

FASCICULE X.2

Annexe D à la Recommandation Z.100

DIRECTIVES POUR LES USAGERS DU LDS

Blanc

MONTAGE: PAGE 2 = PAGE BLANCHE

DIRECTIVES POUR LES USAGERS DU LDS**TABLE DES MATIERES**

		Page D.1	Préface /
		D.2	Introduction /
D.2.1	Aperçu général du LDS /		
D.2.1.1	Le LDS fondé sur un modèle de machine à état finis étendue /		
D.2.2	Forme syntaxique du LDS /		
D.2.3	Applicabilité du LDS /		
		D.3	Concepts de base du LDS /
D.3.1	Système /		
D.3.2	Blocs /		
D.3.3	Canaux /		
D.3.4	Signaux /		
D.3.5	Acheminement des signaux /		
D.3.6	Diagrammes de système et de bloc /		
D.3.7	Commentaires et extension de texte /		
D.3.7.1	Commentaires /		
D.3.7.2	Extension de texte /		
D.3.8	Processus /		
D.3.8.1	Création de processus /		
D.3.8.2	Etats /		
D.3.8.3	Entrées /		
D.3.8.4	Mises en réserve /		
D.3.8.5	Condition de validation et signaux continus /		
D.3.8.6	Sorties /		
D.3.8.7	T ches /		
D.3.8.8	Décisions /		
D.3.8.9	Branchements et connecteurs /		
D.3.9	Procédures /		
D.3.9.1	Corps de procédure /		
D.3.9.2	Appel de procédure /		
D.3.10	Traitement des données /		
D.3.10.1	Déclarations variables /		
D.3.10.2	Variables révélées/vues /		
D.3.10.3	Variables exportées/importées /		
D.3.10.4	Expressions /		
D.3.11	Expression du temps en LDS /		
D.3.12	Utilisation de qualificatifs /		
D.3.13	Syntaxe de noms /		
		D.4	Structuration et affinage des systèmes en LDS /
D.4.1	Considérations générales /		
D.4.2	Critères de subdivision /		
D.4.3	Subdivision des blocs /		
D.4.4	Diagramme d'arbre de blocs /		
D.4.5	Division des canaux /		
D.4.6	Représentation du système en cas de subdivision /		
D.4.6.1	Sous-ensemble de subdivision cohérent /		

D.4.7	Affinage /	
D.4.7.1	Sous-ensemble d'affinage cohérent /	
D.4.7.2	Transformation entre signaux et sous-signaux /	
		D.5
		Concepts supplémentaires /
D.5.1	Macros /	
D.5.2	Systèmes génériques /	
D.5.3	Services /	
D.5.3.1	Considérations générales /	
D.5.3.2	Signaux prioritaires /	
D.5.3.3	Transformation /	
D.5.4	Directives applicables à la représentation en fonction des états et aux éléments graphiques /	
D.5.4.1	Observations d'ordre général sur la représentation en fonction des états /	
D.5.4.2	Illustration d'état et éléments graphiques /	
D.5.5	Diagrams auxiliaires /	
D.5.5.1	Diagramme synoptique d'état /	
D.5.5.2	Matrice état/signal /	
D.5.5.3	Diagramme de séquençement /	
		D.6
		Définition de données en LDS /
D.6.1	Directives applicables aux données en LDS /	
D.6.1.1	Introduction générale /	
D.6.1.2	Sortes /	
D.6.1.3	Opérateurs, littéraux et termes /	
D.6.1.4	Equations et axiomes /	
D.6.1.5	Informations complémentaires concernant les équations et les axiomes /	
D.6.2	Générateurs et héritage /	
D.6.2.1	Générateurs /	
D.6.2.2	Héritage /	
D.6.3	Observations relatives aux équations /	
D.6.3.1	Considérations générales /	
D.6.3.2	Application de fonctions aux constructeurs /	
D.6.3.3	Spécification d'ensemble d'essai /	
D.6.4	Caractéristiques /	
D.6.4.1	Opérateurs cachés /	
D.6.4.2	Relation d'ordre /	
D.6.4.3	Sortes avec champs /	
D.6.4.4	Sortes indexées /	
D.6.4.5	Valeur par défaut de variables /	
D.6.4.6	Opérateurs actifs /	
		D.7
		Directives supplémentaires pour le dessin et l'écriture /
D.7.1	Directives pour le LDS/GR /	
D.7.1.1	Considérations générales /	
D.7.1.2	Points d'entrée et de sortie /	
D.7.1.3	Symboles /	
D.7.1.4	Gabarit /	
D.7.2	Directives applicables au LDS/PR /	
		D.8
		Documentation /
D.8.1	Introduction /	
D.8.2	Types de représentation de systèmes /	
D.8.3	Structure de documents /	
D.8.4	Mécanisme de référence /	
D.8.5	Classification des documents /	
D.8.6	Combinaison de LDS/GR et de LDS/PR /	
		D.9
		Mises en correspondance /
D.9.1	Mise en correspondance du LDS et du CHILL /	

D.9.2	Mise en correspondance du LDS/GR et du LDS/PR /	D.10	Exemples d'application /
D.10.1	Introduction /		
D.10.2	Le concept de service /		
		D.11	Outil pour le LDS /
D.11.1	Introduction /		
D.11.2	Cat'egories d'outils /		
D.11.3	Entr'ee des documents /		
D.11.4	V'erification des documents /		
D.11.5	Reproduction des documents /		
D.11.6	Production des documents /		
D.11.7	Mod'elisation et analyse du syst`eme /		
D.11.8	G'eneration de code /		
D.11.9	Formation /		

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 regroupées en 1984 dans les Recommandations Z.100 à Z.104 (Livre rouge). Au cours de la période d'études 1985-1988, le langage a été encore développé et harmonisé; les Recommandations existantes ont été fondues en une seule et une définition mathématique a été ajoutée.

Des directives sont indispensables aux utilisateurs pour faciliter l'utilisation du LDS dans ses applications à une large gamme de systèmes de télécommunication. Ces directives ont pour but d'aider les utilisateurs à comprendre la Recommandation concernant le LDS et son application à différents secteurs.

