MASTER EN SCIENCES INFORMATIQUES

Software de télétraitement pour un miniordinateur pédagogique

Debot, Danielle

Award date:
1974

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX A NAMUR

Institut d'Informatique

Année académique 1973-1974

SOFTWARE DE TELETRAITEMENT POUR UN MINIORDINATEUR PEDAGOGIQUE

Danielle DEBOT

Jury du Mémoire :
M. J. BRUNIN

Mémoire présenté en vue de l'obtention du grade de Licenciée et Maître en Informatique.

Nous tenons à remercier particulièrement M. BRUNIN, directeur du mémoire, qui nous a aidé à donner une orientation pratique et réaliste à ce travail.

Nous exprimons également notre reconnaissance envers

Messieurs Baily, Windal et Van der Straet, ainsi que l'équipe télétraitement d'IBM.

Monsieur De Hepcée professeur à l'institut d'Informatique.

Monsieur Xhaard de la R.T.T.

qui ont mis leur expérience à notre disposition.

TABLE DES MATIERES

III.2.3.2. Zone de communication du message POOL.

III.2.4. Instructions de communication.

III.3. Moniteur de télétraitement.

III.3.1. Informations.

III.3.1.1. Zones d'information de chaque ligne.
III.3.1.2. Emploi de ces zones dans le déroulement normal d'une opération lecture/écriture.

III.3.2. Schéma de circulation des signaux avec cas d'erreurs.
III.3.3. Modules du moniteur.

III.3.3.1. Programme de gestion de lignes ou PGL.

III.3.3.1.1. Prise de contrôle du PGL.
III.3.3.1.2. Rôle du PGL.
III.3.3.1.3. Fin du PGL.

III.3.3.2. Interruptions.
III.3.3.3. Input/Output.
III.3.3.4. Erreurs.

III.3.3.4.1. Classement et traitement.
III.3.3.4.2. Topographie des erreurs et signalements.
III.3.3.4.3. Tableau général.
III.3.3.4.4. Tableau d'erreurs pour l'utilisateur.

IV. DEUXIEME PROCEDURE.

IV.1. Différences avec la première procédure.
IV.2. Modifications apportées à la première procédure.

IV.2.1. Présentation du problème.

IV.2.1.1. Déroulement général.
IV.2.1.2. Schéma de circulation des messages.

IV.2.2. Jonction avec le programme d'application.

IV.2.2.1. Opérations de lecture/écriture.
IV.2.2.2. Distributeur de tâches.

I. INTRODUCTION.

I.1. DEFINITION D'UN PROBLEME DE TELETRAITEMENT.

Dans une agence de banlieue un employé cherche un numéro de téléphone. Il tape rapidement la question sur un "terminal", en l'occurrence une machine à écrire de télétransmission :

" Nom : Dupont
Prénom : Jean
Numéro de téléphone ? Adresse ? "

Ce message est envoyé vers un ordinateur au centre de la ville. Quelques instants plus tard, le terminal tape "1354 numéros possibles
Profession ?
Ville ? "

L'employé retape alors :

" Boucher
NAMUR "

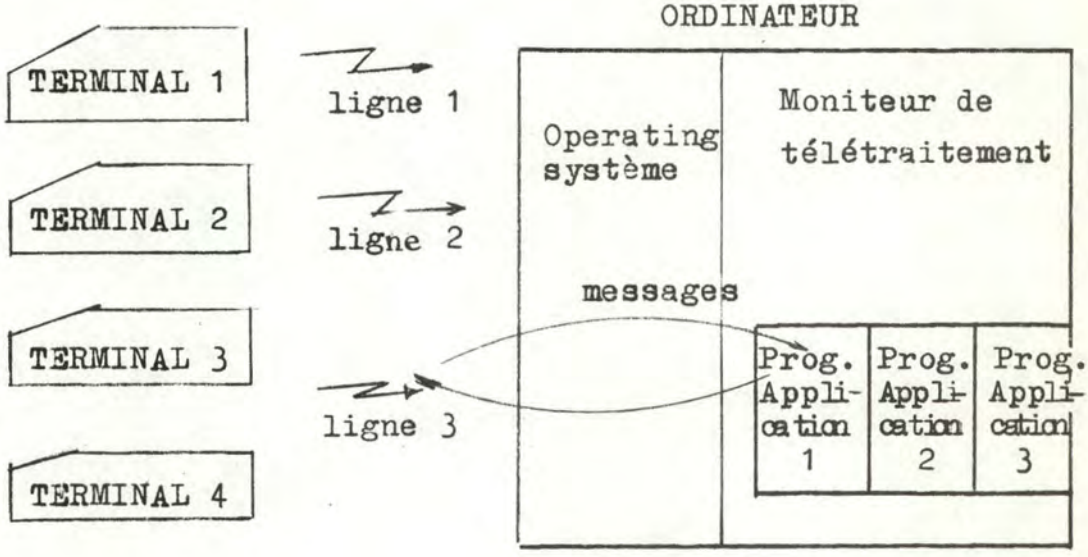
Et peu après il recevra la liste des numéros de téléphone des bouchers namurois nommés Jean Dupont, suivis de leur adresse.

Pourtant l'ordinateur et le fichier regroupant ces renseignements se trouvent à plus de 10 km de là, et sont consultés par plusieurs agences.

Voilà un exemple de télétraitement ou traitement à distance.

Chaque "terminal" peut envoyer un message qui sera traité par un programme d'application. Celui-ci, à son tour, pourra en renvoyer un au terminal.

La gestion des différents messages, en provenance de plusieurs lignes de télécommunication, se fera par l'intermédiaire d'un moniteur de télétraitement.



En effet, un programme d'application peut traiter des messages en provenance de plusieurs lignes. Il doit les recevoir successivement alors que les terminaux les envoient sur les lignes sans se concerter.

Le moniteur règle donc la circulation des messages venant de diverses lignes et les envoie au programme d'application approprié, au moment voulu. De même il renvoie les messages en provenance des programmes d'application, sur la ligne appropriée, pour le terminal voulu.
De plus, il s'occupera du traitement des erreurs de transmission.

REMARQUE : Il peut y avoir plusieurs terminaux par ligne,
plusieurs lignes pour un programme d'application.
Une ligne peut envoyer des messages pour différents programmes.

I.2. LIMITES DU MEMOIRE.

Un moniteur standard, idéal, s'adapterait à toutes les situations.
Il serait :

- compliqué : car il devrait envisager tous les cas possibles, avec toutes leurs répercussions possibles;
- lourd : car il devrait à tout moment exécuter une série de tests pour reconnaître le cas se présentant, et le traiter seulement après.

Il faut donc le limiter et préciser :

- les propriétés et caractéristiques des terminaux choisis;
- le fonctionnement du mini-ordinateur influençant le déroulement des opérations;
- les différents dialogues possibles entre les terminaux et l'ordinateur;
- les caractères adoptés pour la télétransmission.

Dans ce mémoire nous n'examinerons pas des problèmes tels que la sécurité des fichiers servant au télétraitement, l'optimisation de l'occupation mémoire par gestion de pool, ou l'optimisation de programmes.

Le but sera de construire un moniteur logique permettant le fonctionnement cohérent des terminaux de télétransmission.

I.3. DEFINITION DU MEMOIRE.

Le problème sera découpé de la façon suivante :

1. un seul programme d'application recevant des messages de plusieurs lignes suivant la procédure Inquiry définie au chapitre II.3.

2. un seul programme d'application recevant des messages de plusieurs lignes suivant la procédure Contention définie au chapitre II.3 ;

3. Un seul programme d'application recevant des messages de plusieurs lignes suivant la procédure de Polling définie au chapitre II.3 ;

4. Ensuite, toutes les procédures simultanées, s'adressant à plusieurs programmes d'application.

Le problème de télétraitement est supposé de dérouler dans l'ordinateur en multiprogrammation avec d'autres programmes. Il sera seul dans une"partition" qui aura la priorité absolue sur les autres partitions et pourra reprendre la main lors des interruptions provoquées par elle.
Dans"la partition télétraitement ", il pourra y avoir plusieurs programmes.

A. Lorsqu'il y a un seul programme d'application et plusieurs lignes, deux programmes sont en concurrence :

   1. le moniteur : qui doit mettre à jour le travail des lignes en lançant des opérations comme lecture/écriture de message et transmission au programme d'application.

   2. le programme d'application : qui doit traiter les messages.

   Il faudra donc permettre à l'un ou l'autre de se dérouler, par un distributeur de tâches.
   Il faudra des zones de communications accessibles par les deux programmes, pour transmettre des informations de l'un à l'autre.
   Il faudra des instructions permettant de donner et d'obtenir ces informations.
   De plus, le moniteur doit toujours savoir dans quels états se trouvent les lignes qu'il doit gérer, ce qu'il doit y faire, et si des erreurs

s'y sont produites. Ces renseignements se trouveront dans des zones d'information qu'il mettra à jour continuellement. Une grande partie de ce travail sera faite lors de traitements d'interruptions.
Le programme d'application pourra être très varié. Il y a un exemple de schéma en annexe.

B. Lorsqu'il y a plusieurs programmes d'application, plusieurs programmes sont en concurrence :

1. le moniteur qui fait le même travail qu'en A/ et qui transmet les messages à tous les programmes d'application.

2. tous les programmes d'applications qui doivent traiter les messages qui leur sont adressés.

Il faudra donc que le distributeur de tâches permette au moniteur ou à un programme d'application de se dérouler.
Il faudra aussi des zones de communication entre le moniteur et chaque programme d'application.

REMARQUE : Dorénavant, le programme d'application sera appelé P.APPL. car ce nom reviendra très souvent.

II. DEFINITIONS PREALABLES.

II. 1. POSSIBILITES ET CARACTERISTIQUES DES TERMINAUX.

L'état du terminal est indiqué dans le "Status byte" et lorsqu'il y a erreur, dans le "sense byte".

STATUS BYTE :
- 0 non opérationnel
- 1 prêt
- 2 occupé
- 3 interruption (END) en attente
- 4 Mode de fonctionnement
- 5 erreur (détail dans sense byte).

ORDRES AGISSANT SUR LE TAMPON DU TERMINAL.

SIO : si il n'y a pas d'interruption en attente :
- le STATUS BYTE est transmis
- passage de l'état PRET à l'état OCCUPE.

Si l'analyse du STATUS BYTE est positif, la réception des ordres est ettendue.

HIO : permet le passage de l'état OCCUPE à l'état PRET lorsque, pour différentes raisons, on désire rendre le périphérique disponible. Le STATUS BYTE est transmis également.

TIO : ne modifie rien à l'état du périphérique et transmet le STATUS BYTE.

AIO : acquitte la fin de la routine END.

TDV : ne modifie rien à l'état du périphérique et transmet le SENSE BYTE en remettant ce dernier à zéro.

Tableau précisant le fonctionnement des ordres T 1.

ETAT	B.INC.	PUISS.		INSTR.				END	STATUS BYTE	ETAT
		0	1	SIO	HIO	AIO	INV			
1		1	2						0	INOP
2		1	2	3					1	PRET
3		1	3	7	2		7	E	2	OCCUPE
7	1	1	7						5	ERREUR
E		1	E			2			3	END émis

Remarque : Un END est produit lors de la réception des caractères ETB et ETX, et lorsqu'une opération est terminée, si le tampon du terminal est dans l'état occupé.

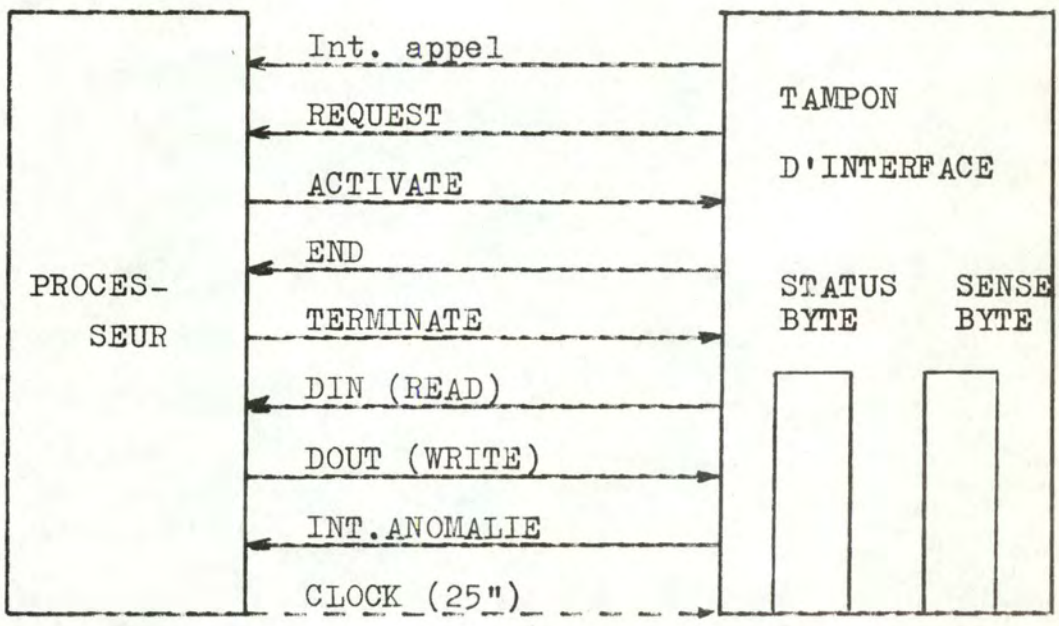
SENSE BYTE :

0 code opér. invalide
1 inopérable ou début
2 cadencement
3 erreur parité
4 lecture bloc trop court (caractères entre terminate et END)
5 inopérable en fonctionnement
6 time out : plus de 25" entre STX et ETX.

Il y a une interruption anomalie (INT AN) pour tout bit du SENSE BYTE qui passe en 1. Le SENSE BYTE est remis à zéro par TDV contenu dans la routine d'interruption.

II.2. FONCTIONS DU MINIORDINATEUR.

II.2.1. L'interface ordinateur-périphérique est caractérisée par les relations électriques suivantes :



Signification :

INT APP : si en état READ du tampon un STX est détecté.

REQUEST : si en état READ le tampon est plein
WRITE le tampon est vide

END : si en état READ le tampon détecte ETB/ETX, et en fin d'opération (fin de CCW) si le tampon est dans l'état occupé.

ACTIVATE : si le REQUEST a été honoré.

TERMINATE : si en READ ou WRITE la longueur du bloc prévue au CCW est atteinte (BC = 0);

INT AN : si une erreur est détectée au niveau du tampon - cfr. ERREURS;

Les fils DIN transmettront les données avec parité vers le processeur.

Les fils DOUT transmettront les données avec parité et les commandes vers le tampon du terminal (cfr.tableau T.1).

II.2.2. TRAITEMENT DES INTERRUPTIONS.

Chaque interruption est mémorisée dans un "banc d'interruptions". A certains moments, par exemple en fin de micro-instruction et de SVC, un décodeur prioritaire permet aux interruptions en attente d'être traitées suivant leur priorité si elles ne sont pas masquées. Les masques sont différents suivant que l'on travaille en mode alarme, superviseur ou problème, ce qui permet de rendre certains programmes non interruptibles par certaines interruptions.

II.3. TYPES DE PROCEDURE ET DE LIGNE.

Les terminaux utilisés seront divisés en trois familles, la première fonctionnera en mode Inquiry et de façon asynchrone, la seconde en mode Contention et de façon synchrone, et la troisième en mode Polling et de façon synchrone.

Tampons : Pour le premier mode, il fonctionnera de façon asynchrone. Pour les deux autres modes, un système de réception de signaux de synchronisation SYN et de génération de ceux-ci, lui permettront de fonctionner de façon synchrone.

II.3.1. PREMIERE PROCEDURE.

Le mode Inquiry est élémentaire en ce sens qu'il correspond :

a. de la part du terminal :

- à l'envoi d'un message STX .... ETX qui constitue une question posée au programme d'application.
- à l'utilisation d'aucun caractère de contrôle. Tout au plus, le tampon de l'ordinateur détectant une erreur au niveau du bloc provoquera-t-il de la part du moniteur la transmission d'un message au terminal l'invitant à renouveler son message.
- plusieurs questions peuvent être posées successivement, donc il n'y aura pas nécessairement alternance question-réponse.

b. de la part de l'ordinateur :

- à la réponse par le programme d'application à la question posée,
- à la possibilité d'envoyer au terminal un message spontané.

PREMIERE PROCEDURE - "QUESTION-REPONSE".

- Le terminal peut prendre l'initiative d'envoyer un message à l'ordinateur et ce dernier y répondra.

- Il y a un programme d'application et plusieurs lignes de transmission point-à-point, c'est-à-dire avec un seul terminal par ligne.

- procédure asynchrone, sans caractères de contrôle, STX et ETX provoquent des interruptions mais pas de Request.

Le système doit le plus vite possible envoyer un ordre de lecture au terminal, le tampon est mis alors en lecture et attend l'arrivée d'un message aussi longtemps qu'il faut.

II.3.2. DEUXIEME PROCEDURE - "SELECTING-CONTENTION".

L'initiative est laissée à la station émettrice.
En cas de conflit, la priorité est accordée à l'une des deux.
La procédure est décomposée en phases (fig. 1).

1. PREPARATION.

L'ordinateur qui désire envoyer fait appel à son module d'émission au moyen d'une instruction CPUT qui provoquera le passage en phase 2 déterminant l'envoi d'un ENQ (WRITE) puis le basculement en phase ③ (READ) de façon à réceptionner si la séquence est normale un ACK Ø qui permettra au maître d'envoyer le premier bloc en phase ⑥.
Toutefois, il est possible que des caractères différents de ACK soient reçus. Il s'agira de cas d'anomalie soit notamment :

a. un ENQ si l'ordinateur distant désire transmettre; c'est l'état ⑤ qui se chargera d'arbitrer ce cas de conflit en fonction de la priorité. Si c'est l'esclave, le maître passera en mode réception (un R de ce module).

b. un NAK si l'ordinateur distant n'accepte pas le message. Le maître terminera par l'émission d'un caractère EOT en phase 4.

c. un caractère non valide auquel cas une réémission de ENQ sera effectuée en phase 2 via un test du nombre limite de ENQ émis via a_0.

d. le mutisme de l'esclave. Le maître ayant eu la précaution de lancer une temporisation T_1 de 3 secondes, elle se manifestera ensuite en déclenchant une réémission de ENQ comme pour un caractère invalide.

2. TRANSMISSION MESSAGE.

L'envoi du premier bloc ayant été effectué en phase 3, il y a lieu de savoir dans quelles conditions cette transmission a été faite; le message étant terminé par ETB (61) ou ETX (62) ou tronqué (63).
En séquence normale il y a donc lieu de distinguer :

- si un ETB (61) a été transmis on doit normalement recevoir, comme il s'agit du premier bloc (il en sera de même des messages impairs), un accusé de réception ACK_1, auquel cas, l'envoi du message pair qui suit sera réalisé en phase (8),
- si c'est ETX (62) qui a été transmis, la réception de ACK_1 provoquera la rupture de la liaison par EOT en phase 4.
  Toutefois il est possible que des caractères différents de ACK_1 soient reçus lorsqu'il y a anomalie de transmission. Ce sera notamment le cas pour une erreur au niveau du message (a/b) ou des caractères (C).

  a. une NAK si l'ordinateur distant décèle une erreur de parité (BCC) du texte. Le maître répètera le bloc incriminé en phase (7) mais via A_1 pour le contrôle du nombre de répétition de bloc. La séquence reprendra la voie normale via 61/62/63.

b. l'absence de caractère reprenant le cas précédent après une temporisation T_1 ce qui se produira lorsque le message à l'arrivée (ordinateur esclave) ne comporte pas de STX et/ou $\overline{ETX}$ ou $\overline{ETB}$ ou si le message au départ a été interrompu (63).

c. un caractère non valide ce qui provoquera le renvoi par le maître d'un ENQ en phase b_1 via a_1 puis passera en b_1' .

La réponse par l'ordinateur esclave aux caractères ENQ émis (b et c) donc en mode réception, suivra les mêmes règles en ce qui concerne le maître.
Ce dernier recevra en phase b_1' soit :

a. un caractère ACK_1 si l'ordinateur distant a été satisfait du message impair précédemment reçu auquel cas le message pair suivant sera transmis (phase (8)).

b. un caractère NAK si l'ordinateur distant n'a pas été satisfait (sauf en ce qui concerne pour STX) du message impair reçu et il le confirme à nouveau. Dès lors, le maître lui enverra à nouveau le message impair précédent en phase (7) via A_1.

c. un $ACK_\emptyset$ qui ne sera renvoyé par l'ordinateur distant qu'en cas de manque de STX. Il provoquera comme pour un NAK la répétition du bloc impair.

d. un caractère invalide ou rien d'où répétition des ENQ via a_1 donc en cas de nouvelle anomalie de transmission de caractère.

Remarquons pour terminer la différence qui existe entre les cas d'un manque de ETB/ETX détecté au départ (état 63) et à l'arrivée. Dans le premier cas c'est-à-dire celui où le message est tronqué à l'émission un état (63) provoquera après temporisation T_1 l'envoi d'un EOT. Dans le second cas (61/62) cette même temporisation provoque la répétition de ENQ.

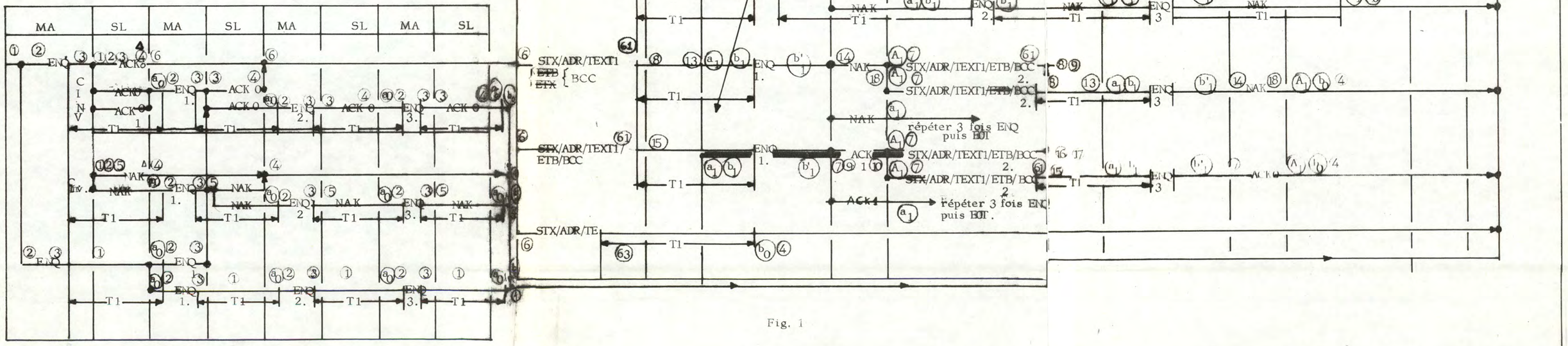
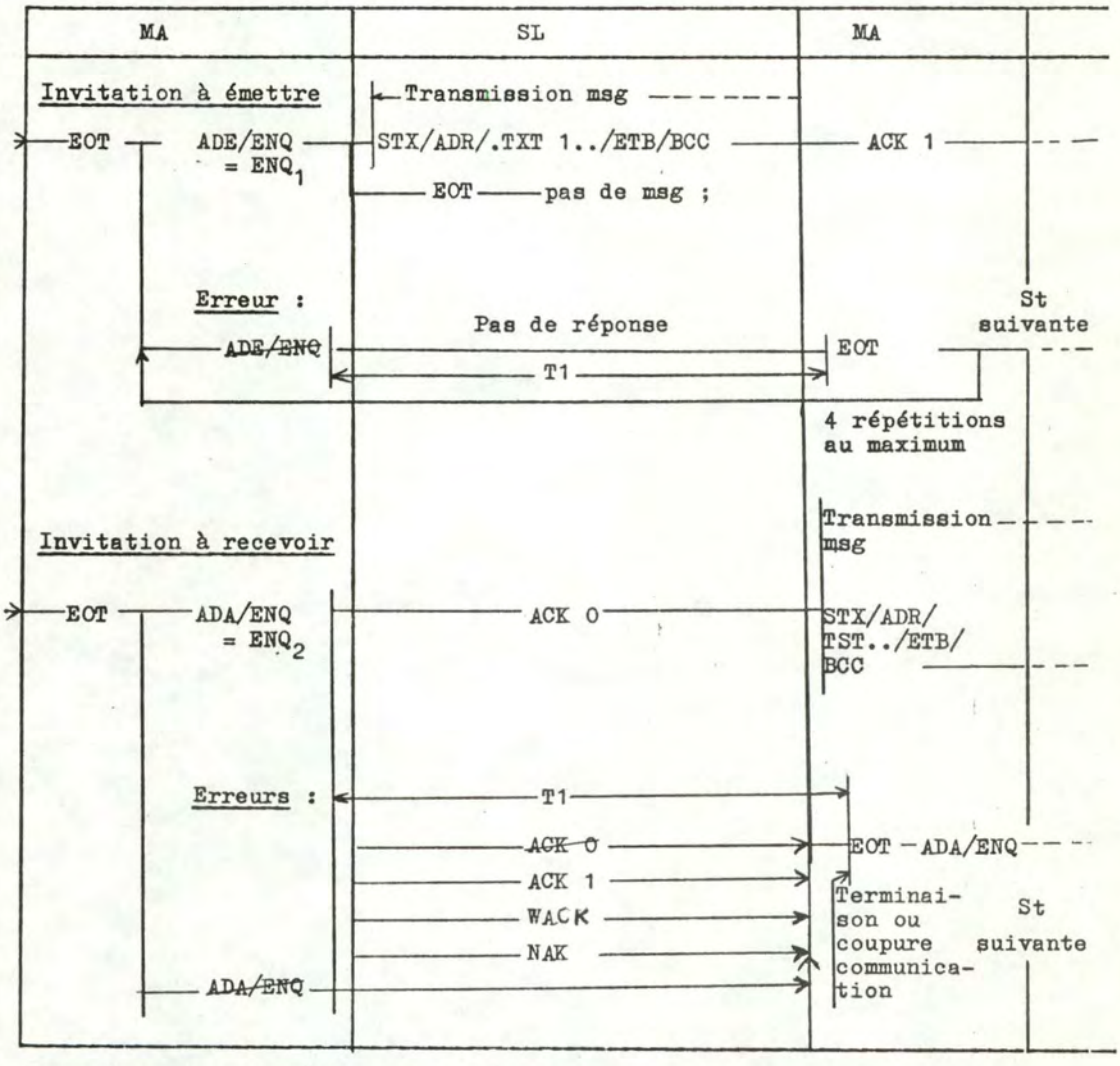

MA
SL
MA
SL
MA
SL
MA
SL
TEXT. 3 ----
STX/ADR/TEXT1/ETB BCC
ACK1
STX/TEXT2/ ETB BCC
ACK0
STX/TEXTn/ ETX/BCC
EOT
ENQ
NAK
T1
répéter 3 fois ENQ puis EOT
répéter 3 fois ENQ puis EOT.

Fig. 1

II.3.3. TROISIEME PROCEDURE -"SELECTING-POLLING".

L'initiative est toujours donnée à l'ordinateur maître (fig. 2).

En procédure SELECTING-POLLING, l'une des stations connectées est la sation maîtresse (il s'agit fréquemment de l'unité centrale). Les autres stations sont dites esclaves et ne peuvent échanger des données qu'avec la station maîtresse. Seule, la station maîtresse peut lancer une transmission de données au moyen d'invitation à émettre (ENQ_2) ou à recevoir (ENQ_1) émises à des intervalles de temps déterminés (polling ou sélective calling) et destinées aux autres stations. La sélection d'une station déterminée, de même que la sélection de l'unité chargée d'émettre ou de recevoir les données dans cette station, est effectuée par le caractère d'adresse contenu dans la demande. Une station esclave ne peut émettre un message que lorsqu'elle reconnaît son adresse dans l'invitation à émettre qu'elle reçoit. Lorsque les stations esclaves reçoivent le caractère EOT, elles sont automatiquement remises au repos, pour être en mesure de recevoir des invitations à émettre ou à recevoir.



Les suites de caractères dont la transmission est erronnée sont barrées.

Légende : ADE = Adresse de la station esclave et du terminal d'entrée.
ADA = Adresse de la station esclave et du terminal de sortie.
ADR = Adresse du terminal d'entrée, si nécessaire, seulement.

II.4 ROLE DES CARACTERES DE CONTROLE.

SIGNE	NOM	ETAT DE CONTROLE (Mode contrôle)	TRANSFERT DE MESSAGE (Mode texte)	COMMENTAIRES	
				ETAT DE CONTROLE (Mode contrôle)	TRANSFERT DE MESSAGE (Mode texte)
ENQ	Demande	Pt à Pt : Pouvez-vous accepter une transmission	Entre blocs : Répondez ou répétez la dernière réponse	En Pt à Pt et écriture multipoint : la réponse sera ACKO ou NAK	Pour demander une réponse en cas de délai du récepteur ou pour faire répéter la dernière réponse en cas de rupture de séquence paire/impaire.
		Multipoint : Répondez à votre adresse	En fin de bloc : Annulez ce bloc et répondez NAK	En lecture multipoint la réponse sera EOT si pas prêt ou ETX suivi du texte	
ACKO	Réponse affirmative paire	Je peux accepter la transmission	Bloc pair bien reçu	La séquence paire/impaire permet de détecteur la perte d'un bloc.	
ACK1	Réponse affirmative impaire	pas utilisé	Bloc impair bien reçu		
NAK	Réponse négative	Je ne peux accepter la transmission	Bloc reçu non valide J'accepte la retransmission		
STX	Début de texte	Les informations suivantes seront en mode texte	Restaure les circuits de vérification et démarre le calcul du nouveau caractère de vérification.		
ETB	fin d'un bloc de texte	Pas utilisé	Caractère de vérification suit. Vérifiez et répondez, un autre bloc suit.		
ETX	fin de texte	Pas utilisé	Caractère de vérification suit. Vérifiez et répondez. Plus de texte pour ce message, liaison non coupée		
TTD	Délai temporaire de texte	Transmission va commencer. Répondez NAK et attendez.	Transmission va continuer. Répondez NAK et attendez.	L'émetteur a besoin de temps, liaison non coupée.	
WACK	Réponse affirmative et demande de délai	Redemandez l'autorisation de transmettre et retardez la transmission jusqu'à réponse affirmative.	Redemandez l'autorisation de transmettre et retardez la transmission jusqu'à réponse affirmative	le récepteur veut retarder la transmission, l'émetteur envoie alors ENQ	
EOT	fin transmission	Termine une liaison, retour au mode contrôle ou réponse négative à une demande de lecture en multipoint.	Termine une liaison. Retour au mode contrôle. Pas valide à l'intérieur du texte.		

III. PREMIERE PROCEDURE.

III. 1. PRESENTATION DU PROBLEME.

III. 1.1. DEROULEMENT GENERAL.

- Pour une ligne :

Le p.applic. doit d'abord signaler à l'ordinateur qu'il est prêt à recevoir des messages d'une ligne, il le fait par la macro CPUT ACTIVATE sur cette ligne, elle devient alors "ligne ouverte". Par exemple, il pourra ouvrir une ligne à 9 h. du matin, une autre à 11 h. et une troisème seulement à 14 h. Aucun message ne passera avant d'avoir ouvert la ligne.

Un message (question) est envoyé par le terminal de cette ligne et vient remplir le "pool de lecture" de cette ligne. Ce message doit être traité par le p. applic. et pour cela celui-ci recevra un signal.

Lorsqu'il a reçu ce signal, le p.applic. désigne une zone de lecture ZL et une zone de renseignements (Z NOTIF), et demande le message par la macro CGET. Le message vient alors remplir ZL tandis que les renseignements (erreurs) le concernant sont positionnés dans ZNOTIF.

Ensuite, le P.APPL. traite la question et envoie le message réponse au terminal par la macro CPUT, en indiquant la zone contenant le message ou zone écriture ZE, et une zone de renseignements ZNOTIF.
Ce message est alors envoyé dans le pool d'écriture et puis sur la ligne tandis que les renseignements concernant le lancement de l'opération se mettront dans ZNOTIF, accessible au p.applic.

Ensuite, le p.applic. attend qu'on lui signale l'arrivée d'un autre message, il se mettra dans cet état d'attente par la macro SUSP.

- Plusieurs lignes :

Mais le P.APPL. doit traiter des messages en provenance de toutes les lignes, sans se préoccuper de quelle ligne ils viennent ni du fait qu'ils viennent sur une ligne de télétransmission.

Il faudra donc qu'un autre programme (moniteur de télétraitement) s'occupe de tous les problèmes relatifs à la télétransmission, et de la gestion des messages en provenance de différentes lignes.
C'est le moniteur qui signalera au P.APPL. qu'il doit traiter un message.

- Le programme application pourra donc être :

1 en exécution c'est-à-dire traitement de message.

2. en attente du signal d'arrivée d'un message, au moyen de l'instruction SUSP (expliquée plus loin).

DEUX POSSIBILITES sont encore données au programme application.

1. S'il ne veut pas envoyer de réponse à la question mais veut recevoir une autre question, il peut employer la macro CPUT ACTION qui enverra un ordre de lecture au terminal.

2. Si le programme d'application veut attendre qu'une opération de lecture ou d'écriture se termine sur une ligne, et puis continuer à traiter des messages pour cette même ligne (sans se mettre en attente d'un signal et donc sans devenir susceptible de s'occuper de messages en provenance d'une autre ligne), il peut employer la macro WAIT.
Par cette instruction il arrête de se dérouler, perd la main, et recevra de nouveau la main pour continuer de se dérouler lorsque l'opération d'I/O sera terminée.

