JAERI-M
9719

FORTRAN構文解析プログラム SCANの機能と構造

1981年10月

富山　峯秀・浅井　清

日本原子力研究所
Japan Atomic Energy Research Institute

FORTRAN構文解析プログラムSCANの機能と構造

日本原子力研究所東海研究所計算センター
富山 峯秀・浅井 清

（1981年9月11日受理）

SCANはFORTRAN構文解析プログラムであり，解析の対象となるFORTRANプログラムを1ステートメントずつ読み込み，文法のチェックを行ない，内部コード表現のステートメントに変換するものである。SCANが扱うFORTRANの言語レベルはJIS－7000を基本とし，これにいくつかのFORTRAN Ⅳ－Hの機能を追加したものである。

本報告書ではSCANのプログラム構造と構文の解析方法，解析の過程で作成されるテーブル類について説明している。また解析の結果作成される内部コード表現化されたFORTRAN文について述べている。

SCAN 自身は約5000枚のFORTRANで記述されている。FORTRANの構文解析を必要とする他の応用に対しては，SCANを少し変更することによりうまく適用できると考えている。

SCAN: A Fortran Syntax Analyzer - Its Structure and Facilities.

Mineyoshi TOMIYAMA and Kiyoshi ASAI

Computing Center,

Tokai Research Establishment, JAERI

(Received September 11, 1981)

SCAN is a computer program which analyzes the syntax of a Fortran program. It reads statements of a Fortran program, checks the grammatical validity of them, and produces tables of the analyzed results and intermediate codes for further use. SCAN recognizes the Fortran syntax of the Japan Industrial Standards-7000, plus some Fortran-H statements.

In this report, the structure of SCAN, the methods used by the SCAN to analyze statements, tables and intermediate Buckus form texts produced by the SCAN, are presented. The SCAN itself is also written in Fortran language and consists of about 5000 statements. By slight modifications the SCAN may be useful for any application which needs analysis operations of Fortran syntax.

Keywords; Computer, Fortran, Analyzer, Syntax Checker, Preprocessor, Application

目　　次

Contents

1. はじめに

1.1 経緯

FORTRANやPL／Iなど高級プログラミング言語はコンパイラにより計算機語（マシンコード）に変換される。コンパイラはソース・プログラムを解釈してマシンコードを作り出すもので字句解析部，構文解析部，コード生成部，最適化部などから構成されている。筆者らの作成したSCANはコンパイラの字句解析部及び構文解析部に相当するもので，入力となるFORTRANプログラムを解釈し，各種テーブルを作成するものである。

筆者らは当初コンパイラを作成する予定であった。というのは，当時のコンパイラは現在のように発達して居らず，同じFORTRANコンパイラでもメーカにより，またコンパイラどうしでも機能やレベルが異なっていた。このため他の計算機を利用する場合はしばしばプログラムの一部手直しが必要であった。特に計算機機種交換においてはプログラマは計算コードの移行に多くの時間を費さなければならなかった。コンパイラの機能，レベルを常に変えないためには原研独自のコンパイラを作成する必要があった。計算機が交換された場合には原研コンパイラのうちのコード生成部を変更する。こうすることによりユーザのプログラムの変更なしに計算機間の差を吸収することができると考えたからである。

コンパイラは頻繁に使用されるものであるから，無駄のないようにマシンコードに近いアセンブラ言語等で記述されたものでなければならない。他方，計算機交換ごとにコンパイラの修正が必要となる点から，コンパイラは後日解読しやすい高級言語で記述されている必要がある。FORTRANの構文解析プログラムSCANはGPL（Genken Programming Language）[1,2]で記述されていた。GPLは筆者らが開発したシステム記述むきの言語でALGOLに似た言語である。GPLで記述されているプログラムはGPLコンパイラにより機械語に変換される。変換生成された機械語はアセンブラで記述されたものと語数比較するとIBM7044のSCANの場合で約10％程度しか増加しない。原研用FORTRANコンパイラはGPLを用いて作成にはいった。

しかし原研用FORTRANコンパイラ作成中において，JIS・FORTRANの発表による言語の統一，機能の向上，計算機の大型化などにより，SCAN開発の段階で作成を打切った。ただしSCANそのものはFORTRAN文を解釈しようという場合には不可欠なものであり，それ自身単独で利用が可能である。この度SCANをGPLからFORTRANに書き換えたのを機にその構造を明らかにし，一般の用に供するものである。この報告書は4章からなり第1章ではこのあとSCANの機能と構造の概略，及び応用例について記した，第2章においては内部構造について述べている。第3章ではソース・プログラムが内部コードに変換された場合の形式について例を掲げて述べてある。本報告書がFORTRANの構文解析ルーチンを必要とする方に少しでも役に立てば幸いである。

1.2 SCANの機能と構造の概略

SCANはFORAN解釈プログラムで，入力となるFORTRANプログラムをサブプログラム単位で一文づつ文法チェックと解釈を行ない，その結果をテーブルにして出力する。

出力されるテーブルは次のようなものである。

・内部コード・テーブル

プログラムを内部コードで表現したもの

・SYMBOLテーブル

プログラム中に現われた変数名，定数，文番号の属性を記録してある。

・ARRAYテーブル

宣言文で領域定義した変数の領域の大きさを記録してある。

・DO管理テーブル

DOのネストを管理するテーブル

・文番号管理テーブル

プログラム中に現われる文番号の参照関係を管理するプログラム。

現在取扱えるFORTRANの文法の範囲はJIS−FORTRAN7000及びFORTRAN−HEの一部を包含している。さらに宣言文においては多少拡張してある。

構文解析方式の概略を図1.1に示す。

PFCは系全体を制御する部分であり，初期設定や入出力の制御を行なう。初期設定の済んだ時点で解釈すべきFORTRAN文を読み込み文の構成を行なう。この時点で継続行は認識され，注釈文は除かれる。一文が構成されるとサブルーチンSELECTに制御が渡される。このサブルーチンは文の先頭文字をキーとして解釈すべき文をそれぞれの解釈ルーチン，例えば先頭文字がAの場合はSELECTはASSIGN文解釈ルーチンに制御を渡しASSIGN文であるか否かを判定する。ASSIGN文である場合は文法チェックを行ない内部コード・テーブルを作成する。そうでなければ代入文解釈ルーチンARITHMに制御を渡し代入としての処理を行なう。処理が済むと制御はPECに戻される。この操作がプログラム単位に含まれる文の数だけ操返される。END行の処理が済むとPFCは作成されたテーブル類を出力し，次のサブプログラムの処理のための初期設定を行なう。文法解釈の途中で発見されたエラーはその都度出力される。

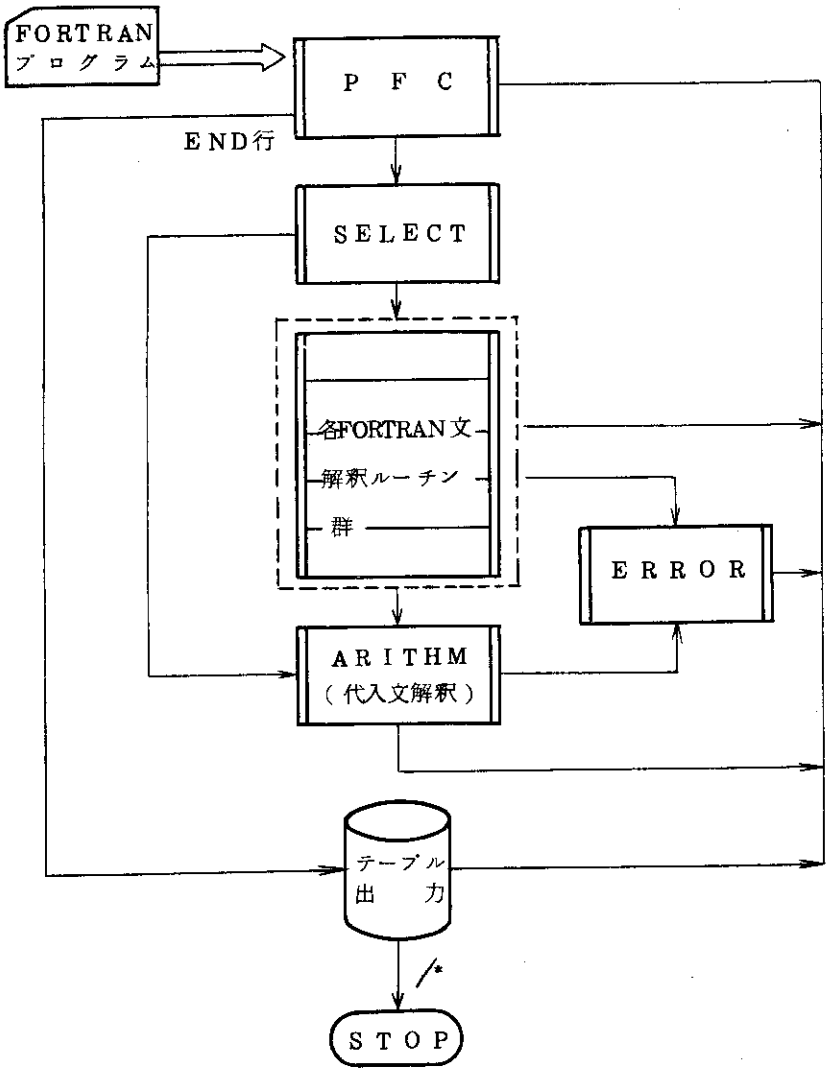


図1.1　　SCANの構文解析方式概略図

1.3　SCANの使用例

SCANはFORTRANプログラムを解釈し内部コードへの変換および各種テーブルを作成するものであるから，これを直接または一部変更することにより様々な適用が考えられる。筆者らが作成しようとしたFORTRANコンパイラはSCANの出力する内部コードから計算機の命令を作成するものである。それ以外の応用としてここでは出力されるテーブル類を利用した例(SOPHY)，SCANの構造を一部変更して利用した例(DATAPOOL)，及びその他の例(TSSFORT)，を紹介する。

1.3.1　SOPHY

近年開発されるプログラムは増大し，またそれらが大規模化，複雑化している。ソフトウェア工学の分野から開発されるプログラムについての生産性，開発時間などについていくつかの

研究が発表されている。その中の1つにプログラム中のオペレイタ（四則演算子，IF，DO等）とオペランド（変数，定数等）の数からプログラムの開発時間等を推定する方法が発表されている。これはM．Halstead[3,4]が発表した理論で，これに興味をもった筆者らはいくつかの実験を行なった。以下にその理論を簡単に紹介する。

H．Halsteadはプログラミング作業の生産性に関し，Software Scienceと呼ぶ理論を発表している。この理論は生産されたプログラムから，そのプログラムを構成するオペレイタとオペランドの数を読み取り，その値をもとにそのプログラムを生産するのに要したプログラムの時間を推定するものである。この理論は以下の計算式で示される。

n_1：ひとつのプログラム中に現われた異なったオペレイタの数

n_2：ひとつのプログラム中に現われた異なったオペランドの数

N_1：ひとつのプログラム中に現われたオペレイタの出現回数の総和

N_2：ひとつのプログラム中に現われたオペランドの出現回数の総和

とするとき

Vocabulary n：　$n = n_1 + n_2$

プログラム長　N：　$N = N_1 + N_2$

Volume　V：　$V = N \log_2 n$

プログラム・レベル　L：　$L = (2 / n_1)(n_2 / N)$

言語レベル　λ：　$\lambda = L^2 V$

プログラム作成に要する努力（effort）　E：　$E = V / L$

プログラミング作成に要する時間　T：　$T = E / S$

（ここでSはプログラマが1秒間に認識できるプログラム要素の数）。

以上の計算に必要なデータはn_1，n_2，N_1，N_2及びSである。SOPHYはSCANの出力した内部コードテーブルとSYMBOLテーブルからオペレイタ，オペランドを数え上げ上の計算を行なっている。Sは別に与えられるがHalsteadの採用した値18を使用した。

Tの値からソフトウェアの生産性が判断できる。しかし我々の実験ではこの理論は数十枚程度のプログラムではよく合うが大きなプログラムでは合わなかった。現在のところこの理論の有効性は不明である。詳細については第21回プログラミング・シンポジウム報告集「科学技術計算におけるソフトウェアの生産性について」[5]を参照されたい。

1.3.2　DATAPOOL

SCANからはその一部の改造，追加によってFORTRAN言語に密接したプリプロセッサを容易に作成することができる。ここではデータプール用に作成したプリプロセッサについて述べる。データプールは次のような考えのもとに作成された。

一般に科学技術計算のデータ・ファイルは計算の状況によりデータ長や書き込み順序が変化する割合がほとんどである。このため作成済のデータ・ファイルを使用する場合関連するプログラムの調査を，その入力データまで含め必要となることが多い。すなわちデータ・ファイルは計算プログラムからは独立ではない。一方当研究所においては大量のデータを扱う大型プログラムが多数使用されており，研究者はこれらプログラム間のデータの整合作業に多大の時間

を費してきた。ＦＯＲＴＲＡＮプログラムとそのデータ・ファイルのもつ難点を解消し，データ・ファイルのモジュラリティを高め，データの標準化と共用を容易にすることがデータ・プール開発の目的である。

この目的に合致するデータ・プール・ファイルのもつべき機能を，

・樹状階層構造の命名

ファイルに格納するデータに樹状階層構造の名前をつけることができ，その名前の指定によりデータの入出力が行なえる，

・データ及びデータ属性の保存

格納されるデータについてその変数名，型，配列の大きさ，入出力型式なども付随して格納される，

・注釈文の保存

データの内容，用途，使用上の注意などの注釈文を格納しておくことができる，

とした。さらにこのファイルに対して，ファイル処理ユーティリティを用意し

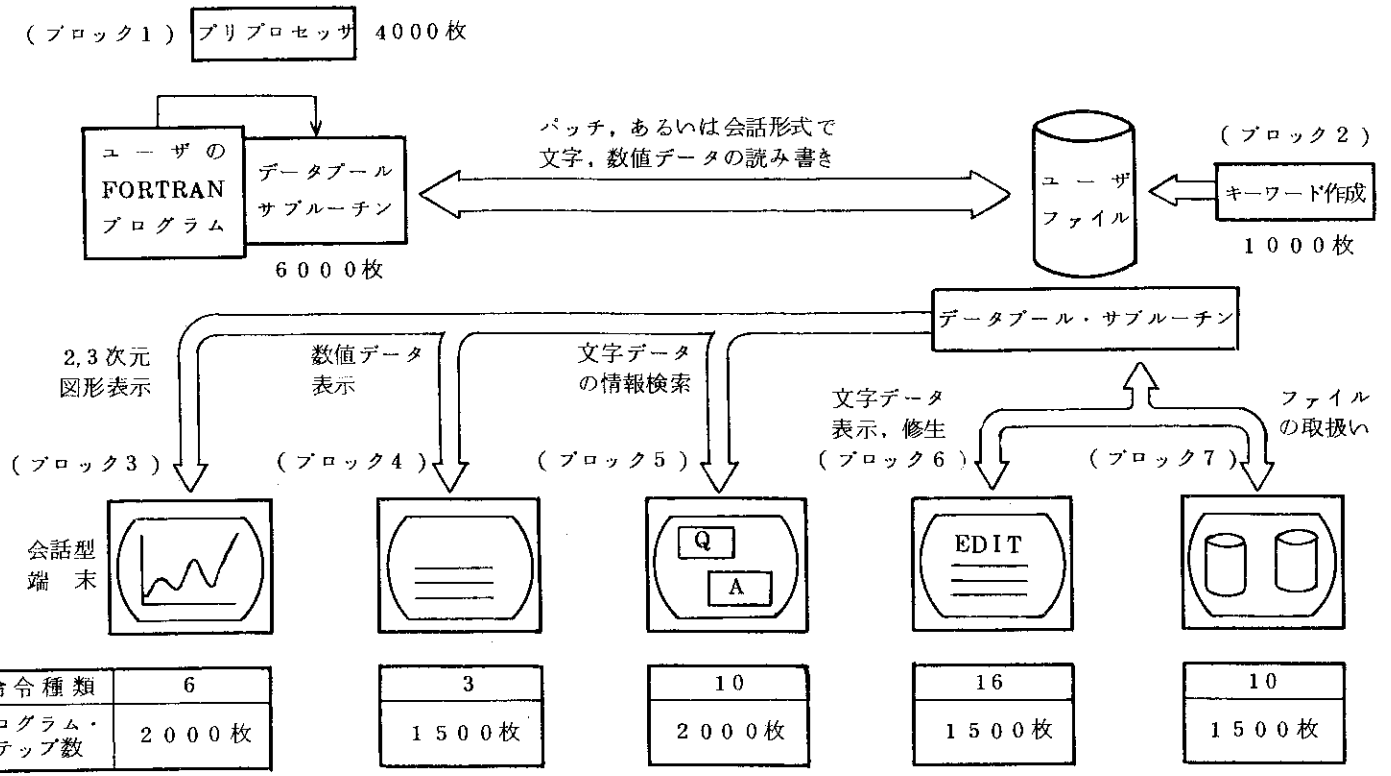
・データおよび属性の検索，表示，修生，消去，保管と管理

などの操作がＴＳＳ用ユーティリティまたは計算コードにおいて容易に行なえるようになっている。データプールの体系を図1.2に示す。

ファイル処理ユーティリティは計算コードの中から呼び出される。呼び出し方には二通りの方法がある。１つは直接ＣＡＬＬ文による方法である。この方法は簡単ではあるが引数が多くなったりＣＡＬＬ文が多くなると誤りを起しやすい。またある概念を記述するうえでＣＡＬＬ文は不適当である。他の１つは概念を記述しやすい文法を新たに定め，これに従って計算コードの中に，例えばデータ入出力等の操作のための文を書く。新しい文法に従って記述された計算コードはプリプロセッサにより通常のＦＯＲＴＲＡＮプログラムに変換するという方式である。この方式は多少面倒ではあるが概念が明確であることからプログラムのデバッグ，後日のメンテナンスに有利である。データプールでは後者を採用している。データプールのためのＦＯＲＴＲＡＮ拡張文をＣＪ文と呼び，表1.1に示す。

この文法に従って計算コード中にデータプール入出力文を記述した例と，それをプリプロセッサにより通常のＦＯＲＴＲＡＮ文に変換した例を図1.3，図1.4に示す。

このプリプロセッサはＳＣＡＮにＣＪ文の解釈部，展開部を付加したものである。



命令種類	6	3	10	16	10
プログラム・ステップ数	2000枚	1500枚	2000枚	1500枚	1500枚

注) プログラムはすべてFORTRANで書かれている。プログラム・ステップ数は実行文のみで，コメント文は含まない。

図1.2　データプールの体系

表1.1　FORTRAN 拡張文

区分	FORTAN拡張文名	パラメータ	機能概要
定義文	FD	Fnn＝データプール名，LRECL＝レコードサイズ／{L/E/U}，MAXRCD＝最大レコード数〔，MODE＝{▼OLD▼/▼NEW▼}，OWNER＝所有者名，DIRECT＝ディレクトリの個数〕〔，MACHINE＝計算機名／{WORD/BYTE}〕〔，DEVICE＝装置名／レコードギャップ／1トラック当りのバイト数〕〕	使用するデータプールの属性を定義する。
入出力文	WRITE	（{データ名/*/#}〔，P＝優先権〕〔，EXT＝展開形式〕〔，FMT＝書式名〕〔，ERR＝文番号〕）　出力並び	データプールにデータを格納する。
	READ	（{データ名/*/#}〔，ERR＝文番号〕）　入力並び	データプールからデータを読む。
	COMMW	（{データ名/*/#}〔，P＝優先権〕〔，ERR＝文番号〕）{▼文字列▼/n，変数名}	注釈文を格納する。
	COMMR	（{データ名/*/#}〔，ERR＝文番号〕）　n，変数名	格納されている注釈文を読む。
	SEARCH	（データ名，データ長〔，ERR＝文番号〕）	指定されたノード名をさがす。
ノード位置設定文	POINT	（データ名，D_{max}，D，v，rc）	ノード名探索の開始点を指定する。
	NEXTH		ノード位置設定ポインタを水平方向右側に移動する。
	NEXTV		ノード位置設定ポインタを下方向左端に移動する。
	PRIORH		ノード位置設定ポインタを水平方向左側に移動する。

区分	FORTRAN 拡張文名	パラメータ	機能概要
ノード位置設定文	PRIORV		ノード位置設定ポインタを上のノードに移動する。
	CONNECT	(データ名1，データ名2，RC)	二つのデータ名を結びつける。
	DISCON	(データ名1，データ名2，RC)	CONNECT文で作られた結びつきを解く。
	RNEXT		ノード位置設定ポインタを現在の位置からCONNECT文で結びつけられているノードに移す。
	FIND		ノード位置設定ポインタが示しているノードのノード名を知る。
	DNEXT	(変数名)	一次元インデックス位置設定ポインタを現在の位置から次の位置へ移す。
保守管理文	COMDENSE	(データプール名，WF=nn)	データプール内の各所に散在する使用不可領域を集め，使用可能領域とする。
	SORT	(ノード名〔，LEVEL={i / ALL}〕〔，WF=nn〕〔，SR=m〕)	指定されたレベルのノード名を並べかえる。
	RECOVER	(データプール名〔，TYPE={W, WF=nn / S, WF=nn / C}〕)	使用不可となったデータプールを再使用可能にする。
宣言文	OPTION	〔BUFFER=作業用領域サイズ／{L / E / U}，〕〔LRUSIZE={30 / LRUテーブルサイズ}，〕〔MAXUDP=最大使用データプール数，〕〔QMAIN={CREATE / NOCRATE}〕	データプール・アクセス・ルーチンの動作環境を設定する。
	SYN	v1=データ名1〔，v2=データ名2，……〕	データ名の同義語を定義する。
	ALIAS	v1=予約語1〔，v2=予約語2〕	予約語を変更する。

```
      SUBROUTINE DPW(A,B,C,D,X,N)
      DOUBLEPRECISION A,B,C,D
      DIMENSION X(100)
CJ    WRITE(A.B.C.D,ERR=200) (X(I),I=1,N)
      RETURN
  200 STOP
      END
```

図1.3　C J 文の使用例

```
      SUBROUTINE DPW(A,B,C,D,X,N)
      DOUBLEPRECISION A,B,C,D
      DIMENSION X(100)
C=====================================================================
CJ    WRITE(A.B.C.D,ERR=200) (X(I),I=1,N)
C=====================================================================
      ILNGTH=(1*(N-1+1)+0)
      CALL QOPEN(4,1,&200,2,1,ILNGTH,IUNIT,IRECRD,0,4,1,0,A,1,0,B,1,0,C,
     11,0,D)
      CALL QWRITE(IRETCD,1,&200,0,12,'(X(I),I=1,N)',4,500,100,400,N)
      WRITE(IUNIT'IRECRD,ERR=200)(X(I),I=1,N)
      CALL QCLOSE(IRETCD)
C=====================================================================
      RETURN
  200 STOP
      END
```

図1.4　プリプロセッサによるC J文の展開例

1.3.3　T S S F O R T

プログラムの開発においてはコーディングよりはむしろその大半の時間をデバック作業に費している。通常ディバックにはチエックのためのW R I T E文を入れたりデバッグパケットを利用して行なわれている。これらの方法では大規模なプログラムの開発と保守においては多大な時間を要する。デバックにはプログラムの実行に合せたダイナミックなデバックが効率的であると考える。そのためにはプログラムの完成された部分は高速に実行され，未完の部分は人間の思考過程に合せゆっくり実行され，また任意の点で実行を停止させて変数の内容の確認や変更，プログラムの変考などが行なえる機能が必要である。人間，計算機の動作にはそれぞれ特徴があり，ソフトウェアの開発にはそれらが活かせることが望ましい。筆者らは１０年前人

間と計算機のそれぞれの長所を活かし，短所を補い人間の思考課程に合せ人間と計算機が一体となってソフトウェアの開発ができるシステムを設計し開発を試みようとした。

ＴＳＳＦＯＲＴはプログラムの開発，保守を効率よく行なうための処理プログラムであり，次のような機能がが考えられた。

1) 会話型式で操作する。

2) テキスト編集の機能をもつ。

3) ＦＯＲＴＲＡＮ文法のチェックが行なえる。

4) ユーザコントロールのもとで実行できる。

a. １ステップごとの実行，指定された区間の実行。

b. 任意の変数の内容が表示，変更ができる。

c. セミルーチンの実行。

d. 任意の点から実行を継続できる。

5) 完成部分は高速に実行する。

この処理プログラムはＦＯＲＴＲＡＮコンパイラ，リンケージエディタをダイナミックに結合し，実行形式プログラムを積極的に利用していくものである。ＴＳＳＦＯＲＴの構造を図1.6に示す。ただしこのうち作成されたのは前述結局ＳＣＡＮだけである。

