第３章　構造化プログラミング言語ＳＰＬの設計思想

第３章　構造化プログラミング言語ＳＰＬの設計思想[C9)]

3.1　概　　要

大規模ソフトウェアの信頼性と生産性向上のために，構造化プログラミングに代表される幾つかのプログラミング方法論が提案されている。これらの方法論に共通な本質は，モジュール分割によって生じるプログラム構造を理解容易なものにすることである。そこで，たとえ従来の言語を用いてプログラムを記述する場合でも，この方法論の主旨に沿った工夫を施せば，ある程度の効果が得られる。しかしながら，このような個人の能力に依存する方法は，多人数で開発する大規模ソフトウェアには適していない。即ち，プログラミング方法論は，それを具現化するプログラミング言語の導入によって，その効果を十分なものにすることができる。

本章では，このような考えに基づき，制御用応用プログラム記述のための構造化プログラミング言語を新たに開発するにあたって，その言語に必要とされた要求機能およびそれを満たすための基本的言語構造について述べる。特に，第１章で述べた従来方式の問題点を解決するために，第２章で述べたプログラミング方法論をどのような言語構造に具体化するかに重点を置き，具体的な言語仕様は４章で述べる。

3.2　新言語への要求機能

本研究の対象である日立の制御用計算機ＨＩＤＩＣ８０用応用プログラムの記述は，従来，制御用高級言語ＰＣＬ[H3)]（Process Control Language）を用いて行われていた。このＰＣＬは，汎用の高級言語Fortranにマルチタスク機能とビットデータ処理機能および構造体のデータ型や主メモリ常駐のデータ属性などを加えたもので，タスク単位の基本的なプログラム構造はFortranと同じである。そのため，制御用応用プログラムの機能に重要な役割を果たすタスク間共通データは次のように扱われていた。

(1)　共通データは機能的なまとまりで幾つかのグループに分けられ，一般に構造体と配列を組合せた複雑なデータ構造を有する。

(2)　タスク内の主プログラムおよび副プログラムは，この共通データの宣言を行うことによりアクセス可能となるが，通常はその一部分にのみアクセスする。

(3)　共通データは，各プログラムの処理内容に依存して異なるデータ型で宣言されることが多い。

このようなことから，1.2.2節で述べたように，従来方式では共通データへのアクセスエラーが多いという問題が生じていた。そこで，共通データに関して，次のような要求項目が新言語に課せられた。

（Ｒ１）　そのデータにアクセスすべきでないタスクからのアクセスを防ぐために，共通データへのアクセス権は厳密に表現されるべきである。

（Ｒ２）　複雑なデータ構造に対する処理方法の誤りを防ぐために，データ構造をカプセル化して，そのデータへのアクセスは専用手続きを経由して行うデータ抽象化技法を導入すべきである。

（Ｒ３）　パラメータ渡しの共通データのデータ型不一致のような手続き間インターフェイス誤りを防ぐために，厳しいデータ型チェック機能を設けるべきである。

以上は，データに関するものであるが，従来言語は制御構造の表現に関しても不十分なものであった。即ち，ＰＣＬでは，各タスクを構成する１つの主プログラムと幾つかの副プログラムを各々独立に記述してコンパイルする。そのため，1.2.2節で従来方式の第２の問題として述べたように，各タスクの処理設計での機能分割によって生じる階層構造をプログラムに表現できないので，設計書とプログラムの対応が不明確になっていた。さらに，各々のプログラム内の制御構造もGOTO文を用いて表現されるために複雑化し，理解しにくいものになっていた。このようなことから，従来方式では，プログラムの修正，保守が難しいという問題が生じていた。そこで，プログラムの制御構造に関して，次のような要求項目が新言語に課せられた。

