TELCOM : un support de terminaux lourds sous un système à temps partagé

Zissimos Papacristodoulou

▶ **To cite this version:**

Zissimos Papacristodoulou. TELCOM : un support de terminaux lourds sous un système à temps partagé. Modélisation et simulation. Université Joseph-Fourier - Grenoble I; Institut National Polytechnique de Grenoble - INPG, 1974. Français. NNT : . tel-00284644

HAL Id: tel-00284644

https://theses.hal.science/tel-00284644

Submitted on 3 Jun 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

TU 681

THESE

présentée à

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

POUR OBTENIR LE GRADE DE

DOCTEUR INGENIEUR

Spécialité INFORMATIQUE

Zissimos
PAPACRISTODOULOU

TELCOM

UN SUPPORT DE TERMINAUX LOURDS SOUS UN SYSTEME A TEMPS PARTAGE

Thèse soutenue le 18 octobre 1974 devant la Commission d'Examen :

Président	:	L. BOLLIET
Rapporteur	:	R. MAHL
Examinateurs		S. KRAKOWIAK R. BOUCHE
Invité	:	J. du MASLE

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

LISTE DES PROFESSEURS

Président : Monsieur Michel SOUTIF
Vice-Président : Monsieur Gabriel CAU

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Georges	Clinique des maladies infectieuses
	ARNAUD Paul	Chimie
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BENOIT Jean	Radioélectricité
	BERNARD Alain	Mathématiques Pures
	BESSON Jean	Electrochimie
	BEZES Henri	Chirurgie générale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BONNIER Etienne	Electrochimie Electrométallurgie
	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques Appliquées
	BRAVARD Yves	Géographie
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	CABANAC Jean	Pathologie chirurgicale
	CABANEL Jean	Clinique rhumatologique et hydrologie
	CALAS François	Anatomie
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et Toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Oto-Rhino-Laryngologie
	CHATEAU Robert	Thérapeutique
	CHENE Marcel	Chimie papetière
	COEUR André	Pharmacie chimique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie Pathologique
	CRAYA Antoine	Mécanique

Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie
	FAU René	Clinique neuro-psychiatrique
	FELICI Noël	Electrostatique
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques Pures
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse numérique
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GIRAUD Pierre	Géologie
	KLEIN Joseph	Mathématiques Pures
Mme	KOFLER Lucie	Botanique et Physilogie végétale
MM.	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques Pures
MM.	MALGRANGE Bernard	Mathématiques Pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Seméiologie médicale
	MASSEPORT Jean	Géographie
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEEL Louis	Physique du solide
	OZENDA Paul	Botanique
	PAUTHENET René	Electrotechnique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET René	Servomécanismes
	PILLET Emile	Physique industrielle
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REULOS René	Physique industrielle
	RINALDI Renaud	Physique
	ROGET Jean	Clinique de pédiatrie et de puériculture
	SANTON Lucien	Mécanique
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SILBERT Robert	Mécanique des fluides
	SOUTIF Michel	Physique générale

MM.	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLAND François	Zoologie
	VALENTIN Jacques	Physique nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
M.	VERAIN André	Physique
Mme	VEYRET Germaine	Géographie
MM.	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	BULLEMER Bernhard	Physique
	HANO JUN-ICHI	Mathématiques Pures
	STEPHENS Michaël	Mathématiques appliquées

PROFESSEURS SANS CHAIRE

MM.	BEAUDOING André	Pédiatrie
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BONNETAIN Lucien	Chimie minérale
Mme	BONNIER Jane	Chimie générale
MM.	CARLIER Georges	Biologie végétale
	COHEN Joseph	Electrotechnique
	COUMES André	Radioélectricité
	DEPASSEL Roger	Mécanique des fluides
	DEPORTES Charles	Chimie minérale
	GAUTHIER Yves	Sciences biologiques
	GAVEND Michel	Pharmacologie
	GERMAIN Jean-Pierre	Mécanique
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	HACQUES Gérard	Calcul numérique
	JANIN Bernard	Géographie
Mme	KAHANE Josette	Physique
MM.	MULLER Jean-Michel	Thérapeutique
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	POULOUJADOFF Michel	Electrotechnique
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie générale
	ROBERT André	Chimie papetière
	DE ROUGEMONT Jacques	Neurochirurgie
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIBILLE Robert	Construction mécanique
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBLARD Pierre	Dermatologie
	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Yves	Chimie
	BEGUIN Claude	Chimie organique
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
	BILLET Jean	Géographie
	BLIMAN Samuel	Electronique (EIE)
	BLOCH Daniel	Electrotechnique
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BOUCHET Yves	Anatomie
	BOUVARD Maurice	Mécanique des fluides
	BRODEAU François	Mathématiques (IUT B)
	BRUGEL Lucien	Energétique
	BUISSON Roger	Physique
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CHIBON Pierre	Biologie animale
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CONTE René	Physique
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	DURAND Francis	Métallurgie
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GRIFFITHS Michaël	Mathématiques appliquées
	GROULADE Joseph	Biochimie médicale
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine préventive
	IDELMAN Simon	Physiologie animale
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	JOLY Jean-René	Mathématiques Pures
	JOUBERT Jean-Claude	Physique du solide
	JULLIEN Pierre	Mathématiques Pures
	KAHANE André	Physique générale
	KUHN Gérard	Physique
	LACOUME Jean-Louis	Physique
Mme	LAJZEROWICZ Jeannine	Physique
MM.	LANCIA Roland	Physique atomique
	LE JUNTER Noël	Electronique
	LEROY Philippe	Mathématiques
	LOISEAUX Jean-Marie	Physique nucléaire
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LUU DUC Cuong	Chimie organique
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et Médecine préventive
	MARECHAL Jean	Mécanique
	MARTIN-BOUYER Michel	Chimie (CUS)

MM.	MAYNARD Roger	Physique du solide
	MICHOULIER Jean	Physique (IUT A)
	MICOUD Max	Maladies infectieuses
	MOREAU René	Hydraulique (INP)
	NEGRE Robert	Mécanique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PELMONT Jean	Physiologie animale
	PERRET Jean	Neurologie
	PERRIN Louis	Pathologie expérimentale
	PFISTER Jean-Claude	Physique du solide
	PHELIP Xavier	Rhumatologie
Mlle	RIERY Yvette	Biologie animale
MM.	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RENAUD Maurice	Chimie
	RICHARD Lucien	Botanique
Mme	RINAUDO Marquerite	Chimie macromoléculaire
MM.	ROMIER Guy	Mathématiques (IUT B)
	SHOM Jean-Claude	Chimie générale
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VAN CUTSEM Bernard	Mathématiques appliquées
	VEILLON Gérard	Mathématiques appliquées (INP)
	VIALON Pierre	Géologie
	VOOG Robert	Médecine interne
	VROUSSOS Constantin	Radiologie
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES ASSOCIES

MM.	BOUDOURIS Georges	Radioélectricité
	CHEEKE John	Thermodynamique
	GOLDSCHMIDT Hubert	Mathématiques
	SIDNEY STUARD	Mathématiques Pures
	YACOUD Mahmoud	Médecine légale

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

Mme	BERIEL Hélène	Physilogie
Mme	RENAUDET Jacqueline	Microbiologie

Fait le 30 mai 1972.

Je tiens à remercier,

Monsieur le Professeur Louis BOLLIET qui a orienté l'ingénieur étranger que j'étais, à mon arrivée en 1970 à Grenoble, dans le domaine de systèmes et qui a bien voulu me faire l'honneur de présider le jury de cette thèse,

Monsieur Sacha KRAKOWIAK qui s'est beaucoup intéressé à mon travail et a accepté de faire partie du jury,

Monsieur Robert MAHL qui a bien voulu me faire l'honneur d'être rapporteur de cette thèse,

Monsieur Richard BOUCHE qui a accepté de faire partie du jury,

Monsieur Jean du MASLE, auteur du sujet, qui a guidé mes premiers pas dans le domaine de systèmes et supervisé ce travail sans jamais ménager son temps pour m'aider dans la réalisation du projet.

Je voudrais encore remercier,

Tous mes collègues de l'équipe de recherche de l'ENSIMAG et du CICG qui m'ont encouragé tout au long de ce travail, et en particulier J.P.ANSART; la réalisation de ce projet doit beaucoup à ses remarques et à sa collaboration.

Monsieur GASTINEL, Directeur du Centre Interuniversitaire de Calcul de Grenoble qui, conscient de l'intérêt de mon entreprise a fait tout ce qui était en son pouvoir pour faciliter mes conditions de travail et sans lequel je n'aurais pu mener ma thèse à terme.

Je me dois aussi de rappeler que l'origine de mon stage à Grenoble a été l'obtention d'une bourse accordée par le Gouvernement français.

La composition matérielle de ce texte est réalisée par moi-même avec le système SCRIPT de CP67/CMS.

J'exprime, enfin, ma gratitude à mon ami Pierre GROUSSIS qui m'a aidé à la présentation des figures, et au Service de Reprographie de l'IRMA qui a terminé la réalisation matérielle de cet ouvrage.

TELCOM

UN SUPPORT DE TERMINAUX LOURDS

SOUS UN SYSTEME A TEMPS PARTAGE

TABLE DES MATIERES

page

CHAPITRE 1 : INTRODUCTION 1

CHAPITRE 2 : LE SYSTEME TELCOM 8

2.1. LA STRUCTURE STATIQUE DU SYSTEME 9

2.1.1. L' écriture du système 9

2.1.2. L' organisation et le chargement du système 10

2.1.3. Emploi standard des registres 16

2.1.4. Conventions de liaison 17

2.1.5. Problèmes dûs à la réentrance 22

2.2. NOTIONS DE BASE DU SYSTEME 24

2.2.1. La notion du 'programme' 24

2.2.2. La notion de 'tâche' 24

2.2.3. La notion du 'processeur' 27

2.2.4. La notion d' 'événement' 28

2.2.5. La notion des 'primitives' 29

2.2.6. La notion de 'section critique' 30

2.2.7. La notion de 'ressource' 31

2.2.8. La notion de 'superviseur' 31

2.3. STRUCTURE FONCTIONNELLE DU SYSTEME 32

2.3.1. Fonctions du système TELCOM 32

2.3.2. Choix des tâches du système 36

2.3.2.1. Changement d' état des tâches 36

2.3.2.2. Le fonctionnement des tâches du système 38

2.3.3. Contraintes dûes à 1' environnement du système 40

2.3.4. Problèmes liés au fonctionnement du système et solutions adoptées 41

2.3.5. Vue générale du fonctionnement global du système 44

2.4. GESTION DES RESSOURCES - SYNCHRONISATION DES TACHES 48

2.4.1. La ressource 'processeur' 48

2.4.1.1. Le 'distributeur' 49

2.4.1.2. La primitive 'WAIT' 50

2.4.1.3. La primitive 'POST' 51

2.4.1.4. Evénements mémorisés ou non 51

2.4.2. La resource 'mémoire' 53

2.4.2.1. La primitive 'GTMN' 55

2.4.2.2. La primitive 'FRMN' 57

2.4.3. Les autres ressources du système 58

2.4.3.1. La primitive 'ENQ' 60

2.4.3.2. La primitive 'DEQ' 61

2.4.4. Un cas particulier des ressources : 'les listes' 61

2.4.4.1. La procédure 'PUTLST' 63

2.4.4.2. La procédure 'GETLST' 64

2.5. MECANISMES DE BASE DU SYSTEME 65

2.5.1. Les opérations d' entrée-sortie 65

2.5.1.1. Lancement d' une opération d' entrée-sortie 68

2.5.1.2. Le module 'EXCP' 69

2.5.1.3. Le module 'IOINT' 70

2.5.2. L' horloge (timer) du système 71

a. Généralités 71

b. Gestion de l' horloge 72

2.5.2.1. L' heure du jour dans le système TELCOM

- module 'TIME' 80

2.5.2.2. Les demandes d' intervalles de temps 81

2.5.2.3. Module 'STIMER' 82

2.5.2.4. Les interruptions de l' horloge

- module 'EXTINT' 83

2.5.3. Les appels au superviseur (SVC) 84

2.6. GESTION DES UNITES D' ENTREE-SORTIE 88

2.6.1. Les "device-routines" 88

2.6.2. Les unités virtuelles 90

2.6.2.1. Lecteur virtuel 90

2.6.2.2. Perforateur virtuel 96

2.6.2.3. Imprimante virtuelle 98

2.6.3. Les unités réelles 98

2.6.3.1. Le télétype opérateur du système 99

a. Particularités du comportement d' un télétype 99

b. L' automate de la gestion du télétype 100

CHAPITRE 3 : COMMUNICATION DES DONNEES 103

3.1. GENERALITES 104

3.1.1. Caractéristiques des liaisons par lignes téléphoniques 105

3.1.2. Le problème de synchronisation 106

3.1.3. La nécessité d' un protocole de transmission 107

3.2. LE PROTOCOLE B.S.C. 109

3.2.1. Les caractères de contrôle 109

3.2.2. Les messages 111

3.2.3. La synchronisation 112

3.2.4. Etablissement de liaison 113

3.2.5. Les réponses ACK0 et ACK1 114

3.2.6. Le contrôle d' erreurs de transmission 114

3.2.7. Le mode "transparent" 116

3.3. LES PROCEDURES DE TRANSMISSION 119

3.3.1. La procédure élémentaire B.S.C. 119

3.3.1.1. Son protocole de transmission 120

3.3.1.2. La structure des messages 121

3.3.1.3. L' automate de la procédure 122

3.3.2. La procédure "multileaving" 125

3.3.2.1. Son protocole de transmission 125

3.3.2.2. La structure des messages 126

a. BCB 126

b. FCS 128

c. RCB 129

d. SRCB 130

e. SCB 131

3.3.2.3. Quelques cas particuliers 132

3.3.2.4. L' automate de la procédure 134

3.4. LA GESTION DES LIGNES DANS LE SYSTEME TELCOM 137

3.4.1. Le module 'LINE' 138

3.4.2. Les modules de compactage (COMPCT) et de décompactage (DCMPCT) 141

CHAPITRE 4 : CONCLUSIONS 143

ANNEXE I : CHARGEUR INITIAL A PARTIR DE CARTES 150

ANNEXE II : MANUEL OPERATEUR 159

ANNEXE III : MANUEL DE MAINTENANCE 166

ANNEXE IV : DESCRIPTION DES PRIMITIVES EN PSEUDO-ALGOL 170

INDEX BIBLIOGRAPHIQUE 178

TABLE DES FIGURES

page

1. Chargement des sections de contrôle du système 13
2. Procédure de chargement 15
3. Liaison inter-section 19
4. Etats d' une tâche 25
5. Etats d' un processeur 28
6. Communication TELCOM - terminal lourd T1 33
7. T.C.B. 35
8. Changement d' état des tâches 37
9. E.C.B. 37
10. Les grandes lignes du fonctionnement des tâches 39
11. Vue générale du fonctionnement global du système 45
12. Image de la mémoire après le chargement du système 52
13. F.Q.E. 54
14. Tables des ressources 59
15. Liste 62
16. U.C.T. 66
17. B.I.O. 67
18. Graphique du 'timer' 73
19. Pile du 'timer' 75
20. I.R.B. 81
21. Pile de gestion de l' horloge 82
22. Trace 86

23. Matrice de l' automate du lecteur 95

24. Matrice de l' automate du perforateur 97

25. Matrice de l' automate du télétype 102

26. Signal de communication 105

27. La forme d' un message B.S.C. 111

28. Les enregistrements dans la procédure 'élémentaire B.S.C.' 122

29. La matrice de l' automate de la procédure 'élémentaire B.S.C.' 124

30. Les messages 'multileaving' 127

31. L' automate 'multileaving' 136

32. Extention d' un U.C.T. de ligne 139

CHAPITRE 1

INTRODUCTION

La troisieme génération des ordinateurs est marquée, du point de vue "software", par l' importance des systèmes d' exploitation et l' apparition des notions nouvelles de multiprogrammation, multitraitement, temps réel, temps partagé, et traitement à distance.

Un système d' exploitation est conçu en fonction du type de service auquel il est destiné. Il est classique de distinguer pour ne citer que les plus usuels:

-les systèmes d' exploitation par lots, où chaque utilisateur soumet son travail à un peripherique d' entrée (lecteur de cartes) et attend ses résultats à un peripherique de sortie (imprimante) sans aucune interaction avec le système,

-et les systèmes à partage de temps qui offrent virtuellement une machine à chacun des utilisateurs.

Au cours de ces dernières années, un effort important a été consacré au développement des communications "homme-machine" et "machine -machine" .

Le terme "télé-informatique" a fait son apparition pour couvrir l' ensemble des moyens et des techniques concernant le traitement de l' information à distance . L' intérêt de cette branche de l' informatique est de pouvoir offrir à un utilisateur distant les facilités d' un système (communication homme-machine) ou d' un ensemble de systèmes (communication machine-machine) .

Un effort parallele, donc, est consacré à la création des systèmes specialisés au télé-traitement et aussi à donner aux systèmes d' exploitation existants les moyens de télé-traitement. Les premières réalisations en téle-informatique (constituant aussi les premiers réseaux informatiques) avaient la forme d' un ordinateur central de grande taille relié avec plusieurs terminaux éloignés par des lignes téléphoniques.

Les terminaux sont des appareils capables de saisir l' information et de la transmettre, de même qu' ils devront être aptes à recevoir et à produire des éléments directement exploitables. Ces terminaux sont en réalité très specialisés et entièrement "hardware". Ils devront, donc, être rigoureusement compatibles et dans la plu-part des cas d' un seul constructeur. Si, par conséquent, un réseau existant nécessitait pour une nouvelle application l' acquisition d' un terminal ne figurant pas chez le constructeur initial, alors, de graves problèmes se posaient.

C' est à ce moment, que les ordinateurs de petite taille que l' on aurait pu croire condamnés par l' évolution, manifestent un regain de vitalité: ils offrent en effet l' avantage d' être programmables et de rendre ainsi possible la simulation de telle ou telle unité, et apportent toutes les possibilités de communication complète avec un grand ordinateur central.

Nous avons appelé ces petits ordinateurs des 'terminaux lourds'.

La présente étude se place dans le cadre de la "télé-informatique" . Son sujet particulier est la réalisation d' un support de terminaux lourds sous un système interactif à temps partagé .

Un tel système simule pour chacun de ses utilisateurs une machine réelle d' une configuration déterminée et à laquelle il est le seul à avoir accés . Cette machine est appelée "machine virtuelle" . Chaque machine virtuelle lorsqu' elle est activée peut utiliser les ressources de la machine réelle.

Pour notre réalisation le système interactif à temps partagé est le système CP67-CMS implanté sur la machine IBM 360/67 du centre interuniversitaire de calcul de Grenoble ; les terminaux lourds déjà supportés étant :

a. Le 1130 (IBM) de Mecanique des Fluides

b. Le T1600 (Télémecanique) de l' INPG

c. L' Ordoprocesseur de la MIAGE

d. Le T1600 (Télémecanique) de la Télémecanique et

e. L' Ordoprocesseur du DEUG.

Dans la perspective de connexion du centre interuniversitaire de calcul de Grenoble avec les réseaux d' ordinateurs SOC et CYCLADES ce support pourrait aussi être utilisé .

Pour la réalisation de ce projet nous avons créé le système TELCOM , un système simple "multitâche" , capable de gérer des lignes de communication vers des terminaux lourds . Nous avons essayé de construire ce système entièrement indépendant faisant intervenir au strict minimum la contrainte qu' il est destiné à fonctionner comme sous-système de CP67 . De ce fait il est autochargable et implanté sur une machine virtuelle nue (simulation par CP67 d' une machine réelle nue, de type IBM/360 standard). CP67 effectue à la demande l' attachement des lignes vers les terminaux lourds à cette machine.

Cette étude sera développée suivant deux axes :

A. Le système TELCOM (chapitre 2). Le principal intérêt de l' étude de ce système est sa simplicité . Simplicité qui fait apparaître clairement les problèmes essentiels de tout système et en même temps d' avoir une vue générale de son fonctionnement .

Le paragraphe 2.1. décrit la structure 'statique' du système c' est-à-dire son écriture , son organisation, et son chargement en évoquant les problèmes abordés dans ces domaines et leurs résolutions .

Les notions de base du système telles que programme , tâche, processeur , événement , superviseur etc. sont étudiées au paragraphe 2.2. Pour la compréhension du fonctionnement du système , il nous est apparu nécessaire de définir précisément toutes ces notions comme cela a été nécessaire pour sa conception.

La structure fonctionnelle du système est abordée au paragraphe 2.3. en s' appuyant sur les notions définies précédement . Une analyse des fonctions à accomplir par le système conduit aux solutions adoptées pour sa réalisation . Les principes , les contraintes , et les conventions ainsi qu' une vue globale du fonctionnement du système sont décrits .

Le paragraphe 2.4. est consacré à la gestion des ressources. Nous distinguons les ressources partageables et les ressources non partageables . Ressources partageables sont par exemple le code qui peut être partagé par plusieurs tâches , les différentes tables qui peuvent être consultées aussi par plusieurs tâches etc. Ces ressources ne posent aucun problème et notre seul souci

est de veiller à ce qu' elles ne soient jamais modifiées (réentrance du code) . Au contraire les ressources non partageables posent le problème de leur allocation et libération et en conséquence les problèmes de l' exclusion mutuelle et de la synchronisation des tâches.

Du reste le terme 'ressource' ne représentera que les ressources non partageables du système .

La description des principaux mécanismes du système TELCOM est abordee au paragraphe 2.5.: entrées-sorties, horloge, et appels au superviseur.

Une unité est gérée par une tâche et une seule, ce qui simplifie la gestion des entrées-sorties (plus de problèmes de conflits).

Une définition de l' horloge d' une machine IBM/360 précède la description de sa gestion et précise les services que nous pouvons attendre de ce mécanisme.

Enfin la description du mécanisme des appels au superviseur montre comment nous utilisons ces appels pour réaliser différentes fonctions du système; par exemple: garder trace du déroulement des programmes, masquage etc.

Dans le paragraphe 2.6. la gestion des unités d' entrée-sortie, tâche primordiale pour le système TELCOM est decrite. Nous abordons d' abord les points communs de cette gestion pour toute unité et puis nous distinguons les cas des unités virtuelles et des unités réelles.

B. La communication de données (chapitre 3).

Le paragraphe 3.1. est consacré à une introduction générale sur les systèmes de communication de données. Une étude est faite sur les différentes méthodes et techniques de transmission et la nécessité d' un "protocole de transmission" pour toute communication.

Le protocole le plus répandu à haute vitesse sur matériel IBM, le protocole de transmission binaire synchrone (B.S.C.), est décrit en détail dans le paragraphe 3.2.

Au paragraphe suivant (3.3.) nous detaillons les deux procédures de transmission: "élémentaire B.S.C." et "multileaving", construites sur le protocole B.S.C.

Une procédure de transmission se basant sur un protocole entreprend la gestion globale des lignes de communication.

Le paragraphe 3.4. décrit la gestion des lignes faite par le système TELCOM englobant aussi le problème de compactage et décompactage des messages transmis sur les lignes.

Le dernier chapitre (chapitre 4) est consacré à une critique postérieure de ce travail. L' expérience de quelques mois d' utilisation du système nous permet de voir ses défauts et ses avantages et de proposer des améliorations et des extensions possibles.

En annexe, enfin, figurent: le chargeur initial du système, le manuel opérateur, le guide de maintenance pour des modifications toujours envisageables et la description detaillée en pseudo-algol des sections de contrôle décrivant les primitives du système.

CHAPITRE 2

LE SYSTEME TELCOM

2.1. LA STRUCTURE STATIQUE DU SYSTEME .

Dans le terme 'structure statique' nous incluons tout ce qui concerne les conventions et les techniques de création et de chargement du système.

Nous traitons en particulier l' écriture et l' organisation du système en sections, son initialisation, et les conventions de liaison entre ses différents composants.

Cette structure constitue la base de la réalisation du système et a rendu la tâche de programmation systèmatique et donc plus aisée.

2.1.1. L' écriture du système.

Le système TELCOM est écrit en assembleur 360, dont on a utilisé intensivement les macro-instructions, ne disposant pas à ce moment de langage spécialisé utilisable (PL360, BSL, etc.)., Pour éviter les difficultés de programmation d' un tel langage nous avons adopté les règles suivantes:

a. Programmation modulaire.

C' est-à-dire, nous avons subdivisé le système total en sections dont chacune a une fonction bien définie et un interface clair avec les autres. Ce découpage a facilité non seulement la

programmation mais aussi la mise au point qui a pu s' effectuer section par section. Ceci nous était aussi imposé car le système a été entièrement développé en mode interactif.

b. Programmation réentrante.

C' est-à-dire, nous avons créé des programmes qui ne se modifient pas en cours d' exécution de façon à ce que plusieurs utilisateurs puissent utiliser le même programme en même temps, et utilisent des espaces de travail acquis dynamiquement.

c. Augmentation de la bibliothèque de macro-instructions.

Grace à cette règle nous avons réduit la codification répétitive et en conséquence la probabilité d' erreurs.

2.1.2. L' organisation et le chargement du système.

Le système est organisé en sections de programmes et sections de données pour permettre la réentrance. Toute section a une longueur inférieure à 4k octets évitant ainsi les problèmes de bases multiples et d' adressage en général.

Nous avons déjà mentionné que le système TELCOM est autochargeable. Il comporte un chargeur initial à partir de cartes et un chargeur secondaire à partir d' un disque.

Notre système de chargement est inspiré par celui du système MTS (Michigan Tecnical System), et s' effectue de la manière suivante:

Sur le lecteur de cartes nous mettons le chargeur du système suivi de toutes les sections de contrôle du système (cartes résultat d' assemblage) et nous procédons à un chargement initial (IPL) à l' adresse du lecteur.

Bien que nous parlons ici de lecteur de cartes il faut remarquer que néanmoins la construction du système n' a impliqué à aucun moment de manipulation objective des cartes qui sont toujours restées au niveau d' images de cartes sur un lecteur virtuel.

Le chargeur du système (SALOADER) débute par un chargeur absolu qui permet de charger le chargeur des sections du système. Une description plus détaillée du chargeur est donnée en annexe I. Ce chargeur lit les sections sur le lecteur une par une et en les chargeant en mémoire, remplit un vecteur d' adresses des sections (TVT) selon un schéma directement inspiré du système PL360 de N. Wirth. L' allocation d' un niveau de ce vecteur à une section du système est indépendant de l' ordre dans lequel se trouvent les sections sur le lecteur. Elle a été rendue systématique par l' adoption de la règle suivante:

Une section de code étant la plus petite entité pouvant être manipulée par un chargeur, son nom a la forme:

XXXXXXNN

où XXXXXX réprésentent 1 à 6 caractères alphanumériques,
(le premier étant toujours une lettre) et

NN deux chiffres de 0 à 9.

Ces deux chiffres indiquent l' entrée de la TVT correspondant à la section, par exemple: ALPHA07 et BETA112 sont associés respectivement aux entrées 7 et 12 de la TVT.

Mais le programmeur n' a pas à connaître le numéro NN de cnaque section. Il lui suffit de connaître le nom alphanumérique de la section et ces numéros sont affectés automatiquement par l' intermédiaire de macro-instructions.

Deux tableaux de caractères contiennent respectivement les noms (tableau BOX) et les nom qualifiés (avec les numéros NN) des sections (tableau COX). Ces tableaux sont déclarés comme des paramètres globaux du système:

```
GBLC  &BOX(100),&COX(100)     -déclaration-
```

Une suite d' affectations initiales à BOX des noms connus du programmeur, telle que par exemple:

```
        BOX(1)  SETC   'ALPHA'
        BOX(2)  SETC   'BETA'
```

permet par l' emploi de la macro FOX d' obtenir les affectations:

```
        COX(1)  SETC    'ALPHA01'
        COX(2)  SETC    'BETA02'
```

Le programmeur ne connaissant que les noms non qualifiés, pourra obtenir la correspondance avec les noms qualifiés, à la création des sections de contrôle, par la macro:

```
        ALPHA   $CSECT
```

qui a comme expansion:

```
        ALPHA01  CSECT
                 USING  ALPHA01,R15
```

(à remarquer que le registre R15 est utilisé comme base d' adressage de toute section de contrôle).

De cette façon, le chargeur trouve le numéro de chaque section à charger, alloue l' entrée correspondante dans le vecteur d' adresses, charge la section, et remplit l' entrée avec l' adresse de chargement de la section (figure 1).

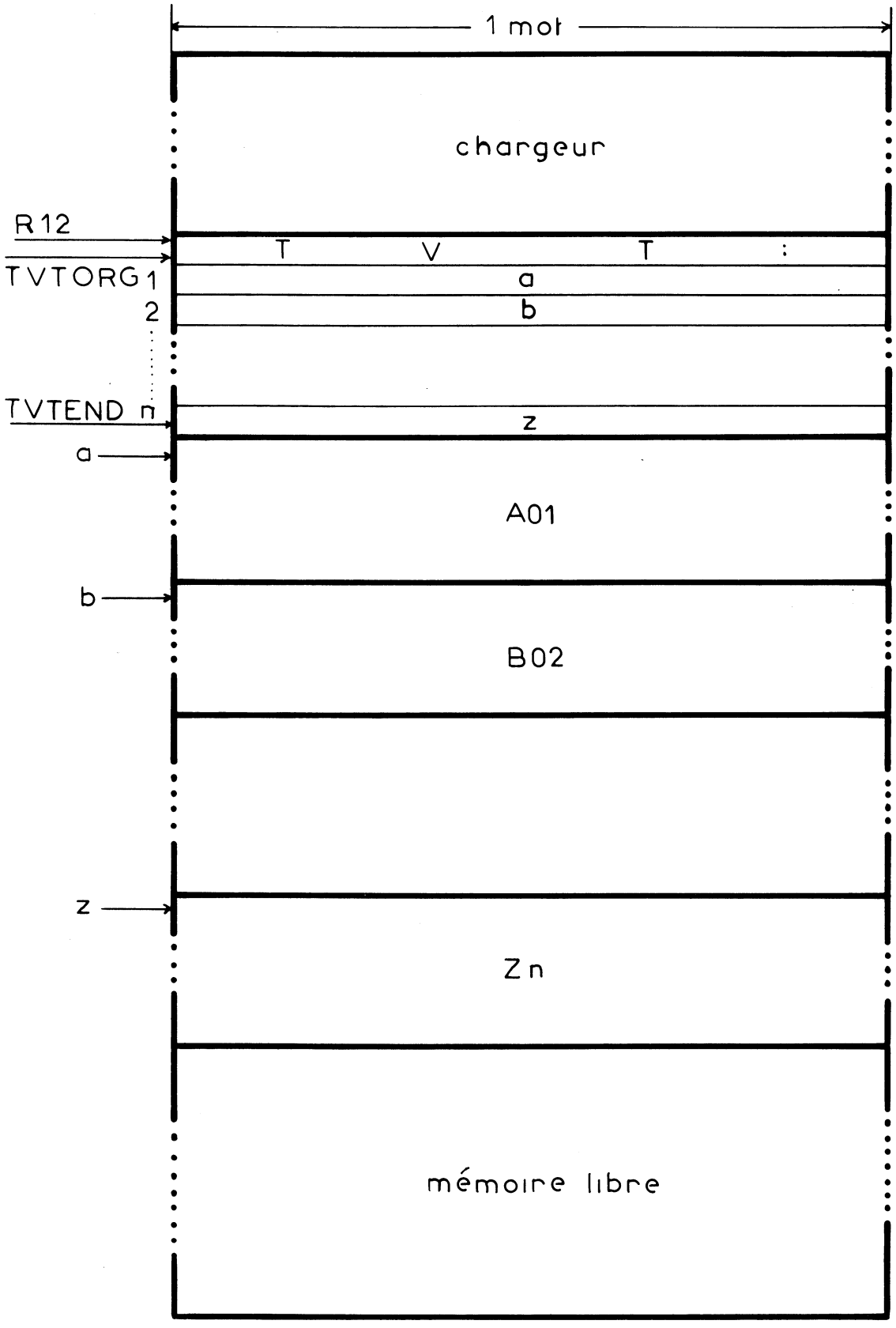


figure1.chargement des sections de controle du système

En fin de chargement le chargeur passe le contrôle à une section spéciale du système (INIT) qui effectue l' initialisation du système.

Pour la phase 'utilisation', la mise au point du système étant réalisée, nous avons créé un chargeur à partir de disque beaucoup plus rapide, qui se compose de deux modules:

Le premier (DSKWTR) recopie sur le disque une image du système chargé en mémoire.

Le second (DSKRDR) recopie en mémoire l' image du système qui se trouve sur le disque.

Le type de disque utilisé étant un IBM/2314 ou un mini-disque virtuel équivalent, ce disque est initialisé selon les normes du système CMS (sections de 800 octets).

Une fois le chargement à partir des cartes terminé le chargeur (à partir des cartes) ne passe plus le contrôle au module d' initialisation mais à DSKWTR, qui écrit sur un disque une image du système chargé en mémoire en ajoutant aussi la séquence d' initialisation sur le disque.