ただし，このうちＳＣＡＮのみが実現されている。以下にＴＳＳＦＯＲＴの考え方のあらましを述べておく。

ＣＯＮＴＯＬＬＯＲは端末とのインタフェースであり，入力されるコマンドを解釈して系全体を制御するルーチンである。ＥＤＩＴＯＲはソース・ファイルにあるソース・プログラムをテキスト編集するもので，ソース・ファイル上のテキストを作業用ファイルに移し，さらにその一部（指定された部分）をメモリ上の作業域に移しここでテキスト編集の作業を行なう。ＳＣＡＮはソース・ファイル端末からの入力，または作業域にあるテキストを読み込み，メモリ上の作業域ＷＡに内部表現のコードを頁単位で作成する。ＷＡで作成された内部表現コードは内部ソース・ファイルに保存される。ＥＸＥＣＵＴＥＲは内部表現ソース・ファイルからＷＡ経由で擬似バイナリ（ＰＢ）形式プログラム・エリアＰＢＡにプログラムを読み込み，実行させる。ＰＢＡにはサブルーチン単位でプログラムが作成され実行後は大記憶上のＰＢＡに退避される。次にそのサブルーチンが参照されると大記憶からメモリ上のＰＢＡにコピーされる。ＬＩＮＫＥＲはＰＲＥＰＲＯ．ＦＯＲＴＲＡＮ，ＬＩＥＤをコールしそれぞれの作業をさせる。またＬＩＥＤにより作成され，ダイナミック・リンク形式のサブルーチンが格納される実行形式ファイルＥＢからメモリ上の領域ＬＤＭＯＤにロードする。ＥＸＥＣＵＴＥＲはこれを実行させる。ＰＲＥＰＲＯは内部表現ソース・ファイルの内容をカード・イメージのテキストに変換し，作業用ファイルＳに格納する。ＦＯＲＴＲＡＮはソース・ファイル上またはＳ上のソース・ファイルをコンパイルし相対形式バイナリ・プログラムを作り出す。ＬＩＥＤはこのＲＢをダイナミック・リンク形式の実行形式プログラムを作成する。

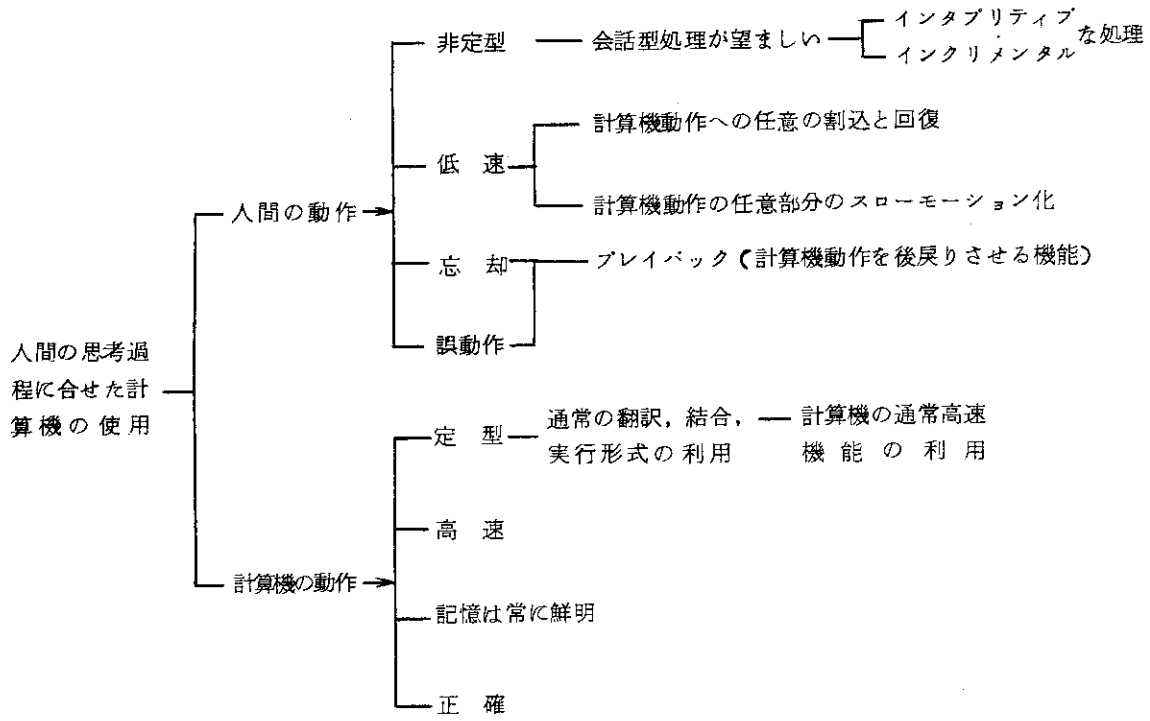


図1.5　人間と機械の動作

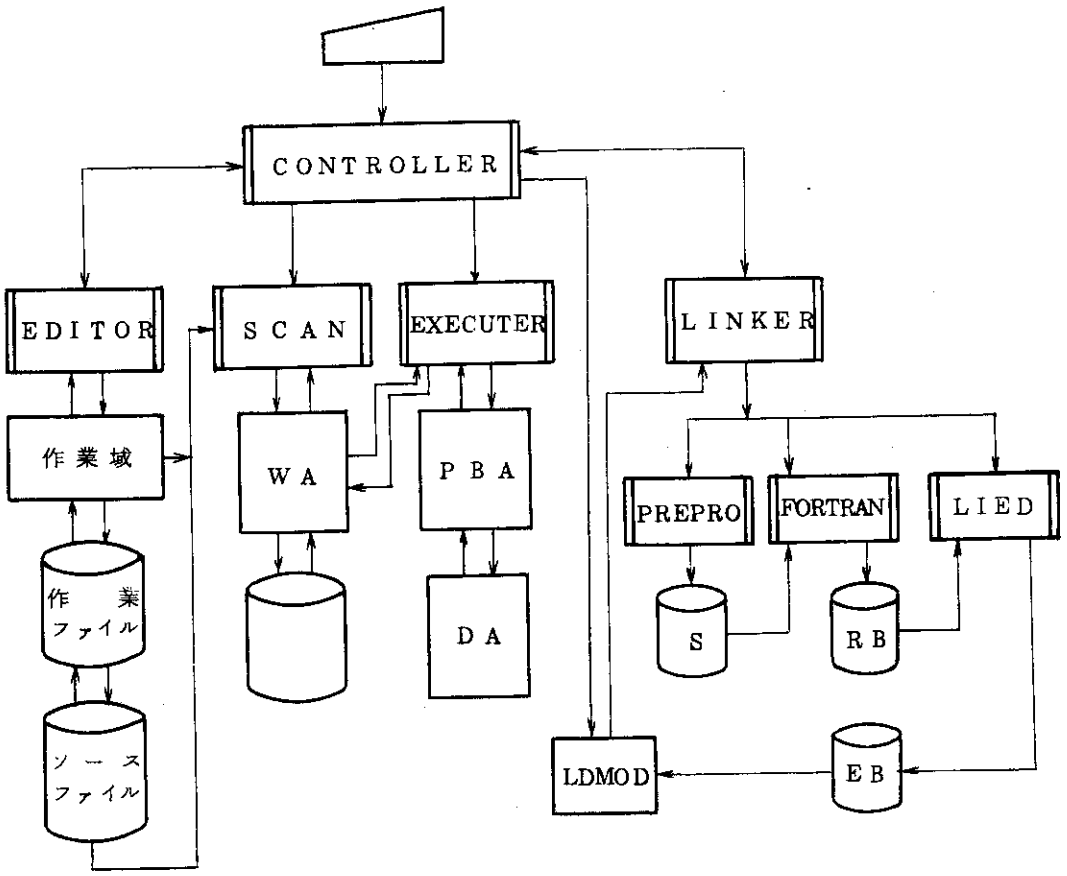


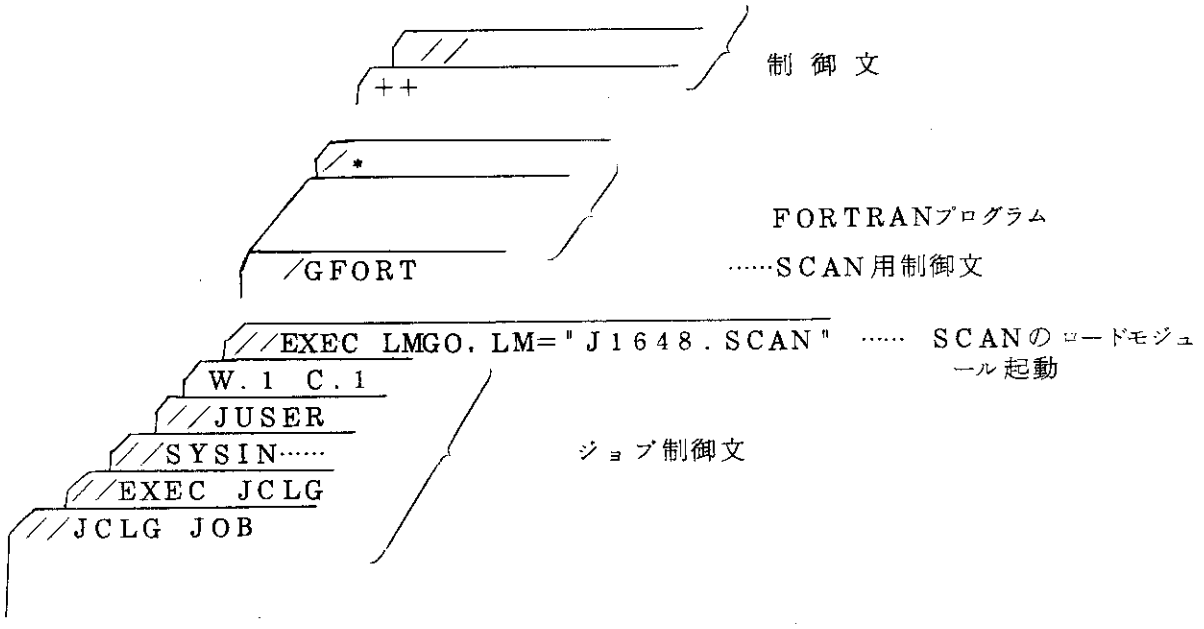
図1.6　TSSFORTの体系

2. ＳＣＡＮの内部構造

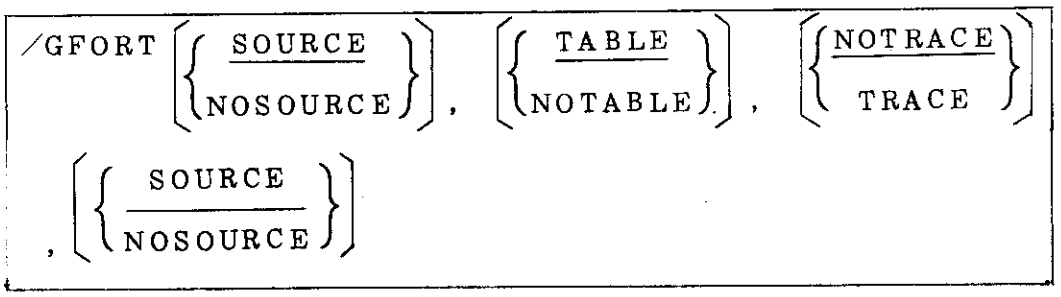
この章ではＳＣＡＮがＦＯＲＴＲＡＮ文解釈のために用いる処理ルーチン，領域，解釈方式，作成されるテーブル等について説明する。図 2.1 に処理の骨格となる部分を示す。

各処理ルーチンは多くの補助ルーチンを呼び出すが，ここでは繁雑さを避けるため省略してある。

解釈されるべきＦＯＲＴＲＡＮプログラムは次の形式で与える。ただし制御情報となる／ＧＦＯＲＴ文と／＊文は省略してもよい。



／ＧＦＯＲＴ文はＳＣＡＮへの制御情報を与えるもので次の形式をしている。



ＳＯＵＲＣＥは入力となったＦＯＲＴＲＡＮプログラムのリストの出力を指定し，ＮＯＳ－ＯＵＲＣＥは出力しないことを指定する。ＮＯＳＯＵＲＣＥを指定してもエラーメッセージは出力される。このパラメータを省略するとＳＯＵＲＣＥが指定されたものとみなす。

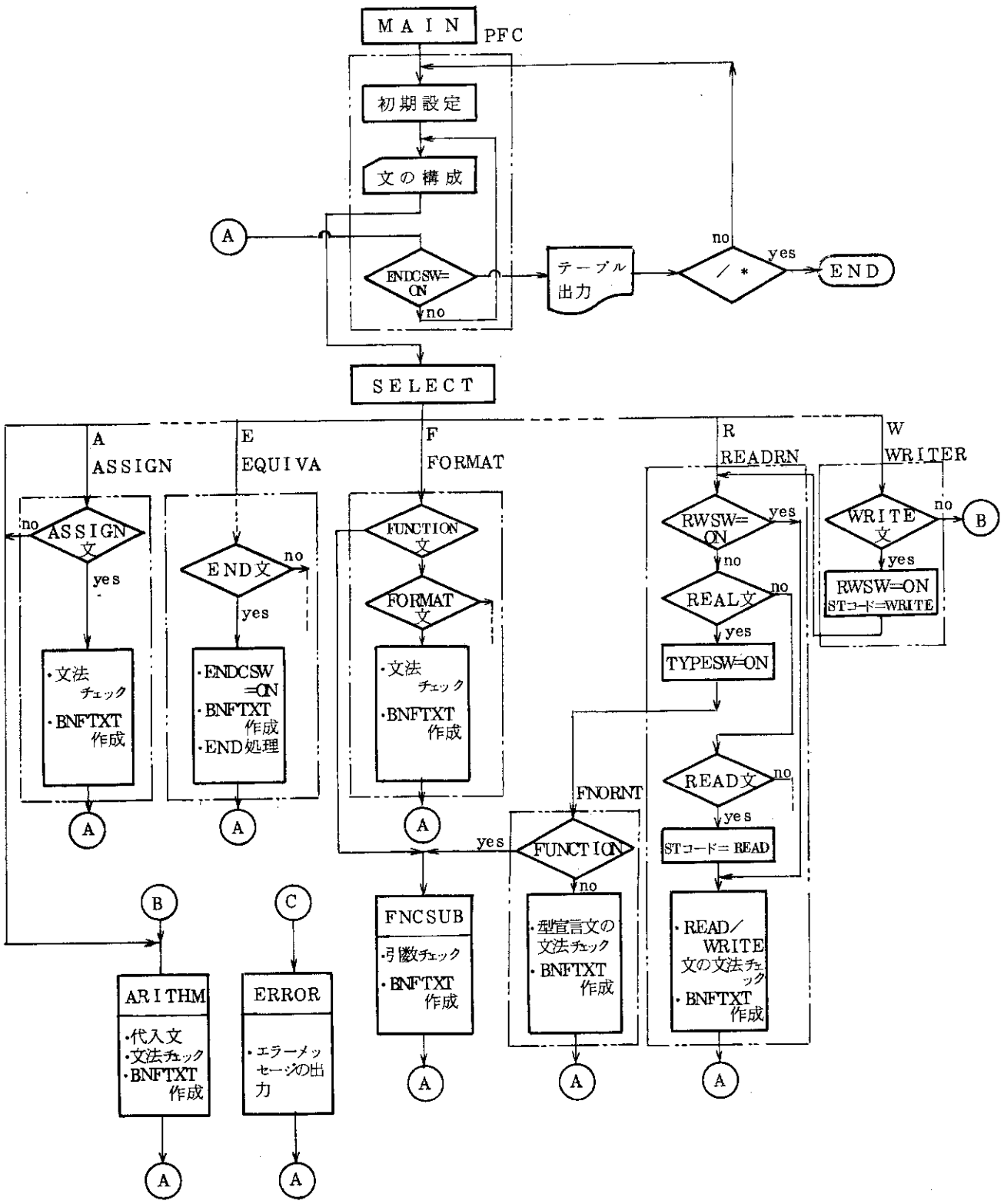


図 2.1　　SCANの構造概略

$\left[\left\{\begin{array}{c}\text{NOTABLE}\\ \hline \text{TABLE}\end{array}\right\}\right]$

ＴＡＢＬＥは作成されたテーブルの内容をプリンタにも出力することを指定し，NOTABLEは出力しないことを指定する。テーブルの内容は１６進数で表示される。このパラメータが省略されるとＮＯＴＡＢＬＥが指定されたものとみなす。

$\left[\left\{\begin{array}{c}\text{NOTRACE}\\ \hline \text{TRACE}\end{array}\right\}\right]$

ＳＣＡＮデバッグ用情報出力の指定を行なうもので，ＴＲＡＣＥはデバッグ情報の出力を指定し，ＮＯＴＲＡＣＥは出力しないことを指定する。このパラメータが省略されるとNOTRACEが指定されたものとみなす。

図2.1においてプログラムＭＡＩＮは／ＧＦＯＲＴ文の解釈を行なった後制御をＰＦＣに渡す。ＰＦＣはＳＣＡＮ全体を制御するサブルーチンであり，テーブルや各種スイッチ，インデックス等の初期設定を行なう。次に継続行を考慮して一文の構成を行ないＳＥＬＥＣＴに制御を渡す。ＳＥＬＥＣＴでは文番号の処理をした後，文の第１文字から，いずれかの文解釈ルーチンを選択し制御を渡す。各解釈ルーチンでは入力された文がそのルーチンで処理すべき文か否かをチェックする。該当する文であれば文法チェックを行ない内部コードを作成する。また該当しなければ算術文解釈ルーチンＡＲＩＴＨＭに制御を渡し，算術文としての処理を行なう。解釈が終了すると制御はＰＦＣに戻る。解釈中に文法エラーを発見した場合はエラー処理ルーチンＥＲＲＯＲに制御が渡され，エラー処理終了後制御は再びＰＦＣに戻される。ＥＮＤ行が処理されるとＰＦＣはそこまでを一区切としてテーブルの出力を行なう。以下にこれらの処理について述べる。

2.1　ＳＣＡＮで用いるＣＯＭＭＯＮ領域と制御変数

ＳＣＡＮではプログラム制御用のスイッチやインデックス，各種テーブルのための領域など２９のＣＯＭＭＯＮ領域が定義してある。ここでは主な領域について説明しておく。

ＣＯＭＭＯＮ／ＡＲＥＡＣ／

内部コードを格納するための領域。

・ＣＥＬＬ（２０００）　：　１０語／１セル

ＣＯＭＭＯＮ／ＡＲＥＡＤ／

Ｄｉｍｅｎｓｉｏｎを定義されている変数の次元数とサイズを記録するための領域。

・ＡＲＹＴＢ（４００）　：　可変長

ＣＯＭＭＯＮ／ＡＲＥＡＦ／

文関数が定義された場合のパラメータを格納するための領域。

・ＦＰＴ（１００）　：　１語／１セル

ＣＯＭＭＯＮ＝ＡＲＥＡＨ／

ハッシュテーブル用の領域。

・ＨＴＤＩＣＴ（１０００）

COMMON／AREAR／

定義された文番号が格納されて位置を格納する領域。

・RLT（100）

COMMON／AREAS／

プログラム中に現われた変数名，定数，文番号，関数名，サブルーチン名を登録する領域。

・SYMBOL（5000）　：　10語／1セル

COMMON／BLOCKN／

プログラム中に現われたコモンブロック名を記録する領域。

・BLOCKN（500）　：　5語／1セル

COMMON／COMARY／

サブルーチンSBNAMEで認識された変数名，定数，が格納される領域。

・NAME（2）

・DOSAVO（2）

COMMON／COMVAR／

SCANの用いる共通変数。

・DPOINT：SYMBOLテーブルへのポインタが格納される。

・IRET　：各ルーチンからのリターンコードが格納される。

・MODE　：現在処理中の変数名，定数の型が格納される。

・DELM　：現在処理中の区切記号の種類が格納される。

・CH　：現在処理中の文字，特殊文字が格納される。

・SAVXR1：インデックスX1の内容を保存する。

・INTNUM：現在処理中の変数名，定数のサイズが格納される。

・TEMP1　：作業領域。

・I　：作業領域。

COMMON／DHTTBL／

文字定数を格納しておくための領域。

・IDHT　：

・IXDHT　：DHTの使用可能位置を示す。

・DHT（2500）

COMMON／DOIX／

DOの定義を管理するための領域。

・DODPTH：定義中のDOの入れ子の深さを示す。

・DOCLSE：DOが閉じられたか否かを示すスイッチ，0なら閉じられていない。
1なら閉じられた。

・MXDPTH：DOの入れ子の最大深さを定義する。

・RANGE　：DOの入れ子の番号。

・MRANGE：DOの最大定義数。

・ＤＯＮＥＳＴ（４００）：ＤＯの管理テーブル。

ＣＯＭＭＯＮ／ＩＮＤＥＸ／

ＳＣＡＮで用いるインデックスをまとめて定義してある。

・ＩＸ(1)　：　ＩＳＹＭＢＬ　　ＳＹＭＢＯＬテーブルの使用可能位置を示す。
　ＩＸ(2)　：　ＩＢＮＦ　　ＢＮＦＴＸＴの使用可能位置を示す。
　ＩＸ(3)　：　ＩＲＬＴ　　ＲＬＴの使用可能位置を示す。
　ＩＸ(4)　：　ＩＦＰＴ　　文関数の引数管理テーブルの使用可能位置。
　ＩＸ(5)　：　ＩＡＲＹＴＢ　　ＡＲＹＴＢＬの使用可能位置。
　ＩＸ(6)　：　ＩＣＥＬＬ　　ＣＥＬＬの使用可能位置。
　ＩＸ(7)　：　ＬＩＭＩＴ
　ＩＸ(8)　：　ＮＢＬＯＣＫ
　ＩＸ(9)　：　ＮＸＣＥＬＬ
　ＩＸ(10)　：

ＣＯＭＭＯＮ／ＩＮＤＥＸＲ／

ＳＣＡＮがＧＰＬで書かれていたときのインデックスレジスタの定義・現在は，Ｘ１がＳＴＡＴＥＭのインデックスになっているがそれ以外は作業用領域として使用・主にアドレス計算に使用している。

ＣＯＭＭＯＮ／ＩＮＰＵＴ／

ＦＯＲＴＲＡＮ文の読み込み領域と文の構成領域（２.２参照）。

・ＲＡＲＥＡ（８０）
・ＳＴＡＴＥＮ（１３２６）

ＣＯＭＭＯＮ／ＩＰＣＴＴＢ／

Implicit 文で定義する型を格納する領域。

・ＩＰＣＴＭＤ（２９）

ＣＯＭＭＯＮ／ＩＳＮＤ／

内部発行文番号を記録しておく領域。

・ＩＳＮ

ＣＯＭＭＯＮ／ＳＩＺＥ／

主なテーブルのサイズを定義している。

・ＭＳＹＭＢＬ：ＳＹＭＢＯＬのサイズ。
・ＭＣＥＬＬ　：ＢＮＦＴＸＴのサイズ。
・ＭＲＬＴ　　：ＲＬＴのサイズ。
・ＭＦＰＴ　　：ＦＲＴのサイズ。
・ＭＨＤＩＣＴ：
・ＭＡＲＹＴＢ：ＡＲＲＡＹのサイズ。
・ＭＩＸ　　：ＩＮＤＥＸのサイズ。
・ＭＳＷ　　：ＳＷＩＴＣＨのサイズ。
・ＬＢＬＯＫＮ：

・MDONST：DOIXのサイズ。

・MLENGT：

COMMON／SWITCH／

SCANの中で使われているスイッチが定義されている。20語。

・SW(1)　：　ON

・SW(2)　：　OFF

・SW(3)　：　CALSW

・SW(4)　：　CTCDSW

・SW(5)　：　DCTSW

・SW(6)　：　DECLSW

・SW(7)　：　DPCXSW

・SW(8)　：　ENDCSW

・SW(9)　：　IFSW

・SW(10)　：　IOSW

・SW(11)　：　RELGSW

・SW(12)　：　SWCLO1

・SW(13)　：　DOSW

・SW(14)　：　SWENDO

・SW(15)　：　SFSW

・SW(16)　：　SUBSW

・SW(17)　：　TYPESW

・SW(18)　：　WSW

・SW(19)　：　STMNSW

・SW(20)　：　ARGUSW

COMMON／TABLE1／

変数名や特殊記号など識別コードが定義されている。

・V(74)

COMMON／TABLE2／

FORTRAN文の識別番号が定義してあるテーブル。

・SCLASS(47)

COMMON／TABLE3／

FORMATコードを定義してあるテーブル。

・FCODE(14)

