source files. EXAMPLE.TXT contains the pascal main program, pascal interrupt routine and all the hardware definitions etc. EXAMPLE.PS is an assembly language file containing the startup code to initialise the 'pascal machine' and the machine code part of the interrupt.

A pascal program uses two stacks, a system stack to push return addresses and intermediate calculations for procedure and function calls and a data stack which is used for local variables and parameter passing. The system stack is pointed to by the s-register and the data stack by the u-register. In addition there is the 26 byte global stack frame which normally sits between the system stack and the user stack. It contains pointers to the input output descriptors,error flags and stack limits for runtime stack checking etc. The y-register points to the global stack mark which is one of the entries in the global stack frame.



When an interrupt occurs the system stack will be used to push the current registers and program counter. Any interrupt routine will then put its procedure/function call return addresses on this stack. With the y and u registers the situation is more complicated. These could be pointing at anything, especcially if the interrupt accured while one of your machine code routines which used the y and u registers was running. Therefore the interrupt routine must initialise the y and u registers. The y to point to the global stack mark and the u to an area of memory which can be used for the interrupt routines, local variables and parameters. In addition the A register must be set to -1 to indicate the lexical level difference between your machine code interrupt routine (lexical level 1) and the pascal interrupt procedure it calls (lexical level 2). (For more explanation refer to the manual).

The reason a separate system stack for interrupts is not required is that it is assumed that you have not been using it as a general purpose register, ie the memory below where it is pointing is ram and does not contain data.

Reason of the 256 bytes of interrupt data stack

I have set up the stacks as follows:

system stack	$BFFF - $BF00 256 bytes
spare bytes	$BEFF - $BEFC in case the system stack overruns

↘ + System

global stack frame	$BEFB - $BEE2 26 bytes
main data stack	$BEE1 - $AEE2 4k bytes
interrupt data stack	$AEE1 - $ADE2 256 bytes

The global stack mark is located 10 bytes from the bottom of the global stack frame ie $BEEC. The starting position and the number of bytes allocated can be changed according to requirements, except for the global stack frame. Pascal performs run time error checking of the stacks using the values loaded into the global stack frame during startup. In this case the system stack limit will be at $BF00 and the data stack limit will be at $ADE2. Note that overrun of the main data stack into the interrupt data stack is therefore not checked.

The order of the stacks should not be changed unless you understand how the stack limit checking works, otherwise you will get 'run time error £20' all the time.

If you decide to change the names of your pascal main program and interrupt routines then you must change the calls to them in EXAMPLE.PS and the external references (XREFs). Similarly if you wish to change the name of the assembly language routine irq then its XDEF must be changed and the referenee in the pascal main program. To use an firq interrupt instead of an irq interrupt remember to push all the registers aswell. Using both types simultaneously will mean making an extra firq stack. NMI and SWI may also be used.

In the pascal main program you must for flex operation change the irq vector to point to the irq routine. The actual vector is in the monitor rom and unchangeable but it points to some code which uses location $DFC8 in ram as a modifiable vector. (normally this points to an RTI instruction). Naturally you do not know the absolute address of the irq routine because it is decided at link time so the pascal function addr can be used to find it at runtime. The irq routine must therefore be externally referenced in the pascal program. The compiler will produce an XREF to irq for the declaration 'irq:hex per;', Loading the irq vector is then done by 'irq_vec:=addr(irq);'.

In a standalone system you will have make a rom to sit at the top of memory with the reset and interrupt vectors pointing to the start of the program and the irq routine respectively. This is best done by making a small assembly language file using FDB directives to point to them and XREFs so that the linker can calculate the values to load at link time. The absolute binary output from the linker can then be romed directly.

The 6809 processor interrupts must be enabled at some time. I have done this in the main program by calling the assembly language routine IRQ_ON declared as external in the main program, declared by XDEF in EXAMPL.PS.

Remember that writing interrupt service procedures needs careful thought about the way you use variables and non re-entrant procedures. Code that uses fixed memory locations such as global variables for scratchpad calculations or writes to hardware cannot normally be re-entrant. For example using writeln in your interrupt procedure and in your main program means that output from the interrupt routine may be scrambled with output from the main program. The example program will do this all though it is not noticeable because only one character is output in the main program. What is needed therefore is a method of communication between the interrupt routine and the main program such as

semaphores or to disable interrupts during critical sections of code.

For a standalone program the input, output and error drivers in the pascal runtime library cannot be used because they use flex and monitor routines. I have written standalone drivers for the signetics 2661 ACIA which are linked in instead of the standard drivers. They can easily be changed for different hardware configurations.

Another thing to remember about standalone programs is that they must never finish as there is nowhere to go afterwards so it best to make the main program a repeat...until false loop. If the pascal program ever got to the 'end.' statement it would return to the address on the top of the stack, which could be anywhere! This is why the example program can only be stopped by a reset or power off.

Final points:

The debugger cannot easily be used with interrupts or modified drivers and so testing of programs must be performed by linking the modules (normally using a chain file). There are many reasons for this:

1. the stack set up code has been modified.
2. the pascal program externally references the machine code and vice-versa.
3. you cannot fully test a real time system at half speed. (debug is slow).
4. the debugger uses the standard drivers only.

In practice, if there is a routine that can only be debugged with the debugger then it best to make a special program to test it without interrupts etc.

The chain file required to assemble and link these modules is as originally produced by the linkage creator. If you change the names of the files, starting address or add more modules then EXAMPLE.CF must be edited as required.

Inorder to understand the pascal environment, which is what this note is all about, you must understand exactly what the .PS files and the .CF files made by the linkage creator do and how to modify them for your own application.

end of note.txt.

```
 TTL startup and interrupt for example pascal program using interrupts
 NAM START
 XDEF START.BIOSIN.IRQ,IRQ.ON
 XREF EXAMPL.INT            ;note the 6 character limit on XDEFs and XREFs

* stack areas in ram - note that 6809 stacks go downwards
SSBEG EQU $CDDD       ;<-S main ! start stacks
SSEND EQU SSBEG-256   ;         ! system stack of 256 bytes
GLSTM EQU SSEND-20    ;<-Y      ! 16 bytes (+4 for safety) above GLSTM ! stack
DSBEG EQU GLSTM-10    ;<-U main ! 10 bytes below global stack mark     ! frame
DSINT EQU DSBEG-4096  ;<-U int  ! 4k byte main data stack
DSEND EQU DSINT-256   ;         ! 256 byte interrupt data stack

* pascal startup routine - reset vector must point here for standalone system
START LDS #SSBEG ;set system stack beginning
 LDU #DSBEG      ;set data stack beginning
 LDY #GLSTM      ;set global stack mark
 LDX #DSEND
 STX -2,Y        ;set data stack limit
 LDX #SSEND
 STX -4,Y        ;set system stack limit
 STY -6,Y        ;set global mark pointer
 CLR -7,Y        ;disable errors
 LDX #$FFFF
 STX 0,Y         ;set heap limit (no heap)
 LDX #$0000
 STX 2,Y         ;set heap start (no heap)
 CLR 4,Y         ;no conversion error yet
Pour utilisation sans FLEX ]   avec FLEX
 LBRA EXAMPL     ;jump to pascal main program EXAMPLE

* irq interrupt routine - irq vector must point here
IRQ LDU #DSINT   ;set interrupt data stack
 LDY #GLSTM      ;set global stack mark
 LDA #-1         ;set lexical level to -1
 LBSR INT        ;call pascal interrupt procedure INT
 RTI             ;return from interrupt

 END START
```

