Langage de haut-niveau de programmation des organes centraux d'un autocommutateur téléphonique

Amal Olaiwan - El Karah

THESE

présentée à

Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR INGENIEUR

par

Amal OLAIWAN – EL KARAH

LANGAGE DE HAUT NIVEAU DE PROGRAMMATION DES ORGANES CENTRAUX D'UN AUTOCOMMUTATEUR TELEPHONIQUE

Thèse soutenue le 26 juin 1979 devant la commission d'examen

L. BOLLIET	Président
F. ANCEAU Ph. JORRAND J.M. PITIE	Examinateurs

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1977-1978

Président : M. Philippe TRAYNARD

Vice-présidents : M. René PAUTHENET
M. Georges LESPINARD

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Electronique - automatique
	BESSON Jean	Chimie minérale
	BLOCH Daniel	Physique du solide - cristallographie
	BONNETAIN Lucien	Génie chimique
	BONNIER Etienne	Métallurgie
*	BOUDOURIS Georges	Electronique - automatique
	BRISSONNEAU Pierre	Physique du solide - cristallographie
	BUYLE-BODIN Maurice	Electronique - automatique
	COUMES André	Electronique - automatique
	DURAND Francis	Métallurgie
	FELICI Noël	Electronique - automatique
	FOULARD Claude	Electronique - automatique
	LANCIA Roland	Electronique - automatique
	LONGEQUEUE Jean-Pierre	Physique nucléaire corpusculaire
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie - physique
	PAUTHENET René	Electronique - automatique
	PERRET René	Electronique - automatique
	POLOUJADOFF Michel	Electronique - automatique
	TRAYNARD Philippe	Chimie - physique
	VEILLON Gérard	Informatique fondamentale et appliquée

* en congé pour études

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuël	Electronique - automatique
	BOUVARD Maurice	Génie mécanique
	COHEN Joseph	Electronique - automatique
	GUYOT Pierre	Métallurgie physique
	LACOUME Jean-Louis	Electronique - automatique
	JOUBERT Jean-Claude	Physique du solide - cristallographie

.../...

MM.	ROBERT André	Chimie appliquée et des matériaux
	ROBERT François	Analyse numérique
	ZADWORNY François	Electronique - automatique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	CHARTIER Germain	Electronique - automatique
	CHIAVERINA Jean	Biologie, biochimie, agronomie
	IVANES Marcel	Electronique - automatique
	LESIEUR Marcel	Mécanique
	MORET Roger	Physique nucléaire - corpusculaire
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	SABONNADIERE Jean-Claude	Informatique fondamentale et appliquée
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie Physique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahim	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique)

E.N.S.E.E.G.

MM.	BISCONDI Michel	Ecole des Mines St. Etienne (dépt. Métallurgie)
	BOOS Jean-Yves	Ecole des Mines St. Etienne (Métallurgie)
	DRIVER Julian	Ecole des Mines St. Etienne (Métallurgie)

.../...

MM.	KOBYLANSKI André	Ecole des Mines St. Etienne (Métallurgie)
	LE COZE Jean	Ecole des Mines St. Etienne (Métallurgie)
	LESBATS Pierre	Ecole des Mines St. Etienne (Métallurgie)
	LEVY Jacques	Ecole des Mines St. Etienne (Métallurgie)
	RIEU Jean	Ecole des Mines St. Etienne (Métallurgie)
	SAINFORT	C.E.N. Grenoble (Métallurgie)
	SOUQUET	U.S.M.G.
	CAILLET Marcel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	COULON Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	GUILHOT Bernard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LALAUZE René	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LANCELOT Francis	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SARRAZIN Pierre	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SOUSTELLE Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THEVENOT François	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THOMAS Gérard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TOUZAIN Philippe	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TRAN MINH Canh	Ecole des Mines St. Etienne (Chim. Min. Ph.)

E.N.S.E.R.G.

MM.	BOREL	Centre d'études nucléaires de Grenoble
	KAMARINOS	Centre national recherche scientifique

E.N.S.E.G.P.

M.	BORNARD	Centre national recherche scientifique
Mme	CHERUY	Centre national recherche scientifique
MM.	DAVID	Centre national recherche scientifique
	DESCHIZEAUX	Centre national recherche scientifique

Je remercie Monsieur L. BOLLIET, Professeur à l'I.U.T. de Grenoble, qui m'a fait l'honneur de présider le jury de ma thèse.

Monsieur F. ANCEAU, Maître de Conférences à l'ENSIMAG, a été mon Directeur de recherches. Je le remercie pour ses encouragements, ses critiques, ainsi que pour la confiance qu'il m'a témoignée.

Je tiens à exprimer ma gratitude à Monsieur P. JORRAND, Maître de Recherches à l'ENSIMAG, pour ses conseils et ses encouragements qui ont toujours été précieux, et pour les nombreuses discussions que nous avons eues sur cette étude.

Je remercie Monsieur J.M. PITIE, Directeur du Groupe Architecture et Fonction de Commande au C.N.E.T.-Lannion, qui a bien voulu s'intéresser à mon travail et a accepté de le juger.

C'est grâce à Monsieur J.F. PONS que j'ai pu aborder et étudier les problèmes téléphoniques. Je le remercie vivement pour l'attention et le soutien qu'il a accordés à ce travail.

Je félicite Madame C. CHALAND pour son dévouement et sa rapidité à frapper ce texte.

Je remercie le service de reprographie de l'IMAG qui a assuré le tirage de ce document.

TABLE DES MATIERES

CHAPITRE 1 - INTRODUCTION

1. AVANT PROPOS P. 1
2. HISTORIQUE P. 2
3. PROGRAMMATION DES SYSTEMES DE COMMUTATION A PROGRAMME ENREGISTRE P. 3
4. OBJECTIFS POURSUIVIS P. 5
5. DEMARCHE SUIVIE P. 7

CHAPITRE 2 - STRUCTURE DES ORGANES DE COMMANDE

1. INTRODUCTION P. 9
2. MODULES DE COMMANDE P. 11
3. ORGANISATION D'UN PROCESSEUR DE TRAITEMENT P. 13
   - 3.1. DONNÉES P. 13
   - 3.2. PROGRAMME TÉLÉPHONIQUE P. 13
4. ORGANISATION DU LOGICIEL D'UN MODULE DE COMMANDE P 14
   - 4.1. COUCHE D'APPLICATION P. 14
   - 4.2. COUCHE DE SERVICE P. 14
   - 4.3. COUCHE INTERMÉDIAIRE P. 15
   - 4.4. COUCHE ÉLÉMENTAIRE P. 15
5. GESTION DES ÉCHANGES ENTRE LES DIFFERENTES COUCHES DU LOGICIEL P. 16
   - 5.1. RECEPTION D'UN MESSAGE P. 16
   - 5.2. EMISSION D'UN MESSAGE P. 17
   - 5.3. ATTENTE D'UN MESSAGE P. 18

CHAPITRE 3 - LANGAGE DE PROGRAMMATION

1. INTRODUCTION P. 20
2. NOTATION ET VOCABULAIRE P. 22
3. STRUCTURE DU PROGRAMME P. 24
4. DONNÉES P. 25
   4.1. DÉFINITION DES CONSTANTES P. 26
   4.2. DÉFINITION DES TYPES P. 27
      4.2.1. TYPE SIMPLE P. 27
         4.2.1.1. TYPE SCALAIRE P. 29
         4.2.1.2. TYPE BOOLEEN P. 29
         4.2.1.3. TYPE ENTIER P. 30
         4.2.1.4. TYPE INTERVALLE P. 30
      4.2.2. TYPE STRUCTURÉ P. 31
         4.2.2.1. TYPE ENREGISTREMENT P. 31
         4.2.2.2. TYPE TABLEAU P. 32
         4.2.2.3. TYPE ENSEMBLE P. 33
      4.2.3. TYPE POINTEUR P. 33
   4.3. DÉCLARATION DES VARIABLES P. 35
   4.4. DÉCLARATION DES EXCEPTIONS P. 36
   4.5. DÉCLARATIONS DES PROCÉDURES P. 37
5. INSTRUCTIONS P. 39
   5.1. LES INSTRUCTIONS COURANTES P. 39
      5.1.1. LES INSTRUCTIONS SIMPLES P. 40
         5.1.1.1 INSTRUCTION D'AFFECTATION P. 40
         5.1.1.2 INSTRUCTION D'APPEL DE PROCÉDURE P. 42
         5.1.1.3. INSTRUCTION DE PROVOCATION D'EXCEPTION P. 42
         5.1.1.4. INSTRUCTION VIDE P. 43

5.1.2. LES INSTRUCTIONS STRUCTURÉES P. 44
5.1.2.1. INSTRUCTION COMPOSÉE P. 44
5.1.2.2. INSTRUCTION CONDITIONNELLE P. 44
5.1.2.3. INSTRUCTION RÉPÉTITIVE P. 46
5.2. LES INSTRUCTIONS SPÉCIALISÉES P. 49
5.2.1. INSTRUCTION QUAND P. 50
5.2.2. INSTRUCTION QUANDNON P. 51
5.2.3. INSTRUCTION SUIVANT P. 51
5.2.4. INSTRUCTION ENVOI P. 52
5.2.5. INSTRUCTION RECEVOIR P. 53
5.2.6. INSTRUCTION AVANT-ATTENDRE P. 54
5.2.7. INSTRUCTION TERMINER-PROCESSUS P. 55

CHAPITRE 4 - NOTES SUR LA LANGAGE PROPOSE

1. INTRODUCTION P. 57
2. COMMENTAIRES, IDENTIFICATEURS ET NOMBRES P. 58
3. STRUCTURE DU PROGRAMME P. 59
4. DONNÉES P. 60
4.1. DÉFINITION DES CONSTANTES P. 60
4.2. DÉFINITION DES TYPES P. 60
4.2.1. TYPES SIMPLES P. 61
4.2.2. TYPES STRUCTURÉS P. 62
4.2.2.1. TYPE ENREGISTREMENT P. 62
4.2.2.2. TYPE TABLEAU P. 62
4.3. DÉCLARATION DES EXCEPTIONS P. 63
4.4. DÉCLARATION DES PROCÉDURES P. 63
5. INSTRUCTIONS COURANTES P. 64
6. INSTRUCTIONS SPÉCIALISÉES P. 66

CHAPITRE 5 - IMPLEMENTATION

1. INTRODUCTION P. 69
2. NOTES PRÉLIMINAIRES P. 70
2.1. LES MACROS QUI GÉNÈRENT DU CODE EXÉCUTABLE P. 71
2.2. LES MACROS QUI FONT APPEL AUX MODULES SYSTÈME P. 72
2.3. LES MACROS QUI FIGURENT DANS D'AUTRES MACROS P. 73
2.4. LES PROCÉDURES P. 73
3. FORMATS INTERMÉDIAIRES ET ORGANIGRAMMES DES INSTRUCTIONS DU LANGAGE P. 75
CHAPITRE 6 - CONCLUSION P. 95

ANNEXE : PROGRAMME TÉLÉPHONIQUE

REFERENCES

BIBLIOGRAPHIE

CHAPITRE 1

INTRODUCTION

1. AVANT-PROPOS

Les progrès de l'informatique et les possibilités apportées par les technologies modernes, on ouvert dans le domaine de la commutation téléphonique, la voie à la technique du programme enregistré. Ceci a permis une plus grande souplesse d'utilisation et une extension des applications et des services fournis à l'usager.

Le sujet de la présente thèse est la conception d'un langage de programmation pour les organes de commande d'un autocommutateur téléphonique décentralisé.

Cette étude est poursuivie par l'équipe Architecture d'Ordinateurs du Laboratoire IMAG, en parallèle avec celle menée sur la structure matérielle des organes de commande. Le langage présenté dans cette thèse est basé sur cette structure décentralisée.

Nous décrivons la structure matérielle au chapitre 2. Le langage lui-même est présenté au chapitre 3. Des notes sur le langage sont données au chapitre 4. Enfin, une implémentation du langage est décrite au chapitre 5.

2. HISTORIQUE

Les systèmes de commutation téléphonique sont les plus anciens et les plus importants de tous les systèmes de télécommunication.

Après l'étape de la technique électromécanique, puis celle de l'électronique (logique câblée), l'étape suivante du progrès de la téléphonie a été marquée par l'apport des ordinateurs qui ont ouvert la voie à la technique du programme enregistré.

Ceci a eu pour avantage de rendre plus souple la gestion et la maintenance des systèmes et d'étendre les applications et les services fournis à l'usager.

Aujourd'hui, avec l'avénement des microprocesseurs et des circuits intégrés de grande complexité, il est devenu possible de réaliser des systèmes décentralisés, plus fiables, plus modulaires, plus puissants, moins encombrants et moins coûteux.

3. PROGRAMMATION DES SYSTÈMES DE COMMUTATION À PROGRAMME ENREGISTRÉ

La grande souplesse de la commande par programme enregistré a permis de concevoir des fonctions logiques beaucoup plus complexes ; cependant, cette souplesse soulève des problèmes très importants relatifs à la programmation:

- les programmes de téléphonie sont extrêmement volumineux (on mesure leur taille en centaines de milliers d'instructions en langage de base),
- de plus, ils doivent répondre aux exigences de sûreté et d'efficacité des systèmes téléphoniques.

La nécessité d'obtenir un code efficace et d'optimiser l'implantation des données, a d'abord entraîné l'usage en téléphonie de langages d'assemblage, puis de langages plus évolués comportant néanmoins un grand nombre de concepts très proches de la machine. On peut citer le: langage ESPL-1 et les langages anglais conçus à l'Université d'Essex: TPL1, TPL1.5, TPL2.

Dans TPL1 et ESPL-1, par exemple, le programmeur a la possibilité de faire chevaucher des données en spécifiant directement leur implantation en mémoire. Il peut aussi indiquer à la fois l'adresse absolue ou relative des objets qu'il manipule. Ces deux possibilités lui permettent de gagner de l'espace mémoire, mais rendent malheureusement les programmes confus et peu fiables. Aucun contrôle n'est possible pour veiller à ce qu'aucune mauvaise utilisation ne soit faite d'un même emplacement.

Ces langages orientés-machine sont devenus d'une grande importance pour les systèmes de commutation à programme enregistré, en raison des deux avantages suivants:

. ils permettent d'utiliser les facilités fournies par le matériel des processeurs de ces systèmes, qui est en général très spécifique ;

. ils présentent des avantages sur les langages d'assemblage du point de vue de la rapidité de production, de la lisibilité et de la facilité de modification et de maintenance.

Cependant, ils presentent en général des faiblesses du point de vue de la sécurité et de l'adaptabilité aux systèmes téléphoniques.

Le logiciel écrit à l'aide de ces langages est difficilement transportable d'un modèle de calculateur à un autre.

Au début du développement de ces langages, il y a eu de nombreuses discussions à propos de leur niveau, car l'un des principaux objectifs des langages évolués était l'indépendance de la machine et, en fait, les concepteurs de l'un d'eux l'ont appelé "langage d'assemblage de haut niveau" [Wichmann, 1973].

Afin de pallier les lacunes rencontrées dans les langages orientés-machine que nous venons de citer, nous avons tenté de définir un langage de haut niveau qui soit adapté à l'application téléphonique d'une part, et transportable d'un modèle de calculateur à un autre d'autre part.

4. OBJECTIFS

Le principal objectif poursuivi est la satisfaction des exigences de sûreté et d'efficacité qui sont strictes en téléphonie.

La sûreté de fonctionnement demandée dans les systèmes téléphoniques est extrêmement grande (2 heures de panne totale tolérées pour 40 ans de fonctionnement !).

Le langage de programmation utilisé joue un rôle important pour obtenir des programmes présentant un haut degré de sûreté. Pour atteindre cet objectif, il doit [Rannou, 1976] :

. être simple,
. être lisible,
. être adapté aux principes de la programmation structurée,
. permettre le plus possible de contrôles à la compilation,
. posséder des concepts qui permettent de traiter les cas d'erreur.

La simplicité et la lisibilité sont atteintes en introduisant un petit nombre de concepts. Pour cela, on a adopté uniquement les instructions jugées nécessaires et suffisantes pour exprimer l'algorithme de la commande. Ceci réduit évidemment le champ d'application du langage, mais le rend plus facile à utiliser et à implémenter, et plus apte à l'exploitation.

Les principes de la programmation structurée sont strictement appliqués. Le langage comprend des schémas de contrôle tels que REPETER, TANTQUE, SI-ALORS-SINON, ... Les instructions de branchement ALLEA et EXIT sont complètement éliminées, ce qui a grandement favorisé la lisibilité du programme téléphonique.

Le contrôle à la compilation a été augmenté par l'adoption de types associés aux objets manipulés dans le langage. Ceci permet de vérifier le type des opérandes, au niveau des opérations effectuées.

Enfin, par l'introduction du concept d'EXCEPTION, nous avons pu traiter *les cas d'erreurs* dans le même programme et dans le même langage.

Le second objectif du langage est l'efficacité. En téléphonie, l'efficacité des programmes est souvent estimée être le principal objectif [Rannou, 1976]. Ceci a conduit à la définition de langages comportant des concepts très proches de la machine. Ces concepts ont bien sûr favorisé l'efficacité, mais ils ont été nuisibles à la lisibilité, à la transportabilité et à la possibilité de contrôle à la compilation.

Afin d'éviter ces inconvénients, il est important qu'un langage soit indépendant du calculateur utilisé. Pour cela il faut qu'aucune définition ne soit trop inspirée de la machine cible. Ceci nous a amené à définir un langage de type évolué, adapté à l'application, mais le plus possible transportable d'un calculateur à un autre.

Ce choix peut se justifier, car il est admis aujourd'hui que si un langage et son compilateur sont bien conçus, on peut atteindre un haut degré d'optimisation, donc d'efficacité.

On décrit brièvement dans ce qui suit la démarche suivie pour arriver à la définition du langage proposé.

5. DÉMARCHE SUIVIE

Nous avons commencé par étudier l'algorithme des organes de commande pour acquérir une connaissance de l'environnement fonctionnel. Le but recherché était de définir un langage le plus simple possible, qui puisse décrire cet algorithme.

L'algorithme étudié exprime les fonctions des organes de commande qui sont limitées au traitement des appels, à la taxation, à la surveillance et à l'observation.

La commande peut traiter plusieurs milliers d'appels au même moment, à différents stades d'avancement. Un appel commence lorsqu'un abonné décroche son combiné. Cet événement est détecté par les organes périphériques qui l'envoient à la commande sous la forme d'un message. La commande alloue alors un enregistreur pour le traitement de la communication et lance un ordre d'envoi de tonalité d'invitation à numéroter, qui cesse dès que le premier chiffre arrive. Chaque chiffre est analysé pour déterminer quel est le numéro d'annuaire de l'abonné demandé. Ce numéro est ensuite converti en un numéro d'équipement. L'état de l'équipement est testé pour savoir s'il est libre et c'est à ce moment seulement que la connexion est établie entre les deux abonnés.

On voit donc que la logique du traitement suit essentiellement le schéma suivant:

. attente d'un message,
. traitement de ce message qui comprendra éventuellement l'envoi d'un message à un organe périphérique,
. attente d'un nouveau message.

L'étude de l'algorithme de la commande nous a montré que:

. le traitement d'un appel est constitué de phases généralement assez courtes et séparées par des temps d'attente qui sont parfois à l'échelle humaine (numérotation des chiffres, réaction de l'abonné appelé à la sonnerie d'appel, conversation entre les deux abonnés). Ces phases sont appelées transitions.

. Le traitement d'un message, qui représente la transition proprement dite, est constitué de deux types d'actions:

- des actions très simples qui effectuent en général des tests et des mises à jour des données de l'enregistreur,
- des actions qui expriment l'émission et la réception des messages qui s'échangent entre l'organe de la commande et les différents organes périphériques qu'il pilote.

Ces remarques nous ont conduit:

. pour décrire le premier type d'action, à choisir des instructions qui sont déjà définies dans la plupart des langages évolués courants, telles que SI-ALORS-SINON, TANTQUE, REPETER ...

. et à définir de nouvelles instructions spécifiques pour décrire non seulement le second type d'actions, mais aussi l'attente des messages. L'introduction de ces instructions est due à la structure décentralisée de l'autocommutateur. Ce sont des instructions d'entrée-sortie qui expriment la communication et la synchronisation entre les processus de la commande et les processus des organes périphériques. Tous ces processus opèrent en parallèle et communiquent entre eux selon le "modèle du producteur et du consommateur".

En conclusion, le but de cette thèse est de concevoir un langage de haut niveau qui soit d'une part le plus simple possible, et d'autre part adapté à la programmation des organes de commande de l'autocommutateur téléphonique.

CHAPITRE 2

STRUCTURE DES ORGANES DE COMMANDE

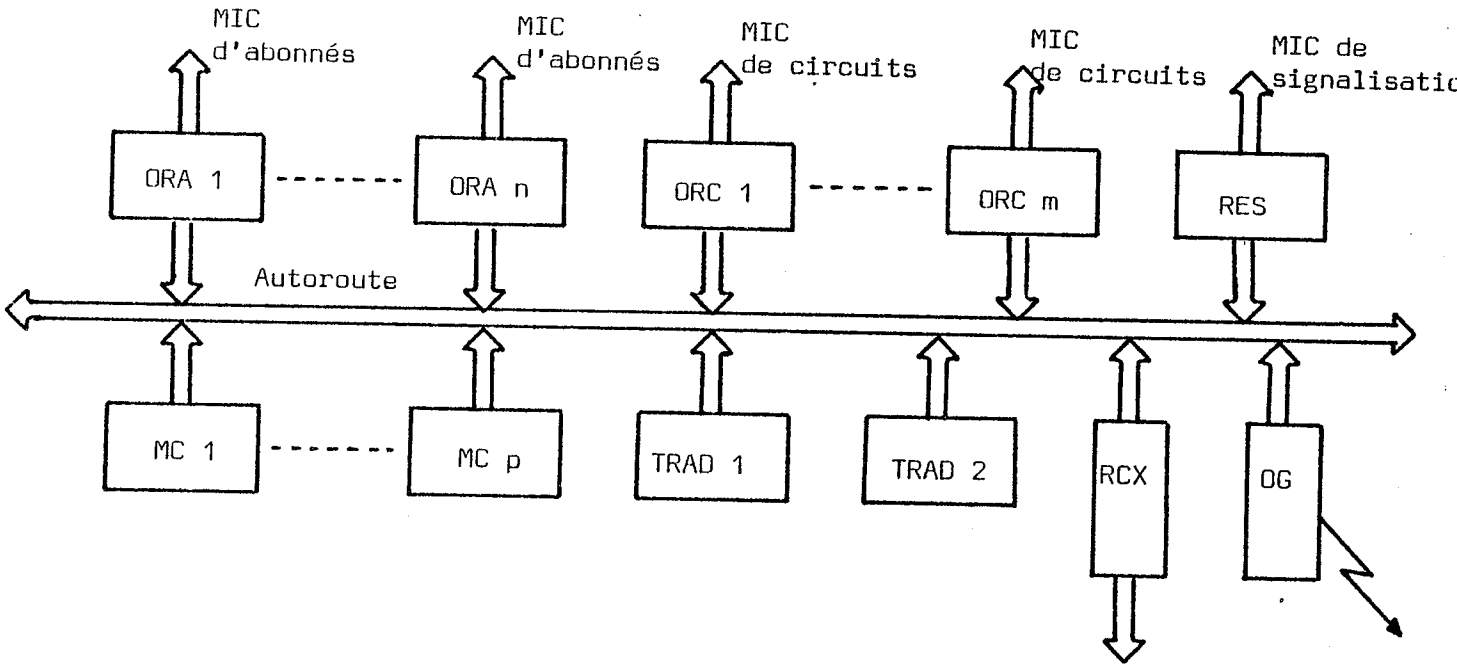
1. INTRODUCTION

Les modules de commande (MC) (ou organes) constituent le coeur de l'auto-commutateur téléphonique, dont la structure est représentée dans la figure 1 [ISS, 1979]. Les autres organes de l'auto-commutateur sont les organes périphériques qui comprennent:

- les organes de raccordement d'abonnés (ORA) et de circuits (ORC) qui assurent les fonctions de numérisation, d'exploration de concentration, d'affectation des voies numériques, de transformation des signalisations par états en signalisation par événements, de gestion de la procédure de demande de contexte et de régulation de la charge par rejet sélectif des appels ;
- les récepteurs/émetteurs de signalisation (RES) qui analysent et génèrent les signalisations multi-fréquences et génèrent les tonalités ;
- le réseau de connexion (RCX) qui assure les connexions et déconnexions des voies de parole des multiplex entrants sur les multiplex sortants ; il assure également la fonction de mémorisation des communications établies ;
- l'organe de gestion (OG) qui assure la liaison avec l'exploitation et gère les comptes auxiliaires d'abonnés ;
- les traducteurs (TRAD) qui gèrent les informations d'acheminement concernant les abonnés et les circuits.

Les organes de l'auto-commutateur sont reliés entre eux par un système de liaisons (autoroute) constitué d'un ensemble de lignes série banalisées, travaillant en partage de trafic.

L'échange d'informations entre les différents organes de la structure se fait par des messages ayant une grande densité d'informations.



