UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

CCITT

COMITÉ CONSULTATIF
INTERNATIONAL
TÉLÉGRAPHIQUE ET TÉLÉPHONIQUE

LIVRE ROUGE

TOME VI – FASCICULE VI.11

LANGAGE DE SPÉCIFICATION ET DE DESCRIPTION FONCTIONNELLES (LDS)

ANNEXES AUX RECOMMANDATIONS Z.100 À Z.104

VIIIe ASSEMBLÉE PLÉNIÈRE
MALAGA-TORREMOLINOS, 8-19 OCTOBRE 1984

Genève 1985

A NOTE FROM ITU LIBRARY & ARCHIVES

Due to technical restrictions, the template of SDL symbols has not been included in the scanned version of this document.

****

En raison de contraintes techniques, le gabarit de symboles du LDS n'a pas été inclus dans la version scannée de ce document.

****

Debido a restricciones de técnicas, la plantilla de símbolos del LED no se ha incluido en la versión escaneada de este documento.

UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

CCITT

COMITÉ CONSULTATIF
INTERNATIONAL
TÉLÉGRAPHIQUE ET TÉLÉPHONIQUE

LIVRE ROUGE

TOME VI – FASCICULE VI.11

LANGAGE DE SPÉCIFICATION ET DE DESCRIPTION FONCTIONNELLES (LDS)

ANNEXES AUX RECOMMANDATIONS Z.100 À Z.104

VIIIe ASSEMBLÉE PLÉNIÈRE
MALAGA-TORREMOLINOS, 8-19 OCTOBRE 1984

Genève 1985

ISBN 92-61-02242-1

CONTENU DU LIVRE DU CCITT EN VIGUEUR APRÈS LA HUITIÈME ASSEMBLÉE PLÉNIÈRE (1984)

LIVRE ROUGE

Tome I – Procès-verbaux et rapports de l'Assemblée plénière.

Vœux et résolutions.

Recommandations sur:

- l'organisation du travail du CCITT (série A);
- les moyens d'expression (série B);
- les statistiques générales des télécommunications (série C).

Liste des Commissions d'études et des Questions mises à l'étude.

Tome II – *(Divisé en 5 fascicules vendus séparément)*

FASCICULE II.1 – Principes généraux de tarification – Taxation et comptabilité dans les services internationaux de télécommunications – Recommandations de la série D (Commission d'études III).

FASCICULE II.2 – Service téléphonique international – Exploitation – Recommandations E.100 à E.323 (Commission d'études II).

FASCICULE II.3 – Service téléphonique international – Gestion du réseau – Ingénierie du trafic – Recommandations E.401 à E.600 (Commission d'études II).

FASCICULE II.4 – Services télégraphiques – Exploitation et qualité de service – Recommandations F.1 à F.150 (Commission d'études I).

FASCICULE II.5 – Services de télématique – Exploitation et qualité de service – Recommandations F.160 à F.350 (Commission d'études I).

Tome III – *(Divisé en 5 fascicules vendus séparément)*

FASCICULE III.1 – Caractéristiques générales des communications et des circuits téléphoniques internationaux – Recommandations G.101 à G.181 (Commissions d'études XV, XVI et CMBD).

FASCICULE III.2 – Systèmes internationaux analogiques à courants porteurs – Caractéristiques des moyens de transmission – Recommandations G.211 à G.652 (Commissions d'études XV et CMBD).

FASCICULE III.3 – Réseaux numériques – Systèmes de transmission et équipement de multiplexage – Recommandations G.700 à G.956 (Commissions d'études XV et XVIII).

FASCICULE III.4 – Utilisation des lignes pour les transmissions des signaux autres que téléphoniques – Transmissions radiophoniques et télévisuelles – Recommandations des séries H et J (Commission d'études XV).

FASCICULE III.5 – Réseau numérique avec intégration des services (RNIS) – Recommandations de la série I (Commission d'études XVIII).

Tome IV – *(Divisé en 4 fascicules vendus séparément)*

FASCICULE IV.1 – Maintenance: principes généraux, systèmes de transmission internationaux, circuits téléphoniques internationaux – Recommandations M.10 à M.762 (Commission d'études IV).

FASCICULE IV.2 – Maintenance des circuits internationaux pour la transmission de télégraphie harmonique ou de télécopie – Maintenance des circuits internationaux loués – Recommandations M.800 à M.1375 (Commission d'études IV).

FASCICULE IV.3 – Maintenance des circuits radiophoniques internationaux et transmissions télévisuelles internationales – Recommandations de la série N (Commission d'études IV).

FASCICULE IV.4 – Spécifications des appareils de mesure – Recommandations de la série O (Commission d'études IV).

Tome V – Qualité de la transmission téléphonique – Recommandations de la série P (Commission d'études XII).

Tome VI – *(Divisé en 13 fascicules vendus séparément)*

FASCICULE VI.1 – Recommandations générales sur la commutation et la signalisation téléphoniques – Interface avec le service maritime et le service mobile terrestre – Recommandations Q.1 à Q.118 *bis* (Commission d'études XI).

FASCICULE VI.2 – Spécifications des Systèmes de signalisation n° 4 et 5 – Recommandations Q.120 à Q.180 (Commission d'études XI).

FASCICULE VI.3 – Spécifications du Système de signalisation n° 6 – Recommandations Q.251 à Q.300 (Commission d'études XI).

FASCICULE VI.4 – Spécifications des Systèmes de signalisation R1 et R2 – Recommandations Q.310 à Q.490 (Commission d'études XI).

FASCICULE VI.5 – Centraux numériques de transit dans les réseaux numériques intégrés et les réseaux mixtes analogiques-numériques. Centraux numériques locaux et mixtes – Recommandations Q.501 à Q.517 (Commission d'études XI).

FASCICULE VI.6 – Interfonctionnement des systèmes de signalisation – Recommandations Q.601 à Q.685 (Commission d'études XI).

FASCICULE VI.7 – Spécifications du Système de signalisation n° 7 – Recommandations Q.701 à Q.714 (Commission d'études XI).

FASCICULE VI.8 – Spécifications du Système de signalisation n° 7 – Recommandations Q.721 à Q.795 (Commission d'études XI).

FASCICULE VI.9 – Système de signalisation avec accès numérique – Recommandations Q.920 à Q.931 (Commission d'études XI).

FASCICULE VI.10 – Langage de spécification et de description fonctionnelles (LDS) – Recommandations Z.101 à Z.104 (Commission d'études XI).

FASCICULE VI.11 – Langage de spécification et de description fonctionnelles (LDS), annexes aux Recommandations Z.101 à Z.104 (Commission d'études XI).

FASCICULE VI.12 – Langage évolué du CCITT (CHILL) – Recommandation Z.200 (Commission d'études XI).

FASCICULE VI.13 – Langage homme-machine (LHM) – Recommandations Z.301 à Z.341 (Commission d'études XI).

Tome VII – *(Divisé en 3 fascicules vendus séparément)*

FASCICULE VII.1 – Transmission télégraphique – Recommandations de la série R (Commission d'études IX). – Equipements terminaux pour les services de télégraphie – Recommandations de la série S (Commission d'études IX).

FASCICULE VII.2 – Commutation télégraphique – Recommandations de la série U (Commission d'études IX).

FASCICULE VII.3 – Equipements terminaux et protocoles pour les services de télématique – Recommandations de la série T (Commission d'études VIII).

Tome VIII – *(Divisé en 7 fascicules vendus séparément)*

FASCICULE VIII.1 – Communication de données sur le réseau téléphonique – Recommandations de la série V (Commission d'études XVII).

FASCICULE VIII.2 – Réseaux de communications de données; services et facilités – Recommandations X.1 à X.15 (Commission d'études VII).

FASCICULE VIII.3 – Réseaux de communications de données; interfaces – Recommandations X.20 à X.32 (Commission d'études VII).

FASCICULE VIII.4 – Réseaux de communications de données; transmission, signalisation et commutation, réseau, maintenance et dispositions administratives – Recommandations X.40 à X.181 (Commission d'études VII).

FASCICULE VIII.5 – Réseaux de communications de données: interconnexion de systèmes ouverts (OSI), techniques de description du système – Recommandations X.200 à X.250 (Commission d'études VII).

FASCICULE VIII.6 – Réseaux de communications de données: interfonctionnement entre réseaux, systèmes mobiles de transmission de données – Recommandations X.300 à X.353 (Commission d'études VII).

FASCICULE VIII.7 – Réseaux de communications de données: systèmes de traitement des messages – Recommandations X.400 à X.430 (Commission d'études VII).

Tome IX – Protection contre les perturbations – Recommandations de la série K (Commission d'études V) – Construction, installation et protection des câbles et autres éléments d'installations extérieures – Recommandations de la série L (Commission d'études VI).

Tome X – *(Divisé en 2 fascicules vendus séparément)*

FASCICULE X.1 – Termes et définitions.

FASCICULE X.2 – Index du Livre rouge.

PAGE INTENTIONALLY LEFT BLANK

PAGE LAISSEE EN BLANC INTENTIONNELLEMENT

TABLE DES MATIÈRES DU FASCICULE VI.11 DU LIVRE ROUGE

Annexes aux Recommandations Z.100 à Z.104

Langage de spécification et de description fonctionnelles (LDS)

NOTES PRÉLIMINAIRES

1 Les questions confiées à chaque Commission d'études pour la période 1985-1988 figurent dans la contributio N° 1 de la Commission correspondante.

2 Dans ce fascicule, l'expression «Administration» est utilisée pour désigner de façon abrégée aussi bien une Administration de télécommunications qu'une exploitation privée reconnue de télécommunications.

3 La Conférence de plénipotentiaires, Nairobi, 1982, a décidé que le terme «Avis» du CCITT et du CCIR devrait être remplacé par le terme «Recommandation» dans les publications de l'UIT. Pour simplifier le traitement des textes du présent Livre, le mot «Avis» avec «A» majuscule a été systématiquement remplacé par le mot «Recommandation»; en conséquence, les Avis des CCI publiés antérieurement au Livre Rouge seront désignés, à partir de maintenant, par le mot «Recommandation».

Annexes aux Recommandations Z.100 à Z.104

LANGAGE DE SPÉCIFICATION ET DE DESCRIPTION FONCTIONNELLES (LDS)

PAGE INTENTIONALLY LEFT BLANK

PAGE LAISSEE EN BLANC INTENTIONNELLEMENT

ANNEXE A

(aux Recommandations Z.100 à Z.104)

Glossaire du LDS

Tout terme figurant dans les Recommandations Z.100 à Z.104 du LDS, qui est imprimé en italique et figure dans le présent glossaire est utilisé strictement dans le sens défini ci-après.

Si une phrase ou une expression imprimée en italique, par exemple *identificateur de procédure*, ne figure pas dans le glossaire, il peut alors s'agir d'un enchaînement de termes et, dans le cas particulier de l'expression composée du mot *identificateur* suivi du mot *procédure*. Lorsqu'un mot est imprimé en italique mais ne se trouve pas dans le glossaire, il peut alors s'agir d'un dérivé d'un terme du glossaire. Par exemple (en anglais), *exported* est le temps passé du verbe *export*.

Après la définition du terme, il existe toujours une référence principale à son utilisation dans les Recommandations de la série Z.100, sauf lorsque ce terme est synonyme d'un autre terme. Ces références sont indiquées entre crochets [ ] après les définitions. Par exemple [Recommandation Z.100, § 3.2] indique que la référence principale se trouve au § 3.2 de la Recommandation Z.100. Etant donné qu'il arrive souvent que ces termes ne soient pas définis, ces références ne constituent qu'une directive.

L'annexe C contient la représentation graphique des symboles et diagrammes, les mots clés utilisés dans la représentation textuelle des expressions et l'utilisation de ces mots clés; ces renseignements ne sont donc pas répétés dans le glossaire.

type de données abstrait

E: abstract data type

Un *type de données abstrait* est un *type* défini par une relation algébrique entre les valeurs du *type* et l'application des *opérateurs* à ces *valeurs* [Recommandation Z.100, § 2.3; Recommandation Z.104, § 1].

syntaxe abstraite

E: abstract syntax

La *syntaxe abstraite* du LDS est donnée dans les Recommandations Z.101, Z.102, Z.103 et Z.104 et elle est résumée dans l'appendice B. Elle a pour objet de décrire la structure conceptuelle d'une définition du *LDS*, par opposition aux *syntaxes concrètes* qui existent pour chaque forme de *LDS*, à savoir: *LDS/GR* (graphique), *LDS/PR* (forme «programme») et *LDS/PE* («pictorial elements», c'est-à-dire éléments graphiques) [Recommandation Z.100, § 3.1].

accès

E: access

L'opérateur *prédéfini* existant implicitement avec tous les types de données, qui s'applique chaque fois qu'une variable donne la valeur associée [Recommandation Z.104, § 4.5].

opérateur actif

E: active operator

Un *opérateur* qui nécessite une ou plusieurs *variables* comme paramètres étant donné qu'il peut changer les *valeurs* associées à ces *variables* [Recommandation Z.104, § 4.5].

paramètre effectif

E: actual parameter

Un *paramètre effectif* est une *valeur* donnée soit à un *processus*, soit à une *procédure* pour le *paramètre formel* correspondant, lorsque le *processus* (ou la *procédure*) est *créé* (ou *appelé* selon les cas) [Recommandation Z.101, § 2.3; Recommandation Z.103, § 2.1].

liste de paramètres effectifs

E: actual parameter list

Une *liste de paramètres effectifs* est la liste des *valeurs* associée à une *demande de création* ou à un *appel de procédure* [Recommandation Z.101, § 2.3; Recommandation Z.103, § 2.1].

ensemble de mise en réserve supplémentaire

E: additional-save-set

Un *ensemble de mise en réserve supplémentaire* est un ensemble de signaux supplémentaires mis en réserve dans une *procédure* [Recommandation Z.103, § 2.1].

symbole d'allocation

E: allocation symbol

Le *symbole* est un *diagramme d'arbre de processus* du *LDS/GR* attaché à un *symbole de processus.* Le *processus* ou *sous-processus* associé au *symbole de processus* est affecté au *bloc* dont le *nom* est contenu dans le *symbole d'allocation* [Recommandation Z.102, § 2.3].

annotation

E: annotation

Dans le *LDS/GR,* une *annotation* est un *commentaire.* Dans le *LDS/PR,* une *annotation* est un *commentaire* ou une *note.* Les *annotations* ne modifient pas la signification du LDS [Recommandation Z.101, § 3.3, 4.3].

arc

E: arc

Un *arc* est une connexion reliant les *nœuds* d'un *graphe de processus* [Recommandation Z.101, § 2.2, 2.1.2].

tableau

E: array

Le générateur prédéfini utilisé pour introduire la notion d'ensembles [Recommandation Z.104, § 5.3].

instruction d'affectation

E: assignment statement

Une *instruction d'affectation* est une *action* qui associe une *valeur* à une *variable* [Recommandation Z.101, § 2.2; Recommandation Z.104, § 4.11].

axiome

E: axiom

Une *expression booléenne* qui reste vraie pour toutes les valeurs possibles des variables utilisées dans l'*axiome* [Recommandation Z.104, § 4.6].

B'

Mot clé du *LDS/PR* qui introduit une dénotation binaire d'une *valeur* numérique [Recommandation Z.104, § 4.8].

LDS de base

E: basic SDL

Sous-ensemble minimum de *LDS* défini dans la Recommandation Z.101.

comportement

E: behaviour

Le *comportement* ou *comportement fonctionnel* d'un système dans le *LDS* est décrit comme des réponses discrètes à des *stimuli* discrets [Recommandation Z.101, § 1].

bloc

E: block

Bloc est synonyme d'*instance de bloc.*

définition de bloc

E: block definition

Une *définition de bloc* définit les propriétés structurelles et de connectivité d'un type de *bloc* nommé [Recommandation Z.100, § 2.2].

diagramme d'interaction de blocs

E: block interaction diagram

Le *diagramme d'interaction de blocs* du *LDS/GR* représente la structure des *blocs*, des *canaux*, des *listes de signaux*, des *listes de données*, des *processus* et de la *circulation des signaux* entre les *processus.* Un *diagramme d'interaction de blocs* peut également montrer la subdivision d'un système en *blocs*, *sous-blocs*, *canaux* et *sous-canaux* [Recommandation Z.101, § 3.1; Recommandation Z.102, § 3.2].

sous-structure de bloc

E: block substructure

Pour un *bloc*, la *sous-structure de bloc* est la *subdivision* du *bloc* en *sous-blocs* et en nouveaux *canaux* à des *niveaux d'abstraction* inférieurs [Recommandation Z.102, § 2.2].

définition de sous-structure de bloc

E: block substructure definition

Une *définition de sous-structure de bloc* fait à titre optionnel partie intégrante d'une *définition de bloc de partie interne* et définit une *sous-structure de bloc* [Recommandation Z.102, § 2.2].

symbole de bloc

E: block symbol

Le *symbole de bloc* représente le concept d'un *bloc fonctionnel* dans un *diagramme d'interaction de blocs* du *LDS/GR* [Recommandation Z.101, § 3.1].

diagramme d'arbre de blocs

E: block tree diagram

Un *diagramme d'arbre de blocs* est la représentation en *LDS/GR* de la subdivision d'un *système* en *blocs* à des *niveaux d'abstraction inférieurs* au moyen d'un diagramme d'arbre inversé [Recommandation Z.102, § 3.1].

booléen

E: boolean

Le *type de données* logique ayant les valeurs TRUE (VRAI) et FALSE (FAUX) et l'ensemble normal des *opérateurs* logiques [Recommandation Z.104, § 5.4].

nœud d'appel

E: call node

Le terme *nœud d'appel* est synonyme de *nœud d'appel de procédure.*

symbole d'appel

E: call symbol

Le terme *symbole d'appel* est synonyme de *symbole d'appel de procédure.*

canal

E: channel

Un *canal* est la classe d'entité qui véhicule des *signaux* d'un *bloc* à un autre. Les *canaux* véhiculent également des signaux à destination et en provenance de l'*environnement.* Les *canaux* sont unidirectionnels, c'est-à-dire qu'ils véhiculent des signaux dans une seule direction [Recommandation Z.101, § 2.1].

définition de canal

E: channel definition

Une *définition de canal* définit les propriétés d'un *canal* nommé. Ces propriétés sont le *bloc* d'origine, le *bloc* de destination, l'ensemble des *signaux* que le *canal* peut véhiculer et, à titre optionnel, une *définition de sous-structure de canal.* Le *bloc* de destination peut être l'*environnement du système* [Recommandation Z.101, § 2.2; Recommandation Z.102, § 2.2].

sous-structure de canal

E: channel substructure

Une *sous-structure de canal* est une *subdivision* d'un *canal* en un ensemble de *canaux* et de *blocs* à un *niveau d'abstraction* inférieur [Recommandation Z.102, § 2.2].

définition de sous-structure de canal

E: channel substructure definition

Une *définition de sous-structure de canal* est une partie facultative d'une *définition de canal* qui définit la *sous-structure du canal* [Recommandation Z.102, § 2.2].

diagramme de sous-structure de canal

E: channel substructure diagram

Un *diagramme de sous-structure de canal* est la représentation en *LDS/GR* d'une *sous-structure de canal* [Recommandation Z.102, § 3.4].

symbole de canal

E: channel symbol

Symbole qui, dans un *diagramme d'interaction de blocs* du *LDS/GR*, représente un *canal* ou un *sous-canal.* La pointe de flèche du symbole est orientée vers le *bloc* de destination, l'autre extrémité du symbole indique le *bloc* d'origine. Le groupe de *signaux* transportés par le *canal* peut être représenté par le *symbole de liste de signaux* associé [Recommandation Z.101, § 3.1].

caractère

E: character

Le *type de données prédéfini* pour lequel les *littéraux* sont les *littéraux de chaîne de caractères* de longueur 1 et les opérateurs sont égal, non égal et concaténation pour former une *valeur de chaîne de caractères* [Recommandation Z.104, § 5.5].

élément graphique de chargement en cours

E: charging in progress PE

Un *élément graphique* indiquant que le chargement est en cours [Recommandation Z.103, § 6.1].

chaîne de caractères

E: charstring

Le *type de données prédéfini* pour lequel les littéraux sont les *chaînes* de caractères de l'Alphabet n° 5 du CCITT et les *opérateurs* sont ceux du *générateur prédéfini de chaîne* instantié pour les *caractères* [Recommandation Z.104, § 5.13].

élément graphique d'émetteur et de récepteur de signalisation combinée

E: combined signalling sender and receiver PE

Un *élément graphique* correspondant à un émetteur et à un récepteur de signalisation combinée [Recommandation Z.103, § 6.1].

commentaire

E: comment

Information qui s'ajoute à un diagramme *LDS* ou qui l'explique. En *LDS/GR*, les *commentaires* peuvent être assemblés au moyen d'un simple crochet relié par une ligne pointillée à une *ligne de liaison* d'un *diagramme de processus* ou à tout symbole concerné. En *LDS/GR*, les *commentaires* sont introduits par le mot clé *COMMENT* [Recommandation Z.101, § 3.3].

opérations composites

E: composite operations

Abréviation représentant une combinaison de concepts *LDS* primitifs. Toute *opération composite* peut être représentée systématiquement au moyen des concepts de la *syntaxe abstraite* du *LDS* [Recommandation Z.103, § 3].

forme syntaxique concrète

E: concrete syntactical form

Le *LDS/GR*, *LDS/PR* et *LDS/PE* sont les *formes syntaxiques concrètes* du *LDS* qui peuvent être mises en correspondance si on considère le graphe *LDS* mathématique conceptuel sous-jacent [Recommandation Z.100, § 3.1].

syntaxe concrète

E: concrete syntax

La *syntaxe concrète* pour les diverses représentations de *LDS* est constituée par les symboles effectivement utilisés pour représenter le *LDS* sous forme de *LDS/GR*, de *LDS/PR* ou de *LDS/PE* [Recommandation Z.100, § 3.1].

expression conditionnelle

E: conditional expression

Une *expression* avec une *expression booléenne* après IF indiquant si l'expression de conséquence qui se trouve après THEN ou l'expression d'*alternative* après ELSE est interprétée [Recommandation Z.104, § 4.10].

élément graphique de chemin de commutation connecté

E: connected switching path PE

Un *élément graphique* indiquant la connectivité entre l'équipement terminal et/ou les dispositifs de signalisation [Recommandation Z.103, § 6.1].

connecteur

E: connector

Un *connecteur* est un *symbole* utilisé en *LDS/GR*. Un *connecteur* (un cercle) est un *connecteur d'entrée* ou un *connecteur de sortie*. Une *ligne de liaison* peut être interrompue par une paire de *connecteurs associés*, la circulation des signaux étant censée partir du *connecteur de sortie* pour aboutir au *connecteur d'entrée* associé [Recommandation Z.101, § 3.1].

valeur constante

E: constant value

Un *littéral* ou un *synonyme* [Recommandation Z.104, § 2.2].

expression constante

E: constant expression

Une *valeur constante* ou un *opérateur* ne comportant que des *expressions constantes* comme *paramètres* [Recommandation Z.104, § 2.2].

signal continu

E: continuous signal

Un *signal continu* est une *opération composite* contenant une *condition.* Lorsque la *condition* se réalise, on quitte l'*état* auquel le *signal continu* est attaché [Recommandation Z.103, § 3.3].

convergence

E: convergence

Dans un *diagramme de processus* du *LDS/GR,* lorsque deux ou plusieurs *symboles* sont suivis d'un seul *symbole,* les *lignes de liaison* aboutissant à ce symbole convergent. Cette convergence peut se traduire par une *ligne de liaison* qui en rejoint une autre ou par plusieurs *connecteurs de sortie* associés à un seul *connecteur d'entrée* ou encore par des *lignes de liaison* distinctes conduisant au même *symbole* [Recommandation Z.101, § 3.3].

action de demande de création

E: create request action

Action qui provoque la création et le départ d'une nouvelle *instance de processus,* à partir d'une *définition de processus* spécifiée [Recommandation Z.101, § 2.3].

nœud de demande de création

E: create request node

Nœud se trouvant dans une *transition,* un *graphe de processus* ou un *graphe de procédure* lors d'une *action de demande de création* [Recommandation Z.101, § 2.2].

symbole de création

E: create request symbol

Symbole, dans un *diagramme d'interaction de blocs* du *LDS/GR,* reliant le *symbole de processus* du *processus créateur* au *symbole de processus* du *processus créé.* La pointe de la flèche identifie le *processus descendant* (*offspring*) tandis que l'autre extrémité du symbole de demande de création identifie le *processus ascendant (parent*) [Recommandation Z.101, § 3.1].

symbole de demande de création

E: create request symbol

Symbole, dans un *diagramme de processus* du *LDS/GR,* représentant une *demande de création* [Recommandation Z.101, § 3.3.]

D'

Mot clé du *LDS/PR* introduisant la notation décimale d'une *valeur* numérique, la notation décimale étant la notation prise par défaut. L'utilisation de *D'* est optionnelle [Recommandation Z.104, § 4.8].

données

E: data

Le terme *données* est synonyme d'*objets(s) de donnée(s)* (ou *information*).

objet de données

E: data item

Un *objet de données* est une *variable* ou une *valeur.*

type de donnée

E: data type

Un *type de données* d'*objet(s) de donnée(s)* détermine *l'intervalle,* la signification des *valeurs* comprises dans cet *intervalle* et l'ensemble des *opérateurs* valides qui peut être utilisé avec les objets de ce *type de données.* (Voir également *type de données prédéfini*) [Recommandation Z.104, § 1].

définition de type de donnée

E: data type definition

Définit le *nom,* les *valeurs de données* et les *opérateurs* d'un *type de données.* [Recommandation Z.104, § 2.2].

décision

E: decision

Une *décision* est une *action* qui se produit à un *nœud de transition,* en cours de *transition,* et qui consiste à poser une question dont la réponse peut être obtenue à ce moment et détermine le choix entre plusieurs *arcs* sortant du *nœud,* pour achever l'exécution de la transition [Recommandation Z.101, § 2.3].

nœud de décision

E: decision node

Un *nœud de décision* est un *nœud* situé dans une *transition,* dans un *graphe de processus* ou un *graphe de procédure* où une décision se manifeste [Recommandation Z.101, § 2.2].

nom de décision

E: decision name

Un *nom de décision* est le *nom* associé à une *décision.* Le nom d'une *décision* découle d'une question qui doit être clairement comprise par l'interprète. Les *noms* des *arcs de décision* partant d'une *décision* doivent constituer tous les aboutissements possibles à partir de la question [Recommandation Z.101, § 2.3].

symbole de décision

E: decision symbol

Symbole représentant le concept LDS d'une décision dans un *diagramme de processus* du *LDS/GR* [Recommandation Z.101, § 3.2].

déclare!

E: declare!

L'opérateur *prédéfini* existant implicitement avec tous les types de données associés à l'instantiation des *instances* de *variables* [Recommandation Z.104, § 4.5].

définition

E: definition

Le terme *définition* est synonyme de *définition de type.*

description

E: description

La mise en œuvre des caractéristiques propres à un système fait l'objet d'une description du système. Les descriptions se composent des *paramètres* du système, tel qu'il est réalisé, et de la *description fonctionnelle* (DF) de son fonctionnement effectif [Recommandation Z.100, § 1.1].

divergence

E: divergence

En *LDS/GR*, lorsqu'un *symbole* est suivi de deux *symboles* ou plus, une *ligne de liaison* partant de ce symbole peut diverger en deux *lignes de liaison* ou plus [Recommandation Z.101, § 3.3].

durée

E: duration

Le *type de données prédéfini* représentant l'intervalle entre deux instants du temps [Recommandation Z.104, § 5.11].

condition de validation

E: enabling condition

La *condition de validation* est une *opération composite* constituant une méthode permettant d'accepter conditionnellement l'*entrée* d'un signal ou sa *mise en réserve*, à une *condition* déterminée [Recommandation Z.103, § 3.2].

symbole de condition de validation

E: enabling condition symbol

Symbole, dans un *diagramme de processus* ou dans un *diagramme de procédure* du *LDS/GR*, représentant une *condition de validation* (lorsqu'il suit un *symbole d'entrée*) ou un *signal continu* (lorsqu'il suit un *symbole d'état*) [Recommandation Z.103, § 3.2].

environnement

E: environment

Le terme *environnement* est synonyme d'*environnement d'un système.* [Recommandation Z.101, § 2.1].

symbole d'environnement

E: environment symbol

Symbole représentant, dans un *diagramme de processus* du *LDS/GR*, l'*environnement d'un système* [Recommandation Z.101, § 3.1.1, 3.1.2].

environnement d'un système

E: environment of a system

L'*environnement d'un système* est une partie du système dont le comportement n'est pas indiqué dans le LDS mais qui réagit avec le reste du système en envoyant des *instances de signal* au *système* [Recommandation Z.101, § 2.1].

comportement équivalent

E: equivalent behaviour

Le terme *comportement équivalent* est synonyme de *comportement fonctionnel équivalent* ou *fonctionnement équivalent.*

comportement fonctionnel équivalent/fonctionnement équivalent

E: quivalent functional behaviour

Deux *systèmes* (ou *blocs* ou *processus*) ont un *comportement fonctionnel équivalent* s'ils donnent la même réponse à une séquence donnée d'*instances de signal* vue de l'extérieur de ces *systèmes* (ou *blocs* ou *processus*) [Recommandation Z.100, § 1.1].

EXPORT

E: EXPORT

L'élément syntaxique *EXPORT* (nom de variable) est utilisé dans le *LDS/GR* et le *LDS/PR* pour représenter l'*exportation* de la *valeur* d'une *variable* [Recommandation Z.103, § 3.1].

export

E: export

Le terme *export* est synonyme d'*opération d'exportation.*

EXPORTED

E: EXPORTED

Mot clé du *LDS/PR* indiquant, dans une *définition de variable*, que la *variable* est *exportée* [Recommandation Z.103, § 3.1].

exportateur

E: exporter

L'*exportateur* d'une *variable* est l'*instance de processus* à laquelle appartient la *variable* dont la *valeur* est *exportée* [Recommandation Z.103, § 3.1].

opération d'exportation

E: export operation

Une *opération d'exportation* est l'*opération composite* qui permet à un processus d'*exporter* la ou les *valeurs* des *données*, de manière qu'un autre processus puisse accéder aux *valeurs* en question [Recommandation Z.103, § 3.1].

expression

E: expression

Une *expression* peut être: un *nom de valeur*, un *nom de synonyme*, un *nom de variable*, une *expression conditionnelle* ou une *opération* sur une ou plusieurs *expressions* [Recommandation Z.104, § 4.7].

extract!

E: extract!

L'*opérateur* qui est sous-entendu en dehors des *axiomes* lorsqu'une *variable* est immédiatement suivie d'*expressions* entre parenthèses (sauf si elle est suivie de : = ; dans ce cas, il faut sous-entendre *insert!* [Recommandation Z.104, § 4.5].

ligne de liaison

E: flow line

Une *ligne de liaison* relie chaque *symbole* au(x) *symbole(s)* qui le suit (suivent), dans un *diagramme de processus* du *LDS/GR* [Recommandation Z.101, § 3.3].

paramètre formel

E: formal parameter

Un *paramètre formel* est un *nom de variable*, contenu dans une *définition de processus* ou une *définition de procédure*, pour lesquels des *paramètres effectifs* sont *affectés* aux *variables*, lorsque le *processus* est créé ou lorsque la *procédure* est appelée [Recommandation Z.101, § 2.2; Recommandation Z.103, § 2.1].

liste de paramètres formels

E: formal parameter list

Liste de paramètres formels dans une *définition de processus* ou dans une *définition de procédure.* Les *paramètres effectifs* sont repérés en fonction de la position qu'ils occupent dans leurs listes respectives [Recommandation Z.101, § 2.2; Recommandation Z.103, § 2.1].

cadre (1)

E: frame (1)

Dans un *diagramme d'interaction de blocs* du *LDS/GR*, un *cadre* représente le *bloc* dont la *subdivision* est définie par le *diagramme d'interaction de blocs* [Recommandation Z.102, § 3.2].

cadre (2)

E: frame (2)

Dans un *diagramme de sous-structure de canal* du *LDS/GR*, un *cadre* représente le *canal* dont la *subdivision* est définie par le *diagramme de sous-structure de canal* [Recommandation Z.102, § 3.4].

comportement fonctionnel ou fonctionnement

E: functional behaviour

(Voir: *comportement.*)

(See: *behaviour.*)

bloc fonctionnel

E: functional block

Un *bloc fonctionnel* est un objet de dimensions commodes, caractérisé par des relations de logique interne bien déterminées, qui a un nom, un ensemble de *canaux* le reliant à d'autres *blocs* et des *processus* internes ou une *définition de bloc* [Recommandation Z.100, § 2.1; Recommandation Z.102, § 2.1].

paramètres généraux

E: general parameters

Dans une *spécification* et dans une *description* d'un système, les *paramètres généraux* se rapportent à des informations telles que: limites de température, construction, capacité d'échange, qualité de service, etc. [Recommandation Z.100, § 1.1].

générateur

E: generator

Le terme *générateur* est synonyme de *générateur de type de données.*

syntaxe graphique

E: graphic syntax

(Voir: *LDS/GR*).

H'

Mot clé du *LDS/PR* introduisant une notation hexadécimale d'une *valeur* numérique [Recommandation Z.104, § 4.8].

identificateur

E: identifier

Un *identificateur* est un *nom* unique pour un *type* ou une *instance* de *type*; il est formé d'une *partie qualificatif* et d'une *partie nom* [Recommandation Z.100, § 2.1].

transition implicite

E: implicit transition

Si, dans un diagramme de processus du *LDS/GR*, on ne trouve ni *symboles d'entrée* explicites, ni *symboles de mise en réserve* explicites, liés à un *symbole d'état* pour l'un des *signaux entrants valides*, il y a alors une *transition implicite* qui est un *nœud d'entrée* connecté directement en retour au même état. En conséquence, ces signaux sont mis au rebut [Recommandation Z.101, § 3.3].

IMPORT

E: IMPORT

La construction syntaxique *IMPORT* (nom de variable, instance de processus) est utilisée dans le *LDS/GR* et le *LDS/PR*, pour représenter l'*importation* de la *valeur* d'une *variable* en provenance d'une *instance de processus* [Recommandation Z.103, § 3.1].

import

E: import

Le terme *import* est synonyme d'*opération d'importation.*

IMPORTED

E: IMPORTED

Mot clé du *LDS/PR* indiquant, dans une *définition de variable,* que la *variable* est utilisée pour contenir une *valeur* qui est importée [Recommandation Z.103, § 3.1].

importateur

E: importer

L'*importateur* d'une *valeur importée* est l'*instance de processus* qui *importe* la *valeur* [Recommandation Z.103, § 3.1].

opération d'importation

E: import operation

Une *opération d'importation* est l'*opération composite* qui permet à un processus d'*importer* la (ou les) *valeur(s)* des *variables* appartenant à un autre *processus* et d'y accéder [Recommandation Z.103, § 3.1].

valeur importée

E: imported value

Valeur vue par un *processus* pour un *élément de données* qui est *importé* [Recommandation Z.103, § 3.1].

IN

E: IN

Attribut de *paramètre formel* dénotant le cas dans lequel une *valeur* est transférée à une *procédure* [Recommandation Z.103, § 2.3].

IN/OUT

E: IN/OUT

Attribut de *paramètre formel* dénotant le cas dans lequel un *nom de paramètre formel* est utilisé comme *synonyme* pour la *variable* [Recommandation Z.103, § 2.3].

connecteur d'entrée

E: in-connector

Une *ligne de liaison* peut être interrompue par une paire de *connecteurs associés,* la circulation de l'information étant censée partir du *connecteur de sortie* pour aboutir au *connecteur d'entrée* associé [Recommandation Z.101, § 3.3].

canal d'arrivée

E: incoming channel

Un *canal d'arrivée* est un nouveau *canal* qui se forme lorsqu'un *canal* est *subdivisé.* Un *canal d'arrivée* véhicule vers les nouveaux *blocs* formés par la *subdivision de canal* tous les *signaux* transmis par le *canal subdivisé* [Recommandation Z.102, § 2.1].

opérateur infixe

E: infix operator

Un des *opérateurs* dyadiques prédéfinis du *LDS/DR* (= > OR XOR AND IN/= = > < <= >= + − // # / MOD REM) qui sont placés entre les deux paramètres plutôt que devant les paramètres entre parenthèses, ou un des *opérateurs* unaires *préfixés* (+ − NOT) [Recommandation Z.104, § 4.5].

accès entrant

E: inlet

Un *accès entrant* d'une *macro* est le point par lequel une ligne pénètre dans la *macro* [Recommandation Z.103, § 4.1].

entrée

E: input

Le terme *entrée* est, par lui-même, synonyme d'*action d'entrée.*

action d'entrée

E: input action

Une *action d'entrée* est une *action* qui reçoit et absorbe un *signal d'entrée* correspondant à un *nom de signal* et permet à l'*instance de processus* d'interpréter l'*action d'entrée* pour accéder à l'information contenue dans le *signal d'entrée* [Recommandation Z.101, § 2.3].

nœud d'entrée

E: input node

Un *nœud d'entrée* est un *nœud* contenu dans un *graphe de processus* ou dans un *graphe de procédure,* lorsqu'une *action d'entrée* se produit et a le même *nom* que le *signal* que l'*action d'entrée* absorbe [Recommandation Z.101, § 2.2].

port d'entrée

E: input port

Le *port d'entrée* d'un *processus* reçoit les *signaux* dans l'ordre de leur arrivée et les retient jusqu'au moment où ces *signaux* sont consommés par une *action d'entrée* [Recommandation Z.101, § 2.3].

signal d'entrée

E: input signal

Un *signal d'entrée* d'un *processus* est l'un des *signaux* nommés de l'ensemble des signaux que le *processus* peut recevoir dans n'importe lequel de ses *nœuds d'état.* L'ensemble des *signaux d'entrée* valides qu'un processus peut recevoir correspond à l'ensemble de tous les *noms de signaux* qui apparaissent dans n'importe quel *nœud d'entrée* du processus [Recommandation Z.101, § 2.3].

symbole d'entrée

E: input symbol

Les symboles, dans un *diagramme de processus,* en *LDS/GR* représentant le concept LDS d'une *entrée* [Recommandation Z.101, § 3.3].

insérer!

E: insert

L'*opérateur* qui est sous-entendu à l'extérieur des *axiomes* lorsqu'une *variable* est immédiatement suivie d'*expressions* entre parenthèses, de ce fait := [Recommandation Z.104, § 4.5].

instance

E: instance

L'*instance* d'un *type* est une valeur qui a toutes les propriétés de ce *type* et peut être distinguée de toutes les autres *instances* du même type par un *identificateur* [Recommandation Z.100, § 2.1].

instantiation

E: instantiation

L'*instantiation* est la création d'une *instance* d'une entité à partir d'un *type* particulier [Recommandation Z.100, § 2.1].

entier

E: integer

Le *type* d'*entiers* est défini par l'ensemble des valeurs comprises entre −infini, . .., −2, −1, 0, +1, +2, . . ., +infini et leurs opérations mathématiques normales, c'est-à-dire: addition, soustraction, multiplication, division, etc. [Recommandation Z.104, § 5.1].

étiquette

E: label

Une *étiquette* est un *nom* facultativement attaché à un *accès entrant* ou à un *accès sortant* d'une *macro* [Recommandation Z.103, 4.2].

niveau

E: level

Le terme *niveau* est synonyme de *niveau d'abstraction.*

niveau d'abstraction

E: level of abstraction

Un *niveau d'abstraction* est l'un des niveaux d'un *diagramme d'arbre de blocs.* Une description d'un *système* est un *bloc* au niveau d'abstraction le plus élevé; il est représenté comme un bloc unique au sommet d'un *diagramme d'arbre de blocs* [Recommandation Z.102, § 3.1].

littéral

E: literal

Un *littéral* désigne une *identité de valeur.* Les littéraux 12, B'1100, 0'14 et M'C désignent tous la même *identité de valeur de nombre entier* [Recommandation Z.104, § 4.4].

macro

E: macro

Une *macro* est une collection nommée d'objets syntaxiques définis par l'usager du LDS, qui remplace l'utilisation du *nom du macro* AVANT que l'interprétation de la représentation de la macro dans le LDS ne soit considérée [Recommandation Z.103, § 4].

définition de macro

E: macro definition

Dans le *LDS/GR,* une *définition de macro* est une partie nommée d'un diagramme *LDS/GR* comportant des *accès entrants* et des *accès sortants* représentés par des lignes de liaison qui conduisent au symbole (diagramme de partie) ou qui en partent. Des *étiquettes* peuvent être attachées à ces lignes de liaison [Recommandation Z.103, § 4.2].

symbole de macro

E: macro symbol

Symbole utilisé en *LDS/GR* pour indiquer une référence à une *définition de macro* par son nom [Recommandation Z.103, § 4.2].

nom

E: name

Le terme *nom* est synonyme de *partie 'nom'* d'un *identificateur.*

partie nom

E: name part

La *partie nom* d'un *identificateur* est une phrase significative du langage naturel qui peut être utilisée en combinaison avec une *partie qualificatif* de l'*identificateur,* pour identifier un *type* ou une *instance* d'une entité [Recommandation Z.100, § 2.1].

nombre naturel

E: natural number

Les *nombres naturels* sont les nombres entiers dont les *valeurs* sont comprises entre zéro et l'infini [Recommandation Z.104, § 5.6].

nouveau type (new type)

E: new type

Un *nouveau type* introduit des ensembles de *littéraux* et d'*opérateurs* qui diffèrent de tout autre *littéral* ou *opérateur* (même s'ils ont les mêmes noms, de sorte que les différentes identités doivent être distinguées par des restrictions). Les propriétés d'un nouveau type sont définies par l'utilisation de *littéraux* et d'*opérateurs* dans les *axiomes* [Recommandation Z.104, § 4.3].

node

E: node

Dans un *graphe de processus*, un *nœud* est un emplacement nommé relié aux autres *nœuds* par des *arcs*. Dans le *LDS*, les catégories de *nœuds* sont les suivantes: *nœuds d'état, nœuds d'entrée, nœuds de tâche, nœuds de sortie, nœuds de décision, nœuds de départ, nœuds d'arrêt, nœuds de départ de procédure, nœuds d'appel de procédure, nœuds de retour de procédure* et *nœuds de demande de création* [Recommandation Z.101, § 2.2].

note

E: note

Annotation en *LDS/PR* qui n'est pertinente que pour la représentation du *LDS/PR*. Une *note* est une chaîne de texte(s) entourée par les signes /* et */ [Recommandation Z.101, § 4.3].

O'

Mot clé du *LDS/PR* introduisant une notation octale d'une *valeur* numérique [Recommandation Z.104, § 4.8].

DESCENDANT (OFFSPRING)

E: OFFSPRING

Le *DESCENDANT (OFFSPRING)* d'un *processus* créateur est un *élément de données* qui a la même *valeur* que l'*élément de données SELF* du *processus* le plus récemment créé par ce *processus* créateur. Si un *processus* n'a pas créé de *processus*, son *élément de données descendant* est *indéfini* [Recommandation Z.101, § 2.3].

opérateur

E: operator

Un *opérateur*, appliqué à une ou plusieurs *valeurs*, donne une valeur déterminée par l'utilisation de l'*opérateur* dans les *axiomes*. Les symboles + − * / sont des opérateurs arithmétiques [Recommandation Z.104, § 4.5].

typage d'opérateur

E: operator typing

Définit les types de données des *éléments de données* auxquels s'applique l'opérateur et le *type* de données de la *valeur* résultante (le cas échéant) [Recommandation Z.104, § 4.5].

option

E: option

Une *option* est un élément de *syntaxe concrète* dans une *définition de processus* permettant que le choix de différents *comportements* puisse être effectué AVANT l'interprétation du *graphe de processus* [Recommandation Z.103, § 5.1].

expression d'option

E: option expression

Expression contenue dans une *option* qui est évaluée dans le but de déterminer le *comportement* à choisir [Recommandation Z.103, § 5.2].

symbole d'option

E: option symbol

Symbole sur un *diagramme de processus* ou un *diagramme de procédure* du *LDS/GR* représentant une *option* [Recommandation Z.103, § 5.2].

connecteur de sortie

E: out-connector

Une *ligne de liaison* peut être interrompue par une paire de *connecteurs associés*, la circulation de l'information étant censée partir du *connecteur de sortie* pour aboutir au *connecteur d'entrée* associé [Recommandation Z.101, § 3.3].

canal de départ

E: outgoing channel

Un *canal de départ* est un nouveau *canal* formé lorsqu'un *canal* est *subdivisé*. Un *canal de départ* véhicule, à partir des nouveaux *blocs* formés par la *subdivision de canal*, tous les *signaux* que véhicule le *canal subdivisé* [Recommandation Z.102, § 2.1].

accès sortant

E: outlet

Un *accès sortant* d'une *macro* est le point par lequel une ligne de liaison quitte la *macro* [Recommandation Z.103, § 4.2].

sortie

E: output

Le terme *sortie* est, par lui-même, synonyme d'*action de sortie*.

action de sortie

E: output action

Une sortie est une *action* accomplie dans une *transition* qui engendre un *signal*, lequel agit à son tour, ailleurs, comme un *signal entrant* [Recommandation Z.101, § 2.3].

nœud de sortie

E: output node

Nœud placé sur un *graphe de processus*, à l'endroit où se produit une *action de sortie* qui a le même nom que le *signal* qu'elle engendre [Recommandation Z.101, § 2.2].

symbole de sortie

E: output symbol

Un *symbole de sortie* dans un *diagramme de processus* du *LDS/GR*, représentant le concept LDS d'une *sortie* [Recommandation Z.101, § 3.3].

ASCENDANT (PARENT)

E: PARENT

L'*élément de données ASCENDANT (PARENT)* d'un *processus* est l'*élément de données SELF* de son *processus* ascendant, c'est-à-dire: le *processus* qui a interprété l'*action de demande de création* qui a initialisé le *processus* [Recommandation Z.101, § 2.3].

subdivision

E: partitioning

Une *subdivision* est l'élaboration du fonctionnement de systèmes complexes et/ou vastes en sous-systèmes, afin d'assurer une subdivision logique du comportement du système et des différents aspects abstraits de ce même système [Recommandation Z.102, § 2.1].

opérateur passif

E: passive operator

Un *opérateur* qui a uniquement besoin des *valeurs* comme paramètres et produit donc une *valeur.* Un *opérateur passif* ne peut modifier les *valeurs* associées aux *variables* [Recommandation Z.104, § 4.5].

élément graphique (PE)

E: pictorial element (PE)

Elément d'un ensemble de formes graphiques normalisées utilisé, en *LDS/PE,* dans *les illustrations d'état,* pour représenter les concepts du système de commutation [Recommandation Z.103, § 6].

Pid

E: Pid

Le type de données prédéfini utilisé pour identifier les *instances de processus* [Recommandation Z.104, § 5.8].

type de données prédéfini

E: predefined data type

Pour simplifier la description, le terme *type de données prédéfini,* s'applique à la fois aux *noms* prédéfinis pour les *types* de données et aux *noms* prédéfinis pour les *générateurs de type* de données. *Booléen, caractère, chaîne de caractères, durée, nombre entier, nombre naturel, Pid, réel, temps* et *temporisateur* sont des *noms de type de données* qui sont prédéfinis. *Tableau, mode ensembliste* et *chaîne* sont des *noms de générateur de type de données* qui sont prédéfinis [Recommandation Z.104, § 5].

mode ensembliste

E: powerset

Le *générateur de type de données prédéfini* engendre des *types de données* avec des *valeurs* qui sont des ensembles de *valeurs* ayant un agencement mathématique. Chaque *valeur* du *mode ensembliste* est un ensemble de *valeurs* du *type de données* utilisé pour calculer les paramètres du *générateur du mode ensembliste* [Recommandation Z.104, § 5.7].

procédure

E: procedure

Section d'un *graphe de processus* pouvant être considérée isolément. Une *procédure* est définie en un point unique mais il est possible de s'y référer plusieurs fois, même dans des *processus* différents. Les *signaux* et les *variables* rendus effectifs par interprétation d'une *procédure* sont contrôlés par le passage des paramètres [Recommandation Z.103, § 2].

appel de procédure

E: procedure call

Un *appel de procédure* est un moyen d'appeler une *procédure nommée* pour l'interprétation de la *procédure* et le transfert des paramètres à celle-ci [Recommandation Z.103, § 2.1].

nœud d'appel de procédure

E: procedure call node

Un *nœud d'appel de procédure* est un *nœud* contenu dans un *graphe de processus* ou dans un *graphe de procédure,* à l'endroit où a lieu un *appel de procédure* [Recommandation Z.103, § 2.1].

symbole d'appel de procédure

E: procedure call symbol

Symbole du *LDS/GR* représentant un *appel de procédure* [Recommandation Z.103, § 2.2].

définition de procédure

E: procedure definition

Une *définition de procédure* définit une section d'un *graphe de processus.* La définition associe le *graphe de procédure* à un *nom de procédure,* une *liste de paramètres formels,* un *ensemble de mise en réserve supplémentaire* et, à titre optionnel, à d'autres *définitions de procédure* et *définitions de données* [Recommandation Z.103, § 2.1].

diagramme de procédure

E: procedure diagram

Un *diagramme de procédure* est la représentation en *LDS/GR* d'un *graphe de procédure* [Recommandation Z.103, § 2.2].

graphe de procédure

E: procedure graph

Un *graphe de procédure* est un graphe dont les nœuds sont reliés entre eux par des *arcs* orientés, dans le but de décrire le *comportement* d'une *procédure* et qui peut former une section d'un *graphe de processus* [Recommandation Z.103, § 2.1].

retour de procédure

E: procedure return

(Voir *retour (return).*)

nœud de départ de procédure

E: procedure start node

Le *nœud de départ de procédure* est un *nœud* contenu dans un *graphe de procédure,* lorsque l'interprétation de la *procédure* commence par un appel de la *procédure* [Recommandation Z.103, § 2.1].

symbole de départ de procédure

E: procedure start symbol

Le *symbole* est un *diagramme de procédure* du *LDS/GR* représentant un *nœud de départ de procédure* [Recommandation Z.103, § 2.2].

processus

E: process

Un *processus* remplit une fonction qui exige divers objets informatifs, en vue de l'accomplissement de sous-fonctions dont l'exécution dépend de l'ordre chronologique dans lequel l'information devient accessible au *processus.* Dans le contexte du LDS, un processus est constitué par une série d'entités dont les instances peuvent se trouver soit dans un *état* d'attente d'une *entrée,* soit dans une *transition.* Le terme *processus* est, par lui-même, synonyme d'*instance de processus* [Recommandation Z.101, § 2.1].

définition de processus

E: process definition

Le déroulement d'un *type* du *processus* de classe est décrit dans une *définition de processus,* au moyen des éléments d'un graphe fermé et orienté, comportant les éléments suivants: *entrées, mises en réserve, états, transitions, décisions, tâches* et *sorties* [Recommandation Z.101, § 2.1].

diagramme de processus

E: process diagram

Un *diagramme de processus* est la représentation en *LDS/GR* d'un *graphe de processus* [Recommandation Z.101, § 3.2].

graphe de processus

E: process graph

Un *graphe de processus* est un graphe connecté par des *arcs* orientés servant à décrire le déroulement d'un *processus* [Recommandation Z.101, § 2.2].

instance de processus

E: process instance

Une *instance de processus* est une *instance* d'un *processus* créée dynamiquement [Recommandation Z.101, § 2.3].

sous-structure de processus

E: process substructure

La *sous-structure de processus* d'un *processus* est la *subdivision* du *processus* en *sous-processus* et la répartition de ces *sous-processus* dans des *sous-blocs* [Recommandation Z.102, § 2.2].

définition de sous-structure de processus

E: process substructure definition

Une *définition de sous-structure de processus* est une partie optionnelle d'une *définition de processus* qui définit la *sous-structure de processus* d'un *processus* [Recommandation Z.102, § 2.2].

symbole de processus

E: process symbol

Symbole dans un *diagramme d'interaction de blocs* ou un *diagramme d'arbre de processus* du *LDS/GR* représentant zéro ou plusieurs *instances de processus.* Le *symbole de processus* contient le *nom de processus,* qui identifie la *définition de processus* et une *liste de paramètres formels* [Recommandation Z.101, § 3.1; Recommandation Z.102, § 3.3].

diagramme d'arbre de processus

E: process tree diagram

Un *diagramme d'arbre de processus* du *LDS/GR* représente la *subdivision* d'un *processus* contenu dans un *bloc* en *sous-processus.* La répartition de ces *sous-processus* dans des *sous-blocs* est indiquée par les *symboles d'allocation* figurant dans le *diagramme de processus.* Ce diagramme a la forme d'un arbre inversé [Recommandation Z.102, § 3.3].

qualificatif

E: qualifier

Le terme *qualificatif* est synonyme de *partie qualificatif* d'un *identificateur.*

partie qualificatif

E: qualifying part

La *partie qualificatif* d'un *identificateur* est l'information qui doit être ajoutée à la *partie nom* de l'*identificateur* pour former un *nom* unique. La *partie qualificatif* d'un *identificateur* peut être déduite du contexte d'utilisation de la *partie nom* [Recommandation Z.100, § 2.1].

réel

E: real

Le *type réel* est défini par l'ensemble de TOUTES les valeurs comprises entre $-\infty$ et $+\infty$, ainsi que les opérations mathématiques normales, c'est-à-dire $+$, $-$, multiplication, élever à une puissance, etc [Recommandation Z.104, § 5.2].

élément graphique de chemin de commutation réservé

E: reserved switching path PE

Un *élément graphique* représentant une connexion réservée entre un équipement terminal et/ou des dispositifs de signalisation [Recommandation Z.103, § 6.1].

RESET

E: RESET

L'*opérateur défini pour le type de données temporisateur* qui permet la libération des temporisateurs [Recommandation Z.104, § 5.12].

signal retenu

E: retained signal

Lorsqu'un *signal* atteint un *processus*, on considère qu'il est *reçu* et *retenu* pour ce *processus* (il se trouve à l'extérieur du *processus*; autrement dit, il n'est pas encore *absorbé* par le processus) [Recommandation Z.101, § 2.1].

retour

E: return

Le *retour* d'une *procédure* est la destruction des *variables* et des *synonymes* créés au moment du *départ de procédure* suivi par l'interprétation du nœud de *départ de procédure* [Recommandation Z.103, § 2.1].

nœud de retour

E: return node

Un *nœud de retour* est le *nœud* contenu dans un *graphe de procédure*, lors du *retour* de la *procédure* [Recommandation Z.103, § 2.1].

symbole de retour

E: return symbol

Symbole figurant dans un *diagramme de procédure* et représentant un *retour* de la *procédure* [Recommandation Z.103, § 2.2].

attribut d'apparition

E: reveal attribute

Un *processus* auquel appartient une *variable* peut avoir un *attribut d'apparition*; dans ce cas, un autre *processus* se trouvant dans le même *bloc* peut *voir* la *valeur* associée à la *variable* [Recommandation Z.101, § 2.3].

mise en réserve

E: save

Une *mise en réserve* est l'ajournement de la *reconnaissance d'un signal*, lorsqu'un *processus* se trouve dans un *état* particulier dans lequel l'*entrée (reconnaissance)* de ce *signal* n'est pas requise [Recommandation Z.101, § 2.2].

ensemble de signaux de mise en réserve

E: save-signal-set

L'*ensemble de signaux de mise en réserve* d'un *état de processus* est l'ensemble des *noms de signal* mis en *réserve* pour cet *état* [Recommandation Z.100, § 2.2].

symbole de mise en réserve

E: save symbol

Symbole dans un *diagramme de processus* du *LDS/GR* représentant le concept LDS de mise en réserve [Recommandation Z.101, § 3.3].

LDS/GR (graphique)

E: SDL/GR

Représentation graphique du LDS [Recommandation Z.100, § 3.1].

LDS/PR (programmation)

E: SDL/PR

Représentation du LDS sous forme de texte [Recommandation Z.100, § 3.1].

LDS/PE (illustration)

E: SDL/PE (pictorial element)

Représentation du LDS sous forme d'éléments graphiques, avec des extensions spécifiques pour les états [Recommandation Z.100, § 3.5].

SELF

E: SELF

L'*objet informatif SELF* d'un *processus* est la valeur d'*instance de processus* unique qui distingue cette instance de toutes les autres *instances de processus* [Recommandation Z.101, § 2.3].

SENDER (émetteur)

E: SENDER

L'*objet informatif SENDER* d'un *processus* est égal à l'*objet informatif du processus d'origine* du *signal* le plus récemment absorbé [Recommandation Z.101, § 2.3].

SET

E: SET

L'*opérateur* défini pour le *type de données de temporisateur* qui permet d'initialiser les temporisateurs [Recommandation Z.104, § 5.12].

valeur partagée

E: shared value

Une *valeur partagée* est la *valeur* associée à une *variable* qui est *apparue* dans un processus et *vue* par un autre processus [Recommandation Z.101, § 2.3].

SIGNAL

E: SIGNAL

Attribut de *paramètre formel* pour un paramètre de *signal* d'une *procédure* [Recommandation Z.103, § 2.3].

signal

E: signal

Le terme signal est, par lui-même, synonyme *d'instance* de signal.

définition de signal

E: signal definition

Une *définition de signal* définit un *nom* comme étant un *nom de signal* et associe une liste de zéro ou plusieurs *types d'information* au *nom de signal* [Recommandation Z.101, § 2.2].

instance de signal

E: signal instance

Une *instance de signal* est l'*instance* d'un *signal* communiquant l'information à une *instance de processus*, soit à partir de l'*action de sortie* d'une autre *instance de processus*, soit à partir de l'*environnement*. Une *instance de signal* peut également transmettre des informations de l'*environnement* à une *instance de processus* [Recommandation Z.100, § 2.3].

élément graphique de récepteur de signalisation

E: signalling receiver PE

Un *élément graphique* représentant un récepteur de signalisation [Recommandation Z.103, § 6.1].

élément graphique d'émetteur de signalisation

E: signalling sender PE

Un *élément graphique* représentant un émetteur de signalisation [Recommandation Z.103, § 6.1].

liste de signaux

E: signal list

Liste des *noms* de tous les *signaux* qui peuvent être véhiculés par un *canal* ou, de manière interne, dans un *bloc*, d'un *processus* à un autre [Recommandation Z.101, § 2.2].

symbole de liste de signaux

E: signal list symbol

Symbole figurant dans un *diagramme d'interaction de blocs* du *LDS/GR* représentant une *liste de signaux* associée à un *canal* ou à un *symbole d'acheminement de signal* [Recommandation Z.101, § 3.1].

symbole d'acheminement des signaux

E: signal route symbol

Symbole dans un *diagramme d'interaction* du *LDS/GR*, indiquant le trajet d'acheminement des *signaux* entre un *processus* et n'importe quel autre *processus* dans le même *bloc* ou dans les *canaux* connectés au *bloc* [Recommandation Z.101, § 3.1].

spécification

E: specification

Les caractéristiques d'un système sont définies dans la *spécification* de ce système. Une spécification se compose des *caractéristiques générales* du système et de la *spécification fonctionnelle* (SF) qui décrit le fonctionnement qu'on attend de ce système [Recommandation Z.100, § 1.1].

langage de spécification et de description (LDS)

E: specification and description language (SDL)

Langage du CCITT utilisé dans la présentation de la *spécification fonctionnelle* et de la *description fonctionnelle* des processus de logique interne des systèmes de commutation à commande par programme enregistré (SPC) [Recommandation Z.100, § 1.1].

action de départ

E: start action

L'*action de départ* d'un *processus* est interprétée avant toute autre *action*. L'*action de départ* initialise les *paramètres formels* du processus [Recommandation Z.101, § 2.3].

nœud de départ

E: start node

Dans un *graphe de processus*, le *nœud de départ* est le seul qui ne suive aucun autre nœud. Chaque *graphe de processus* contient un *nœud de départ*, et un seul, et c'est à ce nœud que commence l'interprétation d'un *processus*. Le *nœud de départ* se trouve à l'endroit où a lieu une *action d'entrée* [Recommandation Z.101, § 2.2].

symbole de départ

E: start symbol

Symbole dans un *diagramme de processus* du *LDS/GR* représentant un *nœud de départ* [Recommandation Z.101, § 3.3].

état

E: state

Un *état* est une condition dans laquelle l'action d'un *processus* est *suspendue* dans l'attente d'un *signal entrant* [Recommandation Z.101, § 2.1].

nœud d'état

E: state node

Nœud figurant dans un *graphe de processus* ou dans un *graphede procédure* à l'endroit où le *processus* entre dans un *état* [Recommandation Z.101, § 2.2].

illustration d'état

E: state picture

Une *illustration d'état* est un *symbole d'état* comportant des *éléments graphiques* utilisés pour étendre le *LDS/GR* au *LDS/PE* [Recommandation Z.103, § 6].

symbole d'état

E: state symbol

Symbole dans un *diagramme de processus* du *LDS/GR*, représentant le concept LDS d'un ou de plusieurs états [Recommandation Z.101, § 3.3].

arrêt

E: stop

Action qui termine une *instance de processus* [Recommandation Z.101, § 2.3].

nœud d'arrêt

E: stop node

Nœud placé dans un *graphe de processus* à l'endroit où a lieu l'*arrêt* [Recommandation Z.101, § 2.2].

symbole d'arrêt

E: stop symbol

Symbole dans un *diagramme de processus* du *LDS/GR*, représentant un *arrêt* [Recommandation Z.101, § 3.1].

chaîne

E: string

Le *générateur de type de données prédéfini* engendrant des *types de données* avec des *valeurs* qui sont des listes de points du *type de données* utilisé pour définir les paramètres du *générateur de chaîne* [Recommandation Z.104, § 5.9].

struct

E: struct

Une *définition du type de données* avec *struct* introduit implicitement des *types de données* pour les noms de champ et les *axiomes* implicites qui définissent l'utilisation des noms de champ avec *extract*! (extraire) et *insert*! (insérer) pour les valeurs partielles des structures [Recommandation Z.104, § 4.3].

sous-bloc

E: sub-block

Un *sous-bloc* est un *bloc* contenu dans un autre *bloc*. Les *sous-blocs* sont formés lors de la subdivision d'un *bloc* [Recommandation Z.102, § 1, 2.1].

définition de sous-bloc

E: sub-block definition

Une *définition de sous-bloc* définit un *bloc* et fait partie d'une *définition de sous-structure de bloc* [Recommandation Z.102, § 2.2].

symbole de sous-bloc

E: sub-block symbol

Symbole dans un *diagramme d'interaction de blocs* ou un *diagramme d'arbre de blocs* du *LDS/GR*; ce symbole représente un sous-bloc et il est identique à un *symbole de bloc* [Recommandation Z.102, § 3.2].

sous-canal

E: sub-channel

Un *sous-canal* est un *canal* formé lors de la *subdivision* d'un *bloc* [Recommandation Z.102, § 2.1].

sous-processus

E: sub-process

Un *sous-processus* est un *processus* formé lors de la subdivision d'un *processus* [Recommandation Z.102, § 2.3].

élément graphique de ligne d'abonné

E: subscriber line PE

Un *élément graphique* représentant une ligne d'abonné [Recommandation Z.103, § 6.1].

élément graphique de panneau de commande

E: switchboard PE

Un *élément graphique* représentant une partie de l'élément de commande d'un équipement terminal [Recommandation Z.103, § 6.1].

élément graphique de module de commutation

E: switching module PE

Un *élément graphique* représentant un module de commutation associé à un chemin de commutation connecté ou réservé [Recommandation Z.103, § 6.1].

définition de synonyme

E: synonym definition

Définition d'un *nom* pour une *valeur de données* [Recommandation Z.104, § 4.12].

diagramme de syntaxe

E: syntax diagram

Les *diagrammes de syntaxe* sont des diagrammes utilisés pour définir la syntaxe concrète du *LDS/PR* [Recommandation Z.100, § 3.4].

syntype

E: syntype

Un *syntype* introduit un ensemble de *valeurs* qui correspond à un sous-ensemble des *valeurs* du *nouveau type* ascendant. *Access*! (accès), *déclare*! (déclarer) et *assign*! (affecter) sont les seuls opérateurs des *syntypes*, étant donné que les *valeurs* avant affectation et après extraction des variables *syntype* sont toujours des *valeurs* du *nouveau type* ascendant [Recommandation Z.104, § 4.3].

système

E: system

Un *système* est un ensemble de *blocs* reliés les uns aux autres et à l'*environnement* par des *canaux*. Le mot *système* est, par lui-même, synonyme d'*instance de système* [Recommandation Z.101, § 2.1].

frontière du système

E: system boundary

La *frontière du système* est la frontière qui sépare les *blocs* – définis selon le LDS – et l'*environnement* [Recommandation Z.101, § 2.3].

définition de système

E: system definition

Une *définition de système* définit les propriétés d'un *système*. Les propriétés d'un *système* sont les *blocs*, les *canaux*, les *signaux* associés aux *canaux*, les *types de données* et les *synonymes* du *système* [Recommandation Z.101, § 2.2].

tâche

E: task

Une *tâche* est une action exécutée au cours d'une *transition*, contenant une séquence d'énoncés *d'affectation*, d'énoncés *d'initialisation* ou d'énoncés *de réinitialisation* ou de texte informel. L'interprétation d'une *tâche* dépend de l'information détenue par le système et peut avoir une incidence sur ces informations [Recommandation Z.101, § 2.2].

nœud de tâche

E: task node

Nœud contenu dans un *graphe de processus* ou dans un *graphe de procédure* à l'endroit où la *tâche* est exécutée [Recommandation Z.101, § 2.2].

symbole de tâche

E: task symbol

Symbole dans un *diagramme de processus* du *LDS/GR*, représentant le concept LDS d'une tâche [Recommandation Z.101, § 3.3].

élément graphique d'équipement terminal

E: terminal equipment PE

Un des six *éléments graphiques* possibles représentant les types suivants d'équipements terminaux: téléphone raccroché, téléphone décroché, circuit, ligne d'abonné, panneau de commande, ou autre [Recommandation Z.103, § 6.1].

symbole d'extension de texte

E: text extension symbol

Le texte associé à ce symbole est considéré comme appartenant au symbole du LDS/GR auquel le symbole d'extension de texte est attaché [Recommandation Z.101, § 3].

time

E: time

Le *type de données prédéfini* représentant le temps absolu [Recommandation Z.104, § 5.10].

timer

E: timer

Le *type de données prédéfini* utilisé pour les temporisateurs et qui définit les *opérateurs SET* et *RESET* des temporisateurs [Recommandation Z.104, § 5.12].

supervision de temps d'un élément graphique de processus

E: time supervision of a process PE

Un *élément graphique* représentant le fonctionnement d'un temporisateur de surveillance [Recommandation Z.103, § 6.1].

transition

E: transition

Une *transition* est une suite d'*actions* se produisant au moment où un *processus* passe d'un *état* à un autre, en réponse à une *entrée* [Recommandation Z.101, § 2.2].

chaîne de transition

E: transition string

Une *chaîne de transition* peut être une séquence de zéro ou plusieurs *actions* comprises entre un *nœud d'entrée* et le *nœud d'état* qui suit, un *nœud d'arrêt* ou un *nœud de retour de procédure* [Recommandation Z.101, § 2.2].

élément graphique de circuit

E: trunk PE

Un élément graphique représentant une interface de ligne de circuit [Recommandation Z.103, § 6.1].

type

E: type

Un *type* est un ensemble de propriétés caractérisant des entités. Dans le LDS, les classes de «types» sont les suivantes: *blocs, canaux, objets informatifs, procédures, processus, signaux* et *systèmes.* Un *type* peut être composé d'entités de la même classe et, dans ce cas, ces entités sont des sous-types (par exemple: un *bloc* composé de *sous-blocs*) [Recommandation Z.100, § 2.1].

définition de type

E: type definition

Une *définition de type* définit les propriétés d'un *type* [Recommandation Z.100, § 2.1].

signal entrant valide

E: valid input signal

Chaque *état* d'un *processus* comporte un ensemble de *noms de signal* pour les *signaux entrants.* Un *signal entrant valide* est un *nom de signal* contenu dans l'un des *ensembles de signaux entrants valides* [Recommandation Z.101, § 2.3].

valeur

E: value

Une *valeur* de données d'un *type de données* est une des *valeurs* qui sont associées à une *variable* de ce *type de données* et qui peut être utilisée avec un *opérateur* nécessitant une valeur de ce *type de données* [Recommandation Z.104, § 1].

variable

E: variable

Une *variable* est une entité appartenant à un *processus* et à laquelle une *valeur* peut être affectée. Une *variable* donne la dernière valeur qui lui a été affectée, lorsqu'elle est atteinte [Recommandation Z.101, § 2.3].

définition de variable

E: variable definition

Une *définition de variable* définit une *instance* d'un *type de données* [Recommandation Z.101, § 2.2].

view (vue)

E: view

Une *variable* peut être *vue* si la *valeur* associée à cette *variable* est révélée par le *processus* auquel cette *variable* appartient et si un autre *processus* peut accéder à cette *valeur* [Recommandation Z.101, § 2.3].

définition de visibilité

E: viewing definition

Une *définition de visibilité* est une partie d'une *définition de processus* par laquelle sont définies les *variables* appartenant à un autre *processus* et ne sont *visibles* que par le processus contenant la *définition de visibilité* [Recommandation Z.101, § 2.2].

expression de visibilité

E: viewing expression

Une *expression de visibilité* est utilisée dans une *expression* pour indiquer que la dernière valeur d'une *variable vue* est obtenue [Recommandation Z.101, § 2.3].

ANNEXE B

(aux Recommandations Z.100 à Z.104)

Résumé de la syntaxe abstraite

Le présent résumé contient toute la syntaxe abstraite ainsi que les règles de formation correcte, qui sont définies dans les Recommandations Z.101, Z.102, Z.103 et Z.104. La syntaxe est expliquée au moyen d'une forme BNF[1] élargie, telle qu'elle est définie dans la Recommandation Z.200, et les règles sont données en anglais. Comme il s'agit d'un résumé, pour les définitions exactes, il convient de se reporter aux textes des Recommandations.

B.1 *Système*

(1) <définition de système> ::=

(Z.101) nom de *système*

(Z.101) [<définition de bloc>]+

(Z.101) [<définition de canal>]*

(Z.101) [<définition de signal>]*

(Z.104) [<définition de données>]*

(Z.103) [<définition de procédure>]*

Définition bien formée

La définition de la structure doit contenir une définition de signal pour chaque nom des listes de signaux dans chacune des définitions de canaux qui y figurent aussi (Recommandation Z.101).

---

[1] BNF = forme Backus-Naur.

B.2 *Bloc*

(2) <définition de bloc> ::=

(Z.101) nom de *bloc*

(Z.101) [<définition de processus>]*

(Z.101) [<définition de signal>]*

(Z.104) [<définition de données>]*

(Z.103) [<définition de procédure>]*

(Z.102) [<définition de sous-structure de bloc> !]

Définition bien formée

Pour chacun des noms de signaux associés aux opérations d'entrée et de sortie des processus qui y figurent, le bloc contient une définition de signal ou est une définition de signal contenue dans la structure environnante.

Si une définition de bloc contient une définition de sous-structure de bloc, il n'est pas nécessaire qu'elle contienne une définition de processus (Recommandation Z.102).

B.3 *Sous-structure de bloc*

(3) <définition de sous-structure de bloc> ::=

(Z.102) [<définition de sous-bloc>]+

(Z.102) [<définition de sous-canal>]

(Z.102) [<définition de canal>]+

(Z.102) [<définition de sous-structure de processus>]*

(Z.102) [<définition de signal>]*

(Z.104) [<définition de données>]*

(Z.103) [<définition de procédure>]*

(3.1) <définition de sous-bloc> ::=

(Z.102) <définition de bloc>

(3.2) <définition de sous-canal> ::=

(Z.102) <définition de canal>

Définition bien formée

Pour chaque définition de processus contenue dans le bloc environnant, la définition de sous-structure de bloc doit contenir une définition de sous-structure de processus.

Pour chacun des points extrémité de terminaison des canaux du bloc environnant, il doit y avoir au moins une définition de sous-canal ayant ce point extrême comme point extrémité de départ, et l'inverse doit être valable pour tous les points extrémité de canaux de départ du bloc environnant. L'ensemble des listes de signaux de définitions de sous-canaux ayant le même point extrémité qu'un canal aboutissant ou partant du bloc environnant doit être identique à la liste de signaux.de ce canal et, de plus, les listes de signaux des définitions de canaux partant d'un point extrémité de terminaison doivent être séparées (Recommandation Z.102).

Une définition de canal contenue dans la sous-structure doit relier les sous-blocs les uns aux autres. Tous les sous-canaux doivent relier des points extrêmes de canaux du bloc environnant à des sous-blocs.

B.4 *Canal*

(4) <définition de canal> ::=

(Z.101) nom de *canal*

(Z.101) [identificateur de *bloc origine*! ENVIRONMENT]

(Z.101) [identificateur de *bloc destinataire*! ENVIRONMENT]

(Z.101) <liste de signaux>

(Z.102) [<définition de sous-structure de canal> !]

(4.1) <liste de signaux> ::=

(Z.101) [nom de *signal*]+

Définition bien formée

Seul un des identificateurs de bloc d'origine ou de bloc de destination peut être remplacé par une référence à l'environnement (Recommandation Z.101).

L'identificateur de bloc d'origine et l'identificateur de bloc de destination liés à un canal doivent être différents et chacun doit être l'identificateur d'un bloc du système ou d'un sous-bloc du bloc, ou encore être l'ENVIRONNEMENT (Recommandation Z.101).

B.5 *Sous-structure de canal*

(5) <définition de sous-structure de canal> ::=

(Z.102) <définition de canal entrant>

(Z.102) <définition de canal sortant>

(Z.102) [<définition de canal>]*

(Z.102) [<définition de bloc>]+

(Z.102) [<définition de signal>]*

(Z.104) [<définition de données>]*

(Z.103) [<définition de procédure>]*

(5.1) <définition de canal entrant> ::=

(Z.102) <définition de canal>

(5.2) <définition de canal sortant> ::=

(Z.102) <définition de canal>

Définition bien formée

La définition du canal entrant a le même point extrémité d'origine que la définition du canal environnant. Le canal sortant a le même point extrémité de terminaison que la définition du canal environnant. Les définitions des canaux d'entrée et de sortie ont une liste de signaux identique à la liste de signaux de la définition du canal environnant (Recommandation Z.102).

B.6 *Signal*

(6) <définition de signal> ::=

(Z.101) nom de *signal*

(Z.101) [identificateur de *type de données*]*

Définition bien formée

Les identificateurs de type de données utilisés doivent être soit prédéfinis soit des noms de définitions de type de données compris dans les entités de structure environnantes.

B.7 *Processus*

(7) <définition de processus> ::=

(Z.101) nom de *processus*

(Z.101) <nombre d'instances>

(Z.101) [<paramètre formel>]

(Z.104) [<définition de données>]*

(Z.101) [<définition de visibilité>]*

(Z.103) [<définition de procédure>]*

(Z.101) <graphe de processus>

(7.1) <nombre d'instances> ::=
<entier identificateur *de valeur*> <entier identificateur *de valeur*>

(7.2) <définition de visibilité> ::=
(Z.101) <identificateur de *variable*>
<identificateur de *type*>
<identificateur de *définition de processus*>

Définition bien formée

Tous les noms de type mentionnés doivent être définis préalablement, dans la définition de processus, ou dans les concepts structurants englobants.

B.8 *Définition de sous-structure de processus*

(8) <définition de sous-structure de processus> ::=
(Z.102) nom de *processus*
(Z.102) [nom de *sous-processus* nom de *sous-bloc*]+

Définition bien formée

Le nom de processus doit être le nom d'une définition de processus contenu dans le bloc environnant. Tous les noms de sous-processus qui y figurent doivent être des noms de sous-processus contenus dans le sous-bloc ayant le nom de sous-bloc associé.

Chaque nom de signal de l'ensemble de signaux d'entrée valables du processus doit apparaître exactement dans l'un des ensembles de signaux d'entrée valables du sous-processus. Chaque nom de signal joint à un nœud de sortie du processus doit être joint à au moins un nœud de sortie de l'un des sous-processus.

B.9 *Graphe de processus*

(9) <graphe de processus> ::=
(Z.101) <nœud de départ de *processus*> <transition de processus>

(9.1) <transition de processus> ::=
<chaîne de transition> <nœud d'état>
!<chaîne de transition> <nœud d'arrêt>

B.10 *Chaîne de transition*

(10) <chaîne de transition> ::=
(Z.101) <nœud de tâche> <chaîne de transition>
(Z.101) !<nœud de sortie> <chaîne de transition>
(Z.101) !<nœud de décision>
(Z.101) !<nœud de demande de création> <chaîne de transition>
(Z.103) !<nœud d'appel de procédure> <chaîne de transition>
(Z.101) !

B.11 *Nœud de départ*

(11) <nœud de départ> ::=

B.12 *Nœud d'état*

(12) <nœud d'état> ::=
(Z.101) nom d'*état*
(Z.101) <ensemble de signaux de mise en réserve>
(Z.101) [<nœud d'entrée>]+

(12.1) <ensemble de mise en réserve> ::=
(Z.101) [identificateur de *signal*]*

B.13 *Nœud d'arrêt*

(13) <nœud d'arrêt> ::=

B.14 *Nœud de tâche*

(14) <nœud de tâche> ::=
(Z.101) [[<instruction>]+ ![<texte informel>]*]

B.14.1 *Instruction*

(14.1) <instruction> ::=
(Z.101) <instruction d'initialisation>
(Z.101) !<instruction de réinitialisation>
(Z.101) !<instruction d'affectation>

(14.1.1) <instruction d'initialisation> ::=
(Z.101) <expression de *temps*> identificateur de *temporisateur*

(14.1.2) <instruction de réinitialisation> ::=
(Z.101) identificateur de *temporisateur*

(14.1.3) <instruction d'affectation> ::=
(Z.101) <opérateur d'*affectation*>
(Z.101) <identificateur de *variable*>
(Z.101) <expression>

B.15 *Nœud de sortie*

(15) <nœud de sortie> ::=
(Z.101) <identificateur de *signal*>
(Z.101) [<expression>]*
(Z.101) <destination>

(15.1) <destination> ::=
(Z.101) <expression-*pid*>

B.16 *Nœud d'entrée*

(16) <nœud d'entrée> ::=
(Z.101) identificateur de *signal d'entrée* <transition de processus>
(Z.103) !identificateur de *signal d'entrée* <transition de procédure>

Définition bien formée

Une transition de processus ne peut apparaître que dans une définition de processus et une transition de procédure ne peut apparaître que dans une définition de procédure.

B.17 *Nœud de décision*

(17) <nœud de décision> ::=
(Z.101) <question>
(Z.101) <réponse> [<réponse>]+

(17.1) <question> ::=

(Z.101) <expression>

!<texte informel>

(17.2) <réponse> ::=

(Z.101) [<identificateur de *valeur*>]+

[<transition de procédure> ! <transition de processus>]

Définition bien formée

Un nœud de décision doit être suivi de deux réponses ou plus (Recommandation Z.101).

Les expressions du nom de décision et les réponses doivent être du même type.

B.18 *Nœud d'appel de procédure*

(18) <nœud d'appel de procédure> ::=

(Z.103) identificateur de *procédure*

(Z.103) [<expression>]*

(Z.103) [identificateur de *signal*]*

(Z.103) <ensemble supplémentaire de mise en réserve>

(18.1) <ensemble supplémentaire de mise en réserve> ::=

[<nom de signal>]*

Définition bien formée

Chaque expression doit être du même type que le paramètre formel correspondant de la définition de procédure ayant le nom de procédure.

Il doit y avoir une expression, pour chaque paramètre formel, de signal non type dans la définition de procédure, et un identificateur de signal pour chaque paramètre formel de signal type.

B.19 *Nœud de demande de création*

(19) <nœud de demande de création> ::=

(Z.101) identificateur de *processus*

(Z.101) [<expression>]*

Définition bien formée

Il doit y avoir une expression pour chaque paramètre formel dans la définition de processus à laquelle se rapporte l'identificateur de processus, et chaque expression doit avoir un type correspondant au paramètre formel associé (Recommandation Z.101).

B.20 *Procédure*

(20) <définition de procédure> ::=

(Z.103) nom de *procédure*

(Z.103) [<définition de données>]*

(Z.104) [<définition variable>]*

(Z.103) [<définition de procédure>]*

(Z.103) [<paramètre formel>]*

(Z.103) <graphe de procédure>

Définition bien formée

Chaque identificateur de signal qui apparaît dans le graphe de procédure doit aussi apparaître comme paramètre formel du signal type.

B.21 *Graphe de procédure*

(21) <graphe de procédure> ::=

(Z.103) <nœud de départ de procédure> <transition de procédure>

(21.1) <transition de procédure> ::=

(Z.103) <chaîne de transition> <nœud d'état>

(Z.103) !<chaîne de transition> <nœud de retour>

B.22 *Nœud de début de procédure*

(22) <nœud de départ de procédure> ::=

B.23 *Nœud de retour*

(23) <nœud de retour> ::=

B.24 *Définition de données*

(24) <définition de données> ::=

(Z.104) <définition de type de données>

(Z.104) !<définition de synonyme>

B.25 *Définition de variables*

(25) <définition de variable> ::=

(Z.101) nom de *variable*

(Z.101) identificateur de *type*

(Z.101) [<attribut de réapparition> !]

B.26 *Identificateur*

(26) <identificateur> ::=

(Z.100) <qualificateur> nom

B.27 *Qualificateur*

(27) <qualificateur> ::=

(Z.100) <nom structural> <nom de type d'entité>

B.28 *Nom structurel*

(28) <nom structural> ::=

(Z.100) nom de *système*

(Z.100) [nom de *bloc*]* [nom de *canal*]*

(Z.100) [nom de *signal*]* [nom de *processus*]*

(Z.100) [nom de *procédure*]* [nom de *type*]*

B.29 *Nom de type d'entité*

(29) <nom de type d'entité> ::=

(Z.100) système !bloc !processus

(Z.100) !procédure !signal !canal

(Z.100) !tâche !décision !début

(Z.100) !arrêt !demande de création

(Z.100) !retour !départ de procédure

(Z.100) !appel !variable !type de données

(Z.100) !opérateur !valeur

B.30 *Définition de type de données*

(30) <définition de type de données> ::=

(Z.104) nom de *type de données* <description de type de données>

(Z.104) !nom de *type de données* <description de type synonyme>

B.31 *Description de type de données*

(31) <description de type de données> ::=

(Z.104) [nom de *valeur*]**

(Z.104) [<introduction d'opérateur>]+

(Z.104) [axiome de type]*

Remarque 1 – La notation [ ]** signifie liste de zéro ou plus et, éventuellement, un nombre infini d'éléments. Exception faite de ce que la liste peut être infinie, cette notation est identique à [ ]*.

Remarque 2 – L'expression «axiome de type» n'est pas définie dans le résumé de la syntaxe. Il existe cependant une syntaxe concrète pour les axiomes.

Remarque 3 – Un signe plus (+) indique que le groupe doit être présent et peut être répété par la suite indéfiniment. Si les éléments syntaxiques sont groupés entre crochets, le groupe est donc facultatif.

B.32 *Introduction d'opérateur*

(32) <introduction d'opérateur> ::=

(Z.104) <opérateur universel>

(Z.104) !nom d'*opérateur* <typage d'opérateur>

B.33 *Opérateur universel*

(33) <opérateur universel> ::=

(Z.104) <opérateur de variable>

(Z.104) !<comparateur>

B.34 *Opérateur de variable*

(34) <opérateur de variable> ::=

(Z.104) affecter

(Z.104) !accéder

(Z.104) !déclarer

B.35 *Comparateur*

(35) <comparateur> ::=

(Z.104) plus petit que

(Z.104) !plus grand que

(Z.104) !égal

B.36 *Typage d'opérateur*

(36) <typage d'opérateur> ::=

(Z.104) <liste de types de données> <identificateur de *type résultat*>

B.37 *Liste de types de données*

(37) <liste de types de données> ::=

(Z.104) [<identificateur de *type de données*>]+

Formation correcte (pour les points 30-37)

Tous les *noms de valeur* doivent s'exclure mutuellement dans une *description.*

Tous les *noms d'opérateur* doivent s'exclure mutuellement dans une *description.*

Pour chaque *typage d'opérateur,* un des *identificateurs de type de données* de la *liste de types de données* doit être l'*identificateur de type de données* du type qui est défini.

B.38 *Syntaxe abstraite de type synonyme*

(38) <description de type synonyme> ::=

(Z.104) <identificateur de *type parent*>
[<identificateur de *valeur*>]*

Syntaxe bien formée

Les *identificateurs de valeur* doivent tous faire partie de l'ensemble d'*identificateurs de valeur* de l'*identificateur de type de données* parent.

B.39 *Texte informel*

(39) <texte informel> ::=

(Z.104) nom *bien compris*

B.40 *Expression*

(40) <expression> ::=

(Z.104) <primaire>

(Z.104) !<opération>
!<expression conditionnelle>

B.40.1 *Opération*

(40.1) <opération> ::=
<opérateur> [<expression>]*

B.40.2 *Expression conditionnelle*

(Z.104) <expression conditionnelle> ::=

(Z.104) <expression *booléenne*>

(Z.104) <expression> <expression>

B.41 *Primaire*

(41) <primaire> ::=

(Z.104) identificateur de *synonyme*

(Z.104) !identificateur de *valeur*

(Z.104) !identificateur de *variable*

B.42 *Opérateur*

(42) <opérateur> ::=

(Z.104) identificateur d'*opérateur*

(Z.104) !<opérateur universel> identificateur de *type*

B.43 *Définition de synonyme*

(43) <définition de synonyme> ::=

(Z.104) nom de *synonyme* <expression constante>

B.44 *Expression constante*

(44) <expression constante> ::=

(Z.104) <valeur constante>

(Z.104) !<opérateur> [<expression constante>]+

B.45 *Valeur constante*

(45) <valeur constante> ::=

(Z.104) identificateur de *valeur*

(Z.104) !identificateur de *synonyme*

ANNEXE C1

(aux Recommandations Z.100 à Z.104)

Résumé du LDS/GR

Dans le LDS/GR, un système comprend les éléments suivants:

- diagramme d'interaction de blocs (DIB)
- diagramme de sous-structure de canal
- arbre de blocs
- arbre de processus
- diagramme de processus
- diagramme synoptique d'état.

Certains de ces éléments sont essentiels pour permettre la spécification et la description des systèmes, tandis que d'autres sont des éléments auxiliaires susceptibles de faciliter la compréhension de la spécification et de la description.

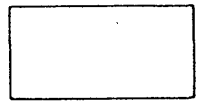
C1.1 *Diagramme d'interaction de blocs (DIB)*

C1.1.1 *Symboles*

Le DIB contient un nom de système, un ensemble de symboles de bloc, des symboles d'environnement, un ensemble de symboles de canal et il peut contenir des symboles de processus.

C1.1.1.1 *Ligne de cadre*

La ligne de cadre entoure le diagramme et représente la frontière des blocs subdivisés.



CCITT-74040

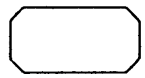
C1.1.1.2 *Symbole de bloc*

Un symbole de bloc contient le nom de bloc (voir la Recommandation Z.101, § 3.1.1 et D.U., § D.4.3.2).

CCITT-74040

C1.1.1.3 *Symbole de processus*

Le symbole de processus contient le nom du processus (voir la Recommandation Z.101, § 3.1.1 et D.U., § D.4.3.4), il peut également comporter des parenthèses entourant la liste des paramètres formels.



CCITT-74040

C1.1.1.4 *Symbole d'environnement*

Le symbole d'environnement est représenté exclusivement par un mot clé (voir la Recommandation Z.101, § 3.1.1):

ENVIRONNEMENT (ENV)

C1.1.1.5 *Symbole de canal*

Le symbole de canal contient un nom de canal. Le symbole de canal a une extrémité d'origine connectée à un symbole de bloc et une extrémité de destination connectée à un autre symbole de bloc. D'autre part, le point de l'extrémité d'origine ou le point de destination (mais pas les deux points) peut être connecté à un symbole d'environnement au lieu d'être connecté à un symbole de bloc.

Un symbole de canal comprend une pointe de flèche au milieu de la ligne montrant le sens de circulation des signaux.

CCITT-74040

C1.1.1.6 *Symbole d'acheminement de signal*

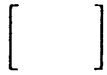
Un symbole d'acheminement de signal figurant dans un bloc peut être associé à un symbole de liste de signaux.

Un symbole d'acheminement de signal conduit soit d'un processus à un autre, soit d'un processus à l'extrémité d'origine d'un canal (à la frontière de bloc), soit de l'extrémité de destination d'un canal (à la frontière de bloc) à un processus.

Les symboles d'acheminement de signal peuvent converger à l'extrémité d'origine d'un canal (à la frontière de bloc) et peuvent diverger à partir de l'extrémité de destination d'un canal à la frontière de bloc. Un symbole d'acheminement de signal comprend une pointe de flèche à l'une des extrémités, pour montrer le sens de circulation des signaux.

CCITT-74040

C1.1.1.7 *Symbole de liste de signaux*



CCITT-74040

Le symbole de liste de signaux contient une liste de noms. La liste des signaux peut avoir elle-même un nom inscrit au-dessus du symbole. Les inscriptions figurant dans la liste, qui sont séparées par des virgules et placées soit en colonnes, soit en rangées, sont des noms de signaux désignés individuellement ou des noms d'autres listes de signaux. Dans la liste, les noms de liste de signaux se distinguent des noms de signal par le fait que chaque nom de liste de signaux est entouré, lui-même, d'une paire de crochets supplémentaire.

C1.1.1.8 *Symbole de création*

CCITT-74040

Ce symbole est utilisé entre les symboles de processus pour indiquer qu'à l'extrémité de la pointe de la flèche, une instance de processus est créée par une instance de processus se trouvant à l'extrémité d'origine de la ligne pointillée. Le symbole de création peut relier le même processus pour indiquer la création d'une instance de ce processus par une autre instance du même processus.

C1.1.2 *Règles*

A chaque bloc correspond un certain nombre de lignes orientées de manière à le relier à d'autres blocs et/ou à l'environnement.

Si des processus sont représentés dans le diagramme d'interaction de bloc, les lignes de signal qui relient les processus doivent être établies uniquement entre les processus du même bloc qui communiquent entre eux.

C1.1.3 *Convention*

Les symboles de canal et les lignes de signal doivent respectivement relier les frontières de symbole de bloc et les frontières de symbole de processus, de préférence, sous un angle de 90 degrés. En cas de besoin, les symboles de canal peuvent contenir des coudes à 90 degrés.

Le croisement de lignes n'a pas de signification.

C1.2 *Diagramme de sous-structure de canal*

Etant donné que les composants sont des blocs et des canaux, le diagramme ressemble à un diagramme de sous-structure de bloc.

C1.2.1 *Symboles*

Les symboles utilisés dans ce diagramme sont les mêmes que ceux utilisés dans le diagramme de sous-structure de bloc.

C1.2.2 *Règles*

Les règles applicables au tracé des connexions dans ce diagramme sont les mêmes que celles qui concernent le diagramme de sous-structure de bloc.

C1.2.3 *Conventions*

Les conventions graphiques décrites pour le DIB/F s'appliquent également au diagramme de sous-structure de canal.

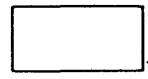
C1.3 *Arbre de blocs*

Le diagramme d'arbre de blocs contient des boîtes et des lignes de subdivision.

C1.3.1 *Symboles*

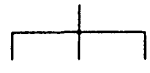
C1.3.1.1 *Boîte*

Une boîte représente un système ou un bloc. Le nom de l'objet représenté doit figurer à l'intérieur du symbole.



C1.3.1.2 *Lignes de subdivision*

Ces lignes de subdivision représentent la manière dont un objet situé «au-dessus» est décomposé en sous-blocs situés «au-dessous».



C1.3.2 *Règles*

Les symboles sont reliés entre eux de manière à former un arbre hiérarchique. Les règles d'interconnexion à l'intérieur du diagramme sont les suivantes:

- il n'y a qu'une boîte «racine» (boîte supérieure);
- toute autre boîte doit suivre une branche de la ligne subdivisée;
- toute boîte peut être suivie d'une ligne de liaison subdivisée;
- une ligne de subdivision doit être suivie de boîtes sur chacun de ses embranchements et doit elle-même sortir d'une boîte.

C1.3.3 *Conventions*

Un arbre doit être tracé de manière que chaque niveau de subdivision se trouve à une hauteur logique dans la représentation; cela signifie que chacune des lignes de subdivision doit avoir une longueur égale vers le bas du diagramme.

C1.4 *Arbre de processus*

L'arbre de processus contient des symboles de processus et des lignes de liaison de subdivision.

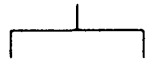
C1.4.1 *Symboles*

C1.4.1.1 *Symbole de processus*

Comme cela est spécifié dans la Recommandation Z.101 (§ 3.1.2), le nom du processus doit figurer à l'intérieur du symbole qui s'y rapporte. La syntaxe de commentaire est utilisée pour indiquer le bloc auquel le processus est alloué.

C1.4.1.2 *Lignes de subdivision*

Les lignes de subdivision représentent la manière dont le processus situé «au-dessus» est décomposé en sous-processus situés «au-dessous» et comment ceux-ci peuvent remplacer le processus.



C1.4.2 *Règles*

Les symboles sont reliés entre eux de manière à former un arbre hiérarchique. Les règles d'interconnexion du diagramme sont les suivantes:

- il n'y a qu'un symbole de processus racine (processus supérieur), qui est suivi d'une ligne subdivisée;
- chacun des embranchements de cette ligne de subdivision conduit à un symbole de processus.

C1.4.3 *Conventions*

Si un diagramme d'arbre de processus est grand, il peut être approprié de scinder le diagramme en plusieurs diagrammes. Cette opération doit être telle que le premier diagramme soit découpé de manière qu'un ensemble de subdivisions de processus apparaisse comme non subdivisé. Dans le diagramme suivant, ces processus apparaîtront comme des racines.

C1.5 *Diagramme de processus*

Un diagramme de processus est un ensemble de symboles reliés entre eux par des lignes de liaison.

C1.5.1 *Symboles*

C1.5.1.1 *Symbole de départ*

Le symbole de départ contient le nom du processus qu'il décrit (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4.3 et D.6.3).

C1.5.1.2 *Symbole d'état*

Le symbole d'état contient le nom d'état ou un astérisque ou un astérisque suivi par des noms d'état entre crochets (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4.3 et D.6.3).

C1.5.1.3 *Symbole d'entrée*

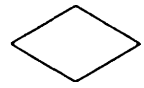
Le symbole d'entrée contient les noms de signal séparés par des virgules ou un astérisque (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4.3 et D.6.3).

C1.5.1.4 *Symbole de mise en réserve*

Le symbole de mise en réserve contient les noms de signal séparés par des virgules ou une astérisque (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4.3 et D.6.3).

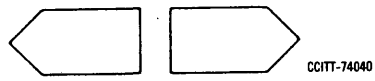
C1.5.1.5 *Symbole de décision*

Le symbole de décision contient le nom de décision (facultatif) et une expression textuelle formelle ou informelle (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4.3 et D.6.3).



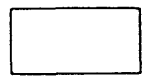
C1.5.1.6 *Symbole de sortie*

Le symbole de sortie contient les noms de signal séparés par des virgules (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4.3 et D.6.3).



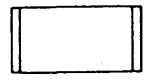
C1.5.1.7 *Symbole de tâche*

Le symbole de tâche contient le nom de tâche (facultatif) et une expression textuelle formelle ou informelle (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4.3 et D.6.3).



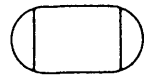
C1.5.1.8 *Symbole d'appel de procédure*

Le symbole d'appel de procédure contient le nom de la procédure et les paramètres effectifs entre parenthèses (voir la Recommandation Z.103, § 2.2 et D.U., § D.4.3 et D.6.3).



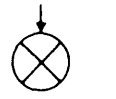
C1.5.1.9 *Symbole de départ de procédure*

Le symbole de départ de procédure contient le nom de procédure et les paramètres formels entre parenthèses (voir la Recommandation Z.103, § 2.2 et D.U., § D.4.3 et D.6.3).



C1.5.1.10 *Symbole de retour*

Le symbole de retour est un cercle à l'intérieur duquel se trouve une croix (voir la Recommandation Z.103, § 2.2 et D.U., § D.4.3 et D.6.3).



C1.5.1.11 *Symbole d'accès entrant de macro*

Le symbole d'accès entrant de macro contient le nom de macro (voir la Recommandation Z.103, § 4.2 et D.U., § D.4.3 et D.6.3).

C1.5.1.12 *Symbole d'accès sortant de macro*

Le symbole d'accès sortant de macro est un cercle comportant une barre à l'intérieur (voir la Recommandation Z.103, § 4.2 et D.U., § D.4.3 et D.6.3).

C1.5.1.13 *Symbole de création*

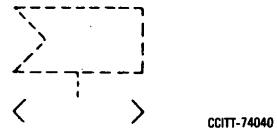
Le symbole de création contient le nom de processus et les paramètres effectifs entre crochets (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4.3 et D.6.3).

C1.5.1.14 *Symbole de signal continu*

Le symbole de signal continu contient le texte de la condition de validation puis le mot clé PRIORITY suivi du numéro de priorité pertinent (voir la Recommandation Z.103, § 3.3.3 et D.U., § D.4).

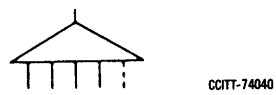
C1.5.1.15 *Symbole de condition de validation*

Le symbole de condition de validation contient l'expression textuelle de la condition (voir D.U., § D.4).



C1.5.1.16 *Symbole d'option*

Le symbole d'option contient le texte de l'option (voir la Recommandation Z.103, § 5.2 et D.U., § D.4).



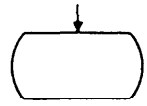
C1.5.1.17 *Symbole de jonction ou connecteur*

Ce symbole contient le nom de jonction/connecteur (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4).

CCITT-74040

C1.5.1.18 *Symbole d'état suivant (NEXTSTATE)*

Ce symbole contient le nom d'état ou un trait d'union (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4).



CCITT-74040

C1.5.1.19 *Symbole d'arrêt*

Le symbole d'arrêt est une croix (voir la Recommandation Z.101, § 3.3.1 et D.U., § D.4).