COMMON／TABLE4／

動作テーブルを定義してあるテーブル。

・TMATRX(1000)

COMMON／TYPEON／

変数の型が定義されている。型は数値で認識される。

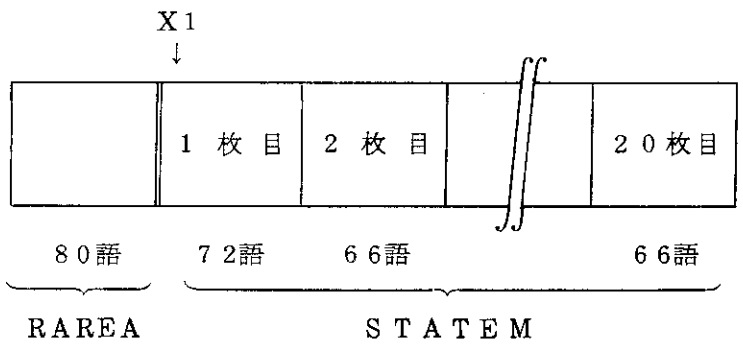
・REALTP　＝　1
・INTTYP　＝　2
・LOGTYP　＝　3
・DBLTYP　＝　4
・CPXTYP　＝　5
・CDTYP　＝　6
・QUDTYP　＝　7
・HOLTYP　＝　8

SCANで定義参照しているCOMMONとサブルーチンを表2.1に示す。

2.2　文単位の認識

通常FORTRNプログラムには注釈文が含まれており，また一文も継続行を含み複数の行から構成されている場合がある。注釈文を除き継続行を考慮して一文を構成するのはPFCで行なっている。文の構成はRAREAおよびSTATEM領域を用いて行なう。まず先頭の一行がRAREAに読み込まれ，注釈文でなければSTATEMに移される。ここでさらに次の一行を読み込み，それが注釈行でなければ継続行か否かをチェックする。継続行の場合はSTATEM中の前行の情報に続けてRAREAの情報を移す。さらに次の行を読み込むという様に繰返す。継続行は20行までとしている。継続行でない行，すなわち次の文がRAREAに読み込まれたら読み込みを中断し，STATEMに作成された一文の解釈を開始する。解釈が終了すると制御はPFCに戻される。ここで既にRAREAに読み込まれている行をSTATEMに移し，次の行を読み込むという手順で文が構成され，解釈されていく。RAREAおよびSTATEMは次のように定義されている。

COMMON／RAREA／RAREA(80)，STATEM(1326)



1文の終りには終端記号(ENDMRK)が記入されている。X1はSTATEMのカラム・カウンタである。

表2.1　SCANで用いているCOMMON領域と参照サブルーチン名

```
=ANALYSIS/77=      LISTINGS OF THE COMMON REFERENCE TABLE                DATE  1981/08/12(WEDENESDAY)     TIME 19:25:27  PAGE 0002
              ==== SUBPROGRAM NAME  VS.  COMMON BLOCK NAME ====
+-------------+-------------------------------------------------------------------------------------------------------------+
I*************]AAABB.CCCCC.DDDDD.DDEEE.EEEEF.FFMGH.IIIIL.LNNNN.NOOPP.PPPMR.RRSSS.SSSSS.SSSSS.BTTTU.WWWWI
I*************]RRSAN.ALDDD.EEIII.RWLNN.NQRRN.NOADD.BFNOB.DACMU.XCPAA.AAFAE.EEBBC.CCEEE.ETTUU.LAARN.DDLRI
I*************]ITSCF.LUDLM.BCCMM.ERMCD.TURRC.URITL.CRILL.GMMLM.CT CC.UUCIA.AWNTA.AVITT.TMUBB.DKKAL.SWIII
I*************]THIKT.KSECM.PPNCE.ATPPP.DI OS.RMNOT.DUGIN.ILDPN.HAIKK.PS ND.DPARL.NNBLU.UNPCP.CEECD.  STI
I*************]HXGSE.EECPD.RRRON.PPRRR.LV RU.NA RH. TEST.CSDRT.ALDNI.RE  C.RRMAA. TN  .DURHR.KLUEA.  TEI
I*************]MPNPX.LFVXN.DDYLS.RRDDD.TA  B.TT N . NRTR.LTEUR.R N  .DR   .NDECR. RF  . MNKD.D  MD.   TI
+-------------+-------------------------------------------------------------------------------------------------------------+
I   1./ADDRES/ IXXXX .  X X.    X.      . XX  .XX X . X X .      .  X  .XXX  .XX   .      .  X X.     .    I
I   2./AREAC / IX    X.      .     .     .     .     .     .     .     .  X  .     .  X  .     .     . X I
I   3./AREAD / I     .      .     .     .     .     .     .     .     .  X  .     .     .     .     . X I
I   4./AREAF / .XX   .      .     .     .     .     .     .     .     .  X  .     .     .     .     .    I
I   5./AREAH / ,     .      . X   .     .     .     .     .     .     .  X  .     .     .     .     .    I
I.............!.....................................................................................................I
I   6./AREAR / .     .      .     .     . X   .     .     X.    .     .  X  .     .X    .     .     . X I
I   7./AREAS / .XX   .  X X.  XXX.     .XX  X.X    . X  X. X   .     . XX  .X    . X   .  X X.     . X I
I   8./BLOCKT/ .     .      X.    .     .     .     .     .     .     .  X  .     .     .     .     .    I
I   9./CUMARY/ .XX   .    XX.  X X.     .X    . X XX. X  X.    X. X   .  X  .XXX  .X    . X   .     .    I
I  10./CUMVAR/ .XXXX .X XXX.  XXX.     .XX  X.XX XX. XX X.XX  X. XX  .XXX  .XXX  .XX   . XX X.   XX.    I
I.............!.....................................................................................................I
I  11./DDUNTE/ .X  X .     X.      .     .     .     . X   .     .     .  X  .     .     .     X.     .    I
I  12./DBLCUM/ .     .      .     .     .     .     .     .     .     .     .  X  .     .     .     .    I
I  13./DHTTBL/ .     .      .     .     .     . X  X.     .     .     .  X  .     .     .     .     .    I
I  14./DMYADD/ .X    .     X.     .     .     .     .     .     .     .     .     .     .     .     .    I
I  15./DDTX  / I     .      .    X.     .X    .     .     .     .     .  X  .     .X X  .     .     . X I
I.............!.....................................................................................................I
I  16./DDMCUM/ I     . X    .X    .XXX X.    X .     .X    .  XX .   XX.    X.   XX.     .   X .     .  XI
I  17./EXECMD/ I     .      .     .     .     .     .     .     .     .  X  .     .     .     .     .    I
I  18./INDEX / .XX  X.    X.  X  .     .XX    .     .    X.     .     .  X  .     .  X  .     .     . X I
I  19./INDEXR/ .XXXXX.XXXXX.XXXXX.XXXXX.XXXXX.XX XX.XXXXX.XXXXX.XXXX .XXX X.XXXXX.XXX  . XXXX.   XX.XX XI
I  20./INPUT / I     . X  .     .     .     .     X. X   .     .XX    .  X  .     .X    .     .     . X I
I.............!.....................................................................................................I
I  21./IPCTTB/ .     .      .     .     .     . X   .     .     .     .     . X  .     .     .     .    I
I  22./ISNU  / .     .     X.     .X    .     .     .     .     .     .  X  .     .  X  .     .     .    I
I  23./OPTTB / .     .      .     .     .  X  .     .     .  X  .  X  .     .  X  .     .     .     .XXX I
I  24./PAGEIX/ .     .      .     .X    .     .     .     .     .     .     .     .     .     .     .    I
I  25./PARAMX/ I     .      .     .     .  X  .     .     .     .     .     .     .     .     .     .    I
I.............!.....................................................................................................I
I  26./POINTE/ I     .      .     .     .     .     .     .     .     .  X  .     .  X  .     .     .    I
I  27./SIZE  / I     .     X.  X  .     .     .     .     .     .     .  X  .     .     .     .     .    I
I  28./SWITCH/ IXX   .  X X.  XXX.     . X   .XX X . X X .X   X.     .XXX  .XXX  .X X  .  X  .     .    I
I  29./TABLE1/ IXX   .  X X.  XXX.     . X  X.XX XX. X   . X   .     . XX  .X X  . X   .  X X.X    .    I
I  30./TABLE2/ IX XX .    X. X  X.   X . X  X.XX X . X   . X   .     . XX  .XX   .  X  .  X  .X    .    I
I.............!.....................................................................................................I
I  31./TABLE3/ .     .      .     .     . X   .     .     .     .  X  .     .     .     .X    .    I
I  32./TABLE4/ . X   .      .     .     .     .     .     .     .     .  X  .     .     .     .X    .    I
I  33./TEMPCD/ .    X .  X  . X   .   X . X   .     .     .     .     .XXX  .XX   .     .  X  .     .    I
I  34./TTTAKA/ .     .      .     .     .     .     .     .     .     .  X  .     .     .     .     .    I
I  35./TYPON / .     .  XXX.  XXX.     .     .XX  X. X   .X    .     .  X  .XXX  . X   .     .     .    I
+-------------+-------------------------------------------------------------------------------------------------------------+
=ANALYSIS/77=      LISTINGS OF THE COMMON REFERENCE TABLE                DATE  1981/08/12(WEDENESDAY)     TIME 19:25:27  PAGE 0002
```

2.3 変数名，文字の認識

解釈されるべきＦＯＲＴＲＡＮ文はＳＴＡＴＥＭに１文字／１語で格納されている。Ｘ１はＳＴＡＴＥＭのカラム・カウンタであり，現在どこまで解釈が進行しているかを示している。ＳＴＡＴＥＭからの変数名，定数，文字等のとり出しは特定のサブルーチンを用いて行なっている。

・一文字のとり出し（サブルーチン　ＮＸＣＨＡＲ）

Ｘ１が示している文字の次の１文字はサブルーチンＮＸＣＨＡＲを用いることにより取り出される。ＮＸＣＨＡＲは次の文字までの空白を読み飛ばし，最初の文字を実引数に格納する。サブルーチンの呼び出し形式は

ＣＡＬＬ　ＮＸＣＨＡＲ（ＣＨ）

ＣＨに次の一文字が格納され，Ｘ１にはその文字の位置がセットされる。

・一文字のとり出しと形の認識（サブルーチン　ＮＣＭＯＤＥ）

次の一文字をとり出すのはＮＸＣＨＡＲと同じであるが，とり出された文字が何であるかを数値で示している。文字に対する数値は次のように割り当ててある。

Ａ－２１	０－０
Ｂ－２２	１－１
～　～	～　～
Ｉ－２９	９－９
Ｊ－３１	
～　～	特殊文字－９０
Ｒ－３９	
Ｓ－４２	
～	
Ｚ－４９	

サブルーチンは次の形式で呼び出される。

ＣＡＬＬ　ＮＣＭＯＤＥ（ＣＨ，Ｎ）

ＣＨには次の一文字が，Ｎには数値が格納される。またＸ１には文字の位置が格納されている。

・複数文字のとり出し（サブルーチンＰＡＣＫＮ）

次の文字から空白を除き指定された文字数だけとり出し実パラメータに格納する。指定された文字数の途中に特殊文字が現われた場合は特殊文字の直前までの文字が格納される。サブルーチンは次の形式で呼び出される。

ＣＡＬＬ　ＰＡＣＫＮ（ＣＨ，Ｎ）

ＣＨは文字の格納される領域，Ｎは文字数である。Ｎが５以上の場合はＣＨは適当な語数ＤＩＭＥＮＳＩＯＮ文で宣言確保されていなければならない。

・変数名，定数のとり出し（サブルーチン　ＳＢＮＡＭＥ）

Ｘ１で示されている位置の次から空白を無視し，次の区切記号（特殊文字）の現われるまでを変数名または定数（文字定数を含む）としてとり出す。サブルーチンは次の形式で呼び出される。

ＣＡＬＬ　ＳＢＮＡＭＥ

実行結果はＣＯＭＶＡＬの次の変数に格納される。

ＮＡＭＥ　：　変数名または数値

ＩＮＴＮＵＭ　：　文字数または桁数

ＭＯＤＥ　：　変数名，数値の型

ＣＨ　：　区切番号

ＩＲＥＴＣＤ　＝　１　：　ＮＡＭＥに格納されているのは変数名である

＝　２　：　ＮＡＭＥに格納されているのは定数である

＝　３　：　次の区切記号までの間に変数名，定数は無かった

変数名，定数の型はＭＯＤＥに格納されている数値によって認識できる。型と数値の対応表を表2.2に示す。

表 2.2　変数，定数の型とコード

型	シンボル	コード
実数型	ＲＥＡＬＴＰ	1
整数型	ＩＮＴＴＹＰ	2
論理型	ＬＯＧＴＹＰ	3
倍精度実数型	ＤＢＬＴＹＰ	4
複素数型	ＣＰＸＴＹＰ	5
倍精度複素数型	ＣＤＴＹＰ	6
四倍精度実数型	ＱＵＤＴＹＰ	7
文字型	ＨＯＬＴＹＰ	8

サブルーチンの実行が終了するとＸ１は区切記号の位置を指している。

2.4　入力文の分類

解釈すべき文が何であるかという認識はいくつかの段階に分けて行なわれる。サブルーチンＰＦＣにおいてＳＴＡＴＥＭ上に構成された文はサブルーチンＳＥＬＥＣＴにおいて文番号の

表2.3　FORTRAN文と解釈ルーチン名

先頭文字	解釈ルーチン名	処理対象FORTRAN文
A	ASSIGN	ASSIGN文
B	BACKSP	BACKSPACE文, BLOCKDATA文
C	COMMON	CALL文, COMMON文, CONTINUE文, COMPLEX文
D	DIMENS	DATA文, DECODE文, DEFINEFILE文, DIMENSION文, DOUBLEPRECISION文, DO文
E	EQUIVA	ENCODE文, END行, ENDFILE文, ENTRY文, EQUIVALENCE文, EXTERNAL文,
F	FORMAT	FIND文, FORMAT文, FUNCTION文
G	GOTORN	GOTO文
H	ARITHM	
I	IFRUTN	IF文, INTEGER文, IMPLICIT文
J	ARITHM	
K	ARITHM	
L	LOGICL	LOGICAL文, LCM文
M	ARITHM	
N	NAMLST	NAMELIST文
O	OPTION	OPTION文
P	PAUSER	PAUSE文, PUNCH文, PRINT
Q	ARITHM	
R	READRN	READ文, REAL文, REWIND文, RETURN文, STOP文,
S	STOPRN	SUBROUTINE文
T	TRACEM	
U	UNLOAD	UNLOAD文
V	ARITHM	
W	WRITER	WRITE文
X	ARITHM	
Y	ARITHM	
Z	ARITHM	

チェックがなされる。文番号があればそれをＤＬＴテーブルに登録する。またその行番号がDOループの終りの行であるか否かをチェックする。ＤＯループの終りであればループの深さのカウンタから１を減じＳＷＥＮＤＯスイッチをＯＮにする。次に文の先頭文字がキーとして大まかな分類がなされそれぞれの処理ルーチンに制御を渡す。各処理ルーチンにおいてさらに詳細なチェックがなされ，文の種類が決定され，文法等のチェックがなされる。先頭の文字による分類を表 2.3 に示す。

文の種類が決定するとＳＥＴＢＮＦサブルーチンを呼び出し内部コードの先頭ブロックを作り出す。次に予約語や変数名，オペレータ等の前後関係を調べ内部コード・テーブルを作成していく。文法解釈の途中で現われた変数名はＳＹＭＢＯＬテーブルに格納されその位置が内部コード・テーブルに記録される。

2.5 変数名，定数，文番号の管理

ＦＯＲＴＲＡＮプログラムの解釈中に現われる変数名，定数，文番号はすべてＳＹＭＢＯＬテーブルに格納される。変数名，定数はＳＢＮＡＭＥにより，また文番号はサブルーチンＳ－ＴＭＮＵＭにより取り出される。取り出された変数名等はサブルーチンＤＩＣＮＲＹによりＳＹＭＢＯＬテーブルに格納される。ＳＹＭＢＯＬテーブルは１０語＝１セルのテーブルである。このテーブルに記録する内容は変数，定数，文番号ごとに異なる。以下に各ＳＹＭＢＯＬテーブルの構造を示す。

a) 変数名，関数名，サブルーチン名

ＳＹＭＢＯＬテーブルの構造

0				
1	Name			
2	a	b	c	
3	P．NEXT		P．BLKN	
4	P．ALIAS		P．ARRAY	
5	P．HASH		P．CDICT	
6	TS code			
7	mode	d	RTIME	
8	offset value			
9		length		

Name　：　６文字以内の変数名，関数名，サブルーチン名が記録される。

a，b，c　：　変数名等の属性を示す。

P．NXET　：　次の変数名の登録されているＳＹＭＢＯＬテーブルへのポインタ。
P．BLKN　：　ブロック名テーブルへのポインタ。
P．ARRAY：　変数がＤimension宣言されている場合，ＡＲＲＡＹテーブルへのポインタ。
P．HASH　：　ＨＡＳＨテーブルへのポインタ。
P．CDICT：　コントロール・ディクショナリへのポインタ（アキ）。
TS　code　：　name で示されているものの区別を示すコード（表2.7）参照。
mode　：　整数型，実数型等を識別するための番号。型により番号は表2.2参照。
d　：　変数がＤimension宣言されている場合の次元数。
RTIME　：　この名前がサブプログラム中に現われた回数。

表2.4　ＳＹＭＢＯＬの属性

a/b/c	値	ＩＤ，ＩＤＡ	ＩＤＦ	ＩＤＳ
a	1	ローカル・セクション	組込み関数	組込みサブルーチン
	2	コモン・セクション	ユーザ関数	ユーザ・サブルーチン
	3		文関数	
	4		関数定義名	サブルーチン定義名
b	0			
	1	external symbol	external symbol	external symbol
c	0	未定義変数		
	1	定義変数		

b) 定　数

ＳＹＭＢＯＬテーブルの構造

0	Value		
1			
2			
3			
4	FORMAT		
5	P．HASH	P．CDICT	
6	TS　code		
7	mode		RTIME
8	offset value		
9			

Value　：　数　値

FORMAT　：　プログラム中での表現形式。4バイトで表現される。第1バイトに形を第2バイト以下に桁数が格納される。例えば

−12.34E+5　→　E322

−12.3　→　F310

−12　→　I300

c) 文番号

0	STMNUM		
1			
2	ISN		
3			
4		RANGE	
5	P. HASH	P. CDICT	
6	TS code		
7		d	RTIME
8	offset value		
9			

STMNUM　：　文番号。文番号は文字型で格納され，さらに最後に文字Sが付加されている。

ISN　：　内部発行文番号

RANGE　：　DOの入れ子の番号

d　：　1なら定義されている

このテーブルに同一の名前が重複して格納されることはない。取り出された名前が既に登録されているか否かのチェックはサブルーチンDICNRYが行なっている。ここではテーブルサーチの方法としてハッシング技法を用いている。以下にハッシング技法について述べる。

・ハッシング技法

プログラム中に現われた変数名等が既に登録されているか否かはテーブル・サーチをしてみなければわからない。一番簡単な方法はリニア・サーチでこの方法は一つづつ順次照合していくものである。この方法では登録された名前が多くなると登録，未登録の判定に時間がかかる。このテーブル・サーチの時間を短縮するために考案されたのがハッシング技法であり，多くの手法が発表されている。多くは変数名をキーとしてハッシュ・テーブルと呼ばれるシンボル・テーブルへのポインタを格納するテーブル上のアドレスを作り出す。ハッシュ・テーブル上の指定されたアドレスの内容が0なら登録しようとする名前は未登録であり，シンボル・テーブルに名前を登録し，そのアドレスをハッシュ・テーブルの指定されているアドレスに登録する。もし指定されたアドレスの内容が0でない場合はその内容の示すシンボル・テーブル上の名前と，登録しようとする名前を照合し，一致すれば

登録済であり，一致しなければハッシュ技法を操返し用いるとか，何らかの方法によりテーブル・サーチを続行する。変数名からハッシュ・テーブル上のポインタの作り方と衝突（ハッシュ・テーブル上のアドレスが一致し，名前が異なる場合をいう）があった場合に次のアドレスの作り出し方が各技法で異なる。ＳＣＡＮではQuadratic Quotient Hashing Method[7]を用いている。これは

① 名前を数値とみなしハッシュ・テーブルのサイズｋ（ｋは素数）で除算し，その商をＱ，余をＲとするとき，Ｒをハッシュ・テーブル上のアドレスＡとする。

② ハッシュ・テーブルのアドレスＡの内容が０ならそこに，新たに登録する名前を格納するためのＳＹＭＢＯＬテーブルのアドレスを格納し終了する。

③ ハッシュ・テーブルのＡの内容が０でない場合は，その内容が示すＳＹＭＢＯＬテーブルへのポインタから，そこに登録されている名前と登録しようとする名前が一致するか否かをチェックする。

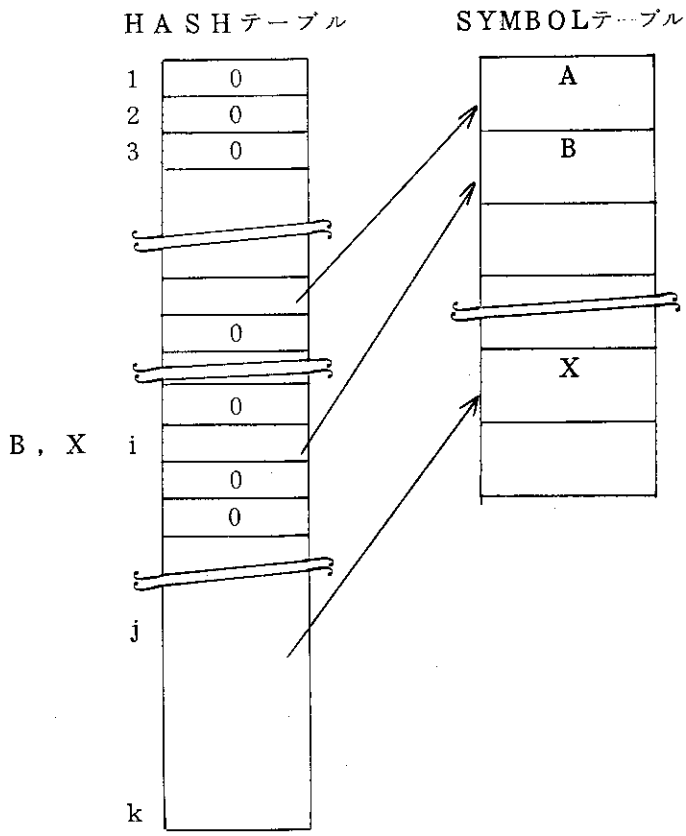
④－１ 一致した場合は登録済であり，ＳＹＭＢＯＬテーブルへのポインタが名前のアドレスである。

④－２ 一致しなければ

$$A = \mathrm{mod}_k\ (i^2 + iQ + R) \qquad (i = 1,\ 2,\ \cdots\cdots)$$

によりハッシュテーブル上のアドレスを再度計算し③から繰返す。

ＨＡＳＨテーブルとＳＹＭＢＯＬテーブル



サブルーチンＤＩＣＮＲＹは名前の登録と同時にmode，ＴＳコードのセット等を行なっている。

2.6 内部コードの管理

各ＦＯＲＴＲＡＮ文解釈ルーチンは入力された文に対応して内部コードを作り出しＢＮＦ－ＴＥＸＴ領域に格納する。ＢＮＦＴＥＸＴは１０語／１セルで文の識別情報をもつ先頭セルと先頭セルだけでは内部コードが格納しきれない場合に作成する継続セルの２つの形式がある。各セルは次のような構造をしている。

ＢＮＦＴＸＴの構造

0	Pointer 1		S－code	
1	Pointer 2		Pointer 3	
2		#CODE	T	U
3	range	ISN		
4			IDS	PBC
5	TEXT			
6				
7				
8				
9				

0	Pointer 4	(Page 4)
1	TEXT	
2		
3		
4		
5		
6		
7		
8		
9		

Pointer 1　：次の文の先頭セルへのポインタ。
Pointer 2　：継続セルへのポインタ。
Pointer 3　：直前の文の先頭セルへのポインタ。
S－code　：文の識別コード（表2.5）参照。
#ＣＯＤＥ　：ＴＳＳＦＯＲＴ で使用。
Ｔ，Ｕ　：ＴＳＳＦＯＲＴ で使用。
range　：この文が属するＤＯの入り子番号。
ＩＳＮ　：内部発行の文番号。
ＩＤＳ　：ＤＯの初期設定。
　０：未処理
　１：終　了
ＰＢＣ　：このセルに記録されている文の記録形式。
　１＝Polish
　２＝ＢＮＦ
　３＝ＣＡＲＤ
ＴＥＸＴ　：本文
Pointer 4　：継続セルへのポインタ。
Page 4　：ＴＳＳＦＯＲＴ で使用。

