ICOT Technical Memorandum: TM-0803

TM-0803

並列オブジェクト指向言語POOLシステム および評価用プログラム使用手引き

佐藤秀樹，末永富美代(富士通)

September, 1989

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

並列オブジェクト指向言語ＰＯＯＬシステム
および評価用プログラム使用手引き

佐藤　秀樹，　末永　富美代
（富士通株式会社）

目　　次

1．概要

本稿は，PSI-Ⅱ上のPseudo Multi-PSI-V1 システムを用いて作成された並列オブジェクト指向言語POOL(Parallel Object Oriented Language) システムおよびPOOLシステムの評価用プログラムとしての協調型ポートフォリオ選択実験システムESCOPS(Experimental System for Cooperative Portfolio Selection) の使用手引きである。

1．1　並列オブジェクト指向言語POOLの特徴

並列オブジェクト指向言語POOLは，PSI-Ⅱ上のPseudo Multi-PSI-V1 システムを用いて実現されている。POOLの主な特徴を以下に示す。

① オブジェクト指向機能

POOLのオブジェクトは，スロットとメソッドとを備える。クラスはオブジェクトの定義を与えるものであり，その内容はクラス名，上位クラス定義，スロット定義（初期値指定を含む），メソッド述語定義，ローカル述語定義から成る。オブジェクトは，対応するクラスのインスタンスである。オブジェクトによる処理は，メソッド述語ゴールの評価，ローカル述語ゴールの評価により進められる。尚，クラス自身は，オブジェクトとしては扱われない。

② 多重継承

クラスは上位クラス定義によって，任意個のクラスとの間に上位―下位関係を持つことができる。この関係に基づき，下位クラスは上位クラスのスロットとメソッド述語の定義を多重に継承することができる。スロットおよび初期値指定の継承は，クラス定義における上位クラス定義の記述順により規定される継承木上の深さ優先探索の順に行われる。また，メソッド述語定義の継承は，下位クラスから上位クラスに向けての並列的な探索に基づき行われる。

③ 並列実行

メソッド述語とローカル述語は，Pseudo Multi-PSI-V1 システムが提供する言語FGHC (Flat Guarded Horn Clause)の節形式により定義される。これらの述語の評価は，Pseudo Multi-PSI-V1 システムの処理系により行われる。従って，全ての述語ゴールの評価は，並列に実行される。言い換えれば，異なるオブジェクトにおける述語ゴールの評価，同一のオブジェクト内の異なる述語ゴールの評価，述語のボディ部の述語ゴールの評価は，全て並列に実行される。

④ 排他制御

オブジェクトによる処理過程に対する状態は，全てのオブジェクトのスロットの値の組により規定される。しかし，③の「並列実行」で述べたようにオブジェクトによる処理は並列に実行される。このため，異なるオブジェクトのメソッド述語ゴールの評価，同一のオブジェクト内の異なるメソッド述語ゴールの評価の間で共有されるスロットに対するアクセスを逐次化しなければ，矛盾した状態を生じうる。こうした事態

を防ぐため，排他制御機能を備えたスロット・アクセスのための組込みメソッド述語が提供されている。

⑤ ESP プログラムの呼出し

専門家システムなどの開発において，利用者インタフェース作成に要する工数が全体工数の中で大きな割合を占めることは，周知の事実である。このため，SIMPOSの主に入出力機能の利用を目的として，ESP プログラムを呼出すための組込み述語が提供されている。

1．2 評価用プログラムESCOPS

協調型ポートフォリオ選択実験システムESCOPSは，入出力に関係する利用者インタフェース部分を除いて並列オブジェクト指向言語POOLを用いて試作された評価用プログラムである。ESCOPSは，複数の投資手段に対する分散投資の組合せであるポートフォリオを選択する。投資手段としては株式，現先，コールなどがあるが，ESCOPSにおいては債券の類を対象としている。こうした債券投資の基本体系は，①投資目標の設定，②運用戦略と戦術の適用，③運用成果の評価といった３段階から構成される。①の段階では，資金の性格を明確にし，経営的判断の下に収益性，安全性，流動性，リスク許容度などに関する基準やガイドラインとして投資目標が設定される。②の段階では，投資目標に基づいて実際に投資が行われる。③の段階では，投資戦略に基づく運用結果の評価が行われる。この評価結果は，前の段階に対してフィードバックされる。ESCOPSが選択するポートフォリオには，①の段階における基準ポートフォリオと②の段階における運用ポートフォリオとがある。

(1)基準ポートフォリオ

基準ポートフォリオは，各債券の過去の投資収益率の時系列データを基に，債券の組全体としてのリターン（収益性）とリスク（安全性）とを考慮して選択されるポートフォリオである。ここで，ポートフォリオのリターンとリスクとは，次のように定義される。

$$\text{ポートフォリオのリターン} \quad E = \sum_{i=1}^{N} X_i E_i$$

$$\text{ポートフォリオのリスク} \quad S^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} X_i X_j S_i S_j R_{ij}$$

N : 債券数

$X_{i(j)}$: 債券i(j)の構成比率

$E_{i(j)}$: 債券i(j)の期待収益率（投資収益率の平均値）

$S_{i(j)}$: 債券i(j)のリスク（投資収益率の標準偏差）

R_{ij} : 債券i と債券j との投資収益率の相関係数

また，ポートフォリオを選択するため，そのリターンとリスクとを総合的に評価するために使われる目的関数は，次のように定義される。

目的関数　$F = E - S^2 / k$

$$= \sum_{i=1}^{N} X_i E_i - 1/k \sum_{i=1}^{N} \sum_{j=1}^{N} X_i X_j S_i S_j R_{ij}$$

上記の目的関数は，プラスの要因となるリターンとマイナスの要因となるリスクをリスク許容度ｋで重み付けしたものとの差によって，ポートフォリオの評価としている。リスク許容度は，投資家が持つリスクに対する態度に関する重み係数である。すなわち，Risk Taker はｋの値を大きくするであろうし，Risk Arbiterはｋの値を小さくすることになる。

(2)運用ポートフォリオ

運用ポートフォリオは，各種の運用手法を適用し，入力として与えられるポートフォリオの内容を入れ替えた結果のポートフォリオを選択する。こうした運用手法は，大別すると積極的運用手法と保守的運用手法とに分けられる。積極的運用手法は，予想金利あるいは銘柄間の利回り格差といったように特定の情報を基に債券銘柄の入替えを行うことにより，収益を得ようとする考え方である。また，保守的運用手法は，機械的な一定の規則を適用する運用手法である。各運用手法は，対応する評価指標に関する目標条件を備えている。選択されるポートフォリオは，これらの目標条件を全て満足することが要請される。表１に，ESCOSPにおいて利用可能な運用手法を示す。目標条件とは，例えば金利予想運用の場合では「特定期間内における所有期間利回りが８％以上，できれば１０％以上」といったような許容範囲から成る。運用ポートフォリオ選択問題における解は，全ての運用手法の目標条件の許容範囲におさまることが要請される。

表1　運用手法

運用手法	カテゴリ	内容
金利予想運用	積極的運用	金利変化の予想に基づき，価格変動リスクを回避するとともに高い収益を確保しようとする運用法。目標条件は，特定期間における所有期間利回りに関する。
利回り格差運用		債券銘柄間に発生する利回り格差の状況に基づき，割高な銘柄から割安な銘柄への入れ替えにより，高い収益を求める運用法。目標条件は，特定期間における所有期間利回りに関する。
免疫型運用	保守的運用	将来の金利変動の影響を避けることを意図して，目標時点のポートフォリオの収益を現時点で固定させる運用法。目標条件は，目標時点からの乖離の程度に関する。
バーベル型運用		ポートフォリオの構成を短期債と長期債で一定の割合に維持しようとする運用法。目標条件は，短期債と長期債の割合のバランスの程度に関する。
ラダー型運用		ポートフォリオの構成を残存期間別に均等に維持しようとする運用法。目標条件は，残存期間別の構成のバランスに関する。

2．並列オブジェクト指向言語POOLシステム

並列オブジェクト指向言語POOL(Parallel Object Oriented Language) システムは，PSI-Ⅱ上のPseudo Multi-PSI-V1 システムを用いて試作された。

2．1　並列オブジェクト指向言語POOL

以下では，並列オブジェクト指向言語POOLに関して，その構文を中心として説明を行う。尚，構文の記述には，次の点でBNF を拡張した拡張BNF を用いるする。

- "X" は，終端記号Xを表す。
- 〔X〕は，Xの０回以上の任意数の繰返しを表す。
- 〔X〕は，Xの出現が選択的であることを表す。

(1)クラス定義

プログラムは，１個以上のクラス定義から成る。クラス定義は，クラス名，上位クラス定義，スロット定義，メソッド述語定義，ローカル述語定義から成る。上位クラス定義列は，当該クラスによってスロットおよびメソッド述語の定義を継承される上位クラスを指定する。スロット定義列は，当該クラスに属するオブジェクトの状態値を格納するスロットを定義する。メソッド述語定義列は，当該クラスが備えるメソッド述語を定義する。ローカル述語定義列は，その呼出しが当該クラス内のみに制限されるローカル述語を定義する。

構文

```
<プログラム>::= <クラス定義>　〔　<クラス定義>　〕
<クラス定義>::=
        "class" "(" "name" "("　<クラス名> ")" ","
                    "super" "("　" 〔" 〔<上位クラス定義列>〕 "〕" ")"　","
                    "slot" "(" " 〔" 〔<スロット定義列>〕 "〕" ")" ","
                    "method" "(" " 〔" 〔<メソッド述語定義列>〕 "〕" ")" "."
                    "local" "("　" 〔" 〔<ローカル述語定義列>〕 "〕" ")" "."
```

(2)上位クラス定義

上位クラス定義列は，当該クラスの直接の上位クラスを定義する。この定義に基づき，当該クラスは上位クラスのスロットとメソッド述語の定義を多重に継承することができる。スロットおよび初期値指定の継承は，クラス定義における上位クラス定義の記述順により規定される継承木上の深さ優先探索の順に行われる。また，メソッド述語定義の継承は，下位クラスから上位クラスに向けての並列的な探索に基づき行われる。

構文

```
<上位クラス定義列>::= <クラス名>〔 ","　<クラス名>　〕
```

(3)スロット定義

スロット定義列は，当該クラスに属するオブジェクトの状態値を格納するスロットを定義する。スロット定義では，スロット名およびオブジェクト生成時のスロットの初期値が選択的に与えられる。スロットの初期値の暗黙値としては，0が割当てられる。尚，データ型として提供されるものは，Pseudo Multi-PSI-V1 システムのそれに同じであるので注意されたい。

構文

```
<スロット定義列>::= <スロット定義> { "," <スロット定義> }
<スロット定義>::= <スロット名> |
                  "(" <スロット名> "," <初期値> ")"
```

(4)述語定義

述語には，メソッド述語とローカル述語とがある。メソッド述語はオブジェクト間での呼出しが可能であり，ローカル述語は当該クラス内のみに呼出しが制限されている。述語定義は，1個以上の節定義から成る。節定義は，Pseudo Multi-PSI-V1 システムと同様にヘッド，ガード・ゴール列，ボディ・ゴール列により構成される節形式を取る。ガード・ゴール列には，Pseudo Multi-PSI-V1 システムで提供されるFGHCガード部組込み述語ゴールを書くことができる。ボディ・ゴール列には，FGHCボディ部組込み述語ゴール，FGHCユーザ定義述語ゴール，ローカル述語ゴール，POOL組込み述語ゴール，ESP プログラム呼出し組込み述語ゴールを書くことができる。