j'utilise des fonctions bas niveau sur un stack (Ron NEN, Ron RELEASE)

```
START
 LDX  $CC14
 STX  6,Y   start of parameter area
 LBSR EXAMPL
 JMP  $CD03   go to warmstart
BIOSIN JMP [$D385]  keyboard input routine vector
```

```
* IRQ-ON    - ACTIVATE IRQ INTERRUPT
IRQ.ON  ANDCC #$CF
        RTS
```

```
{EXAMPLE PASCAL PROGRAM PROGRAM USING INTERRUPTS - alastair bland 19/9/84}
{to compile          : pc (example ) o
 to assemble and link : chain example
 to run              : 1.example.bin
 to stop             : (abort, reset or power off)
 this program uses the facilities provided by flex and the standard monitor.
 the hardware used to generate the interrupt is an isr 7530 double via card
 and a switch on the cb1 port line of ic6 which is one of the 6522 via chips.
 the program can be modified to the desired hardware configuration.
}

program example(input,output);
const
 via_addr=$E100;{address of isr 7530 via card}
var
 iorb1:byte at via_addr+$0;{register to be read to clear cb1 interrupt}
 ifr1:byte at via_addr+$D;{interrupt flag register for ic6}
 ier1:byte at via_addr+$E;{interrupt enable register for ic6}
 pcr1:byte at via_addr+$C;{peripheral control register for ic6}

 irq:hex pcr;{assembly language irq routine}
 delay:integer;{delay loop variable}
 count:integer;{interrupt count}
procedure IRQ_ON ; EXTERNAL;
procedure int:entry;
var
 test:byte;{dummy variable}
begin
 {this pascal interrupt procedure is called by the machine code routine irq
  when there is an interrupt. irq is in file example.ps.
 }
 if (ifr1 and £$90)=£$90
 then
 begin
  {source of interrupt was ic6. cb1 port line}
  test:=iorb1;{clear interrupt source}
  count:=count+1;
  writeln;
  writeln('this is interrupt number ',count:0);{on interrupt display a message}
 end
 else
  writeln('where did that interrupt come from ?');
end {of procedure int};


begin {main program}
 {$S+,R+,C+ enable runtime error checks for debugging}

 {interrupt enabling}
 IRQ_ON;

 {setup line cb1 on ic6 for -ve edge triggered interrupts}
 pcr1:=£%11101111;{bit 4 cleared for -ve edge on cb1}
 ier1:=£%10010000;{bit 4 set to enable cb1 interrupt}
```

```
{initialise variable used by the interrupt procedure}
count:=0;
writeln;

{main program loop}
repeat
 for delay:=1 to 10000 do delay:=delay;{delay loop}
 write('.');
until false;

end.
```

MOPCIF.TXT/BMOPCIF.TXT déclarée EXTERNAL et déclarée XREF dans le fichier MOPCIF.PS/BMOPCIF.PS pour indiquer l'utilisation d'une procédure déclarée extérieurement. De même que pour IRQ, on trouve dans le fichier VECTORS/BVECTORS à l'adresse DFC6 ou FFF6 FDB FIRQ avec FIRQ déclarée comme XREF et déclarée XDEF dans le fichier MOPCIF.PS/BMOPCIF.PS.

Pour comprendre le sens de XREF, XDEF, FDB..., il est vivement conseillé de consulter l'Omegasoft Pascal Language Handbook.

4.2.2 Les propriétés

4.2.2.1 Bref récapitulatif

Suite aux interruptions IRQ génerées par l'appel au Data Module MOPC par l'opérateur, le MIX6809 analyse le type de la propriété demandée et, sur base de la table des propriétés, saute dans le code correspondant. Voilà où nous sommes, que se passe-t'il alors ?

4.2.2.2 Spécification des propriétés

On peut distinguer trois types de propriétés :
- celles qui s'adressent uniquement au G64
- celles directement liées au MOPC du NORD-100 contrôlé
- celles liées au chargement de programmes de test binaires dans le NORD-100 à contrôler.

4.2.2.2.1 Propriétés relatives au G64

Elles permettent :

1) d'effectuer une réservation manuelle du G64, ce qui implique que le G64 est dédicacé à l'usage exclusif de l'opérateur qui l'a réservé.

   Cette réservation doit être annulée explicitement en stipulant le nom sous lequel l'opérateur s'est fait connaître au G64. Si une autre personne tente de s'imiscer, sa demande est rejetée avec stipulation de la personne actuellement en session de travail avec le G64. Cette précaution permet, au cas ou un G64 serait utilisé abusivement par un opérateur, d'annuler sa session en stipulant son nom et éviter ainsi des réservations infinies.

   Pourquoi cette réservation ?
   - Certaines commandes énvoyées au MOPC comme la Memory Dump Instruction peuvent être interrompues par le simple envoi d'un caractère compris entre A-Y et 0-9. Il faut éviter que, par mégarde, deux opérateurs non conscients de leur présence mutuelle se gênent.
   - Tout appel au Data Module réserve le MOPC implicitement pour une durée de 100 msec. Malheureusement, aucun dialogue avec le MOPC ne peut se résoudre dans ces temps vu sa lenteur de réaction. Il faut donc couper toute opération avec le MOPC en trois parties :
     - Write (envoi de la commande)
     - Wait-time
     - Read des résultats.

     Le Write, le Read devant s'effectuer chacun en 100 msec, il faut

éviter qu'une personne étrangère ne vienne s'immiscer entre le Write et le Read. La raison du Wait-time sera discutée plus tard. On aurait pu rallonger le Time-Out mais cette utilisation abusive du CAMAC et de la ligne Multiplex pour rien est à proscrire.

On aurait pu réaliser le Reserve implicitement au début du Write et le Release à la fin du Read. Cependant, pour les opérations de DUMP, de TEST, on ne peut savoir à priori quand effectuer le Release vu que la lecture des résultats s'effectue en plusieurs passes. D'où l'idée de réserver manuellement le G64 pour toute la durée d'une session avec lui.

Le code de cette propriété est incorporé dans la procédure RESERWRITE des fichiers PROP.TXT/BPROP.TXT et correspond au mnémonique #RSV.

2) d'annuler la réservation manuelle du G64.

Pour que cette propriété fonctionne, il faut lui indiquer le nom de l'opérateur ayant effectué auparavant la réservation. Cette précaution a été prise pour éviter qu'une personne étrangère n'éjecte un operateur sans avoir pris connaissance au préalable de son existence.

Cette annulation ne peut être rendue effective que s'il n'y a aucun dialogue en cours avec le MOPC d'un GP contrôlé c'est-à-dire aucun Dump, aucun Test, aucune opération de Write/Read élémentaire.

Le code de cette propriété est incorporé dans la procédure RELEASEWRITE des fichiers PROP.TXT/BPROP.TXT et correspond au mnémonique #RLS.

3) de définir la table de branchement dont il a été question au point 4.1.2.1.

Cette propriété permet d'associer à chacune des actuellement 20 entrées de la table, chaque entrée représentant un canal d'une carte GESSIO-1A, un numéro d'ordinateur du Bâtiment Auxiliaire connecté au G64. (Voir variable TABLE-BRANCH dans les fichiers DECVARE.TXT/BDECVARE.TXT et DECVARL.TXT/BDECVARL.TXT)

Cette propriété existe dans le sens allocation d'un numéro d'ordinateur à un canal et son coding est enfermé dans la procédure VOIEWRITE des fichiers PROP.TXT/BPROP.TXT ainsi que dans le sens lecture du numéro de canal associé à un numéro d'ordinateur dont le coding est enfermé dans la procédure VOIEREAD des fichiers PROP.TXT/BPROP.TXT. Cette propriété a pour mnémonique #TBR.

Remarque : la décision de définir des liaisons canal-ordinateur a été prise pour permettre dynamiquement l'ajout ou le retrait d'un nouvel ordinateur. Sinon, il aurait fallu par programme constituer cette table, ce qui implique une recompilation, réassemblage et remise en EPROM de tout le package G64 si une modification avait dû y être apportée.

4) d'avertir l'opérateur de l'appel d'une propriété non implémentée soit dans le sens Read Call ou soit dans le sens Write Call.

Cette propriété dont le mnémonique est #NOTIMP est implémentée par la procédure NOT-IMPLEMENTED des fichiers PROP.TXT/BPROP.TXT.

4.2.2.2.2 Propriétés relatives au MOPC

1) Ces propriétés permettent, entre autres, l'envoi au MOPC de commandes telles qu'elles auraient pu lui être envoyées en local. Toutes les commandes sont permises conformément à la Section IV du NORD-100 Reference Manual : Operator's interaction with NORD-100, seule la Breakpoint Instruction n'est pas permise suite au problème d'en detecter la fin heureuse.

Tout dialogue avec le MOPC se décompose en trois parties :
- Write operation qui consiste en l'envoi au MOPC d'une commande et dont le coding correspondant de mnémonique $ODR est enfermé dans la procédure ORDREWRITE des fichiers PROP.TXT/BPROP.TXT.
- Wait-Time
- Read des résultats par la propriété de mnémonique $TPR et dont le code est enfermé dans la procédure TRANSREAD des fichiers PROP2.TXT/BPROP2.TXT.

4.2.2.2.2.1 Write operation

Cette opération de Write consiste en :
- l'initialisation du canal de la carte GESSIO-1A connecté à l'ordinateur contrôlé
- l'envoi vers le MOPC de cet ordinateur de la commande et l'attente de la réponse, attente coupée par un Time-Out de 40 msec dont il a été question au point 3.3.6.