La réalisation de ce système pose plusieurs problèmes.

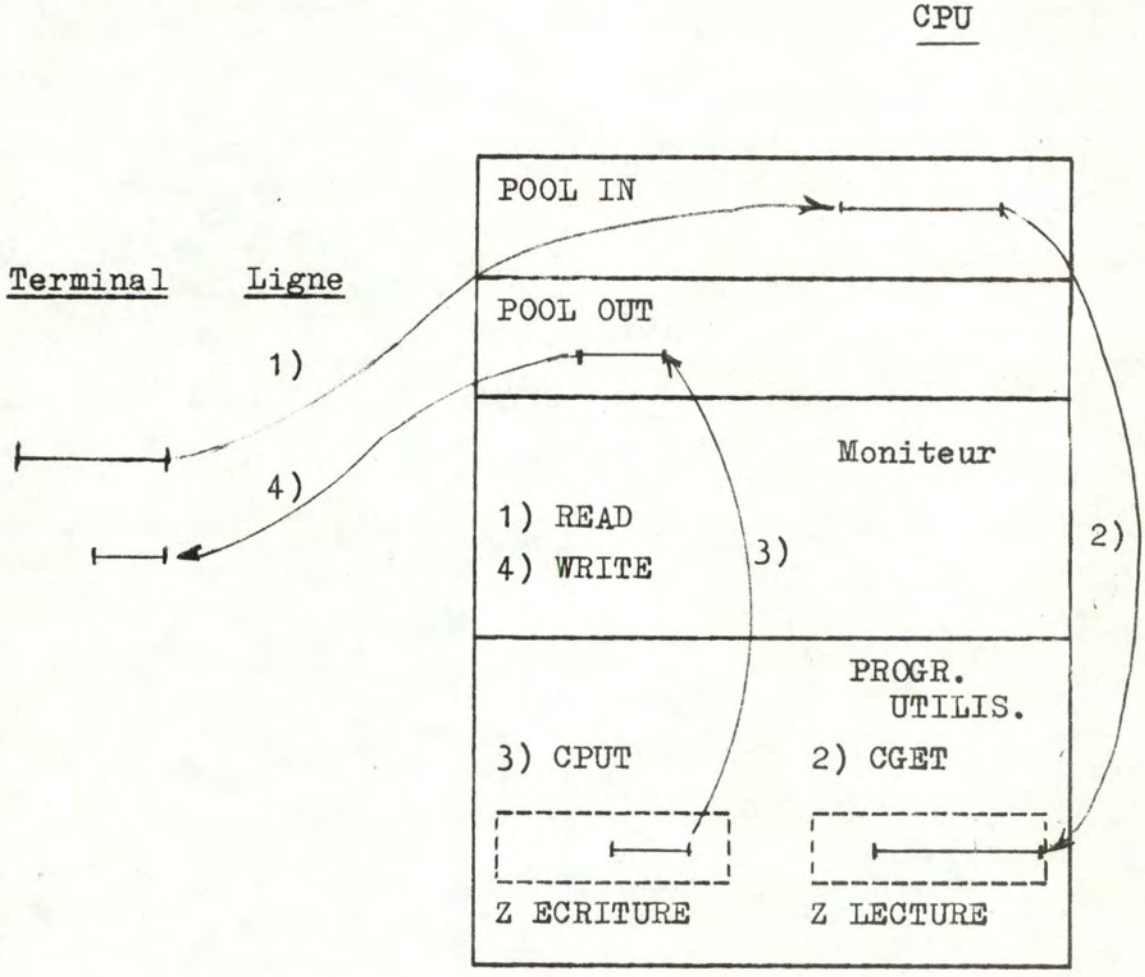
I. Il faut lancer au bon moment, sur chaque ligne, les opérations d'I/O.

II. Il faut régler le déroulement alternatif de deux programmes :

- le moniteur qui signale les arrivées de messages au programme d'application,
- le programme d'application qui traite ces messages,

et il faut que le moniteur soit prévenu de toutes les arrivées de messages, qu'il les mémorise, et les signale, toutes, au bon moment, au P.Application.

III.1.2. SCHEMA DE CIRCULATION DES MESSAGES.



⟵⟶ Message

x) yyy Commande n° x

yyy Zone

x) ↗ Transfert n° x réalisé par commande n° x.

Circuit des messages

Terminal —READ→ Pool in —CGET→ Z lect.de UTILIS.

Terminal ←WRITE— Pool out ←CPUT— Z Ecrit.de UTILIS.

III. 2. JONCTION AVEC LE PROGRAMME D'APPLICATION.

III. 2.1. OPERATIONS DE LECTURE/ECRITURE.

L'ordinateur doit toujours s'attendre à recevoir un message du terminal. Il faut donc lancer une lecture dès l'ouverture de la ligne. Lorsqu'un message est arrivé, la lecture est terminée, et le P.APPL. relancera une écriture (CPUT) ou une lecture (CPUT ACTION).
Après l'écriture, une lecture doit immédiatement être relancée.

Problème : Lorsque le P.APPL. écrit :

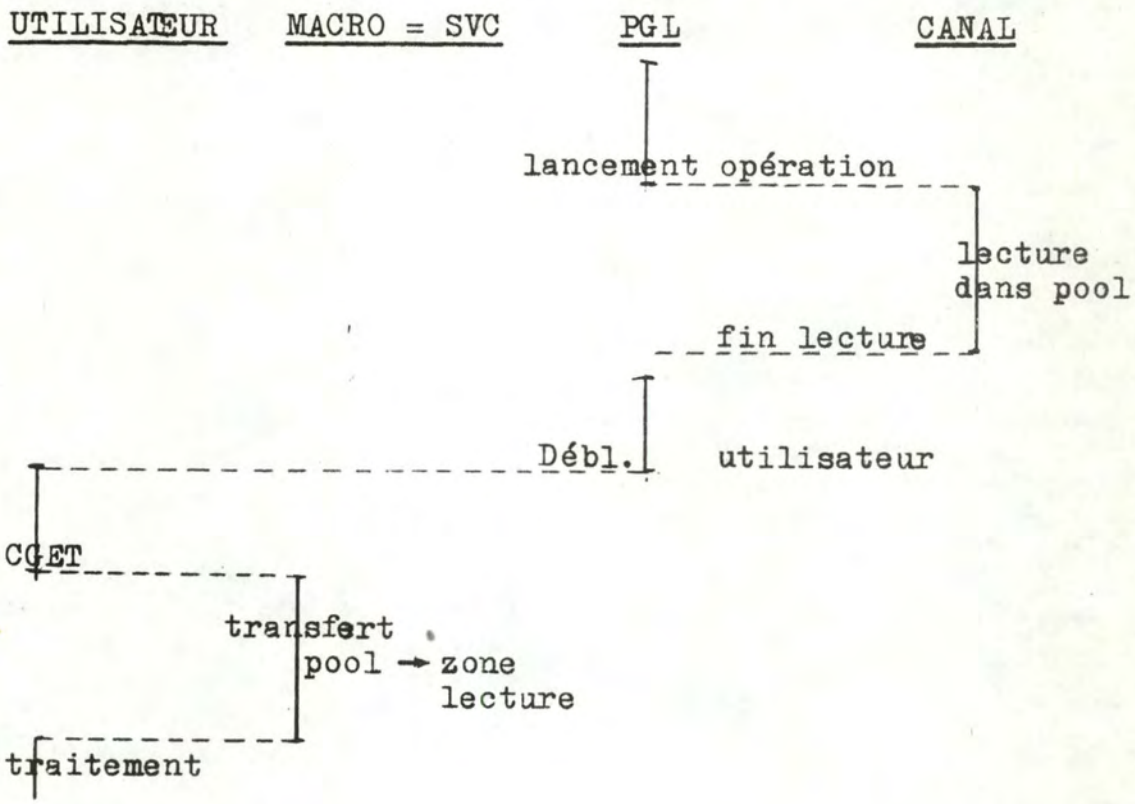
- CPUT ACTIVATE : il faut "ouvrir la ligne" (cfr.III.2.4) et lancer une lecture sur la ligne puis repasser la main au P.APPL.
- CPUT ACTION : il faut lancer une lecture sur la ligne et repasser la main au P.APPL.
- CPUT Message : il faut lancer une écriture sur la ligne et repasser la main au P.APPL.

Lorsqu'il y a une interruption de fin d'écriture, il faut relancer une lecture sur la ligne.

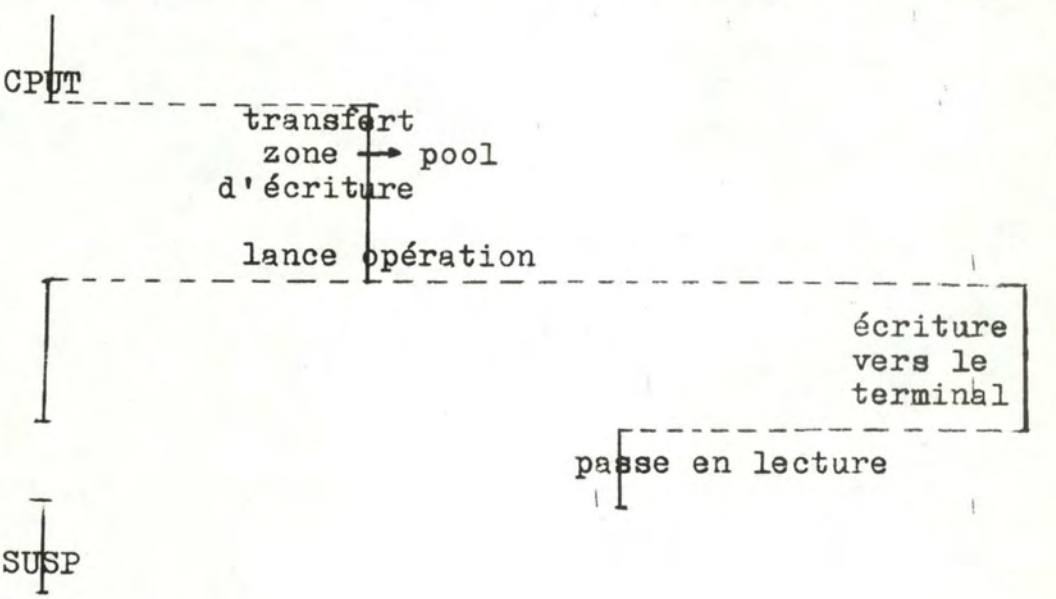
Solution :

- Les macro-instructions CPUT appelleront par un "SVC" une routine qui lancera, en mode superviseur, l'opération demandée sur la ligne voulue.
- L'interruption de fin d'écriture relancera elle-même une lecture car cela doit être faite le plus vite possible.

OPERATION DE LECTURE : principales étapes.



OPERATION D'ECRITURE :



III.2.2. DISTRIBUTEUR DE TACHES.

Le problème de déroulement de programme se situe à 2 niveaux.

1. Niveau global : Il y a plusieurs "partitions" dans l'ordinateur. Dans l'une d'elles se trouve tout le problème de télétraitement, et dans les autres d'autres problèmes. Ce sera le distributeur de tâches principal (propre au superviseur) qui s'occupera de donner la main à l'une ou l'autre partition.

2. Niveau télétraitement : dans ce problème, deux tâches doivent se dérouler alternativement :

   - le PGL du moniteur, qui passera les messages au bon moment au p.applic.
   - le p.appl. de l'utilisateur qui traitera les messages.

   Ce sera le distributeur de tâches qui réglera le déroulement de l'un ou l'autre.

<table>
<tr><td colspan="3">Distributeur de tâches principal
D.T.P.</td></tr>
<tr><td colspan="2">Partition 1
Distr. tâches
D.T.</td><td rowspan="2">Partition
2</td><td rowspan="2">Partition
3</td></tr>
<tr><td>PGL</td><td>P.APPL</td></tr>
</table>

III.2.2.1. NIVEAU GLOBAL.

On considère que la partition télétraitement est la plus prioritaire. Elle ne rend la main au distributeur de tâches principal que si elle n'a plus rien à faire.

On travaille dans un contexte de multiprogrammation, et la partition télétraitement reprendra la main dès qu'elle aura quelque chose à faire, c'est-à-dire qu'une opération d'entrée/sortie s'est terminée :

- soit qu'un programme attendait la fin de cette opération pour continuer à se dérouler,
- soit qu'un nouveau message est arrivé d'un terminal.

Donc, chaque fin d'opération entrée/sortie sera signalée par une interruption (END ou INTAN), cette interruption sera traitée et ensuite le D.T.P. passera la main à la partition télétraitement :

- soit au programme interrompu si c'était la partition télétraitement qui avait été interrompue,
- soit au D.T. si c'est une autre partition qui avait été interrompue.

Par exemple,le D.T. peut positionner un "bit télétraitement" à l'intention du D.T.P. quand il reçoit la main, et l'enlever quand il rend la main au D.T.P.

III.2.2.2. NIVEAU TELETRAITEMENT.

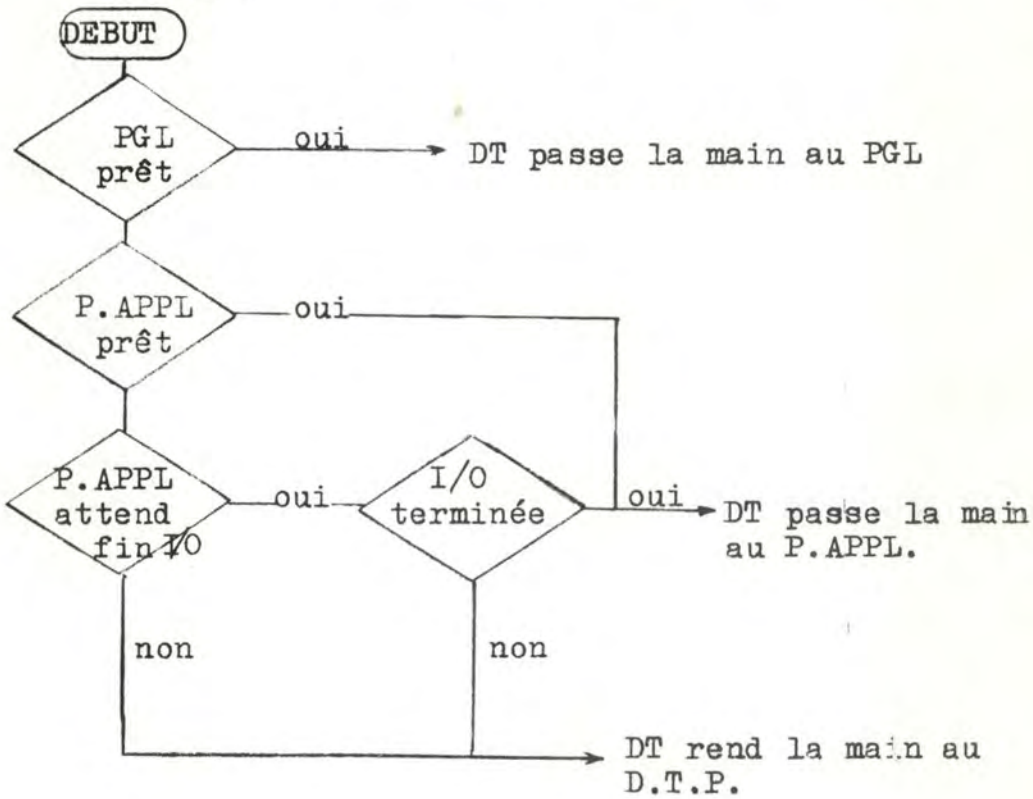
Premier problème : il faut régler le déroulement de deux programmes :

- le PGL qui signale les messages au P.APPL.
- le P.APPL. qui traite ces messages.

C'est le D.T. qui passe la main à l'un ou l'autre et c'est au D.T. qu'ils rendent la m main lorsqu'ils ont terminé.

Travail du D.T.

Si PGL est prêt à se dérouler, DT lui passe la main sinon, si P.APPL. est prêt à se dérouler, DT lui passe la main, sinon, si P.APPL. attend une fin I/O et qu'elle est terminée, DT lui passe la main sinon, DT rend la main au DTP.



Remarque :

- le D.T.P. passe toujours la main au D.T. au même endroit, début.
- le D.T. rend toujours la main au D.T.P. au même endroit.

On aura des bits indiquant l'état dans lequel se trouve chacun des 2 programmes, ces bits seront positionnés à l'intention du DT qui donnera la main au programme voulu. En plus de ces bits, il y aura une adresse indiquant où le DT doit passer la main lorsqu'il la passe au PGL et au P.APPL.

Il faut maintenant rendre le PGL et le P.APPL. prêts à se dérouler au bon moment en positionnant les bits indiquant l'état des deux programmes et les adresses où ils doivent recevoir la main.

Problèmes :

1. P.APPL. doit attendre une fin I/0 pour continuer.
2. P.APPL. doit attendre que PGL lui signale un message.
3. PGL doit être prévenu de toutes les arrivées de message et les mémoriser.
4. PGL doit signaler ces messages au bon moment au P.APPL.

Solutions :

1. Pour attendre une fin I/0, le P.APPL. emploie la macro WAIT, ce qui le met dans un état spécial "WAIT NON READY" reconnaissable par le D.T.
   Lorsqu'une fin d'I/0 se produit, elle provoque une interruption qui positionne un bit dans une zone spéciale : le CCB de la ligne = zone mémoire propre à la ligne. Ce bit est EX.FLAG (cfr. III.3.1.1).
   Il suffira donc que le D.T. regarde si EX.FL. est positionné pour savoir que l'état WAIT peut être enlevé pour le P.APPL. et que celui-ci peut se dérouler de nouveau.

   Donc WAIT :

   - positionne P.APPL. "non ready WAIT"

- positionne l'adresse où on devra lui rendre la main, c'est-à-dire l'adresse de l'instruction suivant le WAIT (P counter).

2. Lorsque le P.APPL. a terminé le traitement d'une question, il emploie la macro SUSP pour attendre que le PGL lui signale un message.
   Cette macro indique :
   - que le P.APPL. ne doit plus se dérouler;
   - que le PGL doit se dérouler pour chercher s'il y a encore 1 message;
   - que le P.APPL. reprendra à l'instruction suivant le SUSP.

   Donc SUSP :
   - positionne P.APPL. non ready;
   - positionne PGL ready;
   - stocke l'adresse de retour = adresse de l'instruction suivant SUSP (P.counter).

3. Chaque fois qu'un message est arrivé dans un pool de lecture, dans une zone propre à chaque ligne un état spécial est positionné accessible au PGL (phase FIN READ) (cfr. IV2.3.).

4. Le PGL examine les lignes successivement et suivant un ordre circulaire. Lorsqu'il en trouve une dans l'état spécial, il prévient le P.APPL. qu'il y a un message à traiter, en indiquant de quelle ligne il provient pour que les macros CGET et CPUT sachent où prendre la question et où renvoyer la réponse.

   Rappel :

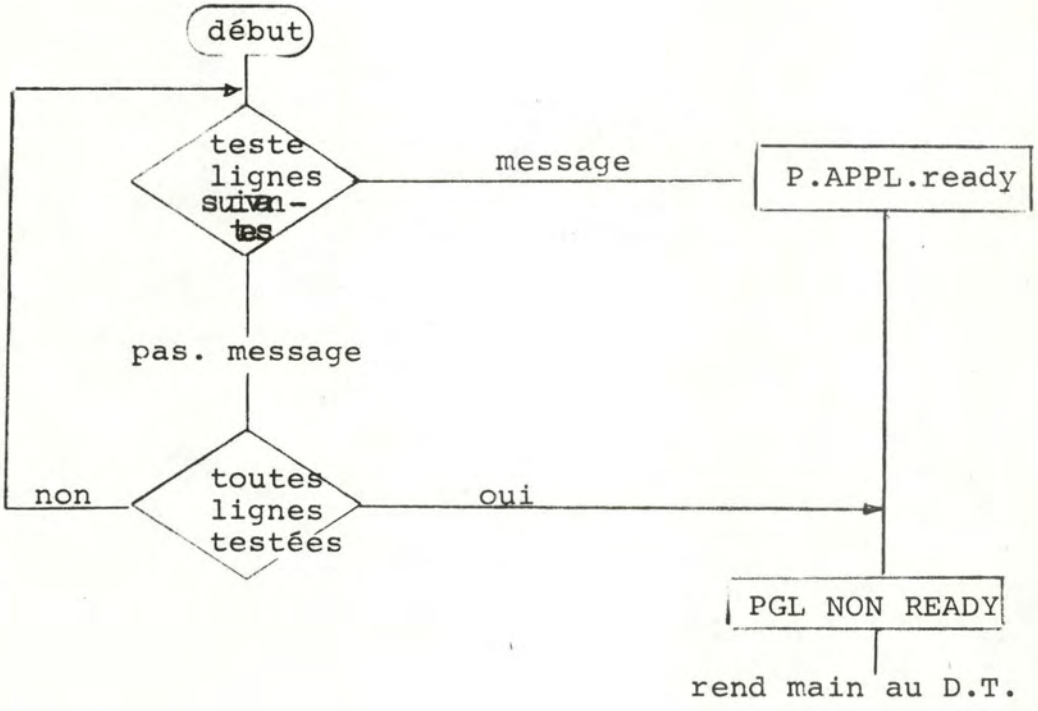
   Lorsque le P.APPL. aura terminé le traitement du message il se remettra en attente (SUSP) et le PGL recommencera à tester les lignes suivantes.

Si le PGL ne trouve rien sur aucune ligne, il rend la main au DT en indiquant que lui même ne doit plus se dérouler. Le DT rendra alors la main au DTP.
Lorsqu'un nouveau message arrivera, le PGL sera de nouveau positionné prêt à se dérouler, lors de l'interruption due à l'arrivée de ce message.

Remarque :

Le PGL reçoit la main du DT toujours au même endroit = début.

DONC travail du PGL : (voir détails plus loin)



Résumé : chaque bit d'informations du D.T. pour chaque programme

bit PGL :

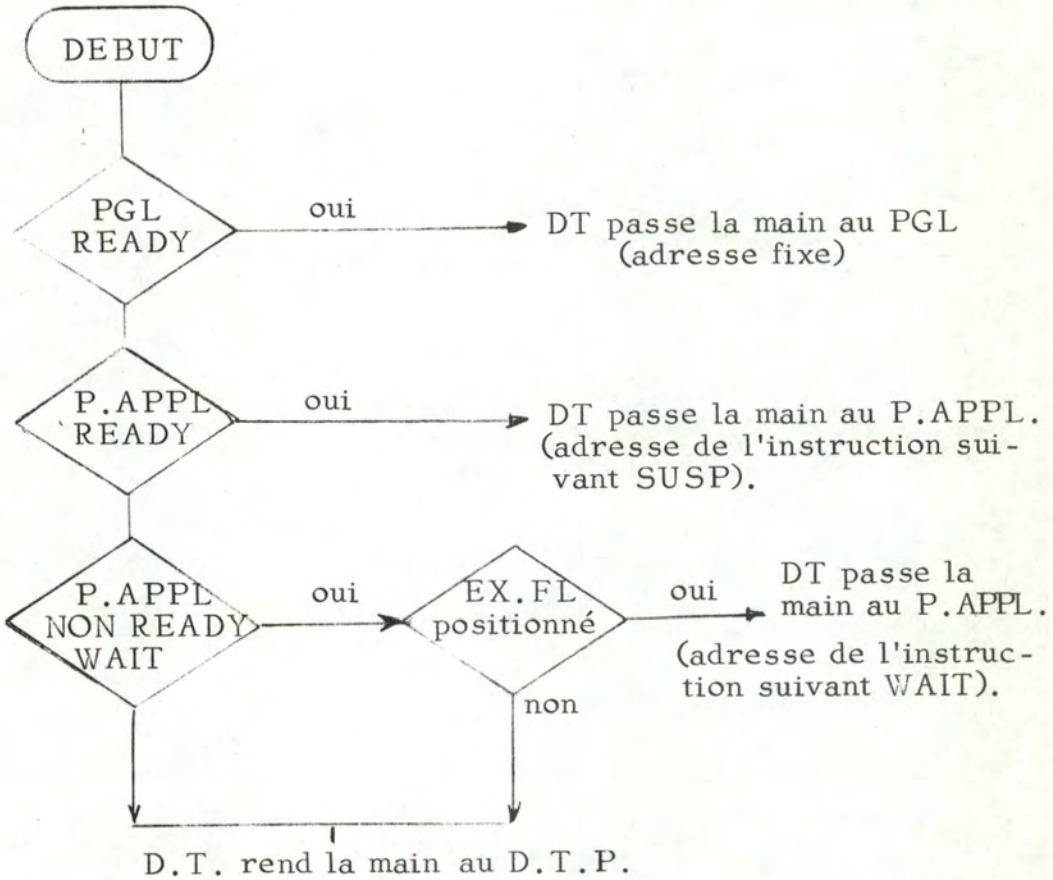
- READY : - par SUSP du P. APPL.
  - par interruption fin de message si P.APPL. est en SUSP.
- NON READY : - lorsqu'il rend la main au DT.

bit P.APPL :

- READY : - lorsque DT lève le WAIT et passe la main au P.APPL.
  - lorsque PGL veut passer la main au P.APPL car un message est arrivé et doit être traité.
- NON READY: - lorsque P.APPL. attend un message (SUSP).
- NON READY WAIT : - lorsque le P.APPL. attend une fin I/O (WAIT).

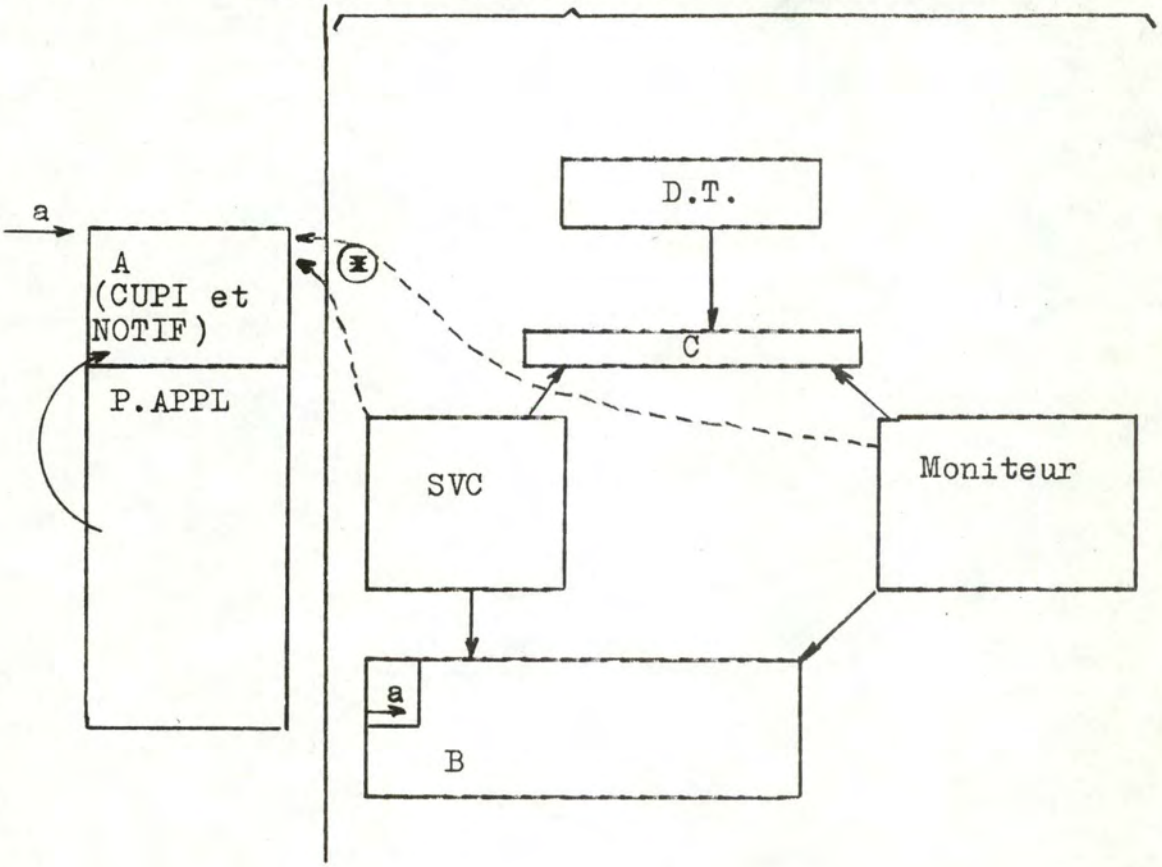
Donc travail DT :

<table>
<tr><td colspan="4">D.T.</td></tr>
<tr><td>bit état</td><td>ADR.RET.</td><td>bit état</td><td>ADR.RET.</td></tr>
<tr><td colspan="2">PGL</td><td colspan="2">P.APPL.</td></tr>
</table>


DEBUT
PGL READY
oui
DT passe la main au PGL (adresse fixe)
P.APPL READY
oui
DT passe la main au P.APPL. (adresse de l'instruction suivant SUSP).
P.APPL NON READY WAIT
oui
EX.FL positionné
oui
DT passe la main au P.APPL.
(adresse de l'instruction suivant WAIT).
non
D.T. rend la main au D.T.P.

III.2.3. ZONES DE COMMUNICATION.

III.2.3.1. ZONES DE COMMUNICATION DES INFORMATIONS.



(*) CPUT ACTIVATE ira mettre l'adresse de A dans la zone B pour que les SVC sachent où se trouve l'adresse des zones CUPI et NOTIF du P.APPL.
Cette adresse servira aux SVC CUPI, CGET, CGET SYS, CPUT ACTIVATE, CPUT DEACTIVATE, CPUT Message, CPUT ACTION.

III.2.3.1.1. ACCESSIBLES AU P.APPL. (et aux macros SVC et aux modules du moniteur).

"ZONE A" :

1. Lorsque le P.APPL emploie une macro CGET, CPUT, il doit savoir comment elle s'est déroulée et s'il peut continuer. Il faut donc une zone pour le signaler au P.APPL. Ce sera la zone NOTIF au début du P.APPL, qui sera positionnée lors du déroulement de la macro mais analysée par P.APPL.

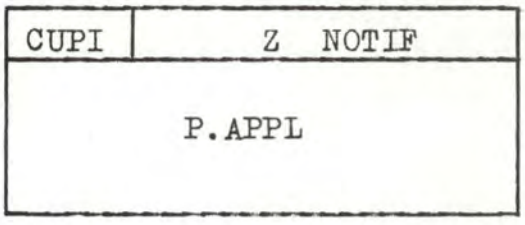
2. Il faut signaler au P.APPL qu'il y a un message dans le pool lorsqu'il attend.

   Lorsque le P.APPL s'est mis en WAIT, et que l'I/O en question est terminée, il faut lui indiquer s'il y a eu erreur ou si l'opération s'est bien déroulée.

   Cela se trouvera dans une zone CUPI au début du P.APPL.
   Il y a pour chaque ligne une zone CUPI accessible à la macro CUPI de l'utilisateur et au PGL. Lors des opérations I/O, la zone CUPI de la ligne concernée est positionnée et la macro CUPI transfère cette zone dans la zone CUPI du P.APPL.

3. Si le P.APPL veut demander le détail des erreurs survenues lors des opérations, il peut le faire par la macro CGET SYS.

   Il y a donc pour chaque ligne une zone SYS NOT indiquant le détail des erreurs survenues lors d'une I/O et cette zone sera transférée dans NOTIF du P.APPL par la macro CGET SYS.



Donc CUPI :

message	erreur lect.	erreur écrit.

- positionné lors des opérations I/O;
- demandé par la macro instruction CUPI de l'utilisateur;
- remis à zéro lors de la macro CUPI ou du lancement d'une autre opération.

NOTIF : cf. chap.erreurs III.3.3.4.

SYS NOT : cf. chap.erreurs III.3.3.4.

III.2.3.1.2. ACCESSIBLES AUX MACROS = SVC (et au moniteur) : "zone B".

Le moniteur et les macros doivent toujours connaître la ligne sur laquelle ils travaillent ainsi que les adresses des différentes zones et paramètres de cette ligne.
Il y a donc une zone ligne = zone B à format standard, dans laquelle se trouvent tous ces renseignements.
Pour plus d'explication, cfr. chapitre III. 3

Ad CUPI, NOTIF : cf. III.2.3.1.1.

N° ligne courant : mise à jour par PGL, permet de retrouver les informations de cette ligne sur laquelle on travaille.

Table n° lignes log/phys. : fixe, permet conversion entre n° ligne logique et physique.

Table codes : fixe, permet la conversion entre les codes des terminaux et de l'ordinateur.

