U.D.C. 800.92:681.323.014.06

制御用トップダウン・ストラクチャードプログラミング言語—SPL—

Top Down Structured Programming Language for Industrial Computer Systems —SPL—

林　利弘*　*Hayashi Toshihiro*
野木兼六**　*Nogi Kenroku*
中田育男***　*Nakata Ikuo*
浜田亘曼****　*Hamada Nobuhiro*

標準化及び保守のしやすい制御用リアルタイムアプリケーションソフトウェアを，高生産性，高信頼性で作成でき，かつ開発管理をサポートできる新高級言語ＳＰＬ（Software Production Language）を開発した。

ＳＰＬは特に最近のソフトウェア工学分野での成果を積極的にユーザーが享受できるようにするために，ブロック構造，構造化文，マクロ機能（抽象データ，抽象処理の機能），自然語（英語，日本語）風マクロ表現，コンパイル時実行機能をサポートし，トップダウンストラクチャードプログラミング，モジュラープログラミング，標準化，問題向き言語化を極めて容易にしている。また，各プログラムは論理的にも物理的にもデータ独立に記述でき，各データはシステムで一元的に管理できるようになっている。

なおＳＰＬを用いた場合の高生産性，高信頼性及び高保守性は既に実システムへの適用を通して確認されている。

1 緒　言

制御用をはじめとするリアルタイムアプリケーションソフトウェアシステムを取り巻く環境は，近年，作成ソフトウェア量の飛躍的増大，単位ソフトウェア量当たりのソフトウェア人口の減少及び短期間でのシステム立上げ，といった形で大きく変化しており，これに対処するため日立製作所では昭和47年にFORTRANをベースに，ビット処理，テーブル処理，タスキング処理などの機能を強化した制御用高級言語ＰＣＬ（Process Control Language）を開発し，以来，今日まで数多くのシステムに適用し大きな効果を挙げてきた。しかし，この適用を通して，

(1)　リアルタイムシステムの建設では従来のように，個々のプログラムのアルゴリズム記述の高級化，簡易化だけでは不十分で，多数のプログラマが作る個々のプログラムをトータルシステム化していく過程をうまく管理，制御していく機能がプログラミングシステムの中にあることが非常に重要で，特に言語システムにそのための各種の機能が備わっていることが必須である。

ということが痛感され，また，一方では，

(2)　リアルタイムシステムは導入後も成長のための保守業務が必要で，これが正確かつ簡単に行なえること。

(3)　制御用リアルタイム計算機の応用も成熟期に入り，過去の経験を生かして，プログラムの標準化，問題向き言語化を行ない，システムの開発，保守費の低減が図れること。

といった新しい要求が切実なものとしてでてきた。

そして，ソフトウェア技術面ではこの数年の間に世界的にも，トップダウン開発技法，モジュール化技法，プロジェクト管理技法といったソフトウェアの信頼性，生産性，及び保守性を高めるための各種ソフトウェア工学的技法に大きな進展があった。

以上のような最近の新しい状況を踏まえ，リアルタイムシステムのライフサイクル全体にわたってソフトウェアの開発と保守が効果的に行なえるプログラミング言語として開発したのがSoftware Production Language（以下，ＳＰＬと略す）であり，以下にその開発思想，方針，特長などについて述べる。

2 SPL開発の基本思想と方針

リアルタイムシステム建設の成否は，大勢の人々によって作られる膨大なソフトウェアを，予定された期間と費用でいかにうまく組み立てるかというプロジェクト管理にかかっているといっても過言ではない。

ソフトウェアプロジェクト管理の第一の難しさは，「ソフトウェアそのものが目に見えない」ところにあり，この問題を解決するには，

(1)　プロジェクトリーダから常にソフトウェア全体が見えるようにし，不具合点を早期に発見できるようにすること。

(2)　不具合点の修正は簡単にでき，他に波及しないこと。

が重要である。

第二の難しさは，「リアルタイムシステムは多くの人々によって，一つの有機的なシステムに作り上げられる」ところにあり，この問題を解決するには，

(3)　プログラミング言語の制約を受けることなく，経験，能力に応じて適切な作業分担を行なって，プログラムのモジュール化が信頼度高く行なえること。

(4)　上位技術者の考えや指示が，下位技術者に的確に伝わり，関係者による作成プログラムのレビューが容易に行なえること。

(5)　モジュールの組立てが論理的，かつ一意的に行なえ，メモリサイズなどの物理的制約については別途手段によって論理的構造を崩すことなく容易に行えること。

といった点が重要である。

* 日立製作所大みか工場　** 日立製作所システム開発研究所　*** 日立製作所システム開発研究所 理学博士　**** 日立製作所日立研究所

これらのことは，リアルタイムアプリケーションシステムでよく発生する，「調整期間中あるいは稼動期間中の機能の変更，追加」に対しても非常に有効な手段となる。

また，上記二つの難しさのいずれに対してもうまくこたえられ，かつ，生産性，信頼性及び保守性のいずれの面からも優れているのは標準化である。しかし，従来標準化には，「適用範囲を広げるには汎用化が必要となり，汎用化すると冗長化し，冗長ゆえに効率が悪くて使えない」というジレンマが生じ，これを解決するためには，

(6) 汎用的な形で標準化しても専用的な形で使えること。

が重要で，また使いやすさという面から，

(7) 標準化されたプログラムの本質的な機能だけを抽象化し，プログラムインタフェースとして一意的な形で表現できること。

が重要である。これらの要求は非常に新しい要求であるため，既存の言語には機能豊富な汎用言語のPL/I（Programming Language/I）といえども上記(1)～(7)の要求を満たす十分な機能は備わっておらず，また，PL/Iには制御用としては不必要な機能も数多くある。一方，最近とみに専門家の間で評価の高い教育用言語PASCALは，簡潔な中にも豊富なストラクチャードプログラミング機能を備えた新しい思想の言語であり，習熟も容易であるが，リアルタイム用としては不足な機能も多い。