Seulement, vu la lenteur de réaction du MOPC, la propriété $ODR ne peut qu'initialiser le traitement et positionner un flag, le flag ORDRE (voir variable ORDRE dans les fichiers DECVARE.TXT/BDECVARE.TXT) avant de rendre la main au MIX6809. Une fois revenu de servir l'interruption IRQ et retombé dans le programme principal qui boucle indefiniement en attente d'interruptions, le systeme repérant le positionnement du flag réagit en conséquence en envoyant au MOPC la commande et en en attendant les résultats. Ce traitement du dialogue en arrière-plan permet de se soustraire de la contrainte des 100 msec du Time-Out de M6809B et d'aller au rythme du MOPC.

Tant que le MOPC n'a pas fini de répondre, il est inutile de lancer des appels Data Module avec la propriété $TPR car ces derniers sont traités au niveau IRQ d'interruption et n'ont pour résultat que de ralentir les opérations au niveau du programme principal, ce qui explique la définition d'un Wait-Time. De plus, pour éviter la lecture des resultats par l'opérateur avant que ceux-ci ne soient disponibles dans le Read Buffer, un flag ETAT.BUSY renseignant sur l'état du G64 a été défini (variable ETAT dans les fichiers DECVARE.TXT/BDECVARE.TXT ou DECVARL.TXT/BDECVARL.TXT). Positionné à TRUE, ce flag empêche la lecture du Read buffer et positionné à FALSE à la fin du traitement d'envoi de la commande/réception des résultats vers/du MOPC, il donne le feu vert à la lecture par l'opérateur des résultats.

Remarque : il est important d'observer que les résultats reçus sont écrits dans un buffer intermédiaire (Voir variable REPONSE des fichiers DECVARE.TXT/BDECVARE.TXT ou DECVARL.TXT/BDECVARL.TXT), cyclique pour des raisons expliquées ultérieurement, et non directement dans le Read Buffer des cartes MPX-RT car l'émission de messages d'erreur, suite à une lecture trop rapide du Read Buffer p.ex., en tant qu'information interne sans rapport direct avec le MOPC passe directement par le Read Buffer et viendrait dans ce cas se superposer à ce qui s'y trouve déjà c'est-à-dire la réponse du MOPC.

4.2.2.2.2.2 Read operation

Cette opération consiste, s'il n'y a pas d'erreur, à écrire dans le Read Buffer des cartes MPX-RT le nombre de bytes utiles ainsi qu'à transférer ces bytes du buffer cyclique REPONSE vers le Read Buffer.

Ce transfert peut s'effectuer :

1) sans intervention du G64 et c'est le but de la procédure TRANSREAD qui travaille en mode transparent c'est-à-dire que les caractères émis par le MOPC sont remontés sans distinction vers l'opérateur;

2) avec intervention du G64 qui filtre les caractères à effectivement renvoyer vers l'opérateur. Cette version de l'opération s'effectue en mode non transparent et n'a pas été implémentée faute de temps. Suit une brève description en pseudo-langage de la manière de l'implémenter :
   - si dernier caractère émis est '?' ou rien n'est émis, envoi d'un message d'erreur;
   - sinon
     - détectez l'écho et le supprimez;
     - si car. prec. est <cr>, laissez passer le caractère '*' significatif d'un début de ligne (s'il est produit);
     - si car.prec. est <SP> et suivant est '/', laissez passer cet ensemble de six caractères représentant l'adresse;
     - si caractere normal, transferez
     - si car. prec. est <SP> et non dernier caractère, envoyez-le;

L'opération consiste, s'il y a erreur, soit qu'un '?' ait été détecté ou que rien n'est été émis par le MOPC de renvoyer vers le MOPC un message d'erreur.

La routine de transfert WRITEBW a dû être écrite en Assembleur et se trouve dans les fichiers MOPCIF.PS/BMOPCIF.PS. Pour mieux comprendre comment passer d'un programme Pascal à un programme ecrit en Assembleur, consulter le chapitre Assembly Language Interface de l'Omegasoft Pascal Language Handbook ainsi que l'exemple fourni en annexe écrit par Mr FABIANI.

4.2.2.2.2.3 Cas particulier : le DUMP de locations-mémoire

Les opérations de DUMP, que ce soit :
- Memory Dump
- Register Dump
- Scratch Register Dump
- IRD ou Internal Register Dump

n'ont pas été implémentées en entier, faute de temps et parce qu'elles nécessitaient l'existence de la propriété de lecture non transparente des résultats. Pour le moment, il est possible d'effectuer un Dump de 0<361 qui représente avec l'écho l'envoi de 2004 bytes par le MOPC.

Pour permettre n'importe quel Dump, il faudrait, après envoi de la commande de Dump par la propriété $ODR par l'opérateur, une fois retourné dans le programme principal dans la zone de traitement du flag ORDRE, découper le Dump en une série de sous-dumps à envoyer séparément 0 < 361, 361 < 722...et ce jusqu'à la valeur demandée.

Si le Dump est terminé et que les résultats du dernier sous-dump ont été lus par l'opérateur par un programme Nodal du type SCHEDL (PROGRAM,ODEV,TIME,INTERVAL), une nouvelle tentative de lecture de résultats

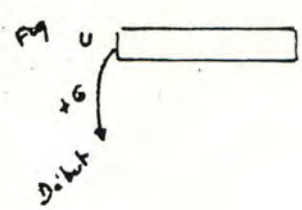
```
            IDATA_REG:= CTRL_REG + $1;
            IDATA_REG^:= CAR;
            WRITE_CHAR:= TRUE;
          END
      ELSE WRITE_CHAR:= FALSE;

END; (* WRITE_CHAR *)

(* ------------------------------------------------------------------------ *)

MODEND.
```

```
          NAM  LIB1
          TTL  multidrop block move library
*
*-------------------------------------------------------------------------
*putblk(var buf:array[1..2048] of byte;var ploc:byte;nch:integer);external;
*ploc=peripheral location
*-------------------------------------------------------------------------
*
          XDEF PUTBLK
PUTBLK    PSHS Y              save y
          LDX  4,U            x=buffer address
          LDY  0,U            y=nch
LP        LDA  0,X+           move block
          STA  [2,U]
          LEAY -1,Y
          BNE  LP
          LEAU 6,U            reset user stack
          PULS Y,PC           reset y and return
*
*-------------------------------------------------------------------------
*getblk(var buf:array[1..2048] of byte;var ploc:byte;nch:integer);external;
*ploc=pripheral location
*-------------------------------------------------------------------------
*
          XDEF GETBLK
GETBLK    PSHS Y              save y
          LDX  4,U            x=buffer address
          LDY  0,U            y=nch
LG        LDA  [2,U]          move block
          STA  0,X+
          LEAY -1,Y
          BNE  LG
          LEAU 6,U            reset user stack.
          PULS Y,PC           reset y and return
*
          END
```




```
{multidrop library (v2) f.fabiani,f.tisserand,a.bland 10/9/84}

module mdlib;
{$idefs}
```

```
   {control/status bits: b7=rp b6=tp b5=it b4=bu b3=sf b2=sr b1=rb b0=tb}
   rb_mask=#$2;
   tb_mask=#$1;
   it_mask=#$20;
   csr_clear=#$c0;
   res_rp=#$80;
   res_tp=#$40;

   e_rwc=-1;{receive word count error}
   e_twc=-2;{transmit word count error}
   e_tou=-3;{transmit timed out error}
   e_par=-4;{write parameter error}

var
   md_csr:byte at md_addr+$1;{control/status register}
   md_rdb:byte at md_addr+$2;{receive data buffer}
   md_tdb:byte at md_addr+$3;{transmit data buffer}
   md_rwl:byte at md_addr+$4;{receive word count low}
   md_twl:byte at md_addr+$5;{transmit word count low}
   md_rwh:byte at md_addr+$6;{receive word count high}
   md_twh:byte at md_addr+$7;{transmit word count high}

procedure putblk(var buf:array[1..len] of byte;var ploc:byte;nch:integer);
external;
procedure getblk(var buf:array[1..len] of byte;var ploc:byte;nch:integer);
external;


function md_init:integer;entry;
begin
 md_csr:=csr_clear;
 md_init:=0;
end {of function md_init};


function md_rb_check:boolean;entry;
begin
 md_rb_check:=(md_csr and rb_mask)<>#0;
end;


function md_write_data(var buf:array[1..len] of byte;bc,msec:integer):
                       integer;entry;
var time:integer;
begin
 {this function writes bc bytes of array buf to the multidrop. The parameter
  msec specifies the number of retries that will be performed due to the
  transmit buffer not being empty. Thus 0 specifies no retries.If transmission
  was succesful then 0 is returned, else an error code is returned. Note that
  interrupts are disabled when the transmit bit is checked and the buffer sent
  because otherwise outputting during an interrupt routine would interfere with
  output during the main program.
 }
 repeat
  if (bc<0) or (msec<0) then md_write_data:=md_error+e_par;exit end;
  md_write_data:=md_error+e_tou;exit end;{assume it will time out}
  for time:=msec downto 0 do
  begin
! PSHS CC
! ORCC #$50;
   if (md_csr and tb_mask)=#0
   then
   begin
    putblk(buf,md_tdb,bc);
    md_csr:=tb_mask;
```

```
    md_write_data:=0;
    exit;
   end;
! PULS CC;
  end {of for loop};
 until true;
end {of function md_write_data};


function md_read_data(var buf:array[1..len] of byte):integer;entry;
var bc:integer;{byte count}
begin
 {this function reads the number of bytes received by the multidrop into the
  buffer passed as a parameter and returns the byte count if succesful or an
  error code otherwise. The value read from the word count registers is doubled
  to give the byte count.
 }
 bc:=ord(md_rwl)<<1+ord(md_rwh)<<9;
 repeat
  if (bc<=0) or (bc>len) then begin md_read_data:=md_error+e_rwc;exit end;
  md_csr:=res_rp;{reset receive buffer pointer}
  getblk(buf,md_rdb,bc);
  {if last byte is 0 then ignore it!}
  if buf[bc]=#$0 then md_read_data:=bc-1 else md_read_data:=bc;
 until true;
end {of function md_read_data};


function md_ack:integer;entry;
begin
 md_csr:=rb_mask;
 md_ack:=0;
end {of function md_ack};


function md_clr_int:integer;entry;
begin
 md_csr:=it_mask;
 md_clr_int:=0;
end;

modend.
```