ORA = organe de raccordement d'abonnés

ORC = organe de raccordement de circuits

RES = récepteurs/émetteurs de signalisation

MC = module de commande

TRAD= traducteur

RCX = réseau de connexion

OG = organe de gestion

MIC = multiplex d'échantillons de parole transmis en modulation par impulsion et codage

Figure 1 - Structure générale de l'auto-commutateur

2. MODULES DE COMMANDE [ISS, 1979]

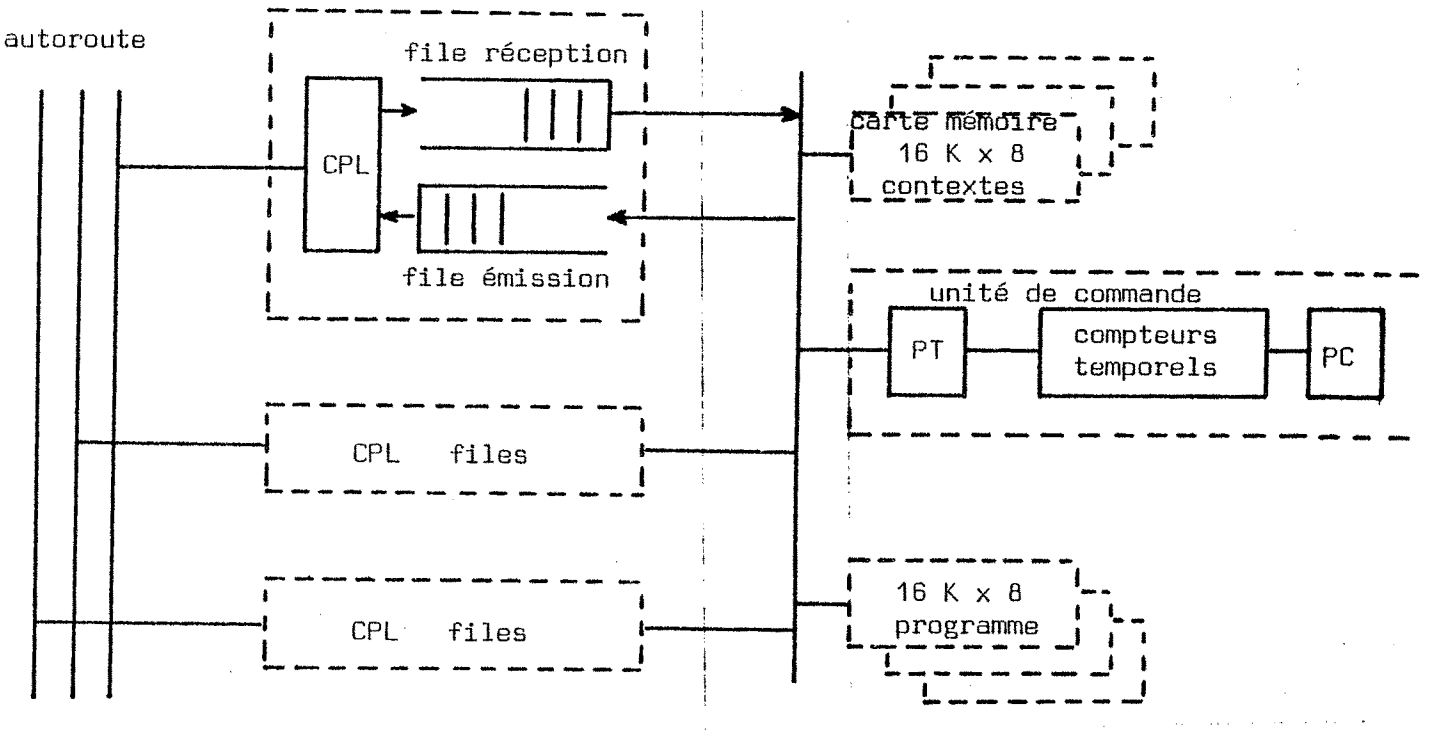
Chaque module (figure 2) comprend un processeur de traitement (auquel est associé un processeur de gestion des temporisations), une mémoire et un coupleur pour chacune des liaisons composant le système de liaisons. Chaque module gère ses propres contextes et il est réalisé à partir de microprocesseurs standards.

Les modules de commande sont les organes de décision de l'auto-commutateur. Toutes les phases successives d'une communication sont supervisées par le même module de commande. Pour cela, le module assure trois fonctions:

. **traitement des appels:**
- création et destruction des processus (communication) et gestion correspondante des contextes,
- réception des messages, association aux contextes et exploitation des paramètres,
- traitement du message par exécution d'une séquence (ou transition) du programme de traitement des communications (programme téléphonique),
- mise à jour des contextes en fonction de la progression de ce programme,
- enchaînement des tâches périphériques par émission d'ordres aux organes spécialisés ;

. **taxation ;**

. **observations:**
- mesure de la répartition des flux de trafic internes au système,
- mesure du comportement des différentes ressources.



CPL = processeur coupleur: réalise le mécanisme d'échange

PT = processeur de traitement: réalise l'application

PC = processeur de comptage: réalise les temporisations

Figure 2 - Structure d'un module de commande

3. ORGANISATION D'UN PROCESSEUR DE TRAITEMENT

3.1. Données

Les données du processeur de traitement sont rangées dans des contextes. Un contexte est constitué d'un ou plusieurs blocs mémoire de taille fixe. Il est susceptible de contenir toutes les informations relatives à une communication qui sont utiles au traitement et aux dialogues. Parmi ces informations on peut citer:

- l'identification du demandeur,
- l'identification du demandé,
- le taux de taxation,
- les discriminations (divers droits et catégories) du demandeur.

3.2. Programme téléphonique

Un module de commande peut traiter plusieurs communications en même temps ; chaque communication est représentée par un processus. Le programme téléphonique décrit le déroulement de chaque processus ; il est écrit de manière séquentielle, comme s'il n'y avait qu'une seule communication en cours de traitement. Il est réentrant, car on peut le réactiver à n'importe quel moment. Il y aura dans le système autant de processus actifs que de communications pouvant être traitées. Le traitement d'un processus est constitué, comme nous l'avons vu au chapitre 1, de transitions.

Une transition est le traitement d'un message concernant ce processus. Les transitions sont généralement assez courtes et séparées par des temps d'attente qui sont parfois à l'échelle humaine. Ceci permet de traiter pendant ces temps d'attente plusieurs transitions relatives à d'autres processus. La succession rapide de ces transitions pour des processus différents assure donc le multiplexage asynchrone des traitements de communication.

Le programme téléphonique est écrit dans le langage présenté au chapitre 3.

4. ORGANISATION DU LOGICIEL D'UN MODULE DE COMMANDE [RAP, 1979]

Le logiciel d'un module de commande est organisé en quatre couches réparties comme suit:

- la couche de plus haut niveau est la couche d'application,
- la couche de service dessert directement la couche d'application,
- la couche intermédiaire réalise l'interface entre les deux couches précédentes et la couche des coupleurs,
- la couche élémentaire est formée du logiciel des coupleurs et du logiciel du processeur de comptage.

4.1. Couche d'application

Elle est constituée du programme téléphonique traitant les établissements et les ruptures de communications.

Ce programme est celui du processus correspondant à la communication. Il est écrit dans le langage proposé (chapitre 3) et fait souvent appel à des routines système utilisées dans la couche de service.

4.2. Couche de service

Elle regroupe les routines de:

- gestion des processus (création, réveil, mise en sommeil),
- gestion mémoire,
- exploration des files d'attente des gardes et des messages,
- certaines routines de sûreté de fonctionnement,
- certaines routines d'exploitation.

4.3. Couche intermédiaire

Elle effectue principalement les entrées-sorties logiques entre les diverses files d'attente (échéance, coupleur) et le tampon d'entrée-sortie.

4.4. Couche élémentaire

Elle est constituée par le logiciel des coupleurs. C'est un ensemble de routines réalisant les échanges entre les différents organes.

5. GESTION DES ÉCHANGES ENTRE LES DIFFÉRENTES COUCHES DU LOGICIEL [RAP 1979]

Le traitement d'une communication consiste à traiter les messages qui la concernent. L'exploitation d'un message nécessite la coopération entre les différentes couches du logiciel décrites plus haut.

Dans ce qui suit, nous décrivons cette coopération lors de la réception, de l'émission et de l'attente d'un message (figure 3).

5.1. Réception d'un message

La réception d'un message émis par un organe périphérique vers la commande se fait par l'un des coupleurs de la couche élémentaire. Le message en question sera copié dans la file de réception de ce coupleur. Sous la commande du moniteur (distributeur de tâches), toutes les files de réception des coupleurs sont auscultées constamment. Dans le cas où l'une d'elles n'est pas vide, cela entraîne l'extraction d'un message et sa recopie dans un tampon d'entrée-sortie. Une analyse est ensuite faite par la couche de service pour classer le message reçu qui peut appartenir à l'une des quatre catégories de messages suivantes:

- messages de panne,
- messages d'exploitation,
- messages de demande de création de processus,
- messages téléphoniques.

Dans le premier cas, il sera dirigé vers le module de traitement des messages de panne.

Dans le deuxième cas, il sera dirigé vers le module de traitement des messages d'exploitation.

Dans le troisième cas, il y aura création d'un processus, si la charge du module de commande le permet.

Enfin, s'il appartient à la quatrième catégorie, il sera associé au processus correspondant, puis traité par la couche de plus haut niveau formée par le programme téléphonique.

Comme on l'a vu au chapitre 1, ce traitement consiste à tester et à mettre à jour les données du contexte du processus concerné et éventuellement à émettre des messages vers les organes périphériques. Le traitement se termine par l'attente d'un ou de plusieurs messages nécessaires pour le bon déroulement du processus.

5.2. Emission d'un message

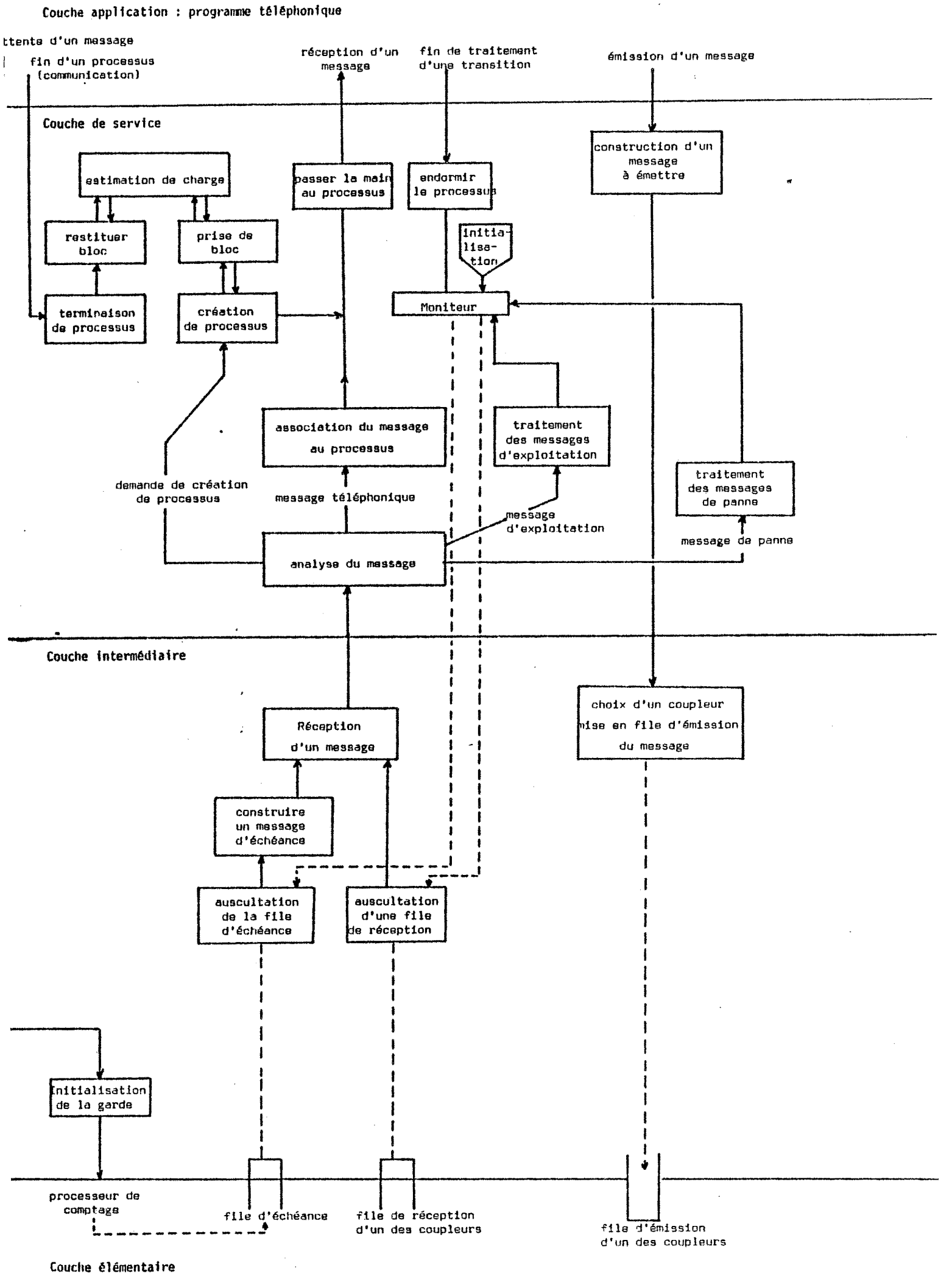
L'émission d'un message fait appel à un module de la couche de service qui se charge de la construction du message à émettre. Ce module fait ensuite appel à un module de la couche intermédiaire qui se charge du choix du coupleur et du transfert du message construit du tampon d'entrée-sortie vers la file d'émission du coupleur choisi. Ce dernier transmet ensuite le message sur l'autoroute.

5.3. Attente d'un message

Les messages envoyés par la commande aux organes périphériques vont entraîner l'arrivée de messages de réponses.

Pour assurer une bonne sûreté de fonctionnement, la politique choisie est de n'attendre un message que pendant un certain temps, c'est-à-dire de le "garder". Tous les messages sont systématiquement gardés. Pour gérer les gardes relatives à tous les processus actifs dans le système, un compteur est associé à chacun d'eux.

L'attente d'un message par un processus, au niveau du programme téléphonique, entraîne l'appel d'un module de la couche intermédiaire qui se charge d'initialiser le compteur correspondant au processus, à la durée voulue de la garde. Le processeur de comptage fait ensuite décrémenter ce compteur. Le passage à zéro de ce dernier entraîne le dépôt par le processeur de comptage d'un message dans la file des échéances des processus. Cette file, comme les autres files des coupleurs, sera constamment auscultée par le moniteur.


Couche application : programme téléphonique
ttente d'un message
fin d'un processus (communication)
réception d'un message
fin de traitement d'une transition
émission d'un message
Couche de service
estimation de charge
passer la main au processus
endormir le processus
construction d'un message à émettre
restituer bloc
prise de bloc
initialisation
terminaison de processus
création de processus
Moniteur
association du message au processus
traitement des messages d'exploitation
traitement des messages de panne
demande de création de processus
message téléphonique
message d'exploitation
message de panne
analyse du message
Couche intermédiaire
Réception d'un message
choix d'un coupleur mise en file d'émission du message
construire un message d'échéance
auscultation de la file d'échéance
auscultation d'une file de réception
initialisation de la garde
processeur de comptage
file d'échéance
file de réception d'un des coupleurs
file d'émission d'un des coupleurs
Couche élémentaire

CHAPITRE 3

LANGAGE DE PROGRAMMATION

1. INTRODUCTION

Le langage est destiné à programmer les fonctions de traitement des appels, de taxation et d'observation de l'auto-commutateur téléphonique.

Vues la taille et la complexité des algorithmes à programmer, le langage défini vise la simplicité et la lisibilité. Il met en oeuvre les principes de la programmation structurée et est adapté à l'application grâce à l'introduction d'instructions spécifiques.

Le programme téléphonique de la commande a la structure d'un bloc ayant un seul point d'entrée et un seul point de sortie. Le bloc est composé d'un entête et d'un corps. Ce dernier est constitué de deux parties essentielles:

- les définitions et les déclarations, qui décrivent les données manipulées par les instructions,
- les instructions, qui décrivent les actions qui doivent être accomplies.

La déclaration d'une variable lui associe un type caractérisant sa structure et l'ensemble des valeurs qu'elle peut prendre. Ceci a pour avantage d'assurer la sécurité des données téléphoniques et de contrôler l'usage que le programmeur fait d'elles.

Le langage dispose des types suivants:

- types simples:
  type scalaire,
  type booléen,
  type entier,
  type intervalle ;

. types structurés:
  les tableaux,
  les enregistrements,
  les ensembles ;

. type pointeur.

Les instructions définies par le langage sont de deux sortes:

. les instructions courantes qui constituent le noyau de tous les langages évolués courants, telles que:
  - SI, ALORS, SINON,
  - REPETER, JUSQUA,
  - l'instruction d'affectation,
  - l'instruction de choix ;

. les instructions spécialisées définies pour être adaptées à l'application, telles que:
  - l'instruction d'attente de message AVANT,
  - l'instruction de choix SUIVANT,
  - l'instruction QUAND.

2. NOTATIONS ET VOCABULAIRE

La syntaxe du langage est décrite en utilisant la notation BNF (Backus-Naur-Form). Sa sémantique est décrite informellement en français.

Les entités syntaxiques sont indiquées par un ou plusieurs mots français, enfermés entre deux crochets: < >. Elles peuvent aussi être groupées en utilisant les deux accolades: { }. Ceci indique leur répétition zéro ou plusieurs fois.

Le langage est construit à partir d'un ensemble de symboles de base qui constituent son alphabet. Ces symboles peuvent être classés en:
lettres: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
chiffres: 0 1 2 3 4 5 6 7 8 9
symboles spéciaux: + - * / ( ) = , . : < > ; [ ] ∨ ∧ ¬

Les mots réservés du langage sont:
A ALORS APPEL ATTENDRE ATTENDREDOUBLE ATTENDRETRIPLE AUTRE AVANT AVEC BIN CONST DCB DE DEBUT EB ENREGISTREMENT ENSEMBLE ENVOI EXCEPTION FAIRE FAUX FIN FINQUAND FINSELON FINSUIVANT FSI JUSQUA LECT MIN MOT NIL POUR PROCEDURE PROVOQUER PTR QUAND QUANDNON RECEVOIR REPETER SEC SECT SELON SI SINON SUIVANT TABLEAU TANTQUE TERMINER-PROCESSUS TYPE VAL VAR VRAI.

Les mots réservés sont reconnus comme des mots particuliers ayant un sens fixe. Ils ne peuvent pas être utilisés comme des identificateurs. Dans la description du langage, ils vont être soulignés pour les différencier des identificateurs.

Les commentaires peuvent être insérés entre deux définitions, entre deux déclarations, ou entre deux instructions du programme, tout en respectant la syntaxe suivante:
/*<tout caractère différent de * et de />*/

Les identificateurs servent à dénoter des constantes, des types, des variables, des procédures et des exceptions. Un identificateur est formé selon la syntaxe suivante:

<identificateur> ::= <lettre> { <lettre>/<chiffre>/-}

Un identificateur ne peut pas se terminer par le caractère -.

Les nombres manipulés dans le programme téléphonique sont des nombres entiers non signés. Ils suivent la syntaxe suivante:

<nombre> ::= <entier non signé>

<entier non signé> ::= <chiffre>{<chiffre>}.

3. STRUCTURE DU PROGRAMME

Syntaxe:

<programme> ::= <bloc>

<bloc> ::= <entête de bloc><corps de bloc><fin de bloc>

<entête de bloc> ::= DEBUT /<identificateur de bloc> : DEBUT

<corps de bloc> ::= <introduction><partie instruction>

<introduction> ::= <partie définition des constantes>
<partie définition des types>
<partie déclaration des variables>
<partie déclaration des exceptions>
<partie déclaration des procédures>

<fin de bloc> ::= FIN / FIN <identificateur de bloc>

<identificateur de bloc> ::= <identificateur>

Sémantique:

Les concepts de structuration du programme sont le bloc, la procédure et l'exception. De plus, dans la partie instruction, plusieurs instructions peuvent être regroupées à la manière d'un bloc pour former une instruction composée.

Les données téléphoniques sur lesquelles vont travailler les procédures et les exceptions sont implémentées globalement. Ceci a pour avantage de diminuer le temps d'accès à ces données, ainsi que la place nécessaire à la gestion de la pile d'exécution.

Les parties formant l'introduction doivent apparaître dans l'ordre indiqué. En outre, chacune d'elles peut être absente.

L'identificateur qui apparaît éventuellement dans la fin du bloc doit être identique à celui qui apparaît dans l'entête du bloc.

4. DONNÉES

Les données du programme téléphonique sont rangées dans des enregistreurs de taille fixe, susceptibles de contenir toutes les informations relatives à une communication en phase d'établissement, de conversation ou de rupture. Les enregistreurs sont des variables de type enregistrement, générées dynamiquement au niveau du système d'exploitation. Elles ne peuvent être référencées que par des pointeurs dont les valeurs sont fournies par la routine de gestion de ces enregistreurs. Un pointeur n'est susceptible de contenir qu'une adresse d'enregistrement. Il reçoit le type de l'enregistrement vers lequel il pointe.

Les données sont les constantes et les variables. A chaque variable est associé un identificateur de variable et un type. Les types précisent la structure des variables, ainsi que l'ensemble des valeurs qu'elles peuvent prendre.

Dans ce qui suit, on décrit les parties définitions et déclarations des données.

4.1. Définition des constantes

Syntaxe:

<partie définition des constantes> ::= <vide>/CONST <définition de constante>
{;<définition de constante>} ;
<définition de constante> ::= <identificateur> = <constante non signée>
<constante non signée> ::= <nombre>/<identificateur de constante>/ NIL
<identificateur de constante> ::= <identificateur>

Sémantique:

La définition d'une constante introduit un identificateur qui est synonyme de cette constante et qui pourra être utilisé à sa place dans la suite du programme. L'utilisation des identificateurs de constante rend le programme plus lisible et apporte une aide à la documentation.

Exemple:

```
CONST Nb-chif-local := 6 ;
```

4.2. Définition des types

Syntaxe:

<partie définition des types> ::= <vide>/ TYPE <définition de type>
{; <définition de type>} ;

<définition de type> ::= <identificateur> = <type>

<type> ::= <type simple>/<type structuré>/<type pointeur>

Sémantique:

Un type est un ensemble de propriétés telles que: taille, opérations permises, etc ..

La définition d'un type lui associe un identificateur qui n'est pas une variable, mais qui peut être utilisé ensuite pour représenter ce type. C'est un mécanisme qui permet d'introduire des nouveaux symtoles de types en plus des types standards du langage. Il faut noter que la définition récursive des types n'est pas permise.

4.2.1. Type simple

Syntaxe:

<type simple> ::= <type scalaire><liste d'attributs>/<type intervalle>/
<identificateur de type>

<liste d'attributs> ::= <attribut de taille><attribut de codage>
<attribut d'accès>

<attribut de taille> ::= <vide>/ EB (<nombre>)/MOT (<nombre>)/SECT (<nombre>)

<attribut de codage> ::= <vide>/ BIN / DCB

<attribut d'accès> ::= <vide>/ LECT

Sémantique:

L'attribut de taille est utilisé pour optimiser l'implantation mémoire. Ceci est dû au fait que la plupart des données téléphoniques occupent 1, 2 ou 5 éléments binaires, ce qui rend très coûteuse leur implantation sur des mots. Ces attributs sont déjà définis dans des langages téléphoniques tels que PAPE et LPE10 [F.ALIZON, 1975].

Les attributs MOT(n), SECT(n), EB(n) indiquent respectivement que la donnée a une taille de:

- n mots,
- n secteurs (2 secteurs = 1 demi-mot),
- n éléments binaires (1 élément binaire = 1 bit).

L'attribut de codage DCB indique que la donnée est codée en "décimal codé binaire" avec quatre éléments binaires par chiffre.

L'attribut de codage BIN indique que la variable est codée en "binaire pure".

L'attribut d'accès LECT indique que la donnée doit être accédée en lecture seule, ce qui permet de contrôler l'accès aux données téléphoniques. L'absence de cet attribut indique que l'on peut accéder à la donnée en lecture et en écriture.

4.2.1.1. Type scalaire

<type scalaire> ::= (<identificateur> {,<identificateur>})

Sémantique:

Un type scalaire est défini par un ensemble ordonné de valeurs dénotées par des identificateurs de constantes, qui sont ainsi déclarés de manière implicite. Sur les données de type scalaire s'appliquent deux fonctions standards:
SUCC donnant le successeur d'une valeur de type,
PRED donnant le prédécesseur d'une valeur de type.

Pour une donnée de type scalaire, l'attribut de taille peut être vide ; la taille est déterminée d'après le nombre de valeurs qu'elle peut prendre.

Exemple:

CODE-DR = (ab, CIRCUIT-Y, CIRCUIT-MF, GSM) ;

Il existe deux types scalaires standards:
. le type booléen,
. le type entier.

4.2.1.2. Type booléen

Le type BOOLEEN est défini implicitement par TYPE booléen = (FAUX, VRAI)
Aux opérandes de ce type, on peut appliquer les opérateurs ET (∧), OU (∨) et NON (¬) qui produisent une valeur de type booléen.
Comme pour le type scalaire, l'attribut de taille peut être absent.

Exemple:

ABONNE-CLAVIER = BOOLEEN

4.2.1.3. Type_entier

Une valeur de ce type appartient à l'ensemble des entiers compris entre 0 et $2^{(\text{longueur du mot})}-1$.

Les opérateurs arithmétiques suivants, appliqués à des valeurs de type entier, donnent un résultat de type entier:

* multiplication.

/ division entière

+ addition

- soustraction

Les opérateurs de relation, appliqués à des opérandes de type entier, produisent un résultat de type booléen.

Exemple:

OR-OR = ENTIER

4.2.1.4. Type_intervalle

<type intervalle> ::= <constante> .. <constante>

<constante> ::= <constante non signée>

Sémantique:

C'est un type scalaire réduit à un sous-ensemble d'entiers consécutifs entre une borne inférieure et une borne supérieure. La taille d'une donnée de ce type est déterminée d'après le nombre de valeurs qu'elle peut prendre.

Exemple:

I = 1 .. 256

4.2.2. Type structuré

Syntaxe:

<type structuré> ::= <type enregistrement> / <type tableau> / <type ensemble>

Sémantique:

Un objet de type structuré est composé de plusieurs objets d'autres types.

4.2.2.1. Type enregistrement

<type enregistrement> ::= ENREGISTREMENT <liste de champs> FIN
<liste de champ> ::= <section d'enregistrement>{;<section d'enregistrement>}
<section d'enregistrement> ::= <identificateur de champ>
{,<identificateur de champ>} : <type>
<identificateur de champ> ::= <identificateur>

Sémantique:

Un enregistrement est un objet composé de champs qui sont des objets qui peuvent être de types différents les uns des autres et qui sont assemblés individuellement par une opération de sélection utilisant l'identificateur de champ qui leur est associé.

Exemple:

```
TYPE ENREG = ENREGISTREMENT
             CODE-FONCTION : (NOUVEL-APPEL, RACCROCHAGE, ..) ;
             OR-DR : ENTIER EB(9) BIN ;
             .....   .....
             .....   .....
             FIN
```

La sélection d'un champ d'un enregistrement s'obtient en faisant précéder l'identificateur de ce champ du nom de l'enregistrement suivi d'un point.

Exemple:
La sélection du champ CODE-FONCTION de l'exemple précédent prend la forme suivante:
ENREG.CODE-FONCTION

Il faut noter qu'il n'est pas possible d'affecter un objet de type enregistrement, mais qu'il est possible d'affecter ses champs.

4.2.2.2. Type_tableau

<type tableau> ::= TABLEAU [<type index>] DE <type de composants>
/TABLEAU [<type index>,<type index>]
DE <type de composants>
<type index> ::= <type intervalle>
<type de composants> ::= <type simple>

Sémantique:

Un tableau est constitué d'un nombre fixe de composants de même type simple. La définition d'un tableau précise le domaine de valeurs de l'index et le type de ses composants. Un tableau peut avoir au maximum deux dimensions.

Exemple:
TYPE ZONE-CHIFFRES = TABLEAU [1..16] DE ENTIER EB(4) DCB

L'indexation d'une variable v de type ZONE-CHIFFRES prend la forme v[i] où i est un entier compris entre 1 et 16.

Il faut noter qu'il n'est pas possible d'affecter un objet de type tableau, mais qu'il est possible d'affecter ses éléments.

4.2.2.3. Type ensemble

<type ensemble> ::= ENSEMBLE DE <type de base>
<type de base> ::= <type simple>

Sémantique:

Une valeur de type ensemble est un ensemble sans répétition d'objets d'un même type simple.
Le type ensemble permet de représenter simplement l'appartenance de plusieurs éléments à un ensemble de valeurs.
La construction d'objets de type ensemble se fait de la façon suivante:
<ensemble> ::= [<élément> {,<élément>}] / [ ]
<élément> ::= <expression> / <expression> .. <expression>
[ ] représente l'ensemble vide.