CCITT-74040

C1.5.2 *Règles*

- Un état doit être relié à un ou plusieurs symboles d'entrée ou symboles de mise en réserve (toute connexion avec d'autres symboles est erronée).
- Un nœud d'appel de procédure peut suivre n'importe quel nœud qui n'est ni un nœud d'état, ni un nœud d'arrêt, ni un nœud de retour.
- Un nœud d'appel peut être suivi par n'importe quel nœud, sauf un nœud de départ ou un nœud de mise en réserve.
- Des conditions de validation peuvent être attachées à tout symbole d'entrée.
- Un signal continu est attaché à une ligne de liaison provenant d'un symbole d'état mais sans symbole d'entrée.
- Le symbole d'appel de macro peut être inséré dans tout diagramme (diagramme de processus, DIB, diagramme d'ensemble d'état) et en n'importe quel point du diagramme. Ce symbole peut avoir un ou plusieurs accès entrants. Dans le cas d'une pluralité d'accès entrants, une étiquette doit être associée à chaque accès entrant. Ce symbole peut avoir zéro, un ou plusieurs accès sortants; dans ce dernier cas une étiquette doit être associée à chaque accès sortant. Les accès entrants sont représentés par des flèches pointées vers le symbole de macro; les accès sortants sont représentés par des flèches partant du symbole de macro. Les accès entrants et sortants de symbole de macro sont reliés à d'autres symboles par des lignes de liaison de type approprié, selon leur signification.
- Le symbole d'arrêt et le symbole de retour d'une procédure et d'une macro ne sont suivis d'aucun symbole.
- Le symbole de création peut être placé au point où se trouve une tâche.
- Le symbole d'option peut être inséré directement dans une transition, uniquement si les variantes de fonctionnement ne comportent pas des états et s'il se termine en un point quelconque.
- Une ligne de liaison pleine peut être interrompue par une paire de connecteurs associés et l'on suppose que la circulation de l'information part du connecteur de sortie pour aboutir au connecteur d'entrée associé (symbole de jonction).
- Lorsque deux symboles ou plus sont suivis d'un seul symbole, les lignes de liaison aboutissant à ce symbole convergent. Cette convergence peut être représentée sous la forme d'une ligne de liaison qui en rejoint une autre, sous la forme de plusieurs connecteurs de sortie associés à un seul connecteur d'entrée, ou encore sous la forme de lignes de liaison séparées entrant dans le même symbole.

Exemple: convergence

CCITT-74040

– Un connecteur de sortie n'est suivi d'aucun autre symbole et aucune ligne de liaison n'en provient.
– Lorsqu'un symbole est suivi de deux ou plusieurs autres symboles, une ligne de liaison partant de ce symbole peut diverger en deux ou plusieurs lignes de liaison.

Exemple: divergence

CCITT-74040

– Des pointes de flèche sont nécessaires chaque fois que deux lignes de liaison se rencontrent et chaque fois que ces lignes entrent dans un connecteur de sortie ou dans un symbole d'état. L'utilisation des flèches est interdite sur les lignes de liaison entrant dans des symboles d'entrée.
– Le symbole de connexion est relié à un symbole ou à une ligne de liaison pleine.

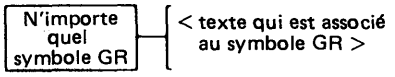
C1.5.3 *Conventions*

– Dans un diagramme donné, tous les symboles du même type doivent de préférence avoir les mêmes dimensions.
– L'orientation des symboles doit, de préférence, être horizontale et le rapport entre la longueur et la largeur des symboles doit de préférence être 2:1.

C1.5.4 *Symboles généraux*

C1.5.4.1 *Symbole d'extension de texte*

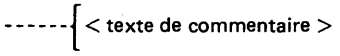
Il peut être associé à tous les symboles du LDS/GR. Le texte contenu dans ce symbole doit être considéré comme figurant dans le symbole auquel le symbole d'extension de texte est associé.



CCITT-74040

C1.5.4.2 *Symbole de commentaire*

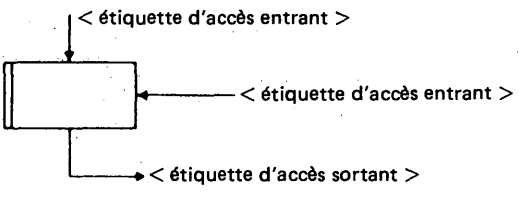
Le symbole de commentaire contient le texte de commentaire.



CCITT-74040

C1.5.4.3 *Symboles de macro, d'accès entrant et d'accès sortant*

Le symbole de macro est la référence à la définition de macro. Les accès entrants à la macro et les accès sortants de la macro sont représentés par des lignes de liaison aboutissant au signal ou partant de celui-ci. Des étiquettes peuvent être associées facultativement aux lignes de liaison. Le symbole de macro contient le nom de la définition de macro.



CCITT-74040

C1.6 *Diagramme synoptique d'état*

Ce diagramme représente la séquence des états dans un processus. Il permet de voir à partir de quels états un état déterminé peut être atteint (voir D.U., § D.4.2 et D.4.3).

C1.6.1 *Symboles*

C1.6.1.1 *Symbole d'état*

Dans ce diagramme le symbole d'état est un cercle contenant le nom d'état (voir D.U., § D.4.2 et D.4.3).

C1.6.2 *Règles*

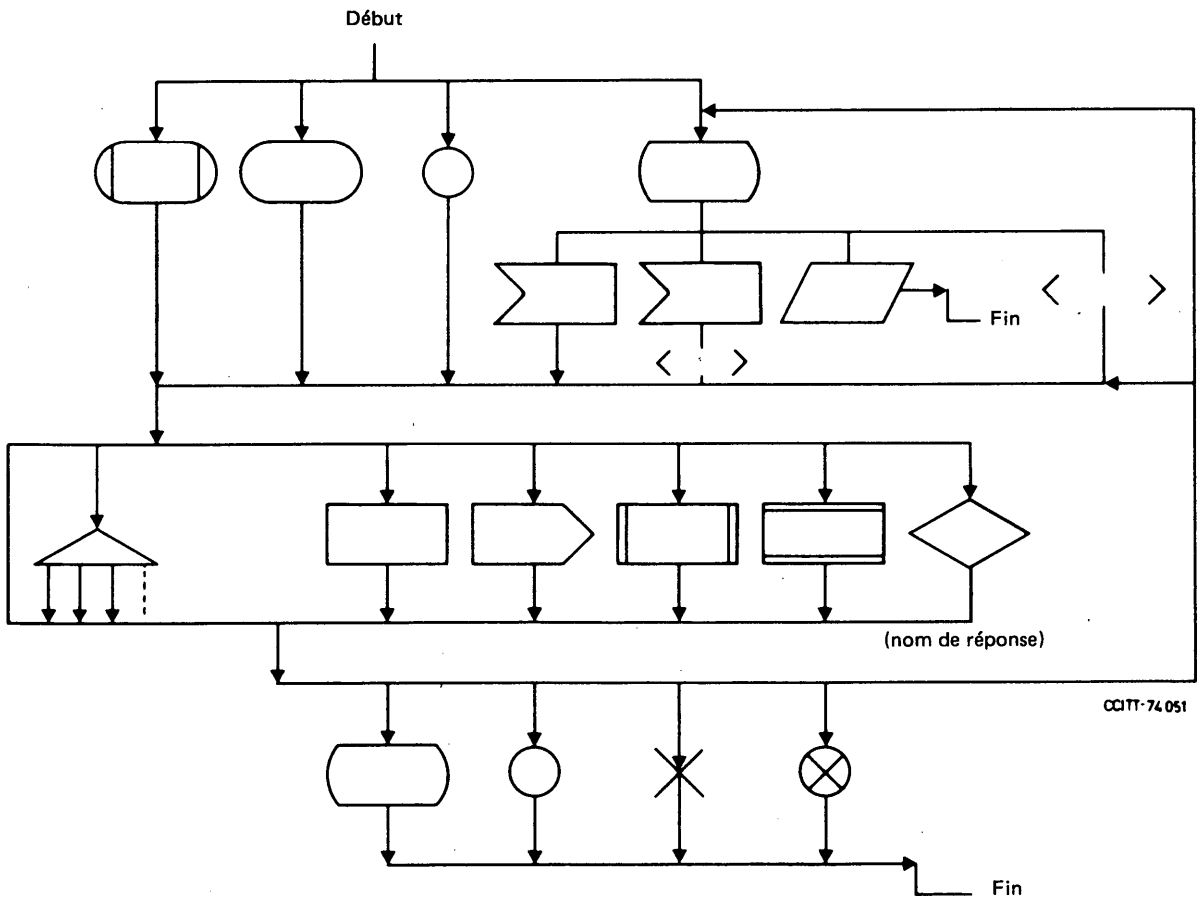
– Les états sont reliés par des arcs. Les noms de signal d'entrée peuvent être facultativement inscrits sur les arcs.

C1.7 *Conventions générales*

La syntaxe graphique comporte certaines conventions de dessin, valables pour tous les types de diagrammes:

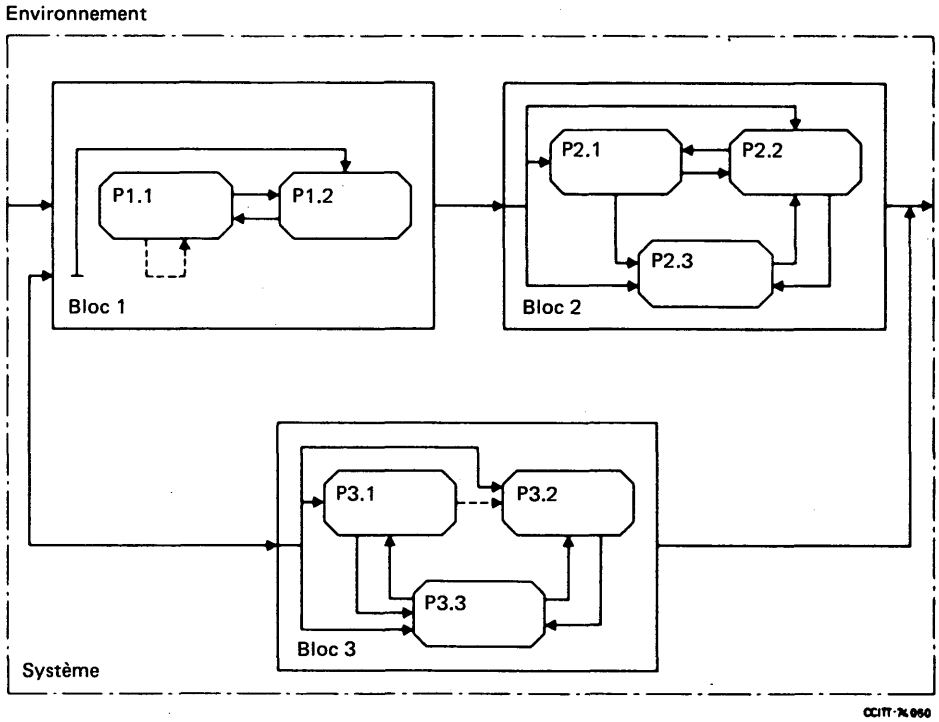
– le rapport d'aspect et la dimension des symboles sont variables, au choix de l'usager;
– les frontières de symboles ne doivent pas se recouvrir, ni se traverser. Une exception à cette règle s'applique aux symboles de canaux et de circulation des signaux qui peuvent se croiser. Les symboles de canaux ou de circulation de signaux qui se croisent n'ont entre eux aucune relation logique.

C1.8 *Relations entre les diagrammes de processus LDS/GR et la syntaxe abstraite et les symboles du LDS*

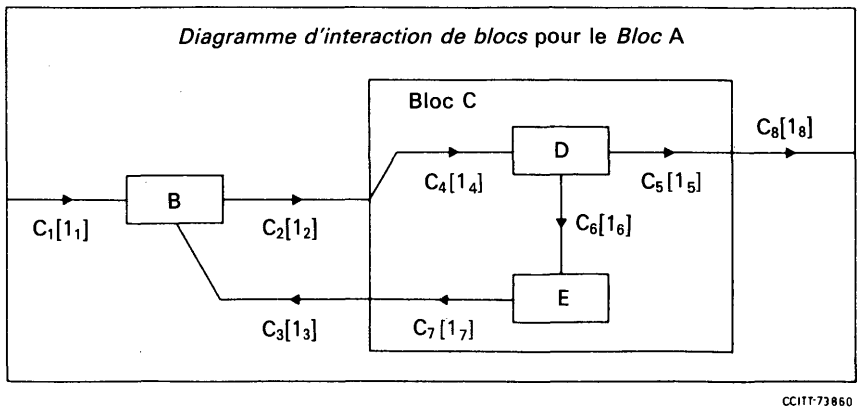


C1.9 *Exemples*

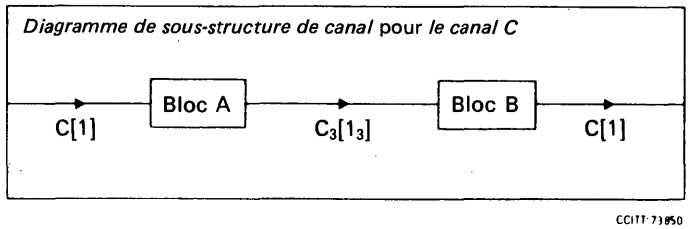
C1.9.1 *Diagramme d'interaction de blocs*



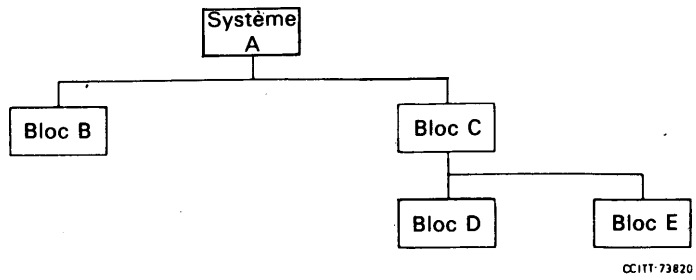
C1.9.2 *Diagramme d'interaction de blocs*



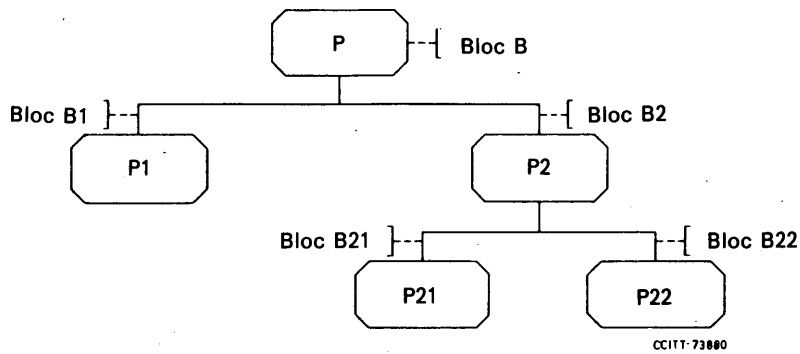
C1.9.3 *Diagramme de sous-structure de canal*



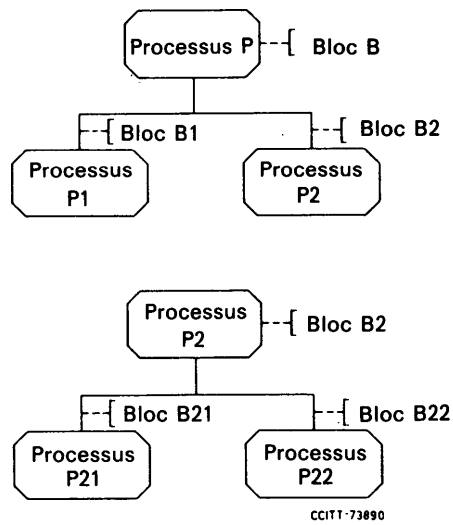
C1.9.4 *Arbre de blocs*



C1.9.5 *Arbre de processus*



Autre présentation du même arbre de processus (représentation utile lorsque l'arbre de processus est grand).



ANNEXE C2

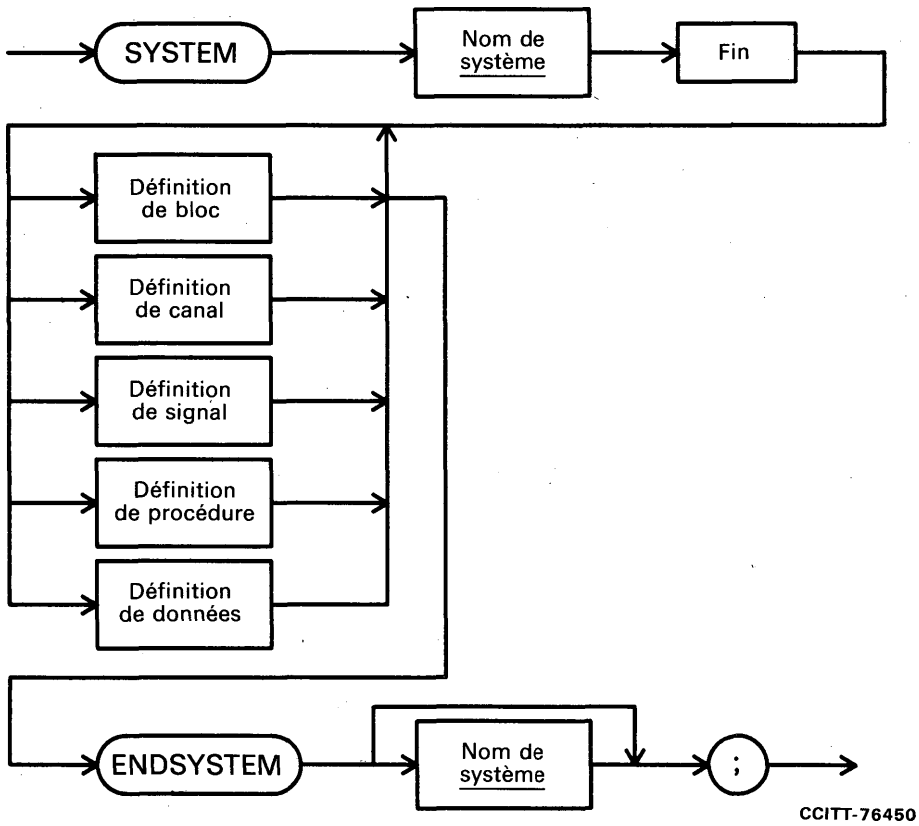
(aux Recommandations Z.100 à Z.104)

Résumé du LDS/PR

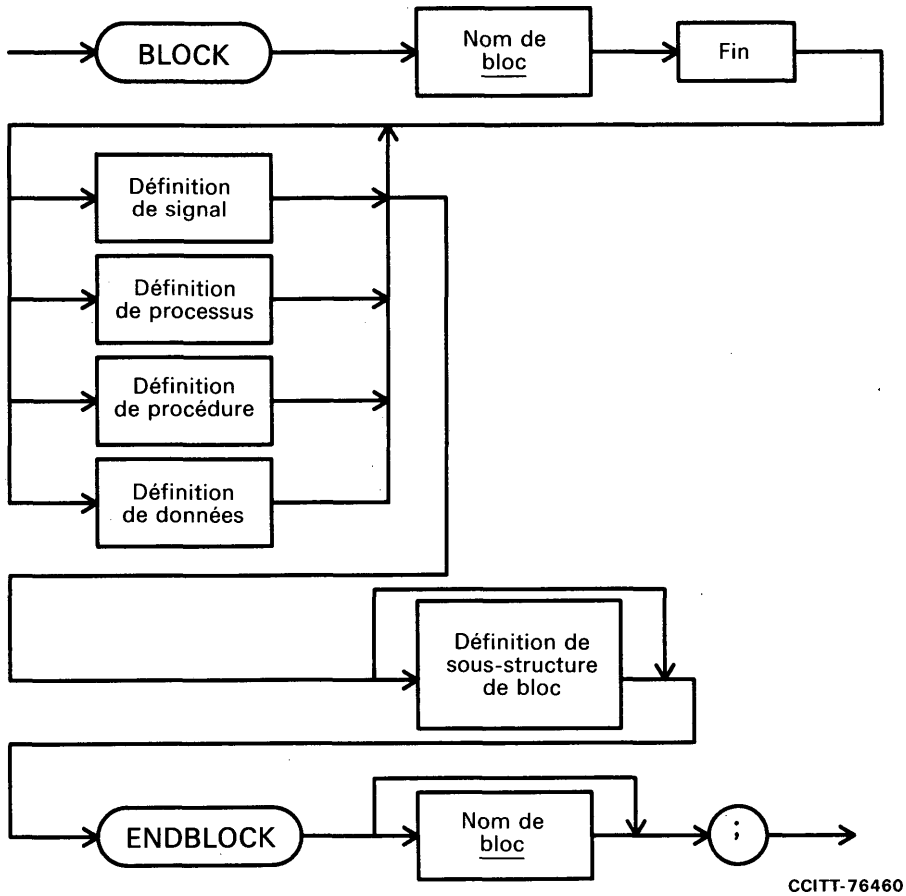
C2.1 *Syntaxe*

Diagrammes de syntaxe

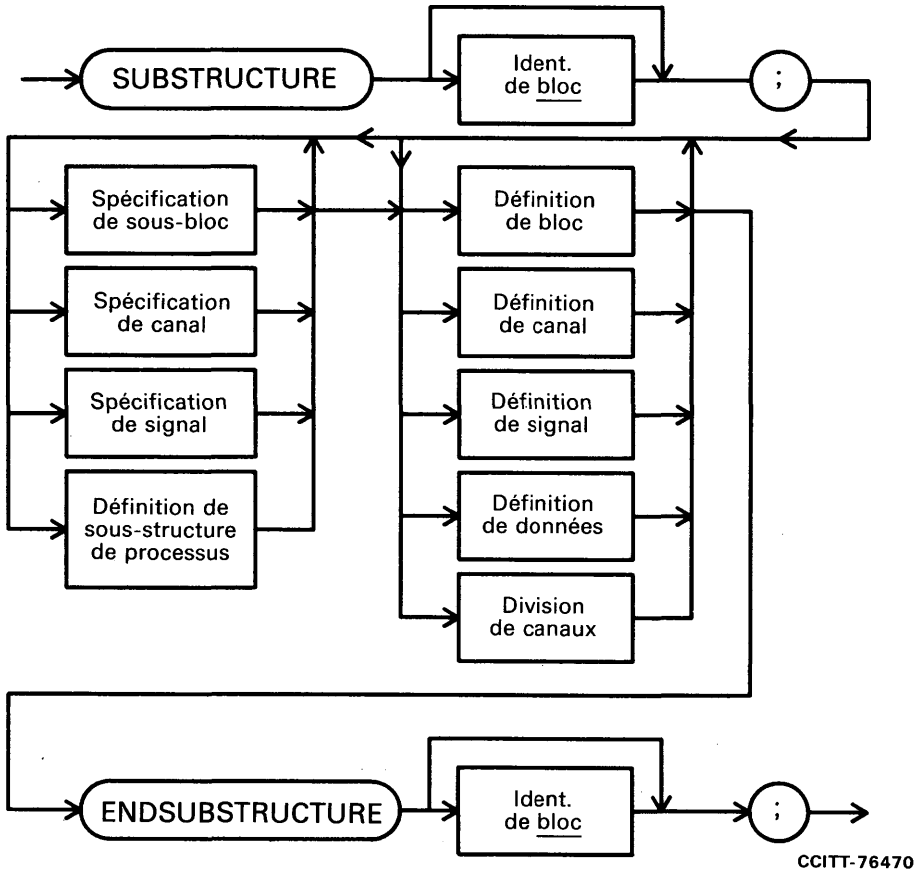
DÉFINITION DE SYSTÈME



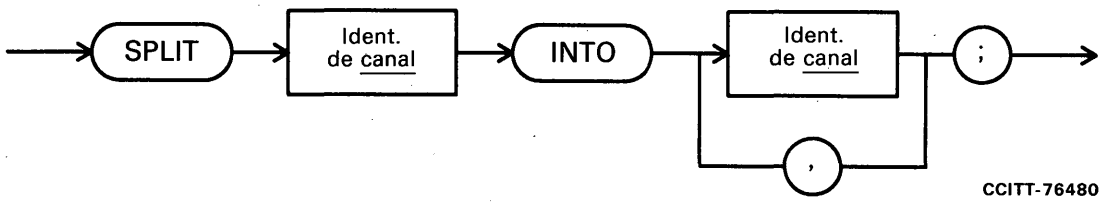
DÉFINITION DE BLOC



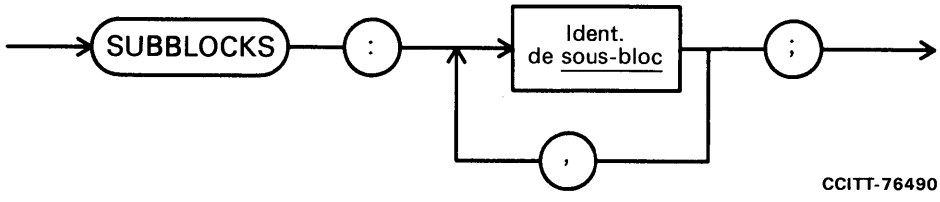
DÉFINITION DE SOUS-STRUCTURE DE BLOC



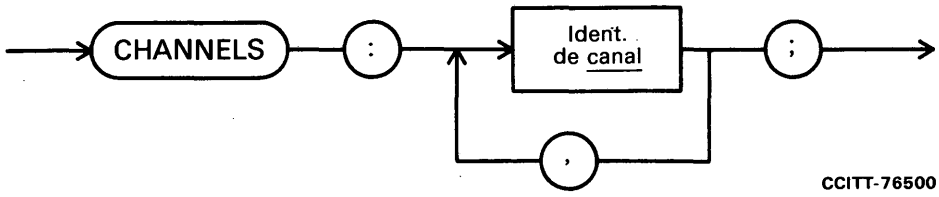
DIVISION DES CANAUX



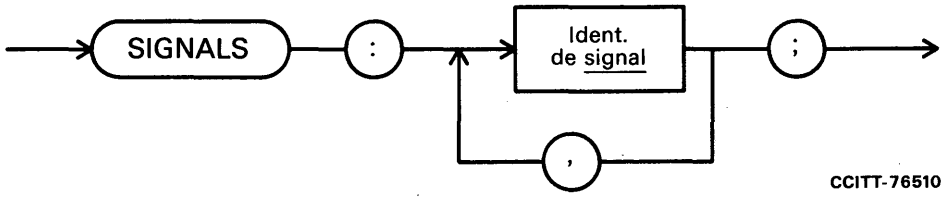
SPÉCIFICATION DE SOUS-BLOC



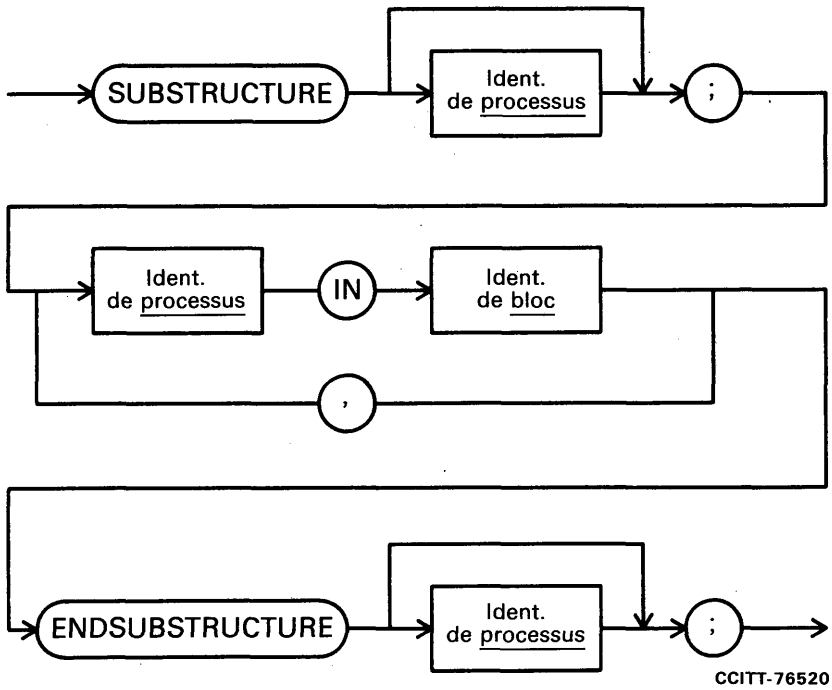
SPÉCIFICATION DES CANAUX



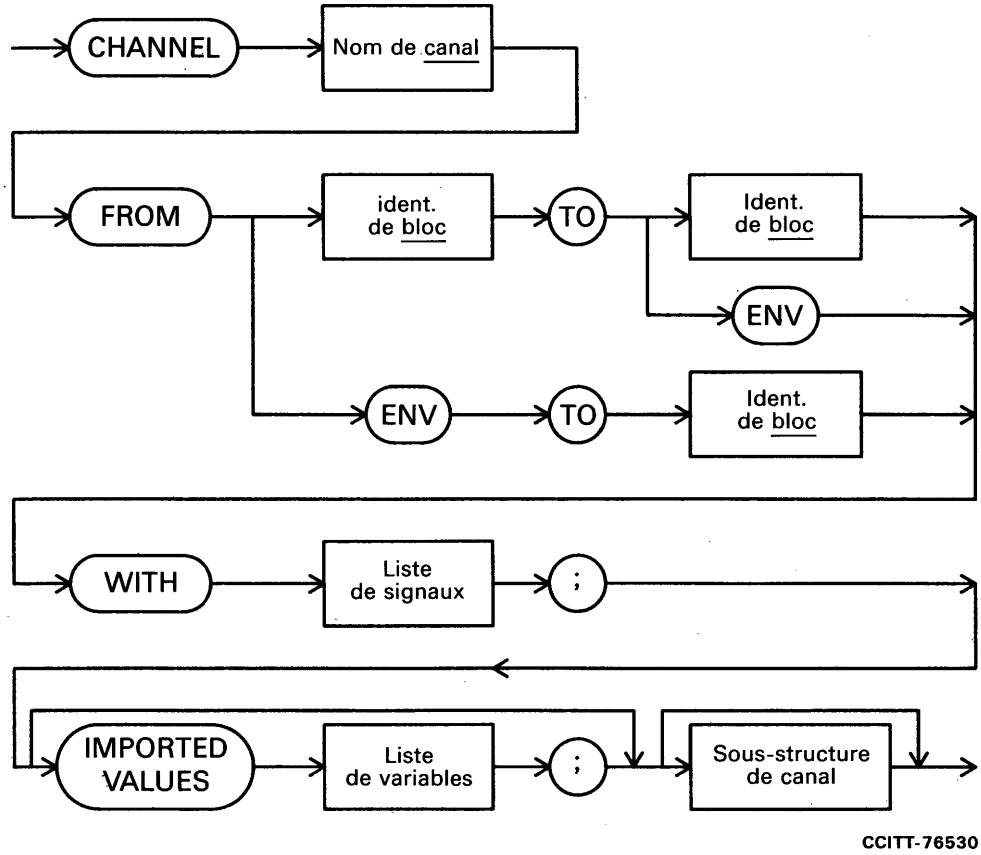
SPÉCIFICATION DES SIGNAUX



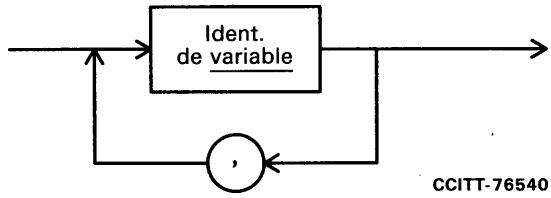
DÉFINITION DE SOUS-STRUCTURE DE PROCESSUS



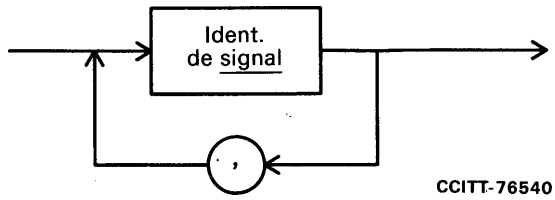
DÉFINITION DE CANAL



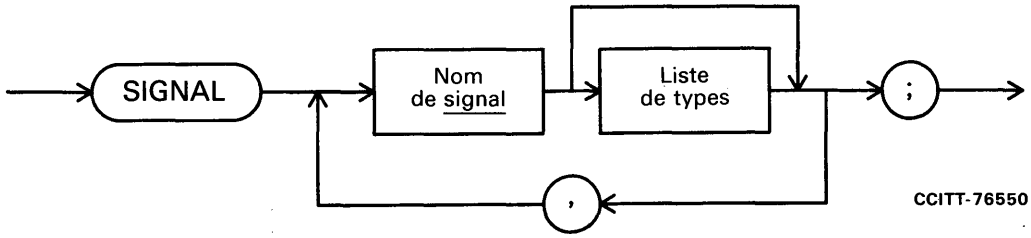
LISTE DE VARIABLES



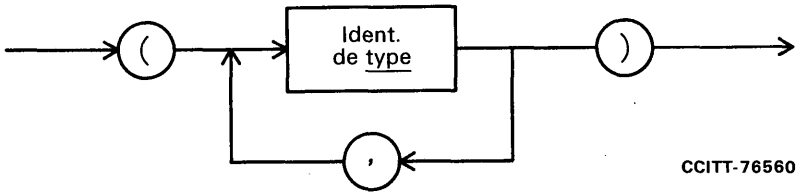
LISTE DE SIGNAUX



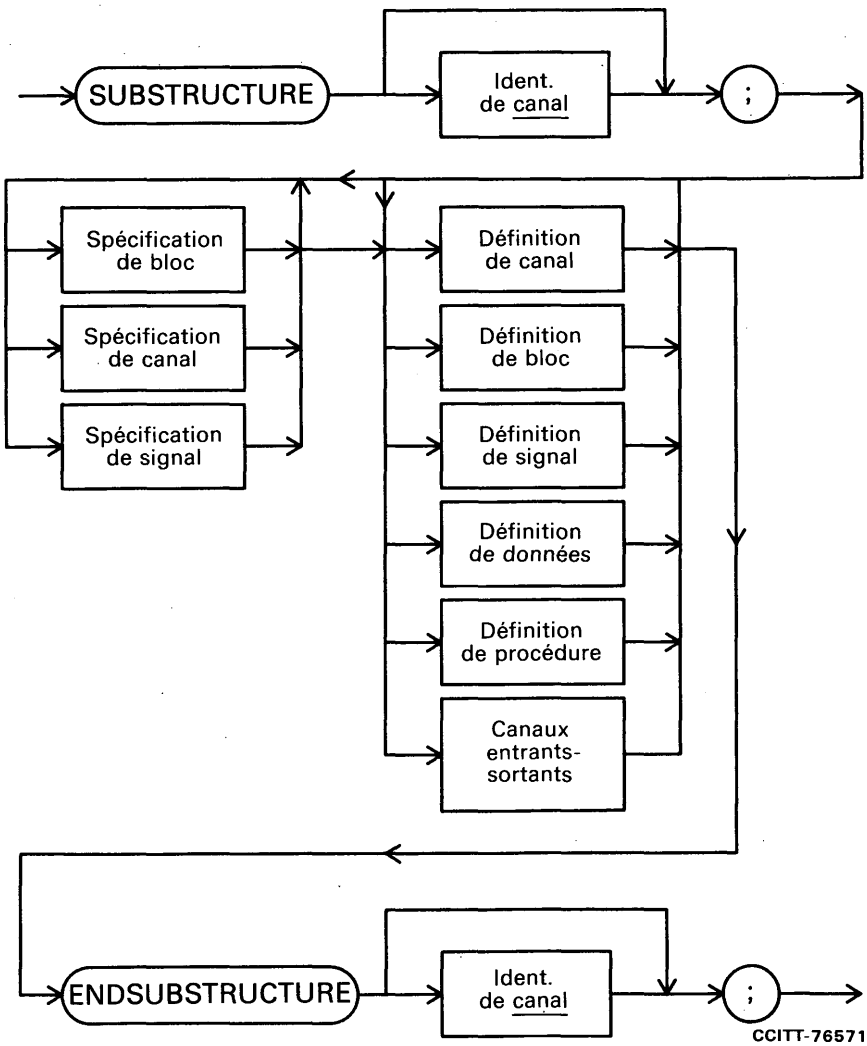
DÉFINITION DE SIGNAL



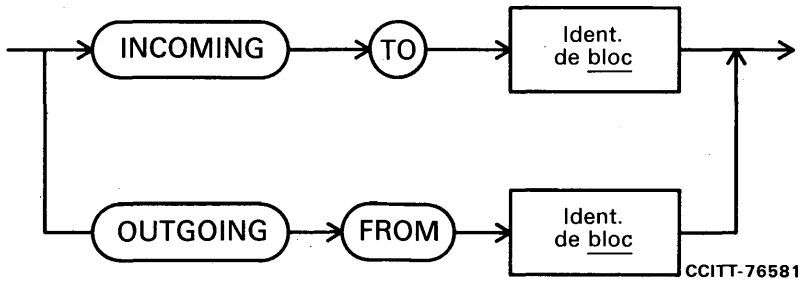
LISTE DE TYPES



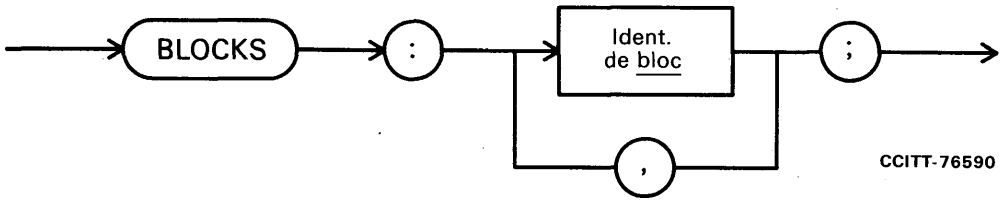
DÉFINITION DE SOUS-STRUCTURE DE CANAL



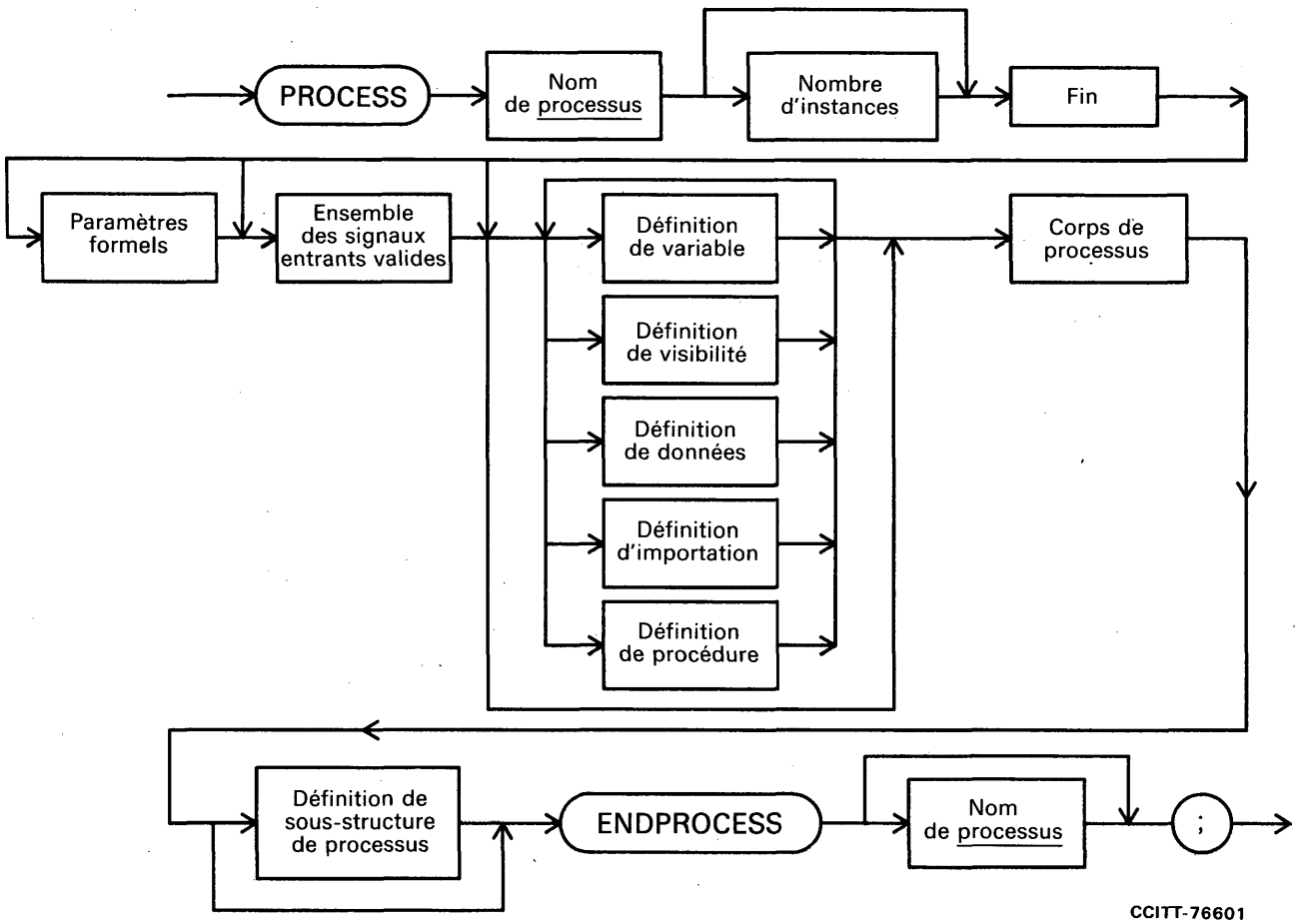
CANAUX ENTRANTS-SORTANTS



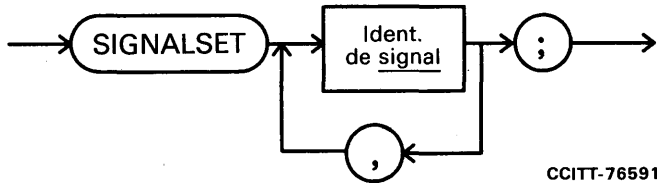
SPÉCIFICATION DE BLOC



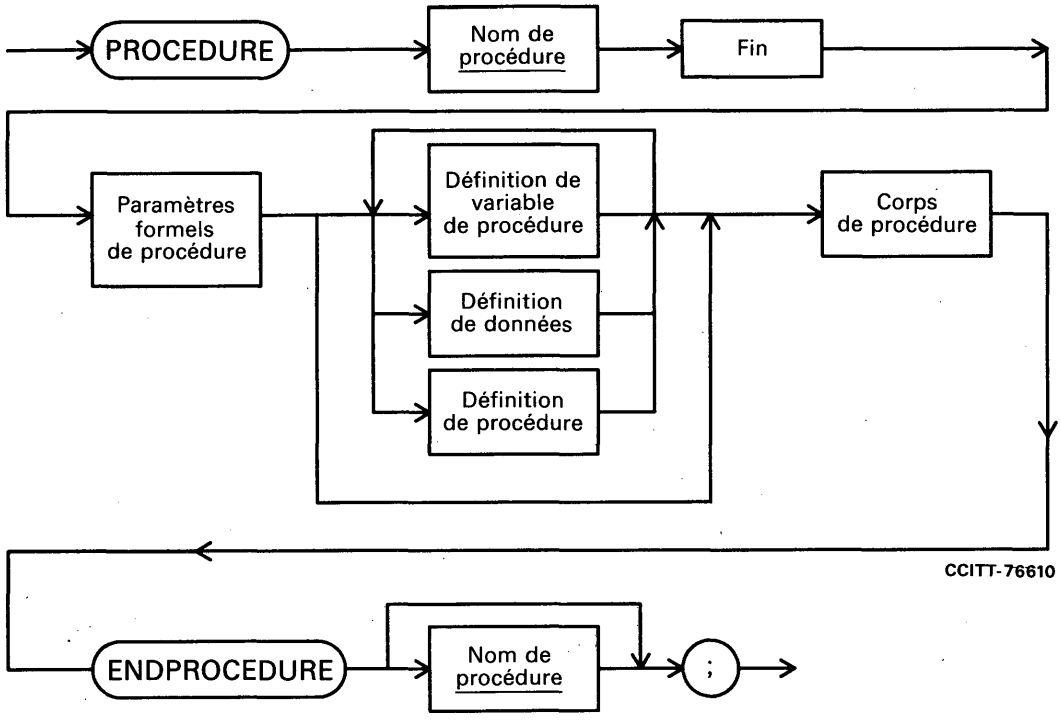
DÉFINITION DE PROCESSUS



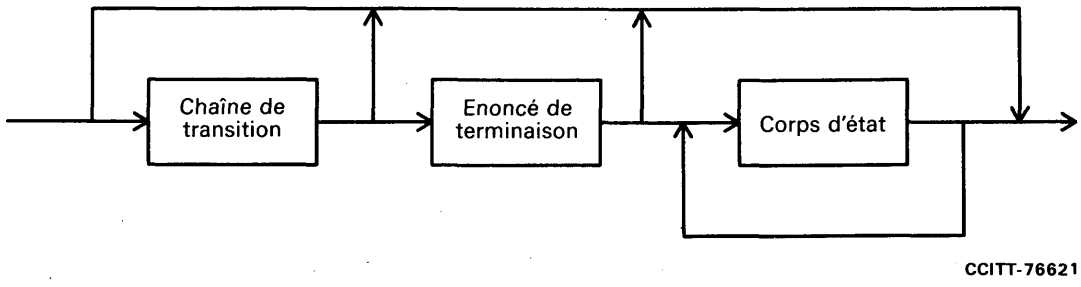
ENSEMBLE DES SIGNAUX D'ENTRÉE VALIDES



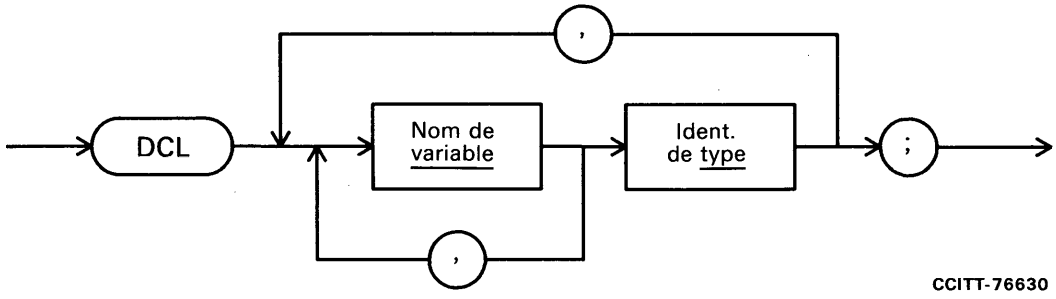
DÉFINITION DE PROCÉDURE



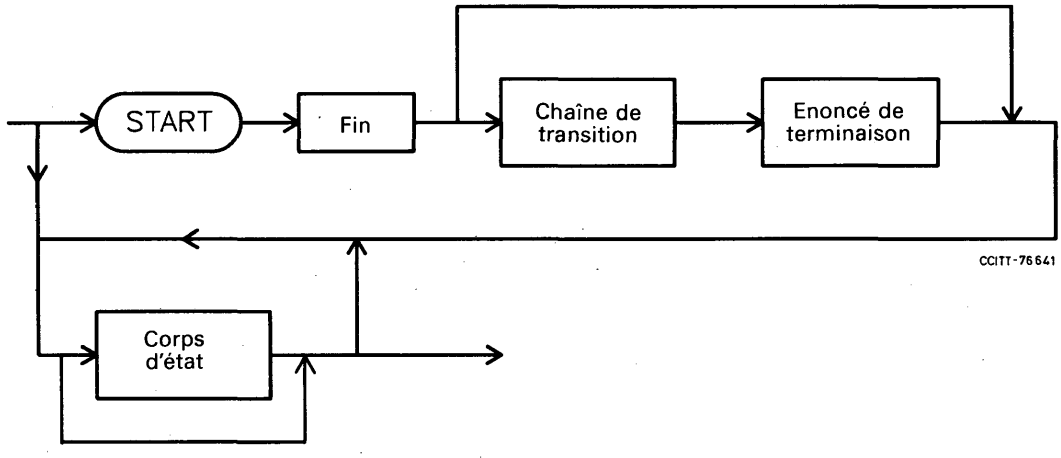
CORPS DE PROCÉDURE



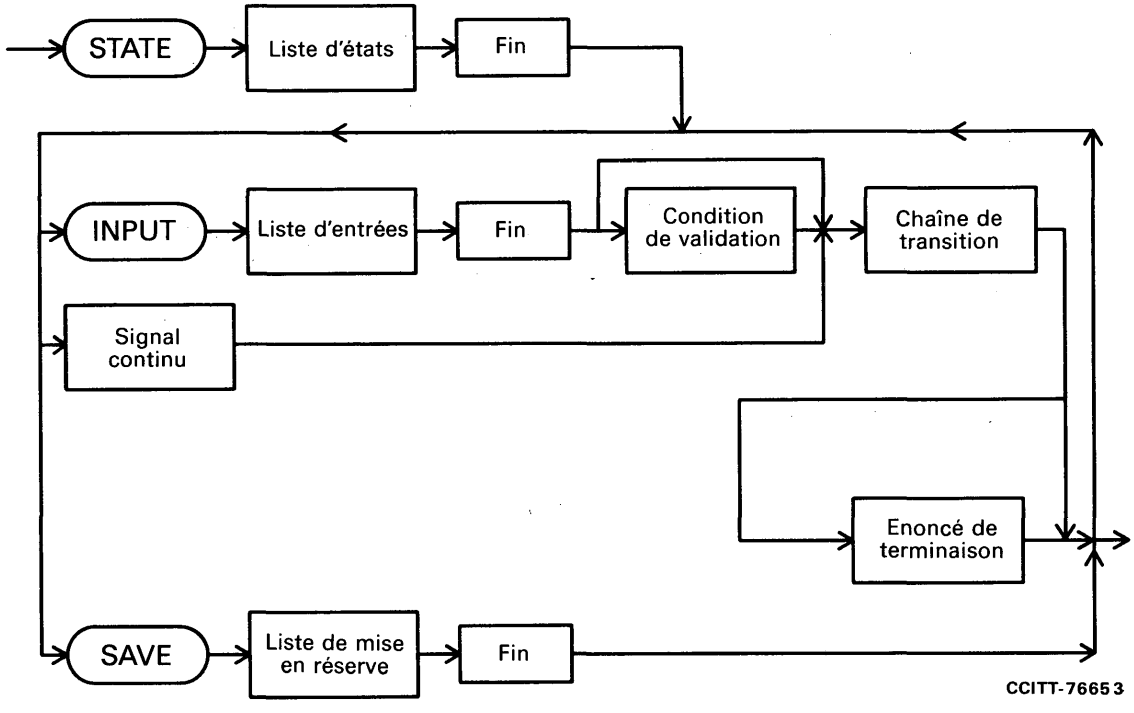
DÉFINITION DE VARIABLE DE PROCÉDURE



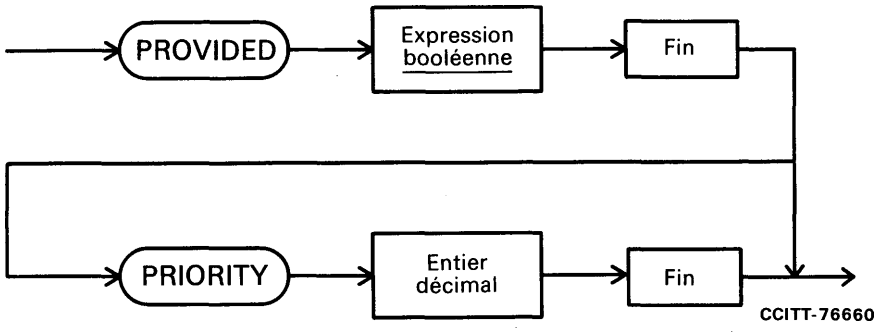
CORPS DE PROCESSUS



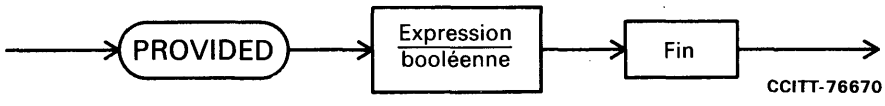
CORPS D'ÉTAT



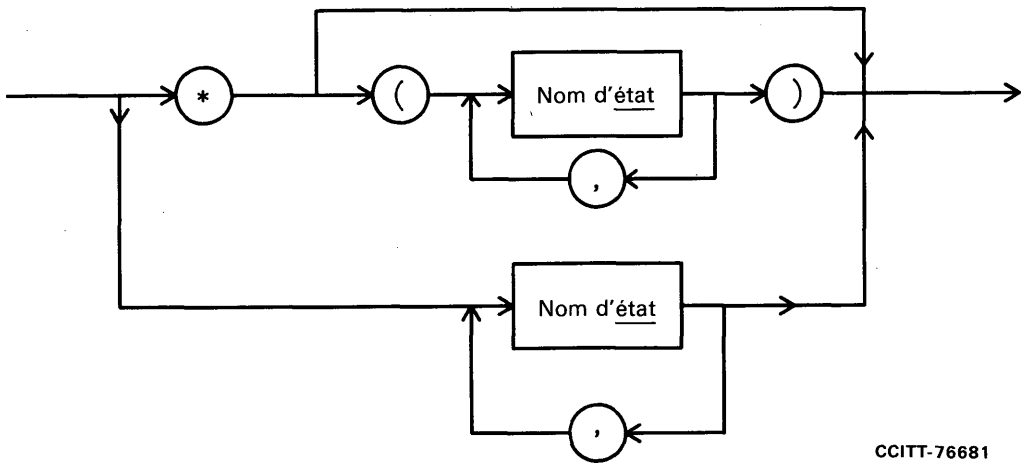
SIGNAL CONTINU



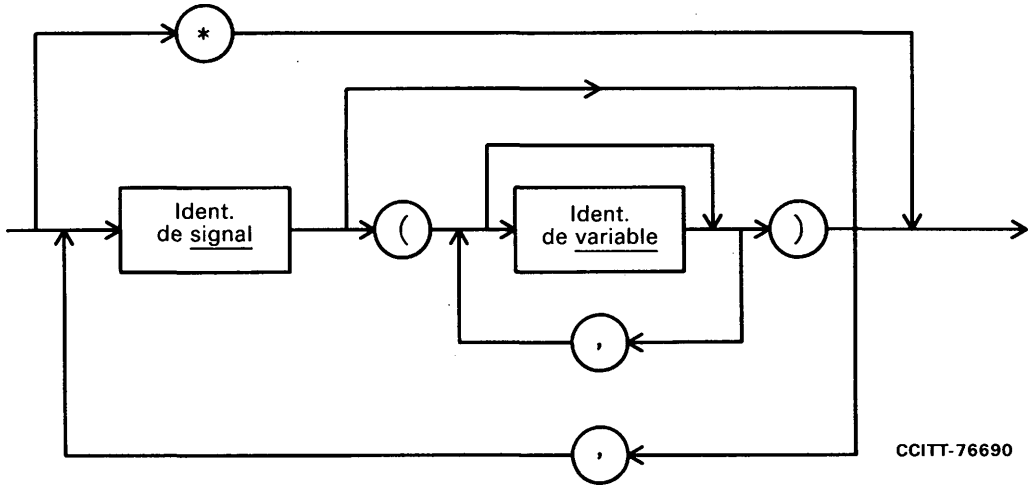
CONDITION DE VALIDATION



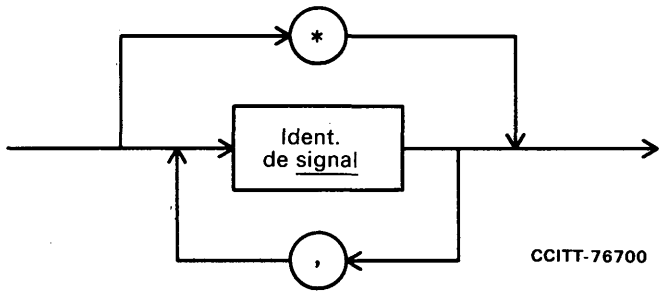
LISTE D'ÉTATS



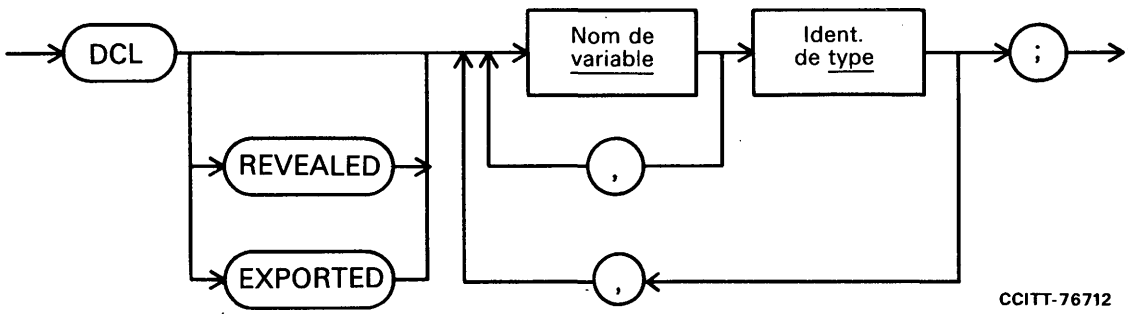
LISTE D'ENTRÉES



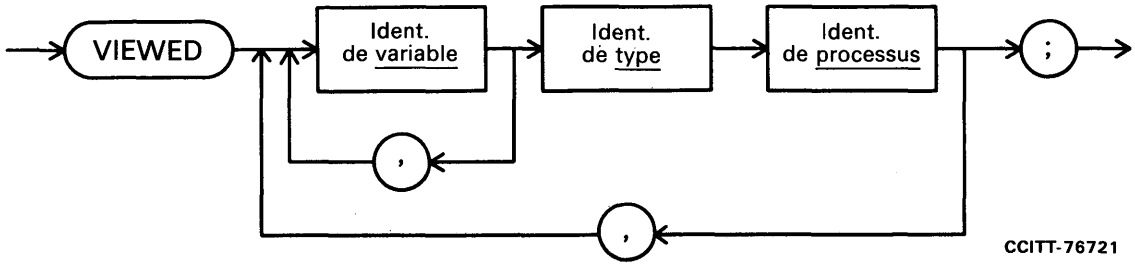
LISTE DE MISE EN RÉSERVE



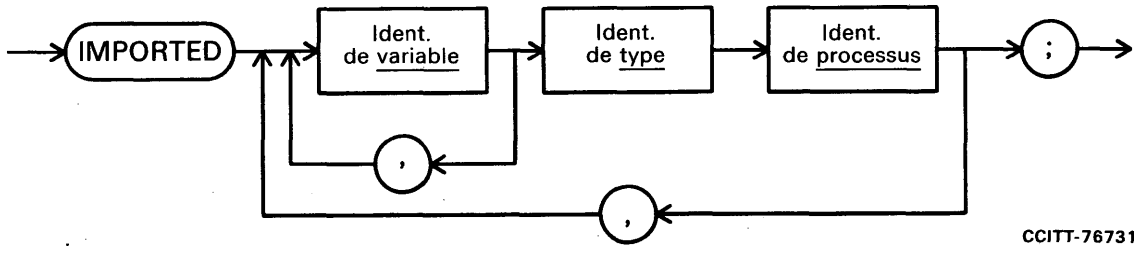
DÉFINITION DE VARIABLE



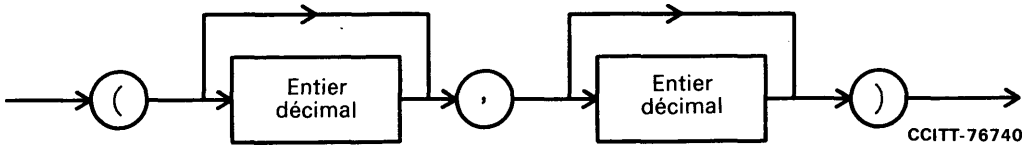
DÉFINITION DE VISIBILITÉ



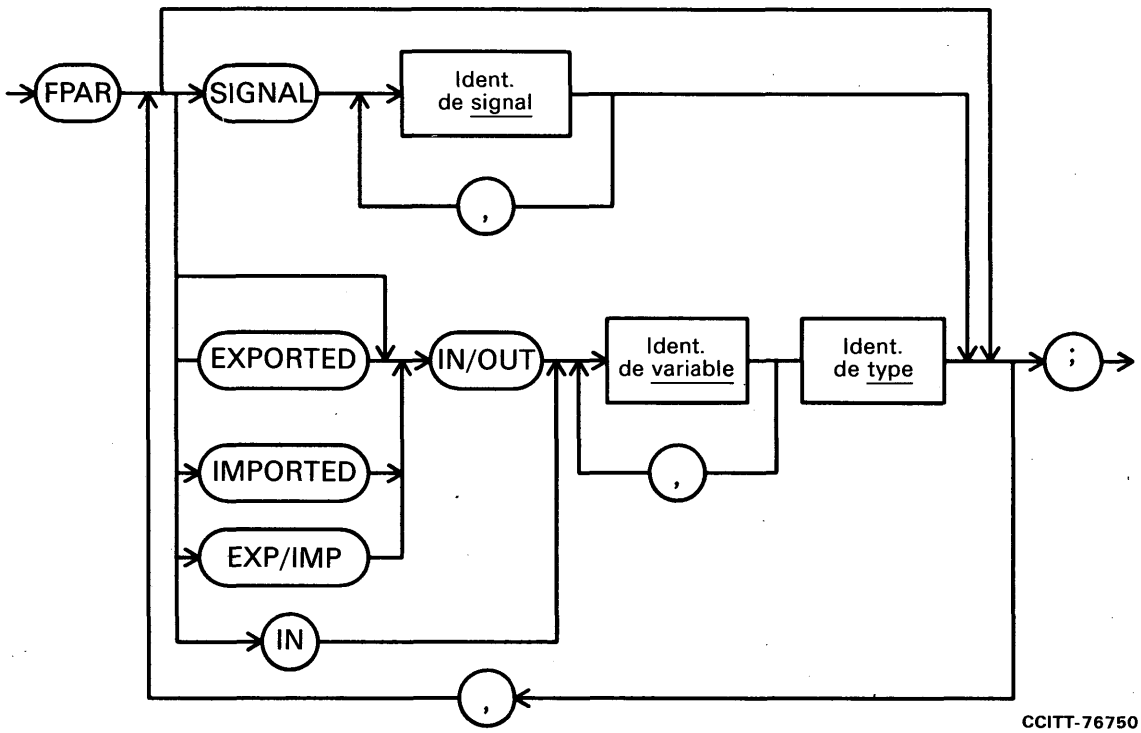
DÉFINITION D'IMPORTATION



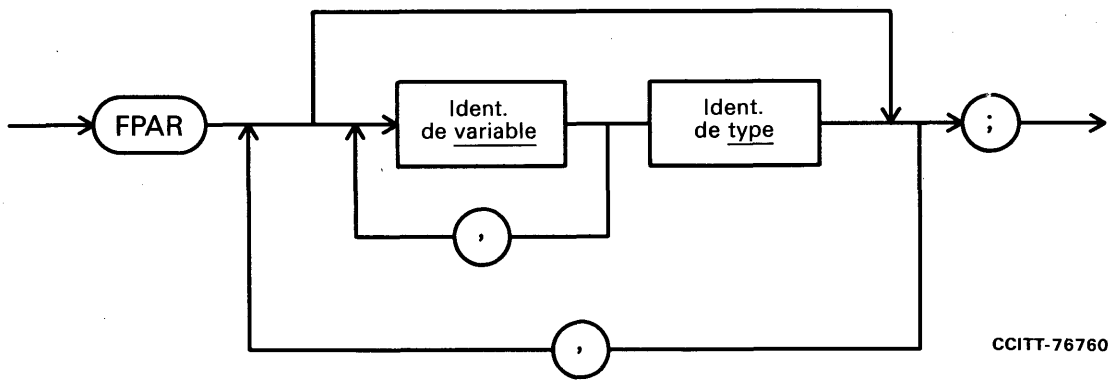
NOMBRE D'INSTANCES



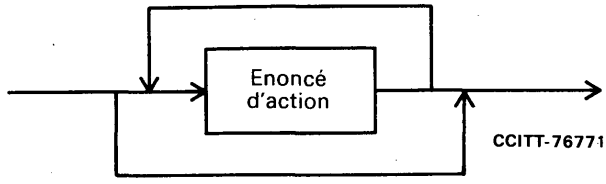
PARAMÈTRES FORMELS DE PROCÉDURE



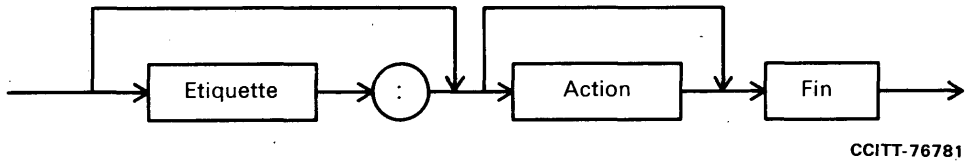
PARAMÈTRES FORMELS



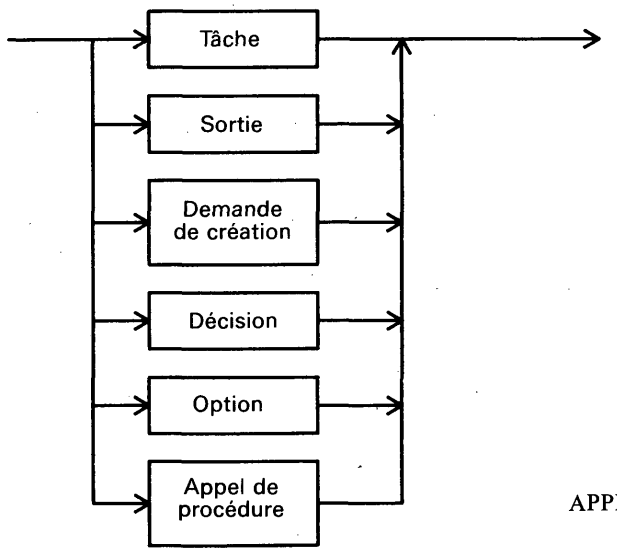
CHAÎNE DE TRANSITION



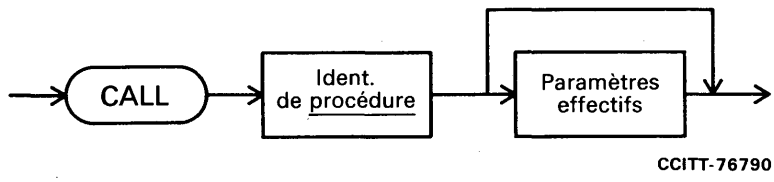
ÉNONCÉ D'ACTION



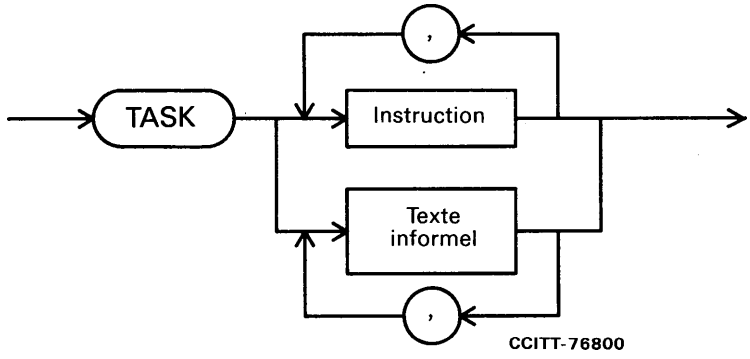
ACTION



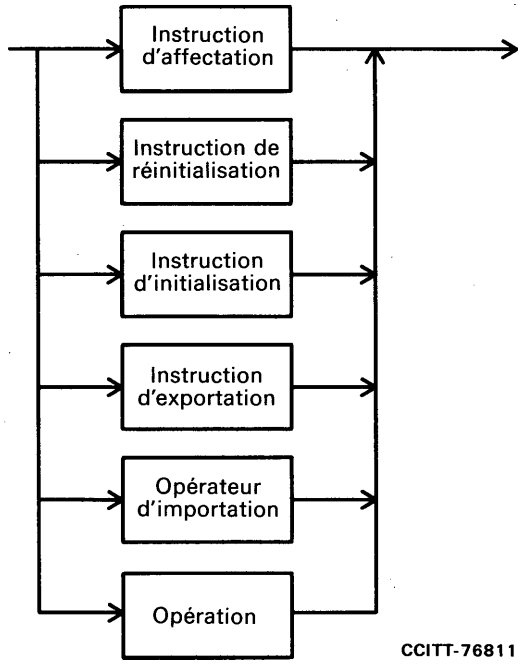
APPEL DE PROCÉDURE



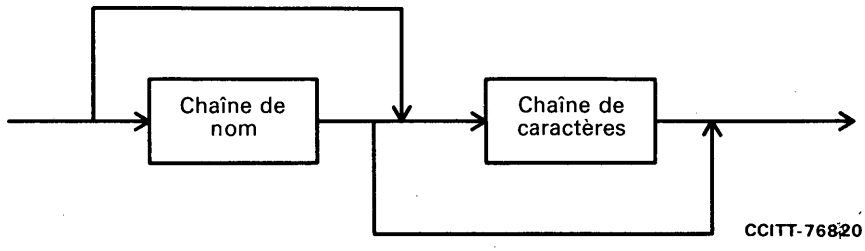
TÂCHE



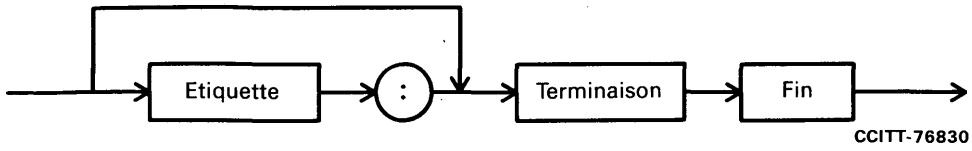
INSTRUCTION



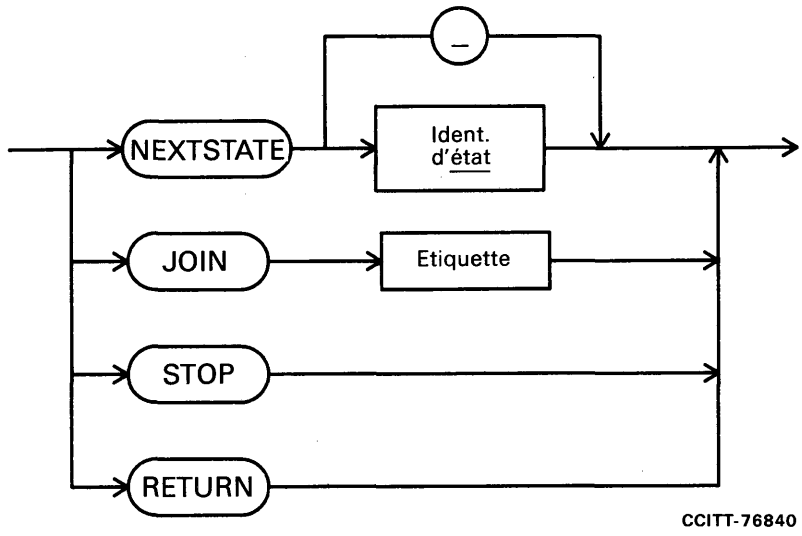
TEXTE INFORMEL



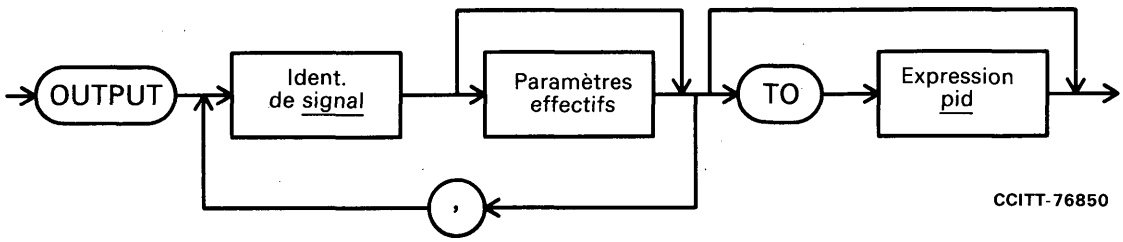
ÉNONCÉ DE TERMINAISON



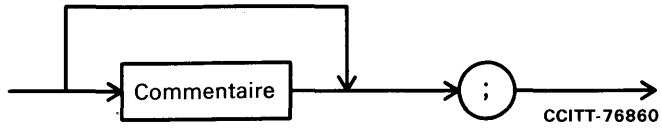
TERMINAISON



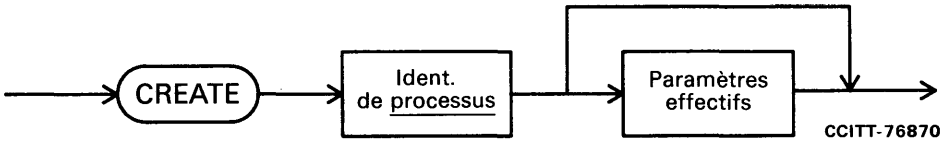
SORTIE



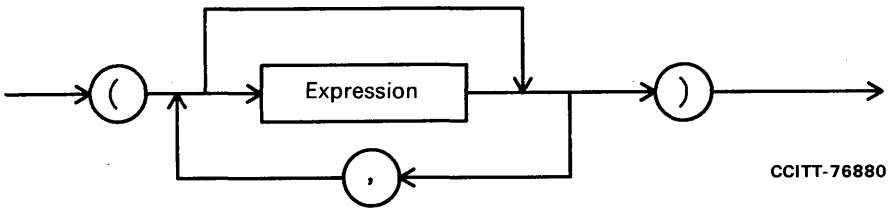
FIN



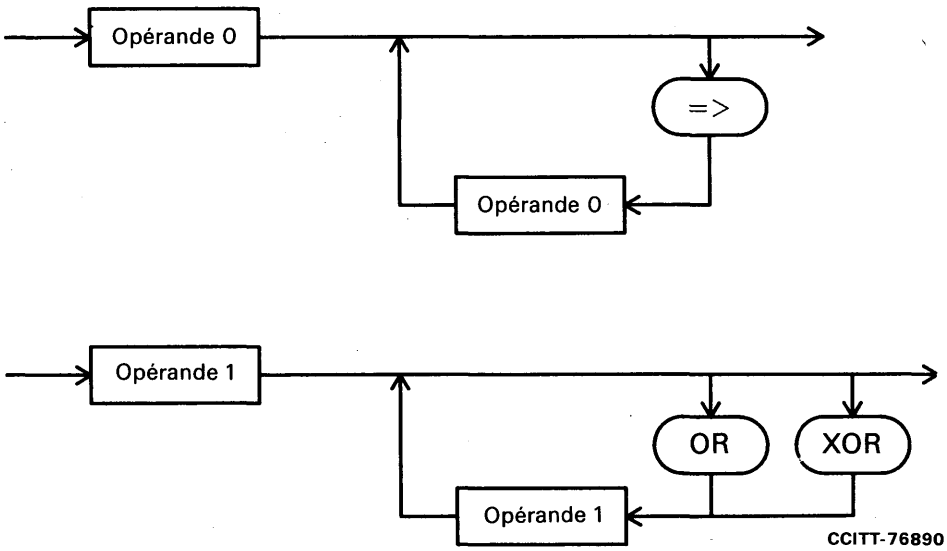
DEMANDE DE CRÉATION

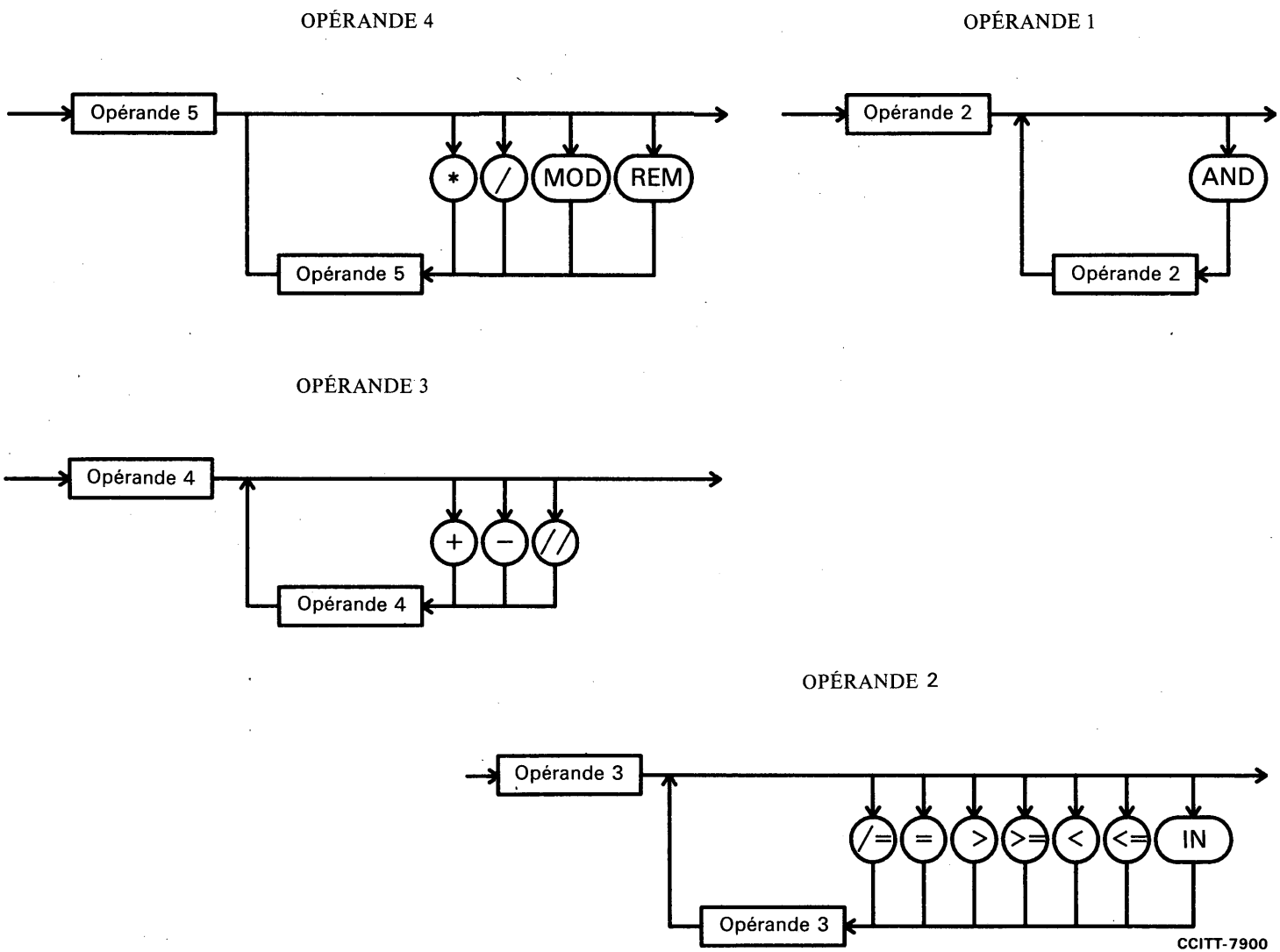


PARAMÈTRES EFFECTIFS

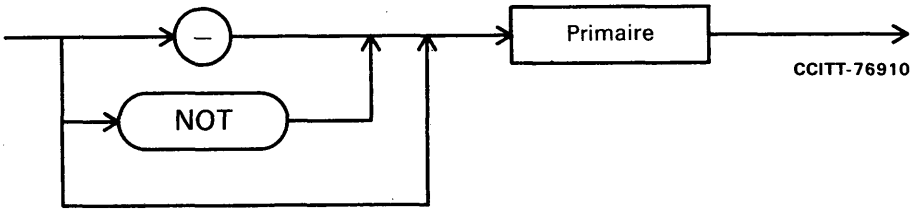


EXPRESSION

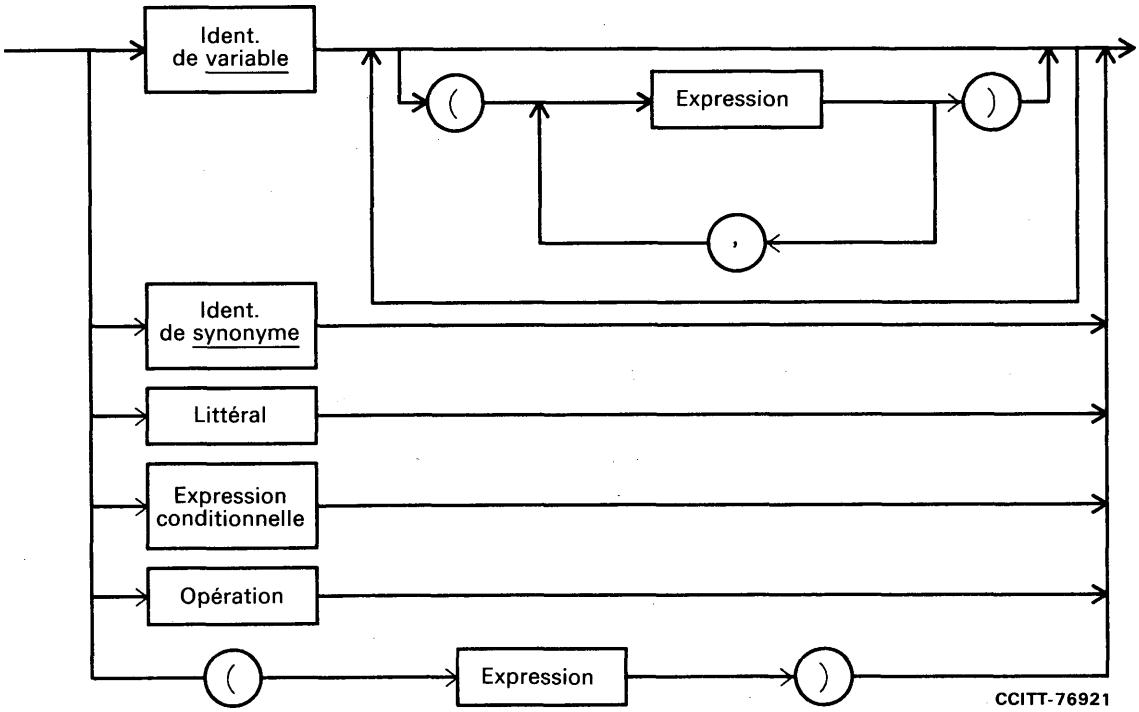



OPÉRANDE 4
Opérande 5
*
/
MOD
REM
Opérande 5
OPÉRANDE 1
Opérande 2
AND
Opérande 2
OPÉRANDE 3
Opérande 4
+
–
//
Opérande 4
OPÉRANDE 2
Opérande 3
/=
=
>
>=
<
<=
IN
Opérande 3
CCITT-7900