devrait se terminer soit :

- par l'envoi d'un message d'erreur du type 'Dump terminé' signifiant qu'il n'y a plus rien à transmettre et remise du système dans un état cohérent prêt à renouer un nouveau dialogue; cette façon permet de rester dans la logique de l'opération de test qui sera expliquée plus tard;
- par l'envoi d'un byte count nul renseignant l'opérateur que le Dump est terminé et remise du syteme dans un état cohérent.

Il ne faut pas oublier non plus dans la procédure de lecture des résultats, de repositionner en fin de l'opération le flag ETAT.BUSY à TRUE pour empêcher une inattention de l'opérateur consistant en l'envoi d'une nouvelle commande au MOPC alors que le Dump n'est pas terminé ainsi que d' interdire purement et simplement l'envoi de nouvelles commandes par la propriété $ODR tant que la lecture du dernier sous-dump n'aura pas été effectuée.

Il est renseigné dans les fichiers MOPCIF.TXT/BMOPCIF.TXT ou PROP2.TXT/BPROP2.TXT où insérer les modifications en cas de développement du Dump.

L'idée de descendre toute la commande de DUMP telle quelle vers le G64 et de laisser ce dernier le décomposer en sous-dumps est le résultat de deux faits :

- la décentralisation des opérations au niveau du G64 doit être maintenue pour soulager l'Acces;
- si la découpe était effectuée par l'opérateur :
  1) cela imposerait un travail supplémentaire pour ce dernier
  2) un autre utilisateur de la ligne Multiplex pourrait s'intercaler entre deux sous-dumps et ralentir l'opération globale de Dump. Eviter qu'une personne ne vienne s'immiscer pourrait être résolu par une opération unique constituée de plusieurs CAMPX dont chacun servirait à envoyer un sous-dump et puis pour la lecture des résultats. Cependant, on ne peut requisitionner le Camac et le Bus Multiplex pour toute la durée d'un Dump car une opération prioritaire peut se présenter.

4.2.2.2.2.4 Remarque

L'idée avait émergé d'imposer la définition d'un password pour l'emploi en toute sécurité de certaines instructions comme le Deposit memory, la Stop, Master clear et Load Instruction. Très vite, elle fut abandonnée car les spécifications du MOPC prévoient de tels mécanismes.
Exemple : Après un Memory Examine, le contenu de la cellule-mémoire peut être changé en tapant un nombre octal terminé par <CR>. Si le CPU est en mode RUNNING, "DEP" doit être intercalé entre le nombre et <CR>. Cette précaution impose aux gens d'intervenir en connaissance de cause dans le changement du contenu de positions-mémoire.

Si le travail s'effectue en local, à partir de la console opérateur, rien n'empêche de commettre des bétises telles que les spécifications du MOPC sont actuellement définies. De plus, le but du projet n'est pas une révision des spécifications du MOPC. Cependant, cette sécurité peut être ajoutée optionnellement au niveau du programme d'application formé des appels au Data Module MOPC et employé par l'opérateur en lui imposant la communication de sa section et capacité.

2) Parmi ces propriétés, il en existe une qui permet d'arrêter tout dialogue

en cours avec le MOPC et elle s'appelle $STP dont le code est présent dans la procédure STOPWRITE des fichiers PROP.TXT/BPROP.TXT.

Cette opération permet le redémarrage d'un nouveau dialogue avec le MOPC et, dans le cas du test, constitue l'unique moyen de remettre le G64 dans un état cohérent si l'on veut mettre un terme à l'opération de test. Il en sera discuté plus tard.

Cette opération ne s'accompagne pas d'un release automatique de la réservation manuelle du G64 car on peut vouloir stopper l'opération en cours sans interrompre sa session d'opérateur.

4.2.2.2.3 Propriétés liées au chargement de programmes de test binaires

C'est dans la réalisation de ces propriétés que le package d'application développé au niveau du G64 acquiert toute sa valeur. Elles permettent sans devoir surcharger l'Acces, le chargement de programmes de test dans la mémoire du NORD-100 contrôlé ainsi que la lecture à distance des résultats.

Ces propriétés une fois complètement implémentées s'exécuteront en quatre phases :

1) Phase de communication au G64 des paramètres de test :
   - adresse de départ de chargement du programme servant à effectuer un Examine Memory;
   - nombre de mots dont est constitué le fichier binaire à charger en mémoire
   - adresse de lancement d'exécution du test servant à effectuer le Start a Program Instruction.

   Cette phase initialise en plus le canal de dialogue avec le NORD-100 à contrôler.

   Cette phase est réalisée par l'appel à la propriete $INT dont le code se trouve dans la procédure INITWRITE des fichiers PROP2.TXT/BPROP2.TXT.

   Pour le moment, cette phase ne repère que l'adresse de lancement du programme de test.

2) Phase de chargement proprement dit dans la mémoire du NORD-100 contrôlé du test stocké en librairie.

   Cette phase sera réalisée par appels consécutifs à la propriété $WRT dont le code est contenu dans la procédure TESTWRITE des fichiers PROP2.TXT/BPROP2.TXT et dont un paramètre est une partie du fichier binaire à charger en mémoire. A remarquer que le chargement effectif s'effectuera en arrière-plan suite au positionnement dans la propriété $WRT du flag WRTEST et que les mots envoyés sont en notation décimale à convertir en octal sous chaîne de caractères ASCII. Une fois, le nombre de bytes spécifiés à charger atteint, l'exécution du test est lancée et il n'est plus possible de charger quoi que ce soit.

Ces deux propriétés ont été développées spécifiquement pour éviter, en cas de chargement d'un test, de devoir effectuer un Examine Memory, une myriade de Deposit et un Start a program à distance et donc autant d'appels Data Module. Ce nombre restreint d'appels limite la charge de l'Acces dont le rôle est de fragmenter un fichier de la librairie en morceaux de 2Kbytes à envoyer au G64 et de réceptionner les résultats sur un autre fichier.

Malheureusement, pour le moment, vu les exigences temporelles, le chargement ne peut encore s'effectuer que de cette manière et l'appel à la propriete $WRT n'a pour effet que de lancer l'exécution du test à l'adresse de lancement specifiée par la propriété $INT.

3) Phase de réception des résultats depuis le buffer cyclique.

La lecture s'effectue comme dans le cas d'un simple dialogue avec le MOPC par appels consécutifs à la propriété $TPR qui renvoie :

- si caractères non disponibles, un message d'erreur;
- sinon, le nombre de bytes utiles et ces derniers.