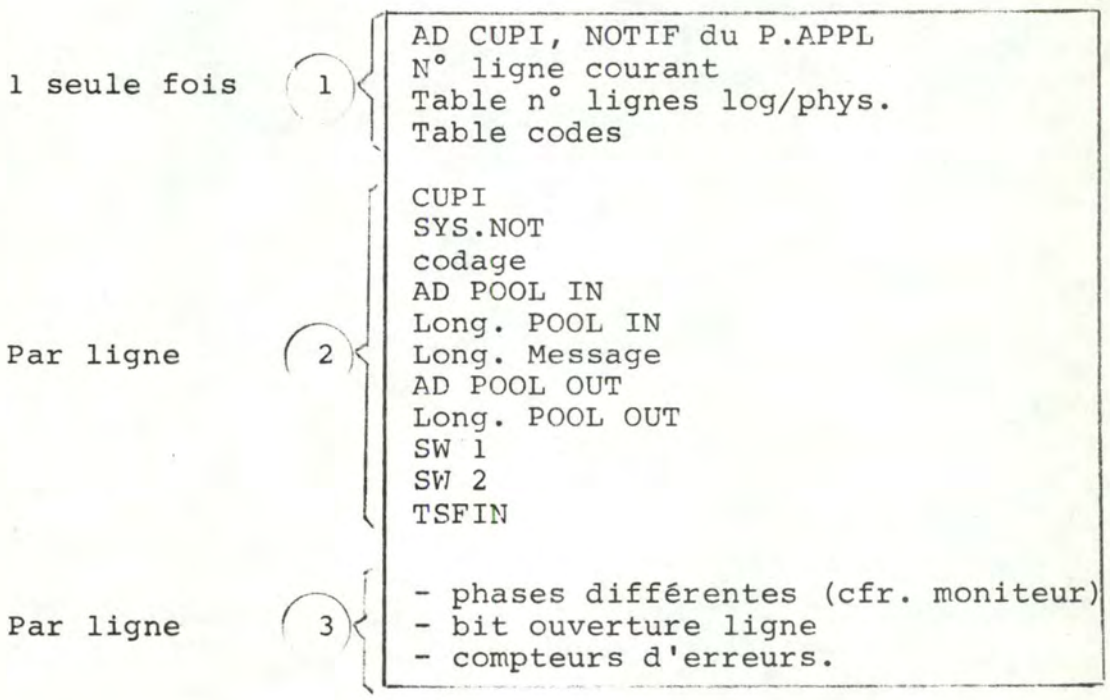
CUPI SYS NOT : remplis lors des opérations 1/0.

Codage : - rempli lors de CPUT ACTIVATE de la ligne
- indique le code employé par le terminal de la ligne;
- employé lors de demande (CGET) ou donnée (CPUT) de message ; si le code est différent de celui de l'ordinateur, le message sera transcodé suivant la table de codes.

Ad/long. de pool IN/OUT : fixes, permettent de retrouver le pool.

Long.message : rempli lors de chaque arrivée de message.

SW1, SW2, TSFIN : indiquent le déroulement des opérations.



III.2.3.1.3. ACCESSIBLES AU D.T. (et au moniteur et aux SVC) : "zone C".

- bits d'état des PGL et P.APPL) cfr.
- Adresse de retour de PGL et P.APPL) III.2.2

III.2.3.2. ZONE DE COMMUNICATION DU MESSAGE : POOL.

Les pools de chaque ligne sont accessibles à partir de leur adresse en zone ligne B2.

- Simplification maximum de la gestion des pools :

C'est-à-dire 2 pools par ligne :

- IN pour la réception d'un message
- OUT pour l'émission d'un message,

leur longueur est la longueur maximum d'un message (+ 2 pour caractères STX, ETX en écriture).

IN SW1 = 1 = ON
- si le pool est rempli par le périphérique,
- positionné lors de la réception du message (end).

SW1 = 0 = OFF
- si le pool est vide et prêt à être rempli par le périphérique,
- positionné par CGET, lancé par l'utilisateur après transfert hors du pool.

OUT SW2 = 1 = ON
- si le pool est rempli par le P.APPL. et prêt à être envoyé vers le périphérique.

- positionné par CPUT lancé par le P.APPL, avant transfert dans le pool.

SW2 = 0 = OFF

- si le pool est vidé par le périphérique et prêt à être rempli par le P.APPL pour une écriture,
- positionné lors de la routine d'interruption end, après écriture.

Caractères de contrôle.

- En début de pool OUT, on a STX (premier caractère envoyé sur la ligne).
- La macro-instruction CPUT ajoutera dans le pool OUT, en fin de message le caractère ETX qui sera envoyé après le message.

III.2.4. INSTRUCTIONS DE COMMUNICATION.

SUSP - le P.APPL. attend qu'on lui passe un message, il rend donc la main au DT en se positionnant non ready et sera de nouveau mis ready lorsque le PAL aura un message à lui donner.

WAIT - le P.APPL. rend la main au DT et demande à la recevoir lorsque l'opération d'I/O lancée sera terminée. Lorsque cette opération est terminée, elle positionne un bit "EX.FLAG" que le DT peut tester pour lever le WAIT.

CUPI - transfère la zone CUPI propre à la ligne courante dans la zone CUPI du P.APPL.

- utile après le début du P.APPL pour savoir si il y a un message et après un WAIT pour savoir comment s'est déroulée l'opération.

CGET SYS - transfère la zone SYS NOT propre à la ligne courante dans la zone NOTIF du P.APPL.

- utile si le P.APPL gère lui même les erreurs.

Remarque :

On pourrait employer CGET SYS : / CUPI : } n° ligne

pour tester une ligne autre que la ligne courante.

Schéma des macro-instructions suivantes :

1. - Vérification de la validité,
2. - soit transfert entre pool et zone utilisateur (CGET),
   - soit lancement d'une opération I/O (CPUT ACTION, ACTIVATE, DEACTIVATE),
   - soit les deux à la fois (CPUT).

Vérification de validité.

TSFIN - aide à surveiller l'emploi correct des macro-instructions,

- positionné lors du lancement d'une lecture et enlevé lors de la fin de lecture,

donc : si le P.APPL. veut écrire et que TSFIN est positionné, il faut vérifier que le terminal n'a pas encore commencé à envoyer.

Si le terminal a commencé : on ne peut pas écrire.

Si le terminal n'a pas commencé : on peut arrêter la lecture (HIO) et lancer une écriture.

Si le P.APPL. écrit :

CGET - il est prévenu par NOTIF s'il n'y a rien dans le pool,

CPUT - il est prévenu par NOTIF si une écriture est en cours ou si le terminal a commencé a envoyer et alors l'écriture n'est pas acceptée. Si il y a message à prendre, l'écriture est lancée.

CPUT ACTION - il est prévenu s'il y a un message à prendre (et la lecture est lancée), si une lecture ou une écriture est déjà lancée (et la lecture n'est pas lancée).

CGET Message - obtient le transfert d'un message depuis le pool IN jusque la zone de lecture du P.APPL.

- paramètres.

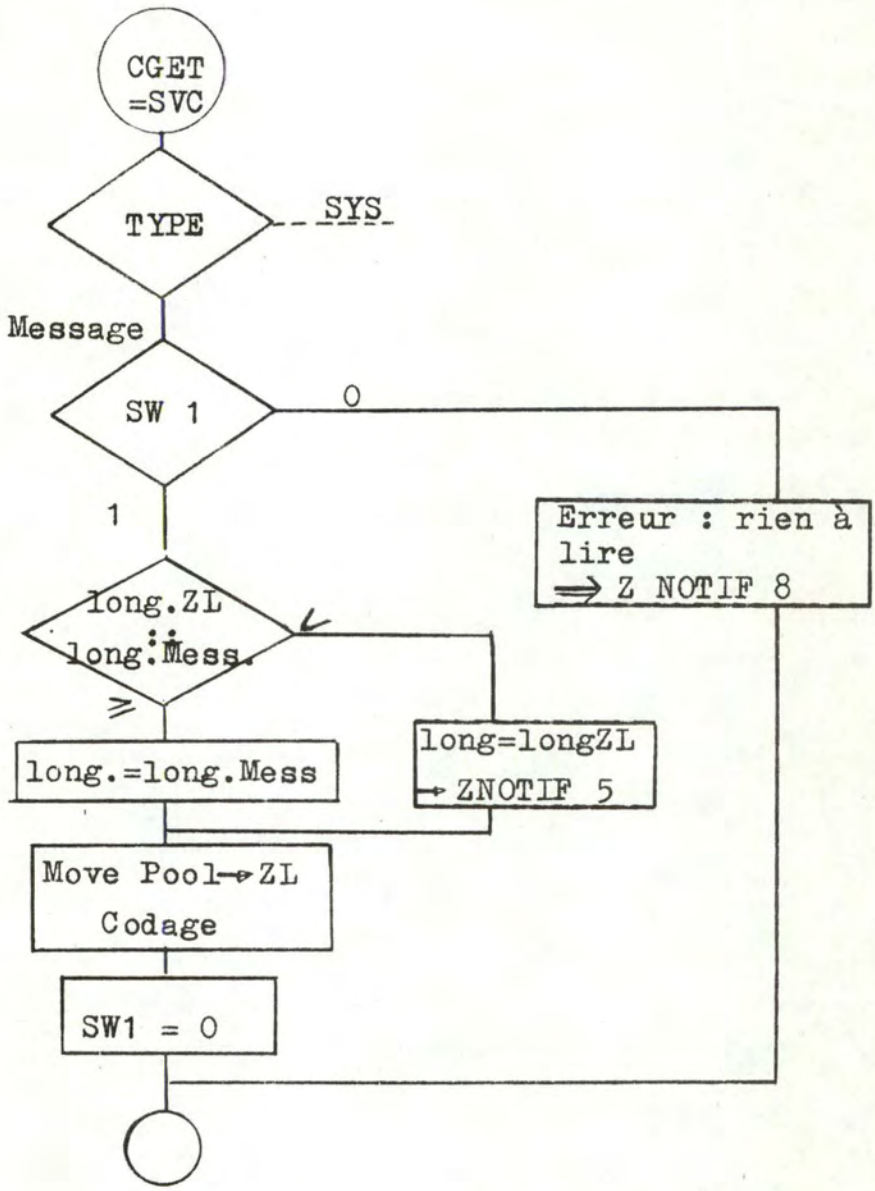
type zone de lecture = ZL longueur ZL

} remplis par l'utilisateur

n°ligne courant	table codes	Ad NOTIF P.APPL	
Ad.Pool IN	Long.message	Codage	SW1

} dans zone B remplie par système

Organigramme.



Remarque :

Longueur Message = longueur Byte Count au début (= long. pool IN) - longueur restant dans BC à la fin de la lecture,

donc : lors de EXCP on place la longueur du BC dans longueur message,

lors du END de lecture : on diminue longueur Message de BC restant.

CPUT Message

- obtient le transfert d'un message depuis la zone écriture du P.APPL jusqu'au pool OUT.
- lance l'opération d'écriture.
- **paramètres**

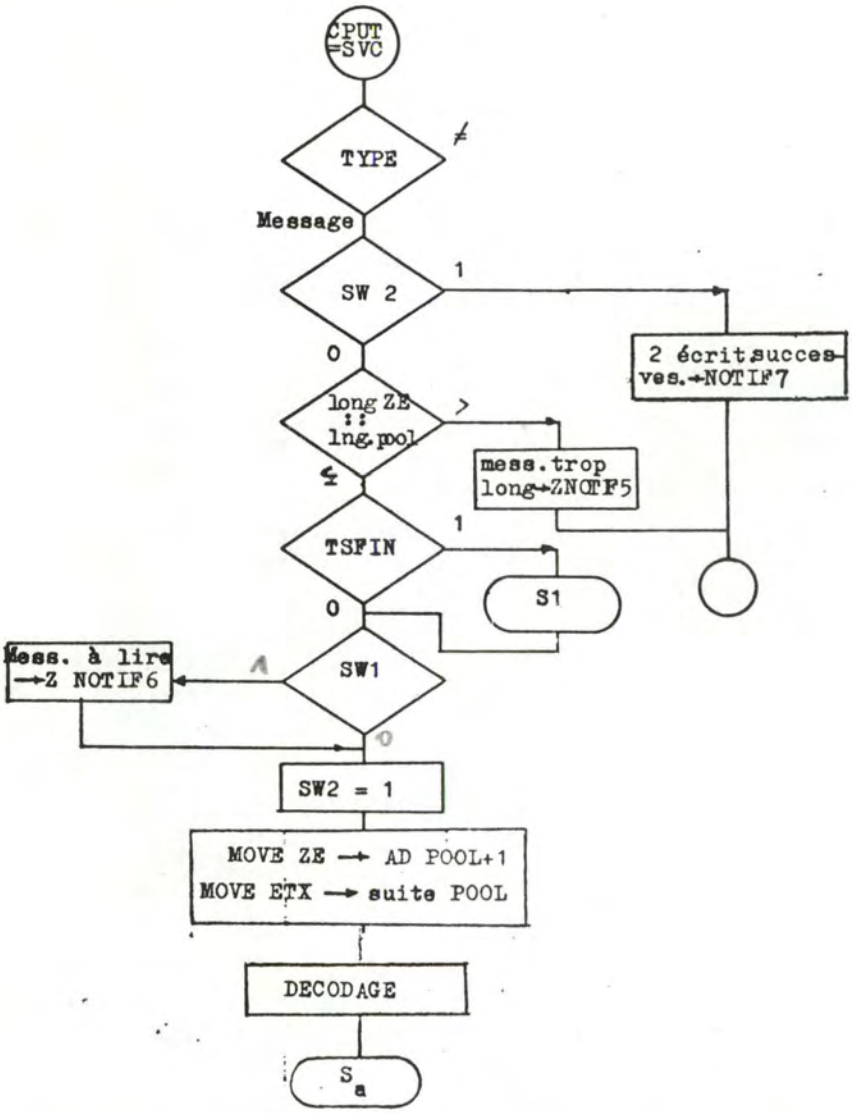
Utilisateur

type zone d'écriture = ZE longueur ZE

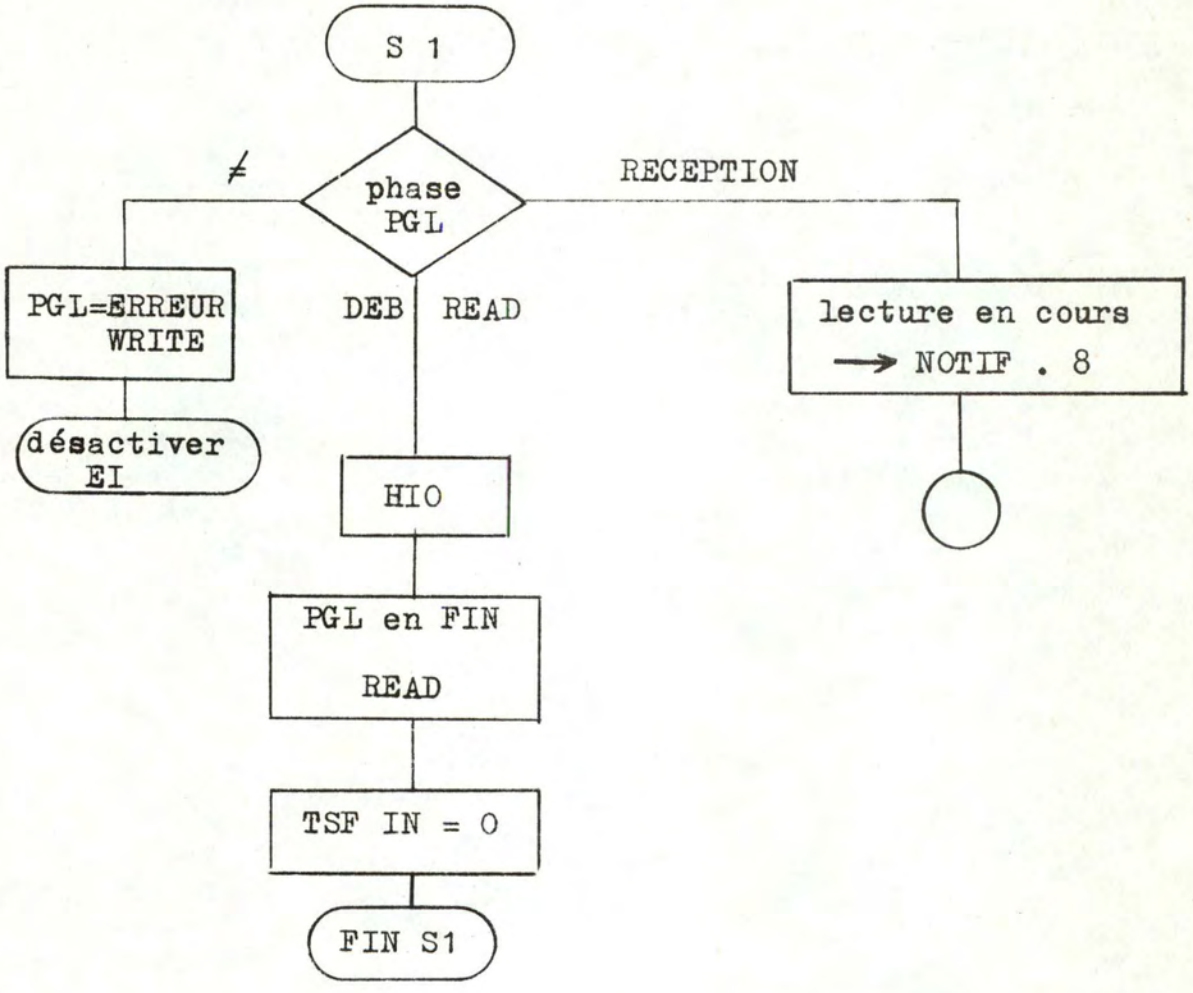
Système

n° ligne courant		table codes	AD NOTIF P.APPL		
AD POOL OUT	long.POOL OUT	SW2	SW 1	TSFIN	Codage
Phases	SYS NOT				

Organigramme.

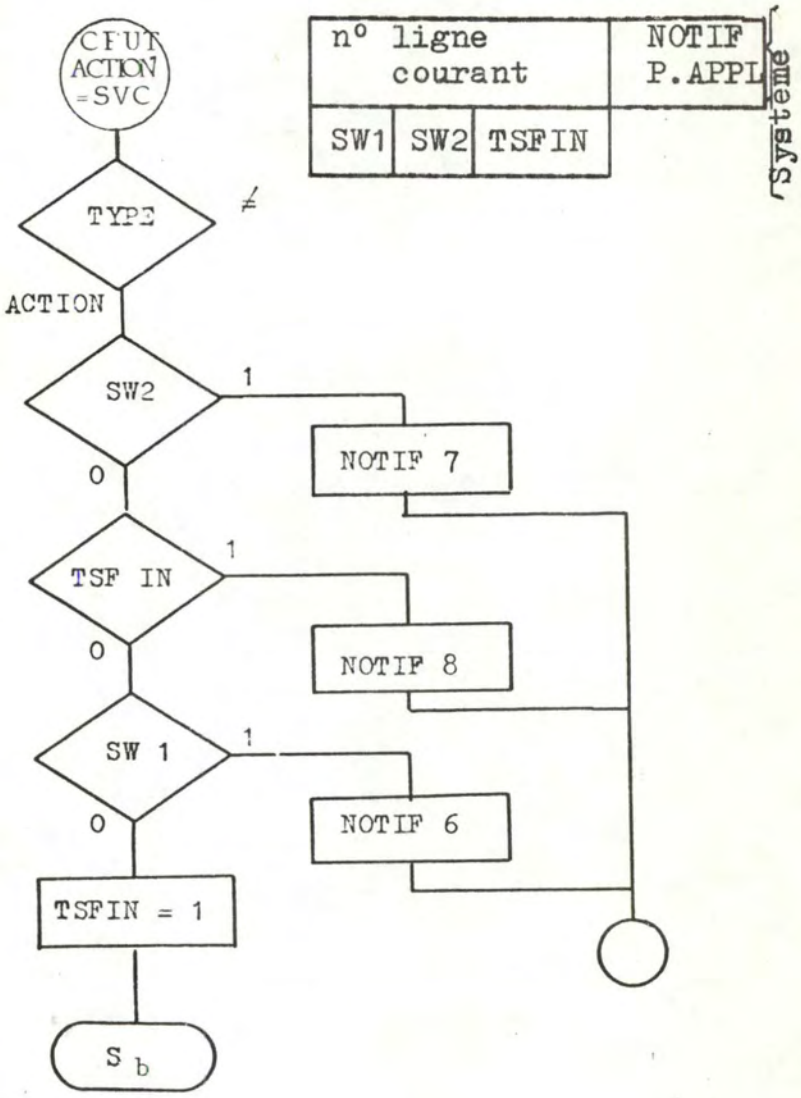


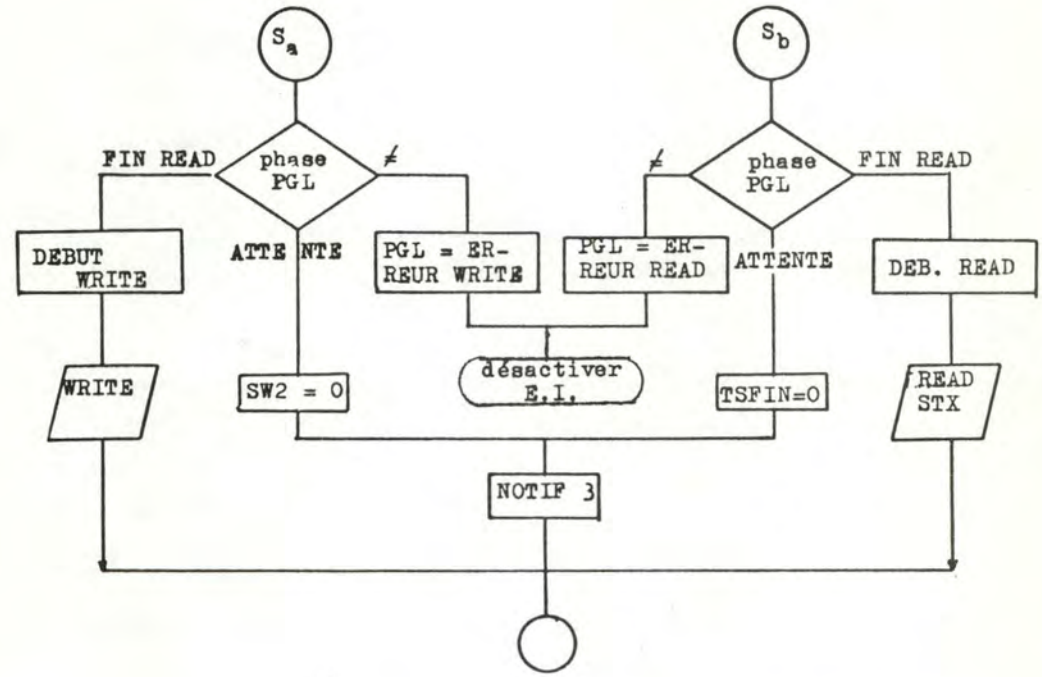
Remarque : 1. pour CPUT de console idem sauf éviter le test SW1 et NOTIF devient impression console.
2. si on permet CPUT n° ligne, il faut commencer par vérifier si la ligne n'est pas inconnue.

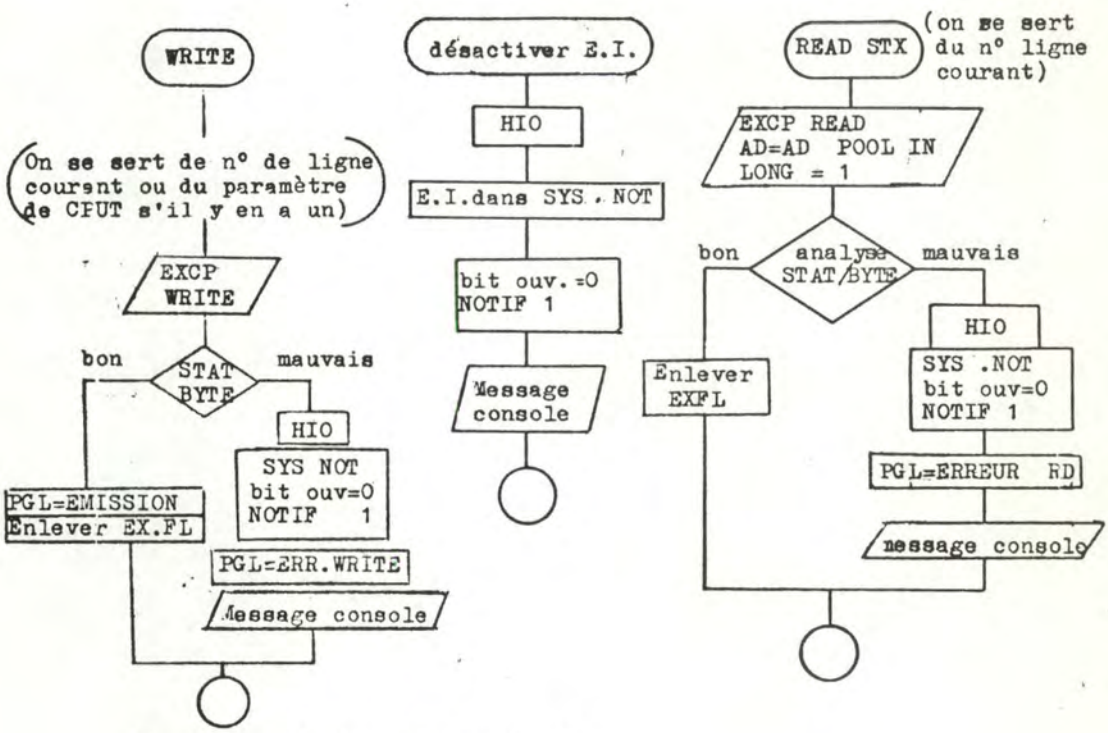

S 1
phase PGL
≠
RECEPTION
PGL=ERREUR WRITE
DEB
READ
lecture en cours
→ NOTIF . 8
désactiver EI
HIO
PGL en FIN READ
TSF IN = 0
FIN S1

CPUT ACTION

- lance une opération de lecture, sans qu'il y ait eu de réponse à une lecture préalable.
- <u>paramètres</u> : Utilisateur { type

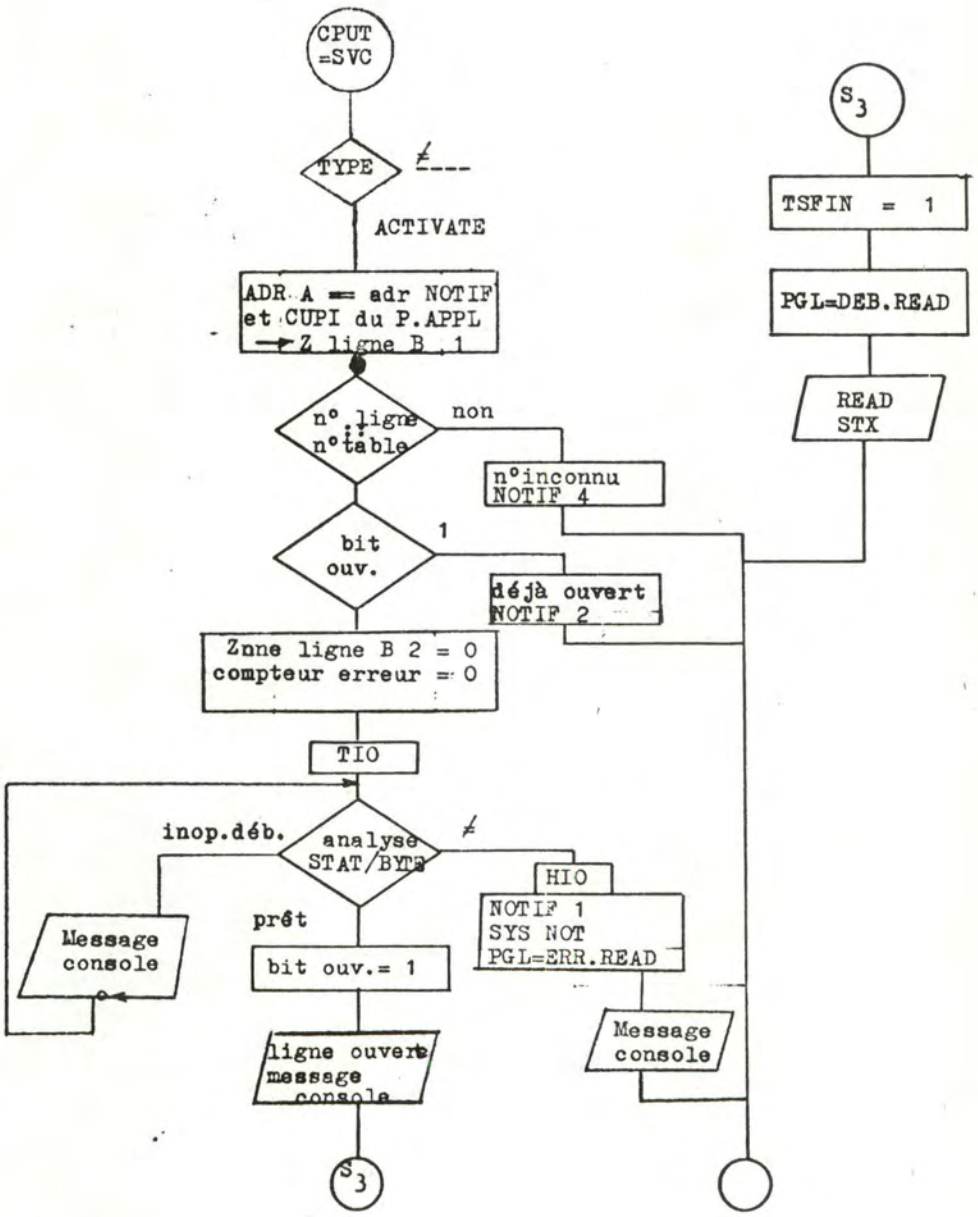






Remarque : EXCP : cfr. T2 et T6.

Organigramme.



Remarque : A partir de la console, on commence en ● et NOTIF est remplacé par 1 message console.

CPUT ACTIVATE

- initialise la ligne
- lance l'opération de lecture
- paramètres

Utilisateur

Adr A.
type
n° ligne
code

table n° ligne	bit ouvert.	phase

NOTIF P.APPL

Code	AD.POOL IN	SW1	TSFIN	long.Mess.	Long. Pool IN
	Phases	SYS NOT			

Système

CPUT DEACTIVATE

- arrête l'opération en cours

  - message entrant non reçu
  - message sortant non envoyé.

- <u>paramètres</u>.

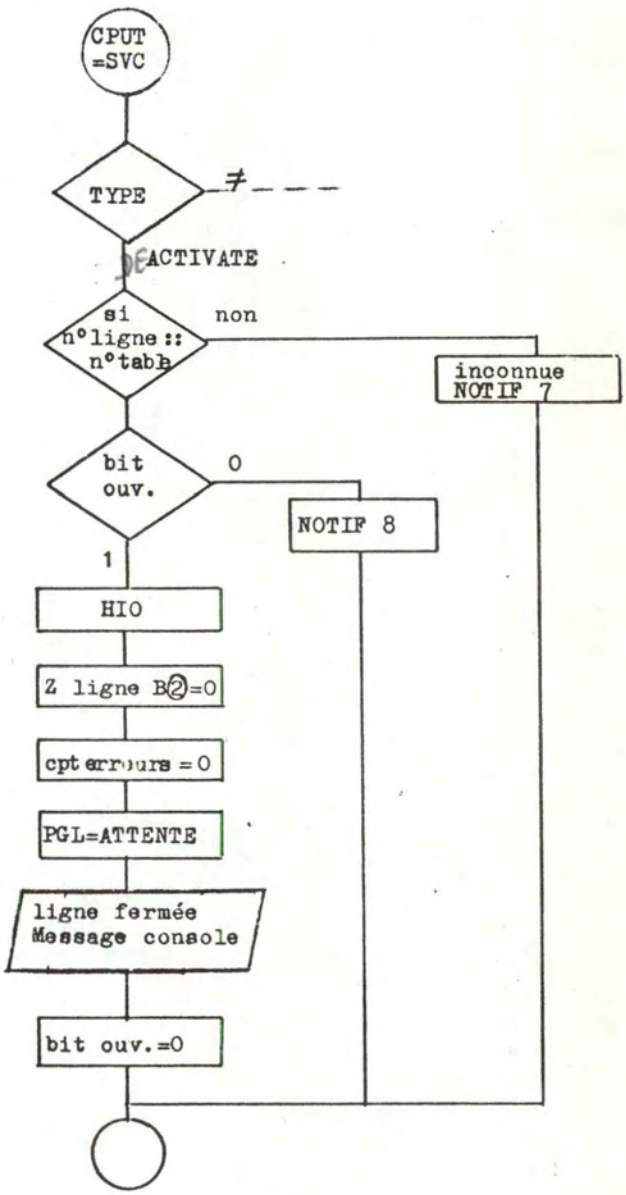
Utilisateur { type, n° ligne } (éventuel)

Système

table n° ligne	bit ouv.	phases

NOTIF P.APPL	n° ligne courant (éventuel).

Organigramme.



Remarque : A partir de la console, NOTIF devient message console.

REMARQUE.

ON PEUT AJOUTER UN DELAI DE PRESENCE DANS LE POOL.

Lorsque le pool est rempli, il faut déclencher un timer qui provoque une interruption après un temps T renseigné dans la macro instruction CPUT ACTIVATE de l'utilisateur et mémorisé dans la zone B (2).

Lorsque le pool est vidé, il faut remettre le timer à zéro.

Si l'interruption se produit, c'est que le temps est dépassé ⟶ signaler l'erreur.

ON PEUT EMPLOYER CGET CELL.
CPUT CELL.

Dans un bloc, on aurait x cellules, et chaque CGET, CPUT remplirait une cellule, en n'envoyant l'opération de lecture/écriture que lorsque un bloc complet devrait être écrit/lu.

LAST CELL indiquerait la dernière cellule.

III.3. MONITEUR DE TELETRAITEMENT.

III.3.1. INFORMATIONS.

III.3.1.1. ZONES D'INFORMATION DE CHAQUE LIGNE.

Le moniteur doit savoir à tout moment où il en est dans son travail pour chaque ligne, et dans quel état se trouve la ligne. Les renseignements se trouvent dans des zones propres à chaque ligne.