（Ｒ４）　段階的詳細化技法によるプログラム開発において，その各々の段階に対応したプログラムが作成され，詳細化の過程がプログラム構造に表現されるべきである。

（Ｒ５）　制御文は，構造化コーディング用の複合文，選択文，反復文に限定し，GOTO文を排すべきである。

以上，新言語に対する５項目の技術的要求について述べたが，その他に，次のような管理運用面からの要求項目がある。

（Ｒ６）　従来言語に慣れている平均的プログラマにとって，修得容易な言語でなければならない。

（Ｒ７）　従来言語で記述された既存プログラムの一部を流用できるようなモジュール結合機能が必要である。

これらの項目は，既にソフトウェアに関する多くの財産，即ち，プログラマとプログラムを有する分野に新しい言語を導入する場合には必須の要件である。

3.3　木構造形式のモジュール階層構造

プログラムの中で用いられるデータのアクセス権を制約するための代表的な方法として，Algol，ＰＬ／Ｉ，Pascalなどで採用されているブロック構造がある。ところが，ブロック構造言語では，データと手続きの名前の有効範囲が同一の規則に従うため，データ抽象化の実現が難しい。即ち，あるデータとその操作手続き群をカプセル化する場合を考えると，そのデータの使用者から操作手続き名は参照できるがデータ名は参照できないようにするために，操作手続きはそのデータを宣言しているブロックの外で定義されなければならない。一方，このデータは，その操作手続き群から共通に参照されるために，これらの手続き定義を含むブロックまたはその外で宣言されなければならない。結局，データ宣言と手続き定義の位置関係に矛盾が生じ，ブロック構造言語ではデータ抽象化が難しい。

そこで，データ抽象化の実現のためには，その支援言語であるＣＬＵ，Alphard，Adaなどで採用しているデータ抽象化機構が必要である。ところが，この場合はブロック構造のような階層的データアクセス権の制約が難しい。しかも，このようなデータ抽象化機構は，従来の汎用手続き型言語

とは異質なため，従来言語に慣れたプログラマが使いこなすのは容易ではないと思われる。

以上の問題を解決するために，ＳＰＬでは，ブロック構造を変形して，データの名前の有効範囲は階層構造にするが，手続き名には制約を設けないようなプログラム構造を導入した。即ち，次のような方法をとった。

(1) 共通データの宣言を手続き定義と分離，独立させ，環境モジュールとしてまとめる。

(2) これらの共通データの有効範囲を厳密に表現するために，環境モジュールは複数可能とし，お互いを木構造形式に結合して，階層化できる。

(3) 手続きは，関連するものを幾つかまとめて処理モジュールとし，適切な環境モジュールの下に結合する。そして，そこからアクセス可能な共通データは，その上位に位置する環境モジュール内で宣言されたものに限る。一方，手続きは他の処理モジュールから引用可能とする。

ＳＰＬにおけるモジュール階層の例を図3.1に示す。この例では，例えば，環境モジュールＥ3で宣言した共通データは処理モジュールＰ3とＰ4からアクセスできる。また，環境モジュールＥ2で宣言されたデータを処理モジュールＰ1からアクセスする場合は，直接の参照はできず，Ｅ2の下の処理モジュールＰ2内の手続きを引用することによってのみ可能となる。

3.4 言語の概要

木構造形式のモジュール階層構造を基本として設計されたＳＰＬの詳細な言語機能は次章で述べるが，ここでは，その概要を述べておく。

そのための例題として，制御用応用プログラムの1つである列車運行管理システム[K5)]を取りあげる。このシステムの設計時に，段階的詳細化技法を適用して最上位レベルから順に機能分割を行っていき，まず図3.2に示したようなモジュール構成を得る。即ち，最上位レベルの機能分割により，次の3個のタスクに分割する。

(1) タスク1 ： 列車運行状態把握

(2) タスク2 ： 進路制御

(3) タスク3 ： 列車運行情報表示

そして，これらのタスク間共通データである，

(1) 列車運行状態テーブル

(2) 運行計画テーブル

を最上位の環境モジュールに置く。次に，各タスクの機能を更に分割して，合計7個の処理モジュールを得る。その時，各タスク内での処理モジュール間共通データは第2レベルの環境モジュール内に置くと共に，各々の処理モジュール内での共通データは第3レベルの環境モジュールに置く。

次に，ここで得られたモジュール構成に基づき，各々のモジュールに対応するＳＰＬプログラムを作成する。図3.3および図3.4は，3番目の処理モジュールの上位環境となっている3個の環境モジュールおよびそれ自身のプログラムリストである。まず，図3.3(a)の最上位環境モジュールは，「シ

ステム」という名前で定義され，タスク間共通データである列車運行状態テーブルと運行計画テーブルを各々ＴＲＮＪ，ＵＰＬＡＮという変数名で宣言している。そのデータ型は，１００個および１０００個の要素を持つ配列で，その配列要素のデータ型は「レッシャジョウタイ」および「ウンコウケイカク」というユーザ定義データ型名で指定されており，その詳細はまだ未定である。なお，プログラム中の行番号の２行目のＧＬＯＢＡＬおよび５行目のＢＵＬＫはメモリ属性を指定しており，前者はそのデータを主記憶に常駐すること，後者はそのデータを通常は補助記憶装置に置き，必要に応じて主記憶バッファに読み込んで使用することを意味し，共に外部名指定を兼ねる。なお，図中の行番号や欄番号は言語処理系が自動的に表示している。

