INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports Techniques

N° 113

Programme 5
Automatique, Productique,
Traitement du Signal et des Données

SYNDEX
UN ENVIRONNEMENT DE PROGRAMMATION POUR MULTI-PROCESSEUR DE TRAITEMENT DU SIGNAL

MANUEL DE L'UTILISATEUR
VERSION V. 0

Christophe LAVARENNE
Yves SOREL

Novembre 1989

RT_0113

SYNDEX

Un environnement de programmation pour multi-processeur de traitement du signal

MANUEL DE L'UTILISATEUR
version v.0

Christophe LAVARENNE - Yves SOREL

sorel@seti.inria.fr

INRIA Domaine de Voluceau Rocquencourt B.P.105 - 78153 Le Chesnay Cedex

Abstract

SYNDEX (acronym for SYNchronous Distributed EXecutive) is an interactive graphical development environment for signal processing applications on multi-processor machines. It allows the high-level description of a target machine and its application program, then generates the system program to be added to the application program, either for the target machine to be simulated on a simulation machine, or for the application program to be executed on the target machine in real time. The application program may be placed on the target machine, either manually, or with the help of optimization routines. In the present version, SYNDEX generates OCCAM source code for a multi-TRANSPUTER machine.

Résumé

SYNDEX (acronyme pour EXécutif Distribué SYNchrone) est un environnement graphique interactif de développement d'applications de traitement du signal pour machines multi-processeur. Il permet une description de haut niveau d'une machine cible et de son programme d'application, puis génère automatiquement le programme système qu'il faut ajouter au programme d'application, soit pour effectuer une simulation de la machine cible sur une machine de simulation, soit pour exécuter le programme d'application sur la machine cible en temps réel. Le placement du programme d'application sur la machine cible peut être fait soit manuellement, soit assisté par des programmes d'optimisation. Dans cette version, SYNDEX génère du source OCCAM pour une machine multi-TRANSPUTER.

* Etude financée dans le cadre de la convention avec le CNET
n° 2.88.A009.000.00.MC.01.1

I. GUIDE DE REFERENCE

I.1. INTRODUCTION

SYNDEX (acronyme pour EXécutif Distribué SYNchrone [SIM]) est un environnement graphique interactif de développement d'applications de traitement du signal pour machines multi-processeur. Il permet une description de haut niveau d'une machine cible et de son programme d'application, puis génère automatiquement le programme système qu'il faut ajouter au programme d'application, soit pour effectuer une simulation de la machine cible sur une machine de simulation, soit pour exécuter le programme d'application sur la machine cible en temps réel. Le placement du programme d'application sur la machine cible peut être fait soit manuellement, soit assisté par des programmes d'optimisation [GHE].

Dans cette version, SYNDEX génère du source OCCAM [OCC] pour une machine de simulation mono-TRANSPUTER ou pour une machine cible multi-TRANSPUTER [TRA]. SYNDEX doit être associé a un système de développement OCCAM [TDS] pour édition et compilation séparée de librairies (librairies système supportant les différents protocoles de communication inter-processeurs, et librairie des activités de calcul de chaque application) et pour compilation et exécution des programmes générés par SYNDEX.

I.2. CYCLE DE DEVELOPPEMENT

L'édition d'une machine et de son programme d'application se fait en trois étapes :

- construction du graphe logiciel, dont les nœuds sont des activités de calcul, et les arcs orientés des canaux de communications qui servent aux transferts de données entre activités

- construction du graphe matériel, dont les nœuds sont des processeurs et des bus et les arcs non orientés des liens de communication bidirectionelle, et construction de chemins dans ce graphe, appelés 'routes structurelles', permettant de connecter logiquement des processeurs deux a deux, en passant éventuellement par des processeurs intermédiaires

- placement des activités de calcul sur les processeurs, et des canaux de communication sur les routes structurelles.

Le source OCCAM peut ensuite être généré, puis compilé et exécuté sous TDS.

I.2.1. Construction du graphe logiciel

Le graphe logiciel peut être construit manuellement ou pourra être ultérieurement généré par le compilateur du langage SIGNAL [SIG], 'langage synchrone' de haut niveau adapté à la description d'applications de traitement du signal.

Les activités de calcul font appel à des PROC OCCAM compilées séparément, dont il suffit de connaître l'interface d'appel. SYNDEX génère pour chaque activité de calcul une enveloppe gérant les canaux (au sens OCCAM) de communication inter-activités et leurs tampons associés (passés en arguments de la PROC).

Le protocole de transfert de données inter-activités associé a chaque canal OCCAM est défini par un PROTOCOL OCCAM (ces protocoles, situés au niveau applicatif, sont à distinguer des protocoles de communication inter-processeurs, situés au niveau système). Tout nouveau PROTOCOL est créé à partir des huit PROTOCOLs OCCAM primitifs (BOOL BYTE INT INT16 INT32 INT64 REAL32 REAL64).

Tout nouveau nœud du graphe logiciel est créé en déclarant son nom, celui de sa PROC OCCAM de calcul et les canaux d'entrée-sortie de l'enveloppe associée.

Les nœuds logiciels doivent être connectés de manière cohérente : une sortie ne peut etre connectée qu'a une seule entrée de même protocole et appartenant à un nœud logiciel autre que celui de la sortie.

I.2.2. Construction du graphe matériel

Un nœud processeur représente une entité matérielle au niveau fonctionnel : un processeur exécute des activités de calcul (programme application) et des activités de communication inter-processeur (programme système).

Le temps nécessaire à l'exécution d'une activité de calcul est donné lors de la déclaration du nœud logiciel appelant cette activité de calcul ; le temps nécessaire à une communication inter-processeur est (dans la version actuelle) uniquement proportionnel au nombre d'octets transmis (40 pour un protocole [10]REAL32, 1 pour un BYTE).

Tout nouveau processeur est créé en déclarant son nom et ses liens de communication. Il existe deux types de lien : procr-procr (processeur-processeur ou point-à-point) et procr-bus (processeur-bus ou multipoint).

Un nœud bus représente un moyen de communication partagé entre plusieurs processeurs.

Les nœuds matériels doivent être connectés de manière cohérente :
- un lien procr-procr ne peut être connecté qu'à un lien procr-procr
- deux processeurs peuvent être connectés par plusieurs liens procr-procr.
- un lien procr-bus ne peut être connecté qu'à un bus
- un processeur ne peut être connecté à un bus que par un seul lien procr-bus

Une route structurelle est déclarée en donnant un processeur de départ, le lien par lequel on en sort, le processeur suivant et le lien par lequel on en sort, etc..., enfin un processeur d'arrivée (cette définition est suffisante pour determiner tous les liens et bus intermediaires utilisés) ; une route peut être parcourue dans les deux sens.

I.2.3. Placement des activités de calcul et des canaux de communication

Le placement des activités de calcul sur les processeurs peut avoir lieu dès que ces premières et ces derniers sont déclarés. Le placement sur des processeurs différents de deux activités de calculs interconnectées, donne lieu à une communication inter-processeur. Au moins une route structurelle doit avoir été créée entre ces deux processeurs pour que cette communication puisse y être placée.

Le placement peut être fait soit manuellement, soit automatiquement à l'aide d'heuristiques d'optimisation [GHE]. Dans sa version actuelle, SYNDEX propose deux étapes d'optimisation :

Etape 1: optimisation du placement des activités de calcul sur les processeurs

L'heuristique d'optimisation est un algorithme glouton tentant de minimiser deux critères hiérarchisés :
- le temps de traversée du graphe logiciel
  (Tgfd, acronyme pour "Temps graphe flot de données")
- une estimation du volume des communications inter-processeur
  (sans tenir compte du routage)

L'algorithme est proposé en deux versions, chacune tentant de minimiser l'un des deux critères en priorité.

Etape 2 : optimisation du placement des canaux de communication sur les routes

L'heuristique d'optimisation est un algorithme glouton tentant de minimiser une estimation du volume des communications inter-processeur (prenant en compte le routage), tout en construisant les connexions physiques et les routes manquantes (construction d'un arbre de poids minimal), ou en profitant des connexions physiques et des routes existantes. L'algorithme existe en deux versions, plaçant les canaux de communication par ordre respectivement croissant ou décroissant du volume des données transmises.

I.2.4. Génération de code source OCCAM

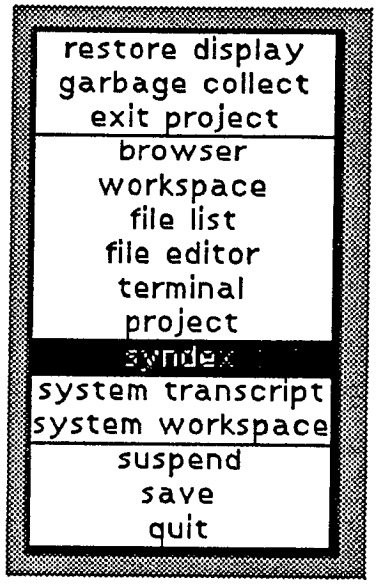
Le source OCCAM ne peut être correctement généré que lorsque :
- le graphe logiciel n'a plus d'entrée/sortie non connectée
- toutes les activités de calcul ont été placées sur des processeurs
- toutes les communications inter-processeur ont été placées sur des routes

Le source OCCAM généré doit être importé sous un environnement de développement OCCAM [TDS] pour compilation et exécution sur transputer. Les librairies système et application doivent être accessibles dans l'environnement de compilation par les directives OCCAM :

```
-- constantes globales (définissant le format des messages, etc...)
#USE CONST

-- code pour PESl et PESb (Portes d'Entrée/Sortie de type PPL et PBL)
#USE PES

-- code pour BL (Bus Logiciel des processeurs)
#USE BL

-- code pour PExxx et PRxxx (pour chaque protocol nommé xxx)
#USE xxx

-- librairie application (code pour les PROC des activités de calcul)
#USE yyy.act -- yyy est habituellement le nom de la machine
```

I.3. INTERFACE UTILISATEUR

A partir d'un environnement SYNDEX (cf §III.1.Installation), on ouvre un nouveau browser SYNDEX en choisissant l'option **syndex** du menu système :



La fenêtre du browser SYNDEX comprend 3 vues :

- **vue zoom** :
en haut à gauche, une vue graphique affichant une représentation simplifiée de la totalité de la machine, et contrôlant la portion affichée dans la vue du bas

- **vue texte** :
en haut à droite, une vue textuelle pour les messages d'aide et l'interprétation des commandes SYNDEX textuelles

- **vue édition** :
en bas, une vue graphique permettant l'édition de la représentation graphique détaillée de la machine (limitée à la portion zoomée)

Le titre du browser est **Syndex Browser**, signifiant qu'aucune machine n'est en cours d'édition. Il présente dans sa vue texte un message de première aide. Une aide contextuelle peut également être obtenue pour chaque option des menus des vues en combinaison avec la touche **Control**.

Les trois vues proposent le même menu lorsqu'on appuie sur le bouton central de la souris :

- l'option **new** permet de créer une nouvelle machine prête à l'édition
- l'option **edit** permet d'éditer une machine existante
(ces deux options sont décrites en détail au paragraphe §I.3.2. Vue zoom).