構文

```
<メソッド述語定義列>::= <述語定義> { "," <述語定義> }
<ローカル述語定義列>::= <述語定義> { "," <述語定義> }
<述語定義>::= <節定義> | "{" <節定義> { "," <節定義> } "}"
<節定義>::= "(" <ヘッド> ":-"
                <ガード・ゴール列> "|" <ボディ・ゴール列> ")"
<ガード・ゴール列>::= <ガード・ゴール> { "," <ガード・ゴール>}
<ガード・ゴール>::= <FGHCガード部組込み述語ゴール>
<ボディ・ゴール列>::= <ボディ・ゴール> { "," <ボディ・ゴール>}
<ボディ・ゴール>::= <FGHCボディ部組込み述語ゴール> |
                    <モジュール名>"@" <FGHCユーザ定義述語ゴール> |
                    <ローカル述語ゴール> |
                    <POOL組込み述語ゴール> |
                    <ESP プログラム呼出し組込み述語ゴール>
```

(5)POOL組込み述語

POOL組込み述語としては，new/2 述語，kill_all/0 述語，send/2述語が提供されている。new/2 述語は，クラス名で指定されるクラスのオブジェクトを作成し，オブジェクト名をその名前とする。kill_all/0 述語は，当該述語の評価時に存在している全てのオブジェクトを削除する。send/2述語は，送信先オブジェクトに対してメソッド述語の評価を依頼する。送信先オブジェクトは，オブジェクト名あるいは"self"により指定される。ここで，"self"は当該メソッド述語の送信元オブジクト自身を指す。すなわち，当該オブジェクト自身に対するメソッド述語の評価が依頼される。メソッド述語ゴールには，組込みメソッド述語ゴールと利用者が定義するメソッド述語ゴールとがある。尚，new/2 述語，kill_all/0 述語，send/2述語と同じ形式のメソッド述語を定義することはできるが，ローカル述語として定義することはできない。

構文

<POOL組込み述語ゴール>::=
　　"new" "(" <クラス名> "," <オブジェクト名> ")" |
　　"kill _all" |
　　"send" "(" <送信先オブジェクト> "," <メソッド述語ゴール> ")"
<送信先オブジェクト>::= <オブジェクト名> | "self"
<メソッド述語ゴール>::= <組込みメソッド述語ゴール> |
　　<利用者定義メソッド述語ゴール>

(6)組込みメソッド述語

組込みメソッド述語としては，kill/0メソッド述語，get/2 メソッド述語，set/2 メソッド述語，set/3 メソッド述語，getandset/3 メソッド述語が提供されている。kill/0メソッド述語は，当該メソッド述語ゴールを受信したオブジェクトを削除する。get/2 メソッド述語は，当該メソッド述語ゴールを受信したオブジェクトのスロット名で指定されるスロットの値を返す。set/2 メソッド述語とset/3 メソッド述語は，当該メソッド述語ゴールを受信したオブジェクトのスロット名で指定されるスロットに値を設定する。また，フラグには，当該メソッド述語の評価が正常終了ならば"ok"が，異常終了ならば"fail"が返される。getandset/3 メソッド述語は，当該メソッド述語ゴールを受信したオブジェクトのスロット名で指定されるスロットの現在の値を返すとともに，新しい値の設定が同時に行われる。たとえ当該メソッド述語の評価時に，第３引数の新スロット値が変数であったとしても，スロットの値と当該変数との間で単一化が行われる。従って，当該スロットに対する別のアクセスによりスロットの値として変数が読出されたとしても，変数に値が与えられる迄，その値を使う述語の評価はFGHCのガード機構により中断されることになる。この組込みメソッド述語は，スロット・アクセスに対する排他制御機能を提供するものである。尚，kill/0メソッド述語，get/2 メソッド述語，set/2 メソッド述語，set/3 メソッド述語，getandset/3 メソッド述語と同じ形式のローカル述語を定義することはできるが，メソッド述語として定義することはできない。

構文

<組込みメソッド述語ゴール>::=
　　"kill" |
　　"get" "(" <スロット名> "," <スロット値> ")" |
　　"set" "(" <スロット名> "," <スロット値> ["," <フラグ>] ")" |
　　"getandset" "(" <スロット名> "," <現スロット値> ","
　　　　<新スロット値> ")"

(7)ESP プログラム呼出し組込み述語

ESP プログラム呼出し組込み述語としては，esp __link@esp__class __call/4述語，esp __link@esp__call/4述語，esp __link@esp__class __waitcall/5述語，esp __link@esp__waitcall/5述語がある。esp __link@esp__class __call/4述語は，ESP のクラス・オブジェクトに対するメソッド述語の呼出しを行う。引数ベクタは，当該メソッドの引数を指定する。ここで引数ベクタの引数は，全て当該メソッド述語に対する入力となっている。また，フラグには，当該メソッド述語の評価が正常終了ならば"ok"が，異常終了ならば"fail"が返される。

esp __link@esp__call/4述語は，ESP のインスタンス・オブジェクトに対するメソッド述語の呼出しを行う。引数ベクタは，当該メソッドの引数を指定する。ここで引数ベクタの引数は，全て当該メソッド述語に対する入力となっている。また，フラグには，当該メソッド述語の評価が正常終了ならば"ok"が，異常終了ならば"fail"が返される。

esp __link@esp__class __waitcall/5述語は，ESP のクラス・オブジェクトに対するメソッド述語の呼出しを行う。引数ベクタは，当該メソッドの引数を指定する。ここで出力変数リストは，引数に含まれる変数の中でESP 側より値が出力される変数を指定する。また，フラグには，当該メソッド述語の評価が正常終了ならば"ok"が，異常終了ならば"fail"が返される。

esp __link@esp__waitcall/5述語は，ESP のインスタンス・オブジェクトに対するメソッド述語の呼出しを行う。引数ベクタは，当該メソッドの引数を指定する。ここで出力変数リストは，引数に含まれる変数の中でESP 側より値が出力される変数を指定する。また，フラグには，当該メソッド述語の評価が正常終了ならば"ok"が，異常終了ならば"fail"が返される。

構文

```
<ESP プログラム呼出し組込み述語ゴール>::=
        "esp_link@esp_class __call" "(" <ESP クラス名> ","
                                        <メソッド名> ","
                                        <引数ベクタ> ","
                                        <フラグ> ")"  |
        "esp_link@esp_call" "(" <ESP オブジェクト> ","
                                <メソッド名> ","
                                <引数ベクタ> ","
                                <フラグ> ")"  |
        "esp_link@esp_class __waitcall" "(" <ESP クラス名> ","
                                            <メソッド名> ","
                                            <引数ベクタ> ","
                                            <出力変数リスト> ","
                                            <フラグ> ")"  |
        "esp_link@esp_waitcall" "(" <ESP オブジェクト> ","
                                    <メソッド名> ","
                                    <引数ベクタ> ","
                                    <出力変数リスト> ","
                                    <フラグ> ")"
<引数ベクタ>::= "(" { <引数> { "," <引数>} } ")"
<出力変数リスト>::= "(" { <出力変数> { "," <出力変数>} } ")"
```

2．2 POOLシステム構成

図1は，POOLシステムの構成概要である．POOLシステムは，POOL翻訳システムとPOOL実行時システムとから構成される．両システムは基本的にはFGHCプログラム（一部ESP プログラムを含む）により実現されており，Pseudo Multi-PSI-V1 システムの配下で実行が行われる．POOL翻訳システムはPOOLソース・プログラムを入力し，FGHCソース・プログラムを出力する．FGHCソース・プログラムは，Pseudo Multi-PSI-V1 システムのMPSIコンパイラによりESP オブジェクト・プログラムにコンパイルされる．POOLプログラムは，Pseudo Multi-PSI-V1 システムおよびPOOL実行時システムの配下で実行される．尚，ここでPseudo Multi-PSI-V1 システムと呼んでいるものには，ESP プログラムの呼出し等のために変更が加えられている．従って，元々のPseudo Multi-PSI-V1 システムとは，厳密には異なっているので注意されたい．

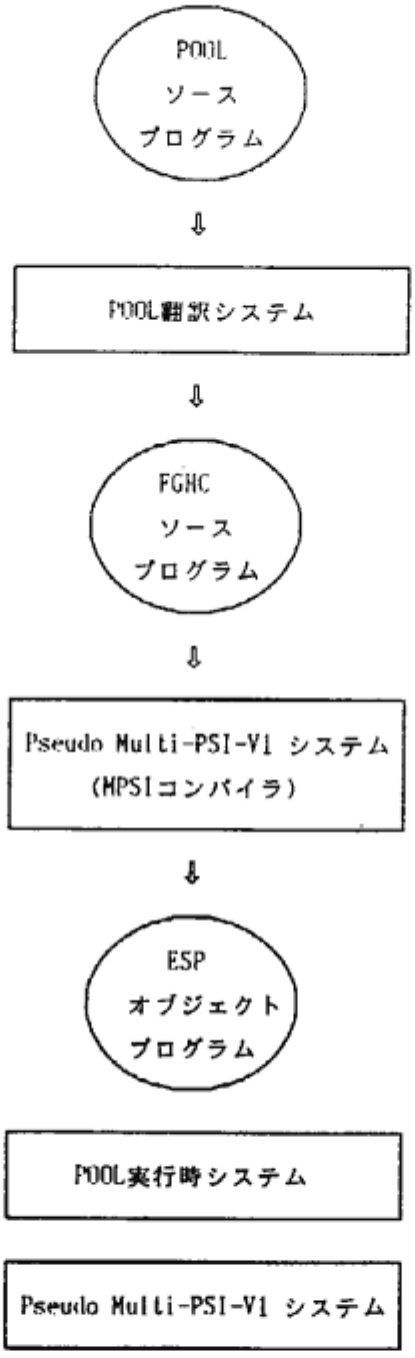


図１　POOLシステム

２．３　POOLシステムの作成

POOLシステムの作成は，Pseudo Multi-PSI V1 システムおよびPOOLシステム自身の作成を含む。ここでPseudo Multi-PSI-V1 システムと呼んでいるものは，「２．２　POOLシステム構成」で述べたように，ESP プログラムの呼出し等のために変更が加えられている。

このため，元々のPseudo Multi-PSI-V1 システムとは，厳密には異なっているので注意されたい．尚，Pseudo Multi-PSI-V1 システムの詳細に関しては，「Pseudo Multi-PSI-V1 システム使用手引き」を参照されたい．

以下では，Pseudo Multi-PSI-V1 システムおよびPOOLシステムの作成について説明する．当該システムは，ソース・プログラムかオブジェクト・コードで提供される予定である．

(1) 提供されたものがソース・プログラムの場合

① システム作成のためにユーザ名を登録する．以降では，ユーザ名を"pool"とする．尚，別のユーザ名を使用する場合には，*.com，MAKE*.COMのファイルの中のパス名の指定を変更することが必要となる．