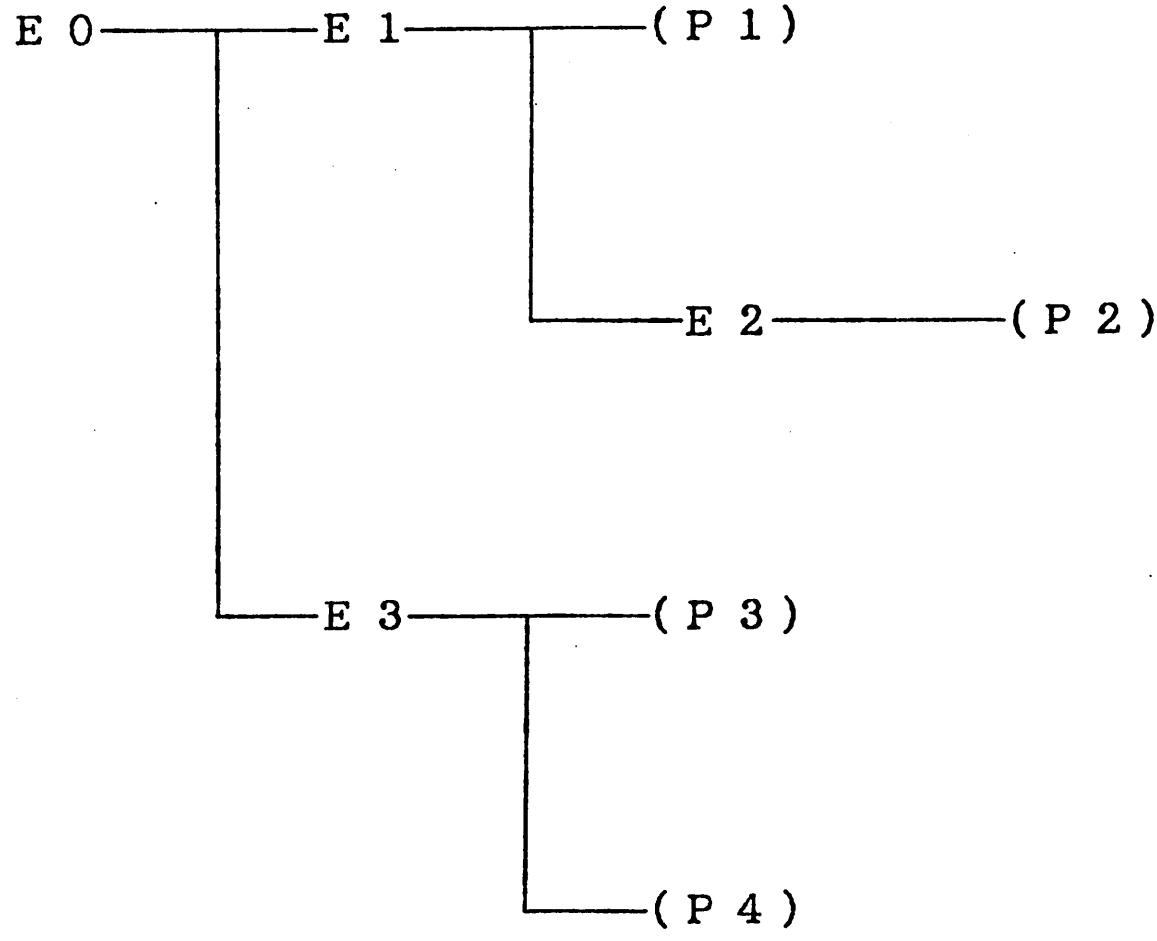
次に，図3.3(b)に示す第２レベルの環境モジュールでは，タスク１の中の４個の処理モジュール間の共通データを宣言している。まず，１行目で，この環境モジュール名を「ジョウタイ」と定義すると共に，その親環境として図3.3(a)の「システム」を指定している。その本体では，共通データである設備状態テーブルとプロセス設備状態テーブルを各々ＳＪＯＴＡＩおよびＰＪＯＴＡＩという変数名で宣言している。そのデータ型は，図3.3(a)と同じく配列とし，配列要素のデータ型にはユーザ定義データ型名を用いている。