Une machine existante peut se trouver en mémoire (représentation interne) ou sur fichier (représentation textuelle SYNDEX); pour être éditée, une machine doit se trouver en mémoire, aussi une machine sur fichier est-elle automatiquement chargée en mémoire avant édition (en convertissant sa représentation textuelle SYNDEX en représentation interne).

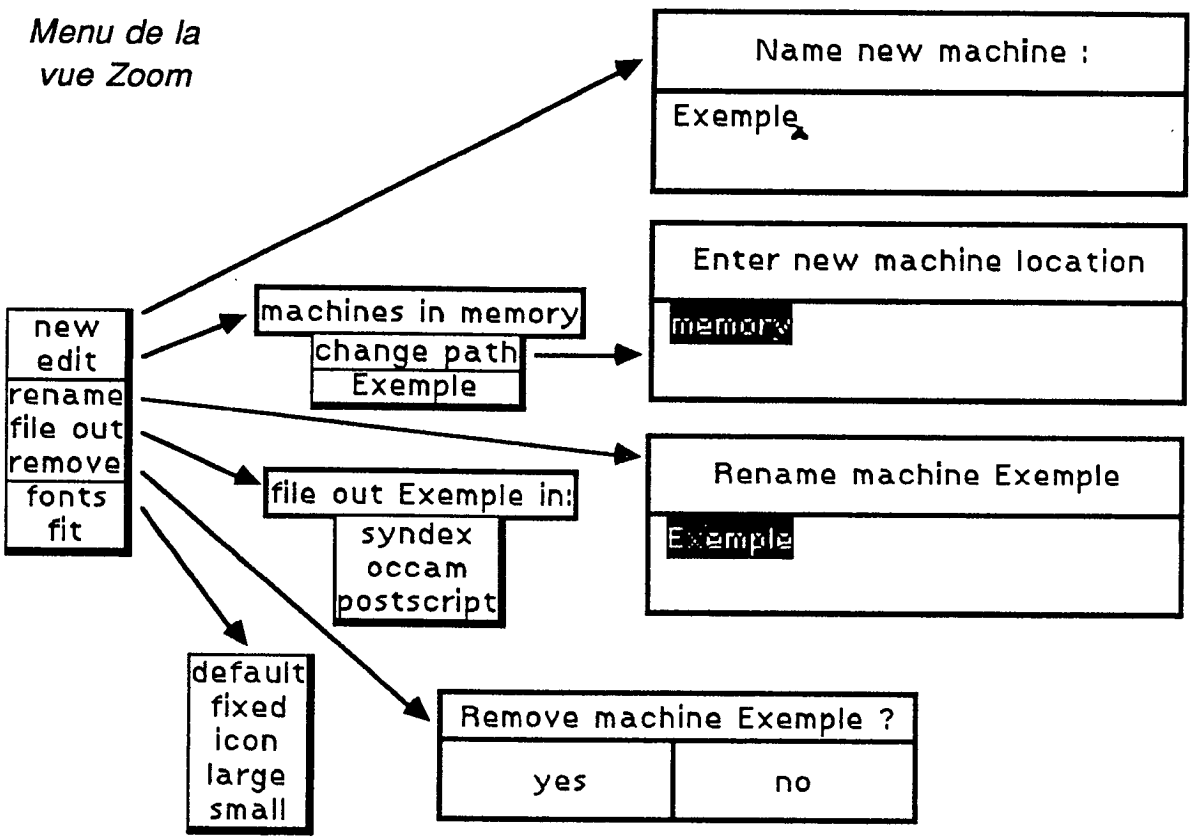
Les trois vues proposent des menus différents dès qu'une machine est éditée. Le label du browser devient alors **Syndex Browser on nom.de.la.machine.**

Le nom de la machine peut être changé par l'option **rename** du menu de la vue zoom (cf §I.3.2). Il ne peut être modifié par l'option **new label** du menu système du browser (bouton droit de la souris).

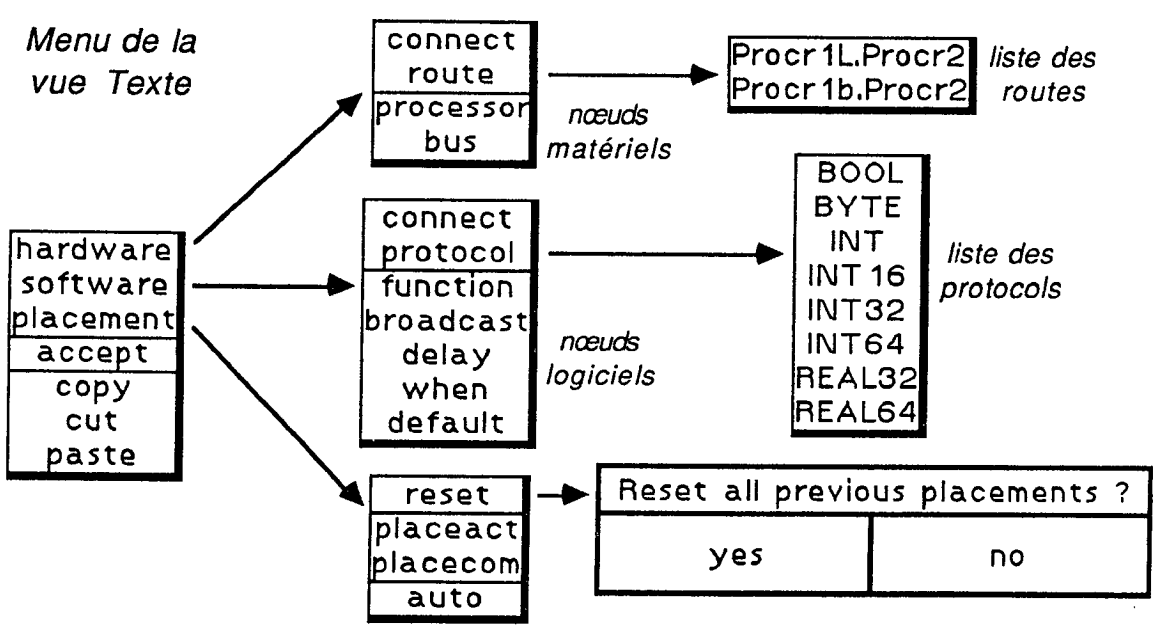
I.3.1. Hiérarchie des menus de l'interface

Tous les menus présentés dans ce paragraphe sont détaillés dans les paragraphes suivants.

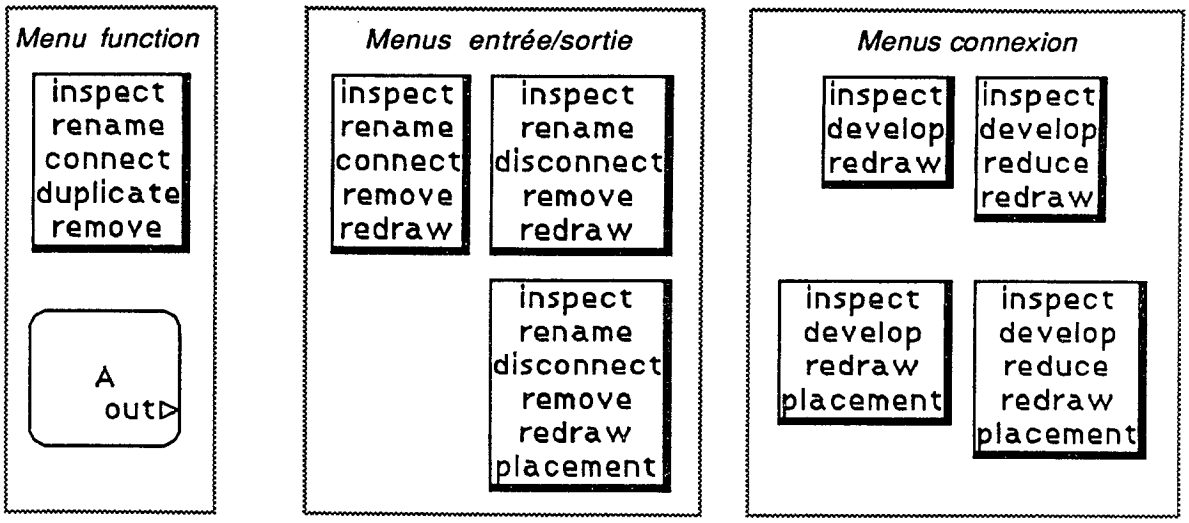
Menus de la vue zoom :



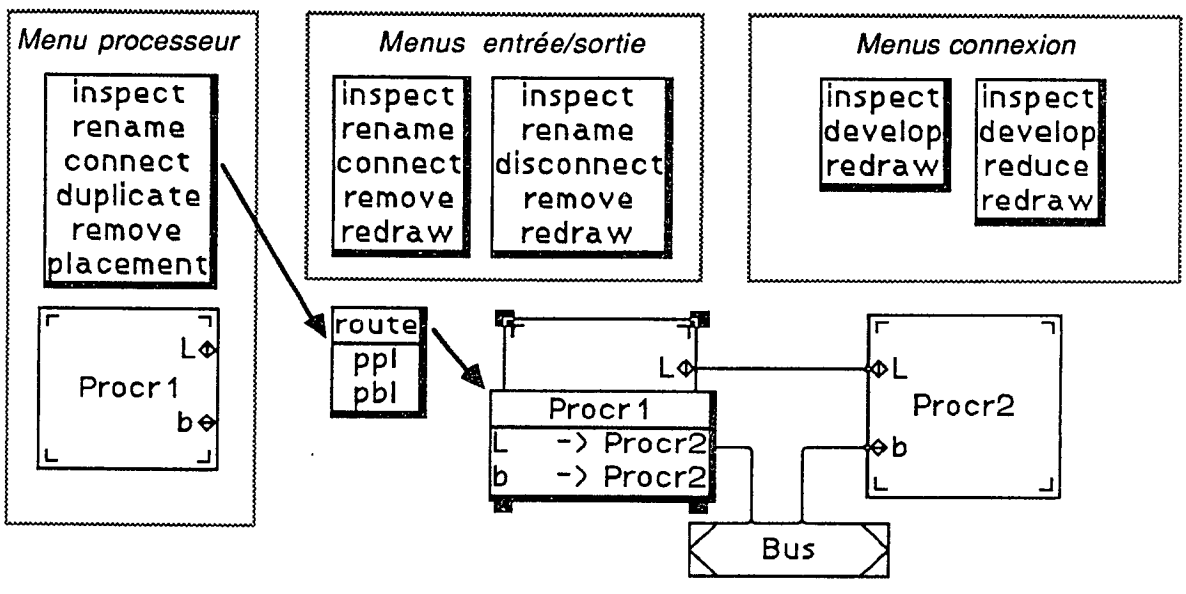
Menus de la vue texte :