Ainsi nous pouvons effectuer un chargement initial (IPL) sur disque en provoquant la lecture en mémoire du second module du chargeur disque (DSKRDR) qui amène en mémoire l' ensemble du système TELCOM, et donne le contrôle au module d' initialisation.

L' ensemble de la procédure de chargement est schématisée par un organigramme (figure 2).

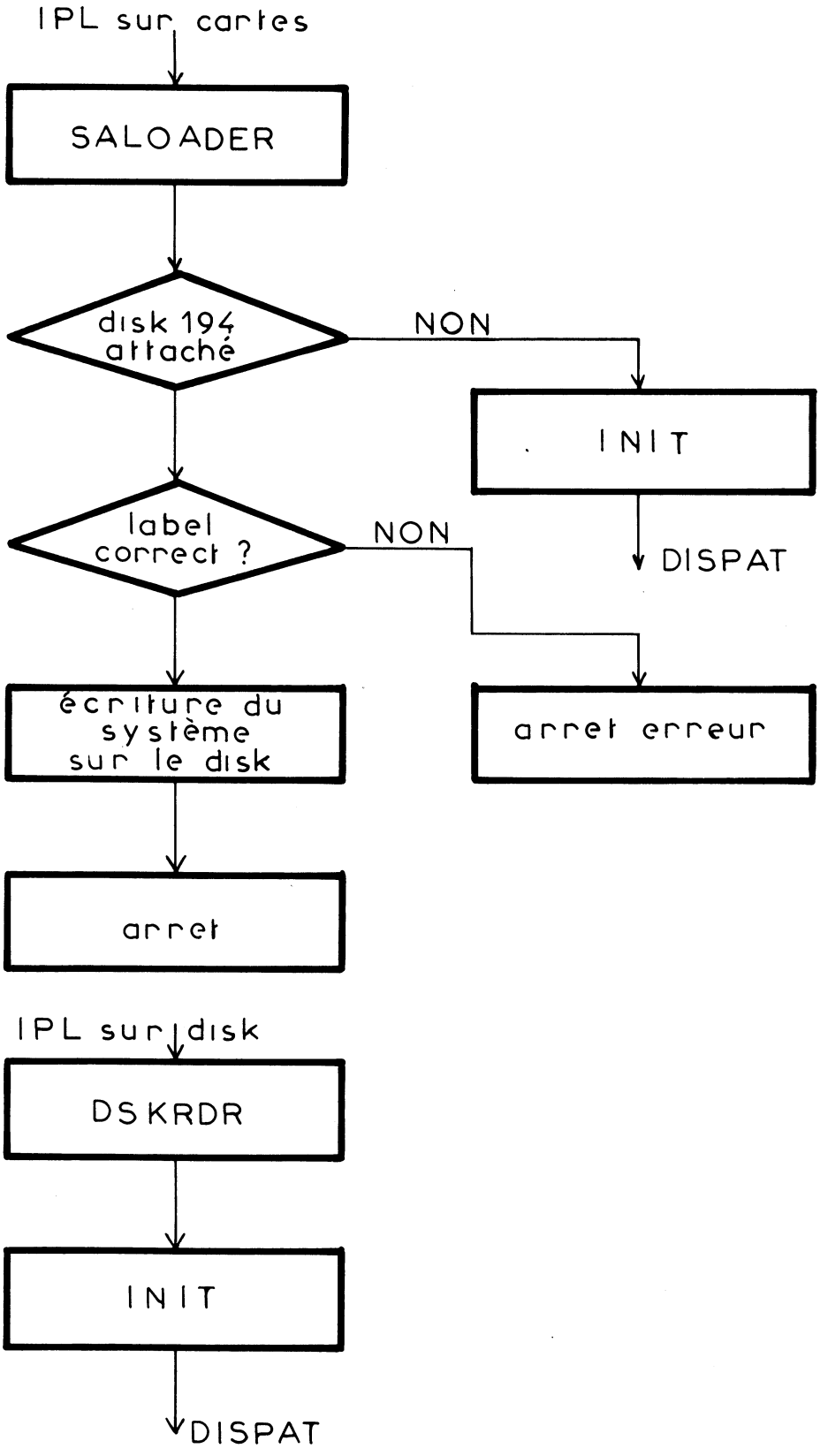


figure 2 procédure de chargement

2.1.3. Emploi standard des registres.

Une programmation modulaire implique la définition des conventions de liaison standards entre tous les modules, ce qui entraine l' emploi standard de certains registres. Les conventions adoptées sont inspirées de celles du système PL360:

R15.- Base d' adressase des sections.

A tout instant il ne peut y avoir qu' une instruction qui se déroule à la fois dans la machine TELCOM, en conséquence il suffit d' une seule base pour l' ensemble des programmes.

Le passage d' une section à une autre se fait par l' intermédiaire de la TVT avec remise à jour du registre R15.

R14.- Adresse de retour.

Au passage d' une section à une autre, R14 contient toujours l' adresse de retour à la section appelante.

R13.- Pointeur vers le bloc de contrôle (TCB) de la tâche active .

A toute tâche du système TELCOM est associé un bloc de contrôle (TCB) que nous détaillerons dans le paragraphe 2.3.2. R13 contient toujours l' adresse du TCB associé à la tâche active.

R12.- Pointeur vers la TVT.

Après le chargement du système, la construction et le remplissage par le chargeur du vecteur d' adresses des sections (TVT), R12 est initialisé avec l' adresse de la TVT et n' est plus modifié.

Remarque: En plaçant la TVT dans les 4 premiers octets de la mémoire on pourrait se dispenser de ce registre.

R11.- Pointeur vers la zone locale de sauvegarde des registres dans une section .

Nous sauvegardons les registres dans cette zone locale (acquise dynamiquement pour assurer la réentrance) lors d' appels inter-sections ne provoquant pas de changement de tâche.

R10.- Pointeur vers la section des données communes à toutes les sections (section DATA) .

R10 prend sa valeur après le chargement du système et ne change plus de valeur.

R9 - R2.- Registres banalisés.

R1 et R0.- Registres de passage des parametres.

Le but évident de ces conventions est d' obtenir des performances élevées car elles donnent accès direct à toutes les informations vitales du système en permanence.

2.1.4. Conventions de liaison.

L' appel d' une section à une autre se fait par la macro:

```
$CALL  ALPHA
```

qui a comme expansion:

```
L      R15,4xNN(R12)
BALR   R14,R15
L      R15,4xN'N'(R12)
```

ou NN est le numéro de la section appelée et

N'N' le numéro de l' appelante .

Si l' appel ne nécessite pas de retour il s' effectue par la macro:

```
$XCTL  ALPHA
```

qui a pour expansion :

```
L      R15,4xNN(R12)
BR     R15
```

Enfin l' accès à une section de données (DATA par exemple) se fait par la macro:

```
$ACCESS  ALPHA,REG=Ra,DUMMY=YES,LOADREG=YES
```

qui a comme expansion :

```
L      Ra,4xNN(R12)     (si LOADREG=YES)
USING  ALPHANN,Ra
```

suivi ou non (DUMMY=YES ou NO) de la DSECT de la section de données. Cela signifie que toute section de données qui peut être référencée par $ACCESS doit être définie sous forme de macro:

```
ALPHA  T=  C
           D
```

qui aura comme expansion :

```
si T=C   ALPHANN    CSECT
                      .
                      .
                      .
```

et

```
si T=D   ALPHANN    DSECT
                      .
                      .
                      .
```

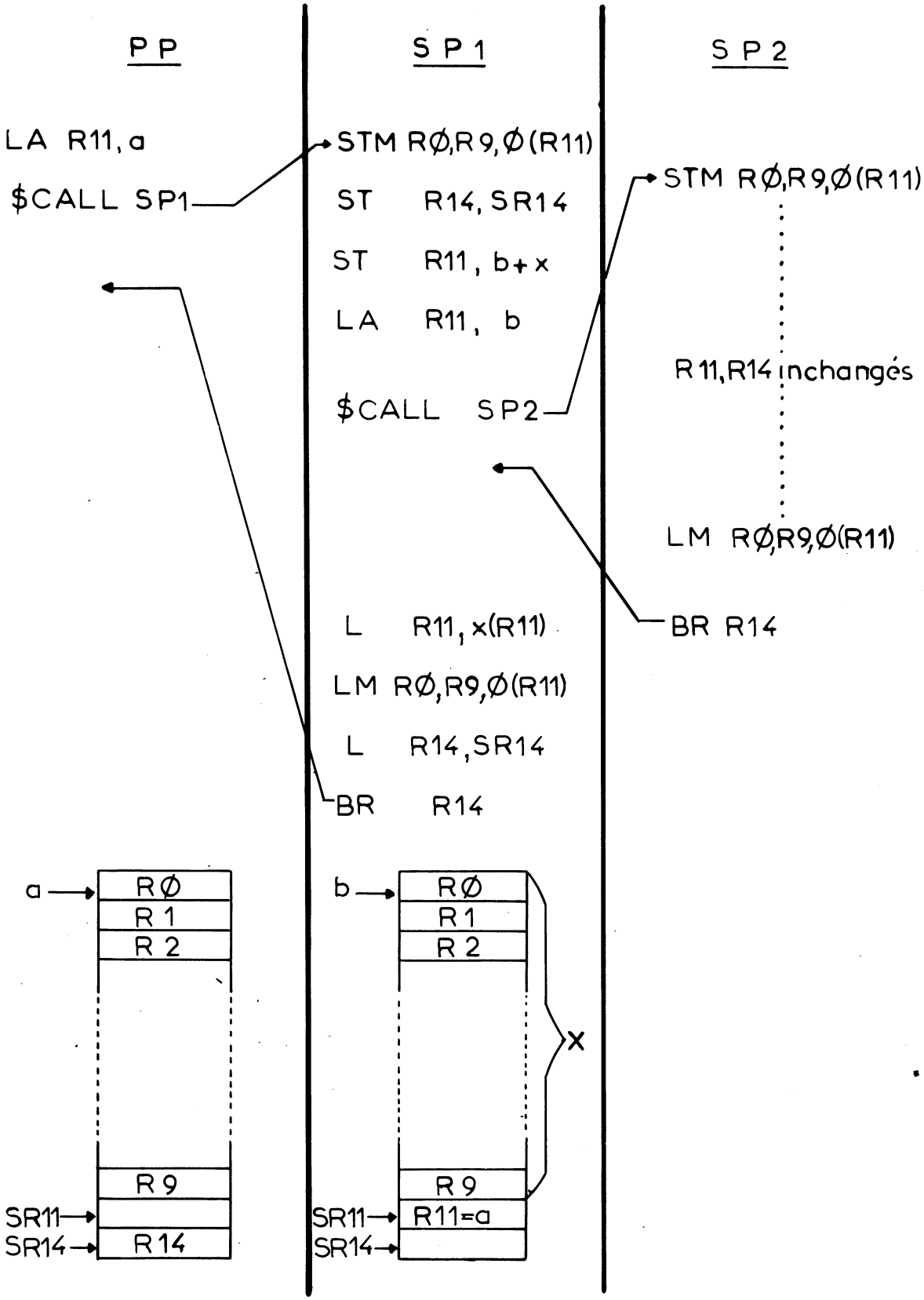


figure 3 liaison inter-section

Explicitons la liaison inter-section ne provoquant pas de changement de tâche sur un exemple. Nous adoptons la règle que l' appelant fournit une zone de sauvegarde à l' appelé, dans laquelle il n' est pas nécessaire de sauvegarder les registres R10 et R12 qui ne changent jamais de valeur ni le registre R13 puisqu' il n' y a pas changement de tâche.

Soit un programme principal PP qui appelle un sous-programme SP1 qui lui à son tour appelle SP2. SP2 lui n' appelle pas d' autre sous-programme et est dit terminal (voir figure 3). Pour SP2 il n' y a pas besoin de prévoir de zone de sauvegarde pour des niveaux inférieurs.

Il est intéressant de détailler ce qui se passe au niveau SP1. A l' entrée R11 contient l' adresse a de la zone de sauvegarde fournie par PP. Une zone de sauvegarde pour TELCOM a une longueur de 12 mots dont les 10 premiers servent à ranger dans l' ordre les registres R0,R1,...,R9 de l' appelant.

Le sous-programme SP1 range donc ces registres dans la zone fournie (STM R0,R9,0(R11)) et sauvegarde dans le dernier mot le contenu du registre R14 (ST R14,SR14), utilisé lors de l' appel de SP2. SP1 doit aussi préparer une zone de sauvegarde pour cet appel. Il acquiert, donc, cette zone à l' adresse b. De ce fait le registre R11 doit adresser b et non plus a; la sauvegarde nécessaire de l' adresse a est réaliser dans l' avant dernier mot de la zone de sauvegarde locale (ST R11,SAVE+x). Le registre R11 peut alors être chargé avec l' adresse b (LA R11,b).

Quand au registre R15 c' est la macro d' appel ($CALL) qui se charge de lui transmettre l' adresse d' entrée de la section de contrôle appelée (SP2), et lors du retour de le réinitialiser

avec l' adresse d' entrée de l' appelant (PP). Il n' est donc pas nécessaire de le sauvegarder.

Quand SP1 appelle SP2 le travail est beaucoup plus facile puisque SP2 n' appelle pas d' autres programmes. Il n' y a qu' à sauvegarder les registres R0 - R9 dans la zone fournie par l' appelant (à l' adresse b) ainsi que le registre R14 si nous voulons l' utiliser dans SP2 et ne pas utiliser le R11. Au retour nous restaurons les registres sauvegardés dans la zone adressée par R11.

Pour le retour au programme principal PP, SP1 doit recharger dans R11 l' adresse de la zone fournie par PP (instruction L R11,x(R11)) restaurer les registres R0 - R9 et R14 et effectuer le branchement de retour.

Ce mécanisme de liaison a été rendu systèmatique par l' exécution des macro-instructions suivantes:

a. A l' appel:

```
$SAVREG
```

qui a comme expansion :

```
STM    R0,R9,0(R11)
ST     R14,SR14
LR     R3,R1
$GETMAIN  SAVELEN
ST     R11,DSR11(R1)
LR     R11,R1
LR     R1,R3
```

b. Au retour:

```
$EXIT
```

qui a pour expansion :

```
LA      R0,SAVELEN
LR      R1,R11
L       R11,SR11
$FREMAIN
LM      R0,R9,0(R11)
L       R14,SR14
BR      R14
```

2.1.5. Problèmes dûs à la réentrance.

Chaque section de contrôle, pour assurer sa réentrance, doit acquérir sa zone de travail dynamiquement. Aussi, si elle désire appeler une autre section, elle doit acquérir une zone de sauvegarde. Cette fonction sera réalisée par appel au module de gestion de la mémoire libre GTMN, qui allouera une telle zone a toute section lui en faisant la demande. La section, donc, doit faire cet appel sans avoir déjà une zone de sauvegarde.

La solution proposée est que les sections de gestion de mémoire GTMN et FRMN ne sauvegardent plus les registres de l' appelant dans une zone acquise dynamiquement mais dans le bloc de contro^le de la tâche (TCB). Cette solution a pour conséquences:

a. GTMN et FRMN peuvent être appelés uniquement lorsqu' une ta^che est active.

b. Toute section qui n' appartient pas à une tâche (elle

appartient donc au superviseur) et qui nécessite une zone de sauvegarde ou de travail doit avoir une zone statique et doit, donc, se dérouler entièrement en mode masqué.

c. GTMN et FRMN doivent aussi se dérouler entièrement en mode masqué. et

d. FRMN qui appelle d' autres sections doit posséder une zone de sauvegarde statique.

Ces conséquences ne sont pas particulièrement lourdes, puisque GTMN et FRMN, primitives du système, ainsi que le superviseur se déroulent en mode masqué de par la conception du système. Ceci sera étudié dans le paragraphe 2.3. consacré à la description de la structure fonctionnelle du système.

Il reste comme conséquence dûe exclusivement à cette solution l' obligation pour toute section qui n' appartient pas à une tâche ainsi que pour FRMN d' avoir une zone statique, conséquence qui interdit le partage de ces modules par plusieurs utilisateurs en même temps.

2.2. NOTIONS DE BASE DU SYSTEME .

Dans la phase de conception du système lorsque nous en avons analysé les buts pour en déduire les principes et les nécessités sur lesquels fonder nos choix, il a été indispensable de bien définir les notions qui apparaissaient comme essentielles. Ces notions étaient déjà connues mais nous avons constaté une certaine divergeance dans leur définition et une certaine confusion dans leur utilisation .

2.2.1. La notion du "programme" .

Un programme est la description d' un algorithme dans un langage, une codification décrivant une séquence d' opérations. Comme telle la notion du programme est évidemment statique.

Les programmes du système TELCOM sont les sections de contro^le dont nous avons déjà fourni une première description.

2.2.2. La notion de "tâche" .

A la différence de la notion du "programme" la "tâche" est une notion dynamique. Elle est une activité, une entité qui est

exécutée ou peut être exécutée à un moment donnée par un processeur physique.

Nous remarquons tout de suite qu' une tâche peut utiliser différents programmes et processeurs pour s' exécuter, et que plusieurs tâches peuvent utiliser le même programme et le même processeur.

Une tâche peut se trouver dans différents états. Nous représentons les états possibles d' une tâche par l' arborescence suivante:

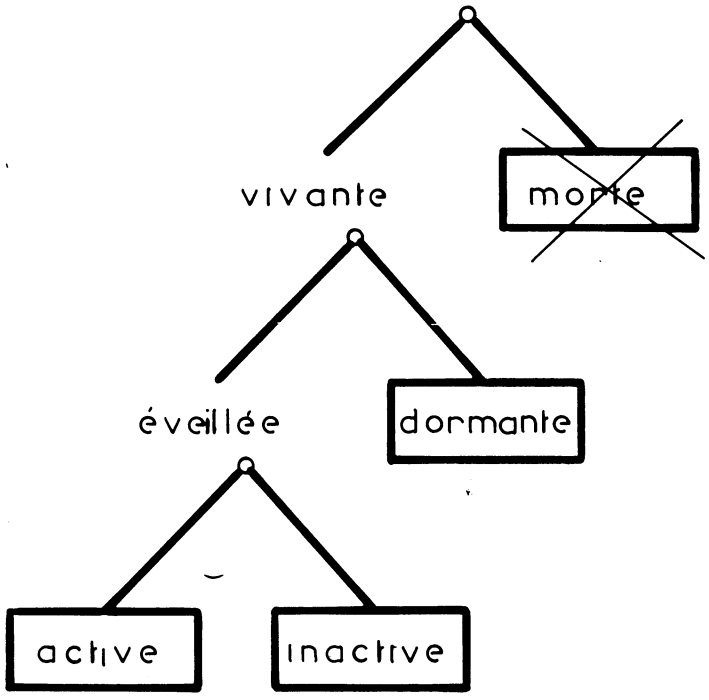


figure 4. états d' une tâche

a. Tâche "morte" : Une tâche se trouve dans cet état quand il n' y a aucune chance qu' elle soit exécutée.

Si elle n' est pas "morte" elle est "vivante".

b. Tâche "dormante" : Une tâche se trouve dans cet état si elle n' est pas candidate à s' exécuter. Il existe un obstacle logique à son exécution. Par exemple la tâche attend l' arrivée d' un événement.

Lorsqu' une tâche "vivante" n' est pas "dormante" elle est "éveillée".

c. Tâche "inactive" : Une tâche se trouve dans cet état quand elle est candidate à s' exécuter mais l' indisponibilité momentanée d' un processeur fait qu' elle ne s' exécute pas. Il n' y a aucune condition logique qui empèche son exécution.

Une tâche "éveillée" si elle n' est pas "inactive" est "active" (elle s' exécute).

A chaque tâche nous avons associé un contexte d' exécution comprenant:

-Le contenu des registres

-Le contenu du mot d' état du programme (P.S.W.)

-L' état de la tâche

-Son identification et

-Différents indicateurs.

Chaque fois qu' une tâche perd le contrôle (ne s' exécute pas) nous sauvegardons son contexte dans un bloc de contrôle associé aussi à la tâche (T.C.B.), pour l' utiliser au moment de sa réactivation.

2.2.3. La notion de "processeur" .

Un processeur est une entité matérielle qui exécute des ta^ches. Nous concevons, alors, comme processeurs l' unité centrale de contrôle (C.P.U.), les canaux et l' horloge.

Nous pouvons ici remarquer qu' il y a une différence entre les deux premiers types de processeurs. Les canaux ne peuvent exécuter une tâche que sur ordre du C.P.U.

Nous appelons le C.P.U. "processeur maître" et les canaux "processeurs esclaves". La communication entre eux s' effectue sous la direction du "maître" et est assurée dans le sens C.P.U.-canal par des instructions (SIO etc.) et dans l' autre sens par des interruptions.

Nous devons signaler ici que cette distinction est faite pour faciliter la compréhension du mécanisme, mais elle est arbitraire. On peut en effet considerer un IBM/360 comme un ensemble de processus de même hiérarchie mais dont certains, les canaux, viennent emprunter le 'hardware' du C.P.U. par le dispositif d' interruptions.

Au processeur maître nous donnons les moyens (superviseur du système) de pouvoir activer une tâche ou une autre. Le C.P.U. confie une partie d' exécution d' une tâche aux canaux.

Un processeur peut se trouver dans différents états qui sont représentés par l' arborescence suivante:

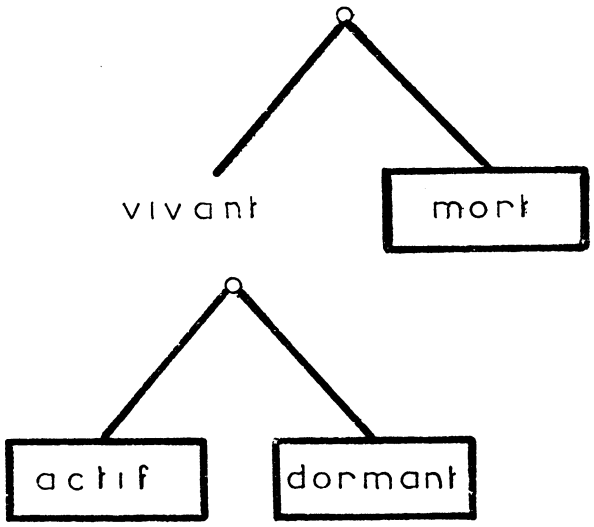


figure 5. états d' un processeur

Un processeur est "mort" quand il est en panne. S' il n' est pas "mort", il est "vivant" et peut être soit "actif" soit "dormant". Il est "actif" quand il exécute une tâche, et "dormant" quand il n' a rien à exécuter et attend (C.P.U. dormant = attente d' une interruption, canal dormant = attente d' une instruction d' entrée-sortie).

2.2.4. La notion d' "événement" .

Nous appelons "événement" un élément qui peut prendre deux valeurs: arrivé ou non, et qui peut conditionner l' exécution des tâches du système. Par exemple la libération d' une zone de mémoire, la fin d' exécution d' une opération d' entrée-sortie, l' élimination d' un élément d' une liste pleine e.t.c. sont des événements du système TELCOM.

A chaque événement nous associons une liste d' attente qui est constituée par les tâches qui attendent son arrivée.

La tâche ou la partie du superviseur qui détecte l' arrivée de l' événement peut:

a. Si des tâches sont en attente de l' événement:

-soit réveiller la première en attente (s' il n' y a qu' une tâche qui peut s' en servir, par exemple: libération d' une ressource non partageable et non partitionnable).

-soit toutes les tâches en attente (si plusieurs tâches ont des chances de s' en servir, par exemple: libération d' une zone de mémoire).

b. Si aucune tâche n' attend l' événement:

-soit ne rien faire, si aucune tâche n' a l' intention de l' attendre

-soit signaler son arrivée dans le cas contraire.

2.2.5. La notion des "primitives" .

A la notion d' événement décrite au paragraphe précédent nous pouvons associer un couple d' opérations qui serviront d' outils de coordination des tâches. Ces opérations:

WAIT (mise en attente d' un événement) et

POST (réveil d' une tâche après l' arrivée d' un événement).

sont appelées "primitives de coordination des tâches".

Il existe deux autres types de primitive, correspondant aux opérations à effectuer sur les ressources du système; d' une part:

ENQ (pour l' allocation d' une ressource) et

DEQ (pour sa libération)

sont des primitives d' allocation de ressources non partitionnables; et d' autre part:

GTMN (pour l' acquisition d' une zone de mémoire) et

FRMN (pour sa libération)

sont des primitives d' allocation de la ressource partitionnable "mémoire".

2.2.6. La notion de "section critique" .

Il existe, dans la vie des tâches, des moments (quand elles manipulent des ressources non partageables par exemple) où logiquement nous ne pouvons accepter que plusieurs tâches effectuent le même traitement en même temps. Nous disons à ce moment que la tâche qui effectue ce traitement est dans une "section critique" où nous sommes obligés d' assurer l' exclusion mutuelle. C' est à dire, qu' une tâche et une seule à la fois peut avoir accès à une "section critique".

2.2.7. La notion de "ressource" .

Nous définissons comme "ressource" toute entité nécessaire à l' exécution d' une ou plusieurs tâches. Par exemple le processeur, la mémoire, les unités d' entrée-sortie,une liste ou la section de données sont des ressources du système.

Une ressource peut être partageable ou non si respectivement plusieurs tâches peuvent s' en servir en même temps ou une et une seule.

Pour la gestion des ressources non partageables l' exclusion mutuelle des tâches est assurée.

Une autre distinction des ressources peut être faite au niveau des ressources partitionnables et non, selon qu' une tâche peut utiliser des morceaux de la ressource ou que la ressource est une entité inséparable.

2.2.8. La notion de "superviseur" .

Le "superviseur" est la partie du système qui assiste le processeur en activant une tâche ou une autre et en assurant les communications avec les canaux et l' extérieur.

Nous distinguons deux parties dans le superviseur du système TELCOM. La partie distributeur de contrôle ou module arbitre ou directeur (DISPAT) qui d' après certains critères choisit la tâche à activer, et les modules de traitement des interruptions qui désactivent la tâche active au moment d' une interruption et traitent les informations associées à celle-ci.

2.3. LA STRUCTURE FONCTIONNELLE DU SYSTEME .

Nous partons d' une analyse des fonctions que le système doit accomplir pour parvenir au choix des tâches et des conventions de son fonctionnement. Cette analyse fera aussi apparaître les contraintes dûes à l' environnement du système.

2.3.1. Les fonctions du système TELCOM.

TELCOM réalise l' interface entre le système CP67 et les lignes auxquelles sont connectés les terminaux lourds. Il doit, donc, assurer la transmission des informations des machines virtuelles vers les terminaux lourds et vice versa.

Explicitons la liaison avec un terminal lourd complet pour faire apparaître clairement les différentes fonctions du système (figure 6).

Sur la machine virtuelle TELCOM nous avons implanté le système TELCOM et nous lui avons attaché la ligne de communication avec le terminal lourd T1.

Le terminal lourd supporte un lecteur et un perforateur de cartes, une imprimante, la console opérateur et un certain nombre de télétypes. Nous supposons qu' il est aussi muni d' un système sachant utiliser la ligne de communication.

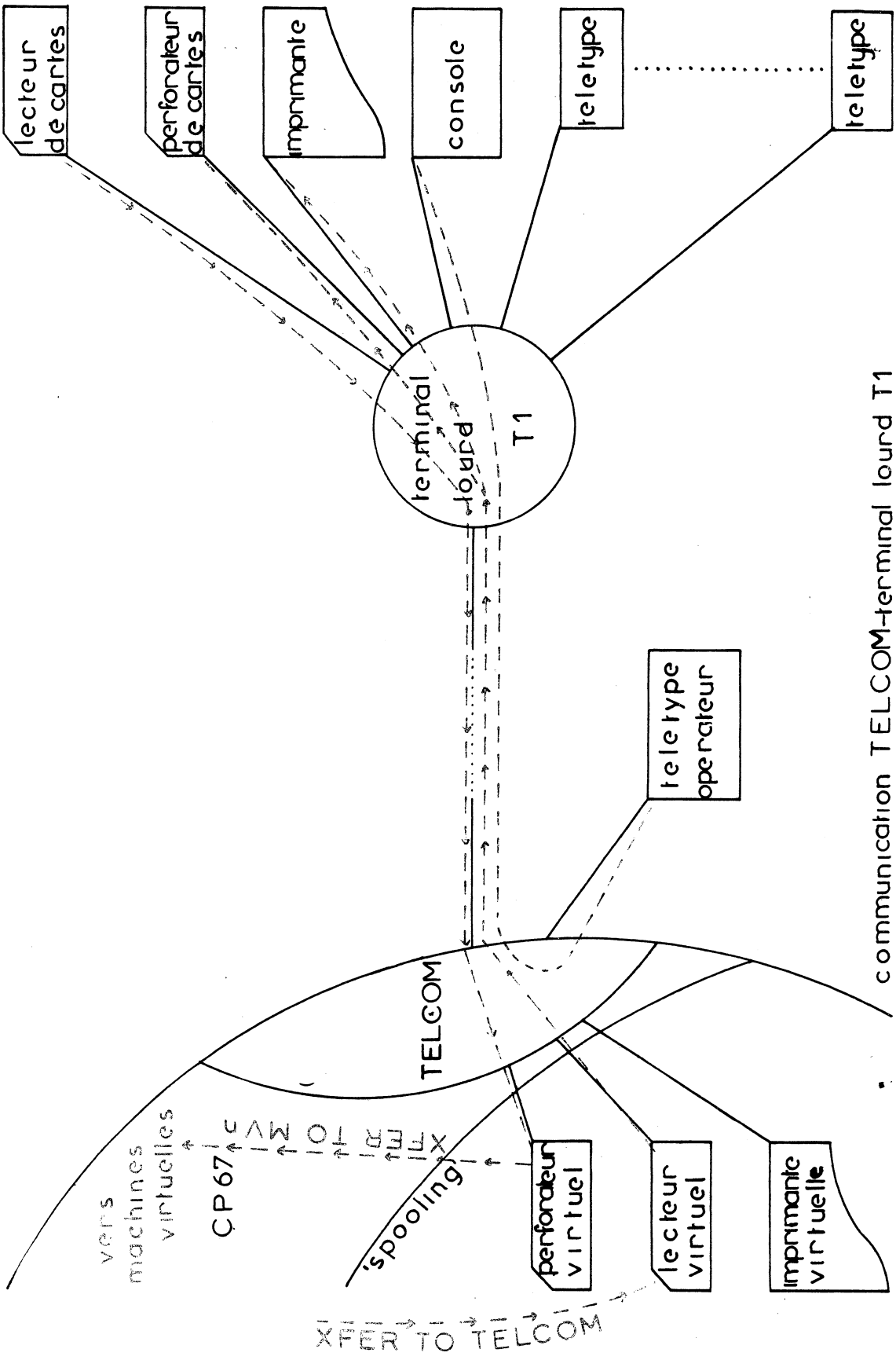

lecteur de cartes
perforateur de cartes
imprimante
console
teletype
teletype
terminal lourd T1
communication TELCOM-terminal lourd T1
teletype operateur
TELCOM
vers machines virtuelles
CP 67
MVT
TOT
RTM
XFM
spooling
perforateur virtuel
lecteur virtuel
imprimante virtuelle
XFER TO TELCOM

figure 6

De 1' autre côtè la machine TELCOM supporte des terminaux virtuels (lecteur, perforateur de cartes et imprimante) son télétype opérateur et la ligne de communication.