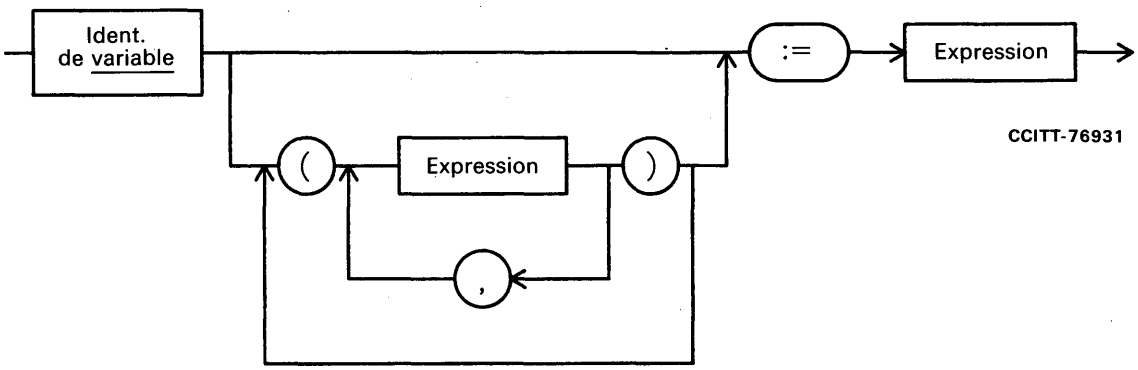
OPÉRANDE 5



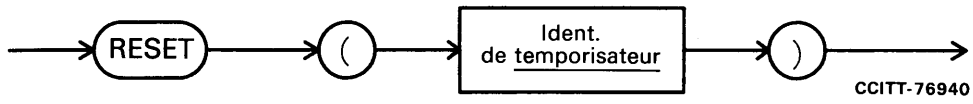
PRIMAIRE



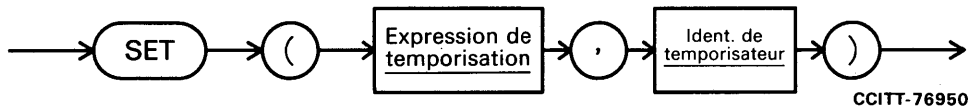
INSTRUCTION D'AFFECTATION



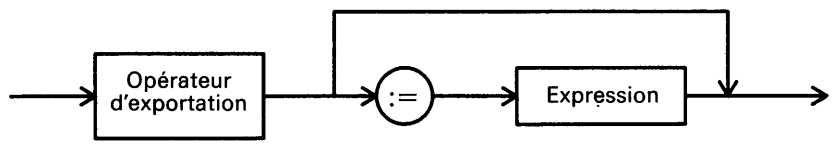
INSTRUCTION DE RÉINITIALISATION



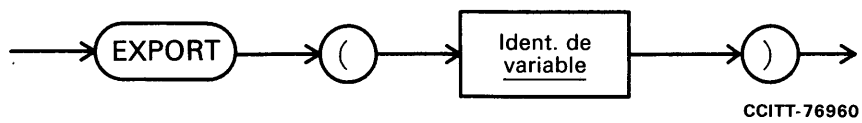
INSTRUCTION D'INITIALISATION



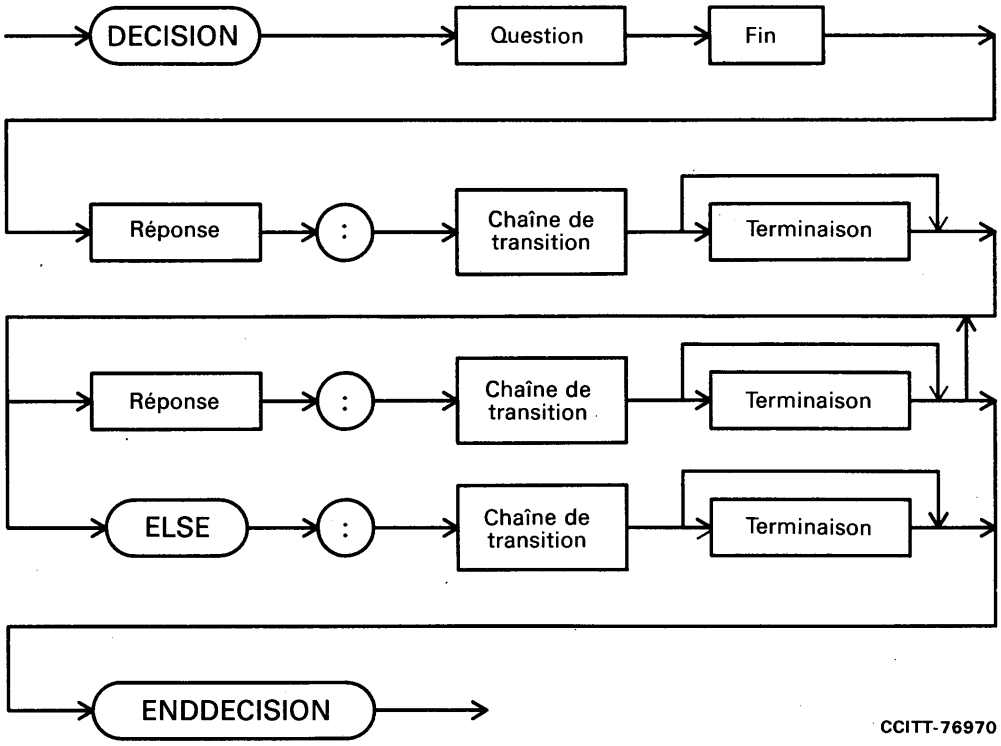
INSTRUCTION D'EXPORTATION



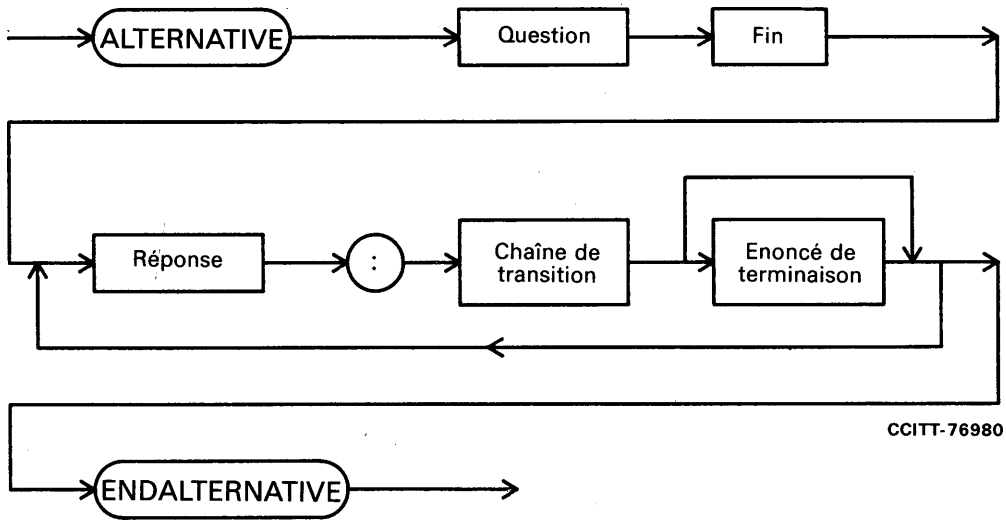
OPÉRATEUR D'EXPORTATION



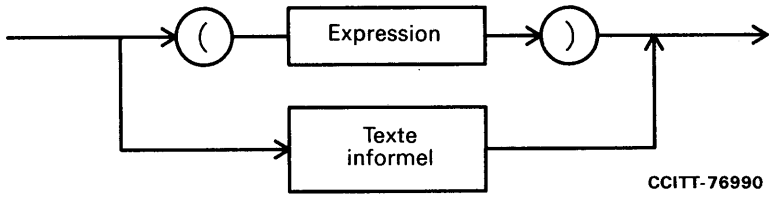
DÉCISION



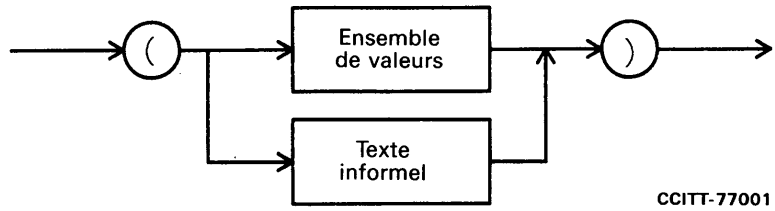
OPTION



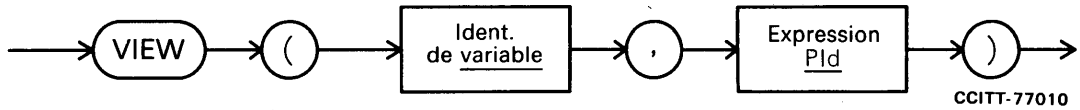
QUESTION



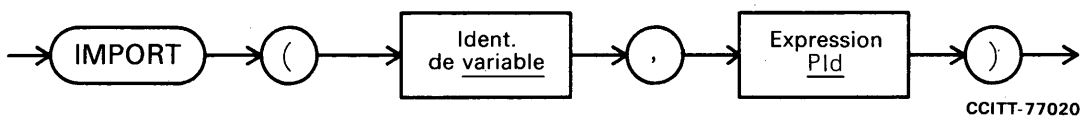
RÉPONSE



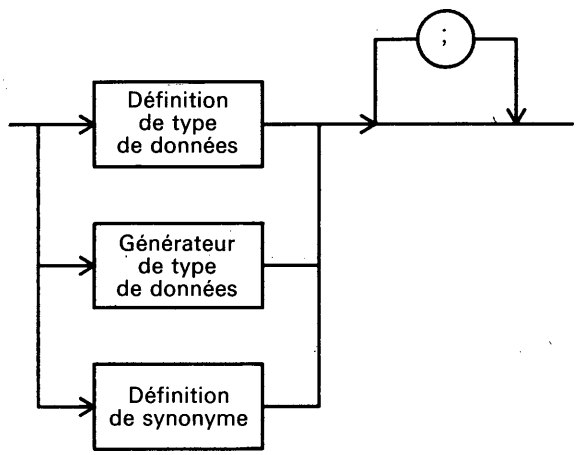
OPÉRATEUR DE VISIBILITÉ



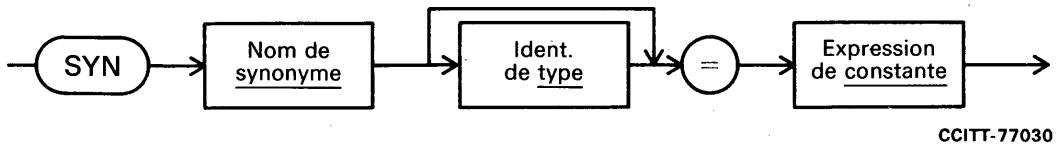
OPÉRATEUR D'IMPORTATION

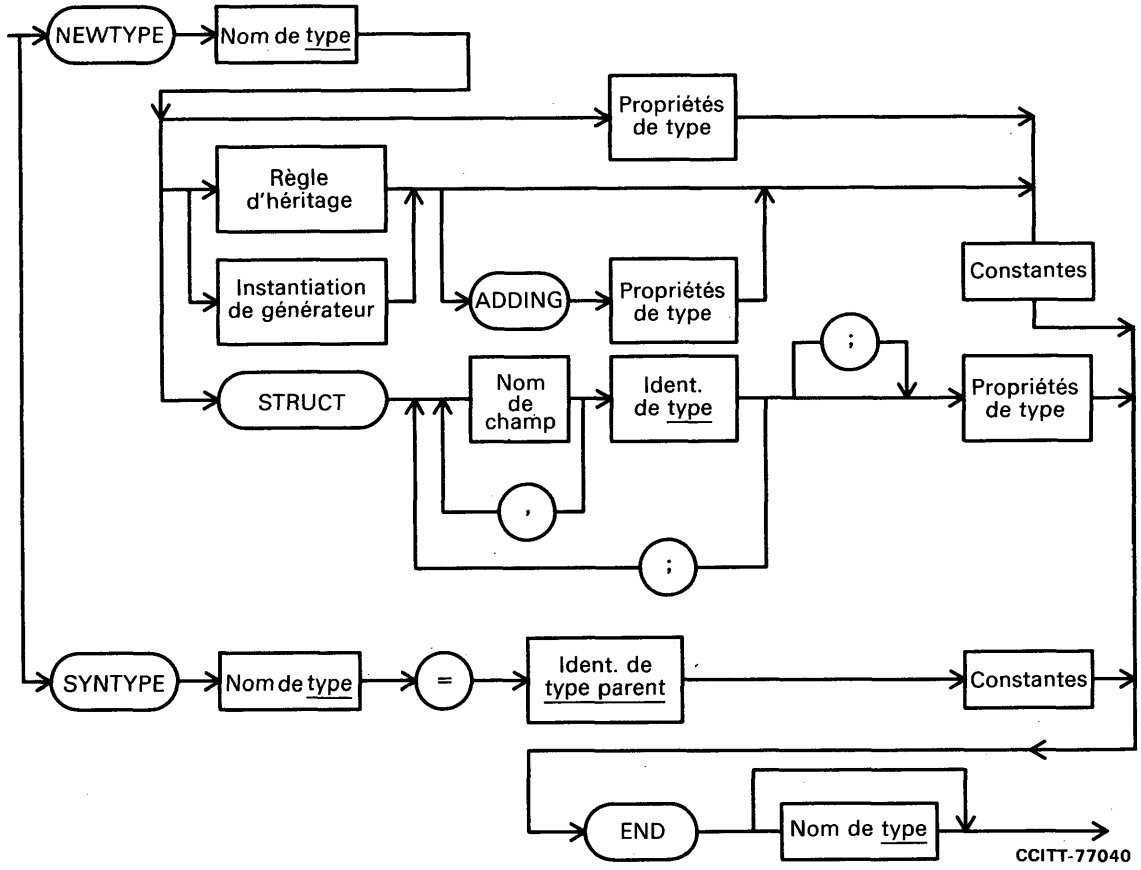


DÉFINITION DE DONNÉES

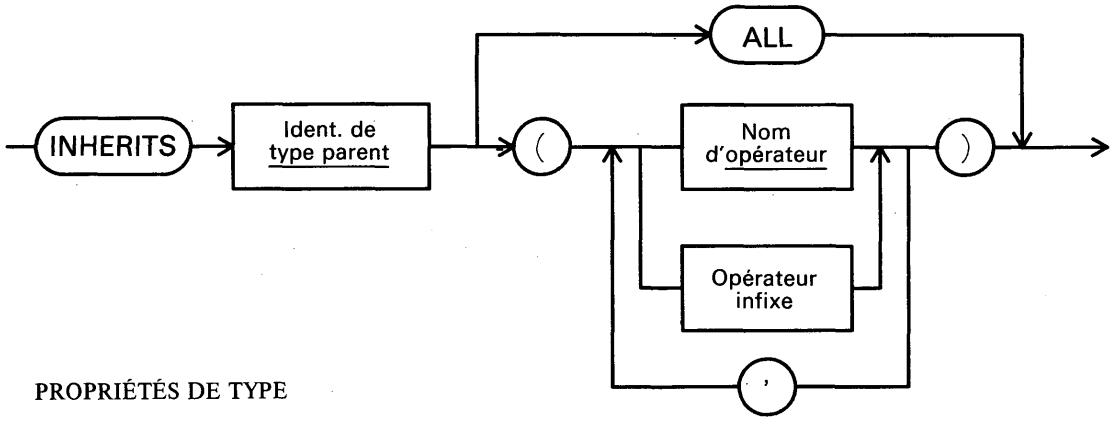


DÉFINITION DE SYNONYME

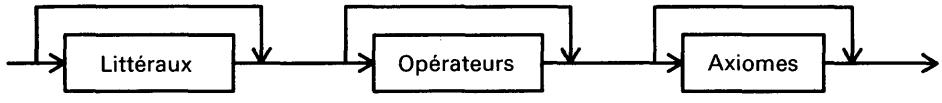




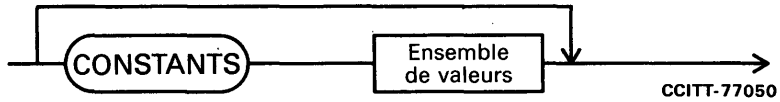
RÈGLE D'HÉRITAGE



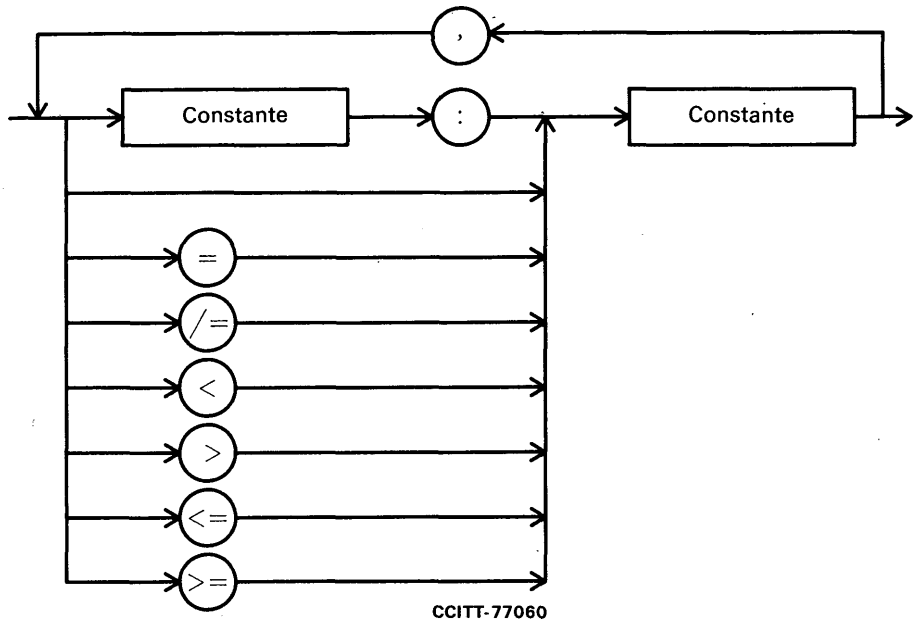
PROPRIÉTÉS DE TYPE



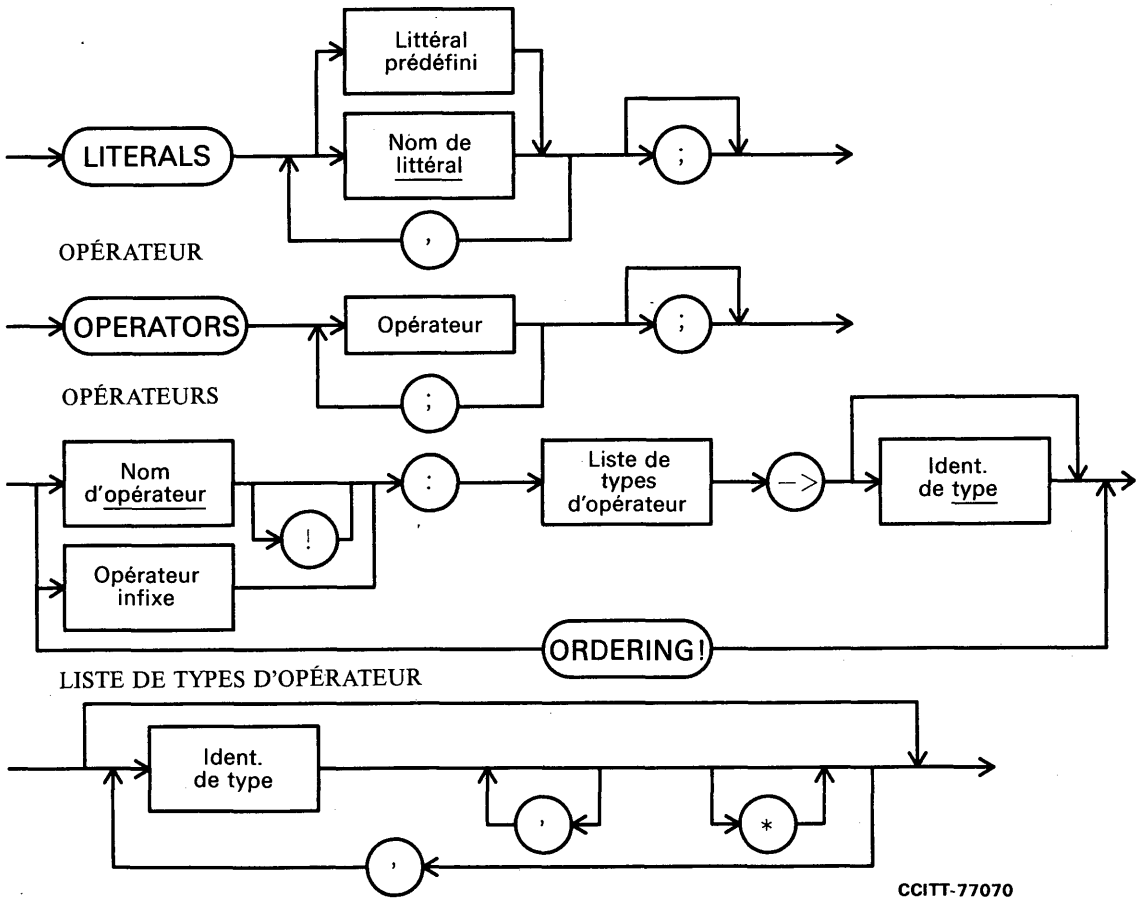
CONSTANTES



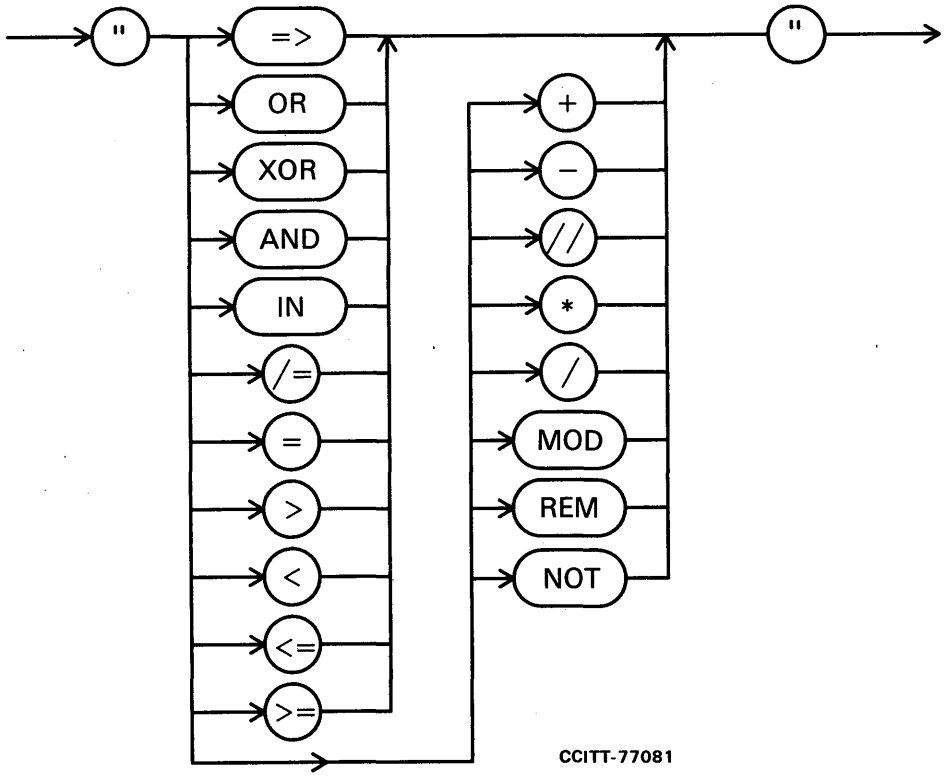
ENSEMBLE DE VALEURS



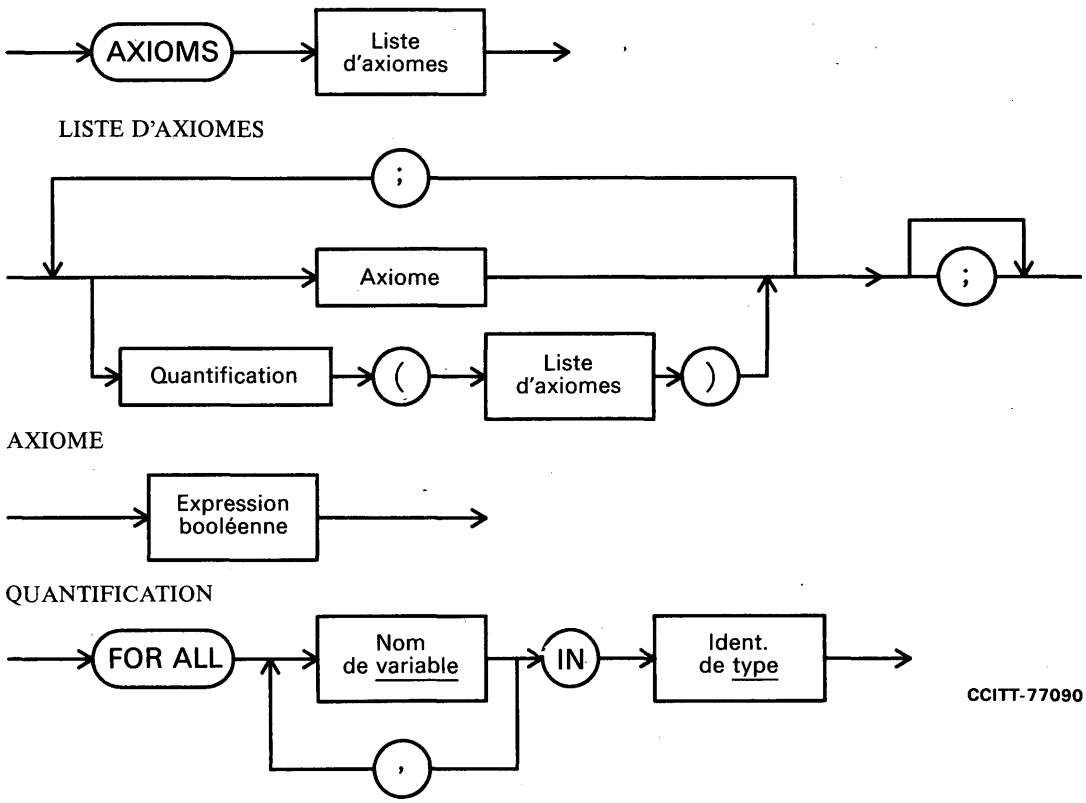
LITTÉRAUX



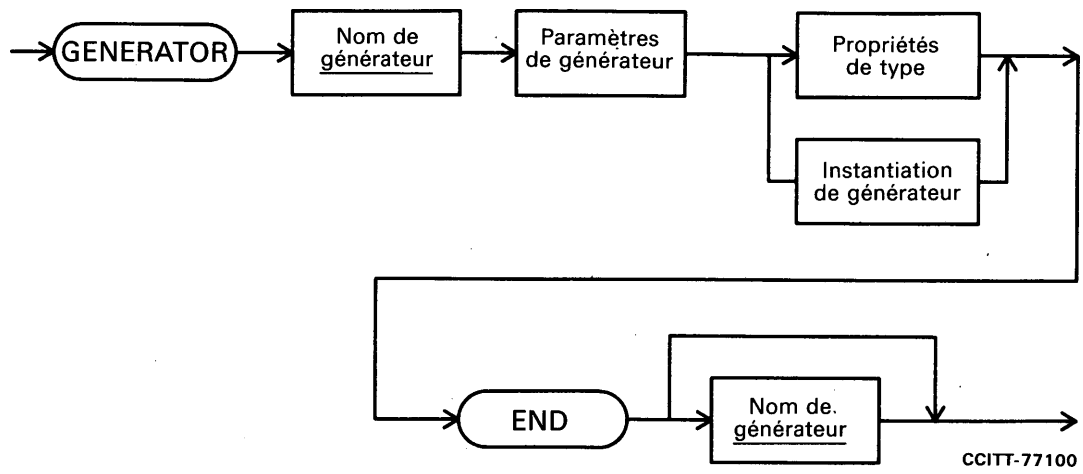
OPÉRATEUR INFIXE



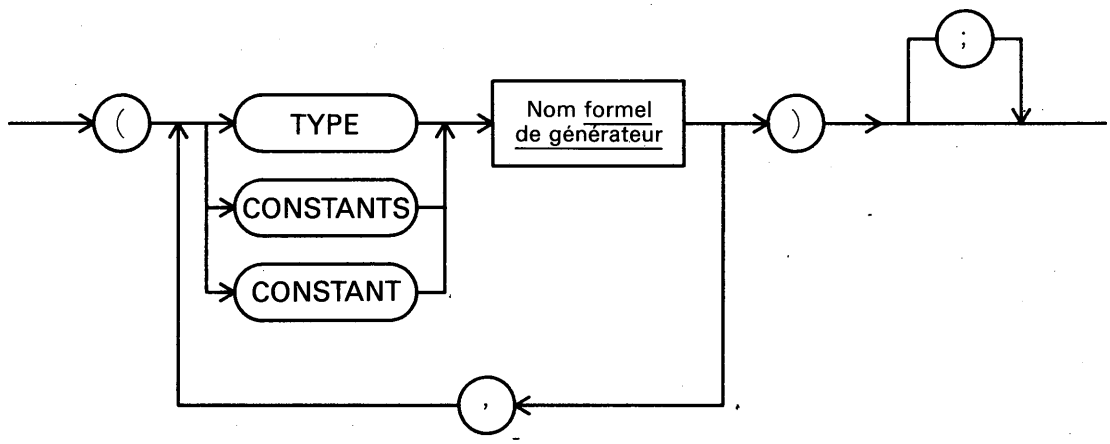
AXIOMES



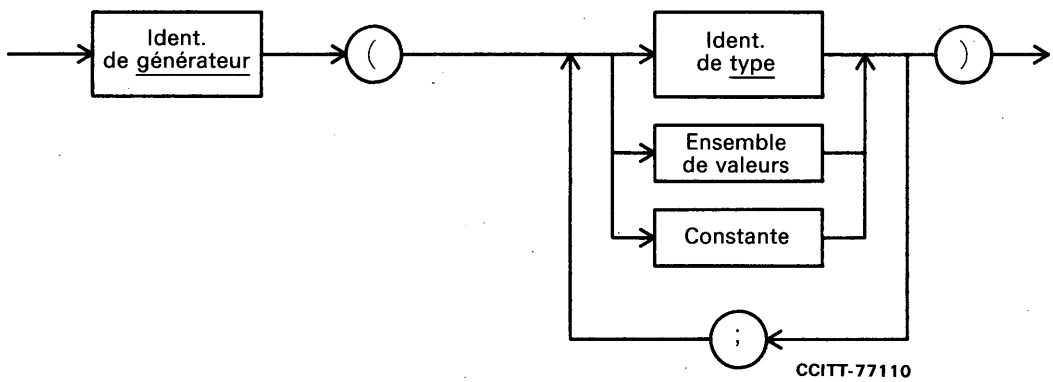
GÉNÉRATEUR DE TYPE DE DONNÉES



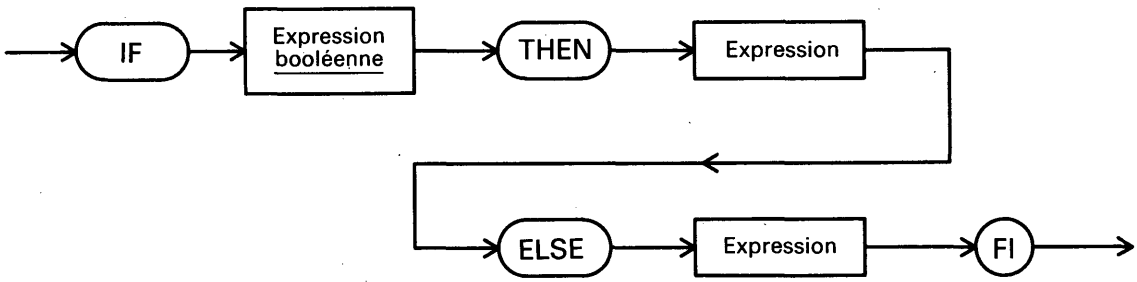
PARAMÈTRES DE GÉNÉRATEUR



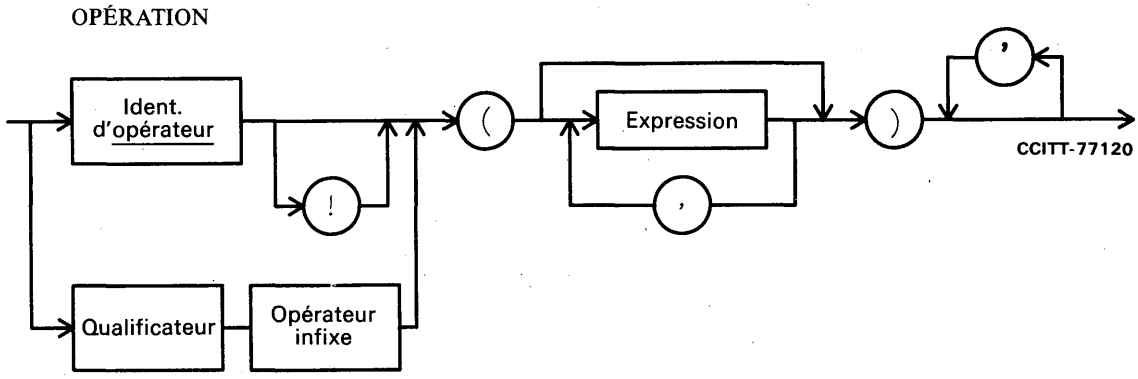
INSTANTIATION DE GÉNÉRATEUR



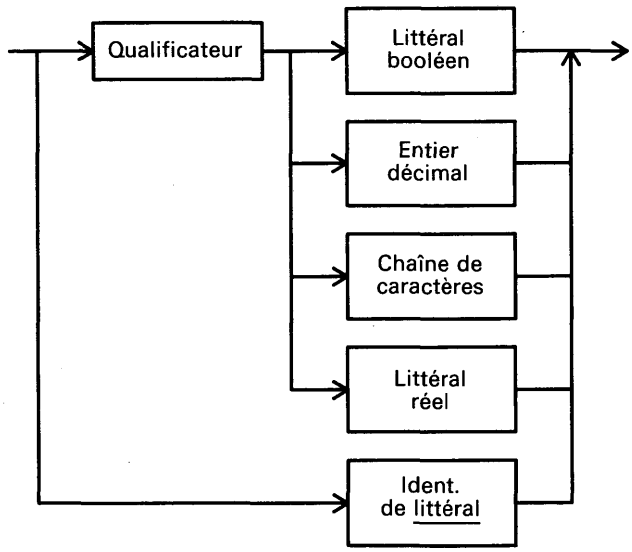
EXPRESSION CONDITIONNELLE



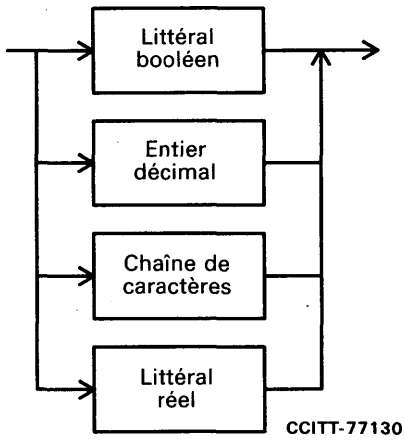
OPÉRATION



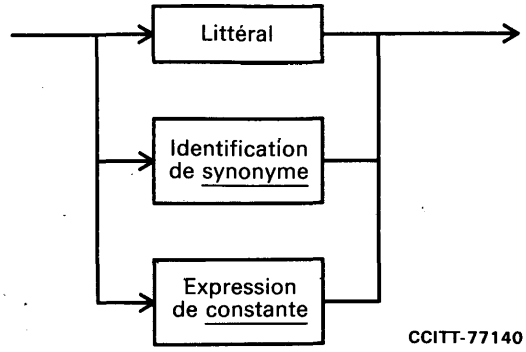
LITTÉRAL



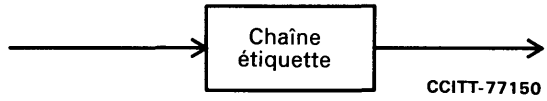
LITTÉRAL PRÉDÉFINI



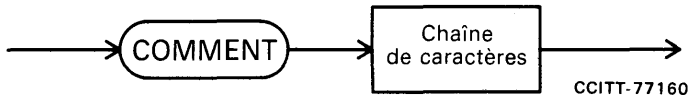
CONSTANTE



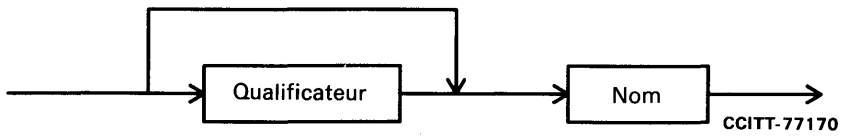
ÉTIQUETTE



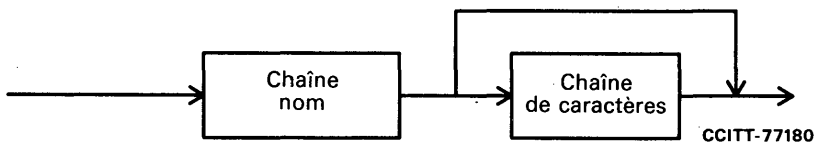
COMMENTAIRE



IDENT.



NOM



C2.2 *Règles lexicales*

1) Tous les signes de ponctuation [par exemple: , . ; ' : ! = ( )] et les symboles de fonctionnement (par exemple: +, −, *, <, > ...) sont des unités lexicales qui peuvent être insérées à la place des espaces.

2) Deux unités lexicales doivent être séparées d'un ou de plusieurs espaces.

3) Les mots clés appartiennent à la même catégorie lexicale, telle que la chaîne de noms, et ils sont réservés.

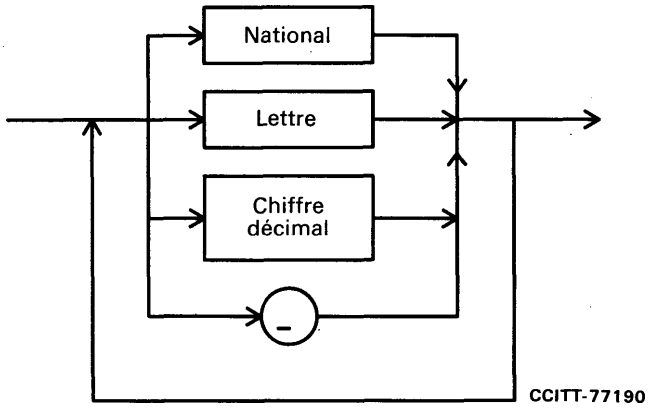
4) En dehors des unités lexicales, plusieurs espaces ont la même «signification» qu'un seul espace.

5) Les caractères de tabulation (VT, HT, CR, BS...) peuvent être considérés comme des espaces.

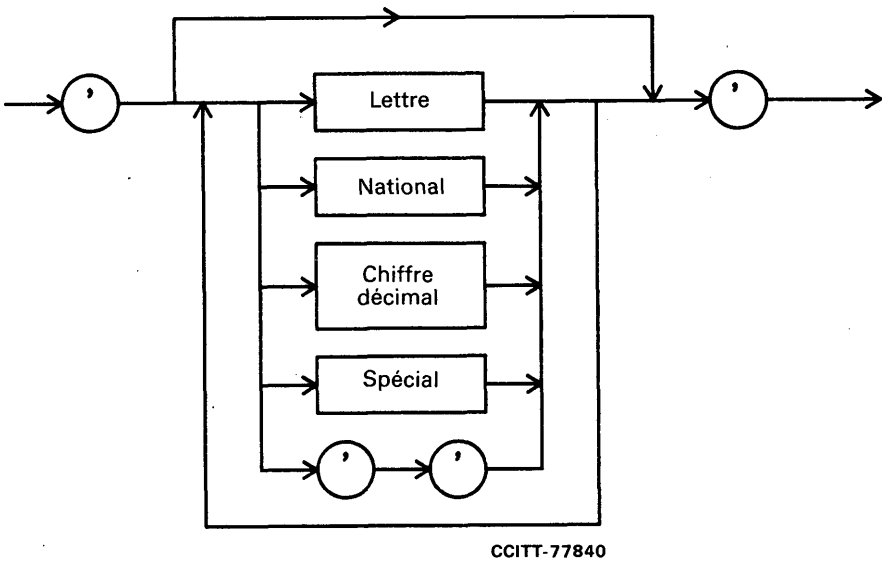
6) Toutes les lettres et tous les caractères nationaux sont toujours considérés comme étant en majuscules, sauf dans une chaîne de caractères.

7) Chaque fois qu'il existe des espaces, il est possible d'insérer des commentaires délimités par ' /*' et '*/', ces commentaires ayant la même signification qu'un espace. Le commentaire ne doit pas contenir la séquence spéciale: '*/'.

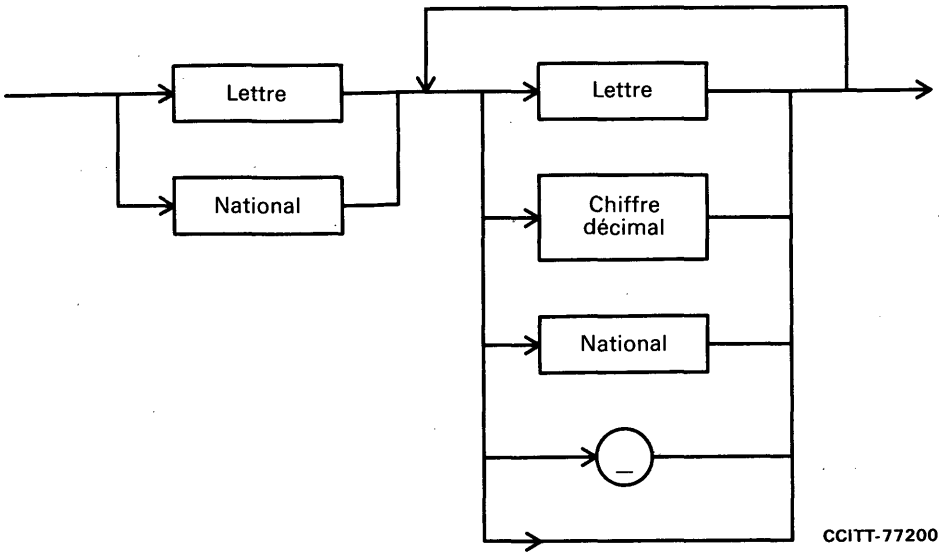
CHAÎNE D'ÉTIQUETTES



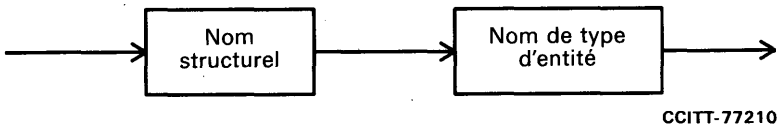
CHAÎNE DE CARACTÈRES



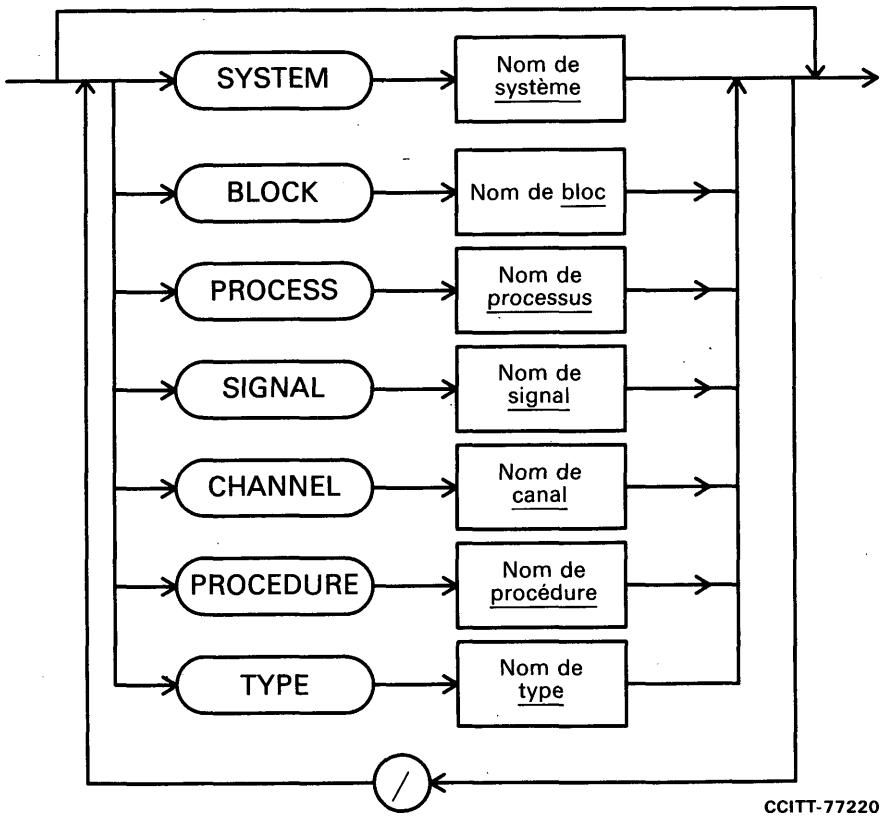
CHAINE NOM



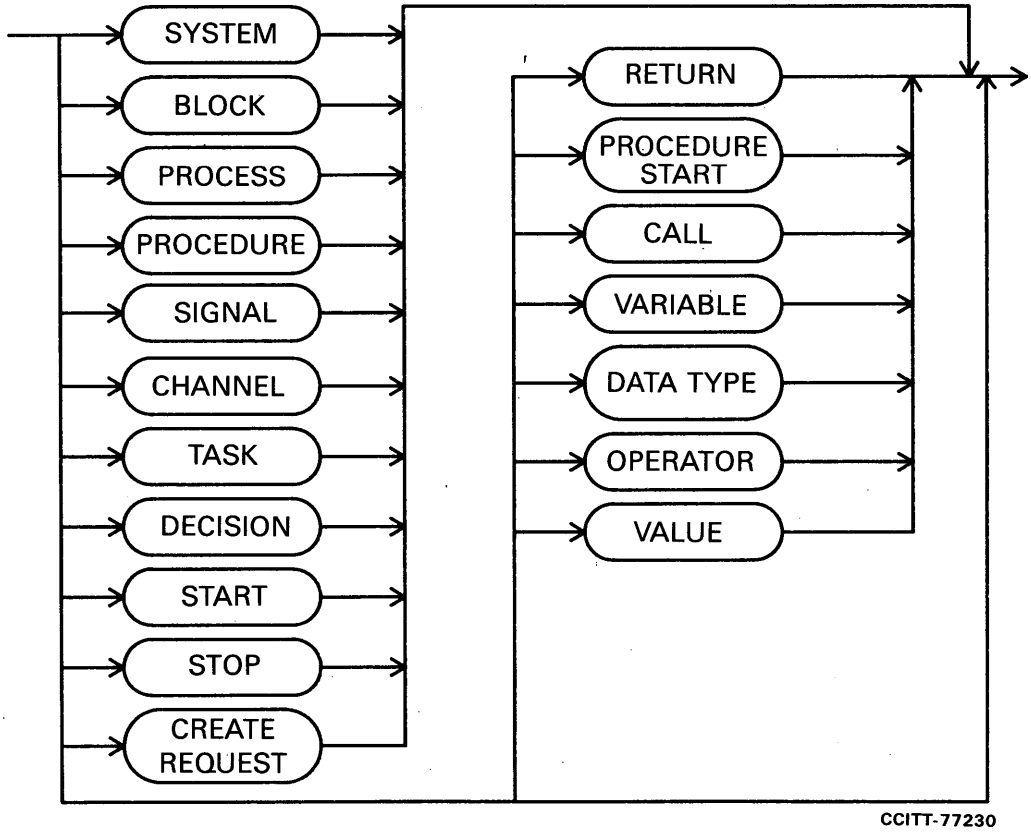
QUALIFICATEUR



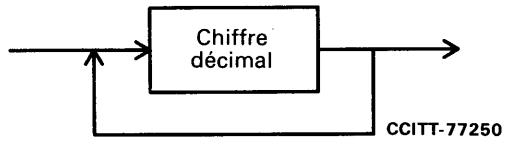
NOM STRUCTUREL



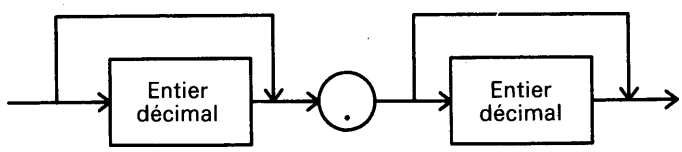
NOM DE TYPE D'ENTITÉ



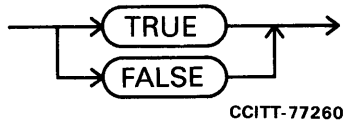
ENTIER DÉCIMAL



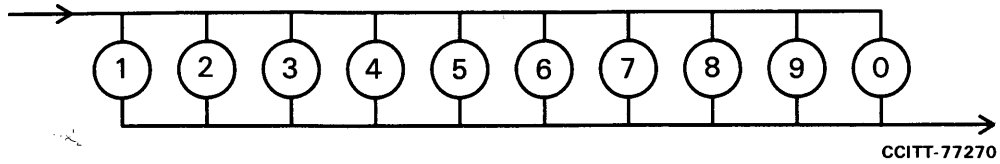
LITTÉRAL RÉEL



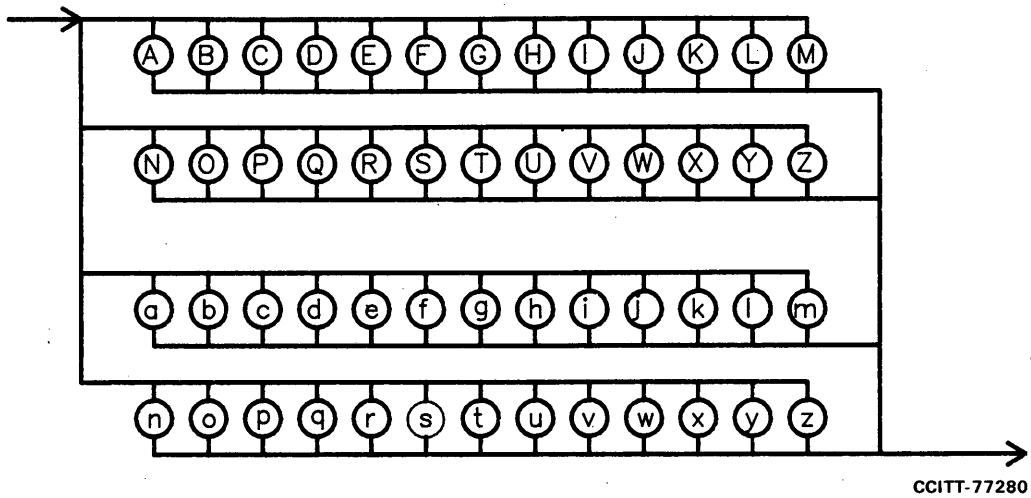
LITTÉRAL BOOLÉEN



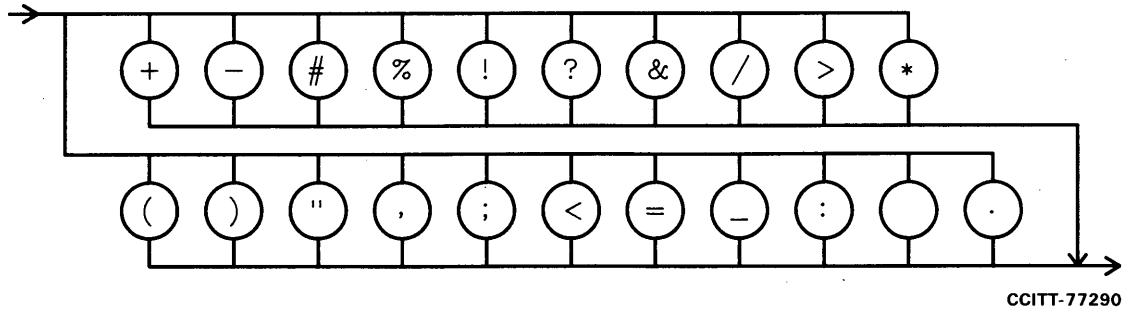
CHIFFRE DÉCIMAL



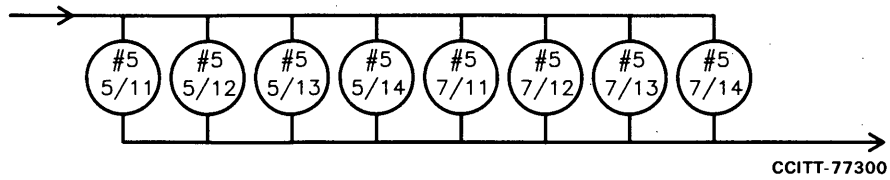
LETTRE



SPÉCIAL

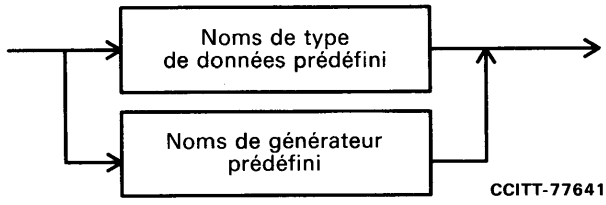


NATIONAL

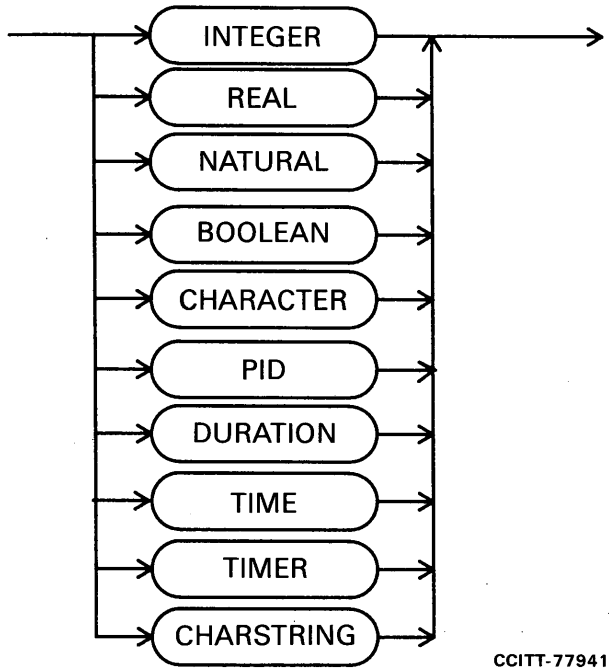


Remarque – Les positions susmentionnées renvoient aux positions de l'Alphabet international n.º 5 du CCITT, réservées pour une utilisation nationale.

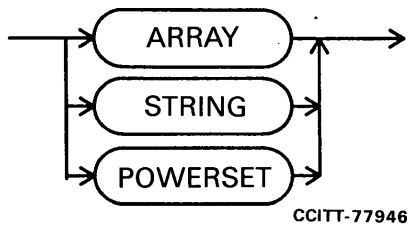
NOM DE TYPE PRÉDÉFINI



NOMS DE TYPE DE DONNÉES PRÉDÉFINI



NOMS DE GÉNÉRATEUR PRÉDÉFINI



C2.3 *Mots réservés utilisés en PR*

SYSTEM
ENDSYSTEM
BLOCK
ENDBLOCK
SUBSTRUCTURE
ENDSUBSTRUCTURE
SPLIT
INTO
SUBBLOCKS
CHANNELS
SIGNALS
IN
CHANNEL
FROM
ENV
TO
WITH
IMPORTEDVALUES
SIGNAL
INCOMING
OUTGOING
BLOCKS
DECISION
ELSE
ENDDECISION
ALTERNATIVE
ENDALTERNATIVE
VIEW
EXPORT
IMPORT
CREATEREQUEST
CALL
DATATYPE
VALUE
PROCESS
ENDPROCESS
PROCEDURE
ENDPROCEDURE
DCL
START
STATE
INPUT
SAVE
PROVIDED
PRIORITY
REVEALED
EXPORTED
VIEWED
IMPORTED
FPAR
EXPORTED/IMPORTED
IN/OUT
IN
TASK
NEXTSTATE
JOIN
STOP
RETURN
OUTPUT
CREATE
COMMENT
RESET
SET
EXPORT
PROCEDURESTART
VARIABLE
OPERATOR

Remarque 1 – L'appel de macro peut être placé dans n'importe quel diagramme en employant la syntaxe:

MACRO nom de macro.

L'expansion de macro est une partie d'un programme LDS/PR commençant par:

MACRO EXPANSION nom de macro;

et se terminant par:

ENDMACRO nom de macro.

Dans le dernier énoncé, le nom de macro n'est pas obligatoire.

Remarque 2 – Dans le diagramme de définition de bloc, la règle suivante est applicable:

- s'il n'existe aucune définition de sous-structure de bloc, il doit exister au moins une définition de processus (qui peut être définie explicitement dans un autre module);
- s'il existe une définition de sous-structure de bloc, il existe une définition de sous-structure de processus pour chaque définition de processus contenue dans la définition de bloc.

ANNEXE D

(aux Recommandations Z.100 à Z.104)

Directives pour les usagers du LDS

Les présentes directives se subdivisent en trois parties.

La première comprend la table des matières (D.0), la préface (D.1), l'introduction (D.2) et les domaines d'application du LDS (D.3), qui donnent des renseignements sur le LDS, son contenu et ses domaines d'application.

La deuxième partie contient une explication générale ainsi que les principes fondamentaux du LDS (D.4) et la structuration du LDS (D.5) et donne des précisions sur la manière de déterminer des modèles de systèmes en LDS et sur les concepts auxquels on a recours. C'est une partie de caractère général qui ne donne pas de détails sur les formes concrètes du LDS.

La dernière partie contient des directives pour la représentation de systèmes utilisant le LDS/GR (D.6), des directives pour la représentation de systèmes LDS/PR (D.7), des mises en concordance du LDS/GR, du LDS/PR et du CHILL (D.8) ainsi que des exemples d'application du LDS (D.9). On y trouvera également des précisions sur l'emploi des deux formes existantes du LDS, à savoir le LDS/GR et le LDS/PR. Finalement, on trouvera une section sur les outils pour le LDS (D.10).

TABLE DES MATIÈRES

D.1 *Préface*

Le langage de spécification et de description du CCITT (LDS) a tout d'abord fait l'objet des Recommandations Z.101 à Z.103 (Tome VI.4 du Livre orange, 1976) puis, sous une forme développée, des Recommandations Z.101 à Z.104 (Livre jaune, 1980) qui ont été complétées et refondues en 1984 dans les Recommandations Z.100 à Z.104 du présent fascicule du Livre rouge.

La Commission d'études XI est consciente de la nécessité d'établir des directives à l'intention des usagers pour faciliter l'application du LDS à une large gamme de processus intervenant dans les systèmes de commutation. Ces directives doivent permettre aux usagers de mieux comprendre les Recommandations relatives au LDS et de les appliquer à divers domaines. Il est difficile d'établir un jeu complet de directives, le domaine d'application de ce langage ne cessant d'évoluer. En conséquence, les directives données dans le présent fascicule devront elles aussi être remaniées et développées au cours de la prochaine période d'études 1984-1988; le CCITT est fort désireux de recevoir des contributions faisant part d'enseignements recueillis dans l'application pratique du LDS.

Aujourd'hui, en 1984, il est évident que l'emploi du LDS est de plus en plus répandu au sein du CCITT et des organisations qui en sont membres, et que la gamme des applications de ce langage ne cessera de se développer. Les présentes directives sont établies à l'intention de ceux qui envisagent d'utiliser ou utilisent déjà le LDS: elles complètent les Recommandations Z.100 à Z.104 en y ajoutant des conseils judicieux et des exemples utiles. Il y aura certes quelques chevauchements entre les directives et les Recommandations; cela semble d'ailleurs souhaitable, si l'on veut que les directives soient autonomes et faciles à consulter. Ce sont néanmoins les Recommandations qui constituent le document de base.

D.2 *Introduction*

D.2.1 *Considérations générales*

Le LDS peut servir tant à la spécification (du fonctionnement que l'on attend d'un système) qu'à la description (du fonctionnement effectif d'un système). Il a été conçu pour spécifier et décrire le comportement des systèmes de commutation qui interviennent dans les télécommunications, mais peut également être utilisé dans une gamme d'applications plus large. De fait, le LDS convient particulièrement bien à tous les systèmes où il est possible de représenter correctement un comportement à l'aide de machines à états finis étendues et où l'on s'intéresse spécialement aux phénomènes d'interaction.

Le LDS peut également servir de point de départ à des méthodes de documentation permettant de représenter intégralement la spécification ou la description d'un système. Dans ce contexte, la signification de la spécification et de la description est liée à leur emploi dans le cycle de vie d'un système. Chacune décrit les propriétés fonctionnelles d'un système d'une façon abstraite. La description comprend généralement certains aspects liés à la conception (par exemple, traitement des erreurs); elle est généralement plus complète en ce qui concerne les détails fonctionnels. Chacune doit concorder avec le modèle concret du système. Elles servent donc toutes deux de spécifications avant la mise en œuvre du système, et de documentation (descriptions) après cette même mise en œuvre.

Le LDS peut servir à représenter à divers niveaux de détail les propriétés fonctionnelles d'un système, d'une fonction ou d'une facilité, qu'il s'agisse de leurs spécifications ou de leurs descriptions. Les propriétés fonctionnelles désignent certaines propriétés structurelles (diagramme d'interaction de blocs) ainsi que le comportement. Par «comportement», on entend la manière dont un système réagit à des signaux reçus (entrées), c'est-à-dire les actions qu'il exécute, par exemple, envoi de signaux (sorties), formulation de questions (aux fins de décision) et exécution de tâches.

Les spécifications peuvent être très générales quand une Administration souhaite étudier les possibilités de mise à jour d'un système en introduisant de nouvelles caractéristiques, de nouveaux services, de nouvelles techniques, etc., tout en laissant au fournisseur la possibilité d'offrir de très nombreuses solutions pratiques. Des spécifications de ce genre ne donneront généralement que peu de détails. A l'autre extrémité, il y a les spécifications par lesquelles une Administration demande le remplacement ou l'extension d'un central existant. Dans ce cas, les détails devront probablement être plus poussés, les spécifications des interfaces devant être très détaillées.

Une spécification et une description peuvent être identiques. Il est toujours préférable de concevoir les nouvelles réalisations à partir de la spécification, afin d'en garantir le respect.

D'une manière générale, ce sont les fournisseurs qui rédigent les descriptions pour donner suite à une spécification (ou pour décrire des systèmes que le fournisseur veut mettre sur le marché). Une description sera généralement plus détaillée que la spécification puisqu'il s'agit de rendre compte du comportement détaillé du système.

Il est à noter également que le LDS permet de décrire un système de manière plus ou moins formelle.

Premièrement, il est possible de décrire un système au moyen de constructions LDS associées au langage naturel. La description ainsi obtenue permet le transfert de l'information à un lecteur qui connaît le contexte, mais pas à une machine. Les contrôles pouvant être effectués automatiquement sont très limités.

Deuxièmement, il est possible d'associer aux constructions LDS des énoncés formels constitués d'éléments de types définis et d'opérateurs sur ces éléments. Les propriétés de ces éléments ne sont pas spécifiées; exemple: «Connecter A-B», où A et B sont du type abonné et «Connecter» est une opération autorisée pour ce type. La description ainsi obtenue permet le transfert de l'information aux lecteurs qui connaissent la signification des opérateurs utilisés. Une machine peut comprendre la description jusqu'à un certain niveau et peut procéder à certains contrôles; elle ne peut pas effectuer des contrôles complets ni «mettre en œuvre» le système car les propriétés des opérateurs sont inconnues.

Troisièmement, il est possible également d'indiquer toutes les propriétés de tous les opérateurs. Dans ce cas, la description est entièrement formelle; une machine peut effectuer tous les contrôles et, en principe, mettre en œuvre les systèmes décrits.

Selon l'objectif visé, les descriptions peuvent être adaptées aux besoins des usagers au moyen de différents niveaux de formalisme. Naturellement, plus la description est formelle, plus un être humain aura de la difficulté à la lire.

D.2.2 *Formes syntaxiques du LDS*

Tel qu'il est défini par les Recommandations Z.100 à 104, le LDS est un langage spécifique qui comporte deux syntaxes différentes, l'une et l'autre procédant du même modèle sémantique. La première s'appelle LDS/GR (LDS *G*raphical *R*epresentation) et repose sur un ensemble de symboles graphiques normalisés et de règles. L'autre s'appelle LDS/PR (LDS textual *P*hrase *R*epresentation) et repose sur des énoncés analogues à un langage de programmation. L'une et l'autre représentent les mêmes concepts du LDS.

D.2.3 *Le LDS fondé sur un modèle de machine à états finis étendue*

En cas d'emploi du LDS, le système à spécifier est représenté par un certain nombre de machines abstraites interconnectées. Une spécification complète comporte obligatoirement:

1) la définition de la structure du système en ce qui concerne les machines et leurs interconnexions,
2) le comportement dynamique de chaque machine, ses interactions avec les autres machines et avec l'environnement, et
3) les opérations sur les données associées aux interactions.

On décrit le comportement dynamique au moyen de modèles qui définissent les mécanismes de fonctionnement des machines abstraites ainsi que la communication entre les machines. La machine abstraite qu'emploie le LDS est une extension de la machine déterministe à états finis (FSM). La FSM est dotée d'une mémoire d'états finis internes et fonctionne avec un ensemble discret et fini d'entrées et de sorties. Pour chaque combinaison d'une entrée et d'un état, la mémoire définit une sortie ainsi que l'état suivant. On considère habituellement que la durée de transition entre deux états est nulle.

L'une des limites de la FSM est la suivante: toutes les données à mémoriser doivent être représentées sous la forme d'états explicites. Il est possible de représenter la plupart des systèmes de cette façon, mais ce n'est pas toujours pratique. On peut être appelé à mémoriser un grand nombre de valeurs importantes pour la future mise en séquence mais qui ne contribuent pas beaucoup à la compréhension globale du système. Cette information ne doit pas faire partie de l'espace des états explicites; en effet, ceci compliquerait la présentation. Il est possible pour ce genre d'applications d'étendre la FSM en la dotant d'une mémoire auxiliaire et d'une capacité de fonctionnement auxiliaire sur cette mémoire. Cette mémoire auxiliaire peut emmagasiner, par exemple, des informations concernant des adresses et des numéros d'ordre.

Les Recommandations relatives au LDS définissent deux opérations auxiliaires qu'il est possible d'inclure dans les transitions de la machine à états finis étendue (EFSM), à savoir les décisions et les tâches. Les «décisions» vérifient des paramètres associés aux entrées et aux données contenues dans la mémoire auxiliaire lorsque ces données sont importantes pour le séquencement de la machine principale. Les «tâches» exécutent des fonctions telles que le comptage, des opérations sur la mémoire auxiliaire et la manipulation de paramètres d'entrée et de sortie.

En LDS, des signaux représentent les interactions entre machines, c'est-à-dire que les EFSM reçoivent des signaux comme entrées et produisent des signaux comme sorties. Les signaux se composent d'un seul identificateur de signal et facultativement d'un ensemble de paramètres. Le LDS prévoit la possibilité d'un temps de transition différent de zéro, et définit un mécanisme théorique de mise en file d'attente «premier entré, premier sorti» pour les signaux qui parviennent à une machine en train d'exécuter une transition. Les signaux sont traités à tour de rôle, dans leur ordre d'arrivée.

D.3 *Applicabilité du LDS*

La figure D-1 donne un éventail de possibilités d'emploi du LDS aux fins d'achat ou de founiture de systèmes de commutation pour les télécommunications.

Dans cette figure, les rectangles représentent des groupes fonctionnels typiques, dont les noms précis, qui peuvent varier d'une organisation à l'autre, ayant des activités représentatives de plusieurs Administrations ou de plusieurs fabricants. Chacune des flèches (lignes de liaison) représente la circulation d'une série de documents entre un groupe fonctionnel et un autre; le LDS peut alors être employé en tant que partie de chacune de ces séries de documents. La figure, donnée seulement à titre d'illustration, n'est ni définitive ni complète.

Les domaines d'application sont ceux effectivement modélisés par des machines à états finis étendues, communiquant entre elles, par exemple: commutation téléphonique télex ou de données, systèmes de signalisation (par exemple, système de signalisation n° 7), interfonctionnement des systèmes de signalisation, protocoles pour données et interfaces d'usagers (LHM).

Si l'on considère plus spécialement les systèmes de commutation SPC, on peut citer comme exemple de fonctions pouvant être spécifiées ou décrites à l'aide du LDS: le traitement des appels (acheminement, signalisation, comptage, etc.), la maintenance, le traitement des dérangements (par exemple, alarmes, relève automatique des dérangements, configuration des systèmes, essais périodiques, etc.), la commande des systèmes (par exemple, protection contre les surcharges), et les interfaces homme-machine. Le § D.9 donne des exemples d'application du LDS.

La spécification des protocoles où intervient le LDS fait l'objet des Recommandations de la série X du CCITT, auxquelles sont annexées les directives auxiliaires pour les usagers concernant l'application du LDS dans ce domaine.

D.4 *Explication générale du LDS et de ses concepts*

Le présent paragraphe expose les concepts du LDS sans entrer dans le détail de leurs formes concrètes, à savoir le LDS/GR et le LDS/PR. Ces détails font respectivement l'objet des § D.6 et D.7.

D.4.1 *Aperçu du LDS*

Le LDS a été mis au point pour représenter les propriétés fonctionnelles telles qu'elles sont spécifiées et réalisées dans un système de télécommunications.

La spécification ou la description d'un système peuvent être subdivisées en deux catégories générales d'information. La première comprend la spécification fonctionnelle ou la description fonctionnelle. La seconde catégorie comprend les paramètres plus généraux du système et donne des renseignements sur l'environnement (limites de température, humidité, etc.), sur les limites de transmission, sur la qualité d'écoulement du trafic par les centraux, ces derniers renseignements n'étant pas représentés en LDS.

L'élaboration du LDS visait en grande partie à permettre aux usagers de se servir de vastes systèmes et de les fragmenter en morceaux suffisamment petits pour pouvoir être compris par lesdits usagers (l'auteur et le lecteur). Ainsi, chaque système représenté en LDS se compose de blocs. Les blocs sont reliés par des canaux qui indiquent le trajet de communication entre les blocs. Ces blocs et ces canaux peuvent à leur tour être subdivisés en canaux et en blocs.

Au niveau inférieur (et souvent aux niveaux plus élevés), les blocs renferment un ou plusieurs processus. Les processus sont les machines à états finis étendues, communiquant entre elles, qui modélisent le comportement dynamique du système. Ces processus peuvent également être vastes et complexes. Des procédures permettent de fragmenter un processus en éléments plus petits, faciles à manier.

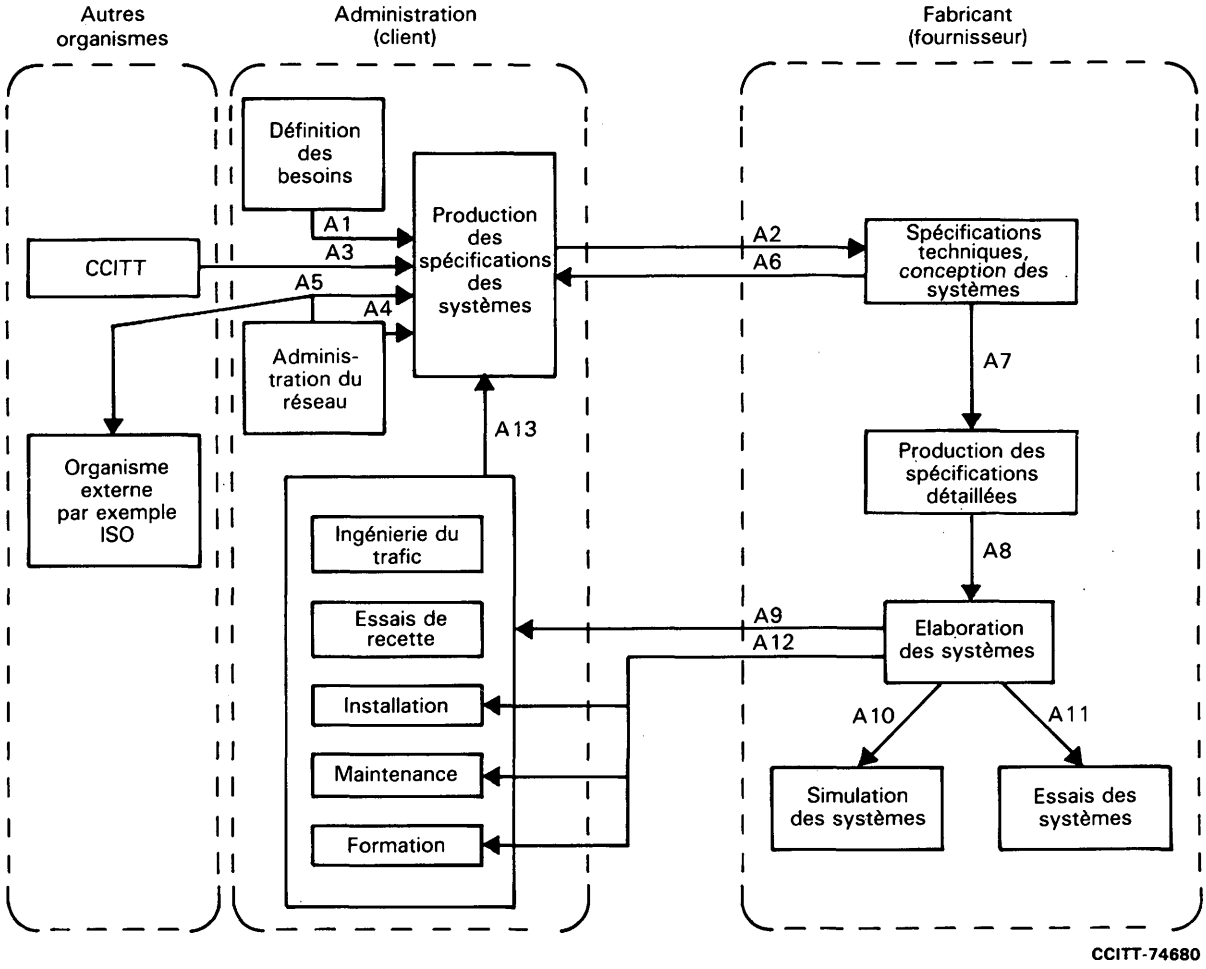
D.4.2 *Structuration des systèmes en LDS*

D.4.2.1 *Considérations générales*

Ce paragraphe traite de certaines constructions en LDS qui autorisent diverses structures de systèmes; ainsi, à partir d'un aperçu général de la structure d'un système, on peut en fournir une description de plus en plus détaillée en examinant la structure interne de chacune de ses parties. La Recommandation Z.101 décrit la structure minimale d'un système en LDS; un système se compose d'un ensemble de blocs connectés au moyen de canaux – ces blocs contiennent des processus (voir la figure D-2).

Il n'est pas nécessaire d'appliquer les concepts décrits dans la Recommandation Z.102 aux systèmes qui n'exigent pas de subdivisions plus détaillées. La Recommandation Z.102 donne les concepts d'une subdivision à plusieurs degrés de détail.

Une observation globale de la structure interne de toutes les parties d'un système révèle ses diverses structures d'une façon plus ou moins détaillée. On dit que la structure du système est représentée à plusieurs niveaux de détail.



A1 Spécification d'une possibilité ou d'une caractéristique, indépendamment de la réalisation et du réseau
A2 Spécification indépendante de la réalisation et dépendante du réseau, comprenant une description de l'environnement du système
A3 Recommandations et directives du CCITT
A4 Contributions à la spécification du système, indiquant les besoins du réseau en matière de gestion et exploitation
A5 Autres Recommandations pertinentes
A6 Description d'une proposition de réalisation
A7 Spécification du projet
A8 Spécification détaillée de la conception
A9 Description complète du système
A10 Documentation relative à la description du système et de son environnement en vue de la simulation du système
A11 Documentation relative à la description du système et de son environnement en vue de l'essai du système
A12 Manuels d'installation et d'exploitation
A13 Contributions à la spécification du système émanant de groupes fonctionnels spécialisés à l'intérieur de l'Administration

Remarque 1 — L'itération est possible à tous les niveaux.
Remarque 2 — Dans certains cas, la documentation LDS indiquée ici comme étant interne à une seule organisation, par exemple A1, A7, A8, pourrait être fournie à une autre organisation.

FIGURE D-1

Scénario général pour l'utilisation du LDS

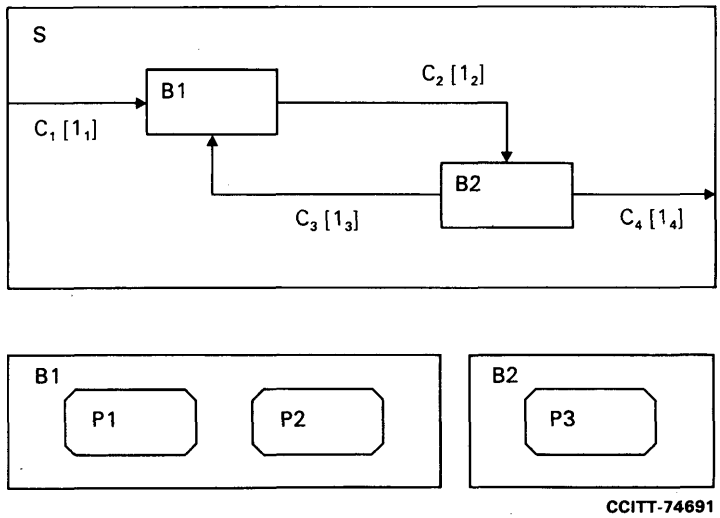


FIGURE D-2

Exemple de structure minimale d'un système

On notera que la structure interne d'une partie d'un système renseigne plus précisément sur la structure, mais pas nécessairement sur le comportement du système. Il est théoriquement possible de distinguer l'aspect d'une représentation plus détaillée du comportement (par exemple, le traitement d'un nouveau signal) de l'aspect d'une structure plus détaillée (couvrant, par exemple, à la fois la structure des parties et celle du comportement du système), mais dans la pratique ces deux aspects sont généralement confondus; ainsi, des détails supplémentaires sur la structure du système en éclairent également le comportement. Le présent paragraphe, pour des raisons de clarté, ne traite que de l'aspect structuration.

D.4.2.2 *Critères de subdivision*

On nomme subdivision la technique qui consiste à fragmenter une représentation d'un système d'un niveau de détail élevé en éléments d'un maniement facile. Le processus de subdivision ajoute une structure à un système.

Parmi les critères qui entraînent la subdivision de la représentation d'un système, on peut citer les suivants:

a) définir des blocs ou des processus qui, par leurs dimensions, facilitent la compréhension du système;

b) établir une certaine correspondance avec les divisions effectives du logiciel et/ou du matériel;

c) utiliser les subdivisions fonctionnelles naturelles;

d) réduire au minimum l'interaction entre les blocs;

e) réutiliser des représentations préexistantes (par exemple un sytème de signalisation;

f) mettre en correspondance une description avec une spécification donnée.

Les critères qui seront effectivement adoptés varient selon qu'il s'agit d'une spécification ou d'une description et ils peuvent dépendre du degré de précision requis.

Chaque bloc peut faire l'objet d'une subdivision supplémentaire selon les mêmes critères ou selon des critères différents.

Etant donné que la relation entre les niveaux dépend des critères de subdivision choisis, il importe de spécifier clairement ces critères afin de faciliter la compréhension de la représentation.

L'usager décide des critères de subdivision mais cetaines restrictions s'appliquent afin de garantir une représentation correcte en LDS. Les chapitres suivants traitent de ces restrictions.

D.4.2.3 *Subdivision des blocs*

On peut subdiviser un bloc en un ensemble de blocs et de canaux pratiquement de la même façon qu'on subdivise un système en blocs et en canaux (voir la figure D-3).

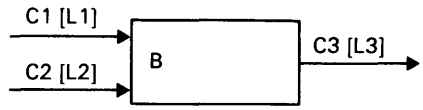
Certaines règles importantes doivent être observées dans le processus de subdivision:

1) Les listes de signaux des sous-canaux (par exemple, C1.1 et C1.2, figure D-3) reliés à un canal d'entrée (par exemple, C1) ne doivent pas comporter de nouveaux signaux; ces listes de signaux doivent inclure tous les signaux du canal initial. Ainsi, dans l'exemple de la figure D-3, L1.1 et L1.2 contiennent tous les signaux de L1. En outre, aucun des signaux de L1.1 ne peut apparaître en L1.2.

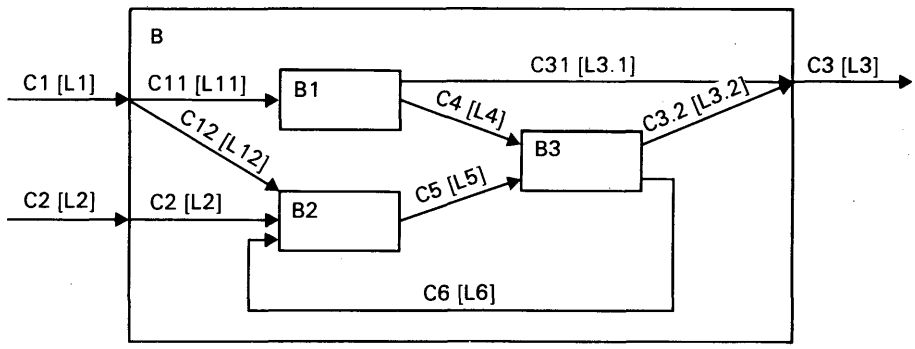
2) Les listes de signaux des sous-canaux (par exemple, C3.1 et C3.2) reliés à une voie de sortie (par exemple, C3) ne peuvent comporter de nouveaux noms de signaux; ces listes de signaux doivent inclure tous les noms de signaux du canal initial. Ainsi, L3.1 et L3.2 contiennent tous les noms de signaux de L3. Contrairement aux sous-canaux connectés à un canal d'entrée, L3.1 et L3.2 peuvent contenir les mêmes noms de signaux.

3) Si les listes de signaux des nouveaux canaux (par exemple, C4, C5 et C6) comportent des noms de signaux qui ne figuraient pas dans le bloc initial, ces nouveaux signaux doivent être définis dans la définition de la partie interne du bloc initial (B).

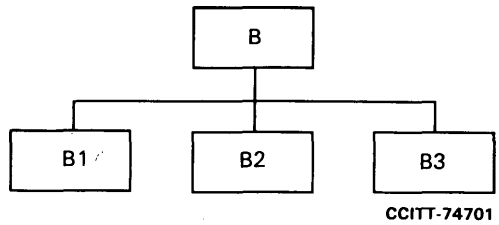
4) Il existe trois options au cas où le bloc initial renferme des processus. Premièrement, chaque processus peut être assigné directement à l'un des nouveaux sous-blocs. La deuxième option prévoit la destruction des processus initiaux et leur remplacement dans les sous-blocs par de nouveaux processus. La troisième option consiste à subdiviser les processus initiaux au moyen des règles qui régissent la subdivision des processus (voir le § D.4.2.4).



a) Diagramme initial d'interaction de blocs



b) Diagramme subdivisé d'interaction de blocs



c) Diagramme d'arborescence de blocs

FIGURE D-3

Subdivision des blocs

5) Les définitions de données contenues dans le bloc «ascendant» sont mises à la disposition des sous-blocs; ainsi chacun d'entre eux peut utiliser un type de données défini dans le bloc «ascendant» sans avoir à le redéfinir.

6) Si un type défini dans le bloc «ascendant» est redéfini dans un sous-bloc, la nouvelle définition s'applique au sous-bloc qui l'a élaborée, tandis que l'ancienne définition s'applique toujours aux autres sous-blocs. Il n'est pas recommandé d'effectuer une nouvelle définition dans le seul but de caractériser un sous-bloc: en effet, cette nouvelle définition peut échapper à un lecteur, qui supposera que l'ancienne est toujours valable. Dans certains cas, et notamment si on a besoin de plus amples détails sur le comportement, une nouvelle définition doit être élaborée. Il convient alors de signaler cette nouvelle définition par des annotations pertinentes.

D.4.2.4 *Subdivision des processus*

Les processus permettent de représenter (en partie) le comportement fonctionnel d'un bloc.

Après examen, on peut estimer nécessaire de subdiviser ce comportement en sous-comportements. Ceci revient au même que de parler de la subdivision d'un processus en sous-processus.

Les sous-processus ainsi obtenus devraient représenter (tous ensemble) le même comportement que celui du processus «ascendant». Chaque signal reçu par le processus «ascendant» doit être reçu par un sous-processus et un seul, et chaque signal produit par le processus «ascendant» doit l'être par un au moins des sous-processus.

La subdivision d'un processus peut résulter de la subdivision du bloc qui contient le processus «ascendant»; elle peut également être sans rapport avec la subdivision de toute autre construction, et donc s'opérer individuellement.

D.4.2.4.1 La subdivision d'un processus peut être la conséquence de la subdivision d'un bloc. Il convient tout d'abord de souligner que la subdivision d'un bloc n'implique pas nécessairement la subdivision du(des) processus qui représente(nt) son comportement. Il peut se produire qu'un processus représente le comportement (ou une partie du comportement) de l'un des sous-blocs. Dans ce cas, il n'est pas nécessaire de subdiviser le processus (voir la figure D-4).

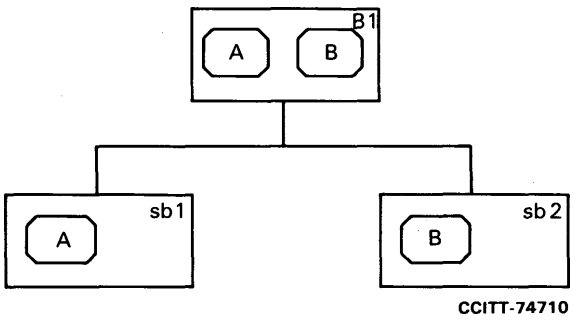


FIGURE D-4

Subdivision d'un bloc sans subdivision de ses processus

Lorsque le comportement que représente le processus correspond à celui de deux sous-blocs ou plus, il faut subdiviser le processus en autant de processus que le demande la règle selon laquelle chaque processus ne peut représenter le comportement que d'un(d'une partie d'un) seul bloc.

Si le bloc est subdivisé en «n» sous-blocs, chacun des processus associés au bloc en question peut être subdivisé en «m» sous-processus, «m» étant tout nombre supérieur à zéro autre que «n». En outre, on notera que chacun des processus associés au bloc est subdivisé indépendamment des autres; l'un d'entre eux peut être subdivisé en deux sous-processus, un autre en quatre sous-processus et un troisième en un seul sous-processus.

Si un processus est subdivisé en un seul sous-processus, ce dernier, du point de vue de la structure, est égal au processus «ascendant».

D.4.2.4.2 La subdivision d'un processus peut se faire indépendamment des blocs auxquels il est associé. Dans ce cas, les sous-processus qu'engendre la subdivision du processus «ascendant» remplacent ce dernier dans le bloc auquel il était associé.

D.4.2.4.3 On peut subdiviser les processus parallèlement au bloc afin de représenter le comportement des divers sous-blocs. La subdivision d'un processus indépendamment du bloc peut être motivée par le souci de simplifier la représentation en isolant des aspects particuliers du comportement.

Il est à noter que si l'on considère ces aspects comme des sous-comportements (tels que la validation de la fréquence ou la reconnaissance des chiffres), il serait préférable de les représenter au moyen de procédures et de macros.

Au contraire, les sous-processus sont plus appropriés pour représenter les comportements isolés lorsque ceux-ci sont des comportements «principaux» et ont tous la même «dignité» (importance). Dans le premier cas, les sous-comportements peuvent traiter les mêmes signaux d'entrée du processus principal, tandis que dans le second cas chaque sous-processus a un ensemble de signaux d'entrée indépendant des ensembles de signaux d'entrée des autres sous-processus.

Pour ce dernier cas, on peut citer à titre d'exemple la subdivision du «processus de traitement d'appel» en «sous-processus de traitement d'abonné», «sous-processus d'acheminement» et «sous-processus de traitement de jonction» (voir la figure D-5).

Dans la subdivision d'un processus, il convient d'observer quelques règles importantes:

1) La subdivision d'un processus entraîne une subdivision de l'ensemble des signaux entrants, de telle sorte qu'un sous-processus donné reçoit un sous-ensemble donné de signaux, un autre sous-processus reçoit un autre sous-ensemble et ainsi de suite. Les sous-ensembles sont disjoints et contiennent collectivement l'ensemble des signaux reçus par le processus «ascendant». Ils peuvent également contenir d'autres signaux engendrés par la subdivision; ces derniers servent au transfert d'information entre les sous-processus.
2) Les sous-processus peuvent utiliser les définitions de données des processus «ascendants» sans avoir à les redéfinir. Si un type de données défini dans le processus «ascendant» est redéfini dans un sous-processus, la nouvelle définition ne s'applique qu'au sous-processus qui a produit la nouvelle définition, et l'ancienne définition continue de s'appliquer aux autres sous-processus.
3) Toutes les variables doivent être redéfinies dans les sous-processus. Les sous-processus ont accès aux procédures définies dans le processus «ascendant» ou dans le bloc qui lui est associé.
4) Il est possible d'avoir un nombre variable d'instances de processus pour chaque définition de processus. Lorsqu'un processus est subdivisé en sous-processus, une demande «créer» adressée au processus entraîne la création d'une instance pour chaque sous-processus énuméré dans la définition de la sous-structure du processus. Chacun de ces sous-processus est un processus et possède toutes les caractéristiques de tout processus LDS.

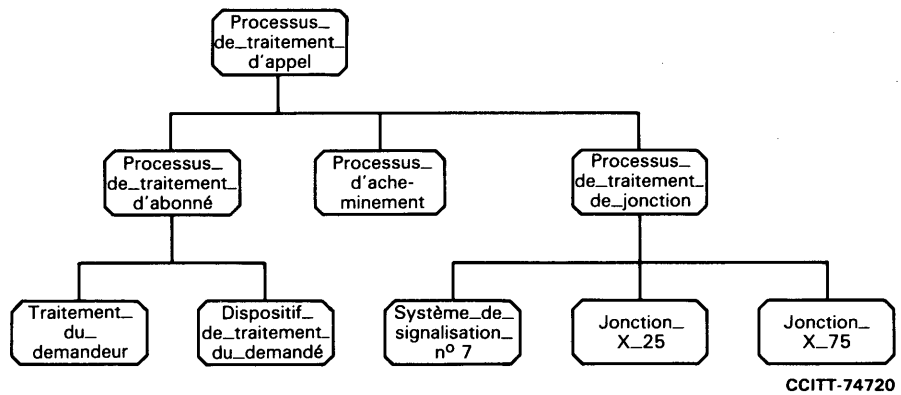


FIGURE D-5

Exemple de subdivision de processus

D.4.2.5 *Subdivision des canaux*

Un canal peut être subdivisé indépendamment des blocs qu'il relie. Cela permet la représentation du comportement du canal en question lorsqu'il achemine des signaux. Il est parfois nécessaire de représenter le comportement du canal afin d'obtenir une représentation exacte du parcours d'un signal. Pour ce faire, on examine le canal en tant qu'élément indépendant dont l'environnement se compose des deux blocs qu'il relie comme l'illustre la figure D-6.

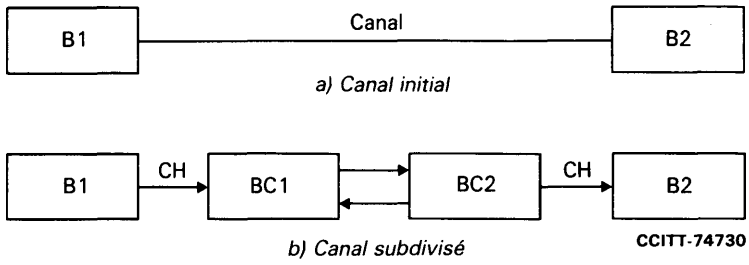


FIGURE D-6

Exemple de subdivision d'un canal

Cette représentation d'un canal en révèle la structure (la structure qu'il forme avec les blocs) ainsi que l'interconnexion de chaque bloc avec les autres blocs et les processus qui représentent le comportement des blocs.

Dans la subdivision de canal, les canaux entrants et sortants ne se partagent pas en de multiples sous-canaux mais doivent être reliés à un seul bloc. Le canal subdivisé a les mêmes caractéristiques qu'un bloc subdivisé, à cette restriction près. Il est possible de subdiviser encore les blocs et les canaux du diagramme d'interaction des voies.

D.4.2.6 *Influence réciproque de la subdivision sur les blocs, les processus et les canaux*

Nous avons jusqu'ici examiné la subdivision des blocs, processus et canaux pris indépendamment les uns par rapport aux autres. Ces activités sont en fait étroitement liées. Ainsi que nous l'avons vu, la subdivision d'un bloc entraîne généralement une subdivision des canaux qui réalisent l'interconnexion et des processus qui lui sont associés.

A l'inverse, la subdivision d'un processus met en évidence des comportements particuliers généralement traités par des parties d'un système (chacune d'entre elles doit pouvoir être mise en concordance selon le concept du bloc).

Comme l'explique le § D.4.2.2, les causes de la subdivision de la représentation d'un système sont variables, mais elles entraînent la subdivision de toutes les entités qui représentent le système.

Les paragraphes suivants traitent ce sujet d'une façon plus détaillée. La subdivision des blocs, des processus et des canaux doit être parfaitement claire, pour ce qui touche à la complexité, la gestion ou la reproduction de la structure effective du système.

Les Recommandations relatives au LDS contiennent des constructions, des règles et des notations qui couvrent la subdivision de ces entités, mais les usagers sont confrontés au fait que les autres entités LDS sont impliquées dans la subdivision et dans la question qui en découle, de l'établissement de rapports mutuels entre les différentes entités.

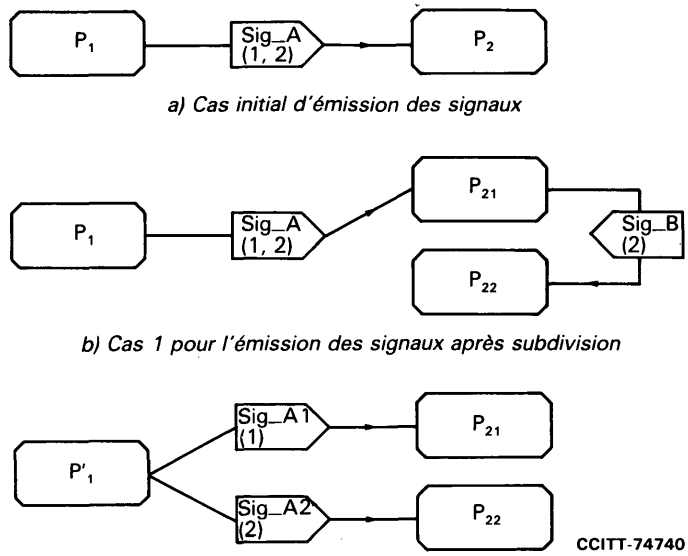
D.4.2.6.1 *Signaux*

Un signal achemine des informations d'un processus à l'autre (un ou plusieurs processus LDS peuvent déterminer le comportement de l'environnement). Ces données peuvent être très complexes; elles sont acheminées à un certain moment de la vie du système, c'est-à-dire lorsque le processus émetteur interprète une transition donnée, puis lorsque le processus récepteur atteint un certain état.

La subdivision d'un processus en sous-processus peut exiger la répartition en plusieurs sous-processus des données «compactées» dans un signal.

Il existe deux possibilités. Dans le premier cas, le signal n'est pas affecté; un sous-processus le reçoit et produit les signaux nécessaires à l'acheminement de la partie des informations dont les autres sous-processus peuvent avoir besoin. La deuxième possibilité affecte le signal qui est alors subdivisé en un certain nombre de signaux, à raison d'un signal pour chaque sous-processus qui demande des données.

La figure D-7 représente ces deux mécanismes.



a) Cas initial d'émission des signaux

b) Cas 1 pour l'émission des signaux après subdivision

c) Cas 2 pour l'émission des signaux après subdivision

FIGURE D-7

Mécanisme d'émission des signaux pendant la subdivision des processus

Les deux mécanismes décrits paraissent équivalents et le deuxième semblerait pouvoir être toujours mis en concordance avec le premier; il existe cependant une différence: en effet, le premier mécanisme opère un transfert séquentiel d'informations et le deuxième mécanisme opère un transfert potentiellement parallèle d'informations. De même, dans le second cas, le processus d'émission n'est pas identique au processus du cas initial: ces deux processus émettent des signaux différents.

Indépendamment de ceci, l'on a obligatoirement recours au deuxième mécanisme dans un cas précis. Il s'agit de l'emploi, à un niveau élevé de description, d'un signal puissant concentrant plusieurs types d'informations.

La situation suivante peut se présenter lors de la subdivision de la représentation du système (qui permet d'obtenir un plus grand nombre de détails concernant sa structure): les données concentrées dans le signal ne peuvent plus l'être du fait qu'elles ne sont pas disponibles au même endroit ou/et au même moment. Ceci peut résulter de la subdivision lorsque les données se trouvent dans différents blocs (signaux de provenances diverses), ou lorsqu'elles ne sont pas disponibles en même temps. Citons, pour illustrer ce dernier cas, le signal «choix du demandé» au niveau élevé; au niveau inférieur, ce signal se transforme en une suite de signaux «chiffre». Une méthode de compactage qui accumule tous les «chiffres» puis envoie le «choix du demandé» est concevable, mais peut être une représentation faussée, voire impossible à réaliser si l'on souhaite montrer qu'une jonction de sorties peut produire une libération en retour après chaque chiffre.

Le signal «établissement de la ligne achevé» peut servir d'exemple: ce signal, à un niveau inférieur, se subdivise en une série de signaux produits par les différents sous-blocs, qui introduisent chacun une partie de l'ensemble du montage.

Ainsi que les exemples l'illustrent, il existe plusieurs cas dans lesquels un signal doit être subdivisé à la suite de la subdivision du système.

Les Recommandations relatives au LDS ne préconisent pas de notation particulière, bien que certaines aient été proposées et examinées par le groupe chargé d'établir le LDS.

Il est conseillé, dans les représentations LDS à plusieurs niveaux, d'annoter les signaux qui résultent de la subdivision d'un signal à un niveau supérieur. Ceci accroît considérablement la lisibilité de la représentation qui relie les différents niveaux entre eux.

L'annotation doit indiquer le signal ascendant d'un signal donné lors de la définition de ce dernier, de même qu'une référence vers l'avant pour indiquer les signaux produits par la subdivision de chaque signal.

L'on notera que la subdivision d'un signal affecte l'émetteur et le récepteur de même que le canal qui contient maintenant de nouveaux noms de signaux. En cas de subdivision de plusieurs signaux, les avantages qu'engendre la conservation de l'ancien canal associé à ses sous-canaux aux extrémités disparaissent; la représentation perd alors en quelque sorte son intérêt, vu qu'il faut redéfinir une partie de l'ancien canal. Cette redéfinition est d'une ampleur variable en fonction du nombre de signaux subdivisés.

La deuxième méthode que mentionne la figure D-7, et qui consiste à remplacer l'ancien canal par de nouveaux, présente plus d'attrait dans ce cas à cause de sa simplicité compte tenu du fait que l'émetteur et le récepteur sont de toute façon affectés.

D.4.2.6.2 *Etats*

La subdivision d'un processus en sous-processus peut également s'appliquer à ses composantes, à savoir les états, les décisions, les tâches et les données (les entrées comme les sorties sont affectées par la subdivision du signal).

Un état doit être la représentation d'une situation logique d'un processus, par exemple, la phase de conversation ou la phase d'essai. La subdivision du processus en sous-processus entraîne la subdivision de l'état logique en un ensemble d'états qui se répartissent entre plusieurs sous-processus (sans toutefois couvrir nécessairement la totalité des sous-processus).

Ceci influe sur l'interprétation par l'homme et par la machine des corrélations qui existent entre les différents niveaux.

Devant la multiplication des détails, l'être humain tend à perdre de vue la situation dans son ensemble, ce qui peut engendrer des effets inopportuns (cf. le nombre de fois qu'une remise à jour d'un programme supposée parfaite produit des effets indésirables dans l'ensemble du système!).

La dispersion de détails qui sont groupés d'une façon logique, exerce un effet encore plus nuisible sur la capacité de compréhension d'une situation dans son ensemble. L'organisation LDS d'une représentation sur plusieurs niveaux cherche à éviter ce problème, mais il est essentiel de mettre les divers niveaux en corrélation les uns avec les autres.

Il est possible d'établir cette corrélation à l'aide de notations appropriées renvoyant à une représentation plus abstraite qui fournit un aperçu d'ensemble (le niveau supérieur), et par l'établissement de références aux autres parties qui doivent être examinées ensemble.

L'aspect de l'interprétation qu'effectue la machine diffère totalement de celui de l'interprétation humaine. La machine n'a aucune vue d'ensemble, et tous les éléments sont pris en compte et considérés avec la même attention; ils sont pris en compte un par un, une relation à la fois. C'est pourquoi la dispersion de données toutes reliées les unes aux autres n'influe pas sur l'interprétation réalisée par la machine (nous ne tiendrons pas compte de la durée de l'interprétation). Le problème, avec cette machine, consiste à formaliser la subdivision du processus de sorte que la machine puisse comprendre que ce qui était précédemment désigné comme «au repos» est devenu une série d'états «au repos» répartis entre plusieurs sous-processus et ainsi de suite.

Les seules possibilités de rendre une machine consciente de cette correspondance consistent à déclarer explicitement cette dernière (ce qui est très inefficace et rarement sûr du fait du risque élevé d'oubli de certaines déclarations), ou à formaliser les règles de subdivision d'un processus en sous-processus.

Des méthodes permettent de traiter (et de résoudre) cette question; c'est pourquoi elle n'est pas étudiée dans les Recommandations relatives au LDS.

D.4.2.6.3 *Décisions*

La subdivision d'un processus peut affecter les décisions en cas de subdivision des données sur lesquelles ladite décision repose. Si une donnée n'est plus disponible pour un sous-processus qui doit prendre une décision à partir de ladite donnée, il convient d'émettre un signal demandant la valeur de la donnée en question et le sous-processus doit attendre la réponse. Le mécanisme d'émission (demande) et de réception (réponse) des signaux peut être secret si les données sont déclarées comme étant importées et si la décision est prise à partir des données importées.

L'on notera que le mécanisme de visibilité des données partagées n'est pas utilisable du fait que les sous-processus sont généralement attribués à des blocs différents.

D.4.2.6.4 *Tâches*

La tâche est la représentation d'un ensemble d'actions qui n'exercent pas d'effet direct à l'extérieur du processus. Lorsque le processus est subdivisé, l'ensemble d'actions exécutées par une tâche peut également être subdivisé de telle sorte que certains parties sont exécutées dans un sous-processus, d'autres dans d'autres sous-processus et ainsi de suite.

L'on notera que le partage ne s'opère que dans les sous-processus du processus en question. Si un ensemble d'actions donné est exécuté dans un processus donné, le même ensemble sera exécuté par la totalité des sous-processus de ce même processus. Dans ce cas, il existe une légère différence avec l'état dont la subdivision peut être liée à la subdivision d'autres processus; en effet, l'état d'un processus fait partie d'un état logique qui peut comporter d'autres processus.

Si nous revenons à la tâche, la subdivision des actions réparties entre plusieurs sous-processus exige généralement l'introduction de nouveaux signaux pour déclencher l'exécution d'un sous-ensemble d'actions dans un sous-processus lorsque les actions exécutées par un autre sous-processus s'achèvent.

La subdivision d'une tâche est généralement un effet secondaire de la subdivision des données associées à un processus. Une méthode fondée sur la conception et la représentation peut donner des notations formelles qui permettent de représenter la subdivision d'un ensemble d'actions entre plusieurs sous-processus.

Les conseils donnés au lecteur concernant l'annotation des signaux s'appliquent à la subdivision des tâches.

D.4.2.6.5 *Données*

A un processus sont associées des données. Lorsque nous subdivisons un processus, nous subdivisons également ses données. Il n'existe pas de façon formelle en LDS pour exprimer cette subdivision. Chaque sous-processus doit définir ses propres variables et éventuellement les nouveaux synonymes et les nouveaux types.

On notera que la subdivision de données n'affecte pas seulement les actions du processus (une certaine donnée porteuse d'une information ne peut être émise si l'information en question n'est plus disponible pour le sous-processus concerné, une tâche travaillant sur cette information ne peut être exécutée, etc.), mais encore que la subdivision des données peut affecter d'autres processus si des données exportées ou révélées sont présentes.

Il convient d'informer les processus de la subdivision du dispositif d'exportation, afin qu'ils puissent accéder au nouveau sous-processus pour obtenir l'information. Une donnée déjà connue ne peut plus l'être si le sous-processus auquel elle appartient est alloué à un autre sous-bloc du dispositif de visibilité; on doit donc importer ou exporter sa valeur, ou bien on peut procéder à un échange de signaux explicites.

D.4.2.7 *Représentation du système en cas de subdivision*

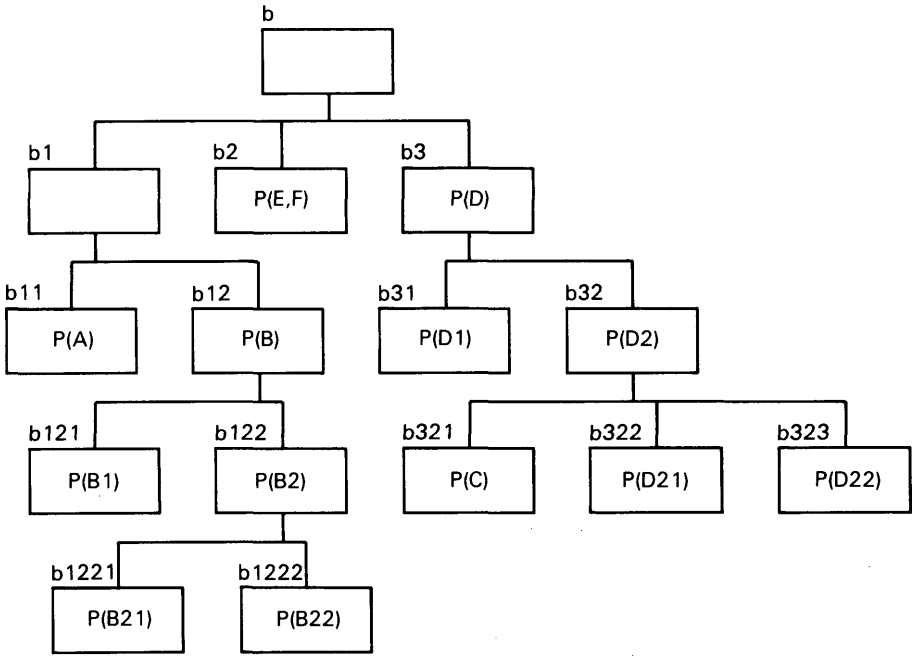
Lorsqu'un système est représenté sous la forme d'un ensemble de blocs interconnectés par des canaux et que l'on exprime le comportement de chaque bloc au moyen d'un ou de plusieurs processus, nous nous trouvons en présence d'une représentation sur un seul niveau. Ceci revient à dire que nous pouvons voir tous les éléments de la représentation au même niveau. Nous introduisons, avec la subdivision une relation hiérarchique entre les différents documents. Le document qui en résulte représente la structure du système lorsque le système se compose de n blocs.

Un autre document peut présenter le système composé d'un ensemble différent de blocs, dont certains proviennent des blocs du document précédent (certains blocs du document précédent ont été remplacés par les sous-blocs résultant de la subdivision des blocs). Ce nouveau document doit être rapporté au précédent document.

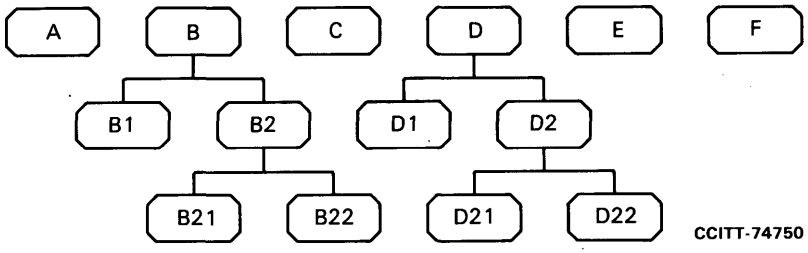
Mais pour obtenir une représentation complète du système, il ne suffit pas d'établir des rapports entre les documents, il faut également les organiser de telle sorte qu'il soit possible d'accéder à la représentation du système par niveaux, en commençant par une vue d'ensemble générale pour passer ensuite à des représentations de plus en plus détaillées. Ceci exige le regroupement des différents documents afin de représenter le système à différents niveaux.

L'on ne devrait pas trouver les mêmes éléments à tous les niveaux. La représentation du système au premier niveau peut se composer de représentations des blocs et des canaux, à l'exclusion des processus qui décrivent le comportement de chaque bloc. A un niveau inférieur, l'on peut souhaiter inclure la représentation du comportement de certains blocs, mais pas de celui d'autres blocs, ou encore l'on peut désirer représenter une partie du comportement d'un bloc, et non pas l'ensemble de son comportement, que l'on ne trouve qu'à un niveau inférieur. Le niveau de représentation le plus bas (c'est-à-dire la représentation la plus détaillée) devrait représenter intégralement les comportements de tous les blocs, c'est-à-dire l'ensemble complet des processus qui expriment ce comportement.

Nous devons, et c'est le premier effet de bord de ce type de représentation par niveaux, indiquer l'association entre un processus et un bloc. La subdivision du système (considéré comme un seul bloc) donne une structure arborescente; la subdivision des «n» processus qui représentent le comportement du système donne «n» arborescences et chacun des processus de ces arborescences devrait être associé à l'un des blocs de l'arborescence. En outre, de nouveaux processus peuvent être introduits à un certain niveau et de nouveau subdivisés à partir de ce niveau. En cas de subdivision, chacun d'entre eux donne une structure arborescente et chaque processus de cette structure doit de nouveau être associé à l'un des blocs de l'arborescence représentée à la figure D-8.



a) Arborescence de blocs



b) Arborescences de processus

Remarque — Les blocs b et b1 n'ont pas de processus associés.
Le bloc b2 contient deux processus (E,F).
Le bloc b321 contient le processus C qui n'apparaît qu'à ce niveau.

FIGURE D-8

Attribution de processus aux blocs

L'observation de l'arborescence de blocs soulève plusieurs réflexions.

En premier lieu, l'arborescence a toujours une et seulement une racine, qui est le bloc du système. Il en est ainsi de même lorsque le système représenté se compose de plusieurs blocs depuis le début. Dans l'arborescence, ces blocs sont représentés au niveau 1. Le nom du système peut suffire pour constituer le bloc qui sert de racine. On peut appliquer les définitions des canaux aux blocs bien que cela ne soit pas exigé, sauf si les blocs ont des processus associés.

L'observation des feuilles de cette arborescence inspire une seconde remarque: elles ne sont pas toutes au même niveau. Ceci est dû à un nombre inégal de subdivisions sur les blocs de l'arborescence. Le nombre de subdivisions dépend de plusieurs facteurs, dont la plupart relèvent de l'appréciation subjective de la personne chargée d'élaborer les spécifications, et du concepteur.

Pour que des blocs feuille ne soient pas subdivisés de nouveau, le LDS n'exige qu'une chose, à savoir que leur comportement soit entièrement spécifié (c'est-à-dire que son attitude puisse être représentée par les définitions des processus). Par conséquent, un bloc feuille doit contenir au moins un processus associé.

Lorsqu'un système est représenté à plusieurs niveaux d'abstraction, nous pouvons représenter le système au niveau de notre choix. Le choix d'un niveau donné exige que les blocs soient examinés à ce même niveau, que leurs processus associés et tous les blocs feuille soient examinés à des niveaux supérieurs avec leurs processus associés (voir la figure D-9).

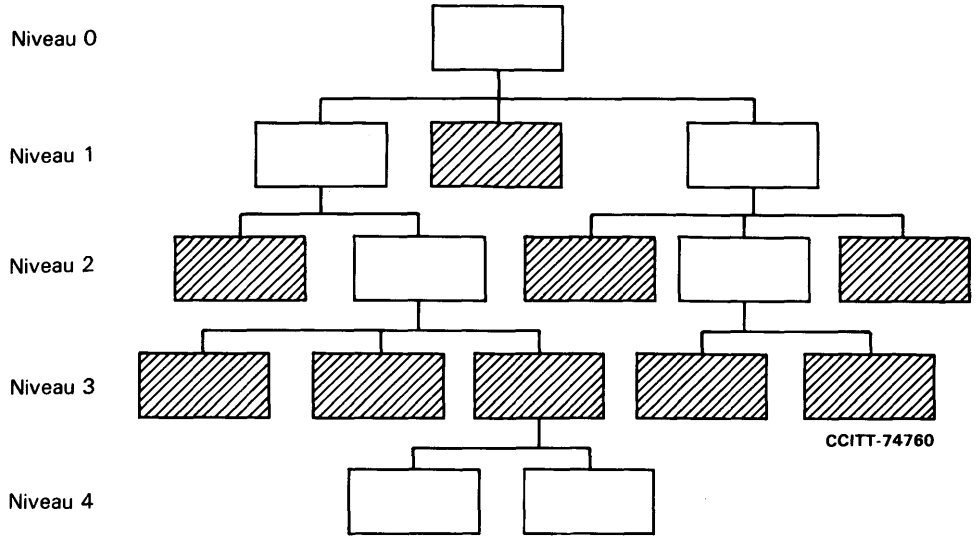


FIGURE D-9

Représentation niveau 3

La représentation d'un système à un certain niveau peut être incomplète dans la mesure où certains des blocs de ce niveau n'ont pas de processus associés, ou encore que ces processus associés ne représentent pas complètement leur comportement.