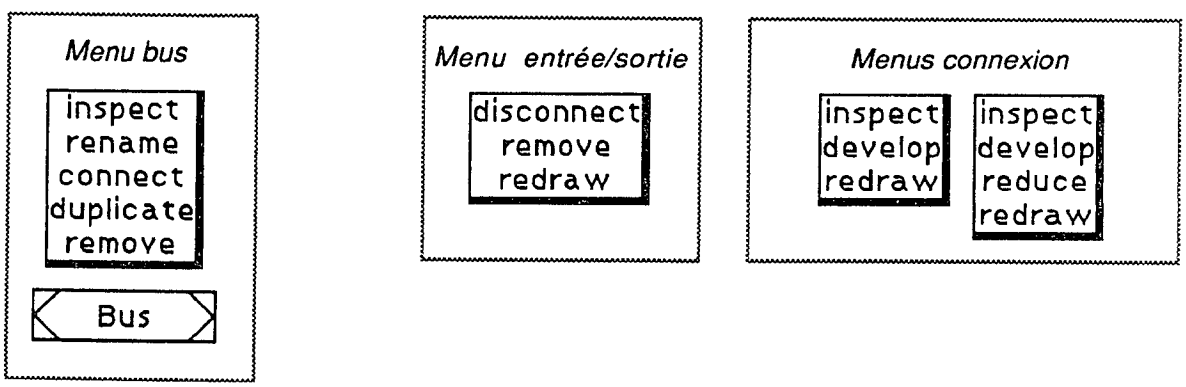
Menus des nœuds, entrée/sorties et connexions logiciels :



Menus des processeurs, de leurs entrée/sorties et connexions procr-procr :

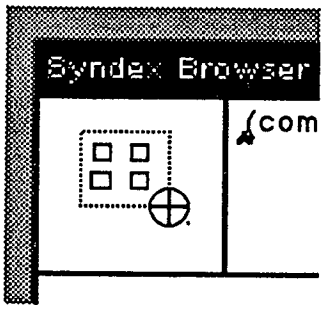


Menus des bus, de leurs entrée/sorties et connexions procr-bus :



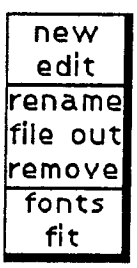
I.3.2. Vue zoom

Dans la vue zoom, seuls les nœuds sont représentés, par de simples boîtes. Un rectangle en pointillé représente le champ visualisé dans la vue édition :

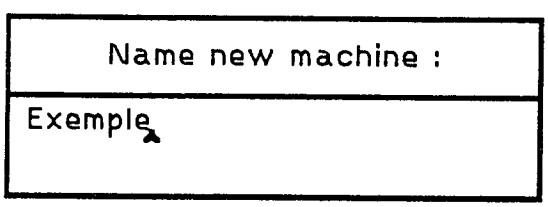


Pour le changer, viser d'abord le centre du nouveau champ en y amenant le curseur viseur, puis appuyer sur le bouton gauche de la souris : le rectangle pointillé se déplace de facon à aligner son centre sur le point visé, puis le viseur saute sur le coin bas droit du rectangle.
Pour changer la dimension du champ, déplacer la souris tout en appuyant toujours sur son bouton gauche. Lorsqu'on lâche le bouton de la souris, la taille du rectangle se fige et la vue édition est mise à jour de façon à visualiser le nouveau champ.

En appuyant sur le bouton central de la souris, on obtient un menu relatif à la machine dans sa totalité :

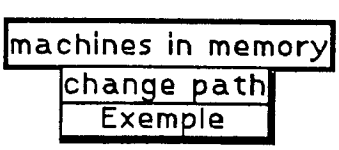


- l'option **new** présente une fenêtre de saisie du nom d'une nouvelle machine :

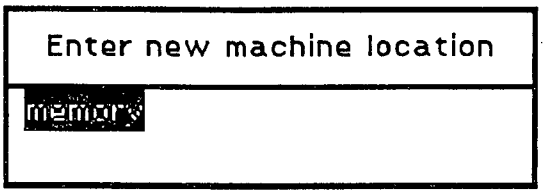


Si ce nom est déjà utilisé pour nommer une autre machine chargée en mémoire, la fenêtre réapparaît avec un nom inutilisé, composé du nom saisi et d'une extension numérique.

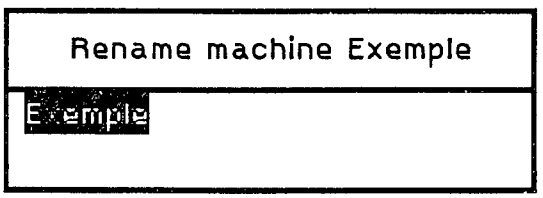
- l'option **edit** présente la liste des machines existantes en mémoire :



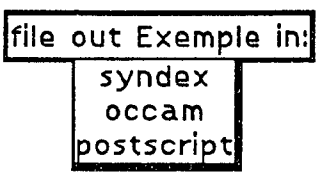
Pour obtenir la liste des machines existantes sur fichier dans un répertoire donné, choisir l'option **change path** qui présente une fenêtre de saisie du nom du répertoire (une saisie vide sélectionne le répertoire courant, le nom **memory** sélectionne la liste des machines en mémoire) ; la machine sélectionnée est affichée pour édition.



- l'option **rename** présente une fenêtre de saisie du nouveau nom de la machine; si ce nom est déjà utilisé, même comportement que pour l'option **new**.



- l'option **file out** présente un second menu à trois options :

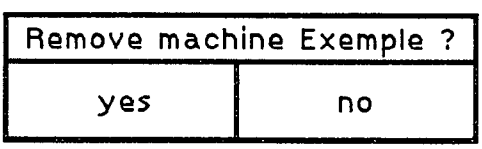


. l'option **syndex** sauve la machine sous forme d'un source textuel SYNDEX dans un fichier nommé comme la machine avec une extension '.syndex'

. l'option **occam** génère le programme OCCAM dans un fichier nommé comme la machine avec une extension '.occam'

. l'option **postscript** génère un fichier source postscript nommé comme la machine avec une extension '.ps'; en combinaison avec la touche **Shift**, le fichier généré est nommé 'hardCopy.ps' et automatiquement envoyé à l'imprimante

- l'option **remove** détruit en mémoire la machine éditée après confirmation :



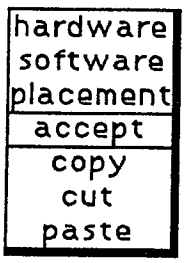
- l'option **fonts** permet de choisir pour la vue édition un nouvelle fonte de caractères, choisie dans un menu des fontes existant sur la station de travail.

- l'option **fit** dilate ou contracte la représentation graphique de la machine, de façon à ce que le rectangle englobant toutes les boîtes de la machine (vues dans la vue zoom) devienne égal au rectangle pointillé affiché dans la vue zoom. De cette manière, il est possible de recadrer ou d'agrandir l'espace de travail.

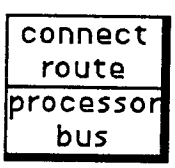
I.3.3. Vue texte

En plus des messages d'aide, cette vue permet d'afficher, éditer et interpréter du source textuel SYNDEX, afin de créer de nouveaux objets ou de modifier des objets existants.

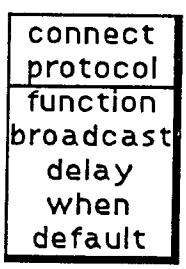
En appuyant sur le bouton central de la souris, on obtient le menu :



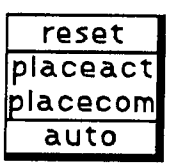
- les options **hardware** **software** et **placement** présentent chacune un menu d'instructions SYNDEX, concernant le graphe matériel pour l'option **hardware**, le graphe logiciel pour l'option **software**, et le placement du logiciel sur le matériel pour l'option **placement** :



menu hardware



menu software

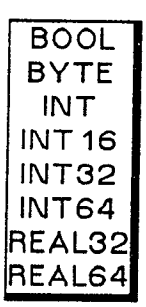


menu placement

En sélectionnant une instruction des menus, on obtient un texte modèle de cette instruction, dont les parties à remplir sont en majuscules et suggèrent les valeurs à saisir, le tout commenté par un mode d'emploi.

Les modèles pour les instructions **when** et **default** du langage SIGNAL [SIG], permettant de créer respectivement des activités de sous-échantillonage conditionné et de mélange de signaux, seront implantées dans une future version de SYNDEX.

En sélectionnant une instruction du menu **hardware** ou **software**, cette fois en enfonçant la touche **Shift**, on obtient la liste des noms des objets existants du type associé à l'instruction. En sélectionnant un nom, on obtient l'instruction qui a permis de créer l'objet correspondant, commentée en général par des références croisées. Le même résultat est obtenu en sélectionnant l'option **inspect** des menus de la vue édition (cf §I.3.4.2), sauf pour les protocols et les routes qui n'ont pas de représentation graphique. La liste des protocols primitifs OCCAM est :

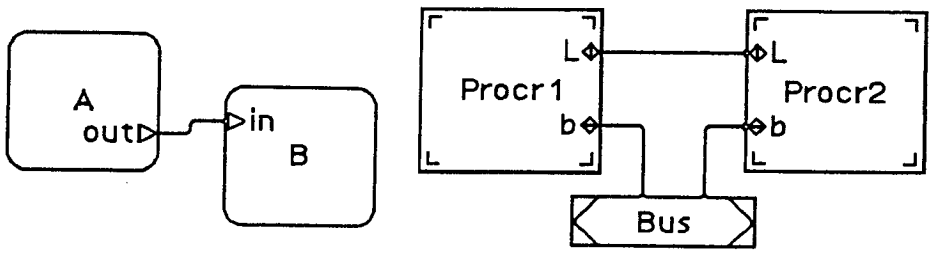


- l'option **accept** lance l'interprétation du source SYNDEX édité.
S'il n'existe aucun objet de même nom que celui saisi dans l'instruction, alors l'interprétation de l'instruction crée un nouvel objet.
Sinon, il faut que l'objet existant soit du type de l'instruction, auquel cas l'objet est modifié.

- les options **copy cut paste** ont le comportement habituel à l'édition de texte en Smalltalk [SMA]

I.3.4. Vue édition

Dans cette vue, chaque nœud est représenté par une boîte avec son nom au centre.
- les nœuds logiciels ont leur coins arrondis
- les processeurs ont leurs coins doublés
- les bus ont leurs coins angulés



Si une boîte est trop petite, ses entrées/sorties ne sont pas visualisées. Sinon elles sont représentées par leur nom et par de petites icônes sur le périmètre des boîtes :

- les icônes des entrées (resp. sorties) des nœuds logiciels sont des triangles pointés vers l'intérieur (resp. l'extérieur) de la boîte

- les icônes des liens procr-procr (resp. procr-bus) sont des losanges pointés sur le périmètre de la boîte, barrés perpendiculairement (resp. parallèlement) au bord de la boîte.

Les connexions entre nœuds sont représentées par des lignes brisées, attachées aux boîtes au niveau des entrées/sorties.

Lorsque le curseur se trouve dans cette vue, l'objet le plus proche du curseur est toujours sélectionné par défaut, et mis en évidence par une (des) marque(s) carrée(s), associée(s) à des points caractéristiques. Pour une boîte, il s'agit des coins, du centre, et des entrées/sorties situées sur le périmètre de la boîte. Pour une connexion, il s'agit des sommets de la ligne brisée.

L'objet sélectionné par défaut peut être modifié :

- soit au niveau de son aspect graphique (déplacement ou redimensionnement) au moyen du bouton gauche de la souris

- soit au niveau de son contenu au moyen du bouton central de la souris qui présente un menu des fonctions qui lui sont applicables.