Il vaut mieux que les résultats soient viusualisés au niveau de l'opérateur pour qu'il sache ce que le programme de test attende de lui.

4) Phase d'arrêt.

Le G64 n'a aucun moyen de détecter la fin d'une opération de test. Si ce dernier n'émet plus de resultats :

- soit il attend une intervention de l'opérateur suite à une question posée. Pour y répondre, il suffit par l'appel à la propriete $ODR d'envoyer la réponse soit RUN, soit EXIT, soit HELP;
- soit le test est terminé. Pour y réagir, il suffit d'envoyer la propriete $STP dont il a été question auparavant et qui remet le système dans un état cohérent.

4.2.2.3 Aspect pratique

Ces propriétés déclarées comme ENTRY sont regroupées dans un module PROP.TXT/BPROP.TXT et PROP2.TXT/BPROP2.TXT et pour les employer dans un programme, il suffit de les déclarer EXTERNAL.(Consultez le chapitre Modular Compilation de l'Omegasoft Pascal Language Handbook)

Ces propriétés qui viennent d'être spécifiées, une fois exécutées endéans le Time-Out de 100 msec du M68098, la main est rendue au MIX6809 assurant l'interfacage avec le M6809B pour la remontée des résultats vers l'opérateur, le Nord-100 contenant le code M68098.

4.3 Mise en pratique

4.3.1 Architecture physique et découpe du logiciel en fichiers sous/sans FLEX

Consultez figures nr 3 et 4 pour avoir une idée des relations entre les fichiers constituant le logiciel sous système de développement FLEX et sans FLEX. Les fichiers dont le nom commence par 'B' sont des versions tenant compte du système de développement FLEX.

4.3.2 Utilisation du logiciel de base

Bref rappel des commandes à utiliser pour générer un package d'application Pascal :

1) Pour créer un fichier texte avec le suffixe .TXT, utilisez l'éditeur par page SCREDITOR 111.
   Ex d'édition du fichier EXEMPLE.TXT : ED3 EXEMPLE
   Ex de création du fichier EXEMPLE.TXT : ED3,,1.EXEMPLE.TXT

2) Une fois tous les programmes entrés par l'éditeur, trois opérations différentes sont nécessaires suivant le type de langage ou la découpe utilisée :
   - Si votre programme est un ensemble de procédures déclarées ENTRY et constitue un module écrit en Pascal,
     - compilez ce module : PC,<EXEMPLE,>EXEMPLE 0
     - assemblez le résultat de la compilation : RA,<EXEMPLE.CO,>EXEMPLE.RO 0
   - Si votre programme est un programme Pascal,
     - compilez ce programme : PC,<EXEMPLE,>EXEMPLE 0
   - Si votre programme est écrit en Assembleur,
     - assemblez ce programme : RA,<EXEMPLE.TXT,>EXEMPLE.RO 0

Remarque : il ne faut pas utiliser la commande LC ou Linkage Creator qui regénère automatiquement les fichiers EXEMPLE.PS et EXEMPLE.CF en surécrivant ceux déjà constitués. En effet, pour l'application, ces fichiers ont été crées manuellement par l'éditeur (Voir les fichiers MOPCIF.PS/BMOPCIF.PS ou MOPCIF.CF/BMOPCIF.CF et VECTORS.TXT/BVECTORS.TXT.)
MOPCIF.PS/BMOPCIF.PS contient le code pour générer les Stacks.
BMOPCIF.CF contient le code :
- pour assembler le résultat de la compilation;
- pour assembler le code de génération des stacks;
- pour charger les vecteurs d'interruption et en plus dans MOPCIF.CF le vecteur de RESET;
- pour lier le tout avec la Runtime librairie, n'importe quel autre fichier relogeable obtenu par RA ou d'autres librairies relogeables.

3) La dernière opération est la Chain commande qui génère la version executable en adresse absolue en faisant passer les programmes par différentes versions. Ex d'appel CHAIN MOPCIF/BMOPCIF.

4) La phase d'exécution débute de deux facons :
   - soit par l'apppel 1.MOPCIF.BIN/1.BMOPCIF.BIN si le système tourne avec FLEX;

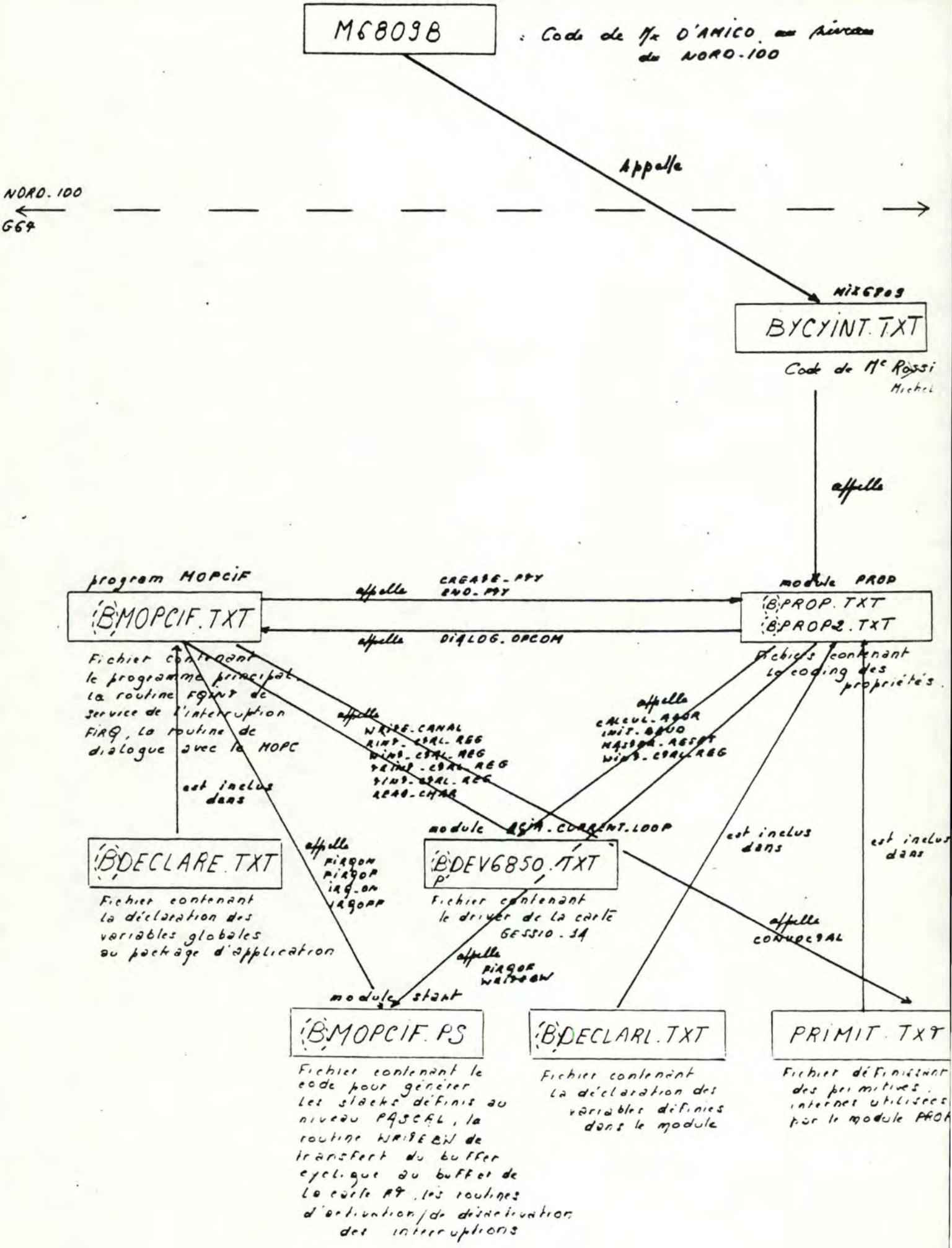
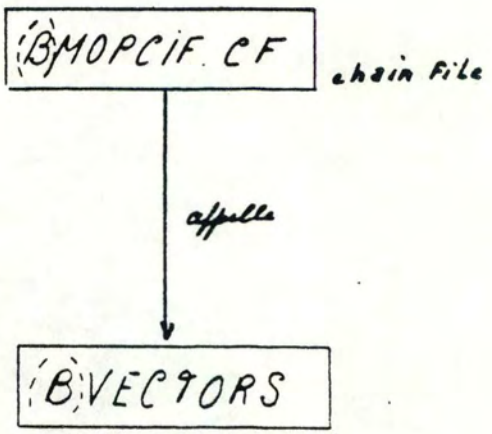