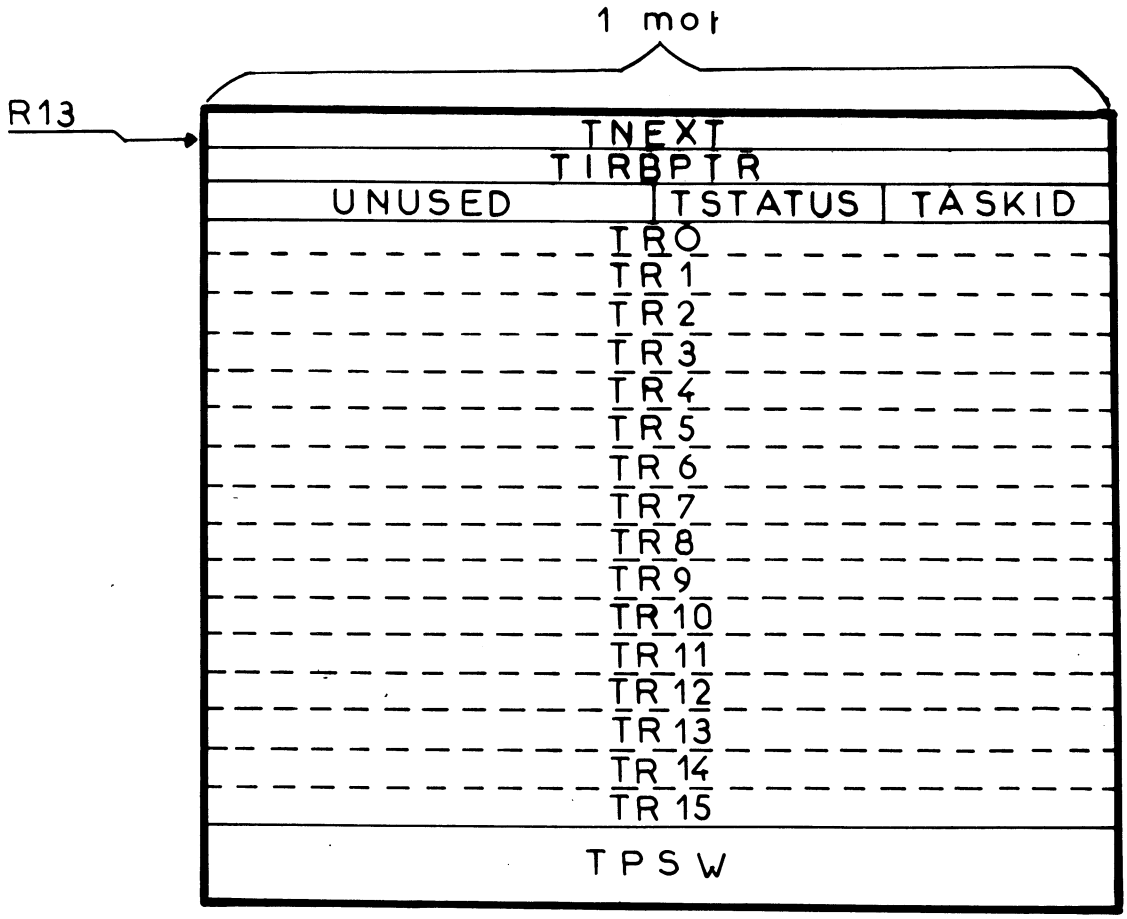
Un utilisateur du terminal lourd T1 met sur le lecteur du T1 un paquet de cartes destiné à une machine virtuelle M1 de CP. Le système local du terminal lourd lit les cartes et les transmet par la ligne à TELCOM. Le système TELCOM doit alors récupérer ces cartes arrivant sur la ligne, reconnaître leur destination (machine M1) et les y transférer (utilisant la fonction XFER de CP).

Dans l'autre sens, la machine virtuelle M1 désirant faire imprimer, par exemple, un fichier sur 1' imprimante de T1, transfére ce fichier à la machine TELCOM (XFER TO TELCOM). Le fichier arrive alors sur le lecteur virtuel de TELCOM. Le système doit le lire, reconnaître sa destination (imprimante de T1) et le transmettre sur la ligne correspondante.

Nous remarquons ici que pour des raisons d' économie d' utilisation des lignes et aussi de compatibilité avec des systèmes de communication très répandus (notamment le "multileaving") nous envoyons sur la ligne des informations compactées. Cela veut dire qu' il y a dans le système TELCOM une phase intermédiaire de compactage ou décompactage suivant la direction de 1' information.

Nous pouvons, donc, conclure que les fonctions du système sont :

- d' une part de gérer les unités d' entrée-sortie et
- d' autre part d' assurer le transfert des messages vers leur destination, leur compactage et décompactage.



TNEXT :adresse du TCB suivant

TIRBPTR :adresse de l'IRB

UNUSED :non utilisé

TSTATUS :état de la tâche (x'80' ready, x'40 waiting, x'20 interrupted, x'00 active)

TASKID :numero d'identification de la tâche

TR0÷TR15 :zone de sauvegarde du contenu des registres

TPSW :zone de sauvegarde du contenu du PSW en cas d'interruption

figure 7. T.C.B.

2.3.2. Choix des tâches du système.

La constatation que les principales fonctions du système sont les entrées-sorties nous a conduit à la solution de créer une tâche par unité d' entrée-sortie, plus deux autres tâches pour le compactage et le décompactage. Cette solution facilite largement la gestion des entrées-sorties résolvant dès sa création le problème d' allocation des unités, puisqu' il n' y a qu' une tâche qui accède à chacune des unités.

Nous avons choisi de permettre l' interruption des tâches et nous associons à chacune un contexte d' exécution et un bloc de contrôle de tâche (TCB) dans lequel nous sauvegardons son contexte au moment de sa désactivation. Le format de ce bloc apparaît dans la figure 7. Les TCB des tâches du système sont chaînés entre eux circulairement et se trouvent dans une section de contrôle (TCBS) pointée par un mot dans DATA (section de contrôle de données du système).

Tant que le système TELCOM est en marche nous n' acceptons pas pour ses tâches l' état "morte". Il ne reste donc que trois états possibles pour une tâche du système TELCOM:

active - inactive - dormante

2.3.2.1. Changement d' état des tâches.

Il existe deux primitives de synchronisation des tâches sur événement : WAIT et POST.

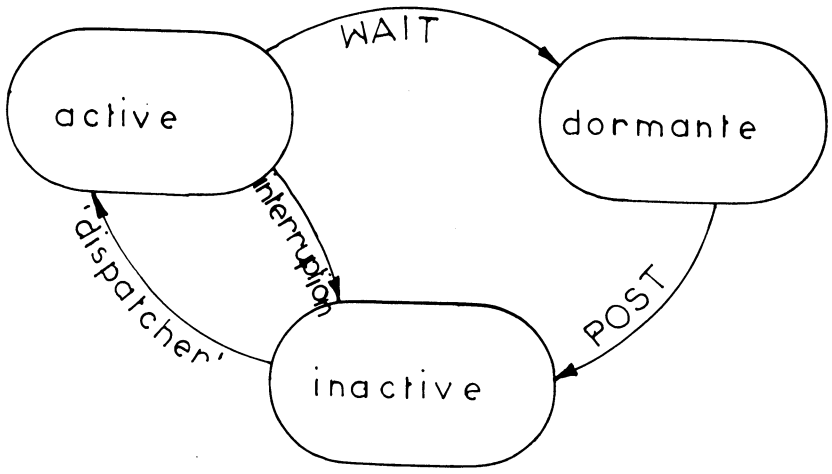
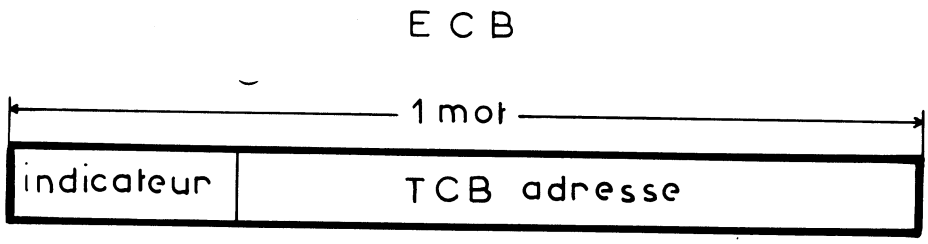


figure 8. changement d' état des tâches

Une tâche "active" qui appelle la primitive "WAIT" devient "inactive" et se met en attente par un ECB rangé dans une liste associée à l' événement. Une telle liste est donc, une file d' éléments (ECB) chacun correspondant à une tâche en attente.

Les ECB ont la forme suivante :



indicateur = x'80' : événement attendu

= x'40' : événement arrivé

figure 9

Le module qui détecte l' arrivée de l' événement appelle la primitive "POST" pour la tâche en attente. La primitive POST, donc, transforme une tâche "dormante" en "inactive".

Une interruption transforme la tâche "active" en "inactive". Et c' est la partie du superviseur qui est chargé de l' allocation du processeur aux tâches, le distributeur, qui entre les tâches "inactives" choisit la tâche à laquelle il alloue le processeur (la rend "active").

2.3.2.2. Fonctionnement des tâches du système.

Nous pouvons maintenant suivre dans les grandes lignes le fonctionnement des tâches du système (figure 10).

La tâche "lecteur" lit des enregistrements arrivés sur le lecteur virtuel de la machine TELCOM. S' il n' y en a pas elle se met en attente de l' événement "un enregistrement sur le lecteur". Supposons maintenant qu' un fichier arrive sur ce lecteur et soit destiné à l' imprimante du terminal T1. La tâche "lecteur" réveillée lit les enregistrements et les met dans une liste. La tâche "compacteur" prend des éléments de cette liste les compacte et les met dans une autre liste où la tâche "ligne" les prend pour les envoyer sur la ligne.

Dans l' autre sens la tâche "ligne" met les buffers arrivés de la ligne dans une liste, où la tâche "décompacteur" les prend, les décompacte et les met dans une autre liste, où une tâche "perforateur" les cherche et les envoie à la machine virtuelle à laquelle ils sont destinés (via XFER).

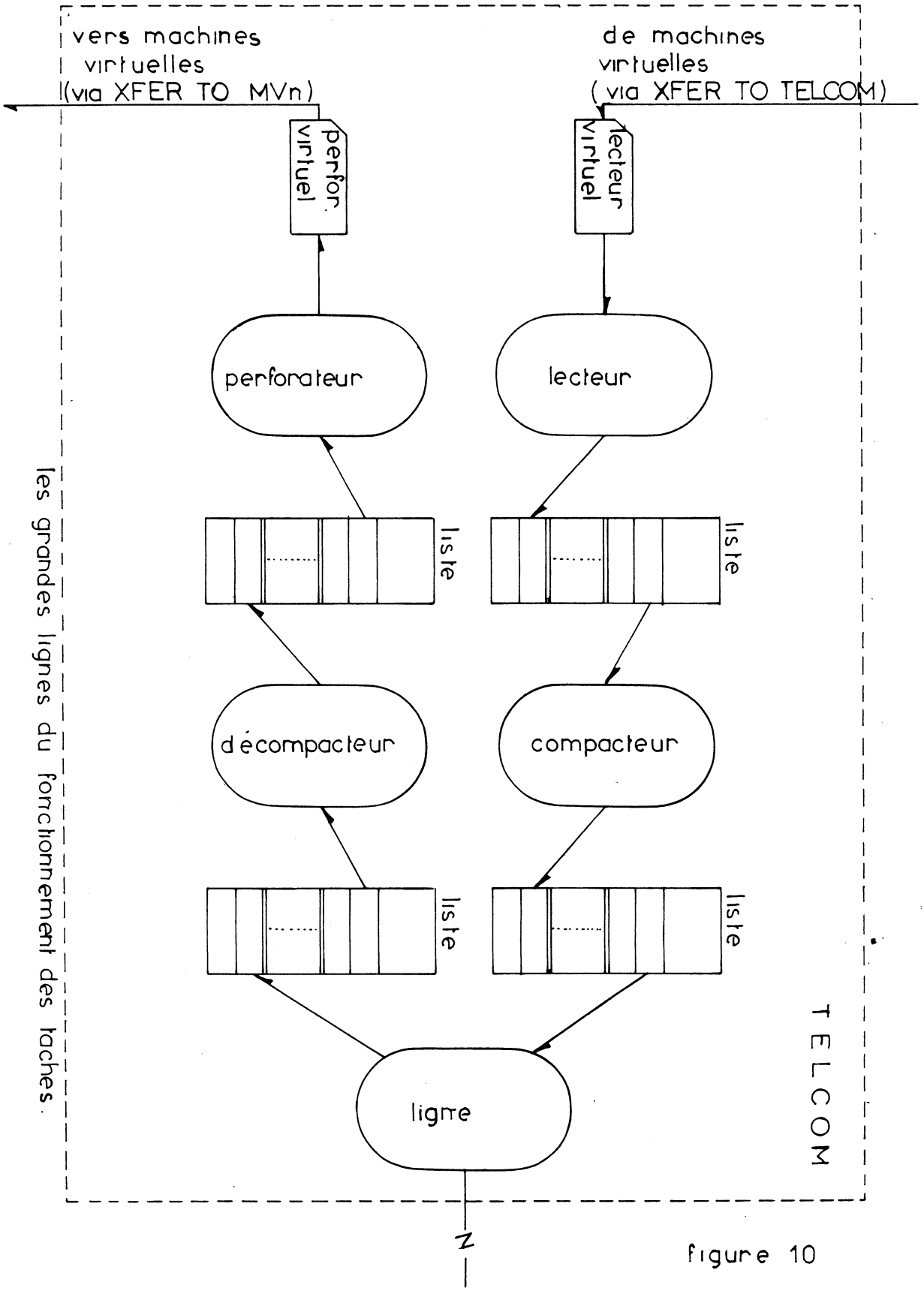

vers machines
virtuelles
(via XFER TO MVn)
de machines
virtuelles
(via XFER TO TELCOM)
perfor
virtuel
lecteur
virtuel
perforateur
lecteur
liste
liste
décompacteur
compacteur
liste
liste
ligne
N
les grandes lignes du fonctionnement des taches.
TELCOM

figure 10

Il est évident que ces listes sont des ressources du système et qu' une synchronisation des tâches du type producteur - consommateur sur elles est nécessaire.

2.3.3. Contraintes dûes à l' environnement du système.

Le système TELCOM est destiné à fonctionner sous CP67. Dans ce système chaque machine virtuelle est activée pour une tranche de temps (50ms). Si elle épuise sa tranche de temps elle est pénalisée par mise dans une file d' attente non prioritaire (parce qu' elle ne fonctionne pas en mode conversationnel). La machine TELCOM doit, donc, si possible ne pas épuiser sa tranche de temps entre deux attentes. Cette contrainte peut être satisfaite par le système TELCOM, qui généralement ne fait que des entrées-sorties, si ses tâches évitent les boucles en se mettant en attente des événements bien choisis.

Le premier principe, donc, du système TELCOM est:

"Le système TELCOM ne fait pas de long traitement, il est généralement en attente."

Une autre contrainte dûe à l' environnement CP67 sous lequel vit le système TELCOM, est qu' il ne peut pas utiliser la console principale de la machine TELCOM comme console opérateur. Chaque fois qu' on appuie sur la touche 'break' de la console principale, alors, CP67 désactive la machine attendant une commande sur la console. Nous étions, donc, obligés de créer une tâche "télétype" pour gérer un télétype attaché à la machine TELCOM (par DIAL) comme console opérateur du système.

2.3.4. Problèmes liés au fonctionnement du système et solutions adoptées .

Le fonctionnement du système a mis en évidence d' autres problèmes:

Un bloquage éventuel du système, par exemple, à cause d' une défaillance d' un terminal du côté terminal lourd. Explicitons le cas:

Supposons que l' imprimante du terminal lourd T1 soit en panne, et que des fichiers arrivent sur le lecteur TELCOM pour cette imprimante mais aussi pour d' autres destinations. Le terminal lourd T1 s' apercevant de la panne de son imprimante informe TELCOM en le priant de ne plus rien envoyer vers cette destination. TELCOM essayant d' arrêter le flot vers l' imprimante arrête son lecteur qui était en train de lire un fichier pour cette destination. C' est ainsi que tous les autres fichiers qui suivent, destinés peut être au perforateur de T1 ou même à une autre ligne sont bloqués.

Il faut remarquer, ici, que ce n' est pas seulement la panne d' un terminal mais aussi sa vitesse très faible qui peut provoquer de tels bloquages temporaires.

Pour remédier à cette situation nous avons utilisé la possibilité de déclarer sous CP67 plusieurs lecteurs virtuels. Le principe dans ce cas est: un nouveau fichier arrive sur le premier lecteur libre. C' est à dire, un lecteur qui n' a pas été utilisé ou qui a été fermé après la lecture du dernier enregistrement d' un fichier.

Ainsi, si des fichiers pour l' imprimante défaillante

bloquent quelques lecteurs, il en reste, tout de même, de libres pour accepter des fichiers pour d' autres destinations.

Il ne reste que le cas extrême où tous les lecteurs sont bloqués par des fichiers pour des destinations bloquées. Dans ce cas nous n' avions que deux solutions: soit ignorer les fichiers, soit les envoyer au terminal homologue local. Nous avons choisi la deuxième solution.

Un autre problème qui apparaît maintenant est le problème de la confusion entre plusieurs fichiers. Supposons qu' en même temps deux machines virtuelles envoient des fichiers pour la même destination (par exemple l' imprimante de T1). Les fichiers arrivent sur des lecteurs différents et sont traités parallèlement. Il serait, donc, possible que sur l' imprimante de T1 arrivent des enregistrements de l' un ou l' autre fichier alternativement, si on n' y prend garde.

La solution adoptée pour pallier cet inconvénient est basée sur la notion de "voie virtuelle". C' est à dire, pour toute destination possible nous créons une voie virtuelle contenant un lecteur, la liste où il met les enregistrements lus, la liste où ses enregistrements sont mis après compactage, l' envoi sur la ligne et le terminal de destination. Ces voies virtuelles constituent des ressources du système.

Pour qu' un lecteur commence à lire un fichier pour une destination donnée il doit demander l' allocation de la ressource: "voie virtuelle correspondante à la destination". Simultanément un autre lecteur voulant lui aussi transmettre vers la même destination, et demandant l' allocation de la même ressource se mettra en attente de sa libération. Le premier lecteur libèrera, bien sûr, la ressource quand il aura transmis

le dernier enregistrement du fichier.

Pour réaliser les solutions adoptées en fonction de ces problèmes nous étions amenés à créer des listes distinctes d' éléments par destination, ce qui a une influence sur le fonctionnement de la tâche "compacteur". Cette tâche a maintenant un certain nombre (égal au nombre des destinations possibles) de listes d' éléments en entrée (à compacter) et le même nombre de listes en sortie (éléments compactés à envoyer sur la ligne). Son fonctionnement normal doit être le suivant: chercher des éléments dans chaque liste d' entrée et s' il y en a les compacter dans la liste de sortie correspondante, sinon parcourir les autres listes circulairement. Mais pour être fidèle au principe du système elle doit se mettre en attente d' un événement bien choisi. La première idée était de choisir l' événement composé: "un élément dans n' importe quelle liste d' entrée". Il faut néanmoins tenir compte du fait que la tâche "compacteur" n' effectuant pas d' entrée-sortie devient ultra-rapide devant les autres. Il ne serait, donc, pas logique de la réveiller chaque fois qu' un élément seulement apparaît dans une de ses listes d' entrée. La solution retenue est une attente sur l' écoulement d' un intervalle de temps. C' est à dire, que toutes les fois que la tâche compacteur ne trouve rien dans ses listes d' entrée, elle se met en attente pour un intervalle d' une seconde (solution qui a donné des résultats satisfaisants).

Dans l' autre sens, enfin, c' est-à-dire dans la direction: lignes - machines virtuelles il y a un problème semblable. Les buffers qui arrivent sur les lignes sont destinés à différentes machines virtuelles et la tâche "décompacteur" après le décompactage les met dans une liste pour le perforateur virtuel.

Celui-ci demande au CP67 un transfert vers la machine virtuelle de destination (par la commande XFER) et perfore les enregistrements. Mais pendant la perforation d' un fichier qui arrive petit-à-petit d' une ligne tous les autres fichiers arrivés des autres lignes sont bloqués, puisque nous ne pouvons pas demander plusieurs XFER du même perforateur virtuel.

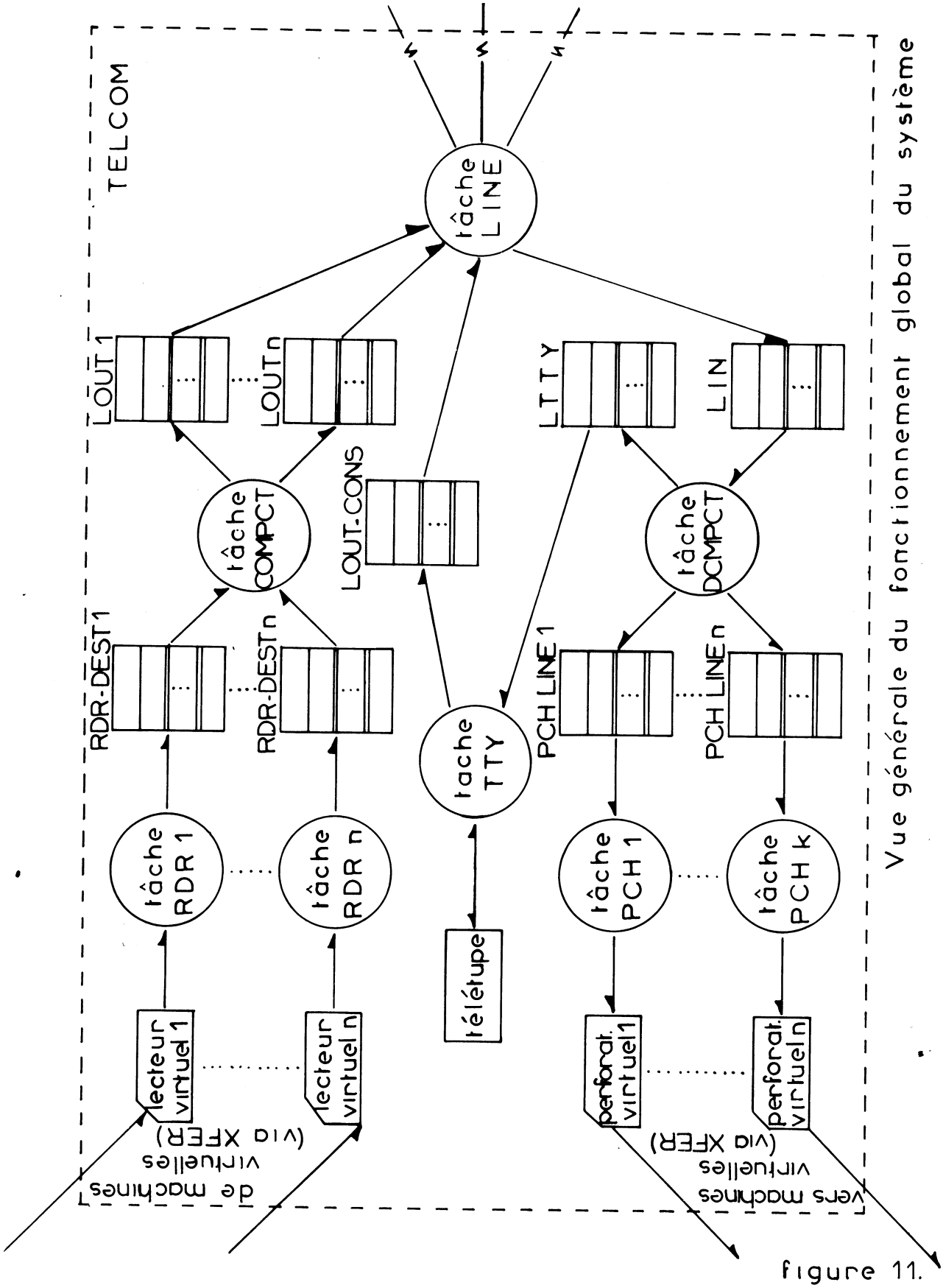
La solution retenue, ici, est analogue à celle choisie pour l' autre sens: nous utilisons la possibilité de déclarer plusieurs perforateurs virtuels et nous avons affecté chacun à une ligne de communication.

Ainsi, les fichiers arrivés sur une ligne peuvent être canalisés vers leur destination via perforateur affecté à cette ligne.

2.3.5. Vue générale du fonctionnement global du système (figure 11) .

Nous pouvons maintenant décrire le fonctionnement global du système en suivant le passage d' un "job" soumis par un utilisateur distant.

Cet utilisateur met son paquet de cartes sur le lecteur du terminal lourd T1 (un de terminaux supportés par TELCOM; figure 6). Il doit mettre en tête une carte de contrôle indiquant la destination de son "job"; par exemple: la carte: ID MV1 qui indique comme destination la machine MV1. Son paquet de cartes est lu et arrive (compacté ou non selon les conventions du terminal lourd) par la ligne à TELCOM.



Vue générale du fonctionnement global du système

figure 11.

La tâche "LINE", activée, lit les informations arrivées et les met dans la liste LIN. La tâche "DCMPCT" (décompacteur) cherche les éléments dans la liste LIN, les décompacte si nécessaire et les met dans la liste du perforateur virtuel associé à la ligne d' arrivée, par exemple dans la liste PCH-LINE1. Enfin, la tâche "PCH1" (tâche perforateur correspondant à cette liste) prend les éléments de sa liste, et trouvant la première carte (ID MV1) demande à CP67 le transfert à la machine MV1 (par un: XFER adr TO MV1, adr étant 1' adresse du perforateur). Elle perfore ensuite les cartes qui suivent, et ainsi le fichier lu au début sur le lecteur du terminal lourd T1 se trouve maintenant sur le lecteur de la machine virtuelle MV1.

Le job s' exécute sur la machine MV1 et à un moment celle-ci décide d' envoyer un fichier (résultats) sur 1' imprimante de T1 (ou une des imprimantes de T1, s' il y en a plusieurs). Elle doit d' abord mettre en tête de ce fichier un enregistrement indiquant la destination; par exemple: T1 PRINT (qui indique destination: imprimante de T1 - s' il y en a plusieurs nous pouvons 1' indiquer par PRINT1,...,PRINTn). Ensuite elle effectue un XFER vers la machine TELCOM et le fichier arrive sur un lecteur virtuel libre de TELCOM.

La tâche "RDR" associée à ce lecteur lit, alors, le premier enregistrement du fichier arrivé et ayant reconnu la destination, demande 1' allocation de la ressource "voie virtuelle vers cette destination". Si elle n' est pas libre, ce qui veut dire qu' une autre machine virtuelle est en train d' écrire sur 1' imprimante de T1, elle attend. Quand la voie se libère, la tâche RDR lit le fichier et met ses enregistrements dans la liste d' entrée du

compacteur associée à la destination (la liste RDR-DEST1). La tâche compacteur ensuite les compacte et les met dans la liste de ligne associée aussi à la destination (LOUT1), d' où la tâche "LINE" les cherche pour les envoyer au terminal lourd T1.

Dans ce cycle entier du fonctionnement du système nous voyons la grande importance de la coordination des tâches dans l' utilisation des ressources du système , par exemple des différentes listes et des voies virtuelles. Nous abordons ce problème dans le paragraphe suivant.

2.4. GESTION DES RESSOURCES. SYNCHRONISATION DES TACHES.

Nous étudions, dans ce paragraphe, les différents types de ressources du système TELCOM, le problème d' allocation et les primitives de synchronisation des tâches dans l' acquisition de ces ressources.

Pour réaliser l' exclusion mutuelle de tâches quand elles utilisent des primitives du système, nous avons choisi que toute primitive se déroule en mode masqué (toute interruption interdite), de sorte qu' une tâche exécutant une primitive ne perd jamais le contrôle.

Suivant l' optique première du système qui est sa simplicité nous avons choisi de synchroniser les tâches par des événements et le jeu des primitives WAIT et POST; la solution des sémaphores aurait été plus coûteuse.

2.4.1. La ressource "processeur" .

Nous considérons le processeur qui exécute les tâches de notre système, comme une ressource, et c' est le distributeur ou module arbitre, partie du superviseur du système, qui se charge de son allocation.

2.4.1.1. Le "distributeur" .

Après l' initialisation du système le contrôle est passé au distributeur (module DISPAT). Toutes les tâches sont initialisées et se trouvent dans l' état "inactive"; c' est-à-dire, elles sont candidates à s' exécuter. Les TCB des tâches sont chaînés entre eux circulairement. Le distributeur cherche dans cette chaîne la première tâche "inactive" et l' active.

Quand le système fonctionne, le distributeur peut prendre le contrôle dans deux cas:

a. après toute interruption qui désactive la tâche active et

b. toutes les fois qu' une tâche active attendant un événement devient "dormante".

Il se déroule entièrement en mode masqué étant partie du superviseur du système.

Dans le système TELCOM il n' existe pas de priorité entre les tâches. C' est une convention de simplicité qui est parfaitement justifiée pour un système généralement en attente. Pour réaliser cette 'non priorité', le distributeur commence la recherche dans la chaîne de TCB par le TCB qui suit celui de la dernière tâche active.

Quand le distributeur trouve une tâche inactive, il examine si elle avait perdu le contrôle par interruption ou en se mettant en attente. Dans le premier cas il lui rend le contrôle en utilisant l' ancien PSW sauvegardé au moment de l' interruption dans le TCB, autrement il utilise le registre R14 sauvegardé aussi dans le TCB.

S' il ne trouve aucune tâche "inactive", il se met en

attente. Tout le système est alors en attente et il sera réveillé par interruption.

2.4.1.2. La primitive "WAIT" .

Une tâche obligée d' attendre un événement appelle la primitive WAIT ayant mis dans le registre R1 l' adresse d' un ECB associé à cet événement et à cette tâche.

"WAIT" regarde si l' événement est déjà réalisé (indicateur "événement arrivé" dans l' ECB positionné). Si c' est le cas il rend la tâche "inactive" et détruit l' ECB, sinon il rend la tâche "dormante" et positionne le drapeau "événement attendu". Dans les deux cas il passe ensuite le contrôle au distributeur.

Par souci de garder toujours le système TELCOM le plus simple possible nous utilisons seulement cette primitive de WAIT simple; c' est à dire, nous adoptons la convention qu' une tâche ne peut attendre qu' un événement à la fois.

L' arrivée d' un événement oblige le module qui la détecte à réveiller les tâches éventuellement en attente de cet événement. Ceci est fait par appel de la primitive POST qui comme WAIT est simple. C' est-à-dire, qu' elle ne peut réveiller qu' une tâche à la fois. Si plusieurs tâches attendent le même événement, nous avons déjà indiqué que l' événement est lié à une liste d' ECB chacun associé à une tâche. Le détecteur est, donc, obligé d' appeler POST autant de fois qu' il y a de tâches en attente.

2.4.1.3. La primitive "POST" .

POST positionne 1' indicateur "événement arrivé" et si 1' événement était attendu réveille la tâche (effectue la transition "dormante" - "inactive"), détruit 1' ECB et revient à 1' appelant. Sinon il revient directement après avoir positionné 1' indicateur.

2.4.1.4. Evénements mémorisés ou non .

L' arrivée d' un événement est significative quand il y a des tâches qui 1' attendent déjà ou des tâches qui se mettront en attente de son occurence plus tard. Dans ce dernier cas nous devons mémoriser son arrivée. Par exemple, 1' arrivée du premier élément dans une liste vide, n' est pas significative si aucune tâche n' a pas encore cherché d' élément dans la liste vide. Elle est au contraire significative quand des tâches ont cherché et sont mises ou se mettront en attente. Pour les tâches qui au moment de 1' arrivée de 1' événement sont déjà en attente il n' y a pas de problème, le détecteur va appeler la primitive POST pour elles. Mais pour les autres tâches, qui ont leurs ECB (dans la liste des ECB associée à 1' événement) à zéro il y a deux possibilités:

-soit qu' elles n' ont pas encore cherché la liste (1' événement n' est pas significatif)

-soit qu' elles ont cherché mais ne sont pas encore mises en attente (1' événement est significatif).

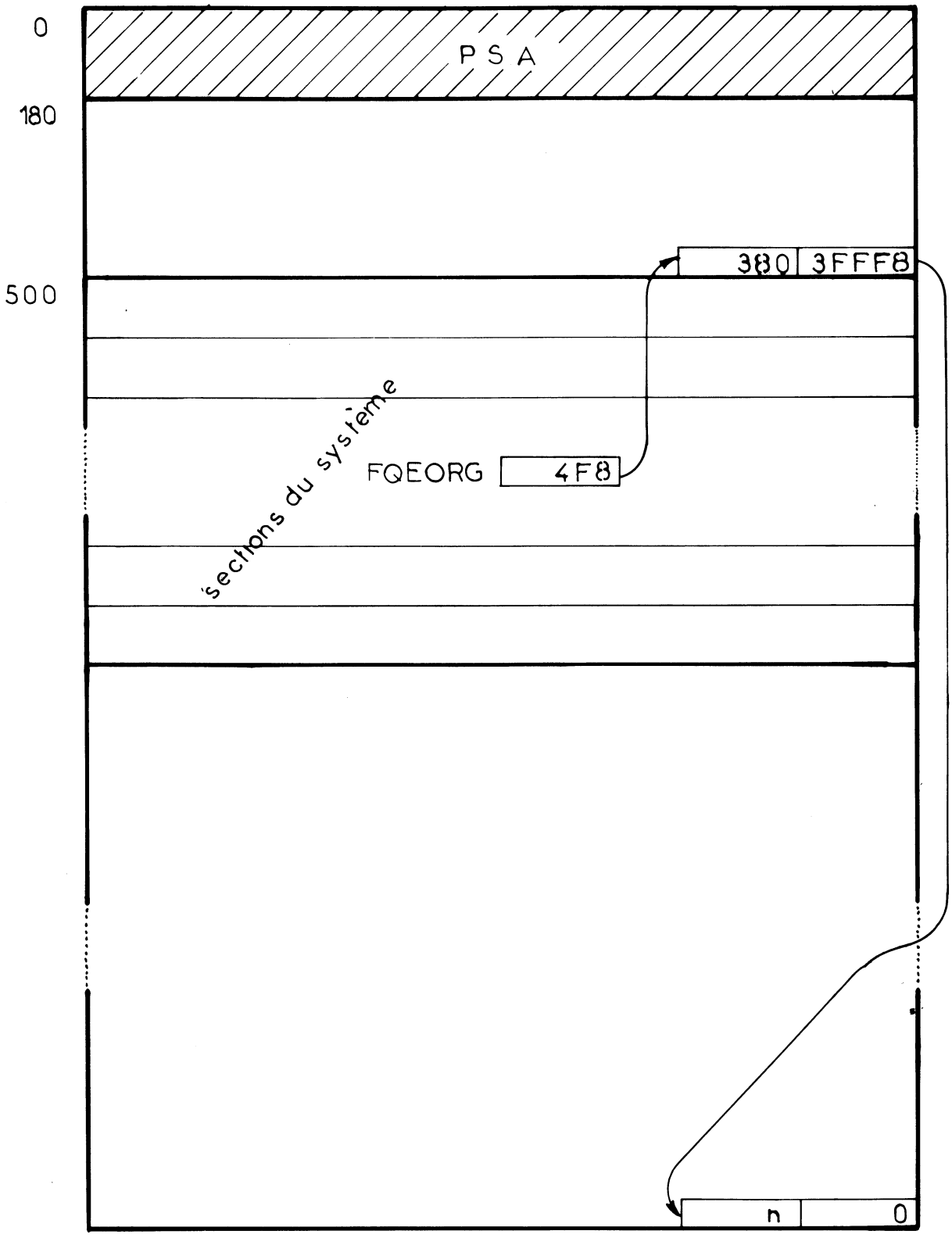


figure 12. image de la mémoire après le chargement du système.