各FORTRAN文に対しては次のようにS－codeを割り当ててある。

表2.5　　FORTRAN文識別コード

STコード	S－code			
記　号	第1バイト	第2バイト	FORTRAN文	備　考
ASSSTM	30	01	代入文	未使用
ARTEXP	30	02	算術代入文	
LOGEXP	30	03	論理代入文	
ASSIGN	30	04	ASSIGN文	
BACKSP	30	05	BACK SPACE文	
BLDATA	30	06	BLOCK DATA文	
CALLCD	30	07	CALL文	
COMMON	30	08	COMMON文	
CONTNU	30	09	CONTINUE文	
COMPLX	30	0A	COMPLEX文	
DATACD	30	0B	DATA文	
DIMENS	30	0C	DIMENSION文	
DOCD	30	0D	DO文	
DOUBLE	30	0E	DOUBLE PRECISION文	
ENDCD	30	0F	END行	
ENDFIL	30	10	END FILE文	
ENTRY	30	11	ENTRY文	
EQUIVA	30	12	EQUIVALENCE文	
EXTERN	30	13	EXTERNAL文	
FMT	30	14	FORMAT文	
FUNCCD	30	15	FUNCTION文	
UGOTO	30	16	単純GOTO文	
AGOTO	30	17	割り当てGOTO文	
CGOTO	30	18	計算型GOTO文	
IFACD	30	19	算術IF文	
IFLCD	30	1A	論理IF文	
INTGER	30	1B	INTEGER文	
IMPLCT	30	1C	IMPLICIT文	
LOGICL	30	1D	LOGICAL文	
LCMCD	30	1E	LCM文	未使用
NAMECD	30	1F	NAME LIST文	

STコード	S－code			
記　　号	第1バイト	第1バイト	FORTRAN文	備　　考
OPTCD	30	20	OPTION文	未使用
PUNCH	30	21	PUNCH文	
PRINT	30	22	PRINT文	
PAUSE	30	23	PAUSE文	
READ	30	24	READ文	
REAL	30	25	REAL文	
REWIND	30	26	REWIND文	
RETURN	30	27	RETURN文	
STFUNC	30	28	文関数定義文	
STOPCD	30	29	STOP文	
SUBRUT	30	2A	SUBROUTINE文	
UNLOAD	30	2B	UNLOAD文	
WRITE	30	2C	WRITE文	
DECODE	30	2D	DECODE文	
ENCODE	30	2E	ENCODE文	
DEFINE	30	2F	DEFINE FILE文	
FIND	30	30	FIND文	

解釈ルーチンではサブルーチンSETBNFを呼び出し先頭セルのセットを行なう。SET－BNFでは現在処理している文の前に既に解釈済の文のセルがあれば，その文の Pointer 1 に先頭セルの位置を記録する。Pointer 1，2，3，4の関係を図2.2に示す。

2.7　式の解釈法

算術式，算術項の文法チェックにはいくつかの方法が考えられる。式の解釈は筆者らは算術式の解釈ルーチンを作成するにあたり，

・効率がよい（処理系が簡単である），

・処理過程が明確である，

・修正追加等が容易である，

・正確である，

ことなどを考慮し文法チェックの方法を検討する中で独自の方式を考案した。

式はいくつかの項からなり，項はまたいくつかのオペランドとオペレイタからなる。式を解釈することは項を認識し解釈していくことである。オペランド，オペレイタをターミナル・シンボル（V_T）と呼び，解釈の過程で成生される中間のシンボルをノンターミナル・シンボル（N_T）と呼ぶ。順次入力されるV_Tに対してN_Tが順次変化しS（文）またはE（式）とい

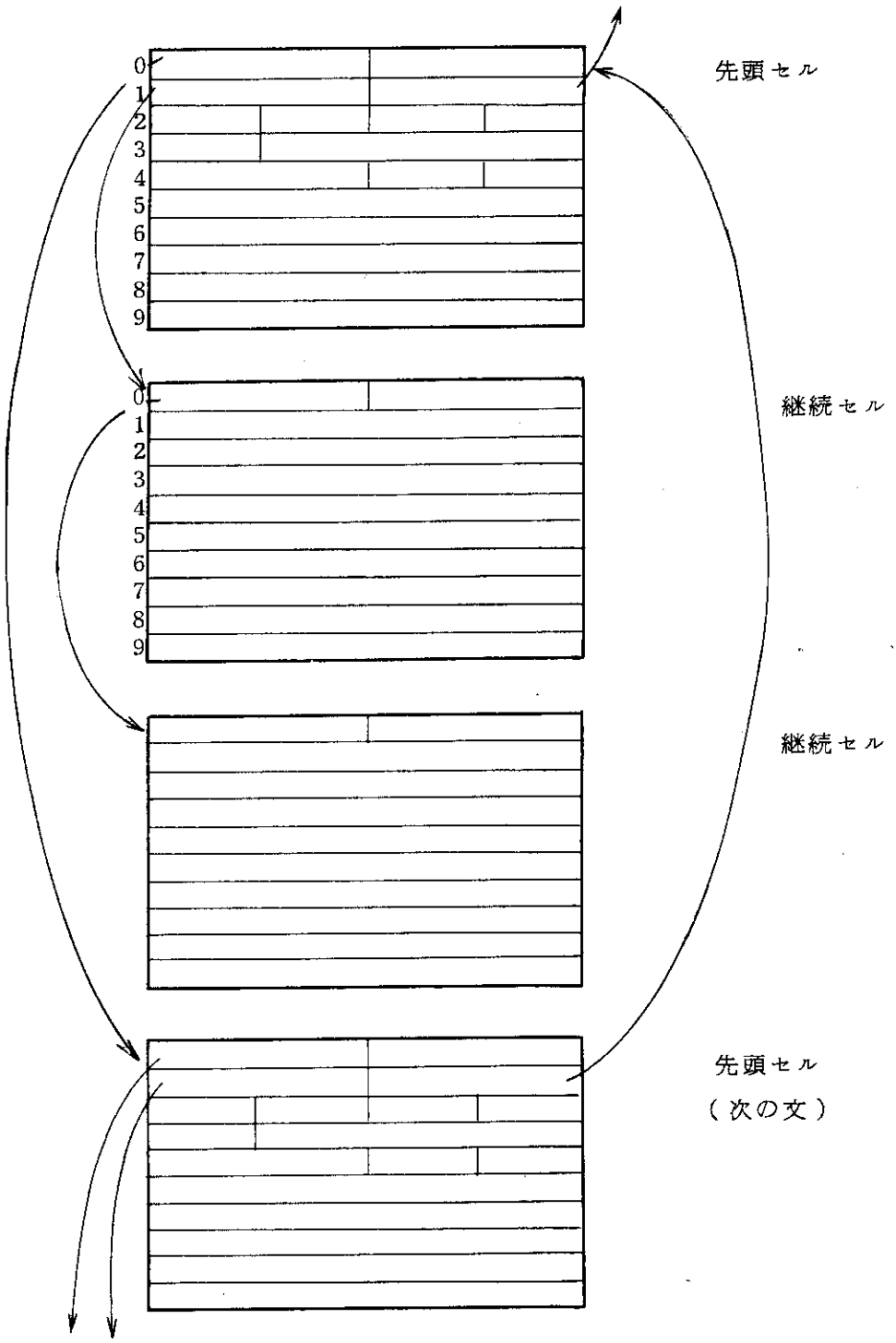


図2.2　ポインタの参照関係

う N_T に変換された時点で解釈は終了する。N_T と V_T N_T の関係及び N_T と N_T の関係は非常に多く，解釈ルーチン作成のうえで非常に繁雑となる。

筆者らの方式は V_T と N_T の関係と解釈手順を記述した表（遷移テーブル）を作成し，これとＳＴＡＴＥと呼ぶ解釈過程を記録するテーブル（プッシュ・ダウン・スタック）を用いるものである。図 2.3 に解釈方式を，表 2.6 に遷移テーブルを示す。遷移テーブルは実際は A_{CT} 番号と V_T ，N_T につけられている番号の２組の数値から構成されている。表 2.7 に遷移テーブルで用いられている V_T ，N_T の定義を示す。表 2.7 においてNo.は記号につけられた番号であり，N_{OHEX} はそれを１６進数で現わしたものである。ＳＹＭＢＬ欄にはＳＣＡＮで用いられている V_T ，N_T が全て記載されている。コードは V_T ，N_T に与えられた個有の番号であり，実際テーブルに記されているのはこのコードだけである。コードは２つの部分からなりａは記号の分類に使用している。ａはさらに２つの部分からなり最初の数値が２０以下のものは V_T のうちオペレイタを，２０はオペランドを２１はそのうち仮引数としてＦＯＲＴＲＡＮプログラム中に現われたオペランドを示す。２２は記号が N_T であることを示している。ａの２番目の数値は識別番号である。オペレイタの数値が細分されているのは，この数ポーランド記法のオペレータの順位を現わしているためである。

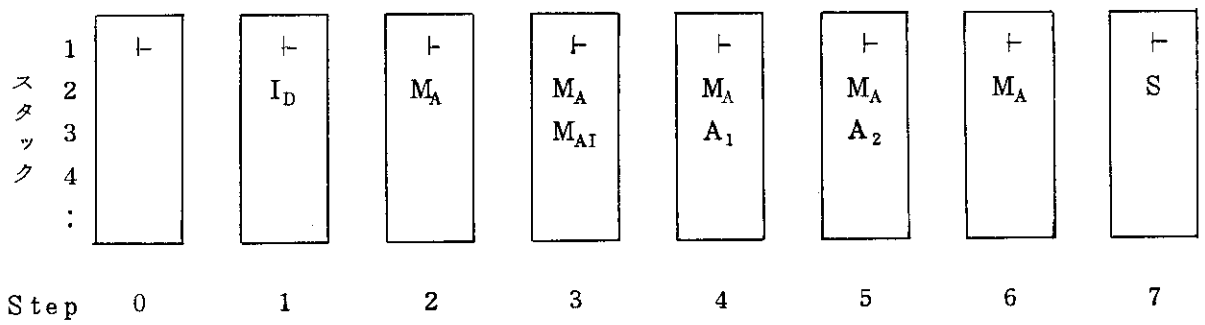
ｂの欄も２つ部分からなり，最初の数値は遷移テーブルの ℓ の欄にこの記号が定義されて位置を，後の数値は遷移テーブルのｒの欄にこの記号が定義されている位置を示している。空白は定義されていないことを示す。

次に簡単な具体例を掲げて解釈の手順を示す。

代入文の解釈例

A＝B＋C　→　idi　＝　idi　aop　idi　⊣

V_T	idi	＝	idi	aop	idi	⊣	
r	idi	＝	idi	aop	idi	⊣	A_2⊣



A＝B＋C

というＦＯＲＴＲＡＮ文の構成要素はサブルーチンＳＢＮＡＭＥにより変数名とオペレータとして分離され取り出される。それぞれのシンボルはＳＢＮＡＭＥやＤＩＣＮＲＹで

$$idi = idi \quad a_{op} \quad idi \quad \dashv$$

として認識する。式の解釈ルーチンにはこの順序で情報が与えられる。ここで a_{op} は算術オペレータを，$\dashv$ は文の終端を示す記号である。ｎはプッシュ・ダウン・スタックの番号を示すものとする。初期状態としてｎ＝１に，プッシュ・ダウン・スタックＳＴＡＴＥ(1)には始端記号 $\vdash$ が初期設定されている。

i) ℓ＝ＳＴＡＴＥ（ｎ），$r = V_{T1}$＝idiとしてＦ（ℓ，ｒ）＝（A_{CT}, S）を求める。すなわち遷移テーブルの（１，３）の内容を取り出す。内容は↓Sの形であるから A_{CT} の値は２であり，ｎ＝ｍ＋１としてＳＴＡＴＥ(2)にＩＤを格納する。（状態１）。次の処理は V_T であるから次の V_{T2} を取り出す。

ii) ℓ＝ＳＴＡＴＥ（ｎ），$r = V_{T2}$＝　＝　として（A_{CT}, S）を求める。すなわち遷移テーブルの（３，１２）の内容を取り出す。内容はＳの形であるから A_{CT} の値は９であり，ＳＴＡＴＥ（ｎ）＝ＳＴＡＴＥ(2)に M_A を格納する。（状態２）。次の処理は V_T であるから V_{T3} を取り出す。

iii) ℓ＝ＳＴＡＴＥ（ｎ），$r = V_{T3}$＝idi，F（ℓ，ｒ）は↓M_AI であるから状態３となる。V_{T4} を取り出す。

IV) $\ell = M_{AI}$，$r = a_{op}$，F（ℓ，ｒ）＝A_1 従って A_{CT} は９であり，結果は状態４となる。V_{T5}を取り出す。

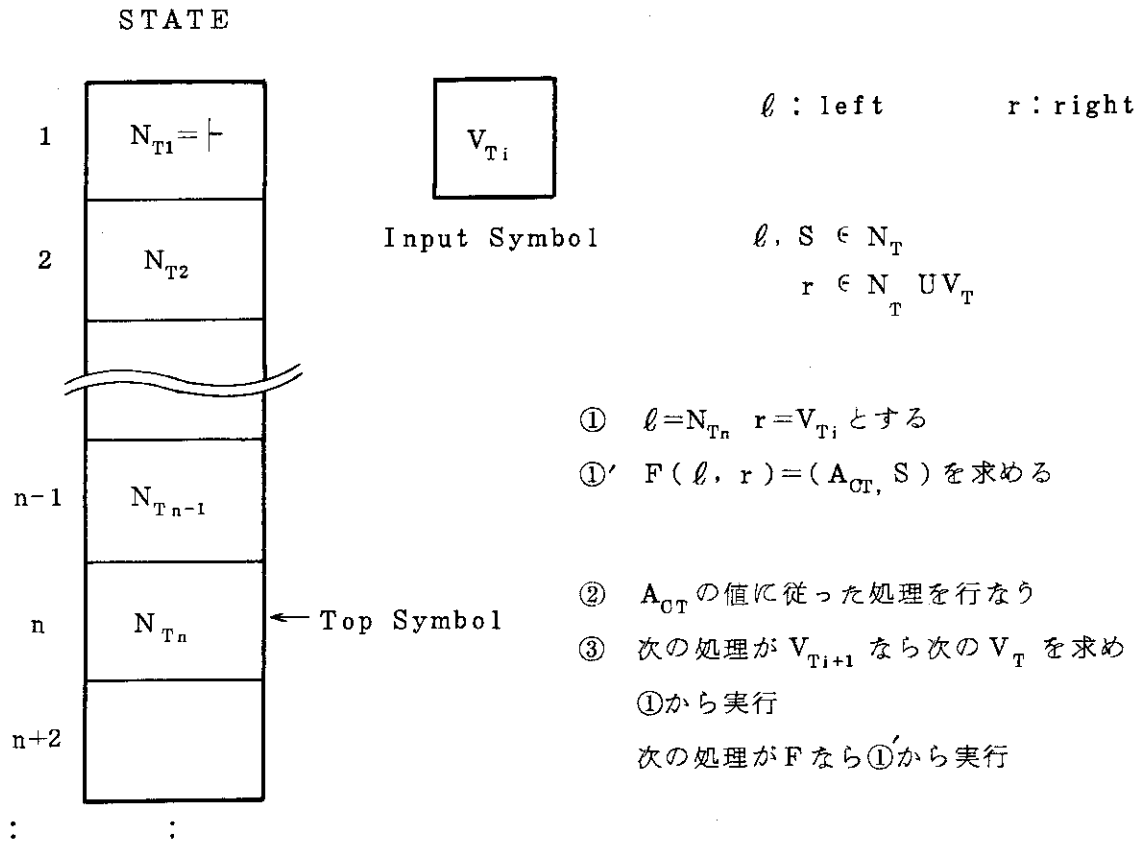
V) $\ell = A_1$，r＝idi，F（ℓ，ｒ）＝A_2 従って A_{CT} は９であり，結果は状態５となる。V_{T6} を取り出す。

vi) $\ell = A_2$，r＝$\dashv$，F（ℓ，ｒ）＝ＩＤ$\dashv$↑ 従ってA_{CT}は４であり，ｒ＝ＩＤ$\dashv$，ｎ＝ｎ－１，ℓ＝ＳＴＡＴＥ（ｎ－１）＝M_A とする。（状態６）。次の処理はＦであるから遷移テーブルの（１２，２２）の内容をとり出す。内容は↑S であるから A_{CT} は６である。従って状態７になる。ここでＳＴＡＴＥ(2)＝５なので処理を終了する。

この解釈方式においてＳＴＡＴＥ(2)にＩ／Ｏリスト処理のための先頭シンボルまたはＤＯ文処理のための先頭シンボルを前もって設置しておくことにより，Ｉ／Ｏリスト，ＤＯのパラメータの並びもチェックできる。

表2.6　遷移テーブル

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
			1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28
		ℓ r	(	)	idi	idn	idl	neg	a_{op}	.R.	.L.	¬	·	=	┤	P_H	A_H	P_{M0}	P_{M1}	P_{M2}	M_A	I_D	I_D)	I_D┤	NI_D	A_2	A_2	A_2┤	R_2	R_2)	R_2┤	L_1	L_1)	L_1┤	L_2	L_2)	L_2┤	F	F)	F┤	D_{ST})	D_{ST}┤
1	1	├	↓		↓I_D	↓I_D	↓L_2	↓				↓				↓	↓				—	—		E	—	—		E	—						—		E	—		E		
2	2	(	↓		↓I_D	↓I_D	↓L_2	↓				↓				↓	↓					—	↑A_2		—	—	↑A_2		—	↑L_2					—	↑L_2		—	↑F		↑I_D	
3	3	I_D		I_D)↑					A_1	I_R	↓		P_{M0}↑	M_A↑	I_D┤↑																	↑L_2	↑L_2)	↑L_2┤								
4	4	A_1	↓		A_2	A_2										↓	↓					↑A_2		↑A_2┤		↑A_2											↑A_2	↑A_2)	↑A_2┤		
5	5	A_2		A_2)↑					A_1	I_R			P_{M1}↑		A_2┤↑																							↑R_2	↑R_2)	↑R_2┤		
6	6	I_R	↓		R_2	R_2		↓								↓	↓					↑R_2		↑R_2┤	↑R_2	↑R_2											↑R_2	↑R_2)	↑R_2┤		
7	7	R_1	↓		R_2	R_2										↓	↓					↑R_2		↑R_2┤		↑R_2																
8	8	R_2		R_2)↑					R_1		↓		P_{M2}↑		R_2┤↑																	↑L_2	↑L_2)	↑L_2┤								
9	9	.L.	↓		↓I_D	↓I_D	↓L_2	↓				↓				↓	↓	↑L_1	↑L_1			—	↑L_1)	↑L_1┤	↑L_1				↑L_1	↑L_1)	↑L_1┤				↑L_1	↑L_1)	↑L_1┤	—	↑L_1)	↑L_1┤		
10	10	¬	↓		↓I_D		↓L_2					↓				↓	↓					↑L_2	↑L_2)	↑L_2┤	↑L_2						↑L_2┤						↑L_2┤	—	↑L_2)	↑L_2┤		
11	11	L_2									↓		P_{M2}↑		L_2┤↑																	↑L_2	↑L_2)	↑L_2┤								
12	12	M_A	↓		↓M_{AI}	↓A_2	↓L_2	↓								↓	↓					M_{AI}		↑S	—	—		↑S	—		↑S				—		↑S	—		↑S		
13	13	M_{AI}							A_1		I_D↓			↑	I_D┤↑																			↑L_2┤								
14	14	A_H	↓		↓I_D	↓I_D		↓								↓	↓	↑	↑			—	↑I_D		—	—	↑I_D											—	↑I_D			
15	15	P_H	↓		↓I_D	↓I_D	↓L_2	↓								↓	↓	↑	↑	↑		—	↑F		—	—	↑F		—	↑F					—	↑F		—	↑A_2			
16	16	neg	↓		NI_D	I_D↑										↓	↓					↑NI_D				↑A_2		↑A_2┤										↑NI_D		↑I_D┤		
17	17	NI_D		I_D)↑					A_1		↓		P_{M1}↑		I_D┤↑																	↑L_2	↑L_2)	↑L_2┤								
18	18	F		F)↑					A_1	I_R	↓		P_{M1}↑		F┤↑																	↑L_2	↑L_2)	↑L_2┤								
19	19	IOL	↓IOL(		↓I_D										↑S		↓	—				—		↑S																		
20	20	IOL(	↓IOL(		↓I_D												↓	—				—	↑I_D																			↑I_D	
21	21	ID_0			↓I_D	↓I_D												↑ID_0P_1	↑ID_0P_1				↑I_D			—																
22	22	P_{MO}	↓IOL(		↓I_D												↓	↑	↑		ID_0	—		↑I_D┤		—																↑D_{ST})	
23	23	ID_0P_1	↓		↓I_D	↓I_D												↑ID_0P_2	↑ID_0P_2				↑D_{ST})	↑D_{ST}┤		—	↑D_{ST})	↑D_{ST}┤															
24	24	ID_0P_2	↓		↓I_D	↓I_D																	↑D_{ST})	↑D_{ST}┤		—	↑D_{ST})	↑D_{ST}┤														↑S
25	25	D_H			↓I_D																	ID_0																				



	A_{CT}の値	(A_{CT}, S)の記号表示	処　　理	次の処理
r ∈ V_T	1	↓	n=n+1, STATE(n)=r	V_{Ti+1}
	2	↓S	n=n+1, STATE(n)=s	〃
	3	S↓	STATE(n)=S, n=n+1, STATE(n)=r	〃
r ∈ N_T	4	S↑	r=S, n=n−1, ℓ=STATE(n)	F
	5	↑S	n=n-1, STATE(n)=S, r=S, ℓ=STATE(n-1)	F
	6	↑S	n=n−1, STATE(n)=S	V_{Ti+1}
	7	↑	n=n−1	〃
	8	—	n=n	〃
r ∈ V_T U N_T	9	S	STATE(n)=S	〃

図2.3　式の解釈方式

表2.7　　TSコード表

No.		記号	V_T/N_T	TSコード				意味
D	H_{EX}			a		b		
1	1	⊢	V_T	1		1		始端
2	2	=	V_T	2			C	
3	3	(	V_T	3		2	1	
4	4	,	V_T	4	1		B	
5	5	)	V_T	4	2		2	
6	6	⊣	V_T	4	3		D	終端
7	7	·OR·	V_T	5		9	9	論理演算子 (.L.)
8	8	·AND·	V_T	6		9	9	
9	9	·NOT·	V_T	7		A	A	
10	A	·LT·	V_T	8	1		8	比較演算子 (.R.)
11	B	·LE·	V_T	8	2		8	
12	C	·GT·	V_T	8	3		8	
13	D	·GE·	V_T	8	4		8	
14	E	·EQ·	V_T	8	5		8	
15	F	·NE·	V_T	8	6		8	
16	10	+	V_T	9	1		7	算術演算子 (a_{op})
17	11	−	V_T	9	2		7	
18	12	＊	V_T	A	1		7	
19	13	／	V_T	A	2		7	
20	14	NEG	V_T	A	3	1	6	unary operator (−)
21	15	＊＊	V_T	B			7	ベキ乗 (a_{op})
22	16	¥	V_T	C				
23	17		V_T	D				

No.		記号	V_T/N_T	TSコード				意味
D	H_{EX}			a		b		
24	18	•	V_T	E				
25	19	idn	V_T	10	1		4	定数
26	1A	idh	V_T	10	2		4	文字定数
27	1B	idl	V_T	10	3		5	論理定数
28	1C	idi	V_T	10	4		3	変数名
29	1D	ida	V_T	10	5	E	F	array 変数名（AH）
30	1E	idf	V_T	10	6	F	E	関数名　（PH）
31	1F	ids	V_T	10	7			サブルーチン名
32	20	iddp	V_T	11			3	ダミー・パラメータ
33	21	iddi	V_T	11	4		3	ダミー・変数名
34	22	idda	V_T	11	5	E	F	ダミーarray変数名（AH）
35	23	iddf	V_T	11	6	F	E	ダミー関数名　（PH）
36	24	idds	V_T	11	7			ダミー・サブルーチン名
37	25	ID	N_T	12	1	3	14	IDentifier
38	26	ID)	N_T	12	2		15	IDの次に右括弧が現われた状態
39	27	ID ┤	N_T	12	3		16	IDの次に終端記号が現われた状態
40	28	A_1	N_T	12	4	4		ID.A_2の次にa_{op}が現われた状態
41	29	A_2	N_T	12	5	5	18	算術項
42	2A	A_2)	N_T	12	6		19	算術項と右括弧
43	2B	A_2 ┤	N_T	12	7		1A	算術項と終端記号
44	2C	R_1	N_T	12	8	7		関係式の次に.R.が現われた状態
45	2D	R_2	N_T	12	9	8	1B	関係式
46	2E	R_2)	N_T	12	A		1C	関係式と括弧
47	2F	R_2 ┤	N_T	12	B		1D	関係式と終端記号
48	30	L_1	N_T	12	C		1E	.L.とR_2の並び
49	31	L_1)	N_T	12	D		1F	L_1 と右括弧
50	32	L_1 ┤	N_T	12	E		20	L_1 と終端記号