次に，図3.3(c)に示す第３レベルの環境モジュールでは，タスク１の中の３番目の処理モジュール内で用いる共通データを宣言している。まず，１行目で，この環境モジュール名を「ツイセキ」と定義すると共に，その親環境として図3.3(b)の「ジョウタイ」を指定している。そして，２～５行目の最初の宣言ユニットでは，上位環境である「システム」や「ジョウタイ」と同じく，２個の共通データをユーザ定義データ型を用いて宣言している。６～１３行目の宣言ユニットでは，これまで引用されたユーザ定義データ型のうちの５個について，その詳細なデータ構造を定義している。最初の３個は１語長（１６ビット）の整数型，４番目は２語長の整数型で，各々ＩＮＴおよびＩＮＴ(2)と指定している。５番目の「レッシャツイセキ」は８ビット長のビット型フィールドを２個組合せた構造体で，次のように定義している。

```
ＴＹＰＥ　レッシャツイセキ　＝　ＳＴＲＵＣＴ　（　ＲＥＳＳＹＡ　：　ＢＩＴ(8)，
                                        ＧＡＩＢＵ　　：　ＢＩＴ(8)　）；
```

ここで，ＲＥＳＳＹＡおよびＧＡＩＢＵは各々のフィールド名である。これらのユーザ定義データ型は，基本的にはどのモジュールで定義しても良いが，その詳細なデータ構造を利用した処理記述は，そのデータ型を定義したモジュールまたはその下位モジュールに限られる。例えば，このプログラム例では，データ型「レッシャツイセキ」の構造体のフィールド名ＲＥＳＳＹＡを引用できるのは，この定義モジュール「ツイセキ」の下位モジュールである「レッツイセキ」だけに限られるが，詳細は5.4節で述べる。

この他，図3.3(c)では，１４～２２行目の宣言ユニットで，７個の定数名を宣言しているが，２～４番目はビット定数，その他は整定数である。１４行目のメモリ属性ＬＯＣＡＬの指定は，これらの定数名が通常のブロック構造と同じスコープルールに従うことを意味している。このＬＯＣＡＬはメ