Dans le dernier cas l' événement doit être mémorisé.

Nous avons adopté la règle suivante pour systématiser la mémorisation des événements quand il le faut. Une tâche qui veut qu' un événement soit mémorisé pour elle, doit mettre l' adresse de son TCB dans l' ECB associé à l' événement et à la tâche.

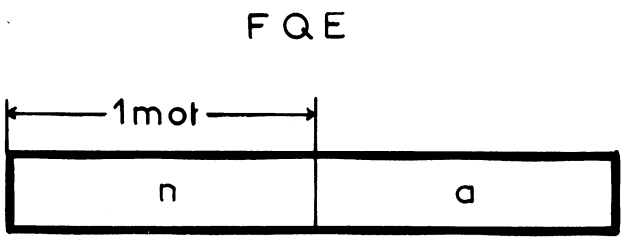
Ainsi les ECB pour des événements à mémoriser ne sont plus à zéro et le détecteur d' arrivée est obligé d' appeler POST pour les tâches correspondantes.

2.4.2. La ressource "mémoire" .

Après le chargement du système, la mémoire a la configuration représentée à la figure 12.

La mémoire libre est celle qui suit les sections de contrôle du système et aussi la mémoire qui était occupée par le chargeur, dont le système n' a plus besoin et qu' il a récupéré. Pour garder trace des différentes zones de la mémoire libre nous adoptons la convention suivante:

Pour chaque zone de mémoire libre nous utilisons le dernier double-mot pour indiquer le nombre d' octets libres et l' adresse de l' indicateur de la zone suivante. Un tel indicateur (FOE) a, donc, la forme suivante:



n = nombre d' octets libres

a = adresse de l' FQE suivant

figure 13

Dans la section de contrôle de données "DATA" nous avons réservé le mot FQEORG, qui contient toujours l' adresse du premier FQE.

La gestion de la mémoire libre se fait par les primitives GTMN et FRMN (deux sections de contrôle du système), respectivement pour la demande et la libération d' une zone de mémoire. Nous suivons pour cette gestion deux principes:

a. Nous n' acceptons que des zones de mémoire multiples de double-mot. C' est à dire, une tâche peut demander un nombre quelconque d' octets, mais les primitives GTMN et FRMN arrondissent au plus petit multiple de 8 supérieur ou égal au nombre demandé.

b. Nous n' accordons que de la mémoire contigüe. C' est à dire, si une zone de mémoire libre n' est pas suffisante pour satisfaire une demande nous examinons la suivante.

A 1' événement "libération d' une zone de mémoire" nous avons associé une liste d' ECB pouvant contenir autant d' ECB qu' il y a de tâches dans le système. Dans cette liste chaque tâche a sa propre place (ECB) suivant son numéro d' identification. C' est sur cet ECB qu' elle se mettra en attente quand elle ne pourra pas obtenir une zone de mémoire et elle sera réveillée après la première libération de mémoire.

2.4.2.1. La primitive "GTMN" .

Nous avons déjà parlé du problème de sauvegarde des registres de 1' appelant. En effet, 1' appelant pour acquérir sa zone de sauvegarde doit appeler GTMN. Solution adoptée: GTMN sauvegarde les registres de 1' appelant dans le TCB de la tâche active. Conséquence: la tâche doit rester active pendant le déroulement de la primitive GTMN. Les interruptions étant interdites, il ne faut pas que GTMN appelle "WAIT", seule autre possibilité pour la tâche de quitter 1' état "active". Si la demande ne peut pas être satisfaite et que la tâche doive se mettre en attente, GTMN revient à 1' appelant avec un code de retour: "demande non satisfaite" et c' est, alors, 1' appelant qui appelle "WAIT". Tout cela devient systématique par 1' utilisation de la macro $GETMAIN qui accepte un seul paramètre (le nombre d' octets demandé). Exemple:

```
EXEMPLE  $GETMAIN  60
```

a comme expansion :

```
EXEMPLE  DS     OH
Lln      DS     OH
         LA     Rl,60     met dans Rl le nombre d' octets
         $CALL  GTMN      appel de GTMN
         LTR    Rl,Rl     test si demande satisfaite?   }
         BP     L2n       oui, continuer                } Z
         LPR    Rl,Rl     non, ECB adresse dans Rl      }
         $CALL  WAIT      appel de WAIT
         B      Lln       après POST essayer de nouveau
L2n      DS     OH        sortie, demande satisfaite.
```

Le module GTMN commence par examiner la file des FQE en partant du premier dont l' adresse se trouve dans le mot FQEORG de DATA. S' il trouve suffisamment de place pour satisfaire la demande, il réserve cette place en modifiant l' indicateur FQE correspondant et il revient à l' appelant après avoir mis dans Rl l' adresse de la zone allouée.

Si au contraire la demande ne peut pas être satisfaite, GTMN cherche dans la liste des ECB associée à l' événement: "libération d' une zone de mémoire", l' entrée correspondante à la tâche active (suivant son numero d' identification). Comme GTMN ne peut pas appeler WAIT, nous sommes ici devant un cas précis d' un événement qui doit être mémorisé (c' est la zone Z d' instructions qui nécessite la mémorisation de l' événement). GTMN, alors, met dans l' ECB correspondant à la tâche active, l' adresse de son TCB demandant ainsi, la mémorisation de l' événement jusqu' au moment où la tâche à l' extérieur de GTMN effectuera l' appel de WAIT. Dans ce cas GTMN met dans Rl l' adresse de l' ECB (utile pour l' appel de WAIT qui suivra), rend

RI negatif pour indiquer "demande non satisfaite" et revient à l' appelant.

2.4.2.2. La primitive "FRMN" .

Une tâche désirant libérer une zone de mémoire doit appeler la primitive FRMN.

Pour les mêmes raisons que GTMN, FRMN sauvegarde les registres de l' appelant dans le TCB de la tâche active. En plus étant obligé d' appeler POST pour les tâches en attente de libération d' une zone de mémoire, elle est munie d' une zone de sauvegarde statique.

L' appelant doit fournir à FRMN le nombre d' octets et l' adresse de la zone à libérer, respectivement dans les registres R0 et R1.

FRMN commence par comparer l' adresse de la zone avec les adresses des FQE pour la cadrer par rapport aux autres zones déjà libres; Ainsi si la zone est voisine avec une autre il en crée une nouvelle, somme de deux, modifiant les FQE. Sinon il insére un nouveau FQE dans la file pour la zone libérée. Ensuite, il cherche dans la liste des ECB et réveille toutes les tâches en attente de mémoire libre, ou avant demandé la mémorisation de l' événement, et revient enfin à l' appelant.

Nous devons remarquer, ici, que FRMN réveille toutes les tâches interessées à l' événement libération d' une zone de mémoire car il peut libérer assez de mémoire pour satisfaire plusieurs demandes.

2.4.3. Les autres ressources du système .

Pour la gestion des autres ressources du système (mises à part le "processeur" et la "mémoire") nous utilisons les primitives ENQ et DEQ, respectivement pour acquérir et libérer une ressource.

Nous avons ordonné les ressources et construit une table (TABRES, un double-mot par ressource) dans laquelle à chaque ressource, dans l' ordre, correspond une entrée. Le premier octet indique si la ressource est occupée (X'80') par une tâche ou non (X'00'). Le reste du premier mot contient l' adresse du TCB de la tâche occupant la ressource, s' il en existe une. Le deuxième mot contient un pointeur vers un ECB associé à une tâche éventuellement en attente de cette même ressource. Etant donné qu' une tâche ne peut attendre qu' un événement à la fois, nous avons construit une table des ECB (RESECB) avec un nombre d' entrées egal au nombre maximum des tâches. Chaque entrée de cette table est un double-mot, le deuxième mot servant à contenir un pointeur vers un ECB d' une autre tâche éventuellement attendant la même ressource, et en premier octet le numéro de la ressource attendue.

Ainsi, les listes des ECB associées à chaque événement: "libération d' une ressource" se trouvent dans cette table chaque ECB étant chaîné à l' ECB suivant du même événement (figure 14).

A noter que pour rendre le travail des primitives ENQ et DEQ plus aisée nous avons adopté comme nom de chaque ressource son numéro d' ordre dans la table de ressources (TABRES).

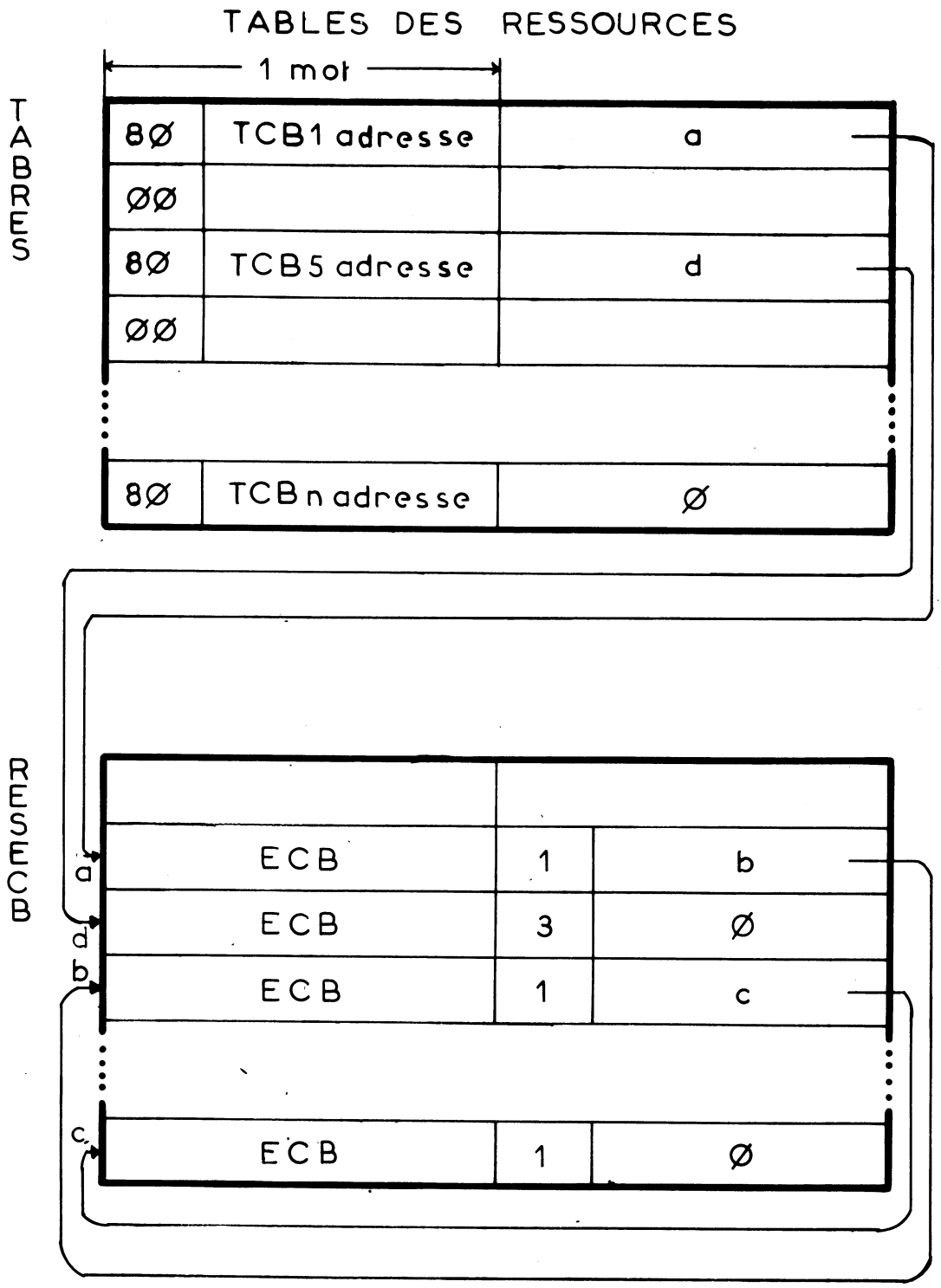

TABLES DES RESSOURCES
1 mot
TABRES
8Ø
TCB1 adresse
a
ØØ
8Ø
TCB5 adresse
d
ØØ
8Ø
TCBn adresse
Ø
RESECB
a
d
b
c
ECB
1
b
ECB
3
Ø
ECB
1
c
ECB
1
Ø

figure 14

2.4.3.1. La primitive "ENQ" .

Une tâche ayant besoin d' une ressource appelle la primitive ENQ avec pour paramètre (dans R1) le nom de la ressource (son numéro d' ordre dans la table des ressources TABRES).

Il existe deux possibilités d' appel suivant que l' on désire ou non se mettre en attente si la ressource est occupée. Le registre R1 est respectivement positif ou négatif selon le cas.

La primitive ENQ examine tout d' abord l' entrée correspondante à la ressource dans la table TABRES. Un test sur le premier octet de cette entrée nous permet de constater si la ressource est occupée (octet = X'80') ou non (octet = X'00').

Si la ressource est occupée et à l' appel R1 était négatif (la tâche ne veut pas se mettre en attente) alors ENQ revient à l' appelant avec R1 différent de zéro; si à l' appel R1 était positif ENQ cherche dans la liste des ECB l' entrée correspondant à la tâche et la chaîne à la file des ECB des tâches qui attendent déjà cette ressource (figure 14).

ENQ se déroulant en mode masqué ne peut pas appeler WAIT. Il doit, donc, demander la mémorisation de l' événement. Il met, alors, dans l' ECB l' adresse du TCB de la tâche, restaure le masque et revient à l' appelant ayant mis dans R1 l' adresse de l' ECB (utile pour l' appel de WAIT qui suivra).

Si par contre la ressource était libre (premier octet = X'00'), ENQ signale qu' elle est maintenant occupée (octet forcé à X'80'), met dans l' entrée du TABRES l' adresse du TCB de la tâche, restaure le masque et revient à l' appelant avec R1=0.

2.4.3.2. La primitive "DEQ" .

La tâche désirant libérer une ressource qu' elle occupe doit appeler la primitive DEQ, passant comme paramètre dans le registre R1 le nom de la ressource.

La primitive DEQ cherche dans la table des ressources l' entrée de la ressource en question, vérifie qu' elle était bien occupée par la tâche appelant et la libère (premier octet = X'00').

Puis elle regarde s' il y avait des tâches en attente de cette ressource et si oui elle poste la première lui allouant la ressource (modifie respectivement les pointeurs dans TABRES et RESECB).

Enfin elle restaure la masque et revient à l' appelant.

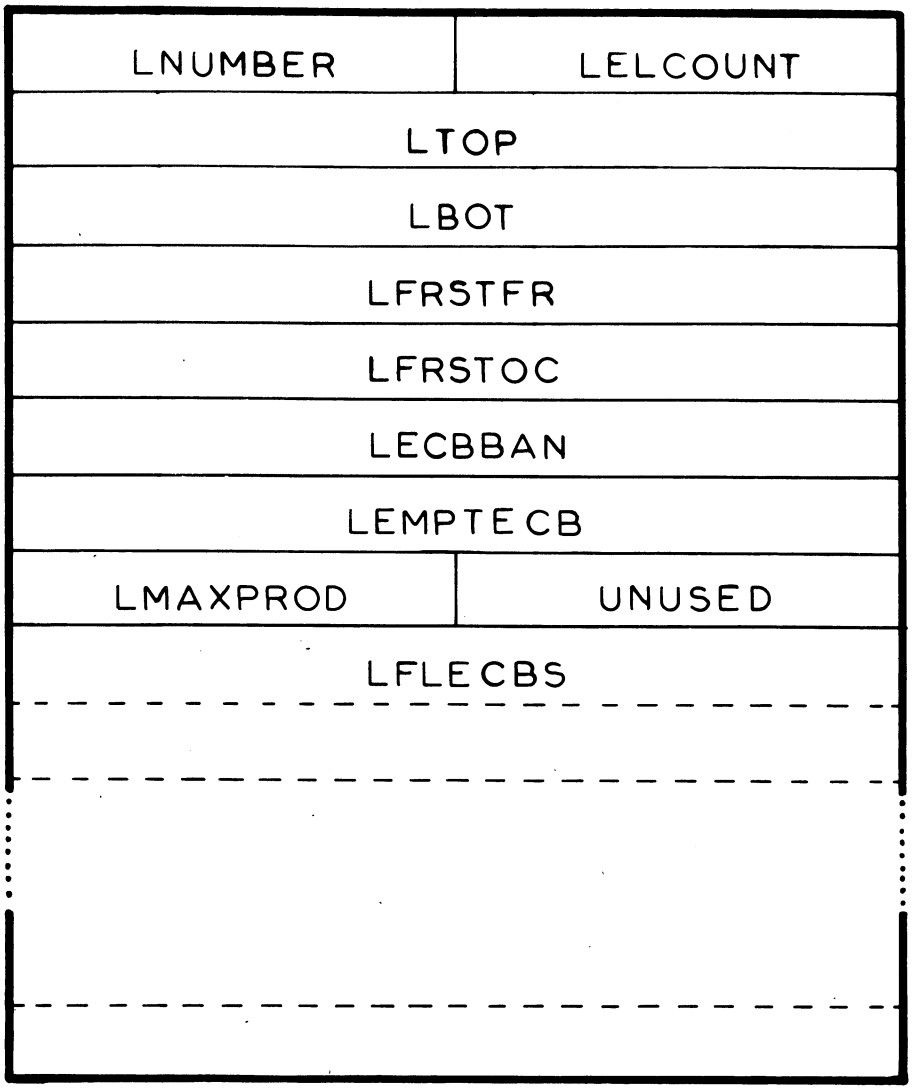
2.4.4. Un cas particulier de ressources. Les listes.

La majorité des ressources du système TELCOM sont des listes d' éléments, listes qui ont toutes la même structure (voir figure 15). Nous avons donc créé une gestion spéciale pour elles, basée bien sûr, sur la gestion générale des ressources par ENQ et DEQ.

Les listes constituent les moyens de communication entre les tâches du système. Ces dernières ont vis-à-vis des listes des fonctions du type producteur - consommateur.

Elles sont conçues par consommateur; c' est à dire, pour une liste donnée il peut y avoir plusieurs producteurs d' éléments mais un seul consommateur.

LISTE (TSKLST)



LNUMBER : numero de ressource
LELCOUNT : nombre d'éléments dans la liste
LTOP : adresse du debut de la liste
LBOT : adresse du dernier mot de la liste
LFRSTFR : adresse du premier élément libre dans la liste
LERSTOC : adresse du premier élément occupe dans la liste
LECBBAN : ECB spécial pour attente multiple
LEMPTECB : ECB de l'événement liste vide (un seul consommateur)
LMAXPROD : nombre maximum de producteurs
LFLECBS : ECBS de l'événement 'liste pleine' (plusieurs producteurs)

figure 15

La gestion des listes se fait par les procédures PUTLST (mettre un élément dans une liste) et GETLST (enlever un élément d' une liste).

2.4.4.1. La procédure "PUTLST" .

Une tâche qui veut mettre un élément dans une liste appelle la procédure PUTLST en passant comme paramètres: l' adresse de l' élément (dans R0) et l' adresse de la liste (dans R1).

Il existe deux possibilités d' appel suivant que l' on veut ou non se mettre en attente si la liste est pleine. Respectivement R1 est positif ou négatif.

PUTLST appelle tout d' abord la primitive ENQ pour la ressource "liste". PUTLIST met la tâche en attente si la ressource est occupée; sinon elle alloue la ressource à la tâche et teste si la liste est pleine. Si oui elle libére la ressource en appelant DEQ et met ou non la tâche en attente sur "liste pleine" selon l' appel. Sinon elle met l' élément dans la liste (met son adresse dans la première entrée libre) en modifiant proprement les pointeurs LFRSTFR et LFRSTOC.

Si la liste était vide, alors, PUTLST appelle POST pour réveiller la tâche consommateur si elle était en attente sur "liste vide" ou si elle avait demandé la mémorisation de l' événement.

Enfin PUTLST effectue DEQ de la ressource de la liste et revient à l' appelant.

2.4.4.2. La procédure "GETLST" .

Une tâche qui veut obtenir un élément d' une liste appelle la procédure GETLST (paramètre: l' adresse de la liste dans R1).

Il existe aussi deux possibilités d' appel suivant que l' on veut ou non se mettre en attente si la liste est vide. Respectivement R1 est positif ou négatif.

GETLST effectue d' abord un ENQ pour acquérir la ressource "liste". Si la ressource est occupée GETLST met la tâche en attente; si non elle alloue la ressource à la tâche et teste si la liste est vide. Si oui, elle libère la ressource et met ou non la tâche en attente suivant l' appel. Si non elle prend le premier élément de la liste en modifiant respectivement les pointeurs.

Si la liste était pleine, GETLST réveille par appel à POST toute tâche producteur qui était en attente sur "liste pleine" ou qui avait demandé la mémorisation de l' événement.

Ensuite GETLST effectue un DEQ sur la ressource de la liste et revient à l' appelant.

Nous devons signaler, ici, que pour éviter un bloquage mutuel nous avons convenu qu' une tâche avant de se mettre en attente doit libérer toute ressource occupée afin de ne pas gêner l' arrivée de l' évenement attendu.

2.5. MECANISMES DE BASE DU SYSTEME .

Nous traitons dans ce paragraphe les mécanismes des entrées - sorties, de l' horloge, et des appels au superviseur du système.

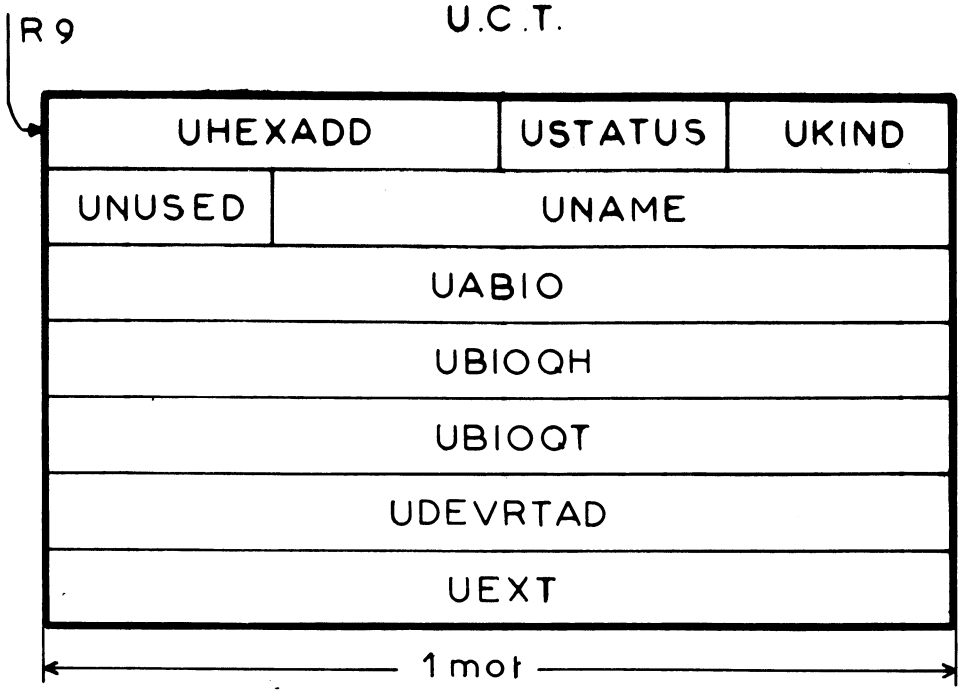
2.5.1. Les opérations d' entrée-sortie .

La gestion des entrées-sorties dans le système TELCOM s' effectue par le superviseur d' entrée-sortie qui se compose de deux modules (EXCP et IOINT) et fait partie du superviseur du système.

Le module EXCP est responsable du lancement des opérations d' entrée-sortie.

Le module IOINT récupère les interruptions et exécute l' action appropriée suivant le code associé à l' interruption et le type d' unité.

A chaque unité d' entrée-sortie nous avons associé une table de contrôle appelée UCT, avec une structure standard (figure 16). Dans cette table nous mettons les informations concernant l' unité telles que: son adresse, son état, son nom, l' opération en cours etc.



UHEXADD :adresse hexadécimale de l'unité

USTATUS :état de l'unité : x'80'=disponible
x'40'=occupée
x'20'=non disponible

UKIND :sorte d'unité : x'80'=réelle
x'00'=virtuelle

UNAME :nom de l'unité en EBCDIC

UABIO :adresse du BIO actif

UBIOQH :adresse de la tête de la queue de BIO

UBIOQT :adresse du dernier BIO dans la queue

UDEVRTAD :adresse de la routine spéciale de cette unité

UEXT :adresse de l'extension de l UCT

figure 16

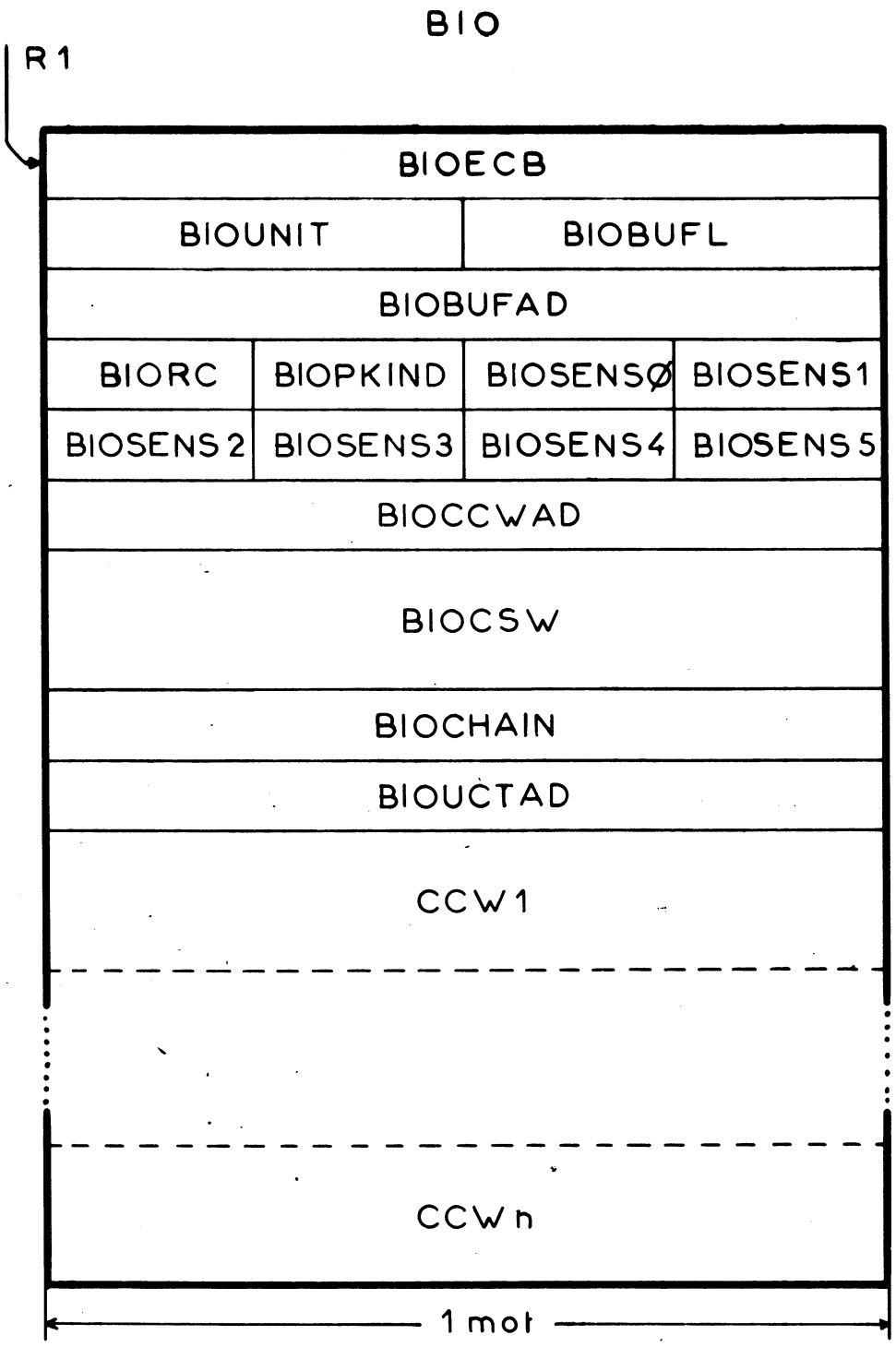

BIO
R1
BIOECB
BIOUNIT
BIOBUFL
BIOBUFAD
BIORC
BIOPKIND
BIOSENSØ
BIOSENS1
BIOSENS2
BIOSENS3
BIOSENS4
BIOSENS5
BIOCCWAD
BIOCSW
BIOCHAIN
BIOUCTAD
CCW1
CCWn
1 mot

Figure 17

Pour certaines unités spéciales qui nécessitent plus d' information nous avons prévu une extension de l' UCT. Dans un tel cas l' adresse de cette extension se trouve dans l' UCT (entrée UEXT).

A chaque opération d' entrée-sortie nous associons, aussi, un bloc de contrôle appelé BIO, avec la structure de la figure 17. Un tel bloc contient toutes les informations nécessaires pour la définition, l' exécution et la description de l' opération, telles que: l' adresse de l' unité en cause, l' adresse et la longueur du buffer, le type de l' opération, le programme canal etc.

2.5.1.1. Lancement d' une opération d' entrée-sortie .

Une tâche qui désire effectuer une opération d' entrée-sortie doit préparer le BIO qui la définit et le confier après au module EXCP pour le lancement de l' opération.

Elle doit remplir toutes les entrées de ce bloc, qui décrivent l' opération; c' est-à-dire: BIOUNIT, BIOBUFL, BIOBUFAD, BIOPKIND, BIOCCWAD, BIOUCTAD, et les CCWS du programme canal.

Le BIO de l' opération étant proprement rempli, la tâche appelle EXCP.

2.5.1.2. Le module "EXCP" .

Le module EXCP, partie du superviseur du système, se déroule entièrement en mode masqué. Il possède deux entrées différentes pour deux appels possibles:

a. par une tâche

b. par IOINT après la fin d' une opération pour lancer la suivante s' il y en a en file d' attente.

La tâche associée à une unité d' entrée-sortie peut lancer plusieurs opérations (c' est-à-dire passer les BIO correspondants à EXCP) sans attendre la fin de chacune . Le module EXCP est, alors, responsable de lancer la première et de mettre en file d' attente les autres. Cela veut dire qu' il existe (ou qu' il peut exister) une file d' attente d' opérations à effectuer (des BIO) pour chaque unité et que toutes ces files d' attente sont gérées par EXCP.

Les informations concernant ces files d' attente se trouvent dans l' UCT de l' unité correspondante.

Quand EXCP est appelé par une tâche sa seule source d' informations est le BIO de l' opération passé par la tâche. Après une vérification de validité du BIO il cherche l' UCT de l' unité en question et regarde s' il s' agit d' un SIO ou d' un HIO.

Dans le cas d' un HIO celui-ci est lancé directement et EXCP revient à l' appelant.

S' il s' agit d' un SIO il teste l' état de l' unité (décrit dans l' UCT). Si l' unité est occupée, c' est à dire qu' il y a une autre opération en cours sur elle, il chaîne le BIO dans la queue des BIO associée à cette unité. Sinon il lance l'

opération, marque dans l' UCT que l' unité est occupée et qu' un BIO est actif (UABIO = adresse du BIO lancé) et revient à l' appelant. Il est bien évident qu' il teste le code de retour sur le SIO et que si celui est anormal il exécute l' action appropriée.

Si EXCP était appelé par IOINT, il teste s' il existe des BIO dans la queue des BIO associée à la même unité et si oui il lance le premier et revient à l' appelant (=IOINT).

2.5.1.3. Le module "IOINT" .

L' autre partie du superviseur des entrées - sorties du système, le module IOINT, prend le contrôle après toute interruption d' entrée - sortie.

Il examine si au moment de l' interruption il y avait une tâche active, auquel cas il sauvegarde ses registres et son PSW dans son bloc de contrôle (TCB), la rend "inactive" en mettant l' état correspondant dans le TCB, et signale, enfin, qu' il n' y a plus de tâche active.