Certaines de ces fonctions nécessitent des informations complémentaires. S'il suffit de désigner un seul autre objet, seuls ceux compatibles avec la fonction seront sélectionnables, le plus proche de la souris clignotera et sera sélectionné en cliquant la souris. S'il faut désigner une collection d'objets, celle-ci doit avoir été construite auparavant en créant une **sélection multiple** (cf § I.3.4.1).

I.3.4.1. Manipulations graphiques

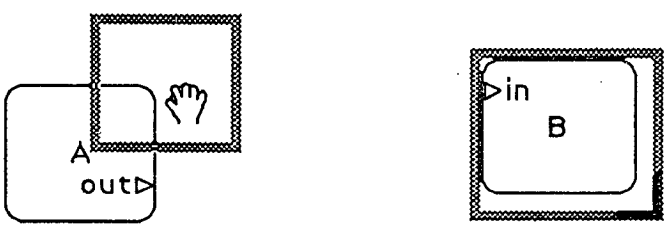
On peut effectuer une **sélection multiple** regroupant plusieurs boîtes en les cliquant l'une après l'autre avec le bouton gauche de la souris tout en maintenant la touche **Shift** appuyée. Les boîtes de la sélection multiple apparaissent en inversion vidéo. Si une boîte fait déjà partie de la sélection multiple, elle en est retirée.
La sélection multiple peut être utilisée par certaines fonctions, ou annulée en cliquant n'importe où le bouton gauche de la souris sans enfoncer la touche **Shift**.

En appuyant (sans **Shift**) sur le bouton gauche de la souris, le curseur saute sur le point marqué (qui est le point caractéristique le plus proche du curseur), puis le déplace en suivant les mouvements de la souris.
Dès que le bouton est relâché, l'objet associé au point déplacé est retracé.

Dans le cas d'une connexion :
- tout déplacement amenant la connexion à traverser une boîte est refusé

Dans le cas d'une boîte :
- si une entrée/sortie est marquée, on peut la déplacer sur le périmètre de sa boîte, de plus si elle est connectée, la connexion associée sera retracée dès que le bouton gauche de la souris sera relâché
- si un seul coin est marqué, on peut changer la dimension de la boîte
- si les quatre coins sont marqués (le curseur est plus près du centre de la boîte), on peut déplacer la boîte

Dans les deux derniers cas, un halo matérialise la marge à respecter autour de la boîte et ne doit rentrer en collision avec aucune autre boîte, sous peine d'annuler le déplacement.



Les connexions liées à la boîte, ainsi que celles qui se trouvent sous le nouvel emplacement de la boîte, sont automatiquement retracées de façon à ce qu'aucune connexion ne traverse de boîte.

I.3.4.2. Description des menus

En appuyant sur le bouton central de la souris, on obtient un menu associé à l'objet marqué et dépendant de son contexte.

Certaines options sont définies seules ou en combinaison avec la touche **Shift**.

Certaines options apparaissent dans presque tous les menus :

- l'option **inspect** affiche dans la vue texte une définition de l'objet marqué : l'instruction SYNDEX qui a permis de créer cet objet, ainsi que quelques commentaires, en général des références croisées

- l'option **rename** présente une fenêtre de saisie d'un nouveau nom pour l'objet marqué. Si le nom existe déjà, la fenêtre réapparait avec un nom inexistant composé du nom saisi et d'une extension numérique

- l'option **connect** est particulière à chaque cas (voir cas par cas ci-dessous)

- l'option **duplicate** n'existe que pour les nœuds.
Elle crée une copie du nœud, indépendante et sans connexion. Une fenêtre est d'abord présentée pour saisir le nom du nouveau nœud, comme pour l'option **rename**, puis la boîte du nouveau nœud doit être positionnée (déplacer halo, cliquer souris, **duplicate** annulé si collision)

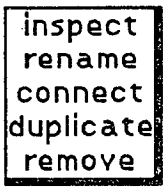
- l'option **remove** détruit, après confirmation, l'objet marqué :
. dans le cas d'un nœud, logiciel ou matériel, détruit également les connexions qui y étaient liées
. dans le cas d'une entrée/sortie, d'un nœud logiciel ou matériel, détruit également, si elles existent, la connexion et l'autre entrée/sortie connectée

- l'option **placement** est particulière à chaque cas (voir cas par cas ci-dessous)

- l'option **redraw** n'existe que pour les entrées/sorties et les connexions ; elle retrace au plus direct la connexion

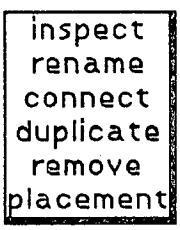
- les options **develop** et **reduce** sont particulières aux connexions (voir cas par cas ci-dessous)

Le menu d'un **nœud logiciel** est :

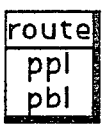


- l'option **connect** crée un nouveau point de sortie sur le nœud marqué (1-choix dans la liste des PROTOCOLs existants, 2-saisie du nom du point de sortie, 3-positionnement du point de sortie), puis permet de sélectionner un autre nœud (le plus proche du curseur clignote et est sélectionné en cliquant la souris) et d'y créer un nouveau point d'entrée de même protocole (1-saisie du nom, 2-positionnement) automatiquement connecté au point de sortie précédemment créé

Le menu d'un **processeur** est :



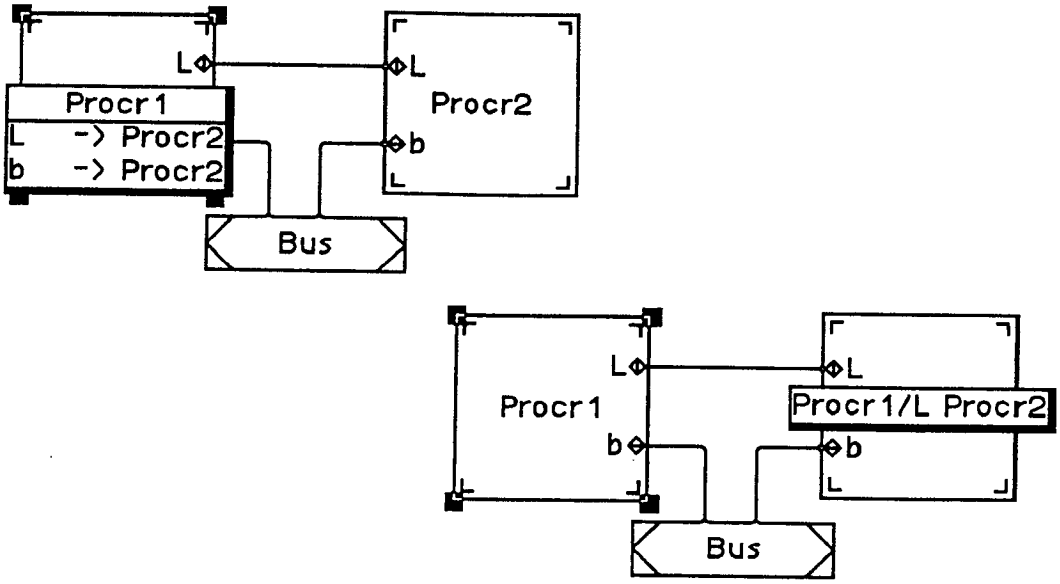
- l'option **connect** présente un menu permettant de choisir entre une connexion logique (**route** = route structurelle) et deux types de connexions physiques (**ppl** = procr-procr et **pbl** = procr-bus)



. l'option **ppl** crée un nouveau lien procr-procr sur le processeur marqué (1-saisie du nom 2-positionnement), puis permet de sélectionner un autre processeur (le plus proche du curseur clignote et est sélectionné en cliquant la souris) et d'y créer un nouveau lien procr-procr (1-saisie du nom 2-positionnement) automatiquement connecté à celui précédemment créé

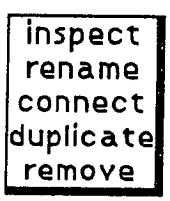
. l'option **pbl** crée un nouveau lien procr-bus sur le processeur marqué (1-saisie du nom 2-positionnement), puis permet de sélectionner un bus (le plus proche du curseur clignote et est sélectionné en cliquant la souris) et d'y créer un nouveau lien procr-bus anonyme (1-positionnement) automatiquement connecté à celui précédemment créé

. l'option **route** crée une route structurelle en proposant des menus successifs, centrés sur les processeurs en commençant par le processeur marqué, avec pour première option la définition de la route accumulée jusqu'à présent, et pour chaque autre option, le nom de l'interface par laquelle la route pourrait sortir du processeur, et le nom du processeur suivant. La sélection de la première option a pour effet d'arrêter là la construction de la route et de présenter une fenêtre de saisie du nom de la route (comme pour les options **rename**), les autres options ont pour effet de poursuivre la construction de la route à partir du processeur suivant. A tout moment la construction de la route peut être abandonnée en ne sélectionnant aucune option.



- l'option **placement** affiche dans la vue texte l'instruction SYNDEX qui a été utilisée pour définir le placement sur le processeur marqué, et change la sélection multiple de façon à faire apparaître ce placement. En combinaison avec la touche **Shift**, place sur le processeur marqué les nœuds logiciels de la sélection multiple

Le menu d'un **bus** est :



- l'option **connect** crée un nouveau lien procr-bus anonyme (1-positionnement), puis permet de sélectionner un processeur (le plus proche du curseur clignote et est sélectionné en cliquant la souris) et d'y créer un nouveau lien procr-bus (1-saisie du nom 2-positionnement) automatiquement connecté à celui précédemment créé

Le menu d'une **entrée/sortie d'un processeur** dépend du contexte :

- l'option **connect** n'apparaît que si l'entrée/sortie marquée n'est pas encore connectée. Elle permet de créer une nouvelle connexion entre l'entrée/sortie marquée et une entrée/sortie compatible (la plus proche du curseur clignote et est sélectionnée en cliquant la souris)

- l'option **disconnect** n'apparaît que si l'entrée/sortie marquée est déjà connectée. Elle détruit la connexion associée

Le menu d'une **entrée/sortie d'un bus** est :

Le menu d'une **connexion entre nœuds matériels** dépend du contexte :

- l'option **develop** ajoute deux nouveaux segments autour du sommet marqué

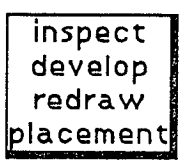
- l'option **reduce** n'apparaît que si le sommet marqué est distant d'au moins deux segments de chaque entrée/sortie. Cette option retire les deux segments autour du sommet marqué, mais seulement si la connexion résultante ne traverse aucune boîte

Le menu d'une **entrée/sortie d'un nœud logiciel** dépend du contexte :

- les options **connect** et **disconnect** fonctionnent comme pour une entrée/sortie d'un nœud matériel

- l'option **placement** n'apparaît qu'avec l'option **connect** et que si les deux nœuds logiciels de part et d'autre de la connexion sont placés sur des processeurs différents. Cette option affiche dans la vue texte l'instruction SYNDEX qui définit le placement de la connexion sur une route structurelle. En combinaison avec la touche **Shift**, place cette communication sur la route choisie dans une liste des routes compatibles (reliant les deux processeurs sur lesquels sont placés les nœuds communicants)