② ユーザ名"pool"でlogin しなおす．

③ SIMPOSの"file manipulator"でフォント・ファイルをフロッピィから >sys>font >test __11.font にコピーする．

④ SIMPOSの"file manipulator"でコマンド・ファイルをフロッピィからコピーし，次のファイルを作成する．

```
>sys>user>pool>login.com
>sys>user>pool>librarian.com
>sys>user>pool>LOADMPSI.COM
>sys>user>pool>LOADPOOL.COM
>sys>user>pool>SAVEMPSI.COM
>sys>user>pool>SAVEPOOL.COM
>sys>user>pool>MAKEMPSI.COM
>sys>user>pool>MAKEPOOL.COM
```

⑤ SIMPOSの"file manipulator"でソース・プログラムを格納するためのディレクトリィ"pmpsi" と"pool"を作成する．

⑥ SIMPOSの"file manipulator"でPseudo Multi-PSI-V1 システム，POOLシステムのソース・プログラムをそれぞれ>sys>user>pool>pmpsi>*.*，>sys>user>pool>pool>*.* にコピーする．

⑦ SIMPOSの"Librarian" でパッケージ"coordination"を定義する（environment は simpos）．

⑧ login しなおす．

⑨ SIMPOSの"Librarian" で"Execute" を選択し，fxd("MAKEPMPSI.COM")とfxd("MAKEPOOL.COM") を順に実行する．

⑩ システム・メニュで"PseudoMultiPSI-FGHC-V1"を選択し，Pseudo Multi-PSI-V1 システムを起動する．この詳細は，「２．４　POOLシステムの起動」を参照のこと．

⑪ POOLシステムのソース・プログラムの中の全てのFGHCソース・プログラム(*.ghc) をコンパイルする．この詳細は，「２．６　POOLプログラムのコンパイルとセーブ (2)」を参照のこと．

⑫ POOLシステムのソース・プログラムの中の全てのPOOLソース・プログラム(*.ocl

) をコンパイルする。この詳細は，「２．６　POOLプログラムのコンパイルとセーブ」を参照のこと。

⑬ SIMPOSの"Librarian" で"Execute" を選択し，さらに"menu"，"user defined"を選択し，"Save PMPSI"と"Save POOL" を順に実行する。

(2) 提供されたものがオブジェクト・コードの場合

① システム作成のためにユーザ名を登録する。以降では，ユーザ名を"pool"とする。尚，別のユーザ名を使用する場合には，*.com のファイルの中のパス名の指定を変更することが必要となる。

② ユーザ名"pool"でlogin しなおす。

③ SIMPOSの"file manipulator"でフォント・ファイルをフロッピィから >sys>font >test __ll.font にコピーする。

④ SIMPOSの"file manipulator"でコマンド・ファイルをフロッピィからコピーし，次のファイルを作成する。

```
>sys>user>pool>login.com
>sys>user>pool>librarian.com
>sys>user>pool>LOADMPSI.COM
>sys>user>pool>LOADPOOL.COM
>sys>user>pool>SAVEMPSI.COM
>sys>user>pool>SAVEPOOL.COM
```

⑤ SIMPOSの"Librarian" でパッケージ"coordination"を定義する（environment は simpos）。

⑥ login しなおす。

⑦ SIMPOSの"Librarian" で"Utility" を選択し，さらに"Load from Fdd" を選択する。マウスの中ボタンを１回クリックし，"all but selected"を選択し，Pseudo Multi-PSI-V1システムとPOOLシステムのオブジェクト・コードを順にパッケージ"coordination"にロードする。

⑧ SIMPOSの"Librarian" で"Execute" を選択し，さらに"menu"，"user defined"を選択し，"Save PMPSI"と"Save POOL" を順に実行する。

２．４　POOLシステムの起動

POOLシステムの起動は，以下の手順により行われる。

① Pseudo Multi-PSI-V1 システムおよびPOOLシステムのオブジェクト・コードがロードされていなければ，SIMPOSの"Librarian" で"Execute" を選択し，さらに"menu"，"user defined"を選択し，"Load PMPSI"と"Load POOL" を実行する。

② システム・メニュで"PseudoMultiPSI-FGHC-V1"を選択し，Pseudo Multi-PSI-V1 システムを起動する。

③ ディスプレイ上の左上部分に図２のようなstandard I/Oウィンドウが表示されるので，擬似プロセッサ個数を1.と入力する。Pseudo Multi-PSI-V1 システムでは擬似プロセッサ個数として２以上が指定可能であるが，POOLシステムでは擬似プロセッ

サ個数は１だけしか指定できないので注意されたい。

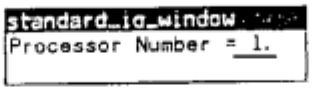


図２　Standard I/Oウィンドウ

④　マウスを使って，
・トップレベルウィンドウ
・PEウィンドウ
・ステイトウィンドウ
・コンパイラウィンドウ
を作る（付録５参照のこと）。

以上でPOOLシステムは，起動された。

２．５　POOLシステムの終了

POOLシステムを終了させるためには，以下の２つの方法がある。

①　SIMPOSの"process manipulator" でプロセス"pmpsi__top __xx" （xxは，プロセス番号）を選択した後，"exterminate" を選択する。

②　図３に示すように，トップレベルウィンドウでhalt. を入力する。

図３　Pseudo Multi-PSI-V1 システムの終了

２．６　POOLプログラムのコンパイルとセーブ

POOLプログラムを動作させるためには，POOLシステムの起動状態で，

・POOLソース・プログラムのFGHCソース・プログラムへのコンパイル

・FGHCソース・プログラムのESP プログラムへのコンパイルとカタログ

を行うことが必要である．以下では，これらの手順について説明する．

(1) POOLソース・プログラムのFGHCソース・プログラムへのコンパイル

POOLソース・プログラムのFGHCソース・プログラムへのコンパイルは，POOLシステムの起動状態で，以下の手順により行われる．

① 図4に示すように，トップレベルウィンドウでゴールodl __translator__top@go(R).を入力し，POOL翻訳システムを起動する．

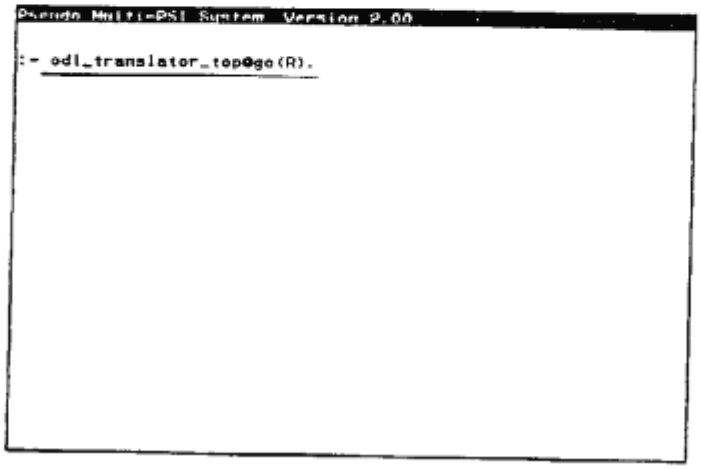


図4　POOL翻訳システムの起動

② マウスを使って，図5に示すPOOL翻訳ウィンドウを作る．

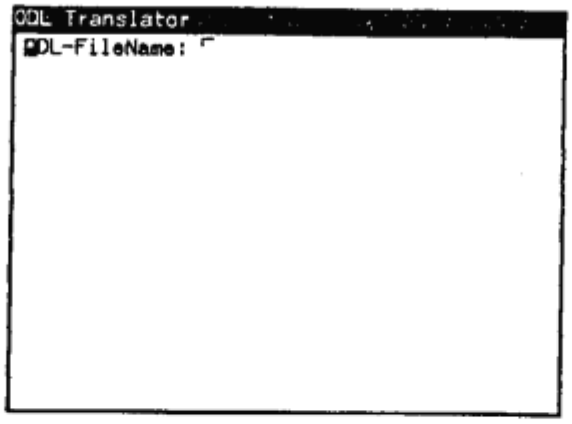


図5　POOL翻訳ウィンドウ

③ 図6に示すように，POOL翻訳ウィンドウにPOOLソース・プログラムの入っているファイルを指定する．この操作により，コンパイルが行われる．ファイル名は*.odlの形式とし，ファイルの指定において下線部は省かれる．同様に，ファイル名は，

パス名あるいは論理ディレクトリ名を用いても指定できる．コンパイル結果としてのGHC ソース・プログラムのファイル名は，*.ghc の形式として作成される．図７はコンパイル終了時のPOOL翻訳ウィンドウを示す．尚，コンパイルに際してはPOOLソース・プログラムの１クラスがFGHCソース・プログラムの１モジュールとして実現され，それらの名前は同じものとなる．

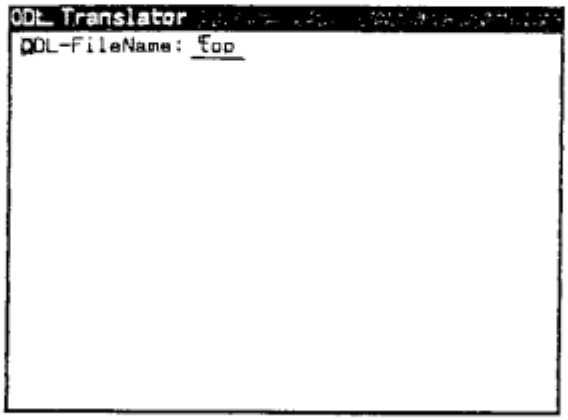


図６　入力ファイルの指定

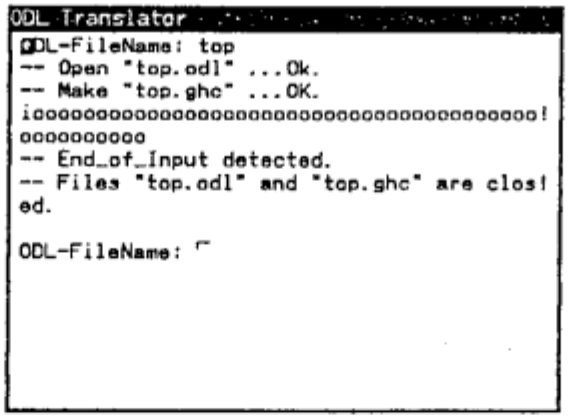


図７　コンパイル終了時のPOOL翻訳ウィンドウ

④　必要であれば，③を繰り返す．

⑤　POOL翻訳システムを終了させるためには，図８に示すようにPOOL翻訳ウィンドウにend を入力する．この時点で，POOL翻訳ウィンドウは消える．

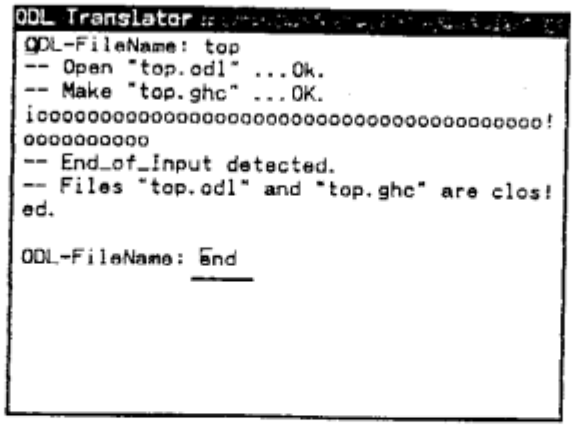


図８　PDOL翻訳システムの終了

(2)　FGHCソース・プログラムのESP プログラムへのコンパイルとカタログ

Pseudo Multi-PSI-V1 システムのMPSIコンパイラは，以下の手順によりFGHCソース・プログラムのESP プログラムへのコンパイルとカタログを行う。