L'emploi du LDS est largement répandu au sein du CCITT et des organisations qui en sont membres; en outre, la gamme des applications de ce langage ne cesse de se développer. Les présentes directives sont établies à l'intention de ceux qui envisagent d'utiliser ou utilisent déjà le LDS; elles complètent la Recommandation sur le LDS en y ajoutant des conseils judicieux et des exemples utiles. Il y aura certes quelques chevauchements entre les directives et la Recommandation; cela semble d'ailleurs souhaitable, si l'on veut que les directives soient autonomes et faciles à consulter. C'est néanmoins la Recommandation qui constitue le document de base.

D.2 *Introduction*

D.2.1 *Aperçu général du LDS*

Le LDS peut servir à spécifier le fonctionnement que l'on attend d'un système et à décrire le 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 (§ D.2.1.1) 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 oeuvre du système, et de documentation (descriptions) après cette même mise en oeuvre.

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 écrire 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 écrire un système de manière plus ou moins formelle.

Premièrement, il est possible de écrire 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 spécification 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 oeuvre>> 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 oeuvre les systèmes écrits.

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.

Dans le texte qui suit, le terme spécification sera utilisé à la fois pour la représentation nécessaire et pour celle des comportements réels.

D.2.2.1 *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 qui 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 le comportement futur 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 séquençement de la machine principale. Les << | ches >> 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 entrée, 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.2.2 Formes syntaxiques du LDS

Le LDS est un langage qui se présente sous deux formes différentes, fondées toutes deux sur le même modèle sémantique. L'une est appelée LDS/GR (LDS graphical representation) et repose sur un ensemble de symboles graphiques normalisés. L'autre s'appelle LDS/PR (LDS textual phrase representation) et repose sur des instructions analogues à un langage de programmation. L'une et l'autre représentent les mêmes concepts du LDS.

Un langage graphique présente l'avantage de montrer clairement la structure d'un système et de permettre à des | tres humains de visualiser facilement le flux de contr | le. La représentation textuelle de phrases convient mieux à l'utilisation par des machines.

En tant qu'outil de conception, le LDS devrait | tre présenté sous une forme permettant à l'utilisateur d'exprimer ses idées clairement et avec concision. Le LDS/GR, qui permet de le faire, correspond davantage à la représentation traditionnelle des machines à états finis étendus.

Le LDS/GR est la forme originale du LDS. Il a été conçu entre 1973 et 1976 a été publiée pour la première fois dans la version de 1976 des Recommandations de la série Z.100.

Le LDS/GR a été établi sur la base de langages graphiques élaborés par différentes organisations pour leurs propres utilisations.

La représentation textuelle de phrases du LDS, c'est-à-dire le LDS/PR, a été conçu pendant la période d'études 1977-1980 mais il a fallu y apporter certaines améliorations avant qu'elle puisse faire l'objet d'une Recommandation. Ces améliorations ont été faites au cours de la période d'études suivante et, dès 1984, le LDS/PR est devenu l'une des syntaxes concrètes recommandées du LDS.

Dans un premier temps, le LDS/PR devait | tre utilisé comme un moyen aisé d'introduire des documents en LDS dans une machine, ce qui était trop difficile avec le GR (en effet, cela nécessitait l'intervention d'équipements périphériques graphiques). C'est pour cette raison que l'on a insisté sur une mise en matière de terminaux graphiques (capacités accrues et réduction des coûts) ont fait que le GR est désormais susceptible d' | tre introduit en machine. Cela ne diminue en rien l'importance et l'utilisation du PR car certains utilisateurs le trouvent plus à leur convenance, particulièrement ceux qui travaillent avec des langages de programmation.

Du fait de cette évolution, la corrélation entre le GR et le PR est moins étroite; cependant, il est encore possible de mettre en correspondance sans difficulté l'une de ces représentations avec l'autre, bien que chacune d'elles ait ses propres particularités. A première vue, le PR ressemble fortement à un langage de programmation (voir la figure D-2.2.1).

Figure D-2.2.1 [T1.100] (à traiter comme tableau MEP, p.

En fait, tout dépend de ce qui caractérise un texte du point de vue du langage de programmation.

Si nous admettons qu'un programme est défini comme une <<information interprétable par une machine >>, non seulement les PR mais aussi les GR sont des <<programmes >>.

Il existe cependant certaines différences entre une spécification en LDS et un programme réel. Tout d'abord, il n'est pas indispensable qu'une spécification en LDS puisse | tre exécutée par une machine (bien que cela ne soit pas

interdit); ce qui est essentiel, c'est sa capacité d'acheminer des renseignements précis d'un |tre humain à un autre |tre humain.

Si nous considérons une spécification en LDS comme un programme, ce qui peut |tre tenu pour une <<spécification en LDS erronée>> (en raison d'un texte informel incomplet) pourrait |tre parfaitement valable si elle était considérée comme une représentation des caractéristiques fonctionnelles d'un système.

Une autre différence réside dans le <<style>> d'une spécification en LDS, si on la compare à la représentation usuelle d'un programme.

Le LDS ayant pour but de faciliter la communication entre |tres humains, on s'est efforcé de préserver la possibilité de différentes présentations, afin que la présentation en LDS puisse servir à orienter le lecteur vers certains aspects considérés comme plus importants que d'autres. Cela est naturellement sans importance pour un programme, qui est censé

|tre interprété par une machine. La machine n'insiste pas sur un quelconque aspect particulier du programme, mais doit traiter de la m |me manière son ensemble; en outre, elle n'essaie pas de <<comprendre>> le programme.

Etant donné ses similitudes avec un programme, le PR a la préférence de certains programmeurs qui utiliseront probablement aussi le CHILL pour la mise en oeuvre des besoins. On serait donc probablement tenté de rechercher une correspondance point par point entre le PR et le CHILL, afin que les besoins exprimés en PR puissent |tre automatiquement transformés en code CHILL. L'inverse est également intéressant car cela permettrait la dérivation d'une spécification en PR à partir d'un programme en CHILL.

On trouvera au § D.9 des exemples de possibilités de mettre en correspondance le LDS avec le CHILL.