そこで，SPLの言語仕様の設定に当たっては，これらPL/IとPASCALの長所を採り入れ，更に，上記(1)～(7)の要求とリアルタイム処理要求とを満たすための機能を大幅に強化することとし，結果としてオブジェクト効率及び保守性が良く，リアルタイムシステムを生産性高く建設でき，更には，ソフトウェアの標準化，問題向き言語への定式化が容易な，バランスのとれた言語とすることを開発の基本方針とした。

3 SPLの言語機能上の特長

先の2.で述べた基本方針に沿って開発したSPLの言語機能のうち，特長的なものについて次に述べる。

3.1 トップダウンストラクチャードプログラミング機能

(1) ブロック構造プログラムとその制御文

SPLはストラクチャードプログラミングを行なうために，1入口1出口のブロック構造の積み重ねでプログラムが構成できるようになっている。そして，これらブロックの流れの制御を行なうための構造化制御文として選択文(IF～THEN～ELSE～END)と，反復文(FOR～REPEAT～END)があり，プログラム構造の単純化により高信頼化とエラー検出の容易化を図っている(平面的構造化，**図1**)。

(2) 空間的構造化とマクロ定義

プログラムの内容が第三者から見て理解しやすいようにするには，(1)の平面的構造化に加えて概略レベルから詳細レベルへと詳細化していく空間的構造化が重要である。

これは，一般的には段階的詳細化といわれる技法であり，この技法を用いれば**図1**は**図2**(a)のように表わされ，非常に理解しやすくなる。SPLではこのための機能として，

(a) 処理部のマクロ定義のためのFUNCTION宣言文(関数定義，**図2**(a))

(b) データ部のマクロ定義のためのTYPE宣言文(新データ型定義，**図2**(b))