①　図９に示すように，Pseudo Multi-PSI-V1 システムのコンパイラウィンドウでcompile.を入力し，MPSIコンパイラを起動する。

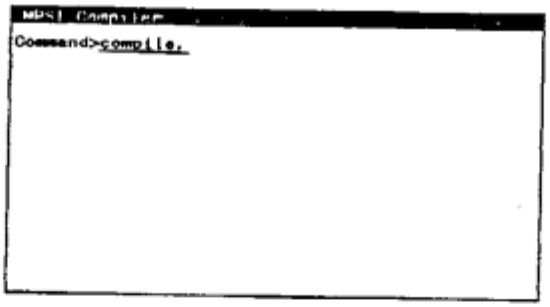


図９　MPSIコンパイラの起動

②　図１０に示すように，Pseudo Multi-PSI-V1 システムのコンパイラウィンドウにFGHCソース・プログラムの入っているファイルを指定する。この操作により，コンパイルが行われる。ファイル名は* .ghcの形式とし，ファイルの指定において下線部は省かれる。同様に，ファイル名は，" ' " でくくられたパス名あるいは論理ディレクトリ名を用いても指定できる。コンパイル結果としてのESP プログラムは，ESP コンパイラによりカタログされる。図１１は，コンパイルおよびカタログ終了時のコンパイラウィンドウを示す。尚，コンパイルに際してはFGHCソース・プログラムの１モジュールがESP の１クラスとして実現され，それらの名前は同じものとなる。

図１０　入力ファイルの指定

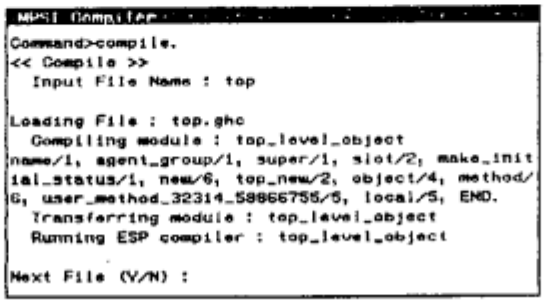


図１１　コンパイルおよびカタログ終了時のコンパイラウィンドウ

③　コンパイラウィンドウにＹを入力すると②に戻る。また，Ｎを入力すると，MPSIコンパイラの実行を終了する。

(3)　セーブ／ロード

POOLソース・プログラムは，上記で述べたようにPOOL翻訳システム，Pseudo Multi-PSI-V1 システムのMPSIコンパイラ，ESP コンパイラを経て，ESP プログラムとしてカタログされる。ここでPOOLのクラス名はESP のクラス名に対応しているので，利用者はSIMPOSの"Librarian" を用いてセーブやロードを行うことができる。

２．７　POOLプログラムの実行

POOLプログラムを動作させるためには，対応するESP のオブジェクト・コードがロードされていることが必要である。この下で，POOLシステムを起動し，トップレベルウィンドウのプロンプト":-"に答えてゴールを入力すれば良い。ゴールの形式は，次の通りである。

＜ゴール＞::= ＜POOLのクラス名＞"@top _new" "("
＜オブジェクト名＞ ","
" {" ＜メソッド述語ゴール＞
{ "," ＜メソッド述語ゴール＞} "} " ")"

図１２は，ゴール入力の例を示す。

図１２　PDOLプログラムのゴールの入力

３．協調型ポートフォリオ選択実験システムESCOPS

協調型ポートフォリオ選択実験システムESCOPS(Experimental System for Cooperative Portfolio Selection)は，並列オブジェクト指向言語POOLシステムの評価用プログラムとして試作された。

３．１　ESCOSPシステムの作成

ESCOPSシステムは，ソース・プログラムかオブジェクト・コードで提供される予定である。尚，ESCOPSシステムの作成に際しては，POOLシステムが既に作成されていることが必要である。

(1)　提供されたものがソース・プログラムの場合

以下の手順でオブジェクト・コードを作成し，セーブする。

① ユーザ名"pool"でlogin する。

② SIMPOSの"file manipulaotr"でフォント・ファイルをフロッピィからコピーし，次のファイルを作成する。

```
>sys>font>ui__font__45.font.1
>sys>font>ui__font__610.font.1
>sys>font>ui__font__816.font.1
>sys>font>ui__font__88.font.1
```

③ SIMPOSの"file manipulator"でコマンド・ファイルをフロッピィからコピーし，次のファイルを作成する。

```
>sys>user>pool>LOADESCOPS.COM
>sys>user>pool>SAVEESCOPS.COM
>sys>user>pool>MAKEESCOPS.COM
```

④ SIMPOSの"file manipulaotr"でソース・プログラムを格納するためのディレクトリィ"escops"を作成する。

⑤ SIMPOSの"file manipulator"でESCOPSシステムのソース・プログラムを>sys>user>pool>escops>*.* にコピーする。

⑥ SIMPOSの"Librarian" で"Execute" を選択し，fxd("MAKEESCOPS.COM") を実行する。

⑦ POOLシステムを起動し，ESCOPSシステムのソース・プログラムの中の全てのPOOLソース・プログラム(*.odl) をコンパイルする。この詳細は，「２．４　POOLシステムの起動」，「２．６　POOLシステムのコンパイルとセーブ」を参照のこと。

⑧ SIMPOSの"Librarian" で"Execute" を選択し，さらに"menu"，"user defined"を選択し，"Save ESCOPS" を実行する。

(2)　提供されたものがオブジェクト・コードの場合

以下の手順でオブジェクト・コードをロードし，セーブする。

① ユーザ名"pool"でlogin する。

② SIMPOSの"file manipulaotr"でフォント・ファイルをフロッピィからコピーし，

次のファイルを作成する。

```
>sys>font>ui__font__45.font.1
>sys>font>ui__font__610.font.1
>sys>font>ui__font__816.font.1
>sys>font>ui__font__88.font.1
```

③ SIMPOSの"file manipulator"でコマンド・ファイルをフロッピィからコピーし，次のファイルを作成する。

```
>sys>user>pool>LOADESCOPS.COM
>sys>user>pool>SAVEESCOPS.COM
```

④ SIMPOSの"Librarian" で"Utility" を選択し，さらに"Load from Fdd" を選択する。マウスの中ボタンを１回クリックし，"all but selected"を選択し，ESCOPSシステムのオブジェクト・コードをパッケージ"coordination"にロードする。

⑤ SIMPOSの"Librarian" で"Execute" を選択し，さらに"menu"，"user defined"を選択し，"Save ESCOPS" を実行する。

３．２　ESCOPSシステムの起動

ESCOPSシステムの起動は，以下の手順により行われる。

① ESCOPSシステムのオブジェクト・コードがロードされていなければ，SIMPOSの"Librarian" で"Execute" を選択し，さらに"menu"，"user defined"を選択シ，"Load ESCOPS" を実行する。
，"user __defined"，"LOAD ESCOPS" と選択する。

② 「２．４　POOLシステムの起動」で述べた手順によりPOOLシステムを起動する。

③ Pseudo Multi-PSI-V1 システムのトップレベルウィンドウのプロンプトに答えて，図１３に示すようにゴール top__level __object@top__new(top, [do(top)]).を入力する。これで，ESCOPSシステムは起動された。

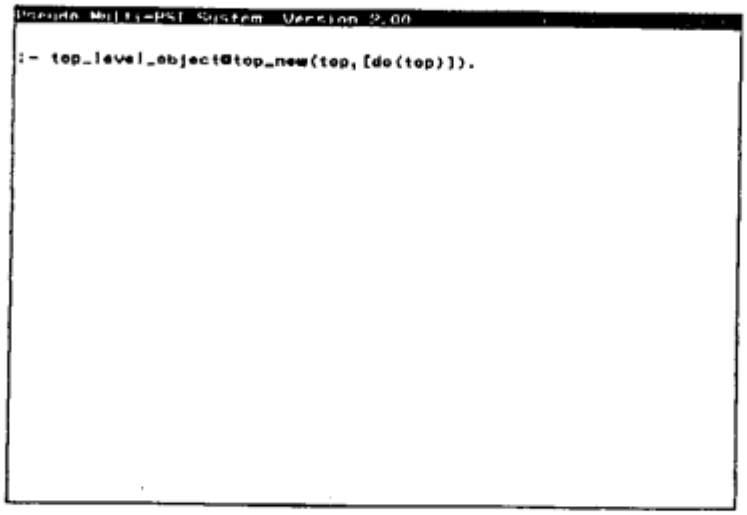


図１３　ESCOPSシステムの起動

３．３　ESCOPSシステムの操作方法

以下では，ESCOPSシステムの操作方法に関して説明する。

(1)トップレベル・コマンドメニュ

ESCOPSを起動すると，ディスプレイ左上部ににトップレベル・コマンドメニュの大きさの矩形枠が表示される。カーソルの移動により，矩形枠の位置は移動する。マウスの左ボタンを１回クリックすると，矩形枠の位置は固定する。図１４は，トップレベル・コマンドメニュである。コマンドには，基準ポートフォリオ計画の①条件入力，②条件表示，③計画実行，運用ポートフォリオ計画の④条件入力，⑤条件表示，⑥計画実行，および⑦終了がある。カーソルをコマンド名の位置にもっていくと，コマンド名が矩形で囲まれる。この状態でマウスの左ボタンを１回クリックすることにより対応するコマンドが選択できる。

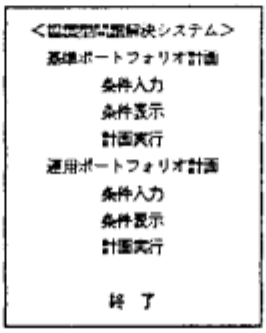


図１４　トップレベル・コマンドメニュ

(2)基準ポートフォリオ計画条件入力

トップレベル・コマンドメニュにおいて，基準ポートフォリオ計画の条件入力コマンドを選択すると，ディスプレイ左上部に条件入力ウィンドウの大きさの矩形枠が表示される。カーソルの移動により，矩形枠の位置は移動する。マウスの左ボタンを１回クリックすると，矩形枠の位置は固定する。図１５は，条件入力ウィンドウである。図１５の条件入力ウィンドウでは，入力の各項目に対する暗黙値が表示されている。

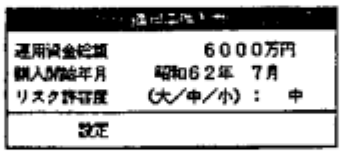


図１５　基準ポートフォリオ計画条件入力ウィンドウ

運用資金総額，購入開始年月の各項目値の表示箇所をマウスの左ボタンで１回クリックすると，項目値の入力可能状態となる。項目値の入力が終了し，改行キーを入力すると，

入力可能状態は終了する．リスク許容度項目値の表示箇所をマウスの左ボタンで１回クリックすると，ポップアップ・ウィンドウが表示される（図１６参照），ポップアップ・ウィンドウ内の「大／中／小」のいずれかをマウスの左ボタンで１回クリックすると，項目値が選択され，ポップアップ・ウィンドウはディスプレイ上から消える．選択された項目値は，条件入力ウィンドウのリスト許容値の値として表示される．条件入力が終了したら，条件入力ウィンドウのボトムに位置する「設定」をマウスの左ボタンで１回クリックすると，条件入力ウィンドウが消える．