No.		記号	V_T/N_T	TSコード				意味
D	H_{EX}			a		b		
51	33	L_2	N_T	12	F	B	21	論理項
52	34	L_2)	N_T	12	10		22	論理項と右括弧
53	35	L_2┤	N_T	12	11		23	論理項と終端記号
54	36	PM_0	N_T	12	12	16	10	IDの次に，が現われた状態
55	37	PM_1	N_T	12	13		11	A_2の次に，が現われた状態
56	38	PM_2	N_T	12	14		12	R_2の次に，が現われた状態
57	39	M_A	N_T	12	15	C	13	代入文左辺と＝
58	3A	NI_D	N_T	12	16	11	17	neg と ID
59	3B	F	N_T	12	17	12	14	関数参照
60	3C	F)	N_T	12	18		25	Fと右括弧
61	3D	F┤	N_T	12	19		26	Fと終端記号
62	3E	I_R	N_T	12	1A	6		IDと .R.
63	3F	IOL	N_T	12	1B	13		I／O list
64	40	IOL(	N_T	12	1C	14		IOLと左括弧
65	41	ID_O	N_T	12	1D	15		DO，文番号，変数名，＝の形
66	42	ID_0P_1	N_T	12	1E	17		ID_0に初期値とコンマの状態
67	43	ID_0P_2	N_T	12	1F	18		ID_0P_1に増分値とコンマの状態
68	44	D_{ST})	N_T	12	20		27	implied DOと右括弧
69	45	D_{ST}┤	N_T	12	21		28	implied DOと終端記号
70	46	D_H	N_T	12	22	19		DO
71	47	M_{AI}	N_T	12	23	D		M_Aの後のI_D
72	48	S	N_T	12	24			文
73	49	D_{END}	V_T	4	4			DOの範囲終了
74	4A	S_{DIV}	V_T	4	5			
75	4B	ID_{SN}	V_T	10	8			文番号
76	4C	ID_{AN}	V_T	10	9			
77	4D	E	N_T	12	25			式

3. FORTRAN文の内部コード表現

ここではSCANが解釈するFORTRAN文と作成するBNFTXTについて説明する。すでに述べたようにBNFTXTには先頭セルと継続セルの二つの形式がある。先頭セルの最初の5語は文の種類と文の前後関係を示すために用いている。残りの5語及び継続セルを用いて文の内容を示している。文の内容はすべて内部コードで記述されるがここではわかり易いように，特殊文字はその記号で，またテーブルへのポインタはP.SYMBOLのようにPの後にピリオドとテーブル名で示している。BNFTXT作成の際にこれらの記録順序は文によって異なるが原則としてオペレイタ，オペランドの出現順序となっている。また省略しても混乱を生じないものは記録していない。

3.1 宣言文

宣言文には

- DIMENSION文，
- IMPLICIT文，
- 型宣言文，
  - INTEGER文，
  - REAL文，
  - DOUBLEPRECISION文，
  - COMPLEX文，
  - LOGICAL文，
- COMMON文，
- EQUIVALENCE文，
- DATA文，
- EXTERNAL文，

がある。ここでは以下の文についての処理方式と内部コードの表現について述べる。

3.1.1 DIMENSION文

DIMENSION	$v_1(i_1)$, $v_2(i_2)$, …… $v_n(i_n)$
	v(i) : 配列宣言子

サブルーチンDIMENSにおいて処理すべき文がDIMENSION文であることが確認されるとSETBNFルーチンが先頭セルを作り出す。次に変数名が現われるごとにサブルーチンDIMCOLを呼び出し配列宣言の処理を行なう。

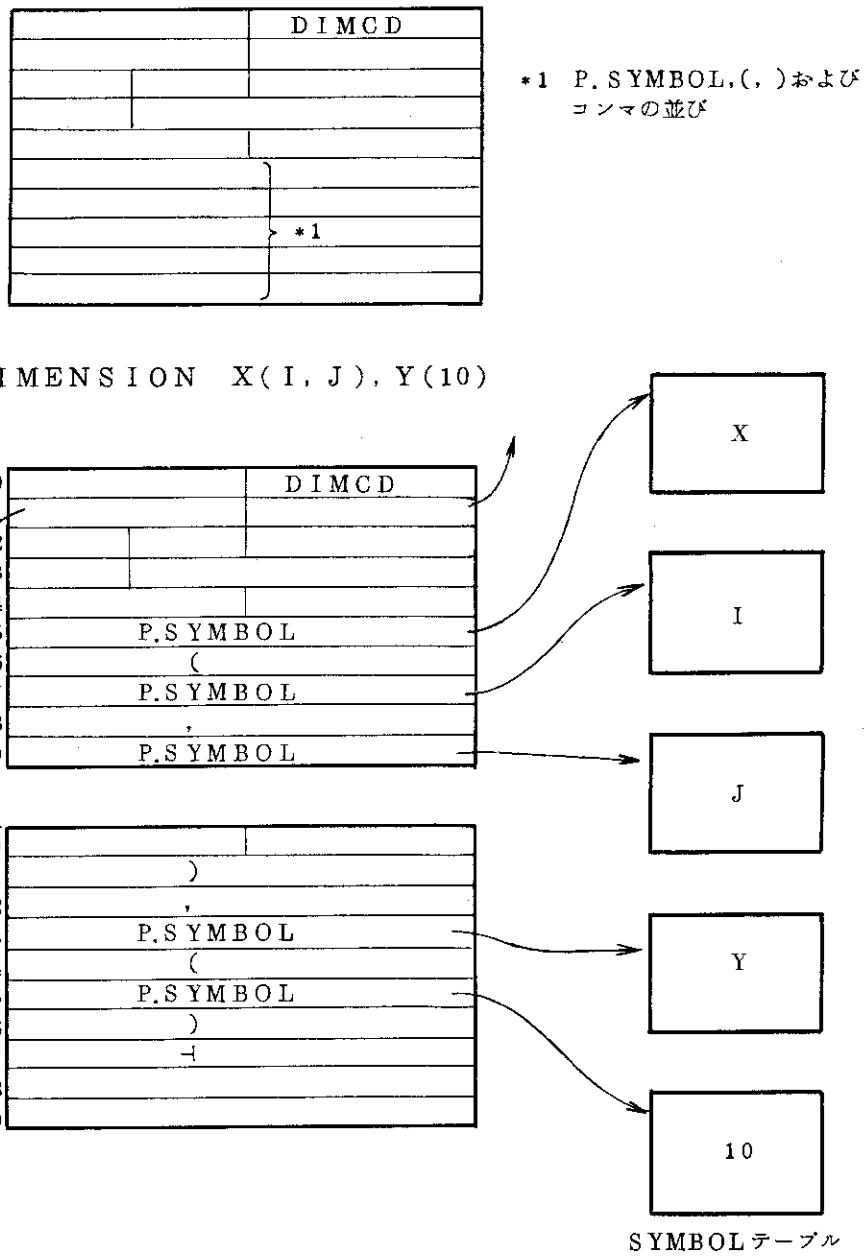
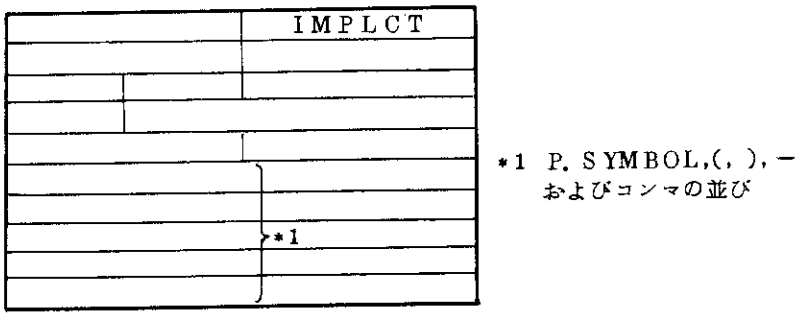


図3.1 DIMENSION文の内部コード表現

3.1.2 IMPLICIT文

IMPLICIT	$type_1$ (a_1, a_2, …) , $type_2$ (b_1, b_2, …)
	type : REAL, INTEGER, COMPLEX, LOGICAL
	a, b : 英字の一文字もしくは(a－b)の型式

IMPLICIT文はサブルーチンIFRUTNで認識される。暗黙の型はサブルーチンPFCで初期設定されている。IMPLICIT文は解釈されると，内部テーブルを作成すると同時に暗黙型宣言のテーブルIPCTTBを変更する。IPCTTBはプログラム単位が変ると初期設定し直される。



例　IMPLICIT REAL(I, J), INTEGER(A－H)

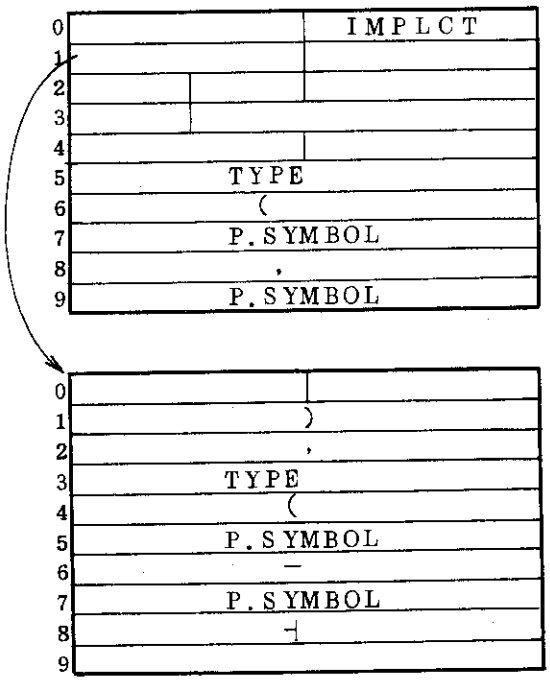
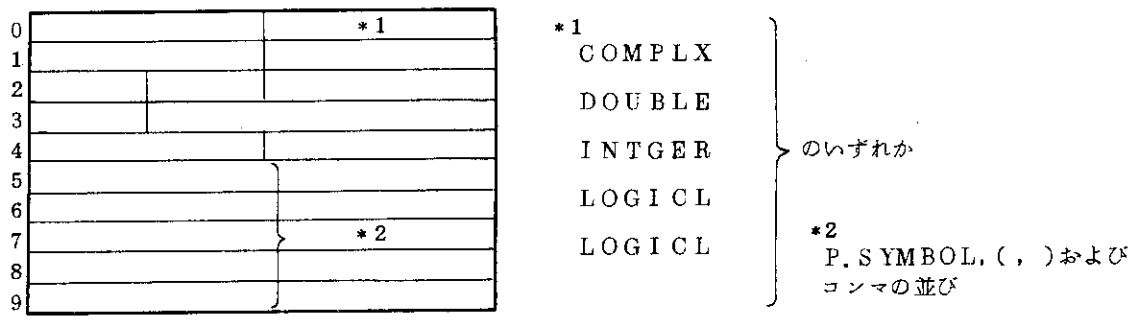


図3.2　IMPLICIT文の内部コード表現

3.1.3 型宣言文

type $*S_0$	$v_1(i_1)$, $v_2(i_2)$, …, $v_n(i_n)$
	v ： 変数名，配列名，関数名
	i ： 宣言子添字

処理すべき文が型宣言文であることが認識されると，TYPESWに型が何であるかをセットし，制御がサブルーチンFNORNTに渡される。FNORNTではその文が関数定義文か否かを確かめる。関数定義文でないことが確認されるとそれぞれの型に従った内部コードテーブルを作成する。また型宣言された変数の登録されているSYMBOLテーブルのMODEを変更する。配列宣言がなされている変数についてはDICNRYを呼び出してその処理も行なう。



例　INTEGER　A，B(100)

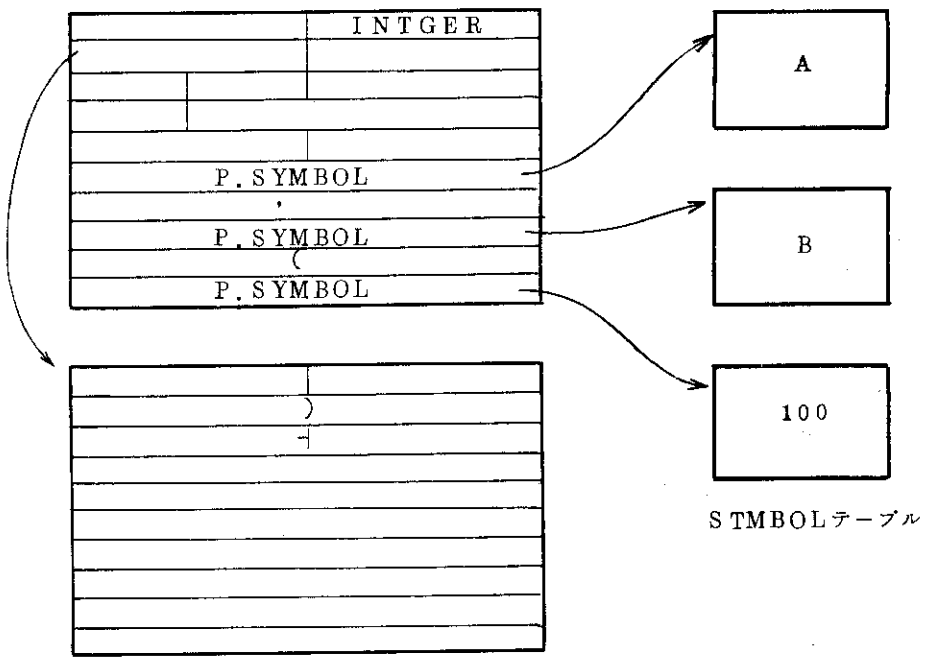


図3.3　型宣言文の内部コード表現

3.1.4 COMMON文

COMMON	$/x_1/a_1, a_2, \cdots\cdots$ $/x_2/b_1, b_2, \cdots\cdots$
	x ： 英字名または空
	a, b ： 変数名，配列名，配列宣言子の並び

COMMON文はサブルーチンCOMMONで認識される。先頭ブロックを作成した後ブロック名を取り出しBLOCKNテーブルに格納する。SCANにおいてはBLOCKNテーブルはプログラム単位に出現するブロック名の収集のみを行なっている。テーブルの構造を図3.4に示す。

0		
1	name	
2	length	
3	Pointer 1	Pointer 2
4	Pointer 3	Pointer 4

name ： COMMONブロック名

length ： ブロックの長さ

Pointer 1： EQUTBLへのPointer（SCANでセット）

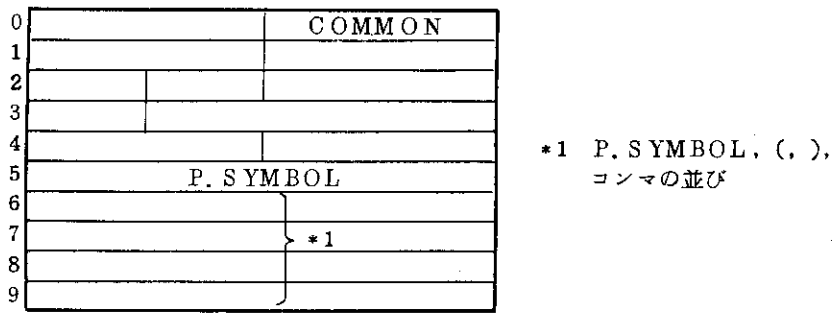
Pointer 2： 〃 （TSSFORTでセット）

Pointer 3： 先頭変数名へのポインタ

Pointer 4： 最終変数名へのポインタ

図3.4 BLOCKNテーブル

ブロック名の処理終了後，変数名，配列名，配列宣言子の並びを順次処理し内部コードテーブルを作成する。



例　COMMON　/AAA/　A(100)，B(200)

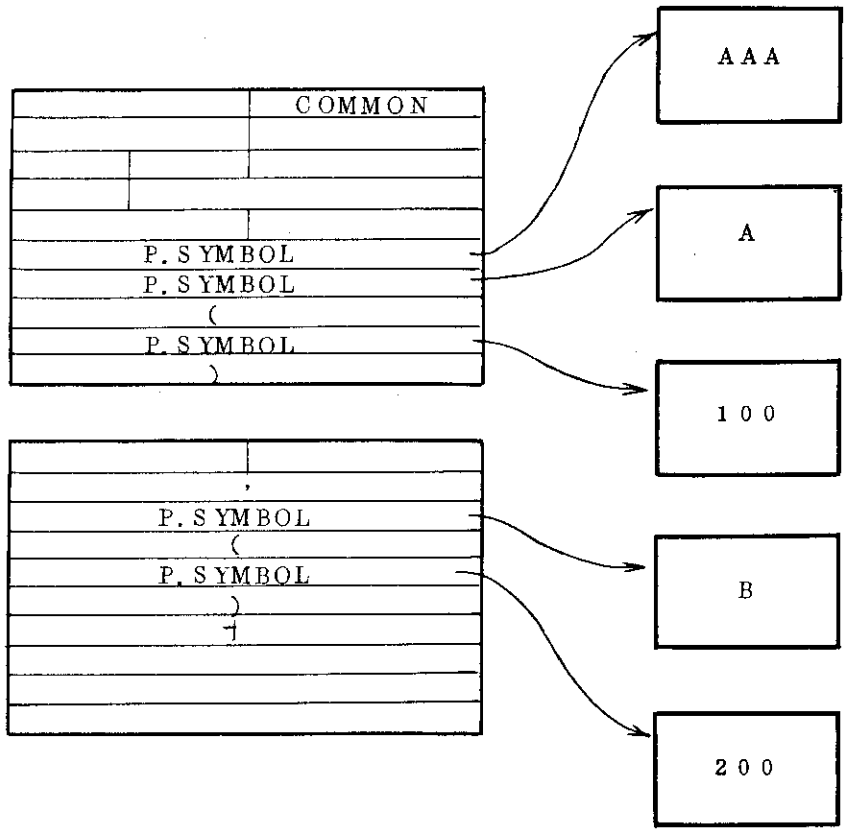
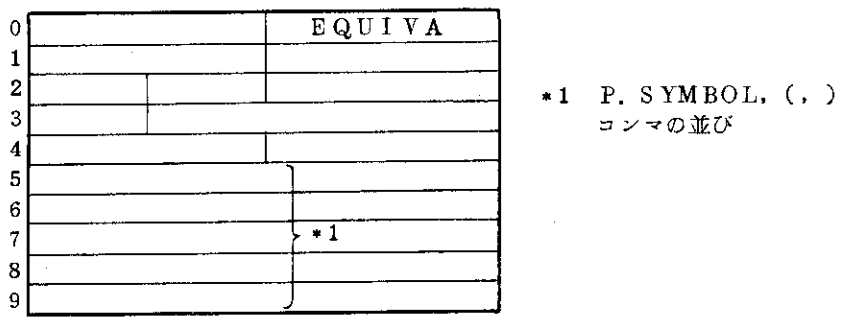


図3.5　COMMON文の内部コード表現

3.1.5 その他

EQUIVALENCE	(a_1, a_2, …), (b_1, b_2, …)
	a, b：変数名，配列要素名
DATA	k_1／d_1／, k_2／d_2／, …… k_n／d_n／
	k ： 変数名．配列名，配列要素名，DO型並びの並び
	d ： 一般の定数の並び
EXTERNAL	v_1, v_2, … v_n
	v ： 外部手順き名

EQUIVALENCE文及びEXTERNAL文はサブルーチンEQUIVAで，DATA文はサブルーチンDIMENSで認識される。その内部コード表現の例を図3.6，7，8に示す。



例　EQUIVALENCE (A(1), X(1, 1))

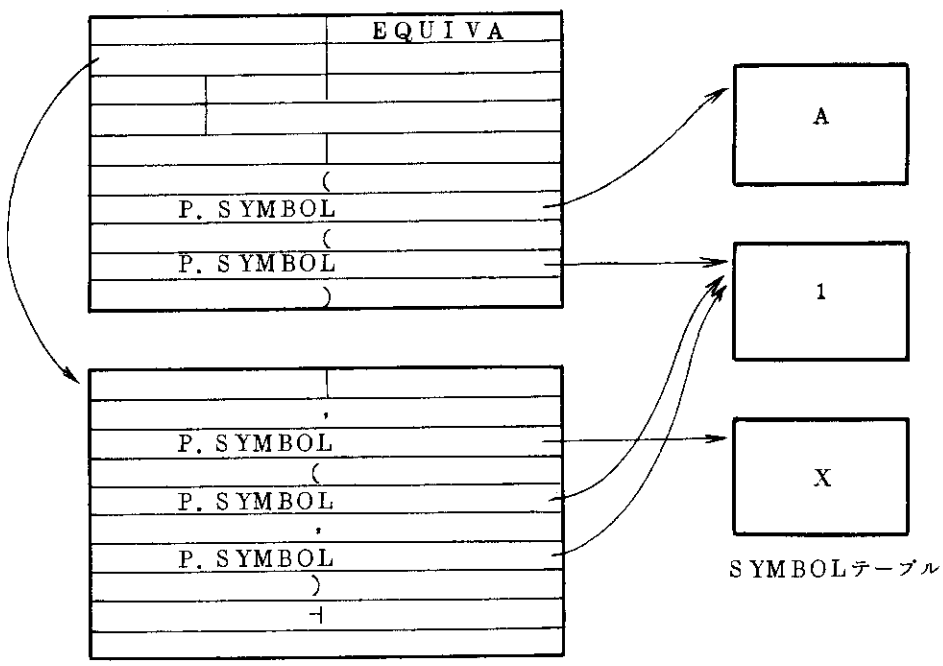
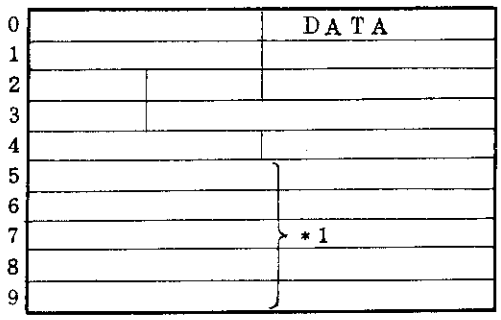


図3.6　EQUIVALENCE文の内部コード表現



*1　P. SYMBOL，特殊文字コードの並び

例　DATA　A，B，C／1.，2 * 2.1

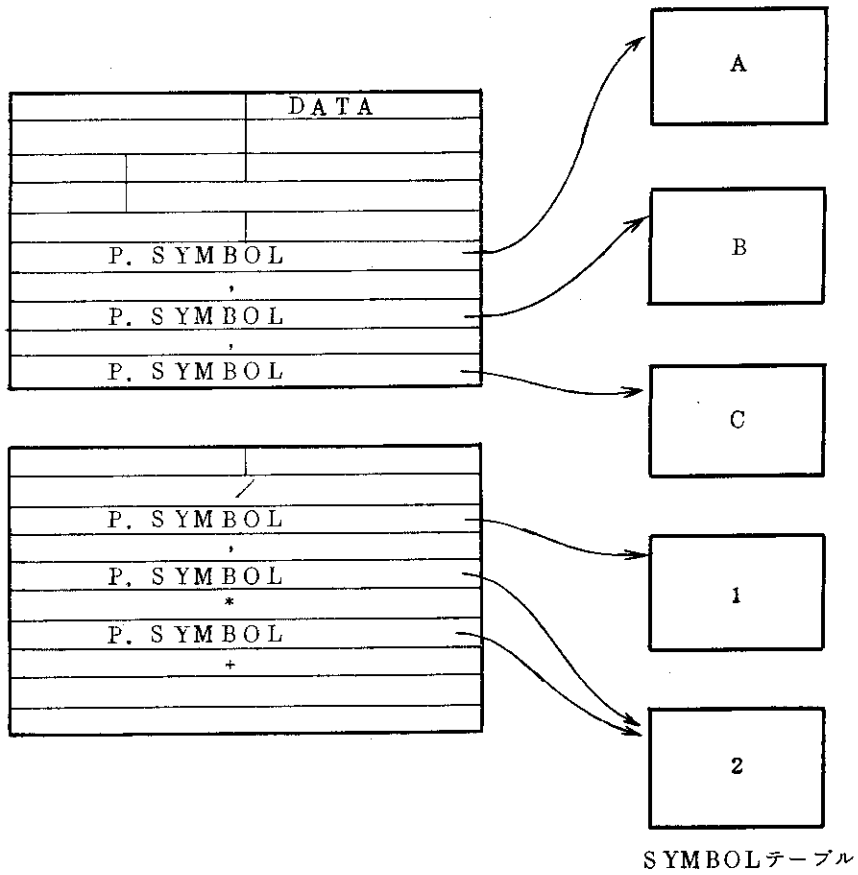
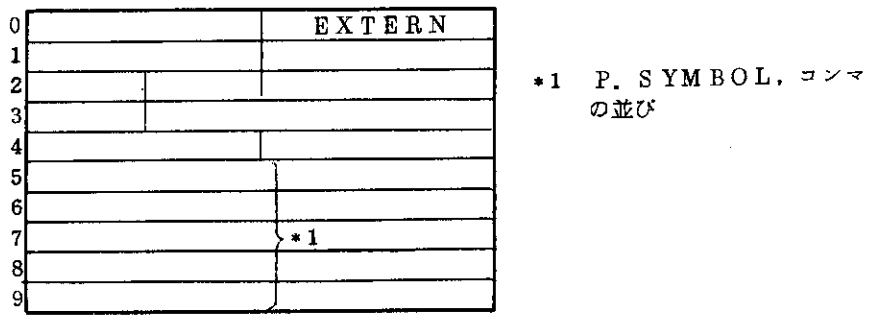