Ensuite ayant comme identificateur l' adresse de l' unité qui se trouve dans l' ancien PSW (IOOPSW), il cherche l' UCT correspondant à l' interruption. Il trouve, après, le BIO de l' opération qui vient de se terminer (UABIO dans l' UCT) et remplit les entrées correspondantes du BIO avec les codes de retour indiquant que l' opération est bien terminée ou la raison pour laquelle elle s' est mal terminée.

Puisque chaque unité a ses particularités dans le traitement des interruptions, le traitement particulier pour chacune tout en

faisant partie de 1' IOINT du système se trouve dans la "device routine" correspondante. Les "device routines" sont des sections associées à chaque unité d' entrée-sortie où nous avons regroupé les différents traitements spéciaux pour chacune. Elles sont décrites au paragraphe 2.6.

Si 1' opération d' entrée-sortie terminée était lancée sans attente, alors, IOINT appelle EXCP pour lancer 1' opération suivante décrite par le premier BIO de la queue des BIO pour 1' unité en cause. Si au contraire elle était lancée avec attente, il poste la tâche qui attend (la rend "inactive") et dans les deux cas revient au distributeur.

2.5.2. L' horloge (timer) du svstème .

a. Généralités.

L' horloge est une mémoire particulière de la mémoire centrale du système 360 située à 1' adresse décimale 80 (50 hexa). Donc dans la mémoire basse avec les autres mémoires spécialisées: les 5 PSW anciens et nouveaux, les mémoires de chargement initial et la zone de diagnostics.

Initialisé à la valeur positive maximum M = $(2^{31} - 1)$ le contenu de 1' horloge décroît continûment jusqu' à la valeur -(M + 1), valeur à laquelle elle repasse à la valeur +M.

Donc si 1' on initialise 1' horloge par la valeur -1 il faudra 2M+1 décréments pour passer à zéro.

Tout se passe comme si le bit 23 (poids 256) de 1' horloge était décrémenté tous les 1/300e de seconde. Cela veut dire que

l' unité de temps horloge (TU) est:

1TU = 1/300X256 sec. = 13,02 µs.

Dans la pratique cet effet est obtenu de deux façons différentes suivant le type d' horloge:

- horloge à haute résolution: décrément de 13 µs.
- horloge standard: en synchronisme avec le secteur 50 Hertz soit une période de 1000/50 ms = 20 ms

(On notera que 20 ms = 20X76,8 = 1536 TU).

Sur machines virtuelles de CP/67 on peut obtenir une horloge décrémentée en temps réel en spécifiant l' option REAL TIMER dans la déclaration de la machine. On obtient alors une horloge standard.

b. Gestion de l' horloge.

Deux types de services sont usuellement demandés à une horloge:

- l' heure du jour
- la mesure d' intervalles de temps multiples.

b1. L' heure courante relative.

Initialisé à la valeur M le contenu de l' horloge décroît linéairement.

L' échelle des temps et le contenu sont exprimées dans la même unité (TU).

L' abcisse indique le temps écoulé depuis le chargement initial de l' horloge.

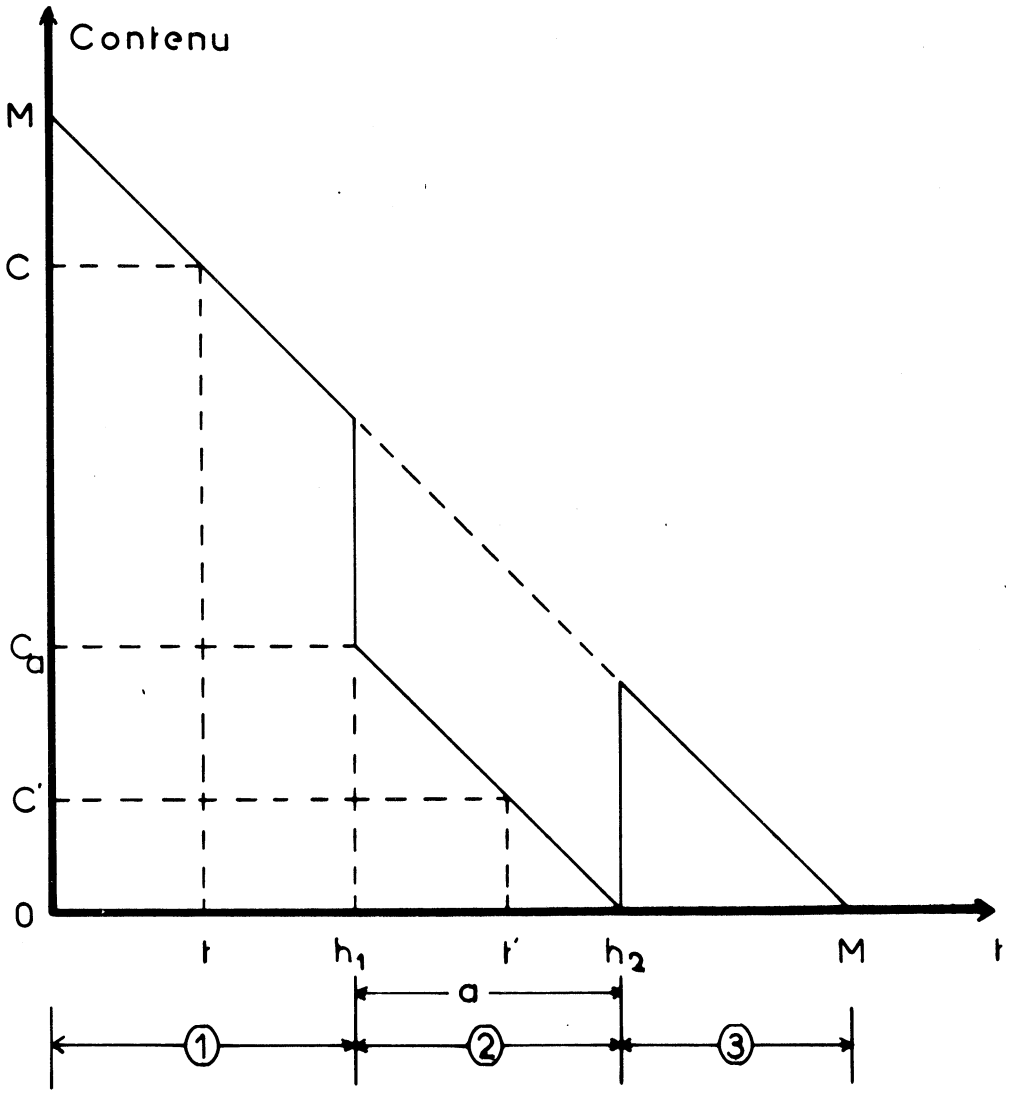


figure 18

Prenant pour valeur initiale le plus grand entier positif (7FFFFFFF hexa) le contenu de l' horloge décroît et arrive à zéro après l' écoulement de $2^{31}-1$ ou $2X10^{9}$TU. soient 7 heures environ.

A chaque passage à zéro une interruption spécifique de type externe se produit.

Pour connaître l' heure courante relative t, il suffit de lire l' horloge, soit C son contenu

$$t = 0 \; ; \; C = M$$

$$t = M-C$$

Du fait que toutes les 7 heures il y a passage par zéro il faut gérer une variable écart E :

A chaque passage par zéro E := E + M ;

On remarquera que si M est représentable sur un mot E ne l' est pas; il faudra donc exprimer l' écart dans une autre unité (plus grande) ou en multiple précision (en double-mot).

b2: Mesure d' un intervalle.

Supposons qu' à l' heure relative h1 on ait désiré une interruption dans un délai α c' est-à-dire à l' heure relative:

$$h2 = h1 + \alpha$$

Il faut réinitialiser l' horloge à la valeur α à l' heure h1, puis à l' heure h2, on reportera la valeur M -h2 qu' elle aurait eue si on n' avait pas fait de mesure d' intervalle.

Considérons les intervalles (1) , (2) , (3) :

Soit Ci le contenu de l' horloge à un temps ti.

Dans l' intervalle (1) : $t = M - C$ (1)

Dans l' intervalle (2) : $t' = h2 - C'$

A noter que h1 est calculable par (1) et que h2 est aussi connu puisque: $h2 = h1 + \alpha$

Ceci conduit à utiliser la gestion de pile suivante:

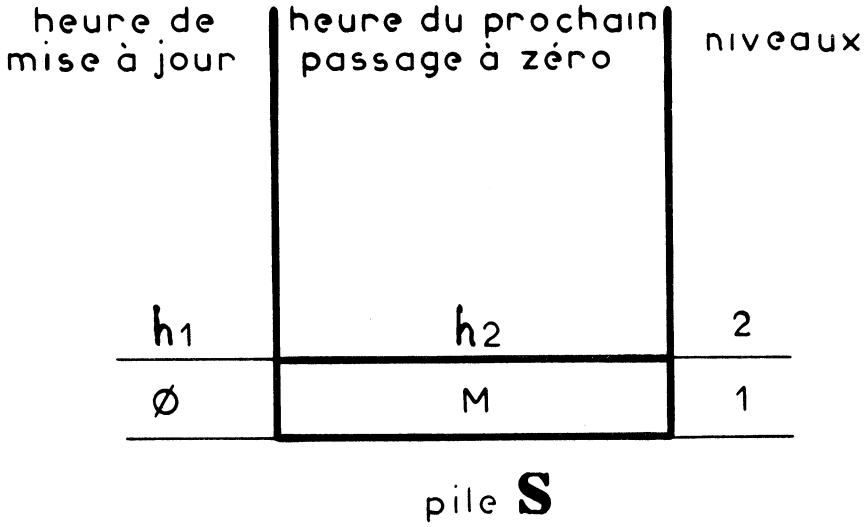


figure 19

Valeur initiale : M

Dans l' intervalle (1)

t = M - C n' est autre t = S(i) - Ci avec i = 1

A l' heure h1, préparation de l' intervalle :

h1 = S(i) - Cα

i = i + 1

S(i) := h1 + α (= h2)

Si l' on désire obtenir l' heure pendant l' intervalle (2), on trouve bien:

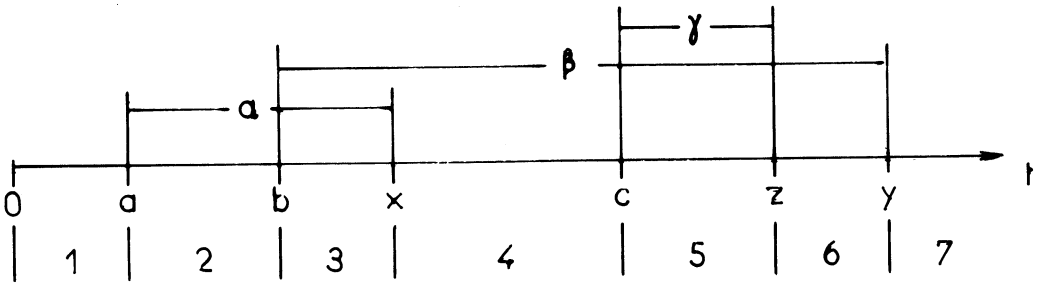
t' = S(i) - C' = h2 - C'

Enfin au passage par zéro, instant h2:

C := S(i-1) - S(i) (soit M-h2)

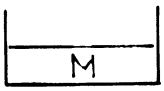
i := i-1

Nous pouvons généraliser cette méthode à un cas plus complexe comportant des intervalles parenthésés et des intervalles recouverts:

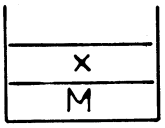


Etudions la pile pendant les différents intervalles 1 à 7:

1: →

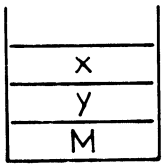


2: →

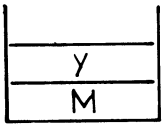


3: On notera que le nouveau passage à zéro demandé est pour l' heure y > x. Il y a donc lieu de faire une insertion dans la pile pour respecter la chronologie

→

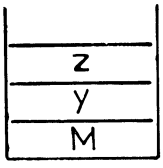


4: →

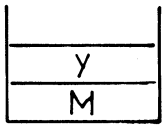


à l' instant x on a fait C := y - x

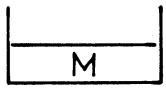
5: →



à l' instant c on a fait C := z

6: → 

à l' instant z on a fait C := y - z

7: → 

à l' instant y on a fait C := M -y

On remarque que la taille de la pile est égale au nombre maximum d' intervalles à mesurer simultanément plus 1.

b3. Le cas général:

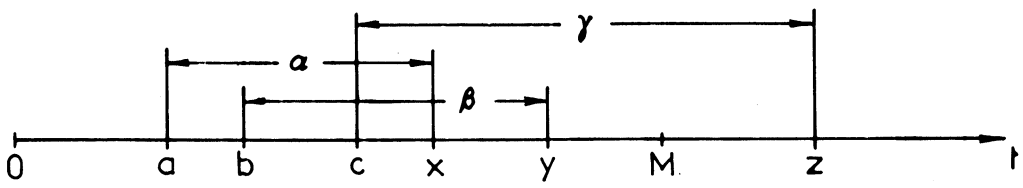
1- On accepte de traiter au plus p intervalles simultanément.

2- Le plus grand intervalle permis est égal à M (7 heures)

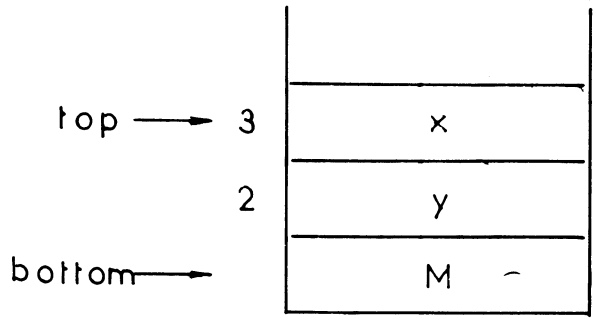
La dimension de la pile sera de p+1 niveaux.

Il faut compléter l' étude précédente par le cas d' intervalle recouvrant une période M de l' horloge.

Supposons le système dans l' état suivant:



A l' instant c la pile a la structure suivante:



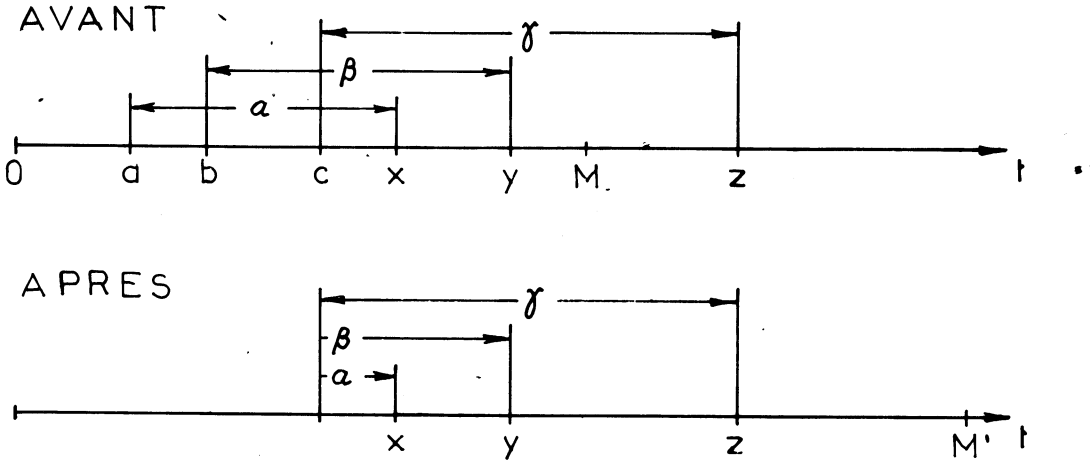
Désirant à l' instant c insérer l' intervalle γ on trouve que l' heure z du passage à zéro correspondant est:

$$z := c + \gamma > M$$

Cette opération provoque un débordement de capacité. Il faut donc opérer un changement de coordonnées. Pour éviter d' avoir à faire cette opération trop souvent il est suffisant de faire le décalage vers la droite de la valeur la plus grande que l' on sache représenter, à savoir M.

Par hypothèse $\gamma < M$,

on peut donc opérer la translation suivante:



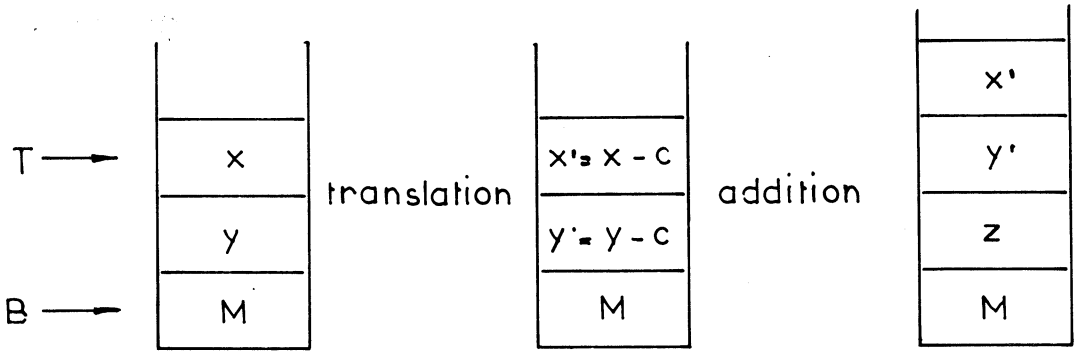
Il suffit donc de retrancher c au contenu de tous les éléments de la pile sauf le premier qui contient toujours M.

On ne peut pas placer la nouvelle origine en x au cas où dans l' intervalle (c,x) un nouvel intervalle serait à définir.

Lors de la translation la variable écart E doit être incrémentée de la valeur c:

E := E + c ; (double précision)

Les différents états de la pile sont les suivants:



b4. Heure du jour.

Pour obtenir la date et l' heure à partir d' une horloge telle que nous l' avons définie il suffit de noter la date et l' heure demandée à l' opérateur au moment de l' initialisation de l' horloge.

Heure réelle = Heure d' initialisation + temps relatif.

Dès lors il est possible de demander un intervalle de temps sous forme d' arrêt à une heure réelle donnée.

2.5.2.1. L' heure du jour dans le système TELCOM .

Le module "TIME" .

Dans le système TELCOM quand nous voulons demander 1' heure du jour nous devons appeler le module "TIME". En retour, ce module rend à 1' appelant dans R0 1' heure et dans R1 la date sous la forme suivante:

R0 = HHMMSSTH R1 = 00YYDDDF

Au moment de 1' initialisation du système nous demandons 1' heure du jour à CP67 et nous initialisons un certain nombre de mots en mémoire (dans la section DATA), telles que:

HYEARS (demi-mot) avec 1' année courante

HMONTHS " avec le mois

HDAYS " avec le jour

HHOURS " avec 1' heure

HMIN " avec les minutes

HSEC " avec les secondes

TELEX (mot) avec la date TELEX

EIPLH et EIPLL (deux mots) avec 1' heure de 1' initialisation

Nous mettons aussi la valeur maximale dans 1' horloge et dans la pile.

"TIME" quand il est appelé, soustrait de la valeur du sommet de la pile la valeur courante de 1' horloge et obtient, ainsi, 1' heure relative à 1' heure de 1' initialisation. A cette valeur, il faut ajouter 1' heure de 1' initialisation se trouvant dans EIPLH-EIPLL, pour avoir 1' heure du jour demandée.

2.5.2.2. Les demandes d' intervalles de temps .

Dans le système TELCOM nous pouvons demander une interruption après l' écoulement d' un intervalle de temps, avec la contrainte d' un intervalle inférieur à 7 heures.

Nous avions deux méthodes pour réaliser ces demandes d' intervalles de temps:

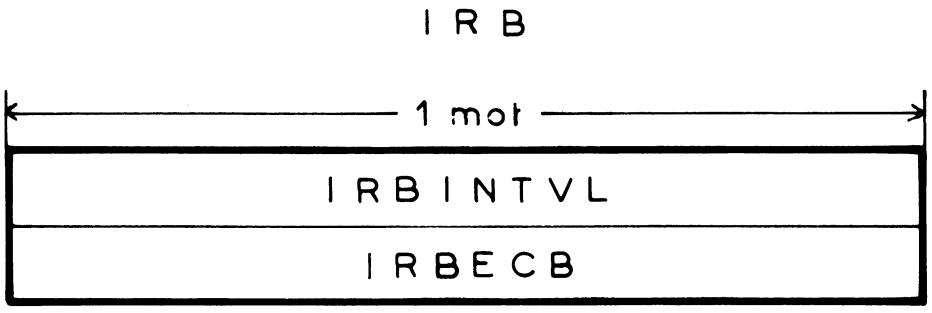
a. Interruption de la tâche au bout de l' intervalle de temps et

b. Création d' un événement: "écoulement de l' intervalle de temps" et réveil de la tâche intéressée.

La première est une méthode asynchrone et donc plus générale alors que la deuxième est une méthode synchrone.

Nous avons choisi la deuxième laquelle était moins coûteuse et pourrait néanmoins couvrir les besoins du système.

Une tâche qui veut faire une telle demande doit construire un bloc de contrôle (IRB) qui est constitué de deux mots. L' un contenant la valeur de l' intervalle en sec. ou en TU. et l' autre l' ECB correspondant à l' événement: "écoulement de cet intervalle de temps".



IRBINTVL : intervalle en sec. ou en T.U.
IRBECB : ECB de l' événement 'écoulement de l' intervalle'

figure 20.

Ensuite la tâche doit appeler le module "STIMER" lui passant comme paramètre l' adresse de l' IRB.

2.5.2.3. Le module "STIMER .

C' est le module qui manipule la pile de l' horloge. Cette pile est installée dans la section "DATA" à l' adresse S et a la forme suivante:

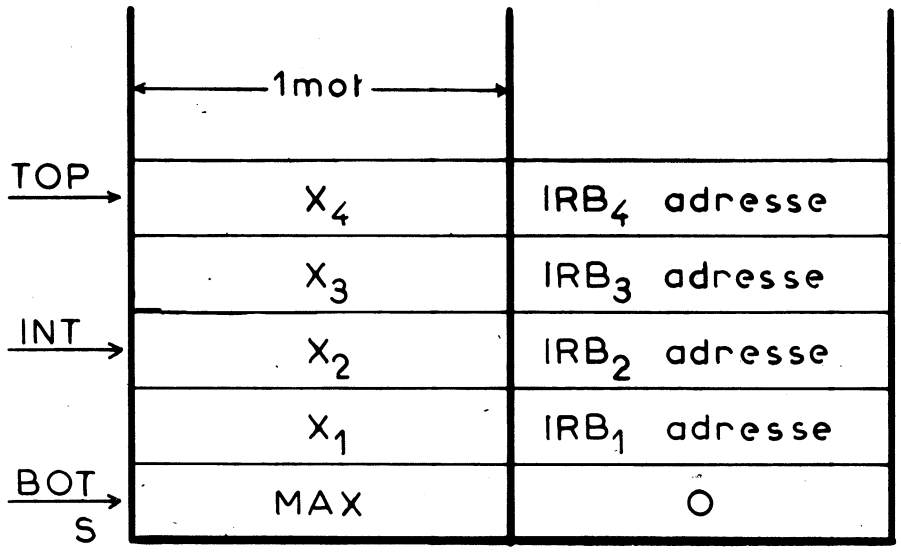


figure 21. pile de gestion de l'horloge

Dans cette pile nous disposons d' un double-mot par niveau. Au premier niveau nous avons la valeur maximale (temps initial du prochain passage à zéro) et le deuxième mot à zéro. Ensuite à chaque niveau nous avons l' heure du prochain passage à zéro (après les demandes d' intervalles) et à côté l' adresse de l' IRB correspondant à cette demande, de façon à pouvoir retrouver la tâche concernée.

La manipulation de la pile étant une action critique STIMER se déroule entièrement en mode masqué.

Ayant reçu une demande (un IRB) d' une tâche, STIMER soustrait le contenu de 1' horloge de la valeur du sommet de la pile (heure du prochain passage à zéro). A 1' heure du moment trouvée de cette façon il ajoute 1' intervalle demandé pour obtenir 1' heure du passage à zéro correspondant. Puis, en comparant cette valeur avec les différents niveaux de la pile il crée un niveau correspondant à cette demande en déplaçant la partie supérieure de la pile (ordre décroissant X4<X3<X2<X1<MAX).

Si cette valeur a été la plus petite (niveau supérieur dans la pile) il met dans 1' horloge 1' intervalle demandé, autrement 1' horloge reste inchangée.

Cas spécial: Si 1' heure du passage à zéro calculée est supérieure à la valeur maximale, nous retranchons de tous les niveaux de la pile 1' heure du moment, nous insérons 1' intervalle demandé dans la pile, et nous gardons en mémoire 1' heure du moment (écart) pour les calculs postérieurs de 1' heure du jour.

2.5.2.4. Les interruptions de 1' horloge - Le module "EXTINT" .

De tous les types des interruptions externes, les seules prises en compte par le système TELCOM sont les interruptions de 1' horloge (passage à zéro).

Le module EXTINT, traitant ces interruptions fait partie du superviseur du système et se déroule en mode masqué.

Prenant, donc, le contrôle après toute interruption de l' horloge il désactive la tâche active, s' il y en avait une.

Ensuite, si la pile de l' horloge n' a qu' un niveau, il remet la valeur maximale et l' ajoute à l' écart pour les calculs postérieurs de l' heure du jour. Si au contraire la pile avait plusieurs niveaux, il trouve au sommet de la pile l' adresse de l' IRB associé à cette interruption et il réveille la tâche correspondante. Si en dessous il y avait d' autres niveaux avec la même valeur il réveille aussi les tâches correspondantes.

Enfin, il met la nouvelle valeur dans l' horloge (égale: niveau en dessous - niveau courant), et élimine le niveau courant et tous les niveaux avec la même valeur.

Ayant terminé son travail EXTINT revient au distributeur du système.

2.5.3. Les appels au superviseur (SVC) .

Les appels au superviseur constituent le seul moyen de demander par programme l' aide du superviseur du système. Nous les avons utilisés pour différentes raisons:

a. Pour donner aux tâches la possibilité de se masquer (évitant la coûteuse instruction privilégiée SSM).

Dans ce cas nous utilisons l' appel avec le code 0, et la séquence d' appel est la suivante:

```
      SVC   0
SVCM  DS    H
```

Avec l' interruption provoquée par SVC le module "SVCINT" qui traite ce type d' interruptions prend le contrôle en mode masqué, puisqu' il fait partie du superviseur du système.

SVCINT dans le cas d' un SVC 0 , met le masque de l' appelant dans le demi-mot qui suit l' instruction SVC et revient à l' appelant en sautant ce demi-mot.

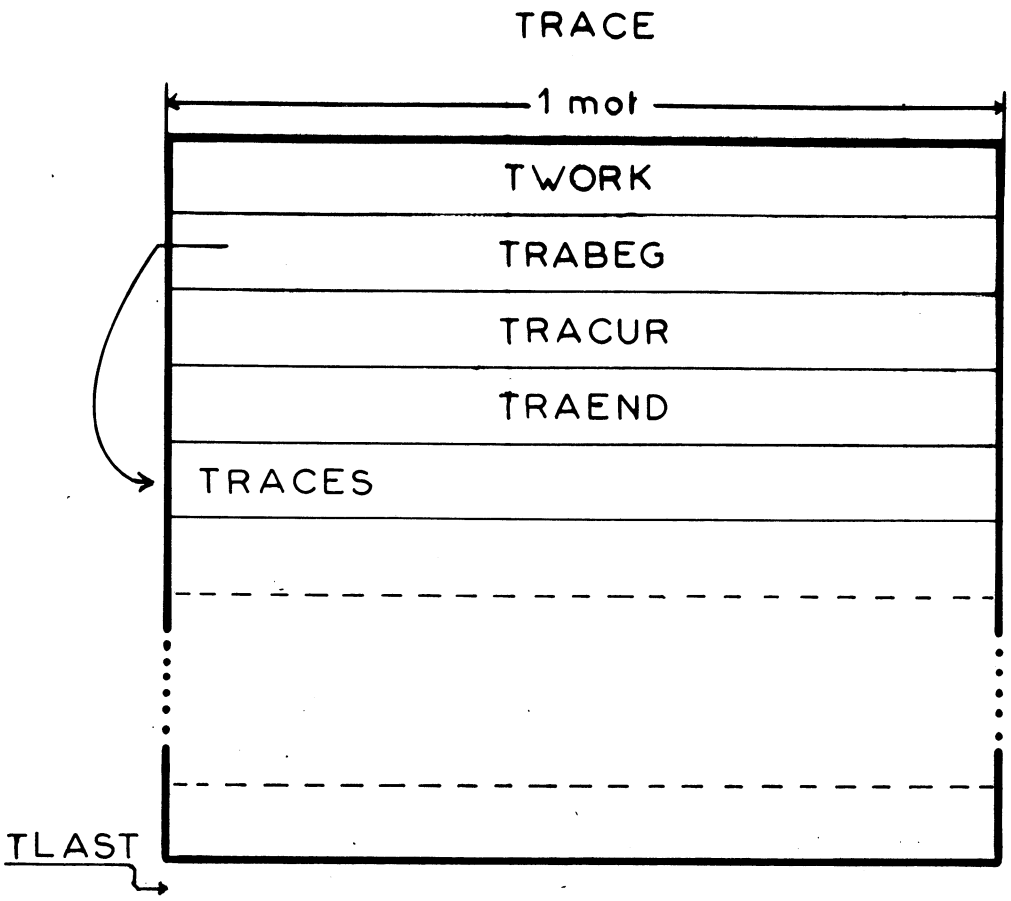
De cette façon l' appelant est déjà en mode masqué et il a toujours son masque antérieur dans SVCM pour pouvoir se démasquer quand il voudra.

b. Pour récupérer une trace du déroulement des programmes.

Dans ce cas nous utilisons l' appel avec le code 1 , et la séquence d' appel est la suivante:

```
      SVC   1
      DC    F'0'
```

Le mot qui suit l' instruction SVC (compteur) est augmenté de un à chaque passage et son adresse est mise dans une liste circulaire ("TRACE"). Cette liste constitue une section de données du système et a la structure de la figure 22.



TWORK : espace de travail

TRABEG : adresse du début de la liste circulaire

TRACUR : adresse de l'entrée courante de la liste

TRAEND : adresse de la dernière entrée de la liste

TRACES : début de la liste

figure.22

c. Les appels au superviseur avec les autres codes sont utilisés pour terminer l' activité du système en cas d' erreur catastrophique, en indiquant l' erreur et faisant sortir sur l' imprimante une image de la mémoire au moment de l' erreur. Les différents codes d' appels caractérisent chaque erreur.

2.6. GESTION DES UNITES D' ENTREE - SORTIE .

Dans ce paragraphe nous étudions d' abord les points communs de la gestion de toute unité d' entrée - sortie au système TELCOM, puis séparément les cas des unités virtuelles et des unités réelles ayant exclu les lignes de communication qui seront étudiées au chapitre 3.

2.6.1. Les "device - routines" .

Le système TELCOM comporte une tâche par unité d' entrée - sortie. Cette tâche utilise une section de contrôle qui décrit l' algorithme de la gestion de l' unité.

Il y a différents types d' opérations que la tâche sera amenée à lancer sur l' unité. Pour rendre le lancement de ces opérations systématique et standard pour toute unité, nous avons créé une section de contrôle en plus par unité, appelée "device routiné". Toute "device routine" commence par un vecteur de transfert (suite de branchements) de la forme suivante:

```
B     ...INIT
B     ...READ
B     ...WRITE
B     ...IOINT
```

```
B       ...SENSE
B       ...CLOSE
B       ...ATTN
B       ...CSWST
```

Il est évident que chaque entrée adressée par ce vecteur correspond à une opération à effectuer sur 1' unité. Bien sûr toutes les entrées ne sont pas utilisées pour chaque unité; 1' entrée WRITE, par exemple, ne peut pas être utilisée pour une unité de lecture.

Ainsi la tâche désirant effectuer une certaine opération sur une unité doit appeler la "device routine" associée et indiquer 1' entrée correspondante. Ceci a été rendu systématique par 1' emploi de la macro: $DEVROUT arg
où arg doit être le déplacement de 1' entrée dans le vecteur de transfert de la "device routine".

Nous devons nous arreter un peu sur cette notion importante de la "device routine" qui est empruntée au système MTS. C' est un moyen simple de traiter de façon standard des périphériques différents. Les tâches associées aux périphériques lancent les mêmes ordres (type: READ;WRITE etc.) et ce sont les "device routines" qui effectuent les traitements spécifiques. Un ordre de lecture par exemple est traduit par 1' emploi de la macro: $DEVROUT DRREAD en un branshement vers 1' entrée READ de la "device routine" qui prepare et lance 1' opération sur 1' unité . Nous pouvons, donc, aussi avec cette méthode simuler un périphérique par un programme et arriver à réaliser 1' équivalence programme - unité physique, but poursuivi dans tous les systèmes développés actuellement.

2.6.2. Les unités virtuelles .

Les unités virtuelles sont des simulations par le système CP67 des unités réelles. Leur programmation est compatible avec celle des unités réelles; leur gestion est plus facile du fait qu' elles se présentent comme "idéales", au sens que certains cas particuliers ne peuvent pas se produire.