L'on parle de niveaux de représentation et de sélection d'un certain niveau de représentation pour les raisons suivantes:

- Il se peut que notre représentation ait atteint un certain «niveau» de précision, représenté par ce niveau de représentation (dans ce cas, les blocs de ce niveau sont des blocs feuille; ils ne sont pas complets car ils demandent un supplément de travail!).
- Nous voulons considérer la représentation du système à un certain niveau de précision; nous choisissons donc le niveau de représentation qui correspond au mieux aux abstractions que nous cherchons. On notera que dans certains cas un niveau donné de représentation peut être constitué de documents à différents niveaux d'abstraction. La représentation d'une partie du système au niveau 2 peut être très détaillée, tandis que celle d'une autre partie au niveau 4 peut demeurer abstraite. Cela signifie qu'en choisissant une représentation au niveau 3, l'on peut obtenir une représentation très détaillée de certaines parties et une simple vue d'ensemble d'autres parties.
- La méthodologie de représentation et conception peut permettre à chaque niveau d'avoir une signification précise. Ainsi le niveau 1 correspond à la spécification, le niveau 2 fournit la structure générale du système, le niveau 3 la structure des modules ("racks", fonctions de logiciel), le niveau 4 donne la structure détaillée (planches imprimées, procédures, modules de logiciel). Dans ce cas, le lecteur choisit un niveau donné en fonction de son propre besoin, et l'on notera que la méthode permet d'éviter les différences entre les niveaux de précision des parties qui constituent un niveau donné de représentation.

Outre la représentation globale du système et la représentation par niveaux, le LDS comporte la notion de «représentation cohérente du système» (voir la figure D-10).

Cette notion permet de définir toute représentation du système en tant que représentation à un seul niveau dont tous les blocs peuvent provenir de n'importe quel niveau de la structure du système s'il est établi que:

- L'on peut choisir un bloc pour l'incorporer à la représentation cohérente du système si on peut le considérer comme un bloc feuille (soit il s'agit d'un bloc feuille, soit on lui associe tous les processus requis pour représenter son comportement).
- Si l'on choisit un bloc, il doit contenir tous les blocs engendrés par la subdivision de son bloc ascendant; ils doivent être inclus directement ou par l'inclusion de leurs descendants.
- Tous les documents définissant le signal qui circule sur le canal de connexion d'un bloc sur la représentation doivent être fournis. Il s'ensuit qu'en fonction de la méthode de subdivision retenue, et suite à la prise d'un bloc, les parties qui définissent les données et les signaux de son bloc ascendant doivent également être prises.

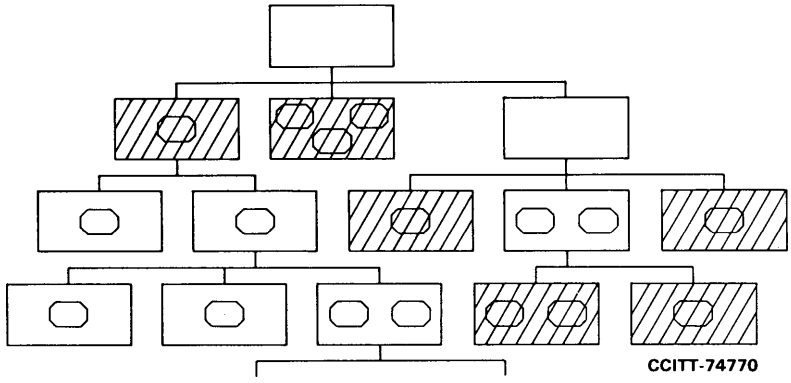


FIGURE D-10

Exemple de représentation cohérente du système

Lorsque certains canaux, considérés comme des systèmes, ont été subdivisés, il convient de fournir la représentation de chacun de ces «sytèmes».

Cette représentation a le même type de documents que la représentation habituelle du système. Il convient d'ajouter la notation permettant de relier ces systèmes au système principal. L'on peut dans une certaine mesure considérer ces systèmes comme des systèmes internes par rapport au «système principal». Chacun d'entre eux peut avoir plusieurs niveaux de représentation et comporter des systèmes internes si l'un quelconque des canaux qu'ils contiennent en vient à être lui-même considéré comme un système (voir la figure D-11).

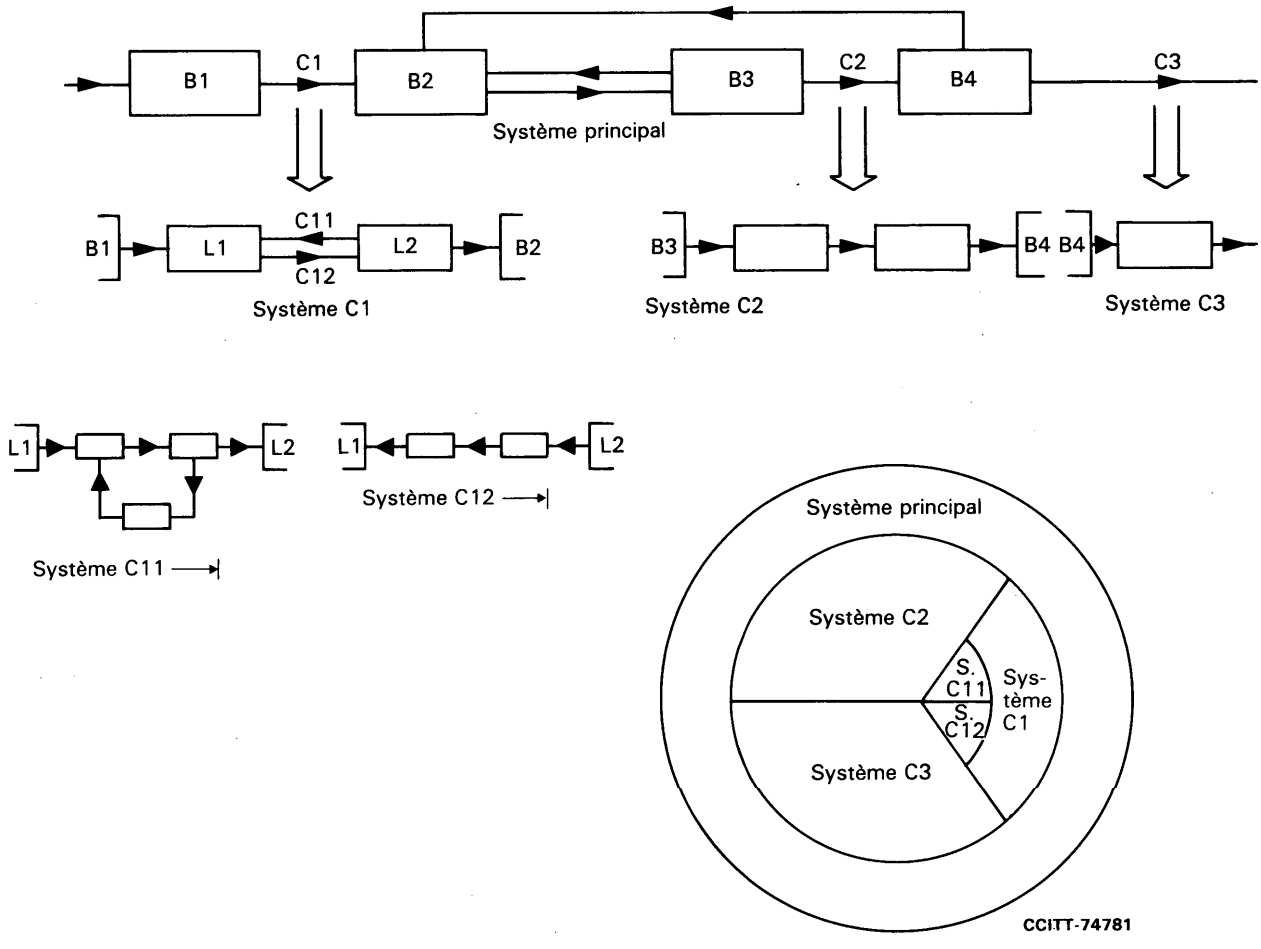


FIGURE D-11

Possibilité de représentation du système par la subdivision des canaux

D.4.3 *Concepts du LDS*

Le § 4.3 présente chacun des concepts du LDS et donne des directives générales sur la façon de les employer. Les exemples feront principalement appel au LDS/GR. Les mêmes directives s'appliquent également au LDS/PR.

D.4.3.1 *Système*

Ainsi qu'il a été vu plus haut, le LDS représente des systèmes à l'aide de modèles. Ainsi ce qui est défini par une spécification ou une description en LDS constitue le système. Un système LDS peut donc représenter à l'aide de modèles une partie d'un système (ou d'un central) téléphonique, ou un réseau complet de systèmes téléphoniques ou des parties d'un grand nombre de centraux téléphoniques (par exemple, les dispositifs de contrôle de circuit aux deux extrémités d'un circuit). Il importe de souligner que le système LDS contient, d'un point de vue LDS, tout ce que la spécification ou la description tente de définir. L'environnement ne relève pas de la spécification et n'est pas défini en LDS.

Des canaux relient le système à l'environnement. Théoriquement il suffit d'un seul canal entrant et d'un seul canal sortant pour connecter le système à l'environnement. Dans la pratique, on définit généralement des canaux pour chaque jonction logique avec l'environnement.

Chaque système se compose d'un certain nombre de blocs reliés entre eux par des canaux. Chaque bloc est indépendant de tous les autres. Seule l'émission de signaux dans les canaux permet la communication entre des processus placés dans deux blocs différents.

D.4.3.2 *Bloc*

Ainsi que nous l'avons vu dans le paragraphe relatif à la structuration en LDS, un bloc peut renfermer une structure de un ou plusieurs processus. Pour que l'on puisse obtenir une définition complète d'un bloc, les blocs placés au bas du diagramme d'arborescence de blocs doivent contenir une ou plusieurs définitions de processus.

Les processus peuvent communiquer entre eux à l'intérieur d'un bloc, au moyen de signaux, de valeurs partagées. Ainsi le bloc ne constitue pas seulement un mécanisme pratique pour le regroupement de processus, mais encore une limite à la visibilité des données. C'est pourquoi il convient, lors de la définition de blocs, de s'assurer du caractère raisonnable et fonctionnel du regroupement de processus au sein d'un bloc. Dans la plupart des cas, il est utile de fractionner dans un premier temps le système (ou le bloc) en unités fonctionnelles, puis de définir les processus à intégrer dans le bloc.

D.4.3.3 *Canal*

Les canaux permettent la communication entre les blocs. Un canal est normalement une entité fonctionnelle pouvant servir à marquer un trajet d'information donné. En fait, il est possible de définir d'une manière formelle le comportement de chaque canal en subdivisant les canaux.

Les définitions des canaux comportent la liste des signaux qui peuvent être acheminés par le canal. Cette liste des signaux doit assurer que chaque signal émis par un processus à une extrémité du canal peut être reçu par le processus du bloc placé à l'autre extrémité du canal. Ainsi la définition du canal s'incorpore à la définition de la jonction de chaque bloc. Dans les projets à grande échelle qui impliquent plusieurs personnes, le risque d'impossibilité de communication entre deux processus peut être réduit si un accord est rapidement réalisé concernant les signaux d'un canal et la définition desdits signaux.

D.4.3.4 *Processus*

Un processus est une machine à états finis étendue qui définit le comportement dynamique d'un système. Les processus sont foncièrement dans un état d'attente des signaux. A la réception d'un signal, le processus répond en accomplissant les actions précises qui correspondent à chaque type de signal pouvant être reçu par le processus. Les processus contiennent de nombreux états qui leur permettent d'accomplir différentes actions en cas de réception d'un signal. Ces états mémorisent les actions précédemment accomplies. Après l'accomplissement de toutes les actions liées à la réception d'un signal donné, un nouvel état est atteint et le processus se met en attente d'un autre signal.

Les processus peuvent soit exister au moment de la création du système, soit être créés suite à une demande de création émise par un autre processus. En outre, les processus peuvent durer indéfiniment ou s'arrêter par le déclenchement d'une action interne d'arrêt. Certaines caractéristiques importantes de la création de processus sont:

1) Après la création du système, les processus ne peuvent être créés que par un autre processus du même bloc. Une possibilité de création d'un processus dans d'autres blocs consiste à avoir un processus spécial dans chaque bloc; ce processus provoque la création d'un processus à la réception d'un signal provenant d'un processus d'un autre bloc. Dans de nombreux cas, ce processus spécial est une sorte de processus «de système en exploitation».

2) Après leur création, les processus ont une durée de vie qui leur est propre. Ils ne meurent que lorsqu'ils accomplissent une action arrêter pendant une transition. L'on peut construire des systèmes qui autorisent les opérations extérieures d'élimination de processus en employant un signal spécial d'élimination. A la réception du signal d'élimination, le processus accomplit une action arrêter.

3) Un processus créé connaît sa propre identité et celle de son ascendant. Il ne peut communiquer avec d'autres processus s'il ne reçoit pas d'autres informations. Ce problème peut se résoudre de deux façons. En premier lieu, un ou plusieurs des paramètres du processus possèdent l'identité des processus avec lesquels le processus en question va communiquer. Deuxièmement, il peut recevoir un signal dont l'un des paramètres effectifs est un identificateur d'instance de processus.

D.4.3.4.1 *Etats*

Un état est une étape du processus pendant laquelle aucune action ne s'accomplit, mais où la file d'entrée contrôle l'arrivée des signaux entrants. Grâce à l'identificateur de signaux que contient le signal d'entrée, l'arrivée du signal fait sortir le processus de l'état et lui fait accomplir une succession donnée d'actions. Un signal qui est arrivé et a entraîné une transition a été «absorbé» et cesse d'exister. Au cours d'une transition un processus ne sait pas explicitement quel signal d'entrée a occasionné la transition. Seul le contexte permet de le déduire (c'est-à-dire que cette transition ne peut se produire qu'en cas de réception d'un signal donné). Dans la figure D-12, la tâche T1 ne s'accomplit qu'en cas de réception de I1. Toutefois, la tâche T2 s'accomplit en cas de réception de I2 ou de I3. S'il importe que T2 sache quel signal d'entrée a été reçu, il est souhaitable de concevoir le processus comme l'illustre la figure D-13.

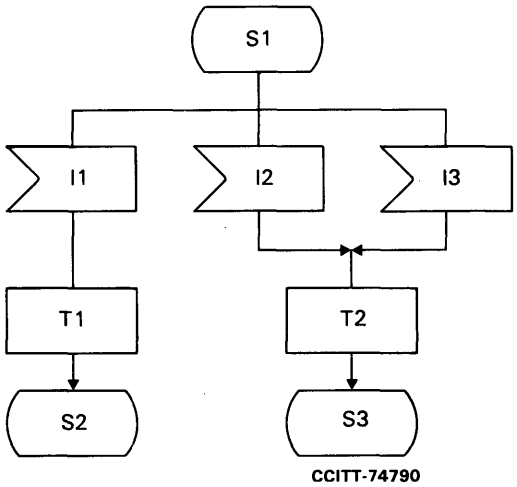


FIGURE D-12

Exécution d'une tâche qui dépend de deux signaux reçus sur trois mais qui est indépendante de chacun

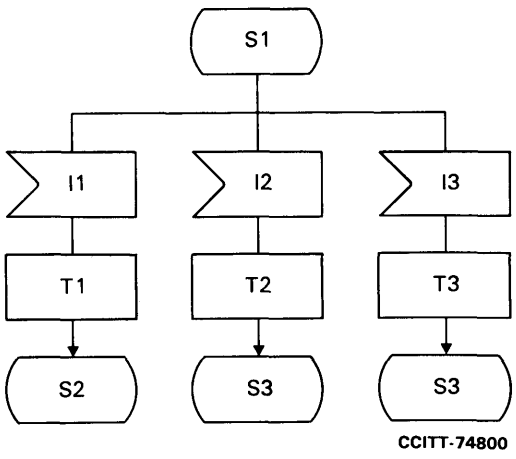


FIGURE D-13

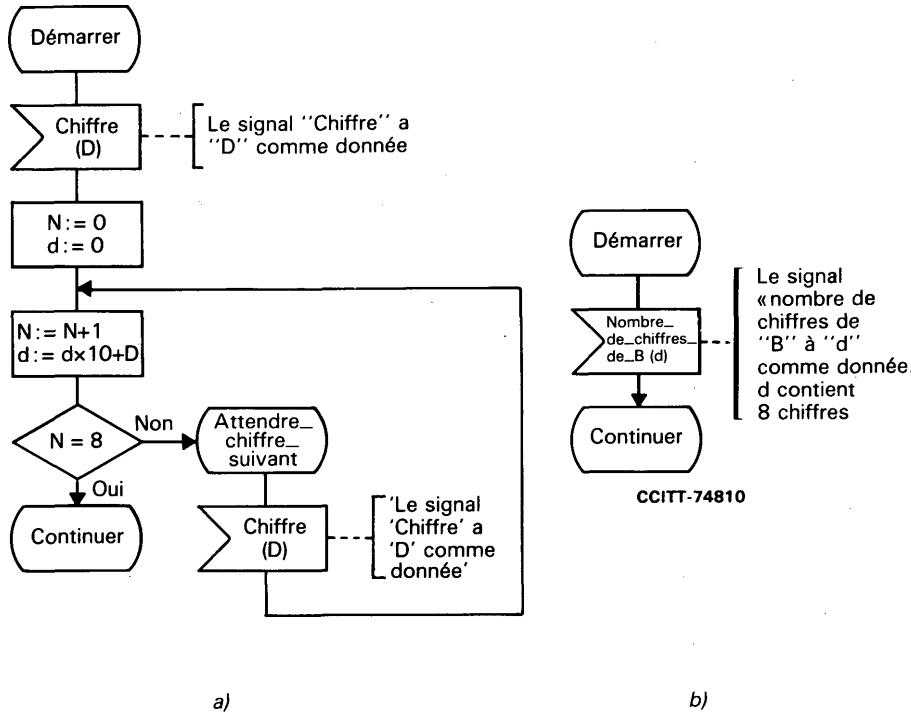
Exécution d'une tâche qui dépend du signal reçu

D.4.3.4.1.1 *Détermination des états requis*

L'auteur d'un diagramme LDS dispose généralement d'une certaine latitude pour définir les états d'un processus. Il peut avoir besoin d'une stratégie qui lui permette d'identifier les états du processus et cette stratégie peut être non formelle ou formelle. Un jugement sûr (que seule une longue expérience permet d'acquérir) est nécessaire si l'on veut établir des diagrammes LDS tels que l'identification d'un trop grand nombre d'états ne les complique pas inutilement ou qu'un nombre artificiellement réduit d'états ne les empêche d'exploiter les avantages qu'offre le LDS. Avant que l'auteur ne commence à établir le diagramme, certaines étapes préliminaires (étudiées au § D.4.2) doivent être achevées, par exemple:

- la structuration du système en blocs fonctionnels;
- la représentation d'un ou plusieurs processus LDS par bloc;
- le choix des signaux d'entrée et de sortie;
- l'utilisation des données dans le processus.

Tous les facteurs ci-dessus exercent un effet important dans la détermination des états de chaque processus. L'effet qu'exerce le choix des signaux d'entrée sur le nombre d'états d'un diagramme LDS est illustré par les deux exemples de la figure D-14.



Remarque — Les exemples *a)* et *b)* représentent la même fonction à différents niveaux de détail. Le nombre des états varie en conséquence.

FIGURE D-14

Réception d'un numéro téléphonique à huit chiffres

D.4.3.4.1.2 *Réduction du nombre des états*

Ayant appliqué une stratégie pour l'identification des états d'un processus, l'auteur d'un diagramme LDS estimera peut-être qu'un trop grand nombre d'états ont été utilisés. Le nombre des états est important car la dimension et la complexité d'un diagramme LDS sont souvent étroitement liées à ce nombre. Il existe certes des moyens qui permettent de réduire le nombre des états mais le fait qu'un diagramme LDS soit complexe n'est pas, en soi, une raison justifiant sa modification; en effet, la complexité du diagramme peut être simplement due à la complexité inhérente au processus défini. Le choix de l'ensemble d'états doit généralement offrir le plus de clarté possible à la séquence d'interactions entre le processus et son environnement. Ce n'est d'ordinaire pas en minimisant le nombre d'états que l'on obtient cette clarté. Le nombre de séquences indépendantes traitées par un processus exerce un effet multiplicateur sur le nombre d'états. Il est donc souhaitable que les séquences indépendantes des processus séparés soient traitées de façon à réduire le nombre d'états et à obtenir une plus grande clarté.

On peut réduire le nombre des états en effectuant la séparation des fonctions communes ou la fusion des états ou encore en recourant au concept de procédure. Certaines structures de données peuvent également conduire à une réduction du nombre des états. Pour d'autres représentations, l'emploi des macros peut être avantageux.

D.4.3.4.1.3 *Séparation des fonctions communes*

Lors de la mise en place d'un diagramme LDS, on peut constater que la définition d'un aspect particulier et répétitif d'un processus nécessite la représentation d'états répétitifs. La figure D-15 représente une partie d'un diagramme LDS définissant un processus de signalisation de ligne et illustrant la condition selon laquelle une tonalité de signalisation de ligne doit être présente pendant une certaine durée avant que l'on considère que le signal de ligne a été détecté.

Pour spécifier cet aspect, il convient d'insérer un état intermédiaire entre les états aucun_signal_de_ligne_n'est_détecté et conversation. Supposons que, dans un diagramme complet, une telle fonction commune doit être répétée à chaque point où le signal est détecté. Une autre solution consiste à définir un processus séparé qui est responsable du contrôle de la tonalité de signalisation de ligne et de la détection des signaux sur la base du temps de reconnaissance spécifié. L'existence de ce nouveau processus permettrait de représenter le diagramme de la figure D-15 de la manière indiquée dans la figure D-16. (Dans un contexte donné, les figures peuvent être rendues équivalentes moyennant l'introduction d'un nouveau signal signal_de_ligne_valable).

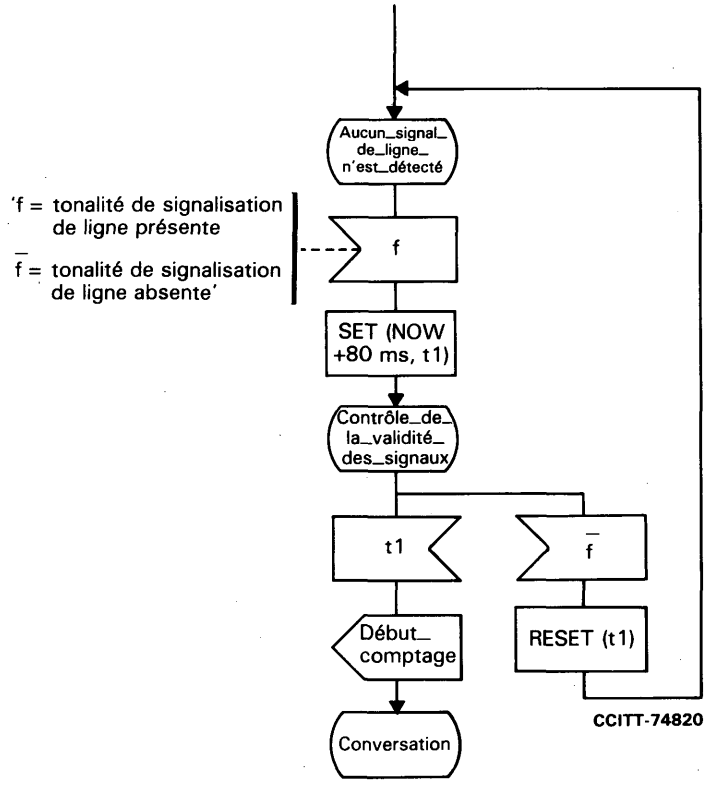
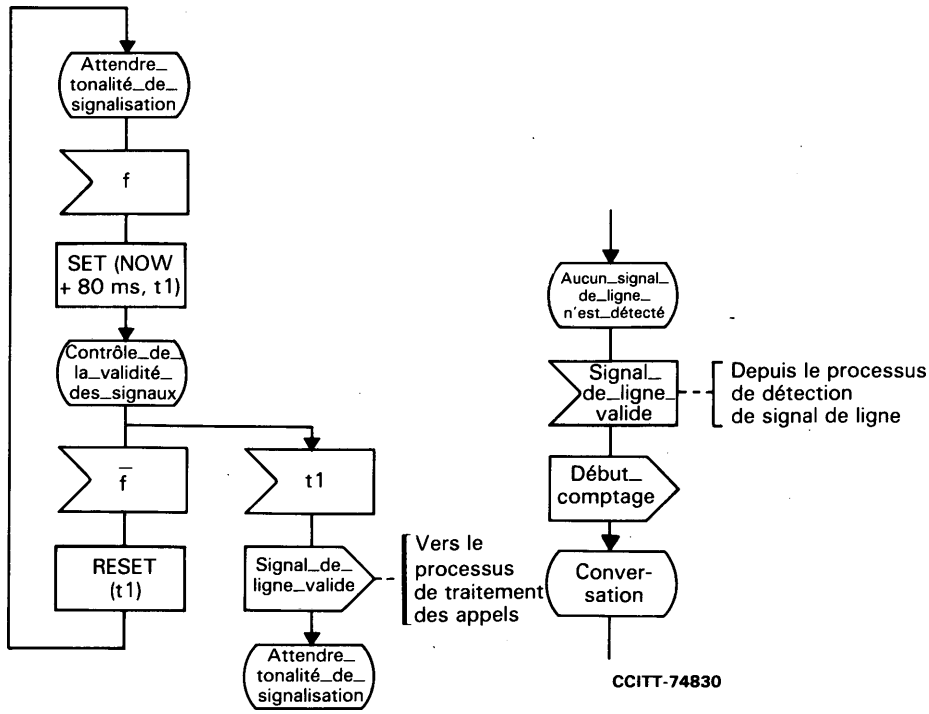


FIGURE D-15

Exemple de diagramme LDS correspondant à une fonction composite de traitement des appels et de détection de signal de ligne



a) Processus de détection de signal de ligne

b) Processus de traitement des appels

FIGURE D-16

Exemple de ségrétation d'une fonction commune (détection de signal de ligne) afin d'éviter des états répétitifs tels que le contrôle de la validité des signaux (à comparer à la figure D-15)

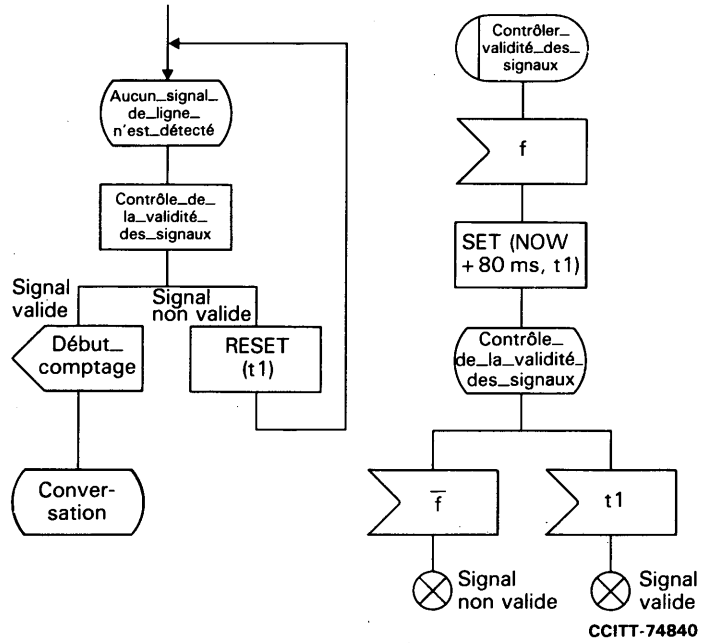


FIGURE D-17

Exemple de la figure D-16 avec emploi de macros

Il convient de noter la légère différence entre le processus de la figure D-15 et les deux processus de la figure D-16.

Le processus de la figure D-15 commence à compter dès réception de t1 alors que le processus de traitement des appels (processus b de la figure D-16) commence à compter dès réception de signal_de_ligne_valable qui est engendré et émis à la réception de t1 par le processus a de la figure D-16. Il s'ensuit que, dans le second cas, le comptage démarre après t1 + temps de génération, d'émission et de réception du signal. En outre, d'autres signaux peuvent avoir été mis dans la file d'attente pendant le temps nécessaire à la génération et à l'émission de signal_de_ligne_valable.

Une seconde particularité est que cette séparation d'une sous-fonction ne serait pas possible si le signal que doit recevoir la sous-fonction devait être reçu également par la fonction principale, un signal étant toujours acheminé vers un seul processus.

Si, dans l'exemple, le même signal f, devait être reçu dans un autre état où il n'est pas besoin de le valider pour le temps t1, il n'aurait pas été possible de séparer la sous-fonction de validation en un autre processus.

Les solutions qui font appel à des processus distincts sont généralement utiles lorsque les signaux doivent être traités indépendamment des états dans le processus principal. Dans ce cas, les opérations qui précèdent et qui suivent les processus peuvent traiter les séquences détaillées et décharger un processus principal de tous les détails. Ceci engendre souvent une modularité utile permettant, par exemple, d'isoler les caractéristiques particulières des systèmes de signalisation, du processus principal plus axé sur le service.

Une solution différente du problème consisterait à employer la notation MACRO, comme indiqué à la figure D-17. Dans ce cas, on obtient pour le diagramme la compacité voulue sans modifier en rien la sémantique du diagramme original. Il est en outre possible d'appeler la MACRO à partir de plusieurs états si la logique du processus l'exige.

D.4.3.4.1.4 *Fusion des états*

Si, dans un diagramme LDS, la destination future de deux états est la même, on peut, indépendamment de leur évolution antérieure, effectuer la fusion de ces deux états en un seul, sans affecter la logique du diagramme.

La figure D-18 représente une partie d'un diagramme LDS comportant deux états dont le «passé» est différent mais dont le «futur» est identique. Dans la figure D-19, les deux états ont été combinés en un seul. Il s'agit là d'un exemple relativement simple dans lequel la réduction de la complexité est peu importante, mais cette même technique peut être utilisée pour obtenir une plus grande simplification. La sémantique du LDS ne prévoit pas une décision consécutive à un état pour déterminer le «passé» du processus antérieurement à cet état (c'est-à-dire de déterminer, pour cet exemple, si A6 ou B4 a été émis), à moins que cette information n'ait été explicitement stockée avant l'entrée dans l'état. A noter, que l'attribution d'un nom aux données d'une entrée a pour effet de mettre la valeur en mémoire.

Un état doit représenter une situation logique du processus et il n'est donc pas souhaitable d'effectuer la fusion de situations logiques différentes en un seul état.

Des précautions sont à prendre si un diagramme fusionné doit être modifié ultérieurement,. L'usager devrait rechercher si la modification envisagée a ou non le même effet sur les divers trajets d'acheminement initiaux.

D.4.3.4.2 *Entrées*

Le présent § D.4.3.4.2 a pour objet d'expliquer la notion d'entrée ainsi que l'utilisation des entrées dans des diagrammes LDS ne faisant pas appel à la notion de mise en réserve. La notion de mise en réserve et l'utilisation de cette notion en même temps que la notion d'entrée font l'objet du § D.4.3.4.3.

D.4.3.4.2.1 *Considérations générales*

Un symbole d'entrée attaché à un état signifie que, si le signal dont le nom figure dans le symbole d'entrée arrive pendant que le processus est dans cet état, il faut interpréter la transition qui suit le symbole d'entrée. Lorsqu'un signal a déclenché l'interprétation d'une transition, ce signal n'existe plus et on dit qu'il a été absorbé.

Un signal peut être accompagné de données associées. Par exemple, un signal portant le nom «chiffre» sert non seulement à déclencher l'exécution d'une transition par le processus de réception mais encore à le véhiculer avec la valeur du chiffre (0 à 9), ces données pouvant être utilisées par le processus récepteur.

Pour que ces données élémentaires soient utilisables par le processus, il faut que, dans les symboles d'entrée, elles portent un nom entre parenthèses. Les données sont ainsi utilisables pendant la transition. Si les données ne sont pas explicitement mises en mémoire, elles ne seront pas accessibles au cours d'une transition ultérieure.

Toute donnée élémentaire est séparée des autres données au moyen de virgules. Les figures D-20, D-21 et D-22 donnent des exemples de la réception des données provenant des entrées.

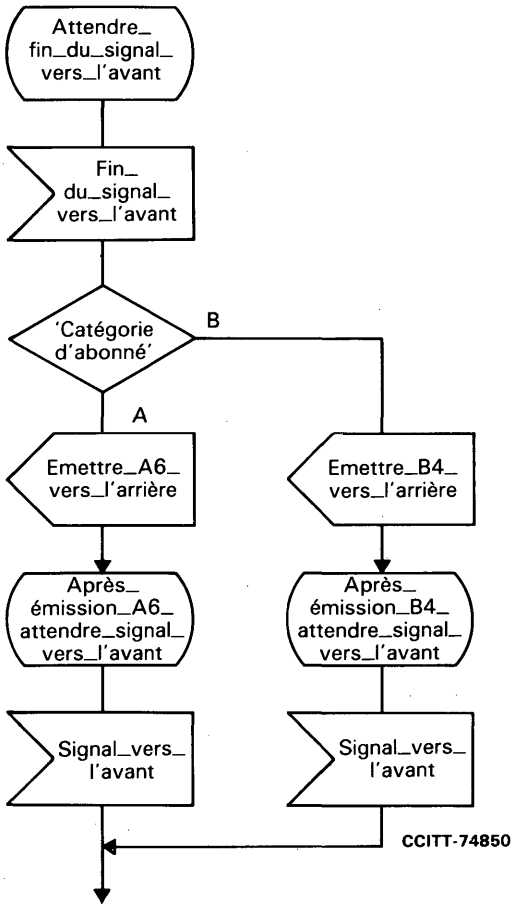


FIGURE D-18

Exemple d'une partie d'un diagramme LDS avant la fusion des états

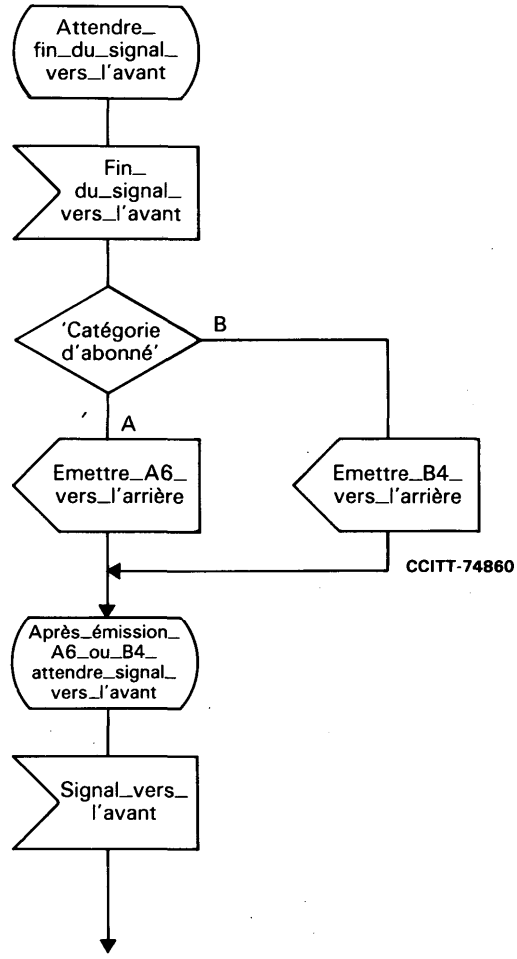


FIGURE D-19

Exemple d'une partie d'un diagramme LDS après fusion des états

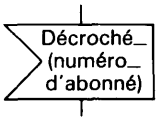
Les données auxquelles un nom est assigné peuvent être utilisées par le processus récepteur quand l'entrée est interprétée.

La figure D-20 montre la réception de signal_décroché. Le signal_décroché est accompagné de données associées (numéro_de_l'abonné) avec pour valeur 9269. Le signal peut être reçu de trois manières, comme indiqué en a-c de la figure.

La figure D-22 montre comment envoyer et recevoir plusieurs éléments de données en un seul signal. Chaque élément doit être séparé du suivant par une virgule. La figure D-22 c) montre comment ignorer les éléments indésirables en laissant un blanc dans la liste.

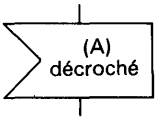
A noter que, dans le signal de sortie, il est impossible d'écrire des expressions pour E1, E2 ou E3 alors que, dans le signal d'entrée, il faut employer des variables pour recevoir les valeurs émises.

Dans le LDS, il n'est pas nécessaire de dessiner des symboles d'entrée pour représenter les signaux dont l'arrivée nécessiterait une transition nulle (c'est-à-dire une transition qui ne contient aucune action et qui ramène au même état). La convention admise est la suivante: pour tout signal qui n'est pas représenté dans un symbole d'entrée explicite à un état donné, il existe, dans cet état, un symbole d'entrée implicite et une transition nulle. Conformément à cette convention, les deux diagrammes représentés dans la figure D-23 sont logiquement équivalents et peuvent être indistinctement utilisés.



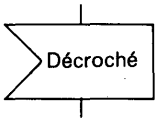
Remarque — La valeur 9269 est stockée dans une variable appelée numéro_d'abonné.

a)



Remarque — La valeur 9269 est stockée dans une variable appelée A.

b)



CCITT-74870

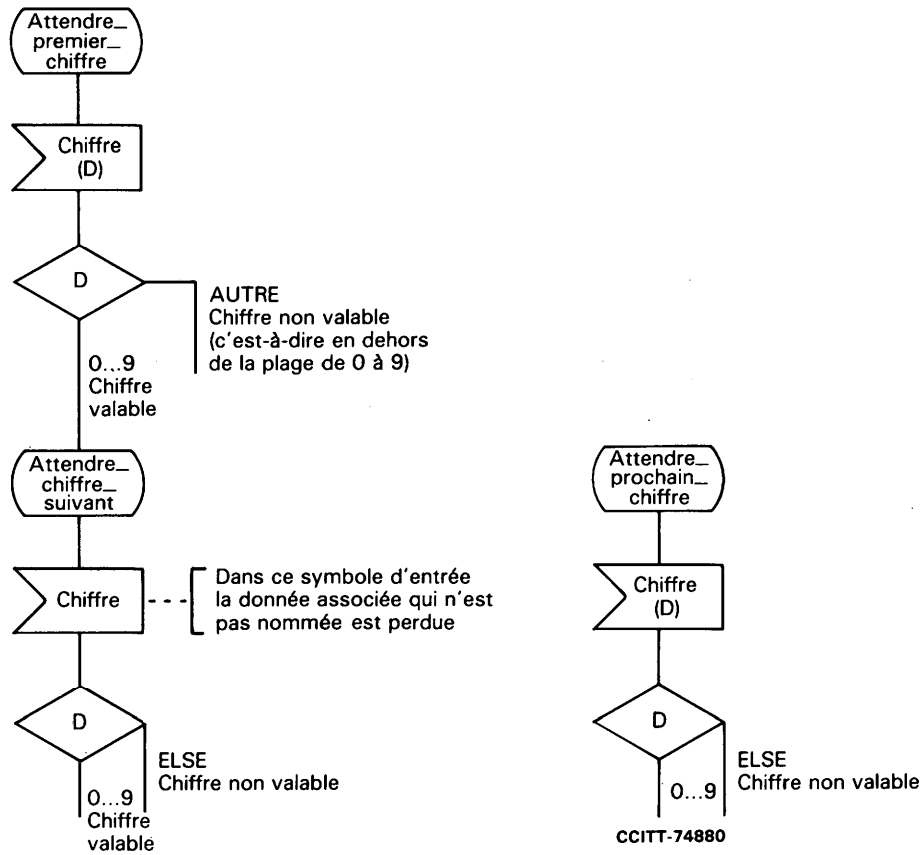
Remarque 1 — La donnée n'a pas de nom, la valeur 9269 est perdue et le processus de réception ne peut l'obtenir.

Remarque 2 — Le nom du signal (décroché) doit correspondre au nom du signal de sortie mais les noms des données peuvent correspondre ou non.

c)

FIGURE D-20

Exemple de réception des données dans un processus



a) Indiquer une version incorrecte *b) Indiquer une version correcte*

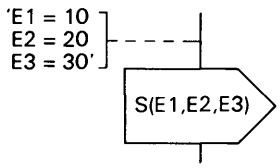
Remarque 1 — Dans *a)* on retiendra pour la deuxième décision la valeur de D obtenue avec le premier chiffre.

Remarque 2 — Dans *b)* on retiendra pour D la valeur du chiffre du moment. Les valeurs des chiffres antérieures seront éliminées.

Remarque 3 — Etant donné que l'on considère D comme un paramètre formel du chiffre, la valeur de D est automatiquement mise en mémoire.

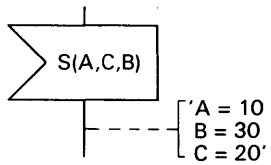
FIGURE D-21

Partie d'un processus récepteur de chiffres



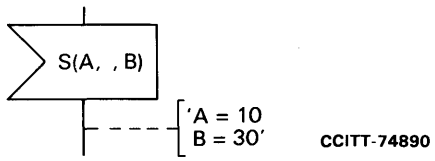
Remarque — Le signal de sorties S a trois variables appelées E1, E2 et E3. Ces éléments correspondent à trois valeurs, dans le cas présent 10, 20 et 30.

a)



Remarque — Le signal d'entrée correspondant S dans le processus de réception nomme respectivement ces éléments A, C et B.

b)

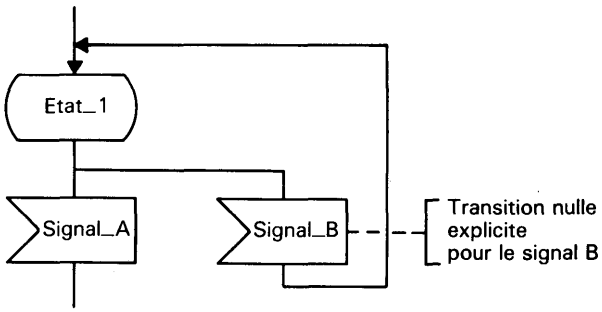


Remarque — Ce signal d'entrée ne nomme que deux variables. La valeur intermédiaire est perdue.

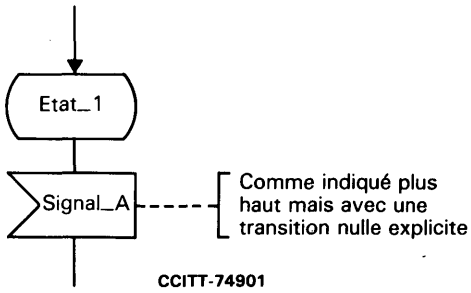
c)

FIGURE D-22

Signal avec plusieurs données élémentaires associées



a) Transition «nulle» explicite



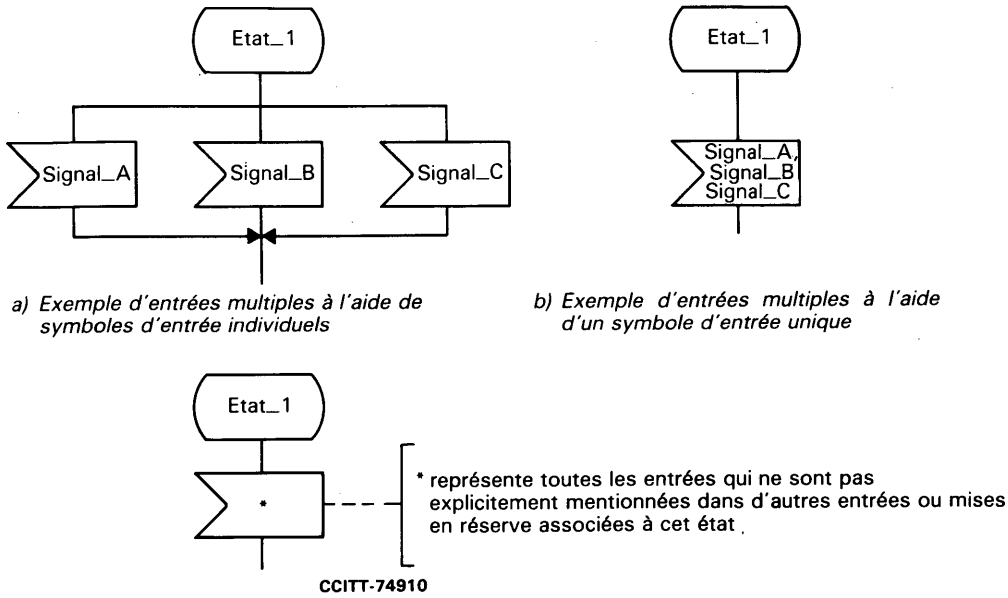
b) Transition «nulle» implicite

Remarque — Si une donnée est associée au signal B, elle est perdue dans l'un et l'autre cas. Si cependant un nom de donnée apparaît (par exemple, signal_B(x)) dans le cas *a)* la valeur de la donnée est maintenue. Les deux cas ne sont plus alors identiques.

FIGURE D-23

Représentation explicite et implicite d'une transition «nulle»

Lorsqu'un certain nombre d'entrées aboutissent à la même transition, tous les noms des signaux qui s'y rapportent peuvent être placés dans un même symbole d'entrée. La figure D-24 illustre cet aspect et les diagrammes qui y sont représentés sont logiquement équivalents. Si tous les noms de signaux sont mentionnés, il faut les séparer par des virgules.



a) Exemple d'entrées multiples à l'aide de symboles d'entrée individuels

b) Exemple d'entrées multiples à l'aide d'un symbole d'entrée unique

c) Exemple d'entrées multiples à l'aide d'une notation par astérisque

FIGURE D-24

Autre représentation possible des entrées multiples

D.4.3.4.2.2 *Mécanisme implicite de mise en file d'attente*

Un ou plusieurs signaux peuvent être en attente d'absorption lorsqu'un processus parvient à un nouvel état. Cela signifie que les signaux doivent être mis en attente d'une certaine manière si l'on veut éviter qu'ils soient perdus. Lorsqu'un signal parvient au bloc de destination, il est placé dans la file d'attente d'entrée du processus de réception. La sémantique du LDS définit pour chaque processus un mécanisme théorique de mise en file d'attente selon lequel le mode de sélection des signaux pour l'absorption par le processus est fondé sur l'ordre d'arrivée des signaux dans ce processus. Lorsque le processus parvient à un état, il reçoit un seul signal en provenance de la file d'attente. Ceci signifie que si la file d'attente n'est pas vide, le processus absorbe le premier signal qui vient de la file d'attente en question. Si cette dernière est vide, le processus demeure en attente dans l'état jusqu'à l'arrivée à la file d'attente d'un signal qui est ensuite absorbé par le processus.

La figure D-25 utilise le concept de file d'attente pour expliquer le fonctionnement d'un processus LDS dans lequel les temps de transition sont différents de zéro. Il convient de noter les éléments suivants:

- le concept de mise en réserve n'est pas appliqué et les signaux sont absorbés dans l'ordre de leur arrivée;
- l'ordre de succession de l'arrivée des signaux est important. Si «C» était arrivé avant «B» dans la transition entre l'état 1 et l'état 2, la séquence des états aurait été 1, 2, 3 au lieu de 1, 2, 4;
- la file d'attente n'étant pas vide lorsque le processus arrive aux états 2 et 4, ce processus ne doit attendre ni dans l'un ni dans l'autre de ces états;
- il n'est pas possible d'attribuer la priorité à un signal.

Si les temps de transition sont nuls, chaque signal sera absorbé au moment de son arrivée dans un processus, sauf si l'on a recours à la mise en réserve (voir le § D.4.3.4.3).

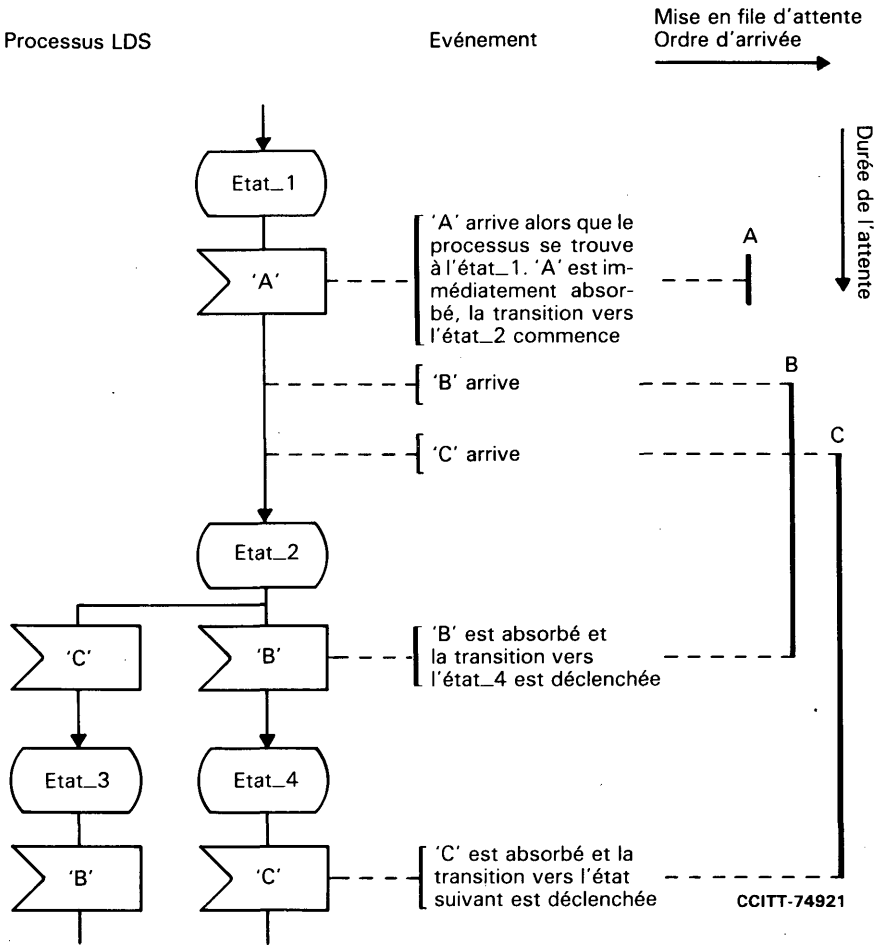


FIGURE D-25

Exemple de fonctionnement du mécanisme de mise implicite en file d'attente

D.4.3.4.2.3 *Réception des signaux qui ne se présentent pas normalement*

Dans chacun des états, tous les signaux possibles doivent être indiqués implicitement ou explicitement. Des exceptions (signaux inattendus mais théoriquement possibles, signaux non définis ou logiquement faux à un endroit donné, etc.) peuvent se présenter dans presque tous les états. Normalement, l'auteur n'indique pas ces possibilités; il s'ensuit qu'un tel signal sera éliminé s'il se présente. Si toutefois l'auteur tient à faire figurer les exceptions dans son diagramme, il doit prévoir pour *tous* les états une entrée supplémentaire.

Une autre possibilité consiste à profiter des apparitions multiples d'un état (voir le § D.6.3.6.5.1) ainsi que du symbole «tous» (*) (voir le § D.6.3.6.21). Par exemple, si le signal A_RACCROCHE peut être reçu dans tous les états et si les actions postérieures sont identiques, on peut recourir à la méthode indiquée à la figure D-26.

D.4.3.4.2.4 *Arrivée simultanée de signaux*

La Recommandation Z.101 prévoit que les signaux peuvent parvenir simultanément à un processus et précise qu'ils seront placés dans un ordre arbitraire.

Si un usager met au point un processus capable de recevoir des signaux simultanés, il doit veiller à ce que l'ordre d'arrivée ne bouleverse pas le fonctionnement escompté du processus.

Le LDS ne préconise pas de priorité parmi les signaux; ainsi, en cas d'arrivée simultanée de signaux, l'un d'eux est choisi arbitrairement.

Si plusieurs signaux sont disponibles au moment de l'entrée du processus dans un état, un seul signal est présenté au processus puis reconnu comme une entrée. Selon la sémantique du LDS, les autres signaux sont en fait retenus.

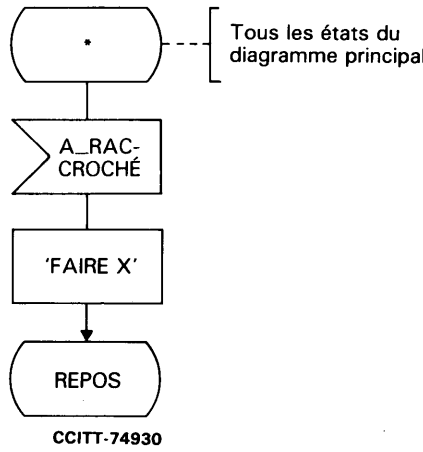


FIGURE D-26

Exemple de traitement des signaux pour qu'ils apparaissent dans plusieurs états

D.4.3.4.2.5 *Entrées externes et entrées internes*

Un signal externe est un signal entre processus appartenant à des blocs fonctionnels différents; un signal interne est un signal entre processus appartenant au même bloc fonctionnel. Sémantiquement, il n'y a aucune différence entre les signaux externes et les signaux internes. Dans les Recommandations antérieures relatives au LDS, on avait toutefois établi une distinction syntaxique. Ces différences syntaxiques ont été éliminées et, aujourd'hui, les signaux internes et les signaux externes ont des syntaxes identiques. L'ancienne syntaxe GR pour un signal interne (deux lignes verticales dans le symbole) n'est plus employée. Les anciens diagrammes où apparaissent des signaux internes sont tolérés mais les signaux internes sont interprétés de la même manière que les signaux externes.

D.4.3.4.2.6 *Identification de l'émetteur*

Chaque signal est porteur de l'identificateur d'instance de processus du processus émetteur. A la réception d'un signal, une variable de processus nommée ÉMETTEUR prend la valeur de l'identificateur d'instance de processus du processus émetteur; cet identificateur est porté par le signal. On trouvera à la figure D-27 un exemple d'emploi de cette variable ÉMETTEUR.

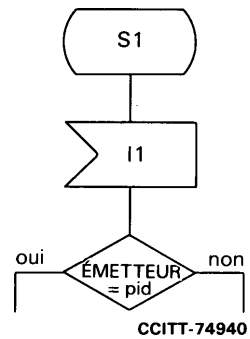


FIGURE D-27

Emploi de la variable ÉMETTEUR

D.4.3.4.3 *Mises en réserve*

Le concept de mise en réserve permet de différer l'absorption d'un signal jusqu'à ce qu'un ou plusieurs signaux ultérieurs aient été absorbés. Comme l'indique le § D.4.3.4.2, les signaux sont absorbés dans l'ordre dans lequel ils se présentent, sauf en cas d'emploi du concept de mise en réserve.

L'on peut faire appel au concept de mise en réserve afin de simplifier les processus lorsque l'ordre relatif d'arrivée de certains signaux n'a pas d'importance et que leur ordre effectif d'arrivée est indéterminé.

Dans chaque état, chaque signal est traité comme suit:

- il est représenté comme un symbole d'entrée ou,
- il est représenté comme un symbole de mise en réserve ou,
- il est, par convention, couvert par une entrée implicite aboutissant à une transition nulle implicite.

Le fonctionnement du mécanisme implicite de mise en file d'attente décrit dans le § D.4.3.4.2 s'applique également au concept de mise en réserve. A leur arrivée, les signaux sont placés dans la file d'attente et lorsque le processus atteint un état donné, les signaux qui se trouvent dans la file d'attente sont examinés un à un dans l'ordre de leur arrivée. Un signal couvert par un symbole d'entrée explicite ou implicite est absorbé et la transition qui s'y rapporte est exécutée. Un signal représenté dans un symbole de mise en réserve n'est pas absorbé et reste dans la file d'attente. Dans la même position séquentielle; le signal suivant de la file d'attente est considéré.

La Figure D-28 représente un exemple d'un processus LDS qui comporte un symbole de mise en réserve. Il convient de noter que les signaux S et R sont consommés dans l'ordre R, S, c'est-à-dire dans l'ordre inverse de leur réception. Un symbole de mise en réserve unique peut servir à mettre un signal en réserve tant que le processus se trouve dans l'état dans lequel le symbole apparaît; ce signal est mis en réserve pour la durée de la transition vers le prochain état. Dans le prochain état, le signal sera absorbé par l'intermédiaire d'une entrée explicite saug dans les cas suivants; a) lorsque le symbole de mise en réserve comportant le nom du signal est répété (voir la figure D-28) ou b) lorsque dans la file d'attente implicite, il existe avant lui, un autre signal de sauvegarde disponible pour l'absorption (voir la figure D-29).

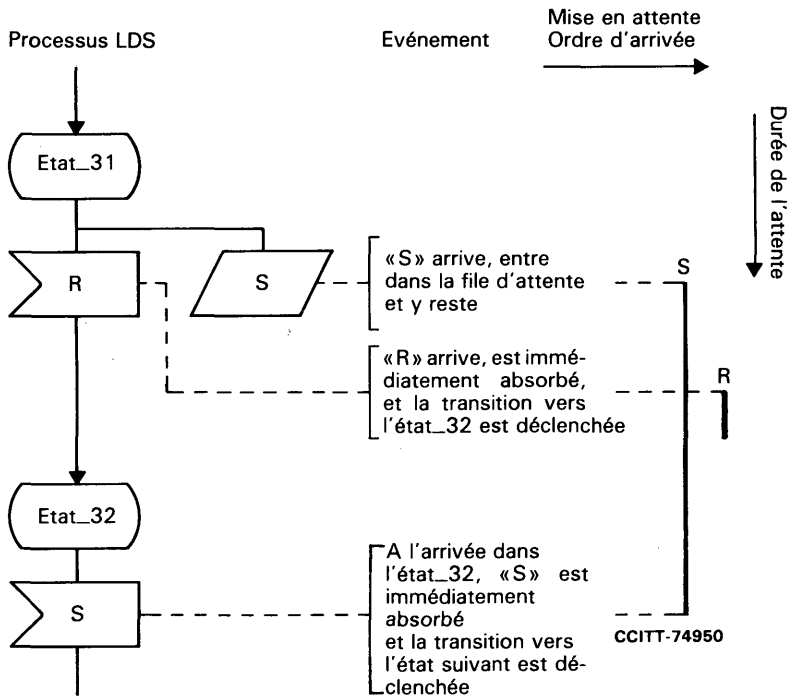


FIGURE D-28

Exemple de diagramme LDS avec symbole de mise en réserve montrant le fonctionnement d'un mécanisme implicite de mise en attente

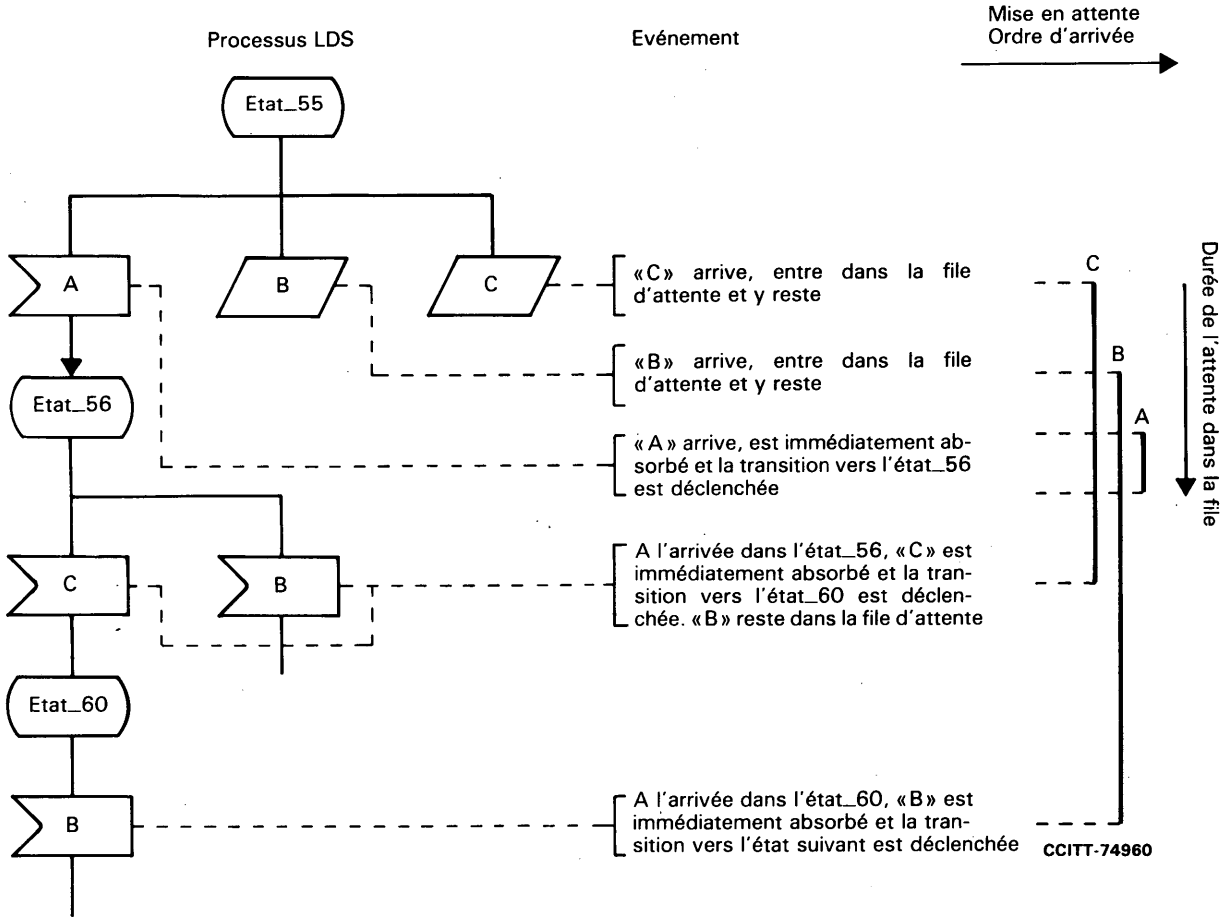


FIGURE D-29

Deuxième exemple de l'utilisation du symbole de mise en réserve

Un signal mis en réserve n'est mis à disposition d'un processus que par l'intermédiaire d'un symbole d'entrée correspondant (explicite ou implicite). Aucune question relative à un signal mis en réserve ne peut être posée dans une décision avant la reconnaissance de ce signal comme une entrée; de même les données qui lui sont associées ne sont pas disponibles.

Dans un état où plusieurs signaux doivent être mis en réserve, on peut attribuer un symbole de mise en réserve à chaque signal ou on peut les représenter tous à l'intérieur du même symbole de mise en réserve.

Si plusieurs signaux doivent être mis en réserve, la sémantique du symbole de mise en réserve exige que l'ordre de leur arrivée soit préservé.

Un troisième exemple de l'utilisation de la notion de mise en réserve est donné dans la figure D-30 et la figure D-31 décrit le fonctionnement du mécanisme théorique de formation de file d'attente.

L'utilisation du symbole de mise en réserve peut simplifier les diagrammes. Ainsi, en mettant un signal en réserve, l'on peut éviter de le recevoir et de devoir mettre en mémoire ses données associées jusqu'à l'état suivant.

Bien que le symbole de mise en réserve puisse être utilisé à chaque niveau de description, il y aurait peut-être lieu, au niveau inférieur, de décrire le mécanisme effectif qui permet cette mise en réserve.

Sans un minimum de précautions dans l'emploi de la mise en réserve, la file d'attente des signaux mis en réserve peut augmenter considérablement ou garder des signaux en mémoire trop longtemps, de sorte qu'un signal ancien peut être absorbé lorsqu'un signal récent est demandé.

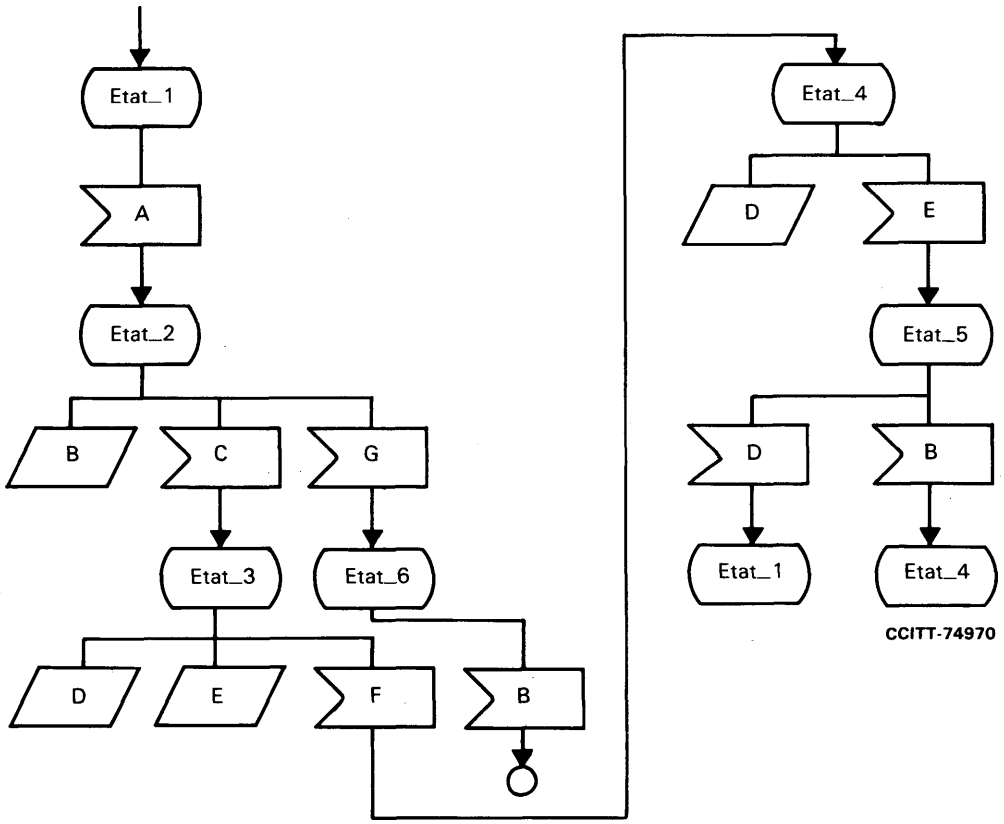


FIGURE D-30

Exemple de diagramme LDS plus complexe avec plusieurs entrées et mises en réserve

Etat actuel	Evénement
Etat 1	(Le processus entre dans l'état 1 avec les signaux «A», «B», «C», «D», «E» dans une file d'attente) Le premier signal de la file d'attente, «A», est absorbé (entrée explicite) et la transition vers l'état 2 est déclenchée
Etat 2	Le premier signal de la file, «B», apparaît dans un symbole de mise en réserve et reste dans la file d'attente
Etat 2	Le deuxième signal, «C», est absorbé (entrée explicite) et la transition vers l'état 3 est déclenchée
Etat 3	Le premier signal de la file d'attente, «B», est absorbé (entrée implicite) et une transition nulle est déclenchée
	«F» arrive et entre dans la file d'attente
Etat 3	(En arrivant à nouveau dans l'état 3) le premier signal de la file, «D», apparaît dans un symbole de mise en réserve et reste dans la file d'attente
Etat 3	Le deuxième signal, «E», apparaît dans un symbole de mise en réserve et reste dans la file d'attente
Etat 3	Le troisième signal, «F», est absorbé (entrée explicite) et la transition vers l'état 4 est déclenchée
Etat 4	Le premier signal de la file, «D», apparaît dans un symbole de mise en réserve et reste dans la file d'attente
Etat 4	Le deuxième signal, «E», est absorbé (entrée explicite) et la transition vers l'état 5 est déclenchée
Etat 5	Le premier (et seul) signal de la file, «D», est absorbé (entrée explicite) et la transition vers l'état 1 est déclenchée

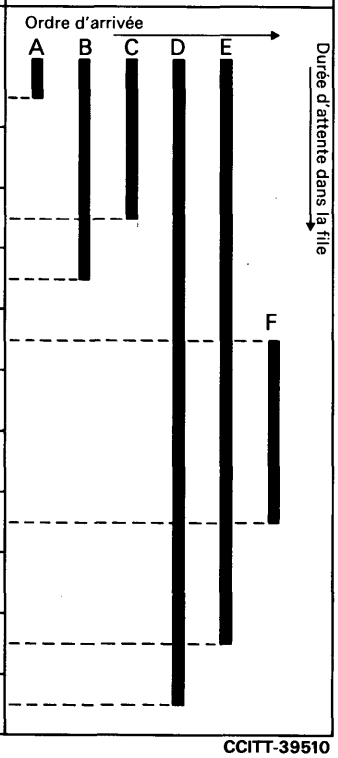


FIGURE D-31

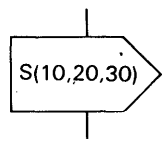
Fonctionnement du mécanisme théorique de mise en file d'attente

D.4.3.4.4 *Sorties*

Un symbole de sortie représente l'émission d'un signal d'un processus vers un autre. Etant donné que le contrôle de la réception et de l'absorption du signal est associé au processus de réception (voir le § D.4.3.4.2), les règles sémantiques se rapportant directement aux symboles de sorties sont relativement simples. Du point de vue du processus d'émission, une sortie peut souvent être considérée comme une action instantanée qui, une fois achevée, n'a aucun autre effet direct sur le processus d'émission, lequel ne sera pas directement conscient du sort du signal.

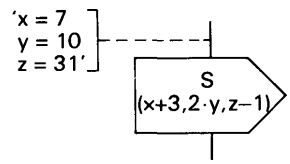
Si l'auteur éprouve de la difficulté à décider si une action doit être présentée comme une sortie ou comme une tâche, il consultera utilement le § D.4.3.4.6. Une action de sortie représente l'émission d'un signal et des valeurs de données qui peuvent éventuellement lui être associées. Il est possible d'associer des valeurs de données à un signal de sortie en plaçant ces valeurs entre parenthèses (voir la figure D-32 et la figure D-22).

Chaque sortie doit être orientée vers une instance de processus donnée. Etant donné qu'il est généralement impossible de connaître l'identificateur d'instance de processus de tout processus au moment de l'élaboration d'une spécification ou d'une description, la méthode normale pour orienter les signaux consiste à employer une variable ou une expression dans le symbole ou le mot-clé TO (VERS). On trouvera dans les figures D-33, D-34 et D-35 des exemples. Dans la figure D-33, le paramètre de processus «sortie vers» prend la valeur d'un identificateur d'instance de processus lors de la création du processus. Le rôle de «sortie vers» au sein du processus est d'assurer la liaison entre le processus en question et le processus auquel il est connecté. Il convient de veiller lors de la conception du système, à ce que le type de processus indiqué par «sortie vers» puisse recevoir les signaux qui sont émis. Dans la figure D-34, la variable de processus intégrée sert à renvoyer un signal au processus qui a émis le signal reçu peu avant. Dans la figure D-35, le signal est envoyé vers le descendant le plus récent du processus.



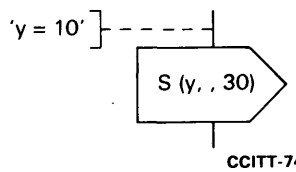
Remarque — Le signal S a trois valeurs, 10, 20 et 30 qui lui sont associées.

a)



Remarque — Pour l'interprétation de S: x, y et z ont (dans le présent exemple) les valeurs 7, 10, 31 respectivement. S émet les valeurs 10, 20, 30.

b)



Remarque — Pour l'interprétation de S, y a (dans le présent exemple) la valeur 10. S émet les valeurs 10 ainsi qu'une valeur indéfinie et 30.

c)

FIGURE D-32

Sortie avec données associées

```
PROCESSUS X; ("sortie_vers" PID);
.
.
.
SIGNAL DE SORTIE VERS "sortie_vers";
.
.
.
```

FIGURE D-33

Adressage de signaux à l'aide de paramètres formels

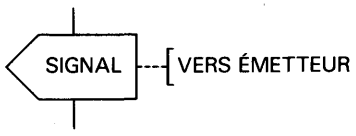


FIGURE D-34

Renvoi de signaux à ÉMETTEUR

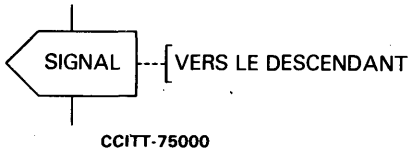


FIGURE D-35

Adressage d'un signal vers un descendant

D.4.3.4.5 *Condition de validation et signaux continus*

Les conditions de validation permettent une réception conditionnelle de signaux fondée sur la condition de validation spécifiée. Le signal est reçu et la transition interprétée si la condition est vraie. En cas de condition fausse, le signal est mis en réserve et le processus ne change pas d'état jusqu'à ce qu'un autre signal arrive ou que la condition fausse devienne vraie. L'exemple de la figure D-36 illustre ceci. Le processus commence à l'état S1. Dans le processus P2, la valeur de X est égale à 9. A l'arrivée du signal B, on compare la valeur de X avec 10. L'on notera, étant donné qu'il s'agit d'une valeur d'un autre processus, que l'opération d'importation est nécessaire afin de déterminer sa valeur présente. Vu que X n'est pas égal à 10, le processus demeure en S1 jusqu'à réception d'une autre entrée ou jusqu'à l'émission d'une nouvelle valeur de X par P2. Dans cet exemple A arrive et provoque la transition vers S2. Au cours de cette transition, X a pris la valeur 11; ainsi, la condition attachée au signal B dans l'état S2 est à présent vraie. Vu que B est le premier signal de la file d'attente, la transition qui suit s'opère, et le processus prend fin à l'état S3.

Certaines caractéristiques importantes des conditions de validation sont:

1) L'on ne teste la condition de validation que lorsque le processus arrive à un état. Ainsi, si la valeur de X était passée de 9 à 11, puis à 12 pendant la transition qui faisait suite à la réception de A, le processus serait resté à S2.

2) Les conditions de validation ne peuvent se fonder que sur des expressions qui contiennent des variables ayant l'attribut IMPORTED ou locales, et n'ayant pas l'attribut VIEWED. Ceci est dû au mécanisme fondamental servant à la mise en œuvre de la condition de validation. La condition de validation fonctionne au moyen de signaux qui détectent les changements de valeurs dans la condition de validation. Pour les variables locales, on vérifie la condition avant l'entrée dans l'état. Vu que ces valeurs sont locales par rapport au processus, elles ne peuvent changer pendant que le processus est dans l'état. Ainsi, il n'est pas nécessaire de contrôler la condition en permanence. Le LDS fait appel à des signaux émis pour détecter les changements des variables ayant l'attribut IMPORTED; cela permet de savoir à quel moment il convient de vérifier la condition de validation. Les variables ayant l'attribut VIEWED parviennent directement à la valeur de la variable. Etant donné qu'aucun signal n'est émis après que le processus soit entré dans un état, le processus ne peut en aucune façon vérifier la valeur de la variable.

3) Alors qu'il est possible d'employer plus d'une condition de validation par état, l'emploi de plus d'une condition de validation pour le même signal n'est pas autorisé. La condition que présente la figure D-37 n'est donc pas autorisée. Si un signal donné exige de multiples conditions, il est possible de les combiner en une expression Booléenne ainsi que le montre la figure D-38. Ceci supprime l'ambiguïté qui se présente lorsque deux ou plus de deux conditions sont vraies au même moment. Dans la figure D-38, l'usager a décidé de donner la priorité à la condition de X par rapport à celle de Y. Ainsi, si les deux conditions étaient vraies, le trajet suivi serait le trajet C1. De nouveau, ceci est dû au fait que le système fondamental du LDS n'accorde pas de priorité aux signaux.

Les signaux continus ont les mêmes propriétés fondamentales que la condition de validation, excepté le fait qu'ils ne sont accompagnés d'aucun signal. Ainsi, en l'absence de signaux dans la file d'attente susceptibles de provoquer une transition à leur entrée dans l'état, les signaux continus sont contrôlés; si l'un d'entre eux est vrai, la transition qui le suit s'opère. L'exemple de la figure D-39 en donne l'illustration. A l'origine, le processus se trouve à l'état S1 et la valeur de X est 9. En arrivant, le signal A provoque la transition vers l'état S2. Pendant la transition, X prend la valeur 11. Vu qu'aucun autre signal ne se trouve dans la file d'attente, la transition vers l'état S3 s'accomplit.

L'on trouvera ci-dessous certaines caractéristiques importantes des signaux continus:

1) De même que pour les conditions de validation, la valeur de la condition n'est contrôlée qu'à l'arrivée à un état.

2) De même, seules les variables ayant l'attribut IMPORTED et locales peuvent être employées dans les signaux continus.

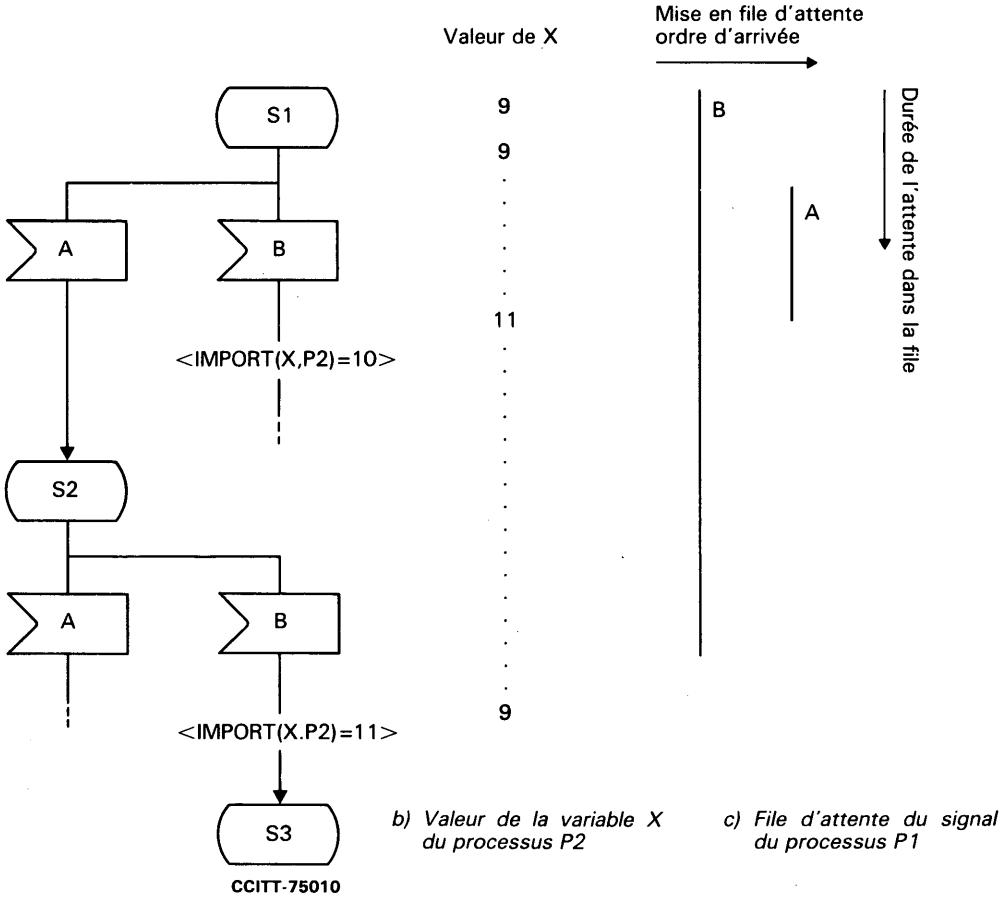
3) Les signaux continus multiples sont autorisés pour chaque état. Lorsque plusieurs signaux continus sont liés à un état, le premier à faire l'objet d'une mesure est le signal continu ayant le rang de priorité le plus bas. Deux signaux continus ne peuvent avoir le même rang de priorité. Le signal continu est toujours moins prioritaire que tout autre signal. Le LDS ne fait appel à des priorités que pour les signaux continus. Ceci est de nouveau dû au système fondamental du LDS; toutefois, la représentation à l'aide de modèles des signaux continus en LDS (emploi des signaux émis pendant l'importation de variables), permet le recours à des priorités pour les signaux continus; ceci est d'ailleurs nécessaire afin d'éviter toute ambiguïté en cas de présence de plusieurs signaux continus. La figure D-40 en donne une illustration. Le processus commence à l'état S1 et ses variables locales X et Y ont respectivement les valeurs 10 et 11. Etant donné que les deux signaux continus sont vrais, celui qui a la plus haute priorité (rang de priorité le plus bas) est choisi et la transition vers l'état S2 s'accomplit. En S2, la condition de Y n'est plus vraie, et bien que la priorité du signal continu de X soit inférieure à celle de Y, la transition qui le suit s'accomplit et le processus parvient à l'état S3.

D.4.3.4.6 *Tâche*

Une tâche sert à représenter des opérations sur des données, accomplies pendant une transition, à l'exception de la production de signaux de sortie et de décisions. Seul le processus peut modifier les données s'il en est le détenteur. La nature des tâches qui apparaissent dans un diagramme quelconque sera déterminée par la nature du processus décrit et par le niveau de détail requis. Comme exemple d'action que l'on peut représenter par des tâches, on peut citer:

a) les actions se rapportant au matériel, telles que envoyer_tonalité_d'occupation;

b) la manipulation des données.



a) Partie du processus P1

b) Valeur de la variable X du processus P2

c) File d'attente du signal du processus P1

FIGURE D-36

Condition de validation

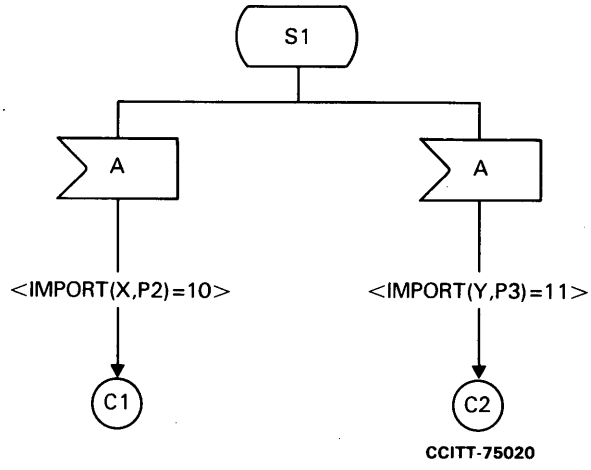


FIGURE D-37

Condition de validation illicite

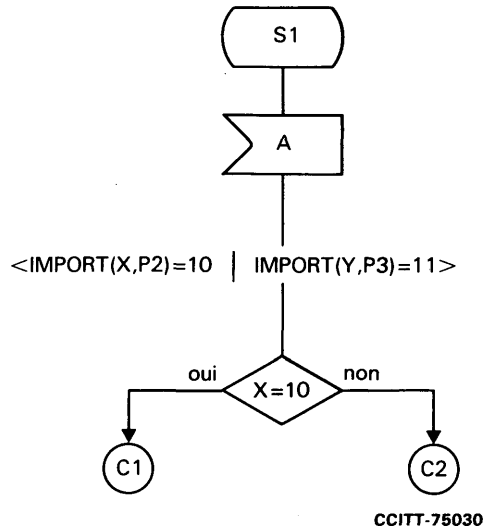
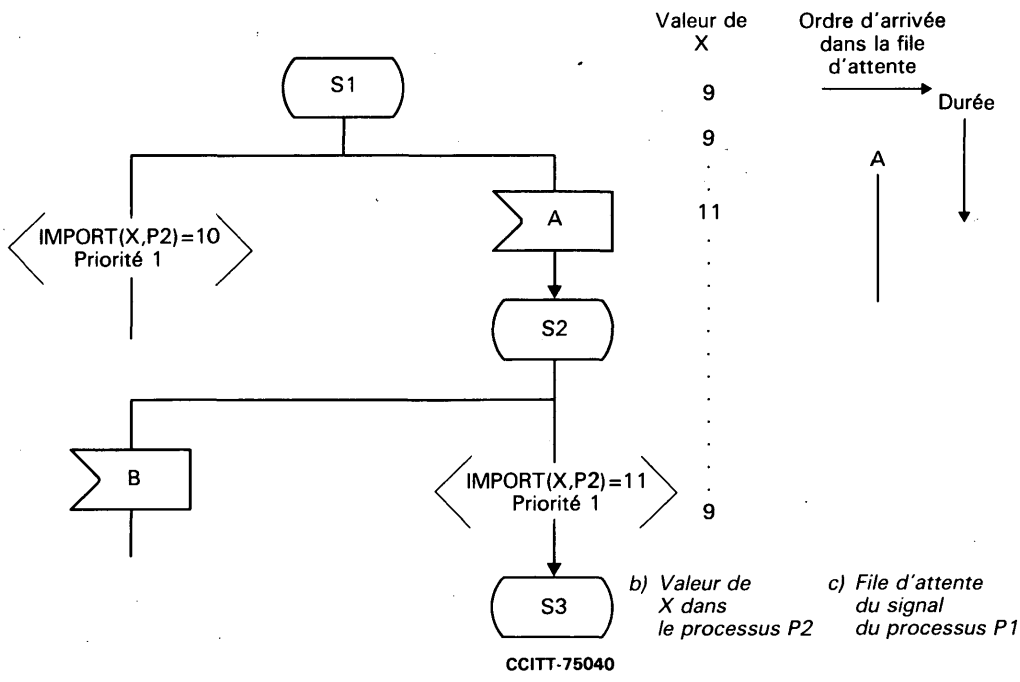


FIGURE D-38

Solution correcte applicable à la figure D-37



a) Partie du processus P1

b) Valeur de X dans le processus P2

c) File d'attente du signal du processus P1

FIGURE D-39

Signaux continus

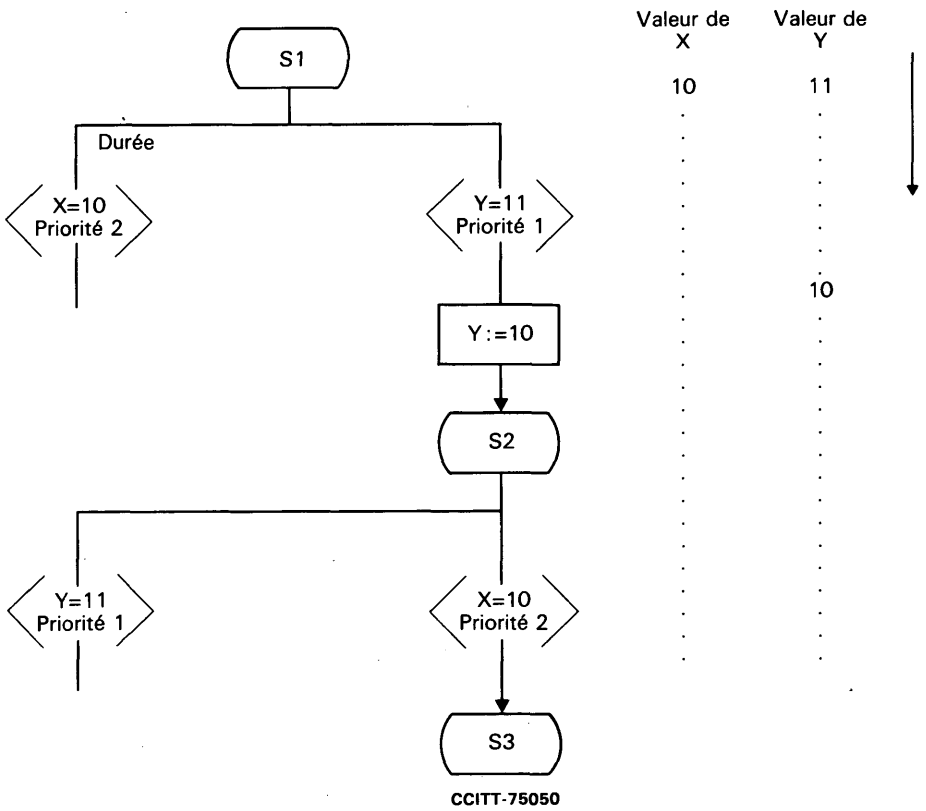


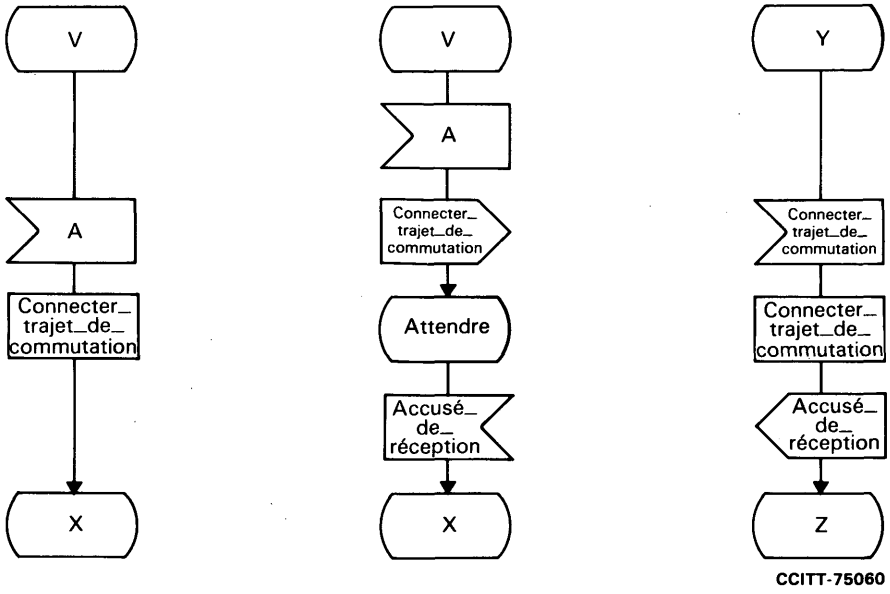
FIGURE D-40

Signaux continus avec priorité

Les usagers du LDS peuvent parfois éprouver de la difficulté à décider si un aspect du système défini doit être représenté par une tâche ou un processus distinct. Considérons l'exemple du processus représenté dans la figure D-41; l'action «connecter-trajet-de-commutation» doit-elle être représentée par une tâche ou par un processus distinct? Si l'on n'a pas identifié un processus distinct de commande de trajet de commutation, il serait indiqué d'utiliser le symbole tâche [figure D-41 a)]. Si on a identifié un processus distinct de commande de trajet de commutation, on doit utiliser les signaux qui communiquent avec le processus de commande [figure D-41 b)].

D.4.3.4.7 *Décisions*

Une décision est une action qui se produit au cours d'une transition et qui correspond à une question concernant la valeur des données élémentaires disponibles pour le processus au moment de l'exécution de l'action; le processus a le choix entre deux ou plusieurs trajets, ce choix étant déterminé par la réponse consécutive à la décision. Les auteurs des diagrammes LDS doivent veiller à ce que les processus soient définis de manière qu'ils ne puissent tenter d'exécuter des décisions pour lesquelles des réponses (ou les données) ne sont pas disponibles; de telles décisions rendraient le diagramme tout à fait incorrect et entraîneraient une confusion considérable.



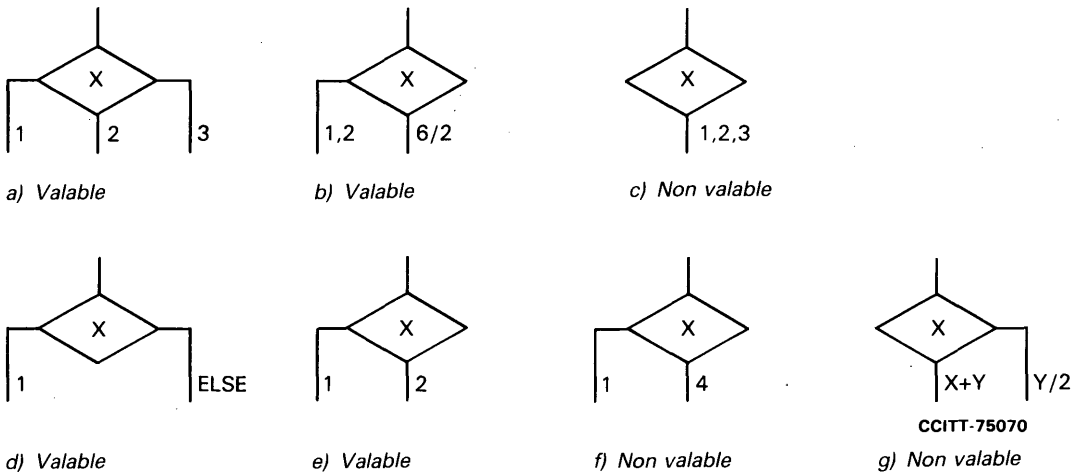
a) Solution utilisant un processus

b) Solution utilisant deux processus

FIGURE D-41

Deux solutions possibles pour « connecter_le_trajet_de_commutation »

La figure D-42 donne quelques exemples d'utilisation des décisions. La variable X dans ces exemples peut recevoir les valeurs 1, 2 ou 3.



a) Valable

b) Valable

c) Non valable

d) Valable

e) Valable

f) Non valable

g) Non valable

FIGURE D-42

Exemples d'utilisation de décisions

La variante c) n'est pas valable car la décision n'est pas suivie d'au moins deux trajets. La variante e) est valable car la convention suivante s'applique en LDS: si le choix d'une solution est impossible, l'on doit interpréter la décision comme une solution implicite débouchant directement sur un symbole arrêt. La variante f) n'est pas valable car l'une des variantes est hors de portée. La variante g) n'est pas valable étant donné que les conditions des variantes comportent des variables; il n'est donc pas possible de procéder à des évaluations statiques.

Si une réponse ramène à la décision située dans la même transition, il faut accomplir certaines actions qui influent sur la question qui se trouve dans la décision. Cependant, même avec cette règle, il peut y avoir des cas insolubles, comme indiqué à la figure D-43. Il convient donc d'être toujours prudent quand les réponses ramènent à une décision comprise dans la même transition.

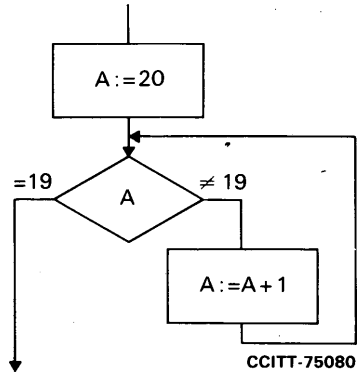


FIGURE D-43

Exemple d'emploi autorisé d'une décision donnant lieu à une situation insoluble

Les décisions peuvent être prises à l'aide des valeurs dont le processus dispose à un instant donné et notamment:

- données stockées pour une tâche,
- données reçues par une entrée,
- données fournies lors de la création du processus (données transmises en tant que paramètre effectif),
- données partagées.

Pour une décision, la question posée peut être: «quelle est la valeur d'une expression?» l'expression pouvant contenir des constantes et l'un quelconque des types de valeurs de données précitées.

D.4.3.5 *Procédures*

Les procédures constituent une technique permettant d'ajouter une structure à des processus. Les processus en LDS sont, à maints égards, identiques aux procédures des langages de programmation. Toutefois, l'usager doit savoir qu'il existe des différences majeures; une certaine prudence est conseillée. Compte tenu de cela, nous pouvons aborder le sujet des procédures.

D'un point de vue syntaxique, une procédure ressemble fortement à un processus. Le nœud de départ d'une procédure du graphe d'une procédure est légèrement différent du nœud de départ d'un processus; la procédure est dotée d'un nœud de retour au lieu d'un nœud d'arrêt. D'un point de vue sémantique, cette différence est relativement importante bien que des nœuds identiques de processus et de procédures puissent avoir la même signification.

Des appels de procédures peuvent se présenter dans un graphe de processus ou dans un graphe de procédure partout où un nœud de tâche est autorisé. On peut d'une certaine façon considérer une procédure comme une tâche, avec les exceptions suivantes:

1) Une procédure peut comporter des états; elle est donc appelée à recevoir des signaux. Une partie de la définition de la procédure concerne un ensemble de mise en réserve supplémentaire qui énumère les signaux à mettre en réserve en cas d'appel de la procédure. Etant donné qu'il est possible de définir des procédures au niveau du système, il importe de répertorier tous les signaux que l'on doit mettre en réserve dans n'importe lequel des processus qui appellent la procédure.
2) Une procédure peut émettre des signaux de sortie; l'identificateur d'instance du processus de départ est l'identificateur d'instance de processus de celui qui a appelé la procédure.
3) Une procédure possède ses propres variables locales et n'a aucun accès aux variables du processus à moins que ces dernières aient été transmises à la procédure sous la forme de paramètres avec l'attribut IN/OUT.
4) Les paramètres SIGNAL sont autorisés à créer des synonymes des noms effectifs des signaux du processus d'appel, et des noms des signaux utilisés dans la procédure afin que la procédure puisse être appelée à plusieurs endroits.

En cas d'appel d'une procédure, l'environnement de cette dernière est créé; l'interprétation de la procédure commence, et se poursuit jusqu'à ce que le nœud de retour soit atteint. Pendant l'interprétation de la procédure, celle-ci met en réserve ou reçoit tous les signaux adressés vers le processus. La procédure n'a pas de file d'attente d'entrée propre, mais elle se sert de la file d'attente d'entrée du processus qui l'a appelée. C'est pourquoi il importe qu'une procédure dispose de l'ensemble supplémentaire de mise en réserve adéquat. L'ensemble supplémentaire de mise en réserve a une construction dynamique en cas d'appel de procédure; il est formé de tous les signaux d'entrée valables de la procédure d'appel qui ne sont pas considérés comme des paramètres joints à l'appel. Le signal est éliminé s'il ne se trouve pas dans ce dernier ensemble ni dans l'ensemble normal de mise en réserve du nœud d'état du graphe de la procédure.

Le § D.9 donne un exemple d'emploi des procédures. Il illustre l'emploi de la procédure relative aux spécifications de protocole.

D.4.3.6 *Expression du temps en LDS*

Dans le LDS, il existe plusieurs situations où une représentation du temps peut être nécessaire:

a) activation d'un processus à un instant donné (temps absolu ou relatif);
b) temps passé pour exécuter des actions de transition;
c) temps passé pour transférer un signal d'un processus à un autre processus qui peut faire intervenir une voie.

D.4.3.6.1 *Temporisateurs et temporisation*

Il est possible de mesurer le temps et de demander des temporisations à l'intérieur d'un système grâce à des temporisateurs et à l'ensemble d'opérations qu'ils permettent d'exécuter.

Le «temporisateur» est d'un type préalablement défini, et tout emploi de temporisateur dans un processus doit faire l'objet d'une déclaration. Les temporisateurs permettent les opérations «SET» (Positionnement) et «RESET» (Remise à zéro). Pour l'opération SET, une temporisation doit se produire à un moment donné; l'opération RESET supprime toute demande de temporisation (à noter: une opération SET s'accompagne automatiquement du RESET de toute temporisation en cours dans le temporisateur).

```
DCL T1 Temporisateur;
SET (NOW + 20 s, T1);
RESET (T1).
```

FIGURE D-44

Exemple de déclaration et d'emploi d'un temporisateur

Un temps absolu doit nécessairement être spécifié au cours de l'opération SET. Pour transformer un temps relatif en une valeur de temps absolue, il convient d'ajouter l'opération «NOW» (instant présent); l'on obtient ainsi l'instant présent. Ainsi, NOW + 20 s désigne un instant situé 20 secondes après l'instant présent.

Un contrôle des temporisateurs n'a lieu que lorsque le processus est dans un état. Un signal est introduit dans la file d'attente d'entrée du processus en cas de vérification du temporisateur et s'il est inférieur ou égal à l'instant présent. Le temporisateur est ensuite implicitement remis en marche.