図3.7　DATA文の内部コード表現



例　EXTERNAL XFUNC, YFUNC

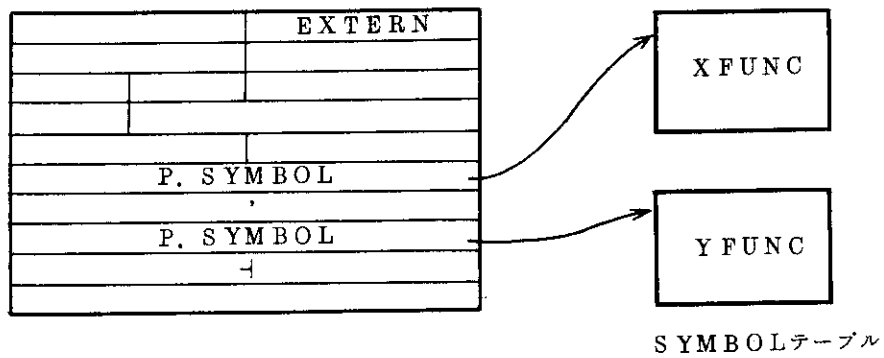


図3.8　EXTERNAL文の内部コード表現

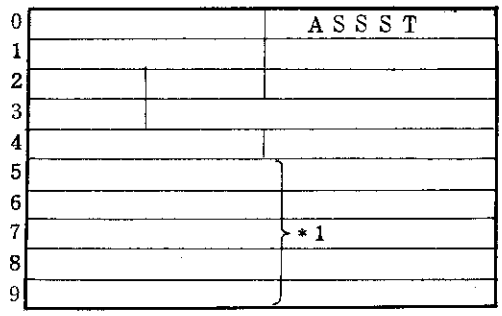
3.2 代入文

v = e

v ： 論理型以外の配列名または配列要素名

e ： 算術式

代入文はサブルーチンARITHMによって処理される。ARITHMはSELECTから直接呼び出されるだけでなく，SELECTから呼び出された他の処理ルーチンにおいて，解釈すべき文がその処理ルーチンに該当しないと判断されたとき，そのルーチンから呼び出される。ARITHMでは式の等号までの形式が正しいことを確認するとARTHXPを呼び出し文法のチェックを行なう。代入文は内部コード表現される場合すべての変数名，特殊文字がその並びの順にコード化され格納される。以後式を文字eを用いて現わす場合がある。



*1 P.SYMBOL，特殊文字コードの並び

例　A(I)=SIN(X)*Y

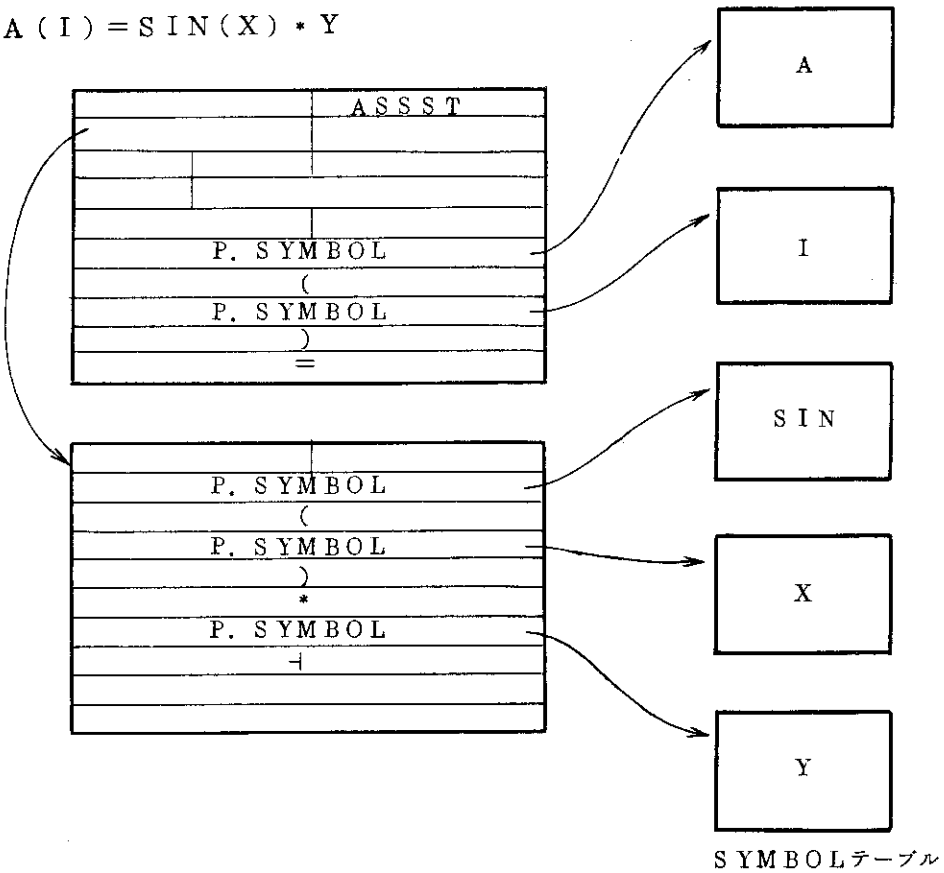


図3.9　代入文の内部コード表現

3.3 制御文

3.3.1 GOTO文

・単純GOTO文
　GO TO k
　　k：同一プログラム単位内の実行文につけられた文番号

・割り当て型GOTO文
　ASSIGN k TO i
　　⋮
　GO TO i, (k_1, k_2, … k_n)
　　k：同一プログラム単位内の実行文につけられた文番号
　　i：整数型の変数名

・計算型GOTO文
　GO TO (k_1, k_2, … k_n), i
　　k：同一プログラム単位内の実行文につけられた文番号
　　i：整数型の変数名

GOTO文には単純GOTO文，割り当て型GOTO文，計算型GOTO文があり，それぞれサブルーチンGOTORNで認識される。またASSIGN文はサブルーチンASSIGNで認識される。GOTO文で指定された文番号はSYMBOLテーブルに格納されるが，同時にRLTテーブルに内部発行の文番号と，SYMBOLテーブルへのポインタを格納する。RLTテーブルはEND行が認識された時点でDLTテーブルと照合され，未定義の文番号がないか否かがチェックされる。RLTテーブル構造及び各GOTO文，ASSIGN文の内部コード表現の例を図3.10，11，12，13，14に示す。

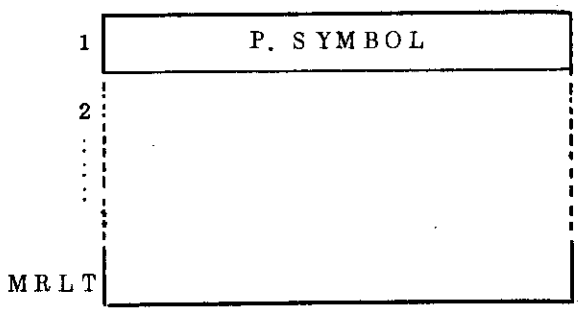
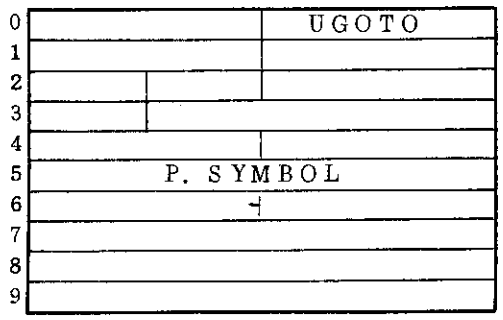


図3.10　RLTの構造



例　GOTO　100

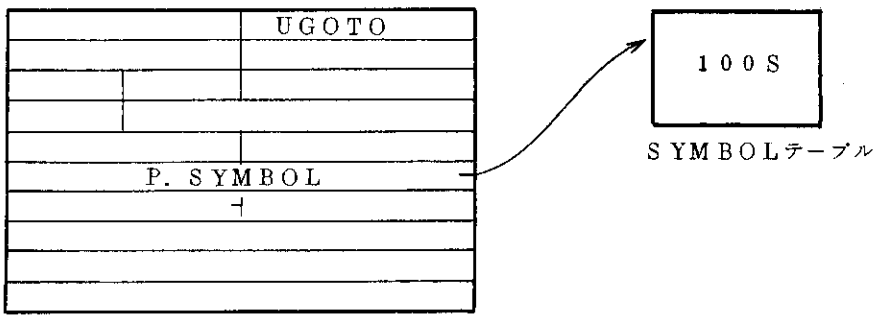
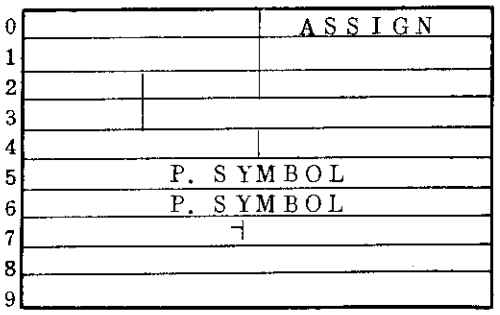


図3.11　単純GOTO文の内部コード表現



例　ASSIGN　100　TO　ISN

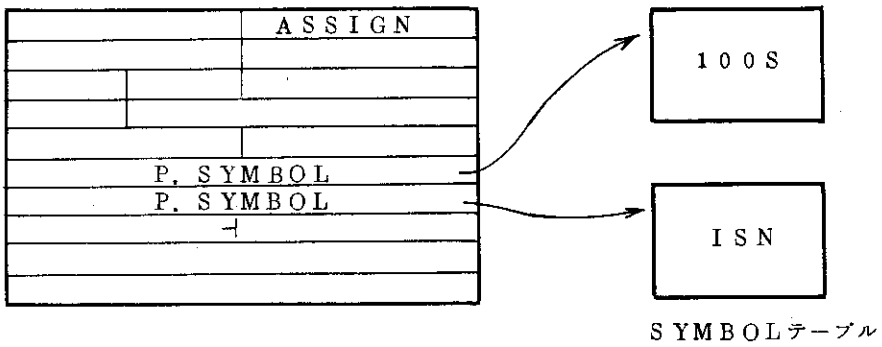
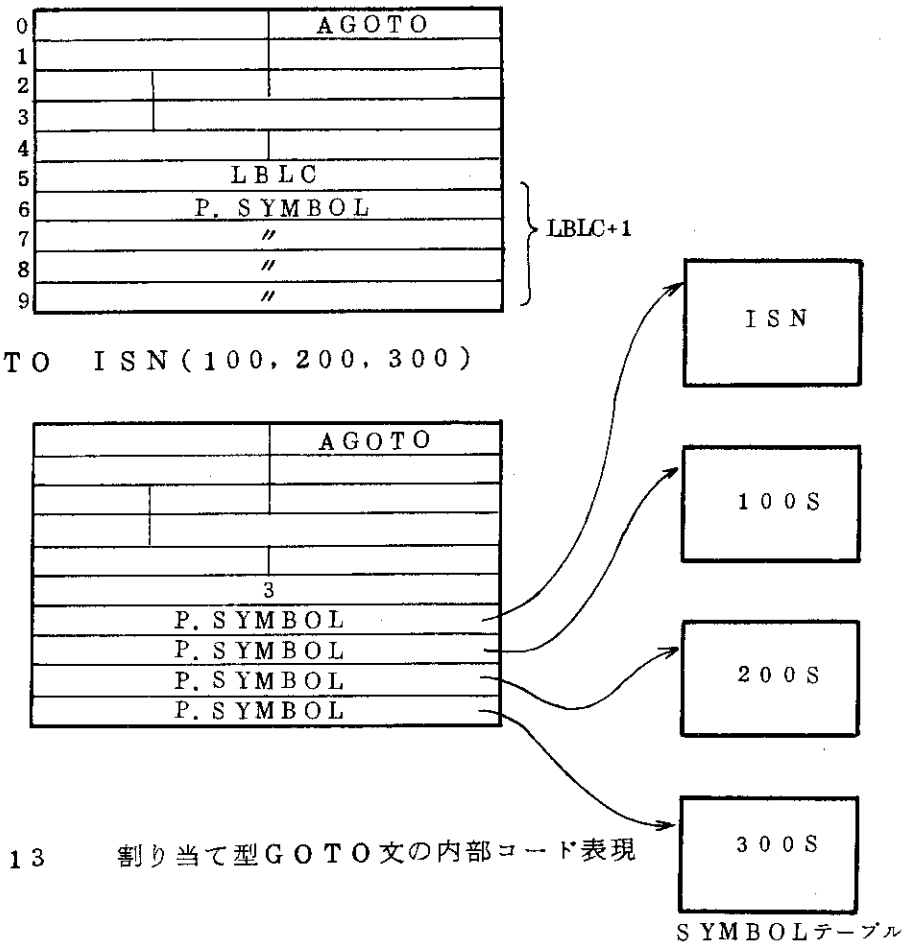
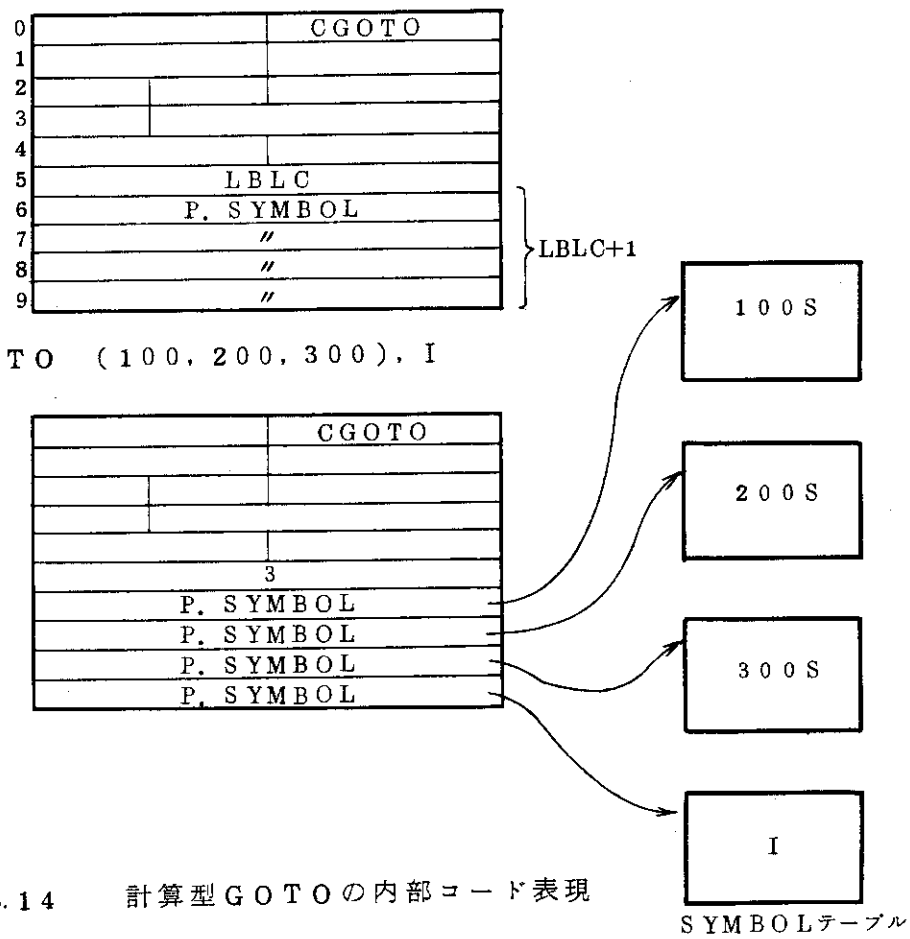


図3.12　ASSIGN文の内部コード表現



例　GOTO　ISN(100, 200, 300)

図3.13　割り当て型GOTO文の内部コード表現



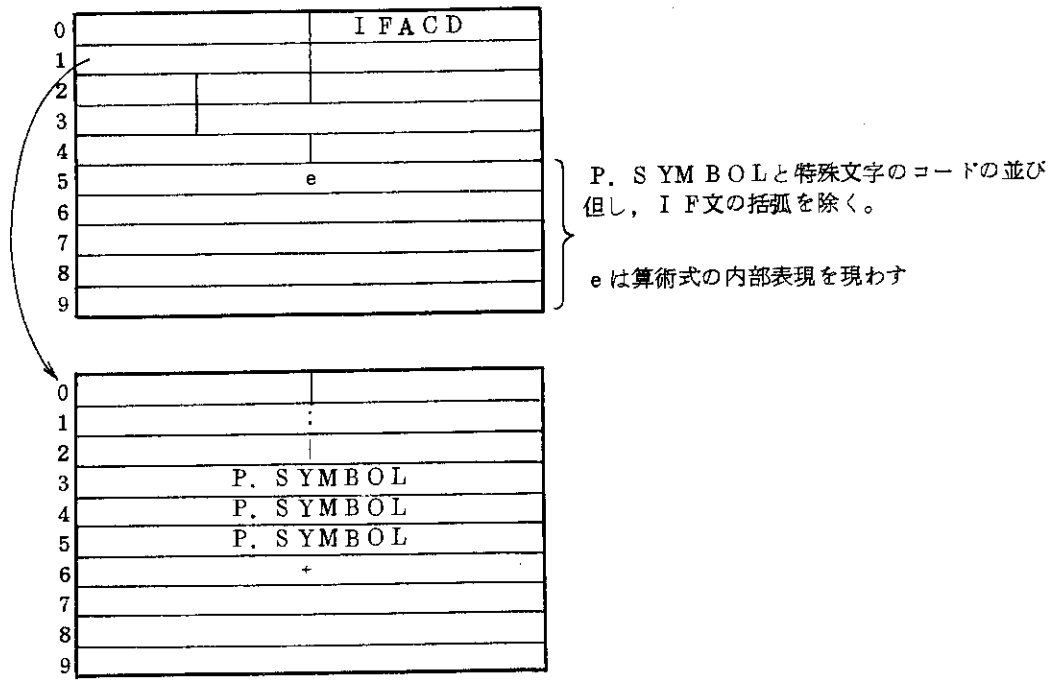
例　GOTO　(100, 200, 300), I

図3.14　計算型GOTOの内部コード表現

3.3.2 ＩＦ文

ＩＦ（ｅ） k_1，k_2，k_3	
	ｅ：整数型か実数型の算術式
	ｋ：文 番 号
ＩＦ（ｅ） Ｓ	
	ｅ：論 理 式
	Ｓ：ＤＯ文，ＩＦ文以外の実行文

ＩＦ文には算術ＩＦ文，論理ＩＦ文がある。両者はＩＦ文の式の終りを示す右括弧の次に現われるものが文番号か変数名かによって区別される。括弧でくくられている式はARTHXPにより解釈される。また論理ＩＦ文の場合，後続の実行文を独立した文として扱う。すなわち，ＩＦSWをＯＮにし，ＰＦＣのＬＢＳＣＡＮに制御を戻す。以後の解釈は文の途中から始める以外は通常のＦＯＲＴＲＡＮ文と同様に扱われる。



例　IF(X−Y)100，200，300

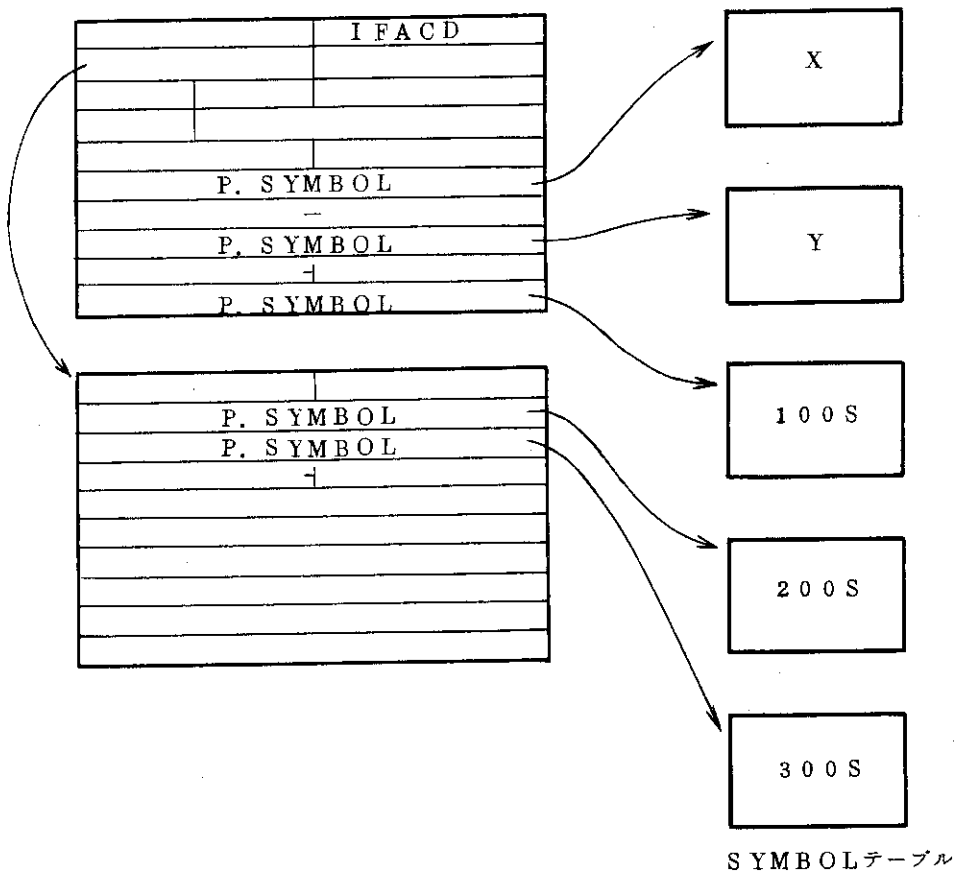
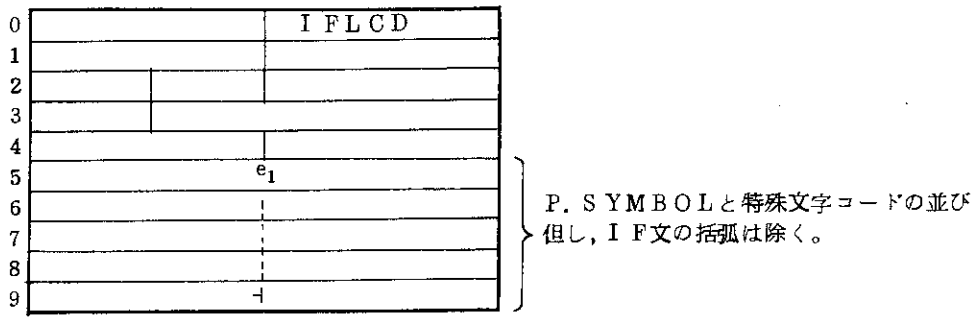
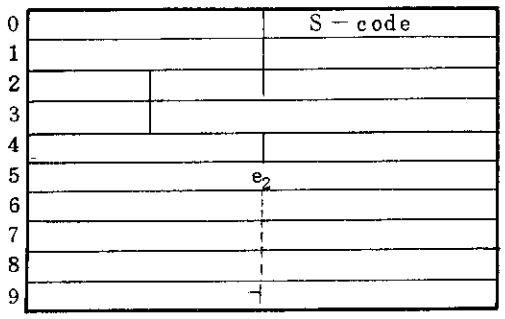