Etat de la ligne - Rappel.

Non Opérat.	PRET	OCCUPE	INT. PEND.	MODE FONCT.	ERREUR

Ces informations se trouveront dans le status byte, que le PGL peut demander par TIO.

bit	0	1	2	3	4	5	6
	Code CDe Commande non légale	INOP DEBUT	ERREUR Cadencet.	ERREUR PARITE	BLOC TROP COURT	INOP FONCT	ETX pas 25" après STX

Ces informations se trouveront dans le sense byte que le PGL peut demander par TDV.

Etat du travail du PGL.

a. Différentes phases permettent le contrôle du dialogue.

ATTENTE	DEBUT READ	RECEPT	FIN READ	ERREUR READ	DEBUT WRITE	EMIS-SION	FIN WRITE	ERREUR WRITE

Elles se trouveront dans la zone ligne B 3

Emploi de ces phases : cfr. III.3.3.

b. CCB.

Code termi-nal	Adresse zone ligne	~~USER'S FLAG~~	EXEC FLAG	BYTE COUNT	Adresse CCW courant	~~Adresse CCW suivant~~	STATUS BYTE	SENSE BYTE

User's flag : pas employé.

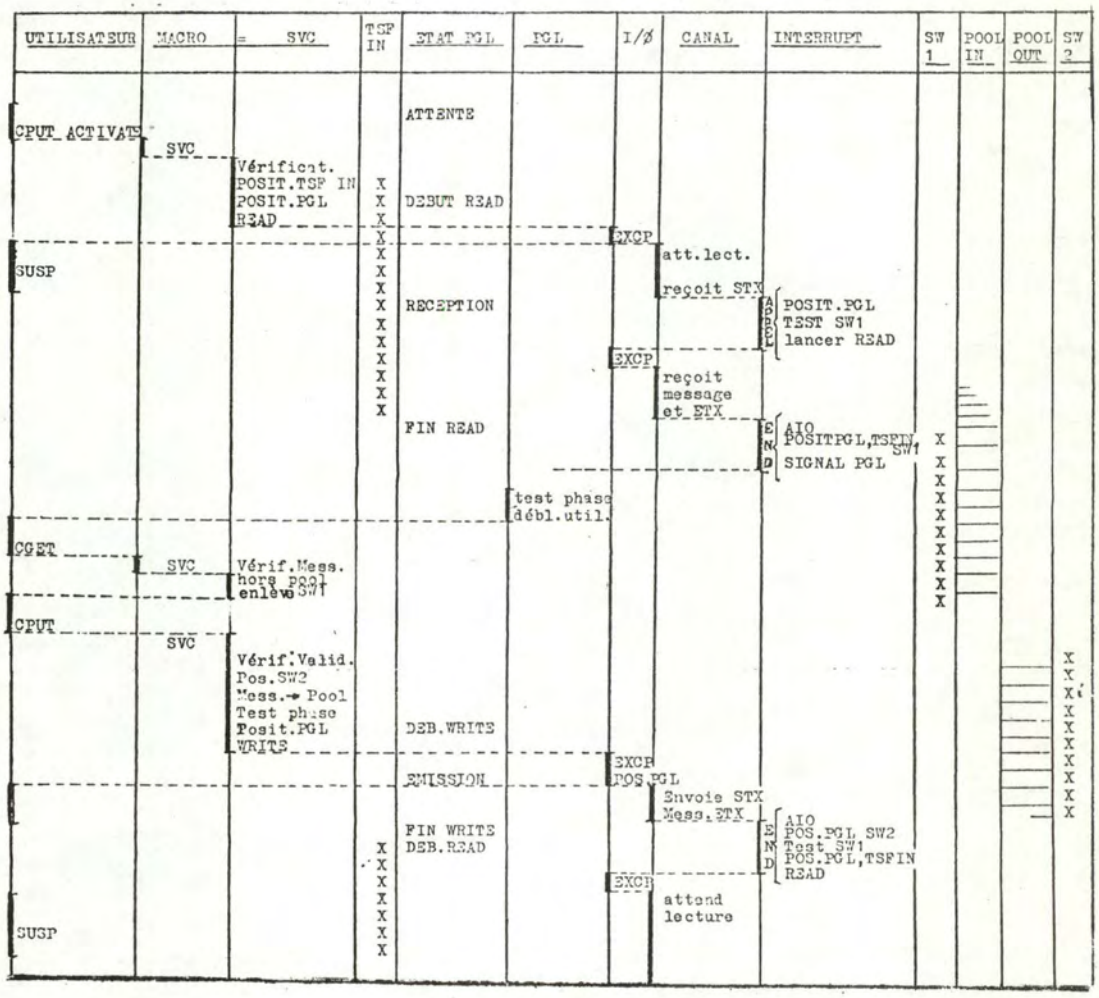
Adresse CCW suivant : servirait en cas de chaînage.

Adresse zone ligne : permet de retrouver les informations concernant la ligne.

EXEC Flag : le bit 7 sera positionné en fin de routine END pour lever le WAIT.

BYTE COUNT : au début d'une opération, il contient le nombre de caractères, en fin d'opération, il contient le nombre de caractères inemployés.

III.3.1.2. EMPLOI DE CES ZONES DANS LE DEROULEMENT NORMAL D'UNE OPERATION LECTURE/ECRITURE.



III.3.1.2. EMPLOI DES ZONES DANS LE DEROULEMENT NORMAL D'UNE OPERATION DE LECTURE/ECRITURE.

Au départ, on suppose qu'une opération de lecture a été lancée sur la ligne 1. (cf. schéma : déroulement général).

1. Le terminal envoie STX, ce qui provoque une interruption, INT.APP.

2. Lorsque l'interruption n'est pas masquée, le programme se déroulant est suspendu pour permettre à la routine d'interruption de se dérouler. Celle-ci lance l'opération de lecture du message dans le pool IN.

3. Le message est envoyé dans le pool IN.

4. Le terminal envoie ETX, ce qui provoque une interruption END.

5. La routine d'interruption positionne PGL prêt s'il ne devait plus reprendre la main c'est-à-dire si PGL NON READY et P.APPL SUSP.

6. Le DT reçoit la main après le traitement de l'interruption, et il finira par la rendre au PGL (lorsque P.APPL en SUSP).

6' Le PGL cherche une ligne ayant envoyé un message, supposons qu'il trouve la ligne 1 en question.

7. Il positionne le P.APPL ready en fournissant dans une zone accessible le n° de cette ligne 1.

7' PGL se positionne non ready car il a terminé.

8. PGL rend la main au DT.

9. Le P.APPL reçoit la main du DT.

10. Le P.APPL demande le message dans une zone de lecture par CGET.

11. Le P.APPL traite le message.

12. Le P.APPL envoie la réponse au message dans le pool OUT par CPUT.

13. CPUT lance également l'opération sur la ligne.

14. Le message est envoyé au terminal.

15. Le P.APPL a terminé, il positionne le PGL ready.

16. Le P.APPL se positionne non ready.

17. Le P.APPL. rend la main au DT, qui la rendra au PGL pour lui permettre de tester les autres lignes.

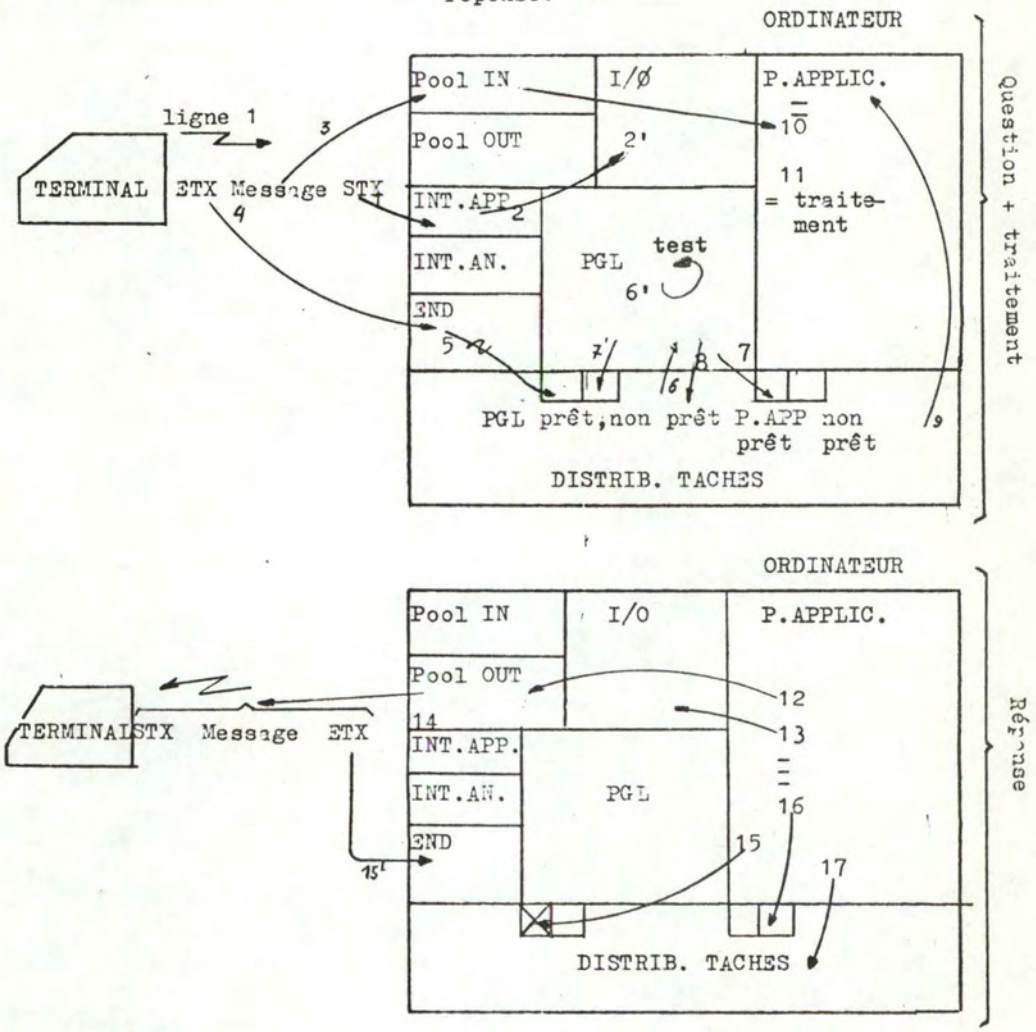
15' Peut arriver à n'importe quel moment après 14.

Le message est envoyé, le dernier caractère, toujours ETX, provoque une interruption END qui initialisera la lecture suivante.

REMARQUE :

Si le P.APPL a écrit WAIT, 15' arrivera toujours avant 15, 16, 17.

DEROULEMENT GENERAL question d'un terminal, traitement, réponse.



III.3.2. SCHEMA DE CIRCULATION DES SIGNAUX AVEC CAS D'ERREURS.

Prenons le déroulement normal d'une opération sur une ligne, les erreurs possibles étant signalées - Tableaux 2 à 7.

Tableau T2.

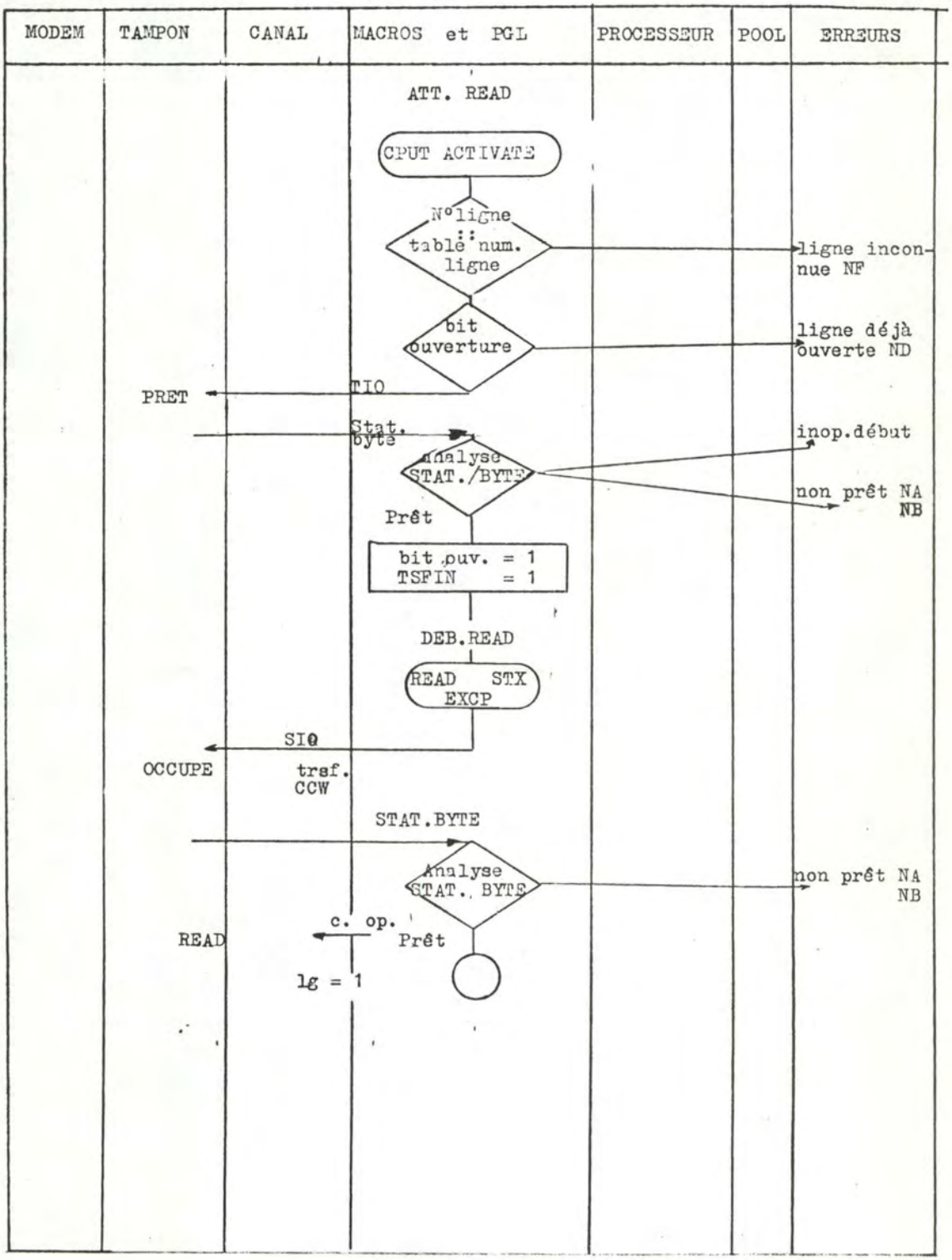

MODEM
TAMPON
CANAL
MACROS et PGL
PROCESSEUR
POOL
ERREURS
ATT. READ
CPUT ACTIVATE
N° ligne
table num. ligne
ligne inconnue NF
bit ouverture
ligne déjà ouverte ND
PRET
TIO
Stat. byte
analyse STAT./BYTE
inop.début
non prêt NA NB
Prêt
bit ouv. = 1
TSFIN = 1
DEB.READ
READ STX EXCP
SIO
OCCUPE
trsf. CCW
STAT.BYTE
Analyse STAT., BYTE
non prêt NA NB
c. op.
READ
Prêt
lg = 1

Tableau 3.

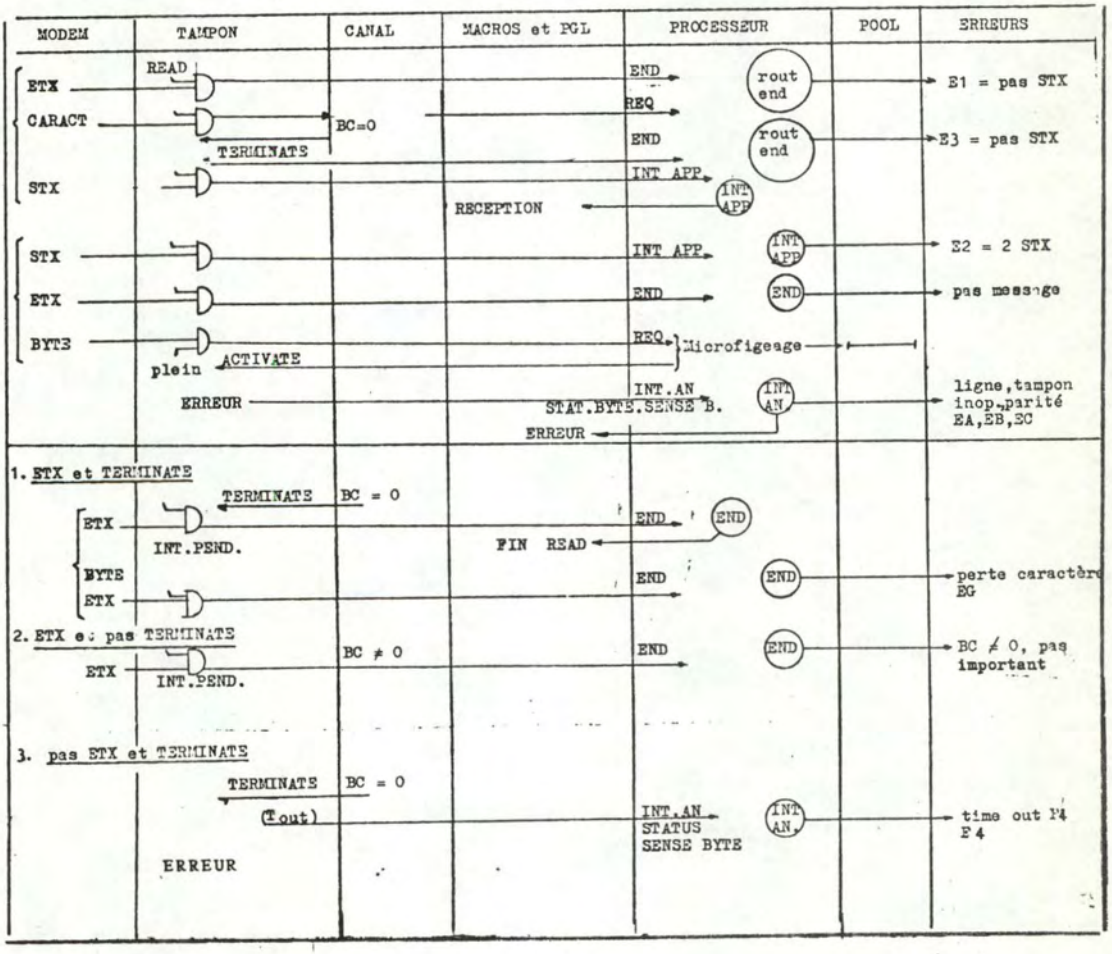

MODEM
TAMPON
CANAL
MACROS et PGL
PROCESSEUR
POOL
ERREURS
READ
ETX
CARACT
STX
BC=0
TERMINATE
END
REQ
INT APP
RECEPTION
rout end
E1 = pas STX
E3 = pas STX
E2 = 2 STX
pas message
BYTE
plein
ACTIVATE
ERREUR
Microfigeage
INT.AN
STAT.BYTE.SENSE B.
ligne,tampon inop.,parité EA,EB,EC
1. ETX et TERMINATE
INT.PEND.
FIN READ
perte caractère EG
2. ETX et pas TERMINATE
BC ≠ 0
BC ≠ 0, pas important
3. pas ETX et TERMINATE
(Tout)
STATUS SENSE BYTE
time out P4

Tableau 4.

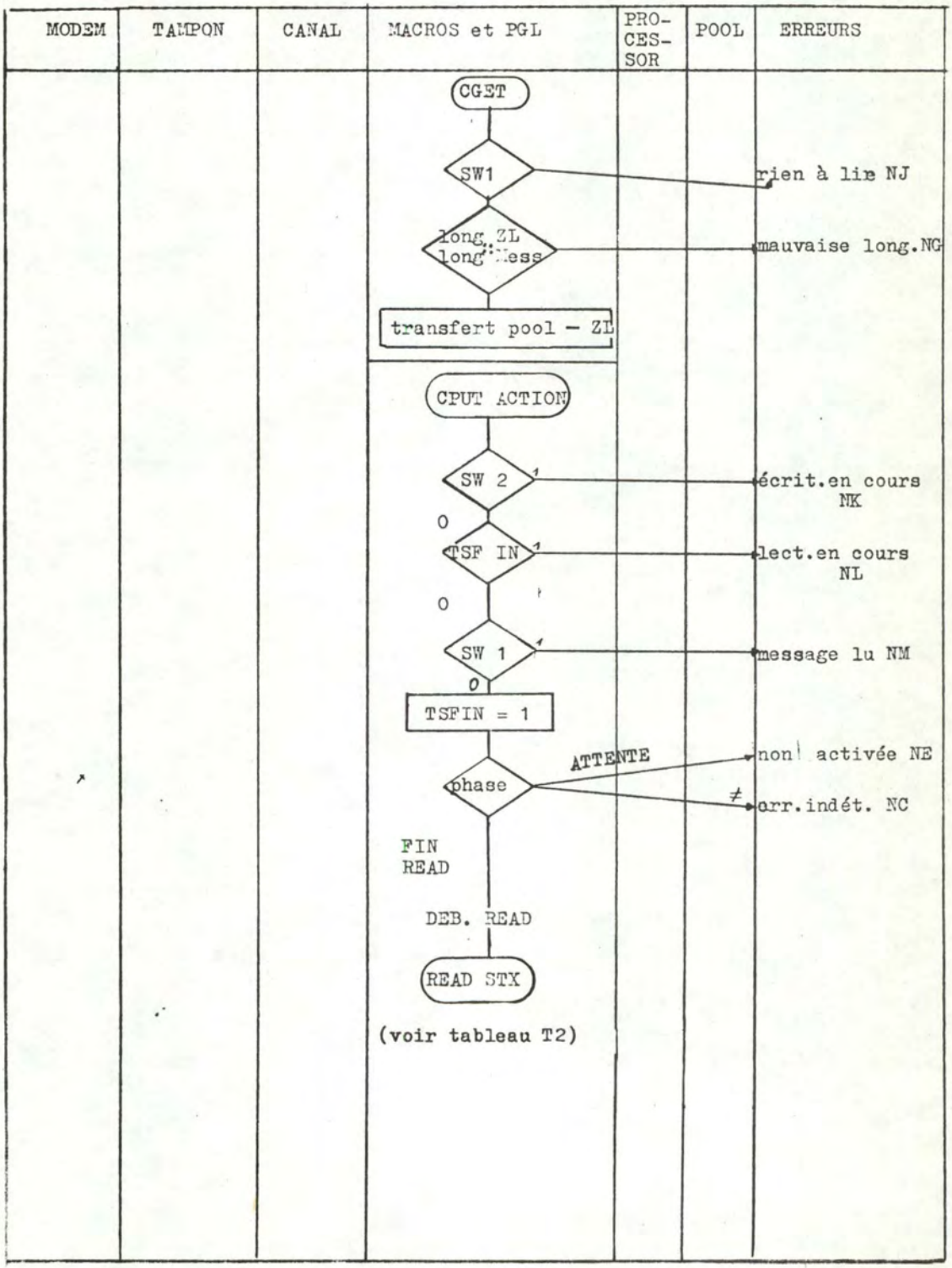
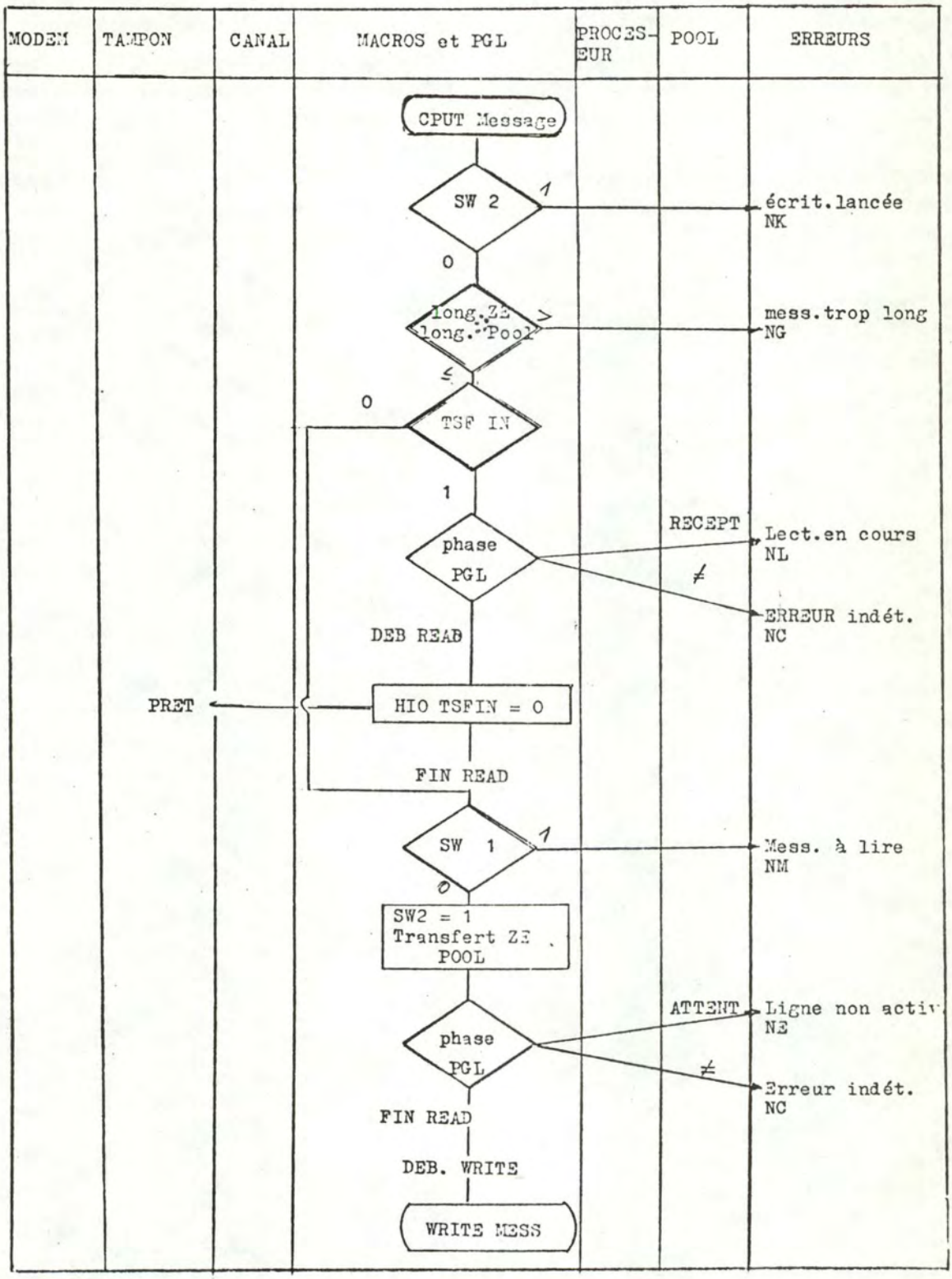

MODEM
TAMPON
CANAL
MACROS et PGL
PRO-CES-SOR
POOL
ERREURS
CGET
SW1
rien à lire NJ
long. ZL
long. less
mauvaise long.NG
transfert pool – ZL
CPUT ACTION
SW 2
écrit.en cours NK
0
TSF IN
lect.en cours NL
0
SW 1
message lu NM
0
TSFIN = 1
ATTENTE
phase
non activée NE
≠
arr.indét. NC
FIN READ
DEB. READ
READ STX
(voir tableau T2)

Tableau 5.


MODEM
TAMPON
CANAL
MACROS et PGL
PROCES-EUR
POOL
ERREURS
CPUT Message
SW 2
1
écrit.lancée
NK
0
long. ZE
long. Pool
>
mess.trop long
NG
≤
0
TSF IN
1
phase
PGL
RECEPT
Lect.en cours
NL
≠
ERREUR indét.
NC
DEB READ
PRET
HIO TSFIN = 0
FIN READ
SW 1
1
Mess. à lire
NM
0
SW2 = 1
Transfert ZE
POOL
ATTENT
Ligne non activ
NE
phase
PGL
≠
Erreur indét.
NC
FIN READ
DEB. WRITE
WRITE MESS

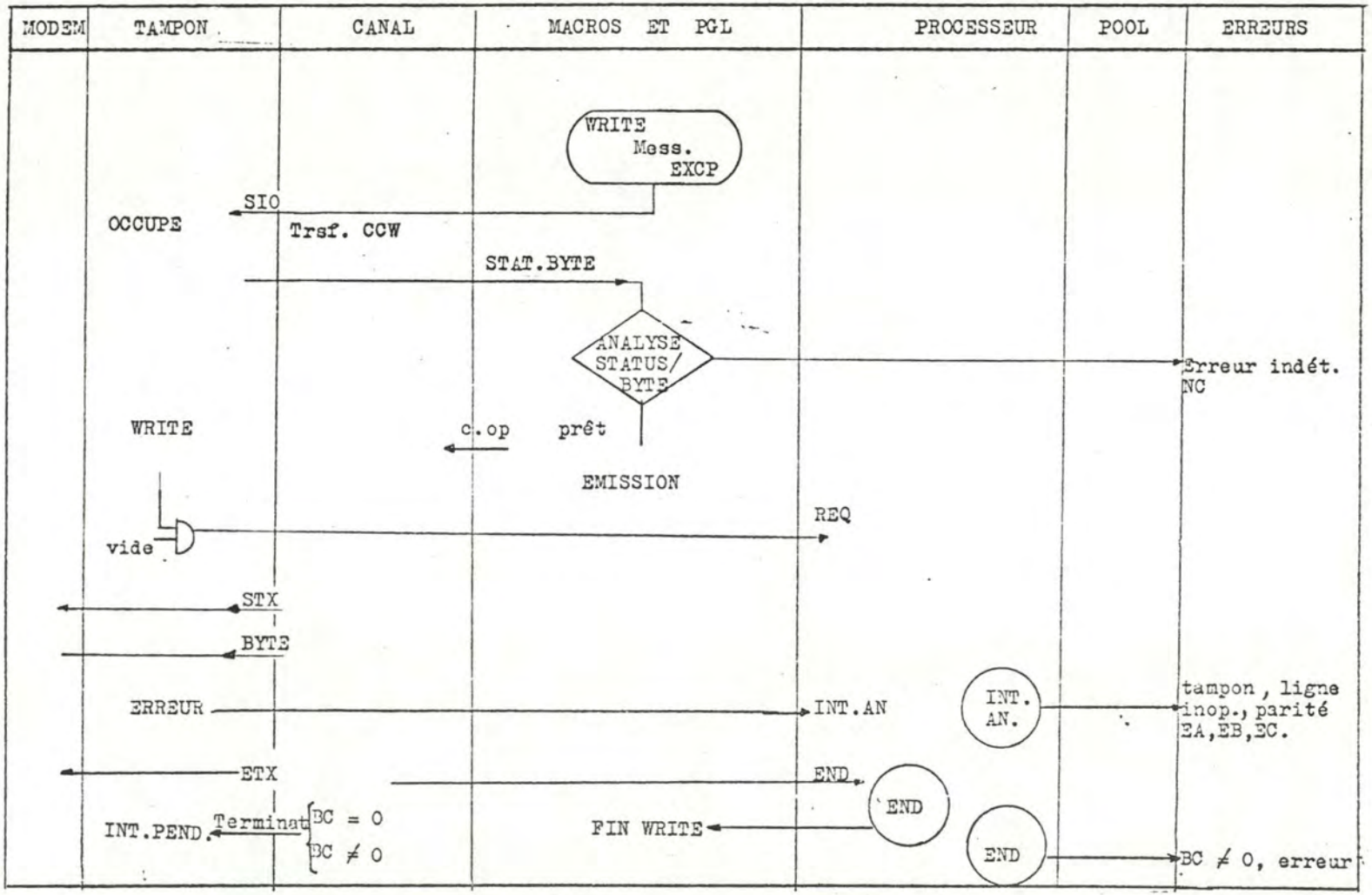
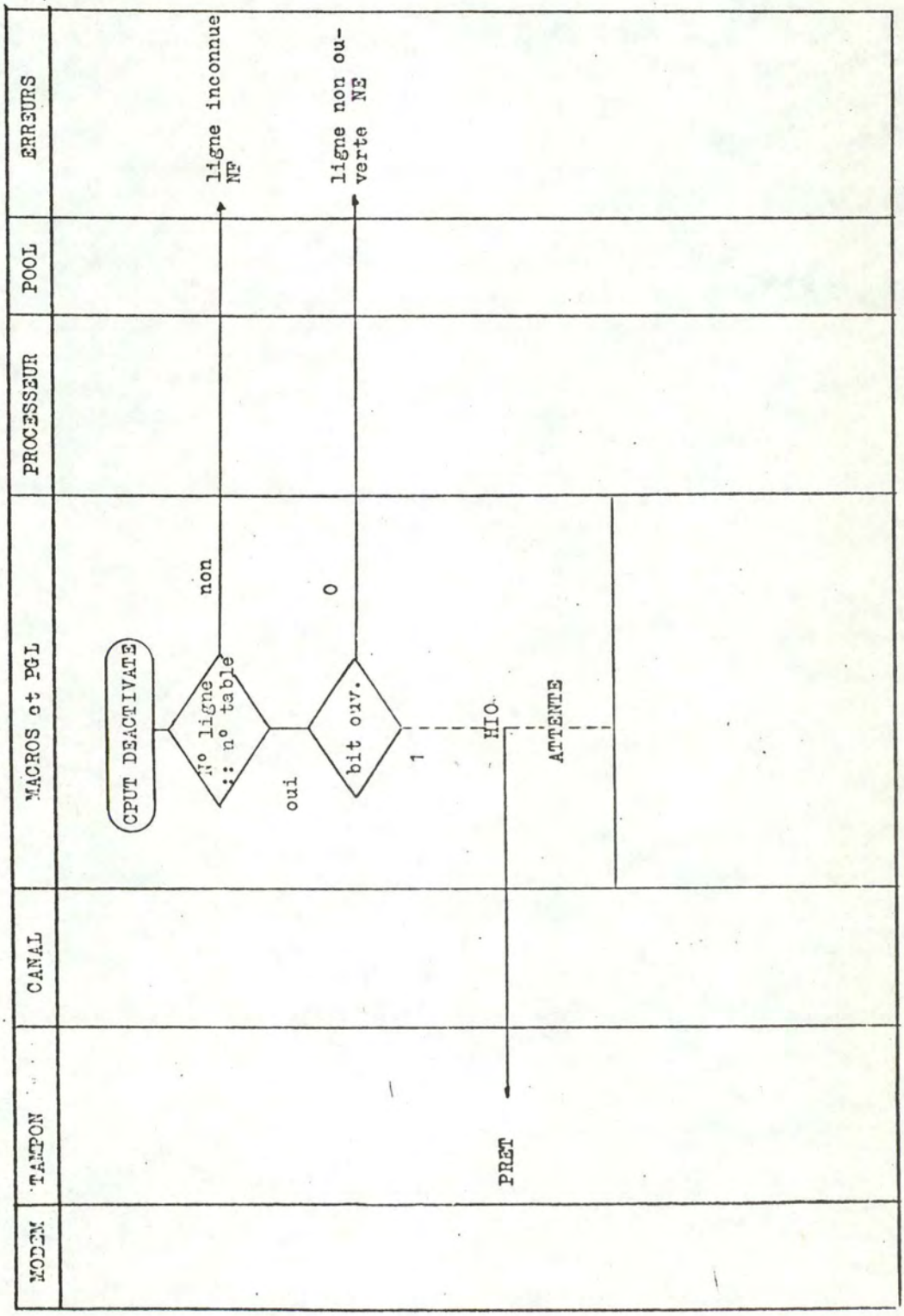

MODEM
TAMPON
CANAL
MACROS ET PGL
PROCESSEUR
POOL
ERREURS
WRITE
Mess.
EXCP
SIO
OCCUPE
Trsf. CCW
STAT.BYTE
ANALYSE
STATUS/
BYTE
Erreur indét.
NC
WRITE
c.op
prêt
EMISSION
REQ
vide
STX
BYTE
ERREUR
INT.AN
INT.
AN.
tampon , ligne
inop., parité
EA,EB,EC.
ETX
END
END
INT.PEND.
Terminat
BC = 0
BC ≠ 0
FIN WRITE
END
BC ≠ 0, erreur

Tableau 6.