があり，いずれもSPLのステートメントを用いてトップダウンストラクチャードにマクロ定義が行なえる。

これらの機能は，**図2**からも分かるように先に詳細レベルが分かっていれば，これらマクロ定義の機能を用いて逆に抽象化でき，標準化に非常に有効である。

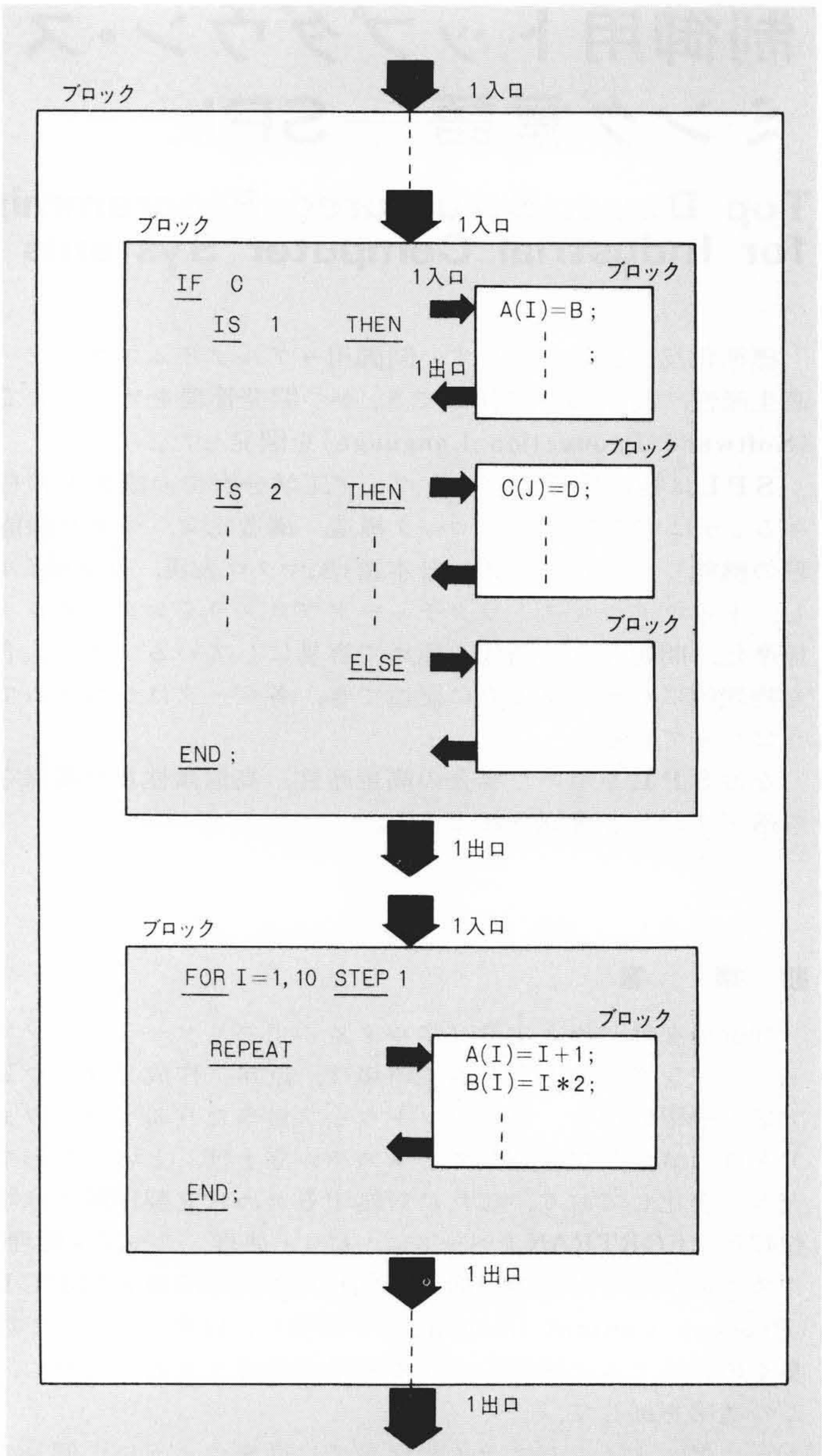
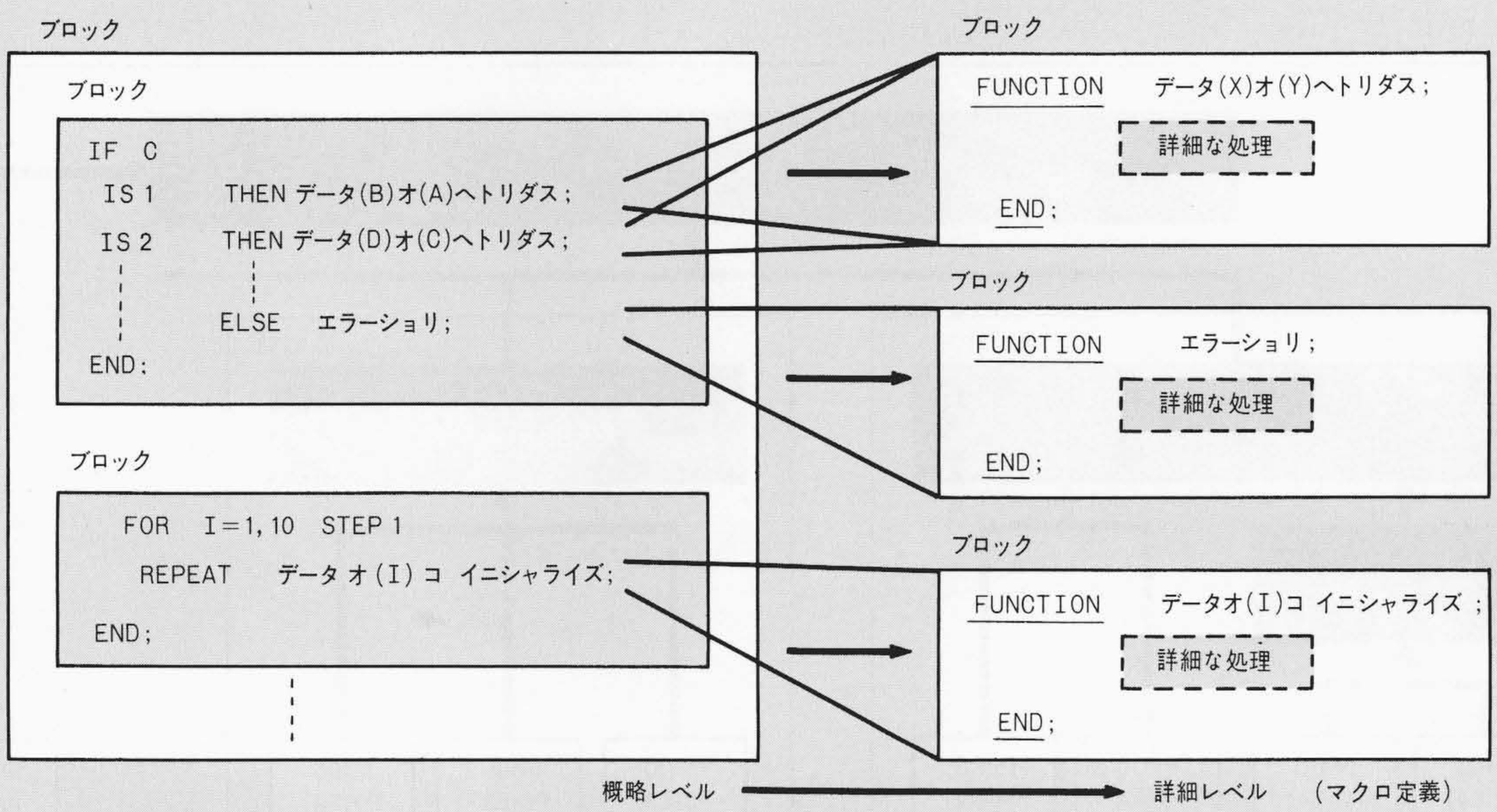


図1 プログラムの平面的構造化と制御文 SPLプログラムは，ブロック構造と構造化制御文により平面的に構造化され，単純化，高信頼化及びエラー検出の容易化が図られる。

(3) データ部と処理部の分離と一元管理機能

前記(1)，(2)の機能によりトップダウン型のストラクチャードプログラミングは一応可能であるが，オンラインリアルタイムシステムではこれら二つだけでは十分でない。すなわちオンラインリアルタイムシステムでは，一つの有機的なシステムを大勢の人々が分担しながらソフトウェアを組織的に構築していくため，プロジェクト組織内で行なわれる上位技術者から下位技術者へのトップダウンな指示のやり方を反映させたプログラミング方式をとれることが重要である。このとき，リアルタイムシステムの基本となるデータ構造(システムアーキテクチャを反映している)をその共通度に応じて関連の取りまとめ技術者が一元的に管理できることがポイントである。

これに対して，従来の言語は一つのコンパイル単位内で必ずデータ部と処理部とが一体化されている必要があった。こ



(a) プログラムの空間的構造化と処理部のマクロ定義

(b) プログラムの空間的構造化とデータ部のマクロ定義

図2　プログラムの空間的構造化とマクロ定義機能　空間的構造化を行なえば概略レベルだけをみることによりプログラムの処理内容を理解できる。この場合，処理部とデータ部とを並行的にマクロ化できることが重要である。