図3.15　算術IF文の内部コード表現





例　IF(A.EQ.B)GOTO 100

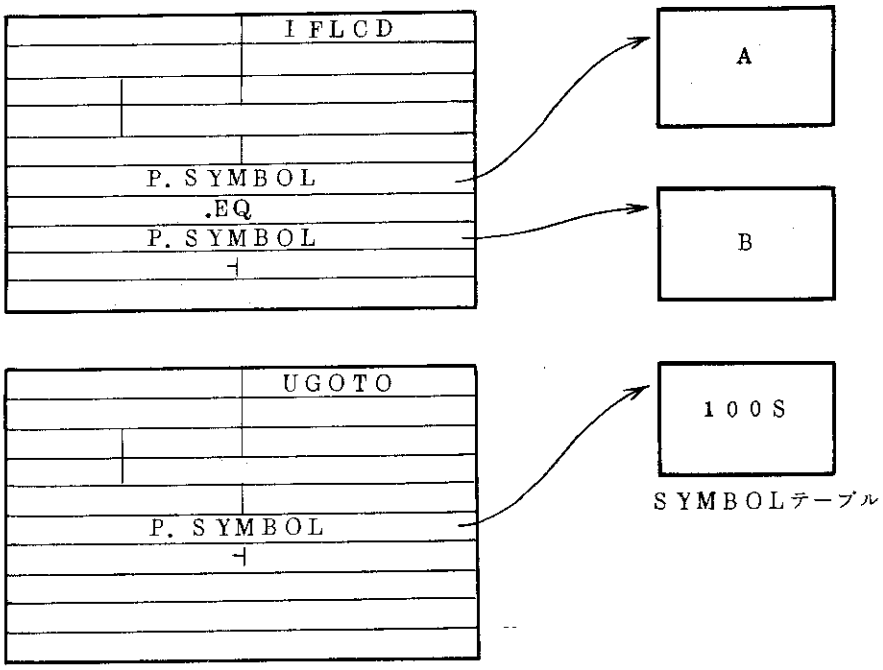


図3.16　論理IF文の内部コード表現

3.3.3 DO文

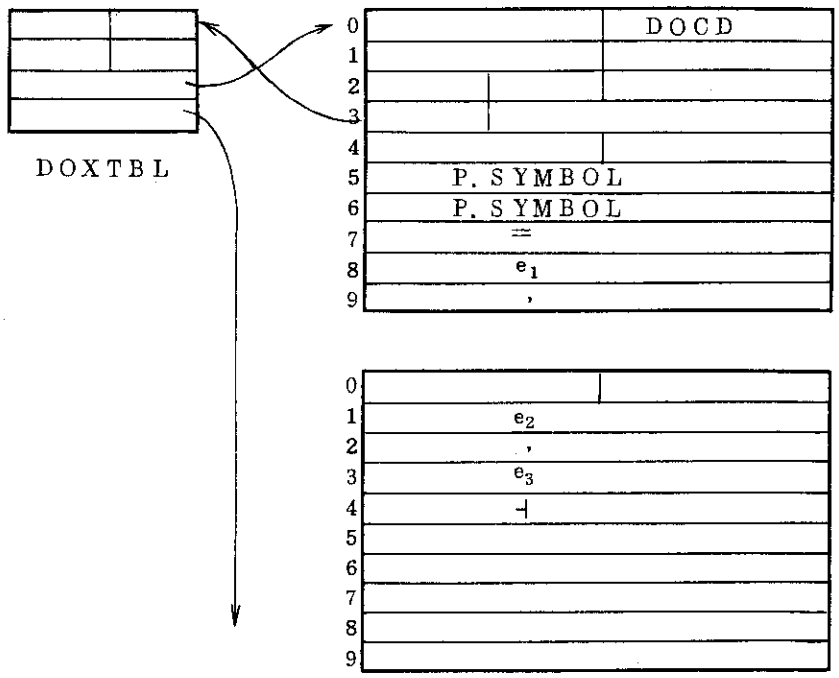
DO　k　$i = m_1, m_2, m_3$	
	k：文番号
	i：整数型の変数名
	m_1, m_2, m_3：整定数かまたは整数型変数の引用

DO文は初期値の次のコンマが現われた時点でDO文であると認識され，DOXTBLを作成する。DOの入れ子の終了はサブルーチンSELECTが端末文を認識するとDOCLSWをONにし，PFCに制御か戻された時点でチェックされ終了の記号を内部テーブルに記録する。

Pointer 1	Pointer 2
Pointer 3	
	P. BNFTXT 1
	P. BNFTXT 2

Pointer 1　：上のレベルを示すDOXTBLへのポインタ
Pointer 2　：同じレベルを示すDOXTBLへのポインタ
Pointer 3　：下のレベルを示すDOXTBLへのポインタ
P. BNFTXT1：Doを宣言しているBNFTXTへのポインタ
P. BNFTXT2：Do終りの文が格納されているBNFTXTへのポインタ

図3.17　DOXTBLの構造



例　DO 123 I＝1, N, M

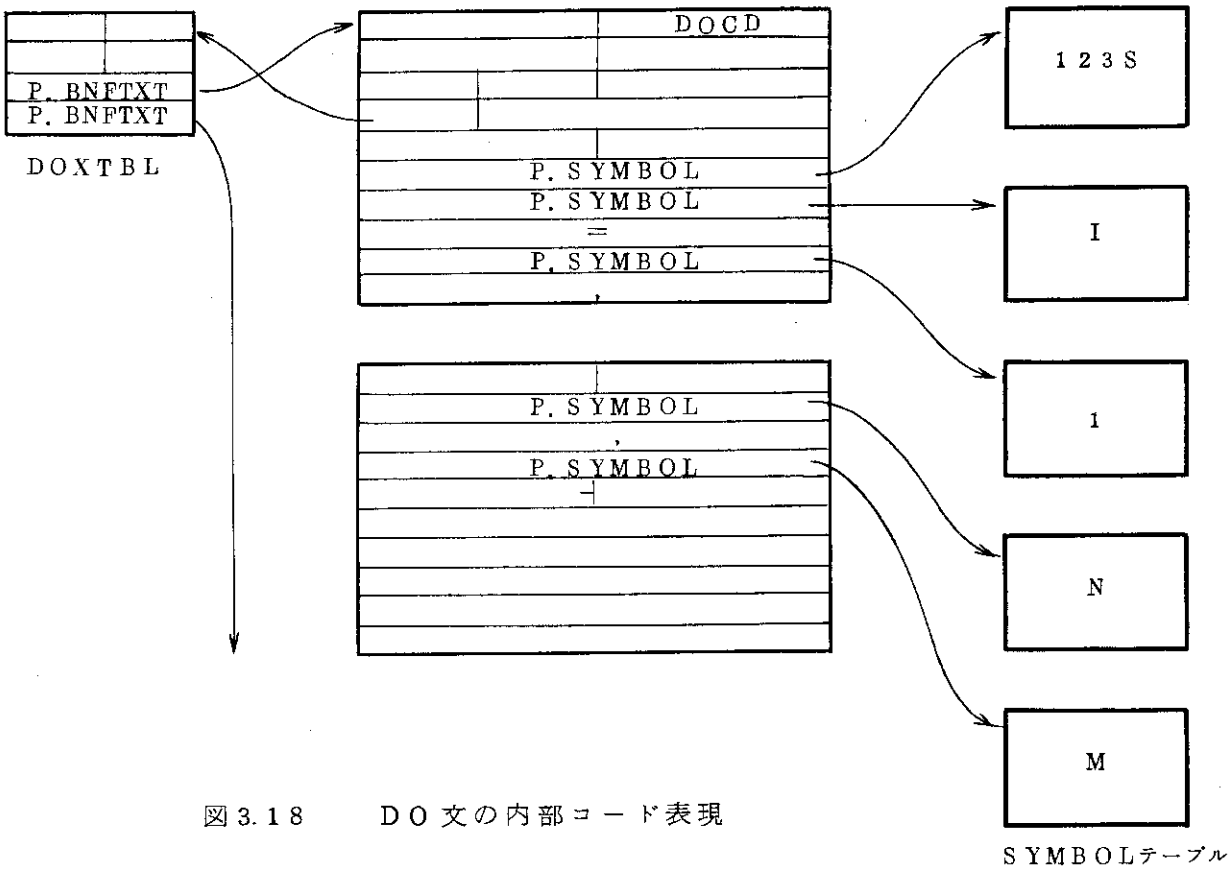


図3.18　DO文の内部コード表現

3.3.4 その他

```
CALL    name( a1, a2, … an )
                name ：サブルーチンの名前
                   a ：実 引 数

RETUEN    i
                   i ：整定数かまたは整変数

CONTINUE

STOP    n
                   n ：5桁以内の整数または文字定数
```

制御文には既に述べたもの以外に

CALL文，
RETURN文，
CONTINUE文，
STOP文，
PAUSE文，

がある。内部コード表現を以下に示す。

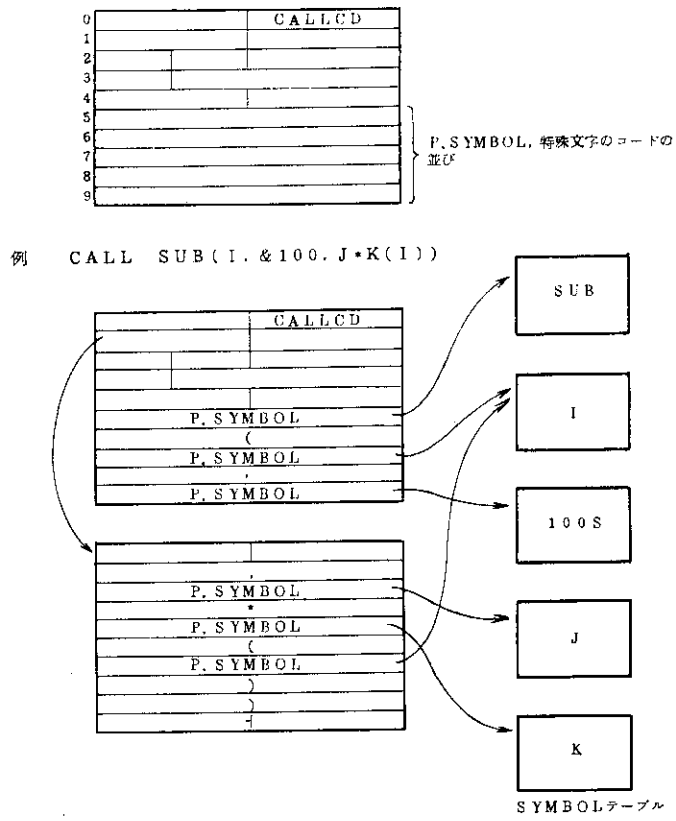
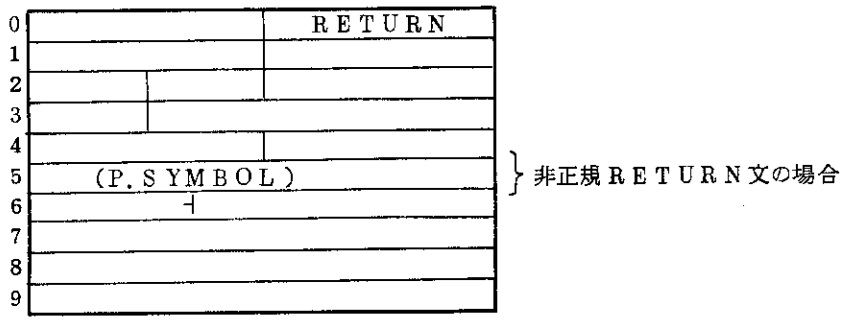


図3.19　CALL文の内部コード表現



例　RETURN　2

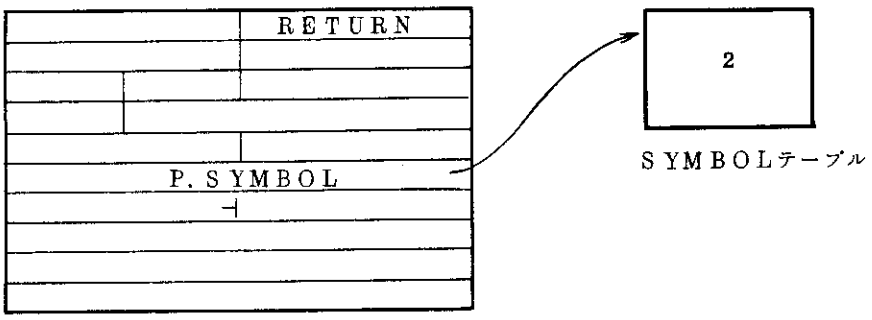
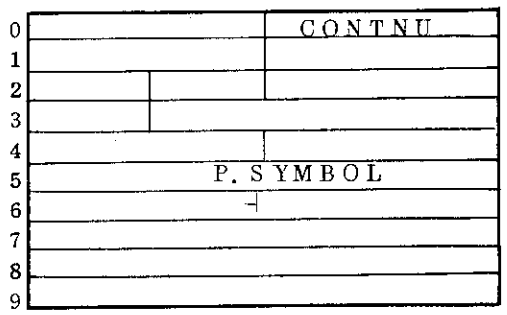


図3.20　RETURN文の内部コード表現



例　1000　CONTINUE

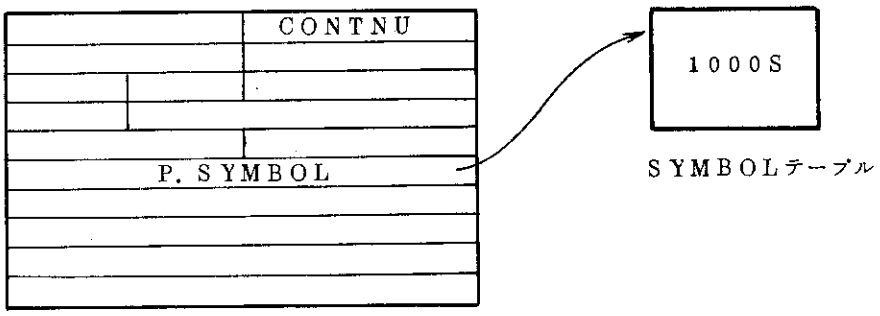
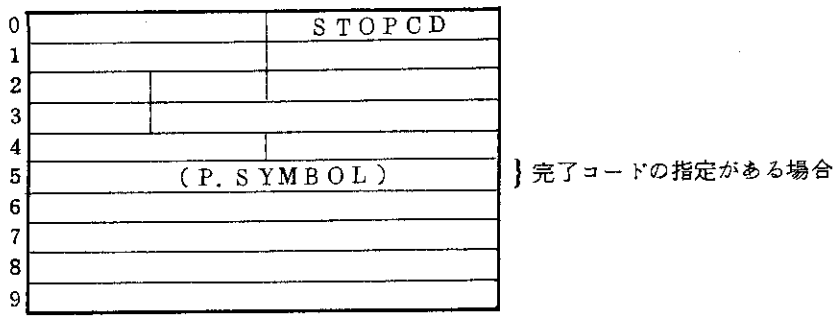


図3.21　CONTINUE文の内部コード表現



例　STOP　100

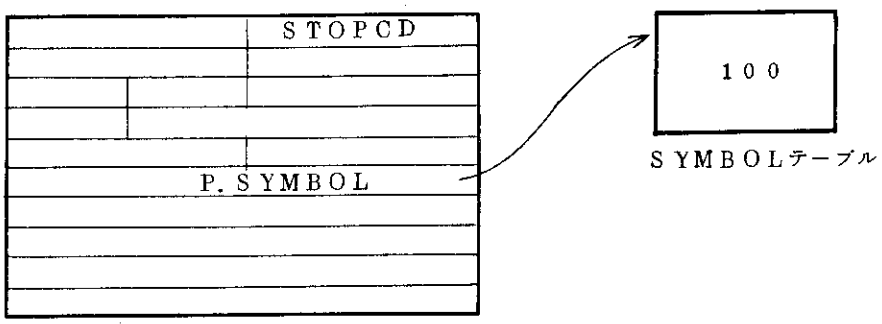


図3.22　STOP文の内部コード表現

3.4 入出力文

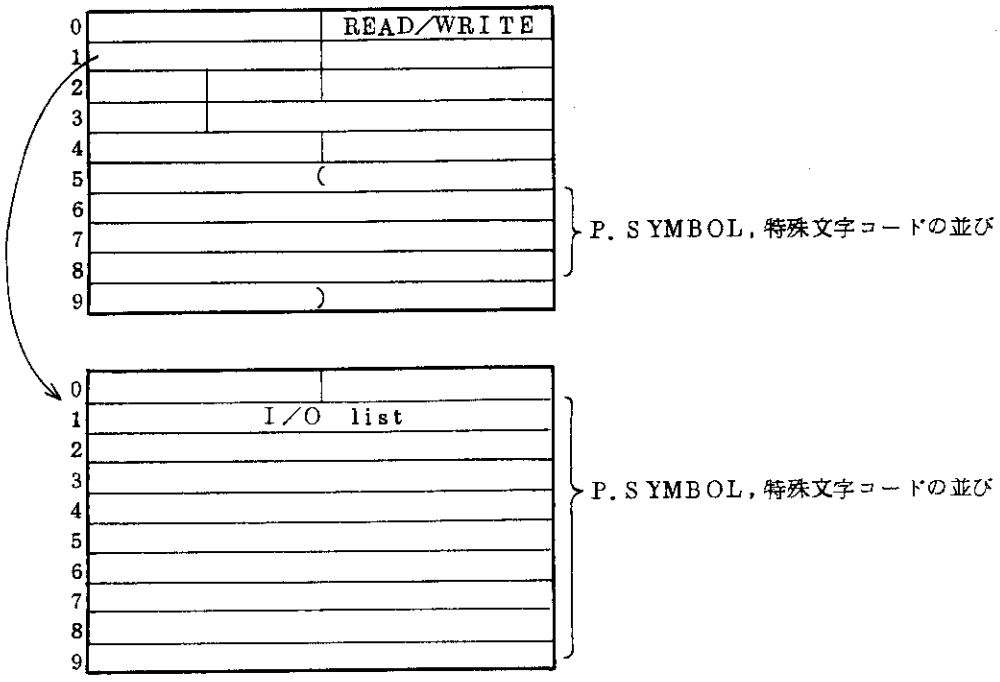
3.4.1 READ/WRITE文

```
READ(u, f, END=n1)k
READ(u)k
READ(u, X, END=n1)
READ(u, *, END=n1)k
READ(U'i, f)k

WRITE(u, f)k
WRITE(u)k
WRITE(u, X)
WRITE(u, *)k
WRITE(u'i, f)k
```

u：1～99までの整数または整数型変数名
f：文番号または整数型変数名
n_1：文番号
X：name list名

READ/WRITE文には書式つき入出力文，書式なし入出力文，NAMELIST名つき入出力文，リストダイレクト入出力文，直接アクセス入出力文がある。SCANにおいては現在これらを区別して扱ってはいない。またREADとWRITE文では左括弧以下の構文が同じであるので，WRITE文の解釈は“WRITE”と認識された時点でWSWをON，TEMPRWをWRITECとしてサブルーチンREADRNに制御を渡している。I/Oリストのチェックは，ARTHXPを呼び出し行なっている。



例　READ(5, 100)I, (A(J), J=1, I)

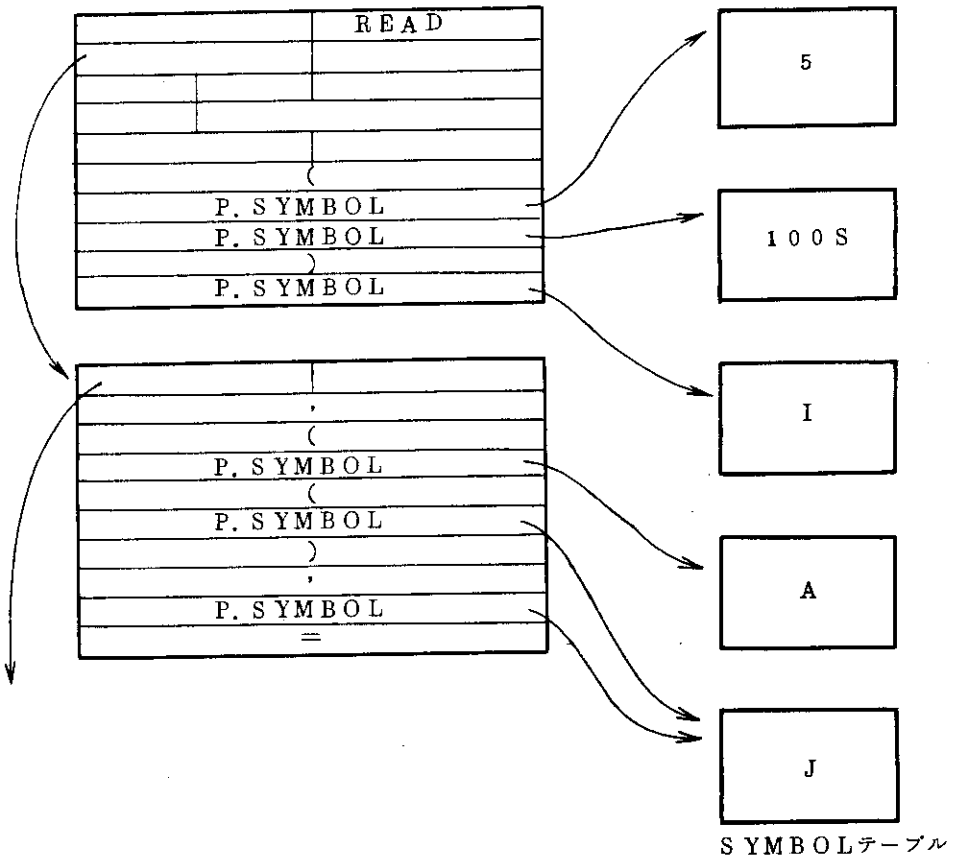


図3.23　入出力文の内部コード表現

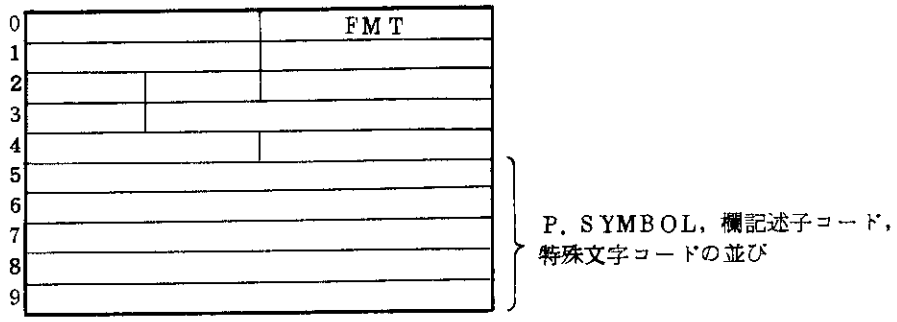
3.4.2 FORMAT文

n FORMAT （書式仕様）

FORMAT文の内部コード表現は定数，欄記述子，区切記号が現われる順に記録される。欄記述子の内部コードを表3.1に示す。

表3.1　欄記述子の内部コード

FORMATコード名	コード		意味
	第1バイト	第2バイト	
A CODE	23	01	文字型
D CODE	23	02	倍精度実数型
E CODE	23	03	実数型
F CODE	23	04	実数型
G CODE	23	05	整数，実数，論理型
H CODE	23	06	文字型
I CODE	23	07	整数型
L CODE	23	08	論理型
Z CODE	23	09	16進型
P CODE	23	0A	桁移動子
Q CODE	23	0B	桁移動子
R CODE	23	0C	文字型
T CODE	23	0D	位置設定
X CODE	23	0E	空白の指定



例　1000 FORMAT(1X, I2, 3(E13.5))