Exemple:

```
TYPE CODE-FONCTION = (NOUVEL-APPEL, RACCROCHAGE, BR, ECHEANCE, ..) ;
     MSG-A-RECEVOIR = ENSEMBLE DE CODE-FONCTION
```

Un objet de type MSG-A-RECEVOIR pourra être construit par:
[NOUVEL-APPEL, ECHEANCE] ;

4.2.3. Type pointeur

<type pointeur> ::= PTR <identificateur de type> <attribut d'accès>

Sémantique:

Les variables déclarées explicitement sont des variables statiques. Par contre, les variables créées au moment de l'exécution sont des variables dynamiques. De telles variables sont adressées par des pointeurs.

Chaque pointeur est associé à un type qui caractérise la variable vers laquelle il pointe. Un pointeur ne peut pointer que vers des variables de type enregistrement, car seules de telles variables sont créées dynamiquement. La valeur NIL indique que le pointeur ne pointe vers aucun enregistrement.

Dans le cas du programme téléphonique, les pointeurs vont servir à pointer vers les enregistreurs qui possèdent une structure d'enregistrement.

Pour des questions de sûreté, la génération de ces enregistreurs est retirée de l'initiative du programmeur. Elle se fait au niveau du système d'exploitation.

Exemple:

```
TYPE ENREG = ENREGISTREMENT
             CODE-FONCTION : (NOUVEL-APPEL, RACCROCHAGE, ECHEANCE, ...) ;
             OR-DR : ENTIER EB(9) BIN ;
             FIN
     P = PTR ENREG LECT ;
VAR  E : P ;
```

Pour sélectionner la variable dynamique de type ENREG pointée par E, on utilise la notation VAL E. Dans l'exemple précédent, pour sélectionner le champ OR-DR de la variable ENREG, on utilisera la variable VAL E.OR-DR.

4.3. Déclaration des variables

Syntaxe:

<partie déclaration des variables> ::= <vide> / VAR <déclaration de variable>
{; <déclaration de variable>} ;

<déclaration de variable> ::= <identificateur>{,<identificateur>} : <type>

Sémantique:

Les variables déclarées dans cette partie sont statiques, car une zone de mémoire leur est allouée pendant toute la durée d'exécution du programme. Le type de chaque variable est unique et donné dans la déclaration de variable.

Exemples:

```
TYPE MSG-A-RECEVOIR : (NOUVEL-APPEL, RACCROCHAGE, ECHEANCE, ..)
VAR MSG-A-ENVOYER : MSG-A-RECEVOIR
```

La variable MSG - A - ENVOYER est du type MSG - A - RECEVOIR.

4.4. Déclaration des exceptions

Syntaxe:

<partie déclaration des exceptions> ::= <entête d'exception><corps d'exception
<entête d'exception> ::= EXCEPTION <identificateur> ;
<corps d'exception> ::= <instruction composée>

Sémantique:

La déclaration d'une exception permet d'identifier un ensemble d'instructions et de pouvoir ensuite les faire exécuter par l'instruction d'appel d'exception qui est l'instruction PROVOQUER.

L'exception est une procédure particulière, car son exécution termine l'exécution de tout le programme.

Exemple:

```
EXCEPTION preselec-mal-déroulé ;
DEBUT
APPEL identifier-erreur ;
APPEL LIB-DCNX-RNC
FIN
```

4.5. Déclaration des procédures

Syntaxe:

<partie déclaration des procédures> ::= {<déclaration de procédure> ;}
<déclaration de procédure> ::= <entête de procédure> <corps de procédure>
<entête de procédure> ::= PROCEDURE <identificateur> ; /
PROCEDURE <identificateur>
(<section de paramètres formels>
{;<section de paramètres formels>}) ;
<section de paramètres formels> ::= <identificateur>{,<identificateur>} :
<identificateur de type>
<identificateur de type> ::= <identificateur>
<corps de procédure> ::= <instruction composée>

Sémantique:

Cette partie définit l'identificateur de la procédure et les paramètres formels éventuels. Le passage des paramètres se fait par référence.

Exemple:

```
PROCEDURE libération-ETA (msg-a-envoyer : CODE-FONCTION) ;
DEBUT
ENVOI (ETA, msg-a-envoyer)
RECEVOIR (ETA, [accuse-restitu]) ;
FIN
```

Exemple d'une partie de description des types de données téléphoniques

```
ENREG = ENREGISTREMENT
        CODE-FONCTION : (NOUVEL-APPEL, RACCROCHAGE, ...) LECT ;
        TYPE-COMM     : (COMM-COURANT, TRANSFERT, CCF, ...) ;
        OR-DR         : ENTIER MOT(1) BIN LECT ;
        VTLRDR        : ENTIER EB(5) BIN LECT ;
        EQ-DR         : ENTIER EB(9) BIN LECT ;
        DISCRI-DR     : ENREGISTREMENT
                        TD, ABS : BOOLEEN LECT ;
                        NC, SR : (NATIONAL, INTERNATIONAL, REGIONAL) LECT ;
                        CAB, IDT, LTI,
                        OBS, OPE, CCF,
                        STR, CL, AE, RT,
                        LE, LSN, IAI : BOOLEEN LECT ;
                        FIN
        ....          : ....
        FIN
P = PTR ENREG ;
```

5. INSTRUCTIONS

Syntaxe:

<partie instruction> ::= <instruction> {; <instruction>}
<instruction> ::= <instruction courante> / <instruction spécialisée>

5.1. Les instructions courantes

Syntaxe:

<instruction courante> ::= <instruction simple> / <instruction structurée>
<instruction simple> ::= <instruction d'affectation> /<instruction d'appel de procédure> / <instruction de provocation d'exception> / <instruction vide>
<instruction structurée> ::= <instruction composée> / <instruction conditionnelle> / <instruction répétitive> / <instruction AVEC>

Sémantique:

Les instructions courantes constituent le noyau de tous les langages évolués classiques. Elles sont particulièrement inspirées du langage PASCAL [WIRTH, 1974].

5.1.1. Les instructions simples

5.1.1.1. Instruction d'affectation

<instruction d'affectation> ::= <variable> ::= <expression>
<variable> ::= <identificateur de variable> / <variable composante> /
<variable référencée>
<identificateur de variable> ::= <identificateur>
<variable composante> ::= <indexation> / <sélection>
<indexation> ::= <variable tableau> [<expression>] /<variable tableau>
[<expression>,<expression>]
<variable tableau> ::= <identificateur de variable> /<sélection>
<sélection> ::= <variable enregistrement> . <identificateur de champ>
<variable enregistrement> ::= <identificateur de variable> / <sélection> /
<variable référencée>
<identificateur de champ> ::= <identificateur>
<variable référencée> ::= VAL <variable pointeur>
<variable pointeur> ::= <identificateur de variable> / <sélection>
<expression> ::= <expression simple> / <expression simple>
<opérateur de relation> <expression simple>
<opérateur de relation> ::= = / < / > / < = / > = / =/ /
<expression simple> ::= <terme> / <terme><opérateur d'addition> <terme>
<opérateur d'addition> ::= + / - / v
<terme> ::= <facteur> / <facteur> <opérateur de multiplication> <facteur>
<opérateur de multiplication> ::= * / / / ∧
<facteur> ::= <variable> / <constante non signée> /(<expression>) /¬ <facteur>

Sémantique:

L'affectation donne une nouvelle valeur à la variable. La valeur est obtenue par l'évaluation de l'expression. L'affectation n'est possible que pour les types simples des variables. En plus il faut que la variable et l'expression soient du même type.

Selon les opérateurs et les opérandes, les types des objets manipulés par une expression sont les suivants:

expression	opérande	opérateurs	type opérande	type résultat
facteur	variable constante (expression)	¬	booléen	booléen
terme	facteur	* / ∧	entier entier booléen	entier entier booléen
expression simple	terme	+ - ∨	entier entier booléen	entier entier booléen
expression	expression simple	= =/ < > <= >=	scalaire type scalaire type scalaire	booléen booléen booléen

La sémantique des opérateurs est la suivante:

+ addition

- soustraction

* multiplication

/ division entière

¬ négation

∨ OU

∧ ET

= égalité

=/ inégalité

< plus petit

<= plus petit ou égal

> plus grand

>= plus grand ou égal

5.1.1.2. Instruction d'appel de procédure

<instruction d'appel de procédure> ::= APPEL <identificateur de procédure> /
APPEL <identificateur de procédure>
(<paramètre effectif>{ ,<paramètre effectif>})

<identificateur de procédure> ::= <identificateur>

<paramètre effectif> ::= <variable>

Sémantique:

Cette instruction a pour effet d'exécuter la procédure appelée. Les paramètres effectifs éventuels sont passés par référence. Le paramètre effectif doit être du type simple.

Exemple:

APPEL identifier-erreur ;

5.1.1.3. Instruction de provocation d'exception [GREEN, 1978]

<instruction de provocation d'exception> ::= PROVOQUER <identificateur d'exception>

<identificateur d'exception> ::= <identificateur>

Sémantique:

C'est un appel à une exception. L'exception est une procédure particulière dont l'exécution remplace le reste de l'exécution du programme téléphonique. La définition de cette instruction s'est révélée nécessaire en raison de la structure particulière de l'algorithme de la commande.

La commande fait avancer une communication en cours d'établissement ou de rupture, par transitions. Chaque transition correspond au traitement d'un message d'événement concernant la communication. Le traitement commence par une analyse fonctionnelle du message arrivé et se termine par l'attente d'un ou de plusieurs messages d'événements qui seront utiles dans la suite pour le bon déroulement de la communication. Le code fonction du message arrivé est comparé au(x) code(s) fonction du ou des message(s) attendu(s) dans le but de détecter l'arrivée d'un message inattendu. Celui-ci est considéré comme une exception qui pourrait être:

. le raccrochage du demandeur avant sa connexion avec le demandé,
. l'échéance de la garde,
. n'importe quel autre message.

Il est normal de faire régresser la communication lorsqu'un tel cas se produit, ce qui se traduit par une sortie de tout le reste du programme téléphonique pour cette communication. Pour des questions de sécurité du système, la politique choisie est de traiter aussi cette exception en utilisant le même langage de programmation.

Pour nous, les exceptions sont restreintes aux messages d'événements qui entraînent la régression de la communication. Ceci veut dire que, quand une exception est rencontrée, le processus correspondant à cette communication doit se terminer.

5.1.1.4. Instruction vide

<instruction vide> ::=

Sémantique:
Cette instruction n'a aucun effet sur le déroulement du programme.

Exemple:

```
QUAND accuse-restitue ALORS
                      SINON PROVOQUER etabli-mal-deroule
                      FINDQUAND
```

5.1.2. Les instructions structurées

5.1.2.1. Instruction composée

<instruction composée> ::= <identificateur d'étiquette> :
DEBUT <instruction> {; <instruction> } FIN /
DEBUT <instruction> {; <instruction> } FIN
<identificateur d'étiquette> ::= <identificateur>

Sémantique:
Elle indique que les instructions qui existent entre DEBUT et FIN doivent s'exécuter en séquence.

Exemple:

```
SI N > 0 ALORS DEBUT
              NCG-a-emettre := N ;
              ENVOI (URDE1, emet-N-chif) ;
              ....
              FIN
        FSI
```

5.1.2.2. Instruction conditionnelle

<instruction conditionnelle> ::= <instruction SI> / <instruction SELON>
<instruction SI> ::= SI <expression> ALORS <instruction> FSI /
SI <expression> ALORS <instruction>
SINON <instruction> FSI
<instruction SELON> ::= SELON <expression> FAIRE <liste SELON>
{;<liste SELON>} AUTRE <instruction> FINSELON/
SELON <expression> FAIRE <liste SELON> {; <liste SELON>
FINSELON
<liste SELON> ::= <liste d'étiquettes SELON> : <instruction>
<liste d'étiquettes SELON> ::= <étiquette SELON> {,<étiquette SELON>}
<étiquette SELON> ::= <constante>

Sémantique:

L'instruction SI indique l'exécution de telle ou telle instruction, selon la valeur d'une expression de type booléen.

L'instruction SELON est formée d'une expression (le sélecteur) et d'une liste d'instructions. Chacune d'elles est étiquetée par une constante du même type que le sélecteur. Le sélecteur doit être du type scalaire.

L'instruction SELON choisit pour l'exécution, l'instruction dont la valeur de l'étiquette est égale à la valeur courante du sélecteur. Après la terminaison de cette instruction, le contrôle passe à la fin de l'instruction SELON. Si la valeur courante du sélecteur ne coïncide pas avec l'une des étiquettes mentionnées, l'instruction qui se trouve derrière le mot réservé AUTRE sera exécutée. Dans ce cas, l'absence de AUTRE entraîne le passage du contrôle à la suite du programme.

Exemple:

```
SI CODE-DE1 = ab ALORS PREMIER-DE-ab := VRAI
                 SINON PREMIER-DE-ab := FAUX
                 FSI
SELON ind-rep-UR-DE1 FAIRE
LIBRE :
OCC : APPEL AE-IAI
...   ...
AUTRE PROVOQUER etabli-mal-deroule
FINSELON
```

5.1.2.3. Instruction répétitive

Syntaxe:

<instruction répétitive> ::= <instruction tantque> / <instruction répéter>/
<instruction pour>

5.1.2.3.1. Instruction TANTQUE

<instruction tantque> ::= TANTQUE <expression> FAIRE <instruction>

Sémantique:

Au début, la valeur de l'expression est calculée. Si cette valeur est VRAI, l'instruction qui se trouve après FAIRE est exécutée une fois, puis l'expression est calculée de nouveau, et ainsi de suite. Si cette valeur est FAUX, l'exécution de l'instruction TANTQUE est terminée.

Exemple:

```
TANTQUE N >= 4 FAIRE DEBUT
                    NCH-attendu := 4 ;
                    ENVOI (OR-DR, recevoir-N-chif) ;
                    .....
                    FIN
```

5.1.2.3.2. Instruction REPETER

<instruction répéter> ::= REPETER <instruction> {;<instruction>}
JUSQUA <expression>

Sémantique:

Au début, la séquence d'instructions comprises entre REPETER et JUSQUA est exécutée, puis la valeur de l'expression est calculée. Si cette valeur est VRAI, la séquence d'instructions sera exécutée de nouveau et ainsi de suite.

Si l'expression prend la valeur FAUX, l'exécution de l'instruction REPETER est terminée.

Exemple:

```
REPETER DEBUT
        APPEL interro-UR-DE1 (test-faisceau) ;
        UR-DE1 := UR-DE1 + 1 ;
        FIN
JUSQUA ind-rep-UR-DE1 =/ tronçon-epuise ;
```

5.1.2.3.3. Instruction POUR

```
<instruction pour> ::= POUR <variable de contrôle> :=
                       <valeur initiale> A <valeur finale> FAIRE <instruction>
<variable de contrôle> ::= <identificateur>
<valeur initiale> ::= <expression>
<valeur finale> ::= <expression>
```

Sémantique:

Cette instruction provoque l'exécution de l'instruction contrôlée pour les valeurs entières croissantes de la variable de contrôle à partir de la valeur initiale jusqu'à la valeur finale.

5.1.2.4. Instruction AVEC

```
<instruction avec> ::= AVEC <liste de variables d'enregistrement>
                       FAIRE <instruction>
<liste de variables d'enregistrement> ::= <variable enregistrement>
                                          {,<variable enregistrement>}
```

Sémantique:

Lorsqu'une variable d'enregistrement figure entre les symboles AVEC et FAIRE, les utilisations des champs de cet enregistrement dans l'instruction suivant le FAIRE peuvent se faire sans spécification d'opération de sélection.

Exemple:

```
SI VAR DATE : ENREGISTREMENT
              MOIS : 1 .. 12 ;
              ANNEE : ENTIER ;
              JOUR : 1 .. 31 ;
              FIN
AVEC DATE FAIRE
SI MOIS = 12 ALORS DEBUT
                   MOIS := 1 ; ANNEE := ANNEE + 1 ;
                   FIN
             SINON MOIS := MOIS + 1
             FSI
```

Ce qui est équivalent à:

```
SI DATE.MOIS := 12 ALORS DEBUT
                         DATE.MOIS := 1 ;
                         DATE.ANNEE := DATE.ANNEE + 1 ;
                         FIN
                   SINON DATE.MOIS := DATE.MOIS + 1
                   FSI
```

Dans une instruction AVEC PTR P FAIRE, la valeur du pointeur P est calculée une fois pour toutes au départ. Ceci n'interdit pas de modifier la valeur de P dans les instructions dépendant de AVEC, mais il n'y a pas d'effet de bord.

5.2. Les instructions spécialisées [A. OLAIWAN, 1978]

Tous les organes de l'auto-commutateur téléphonique communiquent par des messages. La majeure partie des instructions spécialisées servent à manipuler ces messages. Un message est constitué d'un entête et de plusieurs paramètres. L'entête du message est un paramètre de code fonction de type scalaire. Il identifie le message et détermine les informations portées par lui. Les messages sont de type enregistrement, leur structure est prédéfinie au niveau du système d'exploitation. Un message est reçu par le système dans un tampon d'entrée-sortie inaccessible au programmeur. Les transmissions des paramètres du tampon à l'enregistreur à la réception et de l'enregistreur vers le tampon à l'émission, sont implicites.

Syntaxe:

<instruction spécialisée> ::= <instruction QUAND> /
<instruction QUANDNON> /
<instruction SUIVANT> /
<instruction ENVOI> /
<instruction RECEVOIR> /
<instruction AVANT-ATTENDRE> /
<instruction TERMINER-PROCESSUS>

Sémantique:

Les instructions spécialisées ont été définies dans le but d'adapter le langage à la nature particulière du programme téléphonique.
Les instructions ENVOI, RECEVOIR et AVANT-ATTENDRE sont les instructions d'entrées-sorties qui expriment la communication et la synchronisation entre le processeur de la commande et les différents processeurs périphériques qu'il pilote.

5.2.1. Instruction QUAND

<instruction QUAND> ::= QUAND <identificateur d'un message attendu>
ALORS <instruction> FINQUAND / QUAND
<identificateur d'un message attendu> ALORS
<instruction> SINON <instruction> FINQUAND

<identificateur d'un message attendu> ::= <identificateur de constante scalaire>

<identificateur de constante scalaire> ::= <identificateur>

Sémantique:

Cette instruction provoque la comparaison entre la variable CODE-FONCTION du message attendu dont l'identificateur figure après le mot réservé QUAND et la variable CODE-FONCTION du message arrivé.
L'égalité entraîne:

. l'appel d'un module système qui effectue la transmission des paramètres du message arrivé dans l'enregistreur correspondant,

. l'exécution de l'instruction qui se trouve derrière le mot réservé ALORS.

L'inégalité entraîne l'exécution de l'instruction se trouvant derrière le mot réservé SINON.

Exemple:

```
QUAND ECHEANCE ALORS FIN-etabli := blocage-interne
               SINON APPEL identifier-erreur
               FINQUAND
```

Dans le cas de l'instruction QUAND - ALORS - FINQUAND, l'inégalité entraîne le passage du contrôle à la suite du programme.

5.2.2. Instruction QUANDNON

<instruction QUANDNON> ::= QUANDNON <identificateur d'un message attendu>
FAIRE <instruction>

Sémantique:

Elle provoque la même comparaison effectuée par l'instruction QUAND. Dans ce cas, l'égalité entraîne seulement l'appel au module système qui effectue la transmission des paramètres du message arrivé dans l'enregistreur correspondant. L'inégalité entraîne l'exécution de l'instruction se trouvant derrière le mot réservé FAIRE.

Exemple:

QUANDNON rep-reserv-posit FAIRE PROVOQUER etabli-mal-deroule ;

5.2.3. Instruction SUIVANT

<instruction suivant> ::= SUIVANT <identificateur du message reçu>
FAIRE <liste suivant> {;<liste suivant>}
AUTRE <instruction> FINSUIVANT
<liste suivant> ::= <liste d'étiquettes suivant> : <instruction>
<liste d'étiquettes suivant> ::= <identificateur d'un message attendu>
{,<identificateur d'un message attendu>}
<identificateur du message reçu> ::= <identificateur>

Sémantique:

De même que l'instruction SELON est un SI-ALORS-SINON multiple, l'instruction SUIVANT est un QUAND multiple. Elle est formée d'une variable de type scalaire (le sélecteur) et d'une liste d'instructions. Chacune d'elles est étiquetée par un identificateur de constante.

Dans le cas d'une coïncidence entre l'une des étiquettes et la valeur courante du sélecteur, l'instruction SUIVANT entraîne:

. l'appel au module système qui fait la transmission des paramètres du message reçu dans l'enregistreur correspondant,

. l'exécution de l'instruction référencée par l'étiquette en question.

La non-coïncidence entraîne l'exécution de l'instruction se trouvant derrière le mot réservé AUTRE.

Exemple:

```
SUIVANT CODE-FONCTION FAIRE
EQ          : APPEL interro-UR-DE1 (test-eq)
PF          : APPEL interro-UR-DE1 (test-faisceau)
GF          : APPEL interro-toutes-UR-DE1
rep-neg-TR  : PROVOQUER FAUX-DE1
AUTRE         PROVOQUER etabli-mal-deroule
FINSUIVANT
```

5.2.4. Instruction ENVOI

<instruction ENVOI> ::= ENVOI (<identificateur d'organe>,
<identificateur du message à émettre>)
<identificateur d'organe> ::= <identificateur de constante scalaire>
<identificateur du message à émettre> :: <identificateur de constante scalaire

Sémantique:

C'est une instruction de sortie. Elle exprime l'émission d'un message de l'organe de commande vers les organes qu'il pilote. L'identificateur d'organe désigne l'organe destinataire, l'identificateur de message à émettre désigne le message qui doit être envoyé à cet organe.

Exemple:

ENVOI (RCX, CNX-RNC-ENVOI-IN) ;

Cette instruction exprime l'émission à l'organe RCX (réseau de connexion) du message de code fonction CNX-RNC-ENVOI-IN qui exprime la demande au réseau de connexion d'envoyer la tonalité d'invitation à numéroter et de connecter le récepteur à numérotation clavier.

5.2.5. Instruction RECEVOIR

<instruction RECEVOIR> ::= RECEVOIR (<identificateur d'organe>
<ensemble de messages attendus>)
<ensemble de messages attendus> ::= <ensemble>

Sémantique:

La plupart des messages émis par la commande vont entraîner l'arrivée de messages en réponse. La politique choisie est de garder systématiquement tous les messages. Deux types de délais sont distingués, suivant que le message attendu dépend des conditions téléphoniques externes (abonnés, circuits) ou de conditions internes (temps de réponse des organes de l'auto-commutateur). L'instruction RECEVOIR est une instruction d'entrée définie pour ce dernier type de délai.

Exemple:
RECEVOIR (RCX, [compte rendu])

Cette instruction exprime l'attente du message compte-rendu de l'organe réseau de connexion.

5.2.6. Instruction AVANT-ATTENDRE

<instruction AVANT-ATTENDRE> ::= AVANT <temporisation> ATTENDRE
(<nombre>, (<identificateur d'organe>,
<ensemble de messages attendus>)
[, (<identificateur d'organe>,
<ensemble de messages attendus>)]
<temporisation> ::= <nombre> <unité de temps> /
<identificateur de variable de type entier>
<unité de temps>
<unité de temps> ::= SEC / MIN
<identificateur de variable de type entier> ::= <identificateur>

Sémantique:

Cette instruction d'entrée est définie pour le cas où le(s) message(s) attendu(s) dépend(ent) des conditions externes. Le nombre qui suit ATTENDRE indique le nombre d'organes desquels on attend des messages. Il s'agit d'attendre de chaque organe un ou plusieurs messages.

L'instruction AVANT <temporisation> initialise un compteur à une durée voulue. Cette durée est égale à la valeur de la temporisation. Elle représente le temps pendant lequel on attend le(s) message(s). Le processeur de comptage fait décrémenter ce compteur. Le passage à zéro de ce dernier indique l'échéance de la garde.

Exemples:

AVANT 16 SEC ATTENDRE (1, (ORDR, [p-ere-imp-reçu]) ;

Elle exprime l'attente pendant 16 secondes du message première-impulsion-reçue de l'organe de raccordement du demandeur.

AVANT 1 MIN ATTENDRE (2, (OR-DR, [RACCRO-DR]), (OR-DE, [RACCRO-DE])) ;

Elle exprime l'attente pendant une minute de la réception des deux messages: le RACCRO-DR de l'organe de raccordement du demandeur et le RACCRO-DE de l'organe de raccordement du demandé.

5.2.7. Instruction TERMINER-PROCESSUS

<instruction TERMINER-PROCESSUS> ::= TERMINER-PROCESSUS

Sémantique:

L'exécution de cette instruction implique la terminaison du programme téléphonique et la libération de l'enregistreur qui a été alloué par le module de commande pour le traitement du processus.

Exemple d'une partie du programme téléphonique:

```
présélection: DEBUT
              SI DISCRI-DR.CL ALORS DEBUT
                                    ENVOI (ETA, reserv-RNC) ;
                                    QUANDNON rep-posit-reserv FAIRE PROVOQUER
                                                              etabli-mal-derou
                                    ENVOI (RCX, CNX-RNC-envoi-IN) ;
                                    RECEVOIR (RCX, [compte-rendu]) ;
                                    QUANDNON compte-rendu FAIRE PROVOQUER
                                                          presel-mal-deroule ;
                                    UNITE := RNC ;
                                    FIN
                                SINON DEBUT
                                    ENVOI (RCX, CNX-TONAL-IN) ;
                                    RECEVOIR (RCX, [compte-rendu]) ;
                                    QUANDNON compte-rendu FAIRE PROVOQUER
                                                          etabli-mal-deroule ;
                                    UNITE := UR-DR ;
                                    FIN
                                FSI
              FIN
```

CHAPITRE 4

NOTES SUR LE LANGAGE PROPOSÉ

1. INTRODUCTION

Le but de ce chapitre est de présenter les différentes considérations qui nous ont conduits à choisir la plupart des concepts adoptés dans ce langage.

Pour cela, les données et les instructions vont être reprises dans l'ordre dans lequel elles ont été présentées au chapitre précédent.

2. COMMENTAIRES, IDENTIFICATEURS ET NOMBRES

. Les commentaires sont utilisés pour favoriser la lisibilité du programme.

. Un identificateur peut être formé de lettres, de chiffres et du caractère spécial '-' . Il commence par une lettre et ne doit pas se terminer par '-'. L'utilisation du caractère '-' dans les identificateurs est utile pour la lisibilité.

. Les nombres sont uniquement des entiers non signés. Ceci est dû au fait que les entiers négatifs ne sont jamais manipulés par le programme téléphonique.

3. STRUCTURE DU PROGRAMME

Les concepts de structuration du programme sont le bloc, la procédure et l'exception.

La structuration permet d'une part de modulariser le programme et d'autre part de favoriser sa lisibilité et sa documentation.