D.2.3 *Applicabilité du LDS*

La figure D-2.3.1 donne un éventail de possibilités d'emploi du LDS aux fins d'achat ou de fourniture 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 |tre 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 téléx ou de données, systèmes de signalisation (par exemple, système de signalisation n^o 7), interfonctionnement des systèmes de signalisation, protocoles pour données et interfaces d'utilisateurs (LHM).

Si l'on considère plus spécialement les systèmes de commutation SPC, on peut citer comme exemple de fonctions pouvant |tre spécifiées ou écrites 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.10 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.

Figure D-2.3.1, p.

D.3 *Concepts de base du LDS*

Comme on a pu le voir 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 bidirectionnel 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 par des canaux. Chaque bloc du système est indépendant de tous les autres. Chaque bloc peut contenir un ou plusieurs processus écrivant le comportement du bloc. Seule l'émission de signaux dans les canaux permet la communication entre des processus placés dans deux blocs différents. Pour subdiviser le système en blocs, on peut prendre des critères tels que les suivants: définir des parties d'une dimension facilitant la compréhension, créer une correspondance avec la division effective entre le logiciel et le matériel, suivre les subdivisions fonctionnelles naturelles, réduire au minimum les interactions, etc.

Pour de grands systèmes LDS, il existe certaines constructions en LDS qui permettent de spécifier les sous-structures des parties d'un système, de sorte qu'en partant d'un aperçu général du système, on peut donner toujours plus de détails. Dans ce cas, on peut dire que le système est représenté à différents niveaux de détail. Ces constructions sont décrites au § D.4.

Au premier niveau de précision, la spécification en LDS d'un système décrit la structure du système et comprend les points suivants, expliqués dans les paragraphes qui suivent:

- nom du système;
- définitions de signaux: spécification des types de signaux échangés entre les blocs du système ou entre les blocs et l'environnement. Cela comprend la spécification des types de valeurs acheminées par les signaux (liste de sortes);
- définitions de listes de signaux: spécification des identificateurs groupant plusieurs signaux et/ou autres listes de signaux. De tels identificateurs peuvent servir à économiser de l'espace et à donner une spécification plus claire;
- définitions de canaux: spécification des canaux reliant les blocs du système les uns aux autres et à l'environnement. Une définition de canal comprend la spécification des identificateurs des signaux acheminés par ce canal;
- définitions de données: spécification de nouveaux types de syntypes et de générateurs définis par l'utilisateur et visibles dans tous les blocs;
- définitions de blocs: spécification des blocs dans lesquels le système est divisé;
- définitions de macros: des directives concernant l'utilisation des macros sont données au § D.5.1.

Selon la Recommandation concernant le LDS, il existe des types de données prédéfinis qui peuvent être utilisés par chaque système. Ils ne nécessitent pas de définition et peuvent être utilisés au moyen de leurs noms prédéfinis, c'est-à-dire: INTEGER, REAL, CHARACTER, STRING, CHARSTRING, BOOLEAN, PID, TIME, DURATION. Les types de données prédéfinis sont visibles à tous les niveaux de la définition du système; ils peuvent être considérés comme définis implicitement dans une <<bibliothèque>> de système accessible en tout point de la spécification.

Les noms de sorte utilisés dans les définitions de signaux au niveau de système doivent être introduits par des définitions de type partielles visibles au niveau du système, c'est-à-dire des types de données prédéfinis ou des newtypes définis par l'utilisateur ou encore des syntypes définis à ce niveau.

On trouvera des explications complémentaires sur l'utilisation des types de données aux § D.3.10 et D.6.

Le LDS/PR utilisé pour une définition de système consiste en un ensemble d'instructions se terminant par <<;>> (point-virgule). La définition d'une structure de système commence par l'instruction <<SYSTEM nom;>> et se termine par l'instruction <<ENDSYSTEM nom;>>. Le nom de l'instruction terminale est facultatif mais s'il est donné, il doit être le même que le nom donné après le mot de SYSTEM. Il est suggéré de placer toujours le nom dans l'instruction terminale, car cela améliore la lisibilité du document.

Le schéma du LDS/PR d'une définition de structure de système est représenté dans la figure D-3.1.1.

Figure D-3.1.1 [T2.100] (à traiter comme tableau MEP, p.

Pour permettre d'obtenir une représentation plus claire et plus simple de la structure du système ou permettre une spécification descendante de système, le LDS contient un mécanisme général de référence. A ce niveau, le mécanisme de référence peut

être appliqué aux définitions de bloc. Cette caractéristique du langage permet à l'utilisateur de spécifier précisément le nom de bloc à l'intérieur de la définition de la structure de système; la définition de bloc proprement dite peut être donnée séparément (voir la figure D-3.1.2).

Figure D-3.1.2 [T3.100] (à traiter comme tableau MEP, p.

Le mécanisme de référence est particulièrement utilisé dans le LDS/GR parce que la plupart des diagrammes doivent tenir sur une seule page et que la place manque souvent pour des spécifications graphiques embossées.

On trouvera des exemples de LDS/GR pour une définition de système au § D.3.6.

D.3.2 Blocs

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.

A l'intérieur d'un bloc, il est possible (à titre d'option) de définir les trajets de communication entre les processus ou entre ceux-ci et l'environnement du bloc (c'est-à-dire la frontière du bloc). Ces trajets de communications sont appelés acheminement de signaux.

Pour un système LDS de grandes dimensions, il est possible de écrire la sous-structure d'un bloc en fonction d'autres blocs et de canaux, comme si le bloc était un système en soi. Ce mécanisme est expliqué au § D.4.