Tableau 7.


MODEM
TAMPON
CANAL
MACROS et PGL
PROCESSEUR
POOL
ERREURS
CPUT DEACTIVATE
n° ligne :: n° table
non
oui
ligne inconnue NF
bit ouv.
0
1
ligne non ouverte NE
HIO
ATTENTE
PRET

III.3.3. MODULES DU MONITEUR.

III.3.3.1. PROGRAMME DE GESTION DE LIGNES OU PGL.

III.3.3.1.1. PRISE DE CONTROLE DU PGL.

Le PGL devient ready et reçoit la main :

- soit lorsqu'un message est arrivé d'un terminal, si le P.APPL. attend;
- soit lorsque le P.APPL a terminé un traitement de message.

III.3.3.1.2. ROLE DU PGL.

Comme il y a peu de lignes, le PGL les teste toutes

- jusqu'à trouver un message,
- ou jusqu'à les avoir toutes testées sans rien trouver.

Il teste d'abord si elles sont "ouvertes" (sinon il n'ya sûrement pas de message) et ensuite il teste si un message est arrivé, c'est-à-dire si elles se trouvent dans la phase FINREAD.

Le test des lignes se fait de façon circulaire pour donner autant de chances à chacune. A chaque fois que le PGL reçoit la main, il teste la ligne suivante de celle sur laquelle il avait trouvé le dernier message.

III.3.3.1.3. FIN DU PGL.

Si le PGL trouve un message, il prévient le P.APPL. Il recommencera à tester les lignes lorsque le P.APPL aura terminé de traiter ce message et se sera remis en attente par SUSP.

Si le PGL ne trouve de message sur aucune ligne, il rend la main au DT. Il recevra ensuite la main lorsqu'un nouveau message sera arrivé depuis un terminal.

Lorsque le PGL prévient le P.APPL. de l'arrivée d'un message, il lui signale sur quelle ligne le message est arrivé en l'indiquant dans le n° courant de ligne.

A partir de ce n° courant de ligne, on doit pouvoir trouver les informations relatives à cette ligne et se trouvant dans la "zone ligne B".

Il faudra donc une table des lignes donnant l'adresse de ces informations pour chaque ligne.

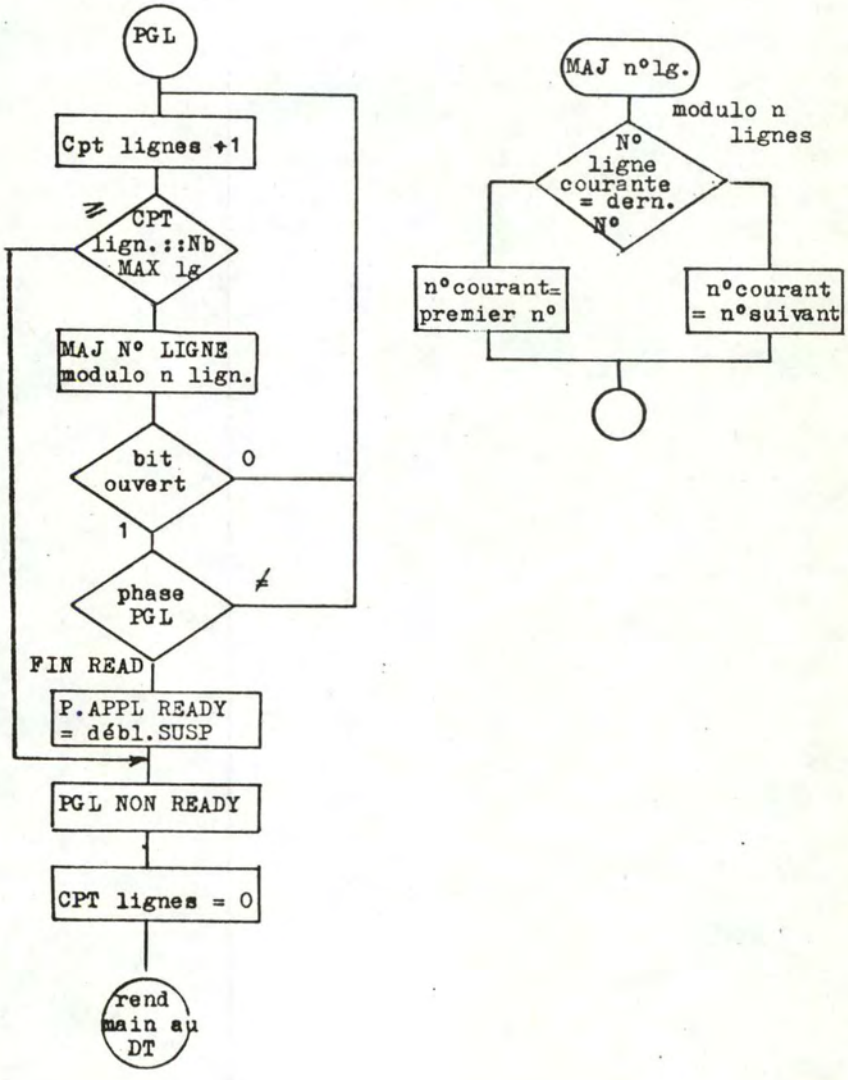
n° logique	n° physique	Adresse dans "zone ligne B"
1 2 3 4 ' '	' ' ' ' ' '	' ' ' ' ' '

A moins que le n° logique ne permette de calculer cette adresse directement.

Le PGL emploie deux zones d'informations qui lui sont réservées :

1. compteur de lignes = CPT lignes.
2. nombre maximum de lignes = Nb Max.lg.

III.3.3.1.4. Organigramme.



III.3.3.2. INTERRUPTIONS.

Priorités décroissantes: INT.APP,INT.AN,END.

INT.APP.

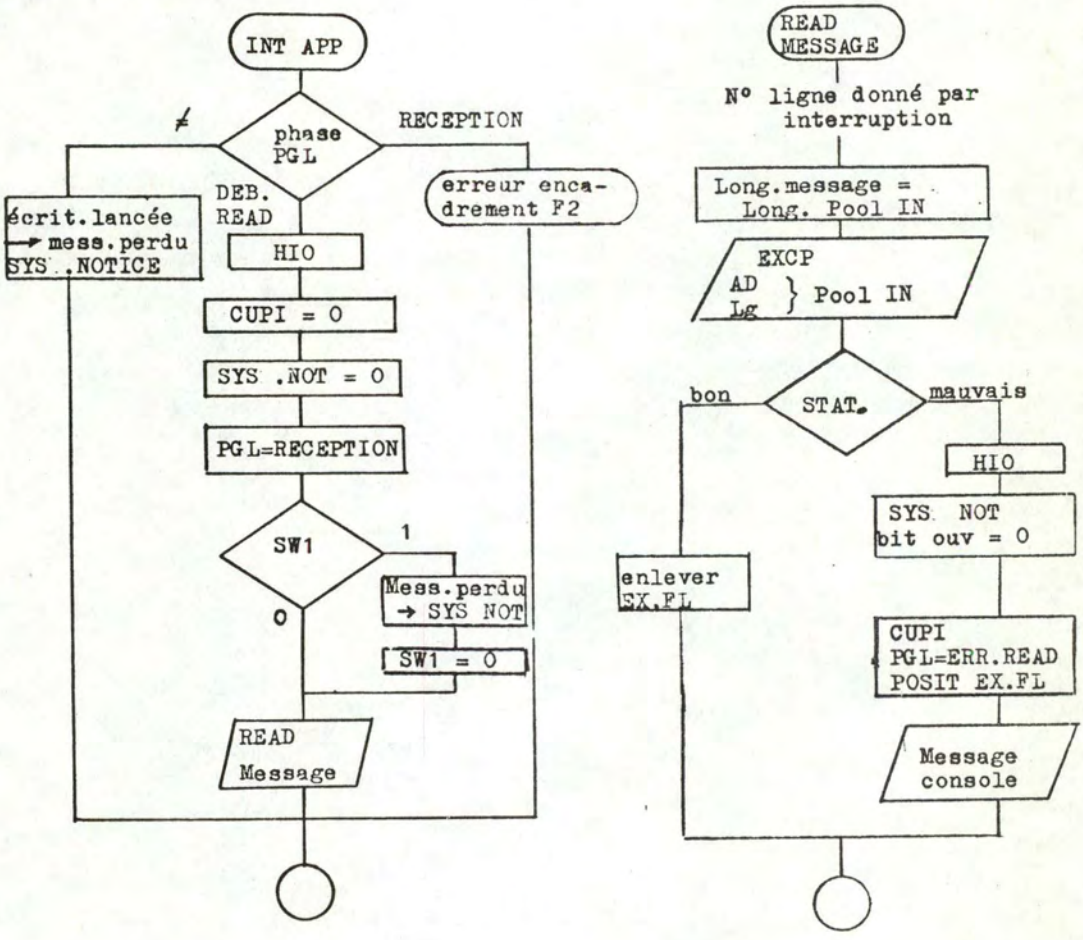
- se produit lors de l'arrivée d'un caractère STX qu'il soit au début ou à la fin du message envoyé par le terminal. Il faudra détecter une erreur possible.

  Détection de l'erreur :

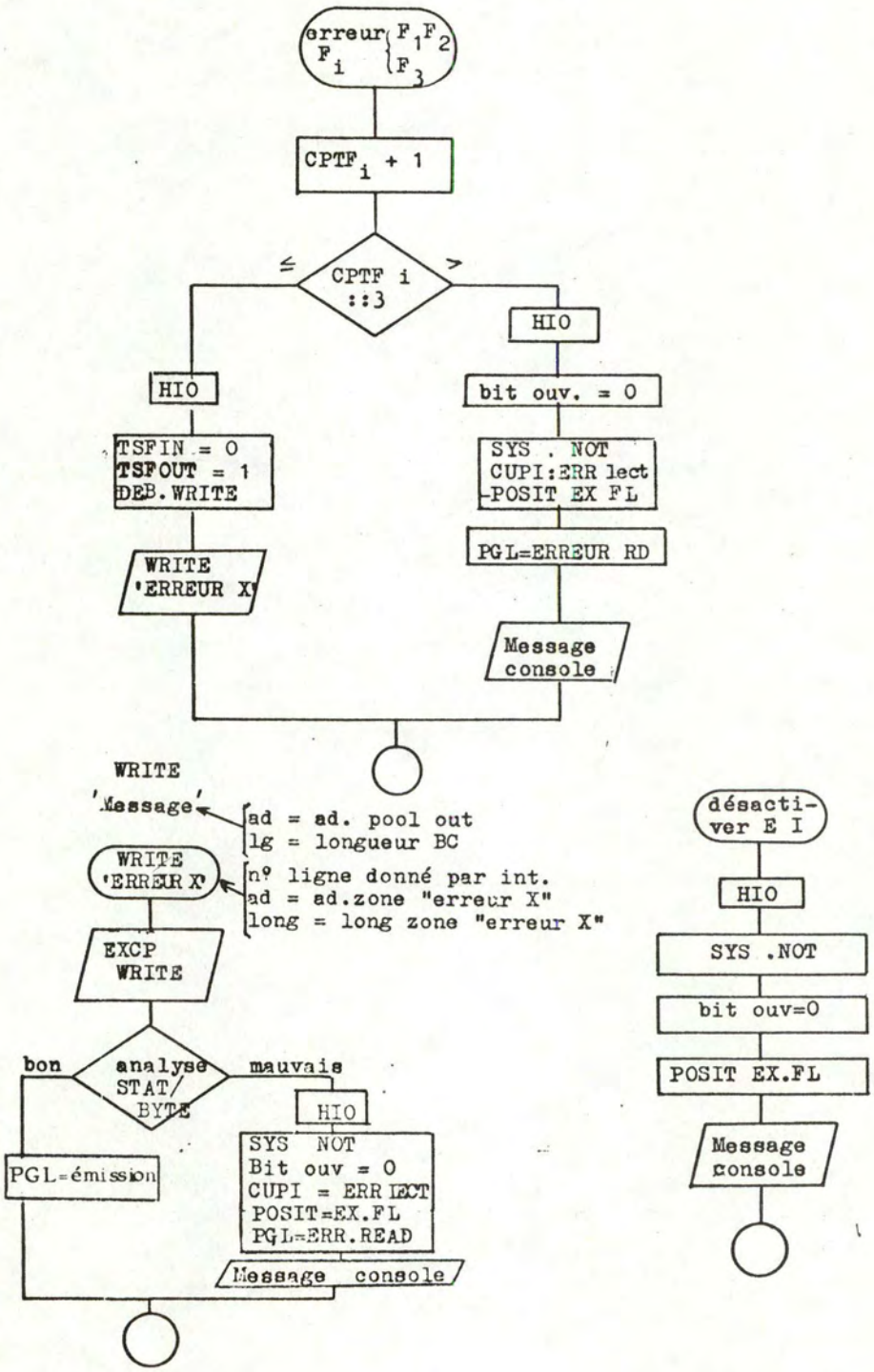
  Avant l'arrivée de tout caractère, la phase de la ligne est DEBREAD, dès l'arrivée du premier STX, elle est modifiée en RECEPTION. Un autre STX, arrivant durant cette phase RECEPTION, sera une erreur.

- routine d'interruption : lors de l'arrivée du premier STX il faut :
  - terminer l'opération de lecture de 1 caractère, qui avait été lancée. En effet, STX ne provoque pas de REQUEST et ne décrémente pas le CCW.
  - changer la phase PGL.
  - tester s'il y a de la place dans le pool pour le message.
  - lancer l'opération de lecture du message et enlever EX.F L.

ORGANIGRAMME.



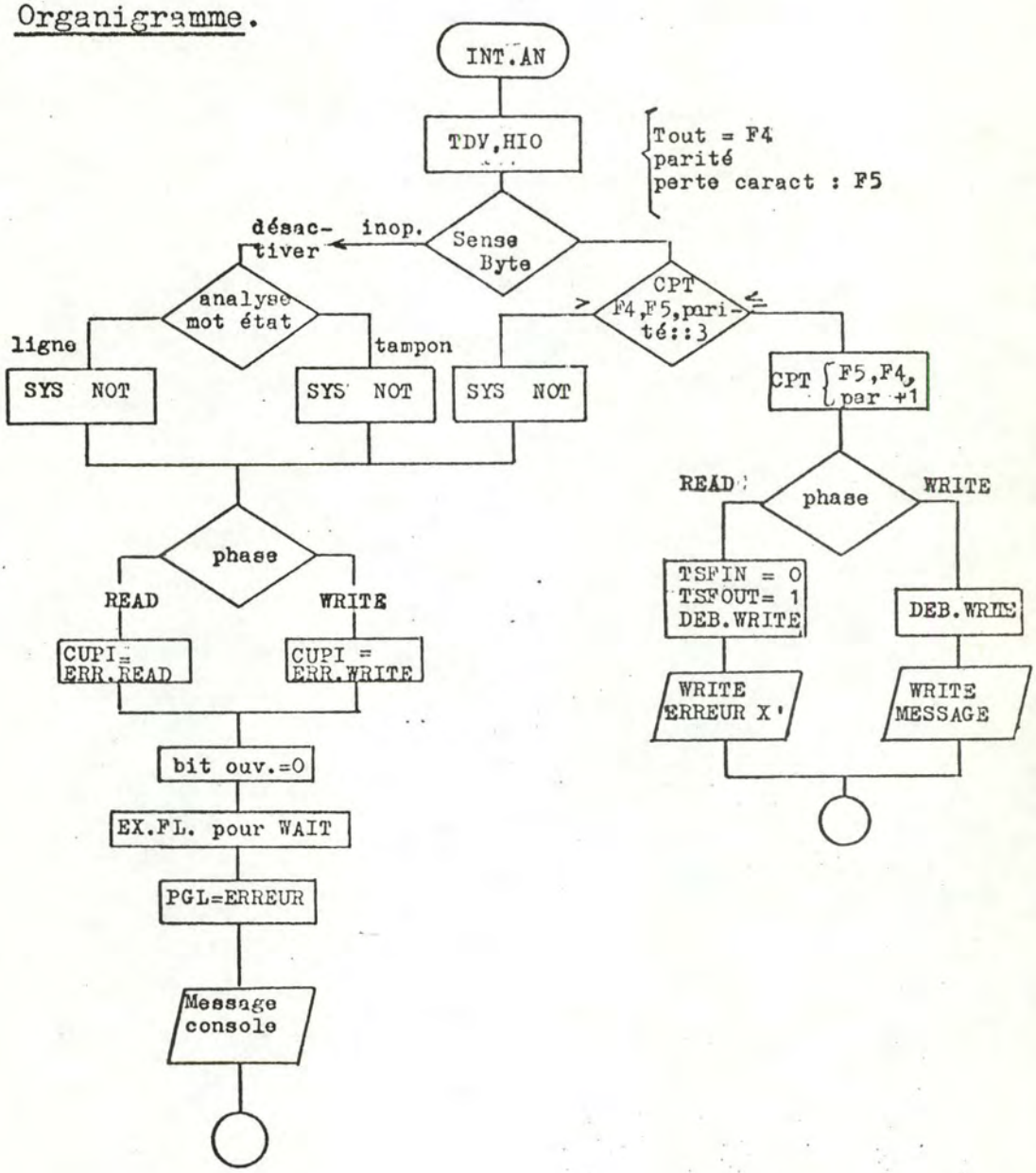
ORGANIGRAMME.



INT. AN

- se produit lorsque le Sense Byte est positionné (inop. ou erreur parité, Tout, caractères perdus).

Organigramme.



Remarque : CUPI sert pour WAIT
SYS NOT pour CGETSYS.

END

- se produit lorsqu' on reçoit ETX du terminal si le tampon est dans l'état READ et en fin CCW si le tampon est dans l'état OCCUPE.

- vérifie que le message a été transmis sans erreur, si il n'y a pas d'erreur : en lecture, il prévient PGL
  en écriture, il relance 1 lecture,

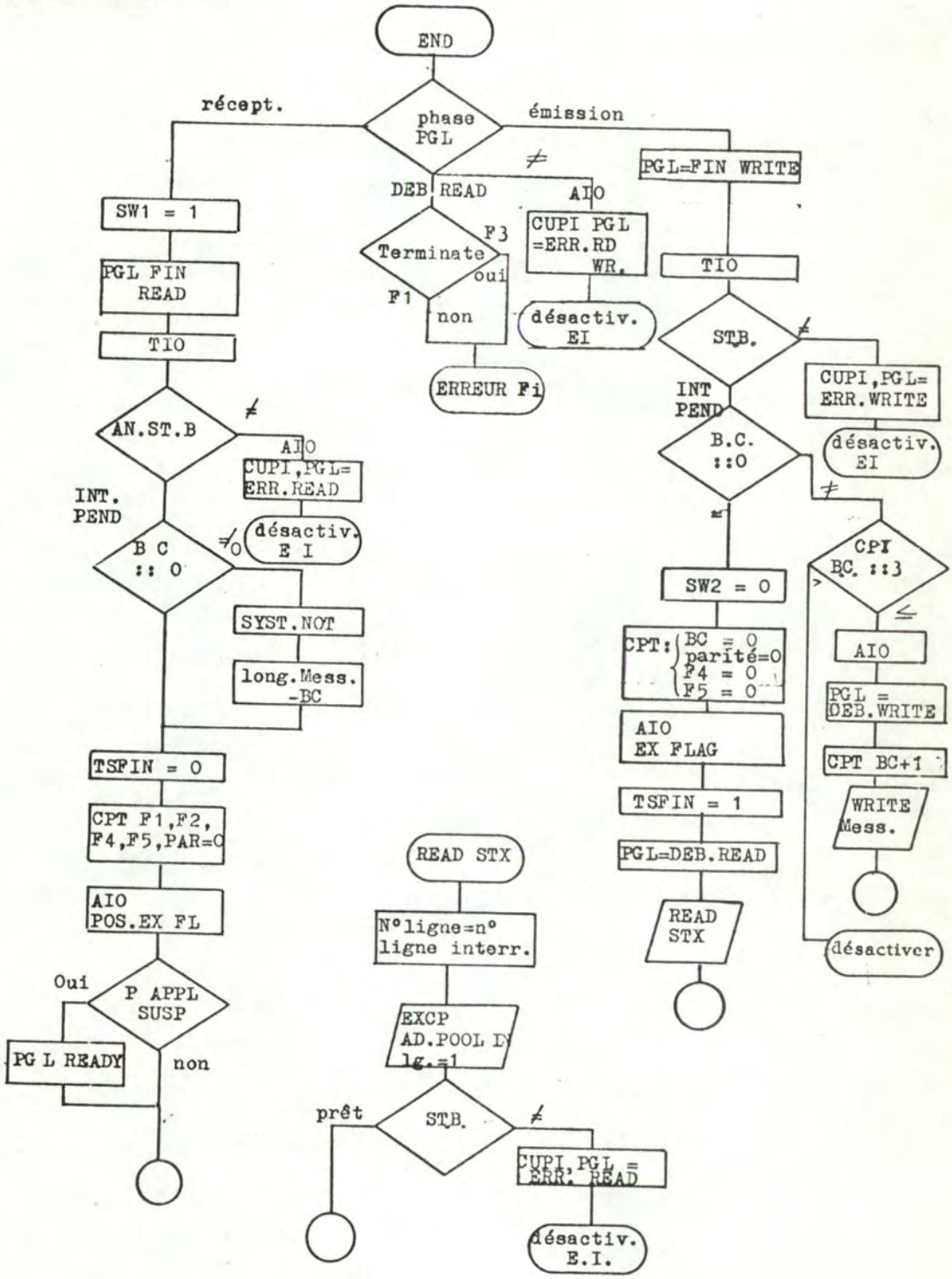
  si il y a 1 erreur, soit il essaie de relancer l'opération, soit il désactive la ligne.

Remarque :

Comme les caractères du message succèdent au STX directement, pour être sûr de ne pas les perdre, on peut lancer la lecture des caractères suivant par microfigeage et ne faire le reste que plus tard, car la routine d'interruption sera masquée lors du déroulement PGL et SVC.

Si il n'y avait pas de place dans le pool, on aura de la même façon une indication message perdu mais ce sera l'ancien au lieu du nouveau.

Organigramme.



III.3.3.3. INPUT/OUTPUT.

Initialisation : EXCP READ ou WRITE-cf. III.3.2.

Description :

- charge les paramètres du CCW indiqués par le CCB;

- lance SIO et teste le STATUS BYTE du terminal.

Si le STATUS BYTE est bon, on lance l'ordre (= code opératoire) au terminal.

Si il y a une erreur, soit on teste le SENSE BYTE, soit on DESACTIVE.

Fin : une interruption END se produit et la routine de fin dépiste les anomalies.

Si une erreur de ligne se produit, une interruption spéciale anomalie se produit et permet le traitement de l'erreur.

Que l'I/O se termine bien ou mal, il faut le signaler pour débloquer un WAIT éventuel, c'est pourquoi l'EX FLAG est positionné.

Dans la zone CUPI :

- on aura l'indication Message normal si l'I/O s'est bien terminée, ou avec une erreur mineure (BC $\neq$ 0 en lecture par ex.) signalée dans SYS NOT.

- on aura l'indication erreur de lecture ou d'écriture si une opération de lecture ou d'écriture s'est terminée de façon anormale.

III.3.3.4. ERREURS.

III.3.3.4.1. CLASSEMENT ET TRAITEMENT.

Classement	Traitement
Erreurs de ligne :	
- permanente :	
- ligne indisponible (1a)	désactiver la ligne
- tampon indisponible (1b)	
- non permanente :	
- transmission	
- parité (2a)	essayer 3 fois et désactiver (M)
- perte de caractères (2b)	
- BC ≠ 0 (2c) en écriture	ignorer
en lecture	
- message perdu (2d)	
- encadrement de message :	
- pas STX au début (2e)	essayer 3 fois et désactiver (M)
- 2 fois STX (2f)	
- pas ETX en fin (time out) (2g)	
Erreurs dues à l'utilisateur :	
- pool incorrect, erreur de séquence (3a)	prévenir l'utilisateur.
- mauvaise longueur (3b)	
- ligne non activée (3c)	
- ligne activée 2 fois (3d)	
- ligne inconnue (3e)	
Erreurs indéterminées (4)	désactiver.

Remarque :

(M) avant d'essayer une nouvelle lecture, il faut prévenir le terminal qu'il y a erreur et qu'il doit retransmettre.

Désactiver :

Dès que l'on désactive, il faut envoyer un message à la console pour prévenir.
Ce message sera composé de l'indication de l'erreur, suivie du numéro de la ligne concernée.

III.3.3.4.2. TOPOGRAPHIE DES ERREURS ET SIGNALEMENTS.

A. Zone NOTIF du P.APPL.

Durant le déroulement d'une macro-instruction, une erreur sera toujours signalée directement dans la"zone NOTIF" du P.APPL (S'il y a une précision à donner, elle le sera dans la zone SYS NOT de la ligne concernée. Cette zone pourra être demandée par le P.APPL. au moyen de la macro-instruction CGET SYS).

	Erreurs	Position dans zone NOTIF
Après CPUT ACTIVATE :		
(1b),(1a),(4) ligne désactivée (précision en SYSNOT)	NA,NB, NC	1
(3d) ligne déjà activée	ND	2
(3e) ligne inconnue	NF	4
Après CPUT DEACTIVATE :		
(3c) ligne non activée	NE	3
(3e) ligne inconnue	NF	4
Après CGET :		
(3a) pool in vide, rien à lire	NJ	8
(3b) mauvaise longueur	NG	5
Après CPUT Message :		
(1b),(1a),(4) ligne désactivée (précision en SYSNOT)	NA,NB NC	1
(3c) ligne non activée	NE	3
(3e) ligne inconnue	NF	4
(3b) mauvaise longueur	NG	5
(3a) message à lire	NM	6

(3a) écriture en cours	NK	7
(3a) lecture en cours	NL	8
Après CPUT ACTION :		
(1b),(1a), (4) ligne désactivée (précision en SYSNOT)	NA,NB NC	1
(3c) ligne non activée	NE	3
(3a) message à lire	NM	6
(3a) écriture en cours	NK	7
(3a) lecture en cours	NL	8
Après CGET SYS et CUPI :		
(3e) ligne inconnue	NF	4

B. Zone SYS NOT :

Durant le déroulement d'une opération, une erreur sera signalée dans la zone SYSNOT de la ligne où se produit l'erreur. En plus, dans la zone CUPI on signalera s'il s'agit d'une erreur de lecture ou d'écriture.

La zone SYSNOT contient aussi des précisions concernant les erreurs survenues lors du déroulement d'une macro-instruction.
Elle est remise à zéro lors du lancement d'une opération.

	Erreur	Position en zone SYS NOT
(1a) ligne indisponible	EA	1
(1b) tampon indisponible	EB	2
(2a) parité	EC	3

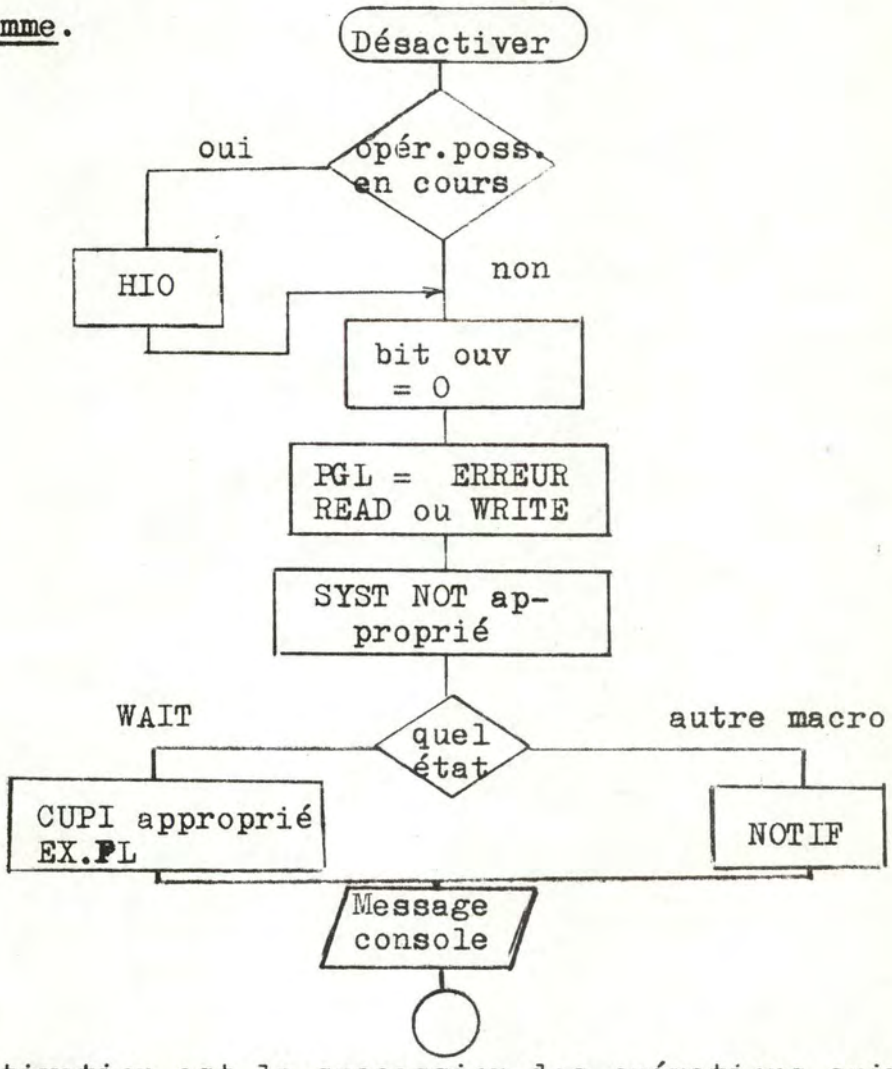
(2b) perte de caractère (réception)	EG	5
(2c) BC $\neq$ 0	EI	4
(2d) message perdu	EH	6
(2e) pas STX au début d'un message	E1	7
	E3	8
(2f) 2 STX dans un message	E2	9
(2g) pas ETX en fin de message	E4	10
(4) erreur indéterminée	E5	11

III.3.3.4.3. TABLEAU GENERAL.