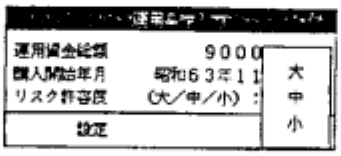


図１６　リスク許容度項目値入力のためのポップアップ・ウィンドウ

条件入力ウィンドウが消えると，つぎに債券選択ウィンドウの位置と大きさの指定が行われる．先ず，カーソルの位置に，債券選択ウィンドウの左上頂点を示す"「"記号が表示される．カーソルの移動により，この記号は移動する．マウスの左ボタンを１回クリックすると，債券選択ウィンドウの左上頂点の位置は固定する．次に，ガーソルの位置に，債券選択ウィンドウの右下頂点を示す"」"記号が表示される．カーソルの移動により，この記号は移動する．マウスの左ボタンを１回クリックすると，債券選択ウィンドウの右下頂点の位置は固定する．図１７は，債券選択ウィンドウである．

銘柄	分類	利付/割引	担保	利金	発行日	満期日	利率	直接利回り	最終利回り	リターン
□13回国債	短期	利付	無	課税	S53/11/20	S63/11/21	6.10%	6.03%	5.00%	7.45%
□16回国債	短期	利付	無	課税	S54/02/19	S64/02/20	6.50%	6.38%	4.97%	7.37%
□19回国債	短期	利付	無	課税	S54/05/20	S64/05/20	7.20%	6.97%	4.98%	7.15%
□21回国債	短期	利付	無	課税	S54/08/20	S64/08/21	7.70%	7.36%	4.98%	6.93%
□25回国債	中期	利付	無	課税	S55/02/20	S65/02/20	8.00%	7.52%	4.96%	6.69%
□28回国債	中期	利付	無	課税	S55/05/20	S65/05/21	8.70%	8.01%	4.93%	6.45%
□31回国債	中期	利付	無	課税	S55/11/20	S65/11/20	8.50%	7.77%	4.97%	6.31%
□34回国債	中期	利付	無	課税	S56/02/20	S66/02/20	8.00%	7.38%	5.06%	6.46%
□36回国債	中期	利付	無	課税	S56/05/20	S66/05/20	7.60%	7.09%	5.20%	6.61%
□41回国債	中期	利付	無	課税	S56/08/20	S66/08/20	8.00%	7.34%	5.21%	6.30%
□43回国債	中期	利付	無	課税	S57/02/20	S67/02/20	7.70%	7.09%	5.28%	6.53%
□45回国債	中期	利付	無	課税	S57/06/20	S67/06/20	7.50%	6.97%	5.46%	6.40%
□46回国債	中期	利付	無	課税	S57/07/20	S67/07/20	7.50%	6.96%	5.46%	6.21%
□49回国債	中期	利付	無	課税	S57/07/20	S67/07/20	8.00%	7.28%	5.40%	6.03%
□53回国債	中期	利付	無	課税	S58/01/20	S68/01/20	7.50%	6.95%	5.57%	6.09%
□57回国債	中期	利付	無	課税	S58/07/20	S68/07/20	7.50%	6.92%	5.57%	5.87%
□59回国債	長期	利付	無	課税	S58/12/20	S68/12/20	7.30%	6.79%	5.64%	5.60%
□63回国債	長期	利付	無	課税	S59/06/20	S69/06/20	7.00%	6.58%	5.69%	5.40%
□64回国債	長期	利付	無	課税	S59/07/20	S69/07/20	7.30%	6.76%	5.66%	5.42%
□68回国債	長期	利付	無	課税	S59/12/20	S69/12/20	6.80%	6.44%	5.79%	5.13%
□75回国債	長期	利付	無	課税	S60/06/20	S70/06/20	6.50%	6.24%	5.77%	4.94%
□78回国債	長期	利付	無	課税	S60/07/20	S70/07/20	6.20%	6.05%	5.77%	4.82%
□87回国債	長期	利付	無	課税	S61/01/20	S71/01/20	5.70%	5.75%	5.84%	4.96%
□89回国債	長期	利付	無	課税	S61/06/20	S71/06/20	5.10%	5.29%	5.67%	4.77%
□91回国債	長期	利付	無	課税	S61/12/20	S71/12/20	5.10%	5.48%	6.22%	4.97%
□99回国債	長期	利付	無	課税	S62/06/20	S72/06/20	4.70%	5.23%	6.29%	5.14%
□101回国	長期	利付	無	課税	S62/06/20	S72/06/20	3.90%	4.65%	6.44%	5.93%

設定

図１７　債券選択ウィンドウ①

債券選択ウィンドウの各行は債券に対応しており，関連する項目が各列に表示される。債券に対応する各行の左端の□をマウスの左ボタンで１回クリックすると，■と表示が反転し，対応する債券が選択されたことになる（図１８参照）。債券選択ウィンドウのボトムに位置する各シンボルをマウスの左ボタンで１回クリックすると，表２の対応するコマンド操作が行われる。

銘柄	方式	利付/割引	担保	特典	発行日	償還日	利率	直接利回り	最終利回り	リターン
■13回国債	短期	利付	無	複利	S53/11/20	S63/11/21	6.10%	6.03%	5.00%	7.45%
□16回国債	短期	利付	無	複利	S54/02/19	S64/02/20	6.50%	6.38%	4.97%	7.37%
□19回国債	短期	利付	無	複利	S54/05/20	S64/05/20	7.20%	6.97%	4.98%	7.15%
■21回国債	短期	利付	無	複利	S54/08/20	S64/08/21	7.70%	7.36%	4.98%	6.93%
□25回国債	中期	利付	無	複利	S55/02/20	S65/02/20	8.00%	7.52%	4.96%	6.69%
■28回国債	中期	利付	無	複利	S55/05/20	S65/05/21	8.70%	8.01%	4.93%	6.45%
□31回国債	中期	利付	無	複利	S55/11/20	S65/11/20	8.50%	7.77%	4.97%	6.31%
□34回国債	中期	利付	無	複利	S56/02/20	S66/02/20	8.00%	7.38%	5.06%	6.46%
□36回国債	中期	利付	無	複利	S56/05/20	S66/05/20	7.60%	7.09%	5.20%	6.61%
■41回国債	中期	利付	無	複利	S56/08/20	S66/08/20	8.00%	7.34%	5.21%	6.30%
■43回国債	中期	利付	無	複利	S57/02/20	S67/02/20	7.70%	7.09%	5.28%	6.53%
□45回国債	中期	利付	無	複利	S57/06/20	S67/06/20	7.50%	6.97%	5.46%	6.40%
□46回国債	中期	利付	無	複利	S57/07/20	S67/07/20	7.50%	6.95%	5.46%	6.21%
□49回国債	中期	利付	無	複利	S57/07/20	S67/07/20	8.00%	7.28%	5.40%	6.03%
□53回国債	中期	利付	無	複利	S58/01/20	S68/01/20	7.50%	6.95%	5.57%	6.09%
□57回国債	中期	利付	無	複利	S58/07/20	S68/07/20	7.50%	6.92%	5.57%	5.87%
□59回国債	長期	利付	無	複利	S58/12/20	S68/12/20	7.30%	6.79%	5.64%	5.60%
□63回国債	長期	利付	無	複利	S59/06/20	S69/06/20	7.00%	6.58%	5.69%	5.40%
■64回国債	長期	利付	無	複利	S59/07/20	S69/07/20	7.30%	6.76%	5.66%	5.42%
□68回国債	長期	利付	無	複利	S59/12/20	S69/12/20	6.80%	6.44%	5.79%	5.13%
□75回国債	長期	利付	無	複利	S60/06/20	S70/06/20	6.50%	6.24%	5.77%	4.94%
■78回国債	長期	利付	無	複利	S60/07/20	S70/07/20	6.20%	6.05%	5.77%	4.82%
□87回国債	長期	利付	無	複利	S61/01/20	S71/01/20	5.70%	5.75%	5.84%	4.96%
■89回国債	長期	利付	無	複利	S61/06/20	S71/06/20	5.10%	5.29%	5.67%	4.77%
□91回国債	長期	利付	無	複利	S61/12/20	S71/12/20	5.10%	5.48%	6.22%	4.97%
□99回国債	長期	利付	無	複利	S62/06/20	S72/06/20	4.70%	5.23%	6.29%	5.14%
■101回国	長期	利付	無	複利	S62/06/20	S72/06/20	3.90%	4.65%	6.44%	5.93%

訂正 キャンセル ヘルプ

図１８　債券選択ウィンドウ②

表２　債券選択ウィンドウのサブ・コマンド

シンボル	コマンド操作
設定	債券選択ウィンドウが消える。
↑	表示項目を上方向に１行分スクロールする。
↓	表示項目を下方向に１行分スクロールする。
←	表示項目を左方向に１列分スクロールする。
→	表示項目を右方向に１列分スクロールする。
⇑	表示項目を上方向に縦方向のサイズ分スクロールする。
⇓	表示項目を下方向に縦方向のサイズ分スクロールする。
⇐	表示項目を左方向に横方向のサイズ分スクロールする。
⇒	表示項目を右方向に横方向のサイズ分スクロールする。

(3)基準ポートフォリオ計画条件表示

トップレベル・コマンドメニュにおいて，基準ポートフォリオ計画の条件表示コマンドを選択すると，ディスプレイ上に条件表示ウィンドウの大きさの矩形枠が表示される。カーソルの移動により，矩形枠の位置は移動する。マウスの左ボタンを１回クリックすると，矩形枠の位置は固定する。図１９は条件表示ウィンドウであり，当該時点において指定されている各項目の値が表示されている。条件表示ウィンドウのボトムに位置する「消去」をマウスの左ボタンで１回クリックすると，条件表示ウィンドウが消える。

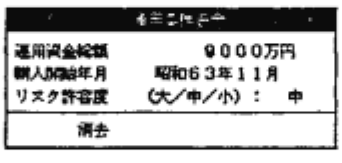


図１９　基準ポートフォリオ計画条件表示ウィンドウ

条件表示ウィンドウが消えると，選択債券表示ウィンドウの大きさの矩形枠が表示され

る。カーソルの移動により，矩形枠の位置は移動する。マウスの左ボタンを１回クリックすると，矩形枠の位置は固定する。図２０は，選択債券表示ウィンドウである。選択債券表示ウィンドウの各行は当該時点において選択されている債券に対応しており，関連する項目が各列に表示される。選択債券表示ウィンドウのボトムに位置する各シンボルをマウスの左ボタンで１回クリックすると，表３の対応するコマンド操作が行われる。