La notion de données locales aux procédures et aux exceptions n'existe pas. Toutes les données téléphoniques sont implémentées globalement. Ceci a pour avantage de diminuer le temps d'accès aux données, ainsi que la place nécessaire à la gestion de la pile d'exécution d'un processus. L'économie de la mémoire nécessaire à la pile d'exécution d'un processus est importante, car il y a environ 1 000 processus actifs dans le système.

La possibilité d'étiqueter un bloc est adoptée puisqu'elle favorise la lisibilité du programme.

4. DONNÉES

La description des propriétés des données forme un complément à la description des instructions. Elle permet d'écarter quelques risques d'erreurs et de contrôler la consistance du programme, ce qui contribue à la sécurité et à l'efficacité.

Dans le langage proposé, toutes les données téléphoniques sont décrites dans les parties définition et déclaration. Les définitions et les déclarations permettent d'introduire de nouveaux objets dans le langage. Elles doivent précéder les instructions et doivent apparaître dans l'ordre indiqué dans le chapitre précédent. Cet ordre est respecté pour des questions de lisibilité.

4.1. Définition des constantes

La définition d'une constante introduit un identificateur qui est synonyme de cette constante. L'utilisation des synonymes est très importante pour la paramétrisation du programme. Elle le rend plus lisible et plus facile à changer.

4.2. Définition des types

L'objectif des types est de déterminer les opérations qui peuvent s'appliquer à un objet. Les types indiquent indirectement la taille de la zone mémoire nécessaire pour un objet et indiquent comment les composants d'une donnée sont liés les uns aux autres. L'utilisation des types favorise la lisibilité du programme et le rend plus facile à écrire et à modifier.

Dans le but de définir un langage très simple, nous avons choisi un nombre restreint de types. Les types simples et structurés définis dans le langage, sont jugés capables de décrire toutes les données téléphoniques. Il faut noter que la définition récursive des types n'est pas adoptée car elle n'est pas nécessaire.

4.2.1. Types_simples

Un type scalaire est défini par un ensemble ordonné de valeurs dénotées par des identificateurs de constante, qui sont ainsi déclarés de manière implicite. L'utilisation du type scalaire favorise la lisibilité du programme.

Exemple:

service-restreint = (local, régional, international)

La variable service-restreint est une variable indiquant si l'abonné a droit à l'appel local, régional ou international.

Les deux types standards du type scalaire sont le type booléen et le type entier. Les types réel et chaîne de caractères ne sont pas adoptés car ils ne sont pas nécessaires dans le cas du programme téléphonique.

L'attribut de taille permet de donner explicitement la taille de la zone mémoire nécessaire pour un objet donné. Ceci est important pour optimiser l'implantation mémoire.

L'attribut d'accès LECT est utilisé pour assurer la sécurité des données. Une donnée est déclarée avec cet attribut dans le but d'être accédée uniquement en lecture. Elle ne devra jamais être modifiée dans le programme.

Pour des questions de lisibilité, les attributs de taille, de codage et d'accès, doivent apparaître dans l'ordre indiqué au chapitre précédent.

4.2.2. Types structurés

4.2.2.1. Type enregistrement

Un enregistrement est composé d'objets qui peuvent être de différents types. Chacun de ces objets s'appelle champ. Un champ peut être de type simple ou structuré.

Un opérande de type enregistrement, ainsi que ses champs de type structuré, ne peuvent pas être affectés globalement. Il est possible d'affecter seulement les champs de type simple.

4.2.2.2. Type tableau

Un tableau est formé d'objets de même type simple appelés éléments du tableau. La taille d'un tableau est connue à la déclaration. Un tableau peut avoir au plus deux dimensions.

Cette restriction sur le type des éléments d'un tableau et sur le nombre de ses dimensions est liée au fait que toutes les données téléphoniques de type tableau sont composées d'éléments simples et ont au plus deux dimensions.

Un opérande de type tableau ne peut pas être affecté globalement, mais il est possible d'affecter ses éléments.

4.3. Déclaration des exceptions

L'exception est une procédure particulière car son exécution termine celle d'un processus. Elle travaille sur des données téléphoniques implémentées globalement. Elle n'a pas de variables locales ni de paramètres formels.

L'utilisation de la notion d'exception est très importante dans le cas du programme téléphonique. Elle permet de sortir du programme d'une manière structurée. Son utilisation a permis d'éliminer les instructions ALLEA et EXIT du langage.

Une version du programme téléphonique a été récemment écrite en utilisant les exceptions et les procédures. Cette version est comparée à une autre version que nous avons écrite en utilisant les indicateurs booléens et l'instruction EXIT. Le résultat de la comparaison a montré que la taille du programme est diminuée de 40 % environ.

4.4. Déclaration des procédures

Une procédure est un moyen fondamental d'abstraction et de structuration du programme. Elle peut avoir des paramètres formels de type simple. Elle ne travaille que sur des données implémentées globalement, ce qui évite l'utilisation d'une pile.

5. INSTRUCTIONS COURANTES

5.1. Instruction d'affectation

Elle permet d'affecter la valeur d'une expression à une variable. Dans le cas du programme téléphonique, l'affectation n'est possible que pour les variables de type simple.

5.2. Instruction d'appel de procédure

Elle est formée du mot réservé APPEL suivi du nom de la procédure, suivi de la liste des paramètres éventuels.

Les paramètres effectifs donnés à l'appel d'une procédure sont des variables de type simple. Le passage de ces paramètres se fait par référence, ce qui est très efficace dans notre cas, car les paramètres ne sont pas fréquemment accédés. Après l'exécution de la procédure, il y aura retour à l'instruction qui suit l'instruction d'appel.

5.3. Instruction de provocation d'exception

Elle est formée du mot PROVOQUER suivi du nom de l'exception. Elle entraîne l'exécution du corps de l'exception, puis la sortie de tout le processus. C'est donc un appel de procédure particulier. La fin de l'exécution de l'exception n'entraîne pas le retour au point d'appel, mais la sortie de tout le processus.

5.4. Instruction composée

Elle est formée de plusieurs instructions qui doivent s'exécuter en séquence. Pour favoriser la lisibilité du programme, une instruction composée peut être étiquetée.

5.5. Instruction AVEC

L'utilisation de l'instruction AVEC permet de sélectionner les champs de l'enregistrement sans spécifier l'identificateur de ce dernier.

L'utilisation de l'instruction AVEC simplifie l'écriture du programme téléphonique, car toutes ses données sont groupées dans un enregistreur de type enregistrement.

6. INSTRUCTIONS SPÉCIALISÉES

6.1. Instruction QUAND

Elle a la signification d'une instruction conditionnelle SI qui possède les caractéristiques suivantes:

. l'expression qui suit le mot SI est toujours une comparaison de deux variables de type scalaire: le code fonction du message attendu et le code fonction du message reçu ;

. la valeur VRAI de l'expression entraîne:

1) l'appel d'un module système qui transmet les paramètres du message reçu du tampon d'entrée-sortie vers l'enregistreur correspondant ;

2) l'exécution de l'instruction qui se trouve après le mot ALORS.

La définition de l'instruction QUAND permet donc d'une part de simplifier l'écriture en remplaçant
SI <identificateur d'un message attendu> = <identificateur du message reçu>
par QUAND <identificateur d'un message attendu>,
et d'autre part, elle permet de faire la transmission des paramètres du message reçu avant l'exécution de l'instruction qui suit le mot ALORS et qui dans la plupart des cas exploite ces paramètres.

6.2. Instruction QUANDNON

Elle provoque la même comparaison que celle effectuée par l'instruction QUAND. Mais dans ce cas, la valeur VRAI de l'expression entraîne seulement l'appel au module système de transmission de paramètres.

Sa définition était nécessaire car les cas de test de non-arrivée des messages attendus (qui sont considérés comme des cas d'erreurs), sont très fréquents. Ceci est dû au fait que ces cas sont traités dans le même langage et dans le même programme.

L'introduction de l'instruction QUANDNON dans le langage a pour but d'éviter l'utilisation de l'instruction QUAND sous la forme

QUAND <identificateur d'un message attendu> ALORS
SINON <instruction>
FSI

et par conséquent de simplifier l'écriture.

6.3. Instruction SUIVANT

De même que l'instruction SELON est un SI ALORS SINON multiple, l'instruction SUIVANT est un QUAND multiple. A la différence de l'instruction SELON, l'option AUTRE est obligatoire. Car si la valeur du sélecteur ne coïncide pas avec l'une des étiquettes, cela signifie que le message attendu n'est pas arrivé, ce qui est considéré comme un cas d'erreur qui nécessite d'être traité. Le traitement se fait par l'instruction qui se trouve derrière AUTRE.

6.4. Instructions d'entrées-sorties

Ce sont les instructions ENVOI, RECEVOIR et AVANT-ATTENDRE. Elles expriment la communication et la synchronisation entre l'organe (le processeur) de la commande et les différents organes (processeurs) périphériques qu'il pilote.

L'instruction ENVOI est une instruction de sortie.
Elle exprime l'émission d'un message déterminé à un organe déterminé.

L'instruction RECEVOIR est une instruction définie pour le cas où le(s) message(s) attendu(s) dépend(ent) des conditions internes (temps de réponse des processeurs de l'auto-commutateur).
Elle exprime l'attente d'un seul ou de plusieurs messages d'un processeur déterminé.

L'instruction AVANT-ATTENDRE est une instruction définie pour le cas où le(s) message(s) attendu(s) dépend(ent) des conditions externes (abonnés, circuits).
Elle exprime l'attente pendant un certain temps (qui dépend parfois de la réaction humaine) de un ou de plusieurs messages de chaque processeur.

CHAPITRE 5

IMPLÉMENTATION

1. INTRODUCTION

L'option choisie pour implémenter le langage est d'écrire le programme téléphonique sous forme de macro-instructions.

La démarche suivie a consisté à faire une analyse de toutes les instructions du langage, à écrire leur format intermédiaire, et enfin à représenter d'une façon détaillée l'organigramme de chaque macro-instruction.

L'étude menée dans cette direction a été largement inspirée du macro-système réalisé par M. GRIFFITHS et M. PELTIER [M. GRIFFITHS, 1969].

2. NOTES PRÉLIMINAIRES

Dans les organigrammes des macros, seules les données locales à ces macros sont déclarées. Celles qui leur sont globales sont:

compteur-etiq: c'est une variable entière utilisée pour éviter les définitions multiples d'une même étiquette à l'intérieur d'une macro ;

pile-etiq: c'est un tableau à une seule dimension permettant de mémoriser les valeurs des compteurs-etiq correspondantes aux différents niveaux d'imbrication de macros ;

pile-code: c'est un tableau à une seule dimension utilisé pour mémoriser les codes des macros ;

pile-pour: c'est un tableau permettant de mémoriser les valeurs des expressions relatives aux instructions POUR qui peuvent bien sûr être imbriquées ;

pile-expression: c'est un tableau utilisé pour mémoriser les valeurs des expressions qui sont des paramètres pour les instructions SELON et SUIVANT ;

pile-utilisateur: c'est un tableau utilisé pour mémoriser des informations relatives à un processus ;

pointeur phase: c'est une variable entière susceptible de prendre la valeur du compteur ordinal ;

code-SI, code-SINON, code-SELON, ...: sont des constantes utilisées pour contrôler le nombre de macros. Elles permettent de détecter s'il y a une macro FINSI sans qu'elle soit précédée par une macro SI, par exemple, et ainsi de suite pour toutes les autres macros.

Dans les organigrammes, la plupart des identificateurs figurent avec l'un des préfixes: gen call ou appel.

. Les identificateurs commençant par gen indiquent des macros qui génèrent du code exécutable.

. Les identificateurs commençant par call indiquent un appel à un module du système d'exploitation.

. Les identificateurs commençant par appel indiquent des macros déjà définies.

. Les identificateurs qui ne commencent pas par un préfixe, indiquent des procédures du macro-assembleur lui-même.

Dans ce qui suit, on donne la description des macros et des procédures citées plus haut.

2.1. Les macros qui génèrent du code exécutable

1) macro genevaluer-condition (relop, A, B)

/* génère la comparaison entre A et B et positionne la condition de branchement. Relop indique l'opérateur de relation désiré */

2) macro genbranch (T, val)

/* génère le branchement d'après l'évaluation de la condition ou de l'expression. Si T prend la valeur M, ceci indique que l'instruction est un branch et non un jump */

3) macro genbranch-incond (T, val)

/* c'est un branchement inconditionnel à l'étiquette dont le type est spécifié par T et dont la valeur est spécifiée par val */

4) macro genevaluer (expression)

/* génère l'évaluation de l'expression qui lui est passée comme paramètre */

5) macro gentransmettre (nom-pile, nom-mesg1, nom-mesg2, nb-mesg, org)
   /* génère l'empilement des paramètres nom-mesg1, ..., dans la pile */

6) macro gendesactiver (nom-pile)
   /* génère la sauvegarde du pointeur de la pile */

7) macro genretour-système
   /* génère la séquence de retour au système */

8) macro genassigner (A, B)
   /* génère l'évaluation de l'opérande B et affecte la variable A de la valeur de B */

9) macro genincrementer (A)
   /* génère l'incrémentation de A */

2.2. Les macros qui font appel aux modules système

1) macro call-transmettre-param-mesg
   /* cette macro génère l'appel au module système qui transmet les paramètres du message arrivé du tampon d'entrée-sortie vers l'enregistreur correspondant */

2) macro call-init-garde (val-tempo, enreg-courant)
   /* elle génère l'appel au module système qui initialise à la valeur val-tempo le compteur correspondant à l'enregistreur courant */

3) macro call-attendmesg
   /* génère l'appel au module système attendmesg */

4) macro call-FIN-PROCESSUS
   /* génère l'appel au module système FIN-PROCESSUS */

2.3. Les macros qui figurent dans d'autres macros et qui sont déjà définies

1) macro appel-AVANT (tempo-système, enreg-courant)

/* c'est la macro AVANT déjà définie, ayant pour paramètres: tempo-système et enreg-courant */

2) macro appel-ATTENDRE (nb-organe, organe1, nb-mesg1, mesg1, mesg2, .., mesgn, organe2, nb-mesg2, mesg1, mesg2, .., mesgn, ...)

/* c'est la macro ATTENDRE déjà définie, ayant pour paramètres: nb-organe, organe1, ... */

2.4. Les procédures

1) procédure incrémenter (P)

P ← P + 1 /* P = compteur d'étiquette */

2) procédure monter-niveau

LV ← LV + 1 /* augmente de 1 le compteur de niveau commun à toutes les piles */

3) procédure empiler (nom-pile, val)

/* fait empiler dans la pile la valeur du second paramètre:

. le paramètre nom-pile spécifie la pile,

. le paramètre val spécifie la valeur à empiler */

4) procédure definiretiq (T, val)

/* le paramètre T spécifie le type d'étiquette à définir, qui pourra être:

C: indique que l'étiquette est une chaîne de caractères ; dans ce cas, le deuxième opérande val sera le nom de l'étiquette définie

M: indique que le deuxième opérande est une valeur ; l'étiquette définie sera M concaténé à (valeur) */

5) procédure vérifier-niveau

/* teste si le niveau commun des piles signifie pile vide
si oui message d'erreur
sinon rien ; */

6) procédure verifier-code (val-code)

/* teste si le sommet de la pile des codes vaut val-code. Val-code est la valeur du code de la macro d'entrée. Ce test se fait dans les macros de sortie "fin nom macro" ; si le test est faux, sortir le message d'erreur correspondant et appeler la procédure descendre-niveau */

7) procédure descendre-niveau

/* LV ← LV - 1 ; elle fait soustraire 1 à la valeur du compteur commun à toutes les piles */

8) procédure verifier-sinon

/* compare le paramètre de la macro FINSI à la chaîne de caractères "sinon". L'inégalité fait sortir un message d'erreurs indiquant que le paramètre de la macro FINSI est illégal */

9) procédure déclarer-local (P)

/* fait déclarer locale à la macro la variable P */

10) procédure affecter (A, B)

/* la variable A prend la valeur de la variable B */

11) procédure depiler (nom-pile)

/* donne la valeur du sommet de la pile spécifiée par nom-pile */

Dans ce qui suit, nous décrivons pour chaque instruction du langage proposé, son format intermédiaire qui détermine le nombre de macro-instructions dont elle est formée et son organigramme détaillé.

3. FORMATS INTERMEDIAIRES ET ORGANIGRAMMES DES INSTRUCTIONS

3.1. Instruction SI

3.1.1. Format intermédiaire

<instruction SI> ::= <macro SI> <expression> <liste d'instructions>
<macro FINSI> <partie SINON>
<partie SINON> ::= <vide> / SINON <liste d'instructions> <macro FINSINON>

La liste d'instructions est formée de zéro ou plusieurs instructions du langage téléphonique, qui sont elles-mêmes écrites sous forme de macro-instructions.

3.1.2. Macros de l'instruction SI

3.1.2.1. Macro SI

La macro SI évalue l'expression booléenne, et si cette dernière prend la valeur FAUX, il y aura branchement vers l'instruction qui se trouve derrière la macro FINSI.

Organigramme:

```
        MACRO
NOM     SI expression
        incrémenter (compteur-etiq) ;
        monter-niveau ;
        empiler (pile-etiq, compteur-etiq) ;
        empiler (pile-code, code-SI) ;
        definiretiq (C, NOM) ;
        genevaluer (expression) ;
        genbranch (M, compteur-etiq) ;
        MEND
```

3.1.2.2. Macro FINSI

Dans le cas où la partie SINON existe, la macro FINSI génère le branchement vers la suite du programme. Dans le cas contraire, elle ne génère pas de code exécutable.

Organigramme:

```
        MACRO
NOM     FINSI SINON
        definiretiq (C, NOM) ;
        verifier-niveau ;
        verifier-code (code-SI) ;
        si 'SINON' ≠ 'vide' alors
        debut
           verifier-SINON ;
           incrementer (compteur-etiq) ;
           genbranch-incond (M, compteur-etiq) ;
           definiretiq (M, depiler (pile-etiq) ;
           empiler (pile-etiq, compteur-etiq) ;
           empiler (pile-code, code-SINON) ;
        fin
        sinon debut
           definiretiq (M, depiler (pile-etiq) ;
           descendre-niveau ;
           fin
        MEND
```

3.1.2.3. Macro FINSINON

La macro FINSINON ne génère pas de code exécutable, mais du code utile au macro-assembleur lui-même pour la gestion des étiquettes.

Organigramme:

```
        MACRO
NOM     FINSINON
        definiretiq (C, NOM) ;
        verifier-niveau ;
        verifier-code (code-SINON) ;
        definiretiq (M, depiler (pile-etiq));
        descendre-niveau ;
        MEND
```

3.2. Instruction SELON

3.2.1. Format intermédiaire

<instruction SELON> ::= <macro SELON> <expression> <macro DEBUTCAS>
<liste d'instructions> <macro FINCAS>
{;<macro DEBUTCAS> <liste d'instructions>
<macro FINCAS> } <macro FINSELON>

3.2.2. Macros de l'instruction SELON

3.2.2.1. Macro SELON

L'évaluation de l'expression est le seul code exécutable généré par la macro SELON.

Organigramme:

```
        MACRO
NOM     SELON expression
        definiretiq (C, NOM) ;
        incrementer (compteur-etiq) ;
        monter-niveau ;
        empiler (pile-etiq, compteur-etiq) ;
        empiler (pile-code, code-SELON) ;
        empiler (pile-expression, genevaluer (expression)) ;
        MEND
```

3.2.2.2. Macro DEBUTCAS

La macro DEBUTCAS génère la comparaison entre l'expression évaluée par la macro SELON et la valeur de l'étiquette qui lui est passée en paramètre. L'inégalité entraîne le branchement vers l'instruction qui se trouve derrière la macro FINCAS.

Organigramme:

```
        MACRO
        DEBUTCAS ETIQ
        incrémenter (compteur-etiq) ;
        empiler (pile-code, code-DEBUTCAS) ;
        empiler (pile-etiq, compteur-etiq) ;
        genevaluer-condition (relop, ETIQ, depiler (pile-expression)) ;
        genbranch (M, depiler (pile-etiq)) ;
        MEND
```

3.2.2.3. Macro FINCAS

Elle génère le branchement vers l'instruction qui se trouve derrière la macro FINSELON.

Organigramme:

```
        MACRO
        FINCAS
        verifier-code (code-DEBUTCAS) ;
        genbranch-incond (M, depiler (pile-etiq)) ;
        definiretiq (M, depiler (pile-etiq)) ;
        MEND
```

3.2.2.4. Macro FINSELON

Elle génère seulement du code utile au macro-assembleur.

Organigramme:

```
        MACRO
NOM     FINSELON
        definiretiq (C, NOM) ;
        vérifier-niveau ;
        vérifier-code (code-SELON) ;
        definiretiq (M, depiler (pile-etiq)) ;
        descendre-niveau ;
        MEND
```

3.3. Instruction TANTQUE

3.3.1. Format intermédiaire

<instruction TANTQUE> ::= <macro TANTQUE> <expression>
<liste d'instructions> <macro FINTANTQUE>

3.3.2. Macros de l'instruction TANTQUE

3.3.2.1. Macro TANTQUE

Elle génère l'évaluation de l'expression booléenne et dans le cas où cette dernière prend la valeur FAUX, il y aura branchement vers l'instruction qui se trouve derrière la macro FINTANTQUE.

Organigramme:

```
        MACRO
NOM     TANTQUE expression
        definiretiq (C, NOM) ;
        incrementer (compteur-etiq) ;
        monter-niveau ;
        empiler (pile-etiq, compteur-etiq) ;
        empiler (pile-code, code-TANTQUE) ;
        definiretiq (M, compteur-etiq) ;
        incrementer (compteur-etiq) ;
        genevaluer (expression) ;
        genbranch (M, compteur-etiq) ;
        MEND
```

3.3.2.2. Macro FINTANTQUE

Elle génère le branchement inconditionnel vers la macro TANTQUE pour évaluer à nouveau l'expression.

Organigramme:

```
        MACRO
NOM     FINTANTQUE
        declarer-local (var-x) ;
        verifier-niveau ;
        verifier-code (code-TANTQUE) ;
        affecter (var-x, incrémenter (depiler (pile-etiq))) ;
        definiretiq (C, NOM) ;
        genbranch-incond (M, depiler (pile-etiq)) ;
        definiretiq (M, var-x) ;
        descendre-niveau ;
        MEND
```

3.4. Instruction REPETER

3.4.1. Format intermédiaire

<instruction REPETER> ::= <macro REPETER> <liste d'instructions>
<macro FINREPETER> <expression>

3.4.2. Macros de l'instruction REPETER

3.4.2.1. Macro REPETER

Elle génère seulement du code utile au macro-assembleur.

Organigramme:

```
        MACRO
NOM     REPETER
        definiretiq (C, NOM) ;
        incrementer (compteur-etiq) ;
        monter-niveau ;
        empiler (pile-code, code-REPETER) ;
        definiretiq (M, compteur-etiq) ;
        incrementer (compteur-etiq) ;
        MEND
```

3.4.2.2. Macro FINREPETER

Elle génère l'évaluation de l'expression booléenne et dans le cas où celle-ci prend la valeur FAUX il y aura branchement vers l'instruction qui se trouve derrière la macro REPETER ; dans le cas contraire il y aura branchement vers la suite du programme.

Organigramme:

```
        MACRO
NOM     FINREPETER expression
        declarer-local (var-z) ;
        verifier-niveau ;
        verifier-code (code-REPETER) ;
        affecter (var-z, incrementer (depiler (pile-etiq))) ;
        genevaluer (expression) ;
        genbranch (M, var-z) ;
        definiretiq (C, NOM) ;
        genbranch-incond (M, pile-etiq) ;
        definiretiq (M, var-z) ;
        descendre-niveau ;
        MEND
```

3.5. Instruction POUR

3.5.1. Format intermédiaire

<instruction POUR> ::= <macro POUR> <expression-1>
<liste d'instructions> <macro FINPOUR>
<expression-2>

3.5.2. Macros de l'instruction POUR

3.5.2.1. Macro POUR

Elle génère l'évaluation de l'expression-1.

Organigramme:

```
        MACRO
NOM     POUR expression-1
        definiretiq (C, NOM) ;
        incrementer (compteur-etiq) ;
        monter-niveau ;
        empiler (pile-etiq, compteur-etiq) ;
        empiler (pide-code, code-POUR) ;
        definiretiq (M, compteur-etiq) ;
        empiler (pile-POUR, genevaluer (expression-1));
        MEND
```

3.5.2.2. Macro FINPOUR

Elle génère l'évaluation de l'expression-2, l'incrémentation de la variable contenant la valeur de l'expression-1 et la comparaison entre ces deux valeurs. Si la première valeur est plus petite que la deuxième, il y aura branchement vers l'instruction qui se trouve derrière la macro FINPOUR.

Organigramme:

```
          MACRO
NOM       FINPOUR expression-2
          declarer-local (var-x) ;
          verifier-niveau ;
          verifier-code (code-POUR) ;
          affecter (var-x, incrémenter (depiler (pile-etiq))) ;
          genevaluer-condition (relop, genincrementer (depiler (pile-POUR)),
                                       genevaluer (expression-2)) ;
          genbranch (M, depiler (pile-etiq)) ;
          definiretiq (M, var-x) ;
          descendre-niveau ;
          MEND
```

3.6. Macro QUAND

3.6.1. Format intermédiaire

```
<instruction QUAND> ::= <macro QUAND> <expression>
                        <liste d'instructions> <macro FINQUAND>
                        <partie FINSINONQUAND>
<partie FINSINONQUAND> ::= <vide> / SINON <liste d'instructions>
                           <macro FINSINONQUAND>
```

3.6.2. Macros de l'instruction QUAND

3.6.2.1. Macro QUAND

La macro QUAND évalue l'expression booléenne et si celle-ci est VRAI, il y aura appel à un module système qui se charge de la transmission des paramètres du message arrivant du tampon d'entrée-sortie vers l'enregistreur: Si l'expression a la valeur FAUX, il y aura branchement à l'instruction qui se trouve derrière la macro FINQUAND.

Organigramme:

```
        MACRO
NOM     QUAND expression
        incrémenter (compteur-étiq) ;
        monter-niveau ;
        empiler (pile-etiq, compteur-etiq) ;
        empiler (pile-code, code-QUAND) ;
        definiretiq (C, NOM) ;
        genevaluer (expression) ;
        genbranch (M, compteur-etiq) ;
        call-transmettre-param-msg ;
        MEND
```

3.6.2.2. Macro FINQUAND

Dans le cas où la partie SINON existe, la macro FINQUAND génère le branchement vers la suite du programme. Dans le cas contraire, elle ne génère pas de code exécutable.

Organigramme:

```
        MACRO
NOM     FINQUAND SINON
        definiretiq (C, NOM) ;
        verifier-niveau ;
        verifier-code (code-QUAND) ;
        si 'SINON' ≠ 'vide' alors
        debut
             verifier-SINON ;
             incrementer (compteur-etiq) ;
             genbranch-incond (M, compteur-etiq) ;
             definir-etiq (M, depiler (pile-etiq)) ;
             empiler (pile-code, code-SINON) ;
        fin
        sinon debut
                  definiretiq (M, depiler (pile-etiq)) ;
                  descendre-niveau ;
              fin
        MEND
```

3.6.2.3. Macro FINSINONQUAND

Elle ne génère pas de code exécutable, mais du code utile au macroassembleur pour gérer les étiquettes.