Numéro	Définition	Détection	Signalée par	Positionnement	Traitée par	Action
Erreurs de ligne permanentes.						
(NA)	ligne indisponible en début	CPUT ACTIV	ST.B.2°et ST.B.2[1]	Mess.console	Opération	Puissance sur la ligne
NA	ligne indisponible	EXCP	ST.B2° et ST.B.2[5] + mot état	NOT IF 1 + SYS NOT 1	PGL	Désactiver
NB	tampon indisponible	EXCP	ST.B2° + ST.B.2[5] + mot état	NOTIF 1 + SYST.NOT.2	PGL	Désactiver
Erreurs dues à l'utilisateur.						
ND	ligne déjà activée	CPUT ACTIV	bit d'ouverture	NOTIF 2	Utilisateur	CPUT, SUSP
NE	ligne non activée	CPUT DEACTIV	bit d'ouverture	NOTIF 3	Utilisateur	CPUT ACTIVATE
NE		CPUT ACTION	bit d'ouverture	NOTIF 3	utilisateur	CPUT ACTIV.ou abandonner
		CPUT Message	bit d'ouverture	NOTIF 3	utilisateur	CPUT ACTIV.ou abandonner
NF	ligne inconnue	CPUT ACTIV. CPUT DEACTIV. CPUT Message CGET SYS	table lignes	NOTIF 4	utilisateur	changer numéro ligne ou abandonner
NG	Mauvaise longueur	CGET Message	test longueur	NOTIF 5	utilisateur	ignorer
		CPUT Message	test longueur	NOTIF 5	utilisateur	redonner CPUT avec ZE correcte
NJ	pool IN vide rien à lire	CGET Message	test SW 1	NOTIF 8	utilisateur	faire CPUT ou CPUT ACTION
NK	écriture en cours	CPUT Message	SW 2	NOTIF 7	utilisateur	SUSP
		CPUT ACTION	SW 2	NOTIF 7	utilisateur	SUSP
NL	lecture en cours	CPUT Message	TSFIN + phase	NOTIF 8	utilisateur	SUSP
		CPUT ACTION	TSFIN	NOTIF 8	utilisateur	SUSP
NM	message à lire	CPUT Message	SW 1	NOTIF 6	Utilisateur	CGET
		CPUT ACTION	SW 1	NOTIF 6	Utilisateur	CGET
Erreurs ligne permanentes.						
EA	ligne indisponible	EXCP END,INTAN	ST.B2°,ST.B2[5] et mot d'état	SYS NOT 1	PGL	désactiver la ligne
EB	tampon indisponible	EXCP END,INTAN	ST.B2°,S.B.2[5] et mot d'état	SYS NOT 2	PGL	désactiver la ligne
Erreurs de ligne non permanentes de transmission.						
EC	parité	INT.AN	ST.B2[5],S.B.2[3]	SYS NOT 3		relancer l'opération 3 fois et désactiver
EG	message trop long en réception	INTAN	ST.B.2[5],S.B.2[4]	SYS NOT 5		relancer l'opération 3 fois et désactiver
EH	message perdu	INT.APP.	SW 1	SYS NOT 6	PGL	ignorer
EI	BC ≠ 0	END écriture	test B.C.	SYS NOT 4		relancer l'opération 3 fois et désactiver
		lecture	test B.C.	SYS.NOT 4		ignorer
Erreure de ligne non permanentes d'encadrement de message.						
E1=P1	Pas STX mais ETX	END	phase DEB.READ	SYS NOT 7		relancer l'opération 3 fois et désactiver
E3=P3	Pas STX mais caractère	END	phase DEB.READ et terminate	SYS NOT 8		relancer l'opération 3 fois et désactiver
E2=P2	2 STX dans le message	INT APP	phase Réception	SYS NOT 9	PGL	relancer l'opération 3 fois et désactiver
E4=P4	pas ETX en fin = Time out	INT AN	ST.B=2[5],S.B=2[6]	SYS NOT 10		relancer l'opération 3 fois et désactiver
Erreurs indéterminées.						
E5	Erreur indéterminée	-	-	SYS NOT 11		désactiver

DESACTIVATION.

Organigramme.



Une Désactivation est la succession des opérations suivantes:

- lancer HIO si une opération est peut-être en cours;
- enlever le bit d'ouverture de ligne;
- mettre le PGL en ERREUR, READ ou WRITE suivant qu'on est en lecture ou en écriture;
- préciser l'erreur dans la zone SYSNOT;
- lors du déroulement d'une macro-instruction, renseigner NOTIF 1;
- s'il y a possibilité que le P.APPL soit dans l'état WAIT, positionner CUPI et EX.FLAG pour renseigner et débloquer le P.APPL.
- envoyer un message à la console pour prévenir que la ligne est désactivée.

Opérations de désactivation après une erreur survenue lors des événements de la première colonne.

	HIO	SYST. NOT	NOTIF	CUPI	EX.FL.	PGL ERREUR	BIT ouv.	Message console
CPUT Message	X	X	X			X	X	X
CPUT ACTION	X	X	X			X	X	X
READ Message	X	X		X	X	X	X	X
WRITE Erreur	X	X		X	X	X	X	X
WRITE Message	X	X		X	X	X	X	X
INT.AN.	X	X		X	X	X	X	X
READ STX	X	X		X	X	X	X	X
END	AIO	X		X	X	X	X	X

MACRO INSTRUCTION	NOTIF										
	1	2	3	4	5	6	7	8	9	10	11
CPUT ACTIVATE	ligne désac-tivée	ligne déjà activée		ligne in-connue							
CPUT DEACTIVATE			ligne non activée	ligne in-connue							
CGET					mauvaise longueur			rien à lire Pool IN vide			
CPUT	ligne désac-tivée		ligne non activée	ligne in-connue	mauvaise longueur	message à lire faire CGET	écriture en cours	lecture en cours			
CPUT ACTION	ligne désac-tivée		ligne non activée			message à lire faire CGET	écriture en cours	lecture en cours			
CGET SYS	ligne indisp.	tampon indisp.	parité	BC ≠ 0	message trop long	message perdu	pas STX début F1	pas STX début F3	2 STX	pas ETX en fin	Erreur indét.

Remarque :

Si l'utilisateur veut corriger lui même les erreurs, il le signalera lors du CPUT ACTIVATE.
Alors le PGL ne fera plus d'essais de correction, ni de désactivation de ligne lorsqu'il y a une erreur permanente.

Il faudrait donc que le CPUT ACTIVATE positionne un bit dans la zone ligne B" et les modules du moniteur testeraient ce bit avant d'agir pour savoir dans quel cas ils se trouvent.

En écriture :

- le P.APPL serait obligé de se mettre en WAIT pour savoir si l'opération s'est bien déroulé.

En lecture :

- que l'opération se termine bien ou mal, le PGL préviendrait le P.APPL, en signalant dans la zone CUPI s'il y a eu une erreur ou un message.

IV. DEUXIEME PROCEDURE.

IV.1. DIFFERENCES AVEC LA PREMIERE PROCEDURE.

La Procédure emploie des caractères de contrôle (voir schéma 1), permettant d'échanger plusieurs blocs composant un message tout en vérifiant la manière dont se déroule le dialogue et en permettant la correction des erreurs.
Elle sera gérée par un module spécial qui enverra et testera les caractères de contrôle.

Le Pool comportera plusieurs buffers, chacun pouvant contenir un bloc du message.

IV.2. MODIFICATIONS APPORTEES A LA PREMIERE PROCEDURE.

IV.2.1. PRESENTATION DU PROBLEME.

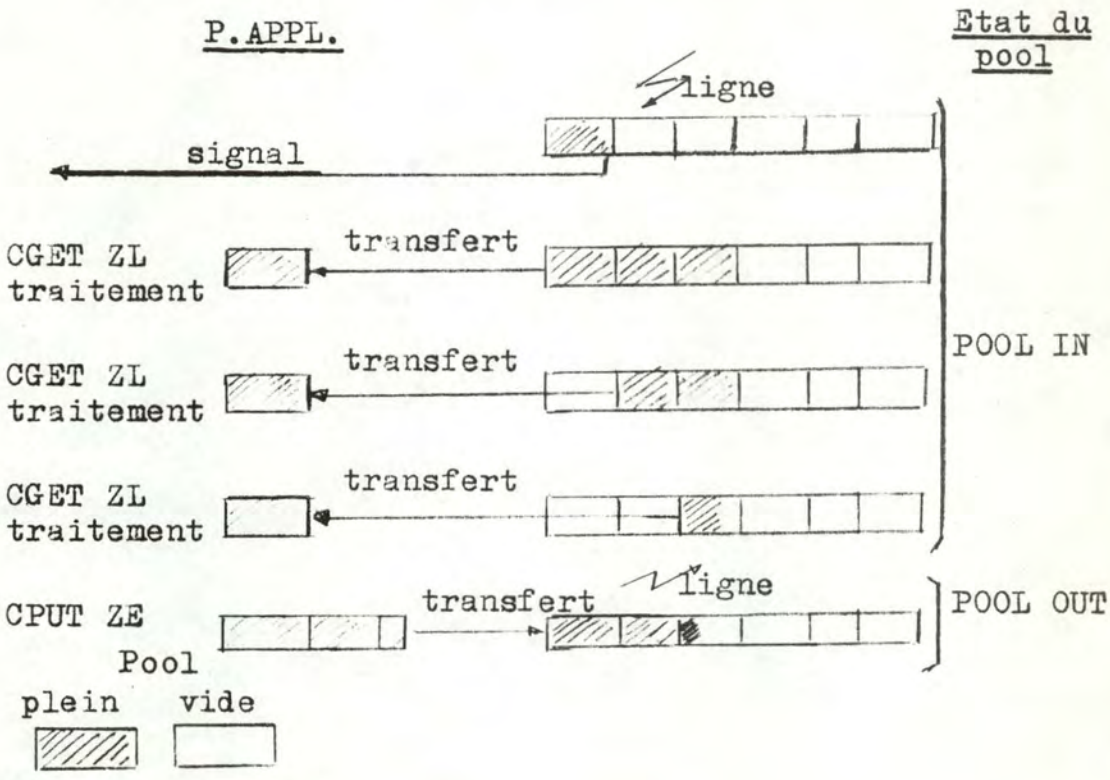
IV.2.1.1. DEROULEMENT GENERAL.

Dès que le premier bloc d'un message est arrivé depuis un terminal, le PGL est prévenu et peut débloquer le P.APPL.
Ensuite, chaque CGET du P.APPL effectue le transfert d'un bloc dans une zone de lecture du P.APPL en positionnant des renseignements dans Z.NOTIF.

Lorsque tous les blocs du message ont été transmis au P.APPL, celui-ci est prévenu dans Z.NOTIF et il peut envoyer une réponse par CPUT.

Si le P.APPL va trop vite, le CGET provoque un WAIT qui attendra l'arrivée du bloc suivant en provenance du terminal.
La réponse envoyée par CPUT peut remplir plusieurs buffers,chaque bloc envoyé videra un buffer.

IV.2.1.2. SCHEMA DE CIRCULATION DES MESSAGES.



Remarque :

On pourrait faire plusieurs CPUT successifs, dès après le traitement d'un bloc envoyé par le terminal.

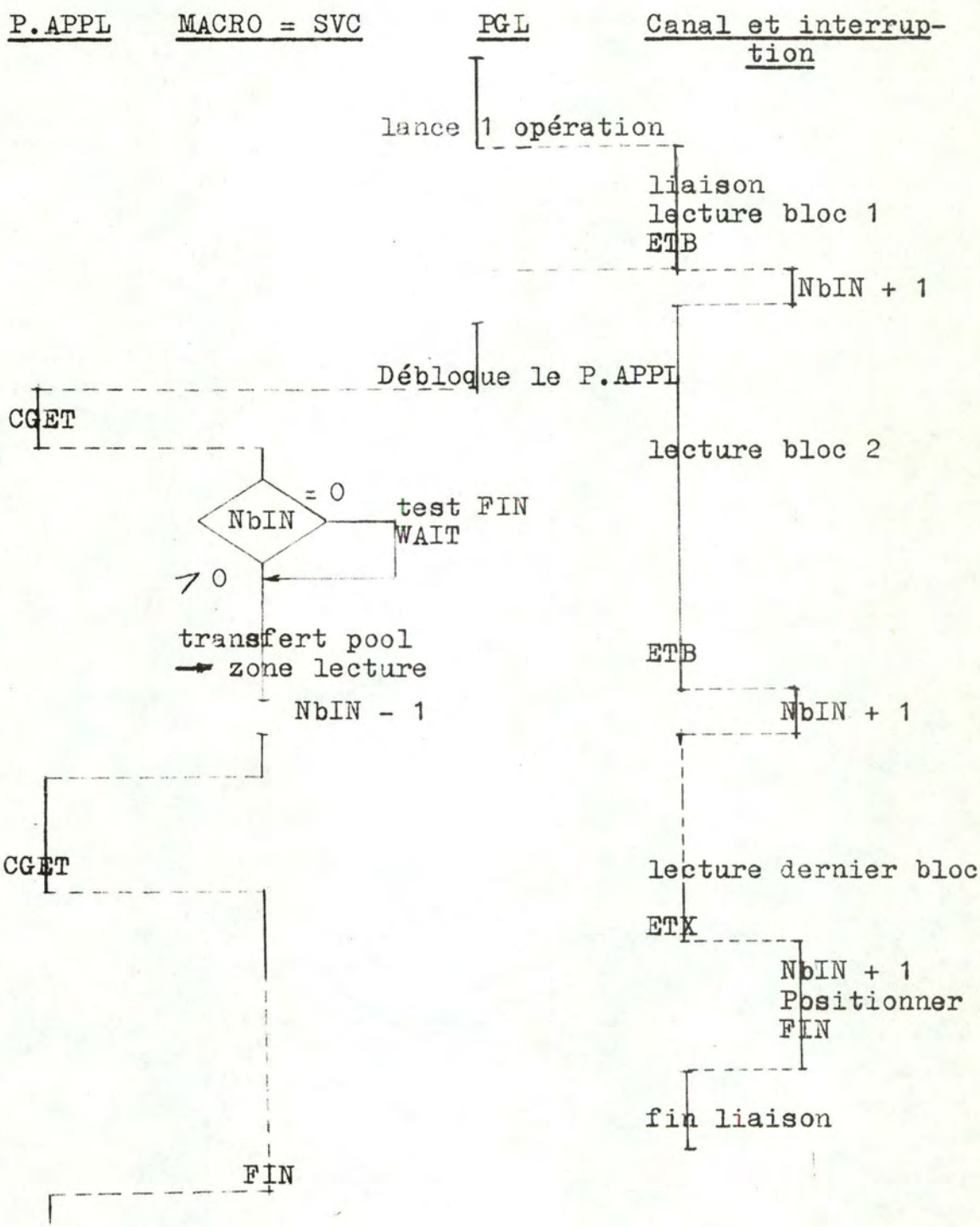
Chaque CPUT remplirait un ou plusieurs buffers du pool et le CPUT suivant recommencerait à remplir le premier buffer vide du pool.
Il faudrait alors dans chaque buffer indiquer la longueur remplie pour envoyer sur la ligne un bloc de longueur voulue.

Le CPUT essayera d'envoyer le message. Si la ligne est occupée il positionnera un bit qui sera testé lors de la libération de la ligne. L'opération d'écriture sera alors initialisée, en enlevant ce bit.

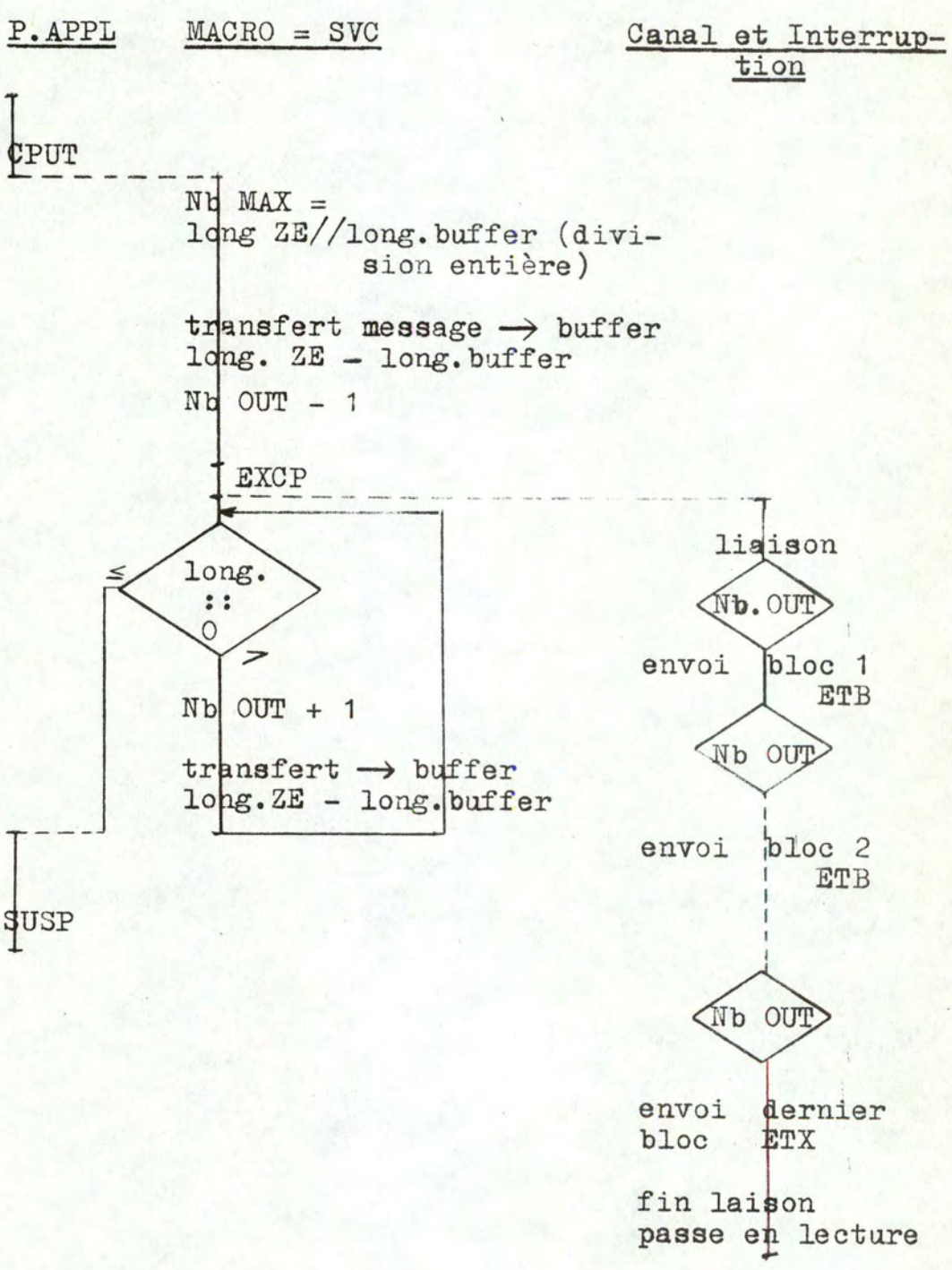
IV.2.2. JONCTION AVEC LE PROGRAMME D'APPLICATION.

IV.2.2.1. OPERATIONS DE LECTURE/ECRITURE.

Opération lecture.



Opération écriture.



IV.2.2.2. DISTRIBUTEUR DE TACHES.

Niveau global :

La partition télétraitement reprend la main dès qu'un bloc est entré dans un pool, en provenance d'un terminal, et pas seulement lorsque tout un message est reçu.

Niveau télétraitement :

Solution 1) : débloquer WAIT en positionnant EX.FL lors de chaque ETB reçu.

Solution 3) : chaque fois qu'un bloc est arrivé dans un buffer, le nombre de blocs "Nb" s'augmente de 1. Si ce Nb est $\neq 0$, le PGL sait qu'il doit débloquer le P.APPL.

Résumé :

- bit PGL :
  - ready : - par SUSP du P.APPL.
    - par interruption en fin ETX ou ETB reçu si P.APPL est en SUSP.
  - non ready : - lorsqu'il rend la main au DT.
- bit P.APPL :
  - ready : - lorsque DT lève le WAIT;
    - lorsque PGL lève le SUSP.
  - non ready: - lorsque P.APPL attend 1 message (SUSP).
  - non ready WAIT :
    - lorsque P.APPL attend une fin d'I/O.

IV. 2.2.3. ZONES DE COMMUNICATION.

IV. 2.2.3.1. "Zone B" ACCESSIBLE AUX MACRO-INSTRUCTIONS.

Zone B 1 :

- AD CUPI et NOTIF de P.APPL.
- N° ligne courant, table codes.

Zone B 2 : par ligne :

- AD POOL IN	AD POOL OUT
AD1POOL IN	AD1POOL OUT
AD2POOL IN	AD2POOL OUT
Long.max.message	Long.max.de bloc = long. de buffer OUT
NbIN	NbMax.
TSFIN	NbOUT
SYS NOT	TSF OUT
Codage	
CUPI.	

Zone B 3 :

- identique à celle de la première procédure.

Explications :

Nb IN, Nb OUT } (cf. IV.2.3.1.2)
TSF OUT }

N° ligne courant, table codes, AD.NOTIF et CUPI du P.APPL, AD.POOL IN/OUT, SYS NOT, TSF IN, Codage } cfr. procéd. 1

AD1 POOL IN : pointeur vers le dernier bloc pris par CGET.

AD2 POOL IN : pointeur vers le dernier bloc envoyé par le terminal.

AD1 POOL OUT: pointeur vers le dernier bloc envoyé par CPUT.

AD2 POOL OUT: pointeur vers le dernier bloc envoyé sur la ligne vers le terminal.

Long.max.Message : sert à lancer la lecture d'un message, c'est la longueur maximum que le terminal pourra envoyer.

Long.buffer out : servira à découper le message envoyé par CPUT en blocs de longueur fixe égale à cette longueur de buffer.

IV.2.2.3.2. POOL.

Il y a toujous deux pools par ligne :

- IN pour la réception,
- OUT pour l'émission.

Mais ce sont des pools de n buffers de longueur fixe, remplis et vidés de façon circulaire.
Il y a dans chaque buffer un bit de présence, et éventuellement une longueur :

- buffer IN : la longueur du bloc s'y trouve toujours;
- buffer OUT : éventuellement, la longueur du bloc s'y trouvera.

IV.2.2.3.2.1. LONGUEUR.

A. Lors de la réception, chaque bloc est placé dans un nouveau buffer, que le précédent soit rempli où non. Il faut donc indiquer la longueur du bloc dans le buffer.

B. Lors de l'émission : première possibilité,

- un seul CPUT découpe une zone écriture en blocs de longueur fixe égale à celle du buffer, et seul le dernier ne remplira peut-être pas complètement le buffer, sa longueur se trouvera en début de pool.

Lors de l'émission : deuxième possibilité.

- chaque CPUT découpera une zone d'écriture en blocs. Un nouveau CPUT commencera dans un buffer différent, que le précédent soit plein ou non. Il faudra donc indiquer les longueurs de bloc dans chaque buffer et plus seulement en début de pool.

IV.2.2.3.2.2. BIT DE PRESENCE DANS CHAQUE BUFFER.

Pool [IN] :

- bit = 1 - si le buffer est rempli par le périphérique et est prêt à être pris par CGET,
  - positionné lors de la réception des caractères ETB et ETX.
- bit = 0 - si le buffer n'est pas encore prêt à être pris par CGET,
  - positionné lors de CGET, après le transfert du message hors du pool.

Pool [OUT] :

- bit = 1 - si le buffer est rempli par l'utilisateur et est prêt à être envoyé sur la ligne vers le terminal,
  - positionné par CPUT, après le transfert dans le pool.
- bit = 0 - si le buffer est vidé par le périphérique,
  - positionné lors de l'envoi des caractères ETB et ETX.

Remarque : Les caractères de contrôle seront envoyés par le module spécial.

IV.2.2.4. INSTRUCTIONS DE COMMUNICATION.

SUSP, CUPI, CGETSYS : idem première procédure.

WAIT : - positionné lorsque le P.APPL a lancé une I/O, attend qu'elle se réalise pour continuer.
- en réception : levé dès qu'un bloc est arrivé.
- en émission : levé lorsque tout le message est envoyé.

Vérification de la validité :

TSFIN - positionné lors du lancement d'une lecture,
- enlevé lors d'une fin de lecture (ETX reçu).

TSF OUT - positionné lors du lancement d'une écriture,
- enlevé lors d'une fin d'écriture.

CGET MESSAGE

Paramètres.

Utilisateur

type
ZL
longueur ZL.

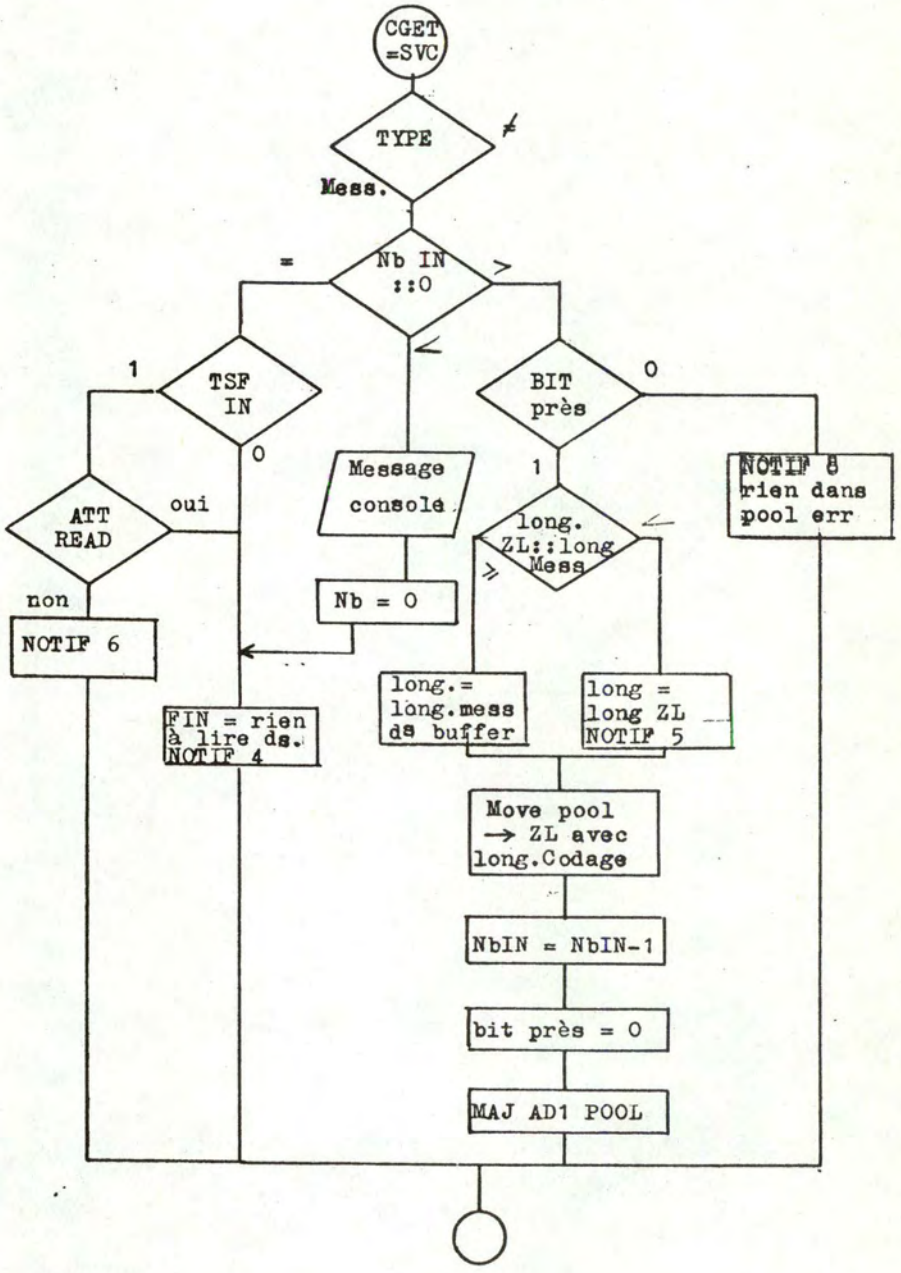
Système :

N°ligne courant	table codes	NOTIF P.APPL.

par ligne

AD.POOL IN	AD1 POOL	CODAGE
Nb IN	TSF IN	
phases du PGL		

Organigramme.



Remarque : MAJ AD1 du
Pool modulo n si il y a n buffers.

CPUT MESSAGE

Paramètres.

Utilisateurs
- type
- ZE
- longueur ZE
- n° ligne (éventuellement).

Système.

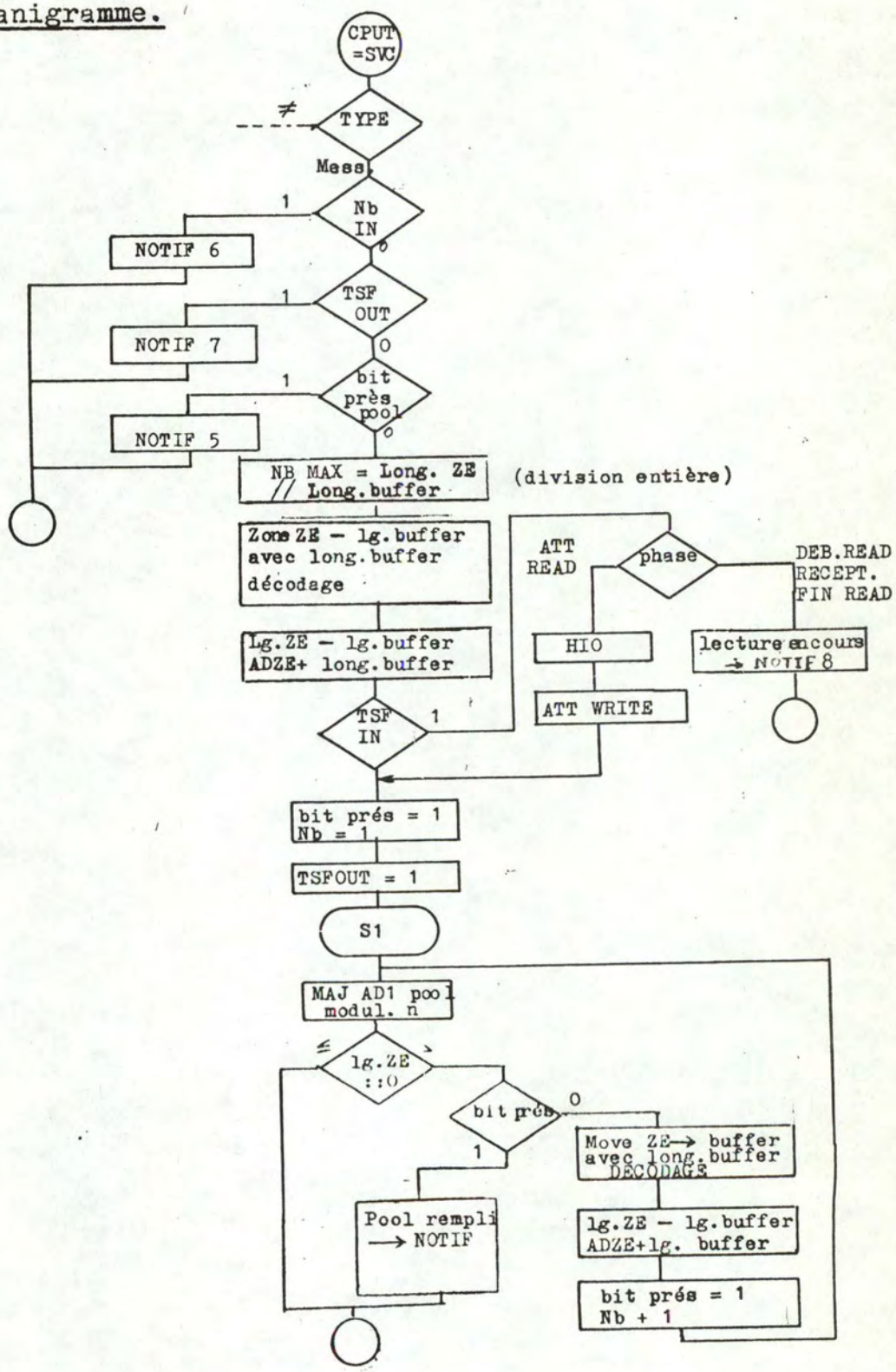
n°ligne courant	table codes	AD.NOTIF EAPPL.

par ligne

AD POOL OUT	AD1 POOL OUT	LG.BUFFER	CODAGE

TSFIN	TSFOUT	NB MAX	NB OUT

Organigramme.



CPUT ACTION

a. repositionne une opération de lecture sans avoir donné de réponse à une lecture préalable.

Paramètres.

Utilisateurs

Adr. A
type

Système

1 fois

n° ligne courant	NOTIF P.APPL.