La définition de la structure d'un bloc peut comporter les points suivants:

- nom de bloc;
- définitions de signaux: spécification des types de signaux échangés à l'intérieur du bloc. Cela comprend la spécification des types de valeurs acheminés par les signaux (liste de sortes);
- définitions de listes de signaux: spécifications ou identificateurs correspondant à des listes de signaux et/ou à d'autres identificateurs de listes de signaux. Ces identificateurs, groupant plusieurs signaux, peuvent être utilisés pour économiser de l'espace et obtenir une spécification plus claire;
- définitions d'acheminement de signaux: spécifications des trajets de communication reliant les processus du bloc les uns aux autres et à l'environnement du bloc. Une définition d'acheminement de signaux comprend la spécification des identificateurs de signaux transportés sur cet acheminement de signaux;

— connexions de canaux vers des acheminements: spécifications des connexions entre les canaux extérieurs au bloc et les acheminements de signaux internes du bloc;

— définitions de processus: spécification des types de processus décrivant le comportement du bloc. Si le bloc n'est pas décrit en fonction de sa sous-structure, il faut qu'il y ait au moins une définition de type de processus à l'intérieur du bloc. Pour la définition du processus, un mécanisme de référence est fourni comme cela est indiqué, dans le cas des blocs, au § D.3.1;

— définitions des données: spécification de newtypes, de syntypes et de générateurs définis par l'utilisateur et visibles dans tous les processus définis du bloc et/ou dans la structure-structure du bloc;

— définitions de macros: des directives pour l'utilisation de macros sont données au § D.5.1.

S'il existe une sous-structure à l'intérieur du bloc, certains des points ci-dessus sont facultatifs (voir le § D.4 pour les explications concernant la structure).

Les types suivants sont visibles dans un bloc:

— des types de données prédéfinis;

— des types de données définis par l'utilisateur, définis dans le bloc proprement dit;

— des types de données définis par l'utilisateur visibles dans le bloc ascendant (en cas de subdivision des blocs).

Dans le LDS/PR, les mots d'e BLOCK et ENDBLOCK servent à circonscrire une définition de bloc. On trouvera des exemples du LDS/GR pour une définition de bloc au § D.3.6.

D.3.3 Canaux

Les canaux sont les moyens de communication entre différents blocs du système ou entre des blocs et leur environnement. Un canal peut relier un bloc à un autre ou un bloc à l'environnement dans une direction (canal unidirectionnel) ou dans les deux directions (canal bidirectionnel). Normalement, un canal est une entité fonctionnelle qui peut être utilisée pour désigner des chemins de communication spécifiques. En fait, par la subdivision des canaux (décrite au § D.4.5), il est possible de spécifier formellement le comportement de chaque canal.

Pour chaque direction indiquée ou chaque chemin de communication, la spécification du canal contient une liste de tous les identificateurs de signaux que le canal peut acheminer dans cette direction. Cette liste de signaux offre le moyen de garantir que chaque signal envoyé par un processus à une extrémité du canal puisse être reçu par le processus du bloc qui se trouve à l'autre extrémité du canal. Ainsi, la spécification de canal devient partie intégrante de la spécification d'interface pour chaque bloc. Dans de grands projets intéressant de nombreuses personnes, un accord dès l'origine sur les signaux d'un canal et sur la spécification de ces signaux réduit sensiblement la probabilité que deux processus ne pourront communiquer l'un avec l'autre comme prévu.

La définition d'un canal comporte les éléments suivants:

— nom du canal;

— un ou deux chemins de communication: un chemin de communication spécifie l'origine et la destination d'une liste de signaux. Des identificateurs de bloc ou le mot d'e <<ENV>> (environnement) peuvent être utilisés dans ce contexte;

— une ou deux listes de signaux: une liste des signaux transportés dans cette direction doit être spécifiée pour chaque chemin de communication. Cette liste peut comporter des identificateurs de signaux ainsi que des identificateurs d'autres listes;

— une définition facultative de sous-structure de canal (ou une référence à une telle définition): voir le § D.4.5.

Dans le LDS/PR, une définition de canal est comprise entre les deux mots clés CHANNEL et ENDCHANNEL. Les mots clés FROM et TO servent à désigner les chemins de communication et le mot clé WITH les listes de signaux. On trouvera dans la figure D-3.3.1 un exemple de définition de canal en LDS/PR.

Dans le LDS/GR, une définition de canal est représentée à l'aide d'une ligne reliant les deux parties mises en jeu dans la communication. Le nom du canal doit être plus proche de la ligne que tout autre symbole. Les trajets sont représentés au moyen de flèches et les listes de signaux doivent être placées entre crochets, comme indiqué dans les exemples de la figure D-3.3.2. Pour éviter une confusion entre les canaux et les acheminements de signaux, les flèches ne doivent pas être placées en l'un des deux points terminaux de la ligne (voir le § D.3.5).

Dans un canal bidirectionnel, chacune des deux listes de signaux doit être aussi proche que possible de la flèche correspondante.

Figure D-3.3.1 [T4.100] (à traiter comme tableau MEP, p.

Figure D-3.3.2, p.

D.3.4 *Signaux*

Les signaux peuvent être définis au niveau du système, au niveau du bloc ou dans la partie interne de la définition de processus. Les signaux définis à un certain niveau peuvent être utilisés à ce niveau ou également aux niveaux inférieurs; cependant, pour simplifier chaque niveau, il est suggéré de définir les signaux de manière aussi circonscrite que possible. Les signaux définis dans le cadre d'une définition de processus peuvent être échangés entre des instances de même type de processus (§ D.3.8) ou entre services d'un processus (§ D.5.3).

La définition d'un signal comprend les points suivants:

- nom du signal;
- liste de sortes (facultative): représente la liste des types de valeurs acheminées par ce signal;
- affinage du signal (facultatif): voir le § D.4.7.

En LDS/PR, une définition de signal est spécifiée par le mot clé SIGNAL. Plusieurs signaux (n'ayant pas fait l'objet d'un affinage) peuvent être définis à l'intérieur de la construction, donnant ainsi le nom du signal et la liste de types. Des exemples de définitions de signaux en LDS/PR sont donnés dans la figure D-3.4.1.

Figure D-3.4.1 [T5.100] (à traiter comme tableau MEP, p.

En LDS/GR, on spécifie une définition de signal en insérant des énoncés linéaires dans un symbole de texte comme indiqué dans la figure D-3.4.2.

Figure D-3.4.2, p.

D.3.5 *Acheminements de signaux*

Des acheminements de signaux servent à représenter des chemins de communication similaires aux canaux. Ils peuvent être utilisés au niveau des blocs et à celui des processus. Comme les canaux, les acheminements de signaux peuvent être unidirectionnels ou bidirectionnels mais ils ne peuvent être subdivisés.

Au niveau des blocs, ils représentent un moyen de communication entre les processus d'un bloc ou entre les processus et l'environnement de ce bloc, c'est-à-dire un canal menant à ce bloc ou venant de celui-ci.