Le signal de temporisation porte le nom du temporisateur et n'achemine pas de valeurs de données.

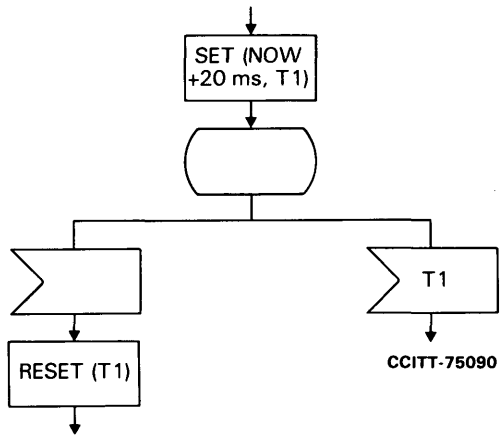


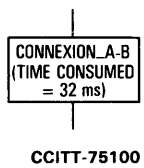
FIGURE D-45

Exemple d'emploi de temporisateurs

D.4.3.6.2 *Spécification du temps passé à l'exécution d'actions*

Le temps consacré aux transitions doit être spécifié lorsqu'on emploie le LDS pour simuler le fonctionnement de systèmes. Le LDS actuel ne prévoit pas cette fonction; cependant, les directives suivantes s'appliquent.

Dans de tels cas, une valeur temporelle (le temps passé à l'exécution d'une tâche donnée) est associée, en tant que paramètre, au nom de cette action. Ce paramètre facultatif peut être identifié par un mot clé (par exemple, TIME CONSUMED = valeur du temps).



TASK CONNEXION_A-B
TIME CONSUMED = 32 ms

a) Une action qui prend 32 ms est représentée par un symbole de tâche comportant indication du temps passé

b) Même action représentée en LDS/PR

FIGURE D-46

Indication du temps de transition

Le temps est indiqué comme paramètre associé du nom et non comme commentaire étant donné que le commentaire est censé fournir une explication à une personne et n'a pas à être compris par une machine.

D.4.3.6.3 *Temps de transfert des signaux*

Le temps qui s'écoule de la production (sortie) d'un signal à son absorption (entrée) est très difficile à exprimer car il dépend de plusieurs facteurs.

Le transport du signal d'un bloc à un autre est une action qui peut occuper un temps absolu qui peut s'exprimer (si le canal est représenté explicitement) comme la somme des temps consacrés aux actions qu'assure le canal.

Le temps passé effectivement est relatif et dépend de la charge du système et de la stratégie adoptée pour le traitement des signaux. De même, le temps consacré à l'activation du système de réception dépend des caractéristiques du système d'exploitation et des priorités attribuées aux processus, de la charge du système, des événements concomitants, etc. Il ne peut donc qu'être l'objet d'estimations et ne présente un intérêt que pour les études relatives à la capacité du système.

Une autre durée qu'il n'est pas possible de déterminer de façon statique est le temps passé dans les files d'attente en attendant l'absorption des signaux qui précèdent.

Normalement, il n'est pas facile de représenter statiquement le temps qui ne présente donc un intérêt que pour les simulations et les études relatives à la capacité des systèmes.

Cependant, à condition que toutes les actions qui ont lieu au niveau plus détaillé de la description du système comportent une indication du temps passé et que la description comporte une représentation des canaux (système d'exploitation) il est possible de simuler le système et d'en évaluer la capacité à l'aide de diverses techniques.

D.4.4 *Texte associé à des contructions en LDS*

Les conventions relatives au texte associé à des constructions en LDS, telles qu'elles sont décrites ci-après, s'appliquent au GR comme au PR.

Un texte en LDS peut être:

- formel
- informel
- un commentaire.

Un texte formel peut être interprété de façon formelle. Un texte informel ne peut être interprété que d'une manière informelle. Les commentaires ne sont que des annotations qui ne peuvent être interprétées. Un diagramme doit avoir la même signification avec ou sans commentaires.

D.4.4.1 *Texte formel*

On représente un texte formel à l'aide de la syntaxe LDS/PR. La syntaxe définit les combinaisons admises pour les textes en LDS/PR. En LDS/GR, le texte formel associé à une construction graphique doit être placé soit dans le symbole en question, soit dans un symbole d'extension de texte relié à la construction (voir le § D.6.3.6.19).

Dans un texte formel, rien n'empêche le mélange des lettres majuscules et des lettres minuscules. Il n'y a aucune différence entre un nom en lettres majuscules et un nom en lettres minuscules. Toutefois, il est recommandé d'employer les lettres majuscules et les lettres minuscules d'une manière cohérente.

D.4.4.1.1 *Nom*

Un nom représente l'identification de l'entité à laquelle il est associé dans le contexte. Il faut un nom pour les entités suivantes:

BLOC, PROCESSUS, SIGNAL, CANAL, ÉTAT, MISE EN RÉSERVE, ENTRÉE, SORTIE, MACRO, CRÉER, COMMENCER, PROCÉDURE et CONNECTEUR.

Tout nom doit débuter par une lettre et peut contenir des lettres, des chiffres, des caractères nationaux et des soulignements. A noter que les espacements ne sont pas autorisés.

Exemples de noms correctement formés (en anglais):

CALL_HANDLER
RINGING_TONE

Exemples de noms incorrectement formés (en anglais):

START CHARGING (il y a un espace)
1982 (il débute par un chiffre)

Le nom de l'ENTRÉE identifie le SIGNAL reçu; inversement, le nom de la SORTIE identifie le SIGNAL émis, de sorte que SORTIE-SIGNAL-ENTRÉE lient le même nom. Les connecteurs [JOINs (liaisons) en LDS/PR] ont un nom qui leur est associé. Ce nom peut débuter par un chiffre. Les connecteurs se distinguent également en ce qu'aucune chaîne de texte ne leur est associée.

Tous les mots-clés en LDS/PR sont réservés; on ne peut les employer comme noms. Le résumé consacré au LDS/PR donne tous les mots clés réservés.

D.4.4.1.2 *Paramètres formels*

On fait appel à des paramètres formels dans des énoncés ou dans les symboles de début de processus et de procédure.

Le mot clé «FPAR» désigne les paramètres formels.

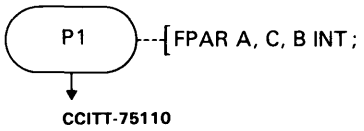


FIGURE D-47

Exemple de paramètres formels

Sur l'emploi des paramètres formels, voir le § D.4.3.5.

D.4.4.1.3 *Paramètres effectifs*

On emploie les paramètres effectifs dans des constructions telles que demande créer, appel de procédure, sortie, etc. On place les paramètres effectifs entre parenthèses.

OUTPUT chiffres (a, b, c, d)

FIGURE D-48

Exemple de paramètres effectifs dans une sortie

D.4.4.1.4 *Enoncés et expressions*

Le texte des tâches et des décisions respecte la syntaxe LDS/PR relative aux énoncés et aux expressions.

D.4.4.1.5 *Définitions et déclarations*

L'on représente toujours les définitions de signaux, de type de données et les déclarations de variables au moyen de la syntaxe LDS/PR pour ce genre de constructions.

D.4.4.2 *Texte informel*

On ne peut interpréter un texte informel associé à des symboles GR ou à des énoncés PR que d'une manière informelle. Les textes informels font appel à la syntaxe LDS/PR destinée aux chaînes de texte, c'est-à-dire que le texte est placé entre apostrophes.

Exemples de chaînes de texte (en anglais):

'Is subscriber free'

'$a^2 + 2ab + b^2$'

A noter que n'importe quel caractère peut figurer dans la chaîne à l'exception de '(apostrophe). En cas de nécessité ' peut être représenté par deux " consécutives.

Exemple:

John's house. – 'John"s house'

On peut omettre les apostrophes si la représentation en LDS emploie uniquement un texte informel. Dans ce cas, il convient d'en faire mention au début de la représentation en LDS.

Un texte informel peut revêtir toute forme convenant à un transfert d'information. Dans ce cas, l'auteur part de l'hypothèse qu'il existe une base de compréhension commune entre lui-même et le lecteur. A noter que le lecteur peut être une machine. Dans ce sens, un texte informel doit être suffisamment formel du point de vue de ce que comprend le lecteur afin qu'il n'existe aucune équivoque (figure D-49).

Exemples de textes informels:

- TASK 'faire avec x<5.....'
- INPUT x 'contient identité_d'abonné'
- DECISION 'abonné occupé?'
- (occupé):

FIGURE D-49

Exemples de textes informels

Un texte CHILL associé à des énoncés en LDS est un exemple de texte «informel» d'un point de vue LDS, bien qu'il s'agisse d'une manière très formelle de représenter des opérations.

Il existe une différence très nette entre un texte informel et des commentaires.

Le rôle des commentaires est d'aider les lecteurs à comprendre la représentation. Leur absence ou leur présence ne change rien à la représentation.

Le texte informel fait partie de la représentation elle-même; on ne peut la comprendre sans examiner le texte qui lui est associé (formel et informel).

D.4.4.3 *Commentaires*

Les commentaires n'ont en LDS aucune signification formelle ou informelle. Leur rôle est d'aider les lecteurs à mieux comprendre le diagramme GR ou le texte PR.

On peut insérer des commentaires à n'importe quel endroit d'un diagramme LDS/GR. On reconnaît les commentaires au fait qu'ils sont associés à un symbole de commentaire ou placés entre les signes '/*' et '*/' (voir la figure D-50).

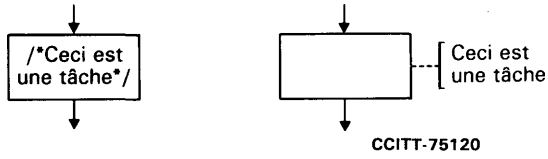


FIGURE D-50

Exemples de commentaires en GR

Les commentaires sont identifiés par le mot clé «COMMENT» (commentaire), ou placés entre les signes '/*' et '*/'.

```
TASK A:=2 COMMENT ceci est une tâche;
ou
TASK A:=2 /* ceci est une tâche */;
```

FIGURE D-51

Exemples de commentaires en PR

D.4.5 *Situations non définies*

D.4.5.1 *Considérations générales*

Pendant la vie utile des systèmes – de leur mise en service jusqu'à leur retrait – des défaillances se produisent. Certaines de ces défaillances (et pratiquement la plupart d'entre elles dans le cas du logiciel) sont dues à des situations non définies dans les spécifications. Il importe donc de vérifier les spécifications en LDS pour en déceler le plus possible avant de les traduire dans la pratique. Une fois qu'une spécification a été mise en œuvre, et qu'on en a établi la description, il convient de s'assurer que la description rend compte du système à 100% ni plus, ni moins.

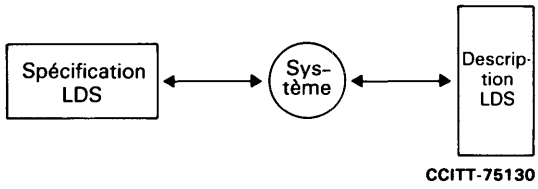


FIGURE D-52

Corrélation biunivoque entre spécification, système et description

D.4.5.2 *Exemples de situations non définies dans le LDS*

D.4.5.2.1 Un objet de données qui essaie d'affecter une valeur en dehors de son domaine ou une valeur qui ne correspond pas à son type. Par exemple, la variable N° DE L'ABONNÉ qui possède le type INTEGER RANGE (100000 : 999999) essaie d'assigner la valeur FAUX, 1 000 000, 87 ou OCCUPÉ.

D.4.5.2.2 Un signal est émis vers un processus qui n'existe pas.

D.4.5.2.3 Un signal émis vers une instance de processus (récemment) disparue. Il se peut cependant que l'erreur ne soit pas imputable au système, mais qu'il existe dans les spécifications ou descriptions du LDS une situation non définie à laquelle le système sous-jacent ou le programme de simulation doit remédier.

D.4.5.2.4 Un signal envoyé vers un canal qui n'existe pas.

D.4.5.2.5 Une erreur dans le comportement du système physique apparaîtra également dans la description du LDS s'il y a corrélation biunivoque.

D.4.5.2.6 Envoi de signaux vers un canal ou un processus sans que les signaux, le canal ou le processus aient été déclarés.

D.4.5.2.7 Une décision pour laquelle il n'existe pas de réponse ou de données quand le processus essaie de l'exécuter.

D.4.5.2.8 Une tentative d'exemplifier un processus pour lequel le nombre maximal d'exemplaires a déjà été atteint.

D.4.6 *Directives sur les données primitives en LDS*

Le § D.4.7 traite du concept de données avancées.

D.4.6.1 *Considérations générales*

En LDS, la communication entre processus est assurée par l'emploi de signaux, de valeurs partagées et de valeurs exportées ou importées.

Un processus émet un signal sous forme de sortie. Ce signal représente une circulation de données véhiculant des informations destinées à un autre processus. Si le processus de réception reconnaît ce signal au moyen d'une entrée, les données qui lui sont associées deviennent disponibles pour le processus.

Un processus peut lire la valeur d'une donnée détenue par un autre processus s'il appartient au même bloc que ce dernier et si la donnée en question a été déclarée valeur partagée.

Un processus peut en outre communiquer un élément de donnée en le déclarant exportable, tous les autres processus déclarés habilités à importer cet élément reçoivent sur demande communication de la valeur de cette donnée.

Si l'on considère un processus donné à un moment donné, il existe deux grandes catégories de données, qui sont complémentaires:

1) les données dont le processus peut disposer;
2) les données dont le processus ne peut pas disposer. (Dans cette seconde catégorie, entrent les données conservées mais dont il ne peut pas encore disposer et qui se présentent sous forme d'un signal en cours d'arrivée.)

Lorsqu'il exécute une quelconque des actions dont il est chargé: décision, tâche ou sortie, un processus ne peut utiliser que les données dont il dispose.

Quand elle sert à une décision ou à une sortie, la donnée proprement dite, à ce niveau d'abstraction, n'est pas modifiée. En revanche, une tâche spéciale peut donner lieu à la création, au stockage, à la modification ou à la destruction de données.

On trouvera des exemples de traitement des données à la figure D-53 (pour une décision), à la figure D-54 (pour une sortie) et à la figure D-55 (pour une tâche).

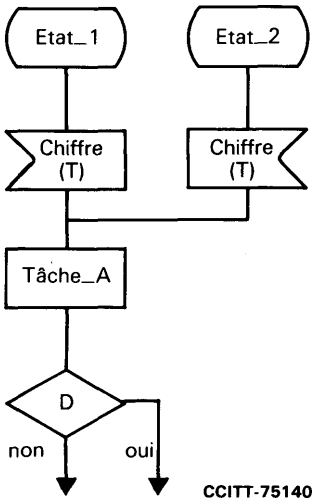
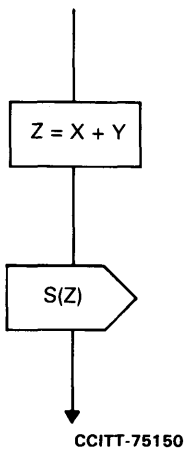


FIGURE D-53

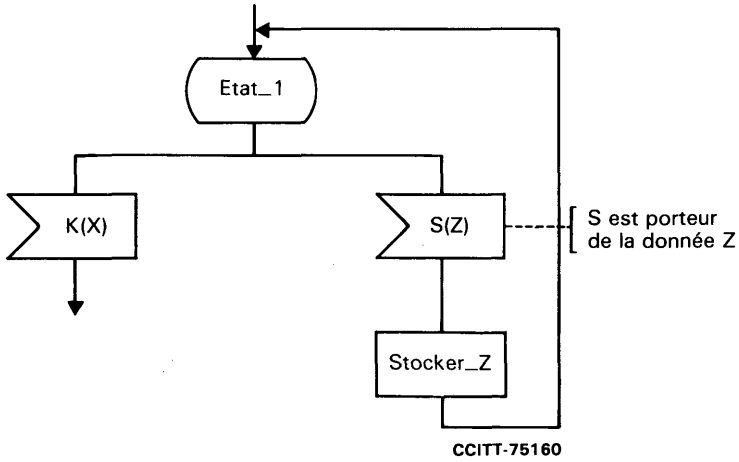
Interrogation des données associées à des entrées au moyen d'une décision



Remarque — S est le nom du signal; Z est la donnée associée.

FIGURE D-54

Emission de données au moyen d'une sortie



Remarque — S et K sont les noms des signaux; Z et X sont les données associées.

FIGURE D-55

Stockage des données d'entrée au moyen d'une tâche en vue d'une utilisation future

Les concepts de valeurs partagées et de valeurs importées ou exportées peuvent remplacer l'usage de signaux aux fins d'échange d'information.

D.4.6.2 *Traitement des données à l'intérieur d'un processus*

Il existe trois possibilités de communiquer entre processus: usage de SIGNAUX, de VALEURS PARTAGÉES, de VALEURS IMPORTÉES ou EXPORTÉES. Ces trois techniques sont très différentes. Par exemple, l'IMPORTATEUR de certaines données élémentaires ne peut pas communiquer lesdites données élémentaires en tant que VALEURS PARTAGÉES.

Les données relèvent d'une instance de processus; il s'ensuit que toutes les données ont une instance de processus détentrice et une seule.

L'instance de processus détentrice de la donnée peut changer la valeur qu'a cette dernière à un moment donné.

Exemple: si les éléments de données a et d sont des nombres entiers locaux au processus P, les exemples de séquences d'actions donnés à la figure D-56 sont permis à l'intérieur de P.

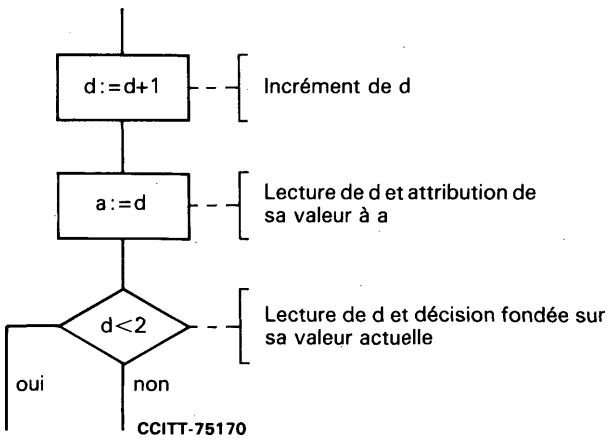


FIGURE D-56

Exemple d'actions sur les données locales d'un processus

D.4.6.3 *Traitement des données entre processus*

Deux processus peuvent échanger des données autrement qu'au moyen de signaux. Tout processus peut lire la valeur d'une donnée d'un autre processus s'il appartient au même bloc et si cette donnée est déclarée «valeur partagée» (voir le § D.4.6.3.1). Tout processus peut en outre lire la valeur d'une donnée d'un autre processus si cette donnée est déclarée «valeur exportable» (voir le § D.4.6.3.2). Le processus qui n'est pas détenteur de la donnée ne peut en modifier la valeur mais peut en faire une copie locale qui peut être manipulée comme tous les autres éléments de données locales.

Un élément de donnée ne peut être à la fois importé et partagé.

D.4.6.3.1 *Lecture des valeurs partagées*

Un processus peut communiquer tout ou partie des valeurs de ses données en les déclarant «valeurs partagées»; il s'ensuit que tous les autres processus (qui appartiennent au même bloc que le processus qui communique) peuvent lire la valeur partagée de la même manière que si cette valeur était une valeur locale. Ainsi, la valeur affichée sur un processus récepteur est toujours la même que celle qui est affichée sur le processus émetteur.

Certaines limites sont imposées à la communication des données.

D.4.6.3.2 *Lecture des valeurs exportables*

Un processus peut déclarer que tout ou partie des données qu'il détient sont des «données exportables»; il s'ensuit que tous les autres processus qui ont la «définition données importables» correspondante ont le droit d'obtenir copie de la valeur au moment où ils la demandent. La copie de la valeur de la donnée communiquée sur demande à un processus importateur reste donc constante, même si la valeur réelle de la donnée est ultérieurement modifiée par le détenteur qui l'a exportée.

D.4.6.3.3 *Exemple de différences entre l'emploi des valeurs partagées et celui des valeurs exportables*

Dans l'exemple de la figure D-57, il n'y a qu'une seule instance de processus 1. S'il peut y en avoir plusieurs, il faut prévoir un moyen d'identifier uniquement l'élément de donnée d correspondant.

D.4.6.3.4 *Valeurs partagées*

D.4.6.3.4.1 *Considérations générales*

L'usager du LDS constatera que la définition de données partagées fournit un moyen commode de spécifier la communication entre deux processus. Toutefois, l'application de systèmes ainsi *spécifiés* soulève un certain nombre de problèmes et le présent paragraphe a été rédigé afin de guider les usagers en sorte qu'ils puissent éviter ou résoudre ces problèmes. La *description* de systèmes en LDS qui comportent des données partagées devrait être moins difficile, car les problèmes auront été résolus dans la réalisation et il devrait être possible de transcrire la solution retenue en LDS.

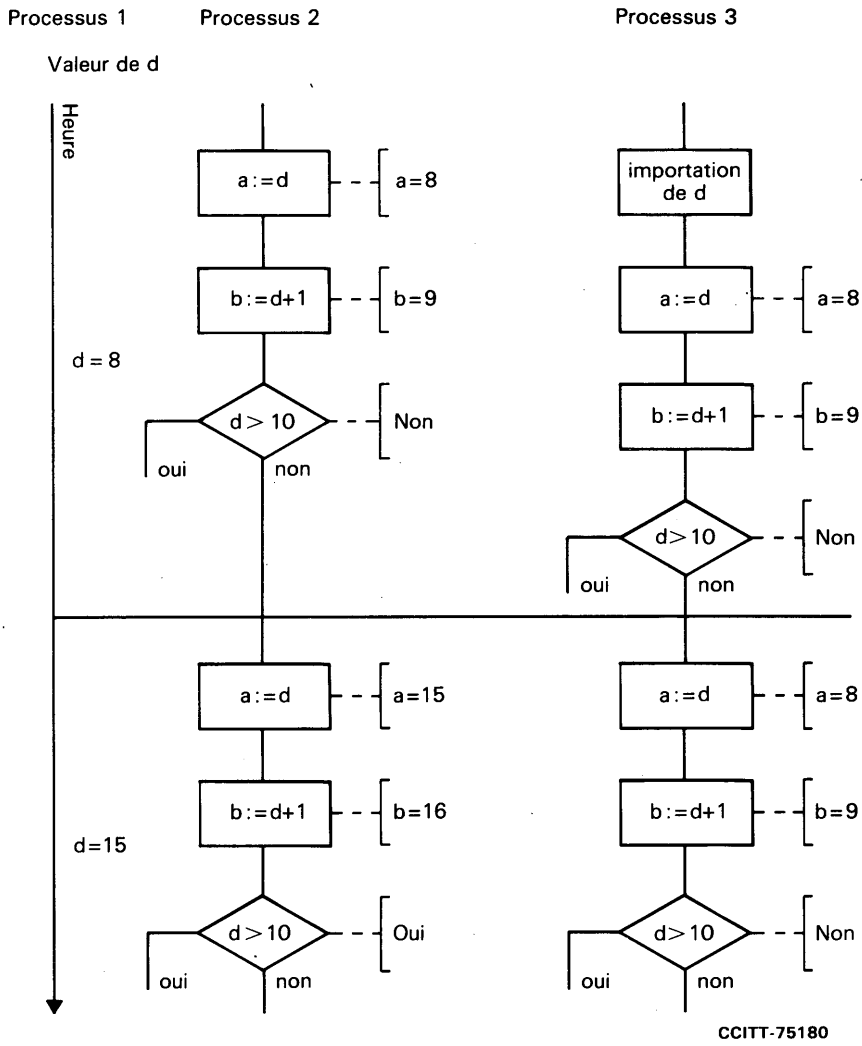
Dans le reste du présent paragraphe, on a admis qu'un processus A détient des données qu'il communique à un processus B, lequel en prend connaissance.

D.4.6.3.4.2 *Problèmes de création*

Toute tentative qui a pour effet de faire apparaître des données dans un processus avant la création du processus et des données constitue une erreur de LDS. L'usager peut éviter ce problème de deux manières:

- soit en veillant à ce que l'instance de processus A qui communique les données soit créée et qu'elle ait commencé à émettre les données pertinentes avant l'instance de processus B qui en prend connaissance;
- soit en veillant à ce que B n'introduise pas de transitions faisant intervenir des données partagées avant que A n'ait été créé et ait commencé à émettre les données correspondantes.

Dans le premier cas, un moyen simple d'obtenir le résultat recherché consiste à faire de A l'ascendant (ou un ancêtre) de B ou de faire en sorte que A soit créé en même temps que le système (création implicite). Dans le second cas, on peut faire en sorte que la transition pertinente en B ne puisse être déclenchée que par un signal émis par A.



Remarque — Le processus 1 est le processus détenteur de d. d est partagée avec le processus 2 et rendue exportable. d se trouve dans le processus 3 en tant que valeur importable.

FIGURE D-57

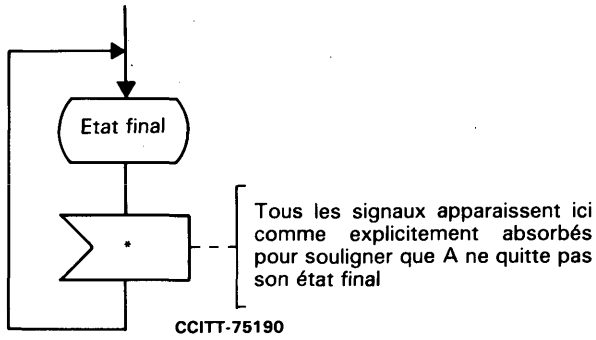
Exemple de différences entre l'emploi de valeurs partagées et celui des valeurs exportables

D.4.6.3.4.3 *Problèmes d'arrêt des processus*

Le processus A ne peut plus communiquer de données une fois qu'il a atteint un symbole d'arrêt. Toute tentative pour prendre connaissance de données constituerait alors une erreur de LDS. L'usage peut éviter cette difficulté de deux manières:

- soit en n'utilisant aucun symbole d'arrêt dans A. Dans la figure D-58, le processus A est indiqué dans un état latent, les données partagées étant visibles, mais ne pouvant plus être modifiées;
- soit en faisant en sorte que B sache que A est sur le point d'arrêter et ne tente plus de prendre connaissance des données. Un exemple où interviennent des signaux est donné à la figure D-59.

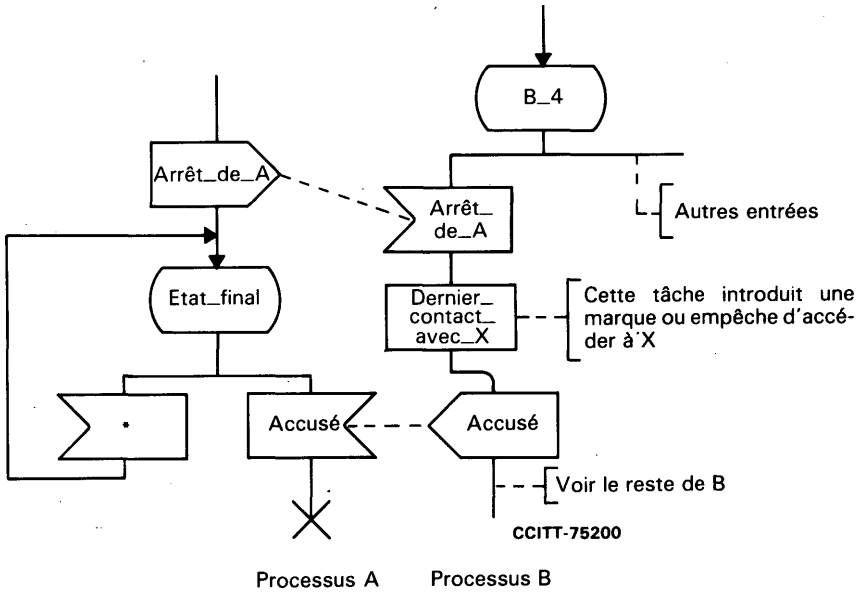
La première solution présente pour l'opérateur, l'inconvénient que A n'a pas libéré les données stockées qu'il utilisait.



Remarque — Le processus A contient la donnée X qui est communiquée.

FIGURE D-58

Processus comportant un état latent qui reste toujours capable de communiquer une donnée mais qui ne peut plus la modifier



Remarque — Le processus A contient la donnée X qu'il communique au processus B.

FIGURE D-59

Le processus A notifie B et obtient son accord avant de s'arrêter

D.4.6.3.4.4 *Problèmes dus à de multiples instances de B*

Si dans le processus B plusieurs instances sont simultanément en activité, par exemple B1, B2, B3 ... toutes pouvant accéder aux données de A, les solutions aux problèmes précédemment exposés dans le § D.4.6.3.4.2 doivent être légèrement modifiées.

Le meilleur moyen d'empêcher qu'une instance de B tente d'accéder à A avant que A n'existe et ait commencé à initialiser les données pertinentes est que A soit l'ascendant de tous les B.

Inversement, si A doit envoyer des signaux à tous les B, il ne peut le faire qu'après que l'ascendant de B lui ait notifié l'existence et l'identificateur de processus de chaque B.

Il faut apporter également une légère modification à la solution exposée au § D.4.6.3.4.3 pour arrêter le processus. De toute évidence, la solution qui correspond à l'état latent de la figure D-58 est toujours valable.

La solution de la figure D-59 n'est plus satisfaisante. On peut alors concevoir une solution d'après les indications suivantes:

a) A existe avant l'ascendant de B;

b) A est informé par l'ascendant de B qui émet un signal chaque fois qu'un nouveau B est créé;

c) A est informé par chaque B qui émet un signal quand ce dernier va s'arrêter ou, du moins, quand il ne pourra plus prendre connaissance des données de A;

d) A tient le compte des B en activité qui ont accès à lui ou peuvent y accéder;

e) A envoie un signal invitant l'ascendant de B à suspendre la création d'autres B et reçoit un accusé l'informant qu'il sera donné suite à son invitation;

f) quand le nombre des B capables d'accéder à B atteint zéro, A peut s'arrêter.

Cette solution est illustrée à la figure D-60 qui décrit le processus A.

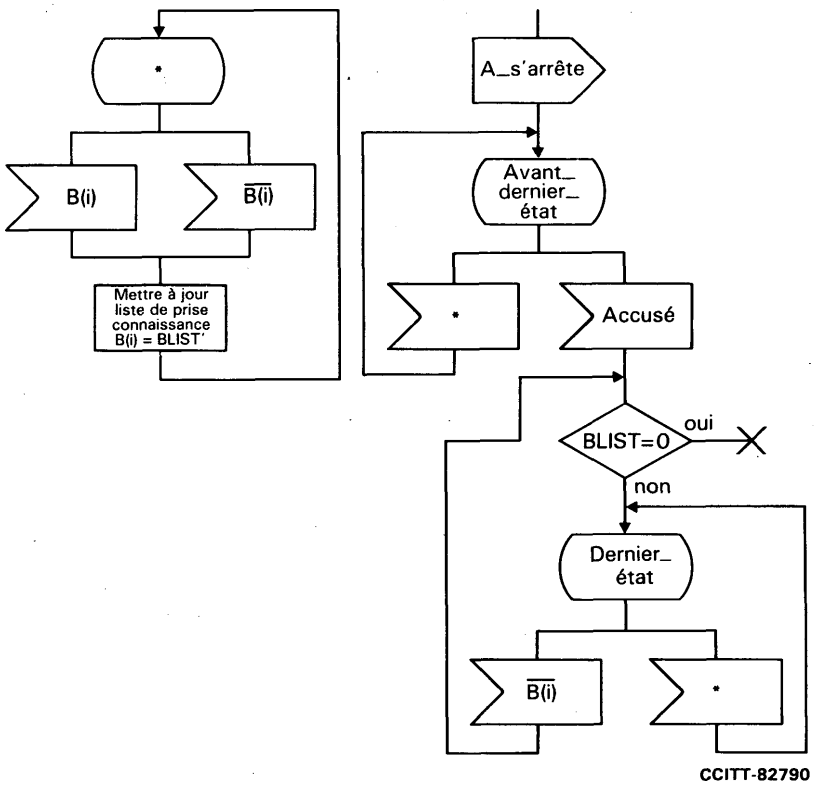


FIGURE D-60

Deux diagrammes du processus A

D.4.6.3.4.5 *Problèmes dus à des instances multiples de A*

Si dans le processus A, plusieurs instances sont simultanément en activité, par exemple A1, A2 et que B veut accéder aux données, B doit alors préciser l'instance A qui détient les données qui l'intéressent. Il s'ensuit que A doit connaître l'identité de l'instance A en question, et être certain que l'instance A aux données de laquelle il veut accéder existe bien.

On peut alors concevoir une solution dans le sens indiqué ci-après:

a) l'ascendant de A envoie un signal au B qui contient l'identificateur de processus d'un de ses descendants (par exemple A6) qui détient les données dont B veut prendre connaissance;

b) B prend connaissance des données;

c) A6 envoie un signal à B pour l'informer que A6 est sur le point d'arrêter et qu'il ne doit plus chercher à prendre connaissance des données qu'il détient;

d) B accuse réception en répondant à A6 par un signal;

e) A6 s'arrête.

D.4.6.3.4.6 *Problèmes dus à des instances multiples de A et de B*

Dans la pratique, le cas le plus fréquemment rencontré d'instances multiples et de processus A et de processus B se présente quand la relation de partage des données a lieu entre paires d'instances de A et d'instances de B. Les problèmes exposés aux § D.4.6.3.4.2 et D.4.6.3.4.3 peuvent être évités de la manière suivante:

a) dès qu'une nouvelle instance de A, par exemple Ai a été créée, elle crée une nouvelle instance de B, par exemple Bj. Ai sait alors que l'identité de processus de Bj est un descendant et Bj sait que Ai est son ascendant;

b) Bj prend connaissance des données;

c) quand Bj a fini de prendre connaissance des données pour la dernière fois, il envoie un signal à son ascendant Ai et s'arrête (ou continue avec d'autres transmissions);

d) Ai s'arrête.

D.4.6.3.4.7 *Partage de plusieurs éléments de données*

Considérons le cas où A détient plusieurs éléments de données, par exemple X1, X2 et X3 auxquels B doit avoir accès. Pour plus de simplicité, supposons qu'il existe une seule instance du processus A et une seule du processus B.

Si A actualise X1, X2 et X3 alors que B tente d'en prendre connaissance, il peut arriver que B obtienne un mélange de valeurs nouvelles et anciennes, c'est-à-dire un jeu incohérent.

Dans certains systèmes, il est plus important d'avoir des valeurs cohérentes que d'avoir des valeurs à jour et dans ce cas, il convient d'appliquer des moyens permettant d'assurer la cohérence. Nous envisageons ici deux possibilités:

a) au lieu de partager X1, X2 et X3 nous définissons une structure composite X qui a X1, X2, X3 comme champs et/ou sous-structures.

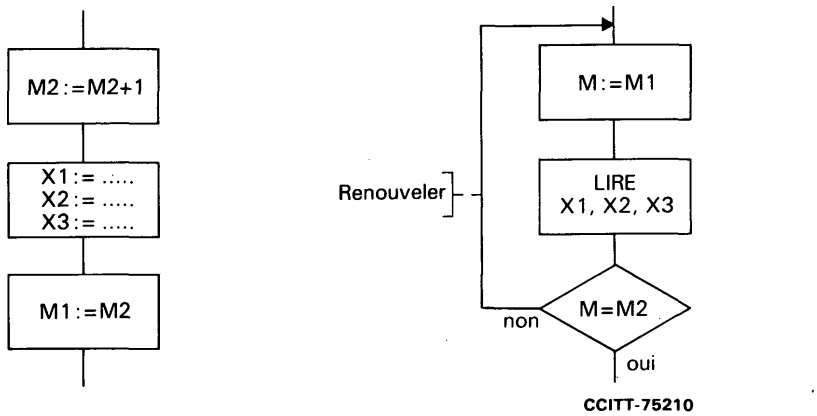
X est communiqué à A et B en prend connaissance.

A obtient alors des valeurs actualisées pour X1, X2, X3 et, quand toutes ces valeurs sont actualisées, affecte tous les nouveaux sous-champs à X dans une seule et même affectation (étant donné que le LDS considère cette affectation comme indissociable).

Ainsi, il n'est jamais possible de prendre connaissance d'un ensemble incohérent.

Le problème n'a cependant pas été vraiment résolu; il est simplement répercuté sur l'implémenteur étant donné que, dans la plupart des applications, les affectations ne sont atomiques que si les champs sont en un mot de machine.

b) pour résoudre le problème tel qu'il est posé, il faut spécifier une certaine forme de verrou dans le LDS. Un exemple simple de cas sans blocage est donné à la figure D-61 qui nécessite l'introduction de deux marqueurs M1 et M2.



a) *Processus A*
Mise à jour de X1, X2, X3.
L'emploi de tâches séparées fait ressortir l'importance de la séquence, d'abord M2, puis les X, puis M1.

b) *Processus B*
Lecture de X1, X2, X3.
Nouvelle tentative s'il apparaît que M1 est différent de M2.

Remarque — Le processus A permet au processus B de prendre connaissance de M1, X1, X2, X3 et M2. M1 et M2 sont du type NOMBRE ENTIER et réglés au départ sur zéro.

FIGURE D-61

Verrou assurant des valeurs cohérentes

Les marqueurs doivent être réalisés de façon à pouvoir les modifier de manière atomique (c'est-à-dire les présenter en un seul mot).

L'implémenteur doit examiner la fréquence et le temps pris pour une mise à jour et rechercher la probabilité de tentatives répétées et s'assurer qu'il obtient les résultats voulus.

D.4.6.3.4.8 *Conclusions*

Dans les paragraphes qui précèdent on a soulevé quelques problèmes que pose la spécification de systèmes où l'on a recours à des données partagées et donné quelques exemples de solutions permettant de les éviter. Les problèmes et les solutions proposés n'épuisent pas le sujet et les usagers pourront trouver d'autres solutions pour éviter les problèmes ainsi que d'autres problèmes qu'il faut chercher à éviter. Quoi qu'il en soit, il est souhaitable de retenir des solutions qui répondent à d'autres caractéristiques du système envisagé, de façon à en réduire la complexité. Il est conseillé aux usagers d'étudier avec soin le comportement des systèmes qui ont recours à des données partagées.

Enfin, il vaut peut-être la peine de souligner que l'on peut éviter la plupart de ces problèmes en faisant appel à des signaux pour envoyer les données.

Dans le LDS, les valeurs des données sont considérées comme «indivisibles» c'est-à-dire qu'un objet de données a telle ou telle valeur dans le domaine correspondant. Quand on implémente des objets de données, l'implémentation ne peut modifier qu'une partie d'un objet complexe à la fois. Si de tels objets sont partagés, l'implémenteur doit prévoir des mécanismes assurant la cohérence nécessaire (par exemple, verrous, etc.).

Remarque – Les données exportables permettent normalement d'éviter cette difficulté vu que le processus détenteur ne pourra pas répondre à une demande quand la donnée est en cours de modification (et qu'elle risque d'être incohérente). La valeur d'une donnée se modifie complètement en une seule transition et, comme le processus ne peut être interrompu que pendant un état, la donnée sera cohérente. L'«exclusion mutuelle» est ainsi assurée.

D.4.7 *Directives sur les données avancées en LDS*

D.4.7.1 *Définition des types de données*

D.4.7.1.1 *Considérations générales*

Le LDS comporte un certain nombre de types de données prédéfinis. Les utilisateurs peuvent toutefois en définir de nouveaux, qui peuvent être employés lors de la déclaration de variables, de signaux, etc., de la même façon que les données de type prédéfini.

Comme pour la plupart des concepts du LDS, le mécanisme de définition des types peut être employé à différents niveaux d'abstraction, depuis les définitions informelles de la syntaxe de base jusqu'à la définition tout à fait formelle de la syntaxe et de la sémantique des types.

Il est possible de «paramétrer» les types, ce qui permet d'obtenir des «blocs de construction» à partir desquels on peut créer une catégorie de types encore plus étendue. On nomme ces derniers des générateurs de types et, comme les processus LDS, ils doivent être instanciés afin de permettre la création d'un ou de plusieurs types similaires aux propriétés légèrement différentes. Le type tableau (Array) prédéfini en est un exemple, étant donné qu'elle ne livre son «contenu» que pendant son utilisation.

Le dispositif permettant de définir de nouveaux types de données doit être utilisé chaque fois qu'il est essentiel d'obtenir une spécification extrêmement rigoureuse et dépourvue de toute ambiguïté. La seule mention du nom d'un type n'est souvent pas suffisante pour permettre à tous les lecteurs d'une spécification de déterminer les propriétés exactes du type en question, même lorsque les lecteurs y sont habitués. Dans ces cas, il convient de faire appel aux concepts de données avancées en LDS.

D.4.7.1.2 *Introduction aux types abstraits*

Tous les types de données en LDS sont définis comme des types abstraits. Un type définit un ensemble de valeurs et un ensemble d'opérations applicables à ces valeurs. Par exemple, l'ensemble de valeurs pour le type booléen sont Vrai et Faux, et les opérations sont les opérations booléennes traditionnelles, Non, Et, Ou, etc. La théorie moderne des langages accorde une large place à un concept de type abstrait d'apparence pourtant restrictive; ce concept permet à l'usager de définir tout type de donnée demandé. Les structures des données des langages de programmation traditionnels sont un simple sous-ensemble des types abstraits. On peut définir le comportement des opérations d'une façon informelle, à l'aide de textes de «commentaires», ou bien au moyen d'un ensemble formel d'axiomes.

Cette méthode permet la définition de types tout à fait indépendamment de leur implémentation; ainsi l'emploi de types de données en LDS n'influe pas sur le choix final des techniques d'implémentation.

D.4.7.1.3 *Définitions de données*

L'on distingue trois catégories de définitions de données: les définitions de types de données, de générateurs de types de données et de synonymes.

D.4.7.1.3.1 *Définitions de synonymes*

Les définitions de synonymes permettent de conférer à un nom la valeur donnée par une expression (dépendante du contexte). Pour obtenir un «typage fort», il faut pouvoir déterminer le type de l'expression sans aucune ambiguïté; dans le cas contraire, le type doit être explicitement indiqué [par exemple, l'expression «4» est ambiguë étant donné qu'elle peut être du type réel (Real) ou du type entier (Integer)].

Les définitions de synonymes servent habituellement d'abréviation en cas d'utilisation répétée d'une expression; elles offrent un point centralisé permettant de modifier facilement une constante souvent accédée.

Exemple:

```
SYN Single Integer = 1
SYN BlockSize = 512 * BitsPerByte
SYN Legs = (4 * Dogs) + (2 * Birds)
```

D.4.7.1.3.2 *Définition des types de données*

Le mécanisme de définition des types de données représente la principale technique permettant de définir formellement de nouveaux types en LDS. Il existe deux possibilités, le *newtype* et le *syntype.*

Le type *syntype* définit un type qui représente un sous-ensemble des valeurs d'un type quelconque déjà existant (que l'on nomme le type ascendant), généralement employé comme une «restriction d'intervalle» d'un langage de programmation afin de permettre un contrôle plus étroit des valeurs possibles d'une variable.

Ceci permet une «équivalence structurelle» entre différents types; ainsi les types sont compatibles si leur structure sous-jacente (type ascendant) est la même malgré la différence de nom.

Exemple:

```
SYNTYPE Digit = Integer
    CONSTANTS 0:9
END Digit

SYNTYPE WarmColours = Colours
    CONSTANTS Yellow, Orange, Red
END WarmColours
```

Un type *newtype* introduit un type de données entièrement nouveau qui peut se baser ou non sur un type existant. De nouvelles constantes et de nouveaux noms d'opérateurs ne peuvent être introduits qu'au moyen de définitions *newtype.*

Les définitions *newtype* assurent l'«équivalence des noms» entre différents types; ainsi on considère que les variables sont du même type seulement si leurs types déclarés portent textuellement le même nom.

Un *newtype* peut être défini indépendamment de tous les autres types, comme l'extension d'un type existant ou l'instantiation d'un générateur de type ou du générateur de type prédéfini proposé pour STRUCT. Dans les deux premiers cas, la possibilité de définir les propriétés du type telles que des axiomes formels est une caractéristique importante. Ces axiomes facultatifs définissent la sémantique des opérations pour ce type en indiquant quels effets peut entraîner la combinaison des opérations. Ainsi toutes les opérations sont définies en fonction des autres. La définition de tous les types abstraits se fonde sur le type booléen, qui donne un noyau à l'ensemble des axiomes pour tous les types (bien qu'un type donné ne fasse pas nécessairement apparaître le booléen dans ses axiomes, un examen récurrent des types révèle que la définition de tous les types se fonde sur le booléen).

L'élaboration de ces axiomes exige une certaine attention de la part de l'usager du LDS qui doit éviter deux pièges importants: l'incomplétude et l'incohérence. Ces deux défauts peuvent rendre une spécification LDS ambiguë. Dans le premier cas, il se peut que l'ensemble d'axiomes soit insuffisant pour caractériser complètement le type en cours de définition. Dans le second cas, les axiomes se contredisent les uns les autres, ce qui invalide évidemment la spécification. Malheureusement c'est l'usager du LDS qui doit se charger de la détection de ces problèmes: en effet, le LDS ne comporte pas de techniques automatiques permettant de déterminer si des définitions de type algébrique sont incomplètes ou incohérentes; l'usager doit donc se baser sur une évaluation intuitive d'un ensemble donné d'axiomes.

D.4.7.1.3.3 *Générateur de types de données*

D'une manière informelle on peut dire qu'un générateur de types de données est un «gabarit» à partir duquel il est possible de créer un nombre indéfini de nouveaux types. On remplace textuellement les paramètres du générateur dans une copie textuelle du générateur initial, et le nouveau générateur obtenu peut être considéré comme faisant partie de la définition du type dans laquelle se produit l'instantiation du générateur.

Ceci évite évidemment une répétition superflue des définitions de types en cas de besoin d'un certain nombre de types identiques, excepté une partie du type qui n'affecte pas les axiomes de base.

D.4.7.2 *Exemples de définitions de données*

Quelques exemples de définitions de données sont présentés ci-après. Ils ne sont pas exhaustifs; ils ne visent qu'à donner un avant-goût de la puissance du mécanisme de définition de données adopté pour le LDS.

Outre des définitions de type, l'on trouvera des exemples de déclarations d'objets de données des types en question et des opérations qui leur sont applicables.

Conformément à la forme LDS employée, les déclarations et les opérations seront inscrites (forme GR) ou non (forme PR) dans des symboles graphiques, tout en conservant cependant la même syntaxe.

D.4.7.2.1 *Exemple 1: définition d'un compteur d'abonné*

Ceci est une définition possible d'un compteur d'abonné. Le compteur d'abonné se définit en utilisant les propriétés générales de type entier naturel et restreint l'ensemble des opérations admissibles à la seule addition.

DÉFINITION

```
NEWTYPE SUB_METER
INHERITS NATURAL ("+")
END SUB_METER
```

EXEMPLE D'UTILISATION

```
DCL A_METER SUB_METER;
.
.
.
A_METER : = A_METER +10
```

D.4.7.2.2 *Exemple 2: définition d'un type générique «tableau à deux dimensions»*

Ceci est la définition d'un tableau à deux dimensions. La définition est générale; elle ne donne aucune limite concernant les intervalles des deux indices ainsi que les composants du tableau. Ainsi, il est possible d'instancier ce type à partir de n'importe quel type d'intervalle et de composante.

Trois opérations sont admises (l'opération d'affectation est implicite): déclarer (DECLARE), insérer (INSERT) et extraire (EXTRACT).

Il est possible, de déclarer de nouveaux objets de ce même type, d'insérer une nouvelle valeur dans une position choisie du tableau et d'extraire une valeur d'une position choisie du tableau.

Conformément à la syntaxe de la définition des données de chaque opérateur autorisé, les domaines et codomaines de l'opérateur sont décrits dans la partie de la définition relative aux OPÉRATEURS (OPERATORS)[1]. La partie réservée aux AXIOMES (AXIOMS) présente la sémantique employée par les opérateurs.

Chaque fois qu'un nom d'opérateur est suivi d'un point d'exclamation (ils le sont tous dans le présent cas), ceci indique que ce nom est utilisé dans la définition et que la syntaxe concrète fait appel à un autre nom lorsqu'elle emploie des objets de ce type.

---

[1] Lorsqu'aucun type de résultat n'est indiqué (après la flèche), le résultat est une modification des paramètres qui sont «primés».

DÉFINITION

```
GENERATOR BIDI_ARRAY (TYPE index1, TYPE index2, TYPE A_TYPE);
    OPERATORS
        INSERT! : BIDI_ARRAY', index1, index2, A_type → ;
        EXTRACT! : BIDI_ARRAY, index1, index2 → A_type;
    AXIOMS
        EXTRACT!(DECLARE!(v), I1,I2)=Error!;
        EXTRACT!(INSERT!(A,Ip1, Ip2, E1),Ipr,Ipc)=
                    if Ipr=Ip1 and Ipc=Ip2 then E1 else
                    Extract!(A,Ipr,Ipc)fi;
END BIDI_ARRAY
```

EXEMPLE D'UTILISATION

```
SYNTYPE INDEX = INTEGER
    CONSTANTS 1:20
END INDEX
NEWTYPE
    ARRAY2 BIDI_ARRAY (INDEX; INDEX; SUB_METER);
END ARRAY2
.
.
DCL DOUB_COL ARRAY2;
DCL A SUB_METER;
.
.
.
.
A :=DOUB_COL(5,16)
.
.
.
DOUB_COL(5,16):=A
.
.
```

D.4.7.2.3 *Exemple 3: définition d'un type bit*

Le nouveau type bit se caractérise par le fait qu'il a 0 et 1 pour seules valeurs. On introduit l'opérateur de permutation (FLIP) pour mettre à 1 ou à 0 la valeur d'un bit (le codomaine est absent) et toutes les autres opérations (Not, And, Or, "+") restituent une valeur.

DÉFINITION

```
NEWTYPE bit
    Literals (0,1);
    OPERATORS
        NOT : bit → bit;
        AND : bit,bit → bit;
        OR : bit,bit → bit;
        "+": bit,bit → bit;
        FLIP : bit' → ;
AXIOMS
    Flip (0)'=1;
    Flip (1)'=0;
    Not (1)=0;
    Not (0)=1;
    And (A,B)=if A=0 then 0 else B fi;
    Or (A,B)=if A=1 then 1 else B fi;
    "+" (A,B)=if A=0 then B else if B=1 then 0 else 1 fi;
END bit
```

EXEMPLE D'UTILISATION

```
DCL CNTRL_BIT BIT;
DCL TMP_BIT, DUM_BIT BIT;
.
.
.
.
TMP_BIT :=NOT(CNTRL_BIT)
DUM_BIT :=OR(TMP_BIT, CNTRL_BIT)
.
.
.
.
FLIP(CNTRL_BIT)
.
.
.
```

D.4.7.2.4 *Exemple 4: définition d'un type octet*

On définit un type octet comme un tableau de bits (newtype bitarray); en tant que tableau, il possède toutes les propriétés des tableaux. Les autres opérations admises sont Shr, Shl, And-Mask produisant respectivement un décalage gauche de n positions ($0 \leqslant n \leqslant 7$), le décalage à droite de n positions ($0 \leqslant n \leqslant 7$) et l'opération ET avec un autre octet.

DÉFINITION

```
NEWTYPE bitarray (integer, bit) END bitarray;

NEWTYPE byte INHERITS bitarray ALL
    ADDING OPERATORS
                    shr : byte', integer → ;
                    shl : byte', integer → ;
                    And-Mask : byte', byte → ;
    AXIOMS
                    extract!(shr(B,1)',J)=    if J<O or J>7 then Error!
                                              else if J=7 then 0 else extract!
                                              (B,J+1)' fi
                                              fi;
                    extract!(Shl(B,1)',J)=    if J<O or J>7 then Error!
                                              else if J=O then 0
                                              else extract (B,J-1)!
                                              fi
                                              fi;
                    shr(B,n)'=  if n<0 or n>7 then Error!
                                else if n=0 then B
                                else shr(shr(B,1)',n-1)'
                                fi
                                fi;
                    shl(B,n)'=  if n<0 or n>7 then Error!
                                else if n=0 then B
                                else shl (shl(B,1)',n-1)'
                                fi
                                fi;

FOR ALL i IN INTEGER (extract!(And-Mask(B,MB)', i)
                            = IF i<0 or i>7 THEN error!
                              ELSE
                              and (extract!(B,i),extract!(MB,i)) FI;
                              /* Quand une opération ET est réalisée entre deux octets, chaque bit de
                              l'octet résultant est égal au résultat de l'opération ET entre des bits correspondants des octets originaux */

end byte
```

EXEMPLE D'UTILISATION

```
DCL F_BYTE, S_BYTE, T_BYTE BYTE;
DCL TMP_BIT BIT;
.
.
.
shr (F_BYTE, Z)
.
.
shl (S_BYTE, 3)
.
.
.
And-Mask (F_BYTE, S_BYTE)
.
.
.
TMP_BIT := F_BYTE (7)
.
.
```

D.4.7.2.5 *Exemple 5: définition simplifiée d'un type octet*

Les définitions de données LDS permettent aux usagers de définir des types de données d'une façon «informelle». Cette ressource peut s'avérer très utile, notamment lorsque la définition du système exige l'emploi d'une méthode progressive d'affinage. L'exemple suivant donne une définition «informelle» du même type octet que dans l'exemple 4. (Remarque – On présume que le type tableau de bits est défini.)

DÉFINITION

```
NEWTYPE byte INHERITS bitarray ALL
        ADDING OPERATORS
          shr : byte', integer → ;
          shl : byte', integer → ;
          And-Mask : byte', byte → ;

    /* Shr produit un décalage à droite de n positions, avec 0⩽n⩽7 */

    /* Shl produit un décalage à gauche de n positions, avec 0⩽n⩽7 */

    /* And-Mask produit l'opération ET entre les bits correspondants des deux différents octets; le résultat est
       rangé dans le premier opérande */

end byte
```

D.4.7.2.6 *Exemple 6: définition d'un abonné*

Ceci est un exemple sommaire de définition d'un abonné. Jusqu'à ce niveau, un abonné est une structure (un enregistrement dans de nombreux langages de programmation, ou d'une façon plus formelle un regroupement de plusieurs champs, pouvant tous être de type différent et être sélectionnés en cas de besoin); cette structure comprend un certain nombre de champs (le numéro téléphonique de l'abonné) et un champ d'état (l'état de l'abonné). La définition de ces deux champs a été donnée précédemment.

Les seules opérations admises sur un objet abonné sont connecter (connect) et déconnecter (disconnect); elles placent respectivement l'état d'un abonné en position occupée (busy) ou libre (free).

DÉFINITION

```
NEWTYPE digitstring string(digit) END digitstring;
NEWTYPE status_id LITERALS(free, busy) END status_id;

NEWTYPE subscriber
    STRUCT    number digitstring;
              status status_id;

    OPERATORS
              connect : subscriber', subscriber" → ;
              disconnect : subscriber', subscriber" → ;
```

```
    AXIOMS
            S=connect(S1,S2)' => s1(status)=busy;
            S=connect(S1,S2)" => s2(status)=busy;
            S=disconnect(S1,S2)' => s1(status)=free;
            S=disconnect(S1,S2)" => s2(status)=free;

end subscriber
```

EXEMPLE D'UTILISATION

```
DCL SUB_EX, CALD_SUB SUBSCRIBER;
DCL NUM DIGITSTRING;
.
.
.
SUB_EX (STATUS) := FREE
.
.
.
NUM := SUB_EX (NUMBER)
.
.
.
CONNECT (SUB_EX, CALD_SUB)
.
.
.
```

D.4.7.2.7-8 *Exemples 7 et 8: définition plus détaillée d'un abonné*

On trouvera ci-après deux autres exemples de types de données définis de façon informelle. Ils se rapportent à la possibilité de structuration. Chaque nouveau type doit comprendre l'ancien type parmi ses champs. Cette caractéristique permet à l'usager de se concentrer sur des aspects particuliers du type de données et de différer l'étude des autres aspects à un niveau plus détaillé.

DÉFINITION

```
NEWTYPE subscriber_rev1
    STRUCT   first_app subscriber;
             counter sub_meter;

END subscriber_rev1
```

La présente définition ajoute la fonction de comptage à un abonné, désigné ici par subscriber_REV1.

```
NEWTYPE subscriber_REV2
    STRUCT   old_sub subscriber_rev1;
             enabling Three_cond;

END subscriber_rev2
```

Cette définition ajoute à un abonné des conditions de validation (validation du champ); la définition du type Three_cond est la suivante:

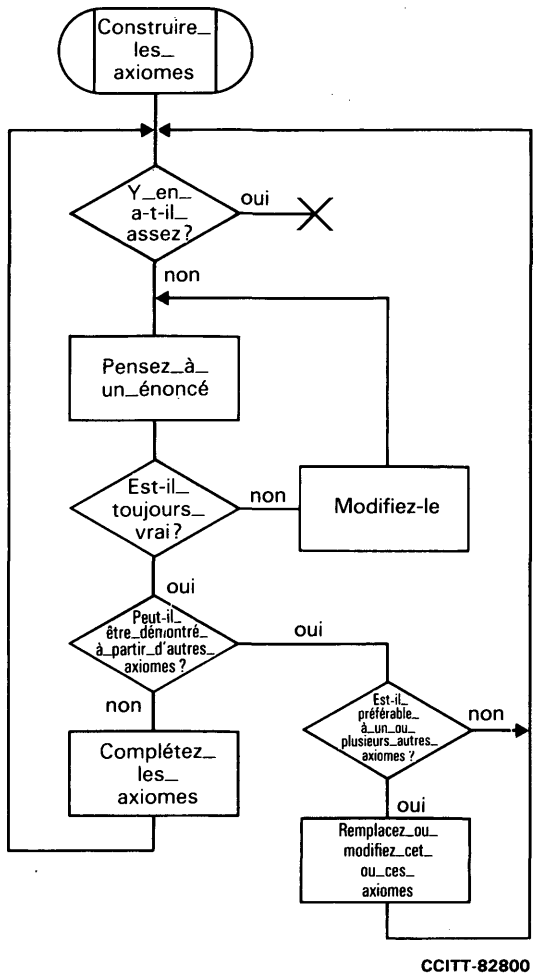
```
NEWTYPE Three-cond LITERALS (LOCAL_AB, LONG_AB, INT_AB) END Three-cond;
```

D.4.7.2.9 *Exemple 9: procédure de construction d'axiomes pour un NEWTYPE*

Lors de l'introduction de nouveaux types de données, il est indispensable d'établir suffisamment d'axiomes, ce qui suppose:

- que chaque opérateur apparaisse au moins une fois dans l'ensemble d'axiomes;
- que la véracité des énoncés (autres que les axiomes) soit vérifiable;
- qu'aucune incohérence ne puisse être détectée,

afin de répondre aux besoins du nouveau type de données.



D.5 *Documentation*

D.5.1 *Qu'est-ce qu'un document?*

L'ISO définit un document comme étant «une quantité limitée et cohérente d'informations stockées sur un support et que l'on peut rechercher». On peut donc le considérer comme une unité logique strictement délimitée. Les documents servent à véhiculer toutes les informations relatives à un système spécifié ou décrit en LDS.

Lorsque le papier sert de support physique au stockage d'un document, on applique souvent improprement le terme «document» aux feuilles de papier plutôt qu'à leur contenu logique. L'emploi des supports de stockage magnétiques se répandant, ce terme retrouve petit à petit sa signification initiale.

D.5.2 *Introduction*

Le présent § D.5 traite des aspects relatifs à la documentation. Il aborde plutôt l'organisation logique des documents que leur organisation physique. Ce dernier point est laissé à la discrétion de l'usager. Du fait de la ressemblance entre les conditions exigées par l'organisation logique et par l'organisation physique des documents, les conseils contenus dans le texte ci-après peuvent utilement aider un usager à mettre en place une organisation physique de documents.

D.5.3 *Pourquoi avons-nous besoin de documents?*

Si l'on fractionne les rapports entre l'information et un système en plusieurs documents, le système devient plus facilement lisible et exploitable. Etant donné que le LDS ne possède pas de concept formel de document, l'usager doit faire appel à un mécanisme informel, ou en emprunter un à un autre langage (CHILL de préférence). Il n'est pas nécessaire de fractionner le système en documents distincts dans le seul cas où l'on spécifie ou décrit un système complet sous la forme d'une seule chaîne de texte en PR.

D.5.4 *Structure du document*

On peut considérer l'ensemble des documents qui englobent la définition du système comme un ensemble de structures.

Lorsque des documents renvoient à des sous-documents, la structure d'un document est attachée à chaque bloc de la structure du système, y compris le niveau du système. Voir la figure D-62.

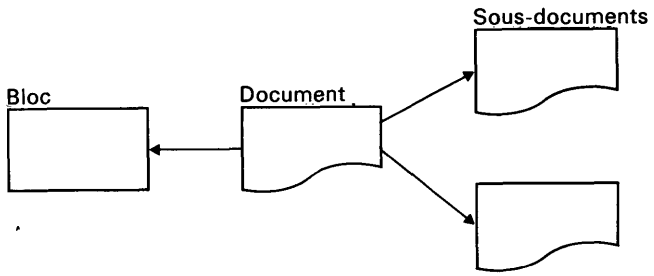


FIGURE D-62 CCITT-75220

Structure d'un document

D.5.5 *Mécanisme de repérage*

Tous les documents doivent être reliés les uns aux autres d'une façon précise. Ceci est d'autant plus nécessaire que l'actualisation de l'un d'entre eux peut exercer un impact sur les autres documents.

Chaque document doit avoir une identification unique et sa frontière doit être clairement spécifiée. L'on peut indiquer cette frontière en donnant les numéros des feuillets qui font partie du même document. Lorsque le document ne comporte qu'une seule page, comme, par exemple, un diagramme d'interaction de blocs, la frontière peut être indiquée par le symbole de trame.

Un même document peut servir de sous-document à plusieurs blocs du même système.

Dans le § D.9 intitulé «Exemples», l'on trouvera deux méthodes de repérage. L'une se fonde sur une syntaxe PR, l'autre fait appel au mécanisme de commentaire utilisé dans les diagrammes GR. Voir le § D.9 pour des renseignements plus détaillés.

D.5.6 *Types de documents*

Les différents types de diagrammes de la syntaxe GR sont généralement des documents si l'on s'en tient à la définition précédemment donnée. Ces documents doivent également renseigner sur leurs propres frontières. Comme précédemment indiqué, ils doivent former une structure de documents lorsqu'ils englobent ensemble toutes les définitions introduites dans la définition d'un bloc. Dans cette structure, il convient de faire figurer au moins les définitions des processus dans des documents distincts. En cas d'emploi d'un diagramme d'interaction de blocs, le même document donne également les définitions des canaux. Il est préférable de regrouper dans des documents distincts les définitions des signaux et des données.

De même, les définitions de procédures et les macrodéfinitions doivent constituer des sous-documents du document de processus.

On peut faire figurer les définitions des signaux qui constituent une liste de signaux dans des documents distincts portant le titre: Liste de signaux X, Y, Z, etc. La figure D-63 ci-dessous présente une structure de document possible pour un bloc.

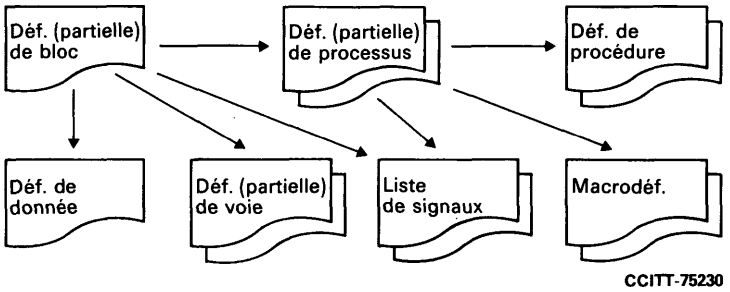


FIGURE D-63

Structure de document possible pour un bloc

D.5.7 *Mélange de GR et de PR*

Il est possible de définir en partie le processus au moyen de la forme GR. Le diagramme du processus (en GR) ou le corps du processus (en PR) englobe le graphe du processus. La définition du graphe du processus doit former un document indépendant afin qu'il soit possible de mélanger la définition du graphe du processus en GR avec l'autre partie en PR.

On peut aussi définir partiellement un système ou un bloc sous la forme GR. Le diagramme d'interaction des blocs doit alors constituer un document. Etant donné que le diagramme d'interaction des blocs n'englobe que des parties de la définition d'un système ou d'un bloc, le reste doit être placé dans des documents distincts.

Lors du mélange de GR et de PR, la représentation en PR doit être fractionnée en plusieurs documents. Malheureusement la syntaxe en PR ne le permet pas; le fractionnement doit donc se produire de façon informelle.

Le § D.7 traite des méthodes de formation syntaxique des parties qui assurent une connexion dans une représentation en PR.

D.6 *Directives pour la représentation de systèmes utilisant le LDS/GR*

Le § D.6 des Directives pour les usagers du LDS explique l'emploi de la syntaxe LDS fondée sur la représentation graphique, le LDS/GR.

D.6.1 *Pourquoi une syntaxe graphique?*

L'avantage d'un langage graphique est qu'il donne une représentation claire de la structure d'un système et qu'il permet de visualiser facilement l'ordre d'exécution. Employé à des fins de conception, le LDS doit prendre une forme qui permette à l'usager d'exprimer ses idées d'une façon claire et concise. La syntaxe graphique permet cela; elle correspond plus à la représentation traditionnelle des machines à états finis étendues, alors que la représentation textuelle en phrases est mieux adaptée à l'emploi en machine.

La syntaxe avec représentation graphique du LDS (LDS/GR) est la forme initiale du LDS. Imaginée entre 1973 et 1976, elle a été publiée pour la première fois dans l'édition de 1976 de la série de Recommandations Z.100.

La syntaxe avec représentation graphique s'est inspirée des langages graphiques conçus par divers organismes à des fins d'usage interne.

D.6.2 *Diagrammes LDS/GR*

Le présent paragraphe explique les diagrammes suivants:

Le diagramme d'interaction des blocs décrit la structure interne d'un système ou d'un bloc

Le diagramme d'arborescence de blocs décrit la subdivision d'un système en blocs et en sous-blocs

Le diagramme d'arborescence de processus décrit la subdivision d'un processus en sous-processus

Le diagramme de sous-structure de canal définit la structure interne d'un *canal*

Le diagramme de définition de macro définit un ensemble de symboles. Chaque référence à la définition de macro doit céder la place à la définition avant d'effectuer l'interprétation

Le diagramme de procédure définit le comportement de l'exécution d'une *procédure*

Le diagramme de processus définit le comportement de toutes les *instances de processus* de la *définition de processus en question.*

Les définitions de signaux et de données sont définies en utilisant la syntaxe du LDS/PR (voir le § D.7.3.2) et sont référencées à partir des diagrammes dans lesquels l'information est utilisée.

D.6.2.1 *Directives générales*

Les directives générales qui s'appliquent à l'élaboration de diagrammes graphiques sont les suivantes:

- les diagrammes doivent être élaborés de sorte à avoir le sens de lecture suivant: de haut en bas et de gauche à droite;
- les diagrammes ne doivent pas comporter trop de détails; il est souvent judicieux de subdiviser de grands diagrammes en sous-diagrammes consacrés aux différentes parties ou aux différents aspects;
- il est préférable qu'un segment connecté d'un diagramme puisse tenir sur une page;
- on peut insérer les commentaires n'importe où sur un diagramme, soit en employant un symbole de commentaire en GR, soit en faisant appel à la syntaxe de commentaire en PR;
- il est préférable de placer un texte dans le symbole auquel le texte est associé. Si cela entraîne des problèmes d'ordre pratique, on peut placer le texte dans un symbole d'extension de texte relié au symbole concerné.

D.6.2.2 *Repérage*

L'actuelle série de Recommandations relatives au LDS ne prévoit pas de moyens permettant d'établir des références entre les diagrammes en GR eux-mêmes, entre les définitions en LDS/PR ou entre les diagrammes en GR et les définitions en PR. La présence d'un mécanisme de repérage est nécessaire à la fois pour obtenir une représentation complète d'un système en LDS et pour structurer la documentation d'un système.

Il existe plusieurs façons de concevoir un mécanisme de repérage, en fonction de l'organisation de la documentation et des méthodes retenues. Etant donné que les Recommandations ne prévoient aucun mécanisme, les usagers ont toute latitude quant au choix d'un mécanisme.

Citons à titre d'exemple d'emploi de références avec la syntaxe en GR, l'utilisation de remarques ou de la syntaxe des commentaires, comme indiqué sur la figure D-64.

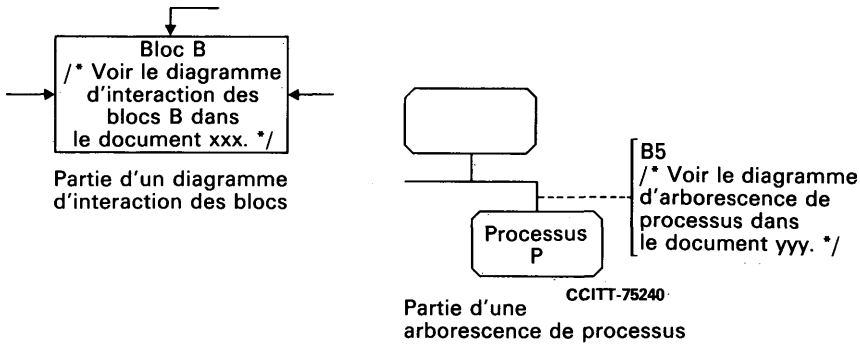


FIGURE D-64

Exemple d'utilisation de remarques ou de commentaires en guise de références

Plusieurs autres méthodes sont envisageables. Le § D.9 donne d'autres exemples.

D.6.2.3 *Gabarit*

On trouvera à l'intérieur de la couverture arrière du présent fascicule un gabarit permettant la représentation graphique des symboles du LDS. Une représentation schématique de ce gabarit est donnée dans la figure D-65.

Cette figure reproduit directement et en trois formats (20 × 40 mm, $20/\sqrt{2} \times 40/\sqrt{2}$ et 10 × 20 mm) les symboles suivants: ENTRÉE, SORTIE, DÉCISION, VARIANTE, PROCESSUS, DÉBUT, TÂCHE, ÉTAT, MISE EN RÉSERVE, CONNECTEUR et ARRÊT. Les cotes intérieures des symboles de grand format sont indiquées.

On peut réaliser les symboles correspondant à l'APPEL DE PROCÉDURE, MACRO, et CRÉER à partir du symbole TÂCHE en traçant la(les) ligne(s) horizontale(s) ou verticale(s) supplémentaire(s) indiquée(s).

On peut réaliser les symboles DÉFINITION DE PROCÉDURE et MACRO ENTRÉE à partir du symbole DÉBUT en traçant la(les) ligne(s) verticale(s) supplémentaire(s) indiquée(s).

On peut réaliser le symbole MACRO SORTIE à partir du symbole CONNECTEUR en traçant la ligne verticale supplémentaire indiquée.

Le symbole RETOUR résulte de la combinaison des symboles CONNECTEUR et PROCESSUS DE FIN.

On réalise le symbole COMMENTAIRE à l'aide de l'une quelconque des extrémités du symbole TÂCHE.

On peut dessiner le symbole CONDITION DE VALIDATION à l'aide du symbole ARRÊT.

Le «symbole» TOUT (astérisque) ne figure pas dans le gabarit car il doit être accompagné du texte associé aux symboles.

Une ligne pleine symbolise un canal ou la ligne qui aboutit au symbole d'EXTENSION DE TEXTE.

La ligne de signal et la ligne qui aboutit à un COMMENTAIRE sont symbolisées par une ligne pointillée dont le rapport est 1:1.

Les figures D-209 et D-210 reproduisent tous les symboles recommandés.

D.6.3 *Emploi du LDS/GR*

Pour la description de tous les symboles utilisés en LDS/GR, voir le résumé relatif à la syntaxe graphique (*annexe C* aux Recommandations Z.100-Z.104).

D.6.3.1 *Expression de la structure en LDS/GR*

Le présent paragraphe traite de la subdivision d'un système en blocs et en canaux ainsi que de la subdivision facultative des blocs et des canaux.

L'on notera qu'en cas de subdivision d'un système à plusieurs niveaux de détail, le système peut comporter des diagrammes de variante (voir la Recommandation Z.102). Lorsqu'il existe deux descriptions de variante, il convient de considérer la description la moins détaillée comme une vue d'ensemble.

D.6.3.1.1 *Diagramme d'interaction de bloc*

Le diagramme d'interaction de bloc représente les *canaux* et les *blocs* qui composent la structure interne d'un *système* ou d'un *bloc.*

Il peut représenter l'ensemble des *processus* que contiennent ces blocs, ainsi que les *processus* qui peuvent créer d'autres *processus.*

Le diagramme se compose des éléments suivants:

- symbole de cadre (obligatoire pour les blocs mais facultatif en cas d'emploi autour d'un système)
- symbole de bloc
- symbole de canal
- listes de signaux

et peut, à titre facultatif, comporter les éléments suivants:

- symbole de processus
- symbole de trajet des signaux
- symbole de demande
- symbole d'environnement.

La figure D-66 donne un exemple sommaire de la représentation d'un petit système (S) par un diagramme d'interaction des blocs. Ce système se compose de blocs (B1 et B2). On notera que le cadre de délimitation est facultatif pour un système.

D'après la convention de haut en bas/de gauche à droite, il est préférable de placer les noms dans l'angle supérieur gauche ou à côté de l'objet nommé. Les définitions qui ont fait l'objet d'un renvoi peuvent être placées à l'extérieur du diagramme, comme la liste de signaux L_1 dans l'exemple ci-dessus.

Les définitions de signaux et de données décrites en syntaxe PR, peuvent être placées dans le diagramme, de préférence dans l'angle supérieur gauche ou encore à l'extérieur du diagramme. Si les définitions sont placées à l'extérieur du diagramme et que leur localisation risque de poser des problèmes, le diagramme doit indiquer où elles se trouvent au moyen d'une référence.

Dans le symbole de bloc, on peut représenter les processus contenus, comme indiqué à la figure D-67.

Afin de rendre les diagrammes plus lisibles, on peut les subdiviser de sorte que l'intérieur des blocs soit représenté séparément. Par exemple, le segment que présente la figure D-67 peut, s'il est pris séparément, représenter un sous-diagramme du diagramme d'interaction des blocs.

Il est souvent utile de représenter plusieurs niveaux d'une structure dans le même diagramme d'interaction des blocs. Pour ce faire, l'on remplace dans un diagramme le symbole d'un bloc par le diagramme d'interaction de blocs correspondant à ce même bloc. La figure D-68 illustre ce cas.

Cette opération peut être répétée un nombre indéfini de fois.

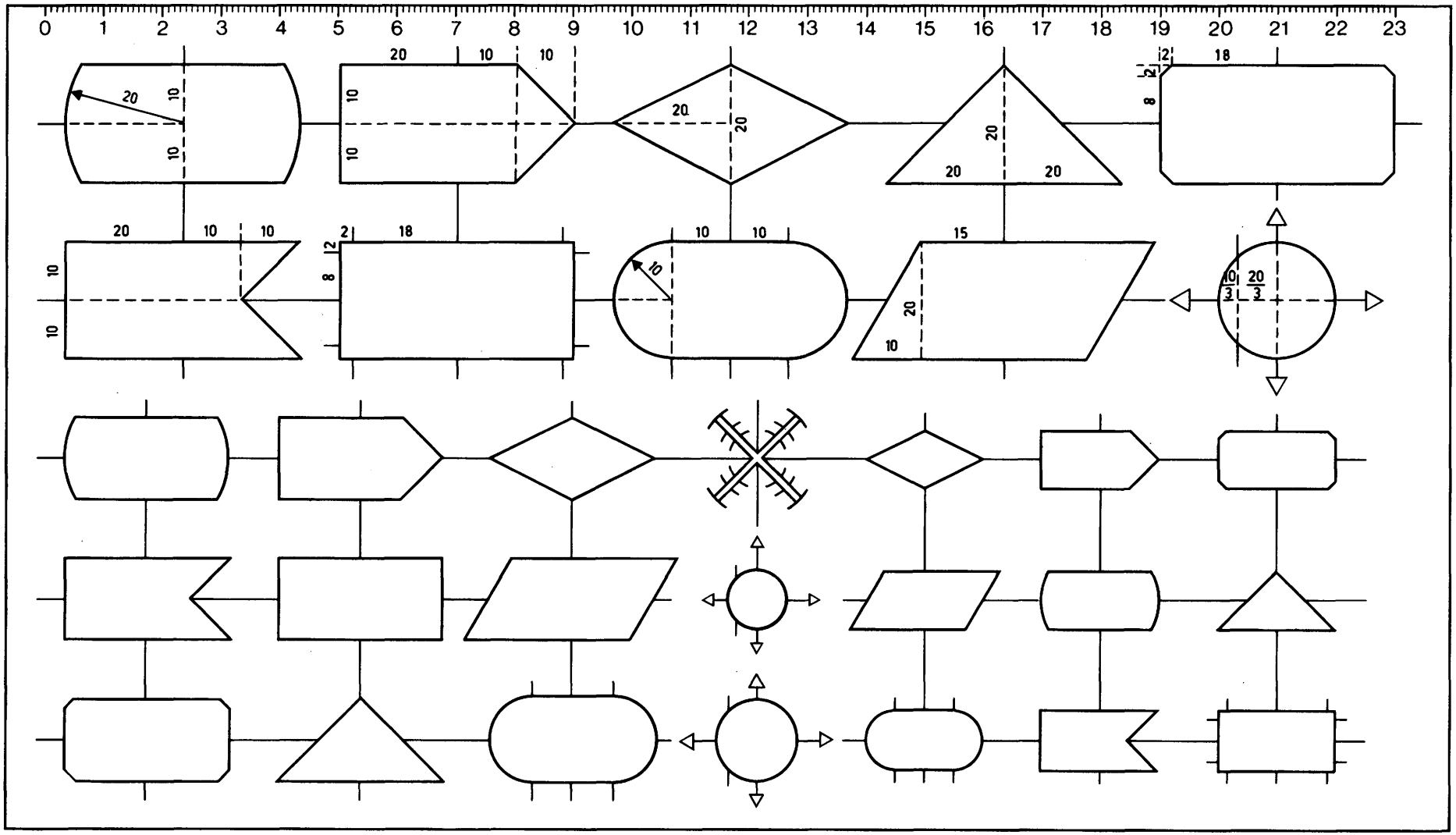
Afin d'obtenir un diagramme bien structuré et lisible, il convient de noter que:

- les diagrammes ne doivent pas être trop complexes afin de tenir sur une page;
- l'utilisation de la possibilité de subdivision du diagramme en sous-diagrammes réduit le nombre de détails et rend le diagramme plus lisible;
- on peut faire appel à une macro pour simplifier de vastes diagrammes.

D.6.3.1.2 *Diagramme d'arborescence de bloc*

Un diagramme d'arborescence de bloc représente les blocs et sous-blocs qui constituent la structure d'un système. Le diagramme vise à donner au lecteur une vue d'ensemble de la structure du système.

Le diagramme est une arborescence hiérarchique de symboles de blocs; il est «subdivisé» en lignes.



Remarque 1 – Rapport hauteur/largeur = 1 :2
Remarque 2 – On trouvera 3 dimensions : largeurs de 40 mm, 28 mm et 20 mm. 40 mm/$\sqrt{2}$ ≃ 28 mm/$\sqrt{2}$ ≃ 20 mm, etc.
Cela permet la photoréduction du format A3 au format A4 avec des symboles compatibles.

FIGURE D-65

Représentation schématique du gabarit LDS

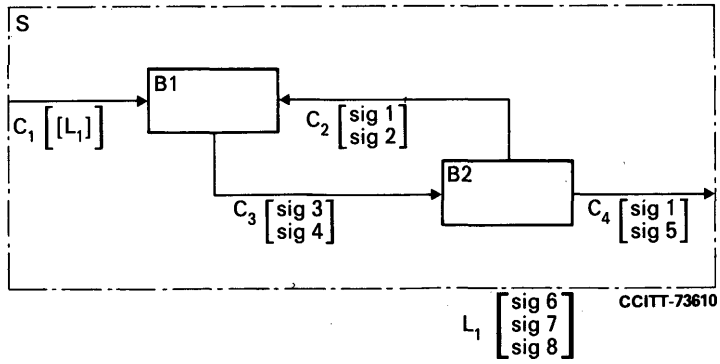


FIGURE D-66

Exemple de représentation d'un système par un diagramme d'interaction des blocs

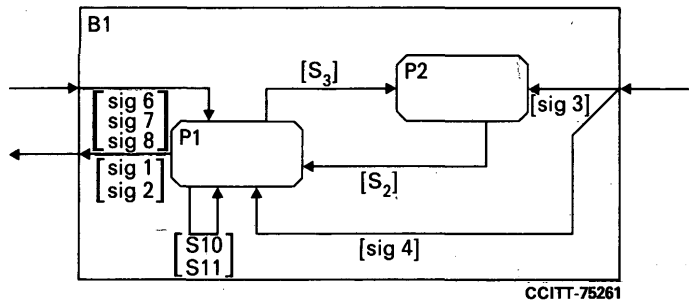


FIGURE D-67

Segment d'un diagramme d'interaction de bloc

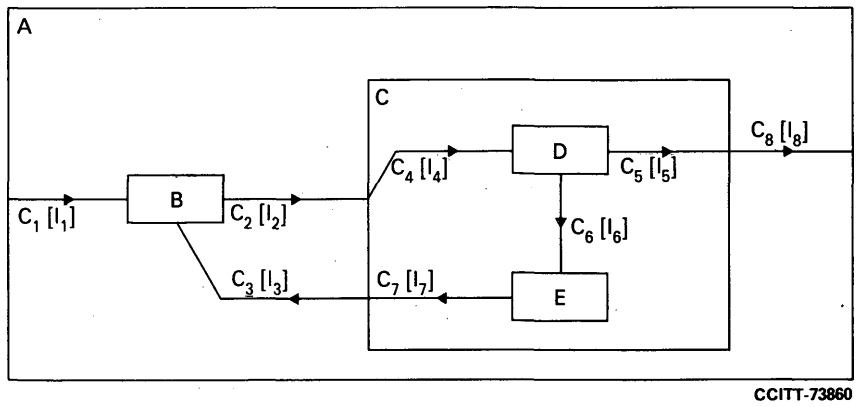


FIGURE D-68

Exemple de représentation à plusieurs niveaux d'une structure

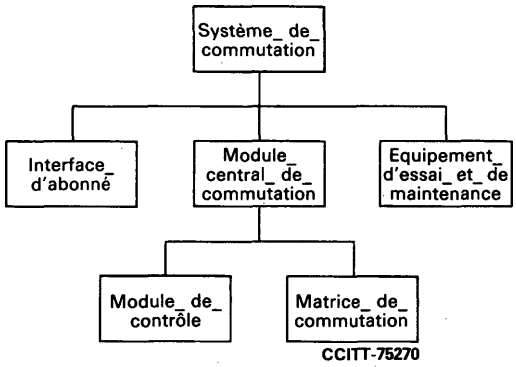


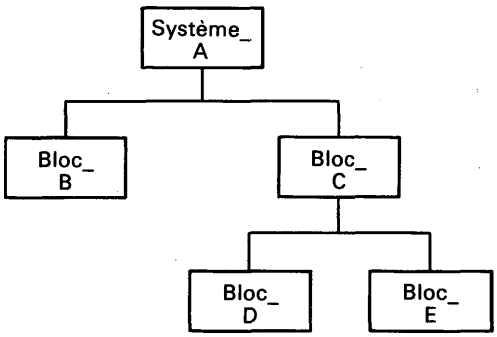
FIGURE D-69

Exemple de diagramme d'arborescence de bloc

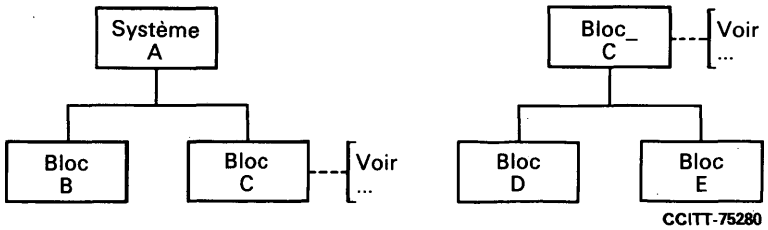
Il est préférable que tous les symboles de blocs de diagramme aient la même taille. Ainsi, des blocs qui sont à un même niveau de subdivision sont représentés au même niveau sur le diagramme. Si la taille du diagramme empêche ce dernier de tenir sur une page, il convient de le subdiviser en diagrammes d'arborescence de blocs «partiels» ainsi que le montre la figure D-70.

La subdivision d'un diagramme d'arborescence de blocs en diagrammes partiels est une opération souvent utile.

Le fractionnement en plusieurs diagrammes partiels s'opère de la façon suivante: le premier diagramme, qui a un système pour racine, est coupé de façon qu'un ensemble de blocs au stade suivant de subdivision n'apparaisse pas comme le résultat d'une subdivision. Les blocs au niveau desquels le diagramme initial a été coupé deviennent les racines de diagrammes qui représentent le stade suivant de subdivision.



a) Diagramme non subdivisé



b) Diagramme subdivisé

FIGURE D-70

Exemple de diagrammes partiels d'arborescence de bloc

Lorsqu'on utilise des diagrammes partiels, sans être certain qu'un bloc est lui-même subdivisé et/ou sans savoir où se trouvent les diagrammes suivants, il convient d'insérer des références au moyen du symbole de commentaire.

D.6.3.1.3 *Diagramme d'arborescence de processus*

Un diagramme d'arborescence de processus décrit la subdivision d'un processus en sous-processus et indique où ces processus sont affectés. Le diagramme est une arborescence hiérarchique de symboles de processus; il est «subdivisé» en lignes. On décrit l'affectation des processus au moyen du symbole de commentaire.

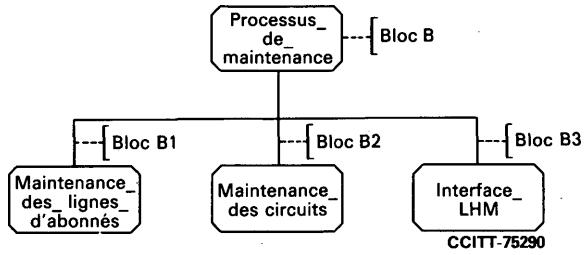
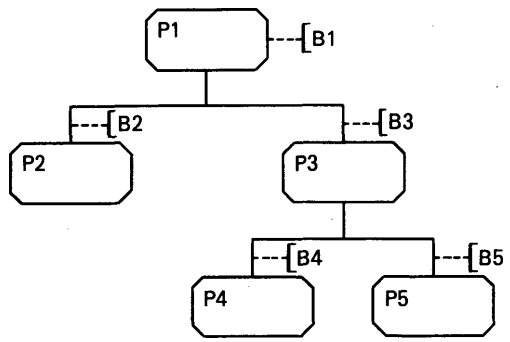


FIGURE D-71

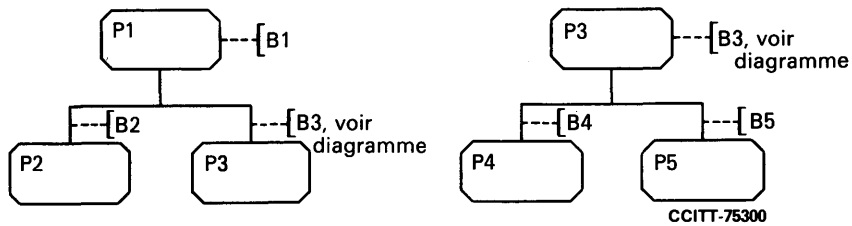
Exemple de diagramme d'arborescence de processus

Ajoutons que pour être complet, le processus qui sert de racine ne doit pas être le sous-processus d'un quelconque autre processus. De même, aucun des processus feuilles ne doit être à son tour subdivisé. Il est préférable que tous les symboles de processus du diagramme aient la même taille afin que les processus d'un même niveau de subdivision apparaissent dans le diagramme à un niveau homogène.

Pour des raisons de lisibilité, ou si, du fait de sa taille, le diagramme ne peut tenir sur une seule page, on peut le scinder en diagrammes partiels. On obtient un ensemble de diagrammes partiels faisant apparaître, comme s'ils n'étaient pas subdivisés, un ensemble de processus subdivisés dans le diagramme initial. Ces processus deviennent donc des racines dans d'autres diagrammes qui illustrent les stades suivants de subdivision.



a) Diagramme non subdivisé



b) Diagramme subdivisé

FIGURE D-72

Exemple de diagrammes partiels d'arborescence de processus

Si on emploie des diagrammes partiels sans être certain qu'un processus est lui-même subdivisé, ou sans savoir où se trouve le diagramme qui y fait suite, il convient d'insérer des références au moyen de symboles de commentaire.

D.6.3.1.4 *Diagramme de sous-structure de canal*

Un diagramme de sous-structure de canal décrit de quelle façon un canal est subdivisé en sous-composants. Ce diagramme ressemble à un diagramme d'interaction de blocs. Toutes les directives du § D.6.3.1.1 s'appliquent également au diagramme de sous-structure de canal.

Lorsqu'un canal est subdivisé dans un diagramme d'interaction de blocs, l'on doit trouver un renvoi au diagramme de sous-structure de canal qui décrit la subdivision. Voir les figures D-73 et D-74.

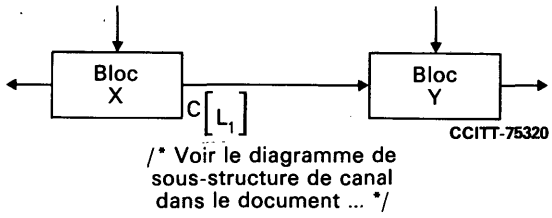


FIGURE D-73

Segment d'un diagramme d'interaction de blocs avec subdivision d'un canal

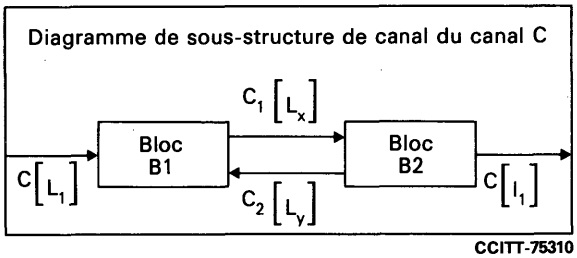


FIGURE D-74

Exemple de diagramme de sous-structure de canal

Lorsque l'on sait que le canal est subdivisé et que la localisation du diagramme de sous-structure ne pose aucun problème, la référence n'est pas indispensable.

D.6.3.2 *Définitions de signaux*

Il n'existe pas de syntaxe graphique spécifique concernant les définitions de signaux. Les diagrammes LDS/GR qui font appel aux définitions de signaux emploient la syntaxe LDS/PR (voir le § D.7.3.2). On peut insérer le texte en PR dans les diagrammes mais il est préférable de placer des renvois dans les diagrammes.

Les diagrammes qui peuvent renvoyer à des définitions de signaux sont:

- les diagrammes d'interaction des blocs
- les diagrammes de sous-structure de canaux.

Lorsque les définitions figurent dans les diagrammes, il est préférable qu'elles soient placées à l'angle supérieur droit de ce même diagramme, comme indiqué à la figure D-75.

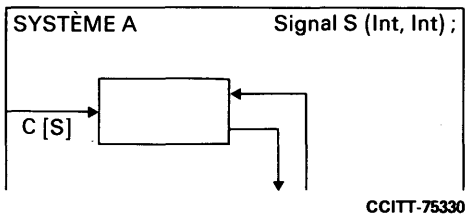


FIGURE D-75

Segment d'un diagramme d'interaction des blocs avec une définition de signal

Si un diagramme contient une quantité importante de textes en PR, il devient moins lisible.

Il est préférable de remplacer ce texte en PR par une référence à son emplacement, comme par exemple:

/* Pour les définitions des signaux, types et procédures, voir le document .....*/.

D.6.3.3 *Définitions de données*

Il n'existe aucune syntaxe graphique spécifique en LDS/GR concernant les définitions de données. En cas de besoin de définitions de données en GR, on fait appel à la syntaxe en PR (voir le § D.7.3.4).

Il existe deux catégories de définitions de données:

- définitions de types de données
- définitions de variables.

En ce qui concerne les définitions de signaux, il est préférable de placer dans les diagrammes en GR des références au texte en PR.

D.6.3.3.1 *Définitions de types de données*

On peut trouver des références à des définitions de types de données dans les diagrammes LDS/GR suivants:

- diagrammes d'interaction de blocs
- diagrammes de sous-structure de voies
- diagrammes de définition de macro
- diagrammes de processus
- diagrammes de procédure.

Les directives applicables à l'insertion de définitions de signaux s'appliquent aussi aux définitions de types de données.

D.6.3.3.2 *Définitions de variables*

Les diagrammes LDS/GR suivants peuvent comporter des définitions de variables:

- diagrammes de processus
- diagrammes de procédures.

Les directives applicables à l'insertion de définitions de signaux s'appliquent aussi aux définitions de variables.

D.6.3.4 *Diagrammes de définition de macro*

Un diagramme de définition de macro définit l'ensemble de symboles devant remplacer chacune des références de la macro en question avant l'interprétation.

Le diagramme peut comporter un ou plusieurs symboles d'entrée, suivi d'une ligne de liaison aboutissant au diagramme ainsi qu'un symbole de sortie suivant une ligne de liaison qui sort du diagramme. En outre, un diagramme de définition de macro peut comporter n'importe quel ensemble de symboles graphiques et de textes, quel qu'en soit l'arrangement. On ne peut déterminer la justesse ou la signification de la macro qu'après avoir effectué le remplacement.

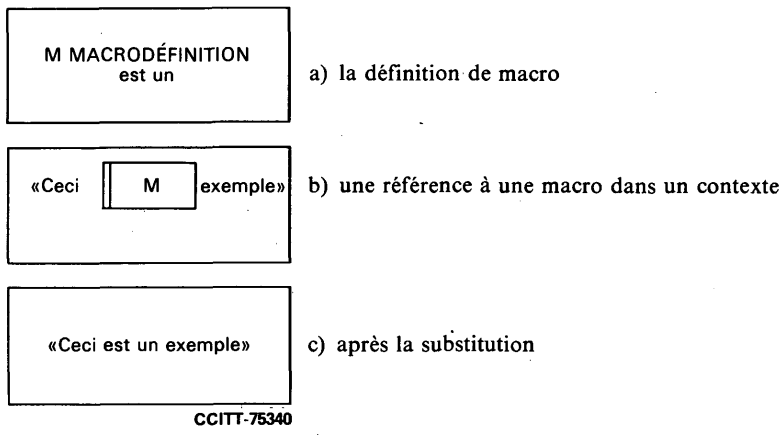


FIGURE D-76

Exemple d'emploi de la macro

Les directives générales pour les diagrammes bien structurés s'appliquent aussi aux diagrammes de définition de macro.

D.6.3.5 *Diagramme de procédure*

Le diagramme de procédure décrit le comportement à appliquer à chaque appel de cette procédure. Les procédures LDS sont semblables aux procédures CHILL et à celles d'autres langages de programmation. Elles visent à:

a) permettre de structurer à plusieurs niveaux de précision un *graphe de processus;*

b) conserver la concision des spécifications en permettant la représentation par un seul élément du montage complexe d'éléments pouvant être considérés individuellement;

c) permettre la définition préalable et l'emploi répété de montages d'éléments d'un usage courant.

Etant donné que les définitions de procédures peuvent être incluses dans les définitions de systèmes, de blocs, de processus et de procédures, on peut appeler la même procédure à partir de plusieurs processus. L'emploi de bibliothèques de procédures communes est autorisé.

Le diagramme qui définit la procédure est semblable à un diagramme de processus. A la différence que le symbole de démarrage et le symbole d'arrêt du diagramme de processus sont respectivement remplacés par le symbole de démarrage de procédure et par le symbole de retour.

Les directives applicables aux diagrammes de processus s'appliquent aussi aux diagrammes de procédure (voir le § D.6.3.6). Un exemple de diagramme de procédure est représenté à la figure D-77.

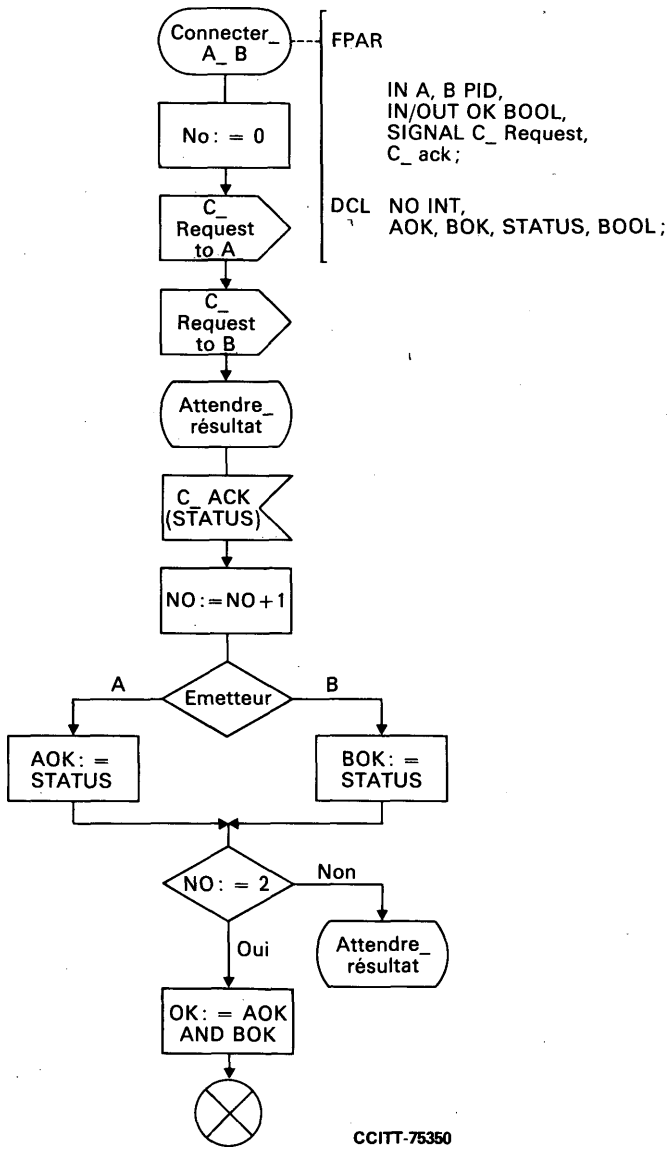


FIGURE D-77

Exemple de diagramme de procédure

D.6.3.6 *Diagramme de processus*

Un diagramme de processus définit le comportement d'un processus fonctionnant en LDS/GR. Des documents synoptiques auxiliaires forment souvent des préambules utiles lorsque les processus décrits sont vastes ou complexes. L'on trouvera quelques exemples relatifs à ce genre de documents à la fin du présent paragraphe.