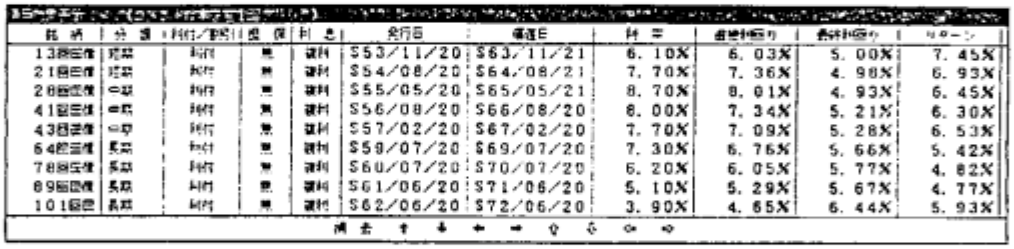


図２０　選択債券表示ウィンドウ

表３　選択債券表示ウィンドウのサブ・コマンド

シンボル	コマンド操作
消去	債券選択ウィンドウが消える。
↑	表示項目を上方向に１行分スクロールする。
↓	表示項目を下方向に１行分スクロールする。
←	表示項目を左方向に１列分スクロールする。
→	表示項目を右方向に１列分スクロールする。
⇑	表示項目を上方向に縦方向のサイズ分スクロールする。
⇓	表示項目を下方向に縦方向のサイズ分スクロールする。
⇦	表示項目を左方向に横方向のサイズ分スクロールする。
⇨	表示項目を右方向に横方向のサイズ分スクロールする。

(4)基準ポートフォリオ計画計画実行

トップレベル・コマンドメニュにおいて，基準ポートフォリオ計画の計画実行コマンドを選択すると，ディスプレイ上に計画実行ウィンドウが表示される。計画実行ウィンドウには，基準ポートフォリオ計画における実行時情報が出力される。図２１は，計画実行ウィンドウのスナップショットである。当該ウィンドウの左部分は，投資対象となる各債券に対応する９個の区画に分割されている。債券数が９個以下の場合には各債券に区画が割当てられ，残った区画は使用されない。債券数が１０個以上の場合には９個の債券に区画が割当てられ，残りの債券には区画は割当てられない。各区画は，上部のメッセージ表示部と下部の項目値表示部に分かれる。メッセージ表示部には，各債券に関する実行時のメッセージ情報が出力される。項目値表示部には，上から順に債券番号，当該時点における投資割当額，リターン値，リスク値，および投資額の割当配分を変更した債券の番号が出力される。計画実行ウィンドウの右上部分には，リターン軸とリスク軸とから成る座標面上に（対応する債券番号による）各債券および（"*" 記号による）ポートフォリオの座標点が示されている。計画実行ウィンドウの右下部分には，当該時点における各債券に対する投資割当額の内容が棒グラフにより表示されている。

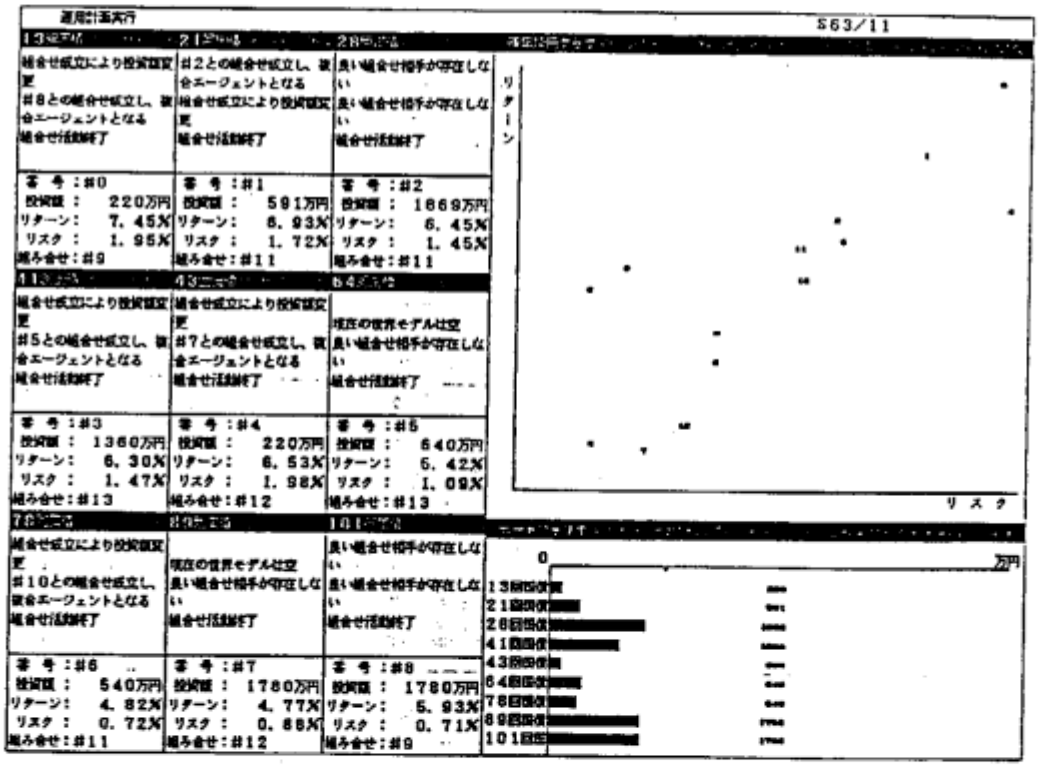


図２１　基準ポートフォリオ計画のスナップショット

(5)運用ポートフォリオ計画条件入力

トップレベル・コマンドメニュにおいて，運用ポートフォリオ計画の条件入力コマンドを選択すると，ディスプレイ左上部に条件入力ウィンドウの大きさの矩形枠が表示される。

カーソルの移動により，矩形枠の位置は移動する。マウスの左ボタンを１回クリックすると，矩形枠の位置は固定する。図２２は，条件入力ウィンドウである。また，表４は条件入力項目を示す。表において，〔＊１〕が付された項目は，表示箇所をマウスの左ボタンで１回クリックすると，ポップアップ・ウィンドウが表示される（図２３参照）。ポップアップ・ウィンドウ内の「活性／非活性」のいずれかをマウスの左ボタンで１回クリックすると，項目値が選択され，ポップアップ・ウィンドウはディスプレイ上から消える。選択された項目値は，条件入力ウィンドウの表示箇所に示される。〔＊２〕が付された項目に関しては，表示箇所をマウスの左ボタンで１回クリックすると，項目値の入力可能状態となる。項目値の入力が終了し，改行キーを入力すると，入力可能状態は終了する。
条件入力が終了したら，条件入力ウィンドウのボトムに位置する「設定」をマウスの左ボタンで１回クリックすると，条件入力ウィンドウが消える。

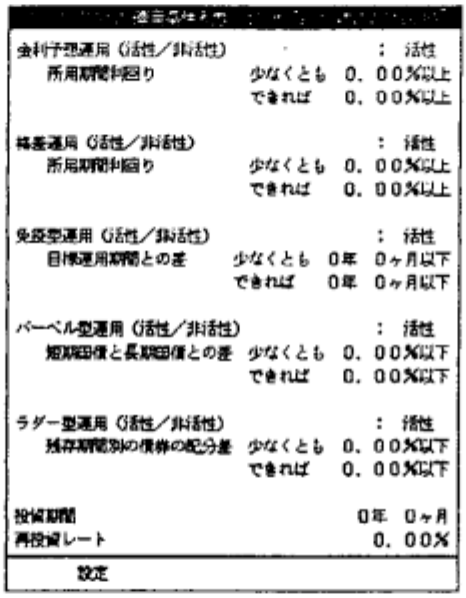


図２２　運用ポートフォリオ計画の条件入力ウィンドウ①

表4　条件入力項目

条件入力項目	項目値の説明
＿＿＿＿運用 （活性／非活性） 〔＊1〕	暗黙値は，「活性」となっている。 活性：　当該運用手法をポートフォリオ計画に参加させる。 非活性：当該運用手法をポートフォリオ計画に参加させない。
＿＿＿＿運用 少なくとも＿＿＿＿ できれば　＿＿＿＿ 〔＊2〕	ポートフォリオ計画における各運用手法の目標の許容範囲を与える。 金利予想運用：投資期間における所有期間利回りに関する。 格差運用：投資期間における所有期間利回りに関する。 免疫型運用：運用期間目標からの乖離き程度に関する。 バーベル型運用：短期債と長期債に対する投資配分のバランスに関する。 ラダー型運用：残存期間別の投資配分のバランスに関する。
投資期間〔＊2〕	金利予想，格差運用におけるポートフォリオの運用期間を示す。 現在のESCOPSで指定できる値は，3ケ月，6ケ月，9ケ月，12ケ月のみに限定されているので注意を要する。
再投資レート 〔＊2〕	投資期間の途中で生ずるクーポン収入や元本を再投資する際の利率を示す。

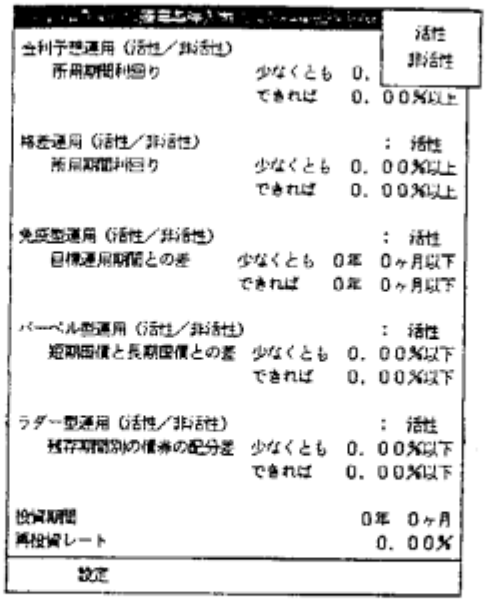


図２３　運用ポートフォリオ計画の条件入力ウィンドウ②

(6)運用ポートフォリオ計画条件表示

トップレベル・コマンドメニュにおいて，運用ポートフォリオ計画の条件表示コマンドを選択すると，ディスプレイ上に条件表示ウィンドウの大きさの矩形枠が表示される。カーソルの移動により，矩形枠の位置は移動する。マウスの左ボタンを１回クリックすると，矩形枠の位置は固定する。図２４は条件表示ウィンドウであり，当該時点において指定されている各項目の値が表示されている。条件表示ウィンドウのボトムに位置する「消去」をマウスの左ボタンで１回クリックすると，条件表示ウィンドウが消える。

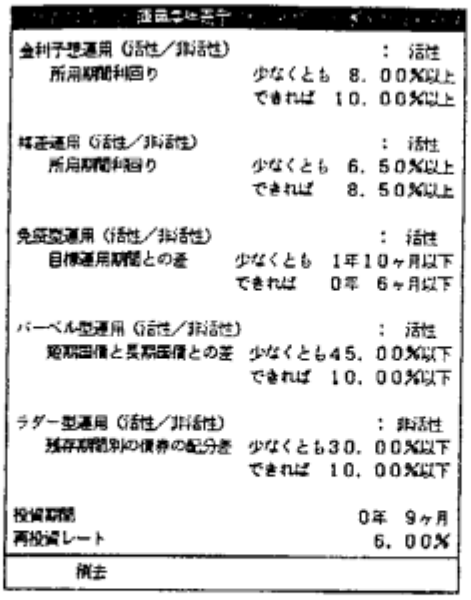


図２４　運用ポートフォリオ計画の条件表示ウィンドウ

(7)運用ポートフォリオ計画計画実行

トップレベル・コマンドメニュにおいて，運用ポートフォリオ計画の計画実行コマンドを選択すると，ディスプレイ上に計画実行ウィンドウが表示される。計画実行ウィンドウには，運用ポートフォリオ計画における実行時情報が出力される。図２５は，計画実行ウィンドウのスナップショットである。当該ウィンドウは，６個の区画に分割されている。右下の区画には，当該時点における各債券に対する投資割当額の内容が棒グラフにより表示さている。残りの５個の区画は各々が各運用手法に対応しており，上部のメッセージ表示部と下部の項目値表示部に分かれる。メッセージ表示部には，各運用手法に関する実行時のメッセージ情報が出力される。項目値表示部には，表５に示す項目値が出力される。