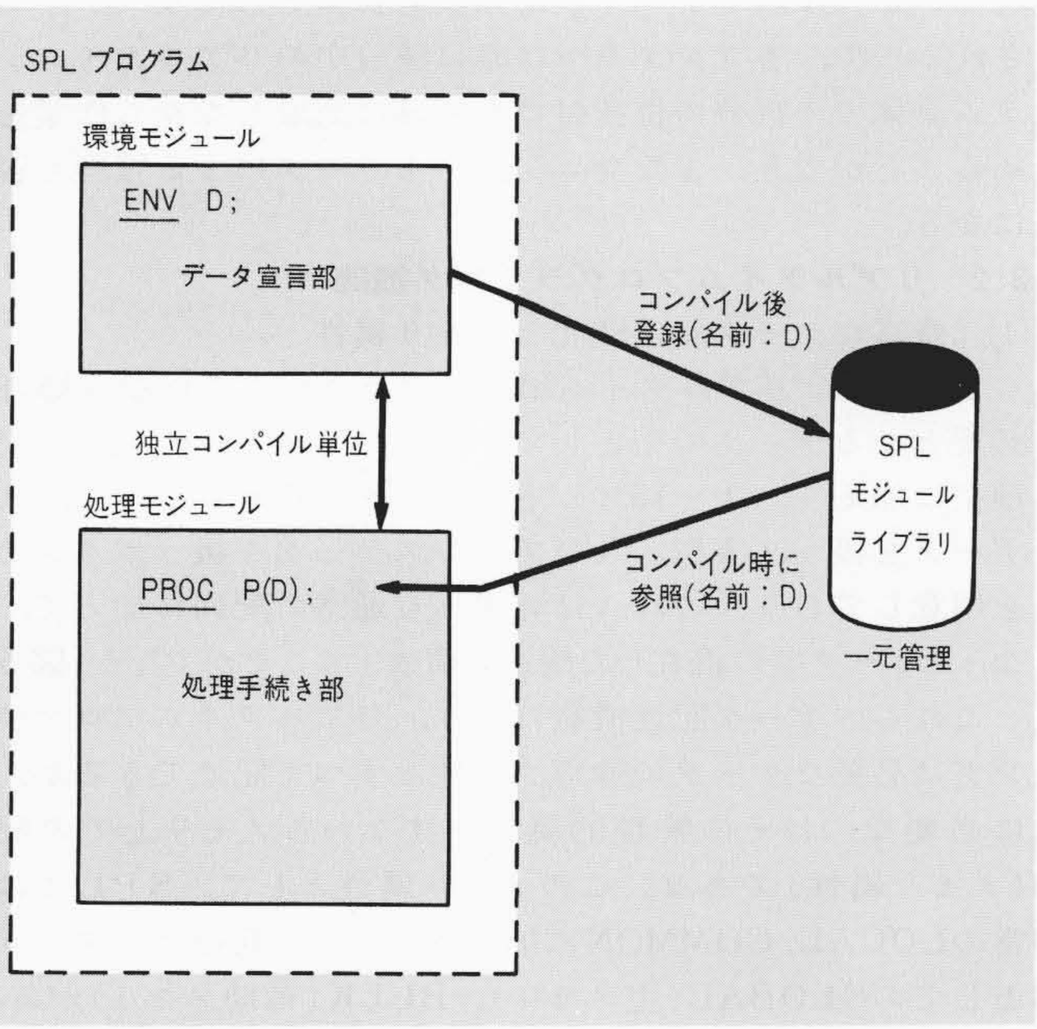


図3　SPLのプログラム構造とデータの一元管理　SPLではデータ部と処理部は独立にコンパイル，ライブラリ登録できる。したがって，共通データ部を先にコンパイル，ライブラリ登録しておくことにより，データの一元管理が可能となる。

のため，共通データは本来一箇所にしか存在しなかったにもかかわらず，その共通データを参照する部分には必ず共通データの宣言が必要で，モジュール化を進めれば進めるほど共通データ部の重複記述が増え，システムあるいはプログラム全体としてみた場合，記述ミスの発生や変更時の弱さから信頼性が低下し，ひいては生産性の低下へとつながっていた。

ＳＰＬではこの問題を解決し，トップダウンプログラミングやモジュール化を能率よく，かつ信頼性高く行なえるように，各プログラムのデータ部と処理部を各々環境モジュール(ENVIRONMENT)，処理モジュール(PROCESS)として分離し，これらを独立にコンパイル後，ライブラリに登録することによりシステム全体にわたるデータの一元管理を可能としている(**図3**)。このとき，テーブルレベルの一元管理だけでなく，テーブルとしては別のものであるが，その部分構造が同一であるものに対しても，前述の新データ型名(**図2**(b))を用いて部分構造レベルまでのきめの細かいデータの一元管理が行なえるようになっている。

(4)　モジュール階層表示機能とリアルタイムアプリケーションシステムの構造表現

一元的に管理される環境モジュール相互の関係，及び環境モジュールと処理モジュールとの関連を示すための機能として，モジュール階層表示機能がある。ＳＰＬによって組み立てるソフトウェアシステムは，常にこの階層表示機能によっ