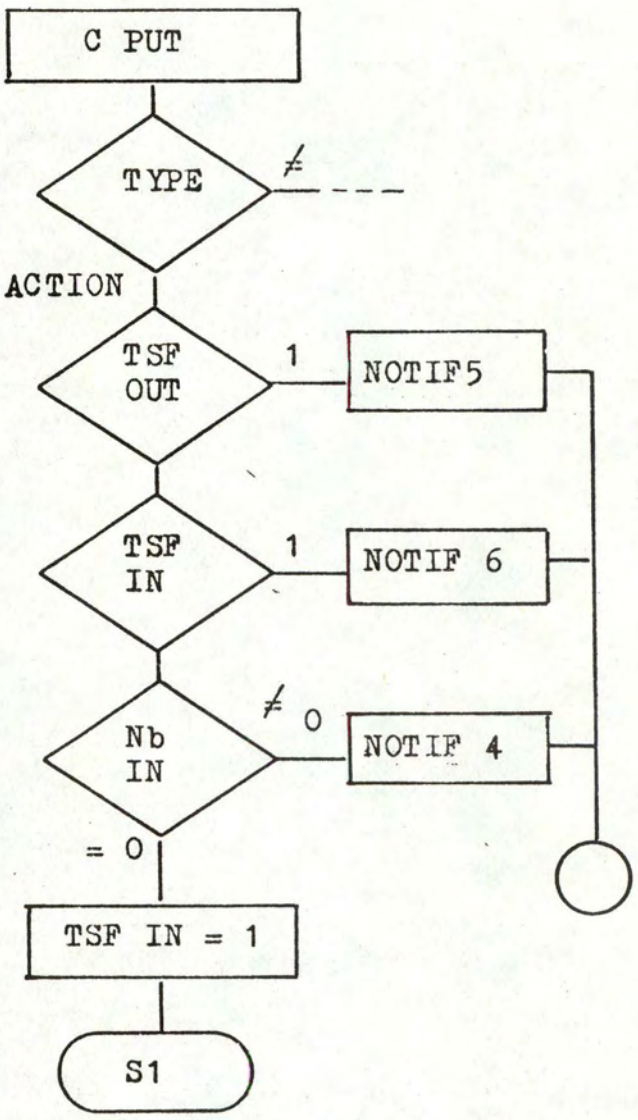
par ligne

TSFIN	TSFOUT	NbIN

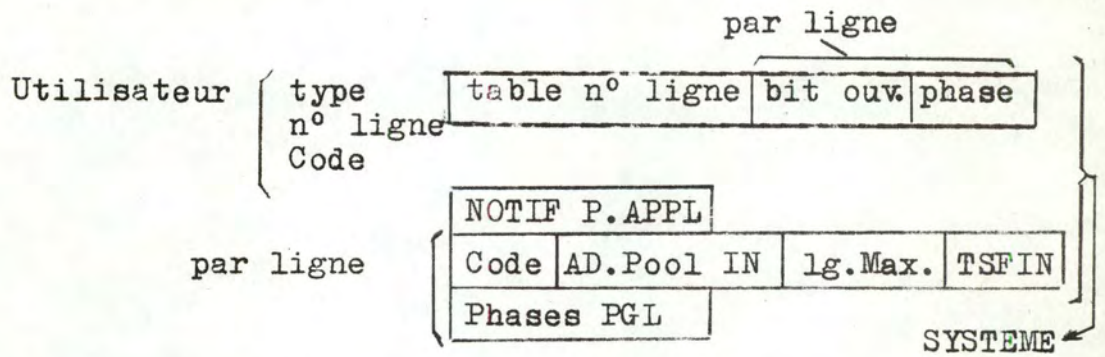
par ligne

AD.POOLIN,	AD2 POOLIN
LONG. MAX,	Z NOTIF.

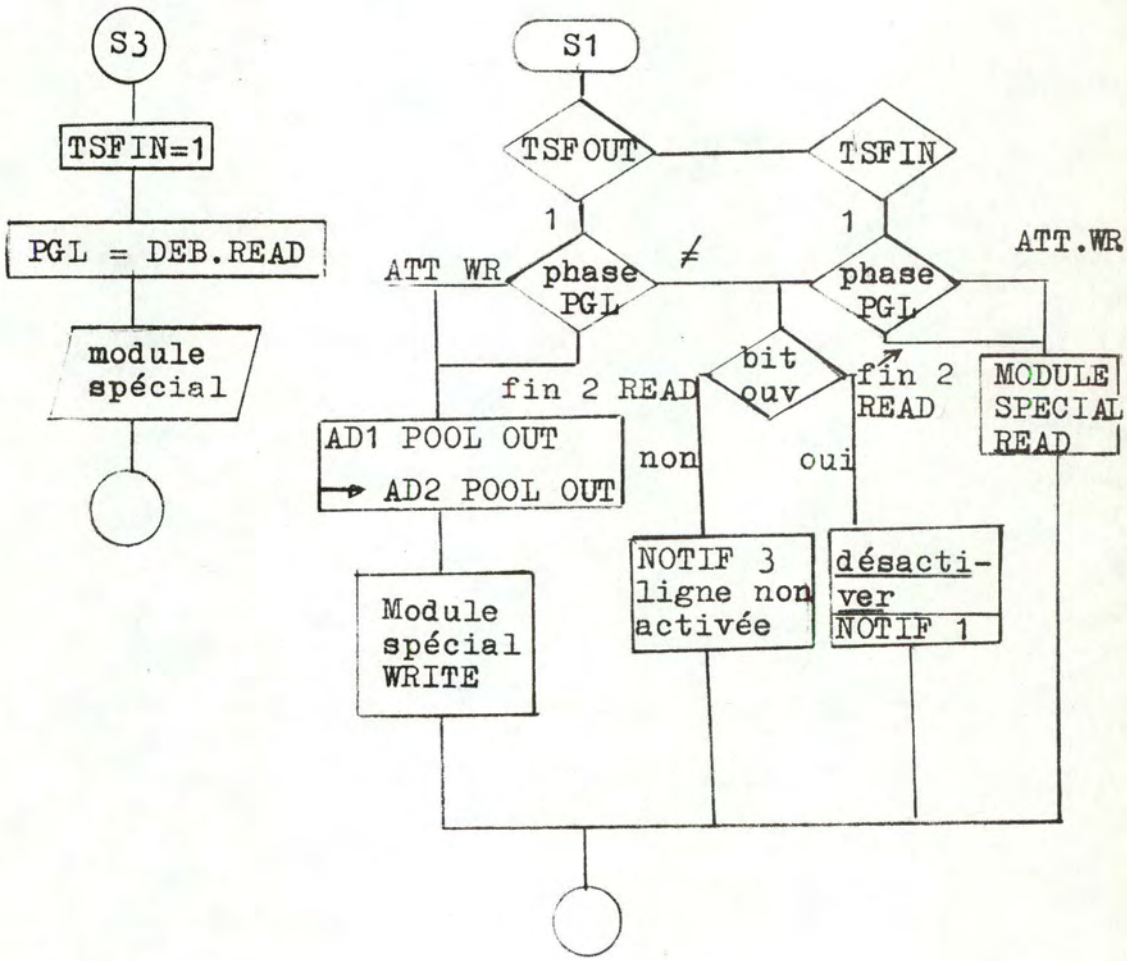
Organigramme.



CPUT ACTIVATE



Organigramme : idem première procédure jusque S3.



IV.2.3. MONITEUR DE TELETRAITEMENT.

IV.2.3.1. INFORMATION.

IV.2.3.1.1. ZONES D'INFORMATION.

Il y a différentes phases possibles pour chaque ligne (cf. emploi IV.2.3.1.2).

ATT.READ	si on veut lire	par l'ordinateur
DEB.READ	si ENQ reçu	
RECEPTION	si STX reçu	
FIN 1 READ	si ETB reçu	
FIN 2 READ	si ETX reçu	
ATT.WRITE	si on peut écrire	par l'ordinateur
DEB.WRITE	si ENQ envoyé	
EMISSION	si STX envoyé	
FIN EMISSION	si ETB envoyé	
FIN2 EMISSION	si ETX envoyé	

IV.2.3.1.2. EMPLOI DE CES ZONES (tableau 8).

TSF IN et TSF OUTserviront à déterminer la possibilité pour d'autres macro-instructions d'être employées. En effet, lorsque TSF IN est positionné, une lecture se déroule, et lorsque TSF OUT est positionné une écriture se déroule.

Pour contrôler la transmission de tous les blocs d'un message :

en réception :

- lors du Début de lecture : TSF IN = 1
- lors de la fin de message (ETX) : TSF IN = 0
- lors de chaque arrivée de bloc (ETB et ETX), on fait NbIN + 1.
- lors de chaque CGET :
  - si NbIN = 0 mais TSFIN=1, il faut attendre une nouvelle arrivée de bloc.
  - si NbIN = 0 et TSFIN=0, c'est terminé.

- si NbIN > 0, on peut prendre un bloc et on fait Nb IN - 1.

en émission :

- lors du début d'une émission (CPUT) on fait TSFOUT = 1 et on détermine le nombre maximum de blocs à envoyer (Nb Max.),

- lors de la transmission d'un bloc dans un buffer , on fait Nb OUT + 1,

- lors de l'envoi d'un bloc au terminal :

  - si Nb OUT = 0 mais Nb Max ≠ 0, il faut attendre le transfert d'un nouveau bloc dans le pool.

  - si Nb OUT = 0 et Nb Max = 0, on fait TSFOUT = 0,

  - si Nb OUT > 0, on envoie un bloc et on fait Nb OUT - 1, Nb Max - 1.

A tout moment, NbIN indique le nombre de blocs restant à traiter par le P.APPL, et NbOUT indique le nombre de blocs restant à envoyer au terminal.

Tableau 8.

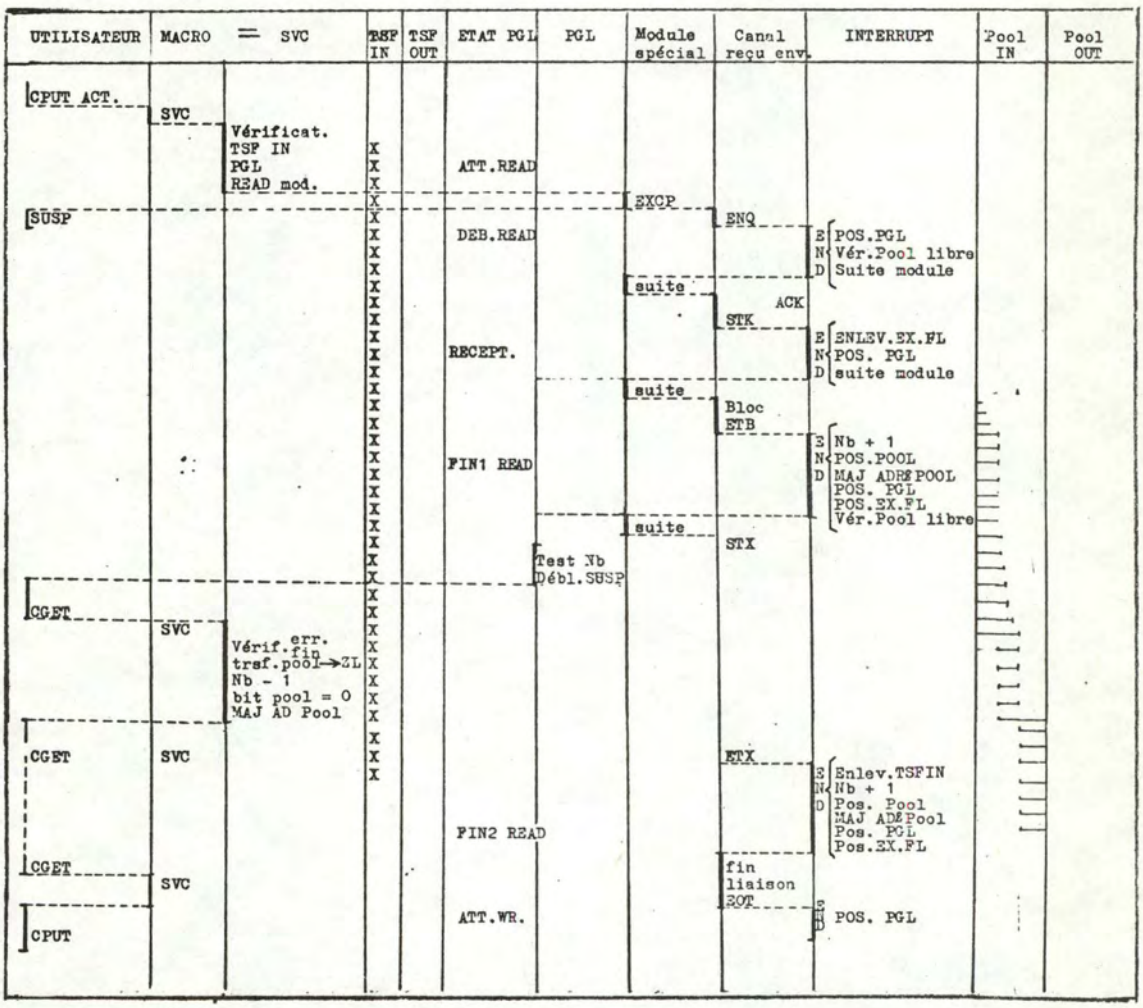
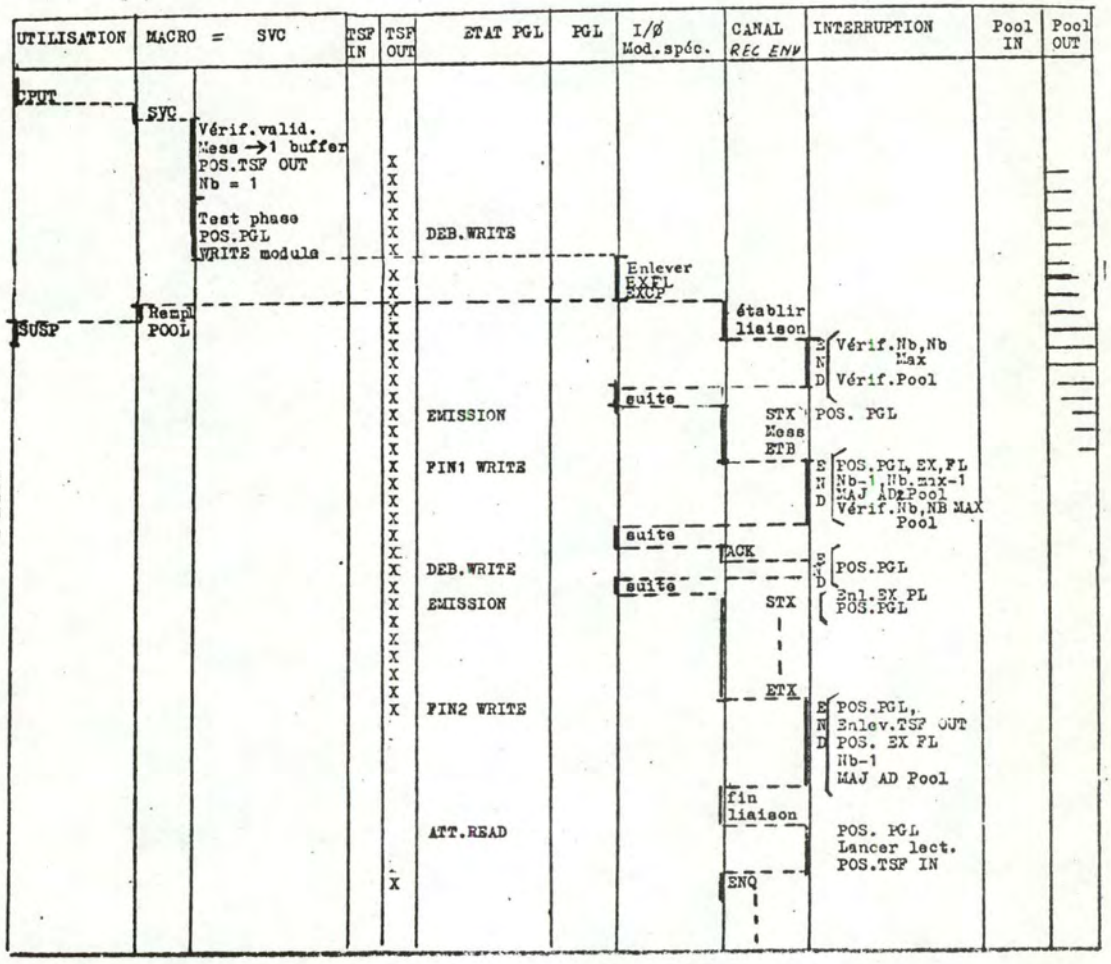

UTILISATEUR
MACRO = SVC
TSF IN
TSF OUT
ETAT PGL
PGL
Module spécial
Canal reçu env.
INTERRUPT
Pool IN
Pool OUT
CPUT ACT.
SVC
Vérificat.
TSF IN
PGL
READ mod.
EXCP
SUSP
ENQ
DEB.READ
POS.PGL
Vér.Pool libre
Suite module
suite
ACK
STK
RECEPT.
ENLEV.EX.FL
POS. PGL
suite module
Bloc
ETB
Nb + 1
POS.POOL
FIN1 READ
MAJ ADR POOL
POS. PGL
POS.EX.FL
Vér.Pool libre
STX
Test Nb
Débl.SUSP
CGET
Vérif. err.
Vérif. fin trsf.pool → ZL
Nb - 1
bit pool = 0
MAJ AD Pool
ETX
Enlev.TSFIN
Nb + 1
Pos. Pool
MAJ ADR Pool
Pos. PGL
Pos.EX.FL
FIN2 READ
fin liaison
EOT
ATT.WR.
POS. PGL

TABLEAU 8.


UTILISATION
MACRO = SVC
TSF IN
TSF OUT
ETAT PGL
PGL
I/Ø Mod.spéc.
CANAL REC ENV
INTERRUPTION
Pool IN
Pool OUT
CPUT
SVC
Vérif.valid.
Mess →1 buffer
POS.TSF OUT
Nb = 1
Test phase
POS.PGL
WRITE module
DEB.WRITE
Enlever
EXFL
SKOP
établir liaison
SUSP
Rempl POOL
Vérif.Nb,Nb Max
Vérif.Pool
suite
EMISSION
STX
Mess
ETB
POS. PGL
FIN1 WRITE
POS.PGL, EX,FL
Nb-1, Nb.max-1
MAJ AD Pool
Vérif.Nb,NB MAX Pool
suite
ACK
POS.PGL
DEB.WRITE
suite
EMISSION
STX
Enl.EX PL
POS.PGL
ETX
FIN2 WRITE
POS.PGL,
Enlev.TSF OUT
POS. EX FL
Nb-1
MAJ AD Pool
fin liaison
ATT.READ
POS. PGL
Lancer lect.
POS.TSF IN
ENQ

IV.2.3.2. MODULE DU MONITEUR : PROGRAMME DE GESTION DE LIGNES.

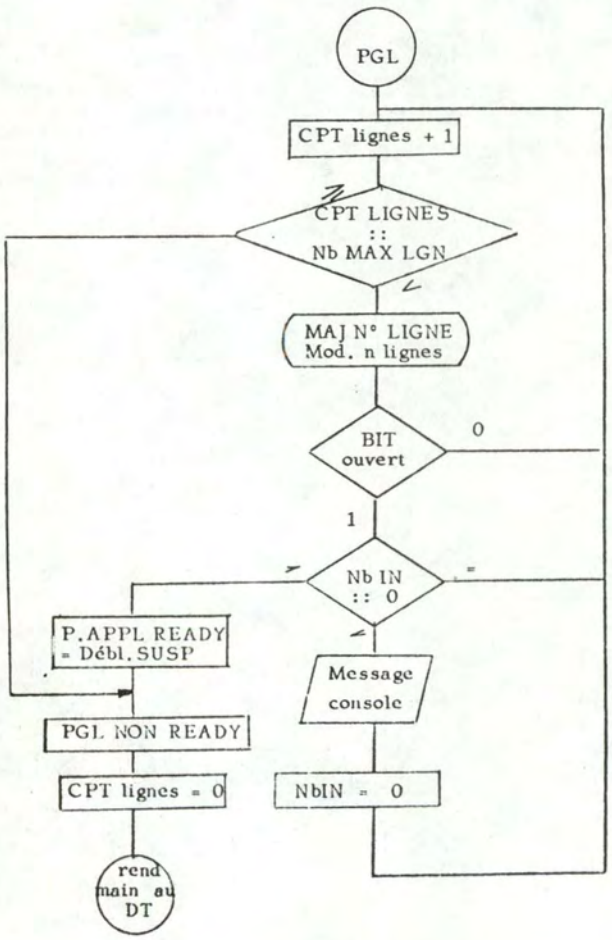
IV.2.3.2.1. PRISE DE CONTROLE DU PGL.

Idem première procédure.

IV.2.3.2.2. ROLE DU PGL.

Le PGL teste d'abord si une ligne est ouverte et ensuite si un bloc est arrivé, c'est-à-dire Nb.IN > 0.

IV.2.3.2.3. ORGANIGRAMME.



IV.2.3.3. MODULE SPECIAL.

IV.2.3.3.1. EXPLICATIONS.

- STX ne provoque pas interruption d'appel mais un simple request.
  Lorsque CCW lancé par module spécial aura reçu un caractère, il sera terminé, et on aura une interruption qui analysera le caractère reçu.

- On aura toujours :

  - soit ETB, ETX provoquant l'interruption END (qui testera - B.C. ≠ 0
    - message trop court).
  - soit CCW terminé provoquant l'interruption END.
  - soit INT.AN (Tout, ligne indisponible, tampon indisponible, parité).

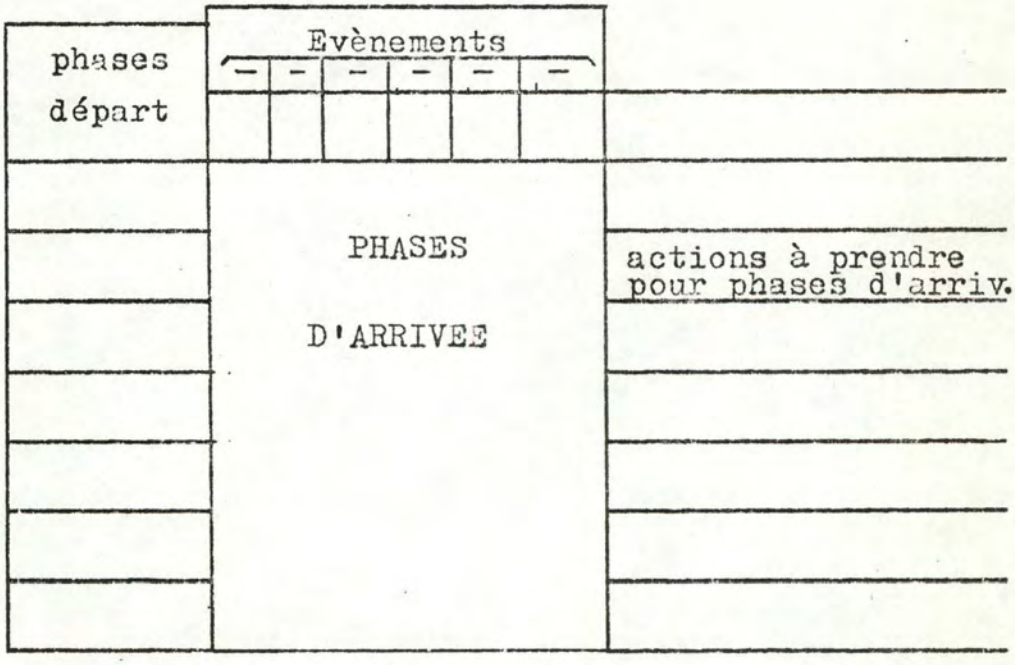
  Ces interruptions passeront au module spécial qui analysera :

  - où il en est (phase),
  - le caractère ou l'évènement reçu (1ère ligne),
  - et décidera l'action à prendre (phase de colonne adéquate).

- En plus des actions indiquées dans le module spécial, chacune de ces interruptions devra :

  - analyser ST, Sense Byte pour connaître la cause d'interruption (employer l'ordre AIO pour terminer l'opération).
  - Mettre PGL ready si c'est fin de bloc (ETB, ETX) et si P.APPL est en SUSP.

- Tableau du module spécial - Explicatinns.



Evènement provoquant une réaction dans le module spécial.

Réception :

BR.INC = BRanchement INconditionnel en fin CCW.

REC.ENQ
REC.EOT
REC.CAR.IN
REC.TTD
: en fin CCW, test du caractère reçu

$\overline{STX}$
STX
: en fin CCW, test du caractère reçu

ETB.CC
ETX.CC
: lors interruption END, test du dernier caractère reçu

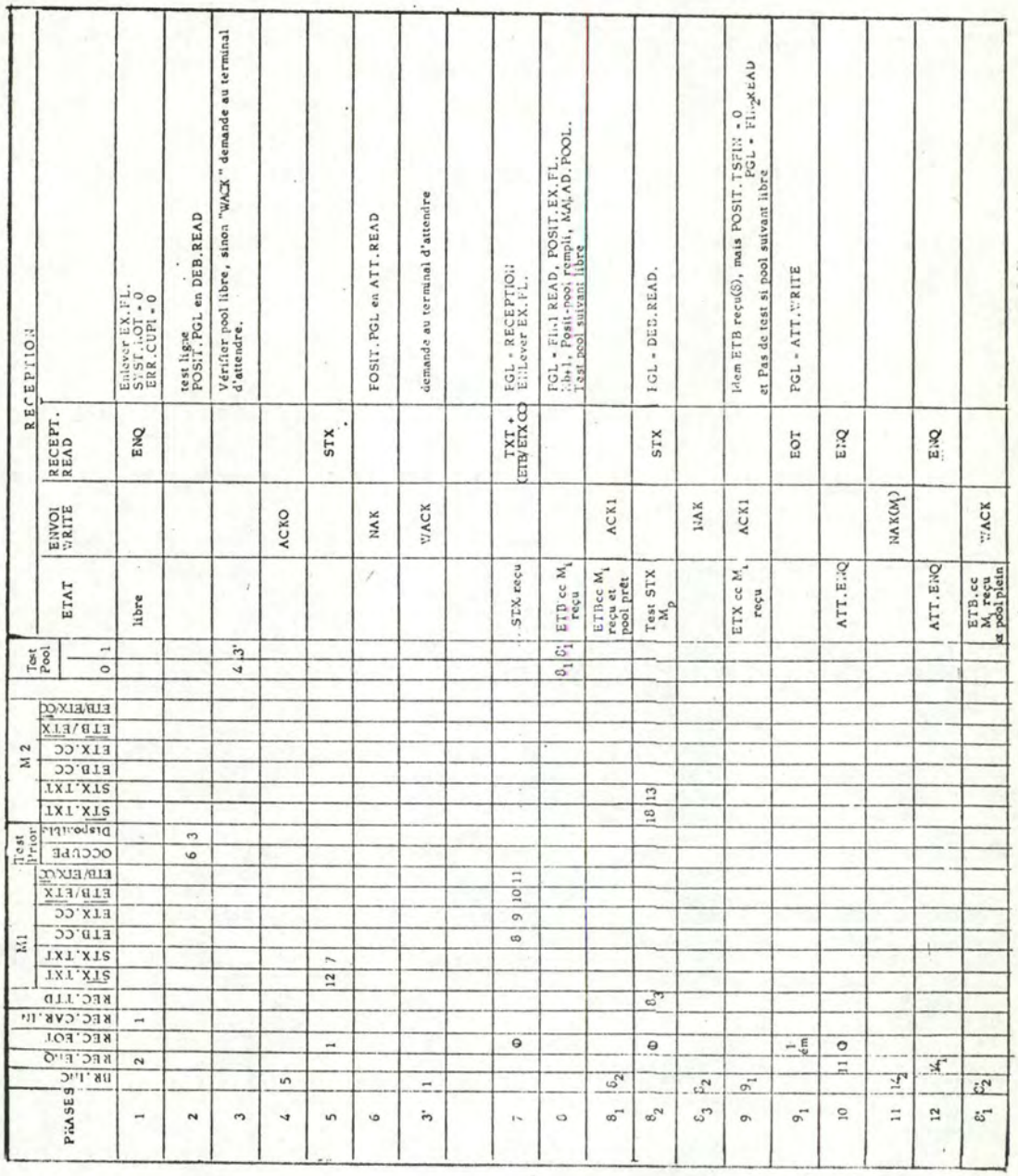
$\overline{ETB/ETX}$: en fin CCW, test du dernier caractère reçu

ETB/ETX.$\overline{CC}$	int. AN
Test prior	= test disponibilité
Test pool	= test du bit de présence du pool.

Emission :

BR.INC	en fin CCW
CPUT	lors du SVC
CPUT.ACT	lors du SVC
EM.ETB	= émission ETB – positionné lors du test du Nombre de messages restant à
EM.ETX	= émission ETX – envoyer (Nb OUT)
$\overline{ETB}.\overline{ETX}$	Time out.
REC.ENQ REC.ACK0 REC.ACK1 REC.NAK REC.WACK REC.C.INV.	en fin CCW, test réponse reçue
ENQ $\overline{F}$ F BLOC $\overline{F}$ F	test de compteurs : - soit le nombre d'essais voulus n'est pas encore réalisé, $\overline{F}$ - soit le nombre d'essais voulus a été réalisé, F
PRIOR L D	si l'ordinateur est relié à un autre, il pourrait être moins prioritaire. Ce cas ne sera pas traité.
Test pool	= test du bit de présence dans le buffer du pool.
T1 T4	interruption du timer

Tableau 10 (1).



MC = message impair

Mp = message pair

① = réception fautive de EOT.

Tableau 10 (2).

Phases	DR.I/C.	REC.ENQ	REC.EOT	REC.CAR.Ill	REC.TTD	M1						Test Prior		M2						Test Pool		RECEPTION			
						STX.TXT	STX.TXT	ETB.cc	ETX.cc	ETB/ETX	ETB/ETX.cc	Occupé	Disponible	STX.TXT	STX.TXT	ETB cc	ETX cc	ETB/ETX	ETB/ETX.cc	0	1	ETAT	ENVOI WRITE	Réception READ	
$8'_2$		C	Φ																			idem $8'_1$		ENQ	
13			Φ													14	15	16	17			$STX\ M_p$ reçu		TXT # (ETB/ETX.CC)	PGL = RECEPTION
14																				14_1	$14'_1$	ETB.cc M_p reçu			PGL = FIN 1 READ, POSIT.EX.FL Nb+1, Posit.pool rempli, MAJ.AD.POOL test si pool suivant libre.
14_1	14_2																					ETB cc M_p reçu et pool prêt	ACK2		
14_2			Φ		14_3	18	7															Test STX M_1		STX	PGL = DEB.READ
14_3	14_2																						NAK		
15	15_1																					ETX cc M_p reçu			Idem ETB mais PGL = FIN 2 READ POSIT TSF.IN = 0 et pas test si Pool suivant libre.
15_1			1 ém.																					EOT	TSFIN = 0 PGL = ATT. WRITE
16		17	Φ																			ATT.ENQ		ENQ	
17	8_2																						NAK		
18		8_1	①																			ATT.ENQ		ENQ	
$14'_1$	$14'_2$																					ETB.cc M_p reçu et pool plein	WACK		
$14'_2$		14	Φ																			idem $14'_1$		ENQ	

Phases	DR.INC.	CPUT	CPUT ACTION	EM.ETB	EM.ETX.	ETB.ETX	REC.ENQ	REC.ACK0	REC.ACK1	REC.NAK	REC.WACK	REC.C.INV.	TEST ENQ F*	TEST ENQ F	TEST BLOC F*	TEST BLOC F	TEST PRER L	TEST PRER D	TEST POOL 0	TEST POOL 1	3' T_1	3' T_4	2'	EMISSION ETAT	EMISSION WRITE	EMISSION READ	EMISSION T_1	EMISSION T_4	Compteurs ENQ	Compteurs BLOC	
1		2	1 réc																					Libre							**Passage en 2 se fait lors de macro CPUT lors de CPUT ACTION, on passe au module de réception.**
2	2_1																							ENQ EMIS	ENQ				+1		Enlever EX.FL SYST.NOT = 0 ERR.CUPI = 0
2_1							6	3_0		4	2_1	a									a					+				Si NAK reu - erreur dans CUPI - posit. SYST.NOT.
4																						1 réc		EOT EMIS.	EOT		0	+			POST.EX.FL. Après T4 : passer en lecture, TSFIN=1 / TSFOUT = 0 PGL = ATT. READ
6																	2	1 réc									0				R1 possible seulement si liaison avec ordinateur plus prioritaire → cas non traité
$2'_1$																						$2'_2$					0 et +				le terminal demande de retarder l'envoi, on attend 3 secondes
$2'_2$	2_1																								ENQ						
a													2	b													0			0	Test cpt. ENQ pour s'arrêter après le Nb d'essais voulus.
b	4																												0	0	Utilis. prévenu si erreur car POSIT (SYST.NOT / CUPI) lors de l'erreur
3_0																			3	3							0		0	0	Test Nb max., Nb, pool → posit. ETB/ETX PGL = DEB.WRITE
3			6 1	6 2	6 3																			Mess. M_i émis.	Texte						PGL = émission.
$6'_1$									7_0	A_1	$3'_1$	a_1									a_1			M_i avec ETB			+				PGL=FIN 1 WRITE Après ACK bon : MAJ ADR2pool, Nb Max-1, Nb-1 sinon posit.erreur (SYST.NOT, CUPI)
6_2								a_1	4	A_1	$3'_2$	a_1									a_1			M_i avec ETX		réponse	+				PGL = FIN 2 WRITE Après ACK bon : MAJ ADR2pool, Nb Max-1 /Nb - 1 sinon : Posit.erreur
6_3																					b			M_i sans ETB ni ETX			+				Si le terminal n'a reçu ni ETB, ni ETX il ne répond pas → T1

Tableau 10 (4).

PHASES	BR.INIC.	CPUT	CPUT ACTION	EM.ETB	EM.ETX	ETB.ETX	REC.ENQ	REC.ACK0	REC.ACK1	REC.NAK	REC.[illegible]ACK	REC.C.INV.	TEST ENQ F1	TEST ENQ F	TEST BLOC F1	TEST BLOC F	TEST PRIOR L	TEST PRIOR D	TEST POOL O	TEST POOL 1	TEST 3' T_1	TEST 25' T_4	EMISSION ETAT	EMISSION WRITE	EMISSION READ	EMISSION T_1	EMISSION T_4	COMPTEURS ENQ	COMPTEURS BLOC	
3'	3"																							TTD		+				
3"										C		C									30				Rép.					
C																					30					+				
$3'_1$																					$3''_1$					0 et +				
$3''_1$	6_1																							ENQ						
$3'_2$																					$3''_2$					0 et +				
$3''_2$	6_2																							ENQ						
$3'_3$																					$3''_3$					0 et +				
$3''_3$	6_3																							ENQ						
5				6_1	6_2	6_3																	M_1 réémis	Texte				0	+1	PGL - EMISSION
a_1													b_2	b												0				
b_1	b'_1																							ENQ				+1		
b'_1								A_1	7_0	A_1											a_1				ACK_x	+				
A_1															5	b										0				

Tableau 10 (5).