Au niveau du processus, les acheminements de signaux peuvent être utilisés lorsque le processus est subdivisé en services (voir le § D.5.3). Dans ce cas, ils relient les services les uns aux autres ou avec les acheminements de signaux du processus.

Si un signal est remis à un acheminement de signaux aboutissant à une frontière de bloc, il passe dans le canal relié à l'acheminement de signaux. Lorsqu'un signal parvient au bloc, à partir d'un canal relié à un ou plusieurs acheminements de signaux, ce signal est placé dans l'acheminement de signaux qui est capable de la transférer.

En LDS/PR, la définition d'un acheminement de signal commence par le mot clé SIGNALROUTE. La syntaxe des chemins de communication et la liste des signaux sont les mêmes que dans le cas des canaux.

En LDS/GR, la seule différence entre un acheminement de signaux et un canal est que, pour les acheminements de signaux, des flèches doivent être placées aux points terminaux des lignes; près de chaque flèche, doit se trouver une liste de signaux appropriés. On trouvera des exemples d'acheminements de signaux dans les figures D-3.6.3 et D-3.6.5 des paragraphes qui suivent.

D.3.6 Diagrammes de système et de bloc

En LDS/GR, une définition de système est représentée au moyen d'un ensemble de diagrammes. La structure d'un système en canaux et en blocs, est représentée à l'aide d'un diagramme de système.

Le diagramme de système se compose des éléments suivants:

- le symbole de cadre: symbole de forme rectangulaire contenant tous les autres symboles. Il représente la frontière du système; l'environnement du système se trouve en dehors de ce cadre;
- l'entête du système: mot clé SYSTEM suivi du nom du système (placé dans l'angle supérieur gauche);
- une numérotation de page facultative (placée dans l'angle supérieur droit du cadre);
- des symboles de texte: un tel symbole peut contenir des énoncés linéaires. Il sert généralement à présenter dans le diagramme des définitions de signaux, des listes de signaux et les données;
- la zone d'interaction de blocs qui comprend la spécification des blocs du système, les canaux et les listes de signaux acheminés par les canaux;
- des diagrammes de macros: les directives concernant l'utilisation des macros sont données au § D.5.1.

Dans le diagramme de système, la spécification d'un bloc peut être l'un des deux points suivants:

- une référence de bloc: symbole de bloc contenant uniquement le nom de bloc;
- un diagramme de bloc: cadre contenant la spécification de la structure de bloc en fonction de ses processus et interactions. Si un bloc est subdivisé en sous-blocs, la spécification de la sous-structure ou la référence à celle-ci doit être placée à l'intérieur du cadre de bloc (§ D.4.3).

On trouvera dans le résumé concernant le LDS/GR la forme des symboles utilisés dans le système et celle des diagrammes de bloc. En ce qui concerne les définitions de symboles, il convient de se référer au § D.7.1.4.

La figure D-3.6.1 donne un exemple d'un diagramme de système pour le système <<s>>. Dans cet exemple, le système s'est divisé en deux blocs B1 et B2 reliés l'un à l'autre et à l'environnement par les canaux C1, C2, C3 et C4. Pour les blocs B1 et B2, on ne donne dans cet exemple qu'une référence. Des explications complémentaires sur les canaux et les symboles de listes de signaux sont données dans les paragraphes qui suivent:

Le même exemple en LDS/PR est représenté dans la figure D-3.6.2.

Figure D-3.6.1, p.

Figure D-3.6.2 [T6.100] (à traiter comme tableau MEP, p.

La structure d'un bloc en ce qui concerne les processus et les acheminements de signaux est représentée en LDS/GR à l'aide d'un diagramme de bloc.

Le diagramme de bloc se compose des éléments suivants:

- le symbole cadre: symbole de forme rectangulaire contenant tous les autres symboles. Il représente les frontières du bloc: au-delà du cadre commence l'environnement du bloc;
- l'entête du bloc: le mot clé BLOCK suivi du nom de bloc (placé dans l'angle supérieur gauche du cadre);
- une numérotation de page facultative (placée dans l'angle supérieur droit du cadre);
- des symboles de texte: ces symboles peuvent englober des énoncés linéaires. Ils sont généralement utilisés pour présenter les définitions de signaux, de listes de signaux et de données;
- la zone d'interaction de processus: cette zone comprend la spécification des processus du bloc et éventuellement des acheminements de signaux et des listes de signaux transportés par les acheminements de signaux. Dans cette zone, il est également possible de représenter les processus qui créent d'autres processus; cette caractéristique est décrite au § D.3.8.1;
- des identificateurs de canaux: si le diagramme fait apparaître des acheminements de signaux aboutissant à l'environnement du bloc ou venant de celui-ci, les identificateurs des canaux reliés à ces acheminements de signaux doivent être spécifiés en dehors du cadre, de façon correspondante aux lignes des acheminements de signaux;
- des diagrammes de macros: on trouvera au § D.5.1 des directives sur l'utilisation des macros.

Dans le diagramme de bloc, la spécification d'un processus peut être:

- soit une référence à un processus: symbole de processus contenant le nom de processus et, en option, la spécification d'instances de processus. Une telle spécification comprend deux couples de nombres entiers séparés par une virgule et placés entre parenthèses (voir le § D.3.8);
- soit un diagramme de processus: cadre contenant un graphique de symboles reliés décrivant le comportement du processus en ce qui concerne les états, les entrées, les sorties, les actions, etc. (voir le § D.3.8). Si le processus est subdivisé en services, le diagramme de processus contient la zone d'interaction de service (§ D.5.3).

Dans la figure D-3.6.3, on trouvera un exemple de diagramme de bloc pour le bloc <<B1>> présentée dans l'exemple de la figure D-3.6.1. Ce bloc B1 est écrit du point de vue des deux processus P1 et P2, reliés par les acheminements de signaux R1, R2, R3, R4 et R5. Pour les processus P1 et P2, on se borne à donner des références dans cet exemple. Les identificateurs de canaux C1, C2 et C3 sont également spécifiés en dehors du cadre.

Le même exemple est présentée en LDS/PR dans la figure D-3.6.4.