E n : 環境モジュール

(P n) : 処理モジュール

図３．１　新言語におけるモジュール階層構造

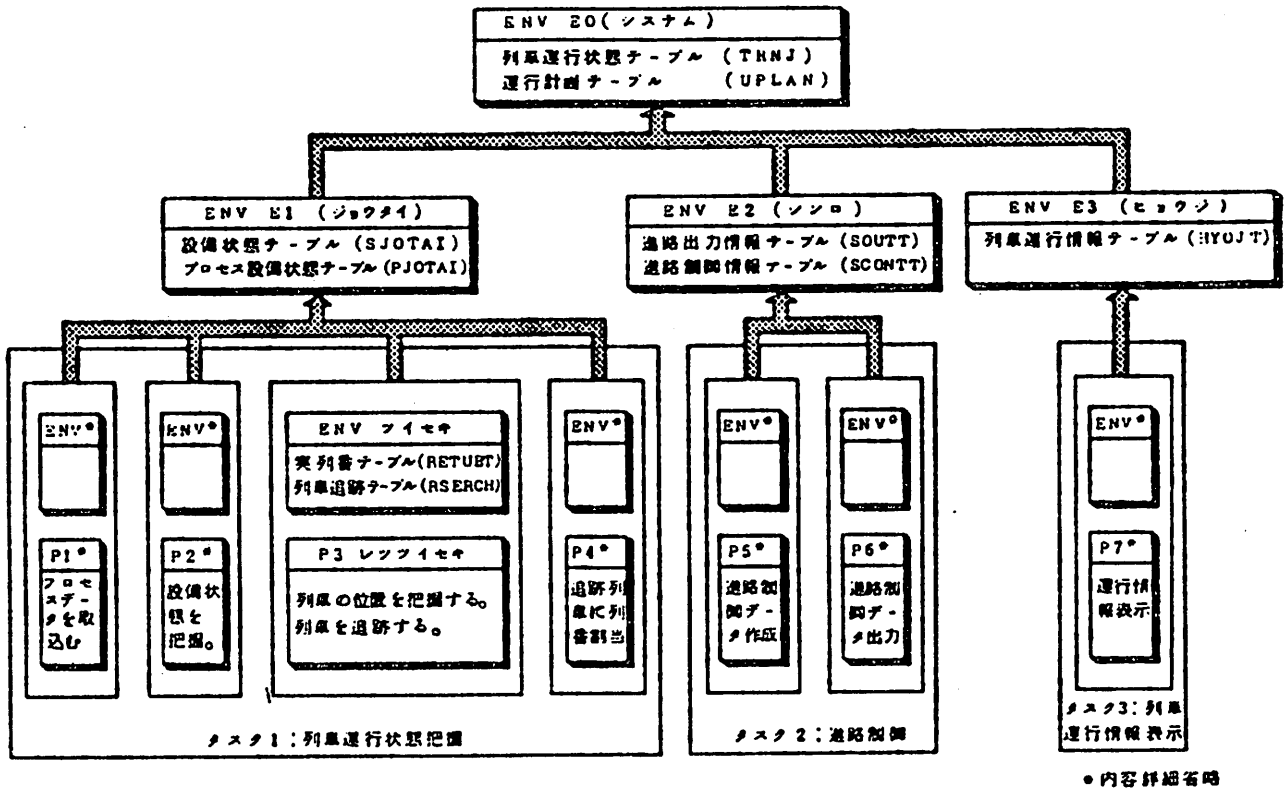


図３．２　列車運行管理システムのモジュール構成

```
MODULE    L.NO     SPL ( FILE INDEX H-80 SPL V1-R3 )  (1977  7/20) SOURCE   STATEMENT     ID.SEQ

          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8

                 *MODULE                                                                  ENV00010
          00001   ENVIRONMENT ｼｽﾃﾑ ;                          /* ｽﾍﾟ ｶﾝｷｮｳ E0 ) ｾﾝﾀｰﾝ        */ ENV00015
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
                                                                                          ENV00020
ｼｽﾃﾑ    . 00002     DCL OPT (GLOBAL) ;
ｼｽﾃﾑ    . 00003       VAR TRNJ    (100)  : ﾚｯｼｬｼﾞｮｳﾀｲ ; /* ﾚｯｼｬ ｳﾝｺｳ ｼﾞｮｳﾀｲ ﾃｰﾌﾞﾙ     */ ENV00025
                                                                                          ENV00030
ｼｽﾃﾑ    . 00004     END ;
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
                                                                                          ENV00035
ｼｽﾃﾑ    . 00005     DCL OPT (BULK) ;
ｼｽﾃﾑ    . 00006       VAR UPLAN  (1000) : ｳﾝｺｳｹｲｶｸ ;   /* ｳﾝｺｳ ｹｲｶｸ ﾃｰﾌﾞﾙ             */ ENV00040
                                                                                          ENV00045
ｼｽﾃﾑ    . 00007     END ;
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
                                                                                          ENV00050
ｼｽﾃﾑ    . 00008   END ｼｽﾃﾑ ;
```

（a）　最上位の環境モジュール　（システム）

```
MODULE    L.NO     SPL ( FILE INDEX H-80 SPL V1-R3 )  (1977  7/20) SOURCE   STATEMENT     ID.SEQ

          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8

                 *MODULE                                                                  ENV00055
          00001   ENVIRONMENT ｼﾞｮｳﾀｲ    (ｼｽﾃﾑ) ;          /* ｺ ｶﾝｷｮｳ E1 ) ｾﾝﾀｰﾝ        */ ENV00060
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
                                                                                          ENV00065
ｼﾞｮｳﾀｲ  . 00002     DCL OPT (GLOBAL) ;
ｼﾞｮｳﾀｲ  . 00003       VAR SJOTAI  (100)  : ｾｿｳﾋﾞｼﾞｮｳﾀｲ ; /* ｾｿｳﾋﾞ ｼﾞｮｳﾀｲ ﾃｰﾌﾞﾙ          */ ENV00070
ｼﾞｮｳﾀｲ  . 00004       VAR PJOTAI  (100)  : ﾌﾟﾛｾｽｼﾞｮｳﾀｲ; /* ﾌﾟﾛｾｽ ｾｿｳﾋﾞ ｼﾞｮｳﾀｲﾘﾖｳ ﾃｰﾌﾞﾙ*/ ENV00075
                                                                                          ENV00080
ｼﾞｮｳﾀｲ  . 00005     END ;
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
                                                                                          ENV00085
ｼﾞｮｳﾀｲ  . 00006   END ｼﾞｮｳﾀｲ     ;
```