D.6.3.6.1 *Méthodes de représentation des diagrammes de processus LDS/GR*

Si, dans un diagramme LDS/GR, les transitions sont décrites exclusivement par des symboles d'action explicites, on parle d'une méthode de représentation en fonction des transitions.

Si, par contre, les états sont décrits au moyen d'illustration d'état et si les différences dans les illustrations d'état sont utilisées pour indiquer des actions de transition implicites, on peut parler d'une méthode de représentation en fonction des états.

La combinaison de ces deux méthodes est appelée la méthode combinée.

Des exemples de ces trois méthodes sont donnés dans les figures D-78 à D-80.

La méthode de représentation en fonction des transitions est applicable lorsque la séquence des actions est plus importante que la description détaillée des états.

La méthode de représentation en fonction des états est applicable lorsque la séquence des actions à l'intérieur de chaque transition revêt peu d'importance et lorsque l'explication graphique et la représentation compacte sont souhaitables. Elle est disponible pour les applications pour lesquelles des éléments graphiques appropriés sont définis (voir le § D.6.3.6.22).

La méthode combinée est applicable lorsque la redondance qui en résulte est jugée utile.

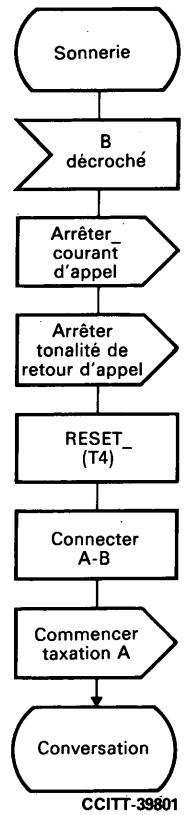
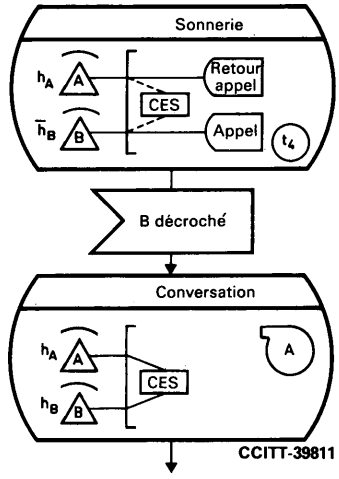


FIGURE D-78

Représentation en fonction des transitions



CES Circuit d'entrée/sortie

FIGURE D-79

Représentation en fonction des états

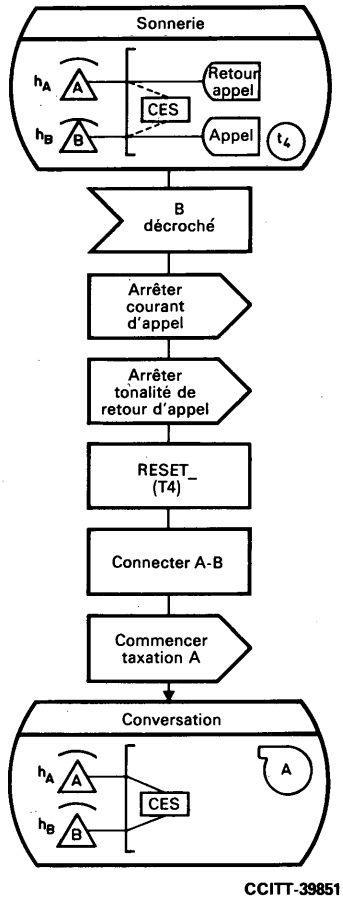


FIGURE D-80

Représentation combinée

D.6.3.6.2 *Symboles et règles de connexion*

L'annexe C aux Recommandations Z.100-Z.104 donne les symboles dont l'emploi est autorisé dans un diagramme de processus, ainsi que les instructions relatives à leur connexion.

D.6.3.6.3 *Diagrammes bien structurés*

Les paragraphes suivants contiennent une série de directives générales pour l'élaboration de diagrammes bien structurés.

D.6.3.6.3.1 *Début et fin d'un diagramme*

Il est possible de créer explicitement des instances de processus et, dans ce cas, il faut établir le diagramme de processus en commençant par le symbole début. Pour les processus où il n'y a pas création explicite d'instances, il existe un état de début, identifié par sa position au sommet du diagramme de processus.

Il est généralement préférable de dessiner un diagramme LDS où l'enchaînement va du haut vers le bas de la page.

Un diagramme peut être réparti sur plusieurs pages. L'enchaînement d'une page à la suivante ne peut être représenté qu'au moyen des connecteurs (voir le § D.6.3.6.16) ou par des apparitions multiples du même état (voir le § D.6.3.6.5).

D.6.3.6.3.2 *Autres dispositions possibles d'un diagramme*

La suspension d'un processus est représentée par des symboles d'état. Il appartient à l'usager LDS de décider si oui ou non ces suspensions doivent être mises en évidence (par l'interruption des diagrammes au début de chaque état). Conformément aux règles du LDS, trois méthodes sont possibles.

Méthode A

Le diagramme est interrompu à chaque symbole d'état et sera poursuivi en commençant par ce symbole d'état.

Selon la méthode A, le diagramme LDS complet comprend une série de *n* diagrammes, *n* étant le nombre des états du processus. Chaque diagramme commence par un état initial différent; il représente toutes les transitions à partir de cet état et se termine par l'état final de chaque transition. Le même état final peut apparaître dans plusieurs diagrammes.

La figure D-81 représente un exemple.

A noter que les transitions ainsi que les entrées sont seulement indiquées par des flèches.

Si le processus a un début, le symbole début et la transition de démarrage seront également représentés par un diagramme distinct [voir la figure D-81 b)]; de même, si le processus comporte un arrêt, les états et les transitions conduisant aux symboles arrêt seront représentés séparément [voir la figure D-81 c)].

Méthode B

Le diagramme est ininterrompu.

Selon la méthode B, le diagramme est dessiné sous la forme d'un graphe entièrement connecté. L'exemple de la figure D-81 se présente dans la figure D-82 sous la forme d'un graphe entièrement connecté.

Méthode C

Le diagramme est interrompu à certains symboles d'état.

Si le diagramme est seulement interrompu à certains symboles d'état, on dispose de plusieurs moyens équivalents de redessiner la figure D-81. L'un de ces moyens, qui consiste à utiliser deux diagrammes, est représenté dans la figure D-83.

Selon la méthode C, certains états apparaîtront dans plusieurs diagrammes. Si l'on applique cette méthode, il est évidemment souhaitable de dessiner des connexions entre les parties logiquement apparentées du diagramme. Par exemple, il est très fréquent qu'une partie relativement réduite du processus représente le comportement correspondant à la plupart des situations habituelles. Il serait souhaitable de représenter cette partie dans un diagramme distinct.

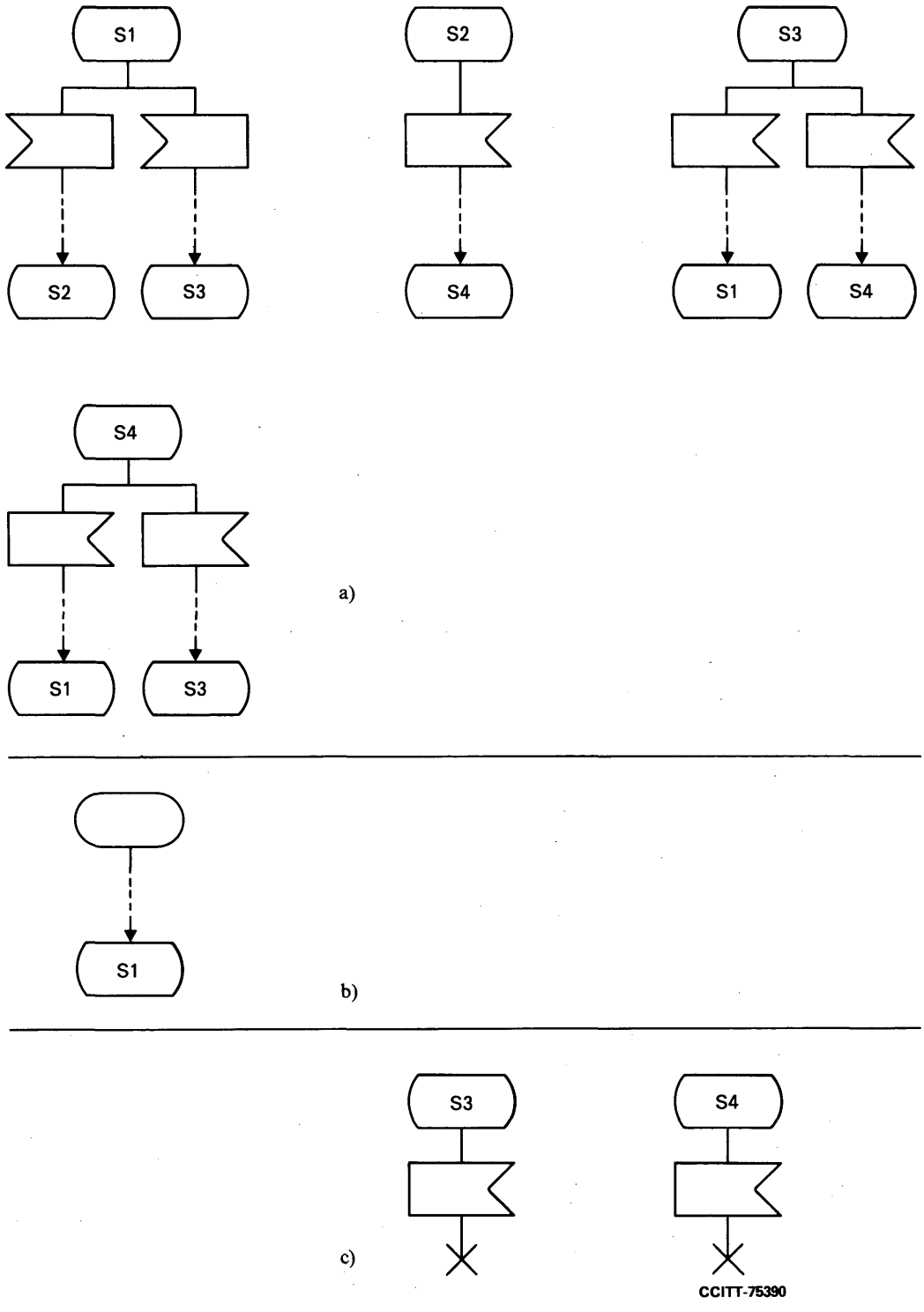


FIGURE D-81

Graphes entièrement séparés

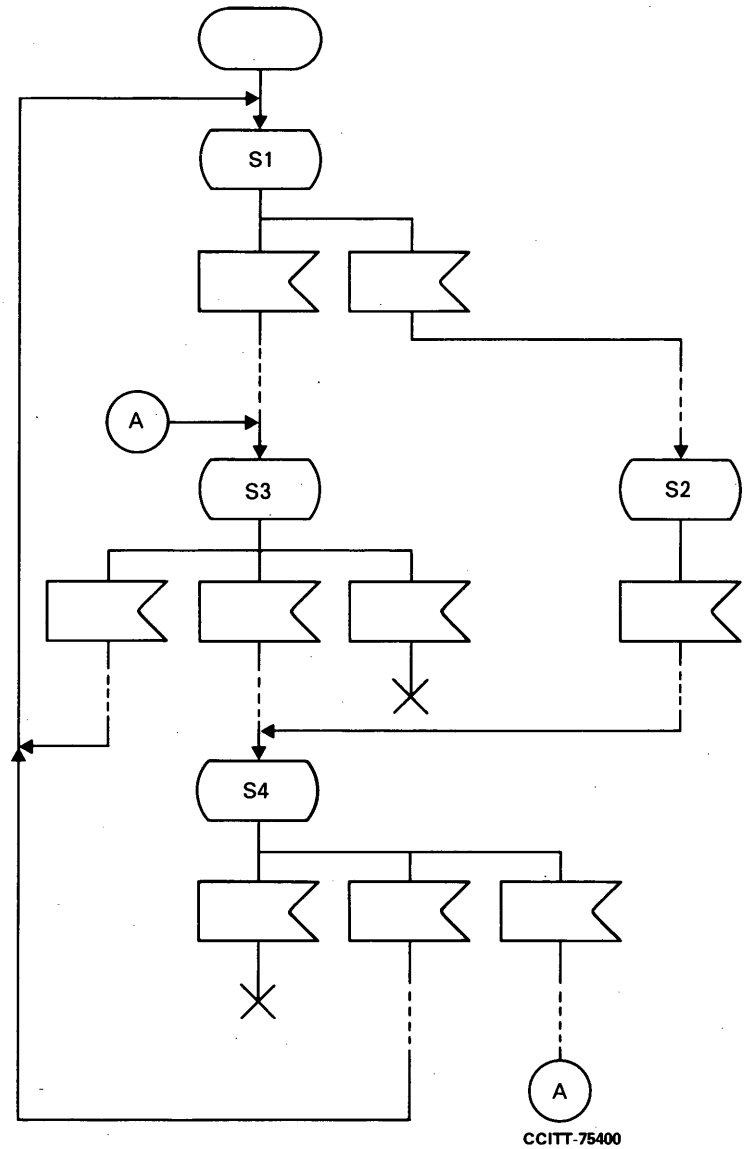


FIGURE D-82

Graphe entièrement connecté

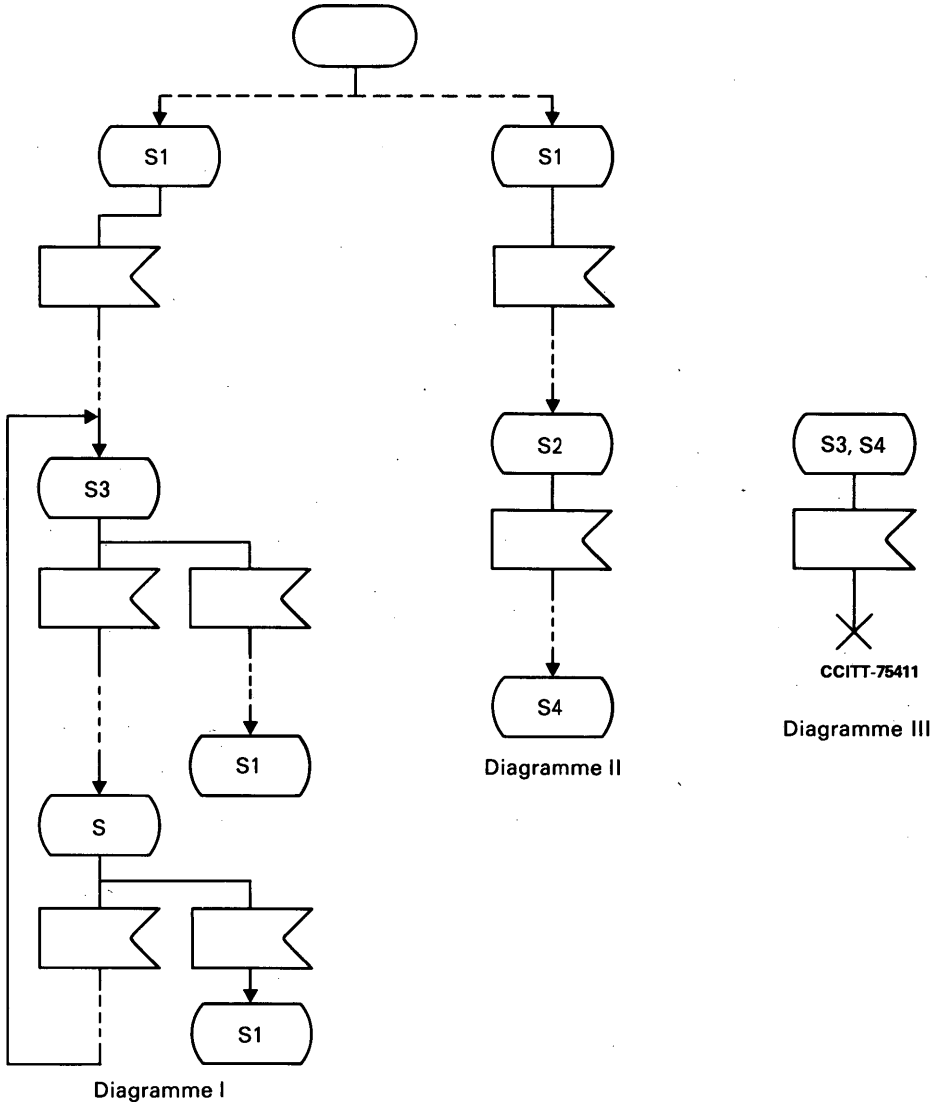


FIGURE D-83

Graphes partiellement connectés

Comparaison des trois méthodes

L'avantage de la méthode A réside dans le fait qu'il faut peu de temps pour décider de la manière dont un diagramme doit être dessiné; l'avantage réside aussi dans la facilité de modifier le diagramme et dans la facilité de la restitution de l'information.

La méthode B a l'avantage de permettre un aperçu global du processus alors que la méthode C aboutit à la structuration du processus conformément aux critères déterminés au préalable et qui permettent de déterminer quelles parties du processus présentent le plus d'intérêt.

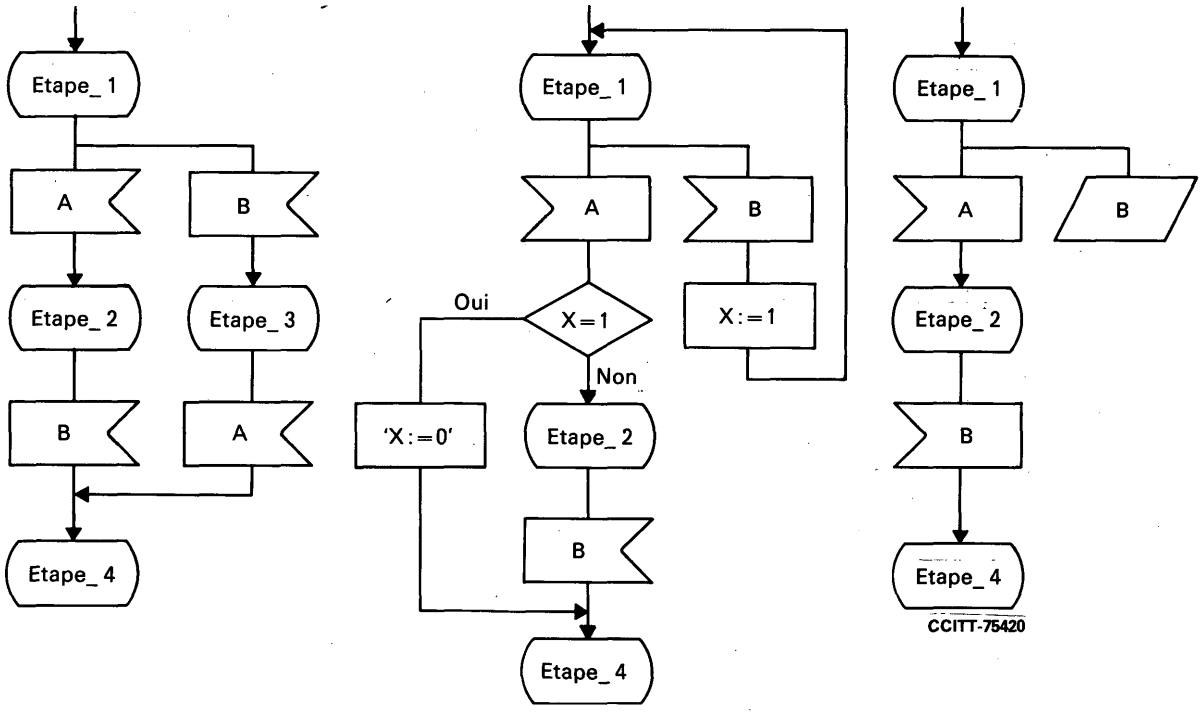
Clarté des diagrammes

Etant donné que les diagrammes LDS constituent un moyen de communication entre les auteurs et les lecteurs, il est hautement souhaitable que ces diagrammes soient clairs. La clarté du diagramme dépend non seulement de la méthode de disposition du diagramme mais également de la complexité du processus (par exemple, le nombre des états, des entrées et des décisions), complexité qui dépend à son tour de la structuration.

Etant donné que les diagrammes doivent être faciles à lire et à comprendre, il faut particulièrement veiller à choisir une disposition appropriée.

En recherchant la meilleure méthode de représentation, l'auteur doit prendre en considération les différentes méthodes pour utiliser le LDS. On peut considérer que les exemples donnés en a), b) et c) de la figure D-84 sont des moyens équivalents de représenter une partie du processus. Etant donné qu'il n'y a ni tâches, ni entrées (sauf la tâche artificielle qui consiste à mettre à 1 ou à 0 la variable X), les trois exemples sont en fait logiquement équivalents.

C'est le diagramme c) qui semble donner la représentation la plus claire.



a) Exemple de diagramme LDS

b) Diagramme LDS équivalent comportant une décision

c) Diagramme LDS équivalent utilisant une mise en réserve

FIGURE D-84

Les trois diagrammes montrent comment le processus passe de l'état 1 à l'état 4 quand il reçoit les entrées A et B dans un ordre aléatoire

D.6.3.6.3.3 *Les textes LDS/PR dans les diagrammes de processus*

On emploie la syntaxe LDS/PR pour décrire les déclarations de variables et les définitions de types de données lors de la description d'une définition de processus en LDS/GR. Ce texte peut soit être directement inséré dans le diagramme, soit figurer dans un document distinct. Lorsque le texte en question n'est pas directement inséré dans le diagramme, une référence doit indiquer dans quel document il se trouve. Voir les figures D-85 et D-86.

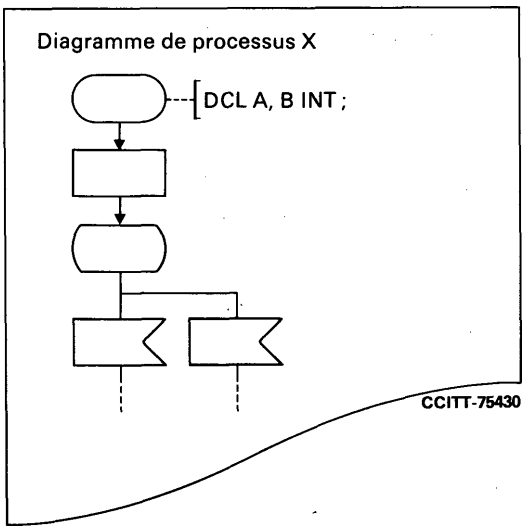


FIGURE D-85

Exemple de texte LDS/PR inséré dans un diagramme de processus

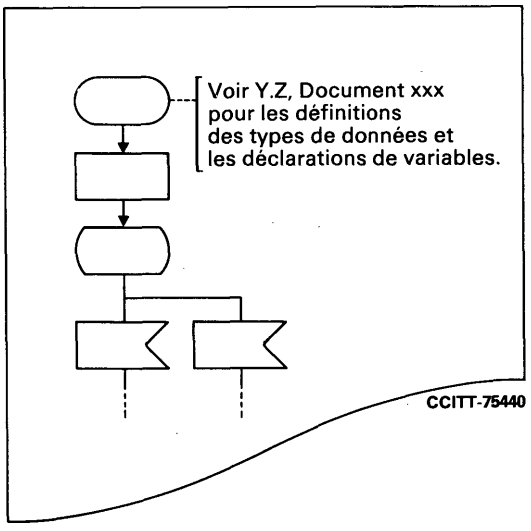


FIGURE D-86

Exemple de renvois à des définitions placées ailleurs

Le choix de la méthode dépend de la longueur du texte à insérer ou à placer en référence. Si l'on insère trop de textes PR, le diagramme risque de perdre de sa clarté; par contre si le texte est court, son inclusion dans le diagramme peut rendre ce dernier plus lisible.

D.6.3.6.3.4 *Définitions de procédures*

Lorsque des procédures sont définies localement aux processus, les diagrammes de procédures peuvent être insérés dans les diagrammes de processus ou cités en référence.

En cas d'insertion des diagrammes de procédures, il importe de séparer le graphe qui définit la procédure de celui qui définit le processus. Pour ce faire, l'on peut subdiviser le diagramme de processus en chapitres ou en sections, comme le montre la figure D-87.

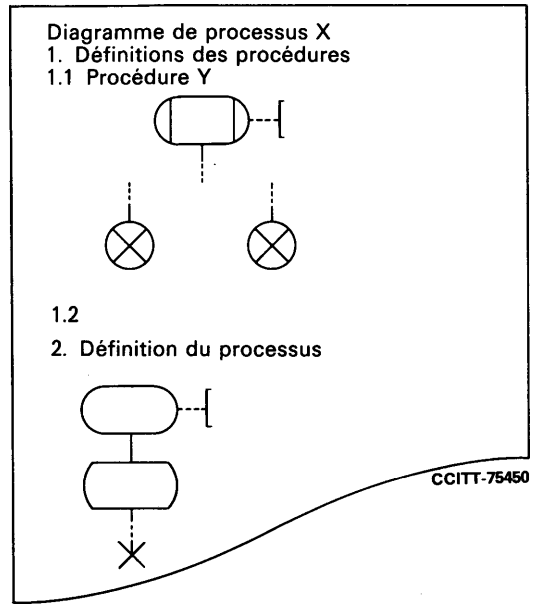


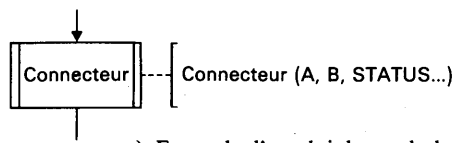
FIGURE D-87

Exemple d'insertion de définitions de procédures dans un diagramme de processus

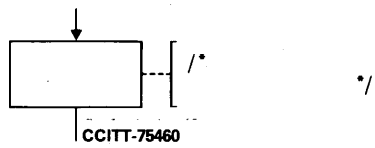
D.6.3.6.3.5 *Textes placés dans les symboles*

Un texte associé à un symbole doit normalement se trouver à l'intérieur du symbole en question. Ceci n'est cependant pas toujours pratique du fait de la longueur du texte et/ou de la taille des symboles.

Le texte peut être placé à l'extérieur du symbole en cas de besoin. Dans ce cas, il convient de préciser que le texte placé à l'extérieur du symbole est associé à ce dernier et qu'il ne s'agit pas d'un commentaire. Pour ce faire, l'on emploie le symbole d'extension du texte représenté dans la figure D-88.



a) Exemple d'emploi du symbole d'extension de texte pour un texte formel placé ailleurs



b) Exemple d'emploi d'un mécanisme de « note » pour une association avec un texte informel.

FIGURE D-88

D.6.3.6.4 *Création de processus*

D.6.3.6.4.1 *Demande de création*

Le symbole de demande de création décrit la création d'une nouvelle instance de processus. L'on notera qu'il n'est possible de créer des instances de processus qu'à l'intérieur du bloc de l'instance créatrice.

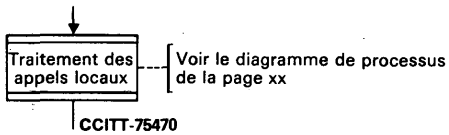


FIGURE D-89

Exemple d'emploi du symbole de demande de création

Afin de rendre le diagramme plus clair, il convient d'inclure sous la forme d'un commentaire une référence au diagramme de processus du processus créé. Si les paramètres effectifs sont demandés, ils doivent être fournis au moyen de la syntaxe LDS/PR.

D.6.3.6.4.2 *Début*

Le symbole début d'un diagramme de processus décrit le point de départ du comportement d'une instance de processus au moment de sa création. Il convient d'associer les paramètres formels du processus à ce symbole.

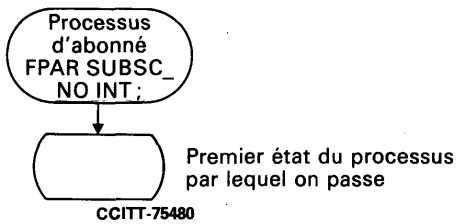
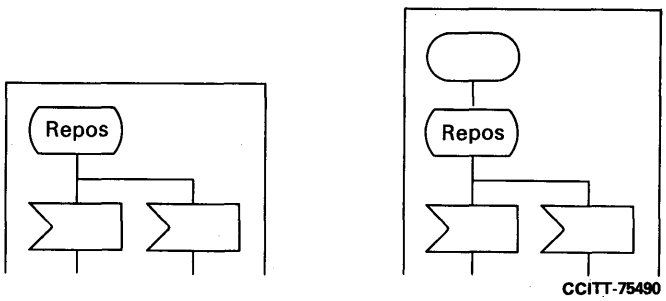


FIGURE D-90

Exemple d'emploi du symbole début

Afin de ne pas invalider les anciens diagrammes LDS, le symbole début demandé peut être implicite; l'on admet alors qu'il figure avant le «premier» symbole d'état du diagramme de processus, le mot «premier» désignant le premier symbole d'état placé en haut de la première page du diagramme.



a) Aspect du diagramme b) Interprétation du diagramme

FIGURE D-91

Symbole début implicite

Le symbole début doit se présenter comme le premier symbole d'un diagramme; en le «dissimulant» à l'intérieur du diagramme, on risque d'embrouiller le lecteur.

D.6.3.6.5 *Etats*

Un état est représenté par un symbole d'état et il comporte des symboles d'entrée, et éventuellement de mise en réserve, qui lui sont associés. Un exemple d'état est représenté dans la figure D-92.

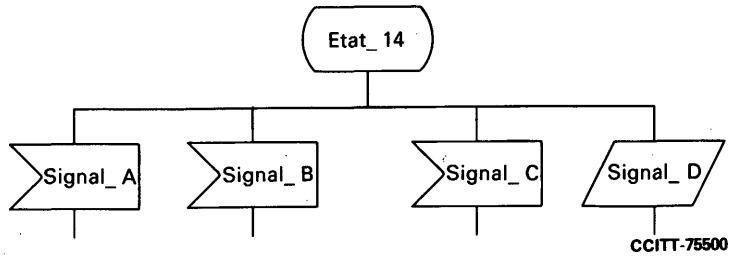


FIGURE D-92

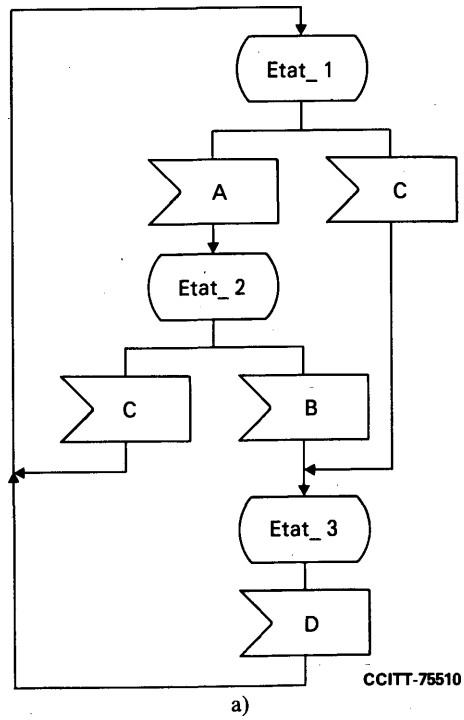
Exemple d'emploi d'un état

D.6.3.6.5.1 *Apparitions multiples et symboles d'état suivant*

Le même état peut apparaître plusieurs fois dans un diagramme LDS pour des raisons de commodité, pour simplifier la représentation ou pour aider à sa compréhension. Lorsque c'est le cas, on considère que le diagramme est totalement équivalent au diagramme que l'on obtiendrait en regroupant toutes les apparitions multiples du même état. Les figures D-93 et D-94 illustrent ceci. Dans la figure D-93 b), un symbole d'état sert de connecteur avec l'état principal de même nom lorsqu'un symbole d'état suivant renvoie au symbole d'état en question. La figure D-94 représente un état au moyen de multiples symboles, qui chacun renferme un sous-ensemble d'entrées (ou de mises en réserve).

Les diagrammes a) et b) de la figure D-93 sont logiquement équivalents. Le diagramme a) ne contient qu'une seule apparition de chaque état tandis que le diagramme b) fait appel à des apparitions multiples. Dans le diagramme b), l'état présente à l'occasion d'une apparition principale tous ses symboles associés d'entrée (et de mise en réserve). Lorsque cet état est accessible depuis d'autres points du diagramme (comme le point terminal d'une transition), cet état est présenté sans entrées ou mises en réserve associées. Un commentaire qui fait référence au symbole d'état à partir du symbole d'état suivant sert à améliorer la clarté, notamment lorsque les apparitions sont réparties sur plusieurs pages.

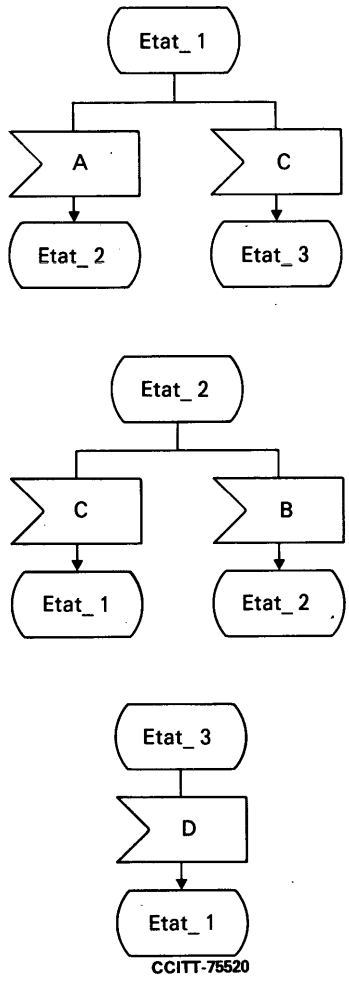
La figure D-94 se sert des apparitions multiples d'un état pour constituer l'ensemble des entrées (et des mises en réserve). Chaque apparition de l'état est représentée accompagnée seulement d'un sous-ensemble d'entrées (et de mises en réserve).



a)

FIGURE D-93

Diagramme complet



b)

FIGURE D-93b

Diagramme a) comportant des états principaux et un état suivant utilisés comme connecteurs pour les états

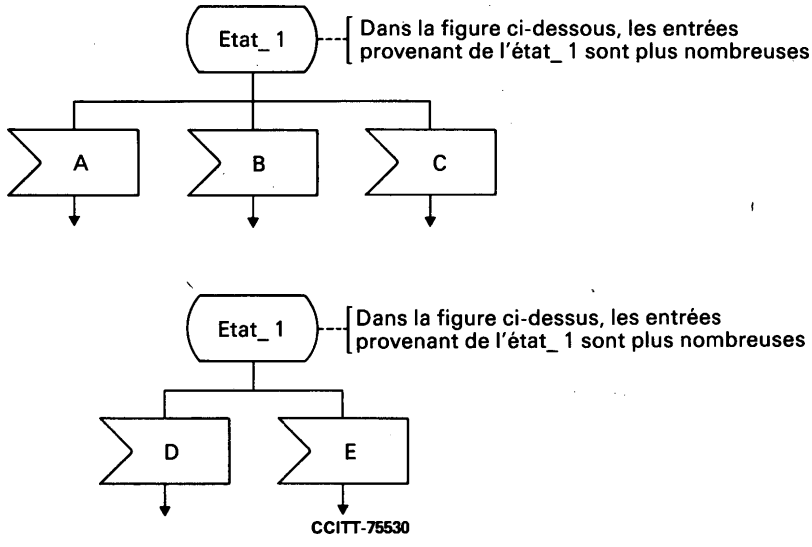


FIGURE D-94

Apparitions multiples d'un état dans le cas où il n'est pas possible de représenter clairement toutes les entrées provenant du même symbole

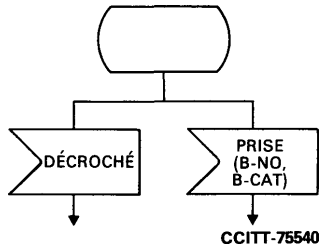
Cette méthode a été appliquée avec succès pour les états qui ont un nombre relativement élevé d'entrées ou de mises en réserve. Mais elle fausse l'interprétation du diagramme si le lecteur n'est pas conscient du fait qu'il existe des apparitions multiples du même état. Pour éviter de tels malentendus, les états ne comportant qu'un sous-ensemble d'entrées ou de mises en réserve doivent être assortis d'un commentaire faisant référence aux autres états ainsi qu'à leurs entrées et mises en réserve associées, comme indiqué à la figure D-94.

L'on peut faire appel d'une manière opportune à l'apparition multiple afin de concentrer l'attention du lecteur sur certains aspects (par exemple, l'ordre normal de traitement des signaux) tout en renvoyant les autres aspects aux autres pages (par exemple, traitement en cas d'urgence).

D.6.3.6.6 *Entrées*

Le symbole d'entrée connecté à un symbole d'état représente le concept d'entrée.

Lorsque le signal qui doit être reçu par l'entrée achemine des données, le symbole en syntaxe LDS/GR donne la liste des variables locales dans lesquelles les valeurs doivent être stockées. La liste peut être omise lorsque aucune donnée n'est acheminée. Le fait que des données soient acheminées en l'absence de la liste doit être interprété comme l'annonce de la perte des valeurs à leur réception. Voir l'exemple de la figure D-95.



Remarque – Aucune flèche ne doit être représentée sur la ligne qui aboutit au symbole d'entrée.

FIGURE D-95

Exemple d'emploi du symbole d'entrée

Si une même transition peut être déclenchée après un état par plusieurs signaux, le même symbole d'entrée contenant une liste de signaux peut représenter toutes les entrées. Dans la liste de signaux, tous les noms de signaux (et leurs listes de variables) sont séparés par des virgules.

Les anciennes méthodes de LDS distinguaient les entrées externes des entrées internes. La différence était la suivante: les entrées externes recevaient des signaux émis depuis l'extérieur du bloc dans lequel se trouvait le processus. Etant donné qu'il n'existe aucune différence sémantique entre des signaux provenant de l'intérieur et des signaux provenant de l'extérieur du bloc, ces concepts ont été supprimés. Lorsque l'on trouve dans des diagrammes le symbole spécial précédemment employé pour représenter une entrée interne, il convient de l'interpréter comme un symbole d'entrée. Voir la figure D-97.

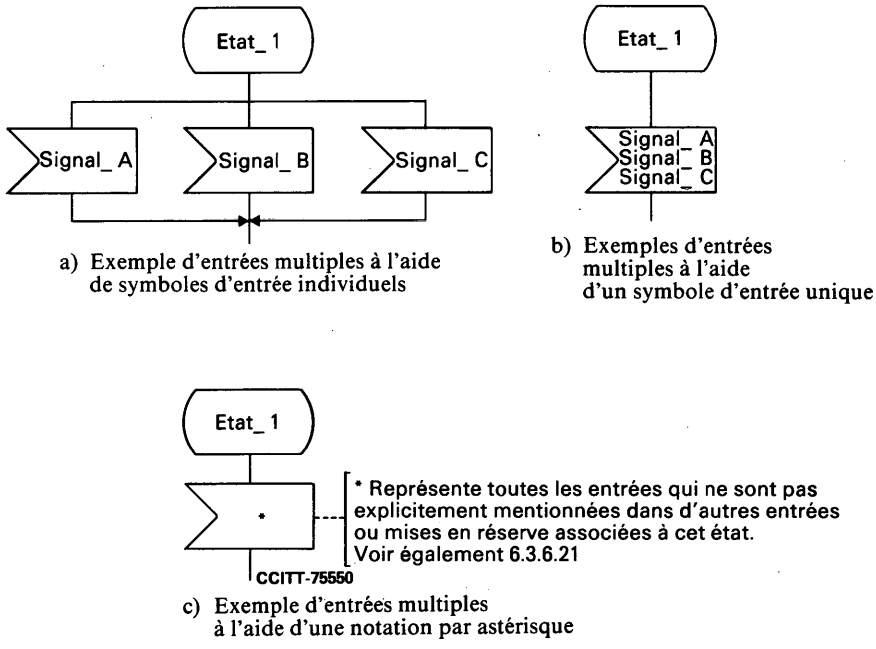
D.6.3.6.7 *Mises en réserve*

Le symbole de mise en réserve représente la liste de signaux mis en réserve d'un état. Il convient de nommer à l'intérieur du symbole les signaux devant être mis en réserve. Si l'on emploie plusieurs symboles connectés à un symbole d'état donné, il faudra les considérer comme un symbole de mise en réserve contenant la liste constituée par les noms indiqués dans tous les symboles.

Les noms que contient la liste doivent être séparés par des virgules. Il existe également une représentation abrégée qui fait appel à la «notation par astérisque» décrite dans le § D.6.3.6.21.

D.6.3.6.8 *Sorties*

Le symbole de sortie représente une sortie qui émet un signal et lui fournit les valeurs à acheminer. La liste des valeurs à acheminer est spécifiée à l'aide du LDS/PR; il est préférable d'inclure cette liste dans le symbole. Le symbole doit également contenir l'adresse de l'instance du processus vers laquelle le signal est émis. Voir les exemples de la figure D-99.



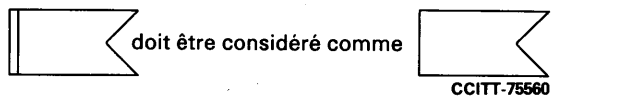
a) Exemple d'entrées multiples à l'aide de symboles d'entrée individuels

b) Exemples d'entrées multiples à l'aide d'un symbole d'entrée unique

c) Exemple d'entrées multiples à l'aide d'une notation par astérisque

FIGURE D-96

Autre représentation possible des entrées multiples

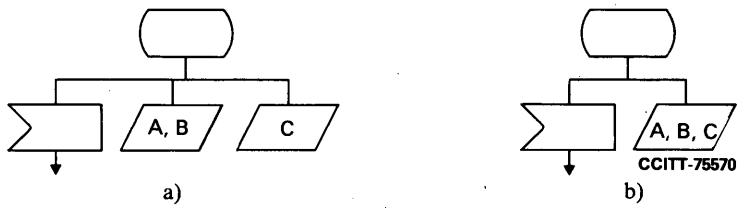


a) ancien symbole

b) nouveau symbole

FIGURE D-97

Equivalence des entrées externe et interne

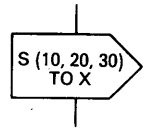


a)

b)

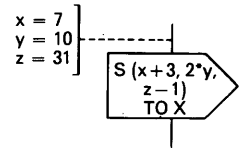
FIGURE D-98

Deux représentations différentes de la même liste de mise en réserve



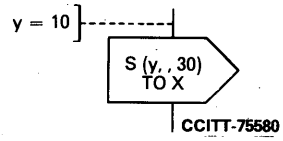
Remarque – Le signal S a trois valeurs, 10, 20 et 30 qui lui sont associées

a)



Remarque – Pour l'interprétation de S : x, y et z ont (dans le présent exemple) les valeurs 7, 10 et 31 respectivement. S émet les valeurs 10, 20 et 30.

b)



Remarque – Pour l'interprétation de S, y a (dans le présent exemple) la valeur 10. S émet les valeurs 10, une valeur indéfinie et 30.

c)

FIGURE D-99

Sortie avec données associées

Les anciennes méthodes de LDS établissaient une distinction entre les sorties externes et les sorties internes, comme pour les entrées. Tout ancien symbole de sortie interne présent dans un diagramme doit être interprété comme un symbole de sortie. Voir la figure D-100.

a) ancien symbole b) nouveau symbole

FIGURE D-100

Equivalence des sorties externe et interne

D.6.3.6.9 *Condition de validation*

Cette condition est représentée par le symbole de condition de validation; ce symbole résulte de la combinaison du symbole d'entrée et d'une syntaxe graphique qui représente la condition booléenne.

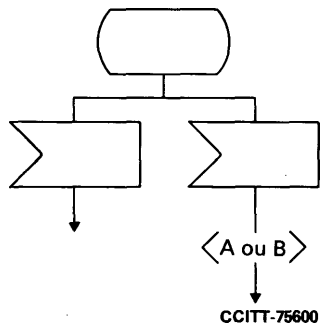


FIGURE D-101

Il convient de considérer cette combinaison comme un seul symbole; c'est pourquoi la ligne qui relie les deux parties ne doit pas être fléchée. Voir la figure D-101.

D.6.3.6.10 *Signal continu*

Le symbole de signal continu représente le signal continu.

L'expression placée à l'intérieur doit être booléenne.

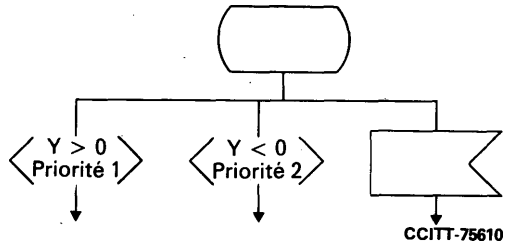


FIGURE D-102

Exemple de conditions de validation

La clause de priorité s'applique obligatoirement si plusieurs conditions de validation sont associées à un état.

D.6.3.6.11 *Tâche*

Une tâche est représentée par un symbole de tâche. Il est préférable de placer dans le symbole de tâche la description de la tâche, que cette description prenne la forme d'un texte LDS/PR ou d'un texte informel.

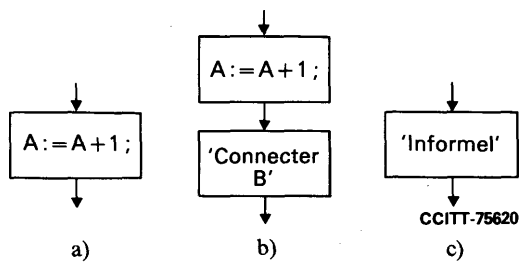


FIGURE D-103

Exemples de symboles de tâche

Dans les diagrammes qui comportent à la fois des énoncés formels et un texte informel, ce dernier doit être placé entre des apostrophes.

D.6.3.6.12 *Décisions*

Une décision est représentée par un symbole de décision.

Le texte d'une question se rapportant à une certaine décision est placé dans un symbole de décision. Il n'est pas nécessaire de faire suivre les questions de points d'interrogations; l'expression à évaluer, par exemple, n°, A + B, Z, suffit. Le symbole doit avoir deux branches ou plus. La réponse à la question doit être inscrite à côté et à droite (ou au-dessus) de la branche correspondante. Les branches réunies doivent contenir toutes les réponses possibles. Chaque réponse peut apparaître avec la valeur de réponse (par exemple, 17, 12) ou avec un opérateur placé comme préfixe (par exemple, > 18, ≠ 34, = 25). La figure D-104 donne quelques formats possibles.

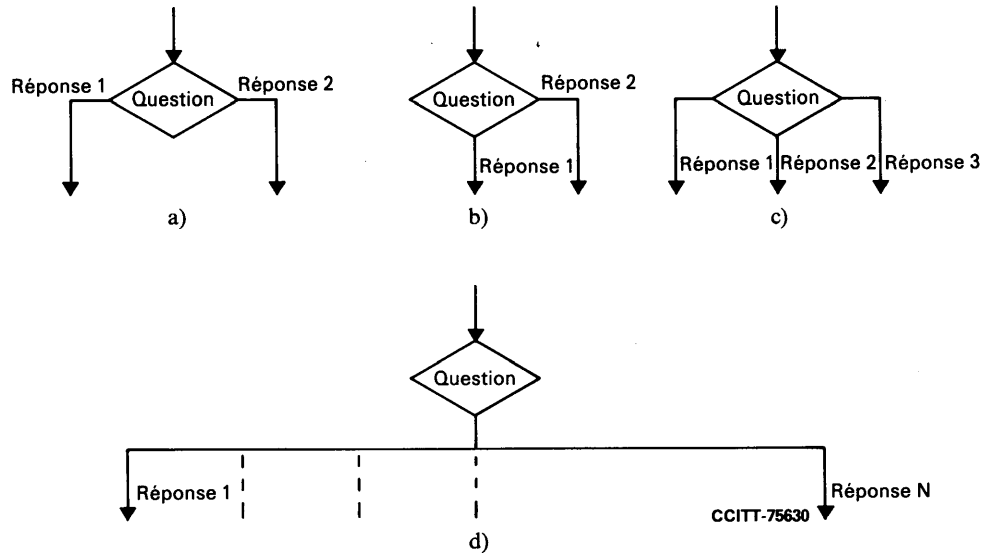


FIGURE D-104

Quelques formats possibles pour le symbole de décision

Plusieurs questions se rapportent à une condition spécifique dans laquelle les seules réponses possibles sont «oui» ou «non», c'est-à-dire VRAI ou FAUX.

Par exemple:

Abonné B occupé

Appareil téléphonique «raccroché»

Chiffre 2 reçu

Z = 1

Un exemple de cette condition est donné à la figure D-105.

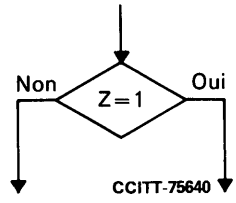


FIGURE D-105

Représentation d'une décision binaire simple

S'il y a plus de deux réponses, les formats de la figure D-106 sont possibles si l'on admet que Z est un nombre entier positif. La réponse ELSE désigne toute réponse non couverte par les autres réponses, c'est-à-dire des valeurs différentes de celles de l'ensemble (1, 2, 3, 9, 10, 11, 12, 13, 14, 15).

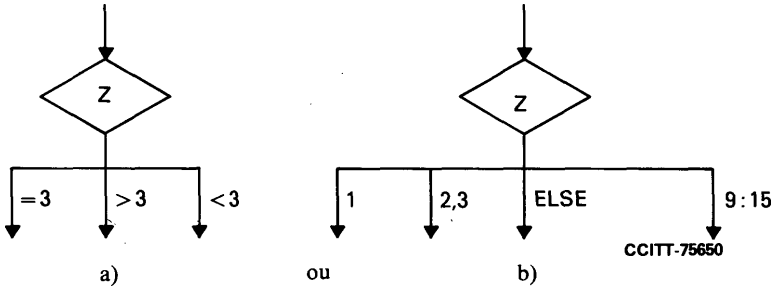


FIGURE D-106

« Formatage » de questions et de réponses

D.6.3.6.13 *Macro*

Les symboles de macro peuvent être placés à n'importe quel endroit du diagramme. On ne peut déterminer l'exactitude du diagramme qu'après avoir remplacé le symbole de macro par le contenu de sa définition. Toutefois, on emploie de préférence les macros dans les diagrammes de processus dans lesquels un symbole de tâche pourrait se présenter.

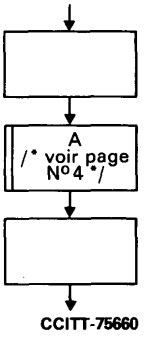


FIGURE D-107

Emploi du macrosymbole

En cas d'emploi de macros, il serait utile de placer dans le symbole de macro une référence renvoyant à l'emplacement de la définition. (La figure D-107 donne un exemple.)

L'usager employant des macros doit savoir que ces dernières peuvent comporter des effets secondaires. Etant donné qu'une macro occupe généralement l'emplacement d'un symbole de tâche, les lecteurs peuvent déterminer la sémantique d'une tâche.

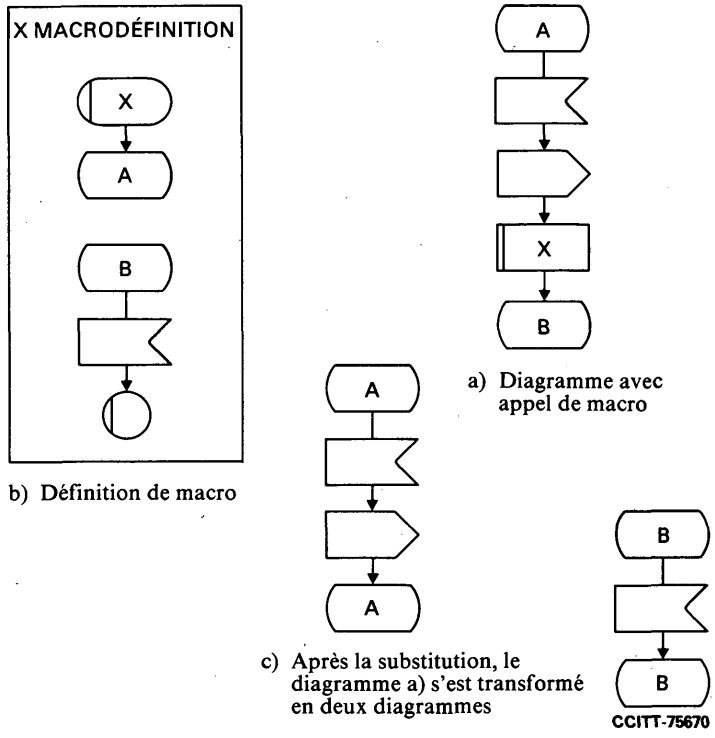


FIGURE D-108

Exemple d'emploi non recommandé de macro

La figure D-108 montre les changements que la séquence du diagramme peut subir suite au remplacement d'un appel de macro. A noter que, lorsque les macros renferment des états, il est possible de fractionner la transition qui contient la référence en plusieurs transitions.

Lorsque des macros ont plusieurs entrées et/ou sorties, elles doivent être clairement identifiées au moyen d'étiquettes, placées de préférence à droite des lignes de liaison d'entrée ou de sortie.

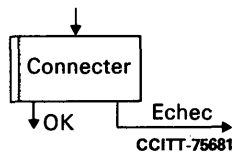


FIGURE D-109

Macro munie de plusieurs sorties

D.6.3.6.14 *Procédures*

Le symbole d'appel de procédure représente l'appel d'une procédure. Il est préférable de placer dans le symbole les paramètres qui accompagnent l'appel; au cas où cela ne serait pas possible, ou pratique, ils peuvent figurer dans un symbole d'extension de texte placé à côté.

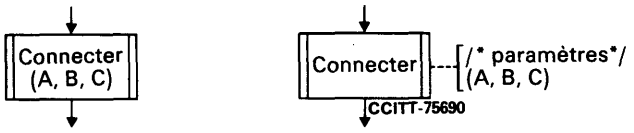


FIGURE D-110

Exemple d'emploi du symbole d'appel de procédure

Ce symbole peut occuper les mêmes emplacements qu'un symbole de tâche.

Les procédures permettent de réduire la complexité d'un diagramme et de recourir à des solutions normalisées.

D.6.3.6.15 *Option*

Le symbole d'option sert à décrire plusieurs autres comportements de processus susceptibles d'accompagner un diagramme donné. Ce dispositif est utile lorsque les différences entre les comportements de plusieurs processus sont minimes. Le symbole renferme une expression d'option. Cette expression doit pouvoir être évaluée avant l'interprétation du processus, c'est-à-dire qu'il ne doit pas s'agir d'une condition dynamique. L'évaluation de l'expression doit pouvoir aboutir à un ensemble discret de variantes servant à l'étiquetage des lignes de liaison de sortie.

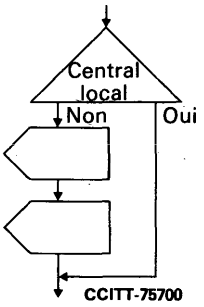


FIGURE D-111

Exemple d'emploi d'options

Il est préférable de placer les variantes des options à droite des lignes de liaison de sortie.

Le diagramme ne doit pas être interprété avant que toutes les expressions d'option aient été évaluées; suite à quoi, le diagramme doit être pris comme si toutes les branches non accessibles qui suivent les options avaient été supprimées.

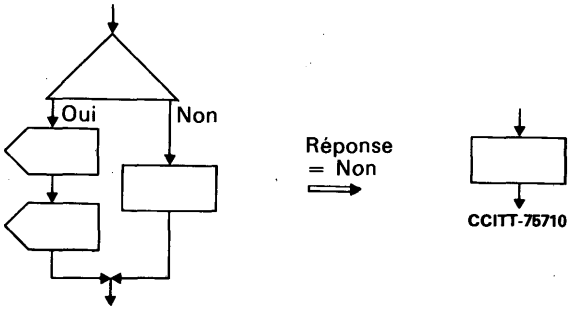


FIGURE D-112

Interpétation d'options

D.6.3.6.16 *Connecteurs*

Considérations générales

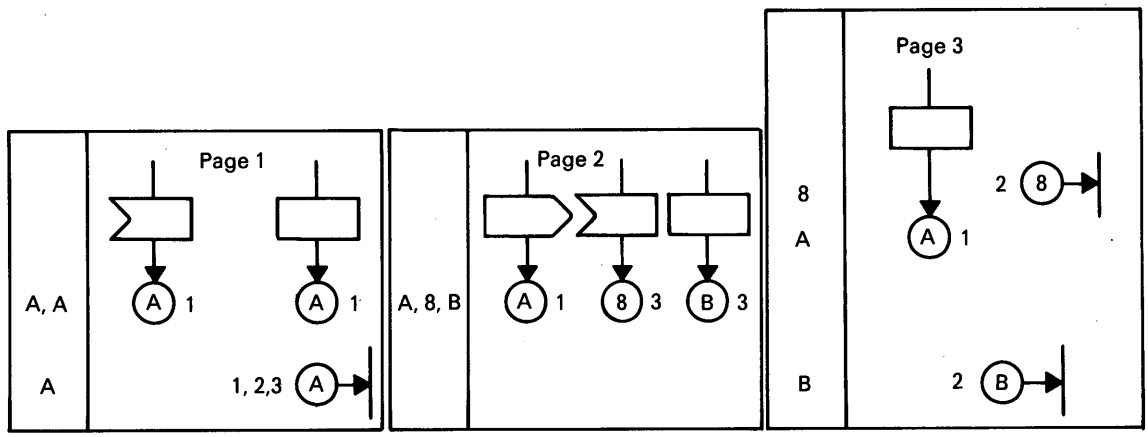
Pour l'établissement d'un grand diagramme en LDS, le manque d'espace peut amener à le répartir sur plusieurs pages. A cette fin, on utilise des connecteurs.

On emploie également des connecteurs pour éviter que les lignes de liaison ne se coupent, ce qui risque de rendre les diagrammes moins clairs. Il est normalement préférable de dessiner un diagramme en LDS en présentant le flux (ou enchaînement) à partir du haut pour arriver au bas de la page.

Utilisation de connecteurs pour la représentation des lignes de liaison

Une ligne de liaison peut être interrompue par une paire de connecteurs associés, et l'on admet que le flux (ou enchaînement) part du connecteur de sortie pour aboutir au connecteur d'entrée. Chaque connecteur a un nom. Les connecteurs associés ont le même nom. Pour chaque nom il n'existe qu'un seul connecteur d'entrée mais il peut y avoir plusieurs connecteurs de sortie.

Il est souhaitable qu'à côté de chaque connecteur de sortie on indique le feuillet où figure le connecteur d'entrée approprié et qu'à côté de chaque connecteur d'entrée on indique le ou les feuillets où figurent le ou les connecteurs de sortie appropriés. Voir la figure D-113. Il est également souhaitable d'avoir en marge une colonne indiquant l'emplacement horizontal et l'ordre des connecteurs.



Remarque – Une autre méthode (également acceptable) d'indication des feuillets consiste à utiliser des commentaires, comme le montre la figure D-114

FIGURE D-113

Méthode d'indication des références de feuillets et des connecteurs

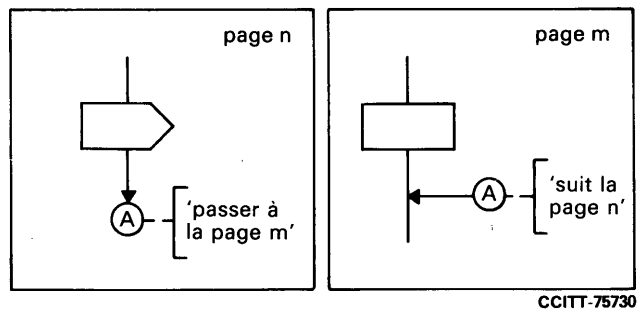


FIGURE D-114

Autre méthode d'indication des références de feuillets

Les connecteurs ne peuvent être employés que pour représenter les connexions à l'intérieur du même processus.

S'il existe de grands trajets de circulation, on peut appeler l'attention sur eux en ne les coupant pas au moyen de connecteurs sauf en haut et en bas des pages. En revanche, un petit trajet de circulation à l'intérieur d'un diagramme LDS peut être interrompu par un connecteur de sortie, le connecteur d'entrée associé étant placé sur une autre page, par exemple, une page suivant la fin des trajets de circulation les plus importants.

D.6.3.6.17 *Divergence et convergence*

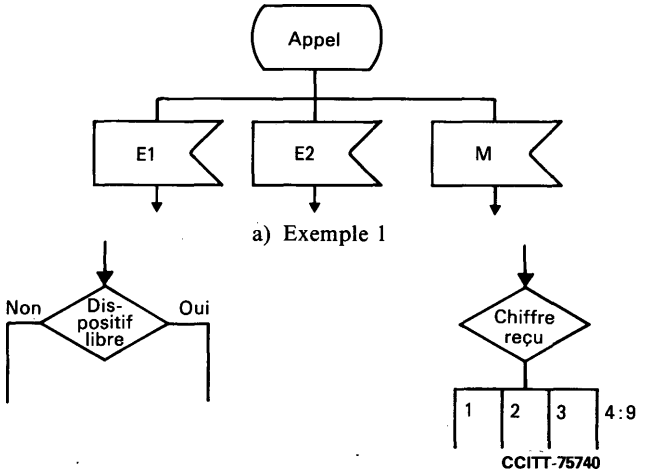
Divergence

Dans une transition d'un diagramme LDS, une divergence peut se produire seulement dans les cas suivants:

a) entre un symbole d'état et les symboles d'entrée et de mise en réserve qui lui sont associés, ou

b) immédiatement après un symbole de décision.

La figure D-115 donne quelques exemples de divergence.



a) Exemple 1

b) Exemple 2

c) Exemple 3

FIGURE D-115

Exemples de divergence

Convergence

Le point de convergence ne peut se trouver entre un symbole d'état et un symbole d'entrée ou de mise en réserve, mais en tout autre point d'un diagramme LDS.

Une convergence peut se présenter dans l'un quelconque des quatre cas de la figure D-116.

L'utilisation de la convergence peut réduire le nombre des symboles dans un diagramme LDS où une séquence de symboles et un texte associé sont répétés comme indiqué à la figure D-117.

D.6.3.6.18 *Commentaires*

Des commentaires peuvent être ajoutés dans un diagramme LDS pour préciser une partie du diagramme. Ils sont destinés à aider le lecteur et n'ont aucun effet sur l'interprétation du diagramme. Les commentaires ne doivent donc ni contredire la sémantique du diagramme, ni y ajouter quelque chose.

Un commentaire est attaché à une ligne de liaison ou à un symbole LDS par un crochet unique connecté par une ligne discontinue à la ligne de liaison ou au symbole concerné. Voir la figure D-118.

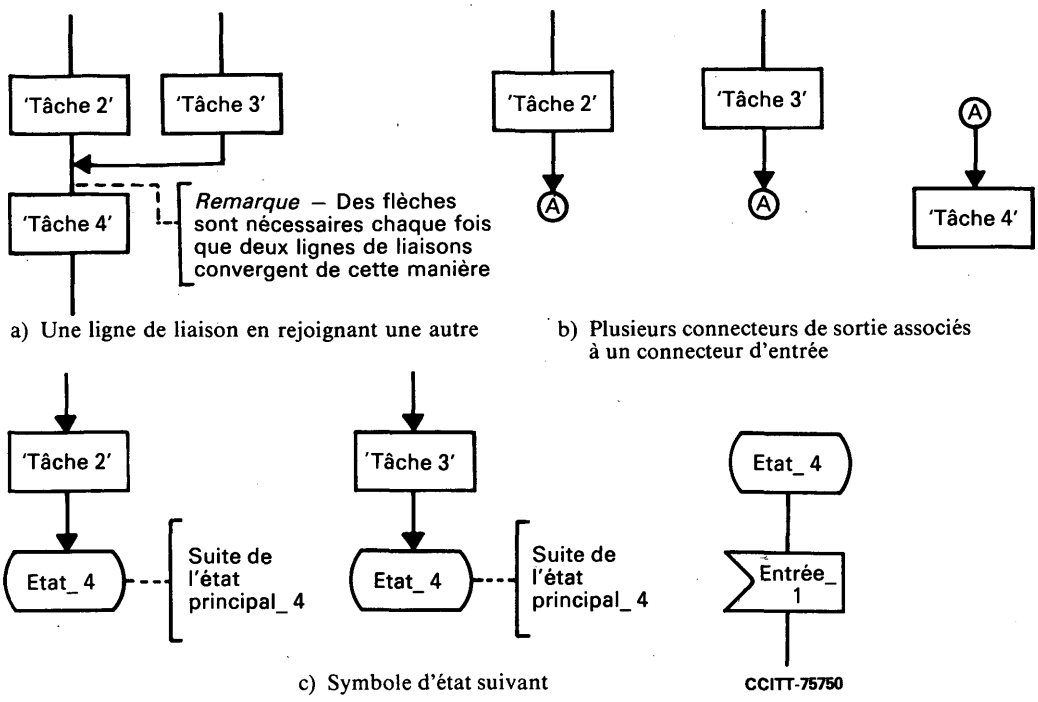
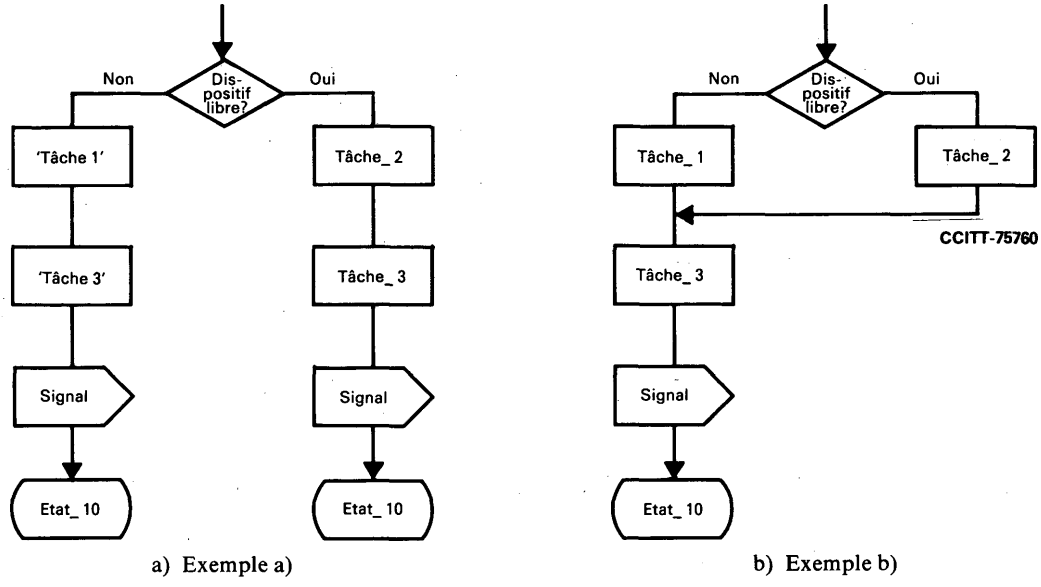


FIGURE D-116

Exemples de convergence



Remarque – L'exemple b) est équivalent à l'exemple a)

FIGURE D-117

Utilisation de la convergence

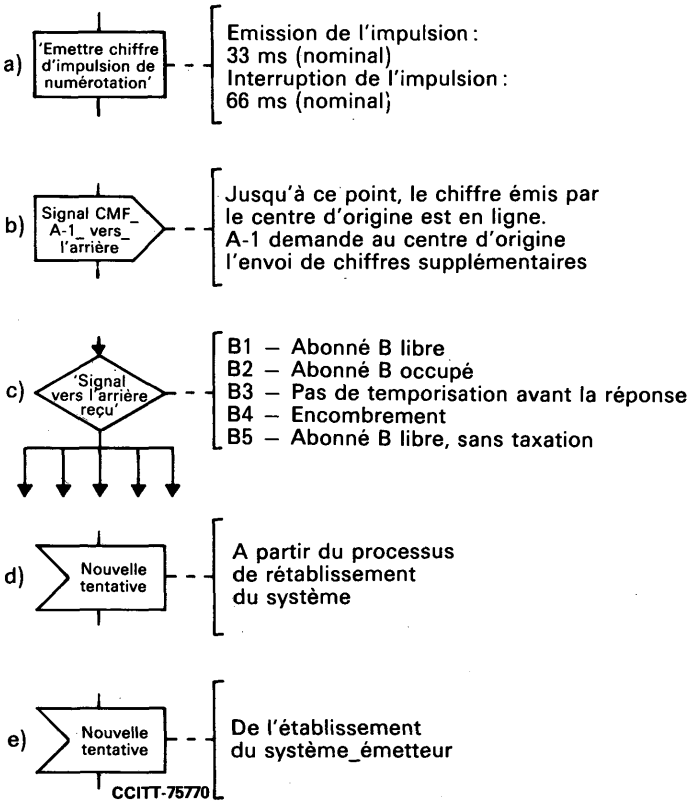


FIGURE D-118

Exemples d'utilisation de commentaires

L'on peut également insérer des commentaires au moyen de la syntaxe LDS/PR (/* ..... */) comme le montre la figure D-119.

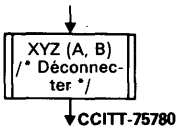


FIGURE D-119

Exemple d'emploi d'une syntaxe de commentaire LDS/PR en LDS/GR

D.6.3.6.19 *Extension de texte*

Le texte associé à un symbole graphique doit normalement être placé à l'intérieur du symbole en question. Ceci ne s'avère toutefois pas toujours possible ou pratique. L'on peut alors mettre le texte à l'intérieur d'un symbole d'extension de texte relié au symbole associé. Voir la figure D-120.

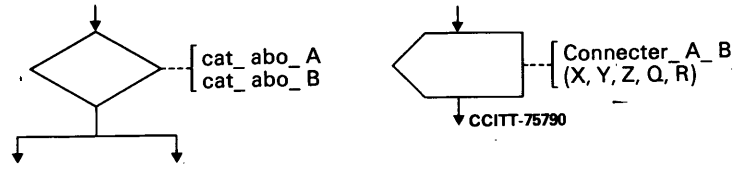


FIGURE D-120

Exemple d'emploi du symbole d'extension de texte

La syntaxe étant la même que celle du commentaire, il importe de tracer une ligne de connexion pleine, contrairement à la ligne du symbole de commentaire qui est pointillée.

Si l'on insère un texte informel dans les symboles d'extension de texte, il convient de le placer entre des apostrophes.

D.6.3.6.20 *Données*

On emploie généralement la syntaxe LDS/PR pour les constructions de données en LDS/GR. Le texte approprié LDS/PR est associé au symbole, en cas d'emploi de données dans des tâches, sorties, décisions ou entrées.

Les définitions des types de données et les déclarations de variables sont contenues dans des documents distincts cités en référence, ou dans le diagramme du processus. Ces définitions doivent être associées au symbole début du diagramme.

FIGURE D-121

Exemples de définitions de données dans un diagramme de processus LDS/GR

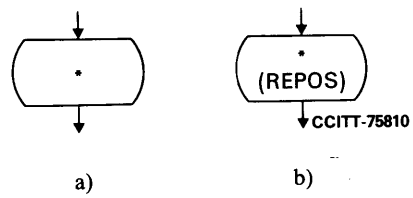
L'emploi du symbole d'extension de texte permet l'association avec le symbole, comme le montre la figure D-121.

D.6.3.6.21 *Notations abrégées*

Les notations abrégées suivantes sont employées afin de rendre les diagrammes plus lisibles et d'éviter de longues énumérations de noms. L'on fait appel aux symboles suivants: l'astérisque (*) et le tiret (–). Le «*» signifie généralement «tout» ou «tout sauf», et le « – » signifie «le(la) même».

L'on peut se servir des notations abrégées dans les symboles d'ÉTAT, d'ENTRÉE, de MISE EN RÉSERVE et d'ÉTAT SUIVANT.

Un astérisque, dans un symbole d'état, peut être utilisé seul ou accompagné d'une liste de noms d'états placée entre parenthèses.



a) b)

FIGURE D-122

Exemple d'emploi d'une notation « * » dans des états

Dans la figure D-122, l'astérisque seul a) signifie «tous les états»; l'astérisque accompagné b) doit être interprété comme «tous les états sauf REPOS». Cette notation fait appel à la possibilité d'apparitions multiples des symboles d'état (voir le § D.6.3.6.5).

Un astérisque ne peut être placé que dans l'une des entrées connectées à un état, à condition de ne pas figurer dans un symbole de mise en réserve connecté à cet état.

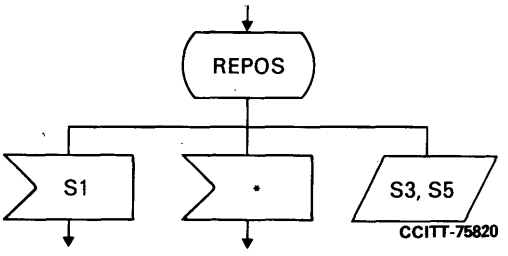


FIGURE D-123

Emploi d'un astérisque dans un symbole d'entrée

L'astérisque de la figure D-123 signifie: «tous les signaux sauf les signaux mentionnés dans d'autres entrées ou mises en réserve, à savoir S1, S3 et S5».

Les conditions d'emploi d'un astérisque dans un symbole de mise en réserve sont les mêmes. On ne peut placer un astérisque que dans l'une des mises en réserve connectées à un état à condition que cet astérisque ne figure dans aucune des entrées connectées à l'état en question.

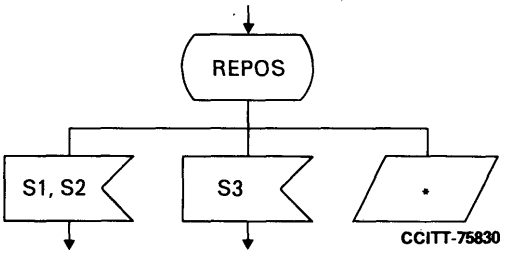


FIGURE D-124

Emploi d'un astérisque dans un symbole de mise en réserve

L'astérisque dans la figure D-124 signifie «tous les signaux sauf S1, S2 et S3».

Un tiret «–» placé dans un symbole d'état suivant peut signifier «le même état que l'état de départ de la transition».

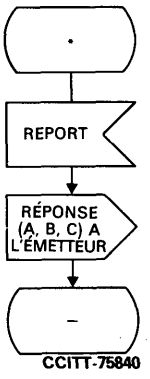


FIGURE D-125

Emploi du tiret dans un symbole d'état suivant

L'interprétation de la figure D-125 est que le signal «REPORT» peut être reçu par n'importe quel état du processus. A réception de ce signal, un signal «RÉPONSE» est émis, et la transition s'achève dans son état de départ.

L'association de ces notations abrégées avec la possibilité d'apparitions multiples offre une grande puissance d'expression. On remarquera que la simple adjonction à un diagramme de notes introduites par «*» et «–» peut modifier le sens de ce diagramme d'une façon inattendue.

Ainsi, par suite de la présence d'un addendum au diagramme de la figure D-126, tous les états de ce diagramme deviennent des transitions non implicites.

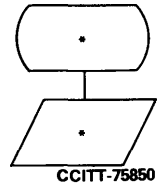


FIGURE D-126

L'une des conséquences de ce diagramme est que l'«*» n'est autorisé dans aucun symbole d'entrée, d'état ou de mise en réserve du diagramme

D.6.3.6.22 *Illustrations d'état*

D.6.3.6.22.1 *Choix des types des illustrations d'état*

Une fois que le type d'illustration d'état a été choisi, il est utile de considérer que chaque illustration comprend trois types d'éléments, qui sont indiqués à la figure D-127 a).

Ces trois types sont souvent combinés pour former un élément graphique composite qui peut être compris indépendamment du reste du graphique. La figure D-127 b), par exemple, donne la signification d'un récepteur de code multifréquence qui attend un signal vers l'avant.

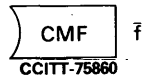
A noter que le symbole d'élément graphique recommandé pour surveiller le temps d'un processus comporte la variable d'entrée t_i correspondante.

1) Eléments graphiques: par exemple  (qui représente un récepteur de signalisation)

2) Variables d'entrée, par exemple $\bar{f}$ (qui signifie qu'un signal vers l'avant n'a pas encore été reconnu) et

3) Texte explicatif, par exemple CMF (qui signifie code multifréquence)

a) Contenu d'une illustration d'état



b) Elément graphique composite

FIGURE D-127

Construction d'éléments graphiques

D.6.3.6.22.2 *Variables de données*

Les variables de données sont très utiles pour représenter les conditions associées au processus qui, si elles changent, entraînent une transition dans ce dernier. Les variables d'entrée doivent être indiquées à l'aide de minuscules de façon à les distinguer facilement du texte explicatif (qui, lui, doit être indiqué à l'aide de lettres majuscules). (Les changements apportés aux variables d'entrée correspondent à des signaux d'entrée qui amènent le processus à quitter l'état dans lequel il se trouve; les modifications du texte explicatif ne correspondent pas à des signaux d'entrée.)

D.6.3.6.22.3 *Texte explicatif*

Le texte explicatif doit être considérablement abrégé et, si cela est possible, placé à l'intérieur d'éléments graphiques appropriés. L'on peut alors facilement savoir sur quels éléments graphiques porte l'explication.

D.6.3.6.22.4 *Illustrations d'état complètes*

Chaque illustration d'état doit comporter un nombre suffisant d'éléments graphiques afin de montrer:

a) quelles ressources le processus met en œuvre au cours de l'état représenté. Exemples: trajets de commutation, récepteurs de signalisation, émetteurs de signalisation et modules de commutation;

b) s'il y a en ce moment un ou plusieurs temporisateurs qui contrôlent le processus;

c) dans le cas où le processus concerne le traitement des appels, si la taxation est ou non actuellement en cours et quels abonnés sont taxés au cours de cette phase de l'appel;

d) les catégories d'équipement actuellement affectées à ce processus, là où cette information influe sur le comportement du processus;

e) le statut des variables d'entrée qui sont contrôlées par le processus pendant l'état considéré.

D.6.3.6.22.5 *Exemple*

Comme exemple de l'application des principes exposés ci-dessus, considérons l'illustration d'état de la figure D-128. On voit que, dans cet état:

a) les ressources affectées aux processus sont: un appareil d'abonné, un récepteur de chiffres, un émetteur de tonalités de numérotation et les trajets de commutation reliant ces organes;

b) un temporisateur T_0 (dont la condition actuelle est t_0) surveille le processus;

c) aucune taxation n'est en cours;

d) l'abonné est reconnu comme l'abonné A mais aucune autre information de catégorie n'est prise en considération;

e) les variables d'entrée en jeu sont les suivantes: h_A (combiné décroché), $\overline{d}$ (le récepteur attend un chiffre) et t_0 (le temporisateur de supervision T_0 fonctionne).

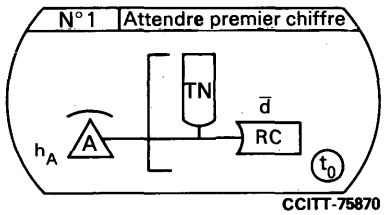


FIGURE D-128

Exemple d'un état au cours du processus de traitement d'un appel

D.6.3.6.22.6 *Vérification de la cohérence de diagrammes LDS avec éléments graphiques*

Si l'on suit le principe exprimé en e) du § D.6.3.6.22.4, on peut toujours trouver l'ensemble des entrées à la suite desquelles le processus quittera chacun de ses états, par un simple coup d'œil aux variables d'entrée indiquées sur l'illustration d'état. En regardant, par exemple, celles de la figure D-128, on voit que ces variables sont au nombre de trois: le passage du combiné à l'état raccroché (entrée $\overline{h_A}$), l'arrivée d'un chiffre (entrée d), l'expiration du délai de temporisation T_0 (entrée $\overline{t_0}$). On peut ainsi éviter toutes les «transitions surprise» lors de la mise en service de ce système aux spécifications LDS très complexes.

L'on constate que le graphique est plus compact et que, d'une certaine manière, il offre au lecteur plus d'informations; cependant, l'identification de la série exacte d'opérations accomplies au cours de la transition exige un examen très attentif du graphique.

En outre, une simple observation du graphique ne permet pas de dire si une action s'accomplit par l'intermédiaire d'une sortie ou d'une tâche (voir le § D.6.3.6.22.8).

D.6.3.6.22.7 *Utilisation du symbole «limites du bloc»*

Les éléments graphiques représentés à l'extérieur du bloc sont des éléments qui ne sont pas directement commandés par le processus donné; ceux qui sont représentés à l'intérieur du symbole «limites du bloc» sont directement commandés par ce processus. Par exemple, le processus d'appel partiellement spécifié dans la figure D-129 peut connecter ou déconnecter le courant d'appel; il peut également déclencher ou arrêter le temporisateur T4, mais il ne peut changer aucune des conditions du combiné de l'abonné.

En concevant la logique à partir d'une spécification de LDS avec éléments graphiques, seuls les éléments graphiques représentés à l'intérieur des limites du bloc ont une influence sur les actions exécutées pendant les séquences de transition. Les éléments graphiques complexes représentés à l'extérieur des limites de ce bloc sont normalement inclus dans une illustration d'état:

a) soit parce qu'ils indiquent des variables d'entrée devant être contrôlées par le processus pendant l'état donné,

b) soit pour améliorer l'intelligibilité du diagramme.

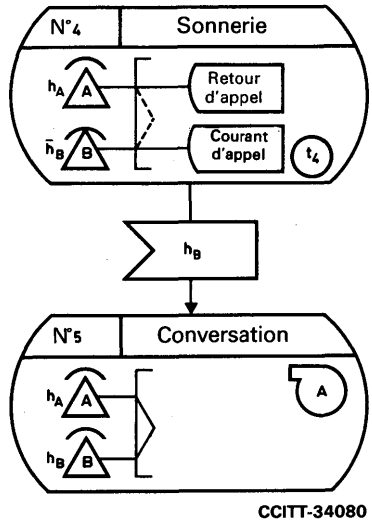


FIGURE D-129

Exemple d'une transition entre deux états, dans lequel toutes les actions de traitement découlent implicitement des différences entre les illustrations d'état

D.6.3.6.22.8 *Tâche ou sortie*

L'interprétation d'une action de traitement, qu'il s'agisse d'une tâche ou d'une sortie, semble quelquefois arbitraire. En fait, la décision d'interpréter l'apparition ou la disparition d'un élément graphique dans les limites d'un bloc fonctionnel soit comme une tâche, soit comme une sortie, ne peut être prise qu'après consultation de la liste des signaux du processus.

D.6.3.6.22.9 *Utilisation du symbole «temporisateur»*

Que l'on emploie ou non des éléments graphiques, l'interruption du processus surveillé à l'expiration d'un délai de temporisation est toujours représentée par une entrée.

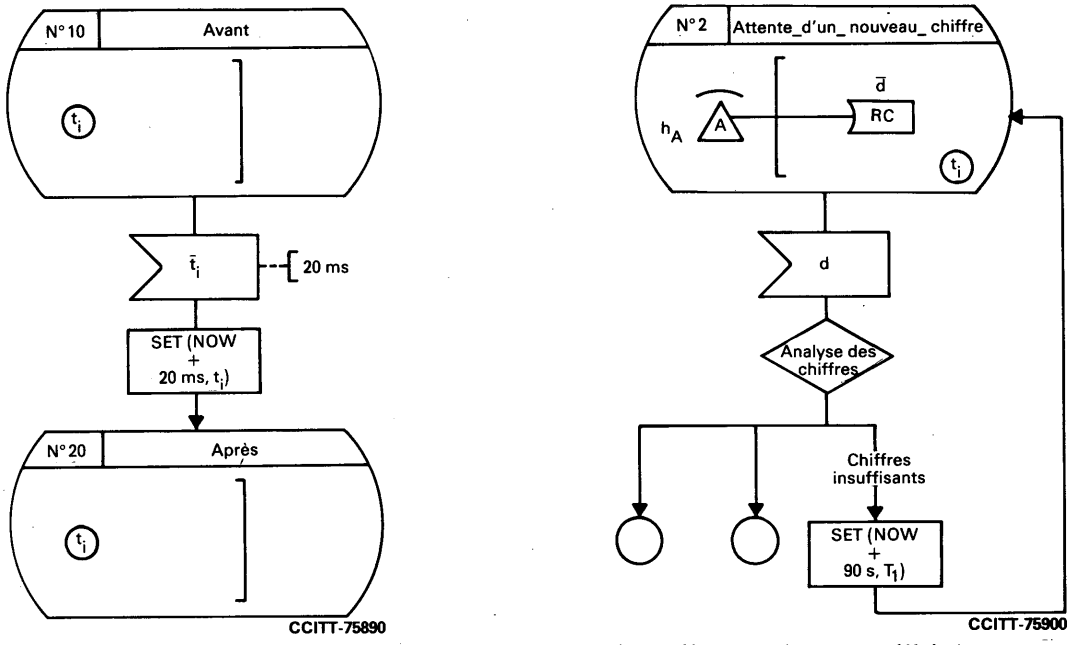
La présence d'un symbole de temporisateur dans une illustration d'état implique qu'un temporisateur fonctionne pendant cet état. Conformément au principe général exposé dans les Recommandations, le démarrage, l'arrêt, le redémarrage et l'expiration du délai d'une temporisation sont représentés à l'aide d'éléments graphiques de la manière suivante:

a) pour montrer qu'une temporisation commence au cours d'une transition donnée, le symbole temporisateur doit apparaître sur l'illustration d'état qui correspond à la fin de cette transition et non sur celle qui correspond à son début;

b) inversement, pour montrer qu'une temporisation s'arrête au cours d'une transition, le symbole temporisateur doit apparaître sur l'illustration d'état qui correspond au début de cette transition et non sur celle qui correspond à sa fin;

c) pour montrer qu'une temporisation est relancée au cours d'une transition, un symbole explicite de tâche doit être représenté dans cette transition (on en voit deux exemples sur la figure D-130);

d) l'expiration du délai d'une temporisation donnée est représentée par un symbole d'entrée associé à un état dont l'illustration porte le symbole «temporisateur» correspondant. Il peut naturellement arriver que plus d'un temporisateur surveille à la fois le même processus (voir la figure D-131).



a) Redémarrage à la suite de l'expiration du délai de temporisation

b) Redémarrage lorsque ce délai n'est pas encore expiré

Remarque – Chaque temporisateur T_i a deux états qui s'excluent mutuellement t_i et $\bar{t}_i$.

FIGURE D-130

Exemples montrant le redémarrage d'un temporisateur

D.6.3.6.23 *Documents auxiliaires*

Des documents auxiliaires offrant des vues d'ensemble facilitent la lecture et la compréhension de diagrammes de processus importants. Ces documents sont informels et se contentent de donner des vues d'ensemble; ce sont des documents «préliminaires» aux diagrammes LDS. Les usagers choisissent librement la syntaxe qui leur convient.

Le présent paragraphe donne des exemples concernant deux types de documents auxiliaires: le diagramme synoptique d'état et la matrice état/signal.

D.6.3.6.23.1 *Diagramme synoptique d'état*

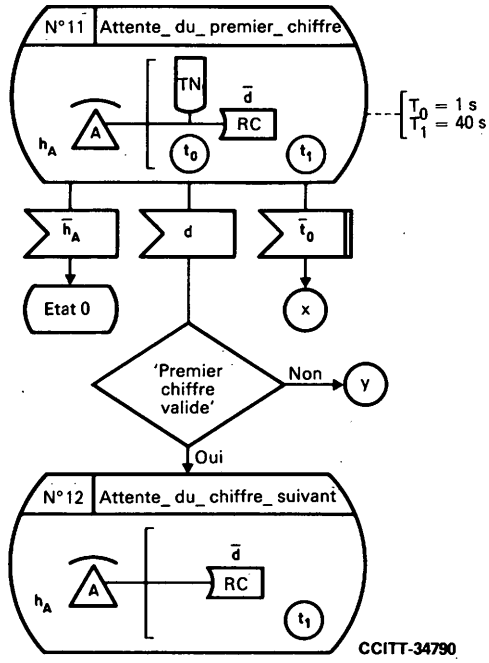
Son objectif est de donner une vue d'ensemble des états d'un processus, et d'indiquer quelles transitions sont possibles entre eux. Etant donné qu'il s'agit de donner un aperçu, l'on peut négliger les états ou les transitions de peu d'importance.

Les diagrammes se composent de symboles d'état, de flèches représentant des transitions et, éventuellement, de symboles de début et d'arrêt.

Le symbole d'état doit indiquer le nom de l'état référencé. Plusieurs noms d'états peuvent être inscrits dans le symbole, et il est possible d'employer un astérisque (*) pour désigner tous les états.

Chacune des flèches peut recevoir le nom du signal ou de l'ensemble de signaux qui déclenchent la transition.

L'emploi des symboles de début et d'arrêt est facultatif.



T_0 surveille l'arrivée du premier chiffre, à la suite de quoi la tonalité de numérotation est éliminée et T_0 arrêté.
T_1 continue à surveiller l'arrivée d'un nombre de chiffres suffisant pour que l'appel soit convenablement aiguillé.

FIGURE D-131

Exemple d'utilisation de deux temporisateurs de surveillance dans le même état

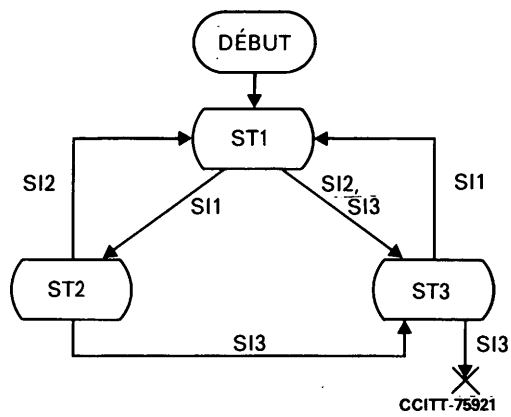


FIGURE D-132

Exemple de diagramme synoptique d'état

Il est possible de répartir sur plusieurs diagrammes le diagramme synoptique d'état d'un *processus;* chacun des diagrammes obtenus porte alors sur un aspect particulier, comme «cas normal», traitement en cas de défaillance, etc.

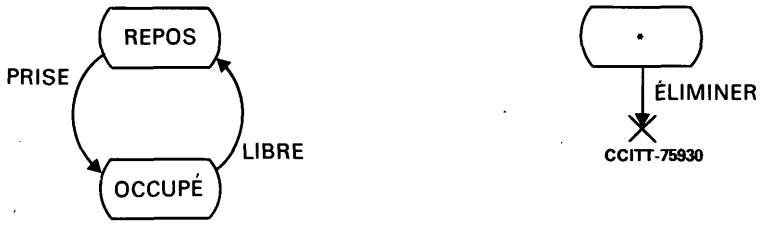


FIGURE D-133

Exemple de diagramme synoptique d'état réparti

Afin que les diagrammes synoptiques d'état soient bien structurés et d'une lecture facile, ils doivent de préférence:

- respecter les sens habituels de lecture: de haut en bas et de gauche à droite;
- tenir sur une seule page.

D.6.3.6.23.2 *Matrice état/signal*

La matrice état/signal doit servir de document «préliminaire» à un diagramme de processus important. Elle indique les endroits où existent des combinaisons entre un état et un signal dans le diagramme.

Le diagramme se compose d'une matrice bidimensionnelle; celle-ci présente sur un axe tous les états d'un processus, et sur l'autre axe tous les signaux d'entrée valides d'un processus. Une référence donne pour chaque élément de la matrice où trouver la combinaison donnée par les indices, si elle existe.

ÉTATS / SIGNAUX	REPOS	OCCUPÉ	BLOQUÉ	------
DÉCROCHÉ	P5	–	–	
RACCROCHÉ	–	P6	–	
PRISE	P7	P8	–	
BLOC	P6	P6	–	

CCITT-75940

FIGURE D-134

Exemple de matrice état/signal

Il est possible de fractionner la matrice en sous-parties réparties sur plusieurs feuilles. Les références sont celles qu'emploie normalement l'usager dans la documentation.

Les signaux et les états doivent être de préférence regroupés de façon que chaque partie de la matrice porte sur un aspect particulier du comportement du processus, ce qui revient à dire que le «cas normal», le «traitement d'exception», la «partie de maintenance», etc., doivent tous se trouver dans des sous-groupes différents.

D.6.3.6.23.3 *Diagrammes de séquencement*

Le diagramme de séquencement peut s'ajouter au diagramme d'interaction pour montrer l'échange des séquences de signaux autorisées entre un ou plusieurs processus et leur environnement.

Ce diagramme doit donner une vue d'ensemble de l'échange de signaux entre les parties du système. Ce diagramme peut représenter l'ensemble ou une partie de l'échange de signaux, en fonction des aspects à mettre en évidence.

La colonne du diagramme indique l'environnement (généralement précisé aux extrémités du diagramme) ainsi que les divers processus ou blocs.

L'ensemble des diagonales fléchées illustre leurs interactions, chaque droite représentant un signal ou un ensemble de signaux. Lorsque l'ensemble de signaux représenté par une droite se dirige dans les deux sens, il serait préférable de tracer la ligne horizontalement (citons à titre d'exemple la «constitution d'une communication»); cette ligne doit être fléchée aux deux extrémités.

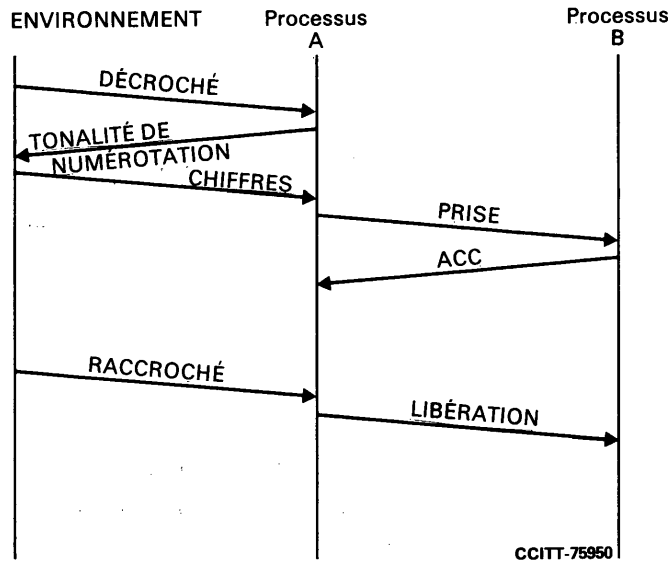


FIGURE D-135

Exemple de diagramme de fonctionnement

On peut annoter chaque séquence afin de faire apparaître clairement l'ensemble d'informations échangé. Chaque ligne est accompagnée d'une annotation qui donne les renseignements requis (noms des signaux ou de procédures).

On peut placer un symbole de décision au point d'aboutissement d'une ligne pour indiquer que la séquence suivante est valide si la condition indiquée est vraie. Dans ce cas, le symbole de décision apparaît généralement plusieurs fois; il indique les différentes séquences produites par chacune des valeurs de la condition.

Ce diagramme peut représenter la totalité ou seulement un sous-ensemble significatif des séquences de signaux échangées.

La représentation de l'interaction réciproque des sous-processus engendrés par la subdivision d'un processus représente une application utile de ce type de diagramme. Dans ce cas, les sous-processus ont pour environnement l'environnement du processus subdivisé.

D.7 *Directives pour la représentation de systèmes utilisant le LDS/PR*

Le § 7 des Directives pour l'usager explique la représentation textuelle de phrase du LDS, appelée PR, qui ressemble fortement à un langage de programmation.

La première section (§ D.7.1) contient des remarques générales sur le LDS/PR, sur son aptitude à servir d'entrée pour une machine, sur sa ressemblance à un programme et ce en quoi il en diffère.

La deuxième section (§ D.7.2) explique comment le LDS/PR est défini pour l'ensemble des Recommandations où on utilise les «diagrammes de syntaxe».

La troisième section (§ D.7.3) est la plus importante pour l'usager. Elle expose les aspects pratiques de l'emploi du LDS. On y étudie notamment les particularités et la puissance expressive de la représentation textuelle de phrase en s'attachant à des questions telles que l'ordre des états, la séquence des transitions à l'intérieur d'un état, la multiplicité des états, les procédures, les macros, les étiquettes et les notations abrégées.

D.7.1 *Raison d'être du PR*

La représentation textuelle de membres de phrase du LDS, LDS/PR, a été mise au point pendant la période d'études 1977-1980; en 1980, il a été décidé d'en joindre la définition en annexe aux Recommandations, des perfectionnements semblant devoir y être apportés avant qu'elle puisse faire l'objet d'une Recommandation. Ces perfectionnements ont été apportés pendant la période d'études qui a suivi et, aujourd'hui, le LDS/PR est une des syntaxes concrètes recommandées pour le LDS.

A l'origine, le LDS/PR devait constituer un moyen commode d'introduire dans une machine des documents établis en LDS, l'introduction des documents établis en GR étant plus difficile (des appareils périphériques à représentation graphique sont alors nécessaires). C'est pour cette raison qu'on s'est attaché à établir une corrélation biunivoque entre la forme PR et la forme GR. L'évolution vers des terminaux à graphiques (offrant des possibilités accrues et réduisant les coûts) a rendu, depuis lors, la forme GR utilisable dans une machine. Cette possibilité ne diminue en rien l'importance de la forme PR que certains usagers, notamment ceux qui travaillent avec des langages de programmation, continuent à employer car ils le trouvent plus à leur convenance.

D.7.1.1 *Similitude entre le LDS/PR et un programme*

L'évolution a abouti à une corrélation plus lâche entre le GR et le PR en sorte qu'il est encore possible (et facile) d'établir une correspondance entre l'un et l'autre mais chacun a des particularités propres. A première vue, la forme PR ressemble beaucoup à un langage de programmation. Voir la figure D-136.

```
–
–
–
–
STATUS raccroché;
INPUT raccroché;
TASK déclencher taxation;
TASK connecter;
OUTPUT réinitialiser temporisateur;
NEXTSTATE conversation
–
–
–
–
```

FIGURE D-136

Exemple de LDS/PR

En réalité, tout dépend de ce qui fait qu'un texte soit un langage de programmation.

Si nous posons comme hypothèse qu'un programme est défini par un ensemble d'informations qu'une machine peut interpréter, alors PR et GR sont tous deux des programmes. Si l'on restreint la définition à un ensemble d'informations et de directives qu'une machine peut interpréter et exécuter, nous constatons une première différence: il n'est pas indispensable qu'une représentation en PR puisse être exécutée par une machine (encore que cela ne soit pas interdit), l'essentiel c'est qu'elle puisse transmettre des informations précises d'une personne à une autre.