2.6.2.1. Lecteur virtuel .

Parmi les unités virtuelles, le lecteur est celle dont l' étude présente le plus grand interêt car il est plus complexe que les autres.

La programmation d' un lecteur de cartes virtuel est compatible avec celle d' un lecteur réel. Cependant du fait que le lecteur est virtuel certains cas particuliers ne peuvent se produire; entre autres tout le traitement des cas relatifs aux opérations demandées à l' opérateur en cas de:

- contrôle de validité (perforation non valide)
- arrêt alimentation (bourrage de cartes).

Toutes ces opérations sont résolues par CP67 au niveau de son système de "SPOOL".

En s' abstenant d' utiliser des possibilités particulières du programme canal (bit PCI éteint, bit SLI allumé) et en l' absence d' erreur de programmation provoquant:

- 'program check' (programme canal invalide)
- 'protection check' (mauvaise clé dans le CAW)

- 'chaining check' (mauvais chainage de commandes)

les bits d' état canal (40 à 47 du CSW) sont toujours nuls. De plus il n' est jamais nécessaire de recourir à une commande 'SENSE' pour éclaircir la nature des cas anormaux.

Supposons que 1' on exécute un SIO sur un lecteur virtuel. Le code condition restitué par 1' opération est le suivant:

CC = 0 - Succès

CC = 1 - CSW rangé

CC = 2 - Canal ou sous-canal occupé

CC = 3 - Non opérationnel

CC = 3 : indique que le lecteur n' est pas connecté ou hors tension. Sous CP67 cela indique que 1' adresse référencée n' a pas été déclarée dans la configuration virtuelle. Ce cas ne doit donc pas se produire sauf en cas d' erreur de programmation.

CC = 2 : Canal occupé : ne peut pas se produire sur un canal multiplexé fonctionnant en 'byte mode' (c' est le cas usuel). Sous canal occupé : pourrait se produire si on émettait deux SIO consécutifs sur le même lecteur ce qui n' est pas une pratique habituelle puisque 1' on s' arrange pour traiter une opération à la fois par périphérique dans tout superviseur classique.

CC = 1 : CSW rangé. Trois cas sont possibles:

a. Il n' y a pas de carte alimentée dans le lecteur, il est donc impossible d' opérer la lecture. Sur un lecteur réel ce cas nécessite une intervention manuelle: poser un paquet de cartes dans le lecteur, mettre la plaquette, appuyer sur départ et fin de fichier.

Sur un lecteur virtuel il faudra une succession de commandes du type:

XFER	00D	TO	MV	On pose un paquet de cartes sur le lecteur.
CLOSE	00C			
READY	00C	On appuiera sur Pret et Fin de fichier.		

Ce cas est indiqué par stockage d' un CSW au moment du SIO (sélection initiale) avec le bit 'unit check' allumé.(X'0200').

b. Il n' y a plus de cartes dans le lecteur car une opération précédente vient de lire la dernière carte. Ici encore il ne peut plus y avoir lecture faute de cartes.

Ceci indique que l' on a terminé la lecture d' un paquet de cartes (fin physique) à l' opération précédente; c' est-à-dire qu' on a atteint la fin de fichier (EOF).

Sous CP67 chaque XFER pose sur le lecteur un paquet de cartes. A la fin de chaque paquet on obtient la condition EOF puis passage au paquet suivant.

Cette condition est indiquée par le bit 'unit exception' (X'0100') dans le CSW.

c. Ayant émis un SIO en mode masqué il se peut que dans l' intervalle de temps qui sépare la mise en mode masqué et le SIO l' opérateur ait alimenté le lecteur comme indiqué dans le cas a. Ceci en mode démasqué aurait provoqué une interruption avec un CSW contenant le bit 'device end' (X'0400') isolé. Etant en mode masqué cette interruption n' a pu être présentée au CPU et est en attente.

L' opérateur SIO est l' occasion qui permet de transmettre cette interruption en attente (pending). Dans ce cas l' opération de lecture demandée par le SIO n' a pas démarrée et nous devons la relancer (réemettre le SIO).

CC = 0 : L' opération de lecture a été lancée normalement. Deux solutions sont alors possibles:

a. Rester en mode masqué, la fin d' opération ne pourra être détectée que par une opération TIO pour trouver quand le lecteur sera à nouveau disponible. Cette méthode est particulièrement coûteuse sous CP67 et doit être évitée (boucle sur une opération privilégiée).

b. Passer en mode démasqué (autoriser les interruptions) en ayant soin d' initialiser convenablement l' 'I/O NEW PSW'.

Le SIO étant bien lancé, le transfert d' information peut débuter immédiatement et le buffer se remplit. Quand l' opération de transfert d' information est terminée le canal est libéré. Ce qui provoque une interruption de type entrée-sortie avec le bit 'channel end' (X'0800') dans le CSW, et l' adresse NN du périphérique dans l' 'I/O OLD PSW'. L' information est alors disponible en mémoire et le lecteur débute la phase électromécanique:

- fait tomber une carte dans le panier de réception
- alimente la suivante
- lit la carte et charge le buffer hardware. Cette opération prend environ 50ms sur un lecteur réel. A la fin de cette phase une nouvelle interruption se produit avec le bit 'device end' (X'0400') allumé.

Dans le cas de ces deux interruptions le CSW pointe l' adresse qui suit celle du CCW que l' on a lancé.

Dans la pratique à chaque interruption on fera une opération de type OR avec le CSW et une mémoire mise à zéro avant l' opération. Après chaque opération on testera la présence des deux

bits 'channel end' + 'device end'; ce n' est que lorsque les deux bits sont présents que l' on considèrera l' opération terminée.

Du fait qu' il est possible de transférer vers le lecteur d' une machine virtuelle des images de cartes (XFER d' une sortie perforation) que des lignes d' imprimante (XFER d' une sortie impimante) le lecteur virtuel peut lire des images de cartes ou des lignes de 132 caractères. On doit donc opérer une lecture avec un compte suffisant dans le CCW avec le bit SLI allumé. En fin d' opération on trouvera dans le CSW le compte résiduel et par différence on trouvera la longueur effectivement lue.

L' algorithme de la gestion d' un lecteur virtuel du système TELCOM peut être décrit par un automate fini déterministe avec la configuration suivante:

Etats:

S0 : Etat initial ('close' du lecteur, envoi systèmatique à S1).

S1 : Lancement de la lecture et test sur lecture initiale.

S2 : Test "unit check" sur lecture initiale.

S3 : Test "unit exception" (fin de fichier).

S4 : Test de fin de lecture.

S5 : Attente de "device end" après "unit check".

S6 : Construction du "record" et mise dans la liste correspondante, envoi systèmatique à S1.

S7 : Test de fin de lecture initiale.

S8 : Envoi de "request to start" et attente de permission.

S9 : Ereur de programmation, sortie par svc.

Evénements:

E0 : Code condition sur SIO = 0.

E1 : Code condition =1 et CSW = "unit exception".

E2 : Code condition =1 et CSW = "unit check".

E3 : Code condition =1 et CSW = "device end"

E4 : Code condition sur SIO 2 ou 3.

E5 : Lecture initiale.

E6 ; Lecture non initiale.

E7 : Fin de lecture avec CE+DE.

E8 : Fin de lecture avec DE

E9 : Fin de lecture différent de CE+DE.

E10 : Permission arrivée.

Matrice de l'automate du lecteur

evenement / états	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
S0	close lecteur ; ⟶ S1										
S1						S2	S3				
S2	S7	S9	S5	S7	S9						
S3	S4	S0	S9	S9	S9						
S4								S6	S9	S9	
S5								S9	S1	S9	
S6	construction 'record';mise dans liste ⟶ S1										
S7								S8	S9	S9	
S8											S6
S9	erreur ; sortie par SVC										

figure 23

2.6.2.2. Perforateur virtuel .

La gestion du perforateur virtuel est extrêmement simple puisque dans son cas sauf erreur de programmation le lancement et la fin de l' opération se passent toujours bien.

Nous décrivons aussi la gestion utilisée par la tâche associée au perforateur par un automate fini déterministe de la forme suivante:

Etats:

S0 : Etat initial ('close' du perforateur, envoi systèmatique à S1).

S1 : Chercher un élément dans la liste.

S2 : Trouver la machine virtuelle de destination, lancer XFER du perforateur vers cette machine, envoi systèmatique à S4.

S3 : Test du premier enregistrement.

S4 : Lancement de perforation, envoi systèmatique à S1.

S5 : Attente d' arrivée d' un élément dans la liste .

Evénements:

E0 : Il existe un élément dans la liste (non EOF).

E1 : Il existe un élément indiquant EOF dans la liste.

E2 : Il n' existe pas d' élément dans la liste.

E3 : Un élément est arrivé dans la liste vide.

E4 : Premier enregistrement du fichier.

E5 : Non premier enregistrement du fichier.

matrice de l'automate du perforateur

evenement / etats	E 0	E 1	E 2	E 3	E 4	E 5
S 0	'close' perfor.; ⟶ S 1					
S 1	S 3	S 0	S 5			
S 2	XFER TO MV; ⟶ S 4					
S 3					S 2	S 4
S 4	perforer; ⟶ S 1					
S 5				S 1		

figure 24

2.6.2.3. Imprimante virtuelle .

La gestion de l' imprimante virtuelle est exactement la même que celle du perforateur.

Dans le système TELCOM nous n' utilisons pas l' imprimante si ce n' est dans le cas du bloquage de tous les lecteurs pour éviter la mort du système. Nous acheminons, alors, tous les fichiers arrivés sur les lecteurs pour une destination défaillante vers l' imprimante locale.

L' arrivée de ce cas anormal indique une mauvaise utilisation du système par ses utilisateurs distants; par exemple, ils persistent à envoyer des fichiers sur leur imprimante qui est en panne !.

2.6.3. Les unités réelles .

Les unités réelles utilisées par le système TELCOM sont les lignes de communication et le télétype opérateur.

La gestion des lignes de communication sera étudiée séparément dans le chapitre 3.

Nous abordons ici l' étude de la gestion du télétype opérateur.

2.6.3.1. Le télétype opérateur du système.

a. Particularités du comportement d' un télétype.

Un télétype est connecté au canal par l' intermédiaire d' une unité de contrôle et d' une ligne à basse vitesse de transmission. Pour utiliser un télétype sous le système CP67 il faut, donc, déclarer une ligne (2703) dans la configuration de la machine virtuelle.

-Pour sa gestion il faut commencer par le lancement d' une commande ENABLE à l' adresse déclarée dans la configuration. Cette commande se termine par l' interruption CE+DE quand sur un télétype réel on frappe la commande: DIAL (nom de la machine virtuelle); exemple: DIAL TELCOM.

-Un télétype utilise le code ASCII. Nous sommes, donc, obligés d' effectuer un transcodage après toute lecture en EBCDIC et avant toute écriture de l' EBCDIC en ASCII.

-En lecture:

La fin normale d' une commande de lecture est provoquée par la touche 'retour chariot' avec "channel end" + "device end" dans le CSW.

Si la lecture n' est pas finie en 28 secondes nous récupérons alors une interruption avec "unit check" dans le CSW et "time-out" dans le premier octet de sense.

Si la lecture se termine par la touche "break" l' interruption a "unit check" dans le CSW et "intervention required" dans le premier octet de sense.

-En écriture:

La fin normale d' une commande d' écriture après l' épuisement des caractères à écrire, se signale par interruption avec "channel end + device end" dans le CSW.

Si nous arrêtons une écriture en cours en appuyant sur la touche "break", l' écriture se termine par interruption avec "unit exception" ou "unit check" et "intervention required" dans le premier octet de sense.

-Nous pouvons encore remarquer que s' il n' y a aucune commande lancée sur le télétype son clavier est débloqué. Nous devons, donc, dans ce cas pour mieux gérer son fonctionnement, lancer une commande PREPARE qui bloque le clavier pour toutes les touches sauf "break".

-La commande PREPARE se termine par la touche "break" avec "channel end + device end" dans le CSW.

-Nous signalons, enfin, que les fonctions de "retour chariot" et de "changement de ligne" s' effectuent par l' écriture des caractères spéciaux réservés à ces fonctions.

b. L' automate de la gestion du télétype.

L' algorithme de gestion utilisé par la tâche associée au télétype opérateur peut être décrit par un automate fini déterministe de la forme suivante:

Etats:

S0 : Etat initial (après la fin d' ENABLE par le DIAL), envoi systèmatique à S1.

S1 : Recherche d' élément dans la liste du télétype.

S2 : Lancement d' un PREPARE

S3 : Lancement d' une lecture.

S4 : Lancement d' une écriture.

S5 : Exécution de la commande lue (eventuellement lancement des écritures) et envoi systèmatique à S1.

S6 : Tuer le PREPARE par HIO et envoi systèmatique à S1

S7 : Mise d' un élément de lecture dans la liste, envoi systèmatique à S1

S8 : Erreur de programmation sortie par SVC.

Evénements:

E0 : Elément de lecture trouvé dans la liste du télétype.

E1 : Elément d' écriture trouvé dans la liste.

E2 : Aucun élément trouvé dans la liste.

E3 : Arrivée d' un élément (d' écriture) dans la liste vide.

E4 : Interruption avec CE+DE seul.

E5 : Interruption avec UC.

E6 : Interruption avec UE.

E7 : Interruption avec d' autres conditions.

matrice de l'automate du télétype.

evenement / etats	E 0	E 1	E 2	E 3	E 4	E 5	E 6	E 7
S 0	fin d'ENABLE; → S 1							
S 1	S 3	S 4	S 2					
S 2				S 1	S 1			
S 3					S 5	S 3	S 8	S 8
S 4					S 1	S 8	S 7	S 8
S 5	exécution de la command lue; → S 1							
S 6	tuer PREPARE; → S 1							
S 7	mettre élément lecture dans liste; → S 1							
S 8	erreur program.; sortie par → SVC							

figure 25

CHAPITRE 3

COMMUNICATION DES DONNEES

3.1. GENERALITES .

Nous pouvons définir la communication comme la transmission des signaux de toute sorte entre un point (station) d' origine et un point de réception sans altération de la séquence ou de l' information.

Les éléments minimum nécessaires pour une communication sont donc: une source d' information, un moyen de transmission d' information et un récepteur d' information.

Pour la communication des données qui nous intéresse particulièrement, la source et le récepteur d' information peuvent être:

-soit des ordinateurs

-soit des terminaux .

La liaison ou le chemin entre la source et le récepteur peut s' effectuer:

-soit par des lignes téléphoniques (à un fil ou groupe de fils)

-soit par le faisceau hertzien.

Dans cette étude nous nous limitons aux liaisons par des lignes téléphoniques.

3.1.1. Caractèristiques des liaisons par lignes téléphoniques:

Nous distinguons deux types de liaison:

-La liaison "point-à-point", lorsque deux stations sont reliées par une ligne.

-La liaison "multipoint", lorsque plusieurs stations sont reliées par une même ligne.

Suivant le sens de la transmission, nous distinguons trois modes d' utilisation des lignes:

-"simplex": un seul sens (A→B)

-"half duplex": deux sens mais un seul à la fois ($A \underset{t'}{\overset{t}{\rightleftharpoons}} B$)

-"duplex" ou "full duplex": deux sens en même temps (A↔B)

La vitesse de transmission sur la ligne se mesure en bit/seconde (ou BAUD, terme réservé aux transmissions télégraphiques). Le "bit" base de l' information est une représentation binaire. Les deux valeurs peuvent se représenter sur une ligne par deux états:

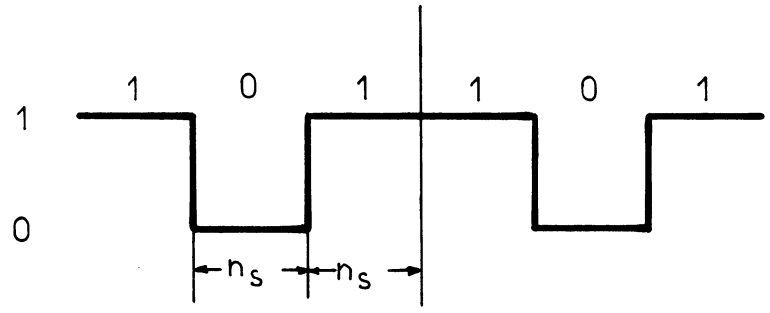


figure 26. signal de communication

Si le plus court intervalle de temps pendant lequel les caractèristiques d' un courant modulé demeurent inchangées est de n secondes, la vitesse de transmission en BAUDS s' exprime par 1/n.

Les lignes normales utilisées à la communication des données sont dans la gamme de vitesses: de 1200bauds à 4800bauds. Mais il existe des lignes expérimentales jusqu' à 1000000bauds.

Pour transférer les informations binaires par une ligne nous procedons à une modulation du courant sur cette ligne. Il en existe trois sortes:

-modulation d' amplitude

-modulation de fréquence

-modulation de phase

3.1.2. Le problème de synchronisation.

La notion de transmission d' information n' est pas indépendante de la notion du temps. La ligne prend une suite d' états discrets entre lesquels se produisent des régimes transitoires non significatifs. Le récepteur ne doit prendre en considération l' état de la ligne qu' en des instants bien déterminés (états significatifs).

La période du signal élémentaire étant supposée connue du récepteur, celui-ci est muni d' une horloge provoquant l' examen de la ligne à la fréquence convenable.

On distingue deux modes de transmission:

-Le mode synchrone, dans lequel l' intervalle entre deux signaux quelconques est toujours multiple de la période du signal élémentaire,

-Le mode asynchrone, dans lequel l' intervalle entre deux signaux quelconques est indépendant de la période du signal élémentaire.

Nous devons, ici, mentionner le mode de transmission "start-stop". Dans ce mode les données sont groupées par caractères (ou blocs de bits). Les groupes peuvent être émis à des intervalles arbitraires, mais les signaux à l' intérieur de chaque groupe sont transmis selon le mode synchrone.

Dans ce cas, on délimite les groupes par des signaux de séparation ("start" et "stop") et la synchronisation s' effectue systématiquement au début de chaque groupe, à la réception du délimiteur "start".

En résumé, le mode "start-stop" est un mode de transmission asynchrone par caractère, synchrone par bit.

3.1.3. La nécessité d' un protocole de transmission.

Dans toute communication pour que les deux stations en liaison se comprennent elles ont besoin d' utiliser un certain langage. Pour la communication de données les mots du langage sont les caractères du code utilisé (EBCDIC, ASCII, etc.) et la grammaire est ce qu' on appelle "protocole de transmission". Ce protocole doit définir la structure des messages et un ensemble de caractères spéciaux pour l' établissement et le maintient de la communication. Il doit définir, par exemple les caractères à

envoyer par la station désirant émettre ("allo"), les caractères de réponse qui assurent que l' autre station est prête à recevoir ("oui"), la séparation entre les différents messages envoyés (pause), la réponse affirmative à un message ("oui"), la demande de répétition d' un message non compris ("pardon?"), etc.

La nécessité d' assurer ces fonctions dans une communication a donc donné naissance au protocole de transmission.

3.2. LE PROTOCOLE B.S.C.

Nous décrivons, ici, le protocole le plus répandu sur matériel IBM, et sur lequel nous avons basé notre gestion de lignes, le protocole B.S.C. (Binary Synchronus Communication).

Comme son nom l' indique il s' agit d' un protocole pour un mode de transmission "synchrone". Il permet les liaisons point-à-point et multipoint et fournit un jeu complet de fonctions de contrôle. Il est indépendant du code de transmission utilisé (EBCDIC, ASCII, 6 bit code, etc.). Tout code ayant un nombre fixe de bits par caractère peut être utilisé à la condition d' avoir réservé une douzaine de caractères par lesquels B.S.C. assure ses fonctions de contrôle.

3.2.1. Les caractères de contrôle.

Ce sont les caractères destinés à assurer le contrôle de la liaison, c' est-à-dire, à établir la liaison entre les correspondants , à délimiter les messages et déclencher les opérations de vérification des données transmises, et de déconnexion.

Liste des caractères:

SYN (synchronous idle): sert à établir le synchronisme entre les deux stations en liaison et à le maintenir dans les intervalles

d' absence de données ou de caractères de contrôle.

SOH (start of heading): précède l' en-tête d' un message.

STX (start of text): précède un bloc de caractères, sert également à terminer un en-tête.

ETB (end of transmission block): indique la fin d' un bloc de caractères précédés par SOH ou STX. Il est immédiatement suivi des caractères de vérification et réclame une réponse du récepteur (ACK ou NAK).

ETX (end of text): termine un message. Il est immédiatement suivi des caractères de vérification et réclame une réponse du récepteur.

EOT (end of transmission): indique la conclusion d' une transmission.

ENQ (enquiry): sert à réclamer une réponse, pour obtenir l' identification d' une station et/ou son état (ACK, NAK,...). Il peut aussi être utilisé pour obtenir la retransmission d' une réponse, dans le cas où une transmission n' a pas été reçue au moment où on l' attendait.

ACK0, ACK1 (affirmative acknowledgement): signal de réponse indiquant l' acceptation par le récepteur du bloc précédent et la disponibilité de ce même récepteur pour recevoir le bloc suivant.

NAK (negative acknowledgement): le récepteur est prêt à accepter la retransmission d' un bloc erroné. Sert aussi à répondre "pas prêt" à une sélection.

DLE (data link escape): sert uniquement à élaborer des signaux de contrôle supplémentaires, tels que WABT, ACK0, ACK1, le départ ou la fin d' un texte transparent ou des séquences de contrôle à l' intérieur d' un texte transparent. (La notion de la "transparence" est décrite plus loin).

ITB (intermediate block check): indique la fin d' un bloc de caractères intermédiaire et est suivi des caractères de vérification. Sa réception ne provoque pas l' inversion du sens de transmission pour répondre.

WABT (wait before transmit): indique que l' émetteur n' est pas pour le moment prêt à recevoir et demande l' attente de l' écoulement d' un délai avant transmettre.

RVI (reverse interrupt): permet au récepteur d' interrompre l' émetteur pour lui envoyer un message plus prioritaire.

3.2.2. Les messages.

Un message se présente sous la forme suivante:

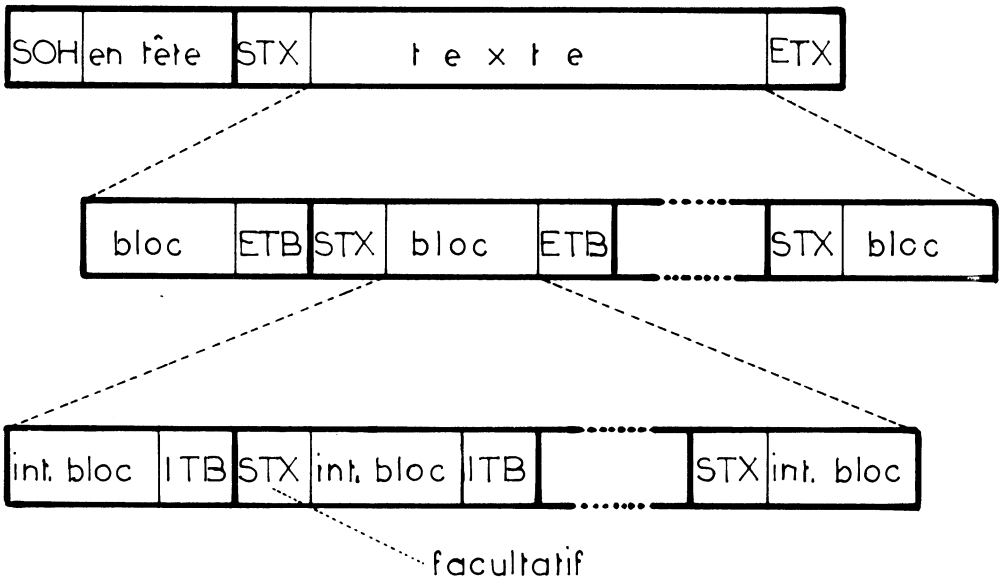


figure 27. la forme d'un message B.S.C.

L' en-tête est précédé par un caractère SOH et terminé par un caractère STX ou ETB ou ENQ. Dans le dernier cas il sert à opérer un contrôle sur la station réceptrice avant l' envoi du message.

Le texte du message débute par un caractère STX et se termine par un caractère ETX. Il est subdivisé en blocs; chaque bloc débute par un caractère STX et s' achève par un caractère ETB à l' exception du dernier qui s' achève par un ETX.

Chaque bloc est subdivisé en blocs intermédiaires; le premier débute par un caractère STX, mais ceci est facultatif pour les suivants. Chaque bloc intermédiaire se termine par un caractère ITB à l' exception du dernier qui se termine par ETB ou ETX.

3.2.3. Synchronisation.

Lors de l' établissement de la liaison ou lors de chaque inversion du sens de transmission, deux opérations doivent être effectuées dans la station réceptrice:

- la synchronisation - bit (reconnaissance du rythme)
- la synchronisation - caractère (cadrage des caractères à l' intérieur du flot de bits).

La synchronisation - bit est effectuée soit dans le "modem" (s' il est muni d' une horloge), soit dans l' unité de contrôle; il faut alors que l' unité de contrôle soit munie d' une horloge.

La synchronisation - caractère est effectuée dans l' unité de contrôle

Dans le protocole B.S.C. la synchronosation est établie à l' aide de caractères SYN; la séquence est la suivante:

SYN-(SYN-SYN-SYN-SYN)-(SYN-SYN)-texte-SYN

Le premier et le dernier caractère SYN servent de tampon. Le dernier s' appelle aussi caractère de renversement de ligne puisqu' il indique le changement de sens de transmission.

Les 4 caractères suivant le premier sont utilisés pour la synchronisation bit et ne sont pas envoyés par l' unité de contrôle si le "modem" est muni d' une horloge. La séquence de deux SYN qui suit assure la synchronisation caractère et est insérée par l' unité de contrôle (IBM-2701 ou 2703 par exemple).

Des caractères SYN servent également à maintenir la synchronisation en l' absence d' informations ou de caractères de contrôle ou encore à intervalles réguliers d' une seconde:

en mode normal groupe de deux SYN

en mode transparent la combinaison DLE-SYN

3.2.4. Etablissement de liaison.

Dans la suite nous ne nous occupons que des liaisons point-à-point.

Dans un tel cas la station désirant émettre doit réclamer la ligne en envoyant un caractère ENQ; la station réceptrice doit, suivant le cas, répondre:

- ACK0 si elle est prête à recevoir,
- NAK au cas contraire et
- WABT si momentanément n' est pas prête à recevoir.

Si les deux stations ont la possibilité d' émettre et

demandent à émettre au même moment, alors, une priorité est affectée à chaque station.

3.2.5. Les réponses ACK0 et ACK1.

Ces réponses ne peuvent être envoyées que par une station réceptrice. La réponse à un bloc de message est alternativement ACK0 et ACK1, suivant que son rang est respectivement impair et pair.

L' alternat des réponses constitue une protection contre la perte ou la duplication de blocs.

3.2.6. Le contrôle d' erreurs de transmission.

Chaque bloc subit un contrôle d' erreurs, qui varie avec les codes de transmission utilisés. Les contrôles habituellement rencontrés dans la pratique sont:

- contrôle d' imparité, par caractère, ou VRC (Vertical redundancy check)
- contrôle par bloc de deux sortes:

  soit LRC (Longitudinal redundancy check)

  soit CRC (Cyclic redundancy check)

Quand le code de transmission utilisé est l' EBCDIC, ce qui est le cas pour nous, le contrôle effectué est le CRC (contrôle cyclique de redondance).

La procédure de contrôle par bloc est déclenchée par les caractères de fin de texte, de bloc et de bloc intermédiaire

(ETX, ETB, et ITB) et effectuée par l' unité de contrôle (2701 ou 2703 par exemple).

En émission elle prépare deux octets de vérification dit CRC à partir des caractères du bloc envoyé et les ajoute après le caractère de la fin du bloc. En réception elle prépare aussi les mêmes octets et les compare avec ceux qu' elle a reçu. S' ils ne sont pas égaux l' unité de contrôle nous le signale ("unit check" et sur SENSE "data check") et nous devons répondre "données non acceptées" (NAK).

Les octets de vérification CRC sont calculés selon l' algorithme de division de polynomes dans le corps des entiers modulo 2.

Nous rappelons que dans ce corps l' addition et la multiplication suivent les règles suivantes:

Addition	Multiplication
1 + 1 = 0	1.1 = 1
0 + 0 = 0	0.0 = 0
0 + 1 = 1	0.1 = 0
1 + 0 = 1	1.0 = 0

Les octets en question se forment comme le reste R(x) de la division des polynomes:

$$B(x).x^{16} / \; x^{16}+x^{15}+x^2+1$$

où

$$B(x) = b_n x^n + b_{n-1} x^{n-1} + \ldots + b_1 x^1 + b_0 x^0$$

Exemple:

bits transmits: 1100 0111 1111 0100

$$B(x) = x^{15}+x^{14}+0x^{13}+0x^{12}+0x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+0x^3+x^2+0x^1+0x^0$$

$$B(x).x^{16} = x^{31}+x^{30}+0x^{29}+0x^{28}+0x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{20}+0x^{19}+x^{18}+0x^{17}+0x^{16}$$
$$+0x^{15}+0x^{14}+0x^{13}+0x^{12}+0x^{11}+0x^{10}+0x^9+0x^8+0x^7+0x^6+0x^5+0x^4+0x^3+0x^2+0x^1+0$$

$$\frac{B(x).x^{16}}{x^{16}+x^{15}+x^2+1} = x^{15}+x^{10}+x^8+x^6+x^4+x^3+x^1+1$$

avec un reste R(x):

$$R(x) = 0x^{15}+0x^{14}+0x^{13}+x^{12}+0x^{11}+0x^{10}+0x^9+0x^8+0x^7+0x^6+x^5+x^4+0x^3+x^2+x^1+1$$

nous aurons alors:

CRC = 0001 0000 0011 0111

3.2.7. Le mode "transparent".

Quand nous utilisons ce mode de transmission, le texte n' est plus composé d' une série de caractères mais d' une série de bits d' information, dont l' interprètation est laissée au programme récepteur.

Ce mode ne concerne que la partie texte d' un message. Les interrogations, les réponses, et les en-têtes sont transmis en mode normal.

Les caractères de contrôle sont remplacés par des configurations obtenues par association du caractère DLE avec chacun des caractères de contrôle; exemple:

DLE-STX, DLE-ETB, DLE-SYN, DLE-ENQ etc.

Une configuration DLE-DLE indique au récepteur qu' on avait voulu transmettre une partie du texte identique au caractère DLE.

Le message en mémoire a la forme des blocs de caractères consécutifs ne comportant aucune configuration de contrôle sauf le DLE-STX en tête de chaque bloc.

Suivons le processus de transmission d' un tel bloc:

-une première commande WRITE achemine vers l' unité de contrôle (contenant l' adaptateur synchrone de la ligne) le bloc situé en

mémoire.

-l' adaptateur envoie alors sur la liaison une séquence de synchronisation.

-il passe en mode transparent à la rencontre du DLE-STX.

-transmet le DLE-STX sur la liaison,

-déclenche l' accumulation du CRC (contrôle d' erreurs),

-transmet en série les caractères du bloc sans aucune reconnaissance de caractère de contrôle autre que DLE; lorsqu' un DLE est rencontré, l' adaptateur en insère un deuxieme pour transmettre DLE-DLE.

-après la transmission du dernier caractère, l' adaptateur reçoit un signal 'stop' du canal et se met en attente après avoir déclenché une temporisation de 3sec. Dans ces 3sec. au plus tard l' adaptateur doit recevoir une deuxième commande WRITE pour la configuration de fin DLE-ETX ou DLE-ETB ou DLF-EOT et en réception de cette séquence:

-il se replace en mode normal et

-transmet la séquence suivie par les caractères de contrôle d' erreurs (CRC) et le caractère tampon (SYN).

L' adaptateur aussi insère dans le texte et à intervalles réguliers des séquences DLE-SYN pour maintenir la synchronisation.

Du côté du récepteur les fonctions sont les suivantes:

-la configuration DLE-STX met le récepteur en mode transparent et déclenche l' accumulation du CRC.

-il reçoit le bloc d' information où il repère les DLE-SYN pour les éliminer et les DLE-DLE pour effacer le premier.

-un DLE reçu seul est une erreur.

-une configuration de fin de séquence (DLE-ETB, DLE-ETX,

DLE-EOT) replace le récepteur en mode normal.

-reception du CRC et contrôle.

3.3. LES PROCEDURES DE TRANSMISSION .

Une procédure de transmission se situe à un niveau supérieur à celui d' un simple protocole. Se basant toujours sur un protocole, elle gère la ligne au niveau logique du flot des messages. Elle doit assurer non seulement l' initialisation et le maintient de la conversation sur la ligne, mais aussi l' aiguillage des messages vers leurs destinations spécifiques, la reprise en cas d' erreur et l' optimisation.

Pour assurer ces fonctions la procédure doit définir son protocole de transmission, le format et la sémantique des messages qui transitent sous sa gestion sur la ligne.

Nous traitons ici les deux procédures basées toutes les deux sur le protocole B.S.C.