Le menu d'une **connexion entre deux nœuds logiciels** dépend du contexte :

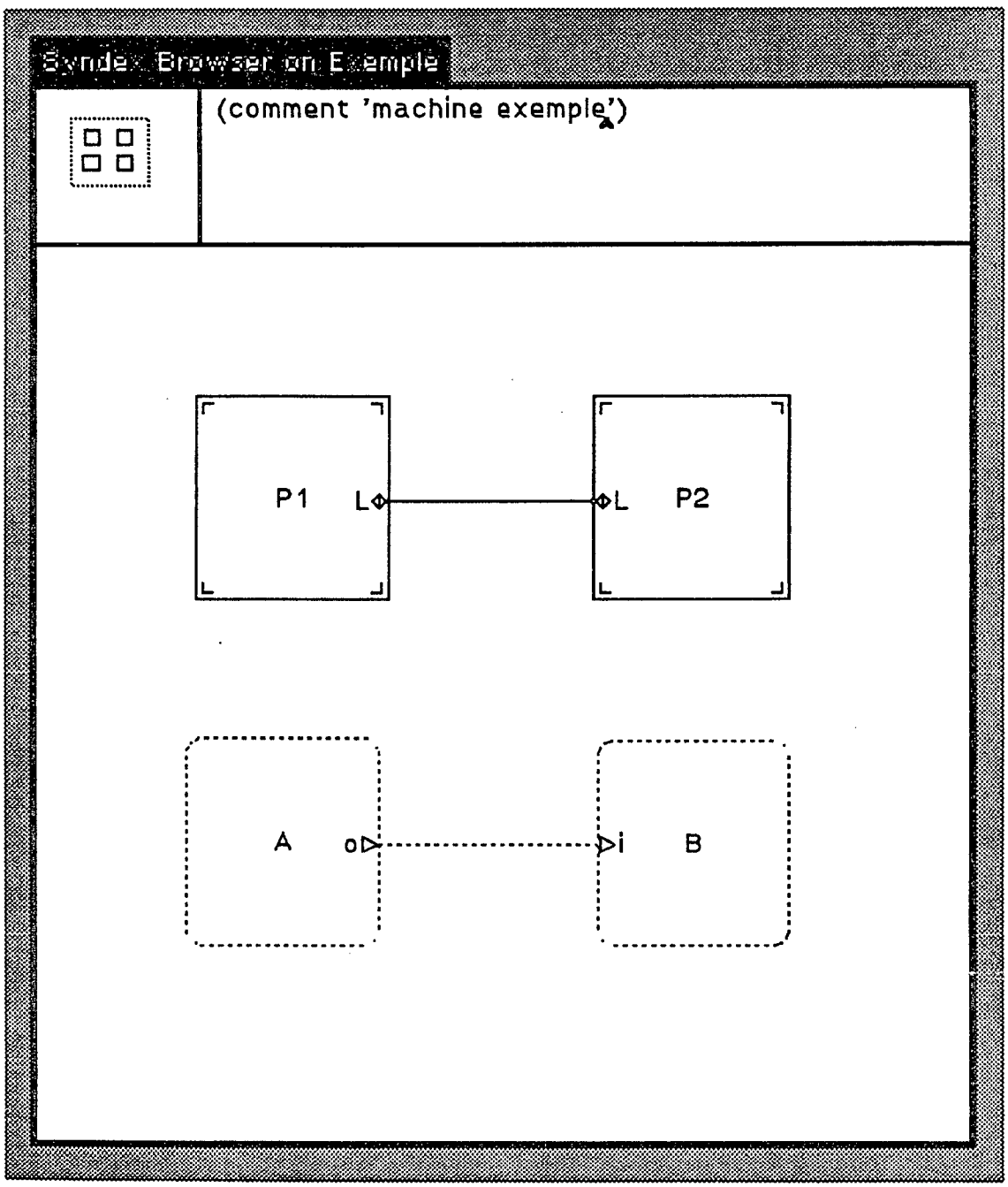


- les options **develop** et **reduce** fonctionnent comme pour une connexion entre deux nœuds matériels

- l'option **placement** n'apparaît que si les deux nœuds logiciels de part et d'autre de la connexion sont placés sur des processeurs différents. Cette option fonctionne comme pour l'entrée et la sortie logicielles de part et d'autre de la connexion

Ce chapitre décrit en détail les étapes à suivre pour créer une machine.

La machine **Exemple** est choisie volontairement simple :
- graphe logiciel
  - . l'application est constituée de deux activités de type function, A et B
  - . A émet vers B des chaines de 10 caractères
- graphe matériel :
  - . le matériel est constitué de deux processeurs P1 et P2
  - . P1 et P2 sont connectés par un lien procr-procr (processeur-processeur)
  - . une route lie les deux processeurs en passant par leur seule connexion
- placement :
  - . A est placé sur P1, B sur P2
  - . leur communication inter-processeur est placée sur la route

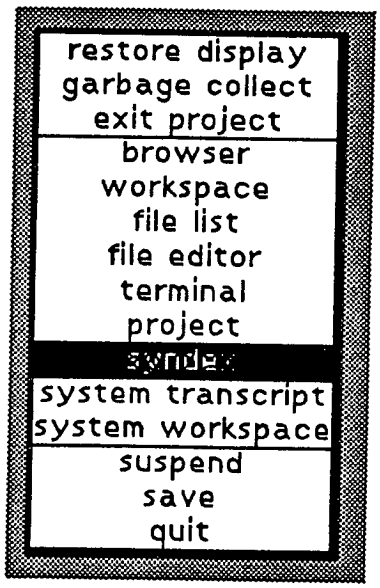


II.1. OUVERTURE D'UN BROWSER SYNDEX

Sous UNIX, charger d'abord l'environnement SYNDEX (créé lors de la procédure d'installation, cf §III.1) en lançant sous shell la commande :

```
Smalltalk80 syndex.im
```

Dans le menu système Smalltalk (obtenu en cliquant avec le bouton central de la souris dans l'arrière plan gris hors fenêtres), sélectionner l'option **syndex** :



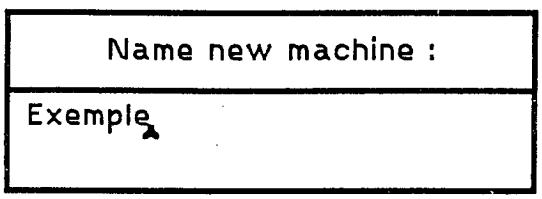
Un cadre en pointillé apparaît, surmonté d'un titre **Syndex Browser**, suivant les mouvements de la souris. Positionner le coin haut-gauche, puis appuyer sur le bouton de la souris tout en positionnant le coin bas-droit, et lâcher le bouton de la souris.

Le browser SYNDEX dessine ses trois vues :
- vue **zoom** en haut à gauche
- vue **texte** en haut à droite, affichant un texte de première aide
- vue d'**édition** graphique en bas

II.2. EDITION D'UNE NOUVELLE MACHINE

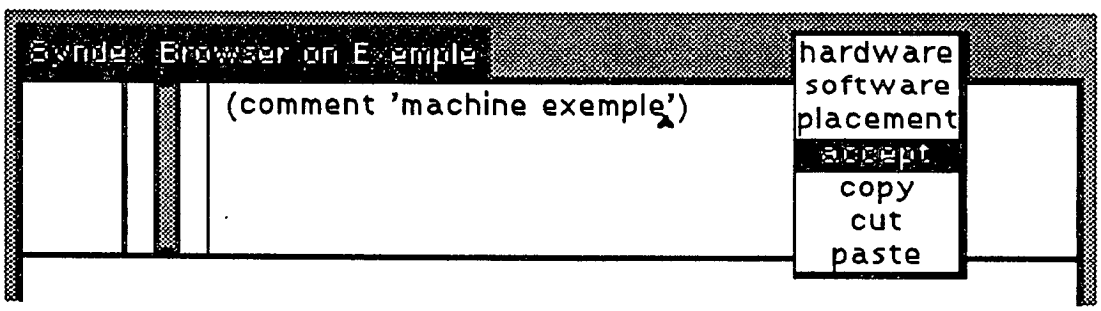
Sélectionner l'option **new** du menu qui apparait dans les trois vues lorsqu'on appuie sur le bouton central de la souris.

Une fenêtre de saisie du nom de la machine apparaît : taper **Exemple<CR>** pour notre exemple.



Un espace est ouvert en mémoire pour une nouvelle machine. Celle-ci devient la machine éditée par le browser, dont le titre devient **Syndex Browser on Exemple.**

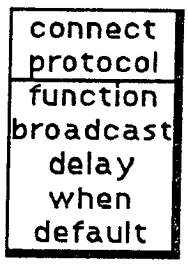
La vue texte affiche **(comment 'no comment')**, invitant à remplacer le **no comment** par un commentaire associé à la machine. Double-cliquer à droite de la quote gauche (ou à gauche de la quote droite) pour sélectionner rapidement tout le texte entre quotes, taper par exemple **machine exemple**, et sélectionner l'option **accept** du nouveau menu de la vue texte pour valider le commentaire (en fait, pour lancer l'interprétation de l'instruction SYNDEX **comment**).



II.3. CREATION DU GRAPHE LOGICIEL

II.3.1. Création de nouveau PROTOCOL

Sélectionner l'option **software** du menu de la vue texte. Un second menu présente la liste des instructions SYNDEX relatives au graphe logiciel.



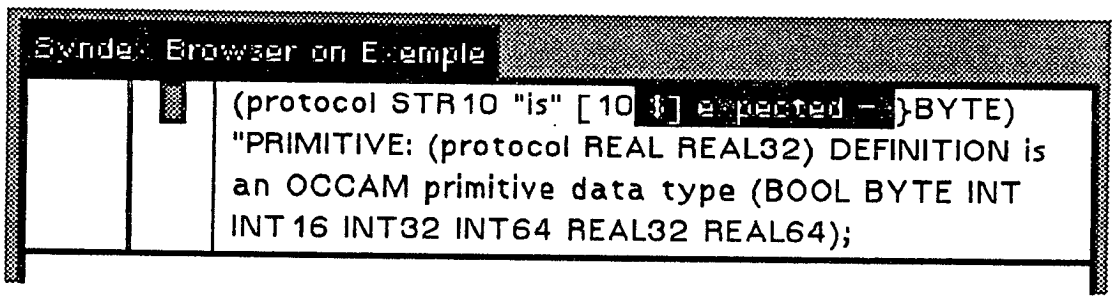
Sélectionner **protocol**. La vue texte affiche une instruction 'à remplir' commentée par un mode d'emploi :

```
(protocol NAME "is" DEFINITION)
"...mode d'emploi..."
```

(tout ce qui se trouve entre guillemets sera ignoré par l'interpréteur SYNDEX).

Double-cliquer sur **NAME** pour sélectionner rapidement le mot complet, et taper **STR10** pour notre exemple ; remplacer de même **DEFINITION** par **[10]BYTE** et sélectionner l'option **accept** du menu de la vue texte.