Ce qui, si on l'envisage en tant que programme, peut sembler être «PR erroné» (parce qu'il est incomplet ou que le texte ne respecte pas les formes prescrites) est un PR parfaitement valable si l'on considère qu'il représente les caractéristiques de fonctionnement d'un système.

Il existe en outre une différence de style si l'on compare une représentation en PR et une représentation à l'aide du programme ordinaire.

Le PR étant conçu pour permettre à des personnes de communiquer entre elles, on a pris soin de ménager la possibilité de recourir à différents types de présentation; le lecteur pourra donc s'attacher à certains aspects qu'il considère plus importants que d'autres grâce à la présentation PR. Ces considérations sont évidemment sans importance pour un programme fait pour être interprété par une machine. La machine, pour sa part, ne s'attache à aucun aspect particulier; elle doit considérer le tout d'une manière égale et elle ne cherche pas à comprendre le programme.

C'est parce qu'il ressemble à un programme (similitude qui relève plutôt de la psychologie puisque la forme GR ressemble autant sémantiquement à un programme que le PR) que certains programmeurs préfèrent le PR, alors qu'ils emploieront le CHILL pour donner suite aux consignes exprimées en PR. La tentation est donc grande de retrouver une concordance biunivoque entre le PR et le CHILL pour pouvoir transformer automatiquement en code CHILL les consignes exprimées en PR. L'opération inverse présente également un intérêt puisqu'elle permettrait d'établir une description en PR à partir d'un programme CHILL.

Le § D.8 expose diverses possibilités de mise en correspondance du LDS et du CHILL.

D.7.2 *Métalangage pour décrire la syntaxe du LDS/PR: diagrammes syntaxiques*

Un diagramme syntaxique se compose de symboles terminaux et de symboles non terminaux reliés entre eux par des lignes de liaison.

Un symbole non terminal représente un autre diagramme syntaxique ayant le même nom. C'est un symbole abrégé qui représente des structures plus complexes, utilisées en divers endroits (structures composées là encore de symboles terminaux et de symboles non terminaux).

Il existe une notation ad hoc pour insérer les commentaires.

Chaque symbole, c'est-à-dire chaque diagramme, doit n'avoir qu'une seule entrée et qu'une seule sortie.

Les lignes de liaison sont unidirectionnelles et une flèche indique le sens du cheminement.

Les symboles terminaux sont indiqués par un rectangle dans lequel les côtés verticaux ont été remplacés par des demi-cercles. Les symboles non terminaux sont indiqués par un rectangle.

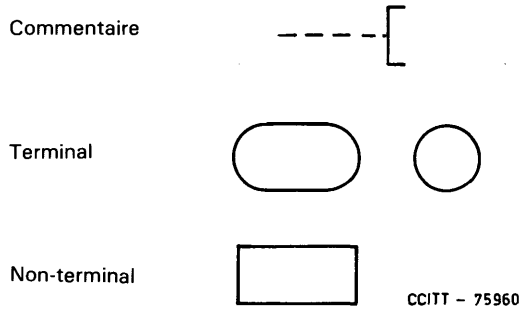


FIGURE D-137

Symboles graphiques dans les diagrammes syntaxiques

D.7.3 *Utilisation du PR*

Dans le LDS/GR, le système complet est décrit dans plusieurs documents, ce qui rend le traitement de l'information plus facile. Le diagramme d'interaction des blocs représente la structure du système et la communication entre les blocs et entre les processus au moyen de canaux et de signaux, alors que les diagrammes de processus sont représentés dans d'autres documents.

Dans le PR, la structure du système et ses communications peuvent être représentées en découpant le programme complet PR du système en différents modules, chaque module décrivant la structure d'un ou plusieurs processus. De cette façon, le traitement de chaque module en est simplifié.

Entre ces modules, il peut y avoir un mécanisme de référence tel que décrit au § D.5.

La version PR se compose d'un ensemble d'énoncés, chacun terminé par «;» (point virgule). Elle commence par l'énoncé

SYSTEM nom;

et finit par l'énoncé

ENDSYSTEM nom;

Dans cet énoncé le mot nom est facultatif.

La représentation en PR ne doit pas nécessairement se composer simplement d'un ensemble d'énoncés inclus entre les énoncés SYSTEM-ENDSYSTEM. On peut avoir plusieurs ensembles commençant par un énoncé SYSTEM et se terminant par un énoncé ENDSYSTEM. Ils n'appartiennent à la même représentation de SYSTEM que si et seulement si ils portent le même nom (voir la figure D-138). Chaque module de processus doit figurer dans un seul ensemble, il ne peut être réparti sur deux ensembles.

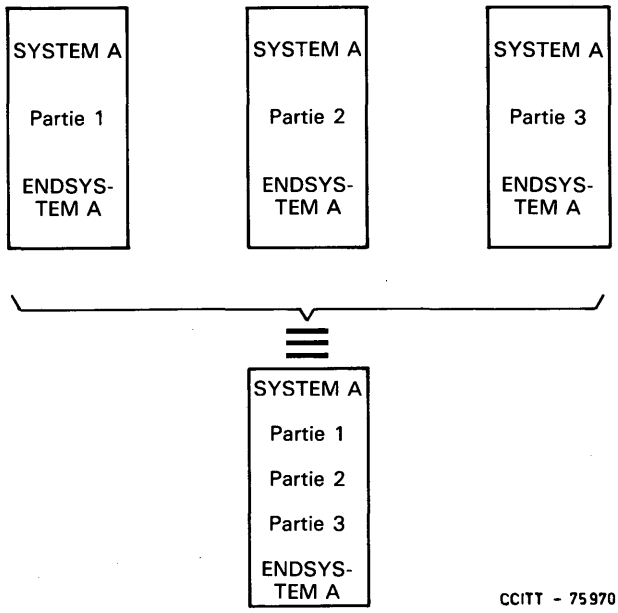


FIGURE D-138

Séparation en trois parties d'une représentation de système

On peut grouper les énoncés compris entre SYSTEM et ENDSYSTEM en modules de bloc et en définitions des signaux, canaux, types de données, macros et procédures comme le montre la figure D-139.

SYSTEM .....

Définitions des signaux

Définitions des canaux

Définitions des types de données

Définitions des macros

Définitions des procédures

BLOCK...
ENDBLOCK } Module de bloc

BLOCK...
ENDBLOCK } Module de bloc

ENDSYSTEM ....

FIGURE D-139

Groupement des énoncés contenus dans un module de système

Chaque module de bloc est encadré par les énoncés BLOCK et ENDBLOCK. Comme pour une représentation des systèmes, une représentation des blocs peut se subdiviser en parties. Là encore nous pouvons considérer chaque module de BLOC comme un ensemble de modules de PROCESSUS plus des définitions des signaux, données, procédures et macros. Chaque module est encadré par les énoncés PROCESS et ENDPROCESS. Le module PROCESS ne peut plus être subdivisé. C'est ce qu'illustrent les exemples de la figure D-140 où un système est divisé en deux parties, l'une contenant le bloc b1 avec le processus p2, le bloc b2 avec le processus p4 et le bloc b3 avec le processus p5 [figure D-140a)] et l'autre contenant le bloc b1 avec les processus p1 et p2 et le bloc b3 avec le processus p6 [figure D-140b)]. La figure D-140c) donne une représentation globale de ces deux parties.

```
SYSTEM a;
BLOCK b1;
  PROCESS p2;
  –
  –
  –
  ENDPROCESS p2;
ENDBLOCK b1;
BLOCK b2;
  PROCESS p4;
  –
  –
  –
  ENDPROCESS p4;
ENDBLOCK b2;
BLOCK b3;
  PROCESS p5;
  –
  –
  –
  ENDPROCESS p5;
ENDBLOCK b3;
ENDSYSTEM a;
```

a) Partie 1

```
SYSTEM a;
  BLOCK b1; à
    PROCESS p1;
    –
    –
    –
    ENDPROCESS p1;
    PROCESS p3;
    –
    –
    –
    ENDPROCESS p3;
  ENDBLOCK b1;
  BLOCK b3;
    PROCESS p6;
    –
    –
    –
    ENDPROCESS p6;
  ENDBLOCK b3;
ENDSYSTEM a;
```

b) Partie 2

```
SYSTEM a;
  BLOCK b1;
    PROCESS p1;
    –
    –
    –
    ENDPROCESS p1;
    PROCESS p2;
    –
    –
    –
    ENDPROCESS p2;
    PROCESS p3;
    –
    –
    –
    ENDPROCESS p3;
  ENDBLOCK b1;
  BLOCK b2;
    PROCESS p4;
  –
    –
    –
    ENDPROCESS p4;
  ENDBLOCK b2;
  BLOCK b3;
    PROCESS p5;
    –
    –
    ENDPROCESS p5;
    PROCESS p6;
    –
    –
    ENDPROCESS p6;
  ENDBLOCK b3;
ENDSYSTEM a;
```

c) Global

FIGURE D-140

Représentation de bloc

D.7.3.1 *Expression de la structure dans PR*

Les documents utiles dans GR sont les arborescences de bloc, les arborescences de processus et les diagrammes de sous-structures des canaux. Il est possible d'obtenir des représentations équivalentes dans PR en utilisant les blocs et les sous-structures de canaux, comme expliqué dans la Recommandation Z.102.

Dans PR, l'arborescence de bloc, l'arborescence de processus et les diagrammes d'interaction des blocs sont regroupés. La structure ainsi obtenue contient également les énoncés des processus, c'est-à-dire qu'elle représente le système complet.

La relation directe avec ces documents est fournie par la liste des sous-blocs, canaux et signaux qui peuvent être incorporés dans la partie de sous-structure de bloc, alors que la structure du système est fournie par les définitions de la série des blocs.

Les figures D-141 et D-142 représentent l'arborescence de bloc et l'arborescence de processus du GR auquel se réfèrent les exemples PR de la figue D-143.

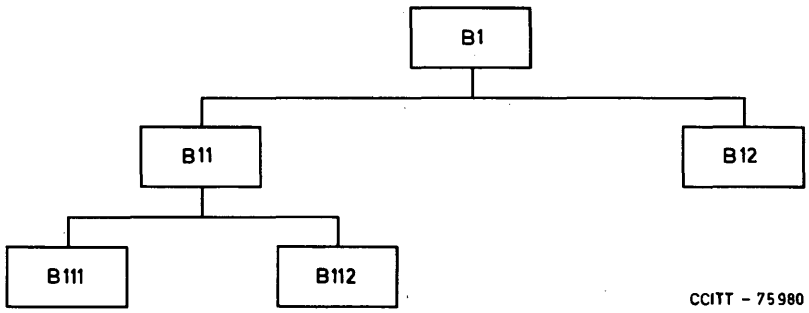


FIGURE D-141

Arborescence de bloc

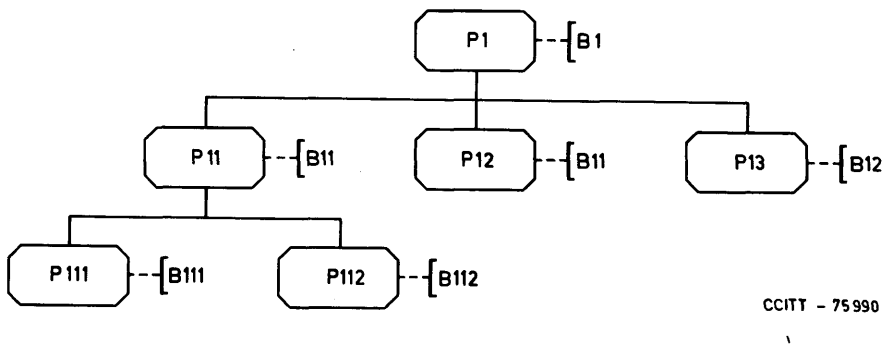


FIGURE D-142

Arborescence de processus

```
SYSTEM J;
- - - -
BLOCK B1;
- - - -
SUBSTRUCTURE B1;
SUBBLOCKS: B11, B12;
- - - -
BLOCK B11;
- - - -
SUBSTRUCTURE B11;
SUBBLOCKS: B111, B112;
- - - -
BLOCK B111;
- - - -
ENDBLOCK B111;
BLOCK B112;
- - - -
ENDBLOCK B112;
ENDSUBSTRUCTURE B11;
ENDBLOCK B11;
BLOCK B12;
- - - -
ENDBLOCK B12;
ENDSUBSTRUCTURE B1;
ENDBLOCK B1;
ENDSYSTEM J;
```

FIGURE D-143

Exemple d'une définition d'une série de blocs

Dans PR, les processus et leur structure sont représentés à l'intérieur de leurs blocs. On peut représenter la sous-structure du processus selon deux méthodes: au niveau supérieur de la sous-structure de bloc ou à l'intérieur de chaque processus du niveau supérieur. Les figures D-144 et D-145 illustrent ces deux méthodes.

```
SYSTEM s;
- - - -
BLOCK b1;
- - - -
PROCESS p1;
- - - -
ENDPROCESS p1;
SUBSTRUCTURE b1;
SUBBLOCKS: b11, b12;
- - - -
SUBSTRUCTURE p1;
  p11 in b11,
  p12 in b12,
ENDSUBSTRUCTURE p1;
BLOCK b11;
- - - -
PROCESS p11;
- - - -
ENDPROCESS p11;
SUBSTRUCTURE b11;
SUBBLOCKS: b111, b112;
- - - -
SUBSTRUCTURE p11;
  p111 in b111,
  p112 in b112,
ENDSUBSTRUCTURE p11;
BLOCK b111;
- - - -
PROCESS p111;
- - - -
ENDPROCESS p111;
ENDBLOCK b111;
BLOCK b112;
- - - -
PROCESS p112;
- - - -
ENDPROCESS p112;
ENDBLOCK b112;
ENDSUBSTRUCTURE b11;
ENDBLOCK b11;
BLOCK b12;
- - - -
PROCESS p12;
- - - -
ENDPROCESS p12;
ENDBLOCK b12;
ENDSUBSTRUCTURE b1;
ENDBLOCK b1;
ENDSYSTEM s;
```

FIGURE D-144

Représentation du système avec sous-structure du processus au niveau de la sous structure du bloc

```
SYSTEM s;
- - - -
BLOCK b1;
- - - -
PROCESS p1;
- - - -
SUBSTRUCTURE p1;
  p11 in b11,
  p12 in b12,
ENDSUBSTRUCTURE p1;
ENDPROCESS p1;
SUBSTRUCTURE b1;
SUBBLOCKS: b11, b12;
- - - -
BLOCK b11;
- - - -
PROCESS p11;
- - - -
SUBSTRUCTURE p11;
  p111 in b111;
  p112 in b112;
ENDSUBSTRUCTURE p11;
ENDPROCESS p11;
SUBSTRUCTURE b11;
SUBBLOCKS: b111, b112;
- - - -
BLOCK b111;
- - - -
PROCESS p111;
- - - -
ENDPROCESS p111;
ENDBLOCK b111;
BLOCK b112;
- - - -
PROCESS p112;
- - - -
ENDPROCESS p112;
ENDBLOCK b112;
ENDSUBSTRUCTURE b11;
ENDBLOCK b11;
BLOCK b12;
- - - -
PROCESS p12;
- - - -
ENDPROCESS p12;
ENDBLOCK b12;
ENDSUBSTRUCTURE b1;
ENDBLOCK b1;
ENDSYSTEM s;
```

FIGURE D-145

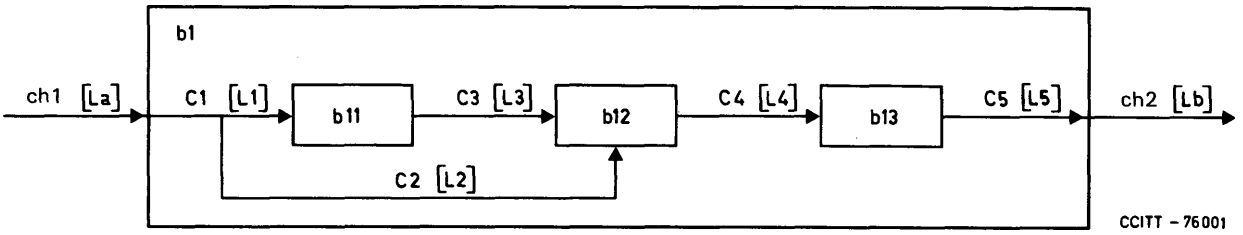
Représentation du système avec sous-structure du processus à l'intérieur de la définition de processus

Les canaux qui relient les sous-blocs obtenus après une découpe sont définis dans la définition des sous-structures des blocs.

Il est préférable de placer la définition des canaux avant la définition des sous-blocs, en raison du nombre considérable de blocs contenus dans le système (voir la figure D-146).

Les signaux de ces canaux sont définis dans la partie relative à la définition des signaux.

La figure D-147 se réfère à la représentation GR de la figure D-146.



$$La = \begin{bmatrix} s1, \\ s2 \end{bmatrix} \qquad Lb = \begin{bmatrix} s3, \\ s4 \end{bmatrix}$$

$$L1 = [s1] \qquad L2 = [s2]$$

$$L3 = \begin{bmatrix} s5, \\ s6 \end{bmatrix} \qquad L4 = \begin{bmatrix} s7, \\ s8 \end{bmatrix}$$

$$L5 = \begin{bmatrix} s3, \\ s4 \end{bmatrix}$$

FIGURE D-146

Une représentation des systèmes à deux niveaux avec utilisation du diagramme d'interaction des blocs

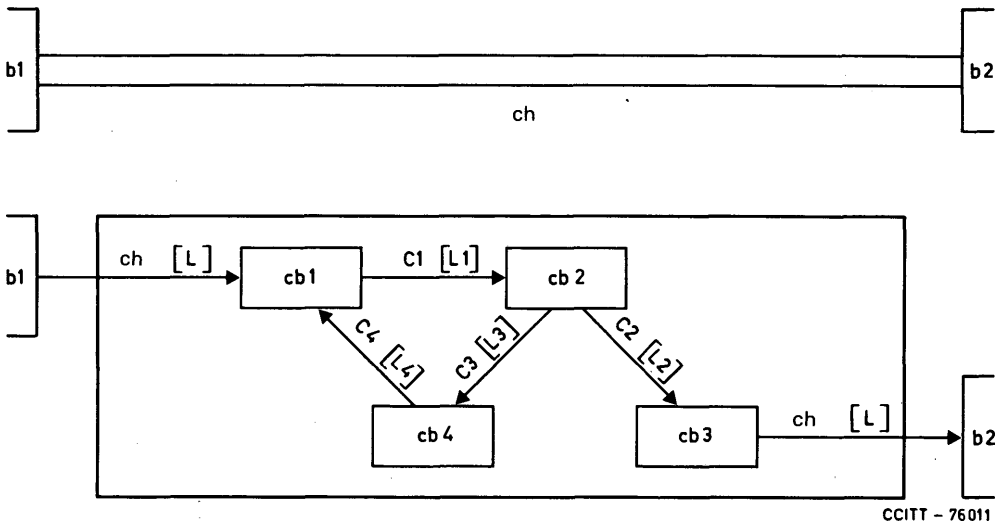
```
SYSTEM s;
CHANNEL ch1
  FROM ENV TO b1
  WITH s1, s2;
CHANNEL ch2
  FROM b1 TO ENV
  WITH s3, s4;
BLOCK b1;
- - - -
SUBSTRUCTURE b1;
SUBBLOCKS: b11, b12, b13;
CHANNELS: C1, C2, C3, C4, C5
  SPLIT ch1 INTO C1, C2;
  SPLIT ch2 INTO C5;
CHANNEL C1
  FROM ENV TO b11
  WITH s1;
CHANNEL C2
  FROM ENV TO b12
  WITH s2;
CHANNEL C3
  FROM b11 TO b12
  WITH s5, s6;
CHANNEL C4
  FROM b12 TO b13
  WITH s7, s8;
CHANNEL C5
  FROM b13 TO ENV
  WITH s3, s4;
    SIGNAL - - - -;
BLOCK b11;
- - - -
ENDBLOCK b11;
BLOCK b12;
- - - -
ENDBLOCK b12;
BLOCK b13;
- - - -
ENDBLOCK b13;
ENDSUBSTRUCTURE b1;
ENDSYSTEM s;
```

FIGURE D-147

Représentation des systèmes à deux niveaux avec utilisation des canaux

Les définitions de canaux se situent, soit au niveau du système (pour la définition des canaux compris entre les blocs composant le système), soit dans la définition de la sous-structure des blocs (pour la définition des canaux compris entre les sous-blocs). Dans GR, le diagramme décrivant la décomposition d'un canal est appelé diagramme de sous-structure de canaux, dans PR, il correspond à une description à l'intérieur de la définition des canaux et également à la définition des blocs et des canaux du niveau inférieur.

La figure D-148 illustre un exemple d'un canal entre deux blocs vu à deux niveaux différents (le second niveau correspond au niveau inférieur et est plus détaillé).



$$L = \begin{bmatrix} s1, \\ s2, \\ s3, \\ s4 \end{bmatrix}$$

$$L1 = [s5]$$
$$L2 = [s6]$$
$$L3 = [s7]$$
$$L4 = [s8]$$

FIGURE D-148

Représentation d'un canal avec blocs et canaux

Dans la partie inférieure de cette figure, les blocs b1 et b2 ne sont pas décomposés en détail, mais la structure interne du canal ch, représenté en tant que système au niveau supérieur, figure dans la représentation au niveau inférieur. Il est préférable de montrer la décomposition du canal, au moment où les blocs de systèmes ne peuvent être décomposés plus en détail, sinon l'interaction respective entre les blocs des canaux et les sous-blocs des blocs devra également être représentée, ce qui augmenterait les difficultés de représentation de la structure.

La figure D-149 illustre la représentation PR de cette décomposition de canaux de la figure D-148.

```
SYSTEM s;
- - - -
CHANNEL ch
  FROM b1 TO b2
  WITH s1, s2, s3, s4;
SUBSTRUCTURE ch;
BLOCKS: cb1, cb2, cb3, cb4;
CHANNELS: C1, C2, C3, C4;
- - - -
INCOMING ch to cb1;
OUTGOING ch FROM cb3;
CHANNEL C1
  FROM cb1 TO cb2
  WITH s5;
CHANNEL C2
  FROM cb2 TO cb3
  WITH s6;
CHANNEL C3
  FROM cb3 to cb4
  WITH s7;
CHANNEL C4
  FROM cb4 TO cb1
  WITH s8;
SIGNAL
- - - -;
BLOCK cb1;
- - - -
ENDBLOCK cb1;
BLOCK cb2;
- - - -
ENDBLOCK cb2;
BLOCK cb3;
- - - -
ENDBLOCK cb3;
BLOCK cb4;
- - - -
ENDBLOCK cb4;
ENDSUBSTRUCTURE ch;
BLOCK b1;
- - - -
ENDBLOCK b1;
BLOCK b2;
- - - -
ENDBLOCK b2;
ENDSYSTEM s;
```

FIGURE D-149

Exemple de découpe de canal

Dans tous les exemples représentés dans ce document, il est possible d'utiliser la notation macro pour décrire, à un autre endroit, les blocs, les processus, les canaux, lorsqu'ils sont trop longs ou trop complexes. Il suffira de placer un appel de macro à l'endroit où est supposée être la description. La définition de macro contient alors l'information.

D.7.3.2 *Définition des signaux*

Les signaux peuvent être définis au niveau du système, au niveau du bloc ou dans la partie interne d'une définition de processus. Les signaux définis au niveau du système représentent les signaux échangés avec l'environnement et entre les blocs du système. Les signaux définis au niveau du bloc représentent des signaux échangés entre les processus du même bloc.

Dans la figure D-150, les signaux SIG1, SIG2 sont listés sur un canal reliant les blocs b1 et B2 ou sur un canal reliant un de ces blocs à l'environnement. Les signaux s1, s2, s3 déclarés dans le bloc 1 sont utilisés par les processus appartenant à ce bloc. Les signaux s1, s2, s4, s5 déclarés dans le bloc B2 sont différents de ceux déclarés dans b1 et ils peuvent être utilisés par les processus du bloc B2.

Dans un système structuré, il peut y avoir des signaux définis au niveau du bloc qui représentent un échange de signaux entre les blocs au niveau inférieur (§ D.7.3.1). La définition de canaux peut également comprendre la définition des signaux des canaux du niveau inférieur générés par la découpe du canal en question (§ D.7.3.1). NB: la référence au § D.3.1 doit être insérée! Les séries de signaux des sous-canaux doivent être disjointes.

```
SYSTEM a;
    .
  SIGNAL SIG1(ty1, ty2), SIG2(ty2, ty3);
    .
  BLOCK b1;
      .
      .
      .
    SIGNAL s1(t1,t2,t3),s2,s3(t4,t2);
      .
  ENDBLOCK b1;
  BLOCK B2;
      .
      .
      .
    SIGNAL s1(t1,t2,t3),s2,s4,s5(t4,t2);
      .
      .
  ENDBLOCK B2;
ENDSYSTEM a;
```

FIGURE D-150

Exemple de définitions de signaux à divers niveaux

D.7.3.3 *Définition des canaux*

Les canaux doivent être définis au niveau du système; ou, si le système est structuré, également au niveau du bloc. Finalement, le canal peut être défini à l'intérieur d'une définition des canaux. Les canaux définis au niveau du système représentent les canaux entre les blocs du système et entre ces blocs et l'environnement.

Dans la figure D-151, les canaux c1 et c2 sont utilisés par le bloc b1 pour sa communication avec l'environnement.

Si le système est structuré, chaque bloc peut contenir la définition des canaux qui permet la communication entre les blocs produits par la découpe du bloc considéré et entre ces blocs et l'environnement (§ D.7.3.1).

Dans la définition des canaux, il peut y avoir des définitions de canaux du niveau inférieur obtenues grâce à la décomposition du canal du niveau inférieur (voir le § D.7.3.1). La définition des canaux contient la liste des signaux du canal qui doivent être définis dans la section de définition des signaux.

```
SYSTEM a;
    .
    .
  CHANNEL    c1 FROM b1 TO ENV
             WITH s1,s2,s3
             REFINEMENT CHANNELS: c1.1,c1.2;
  CHANNEL    c2 FROM ENV TO b1
             WITH s4,s5
             REFINEMENT CHANNELS: c2.1,c2.2;
    .
    .
    .
ENDSYSTEM a;
```

FIGURE D-151

Exemple de définitions de canaux

D.7.3.4 *Définition des données*

Conformément à la Recommandation Z.101, le LDS dispose de types de données prédéfinis qui sont disponibles pour chaque système. Les données sont les suivantes: entier, réel, caractère, chaîne, booléen, identificateur d'instance de processus, temps, durée. Elles ne nécessitent pas une déclaration et peuvent être utilisées dans une définition de variables avec leurs noms prédéfinis, à savoir INTEGER, REAL, CHARACTER, STRING, BOOLEAN, PID, TIME, DURATION.

La définition des variables se situe dans une définition de processus ou de procédure. Toute variable qui doit être EXPORTÉE (EXPORTED), IMPORTÉE (IMPORTED), RÉVÉLÉE (REVEALED) ou VUE (VIEWED) doit être déclarée avec ses propriétés.

Chaque donnée est la propriété d'une instance de processus. Les données sont définies dans la définition de processus [de ce fait, toutes les instances (de processus) d'un processus ont des exemplaires de la définition des données fournis au niveau du processus].

En cas de «révélation» dans la section de déclaration (DCL), il suffit d'ajouter le mot clé «REVEALED» devant le nom du type de variable.

La déclaration «VIEWED» est située en dehors de la section DCL et elle est composée, de façon syntaxique, du mot clé «VIEWED» suivi du nom de la variable, du type de cette variable, et finalement de l'identificateur du processus qui la révèle.

La figure D-152 illustre l'exemple de la déclaration d'une variable «chiffre» «VIEWED» et «REVEALED» respectivement dans les processus p1 et p2. Ces processus appartiennent au bloc b1. Dans cet exemple, la variable «chiffre» appartient au processus p1 et peut être vue par le processus p2.

```
SYSTEM a;
    .
    .
  BLOCK b1;
    .
    .
  PROCESS p1;
    .
    .
  DCL
    REVEALED chiffre INT,
    compteur, n° d'alarme INT,
    a,b,c, BOOL;
      .
      .
      .
  ENDPROCESS p1;
  PROCESS p2;
  (   )
  DCL d,e,f, BOOL;
    VIEWED chiffre p1,
      .
      .
      .
    ENDPROCESS p2;
  ENDBLOCK b1;
ENDSYSTEM a;
```

FIGURE D-152

Exemple de visibilité des données entre les processus du même bloc

Dans la figure D-153, toutes les instances du processus p1 peuvent visualiser la variable «chiffre» étant donné qu'elle est déclarée comme «REVEALED» et «VIEWED».

```
SYSTEM a;
      .
      .
      .
BLOCK b1;
      .
      .
PROCESS p1;
      .
      .
      .
DCL
      REVEALED chiffre INT;
      .
      .
VIEWED chiffre INT p1;
      .
      .
      .
END PROCESS p1;
      .
      .
      .
END BLOCK b1;
      .
      .
END SYSTEM a;
```

FIGURE D-153

Exemple de visibilité entre les instances de processus dans la même définition de processus

Pour indiquer que l'opération de visualisation est nécessaire, il faut utiliser le mot clé «VIEW» suivi du nom de la variable à visualiser et du propriétaire de l'identité de l'instance de processus de la variable. Le nom de la variable est séparé de l'identité de l'instance de processus par une virgule et tous deux sont mis entre parenthèses. Sans le mot clé «VIEW», l'instance de processus est obligée de rechercher une variable possédant ce nom dans ses données locales (figure D-154).

```
SYSTEM s;
.......
  BLOCK b1;
    .......
    PROCESS p1;
    .......
    DCL
      REVEAL chiffre INT,
    .......
    ENDPROCESS p1;
    PROCESS p2;
    .......
    DCL
    t INT,
    VIEWED chiffre INT p1,
    .......
    TASK t:=5*VIEW(chiffre,p1);
      .......
    ENDPROCESS p2;
  ENDBLOCK b1;
ENDSYSTEM s;
```

FIGURE D-154

Exemple d'utilisation d'une variable de visualisation

Les variables susceptibles d'être exportées doivent avoir l'attribut EXPORTED dans leur définition dans le processus exportateur.

Le processus importateur déclare les variables qu'il a l'intention d'importer dans une section d'importation, en utilisant le mot clé IMPORTED suivi des noms des variables à importer, de leur type et des identificateurs du processus d'exportation.

Une variable peut être déclarée IMPORTED et EXPORTED en même temps. Cela permet à une instance de processus d'exporter une variable vers d'autres instances du même processus et d'importer cette variable à partir d'autres instances. Dans ce cas, il faut prendre soin d'éviter les collisions de nom:

v:=IMPORT(v,Pid)

l'affectation entraînera le remplacement d'un des v par le v de l'instance Pid.

Comme on peut le voir dans l'exemple de la figure D-155, chaque opération d'importation nécessite la présence du mot clé IMPORT précédant le nom de la variable et l'identificateur d'instance de processus, le premier étant séparé du deuxième par une virgule et les deux étant placés entre parenthèses.

Si une variable est exportée ou révélée, l'attribut REVEAL devra être rajouté à l'attribut EXPORTED dans la déclaration (voir la figure D-155). Il convient de noter que la révélation n'est valable que pour les instances de processus du même bloc.

Dans l'exemple de la figure D-155, la variable «compteur» appartenant au processus p2 dans le bloc b2 est exportée et révélée. Elle est exportée pour le processus p1 et révélée pour le processus p3 dans le même bloc b2.

```
SYSTEM a;
        .
        .
  BLOCK b1;
        .
        .
    PROCESS p1;
      DCL
    EXPORTED chiffre INT,
    IMPORTED compteur INT BLOCK b2,p1;
        .
        .
    ENDPROCESS p1;
  ENDBLOCK b1;
  BLOCK b2;
        .
        .
    PROCESS p2;
      DCL
        EXPORTED,REVEALED compteur INT,
        .
        .
        .
    ENDPROCESS p2;
    PROCESS p3;
        .
        .
      DCL
        VIEWED compteur PROCESS p2,
        .
        .
        .
    ENDPROCESS p3;
  ENDBLOCK b2;
ENDSYSTEM a;
```

FIGURE D-155

Exemple de visibilité de données entre des processus appartenant à des blocs différents

D.7.3.5 *Définition des macros*

La construction macro est destinée à régler les problèmes de répétition. Dans les programmes PR, on peut avoir une macro pour découper le flot des énoncés et les remplacer par l'appel de macro [§ D.7.3.7.8 et figure D-156b)].

La définition de macro doit être fournie au début de la définition de système, de bloc, de processus, de procédure [figure D-156a)] selon qu'elle peut être appelée à partir de tous les processus, des processus compris dans un bloc, d'un seul processus ou d'une seule procédure. Il convient de noter que dans PR, la macro dispose d'une entrée et d'une sortie, ce qui rend nécessaire l'utilisation d'étiquettes et de liaisons pour représenter dans PR, un diagramme GR disposant de plus d'une entrée et sortie, respectivement.

L'exemple illustré par la figure D-156 représente une macro [figure D-156a)] avec deux sorties a et b. Cela signifie que dans le programme principal [figure D-156b)], il y a deux étiquettes correspondantes a et b.

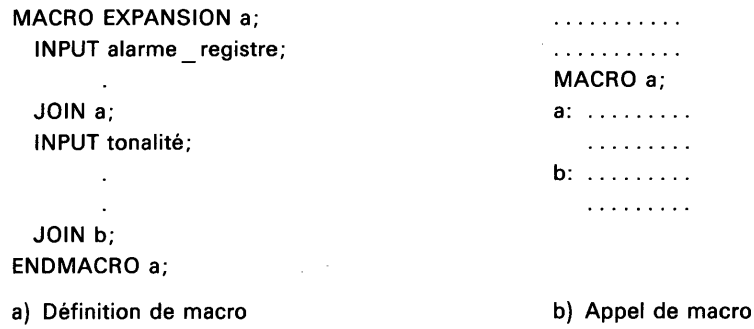


FIGURE D-156

Exemple d'utilisation d'une macro

D.7.3.6 *Définition des procédures*

Les procédures peuvent être définies à divers niveaux dans la hiérarchie des constructions LDS, soit au niveau système, bloc, processus et procédure.

Selon la place où une procédure est définie, elle peut être visible soit pour tous les processus ou procédures dans le système (définition au niveau du système), soit pour tous les processus et procédures d'un bloc (définition au niveau des blocs), soit seulement pour le processus (procédure) dans lequel/laquelle elle est définie.

La définition formelle des paramètres commence avec le mot clé FPAR. Les variables mentionnées comme paramètres formels peuvent avoir les attributs IN, IN/OUT ou SIGNAL si le paramètre effectif correspondant est un signal.

Un paramètre avec le mot clé IN est passé par valeur. Avec le mot clé IN/OUT, un paramètre est passé par référence (voir également le § D.7.3.7.9). Une procédure ne dispose que d'une sortie, mais il peut y avoir un ou plusieurs énoncés RETURN. Après l'interprétation du mot clé RETURN, l'énoncé suivant sera celui situé après l'appel de procédure dans le programme principal (figure D-157).

Le schéma de visibilité est très proche du schéma de définition de types de données. La seule exception est qu'il n'y a pas de procédures prédéfinies, ce qui fait que toute procédure doit être définie de façon que son appelant puisse la voir. La figure D-157 représente un exemple de définition de procédures. Dans la première partie de l'exemple, les paramètres formels sont corrects. Dans le deuxième énoncé de définition de procédures, il y a une erreur due au fait que les types de paramètres formels ne sont pas dans l'ordre, comparés aux paramètres effectifs contenus dans l'appel de procédure.

```
....
....
SIGNAL sig1(int,bool,int),
....
DCL numéro INTEGER,connecté BOOLEAN;
....
PROCEDURE proc1
FPAR
      SIGNAL sig,
      in num INTEGER,
      IN/OUT conn BOOLEAN;
....
....
....
ENDPROCEDURE;
```

Cet énoncé de définition de procédure est correct comparé à l'appel de la procédure, mais

```
PROCEDURE proc1
FPAR
    IN num INTEGER,
    SIGNAL sig,
    IN/OUT conn BOOLEAN;
```

est INCORRECT!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

```
....
....
CALL proc1(sig1,numéro,conn);
....
....
```

FIGURE D-157

Exemples d'utilisation des procédures

D.7.3.7 *Définition de processus*

Ainsi que nous l'avons déjà vu, la définition d'un processus est englobée par les énoncés PROCESS et ENDPROCESS. Un nom doit être associé au mot clé PROCESS. Ce nom (qui a la qualification implicite de processus), avec le nom du bloc englobant et celui du système, constitue l'identificateur du processus.

S'il existe des paramètres formels, ils doivent être déclarés après l'énoncé de processus. Le mot clé FPAR, comme indiqué à la figure D-158, précède les paramètres.

Les paramètres effectifs se trouvent dans l'énoncé CREATE (voir le § D.7.3.7.1).

```
PROCESS nom de processus;
FPAR
nom de variable type;
```

FIGURE D-158

Définition des paramètres formels

Les numéros d'instance, qui doivent être placés entre crochets, sont facultatifs.

Le numéro de l'instance est composé de deux chiffres séparés par une virgule, le premier indiquant le nombre d'instances de processus présentes à l'initialisation du système, le second indiquant le nombre maximum d'instances de processus susceptibles d'exister pendant le cycle de vie du système. Si le premier numéro est omis, il n'y a qu'une instance à l'initialisation du système; si tous les numéros d'instances sont omis, il n'y a qu'une instance de processus dans tout le cycle de vie du système (figure D-159).

```
a) PROCESS p1(5,18);          il y a 5 instances à l'initialisation
                              et un maximum de 18 pendant
                              toute la durée de vie du système;

b) PROCESS p2(,5);            est équivalent à PROCESS p2(1,5);

c) PROCESS p3;                est équivalent à PROCESS p3(1,1);

d) PROCESS erreur (5,3); INCORRECT!!!
```

FIGURE D-159

Exemples de processus

De toute évidence, le nombre maximum d'instances (second chiffre) doit être égal ou supérieur au nombre d'instances existant à l'initialisation du système [figure D-159d)].

Après l'énoncé PROCESS, il nous faut définir les données appartenant au processus et, après les procédures et les macros locales à ce processus (voir le § D.7.3.5 et le § D.7.3.6).

La représentation du comportement commence après ces définitions soit par un énoncé de démarrage (START) suivi d'une chaîne de transitions, soit par un énoncé d'état (STATE).

S'il n'y a pas d'énoncé de démarrage, chaque instance de processus commence à exister au premier énoncé d'état [figure D-160b)].

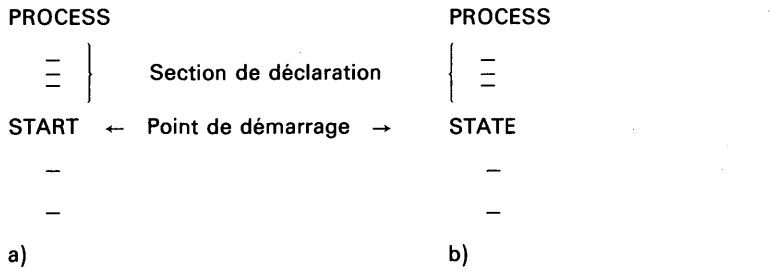


FIGURE D-160

Démarrage d'un processus

D.7.3.7.1 *Création de processus*

Une instance de processus peut créer d'autres instances du même processus ou de différents processus du même bloc, en émettant une action de création.

L'énoncé CREATE peut contenir la liste des paramètres effectifs placés entre parenthèses, comme le montre la figure D-161.

```
SYSTEM s;
  BLOCK b;
    PROCESS p;
    .......
    .......
    DCL     a,c     INTEGER,
            b       BOOLEAN,
    .......
    .......
      CREATE p1(a,b,c);
    .......
    ENDPROCESS p;
    .......
    PROCESS p1;
    FPAR
    .......
      chiffre INTEGER, connecté BOOLEAN, numéro INTEGER;
    START
    .......
    ENDPROCESS p1;
  ENDBLOCK b;
ENDSYSTEM s;
```

FIGURE D-161

Exemple de la création d'une instance de processus

D.7.3.7.2 *Apparitions multiples d'états*

Dans PR, un état est représenté par le mot clé STATE suivi par le nom de l'état.

Un état se termine à l'énoncé d'état suivant (énoncé NEXTSTATE) ou à la fin du processus (énoncé ENDPROCESS ou STOP).

De plus, en PR, il y a toujours des énoncés distincts pour indiquer que l'on passe à un état (NEXTSTATE ) ou qu'on en sort (STATE).

Il n'existe pas un énoncé commun indiquant l'un et l'autre, comme en GR.

Les apparitions multiples d'états ont la même explication et la même relation en modèle sémantique que celles indiquées au § D.6 (GR).

Les apparitions suivantes sont des exemples, dans la version PR, de celles fournies pour GR, au § D.6.3.6.5.

```
b:    NEXTSTATE état_1;
    STATE état_1;
      INPUT A;
      NEXTSTATE état_2;
      INPUT C;
      JOIN a;
    STATE état_2;
      INPUT C;
      JOIN b;
      INPUT B;
a:  NEXTSTATE état_3;

    STATE état_3;
      INPUT D;
      JOIN b;
```

a) Diagramme complet

```
    -
    -
    STATE état_1;
      INPUT A;
      NEXTSTATE état_2;
      INPUT C;
      NEXTSTATE état_3;
    STATE état_2;
      INPUT B;
      NEXTSTATE état_2;
      INPUT C;
      NEXTSTATE état_1;
    STATE état_3;
      INPUT D;
      NEXTSTATE état_1;
```

b) Diagramme avec états principaux et états suivants qui servent de connecteurs à l'état principal

FIGURE D-162

Exemples d'apparitions multiples d'un état (équivaut à la figure D-93)

D.7.3.7.3 *Enoncé PR et utilisation des données*

D.7.3.7.3.1 *Enoncé d'entrée*

L'énoncé d'entrée (INPUT) contient une liste de signaux. Le nom de chaque donnée contenue dans les signaux est établi à partir d'identificateurs de variable. Les identificateurs de variable doivent être du type indiqué dans la définition des signaux, ce qui rend leur position très importante. Ces identificateurs de variable sont placés entre parenthèses et séparés par des virgules (voir la figure D-164). Si une ou plusieurs données de signal sont supprimées, les variables correspondantes sont éliminées, et cela est représenté par deux virgules consécutives (figure D-163).

```
INPUT a(var1,var2,,var4);
```

Remarque – Dans cet énoncé, la troisième donnée du signal a été supprimée.

FIGURE D-163

Signal a) comme entrée avec seulement 3 de ses données définies

```
SIGNAL sig1 (INTEGER,BOOLEAN,INTEGER);
    DCL a INTEGER,b BOOLEAN,c INTEGER,
```

a) déclarations

```
    INPUT sig1(a,b,c);
```

b) une entrée correcte

```
    INPUT sig1(a,c,b);
```

c) une sortie correcte

FIGURE D-164

Enoncés d'entrée

D.7.3.7.3.2 *Enoncé de mise en réserve*

L'énoncé de mise en réserve (SAVE) peut avoir la liste des noms des signaux ou un astérisque, si tous les signaux à l'arrivée n'ayant pas reçu un nom dans les énoncés d'entrée sont mis en réserve. Un exemple simple d'utilisation de mise en réserve est illustré à la figure D-165.

```
–
–
–
STATE Etat_31;
  SAVE S;                    /*S¹ arrive, entre dans la file d'attente et y reste, R arrive
                             et est immédiatement absorbé. La transition allant à
                             l'état_32 est activée.
  INPUT R;

NEXTSTATE Etat_32;
STATE Etat_32;               A l'arrivée dans l'état_32, S¹ est immédiatement absorbé,
                             et la transition allant à l'état suivant est activée. */
  INPUT S;
–
–
–
```

FIGURE D-165

Exemple de l'utilisation d'une mise en réserve

D.7.3.7.3.3 *Enoncé de sortie*

L'énoncé de sortie OUTPUT contient la liste des signaux envoyés vers d'autres processus. Pour chaque donnée contenue dans le signal, il peut y avoir une expression ou une valeur effective correspondante. Si la valeur d'une ou plusieurs données n'est pas définie, elle est omise et la position vide est représentée par deux virgules consécutives (figure D-166).

```
OUTPUT sig1(a,,c);
```

FIGURE D-166

Signal sig1 comme sortie avec seulement 2 de ses données définies sur 3

La liste des valeurs effectives est mise entre parenthèses. La valeur de chaque donnée considérée dans le signal doit être d'un type défini. A la suite des déclarations contenues dans la figure D-164a) pour l'énoncé d'entrée, une sortie correcte et une sortie incorrecte sont montrées à la figure D-167.

```
OUTPUT sig1(2,true,10)
```
a) une sortie correcte

```
OUTPUT sig1(false,true,10)
```
b) une sortie incorrecte

FIGURE D-167

Enoncés de sortie

D.7.3.7.3.4 *Enoncé des tâches*

L'énoncé des tâches (TASK) peut contenir un ou plusieurs énoncés d'affectation, une série d'énoncés de textes et/ou un texte informel. Un texte informel se compose d'un nom et/ou d'une phrase délimités par des apostrophes. Les énoncés ou le texte sont séparés par des virgules (figure D-168).

```
TASK a:=b;
TASK 'connecter l'abonné';
TASK RESET(TEMPS);
TASK affectation_principale, c:=d+e
TASK var1:=var2*var3,
     var4:=var5 MOD var6;
```

FIGURE D-168

Enoncés de tâches

D.7.3.7.3.5 *Enoncé de DECISION*

En LDS/PR, la décision est représentée par le mot clé DECISION suivi du nom et/ou d'une expression ou d'une chaîne de textes correspondant à la décision. Cette dernière est un texte (formel ou non formel) mis entre apostrophes.

L'ensemble des chemins de sortie est délimité au départ par l'énoncé DECISION et, à la fin, par l'énoncé ENDDECISION. (Voir la figure D-169.)

```
DECISION ....;
(résultat): ....;
(résultat): ....;
(résultat): ,,,,;
  ENDDECISION;
```

FIGURE D-169

Délimitation d'une décision

Les résultats d'une décision sont indiqués entre parenthèses et ils sont représentés par une ou plusieurs chaînes de textes délimitées par des apostrophes, ou par des valeurs éventuelles obtenues au moyen de l'évaluation de l'expression contenue dans l'énoncé de décision. Différents résultats mis entre parenthèses sont séparés par des virgules. Les valeurs sont représentées par des expressions constantes ou par des intervalles dont les bornes supérieures et inférieures sont des expressions constantes. Les valeurs des résultats doivent être du type de l'expression contenue dans l'énoncé de décision, comme illustré dans les exemples des figures D-170, D-171 et D-172.

Il est possible d'indiquer certains résultats de façon explicite et de regrouper tous les autres résultats éventuels en utilisant le mot clé ELSE, comme indiqué, par exemple, dans la figure D-170.

```
DECISION 'x';
(2):          .  |
              .  |  chemin 1
(6):          .  |
              .  |  chemin 2
(9,10):       .  |
              .  |  chemin 3
ELSE:         .  |
              .  |  chemin 4
ENDDECISION;
```

Remarque – x est un nombre entier compris entre 1 et 10

Chemin 1 est choisi avec x=2

Chemin 2 est choisi avec x=6

Chemin 3 est choisi avec x=9 ou x=10

Chemin 4 est choisi avec x=1, =3, =4, =5, =7 ou =8.

FIGURE D-170

Décision avec ELSE

```
DECISION 'Catégorie d'abonné';
('international', 'national'):  .  |
                                .  |  chemin 1
('local')                   :   .  |
                                .  |  chemin 2
```

FIGURE D-171

Exemple de DECISION

Exemple

```
DCL x INT,
  DECISION x;
(2+5)          :.....;
(6+8, 10+12) :.....;
ELSE           :.....;
  ENDDECISION;
  DECISION a;
(true) :.....;
(false):.....;
  ENDDECISION;
  DECISION x;
    (10) :.....;
ELSE   :.....;
  ENDDECISION;
```

FIGURE D-172

Exemples des résultats des décisions

L'énoncé ENDDECISION offre des possibilités de structuration comme dans la programmation structurée indiquée à la figure D-173.

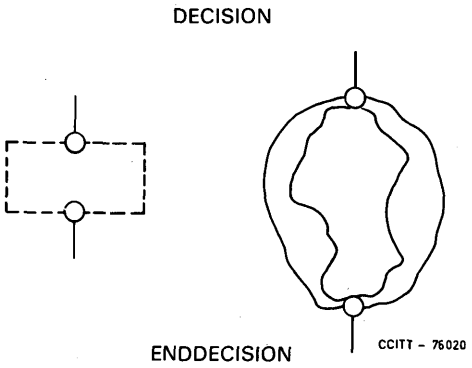


FIGURE D-173

Possibilités de structuration dans la DECISION

Tous les chemins se terminent à l'énoncé ENDDECISION. Les chemins non fermés antérieurement continuent à partir de l'énoncé suivant ENDDECISION, comme l'indiquent les équivalences aux figures D-174 et D-175.

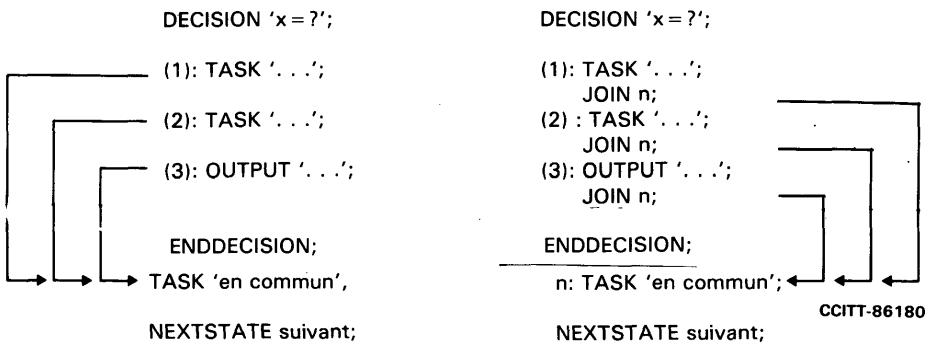


FIGURE D-174

Branchement à partir d'une DECISION

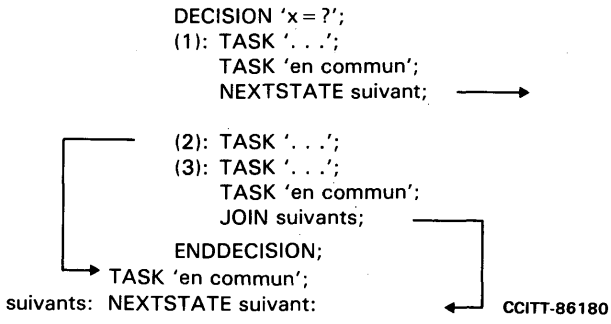


FIGURE D-175

Branchements équivalents à ceux de l'exemple de la figure D-174

L'énoncé de décision peut être utilisé pour servir de modèle à la structure IF-THEN, à la structure DO-WHILE et à la structure LOOP-UNTIL.

La figure D-176 montre la structure IF-THEN; la figure D-177, la structure IF-THEN ELSE; la figure D-178, la structure DO-WHILE et la figure D-179, la structure LOOP-UNTIL.

```
BLOCK a;
  PROCESS b;
    STATE s1;
      INPUT i1;
      DECISION 'i1 = ?'
(0): TASK t1;
ELSE:;
      ENDDECISION;
  ENDPROCESS b;
ENDBLOCK a;
```

FIGURE D-176

Structure IF THEN

```
BLOCK a;
  PROCESS b;
    STATE s1;
      INPUT i1;
      DECISION 'i1=?'
(0): TASK t1;
ELSE: TASK t2;
      ENDDECISION;
  ENDPROCESS b;
ENDBLOCK a;
```

FIGURE D-177

Structure IF THEN ELSE

```
BLOCK b1;
  PROCESS p1;
    STATE s1;
      INPUT i1;
1:    DECISION 'b=?';
      (false): JOIN 11;
      ELSE : ;
      ENDDECISION;
      TASK t1;
      JOIN 1;
11:-
    -
    -
  ENDPROCESS p1;
ENDBLOCK b1;
```

FIGURE D-178

Structure DO WHILE

```
BLOCK b1;
  PROCESS p1;
    STATE s1;
      INPUT i1;
1   DECISION 'b=?';
      (false): JOIN 1;
      ELSE : ;
      TASK t2;
      ENDDECISION;
  ENDPROCESS p1;
ENDBLOCK b1;
```

FIGURE D-179

Structure LOOP...UNTIL...

D.7.3.7.3.6 *Enoncé d'ALTERNATIVE*

Lors de la spécification d'un système, on peut rencontrer des situations où n'importe quel comportement, parmi plusieurs éventuels, répond aux besoins. Nous pouvons alors souhaiter indiquer un certain nombre de comportements permis et laisser à l'opérateur le soin d'en choisir un.

Les diverses variantes sont indiquées par le mot clé ALTERNATIVE auquel est associé un nom ou une chaîne de textes.

L'ensemble des variantes de comportement se termine par l'énoncé ENDALTERNATIVE; chaque variante est identifiée par un nom.

La structure globale est la même que celle de DECISION, la différence se situant dans la sémantique; alors que, pour la DECISION, une des branches est choisie en fonction de la valeur d'une donnée qui change pendant la durée de vie du processus, pour les variantes, il n'existe qu'une seule branche dans le déroulement du processus, en sorte qu'elles sont choisies avant l'exécution. Un exemple est fourni à la figure D-180.

```
                ALTERNATIVE 'Réponse d'alarme'
                  ('première possibilité') : OUTPUT sonnerie;
                  ('deuxième possibilité') : OUTPUT alarme_lumineuse;
                ENDALTERNATIVE;
Au moment de l'exécution on peut avoir
soit
                OUTPUT sonnerie;
soit
                OUTPUT alarme_lumineuse;
```

FIGURE D-180

Exemple d'un énoncé ALTERNATIVE

Les résultats d'un énoncé ALTERNATIVE ont la même syntaxe que l'énoncé des décisions. La figure D-181 en représente un exemple.

```
DCL a INTEGER 1:10;
ALTERNATIVE a;
(1,4)  : ..........;
(5,10) : ..........;
ENDALTERNATIVE;
```

FIGURE D-181

Exemple de l'énoncé d'alternative

D.7.3.7.4 *Modèles d'état*

Il n'existe pas de correspondance en PR.

D.7.3.7.5 *Temps*

Le concept de temps est utilisé dans les énoncés SET (initialisation) et RESET (réinitialisation). Ils peuvent constituer le texte formel dans un énoncé TASK en PR (voir le § D.7.3.7.3.4).

D.7.3.7.6 *Condition de validation*

En PR, la condition de validation est représentée par le mot clé PROVIDED suivi par la condition qui doit être évaluée. Le mot clé est associé à l'énoncé INPUT. La condition associée à INPUT est une expression booléenne. Si la condition est vraie, la transition est permise. Si elle est fausse, le signal est mis en réserve. Etant donné qu'une entrée ne doit paraître qu'une fois par état, la possibilité d'avoir une entrée apparaissant deux fois, à partir du même état, chacune de ces apparitions ayant une condition de validation différente, n'est pas permise.

Les variables utilisées dans l'expression jointe à la condition de validation peuvent être locales ou importées. Elles ne peuvent être vues (viewed).

Un exemple d'utilisation des conditions de validation est illustré à la figure D-182.

```
SYSTEM s;
  BLOCK b1;
  .........
    PROCESS p;
    .........
    DCL
      connecté BOOLEAN,
    IMPORTED alarme BLOCK b2,p2;
    .........
    STATE s1;
      INPUT i1 PROVIDED connecté;
      .........
      INPUT i2 PROVIDED IMPORT alarme,p2;
      .........
    ENDPROCESS p;
  ENDBLOCK b1;
  BLOCK b2;
  .........
    PROCESS p2;
    .........
    DCL
      EXPORTED alarme BOOLEAN;
      .........
    ENDPROCESS p2;
  .........
  ENDBLOCK b2;
ENDSYSTEM s;
```

FIGURE D-182

Exemples de conditions de validation

D.7.3.7.7 *Signaux continus*

Les signaux continus sont représentés en PR par le mot clé PROVIDED suivi par la condition à évaluer. Les signaux continus diffèrent syntaxiquement des conditions de validation par le fait qu'ils ne sont pas associés à un énoncé d'entrée. La condition est une expression booléenne portant sur les données visibles à ce processus. Si, au moment des évaluations, la condition est vraie, la transition est mise en œuvre. S'il existe plusieurs signaux continus, l'ordre d'évaluation doit être indiqué. Il faut pour cela joindre une priorité à chaque condition en utilisant le mot clé PRIORITY, suivi d'une valeur entière. Par définition, les signaux continus reçoivent une priorité moindre que les signaux habituels d'entrée. La condition à évaluer peut ne contenir que des variables locales ou importées. Elle peut ne contenir aucune variable vue (viewed) (figures D-183 et D-184).

```
SYSTEM s;
  .........
  BLOCK b1;
    .........
    PROCESS p1;
      .........
      DCL
        x  INT;
      .........
      STATE s1;
      PROVIDED x=2 PRIORITY 0
      .........
      PROVIDED x=5 PRIORITY 1
      .........
      INPUT i1;
      .........
    ENDPROCESS p1;
  ENDBLOCK b1;
ENDSYSTEM s;
```

FIGURE D-183

Exemple de signaux continus pour variables locales

```
SYSTEM s;
  .........
  BLOCK b1;
    .........
    PROCESS p1;
    .........
    DCL
      connecté BOOL,
    IMPORTED alarme BLOCK b2,p2;
    .........
    STATE s1;
    PROVIDED connecté PRIORITY 0;
    .........
    INPUT i1;
    .........
    PROVIDED IMPORT (alarme,p2) PRIORITY 1;
    .........
    ENDPROCESS p1;
  ENDBLOCK b1;
  BLOCK b2;
    .........
    PROCESS p2;
    .........
    DCL
      EXPORTED alarme BOOL;
    .........
    ENDPROCESS p2;
    .........
    ENDBLOCK b2;
    .........
  ENDSYSTEM s;
```

FIGURE D-184

Exemple de signaux continus pour variables importées

D.7.3.7.8 *Appel de MACRO*

L'énoncé de MACRO est: MACRO nom;

Le nom associé au mot clé MACRO indique une définition de MACRO portant ce nom (voir le § D.7.3.5). L'énoncé de MACRO peut être inséré n'importe où dans une représentation en PR mais doit venir *après* la définition de la MACRO correspondante.

Pour interpréter la représentation, la définition de la MACRO doit remplacer l'énoncé d'appel de MACRO chaque fois qu'il se présente, ainsi que l'indique l'exemple donné à la figure D-185.

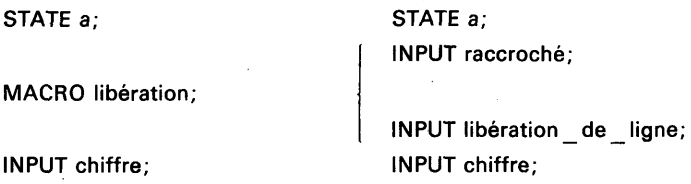


FIGURE D-185

Interprétation d'un énoncé d'appel de MACRO

D.7.3.7.9 *Appel de procédure*

L'appel de procédure contient une liste des paramètres effectifs pour la procédure. Ils sont constitués de signaux d'entrée et de sortie visibles à partir de la procédure. *Tous les signaux visibles par l'appelant et qui ne sont pas déclarés comme paramètres effectifs dans l'appel de procédure sont automatiquement mis en réserve pendant la durée de vie de la procédure.* En outre, les paramètres effectifs contiennent les données visibles par l'appelant et qui sont rendues visibles à la procédure.

La déclaration IN ou IN/OUT est faite dans la définition de procédure, afin qu'elle ne soit pas répétée par l'énoncé d'appel. Les paramètres avec l'attribut IN/OUT dans la définition de procédure correspondent à des variables appartenant directement ou indirectement à l'appelant (lorsque l'appelant est une procédure). Les données mentionnées dans les paramètres effectifs sont associées, dans l'ordre, avec les paramètres formels. Si un paramètre n'est pas fourni, il doit être indiqué par deux virgules consécutives. Dans ce cas, le paramètre formel correspond à la valeur «indéfinie». Un exemple est illustré à la figure D-186.

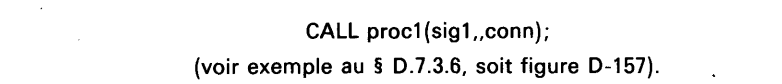


FIGURE D-186

Interprétation d'un appel à une procédure

D.7.3.7.10 *Etiquettes (connecteurs)*

Les étiquettes servent de points d'entrée associés aux énoncés. De ce fait, le contrôle peut être transféré au moyen d'un énoncé JOIN (figure D-187).

FIGURE D-187

Etiquette

Il n'est pas possible de transférer une fonction de commande (et, partant, d'associer des étiquettes) aux types d'énoncés représentés sur la figure D-188.

SYSTEM,	ENDSYSTEM,
BLOCK,	ENDBLOCK,
PROCESS,	ENDPROCESS,
STATE,	ENDDECISION,
INPUT,	SAVE,
ENDMACRO,	ENDPROCEDURE,
ENDALTERNATIVE	

FIGURE D-188

Points d'étiquetage interdits

A un énoncé, il ne peut être associé qu'une seule étiquette. Les étiquettes sont toujours locales et propres à un processus donné. Il n'est pas possible de transférer le contrôle d'un processus à un autre au moyen d'une étiquette.

D.7.3.7.11 *Accès aux énoncés*

Sauf s'il y a indication de transfert de contrôle, les énoncés sont interprétés les uns après les autres.

Les énoncés qui ont pour effet de transférer le contrôle sont indiqués aux figures D-189 à D-193.

STATE: le contrôle est transféré quand un signal arrive à l'énoncé INPUT (entrée) ou SAVE (mise en réserve) qui contient le nom de ce signal.

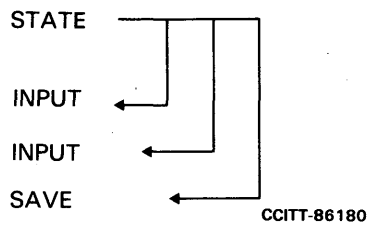


FIGURE D-189

Transfert de contrôle effectué par STATE

JOIN: le contrôle est transféré à l'énoncé qui porte l'étiquette nommée dans le JOIN: il ne doit y avoir qu'un et un seul énoncé à porter une telle étiquette.

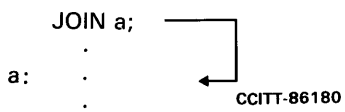


FIGURE D-190

Transfert de contrôle effectué par JOIN

NEXTSTATE: le contrôle est transféré à l'état qui porte le nom indiqué dans l'énoncé NEXTSTATE.

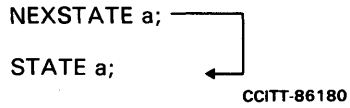


FIGURE D-191
Transfert de contrôle effectué par NEXTSTATE

STOP: l'interprétation prend fin, il n'y a pas de transfert de contrôle.

```
STOP;
```

FIGURE D-192
Pas de transfert de contrôle

Transfert implicite: c'est celui qui a déjà été expliqué pour les branches de décision.

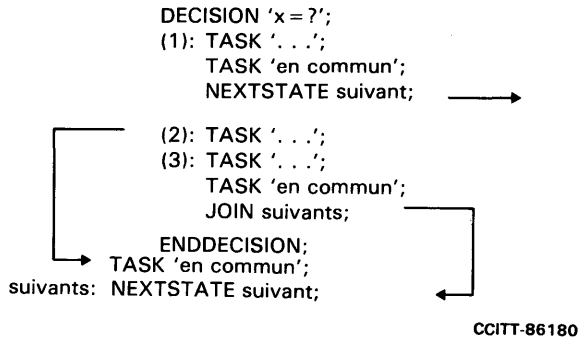


FIGURE D-193
Exemple de transfert implicite

D.7.3.7.12 *Utilisation de divergence et de convergence*

En PR, cette fonction est couverte par des étiquettes également, comme défini dans le § D.7.3.7.10.

D.7.3.7.13 *Commentaires*

Les commentaires sont insérés au moyen du mot clé COMMENT.

Le mot clé peut être inséré en tant qu'énoncé chaque fois qu'un énoncé TASK (tâche) peut l'être. Le mot clé peut en outre être associé à n'importe quel autre énoncé à la fin de l'énoncé:

```
STATE a COMMENT '. . .';
```

A l'exception des énoncés commençant par END auxquels il n'est pas possible d'associer de commentaire.

Il est possible d'ajouter un commentaire CHILL /* .... */ chaque fois qu'il y a un espacement (un blanc). Des exemples de l'utilisation de ce type de commentaires sont illustrés aux figures D-194 et D-195.

```
DCL
a INTEGER, /* explication de l'utilisation de la variable a */
b BOOLEAN; /* explication de l'utilisation de la variable b */
```

FIGURE D-194

Utilisation des commentaires CHILL dans une déclaration de variable

```
PROCEDURE ABC (IN a INTEGER,/* signification de ce paramètre formel */,
IN/OUT b BOOLEAN /* signification de ce paramètre formel */);
```

FIGURE D-195

Utilisation des commentaires CHILL dans une définition de procédure

D.7.3.7.14 *Notations abrégées*

On peut insérer un * (astérisque) dans un énoncé STATE/INPUT/SAVE (état, entrée, mise en réserve) pour indiquer que tous les noms applicables à cet énoncé sont valables.

Exemple:

```
a) STATE *;                    b) STATE A,B,C,D,G;
```

Remarque – a) est équivalent à b) si A,B,C,D,G représente l'ensemble des états définis dans ce processus.

FIGURE D-196

Notation de tous les états

```
a) INPUT *;                    b) INPUT I1, I3, I6;
```

Remarque – a) est équivalent à b) où I1, I3 et I6 sont des signaux qui entrent dans le processus mais *ne sont pas déclarés comme INPUT ou SAVE dans cet état.* Seule une entrée unique (INPUT) ou une mise en réserve unique (SAVE) accompagnée de * peut être rattachée à un état particulier.

FIGURE D-197

Notation de tous les signaux entrants

a) SAVE *;

b) SAVE I1, I3, I6;

Remarque – a) est équivalent à b) où I1, I3 et I6 sont des signaux qui entrent dans le processus mais *ne sont pas déclarés comme INPUT dans cet état.* Aucune autre mise en réserve (SAVE) ne peut être rattachée au même état.

FIGURE D-198

Notation de toutes les mises en réserve

A un STATE *; nous pouvons associer une liste d'exceptions afin d'exclure certains états de la liste. Les noms des états à exclure sont encadrés par des crochets [ ].

a) STATE * [A,C,D];

b) STATE B,E,F,G;

Remarque – a) et b) sont équivalents si A,B,C,D,E,F,G, sont les états du processus considéré.

FIGURE D-199

Notation de tous les états (à l'exception de certains)

On peut insérer un tiret (–) dans un énoncé NEXTSTATE pour indiquer que l'état d'où provient la transition est le même que celui où elle aboutit (figure D-200).

a) STATE a;
INPUT i;

NEXTSTATE –;

b) STATE a;
INPUT i;

NEXTSTATE a;

Remarque – a) et b) sont équivalents.

FIGURE D-200

Etat identique d'origine et de fin

Cette notation est particulièrement utile dans les cas où l'on souhaite accomplir un ensemble commun d'actions dans plusieurs états qui, ultérieurement, restent dans le même état, comme indiqué à la figure D-201.

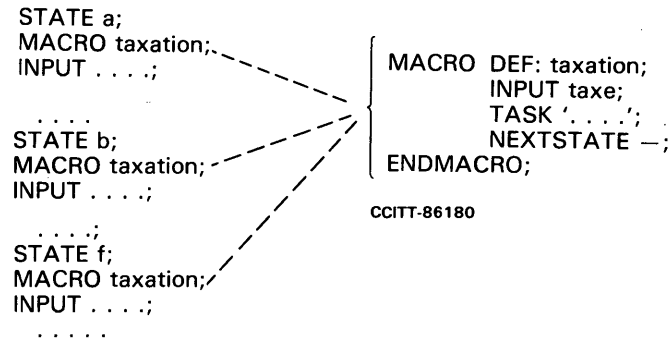


FIGURE D-201
Définition de macro

D.8 *Mise en correspondance*

Le présent paragraphe décrit certains aspects de mise en correspondance du LDS et du CHILL (voir le § D.8.1), du LDS/GR et du LDS/PR (voir le § D.8.2).

D.8.1 *Mise en correspondance du LDS et du CHILL*

Ce paragraphe expose diverses possibilités de mise en correspondance du LDS et du CHILL. Ces solutions sont données à titre d'exemple et ne sont pas exhaustives; dans la pratique, rien ne s'oppose à ce qu'on ait recours à d'autres méthodes.

De fait, pour la mise en correspondance, il ne faudrait pas tenir compte uniquement du compilateur CHILL et de la machine cible; en général, la mise en correspondance nécessite une activité intellectuelle très complexe et ce n'est qu'à force d'expériences que les concepteurs/programmeurs peuvent décider de la structure particulière du programme CHILL qui servira à exécuter une représentation LDS donnée. Cette remarque vaut également pour la représentation LDS des fonctions exécutées à l'aide d'un programme CHILL. Une correspondance biunivoque (si elle est possible) n'est pas nécessairement la meilleure façon d'utiliser le LDS pour représenter les fonctions exécutées en CHILL.

Dans le cadre de cette méthode, la structure globale d'une mise en correspondance d'un diagramme complet LDS et d'un programme (incomplet) CHILL est représentée en figure D-202.

Des exemples de schémas de mise en correspondance de constructions des deux langages sont illustrés aux figures D-203 à D-206. Ils concernent les constructions LDS suivantes:

* état et réception ou mise en réserve des signaux; sélection d'un état suivant;
* sortie;
* liaison;
* décision.

Le module de déclaration contient à la fois la définition et la déclaration de tous les signaux employés dans le diagramme LDS qui sont transformés et de toutes les variables qui leur sont associées: toutes ces variables sont octroyées au module qui représente le bloc fonctionnel du diagramme LDS.

```
Déclaration MODULE

        /*  module CHILL contenant les signaux et leurs variables associées utilisés dans le diagramme
            LDS                                                                              */

            GRANT

        /*  octroi des signaux et des variables                                              */

            SIGNALS

        /*  définition des signaux                                                           */

            SYNMODE (ou NEWMODE)

        /*  définition de types                                                              */

END déclaration

Bloc fonctionnel: MODULE

        /*  module contenant la partie procédurale des diagrammes LDS                       */

            SEIZE

        /*  saisie de tous les signaux et de leurs variables que ce bloc fonctionnel peut recevoir ou
            émettre                                                                          */

        /*  définition et déclaration des données; ces données valent globalement pour tous les
            processus qui relèvent de ce module                                              */

Nom_du_processus: PROCESS ();

        /*  définition et déclaration des données locales                                    */

            état suivant:=...;
            liaison:=néant;

            DO FOR EVER;
                boucle d'état: CASE état_suivant OF

        */  boucle sur la variable état_suivant indiquant l'état LDS                         */

                                        étiquette d'état: RECEIVE CASE
                                        (nom_du_signal_1):
                                                .
                                                .
                                                .
                                        (nom_du_signal_n):

                ESAC boucle d'état;
                DO WHILE liaison/=néant;
                    CASE liaison OF;
                        (étiquette_de_liaison_1): liaison:=néant;
                                                .
                                            .   .
                                                .
                        (étiquette_de_liaison_n): liaison:=néant;
                                                .
                                                .
                    ESAC;
                OD;
            OD;
        END nom_du_processus;
END Bloc_fonctionnel;
```

FIGURE D-202

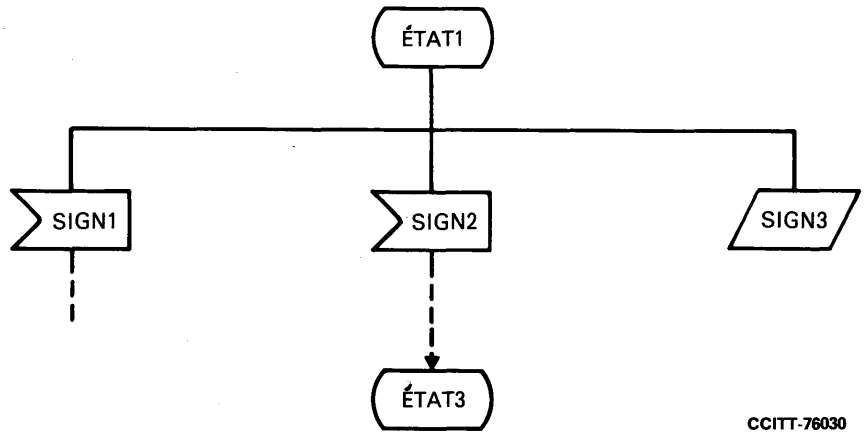
Exemple de mise en correspondance biunivoque de LDS et CHILL

Le module du bloc fonctionnel représente le comportement (partie procédurale) des processus du LDS.

Dans ce schéma de traduction, chaque processus du LDS est représenté par une boucle infinie: une variable nommée «état suivant» indique l'état à examiner et une variable nommée «liaison» indique des points de jonction possibles qui déterminent des ensembles communs d'énoncés.

On choisit au moyen de la construction CASE de CHILL, la valeur de l'état suivant; chaque entrée du CASE identifie un état du LDS. Pour chaque entrée, on choisit entre les signaux d'entrée possibles. Chaque signal d'entrée détermine la série d'actions à accomplir, («Le chemin de transition»).

Chaque trajet de transition se termine soit par une affectation de la variable soit à «état_suivant», déterminant ainsi directement l'état suivant à examiner, ou à la variable «liaison». Une autre boucle de sélection sur la valeur actuelle de la variable «liaison» permet à chaque transition de prendre fin, au sens du LDS, et à la fin, affecte une valeur à la variable état_suivant.



a) LDS

```
ÉTAT1:
   RECEIVE CASE
      (SIGNAL1): .............
                 .............
      (SIGNAL2): .............
                 état_suivant:=ÉTAT3;
      ELSE GETOUT (LIST1)
   ESAC ÉTAT1;
```

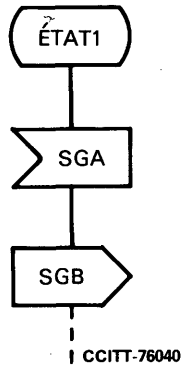
b) CHILL

FIGURE D-203

Exemples de mise en correspondance des énoncés STATE/INPUT/SAVE/NEXTSTATE

L'un des problèmes principaux dans la relation LDS et CHILL réside dans la sémantique différente de la réception des signaux; en fait, alors que le CHILL ne consomme pas (et par conséquent ne détruit pas) les signaux à moins qu'ils ne soient reçus (sigaux persistants), le processus LDS consomme (et par conséquent détruit) tous les signaux reçus, jusqu'à ce qu'il y ait concordance avec l'une des entrées énumérées pour cet état. La différence de sémantique a été résolue en introduisant la fonction prédéfinie: GETOUT comme une alternative (chemin ELSE) dans la construction CHILL RECEIVE CASE, comme indiqué à la figure D-203. La fonction préfédinie GETOUT du CHILL, qui connaît (par paramètres) la liste des signaux d'entrée et de mise en réserve, détruit les autres signaux disponibles au processus lorsque ce dernier est appelé.

Après l'exécution de la fonction GETOUT le sélecteur d'état est mis à 1 de façon à répéter la boucle pour cet état jusqu'à ce qu'un signal d'entrée valable soit sélectionné (ou arrive, s'il n'est pas déjà présent).



a) LDS

```
ÉTAT1;
   RECEIVE CASE
      (SGA): pi:=get_instance_value();
             send SGB to pi;
             état_suivant:=.....;
      ELSE   état_suivant:=état1;
ESAC ÉTAT1;
```

b) CHILL

FIGURE D-204

Exemple de mise en correspondance des sorties

Par exemple, une fois que le signal de sortie SGA, à la figure D-204, a été reconnu, l'instance appropriée du processus destinataire pour le signal SGB est choisie et le signal SGB est envoyé.

Avant d'envoyer le signal SGB, il peut être nécessaire de remplir certains champs d'information qui doivent être acheminés par le signal. Cela peut être fait immédiatement avant ou, bien plus longtemps avant l'envoi du signal.

Quand il existe un point de liaison dans le diagramme (voir figure D-205), on affecte à la variable «liaison» la valeur appropriée. Ainsi qu'il a été expliqué dans la figure D-202, une boucle sur la variable liaison est exécutée pour déterminer le prochain état à examiner. Dans l'optique du langage de programmation, on peut considérer un point de liaison comme une construction «goto» (aller à); la collecte de tous les points de liaison, afin qu'ils puissent être examinés, permet d'écrire entièrement le squelette de programme sans faire intervenir de «goto», ce qui en facilite la lecture.

Une décision en LDS se traduit directement dans la construction CASE en CHILL, comme indiqué à la figure D-206.

D.8.2 *Mise en correspondance de GR et PR*

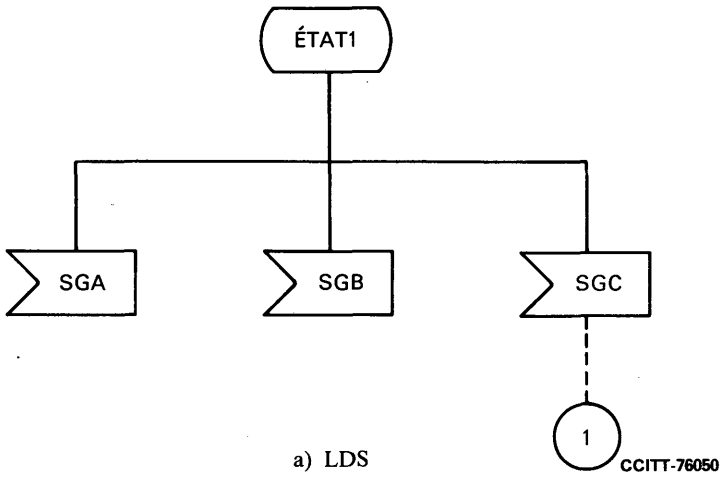
Certaines parties d'un système ne peuvent être décrites qu'au moyen du PR, par exemple, définition des données et définition des signaux. En conséquence, il peut être nécessaire de compléter les systèmes représentés au moyen du GR par une syntaxe PR. Cela signifie que les systèmes représentés au moyen du GR peuvent toujours être complètement représentables en GR.

La figure D-207 représente la partie interne d'un bloc B fournie par un diagramme d'interaction des blocs et ses parties correspondantes en PR.

La figure D-208 présente un exemple de définition d'un processus simple pour un processus PR 1. Elle est donnée sous la forme GR et PR. Seules les parties ayant une représentation graphique sont indiquées.

Pour le diagramme d'interaction des blocs, les constructions GR ont une corrélation avec les constructions PR, comme indiqué à la figure D-209.

La figure D-210 illustre la corrélation correspondante pour le diagramme de processus.



a) LDS

```
ÉTAT1:
   RECEIVE CASE
      (S1 in m): case m.id of
                    (SGA): ..... ;
                    (SGB): ..... ;
                    (SGC): ..... ; liaison:=1;
                    ELSE état_ suivant:=état1
                 esac;
ESAC ÉTAT1;
```

b) CHILL

FIGURE D-205

Exemple de mise en correspondance de JOIN

D.9 *Exemples d'application du LDS*

D.9.1 *Introduction*

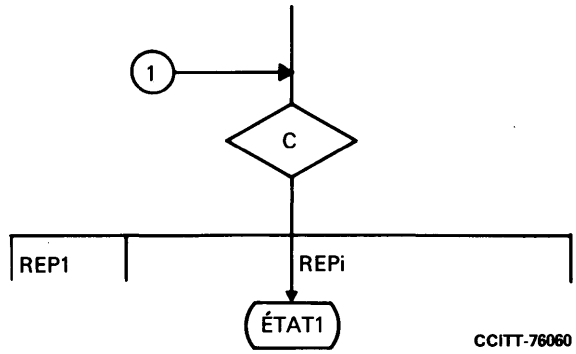
Le § D.9 contient trois exemples d'utilisation du LDS. Les exemples sont tirés de domaines d'application des télécommunications et mettent en œuvre différents sous-ensembles du LDS. On s'est efforcé de prendre des exemples pratiques et de traiter d'autant de concepts LDS que possible.

Dans chaque exemple, un système est présenté à un haut niveau d'abstraction, et seule une partie du système est élaborée en détail, les autres parties étant déclarées «non définies».

Il est essentiel, pour une définition d'un système, que toutes ses parties soient liées, de façon non ambiguë, dans un ensemble complet (voir le § D.5). Etant donné que les recommandations ne fournissent pas de construction de langage susceptible de satisfaire à cette spécification, ces constructions doivent être élaborées dans le cadre des exemples. Ces constructions de référence sont basées sur les règles syntaxiques du LDS.

Etant donné que certains des usagers du LDS préféreront probablement utiliser différents types de constructions de référence, deux méthodes principales sont utilisées dans les exemples. L'une des méthodes est basée sur la syntaxe LDS/PR, et l'autre sur les commentaires dans les diagrammes synoptiques.

Les exemples sont destinés à illustrer l'utilisation du LDS mais ne constituent pas des spécifications internationales.



a) LSD

```
1: CASE C OF
     (REP1): ..... ;
     (REP2): ..... ;
       .
       .
       .
     (REPi): ..... ; état_ suivant := état1 ;
   ESAC 1 ;
```

b) CHILL

FIGURE D-206

Exemple de mise en correspondance des DÉCISIONS

D.9.2 *Système de commutation téléphonique*

Dans cet exemple, seul le LDS de base (voir la Recommandation Z.101) est utilisé.

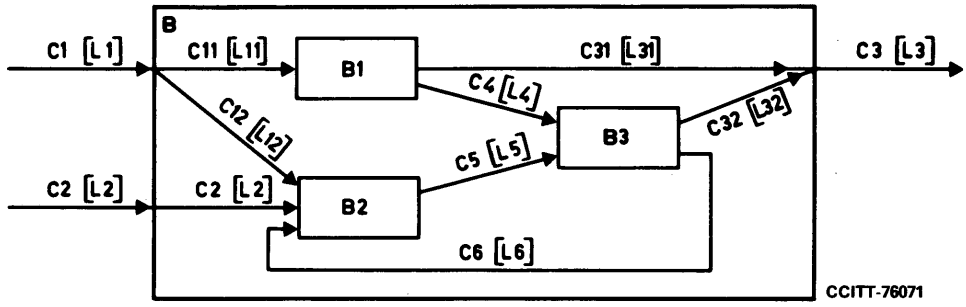
La définition du système est présentée sous toutes les formes syntaxiques concrètes du LDS, afin de faciliter leur comparaison.

Cet exemple traite du processus de traitement d'appel dans un système de commutation téléphonique, et, il est posé comme hypothèse, que l'usager a une connaissance de base en matière de téléphonie. Dans la première phase (établissement de la communication), une connexion est établie entre l'appelant (A) et l'appelé (B), appartenant tous deux au même central. Dans la deuxième phase (conversation), le processus attend les signaux de fin d'appel. Dans la dernière phase (fin de l'appel), les lignes des abonnés sont déconnectées et le processus retourne à l'état libre.

D.9.2.1 *Forme syntaxique LDS/GR dans les deux versions*

La forme syntaxique graphique utilisée habituellement (version orientée vers la transition) ne met pas en œuvre des modèles d'état, alors que l'autre forme (version orientée vers les états) les met en œuvre. Le diagramme d'interaction des blocs est commun aux deux formes.

Dans le présent exemple, aucune définition de variables n'est incluse dans la version utilisant les états et, le positionnement ainsi que la remise à l'état initial des temporisateurs sont montrés de façon implicite. Les états suivants sont également représentés par un symbole d'état de taille réduite contenant le numéro d'état qui, à lui seul, suffit à l'identification.



a) Syntaxe GR

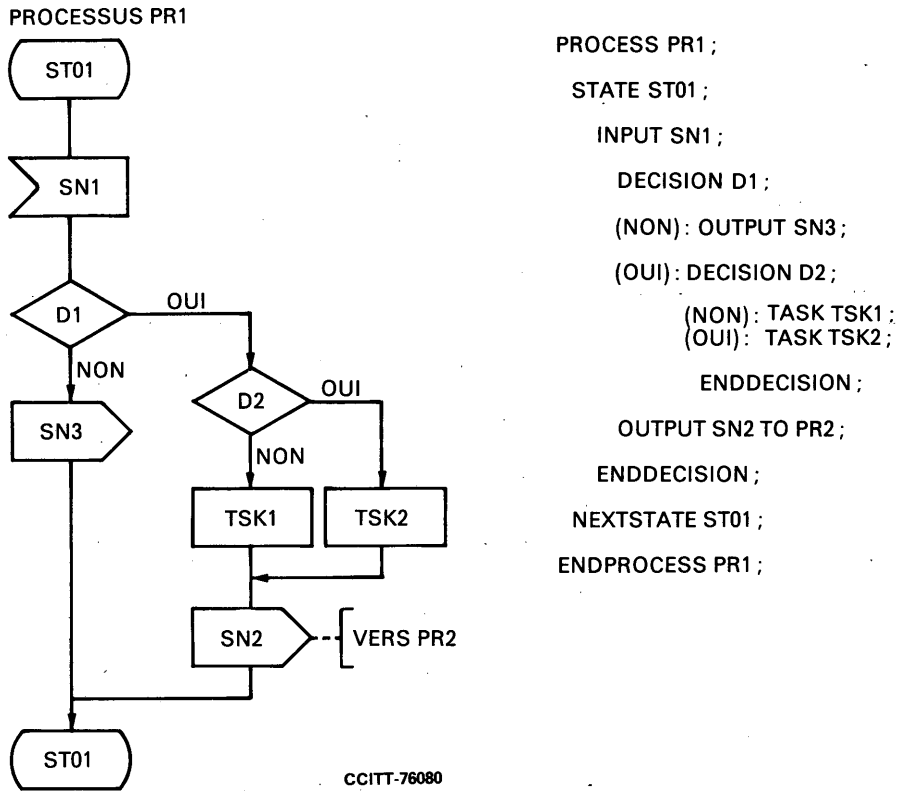
```
BLOCK B;
   SUBSTRUCTURE;
   SUBBLOCKS: B1, B2, B3;
   SPLIT C1 INTO C11, C12;
   SPLIT C2 INTO C2;
   SPLIT C3 INTO C31, C32;
   CHANNEL C11 FROM ENV TO B1 WITH /* L11 */;
   CHANNEL C12 FROM ENV TO B2 WITH /* L12 */;
   CHANNEL C2  FROM ENV TO B2 WITH /* L2 */;
   CHANNEL C31 FROM B1 TO ENV WITH /* L31 */;
   CHANNEL C32 FROM B3 TO ENV WITH /* L32 */;
   /* Les nouveaux canaux */
   CHANNEL C4 FROM B1 TO B3 WITH /* L4 */;
   CHANNEL C5 FROM B2 TO B3 WITH /* L5 */;
   CHANNEL C6 FROM B3 TO B2 WITH /* L6 */;
   ENDSUBSTRUCTURE;
ENDBLOCK B;
```

b) Syntaxe PR

Remarque – Le progamme PR n'est pas complet, d'après les règles de syntaxe.

FIGURE D-207

Corrélation entre GR et PR pour un diagramme d'interaction de blocs



Remarque – D1 et D2 sont des données formellement définies. PR2 est une donnée de type PID dans l'exemple PR, mais peut être le nom du processus dans l'exemple GR.