（b）　第二レベルの環境モジュール　（ジョウタイ）

```
MODULE    L.NO     SPL ( FILE INDEX H-80 SPL V1-R3 )  (1977  7/20) SOURCE   STATEMENT     ID.SEQ

          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8

                 *MODULE                                                                  ENV00155
          00001   ENVIRONMENT ﾂｲｾｷ      (ｼﾞｮｳﾀｲ)     ;      /* P3 ﾉ ｶﾝｷｮｳ ﾉ ｾﾝﾀｰﾝ        */ ENV00160
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
ﾂｲｾｷ    . 00002     DCL OPT (GLOBAL) ;                                                   ENV00165
ﾂｲｾｷ    . 00003       VAR RETUBT  (100) : ｼｭﾂﾚﾂﾊﾟﾝ ;     /* ｼｭﾂﾚﾂﾊﾟﾝ ﾃｰﾌﾞﾙ             */ ENV00170
ﾂｲｾｷ    . 00004       VAR RSERCH  (100) : ﾚｯｼｬﾂｲｾｷ ;     /* ﾚｯｼｬ ﾂｲｾｷ ﾃｰﾌﾞﾙ             */ ENV00175
ﾂｲｾｷ    . 00005     END ;                                                                ENV00180
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
ﾂｲｾｷ    . 00006     DCL ;                                                                ENV00185
ﾂｲｾｷ    . 00007       TYPE ﾚｯｼｬｼﾞｮｳﾀｲ  =  INT ;                                        ENV00190
ﾂｲｾｷ    . 00008       TYPE ｾｿｳﾋﾞｼﾞｮｳﾀｲ  =  INT ;                                       ENV00195
ﾂｲｾｷ    . 00009       TYPE ﾌﾟﾛｾｽｼﾞｮｳﾀｲ =  INT ;                                       ENV00200
ﾂｲｾｷ    . 00010       TYPE ｼｭﾂﾚﾂﾊﾟﾝ     =  INT (2) ;                                   ENV00205
ﾂｲｾｷ    . 00011       TYPE ﾚｯｼｬﾂｲｾｷ     =  STRUCT ( RESSYA : BIT (8),                 ENV00210
ﾂｲｾｷ    . 00012                                    GAIBU  : BIT (8))  ;                 ENV00215
ﾂｲｾｷ    . 00013     END ;                                                                ENV00220
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
ﾂｲｾｷ    . 00014     DCL OPT (LOCAL) ;                                                    ENV00225
ﾂｲｾｷ    . 00015       CONST RAKKA = 1 ;                 /* ﾗｲﾄﾍﾟﾝｶｲﾛ ﾗｸﾗｸ ｼｮｳﾀｲ         */ ENV00230
ﾂｲｾｷ    . 00016       CONST ARI   = '1'B ;              /* ｲﾄﾍﾟﾝ ﾚｯｼｬ ｿﾝﾀﾞｲ ｼﾞｮｳﾀｲ    */ ENV00235
ﾂｲｾｷ    . 00017       CONST NASHI = '0'B ;              /* ｲﾄﾍﾟﾝ ﾚｯｼｬ ﾅｼ               */ ENV00240
ﾂｲｾｷ    . 00018       CONST GAIBU = '1'B ;              /* ｼｽﾃﾑ ｶﾞｲ ｶﾗﾉ ｼﾝﾆｭｳ          */ ENV00245
ﾂｲｾｷ    . 00019       CONST ASR   = 1 ;                 /* ASR ﾆ ﾋｮｳｼﾞ ｽﾙ               */ ENV00250
ﾂｲｾｷ    . 00020       CONST CRT   = 2 ;                 /* CRT ﾆ ﾋｮｳｼﾞ ｽﾙ               */ ENV00255
ﾂｲｾｷ    . 00021       CONST LP    = 3 ;                 /* L/P ﾆ ﾋｮｳｼﾞ ｽﾙ               */ ENV00260
ﾂｲｾｷ    . 00022     END ;                                                                ENV00265
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
ﾂｲｾｷ    . 00023     DCL ;                                                                ENV00270
ﾂｲｾｷ    . 00024       % VAR OUTPUT : INT ;              /* ｼｭﾂﾘｮｸ ｿｳﾁ ｾﾝﾀｸ            */ ENV00275
ﾂｲｾｷ    . 00025     END ;                                                                ENV00280
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
ﾂｲｾｷ    . 00026   END      ﾂｲｾｷ ;                                                      ENV00285
```