Si une erreur est détectée, un message donnant la raison de l'erreur sera inséré à l'endroit même de l'erreur, et sélectionné pour être mis en évidence et également pour pouvoir être effacé d'un seul coup. Par exemple, si par inadvertance vous avez remplacé le crochet fermant par une accolade, la vue texte affichera :



Si aucune erreur n'est détectée, le nouveau 'protocol' est mémorisé, et la vue texte affiche :

```
(protocol STR10 "is" [10]BYTE)
"is used by nobody; to remove it : (protocol STR10)"
```

II.3.2. Création de nouvelles activités

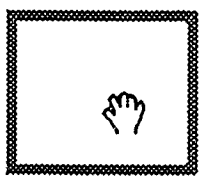
Sélectionner l'option **software** du menu de la vue texte. Un second menu présente la liste des instructions SYNDEX. Sélectionner **function**. La vue texte affiche une instruction 'à remplir' commentée par un mode d'emploi :

```
(function NAME "calls"PROC "dt"DURATION "i/o"INTERFACE)
"...mode d'emploi..."
```

Pour notre exemple, remplacer **NAME** par **A** (le nom de l'activité), **PROC** par **a** (le nom de la procédure OCCAM instanciée par l'activité **A**, compilée séparément dans une librairie à linker plus tard avec le code OCCAM généré), **DURATION** par **5** (la durée estimée d'exécution de la procédure **a** sur un des processeurs), **INTERFACE** par **STR10!o** (une sortie de type **STR10** nommée **o**) et sélectionner l'option **accept** du menu de la vue texte. Si aucune erreur n'est détectée, la vue texte affiche :

```
(function A "calls" a "dt" 5 "i/o" STR10 !o)
```

et un halo, cadre gris aux bords épais, apparaît dans la vue d'édition graphique, suivant les mouvements de la souris, dont le curseur est transformé à cette occasion en petite main :



Positionner le halo pour notre exemple dans le quart bas gauche de la vue, et cliquer un des boutons de la souris : le cadre épais est remplacé par une boîte aux coins arrondis, caractérisant une activité, au centre de laquelle apparaît le nom de l'activité, et sur le bord droit de laquelle apparaît un petit triangle pointé vers l'extérieur de la boîte, caractérisant une sortie, à côté duquel son nom apparaît.

Créer de même la deuxième activité avec une entrée **i**, en interprétant l'instruction SYNDEX :

```
(function B "calls" b "dt" 6 "i/o" STR10 ?i)
```

et positionner sa boîte pour notre exemple dans le quart bas droit de la vue. Sur son bord gauche apparaît l'entrée caractérisée par un petit triangle pointé vers l'intérieur de la boîte.

II.3.3. Menus associés aux objets graphiques

Dans la vue d'édition, une petite marque met en évidence le point clé le plus proche de la souris. Les coins et le centre des boîtes, ainsi que les triangles des entrées/sorties (et les sommets des connexions qui vont être créées) sont des points clés. La marque permet de désigner un objet simplement en s'en approchant (sans avoir besoin de cliquer dessus).
En appuyant sur le bouton central de la souris, on obtient un menu des actions qui peuvent être faites sur l'objet à ce moment.
Pratiquement tous les menus proposent les deux options suivantes :

- **inspect** affiche dans la vue texte une définition de l'objet, commençant souvent par l'instruction SYNDEX qui permettrait de créer le même objet, et d'autres informations du genre références croisées.

- **rename** présente une fenêtre de saisie du nom de l'objet, seule référence textuelle constante à l'objet; le texte saisi est validé par **<CR>**; si le nouveau nom saisi entre en conflit avec le nom d'un autre objet, la fenêtre est présentée une nouvelle fois, proposant par défaut un nom voisin ne provoquant aucun conflit.

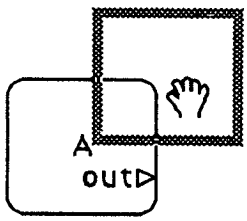
On se reportera au guide de référence (§I.3.4.2) pour la liste complète des actions dans les différents cas.

II.3.4. Création de connexions

Marquer (en approchant la souris) la sortie **o** de l'activité **A** et sélectionner l'option **connect** du menu associé. L'entrée compatible la plus proche de la souris, en l'occurence l'entrée **i** de l'activité **B**, se met à clignoter. Confirmer le choix proposé en cliquant un des boutons de la souris. Une connexion est automatiquement tracée entre l'entrée et la sortie.

II.3.5. Manipulations graphiques

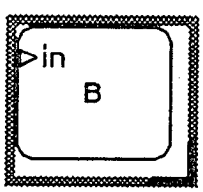
Si le point clé le plus proche de la souris est le centre d'une boîte, ses quatre coins sont marqués; on peut alors déplacer un halo de la boîte (un cadre gris épais) tant que le bouton gauche de la souris est enfoncé.



Au moment où le bouton est relâché, la boîte vient prendre la position définie par l'intérieur du halo, sauf si le celui-ci mord sur une autre boîte, auquel cas une petite fenêtre apparait, affichant un message signalant la 'collision'.

Chaque fois qu'une opération est refusée de cette manière (le message commence généralement par un 'Sorry'), un des boutons de la souris doit être cliqué pour acquitter et effacer le message.

Si le point clé le plus proche de la souris est le coin d'une boîte, on peut le déplacer, faisant apparaître le halo de la boîte, tant que le bouton gauche de la souris est enfoncé; au moment où le bouton est relâché, la boîte prend la dimension définie par l'intérieur du halo (sauf en cas de 'collision'), et les entrées/sorties conservent leurs positions relativement aux bords de la boîte.



Si le point clé le plus proche de la souris est une entrée/ sortie, on peut la déplacer le long du périmètre de la boîte tant que le bouton gauche de la souris est enfoncé. Au moment où le bouton est relâché, la connexion associée (s'il y en a une) est automatiquement retracée vers la nouvelle position de l'entrée/sortie.

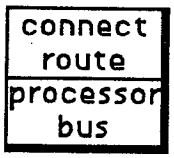
Si le point clé le plus proche de la souris est un sommet d'une connexion, on peut le déplacer tant que le bouton gauche de la souris est enfoncé. Au moment où le bouton est relâché, si la connexion traverse une boîte, le déplacement est annulé.

Chaque fois qu'une connexion est recouverte par une boîte déplacée, elle est automatiquement retracée en contournant les boîtes.

II.4. CREATION DU GRAPHE MATERIEL

II.4.1. Création de nouveaux processeurs

Sélectionner l'option **hardware** du menu de la vue texte. Un second menu présente la liste des instructions SYNDEX relatives au graphe matériel.



Sélectionner **processor**. La vue texte affiche une instruction 'à remplir' commentée par un mode d'emploi :

```
(processor NAME "i/o" INTERFACE)
"...mode d'emploi..."
```

Pour notre exemple, remplacer **NAME** par **P1** (le nom du processeur), **INTERFACE** par **PPL L=100** (un lien procr-procr nommé L, avec 100 octets pour chacune de ses deux mémoires tampons, une en entrée, l'autre en sortie), et sélectionner l'option **accept** du menu de la vue texte.

Si aucune erreur n'est détectée, la vue texte affiche :

```
(processor P1 "i/o" PPL L=100)
```

et comme pour les activités, un halo apparaît dans la vue édition. Positionner la boîte du processeur pour notre exemple dans le quart haut gauche de la vue.

La boîte a ses coins doublés, caractéristiques des boîtes de processeurs, et porte sur son bord gauche un petit losange barré parallèlement au bord de la boîte, caractéristique des interfaces pour lien procr-procr (les interfaces pour lien procr-bus sont barrées perpendiculairement au bord de la boîte).

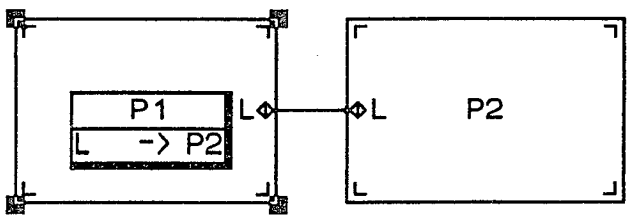
Pour créer le second processeur, réplique du premier, sélectionner l'option **duplicate** du menu associé à **P1**. Une fenêtre de saisie du nom du nouveau processeur apparaît (comme dans le cas de l'option **rename**). Pour notre exemple, nommer le nouveau processeur **P2**, puis positionner son halo dans le quart haut droit de la vue et cliquer un des boutons de la souris. Le processeur **P2** s'affiche. Au nom près, c'est une copie conforme du processeur **P1**.

Pour connecter **P1** et **P2**, procéder comme pour **A** et **B**. Sélectionner l'option **connect** du menu de l'interface **L** du processeur **P1**. L'interface compatible la plus proche de la souris, en l'occurence l'interface **L** du processeur **P2**, se met à clignoter. Confirmer le choix proposé en cliquant un des boutons de la souris. Une connexion est automatiquement tracée entre les deux interfaces.

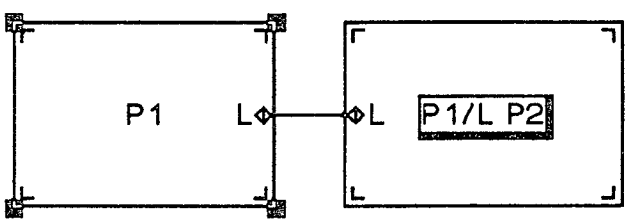
II.4.2. Création d'une route structurelle

Pour créer une route entre P1 et P2, sélectionner l'option **connect** du menu du processeur **P1**. On obtient le menu :

L'option **route** présente un premier menu, centré sur **P1**:



En sélectionnant l'option **L->P2**, on obtient un nouveau menu, centré sur **P2** :



En sélectionnant l'unique option, une fenêtre de saisie du nom de la route apparaît proposant **P1L.P2**; taper **<CR>** pour accepter ce nom. La vue texte affiche :

```
(route P1L.P2 "via" P1/L P2)
"is used by nobody; to remove it : (route P1P2)"
```

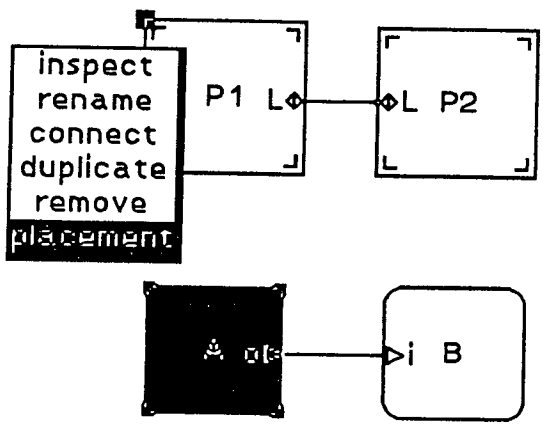
confirmant qu'une route nommée **P1L.P2** a été mémorisée, partant de **P1**, en sortant par son interface **L**, et arrive dans **P2** (sous-entendu par son interface **L** connectée à l'interface **L** de **P1**).

Chacun des menus successifs, centrés sur les processeurs, propose pour première option la définition de la route accumulée jusqu'à présent, et pour chaque autre option, le nom de l'interface par laquelle la route pourrait sortir du processeur, et le nom du processeur suivant.
La sélection de la première option a pour effet d'arrêter là la construction de la route, les autres options ont pour effet de poursuivre la construction de la route à partir du processeur suivant.
A tout moment la construction de la route peut être abandonnée en ne sélectionnant aucune option.