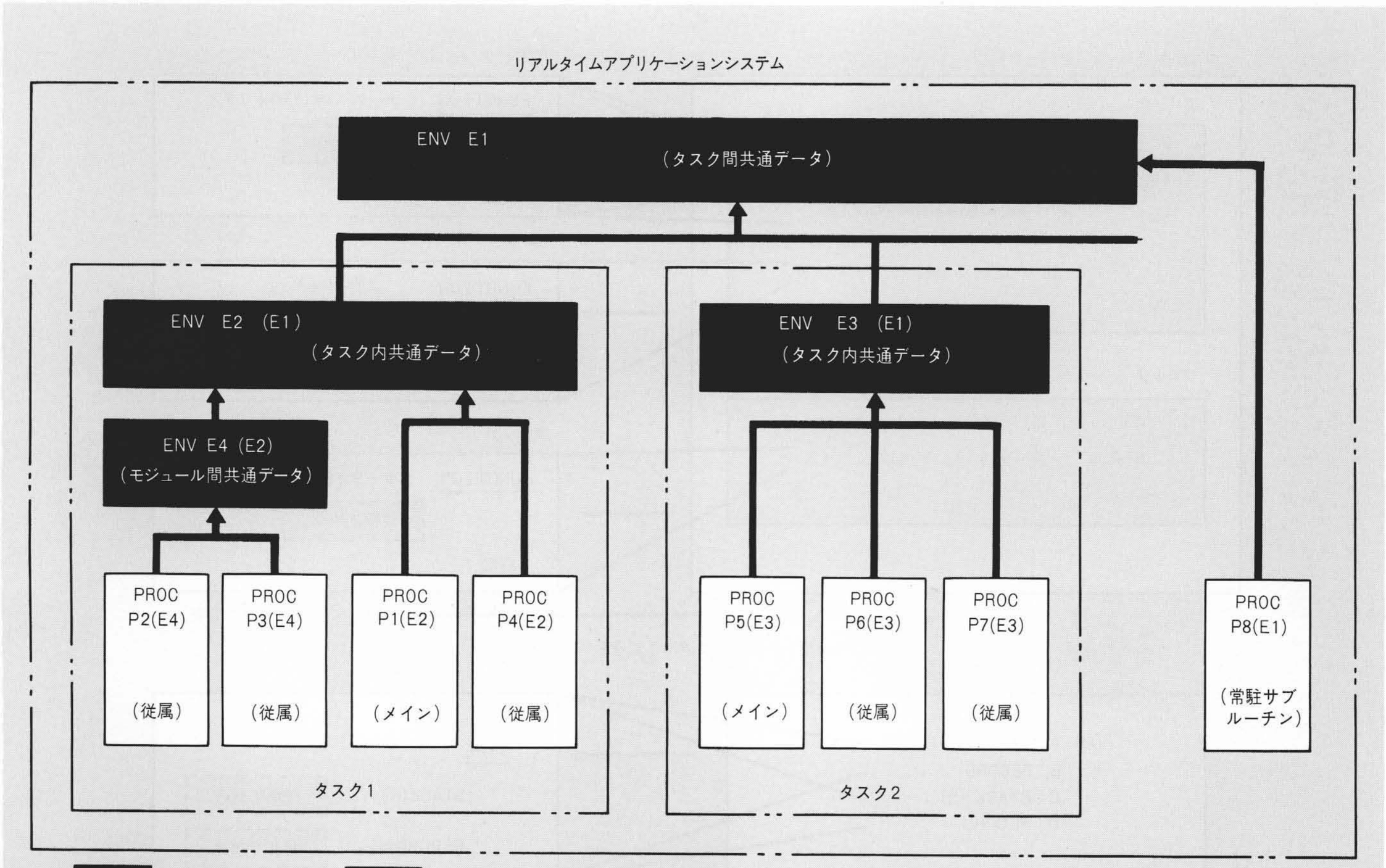
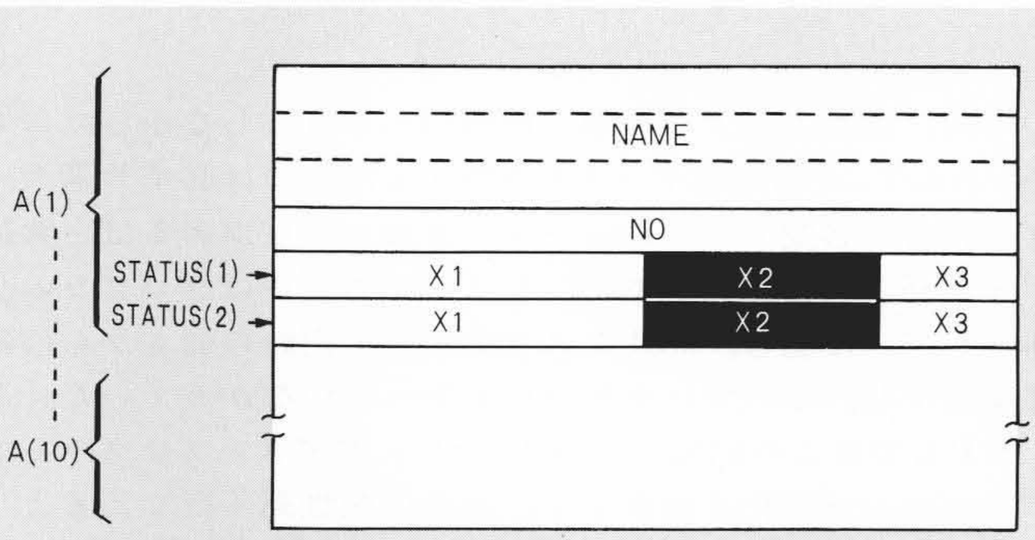


図4 モジュール階層表示機能とシステム構造の反映例 SPLではモジュール相互の関係を階層的に表現でき，これにより，本例のように実際のシステムのソフトウェア構造との対応を容易にとることができる。



(a) 構造テーブルの例

```
VAR  A(10) : STRUCT( NAME : CHAR (6),
                     NO   : INT,
                STATUS(2) : STRUCT ( X1 : BIT(8),
                                     X2 : BIT(5),
                                     X3 : BIT(3)));
```