（c）　第三レベルの環境モジュール　（ツイセキ）

図３．３　環境モジュールのプログラム例

```
                MIDIC-80  SPL     LIST

MODULE     L.NO      SPL ( FILE INDEX H-80 SPL V1-R3 )  (1977  7/20) SOURCE   STATEMENT     ID.SEQ      BLOCK
                                                                                                    NEST
           -----   -L-------1---------2---------3---------4---------5---------6---------7R--------8

           00020    PROCESS  ﾚﾝｻｲｷｾﾂ                (ﾌｲﾙﾓｰﾄﾞ),;                                    PR000100
           -----   -L-------1---------2---------3---------4---------5---------6---------7R--------8
ﾚﾝｻｲｷｾﾂ  . 00021      FUNCTION  TRAIN  ﾌｲﾙﾓｰﾄﾞ ｼﾖｳｷｮ  OPT (MAIN) ;                              PR000105
ﾚﾝｻｲｷｾﾂ  . 00022         ｼﾝｺﾞｳｷ ﾉ ﾍﾝｶﾝ ﾃﾞｰﾀ ﾚﾝｻｲ ｦ ﾌｲﾙﾓｰﾄﾞ ｽﾙ ;                          PR000110
ﾚﾝｻｲｷｾﾂ  . 00023         ｷﾄﾟｳｶｲﾛ ﾉ ﾍﾝｶﾝ ﾃﾞｰﾀ ﾚﾝｻｲ ｦ ﾌｲﾙﾓｰﾄﾞ ｽﾙ ;                          PR000115
ﾚﾝｻｲｷｾﾂ  . 00024      END TRAIN ;                                                          PR000120
ﾚﾝｻｲｷｾﾂ  . 00025  C                                                                        PR000125
           -----   -L-------1---------2---------3---------4---------5---------6---------7R--------8
ﾚﾝｻｲｷｾﾂ  . 00026      FUNCTION  ｷﾄﾟｳｶｲﾛ ﾉ ﾍﾝｶﾝ ﾃﾞｰﾀ ﾚﾝｻｲ ｦ ﾌｲﾙﾓｰﾄﾞ ｽﾙ      OPT (OPEN) ;     PR000130
ﾚﾝｻｲｷｾﾂ  . 00027         VAR I : INT ;                          /* ｺｱ ﾚﾌｧﾚﾝｽ                 */ PR000135
ﾚﾝｻｲｷｾﾂ  . 00028  C                                                                        PR000140
ﾚﾝｻｲｷｾﾂ  . 00029         FOR I = 1, 100 REPEAT                                             PR000150  1
ﾚﾝｻｲｷｾﾂ  . 00030             IF SJOTAI (I)                                                 PR000155  .2
ﾚﾝｻｲｷｾﾂ  . 00031                IS RAKKA THEN                                              PR000160  ..
ﾚﾝｻｲｷｾﾂ  . 00032                          IF RSERCH (I).HESSYA                             PR000165  ..3
ﾚﾝｻｲｷｾﾂ  . 00033                           IS ARI    THEN ｼﾝﾆｭｳ ｦ ｶｸﾆﾝｽﾙ ;                PR000170  ...
ﾚﾝｻｲｷｾﾂ  . 00034                           IS NASHI THEN                                    PR000175  ...
ﾚﾝｻｲｷｾﾂ  . 00035                                     IF RSERCH (I).GAIBU                    PR000180  ...4
ﾚﾝｻｲｷｾﾂ  . 00036                                       IS GAIBU THEN ｿﾄｶﾗ ﾉ ｼﾝﾆｭｳ ;         PR000185  ....
ﾚﾝｻｲｷｾﾂ  . 00037                                                     ELSE ｱﾗｰﾑ ｼｮﾘｼｬ (I) ; PR000190  ....
ﾚﾝｻｲｷｾﾂ  . 00038                                     END ;                                  PR000195  ...4
ﾚﾝｻｲｷｾﾂ  . 00039                          END ;                                            PR000200  ..3
ﾚﾝｻｲｷｾﾂ  . 00040             END ;                                                         PR000205  .2
ﾚﾝｻｲｷｾﾂ  . 00041         END ;                                                             PR000210  1
ﾚﾝｻｲｷｾﾂ  . 00042         RETURN ;                                                          PR000215
ﾚﾝｻｲｷｾﾂ  . 00043      END ｷﾄﾟｳｶｲﾛ ;                                                       PR000220
ﾚﾝｻｲｷｾﾂ  . 00044  C                                                                        PR000225
           -----   -L-------1---------2---------3---------4---------5---------6---------7R--------8
ﾚﾝｻｲｷｾﾂ  . 00045      FUNCTION ｱﾗｰﾑ ｼｮﾘｼｬ (CRB)   OPT (OPEN) ;                             PR000235
ﾚﾝｻｲｷｾﾂ  . 00046         PAR CRB : INT ;                        /* ｺｱ ﾚﾌｧﾚﾝｽ             */ PR000240
ﾚﾝｻｲｷｾﾂ  . 00047  C                                                                        PR000245
ﾚﾝｻｲｷｾﾂ  . 00048         IF RETUBT (CRB) .NE. 0                                            PR000250  1
ﾚﾝｻｲｷｾﾂ  . 00049            THEN  % IF OUTPUT                                              PR000255  .2
ﾚﾝｻｲｷｾﾂ  . 00050                        IS ASR THEN ASR = (RETUBT (CRB)) % ﾋｮｳｼﾞｽﾙ ;        PR000260  ..
ﾚﾝｻｲｷｾﾂ  . 00051                        IS CRT THEN CRT = (RETUBT (CRB)) % ﾋｮｳｼﾞｽﾙ ;        PR000265  ..
ﾚﾝｻｲｷｾﾂ  . 00052                        IS LP  THEN LP = (RETUBT (CRB)) % ﾋｮｳｼﾞｽﾙ  ;        PR000270  ..
ﾚﾝｻｲｷｾﾂ  . 00053                    END ;                                                  PR000275  .2
ﾚﾝｻｲｷｾﾂ  . 00054         END ;                                                             PR000280  1
ﾚﾝｻｲｷｾﾂ  . 00055         RETURN ;                                                          PR000285
ﾚﾝｻｲｷｾﾂ  . 00056      END ｱﾗｰﾑ ;                                                          PR000290
ﾚﾝｻｲｷｾﾂ  . 00057   END ﾚﾝｻｲｷｾﾂ ;
```