II.5. PLACEMENT DES ACTIVITES ET DES COMMUNICATION

II.5.1. Placement des activités de calcul sur les processeurs

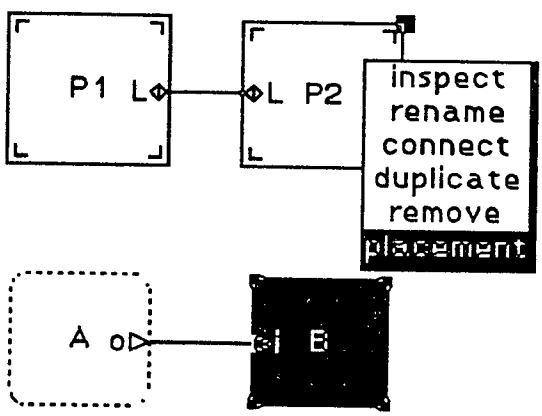
En maintenant la touche **Shift** enfoncée, cliquer sur les activités qui doivent être placés sur le même processeur. Chaque activité sélectionnée est affichée en inversion vidéo. On peut retirer une activité de la sélection en cliquant à nouveau dessus.
Pour notre exemple, sélectionner l'activité **A**, puis, tout en maintenant toujours la touche **Shift** enfoncée, sélectionner l'option **placement** du menu de **P1** :



L'activité **A** se redessine en pointillé, signifiant qu'elle est maintenant placée, et la vue texte affiche :

```
(placeact "on" P1 "partition" A)
```

signifiant que le processeur **P1** prend en charge l'exécution des activités citées après **"partition"**, en l'occurence la seule activité **A**.
Placer de même l'activité **B** sur le processeur **P2**.



La sélection de l'option **placement** au menu d'un processeur, sans enfoncer la touche **Shift**, affiche simplement le placement associé à ce processeur, dans la vue texte par l'instruction SYNDEX **placeact** correspondante, et dans la vue édition en changeant la sélection de manière à refléter ce placement.

Une fois toutes les activités placées sur des processeurs, tous les canaux de communication intra-processeur (entre deux activités placées sur le même processeur) apparaissent également en pointillé, et seuls les canaux de communication inter-processeurs apparaissent encore en trait plein, pour rappeler qu'ils doivent être placés sur des routes structurelles.

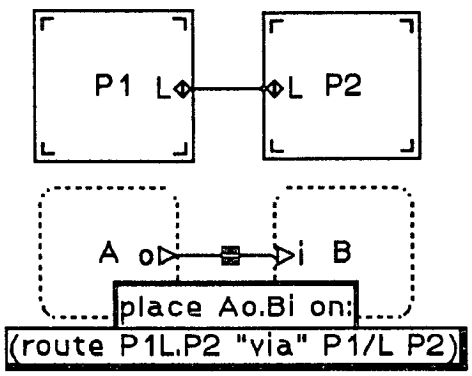
II.5.2. Placement des canaux de communication sur les routes structurelles

Sélectionner l'option **placement** au menu d'une connexion à router, tout en maintenant la touche **Shift** enfoncée.
Si aucune route n'a encore été définie entre les deux processeurs sur lesquels se trouvent les deux activités communiquant, cette situation est signifiée par un message.
Sinon, un menu présente la liste des routes structurelles reliant ces deux processeurs, et la communication inter-processeurs est placée sur la route sélectionnée.

La connexion se retrace en pointillé, signifiant qu'elle est maintenant placée, et la vue texte affiche l'instruction **placecom** associée au placement.

La sélection de l'option **placement** sans enfoncer la touche **Shift** affiche dans la vue texte l'instruction **placecom** correspondante.

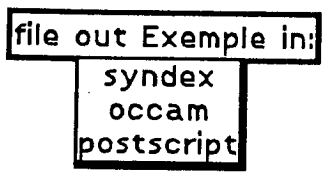
Pour notre exemple, enfoncer la touche **Shift** et sélectionner l'option **placement** du menu de la sortie **o** de l'activité **A** : une seule route est présentée au menu, la route **P1L.P2** déjà créée.



II.6. GENERATION DE CODE

L'environnement SYNDEX permet de générer trois types de code textuel :
- du code SYNDEX pour sauvegarde sur fichier
- du code OCCAM pour exécution sur machine de simulation ou machine cible
- du code postscript pour impression papier

Ces trois fonctionnalités sont accessibles en sélectionnant l'option **file out** du menu de la vue zoom, qui présente le menu :



II.6.1. Impression de la machine sur papier (code postscript)

Sélectionner l'option **file out** du menu de la vue zoom, puis l'option **postscript** du menu suivant, tout en maintenant la touche **Shift** enfoncée. Un fichier postscript, nommé **hardCopy.ps** est alors créé et automatiquement envoyé à l'imprimante.

Si la touche **Shift** n'est pas enfoncée, le fichier est nommé comme la machine avec une extension **.ps**, **Exemple.ps** pour notre exemple, et devra être envoyé à l'imprimante par une commande shell :

```
lpr -c Exemple.ps
```

qu'on pourra taper dans une fenêtre terminal, obtenue en sélectionnant l'option **terminal** dans le menu système Smalltalk (obtenu en appuyant le bouton central de la souris dans l'arrière plan gris hors fenêtres).

Le résultat de l'impression est donné en annexe (cf §III.5).

II.6.2. Génération de code exécutable (code occam)

Sélectionner l'option **file out** du menu de la vue zoom, puis l'option **occam** du menu suivant. Le code source OCCAM est alors généré dans un fichier texte nommé comme la machine avec une extension **.occam.**

Ce fichier doit être ensuite intégré dans l'environnement de compilation OCCAM, compilé et linké avec les bibliothèques système et application (voir §I.2.4 pour plus de détails).

Le contenu du fichier Exemple.occam est donné en annexe (cf §III.4).

II.6.3. Sauvegarde de la machine sur fichier (code syndex)

Sélectionner l'option **file out** du menu de la vue zoom, puis l'option **syndex** du menu suivant. Une définition textuelle complète (graphe logiciel, graphe matériel, placement, graphique) de la machine en l'état est alors sauvegardée dans un fichier dont le nom est contitué de celui de la machine avec une extension **.syndex.**

Le contenu du fichier Exemple.syndex est donné en annexe (cf §III.3).

II.6.4. Sauvegarde de la session de travail

Pour conserver l'état de la session de travail pour la prochaine session, sélectionner l'option **quit** du menu système Smalltalk (obtenu en appuyant sur le bouton central de la souris dans l'arrière plan gris hors fenêtres), puis l'option **Save then quit** du menu suivant, et taper **<CR>** en réponse à la fenêtre de saisie du nom du fichier de sauvegarde de la session.

III.1. Installation de l'environnement SYNDEX sous UNIX

Dans cette version, l'environnement SYNDEX est livré sous forme de sources sur une bande 1/4". Créer un répertoire et y charger la bande au moyen des commandes shell suivantes :

```
mkdir syndex
cd syndex
tar -xvf /dev/rst0
```

Les noms des fichiers chargés s'affichent. Parmi eux, le fichier INSTALL.SYNDEX donne la marche à suivre pour le reste de la procédure d'installation. Son contenu peut être affiché au moyen de la commande shell :

```
more INSTALL.SYNDEX
```

Ce fichier contient, entre autres, les quelques lignes de source Smalltalk [SPS] à interpréter pour créer, à partir des fichiers sources qui viennent d'être chargés, une image Smalltalk comprenant le browser SYNDEX.
L'opération dure environ 10 minutes sur un Sun3/60...

III.2. Syntaxe SYNDEX

La syntaxe est donnée sous la forme Bacchus Naur : chaque élément syntaxique non terminal apparaît en début de ligne, en majuscules, suivi d'un commentaire optionnel entre parenthèses ; sa définition apparaît sur la ligne suivante, avec pour seul méta caractère la barre verticale '|' signifiant l'alternative.

Les éléments syntaxiques suivants permettent de définir identificateurs, entiers et chaînes de caractères :

Name	(identificateur OCCAM [a-zA-Z][a-zA-Z.0-9]*)
Integer	(entier relatif)
String	('tous caractères entre apostrophes')

Tout texte apparaissant entre guillemets est ignoré (commentaires smalltalk).

Chaque instruction SYNDEX est délimitée par des parenthèses, et commence par un mnémonique désignant l'instruction, suivi des paramètres de l'instruction.

Chaque instruction dépend du contexte défini par les instructions précédentes. Cette dépendance se traduit dans la syntaxe par les conventions suivantes :

- chaque fois qu'une instruction définit un nouvel identificateur, celui-ci apparaît dans la syntaxe de l'instruction sous la forme **NewXXXName** avec **XXX** représentant le type de l'objet désigné par l'identificateur

- chaque fois qu'une instruction fait référence à un objet déjà défini, son identificateur apparaît sous la forme **XXXName** dans la syntaxe de l'instruction.

Ces conventions se traduisent par les définitions génériques suivantes :

NewXXXName (nouvel identificateur, pour un objet de type XXX)
    xxxName
XXXName (identificateur déjà defini, pour un objet de type XXX)
    Name
XXXList (liste comportant au moins un objet de type XXX)
    XXX | XXXList

Dernière convention :
les éléments syntaxiques terminaux (ou litéraux) apparaissent en caractères gras.

SyndexSource
    InstructionList

Instruction
    Comment | Iterations |
    Protocol | Activity |
    Processor | Bus | Route |
    Connect | Partition | Communication

Comment
    (**comment** String)

Iterations (number of samples for one simulation)
    (**iterations** Integer)

Protocol (OCCAM protocols for inter-activity chanels)
(**protocol** NewProtocolName ProtocolDef)

ProtocolDef (as in OCCAM, except for counted array protocol)
SimpleProtocol | SequentialProtocol

SequentialProtocol
SimpleProtocol ; SimpleProtocol |
SimpleProtocol ; SequentialProtocol

SimpleProtocol
BasicProtocol | CountedArrayProtocol

CountedArrayProtocol (Integer gives maximum array size)
: OCCAMIntegerType < Integer : BasicProtocol

BasicProtocol (scalar or fixed array)
OCCAMPrimitiveType | [Integer] BasicProtocol

OCCAMPrimitiveType
BOOL | **REAL32** | **REAL64** | OCCAMIntegerType

OCCAMIntegerType
BYTE | **INT** | **INT16** | **INT32** | **INT64**

Activity
Function | Delay | Broadcast

Function (Integer is for function duration in some user-defined units)
(**function** NewFunctionName PROC Integer FunctionInterfaceDef)

PROC (called library procedure)
Name

FunctionInterfaceDef
SameProtocol | SameProtocol , FunctionInterfaceDef

SameProtocol
ProtocolName ConnectorDefList |
OCCAMPrimitiveType ConnectorDefList

ConnectorDef
InputDef | OutputDef | ParamDef

InputDef
? NewInputName

OutputDef
! NewOutputName

ParamDef
: NewParamName = String

Delay (String is the delay initial value)
(**delay** NewDelayName ProtocolName String)

Broadcast
(**broadcast** NewBroadcastName ProtocolName)

Processor
(**processor** NewProcrName ProcrInterfaceDef)