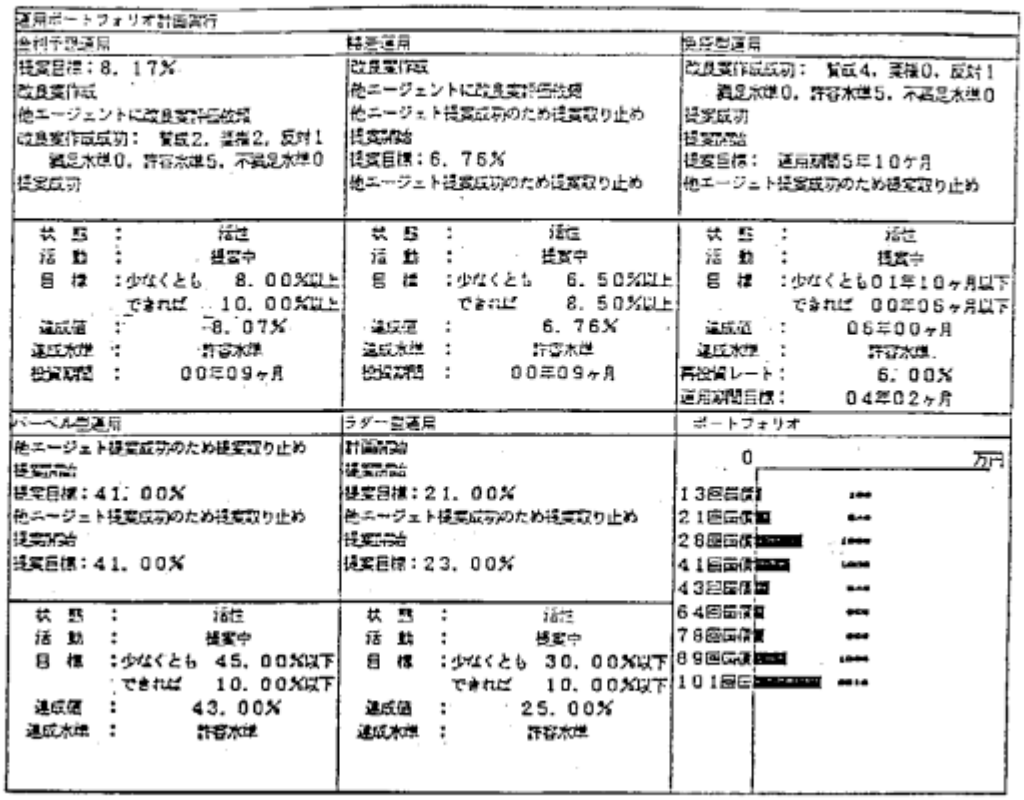


図２５　運用ポートフォリオ計画のスナップショット

表5　運用ポートフォリオ計画における出力項目

項目	項目値の説明
状態	活性：　当該運用手法がポートフォリオ計画に参加している。 非活性：当該運用手法がポートフォリオ計画に参加していない。
活動	提案中：当該時点においてポートフォリオ案を作成中である。 休止：　当該時点においてポートフォリオ案を作成していない。
目標	当該運用手法のポートフォリオ計画に対する目標の許容範囲を示す。
達成値	当該時点におけるポートフォリオに関して達成されている目標の指標値を示す。
達成水準	満足水準：　当該時点におけるポートフォリオにおいて，目標の許容範囲の上限以上を達成している。 許容水準：　当該時点におけるポートフォリオにおいて，目標の許容範囲を満足している。 不満足水準：当該時点におけるポートフォリオにおいて，目標の許容範囲の下限を達成していない。
投資期間（金利予想，格差運用）	当該運用手法を適用する上でのポートフォリオの運用期間を示す。
再投資レート（免疫型運用）	投資期間の途中で生ずるクーポン収入や元本を再投資する際の利率を示す。
運用期間目標（免疫型運用）	当該運用手法で運用する際の目標となるポートフォリオの平均残存期間を示す。

(8)終了

トップレベル・コマンドメニュにおいて，終了コマンドを選択すると，ESCOPSの実行を終了する。

＜参考文献＞

〔杉野　88〕杉野他，Pseudo Multi-PSI-V1 システム使用手引き

〔宮崎　86〕宮崎，FGHCシステム使用手引き

〔Chikayama 84〕Chikayama, T., ESP Reference Manual, ICOT TR-44(1984)

〔Ueda　86〕Ueda, K., Guarded Horn Clause: Parallel Logic Programming Language with the Concept of a Guard, ICOT TR-208(1986)

〔野村総合研究所　81〕野村総合研究所編，債券運用と投資戦略（社団法人金融財政事情研究会，東京，1981年）

〔市来　88〕市来他，協調問題解決システム実現の為の並列オブジェクト指向言語POOL，昭和63年電子情報通信学会秋季全国大会

〔末永　88〕末永他，基準ポートフォリオ選択のための協調問題解決，昭和63年電子情報通信学会秋季全国大会

〔佐藤　88〕佐藤他，運用ポートフォリオ選択のための協調問題解決，昭和63年電子情報通信学会秋季全国大会

<付録１>　POOL構文記述

以下の構文の記述には，次の点でBNF を拡張した拡張BNF を用いる。

・"X" は，終端記号Xを表す。

・〔X〕は，Xの０回以上の任意数の繰返しを表す。

・〔X〕は，Xの出現が選択的であることを表す。

```
<プログラム>::= <クラス定義>　〔　<クラス定義>　〕
<クラス定義>::=
        "class" "(" "name" "("　<クラス名> ")" ","
                    "super" "("　"〔" 〔<上位クラス定義列>〕 "〕" ")"　","
                    "slot" "(" "〔" 〔<スロット定義列>〕 "〕" ")" ","
                    "method" "(" "〔" 〔<メソッド述語定義列>〕 "〕" ")" ","
                    "local" "("　"〔" 〔<ローカル述語定義列>〕 "〕" ")" "."
<上位クラス定義列>::= <クラス名>〔 "," <クラス名> 〕
<スロット定義列>::= <スロット定義>〔 "," <スロット定義> 〕
<スロット定義>::= <スロット名>|
                    "(" <スロット名> "," <初期値> ")"
<メソッド述語定義列>::= <述語定義>〔 "," <述語定義> 〕
<ローカル述語定義列>::= <述語定義>〔 "," <述語定義> 〕
<述語定義>::= <節定義>| "〔"<節定義>〔 "," <節定義> 〕 "〕"
<節定義>::= "(" <ヘッド> ":-"
                <ガード・ゴール列> "|" <ボディ・ゴール列> ")"
<ガード・ゴール列>::= <ガード・ゴール>〔 "," <ガード・ゴール>〕
<ガード・ゴール>::= <FGHCガード部組込み述語ゴール>
<ボディ・ゴール列>::= <ボディ・ゴール>〔 "," <ボディ・ゴール>〕
<ボディ・ゴール>::= <FGHCボディ部組込み述語ゴール>|
                    <モジュール名>"@" <FGHCユーザ定義述語ゴール>|
                    <ローカル述語ゴール>|
                    <POOL組込み述語ゴール>|
                    <ESP プログラム呼出し組込み述語ゴール>
<POOL組込み述語ゴール>::=
        "new" "(" <クラス名> "," <オブジェクト名> ")"|
        "kill_all"|
        "send" "(" <送信先オブジェクト> "," <メソッド述語ゴール> ")"
<送信先オブジェクト>::= <オブジェクト名>　|　"self"
<メソッド述語ゴール>::= <組込みメソッド述語ゴール>|
                        <利用者定義メソッド述語ゴール>
<組込みメソッド述語ゴール>::=
        "kill"|
```

```
        "set" "(" <スロット名> "," <スロット値> ")" |
        "set" "(" <スロット名> "," <スロット値> [ "," <フラグ>] ")" |
        "getandset" "(" <スロット名> "," <現スロット値> ","
                                        <新スロット値> ")"
<ESP プログラム呼出し組込み述語ゴール>::=
        "esp_link@esp_class _call" "(" <ESP クラス名> ","
                                       <メソッド名> ","
                                       <引数ベクタ> ","
                                       <フラグ> ")" |
        "esp_link@esp_call" "(" <ESP オブジェクト> ","
                                <メソッド名> ","
                                <引数ベクタ> ","
                                <フラグ> ")" |
        "esp_link@esp_class _waitcall" "(" <ESP クラス名> ","
                                           <メソッド名> ","
                                           <引数ベクタ> ","
                                           <出力変数リスト> ","
                                           <フラグ> ")" |
        "esp_link@esp_waitcall" "(" <ESP オブジェクト> ","
                                    <メソッド名> ","
                                    <引数ベクタ> ","
                                    <出力変数リスト> ","
                                    <フラグ> ")"
<引数ベクタ>::= "(" [ <引数> { "," <引数> } ] ")"
<出力変数リスト>::= "[" [ <出力変数> { "," <出力変数> } ] "]"
```

<付録２> POOLサンプル・コーディング

```
class(name(goala),
      super([goal]),
      slot([]),
      method([
        [(rank(Estimate,Rank):-
            true
            |
            send(self,get(max_goal,Max)),
            send(self,get(min_goal,Min)),
            rank0(Estimate,Max,Min,Rank))],
        [(estimate(Estimate1,Estimate2,Approval):-
            Estimate1<Estimate2
            |
            Approval=approve),
         (estimate(Estimate1,Estimate2,Approval):-
            Estimate1>Estimate2
            |
            Approval=reject),
         (estimate(Estimate,Estimate,Approval):-
            true
            |
            Approval=draw)],
        [(goal_up(Rank,Estimate,Goal):-
            true
            |
            send(self,get(min_goal,Min)),
            send(self,get(increment,Increment)),
            goal_up0(Rank,Min,Increment,Estimate,Goal))]]),
      local([
        [(rank0(Estimate,Max,Min,Rank):-
            Estimate>=Max
            |
            Rank=maximum),
         (rank0(Estimate,Max,Min,Rank):-
            Estimate<Min
            |
            Rank=minimum),
         (rank0(Estimate,Max,Min,Rank):-
            Estimate<Max,
            Estimate>=Min
            |
            Rank=mid)],
        [(goal_up0(Rank,Min,Increment,Estimate,Goal):-
            Rank¥=minimum
            |
            Goal:=Estimate+Increment),
         (goal_up0(minimum,Min,Increment,Estimate,Goal):-
            true
            |
            Estimate1:=Estimate+Increment,
            max(Estimate1,Min,Goal))],
        [(max(X,Y,Z):-
            X>=Y
            |
            Z=X),
         (max(X,Y,Z):-
            X<Y
            |
            Z=Y)]])).
```