Comme cela a déjà été indiqué précédemment, des diagrammes de bloc peuvent être compris dans les diagrammes de système, en lieu et place de références. Dans la figure D-3.6.5 par exemple, les diagrammes des figures D-3.6.1 et D-3.6.3 sont réunis en un seul diagramme de système.

Une recommandation générale concernant l'établissement de diagrammes de ce genre, est qu'ils ne doivent pas être trop complexes afin de rester lisibles et qu'ils doivent tenir sur une seule page.

Figure D-3.6.3, p.

Figure D-3.6.4 [T7.100] (à traiter comme tableau MEP, p.

Figure D-3.6.5, p.

D.3.7 *Commentaires et extension de texte*

D.3.7.1 *Commentaires*

Il est possible d'ajouter des commentaires à une spécification en LDS pour aider le lecteur et préciser certains points. Deux types de commentaires adaptés au LDS/PR et au LDS/GR ont été introduits dans le LDS.

Le premier type, appelé `<<note>>` et utilisé particulièrement en LDS/PR. Il est délimité par `<<*/>>` au début et par `<<*/>>` à la fin.

En LDS/PR, un tel commentaire peut s'insérer partout où se trouve un espace. Il ne doit pas contenir la séquence spéciale `<<*/>>`.

En LDS/GR, un tel commentaire peut se présenter partout où il existe un espace à l'intérieur des instructions linéaires.

On trouvera dans les figures D-3.7.1 et D-3.7.2 certains exemples de cette forme de commentaire, respectivement en LDS/PR et en LDS/GR.

Figure D-3.7.1 [T8.100] (à traiter comme tableau MEP, p.

Figure D-3.7.2, p.

La seconde forme de commentaire permet une mise en correspondance élément par élément entre le LDS/PR et le LDS/GR; elle convient mieux aux applications comportant des traductions automatiques.

En LDS/PR, un tel commentaire se compose du mot d'œuvre COMMENT suivi d'une chaîne de caractères; il peut être inséré comme une instruction (c'est-à-dire suivi de `<<;>>`) partout où une instruction de type peut être insérée. De plus, il peut être inséré avant le symbole `<<;>>` (;) à la fin de toute instruction.

En LDS/GR, cette forme de commentaire est représentée à l'aide d'un symbole de commentaire contenant le texte du commentaire. Le symbole de commentaire est un symbole de forme rectangulaire auquel le côté gauche ou droit fait défaut. Ce symbole doit être étendu aussi bien horizontalement que verticalement, de manière à contenir tout le texte. Il peut être relié à un symbole quelconque en LDS/GR ou à une ligne de liaison. Pour indiquer la connexion il faut employer une ligne en traits discontinus. Si l'association entre le texte du commentaire et le symbole du commentaire ne présente pas d'ambiguïté, le symbole de commentaire peut se présenter simplement sous la forme de crochets.

Dans les figures D-3.7.3 et D-3.7.4, on trouvera certains exemples de cette forme de commentaire, respectivement en LDS/PR et en LDS/GR.

Figure D-3.7.3 [T9.100] (à traiter comme tableau MEP, p.

Figure D-3.7.4, p.

D.3.7.2 *Extension de texte*

Normalement, le texte associé à un symbole graphique devrait être placé à l'intérieur de ce symbole. Cependant, cela n'est pas toujours possible ou pratique. Une autre solution consiste à placer le texte dans un symbole d'extension de

texte rattaché au symbole associé. Le symbole d'extension de texte est similaire au symbole de commentaire; la seule différence est que la ligne le reliant est en trait continu et non en trait discontinu (voir la figure D-3.7.5).

Figure D-3.7.5, p.

Un caractère de soulignement (<< _>> peut être utilisé à la fin d'une ligne de texte comme caractère de continuation. Dans ce cas, les espaces restant sur la même ligne ne sont pas considérées comme faisant partie du texte. On trouvera au § D.3.13 des directives plus détaillées sur la syntaxe des noms.

D.3.8 *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. À 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 du fait du déclenchement d'une action d'arrêt.

Une définition de processus représente la spécification d'un type de processus; plusieurs instances du même type de processus peuvent

être créées et exister simultanément; elles peuvent fonctionner indépendamment et concurremment. Une définition de processus comprend les points suivants (dont certains sont facultatifs):

- nom de processus;
- une paire de nombres entiers. Le premier de ces nombres entiers spécifie le nombre d'instances de processus créées lors de la création du système; s'il est omis, il a une valeur par défaut de 1; le second entier spécifie le nombre maximal d'instances de processus simultanées; s'il est omis, la valeur par défaut maximale est illimitée;
- paramètres formels: liste d'identificateurs de variables associées à leurs types qui est utilisée pour transmettre l'information au processus au moment de la création. Dans la demande de création de processus, une liste de paramètres réels peut être donnée à cet effet. Les valeurs des paramètres formels de processus créés au moment de la création du système sont indéfinies;
- ensemble de signaux d'entrée valides: liste d'identificateurs de signaux définissant les signaux que le processus peut recevoir;
- définitions de signaux: spécification des signaux qui peuvent être échangés entre instances du même processus ou entre services de ce processus (§ D.5.3);
- définitions de procédures: spécification des procédures qui peuvent être appelées par le processus. Une référence de procédure peut être utilisée dans ce contexte (les procédures font l'objet du § D.3.9);
- définition de données: spécifications des nouveaux types, syntypes et générateurs définis par l'utilisateur et localisés dans le processus;
- définition de variables: déclarations des variables du processus. À titre facultatif, on peut indiquer qu'une variable peut être partagée avec plusieurs autres processus du même bloc (variable REVEALED) ou exportée vers d'autres processus ainsi que vers d'autres blocs (variable EXPORTED). Pour chaque variable déclarée, il faut spécifier l'identificateur de sa sorte. Une variable initiale peut facultativement être spécifiée;
- définitions de visibilité: déclarations d'identificateurs de variable qui peuvent servir à l'obtention des valeurs de variables appartenant à d'autres instances de processus. Pour chaque identificateur de variable, la sorte de variable doit

|tre spécifiée;

— définitions d'import: spécification d'identificateurs de variables appartenant à d'autres processus et que le processus veut importer. Pour chaque identificateur, il convient de spécifier la sorte de la variable;

— définition de temporisateur (timer): fait l'objet du § D.3.11;

— définitions de macros: on trouvera des directives sur l'utilisation des macros au § D.5.1;

— corps de processus: spécification du comportement réel du processus exprimé à l'aide d'états, entrées, sorties, t |ches, etc. Si le processus est divisé en sous-parties (services), la définition du processus comporte une section de décomposition en services en lieu et place du corps du processus. On trouvera des directives à ce sujet au § D.5.3.