Organigramme:

```
        MACRO
NOM     FINSINONQUAND
        definiretiq (C, NOM) ;
        verifier-niveau ;
        verifier-code (code-QUAND) ;
        definiretiq (M, depiler (pile-etiq)) ;
        descendre-niveau ;
        MEND
```

3.7. Instruction QUANDNON

3.7.1. Format intermédiaire

<instruction QUANDNON> ::= <macro QUANDNON> <expression>
<macro FAIRE> <liste d'instructions>
<macro FINFAIRE> <macro FINQUANDNON>

3.7.2. Macros de l'instruction QUANDNON

3.7.2.1. Macro QUANDNON

Elle génère l'évaluation de l'expression et dans le cas où l'expression prend la valeur VRAI, il y aura branchement vers la macro FINQUANDNON. Dans le cas contraire, il y aura exécution de la liste d'instructions.

Organigramme:

```
        MACRO
NOM     QUANDNON expression
        incrémenter (compteur-etiq) ;
        monter-niveau ;
        empiler (pile-etiq, compteur-etiq) ;
        empiler (pile-code, code-QUANDNON) ;
        definiretiq (C, NOM) ;
        genevaluer (expression) ;
        genbranch (M, compteur-etiq) ;
        incrémenter (compteur-etiq) ;
        MEND
```

3.7.2.2. Macro FAIRE

Elle ne génère pas de code exécutable.

Organigramme:

```
MACRO
FAIRE
empiler (pile-code, code-FAIRE) ;
MEND
```

3.7.2.3. Macro FINFAIRE

Elle génère le branchement inconditionnel vers la suite du programme.

Organigramme:

```
MACRO
FINFAIRE
verifier-code (code-FAIRE) ;
genbranch-incond (M, incrémenter (depiler (pile-etiq))) ;
definiretiq (M, depiler (pile-etiq)) ;
MEND
```

3.7.2.4. Macro FINQUANDNON

Elle génère l'appel à un module système qui se charge de la transmission des paramètres du message arrivant du tampon d'entrée-sortie vers l'enregistreur correspondant.

Organigramme:

```
        MACRO
NOM     FINQUANDNON
        definiretiq (C, NOM) ;
        verifier-niveau ;
        verifier-code (code-QUANDNON) ;
        call-transmettre-param-mesg ;
        descendre-niveau ;
        definiretiq (M, incrementer (depiler (pile-etiq))) ;
        MEND
```

3.8. Instruction SUIVANT

3.8.1. Format intermédiaire

<instruction SUIVANT> ::= <macro SUIVANT> <expression>
<macro DEBUTCAS-SUIVANT>
<liste d'instructions> <macro FINCAS-SUIVANT>
[;<macro DEBUTCAS-SUIVANT> <liste d'instructions>
<macro FINCAS-SUIVANT>] <macro FINSUIVANT>

3.8.2. Macros de l'instruction SUIVANT

3.8.2.1. Macro SUIVANT

Elle effectue l'évaluation de l'expression.

Organigramme:

```
        MACRO
NOM     SUIVANT expression
        definiretiq (C, NOM) ;
        incrementer (compteur-etiq) ;
        monter-niveau ;
        empiler (pile-etiq, compteur-etiq) ;
        empiler (pile-code, code-SUIVANT) ;
        empiler (pile-expression, genobtenir (expression)) ;
        MEND
```

3.8.2.2. Macro DEBUTCAS-SUIVANT

Cette macro génère le même code que la macro DEBUTCAS de l'instruction SELON et en plus elle génère l'appel au module système qui fait la transmission des paramètres du message arrivant.

Organigramme:

```
MACRO
DEBUTCAS-SUIVANT  ETIQ
incrementer (compteur-etiq) ;
empiler (pile-code, code-DEBUTCAS-SUIVANT) ;
empiler (pile-etiq, compteur-etiq) ;
genevaluer-condition (relop, ETIQ, depiler (pile-expression)) ;
genbranch (M, depiler(pile-etiq)) ;
call-transmettre-param-mesg ;
MEND
```

3.8.2.3. Macro FINCAS-SUIVANT

Elle génère le branchement vers l'instruction qui se trouve derrière la macro FINSUIVANT.

Organigramme:

```
MACRO
FINCAS-SUIVANT
verifier-code (code-DEBUTCAS-SUIVANT) ;
genbranch-incond (M, depiler (pile-etiq)) ;
definiretiq (M, depiler (pile-etiq)) ;
MEND
```

3.8.2.4. Macro FINSUIVANT

Elle génère du code utile au macro-assembleur.

Organigramme:

```
        MACRO
NOM     FINSUIVANT
        definiretiq (C, NOM) ;
        verifier-niveau ;
        verifier-code (code-SUIVANT) ;
        definiretiq (M, depiler (pile-etiq)) ;
        descendre-niveau ;
        MEND
```

3.9. Instruction ENVOI

3.9.1. Format intermédiaire

<instruction ENVOI> ::= <macro ENVOI>

3.9.2. Macro ENVOI

Elle génère l'appel à un module système qui se charge de la construction du message à émettre de la commande vers un organe périphérique. Les paramètres qui indiquent le message et l'organe destinataire sont passés aussi par cette macro ENVOI.

Organigramme:

```
        MACRO
NOM     ENVOI  organe, message
        definiretiq (C, NOM) ;
        gentransmettre (pile-param, message, organe) ;
        call-envoi ;
        MEND
```

3.10. Instruction RECEVOIR

3.10.1. Format intermédiaire

<instruction RECEVOIR> ::= <macro RECEVOIR>

3.10.2. Macro RECEVOIR

Elle génère l'appel à deux modules système:

. le premier se charge d'initialiser le compteur de temps relatif à l'enregistreur courant à la valeur de la temporisation système,

. le second se charge de tester, lorsqu'un message arrive, s'il appartient à l'ensemble des messages attendus.

Organigramme:

```
        MACRO
NOM     RECEVOIR tempo-système, enreg-courant, nb-organe, organe-1, ...
        definiretiq (C, NOM)
        appel-AVANT (tempo-système, enreg-courant) ;
        appel-ATTENDRE (nb-organe, organe-1, nb-mesg, mesg1, ...) ;
        MEND
```

3.11. Instruction AVANT-ATTENDRE

3.11.1. Format intermédiaire

<instruction AVANT-ATTENDRE> ::= <macro AVANT> <macro ATTENDRE>

3.11.2. Macro AVANT

Elle fait appel au module système init-garde qui se charge d'initialiser le compteur de temps relatif à l'enregistreur courant à la durée indiquée par le paramètre val-tempo de la macro AVANT.

Organigramme:

```
        MACRO
NOM     AVANT val-tempo, enreg-courant
        definiretiq (C, NOM) ;
        call-init-garde (val-tempo, enreg-courant) ;
        MEND
```

3.11.3. Macro ATTENDRE

Cette macro sauvegarde la pile utilisateur, transmet les paramètres indiquant les messages attendus au module système attendmesg, puis donne la main au système. Ceci correspond à la désactivation du processus.

Organigramme:

```
        MACRO
NOM     ATTENDRE nb-organe, organe-1, nb-mesg , mesg1, ..
        definiretiq (C, NOM) ;
        genassigner (zero, code-fonction) ;
        gentransmettre (pile-param ; nb-organe, organe-1, nb-mesg, ..) ;
        genassigner (ETIQ, pointeur-phase) ;
        gendesactiver (pile utilisateur) ;
        genretour-système ;
ETIQ    call-attendmesg ;
        MEND
```

3.12. Instruction TERMINER-PROCESSUS

3.12.1. Format intermédiaire

<instruction TERMINER-PROCESSUS> ::= <macro TERMINER-PROCESSUS>

3.12.2. Macro TERMINER-PROCESSUS

Elle fait appel au module système FIN-PROCESSUS qui se charge de libérer l'enregistreur qui a été alloué pour traiter ce processus.

Organigramme:

```
        MACRO
NOM     TERMINER-PROCESSUS enreg-courant
        definiretiq (C, NOM) ;
        call-FIN-PROCESSUS (enreg-courant) ;
        MEND
```

CHAPITRE 6

CONCLUSION

Le langage de programmation proposé a permis de programmer l'algorithme de la commande de l'autocommutateur téléphonique de façon indépendante d'une implémentation particulière.

La pratique a montré qu'un langage de programmation doit être utilisé expérimentalement pendant plusieurs années avant de pouvoir l'être de façon opérationnelle. L'existence d'un compilateur limite une remise en cause du langage [Pleyber, 1978]. Par conséquent dans un premier temps, l'écriture n'est peut-être pas la meilleure solution à la mise en oeuvre du langage.

L'approche que nous avons choisie pour implémenter le langage est d'écrire le programme téléphonique sous forme de macro-instructions. Ces macros sont programmées en utilisant un langage de niveau plus bas que le langage téléphonique et possédant un compilateur. Ceci débouchera, dans un proche avenir, sur un test de l'ensemble du programme sur la maquette de l'autocommutateur.

Nous espérons dans le futur qu'un compilateur sera écrit pour le langage ainsi défini, ce qui facilitera bien sûr son exploitation.