M68098
: Code de Mr D'AMICO au niveau de NORD-100
appelle
NORD-100
G64
NIXCP09
BYCYINT.TXT
Code de Mr Rossi Michel
appelle
program MOPCIF
BMOPCIF.TXT
appelle CREATE-PTY END-PTY
appelle DIALOG.OPCOM
module PROP
BPROP.TXT
BPROP2.TXT
Fichiers contenant le coding des propriétés.
Fichier contenant le programme principal, la routine FQINT de service de l'interruption FIRQ, la routine de dialogue avec le MOPC
appelle WRITE-CANAL RINT-CSAL-REG WINT-CSAL-REG TRINT-CSAL-REG SINT-CSAL-REG READ-CHAR
appelle CALCUL-ADDR INIT-BAUD MASQUE-ACSINT WINT-CSAL-REG
est inclus dans
module ASM.CURRENT.LOOP
BDECLARE.TXT
appelle FIRQON FIRQOF IRQ-ON IRQOFF
BDEV6850.TXT
Fichier contenant le declaration des variables globales au package d'application
Fichier contenant le driver de la carte GESSIO-A
appelle FIRQOF WAITREW
appelle CONVCSAL
module start
BMOPCIF.PS
BDECLARL.TXT
PRIMIT.TXT
Fichier contenant le code pour générer les stacks définis au niveau PASCAL, la routine WRITEBW de transfert du buffer cyclique au buffer de la carte AP, les routines d'activation/de désactivation des interruptions
Fichier contenant la déclaration des variables définies dans le module
Fichier définissant des primitives internes utilisées par le module PROP

Figure n°3



Fichier contenant le code pour charger les adresses des routines de service des interruptions aux bons endroits
Si Fichier VECTORS, il contient en plus le code pour charger le vecteur Reset avec l'adresse de lancement du programme principal.

Figure n° 4

- soit en mettant le système sous secteur si le programme tourne indépendamment de FLEX, du moniteur.

Définition des suffixes définis pour les fichiers :

- Source code - .TXT
- Compiler output - .CO
- .CO after Assembly - .CA
- Pre-setup source - .PS
- .PS after assembler - .PA
- General relocatable object files - .RO
- Loadable object files - .BIN
- Chain - .CF

4.3.3 Comment passer d'un programme sous FLEX à un programme indépendant

Du point de vue hardware, le problème a déjà été abordé en spécifiant les modifications à apporter à la GESSBS-4.

Du point de vue software, après avoir choisi la configuration-mémoire à utiliser, le problème consiste à adapter la version définitive du package G64 à la configuration-mémoire utilisée. Rappel : Comme les données occupent 8 Kbytes et le programme 16 Kbytes, la configuration 3-4 de GESPAC fut choisie.

A quoi faut-il prêter attention ?

1) Modifiez le fichier BMOPCIF.PS pour obtenir le fichier MOPCIF.PS en retirant :
   - LDX -$CC14
   - STX 6,Y
   - LBSR MOPCIF
   - JMP $CD03

   et en le remplacant par LBRA MOPCIF.

2) Modifiez l'adresse de départ de chargement du module source, si nécessaire. Dans le cas présent, elle a été changée de 0000 à 8000 (Consultez les fichiers MOPCIF.CF/BMOPCIF.CF).

3) Modifiez l'adresse de la borne supérieure de la zone des données par rapport à laquelle définir les stacks. Dans le cas présent, elle est passée de C000 a 1FFF.

4) Localisez l'adresse des vecteurs d'interruption dans lesquels charger l'adresse des routines de service des interruptions, du vecteur Reset dans lequel charger l'adresse où débute le programme après avoir mis le système sous secteur. Consultez les fichiers VECTORS et BVECTORS.

Rappel : si vous deviez changer de carte CPU avec une autre configuration-mémoire et espace I/O, il faudrait, en plus, changer :
- l'adresse de base des cartes GESSIO-1A tant du point de vue hardware que software. (Se référer au point 3.3)
- l'adresse de base des cartes MPX-RT tant du point de vue hardware que software. (Se référer au point 3.4)

4.3.4 Brûlage d'un programme en EPROM

Le programme à brûler sera dans le cas qui nous concerne 1.MOPCIF.BIN version adaptée pour fonctionner sans FLEX. Il est constitué de deux parties pouvant être considerées comme formant un tout allant de l'adresse 8000 à FFFF avec un immense intervalle entre B119 et FFF6. Ce code de, disons 32 Kbytes, se retrouvera sur deux EPROM de 16 Kbytes qui, dans mon cas, étaient des INTEL 27128.

1 ère étape : trouvez un systeme FLEX G64 avec un programmeur d'EPROM comme le système portable DEVLONICS qui contient un programme PROMPROG manipulant ce genre de programmeur.

2 ème étape : chargez le programme 1.MOPCIF.BIN en mémoire et appelez PROMPROG. Une possibilité de PROMPROG est de permettre l'initialisation de la mémoire avec n'importe quelle valeur, ce qui permettra de combler les trous dans le programme 1.MOPCIF.BIN. En sachant qu'une nouvelle EPROM ou une EPROM effacée contient, par défaut, la configuration FF ou en binaire 11111111 dans tous les bytes-mémoire et que PROMPROG ne programme pas en EPROM les éventuels FFs rencontrés, la meilleure valeur à choisir pour éviter de perdre du temps est donc FF
Marche a suivre :

- +++PROMPROG
  Select : 27128
  Select : 'FILL'
  From RAM address : 0000
  To RAM address : 3FFF
  Pattern : FF
  Select : 27128
  Select : 'FILL'
  From RAM address : 4000
  To RAM address : 7FFF
  Pattern : FF
  Quittez le programme PROMPROG qui a eu pour résultat de mettre les locations RAM 0000 à 7FFF à la valeur FF

3 ème étape : Appeler le programme RUN avec la syntaxe RUN/<LOAD ADDRESS> <COMMAND> par la commande +++ RUN/0000 1.MOPCIF.BIN dont l'objectif est de ramener toutes les adresses d'un programme calculées par rapport à l'adresse de base 8000 à l'adresse 0000 avant de charger le programme en RAM. Ceci doit être réalisé car la zone 8000 à FFFF n'est pas constituée uniquement de RAM mais aussi de ROM pour le moniteur et l'espace I/O. RUN, dans cette syntaxe, n'exécute pas le programme mais le charge uniquement.

4 ème étape :

+++PROMPROG
Select : 27128
Select : 'PROGRAM'
From RAM address : 0000
To RAM address : 3FFF
EPROM start address : 0000
Fast programming Y/N : N La programmation rapide coupe court à un système de recouvrement d'erreurs et après trois tentatives, suppose que le byte a été correctement écrit en EPROM tandis que la lente recommence tant que le byte écrit ne correspond pas au byte attendu. Mais cette programmation rapide n'est possible qu'avec certains types d'EPROM. (Quid des INTEL 27128 ?)
Verify byte by byte : yes
Display byte by byte : yes

```
   <Put in EPROM> <CR>
```

L'EPROM, dès ce moment est programmée. En programmation lente, c'est l'affaire de 10 minutes alors qu'en rapide, 3 semblent suffire. Ce temps passé, retirez l'EPROM qui devra être utilisée pour les locations 8000 à BFFF du systeme G64. Le même travail est à réitérer pour l'autre EPROM de 16 Kbytes à utiliser pour les locations C000 à FFFF du systeme G64.

```
+++PROMPROG
   Select : 27128
   Select : 'PROGRAM'
   From RAM address : 4000
   To RAM address : 7FFF
   EPROM start address : 0000
   Fast programming Y/N : N
   Verify byte by byte : yes
   Display byte by byte : yes
   <Put in EPROM> <CR>
```

Quand termine, retirez l'EPROM.

Quand les EPROMs sont prêtes, il ne reste plus qu'à les positionner sur la carte GESSBS-4 de GESPAC, à mettre sur secteur le système G64 et l'application est opérationnelle.

5. Manuel d'utilisation

Tout appel au Data Module se présente sous la forme MOPC (AR,"R"/"W",N-VALUE,#PROPERTY)
avec AR, un tableau d'entiers. Le code de Mr D'AMICO découpe chaque mot en deux bytes.
R/W, le type de l'opération; R pour la lecture et W pour l'écriture.
N-VALUE, entier donnant le nr d'équipement que l'opérateur veut contrôler.
#PROPERTY, chaîne de trois bytes ASCII donnant le nom de la propriété que l'opérateur veut voir exécuter.