PHASES	HR. IntC	CPUT	CPUT ACTION	EM. ETB	EM. ETX	ETD. ETX	REC. ENQ	REC. ACK0	REC. ACK1	REC. NAK	REC. WACK	REC. C.INV.	TEST ENQ	TEST ENQ F	TEST BLOC	TEST BLOC F	TEST PRDR	TEST PRDR L	TEST POOL	TEST POOL D	TEST POOL 0	TEST POOL 1	3" T_1	25" T_4	EMISSION ETAT	EMISSION WRITE	EMISSION READ	EMISSION T 1	EMISSION T 4	EMISSION Compteurs ENQ	EMISSION Compteurs BLOC	
7_0																					7'	7						0		0	0	PGL-DEB. WRITE
7				7_1	7_2	7_3																			Mess M_p émis.	Texte							PGL - EMISSION
7_1								3_0		A_2	7_1	a_2											a_2		Mess. M_p avec ETB			+				PGL - FIN 1 WRITE Après ACK bon : MAJ AD2 pool, Nb Max - 1, Nb-1 sinon POSIT ERREUR 'S. ST...OT, CUPI
7_2								4	a_2	A_2	7_2	a_2											a_2		Mess. M_p avec ETX		Réponse	+				PGL - FIN 2 WRITE Après ACK bon : MAJ AD2 POOL, Nb-1, Nb Max. -1 sinon : POSIT erreur 'S. ST NOT, CUPI
7_3																							b		Mess. M_p sans ETB ni ETX			+				le terminal ne répond pas
7'	7"																									TTD							
7"										c'		c'											7_0				REP						
c'																							7_0					+					
$7'_1$																							$7''_1$					0 et +					on a reçu WACK → on attend 3"
$7''_1$	7_1																									ENQ							
$7'_2$																							$7''_2$					0 et +					
$7''_2$	7_2																									ENQ							
$7'_3$																							$7''_3$					0 et +					
$7''_3$	7_3																									ENQ							
c				c_1	c_2	c_3																			M_p réémis.	Texte				0		+1	PGL - EMISSION
a_2													b_2	b														0					
b_2	b'_2																									ENQ					+1		
b'_2								$3'A_2$	A_2														a_2				ACK	+					
A_2															8	b												0					

IV.2.3.3.2. EMPLOI DU TABLEAU.

Lors de chaque évènement (première ligne), il y a un test de la phase de départ du module pour savoir dans quelle ligne on se trouve :

- si l'intersection de l'évènement et de la ligne de phase 1 de départ donne une phase 2 :
  - il faut vérifier s'il n'y a pas quelque chose à faire,
  - il faut exécuter l'action de cette phase 2,
  - il faut remplacer la phase de départ 1 par la phase 2.
- si l'intersection est vide, il faut terminer la liaison en signalant erreur indéterminée dans SYS .NOT (CUPI) et positionner EX.FL pour signaler que l'opération est terminée.

Exemple :

Soit l'évènement : réception ACK0 :

- le test de la phase donne 21 émission, l'intersection avec ACK0 donne 30,
- il faut se diriger en 30 et exécuter les actions prévues en dernière colonne.

Après on aura l'évènement : fin CCW après avoir envoyé ETB :

- le test de la phase donne 30 émission et le test du caractère positionné donne ETB, donc l'intersection est 61.
- il faut se diriger vers 61 et exécuter :
  - PGL = FIN 1 WRITE
  - READ réponse

Après on aura l'évènement fin CCW, après avoir reçu ACKØ :

- le test de la phase donne 61 émission,
- le test de la réponse donne ACKØ,
- l'intersection est 7.

Il faut donc exécuter les actions prévues et se diriger en 7.

IV.2.3.4. MODULE D'ERREURS.

IV.2.3.4.1. CLASSEMENT ET TRAITEMENT.

Il suffit d'ajouter dans les erreurs dues à l'utilisateur :

(3f) fin de lecture de message : si tous les blocs envoyés par le terminal ont été pris par CGET.

(3g) pool out plein : si le message est trop long pour trouver place dans le pool.

IV.2.3.4.2. TOPOGRAPHIE DES ERREURS ET SIGNALEMENTS.

Après CPUT ACTIVATE :

tampon indisponible	ligne désactivée	NB	1
ligne indisponible		NA	1
ligne déjà activée		ND	2
ligne inconnue		NF	4

Après CPUT DEACTIVATE :

ligne non activée	NE	3
ligne inconnue	NF	4

Après CGET :

erreur pool vide, rien à lire	NJ	8
Fin lecture	NN	4
erreur longueur	NG	5
attendre un message	NO	6

Après CPUT :

pool plein	NG	5
lecture en cours	NL	8
ligne désactivée E.I.	NC	1
tampon ind.	NB	1
ligne ind.	NA	1
ligne non activée	NE	3
ligne inconnue	NF	4
écriture en cours	NK	7
Message à lire	NM	6

Après CPUT ACTION :

ligne désactivée E.I.	NC	1
tampon ind.	NB	1
ligne ind.	NA	1
ligne non activée	NE	3
message à lire	NM	6
écriture lancée	NK	7
lecture lancée	NL	8

Erreurs lors d'une opération.

	Erreur	Traitement
Ligne indisponible	EA	fin de liaison
Tampon indisponible	EB	
Parité	EC	après 3 essais
Perte de caractère	EG	fin de liaison
BC ≠ 0 en réception	EI	ignorer
en émission	EI	après 3 essais
pas de STX en début	E1	fin de liaison
pas de ETX en fin	E4	

Erreur indéterminée	E5	fin de liaison
Erreur de priorité	E6	non traité

Après 3 essais de correction, on coupe la liaison en prévenant par un message console. Ce sera à l'opérateur de décider de désactiver la ligne.

V. TROISIEME PROCEDURE.

V.1. DIFFERENCE AVEC LA DEUXIEME PROCEDURE.

Un terminal ne peut envoyer un message quand il le désire. Il doit attendre que l'ordinateur emploie une méthode de "Polling". Cela signifie qu'à période fixe, il interroge les lignes successivement et demande à chaque terminal s'il a un message à lui envoyer.
L'ordinateur passe à la ligne suivante, soit lorsqu'il reçoit un message sur la ligne, soit lorsqu'il a interrogé tous les terminaux et qu'aucun d'eux ne désire lui envoyer de message.
Lors de l'interrogation, l'ordinateur n'attend pas indéfiniment une réponse. Après un temps déterminé, si le terminal n'a pas répondu, c'est considéré comme une erreur.
Cette procédure implique deux différences essentielles.

1. Il faudra, toutes les secondes par exemple, une interruption qui permettra le lancement du "Polling" sur les lignes.
   L'établissement de la liaison terminal-ordinateur sera donc spécial, il faudra peut-être questionner plusieurs terminaux avant d'entamer la procédure de réception de message.

2. Une opération de lecture, qui sera demandée après une écriture ou par les macro-instructions CPUT ACTION et CPUT ACTIVATE, préparera simplement la ligne concernée pour permettre le lancement du "polling" suivant.

V.2. MODIFICATIONS A APPORTER A LA DEUXIEME PROCEDURE.

V.2.1. DEROULEMENT GENERAL.

Toutes les secondes, une routine d'interruption exécute le "polling" sur les lignes le permettant, c'est-à-dire les lignes se trouvant dans la phase "ATTENTE READ".

Après une écriture, au lieu de relancer une lecture, il faut positionner la ligne dans cette phase "ATTENTE READ".

De même les macro-instructions CPUT ACTION et CPUT ACTIVATE, après vérification normale et ouverture de ligne pour la seconde, positionneront la ligne dans la phase "ATTENTE READ".

V.2.2. JONCTION AVEC LE PROGRAMME D'APPLICATION.

V.2.2.1. OPERATIONS DE LECTURE/ECRITURE.

Opération d'écriture : il suffira d'ajouter au n° de ligne, le n° de terminal concerné. Ces numéros seront les numéros courants (cf. V.2.2.2) ou bien seront passés comme paramètres par "CPUT".

Opération de lecture : pour chaque ligne il y aura une liste de terminaux avec une zone "numéro courant de terminal" pour cette ligne.

Lors d'un "polling" sur une ligne, le module spécial se servira de la liste et interrogera successivement tous les terminaux de la ligne.

Dès que le module aura trouvé un terminal prêt, il entamera la procédure de réception. Le prochain polling recommencera alors à partir du terminal suivant.

Si le module ne trouve aucun terminal désireux de lui envoyer un message, il s'arrête et attend le prochain "polling".

V.2.2.2. ZONES DE COMMUNICATION.

Il faut ajouter dans la "zone B ①", une seule fois, le numéro de terminal courant de la ligne courante.

Il faut ajouter dans la "zone B ②", une fois par ligne, la liste des terminaux de cette ligne

et son numéro de terminal courant.

Il faut ajouter dans cette "zone B ②", une fois par terminal, les zones "CUPI, codage, SYSNOT" et un bit d'activation.

Bit d'activation :

L'opération de "polling" ne prendra en considération que les terminaux dont ce bit est positionné à 0.

La console pourra positionner ces bits à 0 par CPUT ACTIVATION (n° de ligne, n° de terminal) pour rendre les terminaux actifs et les positionner à 1 par CPUT DEACTIVATION n° de ligne, n° de terminal) pour rendre les terminaux non actifs.

V.2.2.3. INSTRUCTIONS DE COMMUNICATION.

CPUT ACTION : consiste simplement, après vérification de validité, à positionner TSF IN et à remettre la ligne dans la phase "ATTENTE READ" pour que le "polling" soit lancé sur la ligne.

CPUT ACTIVATE (n° ligne) : consiste simplement, après vérification de valdité, à préparer les zones de communication d'une ligne, à positionner le bit d'ouverture, TSFIN, et mettre la ligne en phase "ATTENTE READ".

Le "polling" sera lancé après un test du bit d'ouverture et de la phase de la ligne.

Ces deux macro-instructions enlèvent le bit EX.FL qui sera repositionné lors de la réception d'un bloc.

CPUT [DE] ACTIVATION (n° ligne, n° terminal) :
uniquement valable depuis la console.

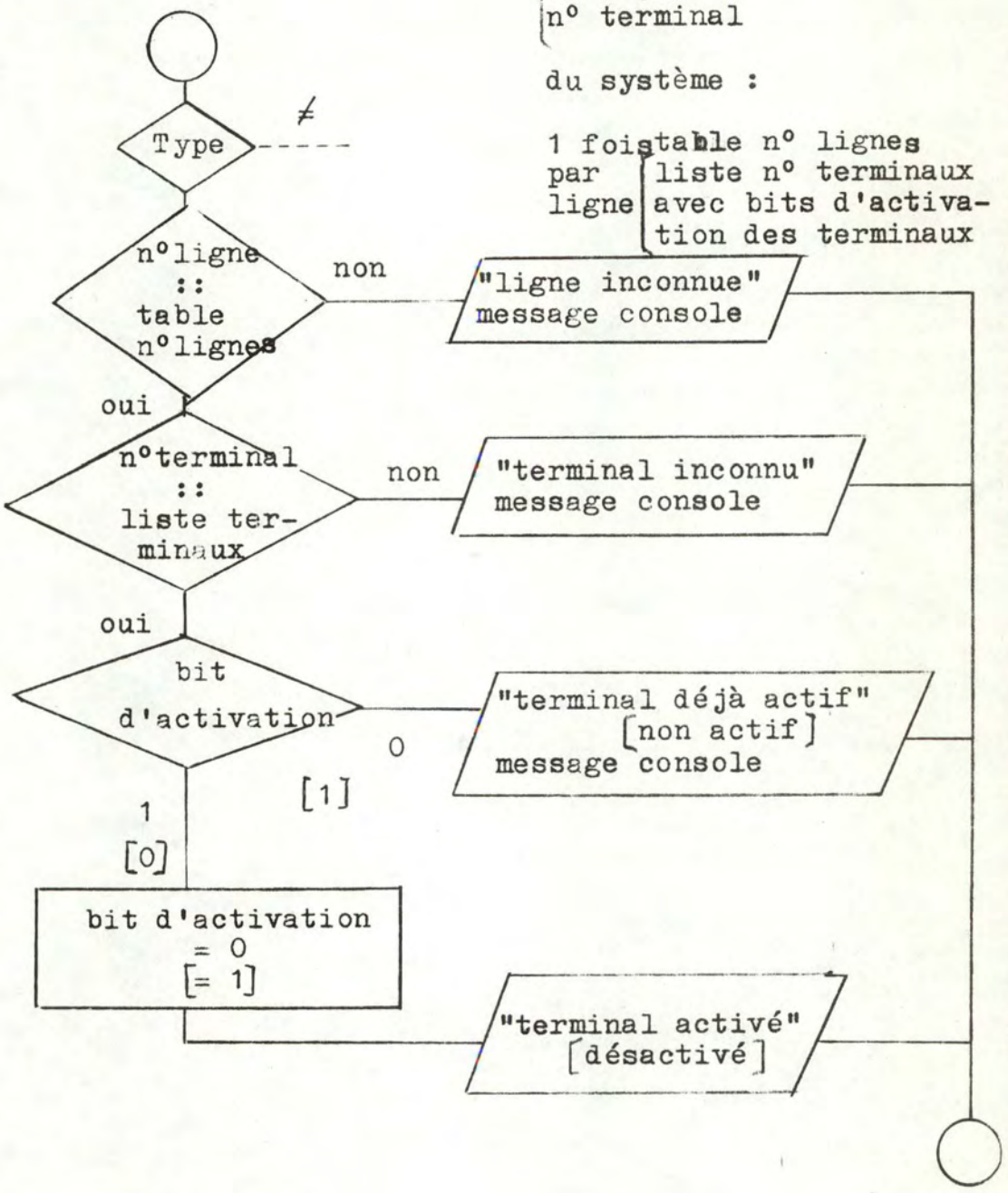
Organigramme.

paramètres de la console :

[n° ligne
n° terminal

du système :

1 fois par ligne [table n° lignes
liste n° terminaux avec bits d'activation des terminaux



V.2.3. MODULES DU MONITEUR.

V.2.3.1. MODULE SPECIAL.

En début de lecture de message, il y a la procédure spéciale de "polling" (cf.schéma 2). Elle consiste de la part de l'ordinateur à envoyer le caractère ENQ sur la ligne courante, vers le numéro courant de terminal. Lorsque le terminal répond affirmativement, il y a réception normale du message. Lorsque le terminal répond négativement, il y a mise à jour du numéro courant du terminal suivant la liste. Lorsqu'on a eu une réponse affirmative ou que l'on a testé tous les terminaux, le "polling" est terminé pour la ligne.

Modifications à apporter au tableau 2.

Emission : il suffit d'enlever l'événement "réc ENQ" qui est impossible (caractère invalide), et la phase 6 disparaîtra également.

En fin de phase 4, lorsqu'on recevra T_4, il ne faut pas passer en lecture mais positionner PGL = ATTENTE READ et TSFIN = 1.

Réception : seul changera l'établissement de la liaison (tableau 11).

Lorsqu'une lecture est lancée, il y a d'abord interrogation sur un terminal, avec déclenchement de l'horloge.
S'il n'y a rien de la part du terminal ou après 3 essais infructueux, il y a test de la liste des terminaux pour continuer à les interroger tous.

Le "polling" des terminaux peut se terminer pour deux raisons :

1. si un terminal répond affirmativement, la procédure de réception est entamée.

2. si aucun terminal ne veut envoyer de message, la ligne est remise en ATTENTE READ, prête à relancer le "polling" suivant. On signalera l'indication "polling négatif" dans SYS NOT.

Tableau 11.

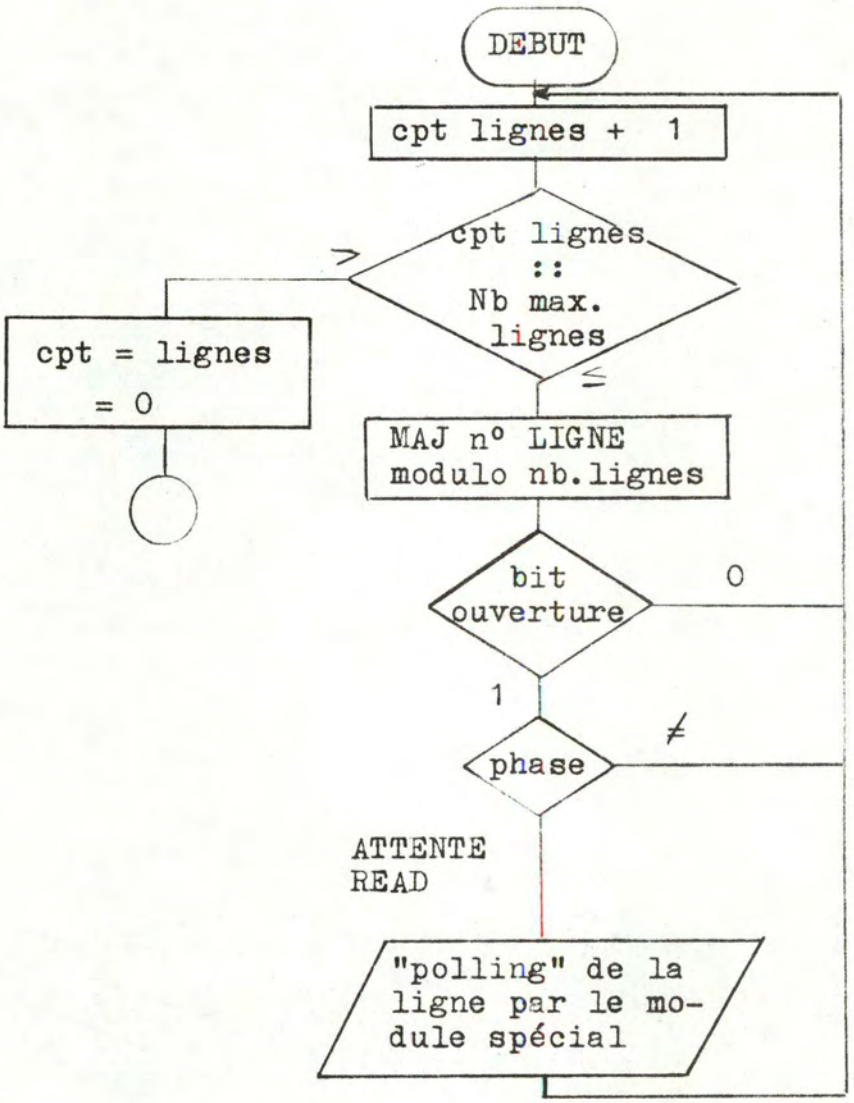
Phases	BR.I.C.	REC.E..Q	REC.EOT	REC. CAR.IN	REC. TTD	STX. IXT	M 1					Test Prior	M2		Lecture	Test Pool		Test E..Q		Test Pool		RECEPTION						
							STX.TXT	ETB.CC	ETX.CC	ETB/ETX	ETMETX.CC	Occupé	Dispo..	Idem précéd.								ETAT	WRITE	READ	ENQ	T1		
1 (émiss.)															1												
1																				2	1 ém						si pool plein → erreur { SYS NOT / CUPI
2	3																						END		+		Enlever EX. FL SYST. NOT = 0 ERR. CUPI = 0
3			1'			a	7							a										REP.		+	
1'															2	1 ém									0	0	- continuer polling si liste non terminée → MAJ AD POLLING - imprimer Mess. erreur à console
a																	2	1'									
7																									0	0	

V.2.3.2. INTERRUPTIONS.

1. INTERRUPTION TIMER TOUTES LES SECONDES :

   Cette routine utilise deux zones d'informations qui lui sont réservées :

   1. cpt lignes : compteur de lignes,
   2. Nb max. lignes : nombre maximum de lignes utilisables.



2. fin de liaison après une écriture :

   il faut positionner TSFIN = 1 et phase ATTENTE READ

au lieu de relancer une lecture comme dans la deuxième procédure.

V.2.3.3. ERREURS.

Il faut ajouter dans SYS NOT l'indication "polling négatif", c'est-à-dire que sur une ligne aucun terminal n'a désiré envoyer un message.

VI. PLUSIEURS PROCEDURES ET PLUSIEURS PROGRAMMES D'APPLICATION SIMULTANES.

VI.1. PLUSIEURS PROCEDURES.

Il faudra un tableau indiquant sous quelle procédure la ligne travaille (L'indication de procédure peut également se trouver dans la "zoneB (2) de chaque ligne).

n° ligne	procédure

Pour chaque opération sur une ligne, il faut d'abord tester sous quelle procédure il faut travailler.

VI.2. PLUSIEURS PROGRAMMES D'APPLICATION.

VI.2.1. RECONNAISSANCE DU PROGRAMME D'APPLICATION.

Le moniteur doit savoir à quel P.APPL. s'adresse un message en provenance d'une ligne. Il faudra donc le tableau suivant :

n° ligne	P.APPL permis

(Les P.APPL permis peuvent se trouver dans la zone B (2)).

Si une ligne peut envoyer des messages pour plusieurs P.APPL, il y aura une liste de P.APPL à côté de son n° ligne (ou dans sa zone B ②).
Le message devra donc indiquer par un code reconnaissable par le moniteur, à quel P.APPL il s'adresse. Et lorsque le programme de gestion de lignes trouvera un message, il devra d'abord tester la validité du code du P.APPL. auquel il s'adresse.
Ensuite seulement, il pourra débloquer le P.APPL.

VI.2.2. PASSAGE DE MAIN ENTRE LES PROGRAMMES.

Le PGL examine ligne par ligne s'il y a un message, vérifie la validité et débloque le P.APPL. approprié si celui-ci attend un message. Tous les P.APPL. possibles sont débloqués, puis le D.T. reçoit la main.

C'est le D.T. qui établira les priorités éventuelles en passant d'abord la main au P.APPL. plus prioritaire.

Remarque :

Ce sont des priorités non préemptives, c'est-à-dire que chaque programme ne prendra la main à un autre moins prioritaire que si celui-ci s'est arrêté par SUSP ou WAIT.

VI.3. MODIFICATIONS A APPORTER.

VI.3.1. ZONES DE COMMUNICATION.

Il y aura une fois : les tables de codes servant à la conversion des messages en codes des terminaux ou de l'ordinateur, et une zone "P.APPLIC.courant".

Il y aura par P.APPL :
- le n° ligne courant
- AD.CUPI, NOTIF
- les tables de lignes permises.

On peut retrouver ces zones à partir du code P.APPL. du message.

Les tables de lignes permises serviront à vérifier la validité des macro-instructions employant comme paramètre un numéro de ligne à laquelle elles s'adressent.

Il faudra ajouter par ligne, éventuellement, la procédure employée et les P.APPL. permis cfr. VI.2.1.).

VI.3.2. DISTRIBUTEUR DE TACHES.

Il débloquera tous les WAIT avant de rendre la main à un P.APPL.

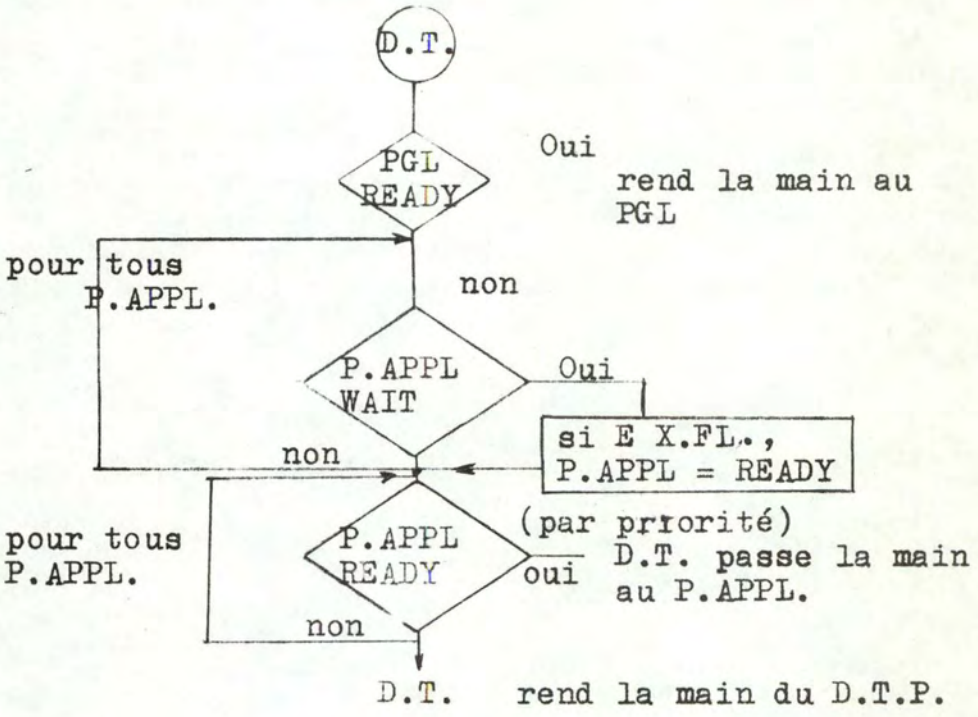


Table :

P.APPL. 1			P.APPL. 2			---- P.APPL.n
Ready	Non Ready	WAIT				——— ------- --------
Adresse retour						

L'adresse de retour sera positionnée par les macro-instructions SUSP et WAIT.

VI.3.3. PROGRAMME DE GESTION DE LIGNES.

Le bit d'état du PGL,à l'usage du D.T., sera positionné "ready" par la routine d'interruption de fin de premier bloc reçu par l'ordinateur, si le P.APPL. concerné est en SUSP.

Lors de la réception du premier bloc, l'interruption provoquée par le caractère ETB analysera le code P.APPL. du message et le mettra dans une zone réservée à la routine d'interruption.

Ce code lui permettra d'accéder aux bits d'état du P.APPL. pour les tester. Si le P.APPL. est en SUSP, il faut positionner le PGL "ready".

Les interruptions de fin de bloc suivantes verront que le code est déjà dans la zone réservée et n'iront plus tester le P.APPL. Ce code sera enlevé par le dernier bloc (réception du caractère ETX).

Le PGL teste toutes les lignes. S'il trouve un message, il débloque le P.APPL. correspondant si celui-ci est en SUSP.

VII. CONCLUSION.

L'exécution de ce travail a permis la mise en évidence de deux points essentiels.

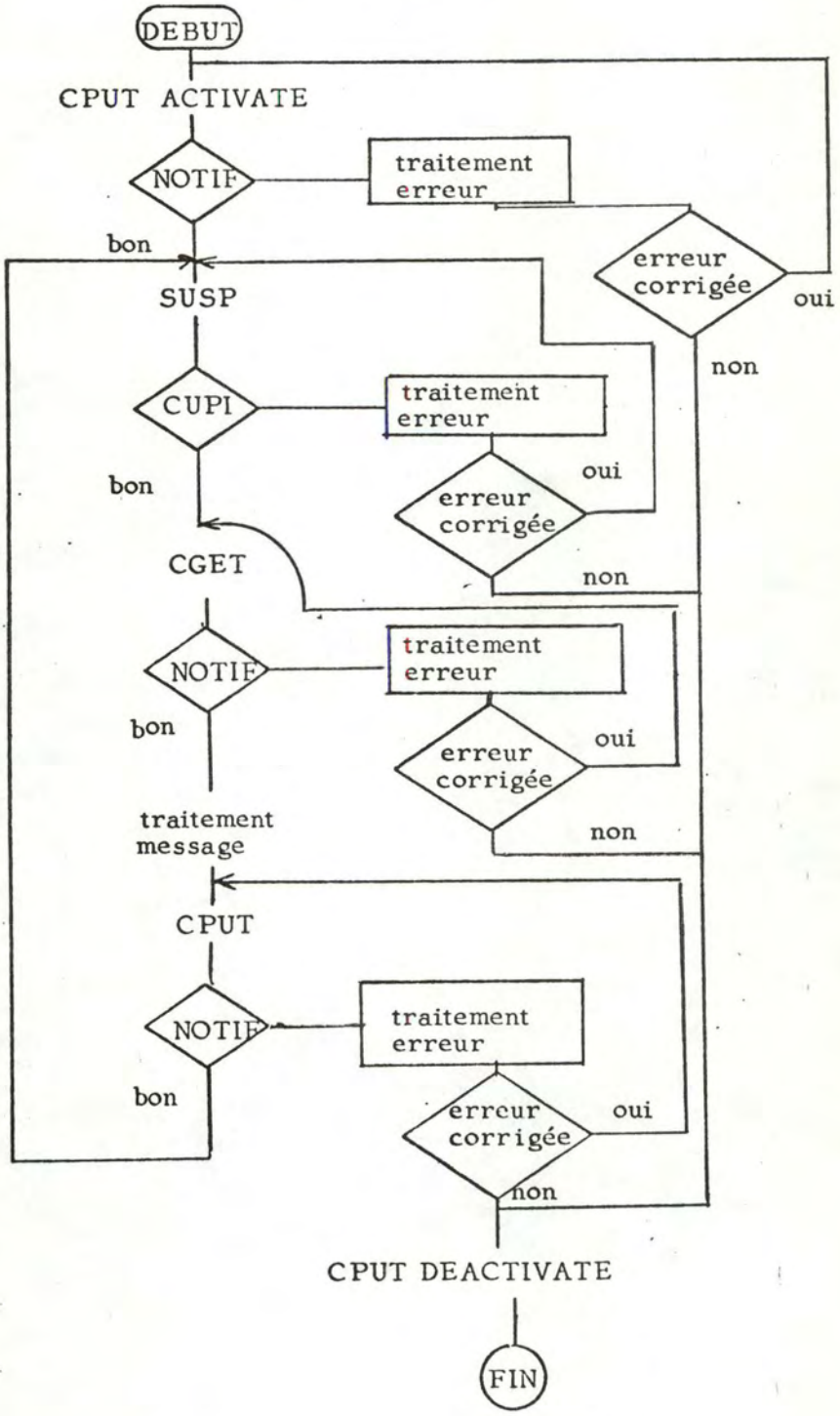
- Le premier est la différence entre une discussion à un niveau assez général et une réalisation.
  On peut garder ses distances vis-à-vis d'un problème en proposant des principes de résolution théoriques, nécessitant souvent des moyens énormes. Et de même par l'analyse d'applications toutes faites pour en retirer des lignes de conduite plus ou moins justifiées.
  Mais pour créer une solution en descendant au niveau de réalisation pratique, on est obligé de se plonger dans le problème et de tenir compte de tous ses aspects.
  Ce mémoire se situe à un niveau plus proche du second point de vue que du premier.

- Le deuxième point est une conséquence du sujet choisi.

  Comme de nombreux problèmes touchant à l'informatique, il se compose d'une multitude de détails et d'interdépendances. Il faut donc une logique rigoureuse pour classer chaque élément, lui accorder son importance relative et arriver au but sans se perdre en chemin.

Annexe 1.

Schéma de programme d'application.



Annexe 2.

Liste des abréviations et expressions utilisées.

lgn pt à pt : ligne point à point, ne reliant qu'un seul terminal à l'ordinateur.

lgn multipt : ligne multipoint, reliant plusieurs terminaux à l'ordinateur.

P.APPL : programme d'application, devant traiter un message envoyé par un terminal.

PGL : programme de gestion de lignes, distribue aux programmes d'application les messages en provenance de différentes lignes et réciproquement.

DTP : distributeur de tâches principal, permet à l'une ou l'autre "partition" (ou ensemble de programmes) de se dérouler.

DT : distributeur de tâches, permet au PGL et aux P.APPL. de se dérouler alternativement et correctement.

Passer la main : Un seul programme peut se dérouler effectivement dans l'ordinateur à un moment donné. S'il "passe la main" à un autre, il s'arrête et permet à cet autre de se dérouler.

Pcounter : tout programme est composé d'instructions successives qui seront exécutées à la suite l'une de l'autre. L'adresse de l'instruction suivante à exécuter se trouve dans le Program counter.

Canal : les opérations de lecture/écriture (I/O), dans l'ordinateur, se déroulent par l'intermédiaire d'un canal et peuvent être alors simultanées au déroulement d'un programme.

CCW : Channel command word(mot de commande du canal).
C'est une espèce de petit programme indiquant à un canal ce qu'il doit faire pour exécuter une opération d'I/O.

CCB : Channel command byte (byte de commande du canal), indique à un canal divers renseignements concernant une opération d'I/O à diriger, par exemple, l'adresse du CCW.

BC : byte Count, indique le nombre de caractères devant intervenir dans une opération d'I/O.

EX.FL. : Exec.flag, information indiquant si une opération d'I/O est en cours ou est terminée.

STB : Status byte, zone indiquant dans quel état se trouve le tampon du terminal, c'est-à-dire la zone de transit du message entre le terminal et l'ordinateur.

S.B. : Sense byte, zone indiquant quelle erreur s'est produite sur une ligne et a mis le tampon du terminal dans l'état "erreur".

S.V.C. : Superviseur call, instruction spéciale. Un programme ne peut pas toujours employer tout le répertoire d'instructions possibles. Cependant, par SVC, il peut demander à d'autres programmes standards d'exécuter une suite d'instructions spéciales qui lui sont interdites.