(b) データ記述例

```
A(1). STATUS(1).X2=10;
A(1). STATUS(2).X2="01010"B;
```

(c) 使用例

図5 構造テーブルの記述と使用例 複雑なテーブルは構造体記述により簡単に表現でき，テーブル要素へのアクセスは修飾名方式により行なわれる。

てモジュール相互の関係が**図4**のようなイメージでライブラリ及び各プログラムの中に記憶されており，この構造は各処理モジュールのコンパイル時に参考資料としてリスティングされるので，各プログラマは常に建設中のリアルタイムシステム全体での自分の位置付けを明確に認識しながら作業を進めることができ，またプロジェクトリーダによる管理も容易になる。

3.2 リアルタイムプログラミング機能

(1) 豊富なデータ記述機能とメモリ属性

SPLにはリアルタイム用プログラミングを行なう場合に必要となるデータの型として，通常の整数型，実数型及び論理型に加えて，1～15ビットのデータを扱うビット型，文字データを扱う文字型，及びアドレスデータを扱うポインタ型を用意しており，これらはいずれも通常の配列に加えて，異なったデータ型の混在した構造を構成することができる(**図5**)。

これらのデータ記述機能により，リアルタイムプログラミングに必要なデータの論理的属性はすべて記述できるが，次に必要なのはその物理的属性，すなわちメモリ上での配置(メモリ属性)である。このメモリ属性として，SPLでは通常のLOCAL, COMMONに加えてタスク間共通データエリアとしてのGLOBAL(主メモリ)，BULK(補助メモリ)がある。

SPLでは，これらのいずれのメモリ属性をもつデータに対しても区別することなく混在して同一の式中に記述し，変数へ代入することができる。そして，特に，BULK変数に対するこの処理を仮想メモリデータ処理と呼び，リアルタイム

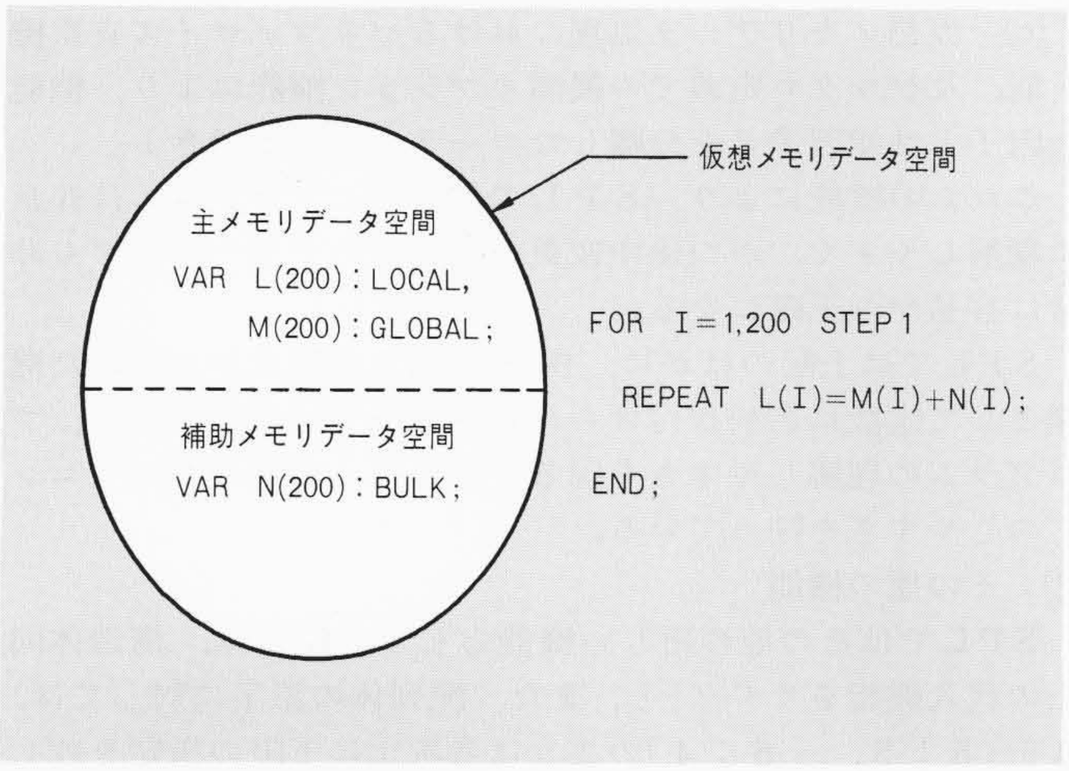


図6　仮想メモリデータ処理　SPLでは主メモリと補助メモリとを区別することなく，全体を一つのメモリ(仮想メモリ)空間と考えて，データの処理ができる。

プログラミングでよく現われる繁雑な補助メモリとの入出力処理を自動化している(**図6**)。

また，プログラム実行時に，処理するテーブルが動的に変わる場合の処理のためにBASED変数と，POINTER変数とを用意している。

(2)　タスク間通信と資源管理のための機能

SPLでは，タスク間の交信や共用資源管理のための機能をシステム交信文として用意し，HIDIC 80オペレーティングシステムとの有機的な結合を図っている。

(3)　豊富なリアルタイム用入出力機能

制御用リアルタイムシステムでは，プロセス入出力装置やタイプライタ，カラーCRT(Color Cathode Ray Tube)などを用いて，プロセスデータの入出力，帳票作成，マンマシン処理が行なわれるが，SPLではこれらのいずれに対してもREAD/WRITE文で統一的に記述でき，また，入出力処理要求タスクと実際の入出力実行動作とが同時並行的に行なえるように標準入出力タスクを別途用意している。また作表のための書式に対しては，ゼロサプレスや，(カンマ)挿入などが行なえる豊富なFORMAT変換機能を備えており，カラーCRTに対してもカラーやカーソルの制御が行なえるようになっている。