5.1 Le tableau AR

Dans le cadre de tout appel au Data Module MOPC, AR doit être configuré de la manière suivante :

- les cinq premiers bytes sont réservés à la définition du mot-clé caractérisant l'opérateur et ce pour remplir l'objectif de réservation manuelle du G64 pendant toute la durée d'une session de travail. Ils doivent se présenter comme suit '(---)' ou - remplace un des bytes du mot-clé;
- les quinze bytes suivants sont largement suffisants pour l'envoi d'une commande au MOPC, l'attribution d'un canal à un ordinateur.

Vu le problème de passage au Data Module du paramètre sous forme de mot, l'envoi d'un nombre impair de bytes se termine par ']' qui est repéré au niveau du G64 comme caractère special pour arrondir le nombre de bytes envoyés.

5.2 La propriété #PROP

Les propriétés sont au nombre de huit.

5.2.1 La propriété #RSV

#RSV assure la réservation manuelle du G64 pour une session d'opérateur.

AR doit simplement donner la définition du mot-clé selon la configuration '(---)'. Il est à rappeler qu'AR n'accepte que des entiers. Il faut donc regrouper les bytes du mot-clé par deux avant de les confier à AR.

Le seul type d'opération permis est un Write Call.

Le N-VALUE ne revêt aucun sens dans cet appel.

Exemple d'appel : MOPC ('(VGD)',"W",1,#RSV).

Messages d'erreur possibles :

- 'G64 busy' : Si le G64 est en phase de dialogue avec le MOPC d'un numéro d'ordinateur, soit en envoi, soit en réception de caractères.
- 'Cle d'accès mal/non définie..."(---)" --> G64 réservé ?' : si l'opérateur

a mal défini sa clé d'accès ou ne l'a pas mentionnée.
- 'G64 sous réservation manuelle' : si le G64 est octroyé à l'usage d'un autre opérateur, avec un autre mot-clé défini.
- 'Propriété non implémentée à ce jour' : si l'opérateur a tenté d'appeler la propriété dans le mode d'opération Read.

5.2.2 La propriété #RLS

#RLS permet la désallocation du G64 de l'opérateur mettant ainsi un terme à sa session de dialogue.

AR doit simplement donner la définition du mot-clé selon la configuration '(---)'. Il est à rappeler qu'AR n'accepte que des entiers. Il faut donc regrouper les bytes du mot-clé par deux avant de les confier a AR.

La définition du N-VALUE ne revêt aucun sens.

Le seul type d'opération permis est un Write Call.

Exemple d'appel : MOPC ('(VGD)',"W",1,#RLS).

Messages d'erreur possibles :
- 'G64 busy' : Si le G64 est en phase de dialogue avec le MOPC d'un numéro d'ordinateur, soit en envoi, soit en réception de caractères.
- 'Cle d'accès mal/non définie..."(---)" --> G64 réservé ?' : si l'opérateur a mal defini sa clé d'accès ou ne l'a pas mentionnée.
- 'G64 sous réservation manuelle' : si le G64 est octroyé à l'usage d'un autre opérateur, avec un autre mot-clé défini.
- 'Propriété non implémentée à ce jour' : si l'opérateur a tenté d'appeler la propriété dans le mode d'opération Read.
- 'Opération en cours à interrompre...' : si le G64 remplit une demande précédemment faite au MOPC d'effectuer soit un Dump, soit un Test ou soit une Manual Instruction, le seul moyen de s'en sortir est l'emploi de la propriété #STP (Voir plus loin).

5.2.3 La propriété #TBR

Cette propriété est utilisable en Write Call et en Read Call.

#TBR, en Write Call, permet de constituer dynamiquement la table de branchement consistant à associer à chaque entree-canal, un numéro d'ordinateur.

Dans ce cas, AR doit fournir, en plus de la définition du mot-clé, le numéro de canal à connecter à l'ordinateur N-VALUE. Ce numéro se présente, comme le mot-clé, sous forme d'une chaîne de caractères ASCII. Exemple d'association au canal 12 du numéro d'ordinateur 1 : MOPC ('(VGD)12',"W",1,#TBR).

#TBR, en Read Call, permet de se souvenir du numéro de canal associé à un ordinateur.

AR doit, dans ce cas, en entrée contenir simplement la définition du mot-clé selon la configuration '(---)'. Il est à rappeler qu'AR n'accepte que des entiers. Il faut donc regrouper les bytes du mot-clé par deux avant de les confier a AR. En sortie, il contiendra le nombre en entier de

bytes utiles à lire par l'opérateur ainsi que le numéro de canal en ASCII structuré en mot de deux bytes par le Data Module et à rescinder par byte.

Messages d'erreur possibles en Write Call :
- 'G64 busy' : Si le G64 est en phase de dialogue avec le MOPC d'un numéro d'ordinateur, soit en envoi, soit en réception de caractères.
- 'Cle d'accès mal/non définie..."(---)" --> G64 réservé ?' : si l'opérateur a mal defini sa clé d'accès ou ne l'a pas mentionnée.
- 'G64 sous réservation manuelle' : si le G64 est octroyé à l'usage d'un autre opérateur, avec un autre mot-clé défini.
- 'Propriété non implémentée à ce jour' : si l'opérateur a tenté d'appeler la propriété dans le mode d'opération Read.
- 'Seuls 20 canaux sont attribuables' : si l'opérateur tente de connecter un numéro d'ordinateur à un canal non disponible sur le G64. Pour le moment, 20 sont utilisables pour une connexion.
- 'Ancien canal dédicacé à l'ordinateur' : si l'opérateur attribue à un autre ordinateur un canal déjà dédicacé à un autre équipement. A remarquer que l'attribution est cependant directe et irreversible.

Messages d'erreur possibles en Read Call :
- 'Ordinateur non connecté à un canal' : si l'opérateur a specifié comme N-VALUE un numéro d'ordinateur non connecté à un canal disponible du G64.
- 'G64 busy' : si le G64 est en phase de dialogue avec le MOPC d'un numéro d'ordinateur, soit en envoi ou soit en réception de caractères.

5.2.4 La propriété $STP

$STP permet l'interruption du dialogue en cours avec le MOPC d'un ordinateur, soit une simple opération de Write/Read, soit un Dump Memory, soit une Manual Instruction ou un Test.

Il est à rappeler que les opérations de Test et de Manual Instruction ne peuvent être conclues correctement que par l'emploi de cette propriété uniquement disponible en Write Call.

AR doit simplement donner la définition du mot-clé selon la configuration '(---)', moyen sous lequel l'opérateur s'est fait connaître du G64. Il est à rappeler qu'AR n'accepte que des entiers. Il faut donc regrouper les bytes du mot-clé par deux avant de les confier à AR.

La définition du N-VALUE ne revêt aucun sens.

Exemple d'appel : MOPC ('(VGD)',"W",1,$STP).

Messages d'erreur possibles :
- 'G64 busy' : Si le G64 est en phase de dialogue avec le MOPC d'un numéro d'ordinateur, soit en envoi, soit en réception de caractères.
- 'Cle d'accès mal/non définie..."(---)" --> G64 réservé ?' : si l'opérateur a mal défini sa clé d'accès ou ne l'a pas mentionnée.
- 'G64 sous réservation manuelle' : si le G64 est octroyé à l'usage d'un autre opérateur, avec un autre mot-clé défini.
- 'Propriété non implémentée à ce jour' : si l'opérateur a tenté d'appeler la propriété dans le mode d'opération Read.
- 'L'OPCOM n'est pas enclenché' : si le MOPC n'a rien émis dans le cas de l'arrêt de la Manual Instruction.
- 'Label or Checksum (loading) error or Incompatible mode' : Si suite à une demande adressée au MOPC par l'opérateur, celui-la a détecté une erreur de syntaxe dans l'arrêt de la Manual Instruction.

5.2.5 La propriété #ODR

#ODR permet l'envoi au MOPC de l'ordinateur contrôlé d'une commande ASCII ou d'une réponse à une question formulée par le programme de test chargé dans le NORD-100 contrôlé ou le chargement de programme de test binaire mot par mot.

AR doit fournir, en plus de la définition du mot-clé sous la configuration '(---)', la commande en ASCII à envoyer au MOPC . Il est à rappeler qu'AR n'accepte que des entiers. Il faut donc , après définition du mot-clé et de la commande ASCII, restructurer l'ensemble des bytes en mots de 2 bytes.