＜付録３＞　POOL翻訳システムのエラー・メッセージ

以下に，POOL翻訳システムがPOOLソース・プログラムのコンパイル時にPOOL翻訳ウィンドウに出力するエラー・メッセージを示す。

① --Open <ファイル名>... ERROR: Open failed!!
POOLソース・プログラムの入った入力ファイル(*.odl) のopen失敗。

② --Make <ファイル名>... ERROR: Make failed!!
出力ファイル(*.odl) の作成失敗。

③ >>ERROR: Input-term error!!
POOLソース・プログラム中のクラス定義の読込み失敗。

④ >>ERROR: Outptu-term failed!!
出力ファイルにデータを書込んだ際にエラー発生。

⑤ >>ERROR: File-close failed!!
入力ファイルあるいは出力ファイルのclose 失敗。

⑥ >>TRANS-ERROR: top-syntax(X)
読込んだクラス定義のterm形式がおかしい。

⑦ >>TRANS-ERROR: name(N,zzzz)
クラス定義におけるクラス名がアトムでない。以降，クラス名をzzzzとして処理を行う。

⑧ >>TRANS-ERROR: super__list(SP)
クラス定義における上位クラス定義列SPのリスト形式がおかしい。

⑨ >>TRANS-ERROR: super__name(S)
クラス定義における上位クラス定義列のクラス名S がアトムでない。

⑩ >>TRANS-ERROR: slots__def(SL)
クラス定義におけるスロット定義列SLのリスト形式がおかしい。

⑪ >>TRANS-ERROR: slot __1 __def(S)
クラス定義におけるスロット定義列のスロット定義S の形式がおかしい。

⑫ >>TRANS-ERROR: predicate__defs(method,L)
クラス定義におけるメソッド述語定義列L のリスト形式がおかしい。

⑬ >>TRANS-ERROR: predicate__defs(local,L)
クラス定義におけるローカル述語定義列L のリスト形式がおかしい。

＜付録４＞　POOL実行時システムのエラー・メッセージ

以下に，POOLプログラムの実行時に，POOL実行時システムがPseudo Multi-PSI-V1 システムのPEウィンドウの上部の実行ウィンドウに出力するエラー・メッセージを示す．尚，エラー・メッセージの出力形式は，次のようである．

＜エラー・メッセージ＞ <CR>
>>ODLSYSERROR:

① make_super _classes _initial _status_failure(SuperName)
上位クラスSuperName のスロット継承に失敗．

② message _is_sent_to_already _closed_stream(Mess)
メッセージ・ストリームが既に閉じられているオブジェクトに対して，メッセージMessを送ろうとした．

③ message _is_sent_to_non _stream(Mess,N)
メッセージ・ストリーム以外のデータN にメッセージMessを送ろうとした．

④ stream_tail_close _failure1(Str)
メッセージ・ストリームStr の終端を閉じるのに失敗した．

⑤ stream_tail_close _failure2(Str)
メッセージ・ストリームStr の終端がおかしい．

⑥ set _to_nonexist_slot(SlotName,Value,Ins)
オブジェクトIns の存在しないスロットSlotNameに値Value を格納しようとした．

⑦ getandset _to_nonexistent _slot(SlotName,Vget,Vset,Ins)
オブジェクトIns の存在しないスロットSlotNameに値Vgetを取り，値Vsetを格納しようとした．

⑧ getting _from_nonexist_slot(Slot,Var,Ins)
オブジェクトIns の存在しないスロットSlotNameの値を変数Var に返そうとした．

⑨ getvalue_return_unification _failure(Slot,Var,Value,Ins)
オブジェクトIns のスロットSlotから取り出した値Value と変数Var との単一化に失敗．

⑩ getandsetvalue_return_unification _failure(Slot,Var,Vset,Value,Ins)
オブジェクトIns のスロットSlotから取り出した値Value と変数Var との単一化に失敗した．

⑪ nomethod(N)
メソッド述語ゴールN を評価すべき節がない．対応するメソッド述語の定義がないか，ガード部が成功する節が存在しない．

⑫ do_body_fail(C,B)
メソッド述語ゴールB に対して，ガード部が成功したクラスC に定義されている節のボディ部の評価が失敗した．尚，B はPOOLソース・プログラム中の形式からPOOL翻訳システムにより次の形式にコンパイルされているので注意されたい．

＜クラス名＞"@user_method_ddddd _tttttttt" "("
＜メソッド述語ゴール＞ "," A "," B "," C "," D ")"

⑬ no_remove_idic_entry(Obj,Ins)

オブジェクトIns から存在していないオブジェクトObj の削除命令が出された。

⑭ no_clear _obj(Obj)

オブジェクトObj の削除命令に失敗した。

⑮ send_to_nonexist_obj _lg(Obj,Mess,Ins)

オブジェクトIns から存在していないオブジェクトObj にメッセージMessが送られた。

⑯ send_to_nonexist_obj _g(Obj,Mess)

存在していないオブジェクトObj にメッセージMessが送られた。

⑰ killed_gdic(Mess)

全てのオブジェクトが削除された後に，メッセージMessが送られた。

⑱ no_getandset _gdic_entry(Obj)

存在していないオブジェクトObj に対して，メッセージを送ろうとした。

⑲ no_remove_gdic_entry(Obj)

存在していないオブジェクトObj に対して，削除操作を行おうとした。

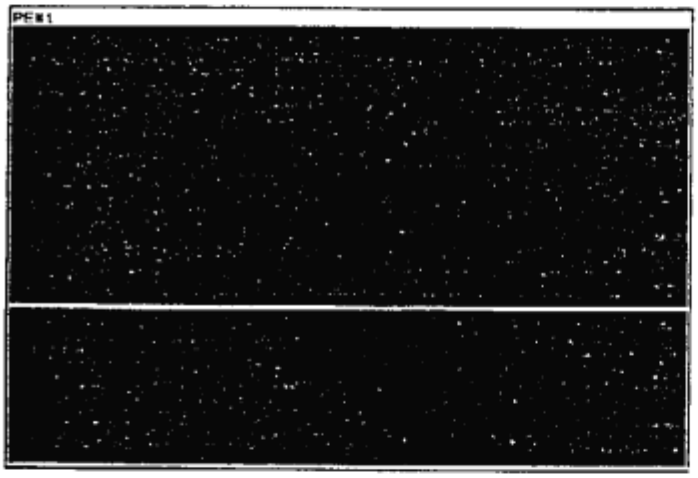
⑳ no_getcopy _gdic_entry(Obj,SendObj)

存在していないオブジェクトObj に対して，メッセージを送ろうとした。

＜付録５＞　Pseudo Multi-PSI-V1 システムのウィンドウ

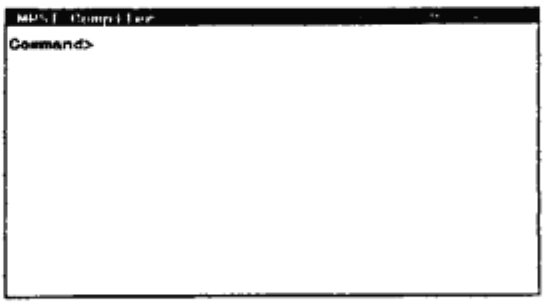
①　トップレベルウィンドウ

②　PEウィンドウ



③　ステイトウィンドウ

④　コンパイラウィンドウ



＜付録６＞　POOLシステムおよびESCOPSシステム関連のフロッピィ一覧

以下は，POOLシステムおよびESCOPSシステム関連のフロッピィである。システムの提供は，これら全てのフロッピィあるいは一部のフロッピィにより行われる予定である。

(1) Command Procedure ファイル，Fontファイル

以下のファイルが，格納されている。

① Command Procedure

login.com
librarian.com
LOADPMPSI.COM
LOADPOOL.COM
LOADESCOPS.COM
SAVEPMPSI.COM
SAVEPOOL.COM
SAVEESCOPS.COM
MAKEPMPSI.COM
MAKEPOOL.COM
MAKEESCOPS.COM

② Fontファイル

test_11.font　　(Pseudo Multi PSI-V1 システム用)
ui_font_45 font　(ESCOPSシステム用)
ui_font_610.font　(ESCOPSシステム用)
ui_font_816.font　(ESCOPSシステム用)
ui_font_88.font　(ESCOPSシステム用)

(2) Pseudo Multi-PSI-V1 システム　ソース・プログラム

(3) Pseudo Multi-PSI-V1 システム　オブジェクト・コード①

(4) Pseudo Multi-PSI-V1 システム　オブジェクト・コード②

(5) POOLシステム　ソース・プログラム

(6) POOLシステム　オブジェクト・コード

(7) ESCOPSシステム　ソース・プログラム(*.odl)

(8) ESCOPSシステム　ソース・プログラム(*.ESP)

(9) ESCOPSシステム　オブジェクト・コード①

(10) ESCOPSシステム　オブジェクト・コード②

＜付録７＞　login.com ファイルのリスト

```
fkpsi211::>fd0>login.com

% fkpsi211::>sys>user>pool>login.com.6, 17-Oct-88 14:45:15, Edit by pool
    % login file

menu :-
     items_list(
        [ [ {"debugger",debugger} ,
            {"pmacs",pmacs} ,
            {"file",file_manipulator} ,
            {"librarian",librarian} ,
            {"shell",shell} ,
            {"spooler",spooler_manipulator} ,
            {"pattern editor",pattern_editor} ,
            {"PseudoMultiPSI-FGHC-V1",coordination#pmpsi_top} ] ]  ).

define :-
         "me" := [ "user:pool" ].

package :- coordination .
```

＜付録８＞　オブジェクト・コードのロード／セーブ

Pseudo Multi-PSI-V1 システム，POOLシステム，ESCOPSシステムのオブジェクト・コードのロード／セーブのために，＜付録６＞に示した(1)「Command Procedure ファイル，Fontファイル」フロッピィに格納されている"librarian.com" ファイルと"*.COM" ファイルを自分のディレクトリにコピーすることにより使用できる。但し，"librarian.com" ファイル中の"*.COM" ファイルのパス名の指定を適宜変更することが必要となる。

```
fkpsi211::>fd0>librarian.com

[
 {"Load PMPSI", fxd(">sys>user>pool>LOADPMPSI.COM")},
 {"Load POOL", fxd(">sys>user>pool>LOADPOOL.COM")},
 {"Load ESCOPS", fxd(">sys>user>pool>LOADESCOPS.COM")},
 {"Save PMPSI", fxd(">sys>user>pool>SAVEPMPSI.COM")},
 {"Save POOL", fxd(">sys>user>pool>SAVEPOOL.COM")},
 {"Save ESCOPS", fxd(">sys>user>pool>SAVEESCOPS.COM")}
]
```

＜付録９＞　ESP ソース・プログラムのカタログ

Pseudo Multi-PSI-V1 システム，POOLシステム，ESCOPSシステムの中のESP ソース・プログラムのカタログのために，＜付録６＞に示した(1)「Command Procedure ファイル，Fontファイル」フロッピィに格納されている"MAKE*.COM" ファイルを自分のディレクトリにコピーすることにより使用できる。但し，"MAKE*.COM" ファイル中の各ESP ソース・プログラムが格納されているファイルのパス名の指定を適宜変更することが必要となる。