ANNEXE

LE PROGRAMME TÉLÉPHONIQUE

Structure du programme téléphonique

```
COM : début
      [/* déclaration des constantes */]
      [/* déclaration des types */]
      [/* déclaration des variables */]
      [/* déclaration des exceptions */]
      début
      RECEVOIR (URDR, [NOUVEL-APPEL] ) ;
      QUAND NOUVEL-APPEL alors début
                                -----
                                -----
                                -----
                                fin
      TERMINER-PROCESSUS
      FIN
      FIN COM
```

```
COM:'DEBUT'
    /*DECLARATIONS DES CONSTANTES*/

    'CONST' NB-CHIF-LOCAL=6;
            NB-CHIF-NATIONAL=8;
            MAX=3276;
            /*3276 'SEC'=PERIODE MAXIMALE DE CONVERSATION ENTRE 2 ABONNES*/

    /*DECLARATIONS DES TYPES */

    'TYPE' ENREG='ENREGISTREMENT'
                 POINTEUR-PHASE:'ENTIER' 'MOT'(2) 'BIN';
                 CODE-FONCTION:(NOUVEL-APPEL,BR-DR,RUPTURE,....) 'LECT';
                 ORGANE:(URDR,URDE1,URDE2,TR,CG,....)'LECT';
                 TYP-COMM:(NORMAL,NUMCOURT,CCF,....);
                 URDR:'ENTIER' 'MOT'(1) 'BIN' 'LECT';
                 VTLRDR:'ENTIER' 'EB'(6) 'BIN' 'LECT';
                 EQDR:'ENTIER' 'EB'(10) 'BIN' 'LECT';
                 DISCRIDR:'ENREGISTREMENT'
                          DR-TD,DR-ABS:'BOOLEEN' 'LECT';
                          DR-NC,DR-SR:(NATIONAL,INTERNATIONAL,REGIONAL) 'LECT';
                          DR-CAB,DR-IDT,DR-LTI,DR-OBSERVE,
                          DR-OPE,DR-STR,DR-CL,DR-AE,DR-RT,
                          DR-LE,DR-LSN,DR-IAI: 'BOOLEEN' 'LECT'
                          'FIN';
                 CODEDR:(AB,CIRCUIT-Y,CIRCUIT-MF,GSM,....) 'LECT';
                 TTX-DR:'ENTIER'  'EB'(5) 'BIN' 'LECT';
                 URDE1:'ENTIER' 'MOT'(1) 'BIN' 'LECT';
                 VTLRDE1:'ENTIER' 'EB'(6) 'BIN' 'LECT';
                 EQDE1:'ENTIER' 'EB'(10) 'BIN' 'LECT';
                 DISCRIDE1:'ENREGISTREMENT'
                           DE1-CAB,DE1-OBSERVE,DE1-T,DE1-ASB,
                           DE1-NEQ,DE1-DENT,DE1-STR,DE1-PBX,
                           DE1-DET,DE1-IAM,DE1-NOSO: 'BOOLEEN' 'LECT'
                           'FIN';
                 CODEDE1:(AB,CIRCUIT-Y,CIRCUIT-MF,GSM,....) 'LECT';
                 URDE2:'ENTIER' 'MOT'(1) 'BIN' 'LECT';
                 VTLRDE2:'ENTIER' 'EB'(6) 'BIN' 'LECT';
                 EQDE2:'ENTIER' 'EB'(10) 'BIN' 'LECT';
                 DISCRIDE2:'ENREGISTREMENT'
                           DE2-CAB,DE2-OBSERVE,DE2-T,DE2-ASB,DE2-NEQ,DE2-DENT,DE2-STR,
                           DE2-PBX,DE2-DET,DE2-IAM,DE2-NOSO:'BOOLEEN' 'LECT'
                           'FIN';
                 CODEDE2:(AB,CIRCUIT-Y,CIRCUIT-MF,GSM,....) 'LECT';
                 IND-REP-URDE1,
                 IND-REP-URDE2:(LIBRE,OCC-CONVERS,INDISPONIBLE,EN-FAUTE,..)'LECT';
                 CHIF-RECU:'TABLEAU'(1..16) 'ENTIER' 'EB'(4) 'DCB';
                 NB-CHIF-CIRCUIT-Y,
```

```
                NB-CHIF-MF:'ENTIER' 'EB'(4) 'BIN' 'LECT';
                PREMIER-UR,
                DERNIER-UR:'ENTIER' 'MOT'(1) 'BIN' 'LECT';
                CHIF-CCF:'ENTIER'  'EB'(4) 'DCB' 'LECT';
                ETA-DR,
                ETA-DE,
                ETA-CCF:'ENTIER' 'MOT'(1) 'BIN' 'LECT';
                FIN-ETABLI:(EFFICACE,INEFFICACE,DE-IMPOSSIBLE,BLOCAGE-INTERNE,.);
                UNITE:(URDR,URDE1,URDE2,TR,CG,....);
                IND-RACCRO-DE1,
                IND-RACCRO-DE2,
                IND-RACCRO-DR:'BOOLEEN';
                FIN-RUPT:(EFFICACE,INEFFICACE);
                N,M,NCH-RECU,
                NCH-ATTENDU,
                NCH-A-EMETTRE,
                NCH-A ENVOYER:'ENTIER' 'EB'(4) 'BIN';
                UNITE-DEMENDEUR-1:'ENTIER' 'MOT'(1) 'BIN';
                CHIF-RETOUR-DE1:'ENTIER' 'EB'(4) 'DCB';
                PREMIER-DE-AB:'BOOLEEN';
                COMPTE,DUREE-CONVERS,
                DUREE-ETABLI,DUREE-SONNERIE:'ENTIER' 'MOT'(2) 'BIN';
                HEURE-DEBUT,
                HEURE-DEBUT-CHRONO,
                NUMERO-SUPPLEMENT:'ENTIER' 'MOT'(3) 'DCB';
                NB-IMP-A- ENVOYER:'ENTIER' 'MOT'(1) 'BIN';
                CODE-SERVICE-SPECIAUX:'ENTIER' 'MOT'(1) 'DCB'
                'FIN';

/*DECLARATION DES VARIABLES */

'VAR'  P:'PTR' ENREG;
```

```
/*DECLARATION DES EXCEPTIONS */

/*ICI COMMENCENT LES DECLARATIONS DES EXCEPTIONS UTILISEES DANS  LA COMM LOCALE,DEPART-Y OU ARRIVEE-MF */

'EXCEPTION' ETABLI-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR;
'APPEL' TEST-RACCRO-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' PRESEL-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR;
'APPEL' LIB-DCNX-RNC;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NUM-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR;
'SI' DR-CLAVIER 'ALORS' 'DEBUT'
                        'APPEL' LIBERATION-ETA(LIB-RNC);
                        'APPEL' DCNX-RNC
                        'FIN'
                        'FSI'
'APPEL' TEST-RACCRO-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NUM-INTERDIT;
'DEBUT'
FIN-ETABLI:=MANOEUVRE-INTERDIT;
'SI' DR-CLAVIER 'ALORS' 'APPEL' LIB-DCNX-RNC
                'SINON' 'APPEL' OCC-DR
                'FSI';
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' LIB-RNC-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR;
'APPEL' DCNX-RNC;
'APPEL' TEST-RACCRO-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' FAUX-DE1;
```

```
'DEBUT'
FIN-ETABLI:=FAUX-NUMERO;
'APPEL' OCC-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' DE1-OCC;
'DEBUT'
FIN-ETABLI:=OCCUPATION-DE1;
APPEL-LIB-OCC-DE1:=APPEL-LIB-OCC-DE1+1;
'APPEL' OCC-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' DE1-IMPOSSIBLE;
'DEBUT'
FIN-ETABLI:=BLOCAGE-INTERNE;
'APPEL' OCC-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' CIRCUIT-IMPOSSIBLE;
'DEBUT'
FIN-ETABLI:=PAS-CIRCUIT;
'APPEL' OCC-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' CNX-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-APRES-DE1;
'SI' CODEDE1=AB 'ALORS' 'APPEL' LIB-DE1-AB 'FSI';
'SI' CODEDE1=CIRCUIT-Y 'ALORS' 'APPEL' LIB-DE1-CIRCUIT-Y 'FSI';
'APPEL' TEST-RACCRO-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-DECRO-DE1-AB;
'DEBUT'
'QUAND' ECHEANCE 'ALORS' FIN-ETABLI:=NON-DECRO-DE1
                 'SINON' 'APPEL' IDENTIFIER-ERREUR-APRES-DE1 'FINQUAND';
'APPEL' DCNX-DR-DE1;
'APPEL' LIB-DE1-AB;
'APPEL' TEST-RACCRO-DR;
'ENVOI'(CG,FIN-ETABLI-DE1);
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'
```

```
'EXCEPTION' APPEL-DE1-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-APRES-DE1;
'APPEL' LIB-DCNX-DE1-CIRCUIT-Y;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-RECU-IT;
'DEBUT'
'QUAND' ECHEANCE 'ALORS' FIN-ETABLI:=BLOCAGE-EXTERNE
                 'SINON' 'APPEL' IDENTIFIER-ERREUR-APRES-DE1 'FINQUAND';
'APPEL' LIB-DCNX-DE1-CIRCUIT-Y;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-DECRO-DE1-CIRCUIT-Y;
'DEBUT'
'QUAND' ECHEANCE 'ALORS' FIN-ETABLI:=NON-DECRO-DE1
                 'SINON' 'APPEL' IDENTIFIER-ERREUR-APRES-DE1 'FINQUAND';
'APPEL' LIB-DCNX-DE1-CIRCUIT-Y;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

/*ICI LA FIN DES DECLARATIONS DES EXCEPTIONS UTILISEES DANS LE   BLOC NOUVEL-APPEL TRAITANT LA COMM ENTRE UN DR
AB ET UN DE AB OU CIRCUIT-Y*/
```

```
/*ICI LE DEBUT DES DECLARATIONS DES EXCEPTIONS UTILISEES DANS LA COMM ARRIVEE-MF */

'EXCEPTION' PRESEL-MF-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR;
'APPEL' TEST-RACCRO-DR-MF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' ETABLI-MF-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR;
'APPEL' LIB-DCNX-RGMF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' FAUX-DE1-MF;
'DEBUT'
FIN-ETABLI:=FAUX-NUMERO;
'APPEL' LIB-DCNX-RGMF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' DE1-OCC-MF;
'DEBUT'
FIN-ETABLI:=OCCUPATION-DE1;
APPEL-ARRIVEE-LIB-OCC-DE1:=APPEL-ARRIVEE-LIB-OCC-DE1+1;
'APPEL' LIB-DCNX-RGMF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' DE1-IMPOSSIBLE-MF;
'DEBUT'
FIN-ETABLI:=BLOCAGE-INTERNE;
'APPEL' LIB-DCNX-RGMF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NUM-SELEC-MAL-DEROULE;
'DEBUT'
FIN-ETABLI:=INEFFICACE;
'APPEL' LIB-DCNX-RGMF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' LIB-RGMF-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-APRES-DE1;
```

```
'APPEL' DCNX-RGMF;
'APPEL' TEST-RACCRO-DR-MF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' DCNX-RGMF-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-APRES-DE1;
'APPEL' TEST-RACCRO-DR-MF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-DECRO-DE1-AB-MF;
'DEBUT'
'QUAND' ECHEANCE 'ALORS' FIN-ETABLI:=NON-DECRO-DE1
                 'SINON' 'APPEL' IDENTIFIER-ERREUR-APRES-DE1 'FINQUAND';
'APPEL' LIB-DE1-AB;
'APPEL' TEST-RACCRO-DR-MF;
'ENVOI'(CG,FIN-ETABLI-DE1);
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' ETABLI-MF-MAL-FINI;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-APRES-DE1;
'SI' ¬IND-RACCRO-DE1 'ALORS' 'APPEL' OCC-DE1 'FSI';
'APPEL' TEST-RACCRO-DR-MF;
'ENVOI'(CG,FIN-ETABLI-DE1);
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

/* FIN DES DECLARATIONS DES EXCEPTIONS UTILISEES DANS LA COMM    ARRIVEE-MF */
```

```
/* DECLARATION DES EXCEPTIONS UTILISEES DANS LE BLOC TRAITANT LE SERVICE SPECIAL CONFERENCE*/

'EXCEPTION' PRESELEC-CCF-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR;
'APPEL' LIBERATION-ETA(LIB-RNC-CCF);
'APPEL' DCNX-RNC-CCF;
'APPEL' TEST-RACCRO-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' ETABLI-CCF-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-CCF;
'APPEL' LIBERATION-OCCUPATION-CCF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' CNX-MAL-DEROULE-CCF;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-CCF-APRES-DE2;
'SI' CODEDE2=AB 'ALORS' 'APPEL' LIB-DE2-AB 'FSI';
'SI' CODEDE2=CIRCUIT-Y 'ALORS' 'APPEL' LIB-DE2-CIRCUIT-Y 'FSI';
'APPEL' TEST-RACCRO-DR-DE1;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-DECRO-DE2-AB-CCF;
'DEBUT'
'QUAND' ECHEANCE 'ALORS' FIN-ETABLI:=NON-DECRO-DE2
                 'SINON' 'APPEL' IDENTIFIER-ERREUR-CCF-APRES-DE2 'FINQUAND';
'APPEL' LIB-DE2-AB;
'APPEL' TEST-RACCRO-DR-DE1;
'ENVOI'(CG,FIN-ETABLI-DE2);
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' APPEL-DE2-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-CCF-APRES-DE2;
'APPEL' LIBERATION-OCCUPATION-CCF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-RECU-IT-CCF;
'DEBUT'
'QUAND' ECHEANCE 'ALORS' FIN-ETABLI:=BLOCAGE-EXTERNE
                 'SINON' 'APPEL' IDENTIFIER-ERREUR-CCF-APRES-DE2 'FINQUAND';
```

```
'APPEL' LIBERATION-OCCUPATION-CCF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-DECRO-DE2-CIRCUIT-Y-CCF;
'DEBUT'
'QUAND' ECHEANCE 'ALORS' FIN-ETABLI:=NON-DECRO-DE2
                 'SINON' 'APPEL' IDENTIFIER-ERREUR-CCF-APRES-DE2 'FINQUAND';
'APPEL' LIBERATION-OCCUPATION-CCF;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' CHOIX-DR-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-CCF-DE1-DE2;
'APPEL' LIBERATION-ETA(LIB-RNC-CCF);
'APPEL' DCNX-RNC-CCF;
'APPEL' TEST-RACCRO-DR-DE1-DE2;
'ENVOI'(CG,FIN-ETABLI-DE2);
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' CCF-MAL-FINI;
'DEBUT'
FIN-ETABLI:=FAUX-MANOEUVRE;
'APPEL' LIBERATION-ETA(LIB-RNC-CCF);
'APPEL' DCNX-RNC-CCF;
'APPEL' OCC-DR-DE1-DE2;
'ENVOI'(CG,FIN-ETABLI-DE2);
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'
```

```
/*DECLARATION DES EXCEPTIONS PROVOQUEES DANS LES PROCEDURES DU   PROGRAMME TRAITANT LA CONFERENCE */

'EXCEPTION' CCF-TROIS-AB-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-CCF-DE1-DE2;
'APPEL' DCNX-RNC-CCF;
'APPEL' TEST-RACCRO-DR-DE1-DE2;
'ENVOI'(CG,FIN-ETABLI-DE2);
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' DCNX-RNC-CCF-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-CCF-DE1-DE2;
'APPEL' TEST-RACCRO-DR-DE1-DE2;
'ENVOI'(CG,FIN-ETABLI-DE2);
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' RETOUR-DE1-MAL-DEROULE;
'DEBUT'
'APPEL' LIBERATION-ETA(LIB-RNC-CCF);
'APPEL' DCNX-RNC-CCF;
'APPEL' TEST-RACCRO-DR-DE1;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' OCC-DR-OU-DE1-CCF;
'DEBUT'
'SELON' CODEDE1 'FAIRE'
AB:'SI' IND-RACCRO-DE1 'ALORS' 'DEBUT'
                          'APPEL' LIB-DE1-AB;
                          'SI' ~IND-RACCRO-DR 'ALORS' 'APPEL' OCC-DR
                          'FIN'
                  'SINON' 'APPEL' OCC-DE1
                  'FSI'
CIRCUIT-Y:'DEBUT'
          'APPEL' LIB-DE1-CIRCUIT-Y;
          'SI' ~IND-RACCRO-DR 'ALORS'  'APPEL' OCC-DR 'FSI'
          'FIN'
'FINSELON';
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' FAUX-NUM-RETOUR-DE1;
'DEBUT'
'APPEL' OCC-DR-DE1;
'ENVOI'(CG,FIN-ETABLI-DR)
```

```
'FIN'

'EXCEPTION' RETOUR-DE1-MAL-FINI;
'DEBUT'
'APPEL' TEST-RACCRO-DR-DE1;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' OBS-RETOUR-DE1;
'DEBUT'
'SI' DR-OBSERVE 'ALORS' 'APPEL' OBS-FIN-ETABLI(DR) 'FSI';
'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-FIN-ETABLI(DE1) 'FSI';
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'
```

```
/* DEBUT DES DECLARATIONS DES EXCEPTIONS FIGURANT DANS LE CAS DE RUPTURE LOCALE,DEPART-Y OU ARRIVEE-MF */

'EXCEPTION' RUPT-DR-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-RUPT-DR;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-TONAL-OCC-DE1;
'DEBUT'
'QUAND' RACCRODE1 'ALORS' FIN-RUPT:=EFFICACE
                  'SINON' 'APPEL'  IDENTIFIER-ERREUR-RUPT-DR 'FINQUAND';
'APPEL' LIB-DE1-AB;
'ENVOI'(RCX,STOP-TONAL-OCC-DE1);
'RECEVOIR'(RCX,|COMPTERENDU|);
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' RUPT-DE1-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-RUPT-DE1;
'APPEL' LIB-DE1;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-REPRISE-COMM;
'DEBUT'
'QUAND' RACCRODR 'ALORS' FIN-RUPT:=EFFICACE
                 'SINON' 'DEBUT'
                         APPEL' IDENTIFIER-ERREUR-RUPT-DE1;
                         'APPEL' OCC-DR
                         'FIN'
                         'FINQUAND';
'APPEL' DCNX-DR-DE1;
'APPEL' LIB-DE1;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' RUPT-DE1-MF-MAL-DEROULE;
'DEBUT'
'APPEL' IDENTIFIER-ERREUR-RUPT-DE1;
'APPEL' LIB-DE1-AB;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

'EXCEPTION' NON-REPRISE-COMM-MF;
'DEBUT'
'SUIVANT' CODE-FONCTION 'FAIRE'
```

```
ECHEANCE:'DEBUT'
         'ENVOI'(URDR,EMET-ALTERNES-2-ATTENT-IMP-LONGUE);
         'AVANT' 5 'SEC' 'ATTENDRE'(1,(URDR,|RACCRODR|));
         'QUANDNON' RACCRODR 'FAIRE' 'APPEL' IDENTIFIER-ERREUR-RUPT-DE1
         'FIN'
RACCRODR:
'AUTRE'  'APPEL' IDENTIFIER-ERREUR-RUPT-DE1
'FINSUIVANT';
'APPEL' DCNX-DR-DE1;
'APPEL' LIB-DE1-AB;
'ENVOI'(CG,FIN-ETABLI-DR)
'FIN'

/*FIN DE DECLARATION DES EXCEPTIONS FIGURANT DANS LE CAS DE RUPT LOCALE,DEPART-Y OU ARRIVEE-MF*/
```

```
/*DECLARATION DES PROCEDURES UTILISEES DANS LE PROGRAMME DE LA  COMM LOCALE ET DEPART DECIMALE*/

'PROCEDURE' RECEPTION-CHIF;
'DEBUT'
'TANTQUE' N>=4 'FAIRE'
'DEBUT'
NCHATTENDU:=4;
'ENVOI'(UNITE,RECEV-N-CHIF);
'AVANT' 64 'SEC' 'ATTENDRE'(1,(UNITE,|N-CHIF-RECU|));
'QUANDNON' N-CHIF-RECU 'FAIRE' 'PROVOQUER' NUM-MAL-DEROULE;
NCHRECU:=NCHRECU+4;
N:=N-4;
'FIN';
'SI' N >0 'ALORS' 'DEBUT'
                 NCHATTENDU:=N;
                 'ENVOI'(UNITE,RECEV-N-CHIF);
                 M:=N*16;
                 'AVANT' M 'SEC' 'ATTENDRE'(1,(UNITE,|N-CHIF-RECU|));
                 'QUANDNON' N-CHIF-RECU 'FAIRE' 'PROVOQUER' NUM-MAL-DEROULE;
                 NCHRECU:=NCHRECU+N
                 'FIN'
                 'FSI'
'FIN'

'PROCEDURE' INTERRO-URDE1(MESG-A-ENVOYER:CODE-FONCTION);
'DEBUT'
'ENVOI'(URDE1,MESG-A-ENVOYER);
'RECEVOIR'(URDE1,|REP-POSIT-URDE1|);
'QUANDNON' REP-POSIT-URDE1 'FAIRE' 'PROVOQUER' ETABLI-MAL-DEROULE
'FIN'

'PROCEDURE' INTERRO-TOUTES-URDE1;
'DEBUT'
URDE1:=PREMIER-UR;
'REPETER'
'DEBUT'
'APPEL' INTERRO-URDE1(TEST-FAISCEAU);
URDE1:=URDE1+1
'FIN'
'JUSQUA' (IND-REP-URDE1 /= TRANCON-EPUISE) 'ET' (URDE1 > DERNIER-UR);
URDE1:=URDE1-1
'FIN'

'PROCEDURE' DETOURNEMENT;
'DEBUT'
'ENVOI'(TR,DETOURNEMENT);
'RECEVOIR'(TR,|PF,GF,REP-NEG-TR|);
'SUIVANT' CODE-FONCTION 'FAIRE'
```

```
PF:'APPEL' INTERRO-URDE1(TEST-FAISCEAU)
GF:'APPEL' INTERRO-TOUTES-URDE1
REP-NEG-TR:'PROVOQUER' CIRCUIT-IMPOSSIBLE
'AUTRE'  'PROVOQUER' ETABLI-MAL-DEROULE
'FINSUIVANT';

/*ON VIENT A CE POINT QUAND LA REP DU TR EST EGALE A PF OU A GF*/

'SELON' IND-REP-URDE1 'FAIRE'
LIBRE:
OCC-CONVERS:'APPEL' AE-IAI
INDISPONIBLE,
PAS-VTLR,
EN-FAUTE:'PROVOQUER'  DE-IMPOSSIBLE
OCC:'PROVOQUER' DE-OCC
TRANCON-EPUISE:'PROVOQUER' CIRCUIT-IMPOSSIBLE
'FINSELON'
'FIN'

'PROCEDURE' TEST-REP-URDE1-DEB;
'DEBUT'
'SELON' IND-REP-URDE1 'FAIRE'
LIBRE:
OCC-CONVERS:'APPEL' AE-IAI
OCC:'PROVOQUER' DE-OCC
INDISPONIBLE,
PAS-VTLR,
EN-FAUTE:'PROVOQUER' DE-IMPOSSIBLE
TRANCON-EPUISE:'APPEL' DETOURNEMENT
'FINSELON'
'FIN'
```

```
/*DECLARATION DES PROCEDURES APPELEES PAR LES EXCEPTIONS*/

'PROCEDURE' IDENTIFIER-ERREUR;
'DEBUT'
'SUIVANT' CODE-FONCTION 'FAIRE'
REP-RESERV-NEG: FIN-ETABLI:=MANQ-AUXI
RACCRODR:'DEBUT'
         RACCRO-PREM:=RACCRO-PREM+1;
         RACCRO-AVANT-TEST-DE:=RACCRO-AVANT-TEST-DE+1;
         FIN-ETABLI:=RACCRO-PREM-DR
         'FIN'
CPT-RENDU-INCIDENT: FIN-ETABLI:=BLOCAGE-INTERNE
'AUTRE'    FIN-ETABLI:=INEFFICACE
'FINSUIVANT'
'FIN'

'PROCEDURE' TEST-RACCRO-DR;
'DEBUT'
'SI' ^IND-RACCRO-DR 'ALORS' 'APPEL' OCC-DR 'FSI'
'FIN'

'PROCEDURE' OCC-DR;
'DEBUT'
'ENVOI'(RCX,TONAL-OCC-DR);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUAND' COMPTERENDU 'ALORS' 'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDR,  |RACCRODR|)) 'FINQUAND';
'QUANDNON' RACCRODR 'FAIRE'  'APPEL' LIB-DR-AB;
'ENVOI'(RCX,STOP-TONAL-OCC-DR);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE' LIB-DR-AB;
'DEBUT'
'ENVOI'(URDR,LIB-DR)
'FIN'

'PROCEDURE' LIB-DCNX-RNC;
'DEBUT'
'APPEL' LIBERATION-ETA(LIB-RNC);
'APPEL' DCNX-RNC;
'APPEL' TEST-RACCRO-DR
'FIN'

'PROCEDURE'  LIBERATION-ETA(MESG-A-ENVOYER:CODE-FONCTION);
'DEBUT'
'ENVOI'(ETA,MESG-A-ENVOYER);
'RECEVOIR'(ETA,|ACCUSE-RESTITU|)
'FIN'
```

```
'PROCEDURE' DCNX-RNC;
'DEBUT'
'ENVOI'(RCX,DCNX-RNC);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE'  LIB-DE1-CIRCUIT-Y;
'DEBUT'
'ENVOI'(URDE1,EMET-IMP-LONGUE-LIB-EQ)
'FIN'

'PROCEDURE' LIB-DE1-AB;
'DEBUT'
'ENVOI'(URDE1,LIB-DE1)
'FIN'

'PROCEDURE' DCNX-DR-DE1;
'DEBUT'
'ENVOI'(RCX,DECONNEXION-DR-DE1);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE' LIB-DCNX-DE1-CIRCUIT-Y;
'DEBUT'
'APPEL' DCNX-DR-DE1;
'APPEL' LIB-DE1-CIRCUIT-Y;
'APPEL' TEST-RACCRO-DR;
'ENVOI' (CG,FIN-ETABLI-DE1)
'FIN'

'PROCEDURE' IDENTIFIER-ERREUR-APRES-DE1;
'DEBUT'
'SUIVANT' CODE-FONCTION 'FAIRE'
RACCRODR:'DEBUT'
         RACCRO-DR-PREM:=RACCRO-DR-PREM+1;
         FIN-ETABLI:=RACCRO-PREM-DR
         'FIN'
CPT-RENDU-INCIDENT:FIN-ETABLI:=BLOCAGE-INTERNE
'AUTRE'    FIN-ETABLI:=INEFFICACE
'FINSUIVANT'
'FIN'

'PROCEDURE' TEST-RACCRO-DR-MF;
'DEBUT'
'SI' ¬IND-RACCRO-DR 'ALORS' 'APPEL' LIB-DR-MF 'FSI'
'FIN'
```

```
'PROCEDURE' LIB-DR-MF;
'DEBUT'
'ENVOI'(URDR,EMET-IMP-LONGUE-LIB-EQ)
'FIN'

'PROCEDURE' LIB-DCNX-RGMF;
'DEBUT'
'APPEL' LIBERATION-ETA(LIB-RGMF);
'APPEL' DCNX-RGMF;
'APPEL' TEST-RACCRO-DR-MF
'FIN'

'PROCEDURE' DCNX-RGMF;
'DEBUT'
'ENVOI'(RCX,DECONNECTER-RGMF);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE' INTERRO-URDE1-MF(MESG-A-ENVOYER:CODE-FONCTION);
'DEBUT'
'ENVOI'(URDE1,MESG-A-ENVOYER);
'RECEVOIR'(URDE1,|REP-POSIT-URDE1|);
'QUANDNON' REP-POSIT-URDE1 'FAIRE' 'PROVOQUER' ETABLI-MF-MAL-DEROULE
'FIN'
```

```
/*DECLARATION DES PROCEDURES UTILISEES DANS LE PROGRAMME TRAITANT LE SERVICE SPECIAL CONFERENCE*/

'PROCEDURE' RECEPTION-CHIF-CCF;
'DEBUT'
'TANTQUE' N >=4 'FAIRE'
'DEBUT'
NCHATTENDU:=4;
'ENVOI'(RNC,RECEV-N-CHIF);
'AVANT' 64 'SEC' 'ATTENDRE'(1,(RNC,|N-CHIF-RECU|));
'QUANDNON' N-CHIF-RECU 'FAIRE' 'PROVOQUER' ETABLI-CCF-MAL-DEROULE;
NCHRECU:=NCH-RECU+4;
N:=N-4
'FIN';
'SI' N >0 'ALORS' 'DEBUT'
                 NCHATTENDU:=N;
                 'ENVOI'(RNC,RECEV-N-CHIF);
                 M:=N*16;
                 'AVANT' M 'SEC' 'ATTENDRE'(1,(RNC,|N-CHIF-RECU|));
                 'QUANDNON' N-CHIF-RECU 'FAIRE' 'PROVOQUER' ETABLI-CCF-MAL-DEROULE;
                 NCHRECU:=NCHRECU+N
                 'FIN'
                 'FSI'
'FIN'

'PROCEDURE' INTERRO-URDE2-CCF(MESG-A-ENVOYER:CODE-FONCTION);
'DEBUT'
'ENVOI'(URDE2,MESG-A-ENVOYER);
'RECEVOIR'(URDE2,|REP-POSIT-URDE2|);
'QUANDNON' REP-POSIT-URDE2 'FAIRE' 'PROVOQUER' ETABLI-CCF-MAL-DEROULE
'FIN'

'PROCEDURE' INTERRO-TOUTES-URDE2-CCF;
'DEBUT'
URDE2:=PREMIER-UR;
'REPETER'
'DEBUT'
'APPEL' INTERRO-URDE2(TEST-FAISCEAU);
URDE2:=URDE2+1
'FIN'
'JUSQUA'(IND-REP-URDE2 /= TRANCON-EPUISE)'OU'(URDE2 > DERNIER-UR)
URDE2:=URDE2-1
'FIN'

'PROCEDURE' FAUX-MANOEUVRE-CCF;
'DEBUT'
FIN-ETABLI:=MANOEUVRE-INTERDIT;
'APPEL' RETOUR-DE1
'FIN'
```

```
'PROCEDURE' FAUX-DE2-CCF;
'DEBUT'
FIN-ETABLI:=FAUX-NUMERO;
'APPEL' RETOUR-DE1
'FIN'

'PROCEDURE' DE2-OCC-CCF;
'DEBUT'
FIN-ETABLI:=OCCUPATION-DE2;
'APPEL' RETOUR-DE1
'FIN'

'PROCEDURE' DE2-IMPOSSIBLE-CCF;
'DEBUT'
FIN-ETABLI:=BLOCAGE-INTERNE;
'APPEL' RETOUR-DE1;
'FIN'

'PROCEDURE' TEST-REP-URDE2-DEB-CCF;
'DEBUT'
'SELON' IND-REP-URDE2 'FAIRE'
LIBRE :
OCC-CONVERS : 'APPEL' AE-IAI
OCC : 'APPEL' DE2-OCC-CCF
INDISPONIBLE,
PAS-VTIR,
EN-FAUTE : 'APPEL' DE2-IMPOSSIBLE-CCF
TRANCON-EPUISE : 'APPEL' DETOURNEMENT-CCF
'FINSELON'
'FIN'

'PROCEDURE' DETOURNEMENT-CCF;
'DEBUT'
'ENVOI'(TR,DETOURNEMENT);
'RECEVOIR'(TR,|PF,GF,REP-NEG-TR|);
'SUIVANT' CODE-FONCTION 'FAIRE'
PF: 'APPEL' INTERRO-URDE2-CCF(TEST-FAISCEAU)
GF: 'APPEL' INTERRO-TOUTES-URDE2
REP-NEG-TR: 'APPEL' NON-CIRCUIT-CCF
'AUTRE'   'PROVOQUER' ETABLI-CCF-MAL-DEROULE
'FINSUIVANT';
'SELON' IND-REP-URDE2 'FAIRE'
LIBRE :
OCC-CONVERS: 'APPEL' AE-IAI
OCC: 'APPEL' DE2-OCC-CCF
INDISPONIBLE,
PAS-VTIR,
```

```
EN-FAUTE :'APPEL' DE2-IMPOSSIBLE-CCF
TRANCON-EPUISE: 'APPEL' NON-CIRCUIT-CCF
'FINSELON'
'FIN'

'PROCEDURE' CHOIX-DR;
'DEBUT'
'QUANDNON' BR1 'FAIRE' 'PROVOQUER' CHOIX-DR-MAL-DEROULE;
'ENVOI'(RCX,TONAL-IN-DR);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' CHOIX-DR-MAL-DEROULE;
NCHATTENDU:=1;
'ENVOI'(RNC,RECEV-N-CHIF);
'AVANT' 16 'SEC' 'ATTENDRE'(1,(RNC,|N-CHIF-RECU|));
'QUANDNON' N-CHIF-RECU 'FAIRE' 'PROVOQUER' CHOIX-DR-MAL-DEROULE;
'ENVOI'(RCX,STOP-TONAL-IN-DR);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' CHOIX-DR-MAL-DEROULE
'FIN'

'PROCEDURE' CCF-TROIS-AB;
'DEBUT'
'ENVOI'(RCX,ETABLI-CCF);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' CHOIX-DR-MAL-DEROULE;
'APPEL' REPRISE-TAXATION(DR-DE1);
'APPEL' DEBUTER-TAXATION(DR-DE2);
'ENVOI'(URDR,SURV2);
'ENVOI'(URDE1,SURV2);
'ENVOI'(URDE2,SURV2);
'APPEL' LIBERATION-ETA(LIB-RNC);
'RECEVOIR'(ETA,|ACCUSE-RESTITU|);
'QUANDNON' ACCUSE-RESTITU 'FAIRE' 'PROVOQUER' CCF-TROIS-AB-MAL-DEROULE;
'ENVOI'(RCX,DCNX-RNC);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' DCNX-RNC-CCF-MAL-DEROULE;
'FIN'
```

```
/*DECLARATION DES PROCEDURES APPELEES PAR LES EXCEPTIONS ET LES PROCEDURES DU PROGRAMME TRAITANT LA CONFERENCE */

'PROCEDURE' RETOUR-DE1;
'DEBUT'
'ENVOI'(RCX,OCC-DR-CCF);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' RETOUR-DE1-MAL-DEROULE;
'AVANT' 16 'SEC' 'ATTENDRE'(1,(URDR,|BR1-DR|));
'QUANDNON' BR1-DR 'FAIRE' 'PROVOQUER' RETOUR-DE1-MAL-DEROULE;
'ENVOI'(RCX,TONAL-IN);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' RETOUR-DE1-MAL-DEROULE;
NCHATTENDU:=1;
'ENVOI'(RNC,RECEV-N-CHIF);
'AVANT' 16 'SEC' 'ATTENDRE'(1,(RNC,|N-CHIF-RECU|));
'QUANDNON' N-CHIF-RECU 'FAIRE' 'PROVOQUER' RETOUR-DE1-MAL-DEROULE;
'ENVOI'(RCX,STOP-TONAL-IN);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' RETOUR-DE1-MAL-DEROULE;
'APPEL' LIBERATION-ETA(LIB-RNC-CCF);
'APPEL' DCNX-RNC-CCF;
'SI' (~IND-RACCRO-DR) 'ET' (~IND-RACCRO-DE1) 'ALORS'
                                              'SINON' 'PROVOQUER' OCC-DR-OU-DE1-CCF 'FSI';
'SI' CHIF-RETOUR-DE1 /= 1 'ALORS' 'PROVOQUER' FAUX-NUM-RETOUR-DE1 'FSI' ;
'ENVOI'(RCX,CNX-DR-DE1);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' RETOUR-DE1-MAL-FINI;
'ENVOI'(URDR,SURV2);
'ENVOI'(URDE1,SURV2);
'PROVOQUER' OBS-RETOUR-DE1
'FIN'

'PROCEDURE' NON-CIRCUIT-CCF;
'DEBUT'
FIN-ETABLI:=PAS-CIRCUIT;
'APPEL' RETOUR-DE1
'FIN'

'PROCEDURE' DCNX-RNC-CCF;
'DEBUT'
'ENVOI'(RCX,DECONNEXION-RNC-CCF);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE' IDENTIFIER-ERREUR-CCF;
'DEBUT'
'SUIVANT' CODE-FONCTION 'FAIRE'
REP-NEG-RESERV:FIN-ETABLI:=MANQ-AUXI
```

```
RACCRODR:'DEBUT'
         FIN-ETABLI:=RACCRO-PREM-DR;
         RACCRO-PREM:=RACCRO-PREM+1;
         RACCRO-AVANT-TEST-DE:=RACCRO-AVANT-TEST-DE+1
         'FIN'
RACCRODE1:'DEBUT'
          FIN-ETABLI:=RACCRO-PREM-DE1;
          RACCRO-PREM:=RACCRO-PREM+1;
          RACCRO-AVANT-TEST-DE:=RACCRO-AVANT-TEST-DE+1
          'FIN'
CPT-RENDU-INCIDENT: FIN-ETABLI:=BLOCAGE-INTERNE
'AUTRE'   FIN-ETABLI:=INEFFICACE
'FINSUIVANT'
'FIN'

'PROCEDURE' LIBERATION-OCCUPATION-CCF;
'DEBUT'
'APPEL' LIBERATION-ETA(LIB-RNC-CCF);
'APPEL' DCNX-RNC-CCF;
'APPEL' TEST-RACCRO-DR-DE1
'FIN'

'PROCEDURE' TEST-RACCRO-DR-DE1;
'DEBUT'
'SELON' CODEDE1 'FAIRE'
AB:'DEBUT'
   'SI' IND-RACCRO-DE1 'ALORS' 'DEBUT'
                               'APPEL' LIB-DE1-AB;
                               'SI' ~IND-RACCRO-DR 'ALORS' 'APPEL' OCC-DR
                               'FIN'
                       'SINON' 'SI' ~IND-RACCRO-DR 'ALORS' 'APPEL' OCC-DR-DE1
                                                   'SINON' 'APPEL' OCC-DE1
                                                   'FSI'
                       'FSI'
   'FIN'
CIRCUIT-Y:'DEBUT'
          'SI' ~IND-RACCRO-DR 'ALORS' 'APPEL' OCC-DR 'FSI'
          'APPEL' LIB-DE1-CIRCUIT-Y
          'FIN'
'FINSELON'
'FIN'

'PROCEDURE' OCC-DE1;
'DEBUT'
'ENVOI'(RCX,TONAL-OCC-DE1);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUAND' COMPTERENDU 'ALORS' 'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE1,|RACCRODE1|)) 'FINQUAND';
'APPEL' LIB-DE1-AB;
```

```
'ENVOI'(RCX,STOP-TONAL-OCC-DE1);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE' OCC-DR-DE1;
'DEBUT'
'ENVOI'(RCX,OCC-DR-DE1);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUAND' COMPTERENDU 'ALORS' 'DEBUT'
                            'AVANT' 1 'MIN' 'ATTENDRE'(2,(URDR,|RACCRODR|),(URDE1,|RACCRODE1|));
                            'SUIVANT' CODE-FONCTION 'FAIRE'
                            RACCRODR:'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE1,|RACCRODE1|))
                            RACCRODE1:'DEBUT'
                                      'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDR,|RACCRODR|));
                                      'QUANDNON' RACCRODR 'FAIRE' 'APPEL' LIB-DR-AB
                                      'FIN'
                            'AUTRE'       'APPEL' LIB-DR-AB
                            'FINSUIVANT'
                            'FIN'
                    'SINON' 'QUANDNON' RACCRODR 'ALORS' 'APPEL' LIB-DR-AB 'FINQUAND';
'APPEL' LIB-DE1-AB;
'ENVOI'(RCX,STOP-TONAL-OCC-DR-DE1);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE' IDENTIFIER-ERREUR-CCF-APRES-DE2;
'DEBUT'
'SUIVANT' CODE-FONCTION 'FAIRE'
RACCRODR:'DEBUT'
         FIN-ETABLI:=RACCRO-PREM-DR;
         RACCRO-PREM:=RACCRO-PREM+1
         'FIN'
RACCRODE1:'DEBUT'
          FIN-ETABLI:=RACCRO-PREM-DE1;
          RACCRO-PREM:=RACCRO-PREM+1
          'FIN'
CPT-RENDU-INCIDENT:FIN-ETABLI:=BLOCAGE-INTERNE
'AUTRE'         FIN-ETABLI:=INEFFICACE
'FINSUIVANT'
'FIN'

'PROCEDURE' LIB-DE2-AB;
'DEBUT'
'ENVOI'(URDE2,LIB-DE2)
'FIN'

'PROCEDURE' LIB-DE2-CIRCUIT-Y;
'DEBUT'
```

```
'ENVOI'(URDE2,EMET-IMP-LONGUE-LIB-EQ)
'FIN'

'PROCEDURE' IDENTIFIER-ERREUR-CCF-DE1-DE2;
'DEBUT'
'SUIVANT' CODE-FONCTION 'FAIRE'
RACCRODR:'DEBUT'
         FIN-ETABLI:=RACCRO-PREM-DR;
         RACCRO-PREM:=RACCRO-PREM+1
         'FIN'
RACCRODE1:'DEBUT'
          FIN-ETABLI:=RACCRO-PREM-DE1;
          RACCRO-PREM:=RACCRO-PREM+1
          'FIN'
RACCRODE2:'DEBUT'
          FIN-ETABLI:=RACCRO-PREM-DE2;
          RACCRO-PREM:=RACCRO-PREM+1
          'FIN'
CPT-RENDU-INCIDENT:FIN-ETABLI:=BLOCAGE-INTERNE
'AUTRE'     FIN-ETABLI:=INEFFICACE
'FINSUIVANT'
'FIN'

'PROCEDURE' TEST-RACCRO-DR-DE1-DE2;
'DEBUT'
'SELON' CODEDE1 'FAIRE'
AB:'SELON' CODEDE2 'FAIRE'
   AB:'SELON' IND-RACCRO-DR 'FAIRE'
      'VRAI':'SELON' IND-RACCRO-DE1 'FAIRE'
             'VRAI':'DEBUT'
                    'APPEL' LIB-DE1-AB;
                    'SELON' IND-RACCRO-DE2 'FAIRE'
                    'VRAI': 'APPEL' LIB-DE2-AB;
                    'FAUX': 'APPEL' OCC-DE2
                    'FINSELON'
                    'FIN'
             'FAUX':'SELON' IND-RACCRO-DE2 'FAIRE'
                    'VRAI':'DEBUT'
                           'APPEL' LIB-DE2-AB;
                           'APPEL' OCC-DE1
                           'FIN'
                    'FAUX':'APPEL' OCC-DE1-DE2
                    'FINSELON'
             'FINSELON'
      'FAUX':'SELON' IND-RACCRO-DE1 'FAIRE'
             'VRAI':'DEBUT'
                    'APPEL' LIB-DE1-AB;
                    'SELON' IND-RACCRO-DE2 'FAIRE'
```

```
                    'VRAI:'DEBUT'
                          'APPEL' LIB-DE2-AB;
                          'APPEL' OCC-DR
                          'FIN'
                    'FAUX':'APPEL' OCC-DR-DE2
                    'FINSELON'
                    'FIN'
              'FAUX':'SELON' IND-RACCRO-DE2 'FAIRE'
                    'VRAI':'DEBUT'
                          'APPEL' LIB-DE2-AB;
                          'APPEL' OCC-DR-DE1
                          'FIN'
                    'FAUX':'APPEL' OCC-DR-DE1-DE2
                    'FINSELON'
              'FINSELON'
        'FINSELON'
    CIRCUIT-Y:'DEBUT'
              'APPEL' LIB-DE2-CIRCUIT-Y;
              'SELON' IND-RACCRO-DR 'FAIRE'
              'VRAI':'SELON' IND-RACCRO-DE1 'FAIRE'
                    'VRAI':'APPEL' LIB-DE1-AB
                    'FAUX':'APPEL' OCC-DE1
                    'FINSELON'
              'FAUX':'SELON' IND-RACCRO-DE1 'FAIRE'
                    'VRAI':'DEBUT'
                          'APPEL' LIB-DE1-AB;
                          'APPEL' OCC-DR
                          'FIN'
                    'FAUX':'APPEL' OCC-DR-DE1
                    'FINSELON'
              'FINSELON'
              'FIN'
    'FINSELON'
CIRCUIT-Y:'DEBUT'
          'APPEL' LIB-DE1-CIRCUIT-Y;
          'SELON' IND-RACCRO-DR 'FAIRE'
          'VRAI':'SELON' IND-RACCRO-DE2 'FAIRE'
                'VRAI':'APPEL' LIB-DE2-AB
                'FAUX':'APPEL' OCC-DE2
                'FINSELON'
          'FAUX':'SELON' IND-RACCRO-DE2 'FAIRE'
                'VRAI':'DEBUT'
                      'APPEL' OCC-DR;
                      'APPEL' LIB-DE2-AB
                      'FIN'
                'FAUX':'APPEL' OCC-DR-DE2
                'FINSELON'
          'FINSELON'
```

```
          'FIN'
'FINSELON'
'FIN'

'PROCEDURE' OCC-DE2;
'DEBUT'
'ENVOI'(RCX,TONAL-OCC-DE2);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUAND' COMPTERENDU 'ALORS' 'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE2,|RACCRODE2|)) 'FINQUAND';
'APPEL' LIB-DE2-AB;
'ENVOI'(RCX,STOP-TONAL-OCC-DE2);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE'  OCC-DR-DE2;
'DEBUT'
'ENVOI'(RCX,OCC-DR-DE2);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUAND' COMPTERENDU 'ALORS' 'DEBUT'
                            'AVANT' 1 'MIN' 'ATTENDRE'(2,(URDR,|RACCRODR|),(URDE2,|RACCRODE2|));
                            'SUIVANT' CODE-FONCTION 'FAIRE'
                            RACCRODR:'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE2,|RACCRODE2|))
                            RACCRODE2:'DEBUT'
                                      'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDR,|RACCRODR|));
                                      'QUANDNON' RACCRODR 'ALORS' 'APPEL' LIB-DR-AB
                                      'FIN'
                            'AUTRE'     'APPEL' LIB-DR-AB
                            'FINSUIVANT'
                            'FIN'
                     'SINON' 'QUANDNON' RACCRODR 'ALORS' 'APPEL' LIB-DR-AB 'FINQUAND';
'APPEL' LIB-DE2-AB;
'ENVOI'(RCX,STOP-TONAL-OCC-DR-DE2);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE' OCC-DE1-DE2;
'DEBUT'
'ENVOI'(RCX,OCC-DE1-DE2);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUAND' COMPTERENDU 'ALORS' 'DEBUT'
                            'AVANT' 1 'MIN' 'ATTENDRE'(2,(URDE1,|RACCRODE1|),(URDE2,|RACCRODE2|));
                            'SUIVANT' CODE-FONCTION 'FAIRE'
                            RACCRODE1:'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE2,|RACCRODE2|))
                            RACCRODE2:'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE1,|RACCRODE1|))
                            'AUTRE'
                            'FINSUIVANT'
                            'FIN'
                            'FINQUAND'
```

```
'APPEL' LIB-DE1-AB;
'APPEL' LIB-DE2-AB;
'ENVOI'(RCX,STOP-TONAL-OCC-DE1-DE2);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'

'PROCEDURE' OCC-DR-DE1-DE2;
'DEBUT'
'ENVOI'(RCX,OCC-DR-DE1-DE2);
'RECEVOIR'(RCX,|COMPTERENDU|);
'QUAND' COMPTERENDU 'ALORS' 'DEBUT'
                            'AVANT' 1 'MIN' 'ATTENDRE'(3,(URDR,|RACCRODR|),(URDE1,|RACCRODE1|
                                                                          ),(URDE2,|RACCRODE2|));
                            'SUIVANT' CODE-FONCTION 'FAIRE'
                            RACCRODR: 'DEBUT'
                                      'AVANT' 1 'MIN' 'ATTENDRE'(2,(URDE1,|RACCRODE1|),(URDE2,|RACCRODE2|));
                                      'SUIVANT' CODE-FONCTION 'FAIRE'
                                      RACCRODE1:'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE2,|RACCRODE2|))
                                      RACCRODE2:'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE1,|RACCRODE1|))
                                      'AUTRE'
                                      'FINSUIVANT'
                                      'FIN'
                            RACCRODE1:'DEBUT'
                                      'AVANT' 1 'MIN' 'ATTENDRE'(2,(URDR,|RACCRODR|),(URDE2,|RACCRODE2|));
                                      'SUIVANT' CODE-FONCTION 'FAIRE'
                                      RACCRODR:'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE2,|RACCRODE2|))
                                      RACCRODE2:'DEBUT'
                                                'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDR,|RACCRODR|));
                                                'QUANDNON' RACCRODR 'ALORS' 'APPEL' LIB-DR-AB
                                                'FIN'
                                      'AUTRE'  'APPEL' LIB-DR-AB
                                      'FINSUIVANT'
                                      'FIN'
                            RACCRDE2:'DEBUT'
                                      'AVANT' 1 'MIN' 'ATTENDRE'(2,(URDR,|RACCRODR|),(URDE1,|RACCRODE1|));
                                      'SUIVANT' CODE-FONCTION 'FAIRE'
                                      RACCRODR:'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE1,|RACCRODE1|))
                                      RACCRODE1:'DEBUT'
                                                'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDR,|RACCRODR|));
                                                'QUANDNON' RACCRODR 'ALORS' 'APPEL' LIB-DR-AB
                                                'FIN'
                                      'AUTRE'   'APPEL' LIB-DR-AB
                                      'FINSUIVANT'
                                      'FIN'
                            'AUTRE'  'APPEL' LIB-DR-AB
                            'FINSUIVANT'
                            'FIN'
                  'SINON' 'QUANDNON' RACCRODR 'ALORS' 'APPEL' LIB-DR-AB 'FINQUAND';
```

```
'APPEL' LIB-DE1-AB;
'APPEL' LIB-DE2-AB;
'ENVOI'(RCX,STOP-TONAL-OCC-DR-DE1-DE2);
'RECEVOIR'(RCX,|COMPTERENDU|)
'FIN'
```

```
/*DEBUT DE DECLARATION DES PROCEDURES FIGURANT DANS LA RUPTURE LOCALE,DEPART-Y OU ARRIVEE-MF */

'PROCEDURE' IDENTIFIER-ERREUR-RUPT-DR;
'DEBUT'
'SUIVANT' CODE-FONCTION 'FAIRE'
CPT-RENDU-INCIDENT:FIN-RUPT:=BLOCAGE-INTERNE
REP-NEG-URDE1:FIN-RUPT:=EFFICACE
'AUTRE'  FIN-RUPT:=INEFFICACE
'FINSUIVANT'
'FIN'

'PROCEDURE' IDENTIFIER-ERREUR-RUPT-DE1;
'DEBUT'
'SUIVANT' CODE-FONCTION 'FAIRE'
REP-NEG-INTERRO-RCX:FIN-RUPT:=EFFICACE
CPT-RENDU-INCIDENT:FIN-RUPT:=BLOCAGE-INTERNE
REP-NEG-URDE1:FIN-RUPT:=EFFICACE
'AUTRE'  FIN-RUPT:=INEFFICACE
'FINSUIVANT'
'FIN'

'PROCEDURE' LIB-DE1;
'DEBUT'
'SI' CODEDE1=AB 'ALORS' 'APPEL' LIB-DE1-AB 'FSI';
'SI' CODEDE1=CIRCUIT-Y 'ALORS' 'APPEL' LIB-DE1-CIRCUIT-Y 'FSI'
'FIN'

/*FIN DE DECLARATION DES PROCEDURES FIGURANT DANS LA RUPTURE LOCALE , DEPART-Y , ARRIVEE-MF*/
```

```
/*DEBUT DE LA PARTIE INSTRUCTION DU PROGRAMME TELEPHONIQUE*/

'DEBUT'
'RECEVOIR'(URDR,|NOUVEL-APPEL|);
'QUAND' NOUVEL-APPEL 'ALORS' 'DEBUT'
                             'AVEC' 'PTR' P,DISCRIDR,DISCRIDE1,DISCRIDE2 'FAIRE'
                             'SI' DR-OBSERVE 'ALORS'  'APPEL' OBS-INIT(DR) 'FSI';

                             /*C"EST L"INITIALISATION DE L"OBSERVATION DU DEMANDEUR*/

                             'SELON' CODEDR 'FAIRE'
                             AB:'DEBUT'
                                'APPEL' ACCEPTERRACCRO(DR);
                                PRESEL:'DEBUT'
                                       'SI' DR-CLAVIER 'ALORS' 'DEBUT'
                                                               'ENVOI'(ETA,RESERV-RNC);
                                                               'RECEVOIR'(ETA,|REP-RESERV-POSIT|);