Des exemples et des explications concernant les données, les variables, les définitions de visibilité et les définitions d'import sont données au § D.3.10.

On trouvera dans la figure D-3.8.1 un exemple partiel de définition de processus en LDS/PR (les mots clés du langage sont écrits en lettres majuscules).

Figure D-3.8.1 [T10.100] (à traiter comme tableau MEP, p.

Le corps de processus représente le graphe réel de la machine à états finis. Il consiste en une séquence d'instructions ordonnées en LDS/PR et, en LDS/GR, c'est une séquence de symboles reliés par des arcs orientés (similaires à un organigramme). La spécification du corps de processus doit toujours commencer par l'indication START suivi d'un ensemble d'actions (transition). L'interprétation d'une instance de processus commence à la création de cette instance de processus.

Les actions qui peuvent |tre exécutées dans une transition sont les suivantes:

— t |che: affectation de variable (ou texte informel);

— exportation: exportation de variable;

— initialisation (set): demande d'activation d'un temporisateur;

— réinitialisation (reset): réinitialisation d'un temporisateur;

— sortie: émission d'un signal en direction d'un autre processus;

— demande de création: création d'une instance d'un type de processus spécifique;

— décision: sélection d'un ensemble d'actions d'après une question;

— appel de procédure: demande d'interprétation d'un ensemble d'actions séparé autonome (m |me usage que dans des langages de programmation);

— branchement (join): spécification d'un <<saut>> vers un autre ensemble d'actions.

Une transition peut se terminer par l'une des actions suivantes:

— nextstate (état suivant): spécification de l'état dans lequel se trouvera l'instance de processus;

— arr |t: arr |t immédiat de l'instance de processus.

Après la spécification de l'action de départ et de la transition de départ facultative, le corps du processus comprend les définitions de tous ses états possibles. Chaque définitif d'état commence par la spécification des stimuli possibles, attendus par le processus dans cet état. Les stimuli possibles sont les suivants:

— entrées: signaux qui peuvent |tre reçus;

— mises en réserve: signaux qui doivent |tre mis en réserve pour traitement ultérieur;

- conditions de validation: d'écrites au § D.3.8.5;
- signaux continus: d'écrites au § D.3.8.5.

Une transition doit être spécifiée en correspondance avec chaque stimulus sauf en ce qui concerne les mises en réserve. De telles transitions représentent la séquence d'actions que le processus exécutera si le stimulus apparaît.

Si un processus exécute une action d'arrêt et que des signaux émis mais non encore reçus se trouvent en suspens, ces signaux sont mis au rebut.

En LDS/GR, une définition de processus est représentée à l'aide du diagramme de processus. Un diagramme de processus se compose des éléments suivants:

- un symbole de cadre: symbole de forme rectangulaire contenant tous les autres symboles. Si aucun acheminement du signal n'est rattaché au symbole de cadre, celui-ci peut être omis;
- l'en-tête du processus: le mot d'entête PROCESSUS suivi de l'identificateur de processus, puis éventuellement de la spécification des paramètres formels. L'en-tête de processus est placé dans l'angle supérieur gauche du cadre;
- une numérotation de page facultative (placée dans l'angle supérieur droit);
- des symboles de texte: dans le cas d'un diagramme de processus, un symbole de texte peut être utilisé pour contenir des définitions de signal, de variable, de visibilité, d'importation, de données et de temporisateur (timer);
- références de procédure: symbole de procédure contenant un nom de procédure représentant une procédure du processus qui est définie séparément;
- diagrammes de procédure: un pour chaque procédure locale du processus qui n'est pas référencée;
- la zone de graphe de processus: spécification du comportement du processus en termes de départ, d'états, d'entrées, de sorties, de tâches, de ports et d'arcs orientés. Si le processus est structuré en services, la zone du graphe de processus contient la spécification des services ou leurs références (voir le § D.5.3). Les symboles en GR utilisés pour le corps du processus sont indiqués dans le résumé du LDS/GR;
- diagrammes de macro: on trouvera au § D.5.1 des directives sur l'utilisation des macros.

La figure D-3.8.2 donne un exemple de définition de processus en LDS/GR; on trouvera des explications et des exemples complémentaires sur les graphes de processus dans les paragraphes qui suivent.

Figure D-3.8.2, p.

Si un graphe de processus ne tient pas sur une seule page, le diagramme peut être présenté sur plusieurs pages; il faut noter:

- qu'une numérotation de page contenant le numéro de la page et le nombre total de pages devrait être fournis, c'est-à-dire: 1 (9);
- que le symbole de brachement (join) ou le symbole d'état suivant (nextstate) peut être utilisé pour représenter des connexions entre différentes parties du graphe du processus.

Un bon critère pour diviser un graphe de processus en plusieurs parties consiste à représenter une définition d'état sur chaque page. Si une définition d'état ne tient pas sur une seule page, il est possible d'utiliser le symbole de brachement pour représenter des connexions avec une partie du graphe qui se trouve sur une autre page.

D.3.8.1 *Création de processus*

Comme indiqué au paragraphe précédent, des processus (c'est-à-dire des instances de processus) peuvent être créés à la suite d'une demande explicite ou lors de la création du système.

La demande explicite de création ne peut être formulée que par un autre processus du même bloc que le processus à créer; elle permet la spécification de paramètres réels pour le transfert de l'information vers la nouvelle instance créée. La figure D-3.8.3 donne un exemple de création en PR et en GR.

Figure D-3.8.3, p.