図３．４　処理モジュールのプログラム例

モリ属性のデフォルト値なので通常は明示的に記述する必要はない。最後の２３～２５行目の宣言ユニットでは，％で始まる変数宣言文により，コンパイル時変数が宣言されている。これはコンパイル時実行文で用いられるが，詳細は4.6節で述べる。なお，本モジュールでは，用途別に４個の宣言ユニットを導入したが，一般に１つの宣言ユニットの中に変数宣言，型宣言，定数宣言等が混在しても良い。

次に，図3.4の処理モジュール「レッツイセキ」は，図3.3(c)の環境モジュール「ツイセキ」を親環境とし，３個の手続きを定義している。ここでも段階的詳細化技法が適用され，まず，図の行番号の２１～２４行目で最初の手続き「ＴＲＡＩＮ　ツイセキ　ショリ」を定義している。２１行目のＭＡＩＮはこの手続き属性が主プログラムであることを指定している。また，手続き定義の先頭のキーワードＦＵＮＣＴＩＯＮは，「機能単位」という意味で用いており，その手続き名が値を返すか否かに依らない。この最初の手続きでは，処理モジュール「ツイセキ」の機能を２分割し，各々の処理を行う次の２個の手続き引用文を含んでいる。

(1)　シンゴウキ　ノ　ヘンカ　デ　レッシヤ　オ　ツイセキ　スル

(2)　キドウカイロ　ノ　ヘンカ　デ　レッシヤ　オ　ツイセキ　スル

このように手続き名は複数の単語の並びで表現でき，英字の代りにカタカナを用いることもできる。但し，外部手続きになるものは，オペレーティングシステムの制約のため，先頭の単語にカタカナは使えない。例えば，本例の最初の手続き「ＴＲＡＩＮ　ツイセキ　ショリ」は主プログラム属性のため，先頭の単語はカタカナを用いていない。

ここで引用された２個の手続きのうち，２番目のものは同じ処理モジュールの２６～４３行目で定義している。２６行目の手続き属性ＯＰＥＮは，この手続きがその引用部分にインライン展開されることを示している。本機能は，段階的詳細化技法を適用して開発されたプログラムの中で，１箇所でのみ引用されるような手続きに用いることにより，その実行効率を良くすることができる。

この手続きの３８行目で引用した手続き「アラーム　シュツリョク」は，すぐ下の４５～５６行目で定義している。その中の４９行目で用いられている％ＩＦ文はコンパイル時実行文である。この場合，図3.3(c)の２４行目で宣言したコンパイル時変数ＯＵＴＰＵＴの値に応じて，５０～５２行目の３個のＴＨＥＮ節のうちのいずれかが選択され，言語処理系の出力であるオブジェクトコードとして生成される。

以上，列車運行管理システムを例にとり，ＳＰＬの概要について述べたが，詳細については次章で述べる。また構文規則は付録２に譲る。

3.5　結　　言

プログラミング言語の設計においては，基本的プログラム構造の設定が最も重要であり，具体的な言語機能はこの基本構造の上に実現される。本章では，従来方式の問題点から導かれた新言語への要求を明らかにし，それを満たす基本言語構造として，従来のブロック構造を変形した木構造形式のモ

ジュール階層構造を導入した。特に制御用応用プログラムの中で重要な役割を果たすタスク間共通データに関して，データアクセス権の制約とデータ抽象化を統一的に実現するため，データおよび手続きの名前の有効範囲の規則を次のように異なるものとした。

(1) データ名の有効範囲はブロック構造の規則に準じる。

(2) 手続き名の有効範囲は制約しない。

この方法により，新言語への要求を満たすことはできたが，共通データへのアクセス誤りを完全に防止できるわけではない。特に手続き名の有効範囲に制約を設けていないため，共通データ操作手続きの引用誤りの可能性が残されている。これを解決する方法としては，その手続きを引用して良いモジュールを明示的に宣言させる方法[W10]があるが，言語仕様が複雑になり，プログラムの記述が難しくなる。この問題の実用的な解決方法は今後の課題である。