FIGURE D-208

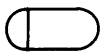
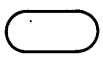
Corrélation entre GR et FR pour un diagramme de processus

CONCEPT	GR	PR
canal		CHANNEL
bloc		BLOCK ENDBLOCK
processus		PROCESS ENDPROCESS
environnement du système	ENVIRONMENT	ENV
système		SYSTEM ENDSYSTEM
liste de signaux	[ ]	
création		CREATE
chemin de signal		
extension de texte		
commentaire	/* */	/* */

CCITT-76090

FIGURE D-209

Corrélation des constructions en GR et PR pour des symboles utilisés dans un diagramme d'interaction des blocs

CONCEPT	GR	PR
état		STATE
état_ suivant		NEXTSTATE
entrée		INPUT
sortie		OUTPUT
tâche		TASK
décision		DECISION ENDDECISION
connecteur d'entrée		x:
connecteur de sortie		JOIN x
extension de texte		
commentaire		COMMENT ou /*.....*/
mélange de transitions		JOIN x x:
tous	*	*
tous (excepté)	* [.....]	* [.....]
mise en réserve		SAVE
appel de procédure		CALL
début de procédure		PROCEDURE
arrêt de procédure		RETURN
appel de macro		MACRO
entrée de macro		MACRO EXPANSION
sortie de macro		ENDMACRO
début		START
arrêt		STOP
demande de création		CREATE
signal continu	< >	PROVIDED
condition de validation		INPUT... PROVIDED
option		ALTERNATIVE ENDALTERNATIVE

CCITT-76101

FIGURE D-210

Corrélation des constructions en GR et PR dans un diagramme de processus

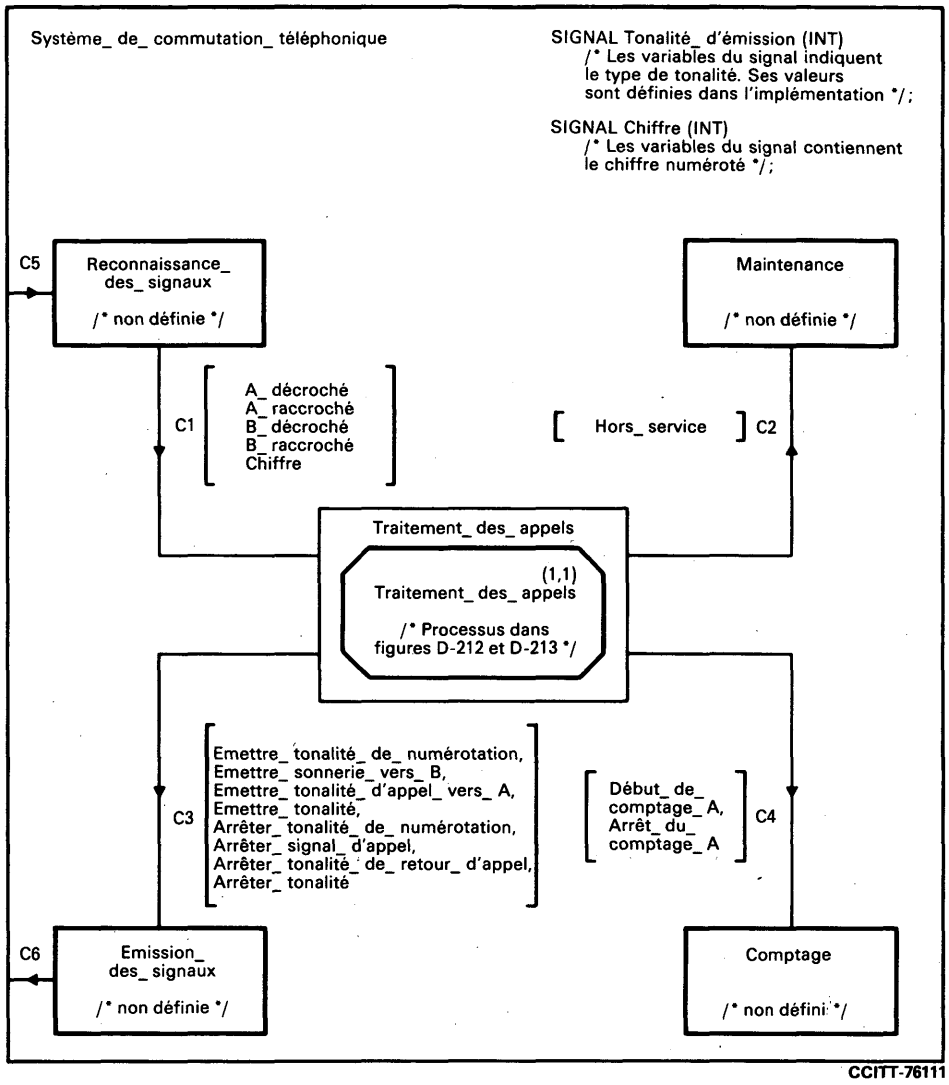


FIGURE D-211

Diagramme d'interaction du bloc pour le système de commutation téléphonique

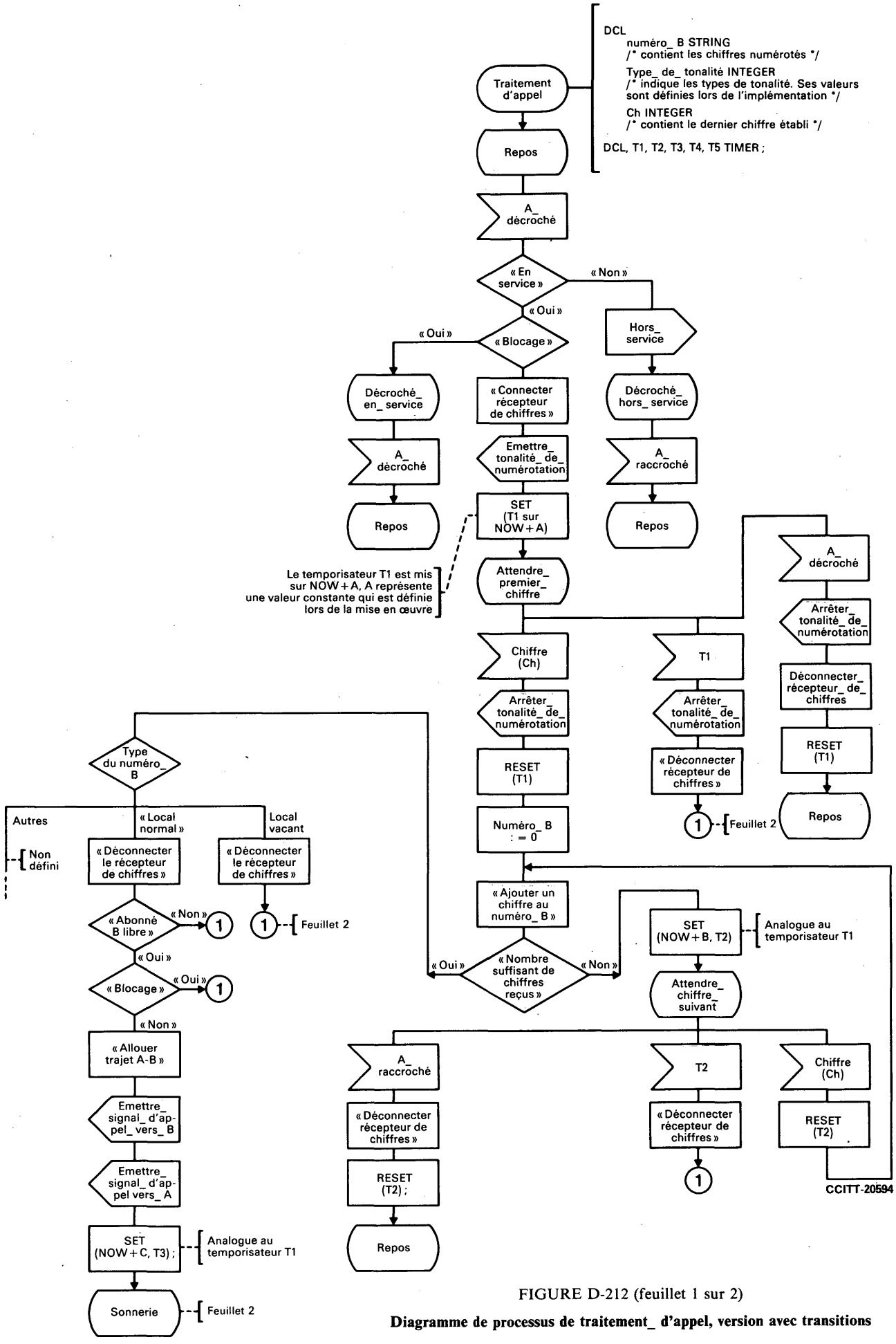


FIGURE D-212 (feuillet 1 sur 2)

Diagramme de processus de traitement_ d'appel, version avec transitions

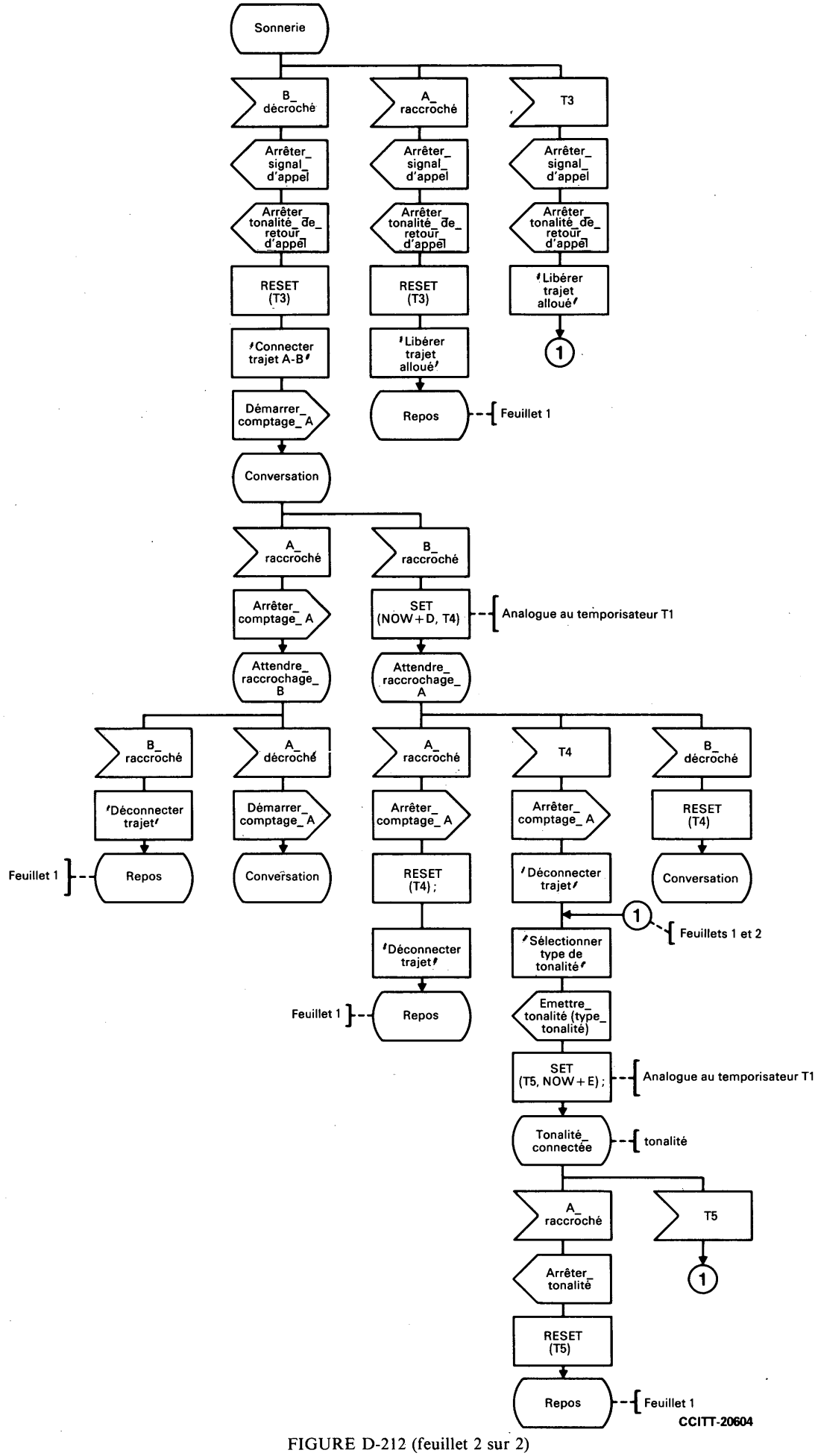


FIGURE D-212 (feuillet 2 sur 2)

Diagramme de processus de traitement_ d'appel, version avec transitions

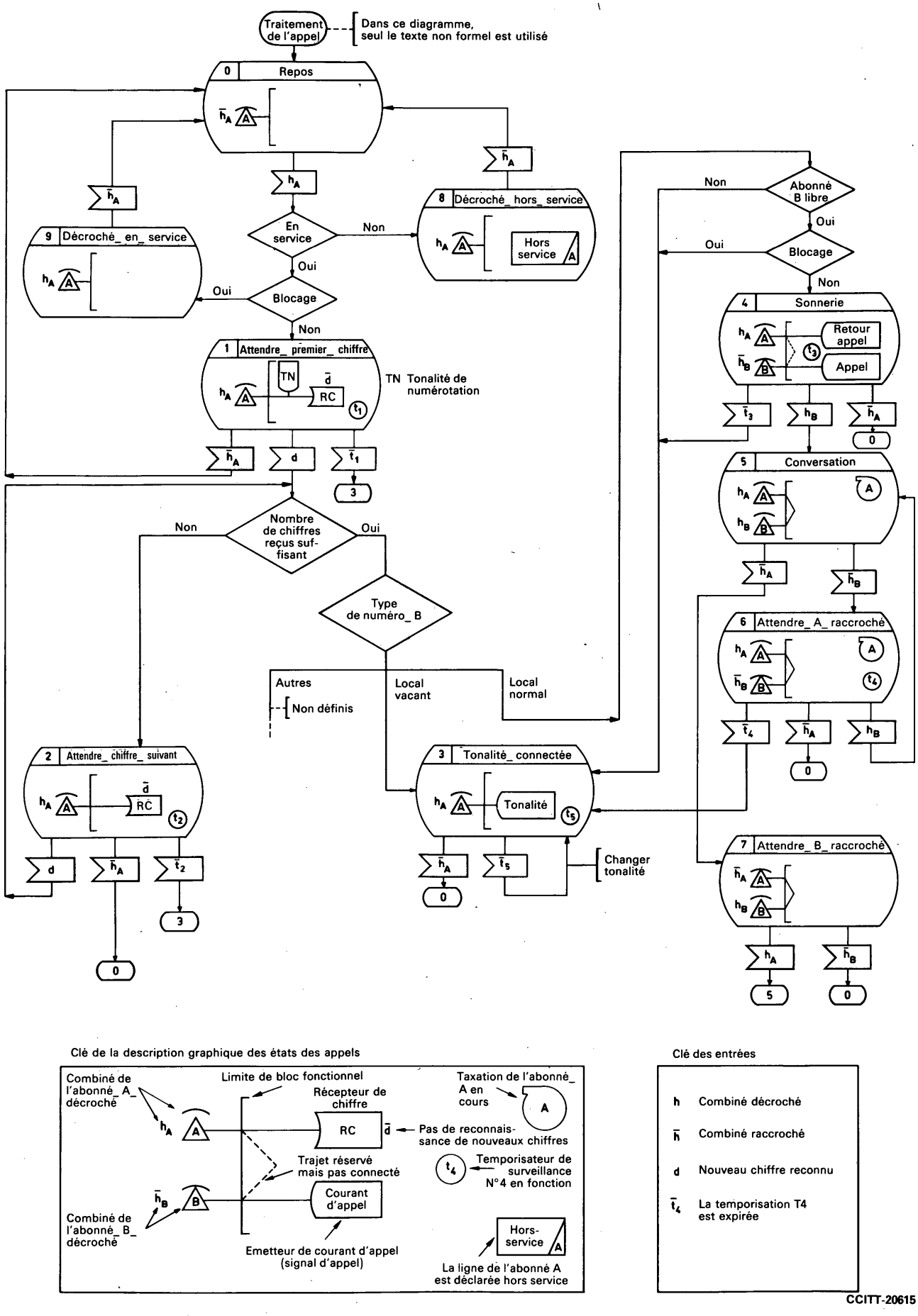


FIGURE D-213

Diagramme de processus de traitement_ d'appel, version avec états

```
SYSTEM Système_de_commutation_téléphonique;
        CHANNEL C1 FROM Reconnaissance des signaux TO Traitement_d'appel
                WITH    A_décroché,
                        A_raccroché,
                        B_décroché,
                        B_raccroché,
                        Chiffre;
        CHANNEL C2 FROM Traitement_d'appel TO Maintenance
                WITH    Hors_service;
        CHANNEL C3 FROM Traitement_d'appel TO Emission_des_signaux
                WITH    Emettre_tonalité_de_numérotation,
                        Emettre_sonnerie_vers_B,
                        Emettre_tonalité_d'appel_vers_B,
                        Emettre_tonalité,
                        Arrêter_tonalité_de_numérotation,
                        Arrêter_signal_d'appel,
                        Arrêter_tonalité_de_retour_d'appel,
                        Arrêter_tonalité,
        CHANNEL C4 FROM Traitement_d'appel TO Comptage
                WITH    Début_de_comptage_A,
                        Arrêt_du_comptage_A;
        CHANNEL C5 FROM ENV TO Reconnaissance_des_signaux
                WITH    /* non définie */;
        CHANNEL C6 FROM Emission_des_signaux à ENV
                WITH    /* non définie */;
        SIGNAL Tonalité_d'émission (INT)
                /* Les variables du signal indiquent le type de tonalité.
                Ses valeurs sont définies dans l'implémentation */;
        SIGNAL chiffre (INT)
                /* Les variables du signal contiennent le chiffre numéroté */;


BLOCK Reconnaissance_des_signaux /* non définie */;
ENDBLOCK;
BLOCK Emission_des_signaux /* non définie */;
ENDBLOCK;
BLOCK Maintenance /* non définie */;
ENDBLOCK;
BLOCK Comptage /* non défini */;
ENDBLOCK;
```

FIGURE D-214 (page 1 de 4)

Définition du système par la syntaxe LDS/PR

```
BLOCK Traitement_d'appel
    PROCESS Traitement_d'appel (1,1);
        DCL Numéro_B STRING
            /* contient les chiffres numérotés */;
        DCL Type_de_tonalité INTEGER
            /* indique les types de tonalité. Ses valeurs sont définies lors de l'implémentation */;
        DCL Ch INTEGER
            /* contient le dernier chiffre établi */;
        START; NEXTSTATE Repos;
        STATE; Repos;
            INPUT A_décroché;
                DECISION 'En service';
                    ('Non'):  OUTPUT Hors_service;
                              NEXTSTATE Décroché_hors_service;
                    ('Oui'):  DECISION 'Blocage';
                                  ('Oui'):  NEXTSTATE Décroché_en_service;
                                  ('Non'):  TASK 'Connecter_récepteur_de_chiffres';
                                            OUTPUT Emettre_tonalité_de_numérotation;
                                            TASK SET (NOW +A, T1);
                                            /* Le temporisateur T1 est mis sur Now +A, A représente une
                                            valeur constante qui est définie lors de la mise en œuvre */
                                            NEXTSTATE Attendre_premier_chiffre;
                              ENDDECISION;
                ENDDECISION;
        STATE Déc;
            INPUT A_décroché_en_service;
                NEXTSTATE Repos;
        STATE Décroché_hors_service;
            INPUT A_raccroché;
                NEXTSTATE Repos;
        STATE Attendre_premier_chiffre;
            INPUT Chiffre (Ch);
                OUTPUT Arrêter_tonalité_de_numérotation;
                TASK RESET (T1),
                    Numéro_B:=0;
            10: TASK 'Ajouter un chiffre au numéro_B';
                DECISION 'Nombre suffisant de chiffres reçus';
                    ('Non'):  TASK SET (NOW +B, T2);
                              NEXTSTATE Attendre_chiffre_suivant;
                    ('Oui'):  DECISION 'Type de numéro_B';
                                  ('Local normal'): TASK 'Déconnecter le récepteur de chiffres';
                                      DECISION 'Abonné_B libre';
                                          ('Non'):  JOIN 1;
                                          ('Oui'):  DECISION 'Blocage';
                                                        ('Oui'):  JOIN 1;
                                                        ('Non'):  TASK 'Allouer trajet A-B';
                                                                  OUTPUT Emettre_signal_d'appel_vers_B;
                                                                  OUTPUT Emettre_signal_d'appel_vers_A;
                                                                  TASK SET (NOW +C, T3);
                                                                  NEXTSTATE Sonnerie;
                                                    ENDDECISION;
                                      ENDDECISION;
                                  ('Local vacant'):  TASK 'Déconnecter le récepteur de chiffres';
                                                     JOIN 1;
                                  ('Autres'):        /* non défini */;
                              ENDDECISION;
                ENDDECISION;
```

FIGURE D-214 (page 2 de 4)

Définition du système par la syntaxe LDS/PR

```
    INPUT T1;
        OUTPUT Arrêter_tonalité_de_numérotation;
        TASK 'Déconnecter_récepteur_de_chiffres';
        JOIN 1;
    INPUT A_décroché;
        OUTPUT Arrêter_tonalité_de_numérotation;
        TASK 'Déconnecter_récepteur_de_chiffres';
        TASK RESET (T1);
        NEXTSTATE Repos;

STATE Attendre_chiffre_suivant;
    INPUT Chiffre (Ch);
        TASK RESET (T2);
        JOIN 10;
    INPUT T2;
        TASK 'Déconnecter_récepteur_de_chiffres';
        JOIN 1;
    INPUT A_décroché;
        TASK 'Déconnecter_récepteur_de_chiffres';
        TASK RESET (T2);
        NEXTSTATE Repos;

STATE Sonnerie;
    INPUT B_décroché;
        OUTPUT  Arrêter_signal_d'appel;
                Arrêter_tonalité_de_retour_d'appel;
        TASK RESET (T3);
        TASK 'Connecter trajet A-B';
        OUTPUT Démarrer comptage_A;
        NEXTSTATE Conversation;
    INPUT A_décroché;
        OUTPUT  Arrêter_signal_d'appel;
                Arrêter_tonalité_de_retour_d'appel;
        TASK RESET (T3);
        TASK 'Libérer trajet alloué';
        NEXTSTATE Repos;
    INPUT T3;
        OUTPUT  Arrêter_signal_d'appel,
                Arrêter_tonalité_de_retour_d'appel;
        TASK 'Libérer trajet alloué';
        JOIN 1;

STATE Conversation;
    INPUT A_raccroché;
        OUTPUT Arrêter_comptage_A;
        NEXTSTATE Attendre_raccrochage_B;
    INPUT B_raccroché;
        TASK SET (NOW +D, T4);
        NEXTSTATE Attendre_raccrochage_A;
```

FIGURE D-214 (page 3 de 4)

Définition du système par la syntaxe LDS/PR

```
        STATE Attendre_raccrochage_B;
            INPUT A_décroché;
                OUTPUT Démarrer_comptage_A;
                NEXTSTATE Conversation;
            INPUT B_raccroché;
                TASK 'Déconnecter trajet';
                NEXTSTATE Repos;
        STATE Attendre_raccrochage_A;
            INPUT A_raccroché;
                OUTPUT Arrêter_comptage_A;
                TASK RESET (T4);
                TASK 'Déconnecter trajet';
                NEXTSTATE Repos;
            INPUT B_décroché;
                TASK RESET (T4);
                NEXTSTATE Conversation;
            INPUT T4;
                OUTPUT Arrêter_comptage_A;
                TASK 'Déconnecter trajet';
            1:  TASK 'Sélectionner type de tonalité';
                OUTPUT Emettre_tonalité (type-tonalité);
                TASK SET (NOW +E, T5);
                NEXTSTATE Tonalité_connectée;


        STATE Tonalité_connectée;
            INPUT A_raccroché;
                OUTPUT Arrêter_tonalité;
                TASK RESET (T5);
                NEXTSTATE Repos;
            INPUT T5;
                JOIN 1;
ENDPROCESS Traitement_de_l'appel;
ENDBLOCK Traitement_de_l'appel;
ENDSYSTEM;
```

FIGURE D-214 (page 4 de 4)
Définition du système par la syntaxe LDS/PR

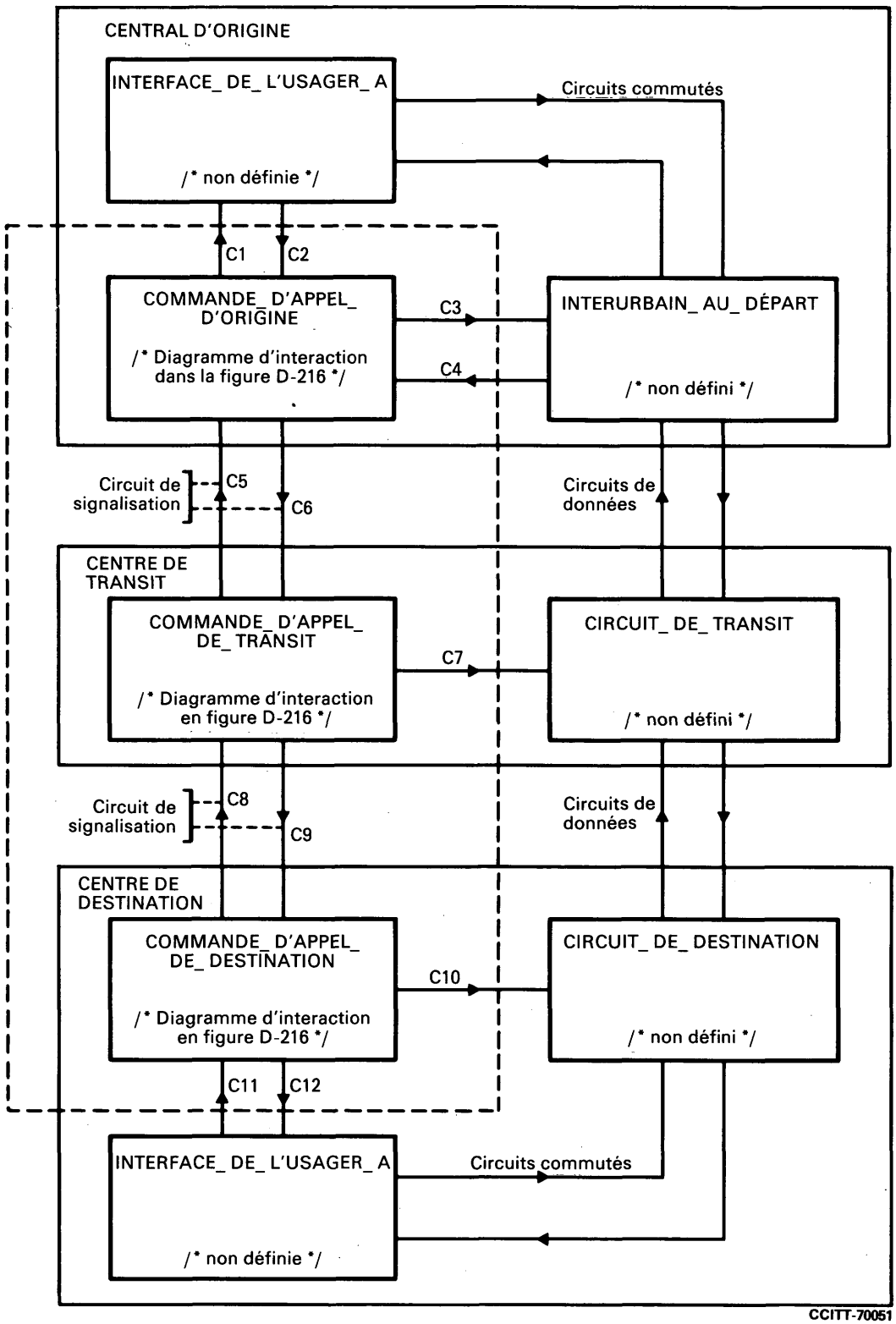
D.9.3 *Sous-système utilisateur données du système de signalisation par canal sémaphore*

L'exemple suivant est tiré de la Recommandation X.61. Il n'est pas nécessairement équivalent à X.61 et il n'est pas destiné à servir de norme internationale. L'exemple illustre l'utilisation de la procédure et les concepts d'option qui sont définis dans la Recommandation Z.103.

Dans un réseau à commutation de circuits mettant en œuvre une signalisation par canal sémaphore, les messages de signalisation pour un groupe de circuits interurbains entre centraux sont acheminés au moyen d'un réseau à signalisation centralisée. La figure D-215 présente un diagramme synoptique d'un réseau de données à commutation de circuits comprenant trois centraux. Le Sous-système Utilisateur Données du système de signalisation par canal sémaphore est représenté par les blocs fonctionnels de commande d'appel dans les centraux d'origine, de transit et de destination. Le diagramme synoptique ne fait pas formellement partie de la spécification mais il y est inclus afin d'expliquer les rapports entre le système de signalisation et les autres éléments du réseau de données. Un diagramme d'interaction de bloc du système de signalisation est représenté en figure D-216.

Les processus d'origine, de transit et de destination (figures D-217, D-218 et D-219) représentent les fonctions nécessaires pour établir, maintenir et terminer une communication de données entre deux interfaces d'usager. Au cours de la phase d'établissement de l'appel, le central de destination peut mettre en œuvre une des deux réponses vers l'appel à l'arrivée. Ces solutions sont établies au moyen de l'utilisation du symbole d'option. L'acceptation d'un appel par l'usager appelé entraîne l'émission d'un signal Prêt pour la Transmission de Données, qui est envoyé vers le processus d'origine une fois que le central de destination a établi la communication.

Les processus d'origine, de transit et de destination mettent tous en œuvre des fonctions similaires pour libérer les circuits interurbains lorsque la communication est terminée. Ces fonctions sont définies par les



Remarque – Sous-système Utilisateur de Données du système à signalisation par canal sémaphore illustré dans les frontières en pointillés.

FIGURE D-215

Diagramme synoptique d'un réseau de données à commutation de circuits comprenant trois centraux

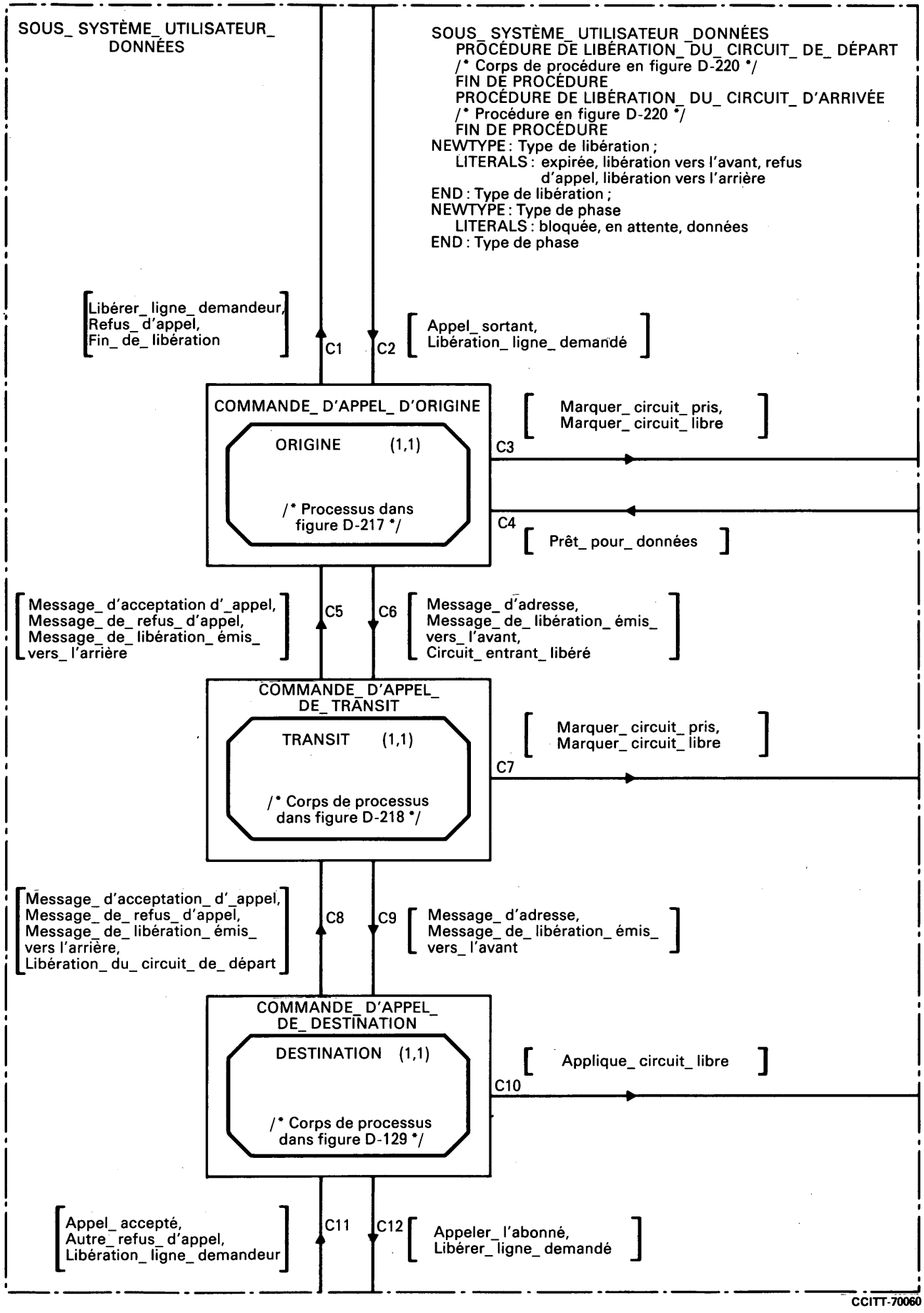


FIGURE D-216

Diagramme d'interaction des blocs pour le système SOUS-SYSTÈME_UTILISATEUR_DONNÉES

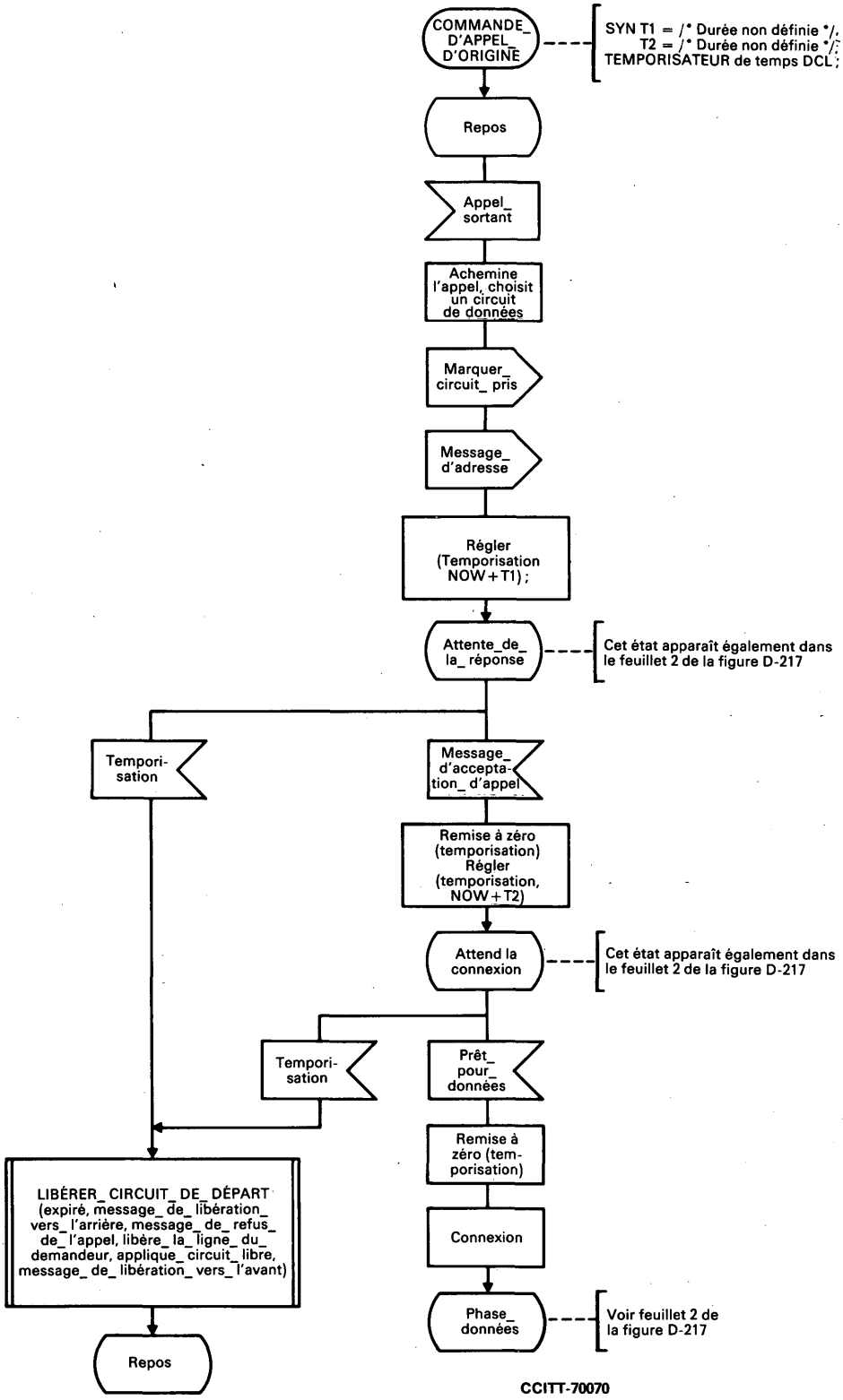


FIGURE D-217 (feuillet 1 sur 2)

Diagramme de processus de commande_ d'appel_ d'origine

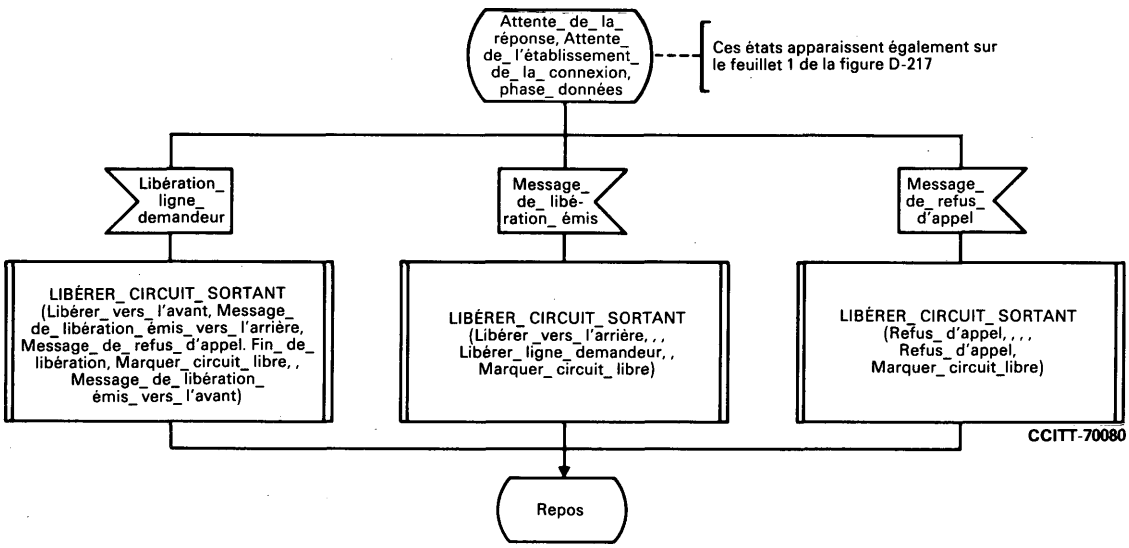


FIGURE D-217 (feuillet 2 sur 2)

Diagramme de processus de commande_ d'appel_ d'origine

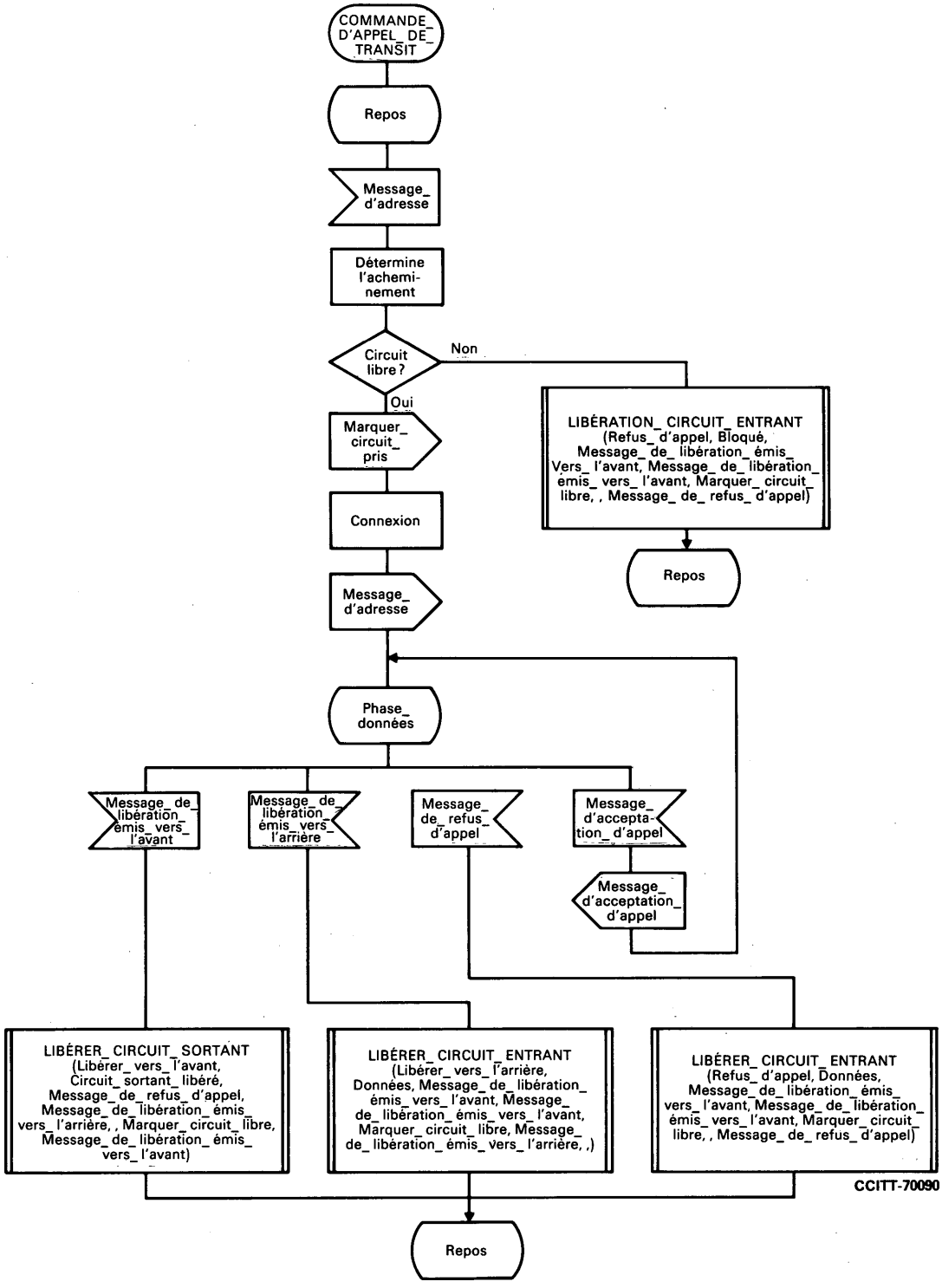


FIGURE D-218

Diagramme de processus de commande_ d'appel_ de_ transit

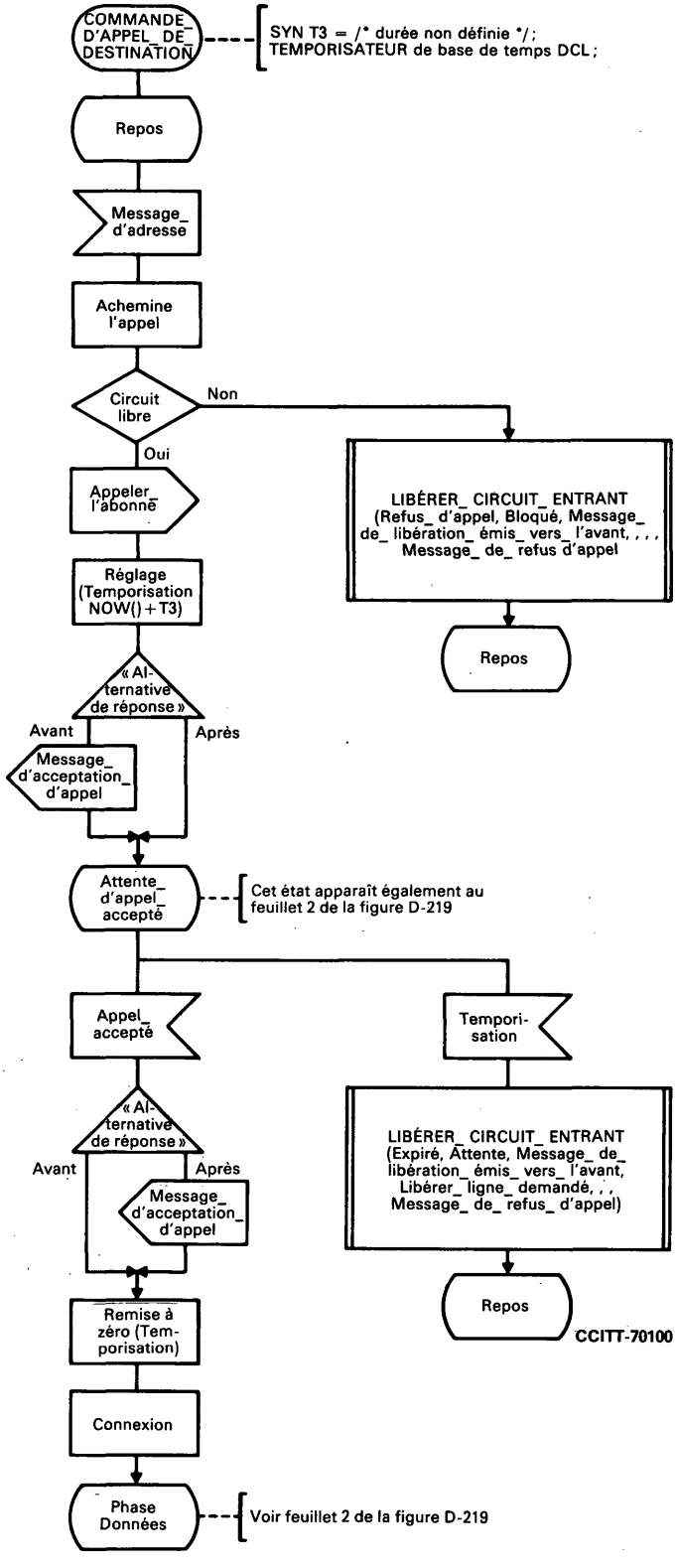


FIGURE D-219 (feuillet 1 sur 2)

Diagramme de processus de commande_ d'appel_ de_ destination

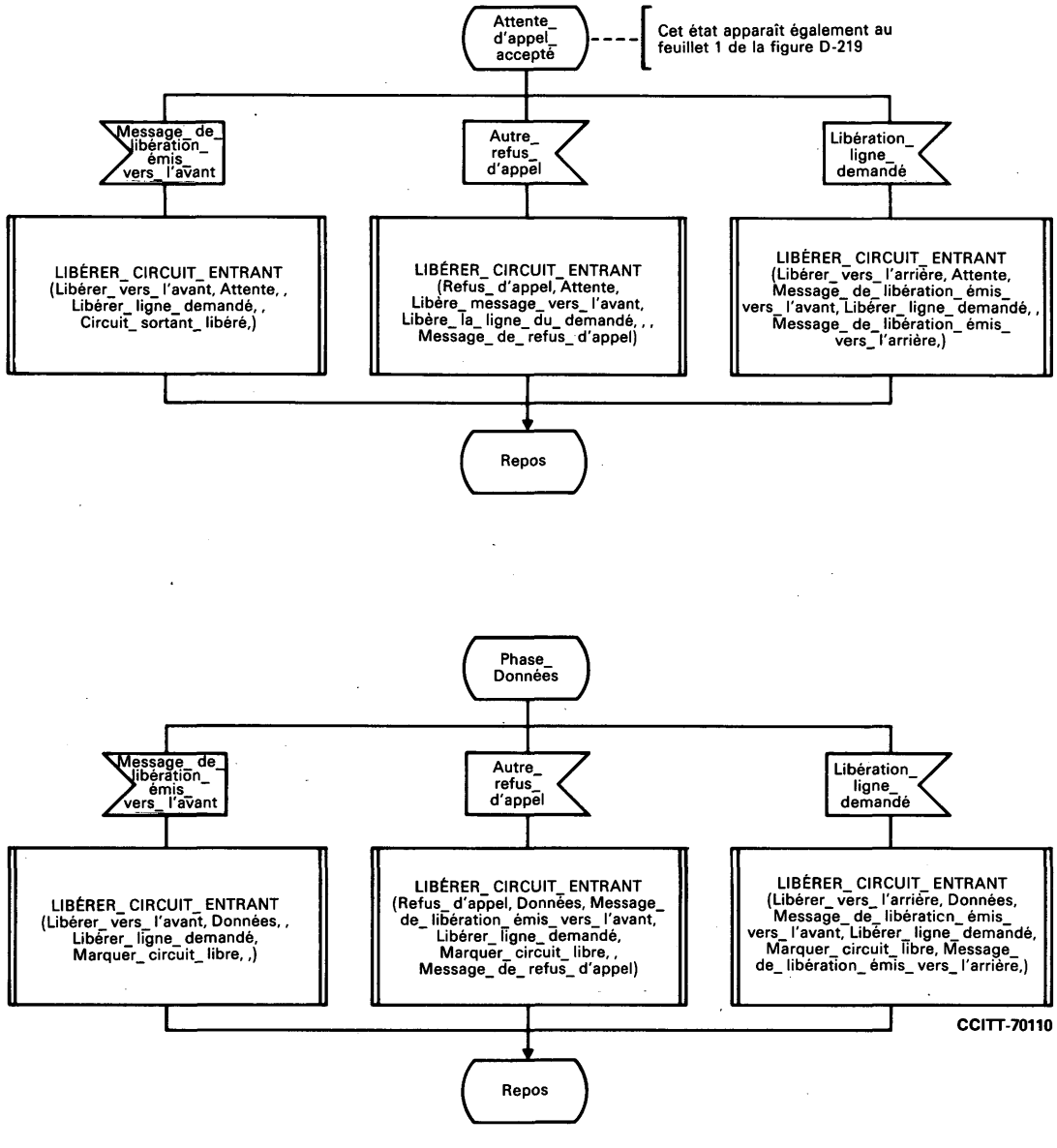


FIGURE D-219 (feuillet 2 sur 2)

Diagramme de processus de commande_ appel_ destination

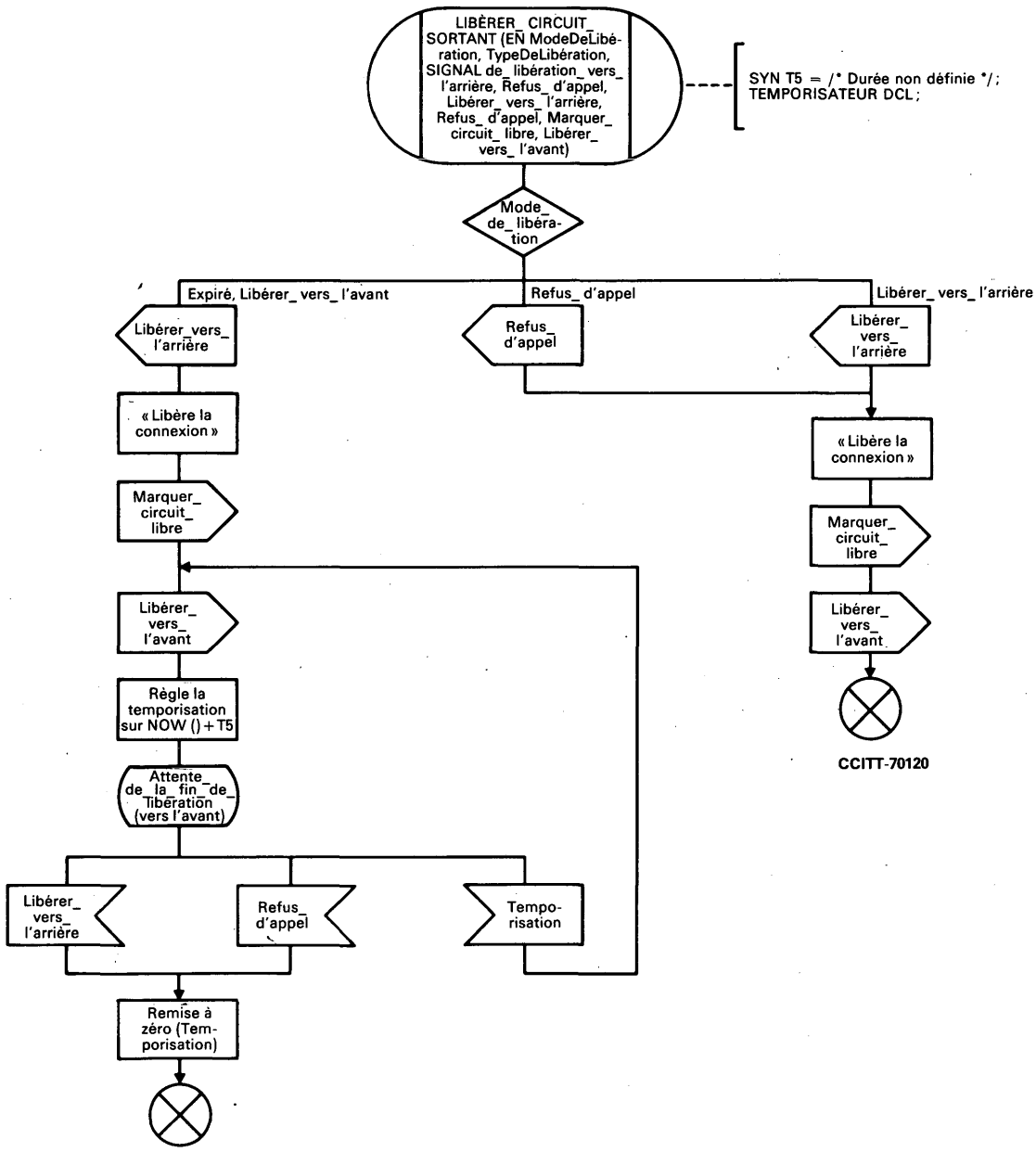


FIGURE D-220

Diagramme de procédure libération_ circuit_ sortant

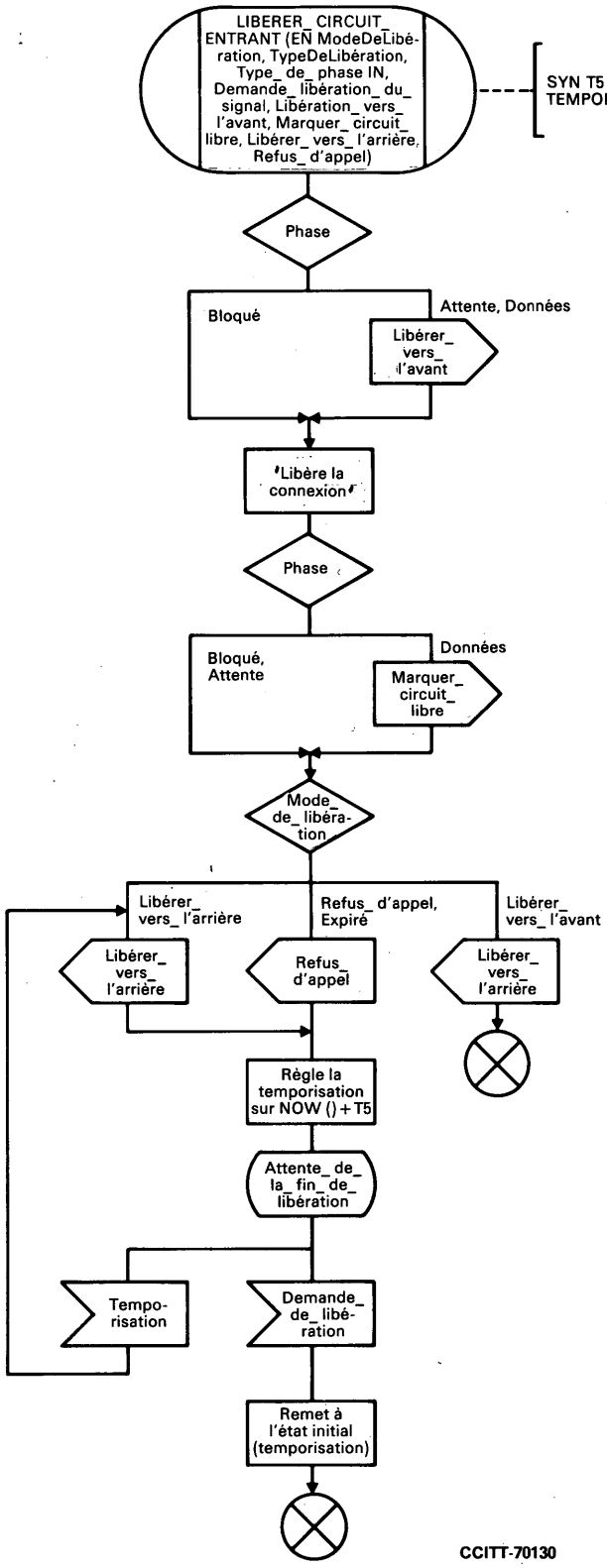


FIGURE D-221

Diagramme de procédure libérer_ circuit_ entrant

D.9.4 *Protocole de transport de classe 0*

L'exemple suivant est fondé sur le protocole de transport de la Recommandation relative à l'interconnexion de systèmes ouverts. Cet exemple représente un sous-ensemble de protocole de transport qui n'est pas nécessairement équivalent à la Recommandation et n'est pas destiné à servir de norme internationale.

Les Recommandations Z.101 et Z.103 du LDS ont également été utilisées. Les données ont été employées mais de manière peu informelle, sans utiliser la Recommandation Z.104. Les types d'énumération contenus dans Z.104 ont été simulés avec des variables INTEGER, en donnant des noms à des constantes non définies. Les opérations relatives aux données n'ont pas été formellement définies.

D.9.4.1 *Aperçu général*

La couche de transport du modèle de référence OSI fournit un service à ses utilisateurs en offrant des voies de communication (appelées connexions de transport) entre deux usagers. A chaque point de connexion de l'usager, la couche de transport est représentée par un organisme de transport. Les organismes de transport communiquent entre eux en utilisant un protocole de transport et les services de la couche Réseau. Un diagramme synoptique est représenté à la figure D-223. L'exemple décrit un organisme de transport qui ne met en œuvre que le protocole de transport de classe 0.

Une connexion de transport est établie entre deux organismes de transport. Le modèle sur lequel la spécification est basée décrit le fonctionnement d'un organisme, de façon indépendante, et peut être appliqué aux deux extrémités de la connexion. Le diagramme d'interaction (figure D-224) décrit un bloc unique représentant un organisme de transport, relié à l'Environnement par des canaux intitulés Point d'accès au Service de transport (TSAP) et Point d'accès au Service de réseau (NASP). L'usager du Service de transport et la couche Réseau sont considérés comme étant l'Environnement.

L'organisme de transport contient deux processus: le premier, le DIRECTEUR, possède une instance unique qui est toujours présente (indiquée par les numéros figurant en haut à droite du symbole de processus). Le second processus, le ENDPOINT (EXTRÉMITÉ) ne possède que des instances établies de façon dynamique, et dont l'une est établie pour le cycle de vie de chaque tentative, afin d'établir une connexion de transport. Si un signal «demande CONNEXION_T» ou «indication CONNEXION_N» est reçu par le DIRECTEUR, ce dernier met immédiatement en œuvre une nouvelle instance de ENDPOINT afin d'acheminer tous les signaux consécutifs associés à la tentative de connexion. Chaque instance de ENDPOINT a une «sorte» de propriété, avec l'une des valeurs INITIATEUR ou RÉPONDEUR qui détermine le fonctionnement initial de l'instance; l'instance prend fin lorsque la connexion est fermée par l'un des usagers du service de transport, ou à la suite d'une erreur.

Le fonctionnement du DIRECTEUR est représenté dans le diagramme de processus de la figure D-225. Celui de ENDPOINT se trouve dans les figures D-226 à D-230. Dans la spécification de ENDPOINT, les procédures LDS sont utilisées pour l'établissement de la connexion, du transfert des données et des phases d'arrêt.

D.9.4.2 *Spécification LDS du système*

La spécification du système se fait au moyen du texte LDS-PR suivant. Le texte s'écarte des Recommandations en ce sens que certaines parties y sont présentées sous la forme LDS-GR avec référence à des parties figurant dans le LDS-PR sous forme de commentaires.

```
ORGANE DE TRANSPORT DU SYSTÈME;
        /*      le rôle de l'ORGANE DE TRANSPORT dans une connexion de transport entre deux usagers est
                représenté dans le diagramme synoptique de la figure D-223 */
        /*      les canaux et signaux associés sont illustrés dans le diagramme d'interaction, figure D-224 */
    SIGNAL /* input-list */
        T_CONN_REQ      (PID,STRING)    ,
        T_CONN_RSP      (PID,STRING)    ,
        T_CONN_RPLY                     ,
        T_DATA_REQ      (STRING)        ,
        T_DISC_REQ                      ,
        N_CONN_IND      (PID,STRING)    ,
        N_CONN_CFM      (PID,STRING)    ,
        N_CONN_RPLY                     ,
        N_DATA_IND      (STRING)        ,
        N_DISC_IND                      ,
        N_RESET_IND                     ;
            MACRO EXPANSION signals;
                T_CONN_REQ, T_CONN_RSP, T_CONN_RPLY, T_DATA_REQ, T_DISC_REQ,
                N_CONN_IND, N_CONN_CFM, N_CONN_RPLY, N_DATA_IND, N_DISC_IND,
                N_RESET_IND,
            ENDMACRO signals;
    SIGNAL /* output-list */
        T_CONN_IND      (PID,STRING)    ,
        T_CONN_CFM      (PID,STRING)    ,
        T_DATA_IND      (STRING)        ,
        T_DISC_IND                      ,
        N_CONN_REQ      (PID,STRING)    ,
        N_CONN_RSP      (PID,STRING)    ,
        N_DATA_REQ      (STRING)        ,
        N_DISC_REQ                      ;
```

FIGURE D-222 (page 1 de 3)

Spécification du système à l'aide du LDS/PR

```
BLOCK TRANSPORT_ENTITY;
    PROCESS DIRECTOR (1, 1) ;
        /* the single instance of this process controls the assignment of ENDPOINTs */
        DCL  conn_params STRING ,
             conn_id PID ;
        /* process diagram, figure D-225, gives process body */
    ENDPROCESS DIRECTOR;
        PROCESS ENDPOINT (0, );
            /* an instance of this process is created by DIRECTOR for each connection requested */
            FPAR
                    kind            INT /* values INITIATOR, RESPONDER */ ,
                    conn_params     STRING ,
                    conn_id         PID;
            DCL
                    ref             STRING ,
                    tc_id           PID ,
                    nc_id           PID ,
                    tpdu_size       INT  /* values                                    */ ,
                    cstat           INT  /* values DATA_TRANSFER, DISCONNECTED        */ ;
                        IMPORTED /* from tc_id via channel TSAP_in */
                    t_ind_flo       BOOL ,
                        IMPORTED /* from nc_id via channel NSAP_in */
                    n_req_flo       BOOL ,
                        EXPORTED /* to tc_id via channel TSAP_out */
                    t_req_flo       BOOL ,
                        EXPORTED /* to nc_id via channel NSAP_out */
                    n_ind_flo       BOOL ;

    PROCEDURE INITIATE_CONNECT;
        SIGNAL      MACRO signals;
        IN          conn_params     STRING ,
        IN          tc_id           PID ,
        IN/OUT      nc_id           PID ,
        IN/OUT      tpdu_size       INT ,
        IN/OUT      cstat           INT /* values DATA_TRANSFER, DISCONNECTED */
      DCL
        n_conn_params STRING ,
        prop_tpdu_size INT ,
        tpdu_flag       INT /* values CC, DR, ER, CR, DT, INV */ ,
        tpdu_data       STRING ,
        nbin            STRING ,
        nbout           STRING ,
        tbout           STRING ,
        taskr           INT /* values CLOSE, NC, TC, NC&TC, NONE */ ;
      /* procedure diagram, figure D-227, gives procedure body */
    ENDPROCEDURE INITIATE_CONNECT;
    PROCEDURE RESPOND_CONNECT
        SIGNAL      MACRO signals;
        IN          conn_params     STRING ,
        IN/OUT      tc_id           PID ,
        IN          nc_id           PID ,
        IN/OUT      tpdu_size       INT ,
        IN/OUT      cstat INT /* values DATA_TRANSFER, DISCONNECTED */ ;
      DCL
        n_conn_params STRING ,
        prop_tpdu_size INT ,
        tpdu_flag       INT /* values CC, DR, ER, CR, DT, INV */ ,
        tpdu_data       STRING ,
        nbin            STRING ,
        nbout           STRING ,
        tbin            STRING ,
        tbout           STRING ,
        taskr           INT /* values CLOSE, NC, TC, NC&TC, NONE */
      /* procedure diagram, figure D-228, gives procedure body */
    ENDPROCEDURE RESPOND_CONNECT ;
```

FIGURE D-222 (page 2 de 3)

Spécification du système à l'aide du LDS/PR

```
PROCEDURE DATA_PHASE;
    SIGNAL      MACRO signals;
    IN          ref             STRING ,
    IN          tc_id           PID ,
    IN          nc_id           PID ,
    IN          tpdu_size       INT ,
    IN/OUT      cstat           INT /* values DATA_TRANSFER, DISCONNECTED */,
    IN/OUT IMPORTED /* from tc_id via channel TSAP_in */
    t_ind_flo                   BOOL ,
    IN/OUT IMPORTED /* from nc_id via channel NSAP_in */
    n_req_flo                   BOOL ,
    IN/OUT EXPORTED /* to tc_id via channel TSAP_out */
    t_req_flo                   BOOL ,
    IN/OUT EXPORTED /* to nc_id via channel NSAP_out */
    n_ind_flo                   BOOL ;
  DCL
    input_present               BOOL ,
    output_present              BOOL ,
    tbin                        STRING ,
    nbin                        STRING ,
    tbout                       STRING ,
    nbout                       STRING ,
    output_segment              STRING ,
    input_tsdu                  STRING ,
    tpdu_flag                   INT /* values CC, DR, ER, CR, DT, INV */ ,
    taskr                       INT /* values CLOSE, NC, TC, NC&TC, NONE */;
  /* procedure diagram, figure D-229, gives procedure body */,
ENDPROCEDURE DATA_PHASE;
PROCEDURE RELEASE_CONNECT
    SIGNAL      MACRO signals;

    IN          taskr           INT /* values CLOSE, NC, TC, NC&TC, NONE */,
    IN          tc_id           PID ,
    IN          nc_id           PID ,

    IN/OUT      cstat           INT /* values DATA_TRANSFER, DISCONNECTED */ ,
    IN/OUT      tpdu_flag       INT /* values CC, DR, ER, CR, DT, INV */ ,
    IN/OUT      last_tpdu       STRING;
  DCL
    nbin                        STRING ,
    nbout                       STRING ,
    tbout                       STRING ;
  /* procedure diagram, figure D-230, gives procedure body */
ENDPROCEDURE RELEASE_CONNECT;
/* process diagram, figure D-226, gives process body */
ENDPROCESS ENDPOINT;
ENDBLOCK TRANSPORT_ENTITY;
ENDSYSTEM TRANSPORT_ENTITY;
```

FIGURE D-222 (page 3 de 3)

Spécification du système à l'aide du LDS/PR

D.9.4.3 *Opérations portant sur les éléments de données*

Les opérations portant sur les éléments de données sont représentées de façon informelle. Elles sont représentées ci-dessous:

alloc_ref: gives a unique Transport Protocol connection reference

form_t_conn_ind:
form_t_conn_cfm:
form_t_data_ind:
form_t_disc_ind:
form_n_conn_req:
form_n_data_req:
form_n_conn_rsp:
form_n_disc_req:

Each of these operations forms the set of data fields for the corresponding service primitive from the parameters supplied, together with background knowledge of the Transport entity (e.g. mapping between transport and network addresses in the case of n_conn_req)

conn_feasible: Transport entity knowledge at the calling TSAP indicates that the connection may be achieved (local resources available, called TSAP reachable, quality of service achievable) – boolean

conn_acceptable: the Transport entity at the called NSAP has resources to accept the network connection – boolean

conn_providable: Transport entity knowledge at the called end indicates that the transport connection may be achieved (called TSAP available, quality of service achieved) – boolean

tpdu-type: determines the content of n_data_ind (values: CR, CC, DT, DR, ER or INV – the last indicates a TPDU invalid for any reason)

prop_tpdu_size: initiator's proposal for value of tpdu_size

agree_tpdu_size: responder's agreed value for tpdu_size

extract_tpdu_size: determines parameter value contained in CC TPDU

take_segment: takes next DT TPDU from TSDU

append_segment: adds current DT TPDU to partially completed TSDU

last_tpdu: current DT TPDU has End of TSDU mark set

Les valeurs d'entrée aux opérations sont représentées entre parenthèses après le nom de l'opération.

FIGURE D-223

Diagramme montrant le rôle d'un ORGANISME_ DE_ TRANSPORT dans une connexion de transport

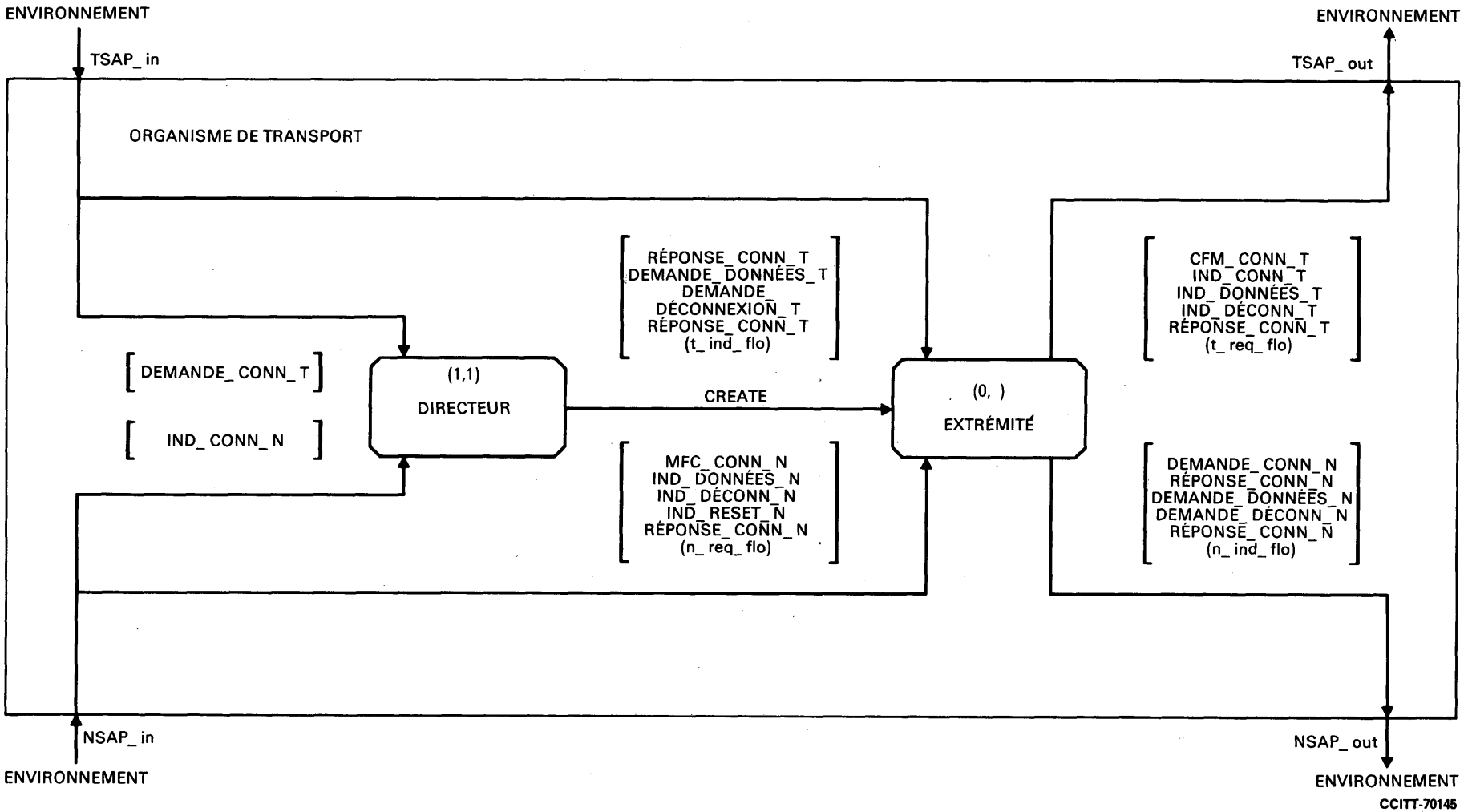


FIGURE D-224

Diagramme d'interaction pour bloc d'organisme de transport

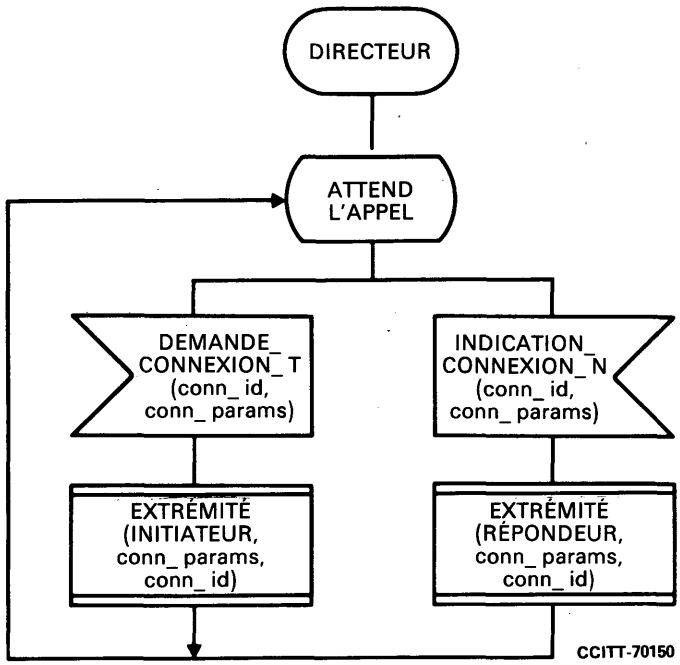


FIGURE D-225

Diagramme de processus pour DIRECTEUR

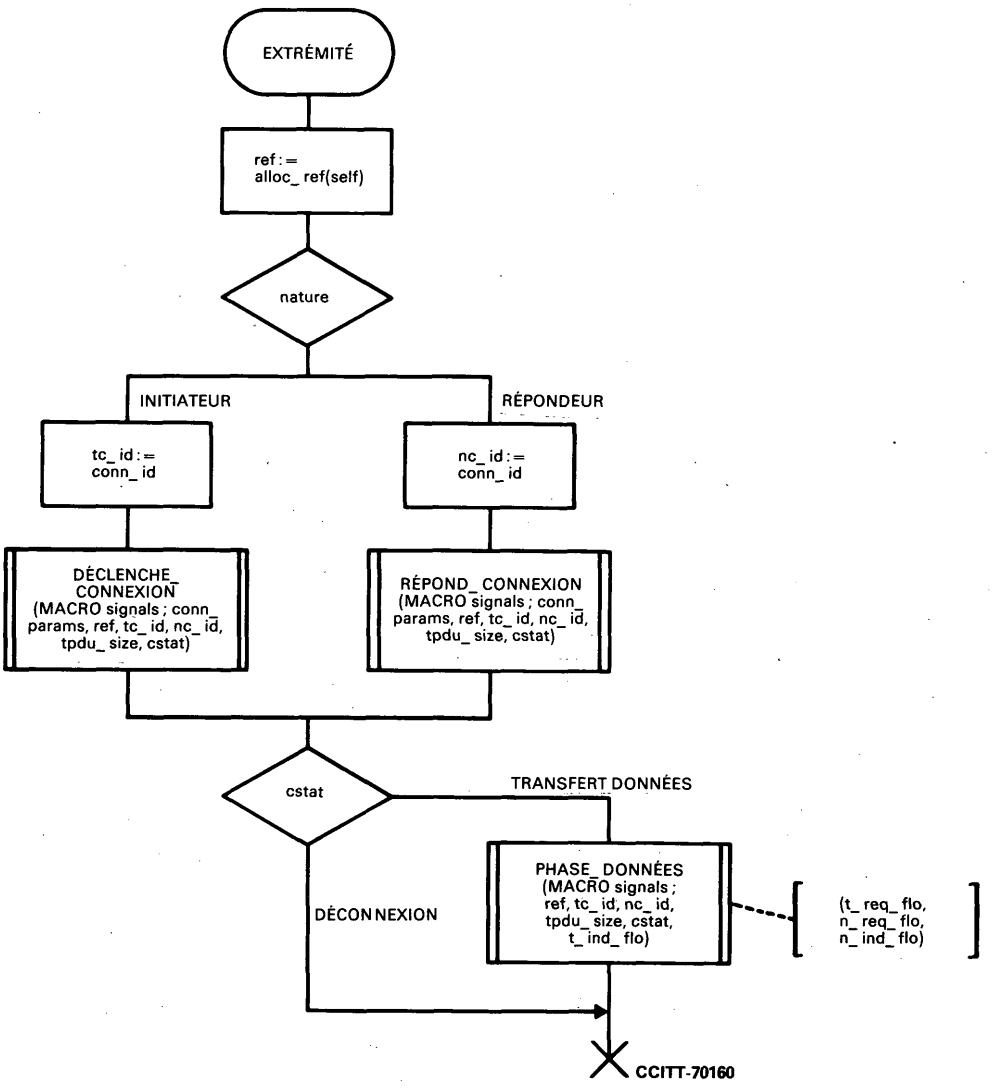


FIGURE D-226

Diagramme de processus pour EXTRÉMITÉ

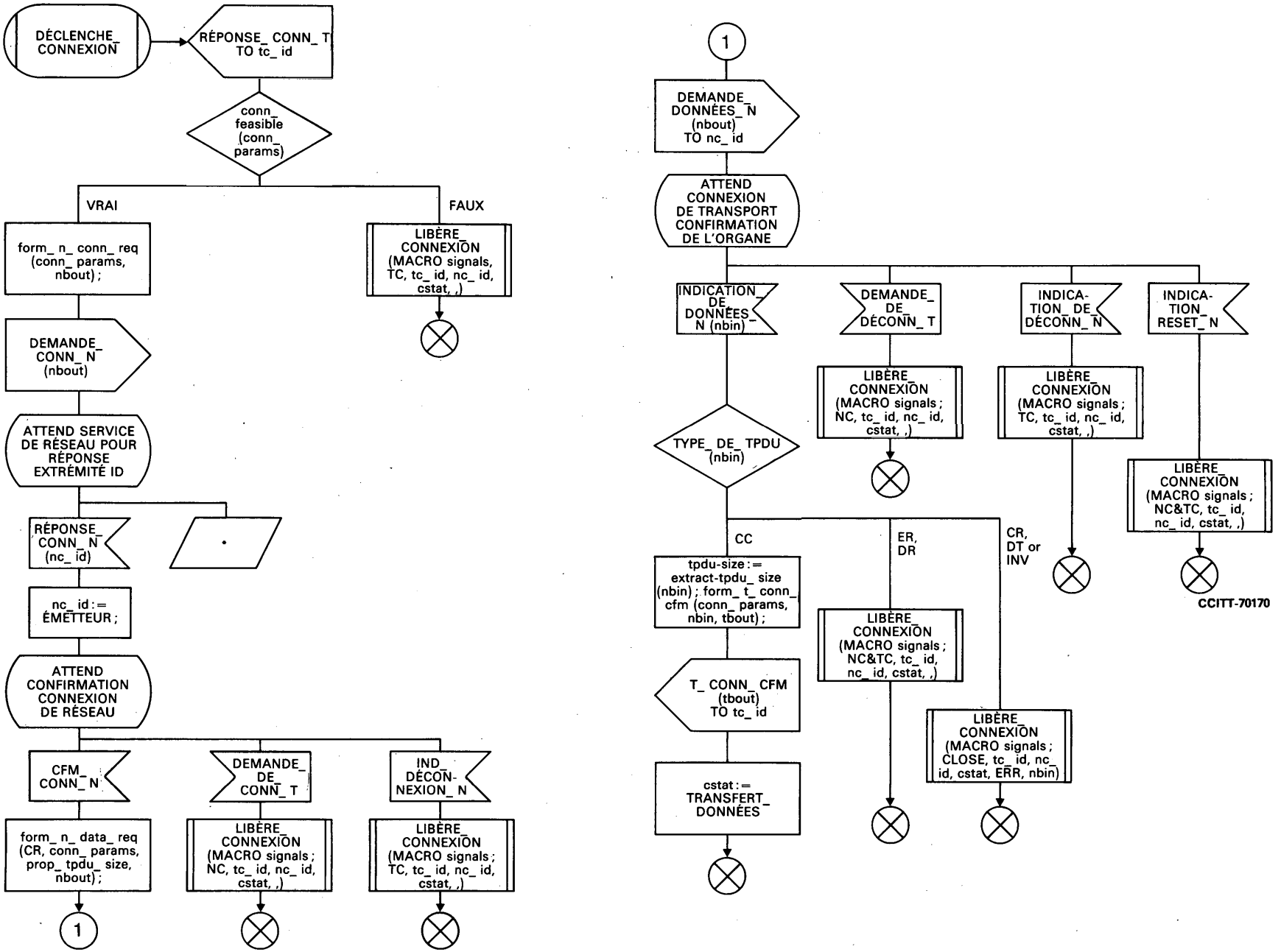


FIGURE D-227

Diagramme de procédure pour DÉCLENCHE_ CONNEXION

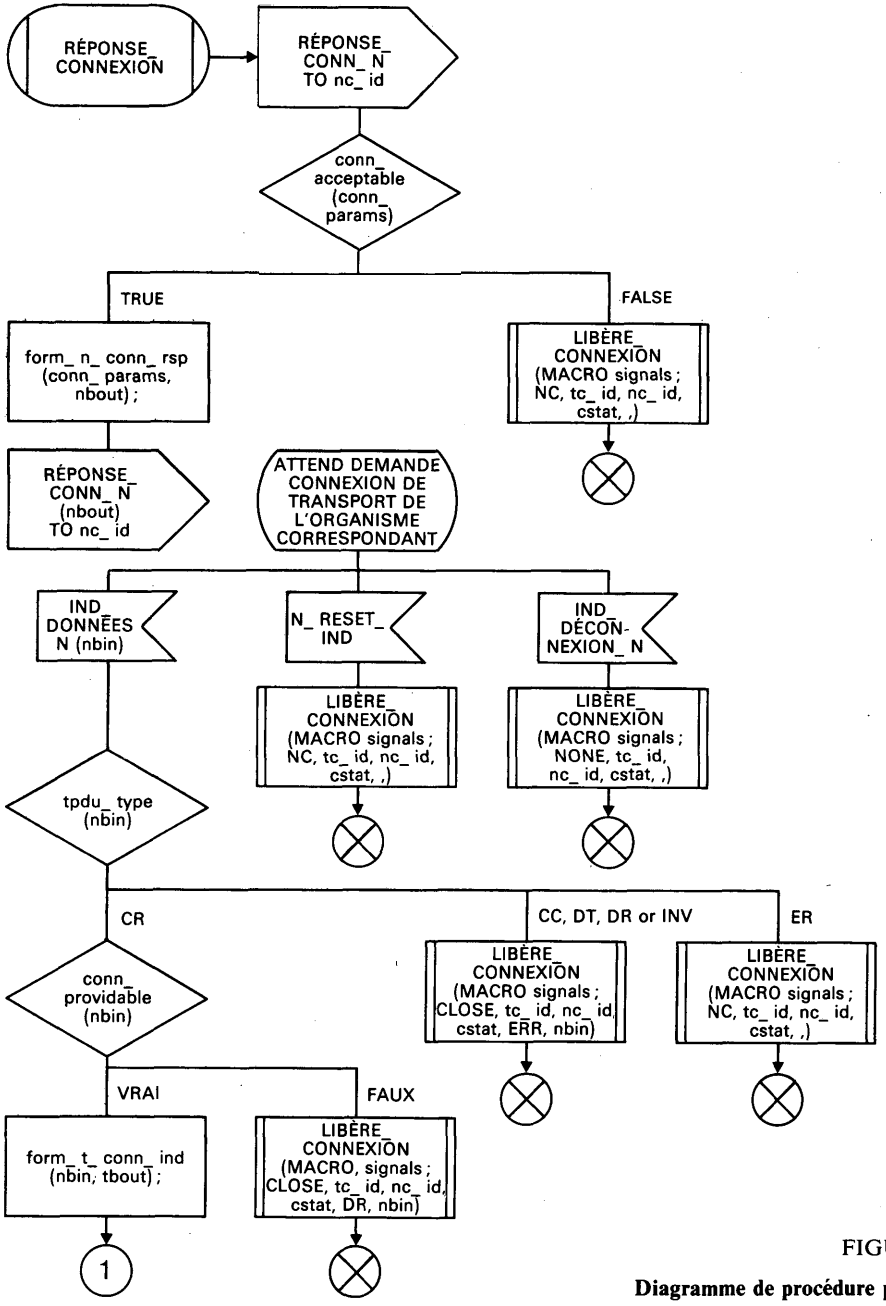


FIGURE D-228

Diagramme de procédure pour RÉPONSE_ CONNEXION

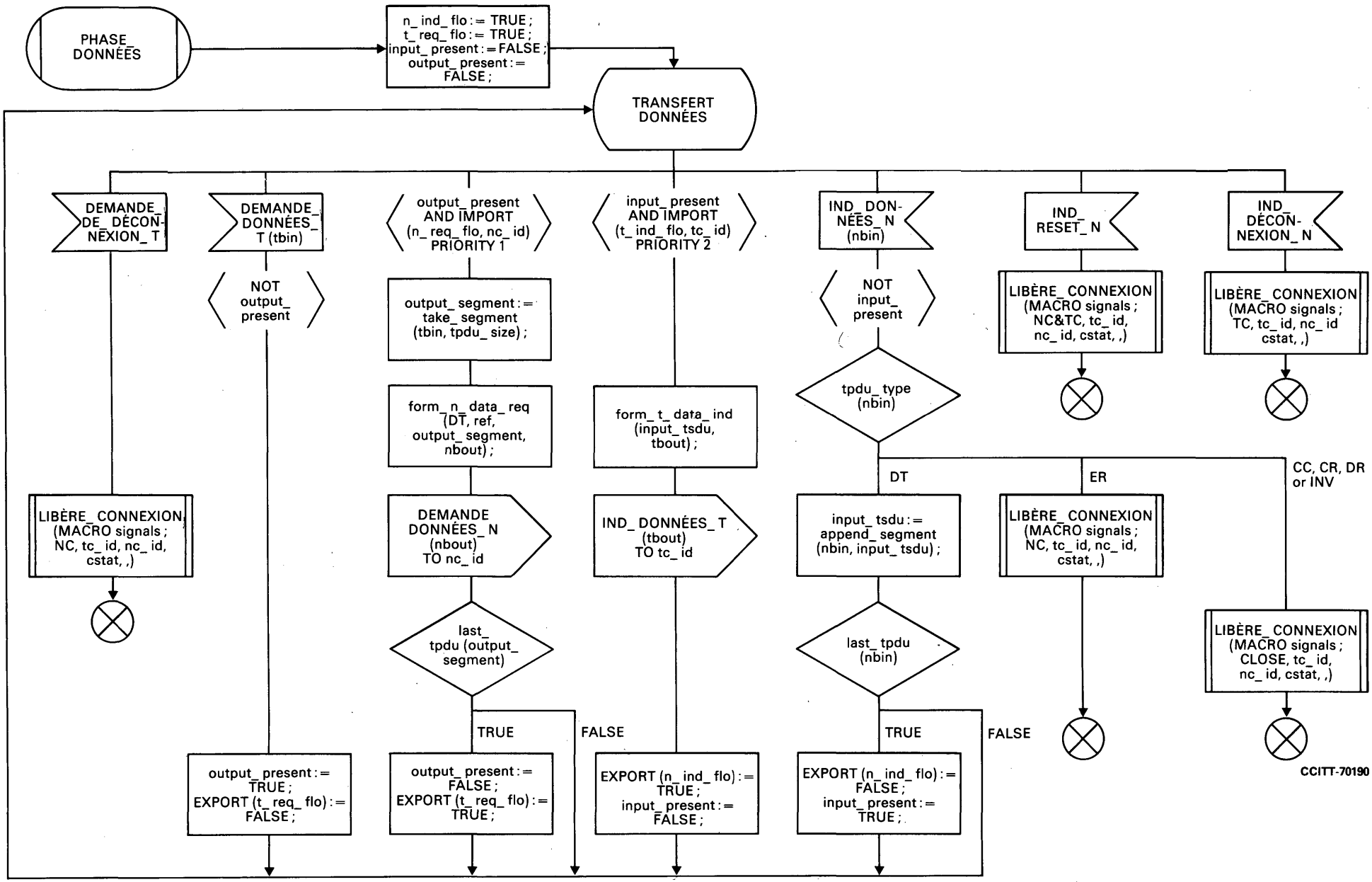


FIGURE D-229

Diagramme de procédure pour PHASE_ DONNÉES

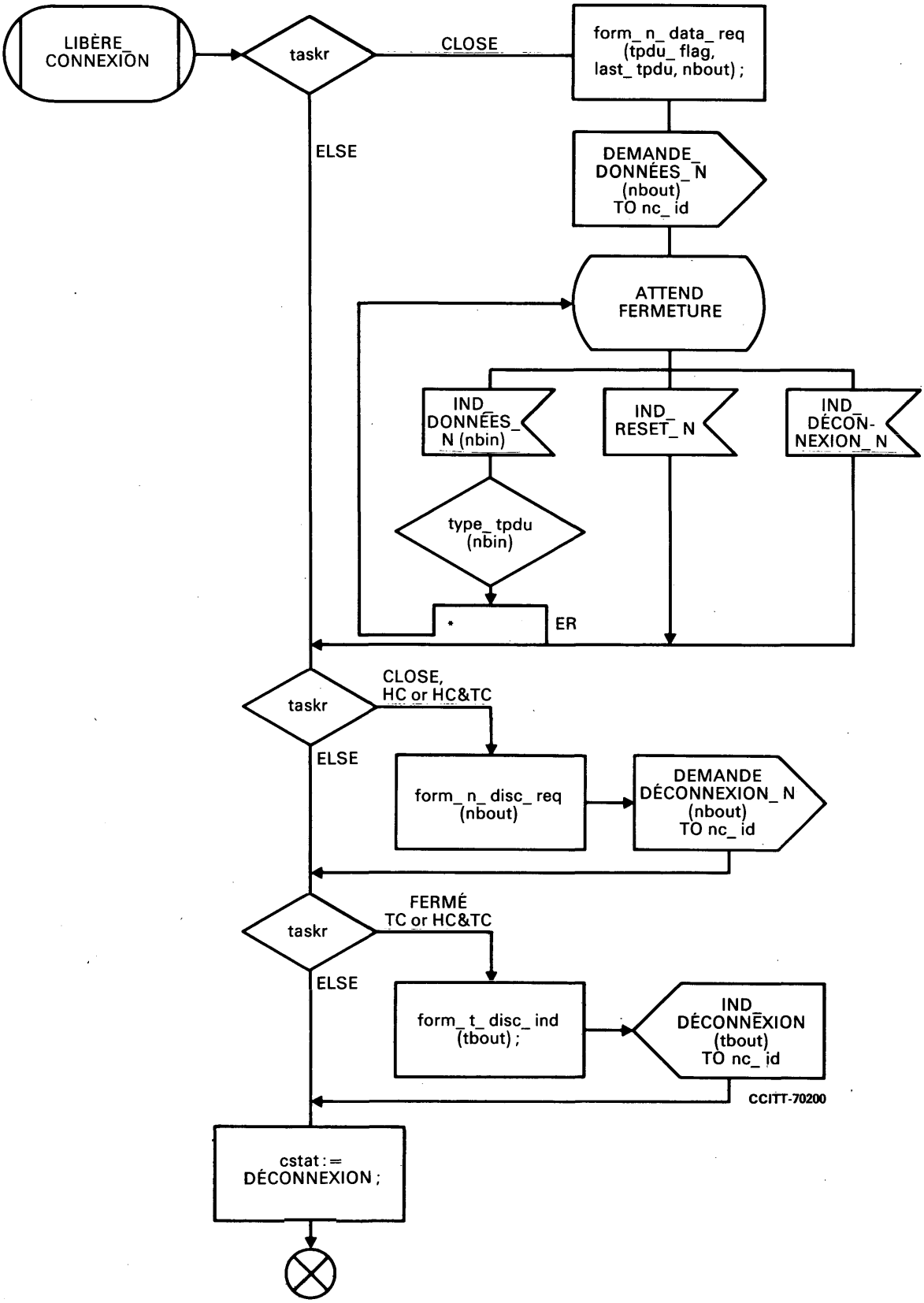


FIGURE D-230

Diagramme de procédure pour LIBÈRE_ CONNEXION

D.10 *Outils pour le LDS*

D.10.1 *Introduction*

Ce paragraphe présente une série d'outils de soutien pour le LDS. Ces outils peuvent aider à la production de documents, de diagrammes LDS (forme GR) ou de listes imprimées (forme PR), et/ou à la validation des représentations LDS.

Ces directives ne contiennent pas une liste exhaustive de tous les outils éventuels. Les outils nécessaires dépendent de la méthode choisie par l'usager.

En principe, le LDS peut être utilisé sans outils. Toutefois, la complexité inhérente des systèmes modernes est telle que les représentations LDS se révèlent souvent compliquées. De ce fait, il est nécessaire de disposer d'outils automatiques pour préparer la spécification, la conception et la documentation de plusieurs systèmes. Par exemple, la complexité et le coût des tracés manuels, et éventuellement, la mise à jour des documents graphiques d'un central de commutation seraient réduits de façon significative, grâce à l'utilisation d'aides appropriées.

En raison des considérations ci-dessus, le LDS a été conçu de façon à intégrer l'utilisation efficace d'outils de soutien.

D.10.2 *Catégories d'outils*

Les outils LDS peuvent être classés en fonction des activités effectuées dans le cadre de la production de documents LDS, par exemple:

* Outils pour les entrées

Selon les formes syntaxiques, nous disposons d'aides d'entrée pour les graphiques LDS, sous forme de membres de phrases ou d'illustrations.

* Outils pour la vérification des syntaxes

Ils comprennent notamment des analyseurs de syntaxe, pour chacune des trois syntaxes.

* Outils pour la production de documents

Une fois les documents LDS enregistrés sur ordinateur, les outils peuvent y avoir accès et les reproduire, en utilisant éventuellement plusieurs périphériques. Ces derniers peuvent utiliser une forme syntaxique différente de celle utilisée pour introduire le document. En outre, les outils peuvent produire des documents à partir de ceux initialement introduits.

* Outils de modélisation et d'analyse des systèmes

A partir des documents LDS représentant un système, on peut tirer un modèle du système. Des vérifications peuvent être faites sur ce modèle. Les outils peuvent rechercher les blocages, faire des comparaisons entre les divers modèles du même système (soit, par exemple, entre une spécification et une description) ou exécuter une simulation du fonctionnement du système, etc.

* Outils pour l'élaboration de codes

Des représentations LDS très détaillées peuvent être utilisées aux fins d'élaboration du logiciel. Les outils peuvent être conçus de façon à pouvoir préparer un questionnaire de code dirigé, semi-automatique.

Il existe également une catégorie d'outils spécifiques mais utiles:

* Les outils de formation LDS

Ils peuvent être utilisés soit seuls, soit intégrés à d'autres outils. Cette intégration permet de les utiliser pour d'autres fonctions, si nécessaire.

Etant donné que le LDS est utilisé pour plusieurs phases du cycle de vie des systèmes, il est facile de voir l'utilisation qui peut être faite de tous les types d'outils dans un environnement de projet intégré.

D.10.3 *Entrée des documents*

Aucune spécification particulière n'est requise pour l'introduction de la forme membres de phrases du LDS, étant donné que la syntaxe PR équivaut, du point de vue de l'entrée, à toute entrée de chaîne de caractères. En conséquence, on peut utiliser les mêmes outils (éditeurs de chaînes). Les deux autres syntaxes supposent toutefois une possibilité de traitement graphique.

C'est un fait qu'il est avantageux de disposer d'outils de soutien pour l'introduction du PR, mais il est indispensable d'avoir des outils de soutien pour l'introduction du GR/PE si nous prévoyons d'utiliser ces syntaxes comme moyens d'entrée.

Il est possible d'étudier de façon conjointe, l'introduction des documents GR/PE, dans la mesure où les deux syntaxes utilisent des graphiques. Il faut disposer d'un éditeur de graphiques et d'une unité de sortie des diagrammes sous forme graphique. Une unité d'entrée des graphiques n'est pas requise étant donné qu'il est possible d'introduire un diagramme sur une grille prédéfinie. Chaque symbole peut être asocié à une chaîne donnée. Par exemple, il est possible d'introduire un état en donnant sa position sur une grille (couple de coordonnées, nombre carré, etc.) et le type (état) en tant que chaîne de caractères.

Une unité d'entrée de graphiques peut offrir un retour d'information immédiat à l'usager. Toutefois, la «méthode de la grille» serait plus rapide et plus facile à mettre en œuvre.

Un éditeur de graphiques est toujours nécessaire pour des fonctions telles que la connexion de deux symboles, le déplacement d'une série de symboles vers une autre partie de la page ou vers d'autres pages, et pour assurer l'enchaînement des suppressions (la suppression d'un symbole entraîne la suppression de la connexion à ce symbole). Comme pour les outils PR, les outils d'entrée GR/PE devront être conçus sur le modèle de la sémantique/syntaxe LDS. En conséquence, il devrait éliminer les connexions non valables et pousser les usagers à remplir toutes les parties non complètes, etc.

Lors de la conception des outils, on est confronté à plusieurs problèmes dus aux contraintes physiques des appareils à graphiques tels que la «résolution». Il est presque impossible de disposer d'un nombre suffisant de caractères qui soient lisibles tout en permettant la visualisation d'un nombre raisonnable de symboles sur l'écran.

Il faudrait passer en revue les solutions telles qu'une fenêtre de zoom ou des déplacements, mais ces solutions ne sont pas totalement satisfaisantes. On peut estimer qu'il n'est pas nécessaire d'avoir un haut niveau de résolution lorsque les diagrammes sont reproduits par l'usager, mais cela devient très souhaitable si les diagrammes sont produits directement par l'usager. Pour la même raison (nécessité d'un tableau synoptique offrant un certain nombre de détails) un niveau élevé de résolution est souhaitable dans la visualisation des diagrammes.

Les outils de soutien des unités d'entrée du PR peuvent être utiles: ils peuvent présenter rapidement à l'usager les mots clés PR attendus, en soulignant que certains mots clés de fermeture sont toujours manquants (tels que décision finale, état final, etc.).

Ils peuvent procéder immédiatement au «formatage» du PR selon les mots clés reçus, insérer automatiquement des séparateurs, et présenter à l'usager des clés de fonctions orientées vers le PR, etc.

La mise en œuvre de ces outils peut être basée sur des éditeurs de chaînes de caractères déjà existants qui peuvent être ensuite élargis de façon à inclure les éléments mentionnés ci-dessus.

D.10.4 *Vérification des documents*

Une fois que les documents sont enregistrés en mémoire, la mesure suivante consiste à les vérifier. Ils doivent tout d'abord être vérifiés un à un, puis avec les diagrammes correspondants jusqu'à ce que le système total soit vérifié.

Si l'entrée a été faite au moyen d'un outil conçu pour le LDS, il se pourrait qu'une bonne partie de la vérification pour chaque document ait déjà été effectuée.

Les erreurs dues à des opérations «pas possibles» (à savoir les entrées ou les mises en réserve précédées d'un quelconque élément, à l'exception d'un état) devront toutes être détectées et corrigées au cours de la phase d'entrée. La détection de certaines erreurs n'est toutefois possible qu'à la fin de la phase d'entrée, aussi bien sur un document unique que dans le cas de contradictions pouvant exister entre des documents.

Plusieurs règles LDS peuvent être automatiquement vérifiées. Par exemple, la nécessité pour toutes les sorties d'avoir une entrée correspondante.

Dans le cas d'une représentation à plusieurs niveaux, la conformité entre les niveaux peut être vérifiée, dans une certaine mesure.

Le modèle formel LDS peut être utilisé comme base pour élaborer une collection de procédures de vérification.

D.10.5 *Reproduction des documents*

Les documents LDS mis en mémoire doivent pouvoir être retrouvés, visualisés et reproduits. Il est nécessaire de disposer d'outils pour toutes ces activités. Il peut s'avérer utile de pouvoir trouver seulement une partie, ou un sous-ensemble, du document. La recherche peut être orientée vers un LDS, par exemple: «trouver tous les processus d'émission» d'un signal donné, ou «dans quels états» est exécutée une action donnée, etc. Les outils de visualisation des informations sont particulièrement importants lorsque l'information doit être visualisée au moyen de la syntaxe graphique. Les mêmes observations que celles faites pour l'entrée des documents dans les syntaxes GR/PE s'appliquent. La reproduction des documents dépend du type de document à reproduire, de la façon dont ces documents sont mis en mémoire et des caractéristiques du périphérique de sortie. Elle peut également dépendre de la façon dont ces documents ont été introduits. Les usagers peuvent désirer une sortie imprimée dans une syntaxe différente de celle utilisée au moment de l'introduction du document.

La reproduction des documents est perturbée par les contraintes pesant sur les périphériques de sortie. Par exemple, un diagramme peut être trop large pour pouvoir être adapté sur une feuille de papier donnée, et de ce fait, il doit être découpé en plusieurs parties. Il faut alors ajouter des connecteurs et des références. Il est quelquefois souhaitable de faire la distinction entre une «adjonction» faite par l'outil et les caractéristiques initiales d'entrée. D'autres contraintes physiques peuvent empêcher la sortie de toutes les informations disponibles, par exemple, une taille particulière de symbole peut s'avérer trop petite pour renfermer la totalité du texte correspondant. Plusieurs méthodes peuvent être choisies, éventuellement sur décision de l'usager. On peut notamment allonger le symbole, découper le texte, le découper mais en ajoutant le texte complet en bas de page, placer le texte à côté du symbole... On peut également souhaiter disposer d'outils permettant un plus grand choix de formats de sorties: ces éléments comprennent notamment différentes tailles de symboles, différents formats de sortie, une présentation verticale ou horizontale, etc.

Un document devrait toujours pouvoir être reproduit exactement de la même façon qu'il a été introduit.

D.10.6 *Production de documents*

Sur la base des documents LDS introduits par les usagers et enregistrés en mémoire, plusieurs autres documents peuvent être produits automatiquement notamment:

- les listes de signaux, regroupés par processus, par bloc ou par système;
- les diagrammes d'interaction de processus, représentant les interactions et les séquences consécutives des actions dans des processus communiquant entre eux;
- les diagrammes synoptiques d'état représentant les graphiques de processus comme un ensemble d'états reliés par des arcs représentant les transitions;
- les tableaux de référence, assemblés par processus, par bloc ou par système;
- le diagramme de découpe, montrant la structure des blocs et des niveaux;
- le fonctionnement du système, comme réponse aux séquences des actions de l'environnement;
- des index: ces documents, une fois produits, devront être reproduits et les même considérations susmentionnées seront également valables.

Les documents LDS introduits sous une forme GR peuvent automatiquement être traduits dans la forme PR équivalente et vice versa.

Pour produire une forme PR correcte, tous les diagrammes GR représentant des processus d'un bloc doivent être examinés conjointement avec les diagrammes d'interaction des blocs GR correspondants.

Les considérations suivantes s'appliquent:

- la forme GR contient des informations visuelles qui ne peuvent être traduites dans la forme PR (ce genre d'information n'existe pas en PR). Par exemple, les coordonnées des symboles sont sans signification dans la forme PR;
- les connecteurs reliant des lignes de liaison sur différentes pages peuvent être éliminés.

La traduction inverse, de PR vers GR, est toutefois plus complexe et il est probable qu'elle ne sera pas tout à fait satisfaisante pour tous les lecteurs éventuels. Hormis le décalage, il n'existe pas de critères subjectifs à l'élégance d'une forme PR, alors qu'il existe une large variété de critères subjectifs relatifs à la forme GR.

En raison de la représentation sur deux dimensions de la forme GR, certaines étiquettes qui ont été insérées afin de répondre à la structure séquentielle du PR, peuvent être supprimées, étant donné qu'une ligne de connexion est suffisante. On peut tirer d'une forme PR, deux représentations GR différentes, à savoir le diagramme d'interaction des blocs fonctionnels et le diagramme de processus (bien évidemment, si le PR représente un seul processus, alors on ne peut en dériver que le diagramme de processus).

Habituellement, la traduction produit un modèle de diagramme GR. Ce modèle comprend tous les renseignements nécessaires à un outil pour pouvoir procéder au formatage et reproduire le diagramme sur une imprimante graphique.

Il convient de noter que deux outils différents traduisant des PR en GR peuvent obtenir deux représentations GR ayant des présentations différentes. Les représentations GR ainsi obtenues sont toutes les deux correctes à condition qu'elles gardent la sémantique exprimée dans la représentation d'origine.

D.10.7 *Modélisation et analyse du système*

Les documents LDS, indépendamment de leurs fonctions de spécification ou de description d'un système, sont fondamentalement un modèle de ce système.

Ce modèle, qui est tout d'abord destiné au transfert de l'information d'une personne à une autre, peut également être interprété par des outils; ces derniers servent à vérifier si le modèle est conforme, complet (cet aspect peut éventuellement être laissé en suspens dans le cas des spécifications destinées à ne spécifier que certaines parties d'un système), s'il est correct et s'il répond aux règles du LDS (telles que décrites dans le paragraphe relatif à la vérification des documents).

En outre, les outils peuvent être conçus de façon à utiliser le modèle pour simuler le comportement fonctionnel des systèmes. Le simulateur peut entrer en interaction avec l'environnement et on peut alors tirer les conclusions quant à l'adéquation du modèle par rapport aux besoins des usagers.

Si l'on ajoute des renseignements supplémentaires pour indiquer le temps passé à exécuter chaque action, et pour évaluer les ressources disponibles, (queues, instances, etc.), la simulation peut également étudier la capacité du sytème.

Les outils doivent être élaborés de façon à créer un modèle de l'environnement, en commençant par le modèle du système, afin d'établir des séquences significatives pour contrôler le système actuel. Une analyse de trajet peut permettre de détecter les blocages dans le modèle.

Le modèle de système peut également être utilisé comme documentation directe. S'il existe des liaisons appropriées entre le système réel et la mémoire de la documentation, on peut élaborer un outil chargé d'imprimer les événements en temps réel du système sur le modèle.

Pour cela, il faudrait établir une corrélation entre les événements physiques tels que vus par le système et les événements logiques traités dans la documentation LDS. Si la documentation est regroupée en plusieurs niveaux d'abstraction, l'usager peut choisir le niveau à imprimer. Cet aspect peut être très utile dans la mesure où il permet aux usagers ayant des niveaux de formation et d'éducation différents, d'inspecter les activités du système.

Les outils destinés à interpréter le modèle LDS peuvent également être utilisés pour faire ressortir les différences de comportement des différents modèles du même système. Ils peuvent également être utilisés pour comparer les différentes descriptions du système (systèmes produits par différentes compagnies), ou pour comparer la spécification du système avec sa description. De cette manière, il est possible de vérifier si une description du système est conforme à la spécification originale.

D.10.8 *Elaboration de code*

Grâce à une syntaxe formellement définie et à une définition mathématique formelle du LDS, il est possible de concevoir des outils capables de faire correspondre la sémantique des représentations LDS et la sémantique des langages de programmation. Ces outils sont peut-être incapables de fournir des programmes complets d'application, mais ils peuvent s'avérer très utiles pour fournir au moins le cadre général pour un programme effectif.

Le § D.8.1 de ces directives à l'intention des usagers, offre un exemple de la façon dont peut être obtenue la mise en correspondance des constructions LDS et CHILL.

D.10.9 *Formation*

Un cours complet de formation pour le LDS a été mis sur pied. Il comprend environ 200 pages de texte et une collection de diapositives (environ 200). Le cours couvre tous les aspects du langage et fournit des exemples et quelques propositions quant à l'utilisation du LDS.

Le cours LDS peut être obtenu auprès de l'Union internationale des télécommunications, Secrétariat général – Section des ventes, Place des Nations, CH-1211 Genève 20 (Suisse).

Imprimé en Suisse — ISBN 92-61-02242-1