Si une ou plusieurs instances de processus d'un type de processus donné sont créées au moment de la création du système, la définition de ce (ou ces) processus ne doit pas avoir accès aux paramètres formels car ils seront indéfinis. En conséquence, chaque définition de processus ayant des paramètres formels doit également faire le nécessaire pour interdire l'accès à ses paramètres formels avant que ceux-ci aient reçu une valeur ou qu'ils comportent explicitement la spécification d'aucune instance de processus au moment de la création du système (voir la figure D-3.8.4).

Figure D-3.8.1 [T11.100] (à traiter comme tableau MEP, p.

Lors de la création d'un processus, il est possible de déterminer des valeurs d'instance de processus à l'aide des expressions prédéfinies suivantes:

(OFFSPRING) DESCENDANT: renvoie la valeur PID de la dernière instance créée,

SELF: renvoie la valeur PID de l'instance de processus proprement dite,

(PARENT) ASCENDANT: renvoie la valeur PID de l'instance qui procède à la création

(SENDER) EMETTEUR: renvoie la valeur PID du processus émettant le dernier signal utilisé.

Lorsqu'un processus est créé par suite de la création du système, seule l'expression SELF renvoie une valeur PID; les expressions OFFSPRING et PARENT donnent la valeur NULL.

De telles valeurs revêtent une grande importance lorsqu'il existe plusieurs instances de processus du même type de processus car elles constituent le seul moyen d'adresser sans ambiguïté les signaux aux instances. En fait, comme cela est mieux expliqué au § D.3.8.6, lorsqu'un processus émet un signal, il doit spécifier l'instance de destination, à moins que celle-ci ne puisse être déterminée sans ambiguïté.

L'utilisateur doit veiller à ce que les instances de processus créées puissent, au besoin, communiquer les unes avec les autres. Pour y parvenir, il faut souvent prévoir certains types de processus d'initialisation. Ces processus, créés en même temps que le système, devront créer les autres processus et éventuellement communiquer les valeurs PID appropriées à tous les processus auxquels elles seront nécessaires.

Il convient de noter d'autres points importants concernant la création de processus:

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;

2) après leur création, les processus ont une durée de vie qui leur est propre. Ils ne cessent d'exister que lorsqu'ils accomplissent une action d'arrêt pendant une transition. 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 d'arrêt.

La relation entre les processus créateurs et les processus créés peut être représentée dans un diagramme de bloc à l'aide de symboles de lignes de création, comme indiqué dans la figure D-3.8.5.

Figure D-3.8.5, p.

D.3.8.2 *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.

En LDS/PR, un état est représenté par le mot d'état STATE suivi du nom de l'état. La spécification de l'état se termine soit à l'énoncé d'état suivant soit à la fin du processus ou au moyen du mot d'état explicite ENDSTATE (FIN D'ÉTAT). On peut utiliser un astérisque dans l'énoncé d'état au lieu du nom de l'état; il s'agit là d'une notation abrégée indiquant que les entrées pour les mises en réserve suivantes et les transitions correspondantes doivent être interprétées pour chaque état.

Le mot d'état NEXTSTATE (ÉTAT SUIVANT), suivi du nom de l'état, est utilisé pour indiquer l'état qui suit. On peut employer un tiret dans l'énoncé d'état suivant au lieu du nom de l'état pour indiquer que l'état suivant est précisément l'état dont la transition actuelle tire son origine.

La figure D-3.8.6 donne un exemple partiel en LDS/PR.

Figure D-3.8.6 [T12.100] (à traiter comme tableau MEP, p.

En LDS/GR, un état est représenté à l'aide d'un symbole d'état contenant le nom d'état et est relié aux symboles d'entrée ou de mises en réserve. Le même symbole d'état, avec une flèche d'arrivée, sert à représenter l'énoncé d'état suivant.

Par commodité, pour simplifier le dessin ou faciliter la compréhension, le même état peut apparaître plusieurs fois dans un diagramme en LDS. Si tel est le cas, le diagramme est considéré comme entièrement équivalent au diagramme qui résulterait, de la fusion de toutes les apparitions multiples du même état. Les figures D-3.8.7 et D-3.8.8 donnent des exemples de cette situation. Dans la figure D-3.8.7 b), on utilise un symbole d'état comme lien avec l'état principal portant le même nom, lorsqu'il est mentionné dans le symbole d'état suivant. Dans la figure D-3.8.8 un état est représenté par des symboles multiples n'ayant chacun qu'un sous-ensemble des entrées (ou des mises en réserve).

Les diagrammes a) et b) de la figure D-3.8.7 sont logiquement équivalents. Le diagramme a) contient une seule apparition de chaque état tandis que le diagramme b) utilise des apparitions multiples. Dans le diagramme b), l'état a une apparition principale, dans laquelle tous ses symboles d'entrées (et de mises en réserve) associées sont indiquées. Lorsque cet état peut être atteint à partir d'autres points du diagramme (par exemple le point de terminaison d'une transition), il est indiqué comme un état sans entrée ni mise en réserve associée. Un commentaire du symbole d'état suivant se référant au symbole d'état améliorera la clarté, notamment lorsqu'ils apparaissent sur des pages différentes.

La figure D-3.8.8 utilise les apparitions multiples d'un état pour constituer l'ensemble total d'entrées (et de mises en réserve). Chaque apparition de l'état est indiquée uniquement avec un sous-ensemble de ces entrées.

Figure D-3.8.7 a, p.

Figure D-3.8.7 b, p.

Figure D-3.8.8, p.

Cette approche a été appliquée avec succès lorsque des états avaient un nombre relativement grand d'entrées ou de mises en réserve mais présentaient le risque que le lecteur interprète de façon erronée le diagramme s'il n'était pas conscient de l'existence d'occurrences multiples. Pour éviter ce malentendu, les états n'indiquant qu'un sous-ensemble d'entrées/mises en réserve devraient être accompagnés d'un commentaire indiquant une référence à d'autres états avec leurs entrées et mises en réserve associées, comme indiqué dans la figure D-3.8.8.

Des apparitions multiples peuvent être utilisées pour attirer l'attention du lecteur sur certains aspects (par exemple la séquence normale des signaux traités), laissant d'autres aspects pour d'autres pages (par exemple le traitement de situations d'alarme).

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-3.8.9, 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-3.8.10.

Figure D-3.8.9, p.

Figure D-3.8.10, p.