3.3.1. La procedure élémentaire B.S.C.

Comme son nom l' indique cette procédure est constituée du protocole B.S.C. auquel on ajoute quelques caractères de fonction pour assurer l' ensemble des fonctions d' une procédure.

Elle est utilisée pour la communication avec des terminaux lourds non programmables (exemple IBM/2780).

3.3.1.1. Son protocole de transmission

La procédure élémentaire B.S.C. utilise tous les caractères de contrôle du protocole B.S.C. sauf que le caractère SOH a le même sens que STX puisque l' en-tête n' existe pas dans la procédure.

Selon cette procédure à chaque instant l' une des deux stations en communication est en mode d' envoi et l' autre en mode de réception. C' est à dire, qu' elles ne peuvent pas envoyer et recevoir du texte en même temps.

Au moment de l' initialisation, pour l' établissement du dialogue, le terminal envoie un caractère ENQ (demande de connexion logique) et le central répond: DLE-ACK0 (d' accord).

Le terminal envoie, alors, sa carte d' identification:

/ SIGNON RMn paramètre

A la reconnaissance de cette carte le central doit répondre: DLE-ACK1 et la communication est établie.

En principe le terminal envoie systématiquement, alors, un EOT et passe en mode de réception pour donner au central la possibilité d' émettre le premier.

Avec l' envoi d' un EOT une station passe en mode de réception. A ce moment l' autre si elle a quelque chose peut l' envoyer et doit d' abord demander la permission (envoi d' un ENQ) sur laquelle la première répond DLE-ACK0 (d' accord). Et l' autre continue,alors, en envoyant du texte sous la forme:

STX-texte-ETB (sans transparence)

DLE-STX-texte-DLE-ETB (avec transparence)

Les réponses sur réception correcte des textes sont: DLE-ACK0, et DLE-ACK1 alternativement.

Si le texte n' est pas bien reçu on demande sa répétition par NAK.

Si par contre, une réponse à un texte n' était pas bien reçue on demande sa répétition par ENQ.

Quand la station-émetteur n' a plus rien à envoyer elle envoie un EOT et passe en mode de réception donnant à 1' autre station 1' occasion de passer en mode d' émission.

3.3.1.2. La structure des messages

La procédure élémentaire B.S.C. ajoute trois caractères de fonction (ESC, HT, EM) au jeu des caractères standards B.S.C. permettant à 1' émetteur de gérer certaines fonctions sur le récepteur. Parmi ces caractères seul EM peut être utilisé en émission à partir du terminal lourd.

Les fonctions remplies par chacun de ces caractères sont les suivantes:

ESC (escape): Ce caractère avec celui qui le suit constituent une séquence de sélection de terminal. Par exemple ESC-4 sélecte le perforateur de cartes du terminal lourd.

Dans la séquence qui sélecte 1' imprimante le deuxième caractère indique aussi 1' opération d' interligne ou de saut de papier correspondante.

HT (horizontal tabulation) : Celui-ci est un caractère de contrôle de tabulation sur 1' imprimante.

EM (end of media) : Indique le fin de 1' enregistrement.

Les messages, donc, dans cette procédure ont le format des messages B.S.C. déjà décrit (figure 27), mais les enregistrements dans chaque bloc ont la forme suivante:

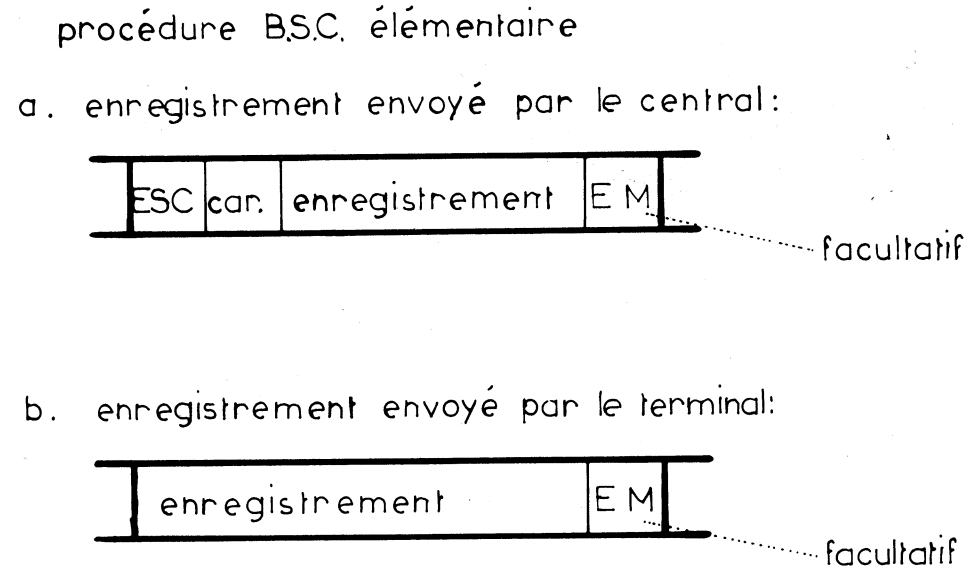


figure 28. les enregistrements dans la procédure "élémentaire B.S.C."

3.3.1.3. L' automate de la procédure

Nous pouvons décrire la procédure élémentaire B.S.C. à l' aide d' un automate fini déterministe avec la forme suivante:

Etats:

S0 : Etat initial, après la réception correcte de la carte SIGNON. Envoi systèmatique à S1.

S1 : Envoi de ACK1 (mode de réception).

S2 : Envoi de ACK0 (mode de réception).

S3 : Envoi de NAK (mode réception).

S4 : Envoi d' un texte (mode d' émission).

S5 : Envoi d' un ENQ (mode d' émission).

S6 : Envoi d' un EOT (mode d' émission - passage en mode de réception).

S7 : Tester s' il y a du texte à envoyer (mode d' émission).

S8 : Comparer la réponse arrivée avec celle attendue (mode d' émission) .

S9 : Erreur grave - sortie par SVC.

Evénements:

E0 : Séquence reçue invalide.

E1 : ACK0 reçu.

E2 : ACK1 reçu.

E3 : NAK reçu.

E4 : ENQ reçu.

E5 : EOT reçu.

E6 : Séquence correcte de texte reçue.

E7 : Réponse attendue reçue.

E8 : Réponse non attendue reçue.

E9 : Il existe un texte à envoyer.

E10 : Il n' existe pas de texte à envoyer.

La matrice de l automate de la procédure élémentaire B.S.C.

evenement / etat	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
S 0	apres reception correcte de SIGNON; S1										
S 1	S 3	S 9	S 9	S 9	S 1	S 7	S 2				
S 2	S 3	S 9	S 9	S 9	S 2	S 7	S 1				
S 3	S 3	S 9	S 9	S 9	S 3	S 7	S1 | S2 *				
S 4	S 5	S 8	S 8	S 4	S 9	S 9	S 9				
S 5	S 5	S 8	S 8	S 5	S 9	S 9	S 9				
S 6	S 6	S 9	S 9	S 9	S 2	S 6	S 9				
S 7										S 5	S 6
S 8								S 7	S 5		
S 9	erreur ; sortie par SVC										

* envoi à S1 ou à S2 suivant que le rang du texte recu est impair ou pair

figure 29

3.3.2. La procédure "multileaving".

La procédure la plus répandue est le "multi-leaving", défini dans le système HASP. Nous avons voulu être compatibles avec cette procédure pour pouvoir communiquer avec des systèmes existants sur les différents terminaux lourds.

3.3.2.1. Son protocole de transmission.

La procédure "multileaving" utilise le protocole B.S.C. Parmi les caractères de contrôle B.S.C. elle n' utilise pas: ACK1, ETX, et EOT, les fonctions correspondantes étant assurées par la structure du message elle même.

Le jeu complet, donc, des caractères de contrôle utilisés est: SYN, ENQ, ACK0, NAK, SOH, DLE, STX, ETB.

On admet que 1' ordinateur ou le système HASP est implanté est le centre principal et les autres calculateurs les terminaux lourds. Au moment de 1' initialisation d' une ligne et de 1' établissement du dialogue, le terminal envoie une sequence:

SOH-ENQ (demande de connexion logique)

et le central repond:

DLE-ACK0 (d' accord)

le terminal envoie, alors, sa carte d' identification de la forme suivante:

/ SIGNON RMn (parametre)

A la reconnaissance de cette carte par le central la communication est établie et la conversation normale se fait par

des textes alternés de la forme:

SOH-STX-texte-SYN-ETB (sans transparence) ou

DLE-STX-texte-DLE-ETB (avec transparence).

S' il n' y a rien de particulier à émettre, la réponse est:

DLE-ACK0, courrament denomée "bulle".

et si on souhaite la répétition du bloc envoyé au cas où il a été mal recu, on répond: NAK.

3.3.2.2. La structure des messages.

La forme générale d' un message "multileaving" est celle indiquée dans la figure 30.

Nous remarquons que dans la procédure "multileaving" aussi nous utilisons des séquences de fonctions. La composition et la structure de chacune est la suivante:

a. BCB (block control byte). Cette séquence a une longueur d' un octet et constitue un compteur des blocs envoyés ou reçus (modulo 16). Elle assure un meilleur contrôle sur la perte ou la duplication des blocs que l' emploi de deux réponses ACK0 et ACK1 de la procédure élémentaire.

Il existe trois types de BCB :

BCB = 1000nnnn : nnnn = compteur des blocs modulo 16.

BCB = 1001nnnn : ignorer le compteur.

BCB = 1010nnnn : remettre compteur attendu à nnnn.

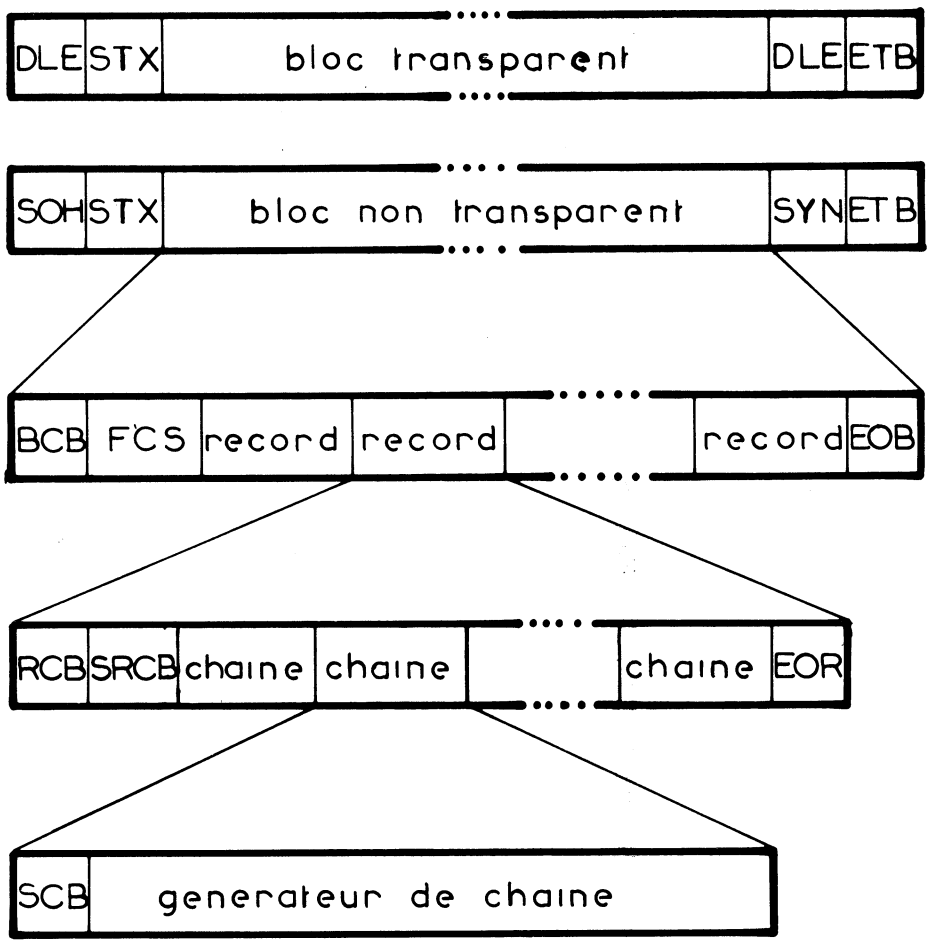

DLE STX bloc transparent DLE ETB
SOH STX bloc non transparent SYN ETB
BCB FCS record record record EOB
RCB SRCB chaine chaine chaine EOR
SCB generateur de chaine

figure 30

b. FCS (function control sequence). Cette séquence a une longueur de deux octets et indique au récepteur quels sont les flots que l' émetteur peut ou ne peut pas récevoir en réponse. Sa forme est la suivante:

FCS = lwrrijkl lcrrmnop

w = 1 : suspendre tous les flots.

w = 0 : consulter les bits des flots particuliers.

r = reservé.

Pour les bits des flots particuliers la signification est:

si le bit est à 1 = continuer ce flot.

si le bit est à 0 = suspendre ce flot.

La correspondance entre bits et flots est différente pour le central et le terminal.

	FCS reçu par le terminal	FCS reçu par le central
i	lecteur 1	imprimante 1
j	lecteur 2	imprimante 2 ou perforateur 7
k	lecteur 3	imprimante 3 ou perforateur 6
l	lecteur 4	imprimante 4 ou perforateur 5
m	lecteur 5	imprimante 5 ou perforateur 4
n	lecteur 6	imprimante 6 ou perforateur 3
o	lecteur 7	imprimante 7 ou perforateur 2
p	-	- perforateur 1
c	commande opérateur	message pour la console

c. RCB (record control byte). Cette séquence a une longueur d' un octet et identifie la destination de l' enregistrement logique qui suit ou une fonction de contrôle. Elle constitue le premier caractère de l' enregistrement logique (record) et si elle est à zéro elle est seule dans le record et signifie 'fin de bloc' (EOF). Autrement sa signification particulière est la suivante:

liii0000 : RCB de contrôle

RCB = X'90' : demande de transmettre
(à une destination indiquée dans le SRCB)

RCB = X'A0' : permission de transmettre
(à une destination indiquée dans le SRCB)

RCB = X'E0' : signal d' erreur dans la séquence de blocs.

RCB = X'F0' : RCB de contrôle généralisé
(la fonction particulière dans le SRCB)

liii0001 : console opérateur distante (output)

liii0010 : console opérateur distante (input)

liii0011 : lecteur distant (input)

liii0100 : imprimante distante (print)

liii0101 : perforateur distant (punch)

iii : indique le terminal particulier (s' il y en a plusieurs).

d. SRCB (sub record control byte). Cette séquence a une longueur d' un octet et donne des informations de contrôle supplementaires associées au record qui suit. Le format et la signification du SRCB sont particuliers à chaque RCB:

i. SRCB pour les demandes et les permissions de commencer une fonction (RCB = X'90' ou X'A0'):

C' est le RCB correspondant à cette fonction; exemple:
pour une demande de commencer à envoyer sur 1' imprimante distante nous avons:

RCB = X'90' et SRCB = X'94'

ii. SRCB pour 1' erreur de séquence des blocs (RCB = X'E0'):
C' est le BCB incorrect reçu.

iii. SRCB pour les records de contrôle généralisé
(RCB = X'F0'):
SRCB = X'C1' : pour le record de contrôle SIGNON
SRCB = X'C2' : pour le record de contrôle SIGNOFF

iv. SRCB pour les records destinés à 1' imprimante
(RCB = X'94'):
10000000 : surimpression
100000nn : sauter nn lignes après 1' impression
1001nnnn : sauter au canal nnnn après 1' impression
101000nn : sauter nn lignes immédiatement
1011nnnn : sauter au canal nnnn immédiatement

v. SRCB pour les records destinés au perforateur

(RCB = X'95'):

100crrss

c =0 : code EBCDIC

c = 1 : code binaire

rr : reservé

ss : sélecteur de panier de reception

vi. SRCB pour les records du lecteur (RCB = X'93'):

100crrrr

c = 0 : code EBCDIC

c = 1 : code binaire

rrrr : reservé

e. SCB (string control byte). Cette séquence a une longueur d' un octet et décrit la chaine de caractères qui suit. Elle a le format et la signification suvante:

00000000 : signal de fin de record (EOR) et si premier dans le record signal de fin de fichier (EOF).

11111111 : chaîne de caractères normale de longueur 111111 $\leq$ 63

10111111 : chaîne d' un caractère répeté de longueur 11111 $\leq$ 31 (le caractère répeté suit)

10011111 : chaîne d' un caractère blanc répeté de longueur 11111 $\leq$ 31; le champs 'chaîne' est vide.

3.3.2.3. Quelques cas particuliers.

a. Demande d' arrêt d' un flot de transmission.

Se fait par positionnement du bit correspondant dans le FCS envoyé. Si on souhaite suspendre tous les flots il faut, alors, positionner à 1 le bit w du FCS. Le récepteur doit alors répondre soit par DLE-ACK0, soit par un "dummy" buffer (sans texte) indiquant que lui aussi ne peut plus recevoir certains flots. Si par exemple, il ne peut recevoir aucun flot le dummy buffer aura la forme suivante:

DLE-STX

BCB = X'8n' ou X'9n'

FCS = X'CFCF'

RCB = X'00'

DLE-ETB

Dans un tel cas 1' entretien se fait par de tels dummy buffers et quand 1' un de deux pourra accepter à nouveau des flots il doit reprendre en envoyant:

- soit un DLE-ACK0
- soit un dummy buffer avec FCS = X'8FCF'

b. BCB arivé erroné.

Le récepteur d' un BCB erroné doit envoyer la séquence suivante:

DLE-STX

BCB = BCB recu + X'90'

FCS = FCS normal

RCB = X'E0' (indique BCB erroné)

SRCB = BCB attendu

SCB = X'00'

DLE-ETB

c. "Time-out" sur lecture.

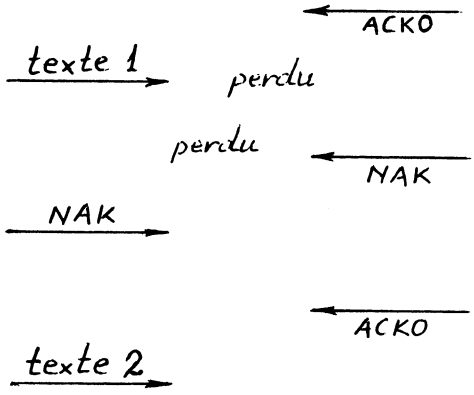
Les deux commandes d' écriture et de lecture sur la ligne étant chaînées, c' est le seul cas de "time-out" qui se pose. Sa signification est que, le récepteur n' a pu répondre dans les 3sec qui ont suivi le lancement de la lecture après la fin de l' écriture. Ayant convenu que si on n' a rien à envover on envoie dans le delai de 3sec. un DLE-ACK0 ou un dummy buffer, la seule explication est que les caractères de synchronisation ont été mal transmis et qu' on n' a pu détecter le message arrivé.

Dans un tel cas, donc, on doit repondre NAK, pour demander la retransmission du message.

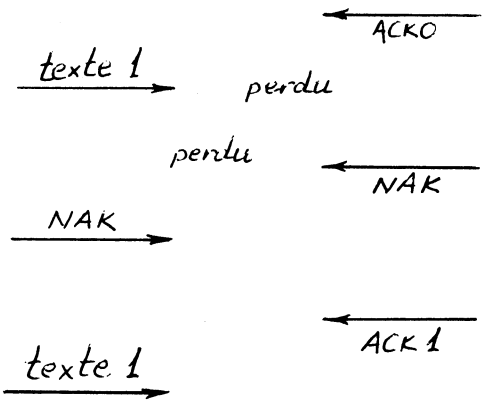
d. Remarque sur l' automate "multileaving".

Considérons le cas suivant:

Sur l' envoi d' un texte le récepteur répond 'NAK' que l' émetteur ne reçoit pas, donc, il répond 'NAK'; le récepteur alors doit envoyer la dernière séquence différente de 'NAK' que l' émetteur considerera comme acceptation de son texte qui en réalité n' a pas été reçu:



Il existe différentes méthodes pour corriger ce défaut. Nous indiquons comme exemple l' utilisation de la réponse ACK1, qui sera envoyée dans le cas de réception d' un 'NAK' sur l' envoi d' un 'NAK':



3.3.2.4. L' automate de la procédure.

Nous pouvons représenter la procédure de transmission sous la forme d' un automate fini déterministe.

a. Définition des états :

S0 : état après 1' initialisation.

S1 : test du dernier FCS arrivé.

S2 : test des caractères : ETB, BCB

S3 : renvoi de la dernière séquence avec FCS actuel.

S4 : envoi d' un NAK.

S5 : envoi d' un ACK0.

S6 : test l' existence d' un buffer à émettre.

S7 : envoi de la séquence sur BCB erroné.

S8 : envoi d' un buffer normal.

S9 : attente d' 1/10 sec. et test d' existence d' un buffer à envoyer.

S10 : envoi de la dernière sequence autre que NAK.

S11 : envoi d' un buffer dummy avec FCS actuel.

S12 : test FCS actuel (interne)

b. Définition des événements :

XX séquence reçue invalide.

DLE-STX réception de DLE-STX-text-DLE-ETB.

ACK0 réception d' un ACK0.

FCSW FCS recu avec le bit w à 1

FCS FCS recu normal.

BUFE il existe un buffer à envoyer.

BUFNE il n' existe pas un buffer à envoyer.

BCBER BCB recu erroné.

BOK bloc recu correct.

ETBER ETB erroné.

NAK réception d' un NAK.

T.O. timeout sur lecture.

INFCSW FCS interne de wait.

INFCS FCS interne normal.

l' automate "multileaving"

états \ evenements	XX	DLE-STX	ACK0	FCS_w	FCS	buff. ex.	buff. n.ex.	BCB er.	BOK	ETB er.	NAK	T.O.	FCS^{in}_{w}	FCS^{in}
S0 après initialisat.	envoi direct à S5 **													
S1 test FCS				S12	S6									
S2 test ETB, BCB								S7	S1	S4				
S3 renvoi de seq.	S4	S2	S6								S3	S4*		
S4 envoi d'un NAK	S4	S2	S6								S10	S4*		
S5 envoi d'un ACK0	S4	S2	S6								S3	S4*		
S6 test exist. buff.						S8	S9							
S7 env. BCB err. seq	S4	S2	S6								S3	S4*		
S8 envoi d'un buff.	S4	S2	S6								S3	S4*		
S9 test ex. buff. (att)						S8	S12							
S10 renv. seq. non NAK	S4	S2	S6								S3	S4*		
S11 env. dummy buff.	S4	S2	S6								S3	S4*		
S12 test FCS interne													S11	S5

* après un certain nombre de "time-out" consécutifs nous devons considerer le correspondant inactif et reprendre à l'initialisation.

** la phase d'initialisation est mise à part parce qu'elle est differente pour le terminal et pour le central

figure 31

3.4. LA GESTION DES LIGNES DANS LE SYSTEME TELCOM.

Le système TELCOM réalisé pour repondre à un besoin réel du centre de calcul de Grenoble: la connexion de la machine IBM 360/67 avec des terminaux lourds déjà existants ou prévus, a été ammené à utiliser les procédures de communication basées sur B.S.C. (protocole adopté par les constructeurs des machines à connecter).

Entre les deux procédures, 'élémentaire' et 'multileaving', la deuxième est de loin préférable,car elle offre une gestion des lignes économique et efficace. Son avantage principal: elle permet de répondre par un texte à un texte optimisant ainsi l' utilisation de la ligne. Son inconvenient: étant plus complexe elle a besoin de plus de place en mémoire que la procédure élémentaire.

Ceci dit, notre choix s' est porté rapidement sur la procédure 'multileaving'. Nous étions néanmoins obligés d' integrer aussi la procédure élémentaire pour pouvoir connecter des terminaux lourds non programmables utilisant cette procédure (type IBM 2780).

3.4.1. Le module 'LINE'.

Les tâches qui gérent les lignes de communication utilisent pour cette gestion la section de contrôle du système 'LINE'. Elle contient la description des algorithmes de deux procédures et l' aiguillage vers l' une ou l' autre. La tâche qui gère une ligne donnée exécute le code du module 'LINE' parametré à l' initialisation par l' adresse de l' UCT correspondant à cette ligne et l' aiguillage s' effectue par test sur les informations contenues dans l' UCT et son extension.

La procédure 'élémentaire B.S.C.' a été décrite au paragraphe précédent. Nous avons apporté à la procédure 'multileaving' une modification du multiplexage des enregistrements: dans les 'buffers' envoyés sur une ligne ne sont compactés que des enregistrements pour une même destination (par exemple imprimante ou perforateur).

C' est-à-dire que la tâche 'compacteur' prépare pour la tâche 'LINE' des listes des 'buffers' à émettre, à raison d' une liste par destination. Ceci a été adopté pour éviter les ennuis au cas du bloquage d' une destination. La dégradation que cette modification semble apporter à l' utilisation optimum de la ligne est sans importance puisque, les envois fonctionnant par fichiers, les enregistrements pour toute destination sont généralement en nombre suffisant pour remplir un buffer.

En revanche en réception nous acceptons des enregistrements totalement multiplexés de façon que notre procédure reste compatible avec celles déjà existantes.

Le module 'LINE' étant utilisé par toute tâche gérant une

ligne est, bien sûr, réentrant mais il a besoin des informations particulières pour chaque ligne. Il les trouve, nous l' avons déjà remarqué, dans l' UCT de chacune et son extension. L' extension d' un UCT de ligne a la forme suivante:

extention d' un U.C.T. de ligne ($LINEXT)

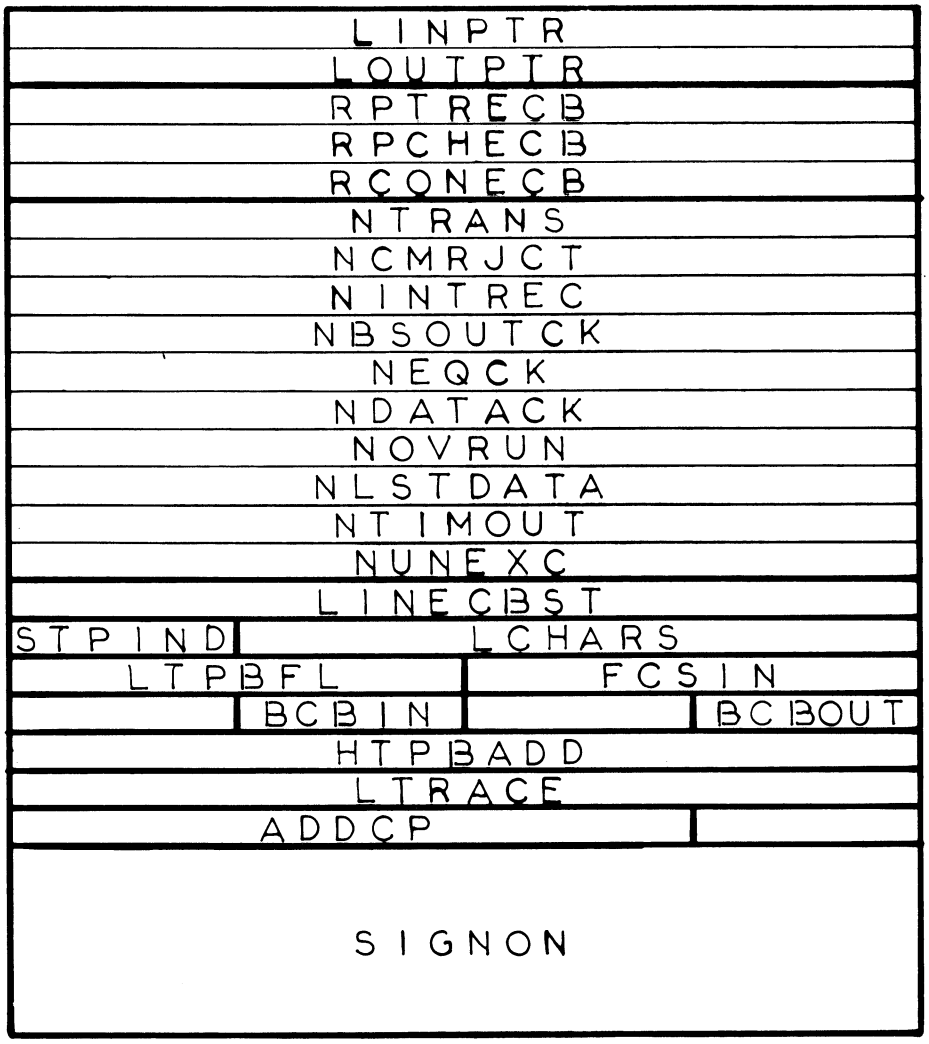


figure 32

où:

LINPTR : adresse de la liste des éléments arrivés sur la ligne.

LOUTPTR : adresse de la liste multiple des éléments à envoyer sur la ligne.

RPTRECB : ECB pour l' attente de permission à envoyer sur l' imprimante distante.

RPCHECB : ECB pour l' attente de permission à envoyer sur le perforateur distant.

RCONECB : ECB pour l' attente de permission à envoyer sur la console distante.

NTRANS : nombre de transmissions effectuées.

NCMRJCT : nb. des cond. 'command reject' sur la ligne.

NINTREQ : nb. des cond. 'intervention requested' sur la ligne.

NBSOUTCK : nb. des cond. 'bus out check' sur la ligne.

NEQCK : nb. des cond. 'equipement check' sur la ligne.

NDATACK : nb. des cond. 'data check' sur la ligne.

NTIMOUT : nb. des cond. 'time-out' sur la ligne.

NUNEXC : nb. des cond. 'unit exception' sur la ligne.

LINECBST : ECB sur ligne arretée (attente de la commande START).

STPIND : indiquation du lancement de la commande STOP.

LCHARS : caractères du nom de la ligne.

LTPBFL : longueur des buffers transmis sur la ligne.

FCSIN : séqence FCS interne.

BCBIN : séquence BCB du buffer attendu.

BCBOUT : séquence BCB du buffer à envoyer.

HTPBADD : adresse d' un buffer pris de sa liste mais pas encore envoyé à cause d' une erreur de transmission.

LTRACE : adresse d' une zone de trace des transmissions .

ADDCP : adresse de la ligne déclarée dans le système CP67.

SIGNON : carte d' identification du terminal lourd en ligne.

Nous remarquons que dans l' extension de l' UCT de la ligne nous avons les caractéristiques de l' état et du fonctionnement de la ligne, des informations concernant la transmission courante et aussi des statistiques et des traces sur les transmissions effectuées.

3.4.2. Les modules de compactage (COMPCT) et de décompactage (DCMPCT).

La tâche 'compacteur' utilise la section de contrôle 'COMPCT' pour effectuer le compactage des enregistrements pris dans ses listes d' entrée et produire des buffers qu' il met dans ses listes de sortie (listes d' entrée pour la tâche 'LINE'). Le compactage, effectué seulement pour la procédure 'multileaving', suit les règles du format des messages décrit au paragraphe 2.3.2.2. Nous précisons que nous compactons tous les n caractères consécutifs identiques et plus; où n est une variable globale du système (n 3).

Cette méthode de compactage est très efficace du point de vue compactage du message mais elle est aussi très coûteuse du point de vue temps d' exécution et elle pourrait être remplacée, si les contraintes de temps le dictaient, par une autre plus rapide.

Le compacteur compactant les enregistrements remplit un buffer pour envoyer sur la ligne; quand ce buffer est rempli ou

quand il n' y a plus d' enregistrement à compacter il met le buffer dans sa liste de sortie correspondante. Quand il ne trouve aucun enregistrement dans aucune de ses listes d' entrée, alors, nous l' avons déjà indiqué le compacteur se met en attente d' un intervalle de temps (d' une seconde).

Inversement la tâche 'décompacteur' utilise le module 'DCMPCT' pour décompacter les buffers arrivés sur les lignes. La tâche 'LINE' correspondante lui indique dans chaque buffer la ligne par laquelle est arrivé.

Le décompacteur décompacte les buffers en enregistrements et les met selon leur destination dans les listes correspondantes. Il exécute aussi l' action appropriée à la rencontre des enregistrements de contrôle. Par exemple à une demande de commencer une fonction il envoie directement la permission, car les terminaux locaux étant virtuels, ils sont toujours disponibles.

C' est pour la même raison qu' il envoie au terminaux des enregistrements décompactés, chose qui pour des terminaux réels aurait provoqué une perte de place mémoire considérable.

CHAPITRE 4

CONCLUSIONS

La nécessité de réaliser le travail présenté dans les chapitres précédants a été dictée par le besoin de faire supporter au système CP67 les terminaux lourds déjà connectés au Centre Interuniversitaire de Calcul et supportés par le système ASP/MVT.

Pour cette réalisation deux possibilités s' ouvraient à nous:

- soit modifier le système CP67 de façon à ce qu' il puisse gérer les lignes vers les terminaux lourds,
- soit se placer en utilisateur du système c'est-à-dire créer une machine virtuelle spécialisée dans la gestion des lignes qui constituerait le support des terminaux lourds pour CP67.