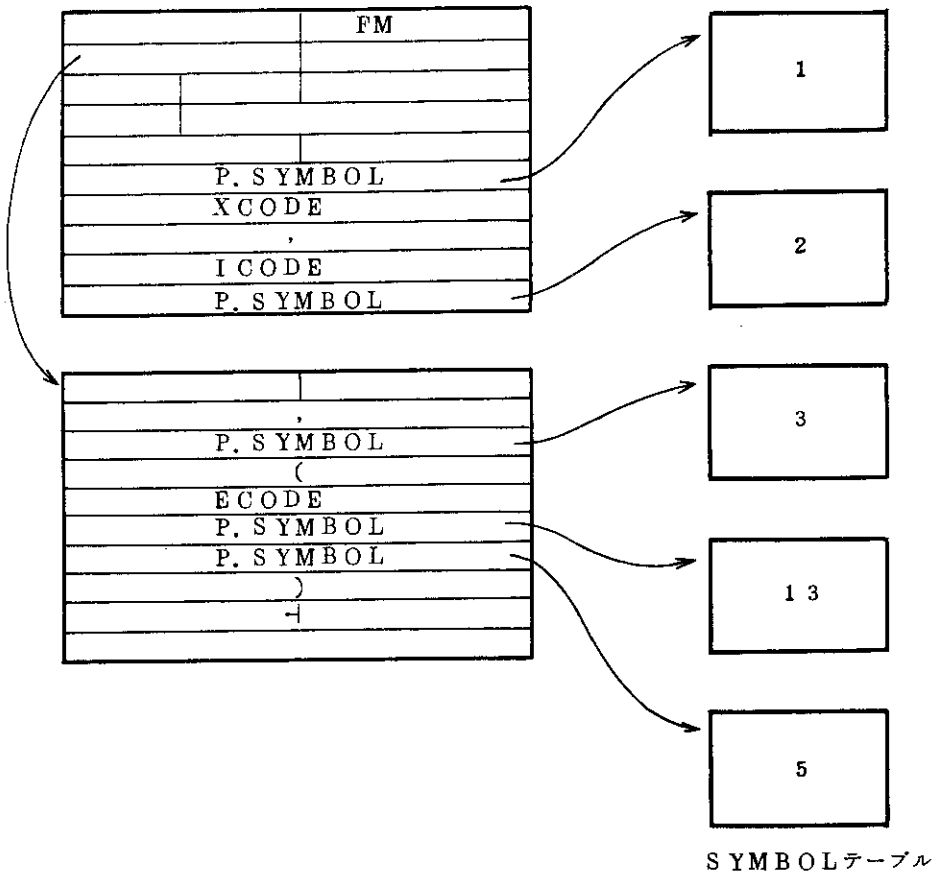


図3.24　FORMAT文の内部コード表現

3.4.3 その他

NAMELIST／x_1／l_1 ／x_2／l_2…… ／x_n／l_n

x：NAMELIST名

l：変数名，配列名の並び

ENDFILE u

u：ファイル参照番号

REWIND u

u：ファイル参照番号

BACKSPACE u

u：ファイル参照番号

DEFINFILE u_1(m_1, r_1, f_1, v_1), …

u：ファイル参照番号

m：レコード数

r：レコードサイズ

f：E／L／U

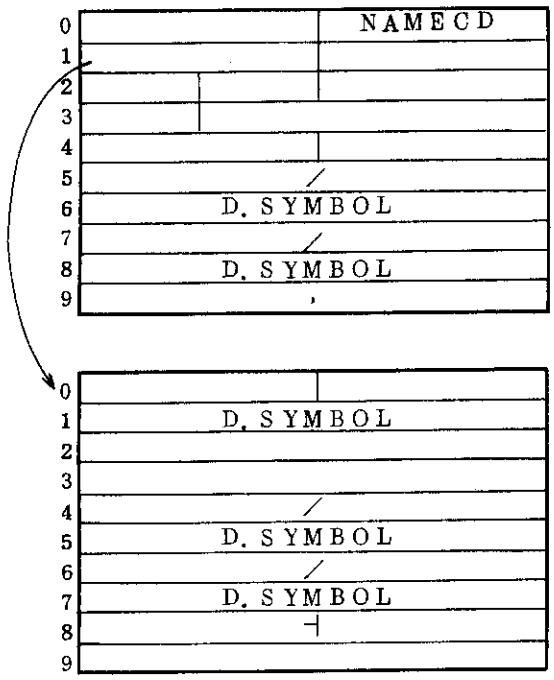
v：結合変数

FIND(u'i)

u：ファイル参照番号

i：記録位置

NAMELIST文，ENDFILE文，BACKSPACE文．DEFINEFILE文，FIND文の内部コード表現を以下に示す。



例　NAME LIST／NML1／X1，X2，X3

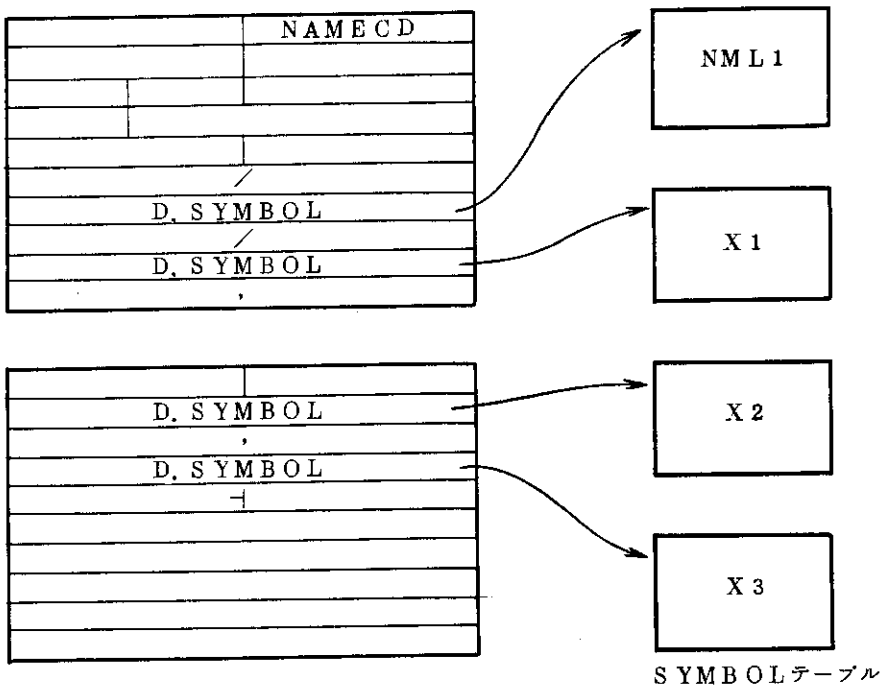
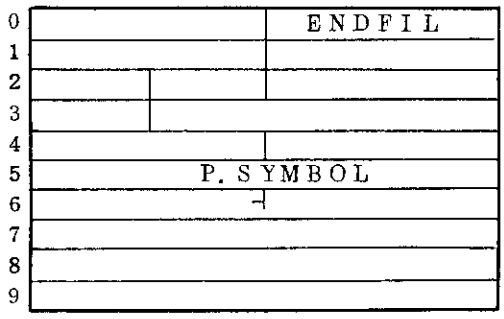


図3.25　NAMELIST文の内部コード表現



例　ENDFILE　1

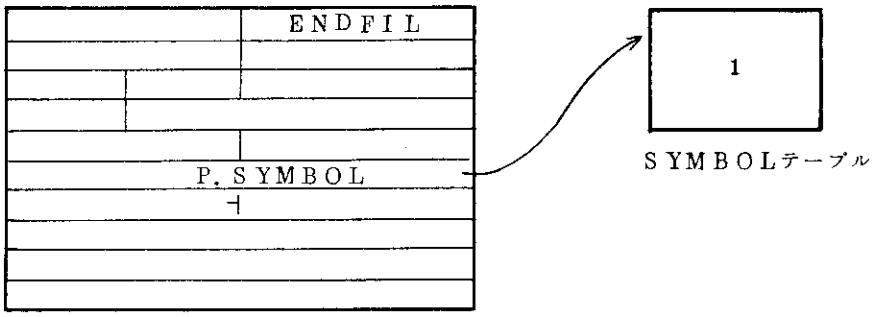
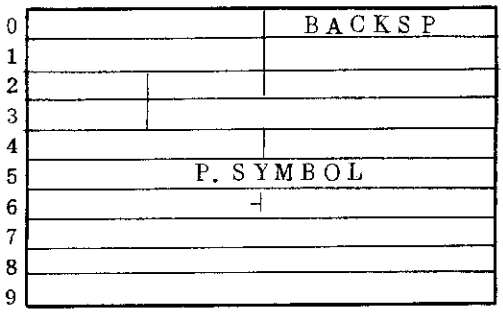


図3.26　ENDFILE文の内部コード表現



例　BACKSPACE　1

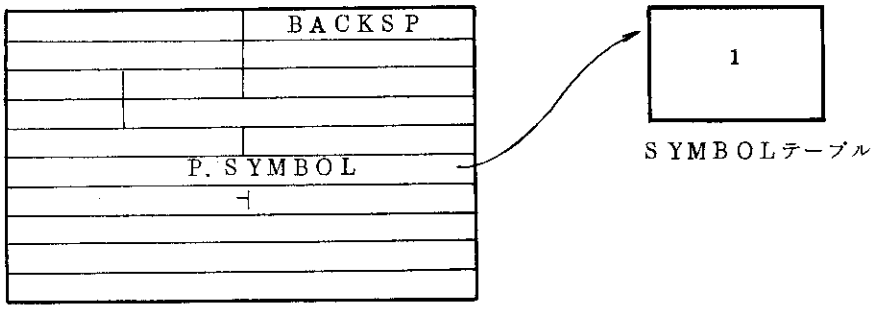
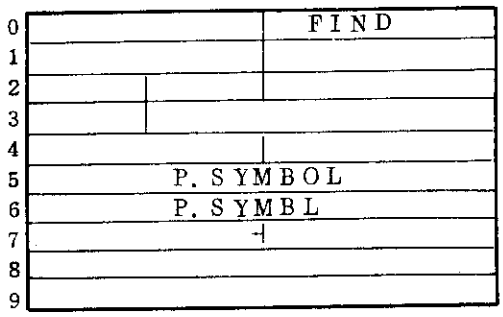


図3.27　BACKSPACE文の内部コード表現



例　FIND(1▼ IX)

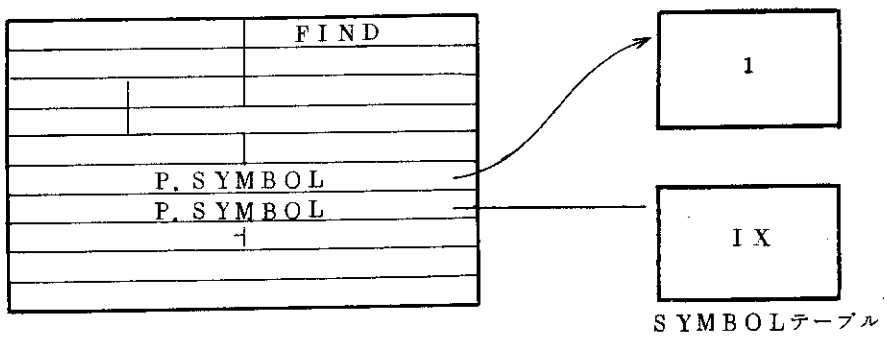
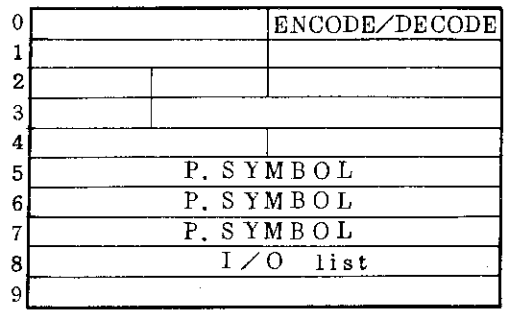


図3.28　FIND文の内部コード表現

3.5 ENCODE／DECODE文

```
ENCODE (C, f, V)k
DECODE (C, f, V)k
                C：文 字 数
                f：文 番 号
                V：変 数 名
                k：入出力並び
```

ENCODE／DECODE文のC，f，Vの解釈はサブルーチンCODECVで行ない，kについてはARTHXPで処理される。



例　ENCODE(N, 100, A)X, Y

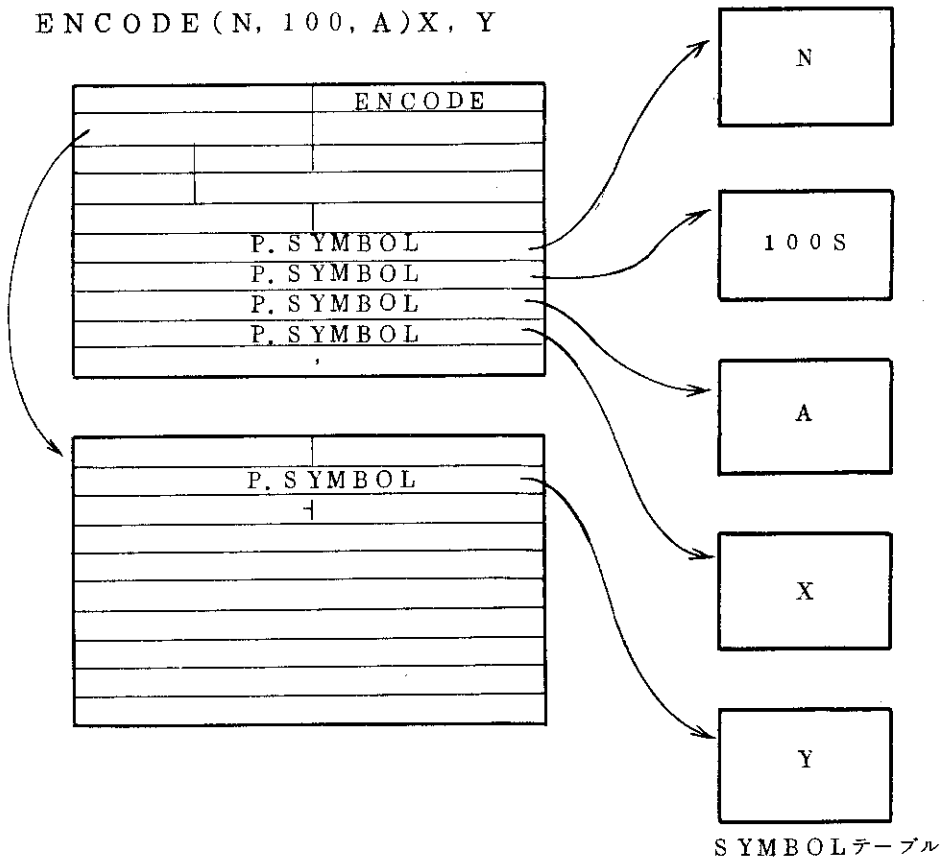


図3.29　ENCODE／DECODE文の内部コード表現

3.6 SUBROUTINE／FUNCTION文

SUBROUTINE	S (a_1 , a_2 , …… a_n)
ENTRY	S (a_1 , a_2 , …… a_n)
type FUNCTION	f (a_1 , a_2 , …… a_n)
	S ：サブルーチン名
	f ：関 数 名
	a ：仮 引 数
	type ：型

SUBROUTINE文はSTOPRNルーチンで認識されBNFTXTの先頭ブロックを作成する引数についてはSUBPROルーチンで処理される。FUNCTION文については型宣言がある場合は，それぞれの型宣言子を認識したルーチンでTYPESWに型をセットした後，FNORNTルーチンでFUNCTIONという文字が現われるか否かをチェックする。現われた場合はFNCSUBルーチンでBNFTXTルーチンの先頭ブロックを作成した後SUBPROルーチンに処理を渡す。

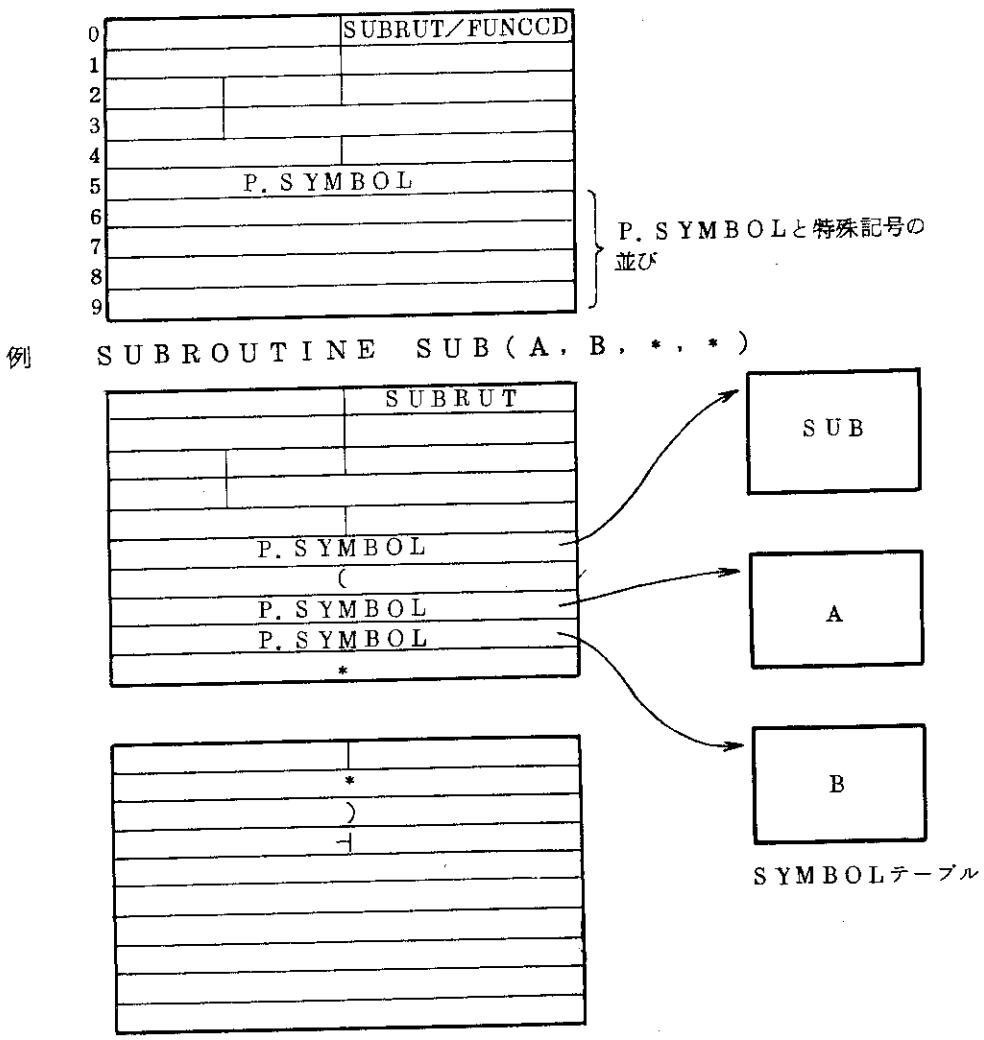


図3.30　SUBROUTINE／FUNCTION文の内部コード表現

4. おわりに

言語はいきものであり，時代と共に変化している。それはプログラミング言語においても例外ではない。計算機の進歩に伴なう計算機の機能の拡大，利用技術の進歩，利用者からの要望などによりFORTRAN言語も拡張されてきている。筆者らがコンパイラ作成を目指した当時と現在のFORTRAN言語ではその機能に大きな差がある。SCANはJIS－7000レベルの言語仕様で作成されたがその後SOPHY，TSSFORT用SCANの作成過程においてFORTRAN－Hの機能が追加されている。

御存知のようにFORTRAN言語の仕様は計算機メーカ毎に異なり，またコンパイラ毎に差がある。SCANにおいてこれらを全て包含することは困難であり，また無意味であろう。FORTRANの構文解析を行なう者はその対象となるコンパイラを明確にする必要がある。

SCANには基本的な解析部が用意されている。また構造的にも追加，修正の容易な構造になっている。言語の拡張，変化に伴なうSCANへの解析部の追加修正は利用者各位でなされたい。どこに追加，修正が必要かは，本書により容易に推測できるものと考えている。

謝　辞

原研用FORTRANコンパイラのための構文解析プログラムSCANを作成してからおよそ10年が経過した。この間SCANはTSSFORT，DATAPOOL，SOPHYへの適用のために少しずつ機能を拡張してきた。SCANの作成，拡張にあたり計算センター平川室長はじめ多くの方から助言，助力をいただいた。ここに深く感謝致します。

参考文献

1. 浅井「順位関数をもつ順位文法」，JAERI－M 4168，1970年10月
2. 浅井，富山「GPL－Genken Programming Language」，JAERI－M 4762，1972年2月
3. Halstead M.H. “Software physics : basic principles”，RJ 1582，IBM Research，Yorktown Hights，N.Y.，1975
4. Ann Fitzsimmos and Tom Love “A Review and Evaluation of Software Science”，
   Computing Surveys，Vol.10，No.1（March 1978），3－18
5. 浅井，富山，田子，松浦「科学技術計算におけるソフトウェアの生産性について」，第21回プログラミング・シンポジューム報告集，情報処理学会

4. おわりに

言語はいきものであり，時代と共に変化している。それはプログラミング言語においても例外ではない。計算機の進歩に伴なう計算機の機能の拡大，利用技術の進歩，利用者からの要望などによりFORTRAN言語も拡張されてきている。筆者らがコンパイラ作成を目指した当時と現在のFORTRAN言語ではその機能に大きな差がある。SCANはJIS－7000レベルの言語仕様で作成されたがその後SOPHY，TSSFORT用SCANの作成過程においてFORTRAN－Hの機能が追加されている。

御存知のようにFORTRAN言語の仕様は計算機メーカ毎に異なり，またコンパイラ毎に差がある。SCANにおいてこれらを全て包含することは困難であり，また無意味であろう。FORTRANの構文解析を行なう者はその対象となるコンパイラを明確にする必要がある。

SCANには基本的な解析部が用意されている。また構造的にも追加，修正の容易な構造になっている。言語の拡張，変化に伴なうSCANへの解析部の追加修正は利用者各位でなされたい。どこに追加，修正が必要かは，本書により容易に推測できるものと考えている。

謝　辞

原研用FORTRANコンパイラのための構文解析プログラムSCANを作成してからおよそ10年が経過した。この間SCANはTSSFORT，DATAPOOL，SOPHYへの適用のために少しずつ機能を拡張してきた。SCANの作成，拡張にあたり計算センター平川室長はじめ多くの方から助言，助力をいただいた。ここに深く感謝致します。

参考文献

1. 浅井「順位関数をもつ順位文法」，JAERI－M 4168，1970年10月
2. 浅井，富山「GPL－Genken Programming Language」，JAERI－M 4762，1972年2月
3. Halstead M.H. “Software physics : basic principles”，RJ 1582，IBM Research，Yorktown Hights，N.Y.，1975
4. Ann Fitzsimmos and Tom Love “A Review and Evaluation of Software Science”，
   Computing Surveys，Vol.10，No.1（March 1978），3－18
5. 浅井，富山，田子，松浦「科学技術計算におけるソフトウェアの生産性について」，第21回プログラミング・シンポジューム報告集，情報処理学会

4. おわりに

言語はいきものであり，時代と共に変化している。それはプログラミング言語においても例外ではない。計算機の進歩に伴なう計算機の機能の拡大，利用技術の進歩，利用者からの要望などによりFORTRAN言語も拡張されてきている。筆者らがコンパイラ作成を目指した当時と現在のFORTRAN言語ではその機能に大きな差がある。SCANはJIS－7000レベルの言語仕様で作成されたがその後SOPHY，TSSFORT用SCANの作成過程においてFORTRAN－Hの機能が追加されている。

御存知のようにFORTRAN言語の仕様は計算機メーカ毎に異なり，またコンパイラ毎に差がある。SCANにおいてこれらを全て包含することは困難であり，また無意味であろう。FORTRANの構文解析を行なう者はその対象となるコンパイラを明確にする必要がある。

SCANには基本的な解析部が用意されている。また構造的にも追加，修正の容易な構造になっている。言語の拡張，変化に伴なうSCANへの解析部の追加修正は利用者各位でなされたい。どこに追加，修正が必要かは，本書により容易に推測できるものと考えている。

謝　辞

原研用FORTRANコンパイラのための構文解析プログラムSCANを作成してからおよそ10年が経過した。この間SCANはTSSFORT，DATAPOOL，SOPHYへの適用のために少しずつ機能を拡張してきた。SCANの作成，拡張にあたり計算センター平川室長はじめ多くの方から助言，助力をいただいた。ここに深く感謝致します。

参考文献

1. 浅井「順位関数をもつ順位文法」，JAERI－M 4168，1970年10月
2. 浅井，富山「GPL－Genken Programming Language」，JAERI－M 4762，1972年2月
3. Halstead M.H. "Software physics : basic principles"，RJ 1582，IBM Research，Yorktown Hights，N.Y.，1975
4. Ann Fitzsimmos and Tom Love "A Review and Evaluation of Software Science"，
   Computing Surveys，Vol.10，No.1（March 1978），3－18
5. 浅井，富山，田子，松浦「科学技術計算におけるソフトウェアの生産性について」，第21回プログラミング・シンポジューム報告集，情報処理学会

6. 富山，浅井「データプールの概念と機能」，JAERI－M 8715，1975年3月
7. James R. Bell "The Quadratic Quotient Method : A Hash Code Eliminating Secondary Clustering"
CACM, Vol. 13, No. 2 (February 1970), 107－109
8. FACOM 230 M－Ⅶ FORTRAN Ⅳ－H 文法書