                                                               'QUANDNON' REP-RESERV-POSIT 'FAIRE' 'PROVOQUER'
                                                                                           ETABLI-MAL-DEROULE;
                                                               'ENVOI'(RCX,CNX-RNC-ENVOI-IN);
                                                               'RECEVOIR'(RCX,|COMPTERENDU|);
                                                               'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                                                                              PRESEL-MAL-DEROULE;
                                                               UNITE:=RNC
                                                               'FIN'
                                                           'SINON' 'DEBUT'
                                                               'ENVOI'(RCX,CNX-TONAL-IN);
                                                               'RECEVOIR'(RCX,|COMPTERENDU|);
                                                               'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                                                                              ETABLI-MAL-DEROULE;
                                                               UNITE:=URDR
                                                               'FIN'
                                                               'FSI'
                                       'FIN';

                               NUMEROTATION:'DEBUT'
                                            NCHATTENDU:=2;
                                            'ENVOI'(UNITE,RECEV-N-CHIF-PERE-IMP);
                                            'AVANT' 16 'SEC' 'ATTENDRE'(1,(UNITE,|PERE-IMP-RECU|));
                                            'QUANDNON' PERE-IMP-RECU 'FAIRE'  'PROVOQUER' NUM-MAL-DEROULE;
                                            'ENVOI'(RCX,STOP-TONAL-IN);
                                            'RECEVOIR'(RCX,|COMPTERENDU|);
                                            'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' NUM-MAL-DEROULE;
                                            'AVANT' 32 'SEC' 'ATTENDRE'(1,(UNITE,|RECU-N-CHIF|));
                                            'QUANDNON' RECU-N-CHIF 'FAIRE' 'PROVOQUER' NUM-MAL-DEROULE;
                                            NCHRECU:=2;
                                            'SELON' CHIFRECU(1) 'FAIRE'
                                            1:'SELON' CHIFRECU(2) 'FAIRE'
```

```
                    0,2,3,
                    4,7,8:
                    5,6:'SELON' DR-SR 'FAIRE'
                        NATIONAL,
                        INTERNATIONAL:'DEBUT'
                                      TYP-COMM:=NORMAL;
                                      N:=NB-CHIF-NATIONAL;
                                      N:=N-2;
                                      'APPEL' RECEPTION-CHIF
                                      'FIN'
                        REGIONAL:'PROVOQUER' NUM-INTERDIT
                        'FINSELON'
                    9:'SI' DR-SR /= INTERNATIONAL 'ALORS' 'PROVOQUER' NUM-INTERDIT
                      'FSI'
                    'AUTRE'  'PROVOQUER' NUM-INTERDIT
                    'FINSELON'
                  2,3,4,5,
                  6,7,8,9,0: 'SELON' CHIFRECU(2) 'FAIRE'
                             =//:TYP-COMM:=NUM-COURT
                             1,2,3,4,5,
                             6,7,8,9,0: 'DEBUT'
                                        NCHATTENDU:=1;
                                        'ENVOI'(UNITE,RECEV-N-CHIF);
                                        'AVANT' 16 'SEC' 'ATTENDRE'(1,(UNITE,
                                                                       |RECU-N-CHIF|));
                                        'QUANDNON' RECU-N-CHIF 'FAIRE' 'PROVOQUER'
                                                                       NUM-MAL-DEROULE;
                                        NCHRECU:=NCHRECU+1;
                                        'SELON' CHIFRECU(3) 'FAIRE'
                                        1,2,3,4,
                                        5,6,7,8,9,0:'DEBUT'
                                                    TYP-COMM:=NORMAL;
                                                    N:=NB-CHIF-LOCALE;
                                                    N:=N-3;
                                                    'APPEL' RECEPTION-CHIF
                                                    'FIN'
                                        =//:TYP-COMM:=NUM-COURT
                                        'AUTRE'  'PROVOQUER' NUM-INTERDIT
                                        'FINSELON'
                                'FINSELON'
                  * :  /*SERVICES SPECIAUX(UTILISATION OU ACTIVATION)*/
                  =// :  /*SERVICES SPECIAUX(ANNULATION OU DESACTIVATION)*/
                  'FINSELON'
                  'FIN'

LIBERATION-RNC:'DEBUT'
                 'SI' DR-CLAVIER 'ALORS' 'DEBUT'
                                         'ENVOI'(ETA,LIB-RNC);
```

```
                                        'RECEVOIR'(ETA,|ACCUSE-RESTITU|);
                                        'QUANDNON' ACCUSE-RESTITU 'FAIRE' 'PROVOQUER
                                                                  LIB-RNC-MAL-DEROULE;
                                        'ENVOI'(RCX,DCNX-RNC);
                                        'RECEVOIR'(RCX,|COMPTERENDU|);
                                        'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                                                  ETABLI-MAL-DEROULE
                                        'FIN'
                                        'FSI'
                'FIN';

SELECTION:'DEBUT'
          'SELON' TYP-COMM 'FAIRE'
          NORMAL:'DEBUT'
                 'ENVOI'(TR,TR-NORMAL);
                 'RECEVOIR'(TR,|EQ,PF,GF,REP-NEG-TR|);
                 'SUIVANT' CODE-FONCTION 'FAIRE'
                 EQ:'APPEL' INTERRO-URDE1(TEST-EQ)
                 PF:'APPEL' INTERRO-URDE1(TEST-FAISCEAU)
                 GF:'APPEL' INTERRO-TOUTES-URDE1
                 REP-NEG-TR:'PROVOQUER' FAUX-DE
                 'AUTRE' 'PROVOQUER' ETABLI-MAL-DEROULE
                 'FINSUIVANT';
                 'SELON' IND-REP-URDE1 'FAIRE'
                 LIBRE:
                 OCC-CONVERS:'APPEL' AE-IAI
                 OCC: 'PROVOQUER' DE-OCC
                 INDISPONIBLE,
                 PAS-VTLR,
                 EN-FAUTE: 'PROVOQUER' DE-IMPOSSIBLE
                 TRANCON-EPUISE:'SELON' IND-TR 'FAIRE'
                                AB: 'PROVOQUER' DE-IMPOSSIBLE
                                CIRCUIT-Y:'DEBUT'
                                          'ENVOI'(TR,DEBORDEMENT);
                                          'RECEVOIR'(TR,|PF,GF,REP-NEG-TR|);
                                          'SUIVANT' CODE-FONCTION 'FAIRE'
                                          PF:'DEBUT'
                                             'APPEL' INTERRO-URDE1(TEST-FAISCEAU);
                                             'APPEL' TEST-REP-URDE1-DEB
                                             'FIN'
                                          GF:'DEBUT'
                                             'APPEL' INTERRO-TOUTES-URDE1;
                                             'APPEL' TEST-REP-URDE1-DEB
                                             'FIN'
                                          REP-NEG-TR:'SELON' IND-DEB 'FAIRE'
                                                     EN-FAUTE,
                                                     OCC:'PROVOQUER' FAUX-DE
                                                     PAS-DEB:'APPEL' DETOURNEMENT
```

```
                                                    'FINSELON'
                                        'AUTRE'  'PROVOQUER' ETABLI-MAL-DEROULE
                                        'FINSUIVANT'
                                        'FIN'

                                CIRCUIT-MF:'DEBUT'
                                           - - - - - - -
                                           - - - - - - -
                                           /*CE BLOC N"EST PAS DECRIT DANS CE PROGR
                                           - - - - - - -
                                           - - - - - - -
                                           FIN'

                                - - - - - - -
                                - - - - - - -
                                'FINSELON'

                        'FINSELON'
                        'FIN'
                NUM-COURT:'DEBUT'
                          - - - - - - - -
                          - - - - - - - -
                          /*CE BLOC TRAITE LA SELECTION DANS LE CAS DU SERVICE
                          SPECIAL NUMERO-COURT, IL N"EST PAS DECRIT DANS
                          CE PROGRAMME */
                          - - - - - - - -
                          - - - - - - - -
                          'FIN'

                - - - - - - - -
                - - - - - - - -
                - - - - - - - -
                'FINSELON';
                'ENVOI'(RCX,CNX-DR-DE1);
                'RECEVOIR'(RCX,|COMPTERENDU|);
                'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' CNX-MAL-DEROULE
                'FIN'

APPEL-DE:'DEBUT'
         'SELON' CODEDE1 'FAIRE'
         AB:'DEBUT'
            APPEL-LOCAL:=APPEL-LOCAL+1;
            'SI' DR-OBSERVE 'ALORS' 'APPEL' OBS-SONNERIE(DR) 'FSI'
            'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-SONNERIE(DE1) 'FSI';
            'AVANT' 4 'MIN' 'ATTENDRE'(1,(URDE1,|DECRO-DE1|));
            'QUANDNON' DECRO-DE1 'FAIRE' 'PROVOQUER' NON-DECRO-DE1-AB;
            APPEL-LOCAL-EFFICACE:=APPEL-LOCAL-EFFICACE+1
            'FIN'
```

```
CIRCUIT-Y:'DEBUT'
          'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-INIT(DE1) 'FSI';
          'ENVOI'(RCX,TONAL-ACH-DR);
          'RECEVOIR'(RCX,|COMPTERENDU|);
          'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                         APPEL-DE1-MAL-DEROULE;
          'AVANT' 1 'MIN' 'ATTENDRE'(1,(URDE1,|RECU-IT|));
          'QUANDNON' RECU-IT 'FAIRE' 'PROVOQUER' NON-RECU-IT;
          'ENVOI'(RCX,STOP-TONAL-ACH-DR);
          'RECEVOIR'(RCX,|COMPTERENDU|);
          'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                         APPEL-DE1-MAL-DEROULE;
          N:=NB-CHIF-CIRCUIT-Y;
          NCHRECU:=0;
          'TANTQUE' N>=4 'FAIRE'
          'DEBUT'
          NCH-A-EMETTRE:=4;
          'ENVOI'(URDE1,EMET-N-CHIF);
          'AVANT' 64 'SEC' 'ATTENDRE'(1,(URDE1,|N-CHIF-EMIS|));
          'QUANDNON' N-CHIF-EMIS 'FAIRE' 'PROVOQUER'
                                         APPEL-DE1-MAL-DEROULE;
          N:=N-4;
          NCHRECU:=NCHRECU+4
          'FIN';
          'SI' N>0 'ALORS' 'DEBUT'
                           NCH-A-EMETTRE:=N;
                           'ENVOI'(URDE1,EMET-N-CHIF);
                           M:=N*16;
                           'AVANT' M 'SEC' 'ATTENDRE'(1,(URDE1,
                                                  |N-CHIF-EMIS|));
                           'QUANDNON' N-CHIF-EMIS 'FAIRE' 'PROVOQUER'
                                                  APPEL-DE1-MAL-DEROULE;
                           NCHRECU:=NCHRECU+N
                           'FIN'
                           'FSI';
          'AVANT' 2 'MIN' 'ATTENDRE'(1,(URDE1,|FSAL,FSAOCC|));
          'SUIVANT' CODE-FONCTION 'FAIRE'
          FSAL:'DEBUT'
               'SI' DR-OBSERVE 'ALORS' 'APPEL' OBS-SONNERIE(DR) 'FSI';
               'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-SONNERIE(DE1) 'FSI';
               'AVANT' 930 'SEC' 'ATTENDRE'(1,URDE1,|DECRO-DE1|));
               'QUANDNON' DECRO-DE1 'FAIRE' 'PROVOQUER'
                                            NON-DECRO-DE1-CIRCUIT-Y
               'FIN'
          FSAOCC:'APPEL' AE
          'AUTRE'  'PROVOQUER'  FAUX-REP-URDE1
          'FINSUIVANT'
          'FIN'
```

```
CIRCUIT-MF:'DEBUT'
           - - - - - - - -
           - - - - - - - -
           - - - - - - - -
           'FIN'

- - - - - - - -
- - - - - - - -
'FINSELON';

'ENVOI'(URDE1,SURV2);
'ENVOI'(URDR,SURV2);
FIN-ETABLI:=EFFICACE;
'APPEL' DEBUTER-TAXATION(DR);
'SI' DR-OBSERVE 'ALORS' 'DEBUT'
                        'APPEL' OBS-CONVERS(DR);
                        'ENVOI'(CG,FIN-ETABLI-DR)
                        'FIN'
                        'FSI'
'SI' DE1-OBSERVE 'ALORS' 'DEBUT'
                         'APPEL' OBS-CONVERS(DE1);
                         'ENVOI'(CG,FIN-ETABLI-DE1)
                         'FIN'
                         'FSI'
'FIN'
'SI' ¬DR-TLTX 'ALORS' 'AVANT' MAX 'SEC' 'ATTENDRE'(2,(URDR,
                      |RACCRODR|),(URDE1,|RACCRODE1|))
'FSI';
```

```
/*SERVICE SPECIAL CONFERENCE */

'SUIVANT' CODE-FONCTION 'FAIRE'
BR-DR:'DEBUT'
      'SELON' IND-BR 'FAIRE'
      DE:'APPEL' IAM

      DR:PRE-CCF:'DEBUT'
                 'APPEL' ACCEPTERRACCRO(DR);
                 TYP-COMM:=CCF;
                 'APPEL' SUSPEND-TAXATION;
                 'APPEL' RESERVATION-ETA(RESERV-RNC-CCF);
                 'QUANDNON' REP-POSIT-RESERV 'FAIRE' 'PROVOQUER'
                                                    ETABLI-MAL-DEROULE;
                 'ENVOI'(RCX,CNX-TONAL-IN-INTERRO-MISE-GARDE);
                 'RECEVOIR'(RCX,|REP-POSIT-INTERRO-RCX|);
                 'QUANDNON' REP-POSIT-INTERRO-RCX 'FAIRE' 'PROVOQUER'
                                              PRESELEC-CCF-MAL-DEROULE;
                 'ENVOI'(URDE1,INTERRO-EQ-SURV1);
                 'RECEVOIR'(URDE1,|REP-POSIT-URDE1,REP-NEG-URDE1|);
                 'SI' CODE-DE1=AB 'ALORS' PREMIER-DE-AB:='VRAI'
                                  'SINON' PREMIER-DE-AB:='FAUX'
                                  'FSI'
                 'FIN';

         NUM-CCF:'DEBUT'
                 'APPEL' ACCEPTERRACCRO(DE1);
                 NCHATTENDU:=2;
                 'ENVOI'(RNC,RECEV-N-CHIF-PERE-IMP);
                 'AVANT' 16 'SEC' 'ATTENDRE'(1,(RNC,|PERE-IMP-RECU|));
                 'QUANDNON' PERE-IMP-RECU 'FAIRE' 'PROVOQUER'
                                                    ETABLI-CCF--MAL-DEROULE;
                 'ENVOI'(RCX,STOP-TONAL-IN);
                 'RECEVOIR'(RCX,|COMPTERENDU|);
                 'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                                ETABLI-CCF-MAL-DEROULE;
                 'AVANT' 32 'SEC' 'ATTENDRE'(1,(RNC,|RECU-N-CHIF|));
                 'QUANDNON' RECU-N-CHIF 'FAIRE' 'PROVOQUER'
                                                    ETABLI-CCF-MAL-DEROULE;
                 NCHRECU:=2;
                 'SELON' CHIFRECU(1) 'FAIRE'
                 1:'SELON' CHIFRECU(2) 'FAIRE'
                   0,2,3,
                   4,7,8:
                   5,
                   6:'SELON' DR-SR 'FAIRE'
                     NATIONAL,
                     INTERNATIONAL:'DEBUT'
```

```
                        TYP-COMM:=NORMAL;
                        N:=NB-CHIF-NATIONAL;
                        N:=N-2;
                        'APPEL' RECEPTION-CHIF-CCF
                        'FIN'
      REGIONAL:'APPEL' FAUX-MANOEUVRE-CCF
      'FINSELON'
    9:'SI' DR-SR/=INTERNATIONAL 'ALORS' 'APPEL'
                                        FAUX-MANOEUVRE-CCF;
    'AUTRE' 'APPEL' FAUX-MANOEUVRE-CCF
    'FINSELON'
  2,3,4,5,
  6,7,8,9,
  0:'SELON' CHIFRECU(2) 'FAIRE'
    =//:TYP-COMM:=NUM-COURT
    1,2,3,4,5,
    6,7,8,9,
    0:'DEBUT'
      NCHATTENDU:=1;
      'ENVOI'(RNC,RECEV-N-CHIF);
      'AVANT' 16 'SEC' 'ATTENDRE'(1,(RNC,|RECU-N-CHIF|));
      'QUANDNON' RECU-N-CHIF 'FAIRE' 'PROVOQUER'
                                    ETABLI-CCF-MAL-DEROULE;
      NCHRECU:=NCHRECU+1;
      'SELON' CHIFRECU(3) 'FAIRE'
      1,2,3,4,5,
      6,7,8,9,
      0:'DEBUT'
        TYP-COMM:=NORMAL;
        N:=NB-CHIF-LOCAL ;
        N:=N-3;
        'APPEL' RECEPTION-CHIF-CCF
        'FIN'
      =//:TYP-COMM:=NUM-COURT
      'AUTRE' 'APPEL' FAUX-MANOEUVRE-CCF
      'FINSELON'
      'FIN'
    'FINSELON'
  *:/* SERVICES SPECIAUX (UTILISATION OU ACTIVATION)*/
  =//:/* SERVICES SPECIAUX (ANNULATION OU DESACTIVATION)*/
  'FINSELON'
  'FIN';

SEL-CF:'DEBUT'
/* C"EST LE BLOC SELECTION CONFERENCE*/
      'SELON' TYP-COMM 'FAIRE'
      NORMAL:'DEBUT'
             'ENVOI'(TR,TR-NORMAL);
```

```
                    'RECEVOIR'(TR,|EQ,PF,GF,REP-NEG-TR|);
                    'SUIVANT' CODE-FONCTION 'FAIRE'
                    EQ:'APPEL' INTERRO-URDE2-CCF(TEST-EQ)
                    PF:'APPEL' INTERRO-URDE2-CCF(TEST-FAISCEAU)
                    GF:'APPEL' INTERRO-TOUTES-URDE2
                    REP-NEG-TR:'APPEL' FAUX-DE2-CCF
                    'AUTRE'   'PROVOQUER' ETABLI-CCF-MAL-DEROULE
                    'FINSUIVANT';

/*ON VIENT ICI QUAND LA REP DU TR EST EGALE A PF,GF OU EQ.LA
PROCEDURE RETOUR-DE1 COMPREND A SA FIN UNE SORTIE DU PROGRAMME*/

                    'SELON' IND-REP-URDE2 'FAIRE'
                    LIBRE:
                    OCC-CONVERS:'APPEL' AE-IAI
                    OCC:'APPEL' DE2-OCC-CCF
                    INDISPONIBLE,
                    PAS-VTLR,
                    EN-FAUTE:'APPEL' DE2-IMPOSSIBLE-CCF
                    TRANCON-EPUISE:'DEBUT'
                                   'SELON' IND-TR 'FAIRE'
                                   AB:'APPEL' DE2-IMPOSSIBLE-CCF
                                   CIRCUIT-Y:
                                   'DEBUT'
                                   'ENVOI'(TR,DEBORDEMENT);
                                   'RECEVOIR'(TR,|PF,GF,REP-NEG-TR|);
                                   'SUIVANT' CODE-FONCTION 'FAIRE'
                                   PF:'DEBUT'
                                      'APPEL'
                                      INTERRO-URDE2-CCF(TEST-FAISCEAU);
                                      'APPEL' TEST-REP-URDE2-DEB-CCF
                                      'FIN'
                                   GF:'DEBUT'
                                      'APPEL' INTERRO-TOUTES-URDE2-CCF;
                                      'APPEL' TEST-REP-URDE2-DEB-CCF
                                      'FIN'
                                   REP-NEG-TR:'SELON' IND-DEB 'FAIRE'
                                              EN-FAUTE,
                                              OCC:'APPEL' FAUX-DE2-CCF
                                              PAS-DEB:'APPEL'
                                                  DETOURNEMENT-CCF
                                              'FINSELON'
                                   'AUTRE' 'PROVOQUER'
                                           ETABLI-CCF-MAL-DEROULE
                                   'FINSUIVANT'
                                   'FIN'

                                   CIRCUIT-MF:'DEBUT'
```

```
                                                    - - - - - - - - -
                                                    - - - - - - - - -
/* CE BLOC N"EST PAS DECRIT DANS CE PROGRAMME*/
                                                    - - - - - - - - -
                                                    - - - - - - - - -
                                                    'FIN'

                                        - - - - - - - - -
                                        - - - - - - - - -
                                        - - - - - - - - -
                                        'FINSELON'
                                        'FIN'
                          'FINSELON'
                          'FIN'

                  NUM-COURT:'DEBUT'
                                  - - - - - - - - -
                                  - - - - - - - - -
/* CE BLOC N"EST PAS DECRIT DANS CE PROGRAMME*/
                                  - - - - - - - - -
                                  - - - - - - - - -
                                  'FIN'

                  - - - - - - - - -
                  'FINSELON';

                  'SI' (CODEDE2=AB) 'OU' (PREMIER-DE-AB='VRAI') 'ALORS'
                                          'SINON' 'APPEL'
                                                  FAUX-MANOEUVRE-CCF
                                         'FSI';
                  'ENVOI'(RCX,CNX-DR-DE2);
                  'RECEVOIR'(RCX,|COMPTERENDU|);
                  'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                                 CNX-MAL-DEROULE-CCF
                  'FIN';

          APEL-CF:'DEBUT'
          /*C"EST LE BLOC D"APPEL DU DEUXIEME DEMANDE*/
                   'SELON' CODEDE2 'FAIRE'
                   AB:'DEBUT'
                      APPEL-LOCAL:=APPEL-LOCAL+1;
                      'SI' DR-OBSERVE 'ALORS' 'APPEL'
                                              OBS-SONNERIE(DR) 'FSI';
                      'SI' DE2-OBSERVE 'ALORS' 'APPEL'
                                             OBS-INIT-SONNERIE(DE2) 'FSI';
                      'AVANT' 4 'MIN' 'ATTENDRE'(1,(URDE2,|DECRO-DE2|));
                      'QUANDNON' DECRO-DE2 'FAIRE' 'PROVOQUER'
                                                   NON-DECRO-DE2-AB-CCF;
```

```
      APPEL-LOCAL-EFFICACE:=APPEL-LOCAL-EFFICACE+1
      'FIN'

CIRCUIT-Y:'DEBUT'
          'SI' DE2-OBSERVE 'ALORS' 'APPEL'
                                   OBS-INIT(DE2) 'FSI';
          'ENVOI'(RCX,TONAL-ACH-DR);
          'RECEVOIR'(RCX,|COMPTERENDU|);
          'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                   APPEL-DE2-MAL-DEROULE;
          'AVANT' 1 'MIN' 'ATTENDRE'
                          (1,(URDE2,|RECU-IT|));
          'QUANDNON' RECU-IT 'FAIRE' 'PROVOQUER'
                                   NON-RECU-IT-CCF;
          'ENVOI'(RCX,STOP-TONAL-ACH-DR);
          'RECEVOIR'(RCX,|COMPTERENDU|);
          'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                   APPEL-DE2-MAL-DEROULE;
          N:=NB-CHIF-CIRCUIT-Y;
          NCHRECU:=0;
          'TANTQUE' N>=4 'FAIRE'
          'DEBUT'
          NCH-A-EMETTRE:=4;
          'ENVOI'(URDE2,EMET-N-CHIF);
          'AVANT' 64 'SEC' 'ATTENDRE'
                          (1,(URDE2,|N-CHIF-EMIS|));
          'QUANDNON' N-CHIF-EMIS 'FAIRE' 'PROVOQUER'
                              APPEL-DE2-MAL-DEROULE;
          N:=N-4;
          NCHRECU:=NCHRECU+4;
          'FIN';
          'SI' N>0 'ALORS' 'DEBUT'
                          NCH-A-EMETTRE:=N;
                          'ENVOI'(URDE2,EMET-N-CHIF);
                          M:=N*16;
                          'AVANT' M 'SEC' 'ATTENDRE'
                             (1,(URDE2,|N-CHIF-EMIS|));
                          'QUANDNON' N-CHIF-EMIS 'FAIRE'
                                                 'PROVOQUER'
                          APPEL-DE2-MAL-DEROULE;
                          NCHRECU:=NCHRECU+N
                          'FIN'
                          'FSI';
          'AVANT' 2 'MIN' 'ATTENDRE'
                    (1,(URDE2,|FSAL,FSAOCC|));
          'SUIVANT' CODE-FONCTION 'FAIRE'
          FSAL:'DEBUT'
               'SI' DR-OBSERVE 'ALORS' 'APPEL'
```

```
                                        OBS-SONNERIE(DR) 'FSI';
                    'SI' DE2-OBSERVE 'ALORS' 'APPEL'
                                        OBS-SONNERIE(DE2) 'FSI';
                    'AVANT' 930 'SEC' 'ATTENDRE'
                                          (1,(URDE2,|DECRO-DE2|));
                    'QUANDNON' DECRO-DE2 'FAIRE' 'PROVOQUER'
                                       NON-DECRO-DE2-CIRCUIT-Y-CCF
                    'FIN'
                FSAOCC:'APPEL' AE
                'AUTRE'  'PROVOQUER' APPEL-DE2-MAL-DEROULE
                'FINSUIVANT'
                'FIN'

        CIRCUIT-MF:'DEBUT'
                    - - - - - - - - -
                    - - - - - - - - -
/* CE BLOC N"EST PAS DECRIT DANS CE PROGRAMME*/
                    - - - - - - - - -
                    - - - - - - - - -
                    'FIN'

        - - - - - - - - -
        - - - - - - - - -
        'FINSELON';
        'SI' DR-OBSERVE 'ALORS' 'APPEL' OBS-CONVERS(DR) 'FSI';
        'SI' DE2-OBSERVE 'ALORS' 'APPEL' OBS-CONVERS(DE2) 'FSI';
        'FIN';

ETAB-CF:'DEBUT'
/*C"EST LE BLOC DE L"ETABLISSEMENT DE LA CONFERENCE*/
        'AVANT' 16 'SEC' 'ATTENDRE'(1,(URDE2,|BOUTON-RAPPEL-1|));
        'APPEL' CHOIX-DR;
        'SELON' CHIF-CCF 'FAIRE'
        3:'APPEL' CCF-TROIS-AB
        2:'DEBUT'
          'ENVOI'(RCX,MISE-GARDE-DE2);
          'RECEVOIR'(RCX,|COMPTERENDU|);
          'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                          CHOIX-DR-MAL-DEROULE;
          'ENVOI'(URDR,SURV1);
          'ENVOI'(URDE1,SURV1);
          'ENVOI'(URDE2,SURV1);
          'APPEL' REPRISE-TAXATION;
          'AVANT' MAX 'SEC' 'ATTENDRE'
                            (1,(URDE2,|BOUTON-RAPPEL-1|));
          'APPEL' CHOIX-DR;
          'SI' CHIF-CCF=3 'ALORS' 'APPEL' CCF-TROIS-AB
                          'SINON' 'PROVOQUER' CCF-MAL-FINI
```

```
                                        'FSI'
                        'FIN'
                    'AUTRE'  'PROVOQUER' CCF-MAL-FINI
                    'FINSELON';
                    FIN-ETABLI:=EFFICACE;
                    'SI' DR-OBSERVE 'ALORS' 'ENVOI'(CG,FIN-ETABLI-DR) 'FSI';
                    'SI' DE1-OBSERVE 'ALORS' 'ENVOI'(CG,FIN-ETABLI-DE1) 'FSI'
                    'SI' DE2-OBSERVE 'ALORS' 'ENVOI'(CG,FIN-ETABLI-DE2) 'FSI'
                    'FIN'
        'FINSELON'
        'FIN'

RACCRODR:'DEBUT'
         'SI' DR-TAXE 'ALORS' 'APPEL' ARRET-TAXATION 'FSI';
         'ENVOI'(RCX,INTERRO-CORRESPT-DR-DE1);
         'RECEVOIR'(RCX,|REP-POSIT-INTERRO-RCX|);
         'QUANDNON' REP-POSIT-INTERRO-RCX 'FAIRE' 'PROVOQUER'
                                                  RUPT-DR-MAL-DEROULE;
         'ENVOI'(URDE1,INTERRO-EQ-SURV1);
         'RECEVOIR'(URDE1,|REP-POSIT-URDE1,REP-NEG-URDE1|);
         'QUANDNON' REP-POSIT-URDE1 'FAIRE' 'PROVOQUER'
                                            RUPT-DR-MAL-DEROULE;
         'SI' CODEDE1=CIRCUIT-Y 'ALORS' 'APPEL'
                                        LIB-DE1-CIRCUIT-Y 'FSI';
         'SI' CODEDE1=AB 'ALORS' 'DEBUT'
                                 'ENVOI'(RCX,TONAL-OCC-DE1);
                                 'RECEVOIR'(RCX,|COMPTERENDU|);
                                 'QUAND' COMPTERENDU 'ALORS' 'AVANT'
                                                      1 'MIN' 'ATTENDRE'
                                                  (1,(URDE1,|RACCRODE1|));
                                                      'SINON' 'PROVOQUER'
                                                      NON-TONAL-OCC-DE1
                                 'FINQUAND';
                                 'APPEL' LIB-DE1-AB;
                                 'ENVOI'(RCX,STOP-TONAL-OCC-DE1);
                                 'RECEVOIR'(RCX,|COMPTERENDU|);
                                 'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER
                                                      RUPT-DR-MAL-DEROULE
                                 'FIN'
                                 'FSI';
         'SI' DR-OBSERVE 'ALORS' 'APPEL' OBS-FIN-RUPT(DR) 'FSI';
         'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-FIN-RUPT(DE1) 'FSI';
         FIN-RUPT:=EFFICACE
         'FIN'

RACCRODE1:'DEBUT'
          'SELON' CODEDR 'FAIRE'
          AB:'DEBUT'
```

```
                        'SI' DR-TAXE 'ALORS' 'APPEL' SUSPEND-TAXATION 'FSI';
                        'ENVOI'(RCX,INTERRO-CORRESPT);
                        'RECEVOIR'(RCX,|REP-POSIT-INTERRO-RCX,REP-NEG-INTERRO-RCX|);
                        'QUANDNON' REP-POSIT-INTERRO-RCX 'FAIRE' 'PROVOQUER'
                                                        RUPT-DE1-MAL-DEROULE;
                        'ENVOI'(URDR,INTERRO-EQ-SURV1);
                        'RECEVOIR'(URDR,|REP-POSIT-URDR,REP-NEG-URDR|);
                        'QUANDNON' REP-POSIT-URDR 'FAIRE' 'PROVOQUER'
                                                    RUPT-DE1-MAL-DEROULE;
                        'AVANT' 3 'SEC' 'ATTENDRE'(1,(URDE1,|DECRO-DE1|));
                        'QUANDNON' DECRO-DE1 'FAIRE' 'PROVOQUER' NON-REPRISE-COMM;
                        'SI' DR-TAXE 'ALORS' 'APPEL' REPRISE-TAXATION 'FSI';
                        'ENVOI'(URDR,SURV2);
                        'ENVOI'(URDE1,SURV2);
                        'SI' DR-OBSERVE 'ALORS' 'APPEL' OBS-FIN-RUPT(DR) 'FSI';
                        'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-FIN-RUPT(DE1) 'FSI';
                        FIN-RUPT:=EFFICACE
                        'FIN'
            'AUTRE'
            'FINSUIVANT'
            'FIN'

CIRCUIT-Y:'DEBUT'
          - - - - - - - - -
          - - - - - - - - -
          /*CE BLOC TRAITE LA COMMUNICATION ARRIVEE-Y,IL N"EST PAS DECRIT
          DANS CE PROGRAMME */
          - - - - - - - - -
          - - - - - - - - -
          'FIN'

CIRCUIT-GSM:'DEBUT'
            - - - - - - - - -
            - - - - - - - - -
            /*CE BLOC N"EST PAS DECRIT DANS CE PROGRAMME*/
            - - - - - - - - -
            - - - - - - - - -
            'FIN'

CIRCUIT-MF:'DEBUT'
            'APPEL' ACCEPTERRACCRO(DR);
            APPEL-ARRIVEE:=APPEL-ARRIVEE+1;
            PRESEL-MF:'DEBUT'
                      'ENVOI'(ETA,RESERV-RGMF);
                      'RECEVOIR'(ETA,|REP-RESERV-POSIT|);
                      'QUANDNON' REP-RESERV-POSIT 'FAIRE' 'PROVOQUER'
                                                        PRESEL-MF-MAL-DEROULE;
                      'ENVOI'(RCX,CNX-DR-RGMF);
```

```
                'RECEVOIR'(RCX,|COMPTERENDU|);
                'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER'
                                                ETABLI-MF-MAL-DEROULE;
                'FIN';

NUM-SELEC-MF:'DEBUT'
                NCHATTENDU:=4;
                'ENVOI'(RGMF,RECEV-N-CHIF-EMET-A1);
                'AVANT' 64 'SEC' 'ATTENDRE'(1,(RGMF,|A1-EMIS-RECU-N-CHIF|));
                'QUANDNON' A1-EMIS-RECU-N-CHIF 'FAIRE' 'PROVOQUER'
                                                        ETABLI-MF-MAL-DEROULE;
                NCHRECU:=4;
                'ENVOI'(TR,TR-INCOMPLETE);
                'RECEVOIR'(TR,|EQ,PF,GF,NM-INF,REP-NEG-TR|);
                'SUIVANT' CODE-FONCTION 'FAIRE'
                NM-INF:'DEBUT'
                /*NUMEROTATION-INSUFFISANTE*/
                        N:=NB-CHIF-MF;
                        N:=N-4;
                        'TANTQUE' N>=4 'FAIRE'
                        'DEBUT'
                        NCHATTENDU:=4;
                        'ENVOI'(RGMF,RECEV-N-CHIF-EMET-A1);
                        'AVANT' 64 'SEC' 'ATTENDRE'
                                        (1,(RGMF,|A1-EMIS-RECU-N-CHIF|));
                        'QUANDNON' A1-EMIS-RECU-N-CHIF 'FAIRE' 'PROVOQUER'
                                                        ETABLI-MF-MAL-DEROULE;
                        NCHRECU:=NCHRECU+4;
                        N:=N-4
                        'FIN';
                        'SI' N>0 'ALORS' 'DEBUT'
                                        NCHATTENDU:=N;
                                        'ENVOI'(RGMF,RECEV-N-CHIF-EMET-A1);
                                        M:=N*16;
                                        'AVANT' M 'SEC' 'ATTENDRE'
                                           (1,(RGMF,|A1-EMIS-RECU-N-CHIF|));
                                        'QUANDNON' A1-EMIS-RECU-N-CHIF
                                                'FAIRE' 'PROVOQUER'
                                        ETABLI-MF-MAL-DEROULE;
                                        NCHRECU:=NCHRECU+N
                                        'FIN'
                                        'FSI';
                        'ENVOI'(TR,TR-COMPLETE);
                        'RECEVOIR'(TR,|EQ,PF,GF,REP-NEG-TR|);
                        'SUIVANT' CODE-FONCTION 'FAIRE'
                        EQ:'APPEL' INTERRO-URDE1-MF(TEST-EQ)
                        PF:'APPEL' INTERRO-URDE1-MF(TEST-FAISEAU)
                        GF:'APPEL' INTERRO-TOUTES-URDE1
```

```
                    REP-NEG-TR:'PROVOQUER' FAUX-DE1-MF
                    'AUTRE'  'PROVOQUER' ETABLI-MF-MAL-DEROULE
                    'FINSUIVANT'
                    'FIN'
               EQ:'APPEL' INTERRO-URDE1-MF(TEST-EQ)
               PF:'APPEL' INTERRO-URDE1-MF(TEST-FAISCEAU)
               GF:'APPEL' INTERRO-TOUTES-URDE1
               REP-NEG-TR:'PROVOQUER' FAUX-DE1-MF
               'AUTRE' 'PROVOQUER' ETABLI-MF-MAL-DEROULE
               'FINSUIVANT';
               'SELON' IND-REP-URDE1 'FAIRE'
               LIBRE:
               OCC:'PROVOQUER' DE1-OCC-MF
               EN-FAUTE,
               INDISPONIBLE:'PROVOQUER' DE1-IMPOSSIBLE-MF
               'AUTRE'  'PROVOQUER' MUM-SELEC-MAL-DEROULE
               'FINSELON'
               'FIN';

LIBR-RGMF:'DEBUT'
          'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-INIT(DE1) 'FSI';
          NCH-A-ENVOYER:=2;
          'ENVOI'(ETA,ENVOI-N-CHIF-RESTITU-RGMF);
          'RECEVOIR'(ETA,|N-CHIF-ENVOYE-ACCUSE-RESTITU|);
          'QUANDNON' N-CHIF-ENVOYE-ACCUSE-RESTITU 'FAIRE' 'PROVOQUER'
                                                    LIB-RGMF-MAL-DEROULE;
          'ENVOI'(RCX,DCNX-RGMF);
          'RECEVOIR' (RCX,|COMPTERENDU|);
          'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' DCNX-RGMF-MAL-DEROULE
          'FIN'

APL-DE1-MF:'DEBUT'
           'SI' DR-OBSERVE 'ALORS' 'APPEL' OBS-SONNERIE(DR) 'FSI';
           'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-SONNERIE(DE1) 'FSI';
           'AVANT' 4 'MIN' 'ATTENDRE'(1,(URDE1,|DECRO-DE1|));.
           'QUANDNON' DECRO-DE1 'FAIRE' 'PROVOQUER' NON-DECRO-DE1-AB-MF;
           'APPEL' ACCEPTERRACCRO(DE1);
           'ENVOI'(URDR,ENVOI-REP-DE1-SURV2);
           'ENVOI'(RCX,CNX-DR-DE1);
           'RECEVOIR'(RCX,|COMPTERENDU|);
           'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' ETABLI-MF-MAL-FINI;
           'ENVOI'(URDE1,SURV2);
           FIN-ETABLI:=EFFICACE;
           'SI' DR-OBSERVE 'ALORS' 'DEBUT'
                                   'APPEL' OBS-CONVERS(DR) 'FSI';
                                   'ENVOI'(CG,FIN-ETABLI-DR)
                                   'FIN'
                                   'FSI';
```

```
                'SI' DE1-OBSERVE 'ALORS' 'DEBUT'
                                         'APPEL' OBS-CONVERS(DE1) 'FSI';
                                         'ENVOI'(CG,FIN-ETABLI-DE1)
                                         'FIN'
                                         'FSI';
                APPEL-ARRIVEE-EFFICACE:=APPEL-ARRIVEE-EFFICACE+1
                'FIN'

'AVANT' MAX 'SEC' 'ATTENDRE'(2,(URDR,|RACCRODR|),(URDE1,
                                                    |RACCRODE1|));
'SUIVANT' CODE-FONCTION 'FAIRE'
RACCRODR:'DEBUT'
         'ENVOI'(RCX,INTERRO-CORRESPT-DCNX-DR-DE1);
         'RECEVOIR'(RCX,|REP-POSIT-INTERRO-RCX|);
         'QUANDNON' REP-POSIT-INTERRO-RCX 'FAIRE' 'PROVOQUER'
                                                  RUPT-DR-MAL-DEROULE;
         'ENVOI'(URDE1,INTERRO-EQ-SURV1);
         'RECEVOIR'(URDE1,|REP-POSIT-URDE1,REP-NEG-URDE1|);
         'QUANDNON' REP-POSIT-URDE1 'FAIRE' 'PROVOQUER'
                                            RUPT-DR-MAL-DEROULE;
         'ENVOI'(RCX,TONAL-OCC-DE1);
         'RECEVOIR'(RCX,|COMPTERENDU|);
         'QUAND' COMPTERENDU 'ALORS' 'AVANT' 1 'MIN' 'ATTENDRE'
                                                  (1,(URDE1,|RACCRODE1|))
                             'SINON' 'PROVOQUER' NON-TONAL-OCC-DE1
         'FINQUAND';
         'APPEL' LIB-DE1-AB;
         'ENVOI'(RCX,STOP-TONAL-OCC-DE1);
         'RECEVOIR'(RCX,|COMPTERENDU|);
         'QUANDNON' COMPTERENDU 'FAIRE' 'PROVOQUER' RUPT DR-MAL-DEROULE;
         'SI' DR-OBSERVE 'ALORS' 'APPEL' OBS-FIN-RUPT(DR) 'FSI';
         'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-FIN-RUPT(DE1) 'FSI';
         FIN-RUPT:=EFFICACE
         'FIN'
RACCRODE1:'DEBUT'
          'ENVOI'(RCX,INTERRO-CORRESPT);
          'RECEVOIR'(RCX,|REP-POSIT-INTERRO-RCX,REP-NEG-INTERRO-RCX|);
          'QUANDNON' REP-POSIT-INTERRO-RCX 'FAIRE' 'PROVOQUER'
                                                   RUPT-DE1-MF-MAL-DEROULE;
          'ENVOI'(URDR,INTERRO-EQ-SURV1);
          'RECEVOIR'(URDR,|REP-POSIT-URDR,REP-NEG-URDR|);
          'QUANDNON' REP-POSIT-URDR 'FAIRE' 'PROVOQUER'
                                            RUPT-DE1-MF-MAL-DEROULE;
          'ENVOI'(URDR,EMET-ALTERNES-2);
          'AVANT' 3 'SEC' 'ATTENDRE'(1,(URDE1,|DECRO-DE1|));
          'QUANDNON' DECRO-DE1 'FAIRE' 'PROVOQUER' NON-REPRISE-COMM-MF;
          'ENVOI'(URDE1,SURV2);
          'ENVOI'(URDE2,SURV2);
```

```
                                        'SI' DR-OBSERVE 'ALORS' 'APPEL' OBS-FIN-RUPT(DR) 'FSI';
                                        'SI' DE1-OBSERVE 'ALORS' 'APPEL' OBS-FIN-RUPT(DE1) 'FSI';
                                        FIN-RUPT:=EFFICACE
                                        'FIN'
                                'AUTRE'
                                'FINSUIVANT'
                                'FIN'

                        'FINSELON'
                        'FIN'
'TERMINERPROCESSUS'
'FIN'
'FIN' COM;
```