Nous avons choisi la deuxième voie pour différentes raisons:

- en modifiant le système CP67 nous aurions été dépendants de sa version en cours et toute nouvelle version nous aurait obligé à réconsiderer nos modifications.
- les modifications dans le but d' ajouter des tâches pourraient provoquer une dégradation de performances du système.
- il est toujours très difficile et dangereux de modifier un système en exploitation.
- en utilisant, par contre, une machine virtuelle spécialisée nous avons obtenu une indépendance de l' evolution de notre réalisation vis-à-vis du système CP67 et une plus grande facilité de la mise au point.

Pour que cette machine virtuelle puisse supporter la gestion

des lignes et les communications avec CP67, elle devait d' abord être munie d' un système de base.

La possibilité d' utiliser un système déjà existant a été vite rejetée parce qu' un tel système conçu pour couvrir certains besoins différents des nôtres aurait été soit inefficace soit très puissant pour notre application.

Nous avons, donc, décidé de construire un système multitâche, le plus simple possible, capable de fonctionner sur une machine virtuelle et d' offrir les outils nécessaires à la gestion des communications des terminaux lourds avec CP67.

Les principales contraintes d' un tel système sont:

a. Il doit être généralement en attente pour ne pas être pénalisé par CP67.

b. Il doit offrir des temps de réponse acceptables étant destiné à une exploitation réelle.

Aujourdhui, le système TELCOM, non seulement satisfait ces deux contraintes mais il nous parait avoir les qualités suivantes:

a. Un système simple.

Nous avons pu apprecier la simplicité comme une qualité fondamentale du système pendant les quelques mois de mise en exploitation, quand il a fallu chercher la cause d' une anomalie à son fonctionnement. Ce travail est vraiement simple dans un système qu' on peut bien connaître en détail.

b. Un système modulaire.

Sa modularité a considérablement facilité la mise au point et permet maintenant de le modifier facilement. En effet nous pouvons modifier des composants du système ou en ajouter de nouveaux sans que cela pose de gros problèmes.

Le système étant choisi comme système de base pour la réalisation sous CP67, d' une station de transport pour le réseau 'CYCLADES' nous avons eu l' occasion d' apprécier sa modularité dans la facilité avec laquelle on a pu effectuer son adaptation.

c. Un système multitâche généralisable.

Le système TELCOM avec quelques modifications dans l' arbitrage des tâches et aussi dans la partie liaisons avec CP67 peut dévenir un système multitâche général, gardant toujours sa simplicité.

Son utilisation pour la station de transport 'CYCLADES' a été plutôt dans cet esprit et il a donné satisfaction.

Son transport enfin pour fonctionner sous le système OS/MVT prouve qu' il peut devenir indépendant du système sous lequel il fonctionne.

Un autre aspect significatif du système TELCOM réside dans son usage systématique des listes d' éléments (ou files d' attente) comme moyens de communication entre ses tâches.

Nous pouvons ainsi agir sur les relations entre les tâches et encore sur le temps de réponse en modifiant la taille des listes. Nous donnons un exemple d' un cas réellement produit:

Ayant adopté au début des listes de la même taille pour les différentes voies virtuelles nous avons constaté que le temps de réponse pour les terminaux utilisant la procédure 'multileaving' était sensiblement supérieur de celui des terminaux utilisant la procédure élémentaire. L' explication était que les informations destinées aux terminaux utilisant la procédure élémentaire ne passaient pas le stade du compactage et, donc; leur voie était beaucoup plus rapide.

En augmentant la taille des listes dans la voie virtuelle vers les terminaux 'multileaving' nous sommes arrivés à des temps de réponse satisfaisants pour les deux directions.

Nous passons maintenent à des critiques sur quelqes points essentiels du système:

a. Le système TELCOM étant dans sa version actuelle un système essentiellement de gestion d' entrées - sorties; nous avons, donc, choisi un distributeur avec préemption afin de traiter au plus vite une interruption. Mais étant donné que c' est un système qui doit être généralement en attente nous aurions pu également choisir des tâches non interruptibles.

Le choix fait donne au système des possibilités futures, notamment la possibilité de créer des priorités entre les tâches en modifiant évidemment le module distributeur.

b. Pour assurer l' exclusion mutuelle des tâches dans l' exécution des séquences critiques nous avons employé le masquage d' interruptions pour les séquences relativement courtes et une sorte de verrou (ENQ, DEQ) pour les séquences assez longues et pour l' aquisition des ressources.

Il faut noter que la réalisation du verrou lui même est aussi faite par masquage des interruptions.

Nous avons préféré ainsi de ne pas adopter une methode standard pour l' exclusion mutuelle mais garder la souplesse d' utiliser le moyen le mieux adapté à chaque cas. Dans cette optique nous aurions pu préféré l' utilisation des sémaphores parfaitement efficaces pour la gestion des listes mais nous avons choisi de considérer les listes comme des ressources du système et utiliser le même type de verrou (ENQ, DEQ) pour ne pas alourdir le système avec de nouvelles sections de contrôle nécessaires à la réalisation de la manipulation des sémaphores.

c. Nous devons aussi mentionner la notion d' événement généralisé qui est l' élément essentiel de la synchronisation des tâches et qui par l' adjonction de la notion d' événement à mémoriser donne entière satisfaction.

d. Quand à la gestion de la mèmoire, elle est veritablement la plus simple possible (vu la petite taille du système, nous n'

avions pas de problèmes majeurs). Mais elle peut par fois gêner le bon fonctionnement du système.

Nous proposons comme première amélioration d' interdire l' acquisition d' une zone de mémoire située à cheval sur deux pages. Une telle acquisition obligerait de garder en même temps les deux pages en mémoire.

Nous pouvons encore envisager une optimisation de la disposition des différents modules et buffers du svstème de façon à ce que chaque tâche ait les modules de la description de son algorithme et ses zones de travail dans la même ou dans deux pages; nous éviterons ainsi la perte de temps pour l' échange des pages.

e. Pour notre réalisation spécifique (connexion avec des terminaux lourds existant et utilisant les procédures basées sur B.S.C.) nous étions obligés de faire notre gestion de lignes suivant le protocole B.S.C. Mais la structure du système est telle qu' on pourrait ajouter des lignes qui fonctionnent selon d' autres protocoles. Il suffit pour cela que l' on ajoute des tâches sachant gérer une ligne selon des procédures basées sur ces autres protocoles.

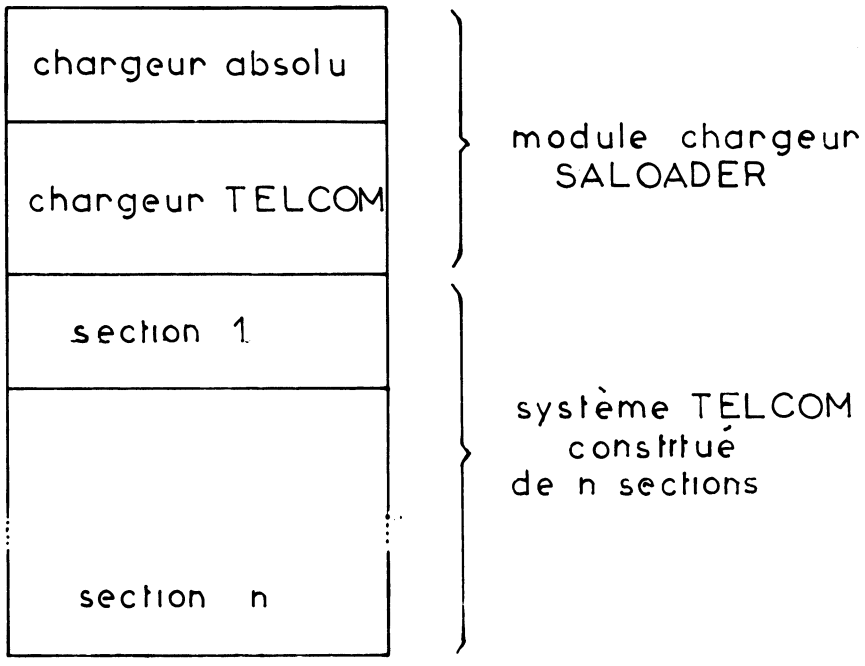
Nous devons mentionner néanmoins que les autres protocoles existant pour la gestion des lignes en 'half-duplex' et en mode synchrone sont analogues à B.S.C., et que B.S.C. peut être étendu pour gérer une ligne en 'full-duplex'.

ANNEXE I

CHARGEUR INITIAL A PARTIR DE CARTES

Ce chargeur a pour but de produire un paquet binaire autochargeable constituant le chargeur absolu. Ce chargeur absolu permet de charger le chargeur de sections du système TELCOM et de lui donner le contrôle.

La structure du paquet autochargeable est donc la suivante:



A. Le chargeur absolu

Un chargeur absolu débute par une carte de 24 octets comportant: IPLPSW, CCW1, CCW2

On peut en effet considérer la séquence d' IPL hardware comme équivalente au programme suivant:

a. Mise en mémoire 0 de CCW0 : read, 0, CC+SLI, 24

b. Mise à zéro du CAW

c. SIO sur 1' unité dont 1' adresse est affichée au pupitre.

La présence du flag de chainage de commande dans CCW0 provoque le chainage du programme canal sur la commande située à 1' adresse 8, c' est-à-dire CCW1 qui vient d' être chargé par 1' exécution de CCW0. Selon le type de chainage réalisé on peut ainsi lancer un programme canal d' un nombre quelconque de CCWS. (CCW0 à CCWn).

Lorsque ce programme canal se termine normalement le processus de chargement se termine par:

LPSW 0

En plus 1' adresse de 1' unité de chargement apparait dans le demi-mot d' adresse 2. (c' est-à-dire dans la partie code interruption du PSW).

La première carte du paquet autochargeur est obtenue par 1' assemblage de la séquence suivante:

```
IPLPSW    PSW   0,0,IO
CCW1      CCW   Read,0,SLI+SKP+CD,16
CCW2      CCW   0,CCW3,SLI+CC,56
          DS    OD
CCW3      ...
```

Assemblée, cette séquence produit une carte TXT qui contient en zone variable les 24 octets désirés. Par duplication décalée à gauche de 16 colonnes cette carte produit la carte de 24 octets de tête du chargeur.

Cette opération se réalise facilement en édition par contexte.

Une fois cette carte obtenue on peut par assemblage conditionnel la reproduire directement par un ordre REPRO. (variable d' assemblage &IPLCARD).

La chaîne de CCW initiale comporte 8 CCW, associés par paires, chaque paire lisant une carte texte.

Chaque paire de CCW comporte la séquence suivante:

```
CCWn    02,0,CD+SKP+SLI,16
CCWn+1  00,CCW3+1/2(n-2),CC+SLI,56
```

Le premier CCW provoque la lecture sans transfert d' information de 16 octets. Le second CCW provoque la lecture des 56 octets de la zone variable d' une carte texte (col. 17 à 72).

La chaîne de 8 CCWS de ce type permet de lire 4 cartes texte soit 4x56=224 octets. Associé à la première carte de 24 octets, c' est un total de 248 (X'0F8') octets qui sont lus par la séquence de chargement initial.

Ceci limite le chargeur absolu à 248 octets ce qui est suffisant.

On remarquera que le chargeur absolu vient se placer dans la PSA (permanent storage area- mémoires spécialisées du S/360). Ceci est un moyen d' initialiser certaines mémoires et en particulier: le CAW, le TIMER, le program check new PSW, l' I/O interrupt new PSW.

Il faut faire attention de ne pas placer de code dans les mémoires qui peuvent évoluer par action du hardware (Timer, CSW etc.).

Le PSW initial passe contrôle en mode superviseur,masqué, à l' adresse L0 où nous trouvons une boucle de mesures de la capacité mémoire de la machine. Cette boucle se termine par une interruption programme de type 'addressing' qui donne le contrôle à l' adresse L1.

On sauvegarde alors la taille mémoire disponible et on implante un buffer de 80 caractères à l' adresse la plus élevée possible. On met à jour un CCW de lecture (CCW9) et le CAW pour pouvoir lire d' autres cartes. On donne alors le contrôle au chargeur absolu qui lit des cartes binaires jusqu' à rencontre d' une carte END.

La zone variable des cartes TXT est mouvementée en mémoire.

La carte END donne le contrôle à l' adresse du point d' entrée spécifié sur la carte END d' assemblage.

B. Le chargeur TELCOM.

Vient immédiatement à la suite du chargeur absolu (provient en fait du même assemblage). Ce chargeur commence par implanter la TVT (nombre d' entrées égal à la variable d' assemblage &LEN)

puis va lire une suite de paquets binaires obtenus par 1' assemblage séparé de chaque section du système.

Le chargement se termine sur la condition de fin de fichiers après la lecture d' une carte END.

Les cartes autorisées sont: ESD, TXT, RLD, END, ALI, REP.

C. Traitement des cartes ESD.

Le noms des sections de contrôle générés par le système de macro-instruction du système TELCOM sont du type:

XXXXXXnn ù XXXXXX représente 1 à 6 caractères alphanumériques dont le premier est une lettre
et nn un entier de 00 à 99 représentant le numéro de 1' entrée dans la TVT pour cette section.

On verifie:

-que le nom de la section a la syntaxe correcte

-que 1' entrée correspondante de la TVT est bien vide

-que la longueur de la section rentre bien dans la mémoire disponible.

Pour faciliter la mise au point, on introduit en tête du code relatif à une section une entrée de 16 octets de la 'map' du chargeur comportant: le nom de la CSECT (8 caractères), le facteur de translation (un mot), et la longueur de la CSECT (un mot). Ce qui permet à la lecture de 1' image mémoire de retrouver rapidement une section et de se repérer par rapport au listing d' assemblage.

On peut encore accepter les cartes à entrées multiples à condition qu' elles proviennent d' un même ssemblage et soient assemblées continûment (pas d' ordre ORG ou START), en effet dans

ce cas toutes les SCECT ont le même facteur de translation.

D. Traitement des cartes TXT.

Il suffit de mouvementer le contenu de la zone variable à l' adresse d' assemblage modifiée par le facteur de translation.

E. Traitement des cartes RLD.

On limite leur traitement à celui des adresses constantes locales à une CSECT. Pour cela on vérifie que le pointeur d' adresse et le pointeur de relocation réferent bien la même entrée du dictionnaire de symboles externes et que cette entrée est bien celle correspondant à la dernière carte ESD traitée.

Il faut alors prendre le contenu du pointeur d' adresse translaté, y ajouter le facteur de translation et remettre en mémoire 2, 3, ou 4 octets selon la longueur de l' adresse constante.

F. Traitement des cartes ALI.

Cette carte permet de forcer l' adresse de première mémoire libre à la frontière de page immédiatement supérieure. Ceci est prévu pour permettre l' optimisation du système sous CP67 et faciliter aussi la lecture d' un champ mémoire en correspondance avec la liste d' assemblage.

G. Traitement des cartes REP.

Des cartes 'replace' simplifiées du type suivant sont acceptées:

.REP AAAA MMMM,MMMM etc.

L' adresse AAAA est une adresse de demi mot relative à l' origine de la SCECT.

MMMM représente le nouveau contenu d' une suite de demi-mots à partir de l' adresse AAAA.

H. Traitement de la carte END.

Aucun traitement particulier n' est effectué.

I. Ordre des cartes binaires:

ALI, ESD, TXT, REP1, RLD, REP2, END,EOF

Les cartes ALIGN peuvent toujours être omises; elles doivent apparaître avant la carte ESD de la CSECT à aligner.

Il y a toujours au moins une carte ESD par section. Il peut y avoir des sections vides sans cartes TXT; dans ce cas on aura la suite ESD, END. Les cartes REP doivent arriver après la carte TXT qu' elles modifient, donc par simplicité après les cartes TXT.

Si une carte REP modifie une adresse constante elle doit apparaître avant la carte RLD correspondante.

END est la dernière carte d' une section.

EOF ne doit apparaître qu' après END.

En fin de chargement appel de la phase d' initialisation du système TELCOM avec passage en paramètre des différents pointeurs de mémoire utiles pour récupérer la place du chargeur et l' espace libre en mémoire.

ANNEXE II

MANUEL OPERATEUR

Pour la mise en marche du système TELCOM sous CP67, nous devons d' abord activer la machine virtuelle TELCOM (mot de passe COMTEL). Ensuite, sur la machine TELCOM, nous effectuons un IPL sur le disque à l' adresse 194.

L' image mémoire du système qui existait sur ce disque se trouve maintenent en mémoire et le module d' initialisation demande quel système à initialiser:

SYSTEME = TELCOM OR ST OR TEST

Avec la réponse : TELCOM le système est initialisé et nous pouvons mettre la machine TELCOM en 'disconnect' si on veut.

Pour attacher le télétype opérateur, nous devons lancer sur un télétype réel la commande: DIAL TELCOM

Nous recevons la réponse:

..CONNECTED..

TYPE YOUR NAME PLEASE:

Le nom demandé est: TELCOM et la réponse du système:

NAME ACCEPTED TELCOM WAITS FOR YOUR COMMANDS

TYPE ATTN TO DEBLOCK KEYBOARD

A partir de ce moment le télétype opérateur est connecté au système TELCOM et nous pouvons utiliser un jeu de commandes pour demander certaines fonctions:

Jeu de commandes du télétype opérateur:

1. START arg

La commande START est utilisée pour démarrer une ligne. arg représente le nom de la ligne; les noms acceptés sont:

LMCF : ligne vers le IBM/1130 de la Mecanique des Fluides

LIPG : ligne vers le T1600 de l' I.N.P.G.

LORD : ligne vers l' ordoproceseur de la MIAGE.

LDEG : ligne vers l' ordoprocesseur du DEUG.

LTLM : ligne vers le T1600 de la Télémecanique.

Le demarrage de la ligne signifie que le système demande à l' opérateur d' attacher la ligne à la machine TELCOM (si ce n' est pas déjà fait) et lance le programme canal d' initialisation sur elle.

Réponses possibles:

COMMAND ACCEPTED ou INVALID COMMAND

2. STOP arg

Cette commande est utilisée pour arreter une ligne. Les arguments acceptés sont les mêmes que pour la commande START.

L' arrêt de la ligne s' effectue après la fin de la transmission courante sur elle.

Réponses possibles:

COMMAND ACCEPTED ou INVALID COMMAND

3. STAT arg

Cette commande provoque la sortie sur le télétype opérateur des statistiques des transmissions effectuées sur la ligne donnée en argument.

Les arguments acceptés sont les mêmes que pour la commande START. Les statistiques à sortir ont la forme suivante:

NUMBER OF TRANSMISSIONS : xxxxxxxx

NUMBER OF COMMAND REJECT xxxxxxxx

NUMBER OF INT/VENTION REQ xxxxxxxx

NUMBER OF BUS OUT CHECK xxxxxxxx

NUMBER OF EQUIPEMENT CHK xxxxxxxx

NUMBER OF DATA CHECK xxxxxxxx

NUMBER OF OVERRUN xxxxxxxx

NUMBER OF LOST DATA xxxxxxxx

NUMBER OF TIME OUT xxxxxxxx

NUMBER OF UNIT EXCPTION xxxxxxxx

où xxxxxxxx : un nombre hexadécimale.

Réponses possibles:

les statistiques (les entrées les différentes à zéro) ou

INVALID COMMAND

4. MSG TO arg1 arg2

Cette commande a pour but d' envoyer un message à l' opérateur du site arg1; arg1 pouvant être:

MCF pour Mecanique de Fluides

IPG pour l' INPG.

ORD pour l' ordoprocesseur de la MIAC.

DEG pour l' ordoprocesseur du DEUG.

TLM pour la Télemecanique.

et arg2 est le message à envoyer.

Réponses possibles:

COMMAND ACCEPTED ou

REMOTE CONSOLE NOT RECEIVING ou

INVALID COMMAND

5. DUMP

Avec cette commande nous demandons la sortie d' une image de la mémoire entière de la machine TELCOM sur l' imprimante locale.

C' est une commande utile pour la mise au point et la détection d' erreurs.

6. DISPLAY arg1 arg2

où arg1 et arg2 sont deux nombres hexadécimaux inférieurs à 40000.

Cette commande provoque la sortie sur le télétype opérateur d' une image de la mémoire (par mots) à partir de l' adresse arg1 jusqu' à l' adresse arg2.

Si arg1 et arg2 ne désignent pas des frontières de mot il y a arrondi, et s' il n' existe pas de deuxième argument le mot d' adresse arg1 est affiché.

Réponses possibles:

image de la mémoire demandée ou

INVALID COMMAND

7. STORE arg1 arg2

où arg1 et arg2 sont deux nombres hexadécimaux, 0 arg1 40000 et arg2 jusqu' à 8 chiffres hexadécimaux.

Cette commande provoque l' écriture en mémoire et dans le mot d' adresse arg1 du nombre hexadécimal arg2.

arg1 doit designer une frontière de mot.

Réponses possibles:

COMMAND ACCEPTED ou INVALID COMMAND

8. MAP arg1

où arg1 est le nom alphanumérique d' une section de contrôle du système.

Cette commande affiche sur le télétype opérateur le 'map' de la section de contrôle nommée en arg1. C' est-à-dire les informations concernant son chargement: son nom qualifié, son facteur de translation, et sa longueur

exemple: MAP LINE affiche:

LINE17 002700 000A70

Réponses possibles:

affichage de la 'map' ou INVALID COMMAND

9. SIZE

Avec cette commande nous demandons l' état de la mémoire libre.

La réponse est l' affichage sur le télétype opérateur de la

chaîne des FQE désignant les différentes zones de la mémoire libre partant du FQEORG (adresse du premier FQE).

10. CANCEL arg1 arg2

où arg1 = XXX est l' identification d' un terminal lourd (ex. MCF ou IPG)

et arg2 = PRINT ou PUNCH

Cette commande provoque l' arrêt d' une sortie sur le terminal lourd (imprimante ou perforateur).

11. DISABLE

Avec cette commande nous effectuons le détachement du télétype opérateur du système TELCOM (pour le rattacher nous devons effectuer un DIAL TELCOM).

Réponse : CP LOGOUT

ANNEXE III

MANUEL DE MAINTENANCE

Cette partie a pour but de décrire la manière selon laquelle nous pouvons apporter des modifications au système TELCOM.

Nous précisons dès le début que nous avons deux disques sur lesquels nous pouvons faire l' IPL du système. Ils sont les disques d' adresses 194 et 195.

Nous transfèrons le système sur ces disques par la procédure de chargement utilisant la machine virtuelle DUMASLE sur laquelle existent tous les modules du système.

Nous avons mis en exploitation normale le disque 194 attaché à la machine TELCOM.

Après toute modification du système (sur la machine DUMASLE) nous chargeons le nouveau système sur le disque 195 attaché à la machine virtuelle TELCOMO avec laquelle nous effectuons tous les tests. Une fois le nouveau système testé et jugé satisfaisant nous pouvons le charger sur le disque 194 pour le mettre en exploitation.

Nous distinguons quatre sortes des modifications:

A. Modifications portant sur des modules du système déjà existant.

C' est le type de modifications qui pose le moins de problèmes. Il suffit de réassembler les modules modifiés, de régénérer le système (commande EXEC SYSGEN) et de le charger sur

disque (commande EXEC ACTIV).

B. Addition des nouveaux modules qui ne sont pas des tâches.

Tout nouveau module du système TELCOM doit respecter les règles de noms alphanumériques et qualifiés (utilisation de la macro: $CSECT) et les conventions de liaison inter-section.

Son nom alphanumérique doit être affecté à une place libre du tableau &BOX des variables globales du système (fichier GBLVAR SYSIN).

Nous devons, bien sûr, assembler ce module et tous les autres faisant appel à lui et aussi modifier respectivement et réassembler le fichier CREIPL SYSIN qui définit les modules du système à charger.

Enfin, nous changeons les caractéristiques du fichier CREIPL TEXT en SYSGEN EXEC, nous générons le système (SYSGEN) et nous le chargeons sur disque (ACTIV).

C. Addition des nouveaux modules qui seront des nouvelles tâches du système.

Dans ce cas nous suivons exactement la procédure précédente et en plus nous devons modifier la variable globale &TCBMAX (nombre maximum de tâches), la section de contrôle TCBS pour ajouter les TCBS correspondant aux nouvelles tâches et le module INIT pour les initialiser.

D. Addition d' une unité d' entrée-sortie.

L' addition d' une unité d' entrée-sortie entraine l' addition d' une tâche gérant cette unité , ce qui revient à effectuer les manipulations décrites précédemment.

De plus nous devons ajouter dans la section de contrôle UCTS le bloc de controle de l' unité ajoutée (UCT) éventuellement son extension, modifier le nombre des unités (&NDEV dans les variables globales du système) et, enfin, modifier le module d' initialisation (INIT) pour initialiser la nouvelle tâche, l' UCT de l' unité, son extension etc.

Nous remarquons , ici, qu' en général l' addition d' une unité nécessite aussi l' addition des listes d' éléments et quelque-fois des voies virtuelles (s' il s' agit d' addition d' une ligne par exemple) ,donc, des changements dans la section de contrôle LISTS et des modifications dans l' initialisation des tables de communication du système.

ANNEXE IV

DESCRIPTION DES PRIMITIVES EN PSEUDO-ALGOL

'WAIT'

```
begin
   if ECBCODE = 'completed'  then
     begin
       ECB := 0 ; return ;
     end ;
   enter masked mode ;
   if ECBCODE = 'completed'  then
     begin
       ECB := 0 ; TSTATUS := 'ready' ; go to 'distributeur';
     end ;
   ECBADDR := TCB ; ECBCODE := 'awaited' ;
   TSTATUS := 'waiting' ; go to 'distrubuteur'
end.
```

'POST'

```
begin
   ECBCODE := ECBCODE or 'completed' ;
   enter masked mode ;
   if ECBCODE = 'awaited'  then go to RETURN ;
   TSTATUS := 'ready' ; ECB := 0 ;
 RETURN: restore mask ; return
end.
```

'GTMN'

```
begin
   if R13 = 0 then store registers in local savearea
              else begin
                     store registers in TCB ;
                     enter masked mode ;
                   end ;
   increase R1 to double-word multiple ;
   take first FQE ;
COMPARE: if R1 ≤ number of free bytes then go to SATISF ;
          if last FQE then go to NOTSAT ;
          take next FQE ; go to COMPARE ;
SATISF: if R1 = number of free bytes then go to GIVALL ;
        number of free bytes := nb. of free bytes - R1 ;
EXIT: put field address in R1 ; clear area ;
RETURN: restore registers - R1 ; reset mask ; return ;
GIVALL: erase FQE from queue ; go to EXIT ;
NOTSAT: R1 := task's entry in ECB's list ; R1 := -R1 ;
         go to RETURN
end.
```

'FRMN'

```
begin
   if R13 = 0 then store registers in local savearea
              else begin
                     store registers in TCB ;
                     enter masked mode ;
                   end ;
   increase R0 to double-word multiple ;
   FQEADD := FQEORG ;
NEXTFQE: if FQEADD = 0 then go to CREATE ;
          if FQEADD+8 = R1 then go to JUSTOVER ;
          if FQEADD+8 < R1 then
              begin take next FQE ; go to NEXTFQE ; end ;
LOW: if free space add. = end of zone then go to CONNECT ;
     if free space add. > end of zone then go to CREATE ;
     go to ERROR ;
CONNECT: FQENBR := FQENBR+R0 ; go to POST ;
CREATE: FQENBR := R0 ; FQEADD := add. in previous FQE ;
         add. in previous FQE := created FQE add. ;
         go to POST ;
JUSTOVER: new FQE add. := add. of old FQE +R0 ;
         if new FQE add. = next FQE add.-8 then
                                        go to CONALL ;
         if new FQE add. <next FQE add.-8 then
                                        go to FILLFQE ;
         go to ERROR ;
```

```
CONALL: FQENBR := sum of bytes (tow FQE's+R0) ;
        add. in previous FQE := add. of second FQE ;
        go to POST ;
FILLFQE: FQENBR := FQENBR+R0 ;
         FQEADD := add. in previous FQE ;
         add. in previous FQE := add. of FQE ;
POST: N := 0 ;
LOOP: if N = TCBMAX then go to RETURN ;
      if GTMNECB+N ≠ 0 then call POST ;
      N := N+1 ; go to LOOP ;
RETURN: restore registers ; reset mask ; return ;
ERROR: call special SVC
end.
```

```
                                'ENQ'
begin
   save registers ; enter masked mode ;
   R7 := R1 ; R1 := ABS(R1) ;
   ressource entry := TABRES+R1-1 ;
   if first byte = X'80' then go to TESTWAIT ;
   first entry word := TCB address ;
   first byte := X'80' ; R1 := 0 ;
EXIT: reset mask ; restore registers - R1 ; return ;
TESTWAIT: if R7 > 0 then go to ENQUEUE ;
          if R7 = 0 then go to ERROR ;
          R1 := R7 ; go to EXIT ;
ENQUEUE: task entry in RESECB := RESECB+TASKID-1 ;
          if entry ≠ 0 then go to ERROR ;
          POINTER := ressource entry pointer ; N := 0 ;
LOOP: N := N+1 ;
      if POINTER = 0 then go to ADDECB ;
      if N = TCBMAX then go to ERROR ;
      POINTER := POINTER+4 ; go to LOOP ;
ADDECB: POINTER := entry in RESECB for the task ;
        R1 := ECB address ; ECB := TCB address ;
        go to EXIT ;
ERROR: call special SVC
end.
```

'DEQ'

```
begin
   save registers ; enter masked mode ;
   ressource entry := TABRES+R1-1 ;
   if TCB add. in entry = cur. TCB add. then go to DEQUEUE ;
   go to ERROR ;
 DEQUEUE: if queue pointer = 0 then go to RESFREE ;
          ECB := entry pointed by pointer ;
          call POST ; go to EXIT ;
 RESFREE: ressource entry := 0 ;
 EXIT: reset mask ; restore registers ; return ;
 ERROR: call special SVC
end.
```

INDEX BIBLIOGRAPHIQUE

Bellino (J) - Superviseurs d' entrée-sortie dans un contexte de multiprogrammation.
Cours C4 maîtrise d' Informatique, IMAG 1970.

Bellino (J) - Mécanismes de base dans les systèmes hyperviseurs: Conception et réalisation d' un système à accès multiples.
Thèse 'docteur es sciences appliquées'.
U.S.M.G., Septembre 1973.

Bellot (M), Guillou (J), Le Heiget (J.P.), Rey (M), Siret (L), Dupuy (J.P.) - Systèmes de programmation générateurs de machines virtuelles.
C.I.C.G., Decembre 1973.

Denning (P.J.) - Third generation computer systems.
ACM. Computing Surveys, Vol. 3, No. 4, December 1971

Dijkstra (E.W.) - Solution of a problem in concurrent programming control.
CACM., Vol. 3, No. 9, September 1965.

Dijkstra (E.W.) - The structure of 'THE' Multiprogramming System.
CACM., Vol. 11, No. 5, May 1968.

Habermann (A.N.) - Prevention of system deadlocks.
CACM., Vol. 12, No. 7, July 1969.

Hansen (P.B.) - The nucleus of a Multiprocessing System.
CACM., Vol. 13, No. 4, April 1970.

Hoare (C.A.R.), Perrot (R.H.) - Towards a theory of parallel programming in Operating Systems technics.
Academic Press, New York, 1973.

I.B.M. - System/360 Principles of Operation (A22-6821)

I.B.M. - System/360 Model 67 Functional Characteristics (A27-2719)

I.B.M. - System/360 Component Description - 2314 Direct Acces Storage Facility and 2844 Auxiliary Storage Control (A26-3599)

I.B.M. - 2701 Data Adapter Unit Component Description (A22-6864)

I.B.M. - 2740/2741 Communications Terminal-Operator's Guide (A27-3001)

I.B.M. - 2780 Data Transmission Terminal - Component Description (A27-3005)

I.B.M. - OS/360 System Programmer's Guide (C28-6550)

I.B.M. - OS/360 Concepts and Facilities (C28-6551)

I.B.M. - OS/360 Loader (C28-6713)

I.B.M. - System/360 Introduction to Teleprocessing (C30-2007)

I.B.M. - Binary Synchronous Communications (GA27-3007)

I.B.M. - Houston Automatic Spooling Priority system
(360D-05.1.014)

Knuth (D.E.) - Additional Comments on a Problem in Concurrent Programming Control.
CACM., Vol. 9, No. 5, May 1966.

Knuth (D.E.) - The art of computer programming
Adisson Wesley Reading, Mass. 1968.

Krakowiak (S) - Conception et réalisation de systèmes à accès multiples: Allocation de ressources.
Thèse d' Etat, Université de Paris-VI, Fevrier 1973.

Lampson (B.W.) - A Scheduling Philosophy for Multiprocessing systems.
CACM., Vol. 11, No. 5, May 1968.

Organick (E) - The multics system: an examination of its structure.
MIT. Press, 1972.

Presser (L); White (J.R.) - Linkers and loaders.
ACM. Computing surveys, Vol. 4, No. 3, September 1972

Wirth (N) - A note on program structures for parallel processing.
CACM., Vol. 9, No. 5, May 1966.

Wirth (N) - On Multiprogramming, Machine coding, and Computer Organization.
CACM., Vol. 12, No. 9, September 1969.