(4)　他言語で作られたプログラムとの容易な結合性

SPLでは他言語で作成されたプログラムとのリンケージのために，リンケージ仕様記述部(SPECIFICATION)があり，そのリンケージ仕様を次のように書くことにより，CALL文を使うことなくSPLに溶け込んだ形で容易にリンケージをとることができる。

(例) SPEC;FUNC
PUT(A:CHAR(8))TO(B:INT)OPT(SUB);

3.3　標準化，問題向き言語化機能

標準化，問題向き言語化機能として，**3.1**(2)の関数定義文，新データ型定義文に加えて，次の二つの機能がある

(1)　コンパイル時編集機能と可変マクロ展開

汎用的に作ったプログラムを専用的な形にするため，ソースプログラムの編集のやり方を，あらかじめコンパイル時実行文を用いてプログラムの中に組み込み，コンパイル時にそれらを実行することにより，ソースプログラムの編集を行なう機能がある。コンパイル時実行文は，PL/Iのそれとは異なり通常の文と同一レベルで文法チェックが行なわれ，コン

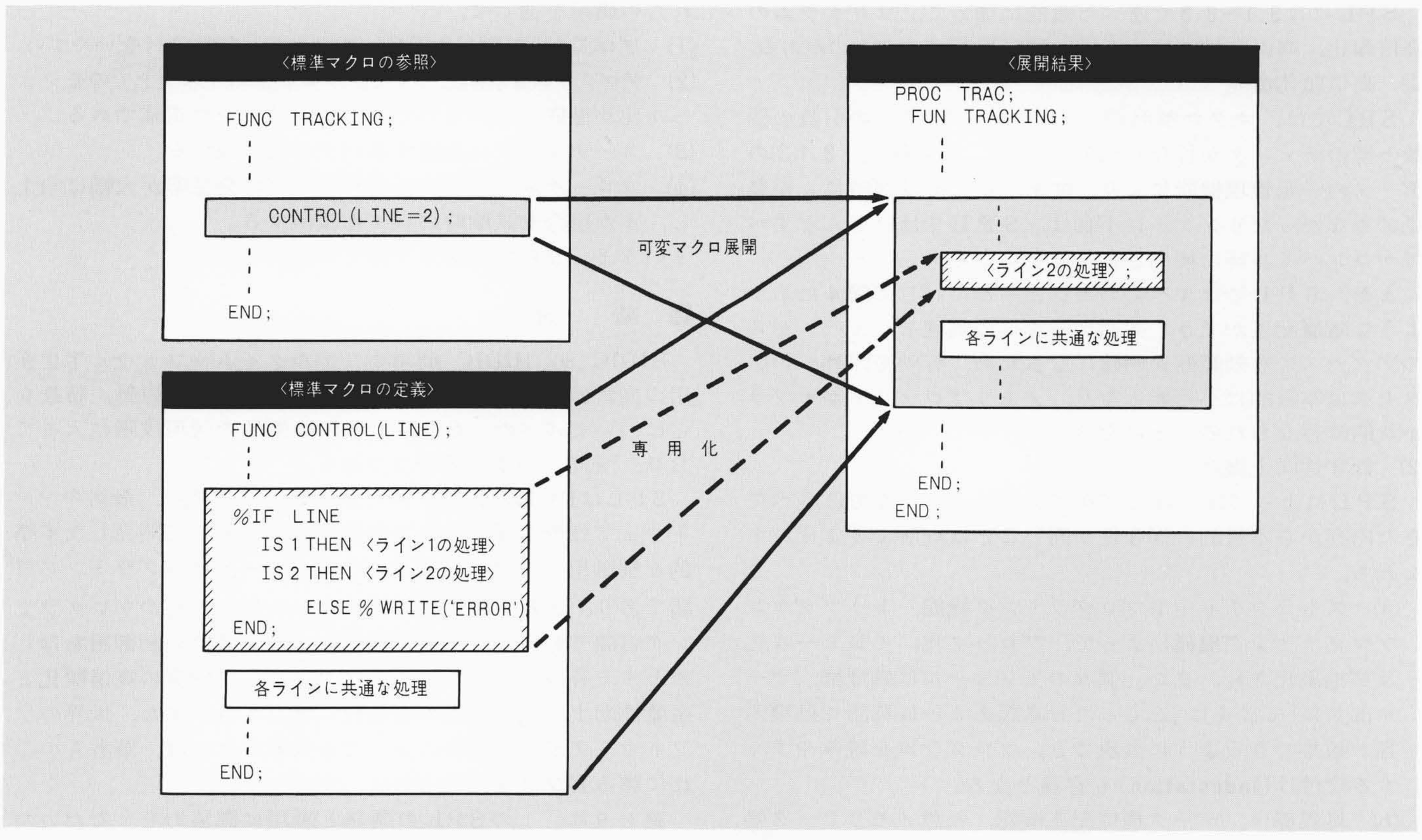


図7　コンパイル時編集機能と可変マクロ展開　汎用的に作られた標準マクロは，コンパイル時実行文(%文)をコンパイル時に実行することにより，専用的な形に可変マクロ展開される。

パイル時編集が行なわれても編集されたプログラムが文法的に矛盾を起こさないように考慮されている。

また，コンパイル時実行文の式の中にはマクロ定義の仮引数名を自由に使用でき，マクロを使用するときのパラメータによって，マクロの展開のやり方を変えることができるようになっている(可変マクロ展開，**図7**)。