Pour les commandes terminées par <CR>, ce <CR> ne doit pas être rajouté par l'opérateur car il s'agit du seul cas où la commande est finie d'être habillée par le G64.

Exemple d'appel : MOPC ('(VGD)1/',"W",N-VALUE,#ODR).

Messages d'erreur possibles :
- 'G64 busy' : Si le G64 est en phase de dialogue avec le MOPC d'un numéro d'ordinateur, soit en envoi, soit en réception de caractères.
- 'Cle d'accès mal/non définie..."(---)" --> G64 réservé ?' : si l'opérateur a mal défini sa clé d'accès ou ne l'a pas mentionnée.
- 'G64 sous réservation manuelle' : si le G64 est octroyé à l'usage d'un autre opérateur, avec un autre mot-clé défini.
- 'Propriété non implémentée à ce jour' : si l'opérateur a tenté d'appeler la propriété dans le mode d'opération Read.
- 'La carte de l"OPCOM n"est pas connectée' : si l'opérateur définit dans l'appel au Data Module MOPC, un numéro d'ordinateur non connecté à un canal d'une carte GESSIO-1A.
- 'La BREAKPOINT INSTRUCTION n"est pas disponible' : si l'opérateur tente l'envoi de ce type de commande au MOPC dont le traitement n'est pas prévu par software.

5.2.6 La propriété #TPR

#TPR permet la lecture en mode transparent des résultats émis par le MOPC suite à une commande ASCII ou par un programme de test.

En entrée, dans l'appel au Data Module, AR doit fournir le mot-clé spécifiant l'opérateur en session avec le G64. En sortie, AR contiendra la valeur entière du nombre de bytes utiles à lire par l'opérateur ainsi que ces bytes regroupés par deux sous forme de mots et qui devront être rescindés en deux (par la fonction NODAL ARTRAN n'acceptant pas le caractère NUL).

Messages d'erreur possibles :
- 'G64 busy' : Si le G64 est en phase de dialogue avec le MOPC d'un numéro d'ordinateur, soit en envoi, soit en réception de caractères.
- 'Propriété non implémentée à ce jour' : si l'opérateur a tenté d'appeler la propriété dans le mode d'opération Write.
- 'L'OPCOM n"est pas enclenché' : si le MOPC n'a rien émis ou si l'opérateur effectue deux appels #TPR successifs alors qu'une première lecture a été réalisée avec succès sans l'envoi d'une nouvelle commande par la suite.
- 'Label or Checksum (loading) error or Incompatible mode' : Si suite à une demande adressée au MOPC par l'opérateur, celui-la a détecté une erreur de syntaxe.
- 'TEST...Buffer empty' : si l'opérateur, en cours de test, tente de lire des

résultats envoyés par le programme chargé dans la mémoire de l'ordinateur contrôlé, cela peut signifier :
  - lecture trop rapide des résultats alors qu'aucun n'est disponible pour le moment;
  - programme de test en attente d'une réponse envoyée par l'opérateur;
  - fin du programme de test.
- 'Overflow du buffer cyclique' : si l'opérateur, en cours de test, n'est pas venu suffisamment rapidement lire les résultats envoyés par le programme binaire chargé dans le Nord-100.
- 'Ordinateur actuel de dialogue' : si l'opérateur, par mégarde, tente de lire les résultats sur un ordinateur autre que celui sur lequel il a envoyé une commande.
- 'Aucun canal initialisé par un dialogue' : si l'operateur après avoir soit mis son systeme sous tension s'il travaille sans FLEX, sans moniteur ou soit appelé le programme 1.BMOPCIF.BIN contenant le code du package s'il travaille avec FLEX, tente de lire de suite des résultats alors qu'aucun dialogue n'est amorcé avec aucun ordinateur.

5.2.7 La propriété #INT

#INT permet l'initialisation de l'opération de test.

AR doit fournir, en plus de la définition du mot-clé sous la configuration '(---)'et à partir du 6 ème byte, trois entiers :
  - le premier indiquant le nombre total de mots à déposer dans la mémoire du NORD-100 testé;
  - le deuxième l'adresse à partir de laquelle effectuer le chargement en mémoire;
  - le troisième l'adresse à partir de laquelle lancer le programme de test.

Cette propriété n'est utilisable qu'en Write Call.

Messages d'erreur possibles :
- 'G64 busy' : Si le G64 est en phase de dialogue avec le MOPC d'un numéro d'ordinateur, soit en envoi, soit en réception de caractères.
- 'Cle d'accès mal/non définie..."(---)" --> G64 réservé ?' : si l'opérateur a mal défini sa clé d'accès ou ne l'a pas mentionnée.
- 'G64 sous réservation manuelle' : si le G64 est octroyé à l'usage d'un autre opérateur, avec un autre mot-clé défini.
- 'Propriété non implémentée à ce jour' : si l'opérateur a tenté d'appeler la propriété dans le mode d'opération Read.
- 'La carte de l"OPCOM n"est pas connectée' : si l'opérateur définit dans l'appel au Data Module MOPC, un numéro d'ordinateur non connecté à un canal d'une carte GESSIO-1A.
- 'Test non permis par l"opération en cours' : si l'opérateur lance un test alors que le G64 remplit une fonction de Dump, de Manual Instruction ou un Test.

5.2.8 La propriété #WRT

#WRT permet d'effectuer le chargement proprement dit dans la mémoire du N-100 contrôlé d'un programme de test découpé en tranches d'entiers, ces tranches étant passées dans le tableau AR après la définition bien entendu du mot-clé sur les 6 premiers bytes, le 6 ème étant occupé par un caractère non significatif.

Actuellement et si rien d'autre n'est implementé d'ici mon départ, #WRT seulement disponible en Write Call ne permet que le lancement du programme. après une série d'appels à la propriete #ODR pour le chargement du programme de test en mémoire par une série de Deposits de chaque mot.

Le N-VALUE ne revêt aucun sens dans cette propriété car le canal de dialogue est initialisé par le propriété #INT.

Messages d'erreur possibles :
- 'G64 busy' : Si le G64 est en phase de dialogue avec le MOPC d'un numéro d'ordinateur, soit en envoi, soit en réception de caractères.
- 'Clé d'accès mal/non définie..."(---)" --> G64 réservé ?' : si l'opérateur a mal défini sa clé d'accès ou ne l'a pas mentionnée.
- 'G64 sous réservation manuelle' : si le G64 est octroyé à l'usage d'un autre opérateur, avec un autre mot-clé défini.
- 'Propriété non implémentée à ce jour' : si l'opérateur a tenté d'appeler la propriété dans le mode d'opération Read.
- 'Opération de test non initialisée' : si l'opérateur, par mégarde, tente le chargement en mémoire de programme de test binaire sans avoir par appel à la propriété #INT communiqué au prealable les paramètres nécessaires à l'opération.

5.3 Exemple de déroulement d'une session avec le G64

1) Lancement du système :
   - soit en mettant le G64 sous secteur si on travaille sans FLEX ou moniteur;
   - soit en appelant 1.BMOPCIF.BIN si on travaille avec FLEX.
2) Opérations indispensables en début de session :
   - MOPC ('(VGD)',"W",---,#RSV) pour la réservation manuelle de début de session;
   - MOPC ('(VGD)1',"W",N-VALUE,#TBR) pour le garnissage de la table de branchement avec au moins une valeur;

3) Opérations en cours de session :
   Opération de W/R normale optionnelle :
   - MOPC ('(VGD)....',"W",N-VALUE,#ODR) pour l'envoi d'une commande au MOPC;
   - MOPC ('(VGD)....',"R",N-VALUE,#TPR) pour la/les lectures de résultats;

   Opération de test optionnelle :
   - MOPC ('(VGD)....',"W",N-VALUE,#INT) pour initialiser le test;
   - MOPC ('(VGD)....',"W",---,#WRT) pour l'envoi des tranches du fichier binaire à charger en mémoire;

   - MOPC ('(VGD)',"R",N-VALUE,#TPR) pour la lecture des résultats;
   - MOPC ('(VGD)....',"W",---,#ODR) pour l'envoi de réponses au programme de test du type RUN ou EXIT;

   - MOPC ('(VGD)',"W",---,#STP) pour arrêt du test;

4) Opération indispensable de fin de session :
   - MOPC ('(VGD)',"W",---,#RLS) pour avorter la session de dialogue G64.

TABLE OF CONTENTS

Section

Section