REFERENCES

ALI 75 ALIZON F., ROUGEOT B.
'Un langage adapté à la programmation structurée des unités de commande E10'
Note technique projet RCI/SIC/4, CNET

GRE 78 "GREEN"
programming language, Manuel de référence préliminaire
CII Honeywell Bull 15 février 1978

GRIF 69 GRIFFITS M., PELTIER M.
'A macro generable language for the 360 computers'
Computer Bulletin, Volume 13 n°11 Novembre 1969

ISS 79 ANCEAU F., COURTOIS B., LECERF C., MARINESCU M., OLAIWAN A.
PONS J.F.
'Projet d'architecture pour la commande d'un autocommutateur téléphonique'
International switching symposium 1979

OLA 78 OLAIWAN A.
'Langage de haut niveau conçu pour la programmation d'un organe de commande d'un autocommutateur téléphonique. Journées d'étude sur la fiabilité des programmes dans les applications industrielles
Chapitre français de l'ACM 26-27 Avril 1978

BIBLIOGRAPHIE

BER 66 BERNAS, CORMARY
'quelques aspects de la programmation aplliquée aux centraux téléphoniques
Commutation électronique n°12, Mars 1966

HAN 72 HANSEN Brinch
' structured multiprogramming'
C.A.C.M. July 1972 volume 15 n°7

C.C.I. C.C.I.T.T.
'C.C.I.T.T. High level language'
Livre bleu

PIM PIMIENTA Daniel
'un langage de description des procédures téléphoniques dans le cadre d'une machine téléphonique'
Thèse présentée à l'IMAN

DIJ 68 DIJKSTRA E.W.
'GO TO Statement considerd Harmful'
C.A.C.M. Volume 11 n°3 March 1968

LAR 74 LARDY D., ROZMARYN C.
'Moduralité et maintenabilité des programmes opérationnels dans les systèmes téléphoniques à programme enregistré
Commutation électronique n°45 Avril 1974

KEV 76 KEVORKIAN K. B., LAGER J.P.
'Les microprocesseurs et l'intégration à grande échelle dans les systèmes téléphoniques à programme enregistré'
Commutation électronique n°54 Juillet

LAM 77 LAMPSON B.W., POPEK G.J., LONDON R.L.
'Notes on the design of EUCLID'
1977

NIK 77 NIKLAUS Wivth
'Modula . a langage for modular multiprogramming'
1977

LUC 77 LUCAS P.
' Les progrès de la commutation électronique dans le monde'
Commutation électronique n°59 octobre 1977

BON 74 BONNARD Patrick, BARBERYE Gérard, MARTIN Michel
'Spécifications du langage PAPE'
Annales des Télécommunications Mars-Avril 1974

CON 75 CONRADI Reidar
'MARY TEXT BOOK'
1975

KAN 76 KANO S., HILLS M.T.
'Programming electronic switching systems'
IEE Telecommunications series 3, 1976

PLEY 78 PLEYBER
'Un langage de description et de programmation de systèmes de conduite de procédés industriels'
Thèse présentée à l'INPG

RAN 76 RANNON
'Programing language for telephon switching systems'
I.R.I.S.A. Publication interne n°44 Juin 1976

RAP 79 ANCEAU F., COURTOIS B., MARINESCU M., NICOLOPOLOS P., OLAIWAN A., PONS J.F., ZAMBRANO A.
'Etude d'une commande répartie à microprocesseurs pour autocommutateurs téléphoniques'
le 27 Mars 1979

WIC 73 WICHMANN B.A., BELL D.A.
'An appraisal of a high-level assembly language'
IFIP/TC-2 Working conference on 'Machine oriented higher level languages' Trondheim, 1973

WIR 74 WIRTH Niklaus, JENSEN Kathleen
'A user manual for PASCAL'
Zurch, April 1974

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU les rapports de présentation de Messieurs :

- F. ANCEAU, Maître de Conférences à l'Institut National Polytechnique de GRENOBLE

- J.M. PITIE, Directeur du groupe Architecture et Fonction de Commande - RCI/PLC - C.N.E.T. - LANNION -

Madame Amal OLAIWAN née EL KARAH

est autorisée à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Génie Informatique".

Grenoble, le 15 Juin 1979

Le Président de l'I.N.P.G.

Ph. TRAYNARD
Président
de l'Institut National Polytechnique