(2) 豊富なマクロ表現機能と展開形式

標準化されたプログラムのマクロ表現形式は，従来のサブルーチンと同じPUT(A, B)のような形に加えて，

PUT(A)TO(B)

プット(DATA=A, FILE=B)

のような英語又は日本語(片仮名)の自然語形式，あるいはキーワード形式，更には，これらの混合形式の任意のマクロ表現がとれ，意味の理解がしやすくなっているとともに，マクロ参照に当たってはCALLといった特別なキーワードは不要なため，問題向き言語への定式化も容易になっている。

また，これらマクロはその展開形式として，インライン展開を行なうOPEN，内部サブルーチンへの展開を行なうCLOSED，及び外部サブルーチンへの展開を行なうSUBの3種類があり，メモリ効率と実行効率と使用頻度の関係から，ユーザーが任意にOPTIONS指定を行なうことにより，これらの選択を行なうことができる。なおこの場合，いずれの展開形式を指定してもマクロ定義のやり方を変更する必要はない。これにより，SPLでは標準プログラムの参照側とは独立に標準プログラムの定義側のOPTIONSを調整することにより，最適なプログラムの生成が可能となる。

(例) FUNC
PUT(A)TO(B) OPT(OPEN / CLOSED / SUB)
〔マクロ定義本体〕
END PUT;

3.4 その他の機能

SPLには**3.1**～**3.3**で述べた機能に加えて，プログラムの高信頼化，高保守性を図るための機能として次のものがある。

(1) 高信頼化機能

SPLでは，マクロ参照時(コンパイル時)にその引数の個数と型のチェックが行なわれる。そして，この機能と**3.1**(3)のデータの一元管理機能により，従来，組合せ試験時にしか発見できなかったリンケージ不良は，SPLではほとんどすべてがコンパイル時に検出されることになる。

また，SPLではすべてのモジュールに対して**図4**に示すような階層記述ができ，共通データへの処理モジュールからのアクセスの有効範囲が明確になるため，有効範囲外へのアクセスは本質的に不可能となり，メモリプロテクトがオフライン的に行なわれることになる。

(2) 保守性向上機能

SPLによって作られたプログラムは，これまでに述べてきた内容から本質的に保守性が高いことは理解できよう。すなわち，

(a) ストラクチャードプログラミング機能，トップダウンプログラミング機能によって，ブロック化，モジュール化及び抽象化され，また，個々のモジュールは処理部，データ部共に「てにをは」をもった日本語あるいは英語で処理内容が類推できるように表現でき，プログラムを読みやすくする段付け(**Indentation**)も容易となる。

(b) 処理部は，データ構造記述機能，仮想メモリデータ処理機能及び環境モジュールの導入により，論理的にも物理的にも完全にデータ独立に記述できる。

(c) 仮想メモリデータ処理におけるバッファサイズ宣言機能，及びマクロ定義での展開オプション機能により，機能因子と性能因子とを分離したコーディングができる。

これらの機能により，SPLで作られたプログラムは非常に理解しやすく，かつ途中変更や完成後のチューニングも非常に容易かつ正確になる。

SPLでは上記のほかに，保守性向上の観点から設けた機能として定数に名前が付けられるCONSTANT文があり，プログラムの理解しやすさを図るとともに，変更やチューニングのしやすさを図っている。

(3) その他の機能

SPLではその他の新しい機能として，配列体，構造体同士の代入機能をサポートし，また，配列体の添字に対しては，A(－5：5，－6：4)のように各次元に下限の指定を許し使いやすさを図っている。

4 SPLの処理系の特長

SPLコンパイラはHIDIC 80下でのセルフコンパイリングシステムを既に開発し，更に顧客側での便宜を考慮して，HITAC 8000シリーズ，Mシリーズ，IBM370シリーズ下でのクロスコンパイリングシステムを現在開発中である。

また，SPLで作成したプログラムのオブジェクト効率は，PCLで直接作成した場合のベストコーディングに比較して，機械語ベースで平均5％の劣化が確認されているが，SPLプログラムの構造の単純さと各種機能により個人差がPCLより圧縮され，システム全体では，PCLで作成した場合よりコンパクトになることが期待されている。

5 SPLの適用状況とその結果

SPLは昭和52年3月にその第1号適用システムを出荷し，その後これまでに十数システムへの適用を行なってきた。これらの適用を通じて，

(1) プログラムの記述がPCLに比べて，非常に行ないやすい。
(2) プログラムが理解しやすく，プログラムリスト上でのモジュール化が徹底しているので，修正が早くかつ正確である。
(3) コーディングに起因するバグが約半減する。
(4) フローチャート不良も含めたバグの発見率が大幅に向上し，また組合せ試験時のバグが激減する。

というような良好な結果を得ている。

6 結　　言

HIDIC 80, HIDIC 08用のリアルタイムソフトウェア生産用の高級言語SPLについて，その開発思想，方針，特長などについて述べた。SPLは現在積極的な適用段階に入っており，所期の効果を挙げつつある。

SPLはPCLの数年にわたる適用経験を基に，最新のソフトウェア技術を採り入れて世界に一歩先がけて開発した本格的な制御用トップダウンストラクチャードプログラミング言語であり，また，標準化，モジュール化のためのプログラミング言語でもある。この言語がユーザー各位の制御用をはじめとする各種のリアルタイムソフトウェア開発の高信頼化と生産性向上，及び保守の容易化に寄与でき，また，世界のソフトウェア工学の進歩に少しでも貢献できれば，筆者らのこれに勝る喜びはない。

終わりに，このSPLの開発と適用に御協力をいただいた関係各位に対し，深謝の意を表わす次第である。