ProcrInterfaceDef
SameType | SameType , ProcrInterfaceDef

SameType (PPL is for procr-procr link, MPL for procr-bus link)
PPL PPLDefList | **PBL** PBLDefList

PPLDef (Integer is PPL buffers size in bytes)
NewPPLName = Integer

PBLDef (Integer is PBL buffers size in bytes)
NewPBLName = Integer

Bus
(**bus** NewBusName BusType)

BusType (only one type for now)
rr ('round robin')

Connect
(**connect** ActOut ActIn) |
(**connect** ProcrPPL ProcrPPL) |
(**connect** BusName ProcrPBLList)

ActOut
ActivityName / OutputName

ActIn
ActivityName / InputName

ProcrPPL
ProcrName / PPLName

ProcrPBL
ProcrName / PBLName

Route
(**route** NewRouteName RouteDef)

RouteDef
ProcrName | (internal route)
ProcrPPL RouteDef |
ProcrPBL RouteDef

Partition
(**placeact** ProcrName ActivityNameList)

Communication
(**placecom** ActIn RouteName)

III.3. Code SYNDEX de la machine Exemple

```
"SYNDEX version v.0
Exemple (31 August 1989 6:22:11 pm )"
  (comment 'machine exemple')

"SOFTWARE SECTION"
  (iterations 100)
  (protocol STR10 "is" [10]BYTE)
  (function A "calls" a "dt" 5 "i/o" STR10 !o)
  (function B "calls" b "dt" 6 "i/o" STR10 ?i)
  (connect A/o B/i)

"HARDWARE SECTION"
  (processor P1 "i/o" PPL L=100)
  (processor P2 "i/o" PPL L=100)
  (connect P1/L P2/L)
  (route P1L.P2 "via" P1/L P2)

"PLACEMENT SECTION"
  (placeact "on" P1 "partition" A)
  (placeact "on" P2 "partition" B)
  (placecom "from A/o to" B/i "on" P1L.P2)

"GRAPHIC POSITIONS SECTION"
  (position A "frame" 152 352 558 776 "o" 2.5)
  (position B "frame" 495 690 560 778 "i" 1.5 495 669 427 667 352)
  (position P1 "frame" 164 350 145 335 "L" 2.51351)
  (position P2 "frame" 495 680 143 332 "L" 1.52703 495 243 435 243 350)

"That's all folks !!"
```

III.4. Code OCCAM de la machine exemple

```
--------------------------------------------------
-- Machine Exemple (31 August 1989 6:41:05 pm )
-- SYNDEX version ALPHA.TEST
--------------------------------------------------
#USE CONST -- global constants
#USE PES -- processor PES1 and PESb library
#USE BL -- processor BL library
#USE STR10 -- PROTOCOL STR10 IS [10]BYTE:
#USE Exemple.act -- application activities library

-- destIDs in Procr P1 -- A --
VAL P1L IS 0 (BYTE): -- PPL >< P2L
VAL Ao IS 1 (BYTE): -- PE >  0:1:D >Bi/P2> 1:1:S

-- destIDs in Procr P2 -- B --
VAL P2L IS 0 (BYTE): -- PPL >< P1L
VAL Bi IS 1 (BYTE): -- PR < Ao/P1

-- inter-processor routeIDs --
VAL P1L.P2.dir IS 0 (INT16): -- route P1 -> P2 = ( P1L P2L )
VAL P1L.P2.rev IS 1 (INT16): -- route P1 <- P2, reverse route
-- 0:1:D   Ao -> Bi
-- 1:1:S   Ao <- Bi

-- Procr P1 -- L><P2L
-- in machine Exemple ---- includes A --
CHAN OF MESSAGE P1L.P2L:
PROC P1 (CHAN OF MESSAGE iL, oL )
  -- fifos --
  [100]BYTE P1L.fifoIn, P1L.fifoOut: -- PPL >< P2L
  -- software bus BL -- connects -- P1L Ao --
  [2]CHAN OF MESSAGE iBL, oBL:

  -- FunctionNode A -- o>PESTR10>Bi --
  -- in Procr P1 --
  CHAN OF STR10 Ao.PE:
  CHAN OF SYNC PE.Ao:
  PROC A (CHAN OF SYNC So, CHAN OF STR10 Do )
    [10]BYTE do:
    INT so: -- synchro output buffer
    SEQ clock=0 FOR NbIterations
      SEQ
        -- no input
        a ( do )
        PAR -- output
          SEQ
            Do ! do
            So ? so
  :

  PAR -------- body of Procr P1 --
    PES1 ( oBL[0], iBL[0], iL, oL, P1L.fifoIn, P1L.fifoOut )
    PESTR10 ( Ao.PE, PE.Ao, oBL[1], iBL[1],
        Ao, P1L.P2.dir, Bi )
    BLrr ( iBL, oBL, [P1L,local] )
    A ( PE.Ao, Ao.PE )
:
```

```
-- Procr P2 -- L><P1L --
-- in Machine Exemple ---- includes B --
CHAN OF MESSAGE P2L.P1L:
PROC P2 (CHAN OF MESSAGE iL, oL )
  -- fifos --
  [100]BYTE P2L.fifoIn, P2L.fifoOut: -- PPL >< P1L
  -- software bus BL -- connects -- P2L Bi --
  [2]CHAN OF MESSAGE iBL, oBL:

  -- FunctionNode B -- i<PRSTR10<Ao --
  -- in Procr P2 --
  CHAN OF STR10 PR.Bi:
  CHAN OF SYNC Bi.PR:
  PROC B (CHAN OF STR10 Di, CHAN OF SYNC Si )
    [10]BYTE di:
    INT si: -- synchro input buffer
    SEQ clock=0 FOR NbIterations
      SEQ
        PAR -- input
          SEQ
            Di ? di
            Si ! si
        b ( di )
        -- no output
  :

  PAR -------- body of Procr P2 --
    PES1 ( oBL[0], iBL[0], iL, oL, P2L.fifoIn, P2L.fifoOut )
    PRSTR10 ( Bi.PR, PR.Bi, oBL[1], iBL[1],
        Bi, P1L.P2.rev, Ao )
    BLrr ( iBL, oBL, [local,P2L] )
    B ( PR.Bi, Bi.PR )
:

PAR -------- body of Exemple--
  P1 (  P2L.P1L, P1L.P2L )
  P2 (  P1L.P2L, P2L.P1L )

-- That's all folks !! --
```

III.5. Références bibliographiques

[GHE] Quelques méthodes de répartition et d'ordonnancement de processus de traitement du signal sur un réseau de transputers. Thèse de doctorat, Université de Paris-Sud Orsay, 1989. Ghezal N.

[OCC] INMOS. OCCAM 2 Reference Manual. Prentice Hall, 1988.

[SIG] Hybrid dynamical systems theory and the language SIGNAL. Rapport de recherche INRIA, 1988. Benveniste A., Le Goff B., Le Guernic P.

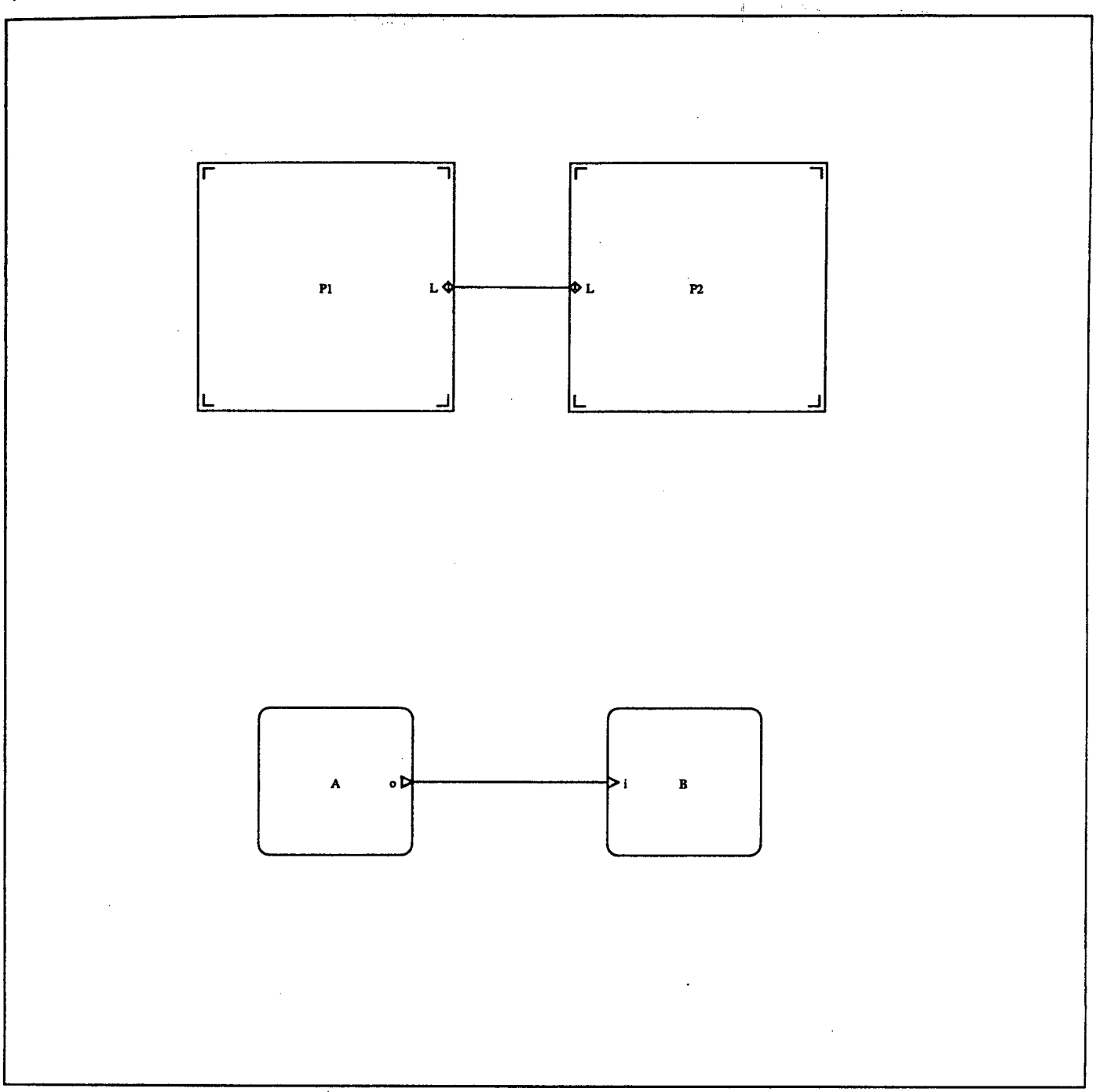
[SIM] Simulateur et exécutif pour machine de traitement du signal. Rapport de recherche INRIA, 1989. Matiatos S., Piovesan P., Sorel Y., Sorine M.

[SMA] Smalltalk-80, the langage and its implementation. Addison-Wesley, 1983. Goldberg A., Robson D.

[SPS] The Smalltalk-80 Programming System, version 2.4. ParcPlace Systems, 1988.

[TDS] INMOS. Transputer Development System. Prentice Hall, 1988.

[TRA] INMOS. Transputer Reference Manual. Prentice Hall, 1988.



Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique