資　料

マイクロコンピュータにおける
ＰＡＳＣＡＬコンパイラ
取　扱　い　説　明　書

昭和56年3月

財団法人　日本情報処理開発協会

この報告書は、日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて、昭和５５年度に実施した「マイクロコンピュータの応用に関する調査研究」の一環としてとりまとめたものであります。

目　　次

1、概　説

プログラミング言語ＰＡＳＣＡＬは、ＡＬＧＯＬの精神を受け継いだ言語として、Prof. Dr. Niklaus Wirthによって１９６８年に設計された（標準ＰＡＳＣＡＬ）。実際のコンパイラは１９７０年にＣＤＣ６０００上で実現され、以後、他機種に対しての移植を目的とするＰＡＳＣＡＬ－Ｐ1、Ｐ2、Ｐ3、Ｐ4コンパイラが開発された。

一方、マイクロコンピュータ上で走るＰＡＳＣＡＬは、１９７８年にKenneth. L. Bowlesによって開発された。このシステムは、ＵＣＳＤ　ＰＡＳＣＡＬシステムと呼ばれ、６４Ｋバイトの主記憶とフロッピーディスクだけの非常に小さな環境の中で動作可能とした点が画期的である。

ＭＣＣ　ＰＡＳＣＡＬは、Ｐ4コンパイラをベースに、８０８０相当のマイクロプロセッサ、６４Ｋバイトの主記憶、フロッピーディスクから構成されるシステムのＣＰ／Ｍ上で走ることを可能にするとともに、言語仕様としていくつかの拡張を行い、また、システムとして８０８０へのトランスレータを備えるなどユーザにとってかなり使い易いシステムとなっている。

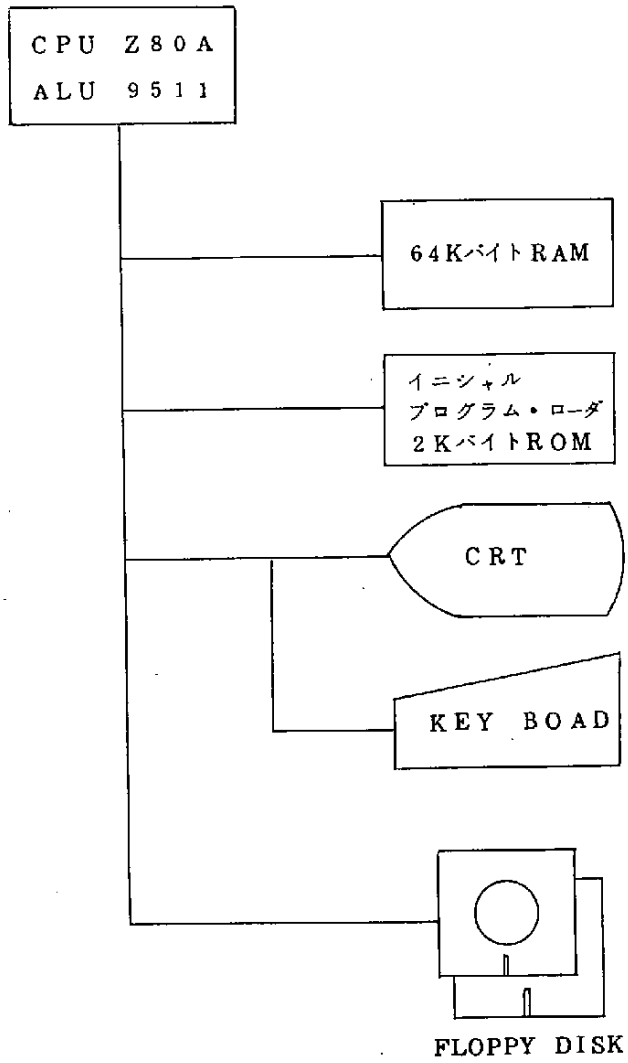
＊　Niklaus Wirth：Institut für Informatik KTH Zürich

＊　Kenneth L Bowles：Institute for Information Systems University of Califernio at San Diego

＊　UCSD PASCAL：Tread mark of the Regent of University of California

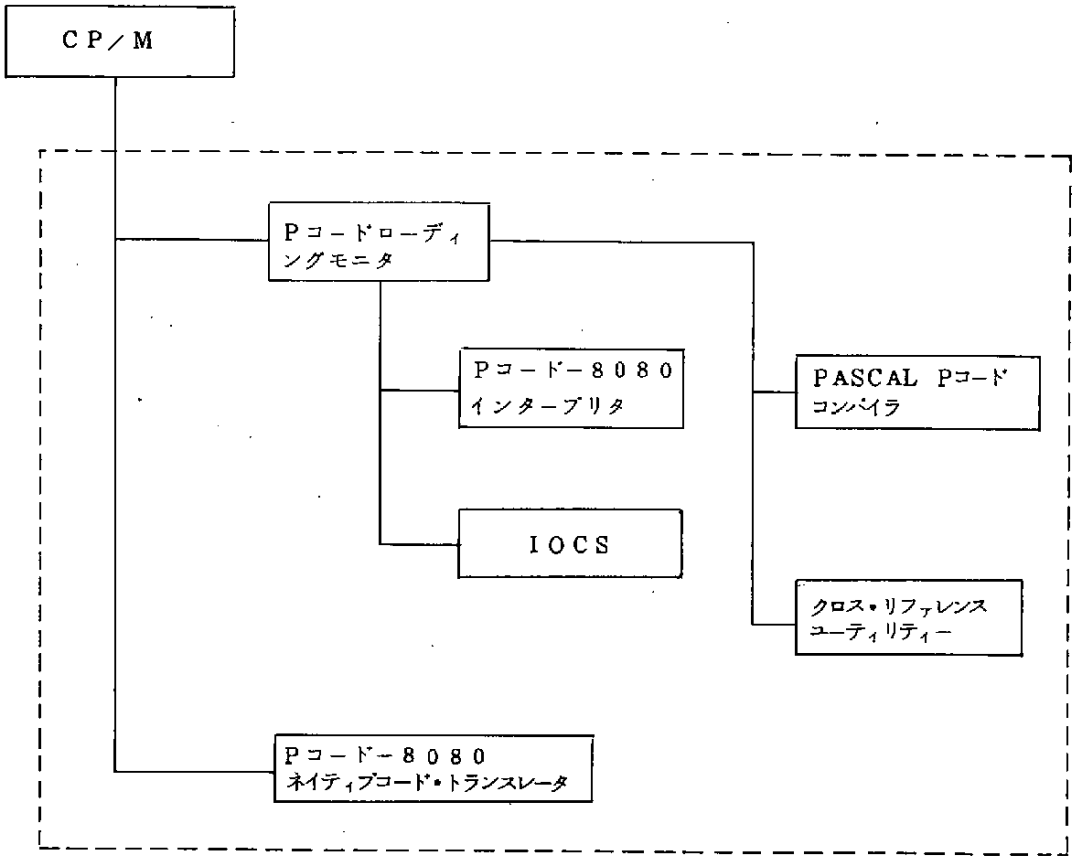
1.1　機器構成

MCC　PASCALを使用するためのハードウェア構成を示す。



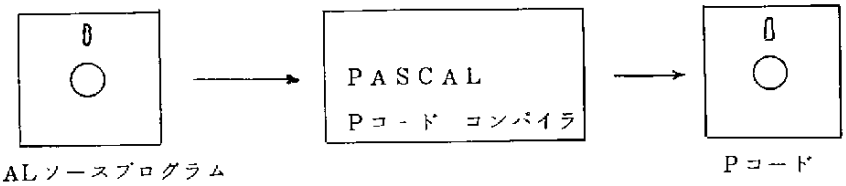
1.2　ソフトウエア構成

本システムのソフトウェアは次図のとおり構成されている。



(1) PASCAL Pコード コンパイラ

PASCAL Pコード コンパイラ（以下コンパイラ）は，MCC PASCAL 文法に従って記述されたソース・プログラムをPコードに変換するプログラムである。



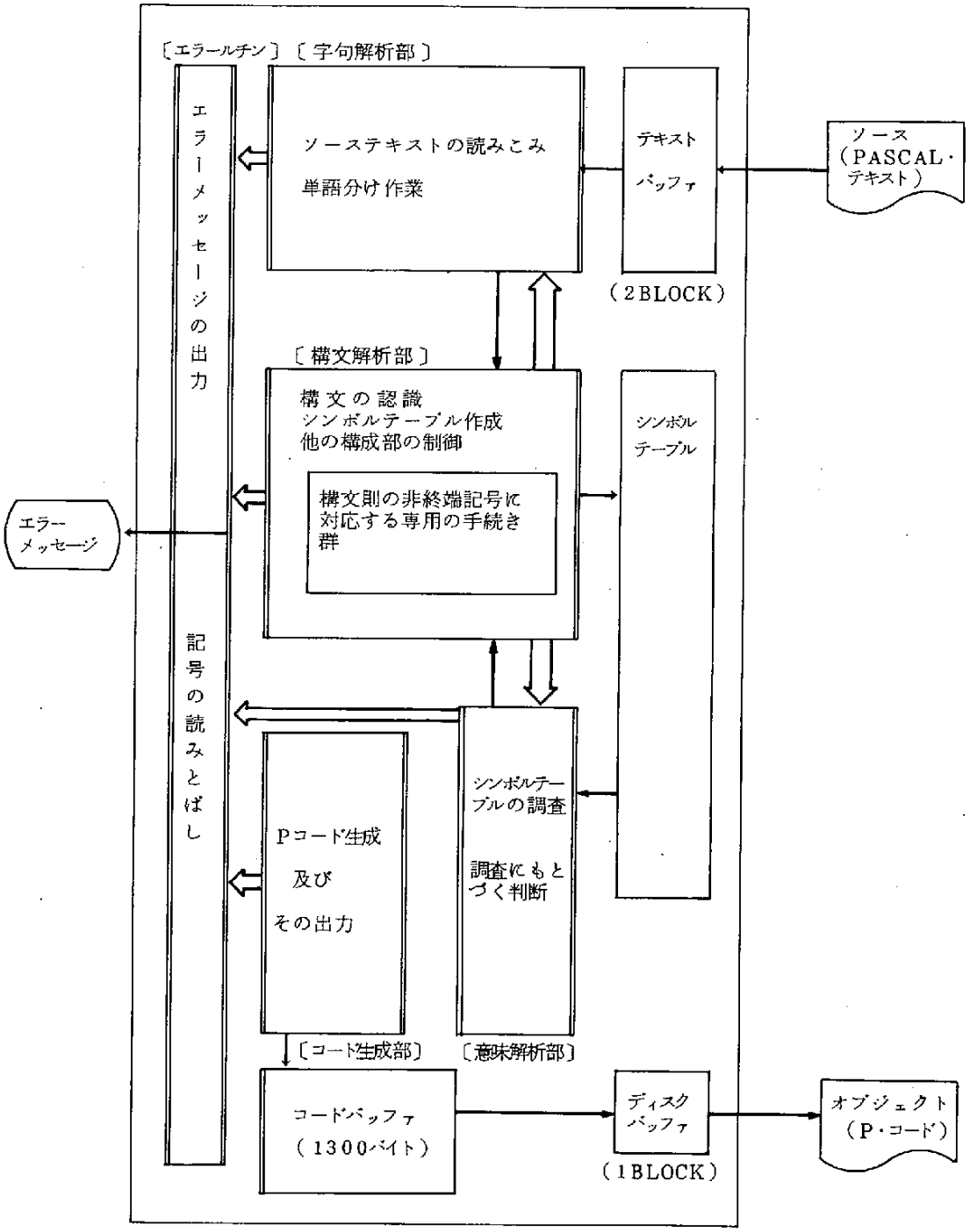
PASCALソースプログラム　　Pコード

コンパイラは，大きく分けて，構文解析部，意味解析部，コード生成部から成る。

構文解析部は，ソース・プログラムからテキストを1文字づつ読み取り，PASCAL 構文フローに基き文法のチェックを行う。

意味解析部では，構文解析部でチェックされた各文の意味づけを行う。意味づけされた文は，コード生成部で，Pコードに変換され出力される。

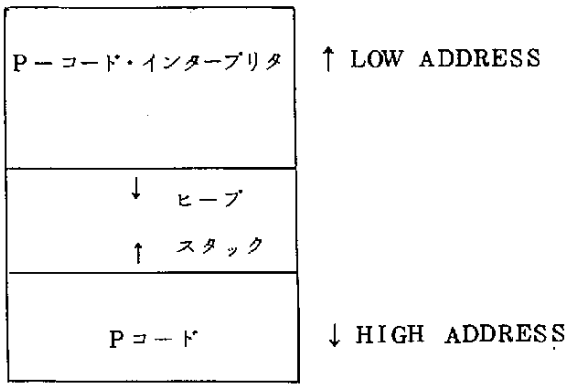
コンパイラの論理構成



(注) 1 BLOCK=512バイト

(2) Pコード－８０８０インタープリタ

コンパイラから出力されるPコードは，仮想されたスタック・マシンの機械語である。従ってPコードを直接実行できる機械は数少く，ほとんどの機械では，Pコードを解釈実行する機械をプログラムで行わなければならない。このように，Pコードを解釈実行するプログラムをPコード・インタープリタと呼ぶ。



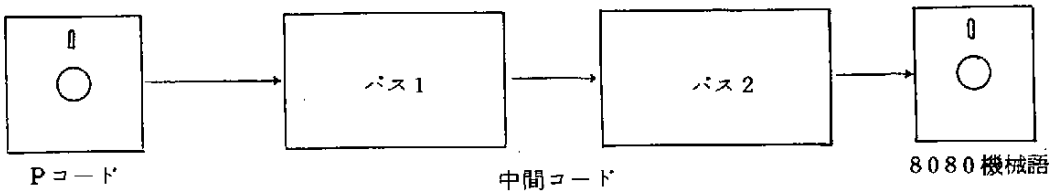
Pコード・インタープリタは，Pコード解読部と各Pコードに対応した実行部で構成される。解読部は，仮想マシンの命令カウンタ（IPC）の指すPコードを取り，そのコードに対応した実行ルーチンに制御を渡す。実行部は，そのコードに対応した機能を果し，制御を解読部に返す。

(3) Pコード－８０８０ネイティブコード　トランスレータ

トランスレータは，Pコードを入力として，それに等価な8080機械語の並びを出力する。

トランスレータは，２パス方式をとり，パス１で飛び先番地テーブルの作成及び局所最適化を含んだ命令カウンタの計数を行い，中間コードを生成する。パス２では，パス１で生成された中間コードを入力として読み，番地修正などの加工を行い，8080機械語（ネイティブ・コード）を出力する。

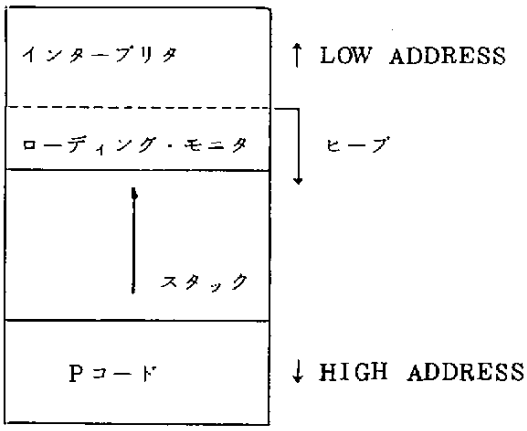
トランスレータで生成された機械語は，ランタイム・ルーチンを付加して実行可能となる。



(4) Pコード　ローディング　モニタ

ローディング・モニタは，CP/MのCCPとインターフェイスをとり，仮想スタック・マシンの初期化，Pコード又はネイティブ・コードのロードと，ランタイム・ルーチンの付加を行う。

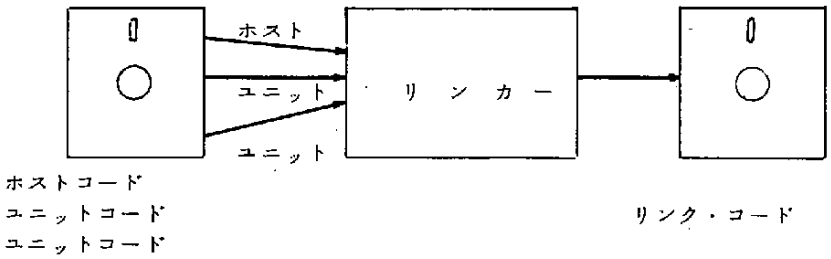
ローディング・モニタは，Pコード・インタープリタと同居しており，Pコード実行時（インタープリタに制御が移った時）に，フリー・エリアとして解放される。



(5) リンカ

リンカは，分割してコンパイルされた複数のプログラムを結合し，一本の完成されたプログラム（Pコード・ファイル）を生成する。

リンカは，分割コンパイルされた手続部分共通領或の参照オフセット，JUMPオフセット，プロセジャ・ディクショナリの書き替を行い一本のPコード・ファイルを出力する。



１．３　システムの起動

(1)　本体の電源釦をＯＮにする。

（電源ランプ及びＲＵＮインジケータが点灯する。）

(2)　ＣＲＴの電源釦をＯＮにする。

（電源インジケータが点灯する。）

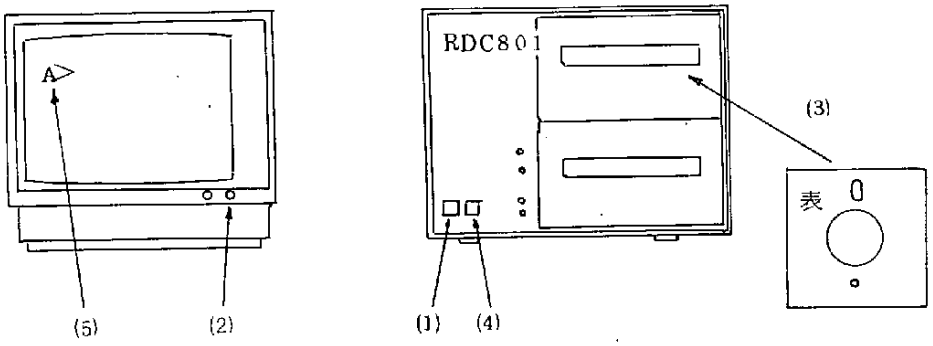
(3)　上段のフロッピーディスク・ドライブ（以下ドライブＡ）に、ＣＰ／Ｍディスケットを挿入する。

フロッピー・ディスクは、表を上にして挿入する。

（ＲＥＡＤＹインジケータが点灯する。）

(4)　ＩＰＬ釦を押下するとＣＰ／Ｍのロードを開始する。

(5)　ＣＰ／Ｍのロードを完了すると、ディスプレイ上にサインオン・メッセージ及びプロンプト・ライン　Ａ＞　が表示される。



１．４　コンパイル操作

(1)　ＣＰ／Ｍエディタを使用し、ＰＡＳＣＡＬソース・プログラムを作成する。

作成したソース・プログラムは、'ＰＡＳ'を第二識別名として持つファイルとすること。

例

Ａ＞ＥＤ　ＳＡＭＰＬＥ．ＰＡＳ

(2)　ＪＯＢコマンドで、ＰＡＳＣＡＬローディング・モニタとコンパイラを呼び出す。

Ａ＞ＰＡＳＣＡＬ△ＣＯＭＰＩＬＥＲ

(3)　コンパイラに実行が移ると、入出力ファイル及びリスティング機器の指定を要求するプロンプトが表示される。

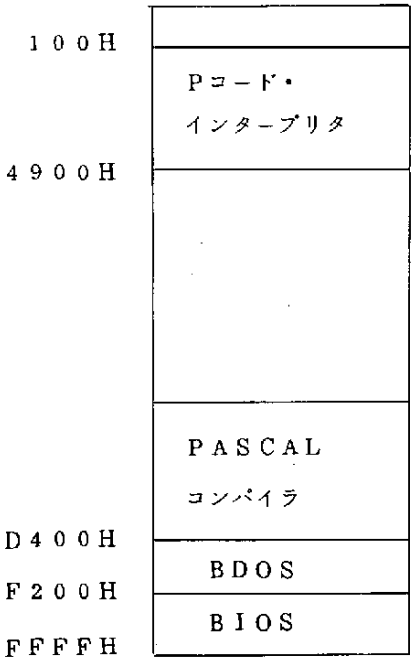
要求に従い、キー・ボードからソース・ファイル名、オブジェクト・ファイル名、リスティング・ファイル名の指定を行う。

例

```
    WHAT  TEXT  FILE: SAMPLE. PAS↙
TO  WHAT  CODE  FILE: SAMPLE. PCD↙
TO  WHAT  LIST  FILE: CON:
```

前記例では、ＰＡＳＣＡＬで書かれたソース・ファイル'ＳＡＭＰＬＥ．ＰＡＳ'を入力とし、'ＳＡＭＰＬＥ．ＰＣＤ'をＰコード・オブジェクト・ファイルとして出力する。

コンパイル・リストは、最後に指定されたＣＯＮＳＯＬＥに出力される。



コンパイル時メモリ・マップ

１．５　トランスレート操作

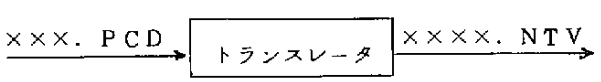
(1) ＰＡＳＣＡＬコンパイラで作成された'Ｐコード'を８０８０の機械語に変換する。

(2) ＪＯＢコマンドでトランスレータと被変換Ｐコード・ファイルを指定する。

例

```
A>TRNSLT  SAMPLE. PCD↙
```

(3) 指定したＰコードファイルを入力として読み込み、Ｐコード・ファイルと同じファイル名に、第二識別名'ＮＴＶ'を付けたネーティブ・コード・ファイルを出力する。



例では、ＳＡＭＰＬＥ．ＰＣＤを入力ファイルとし、ＳＡＭＰＬＥ．ＮＴＶをネーティブ・コード・ファイルとして出力する。

１．６　Ｐコードの実行操作

(1) ＰＡＳＣＡＬローディング・モニタと実行すべきＰコードファイルを呼び出す。

```
A>PASCAL  SAMPLE. PCD↙
```

１．７　ネーテイブ・コードの実行操作

(1) ＲＵＮタイム・ルーチンと実行すべきネーティブ・コードファイルを呼び出す。

```
A>PASCAL  SAMPLE. NTV↙
```

１．８　リンカーの操作

(1) 分割コンパイルされたＰコードを結合し、実行可能なＰコード・ファイルを作成する。

```
A>PASCAL  LINK. PCD↙
```

(2) プロンプトに従い、メイン・プログラム・ファイル名及び出力ファイル名を指定する。

```
WHAT  MAIN  FILE:SAMPLE. PCD↙
TO  WHAT  LINKED:FILE:GO. PCD↙
```

１．９　クロス・リファレンス出力操作

(1) ＰＡＳＣＡＬ、ソース・ファイルを入力し、クロス・リファレンス・リストを出力する。

```
A>PASCAL  CROSREF. PCD↙
```

(2) プロンプトに従い入力ファイルを指定する。

```
WHAT  TEXT. FILE:SAMPLE. PAS↙
```

(3) 出力として、左から順に、名前、定義した行番号、参照している行番号を印字する。

<名前>　<定義行番号>　<参照行番号>　<……>

2、MCC PASCAL 言語仕様

2.1 文 法

MCC PASCALは、Kathleen Jensen & Niklaus Wirthの'PASCAL USER MANUAL AND RERORT'に記されている標準PASCALに準じる。

2.2以降では、MCC PASCALでの制限、及び拡張について記す。

2.2 制 限

2.2.1 整 数

整数型のとり得る値の範囲は、16ビット(−32768～32767)とする。

2.2.2 実 数

実数型のとり得る値の範囲は、32ビット($\pm 3.4028 \times 10^{\pm 19}$)とする。

実数型の入出力は、有効桁6けたのをE型表現とする。

2.2.3 Set型

Set型で、扱い得る要素は、4,080とする。

2.2.4 動的メモリ割り付け

動的メモリ割り付けは、MARK及びRELEASEに依る。

2.2.5 GOTO

GOTOは、同一ブロック内のラベルに対してのみ有効とする。

2.2.6 手続き及び関数

手続き、または関数の引数として、手続き名、関数名を引き渡すことができない。

2.3 拡 張

2.3.1 ランダム・アクセス・ファイル

ディスク・ファイル内の特定レコードに対し、ファイル・ポインタをセットする標準関数'SEEK'を備えている。

SEEKは、引数としてファイル名とレコード番号をもつ。

SEEK(ファイル名、レコード番号)

ファイル名で指定するファイルは、Structureで定義されたものである。レコード番号は、整数型であり0から数える。

SEEK動作は、ファイル名で指定されたファイルのファイル・ポインタをレコード番号で指定したレコード位置に変更する。したがって、SEEKにつづくPUT、GETで要求したレコードに対しアクセスできる。

```
SEEK(FIB, 5);
GET(FIB);
```

2.3.2 Case文

Case文の選択子が、いずれのラベルである定数にも該当しない時、エラーを発生させずに次のステートメントを実行する。

2.3.3 分割コンパイル

MCC PASCALでは、メイン・プログラムから呼ぶ手続きの一部をメイン・プログラムのファイルと異なったファイルとしてコンパイルすることができる。

分割コンパイルを行うために、メイン・プログラム側のファイルと、呼ばれる手続側のファイルで最小限の情報を書く必要がある。この情報は、リンカ情報としてコンパイラから出力され、リンカによって単一セグメントの実行可能なPレコードを完成させる。

メイン・プログラム側では、外部手続き名とそれらを含むファイル名を'USE'で宣言する。この宣言は、プログラム・ヘッダーの直接に書かなくてはいけない。

外部手続き側では、メイン・プログラムで呼ばれる手続き名と、そのファイル名を'UNIT'及び'HEADER'で宣言の後から'BODY'と'END'で囲まれたブロックの中に各手続きを記述する。

<メイン・プログラム側>

```
PROGRAM プログラム名;
  USE ファイル名;
        手続き名;手続き名;
          :
        手続き名;
  USE ファイル名;
        手続き名;
          :
        手続き名
  END;
          ≀
        ブロック
          ≀
END.
```

<外部手続き側>

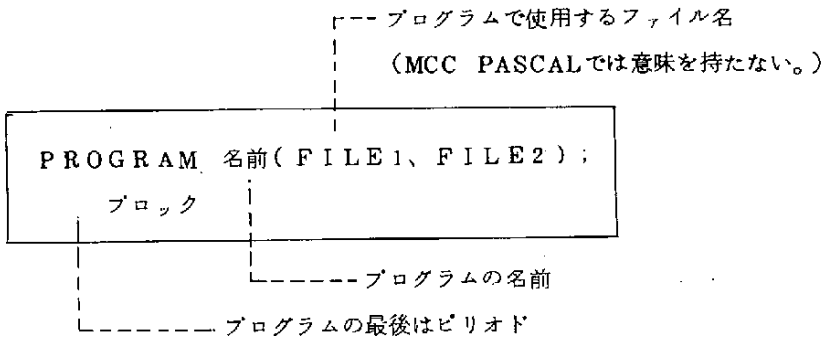
```
PROGRAM プログラム名;
  UNIT ファイル名;
  HEADER
        手続き名;
          :
        手続き名;
  BODY
          ≀
        各手続き
          ≀
END.
```

別　添

MCC　PASCAL文法（標準pascalに準拠）

1．プログラム

プログラム（仕事全体を記述する）



名　前（定数・型・変数・手続き・関数などを表わす）

英字〕　英字または数字２　英字または数字３……

（コンパイラは８文字まで判読）

ブロック（プログラムや手続きなどの本体を記述する）

〔名札の宣言部〕
〔定数の定義部〕
〔型の定義部〕
〔変数の定義部〕
〔手続き・関数の宣言部〕
ＢＥＧＩＮ　文１；
　　文２；
ＥＮＤ；

名札の宣言部（名札をブロックのなかで用いることを宣言する）

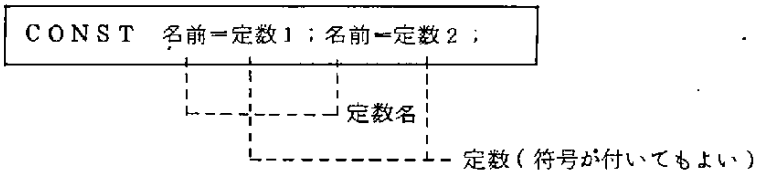
ＬＡＢＥＬ　名札１，名札２，……；

ブロックのなかで用いる名札は必ず宣言する

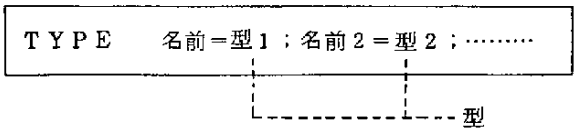
名札（ＧＯＴＯ文で行先を表わす）

数字１，数字２，……；

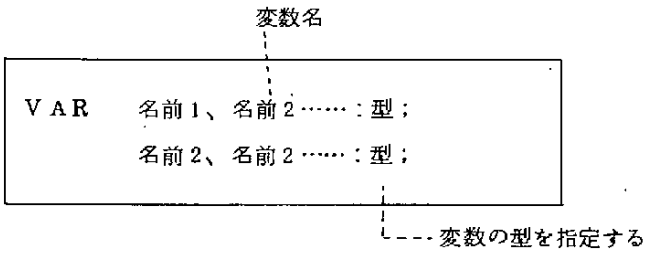
定数の定義部（名前ｉが定数ｊを表わすものであると定義する）



型の定義部（名前ｊが型ｉを表わすことを定義する）

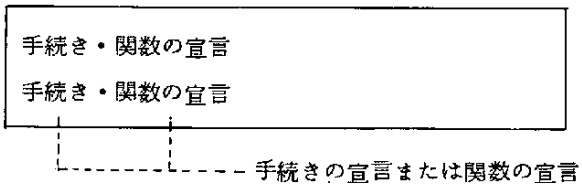


変数の宣言部（ブロックで用いる変数名を宣言する）

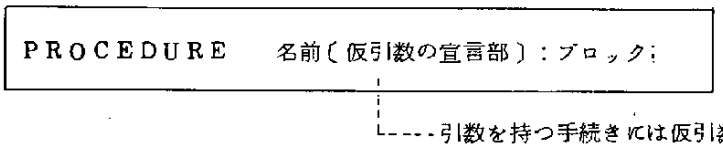


手続き・関数の宣言部

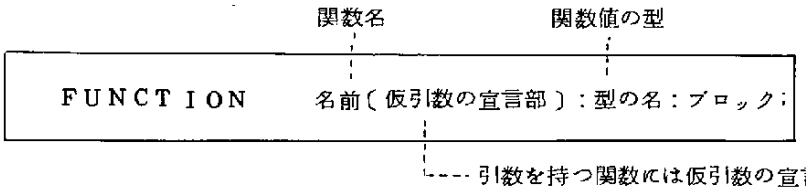
（ブロックで用いる手続きと、関数を宣言する）



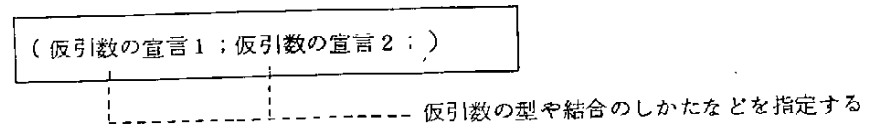
手続きの宣言



関数の宣言



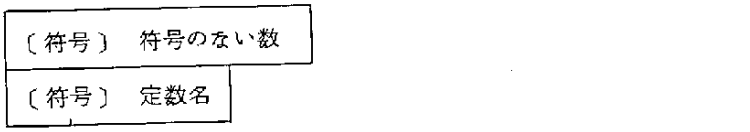
仮引数の宣言部



名前1、名前2……：型の名	値の置換えをうける仮引数
VAR 名前1、名前2……：型の名	変数の置換えをうける仮引数
FUNCTION 名前1、名前2……：型の名	関数の仮引数
PROCEDURE 名前1、名前2，……	手続きの仮引数

2．定　数　型

定数（数または文字例）



文字例　定数

符号のない数

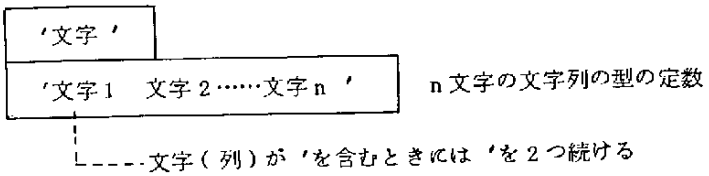
数字１、数字２……数字ｎ　　整数型の定数

数字１、数字２……数字ｎ　　実数型の定数

数字１……数字ｍ・数字ｍ＋１……数字ｎ

数字１……数字ｍ

文字列定数



符号

＋	正
－	負

指数部

E〔符号〕数字1……数字p

単純な型

型の名 ---------- 標準の型の名、あるいは型の定義で定義された名前

（名前1、名前2，……） 名前iを値として持つ列挙型

定数1…定数2 定数iの型の一部を値とする範囲型

単純な型

配列の添字の型

〔PACKED〕ARRAY〔単純な型1、単純な型2，…〕OF型； 配列型

詰めた型は記憶場所を節約する

配列要素の型

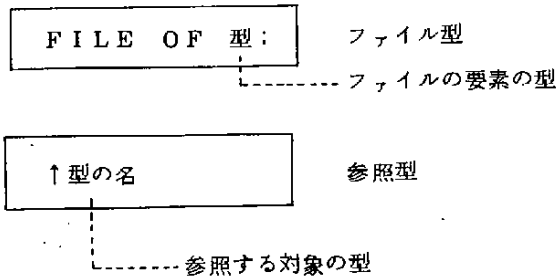
とくにPACKED ARRAY〔1…n〕OF CHAR（n≧2）は文字列の型

RECORD欄の並び END； レコード型

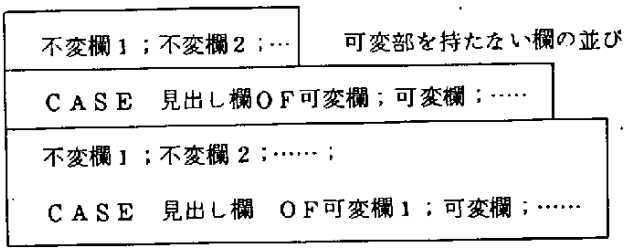
欄を識別する名前と欄の型を並べる

SET OF単純な型； 集合型

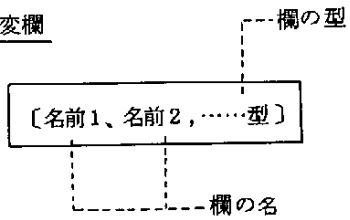
集合の要素の型



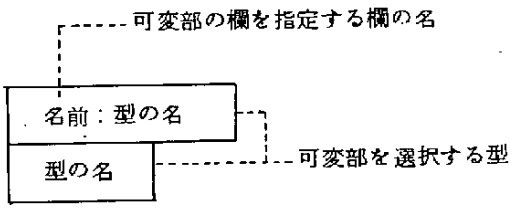
欄の並び



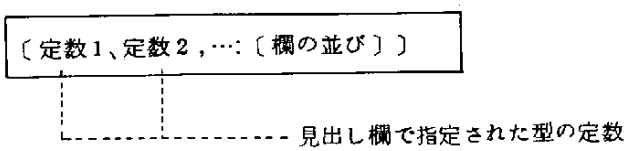
不変欄



見出し欄



可変欄



3．文

文

〔名札：〕単純な文

〔名札：〕構造をもつ文

名札：のついた文へはＧＯＴＯ文によって制御が移ることがある

単純な文

変数：＝式

関数名：＝式

代入文

関数の値を定めるために式の値を代入する

手続き名

手続き名（実引数１、実引数２…）

引数を持つ手続きには実引数を与える

ＧＯＴＯ　名札；　ＧＯＴＯ文

名札のついた文へ制御を移す

＜空＞　空文（文のあるべきところに何もない）

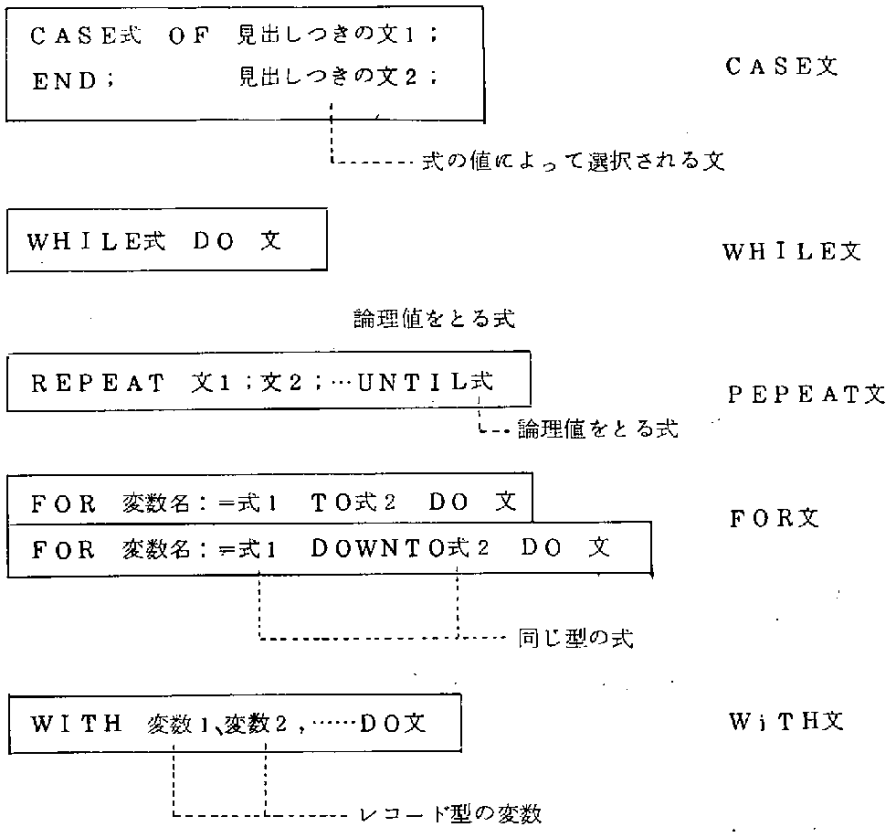
構造をもつ文

ＢＥＧＩＮ　文１；文２；…ＥＮＤ；　複合文

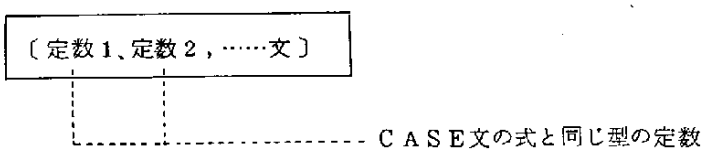
ＩＦ式　ＴＨＥＮ文１　ＥＬＳＥ文２；　ＩＦ文

ＩＦ式　ＴＨＥＮ文；　ＩＦ文

論理値をとる式



見出しつきの文



実引数

式	値の置換えを行なう引数
変　数	変数の置換えを行なう引数
手続き名	手続きを移す引数
関数名	関数を移す引数

4・変数・式

変数

変数名　　全変数

変数〔式1、式2,…〕

- 配列の添字の型の式
- レコード型の変数

変数・欄の名　　欄指定

- レコード型の変数

変数↑　　ファイルの窓

- ファイル型の変数

変数↑

- 参照型の変数

式

単純な式

単純な式　関係演算子　単純な式　　式の型は論理型になる

単純な式

整数型、実数型の項には符号がついていてもよい

〔符号〕項

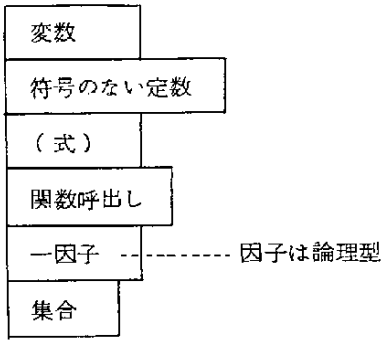
単純な式　加法演算子　項

- 〔符号〕項1　加法演算子1　項2　加法演算子2　項3…

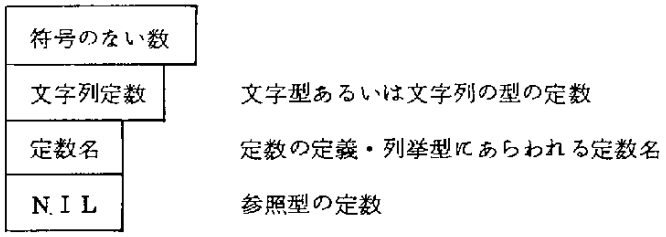
項

因子１　乗法演算子　因子２　乗法演算子２　因子２……

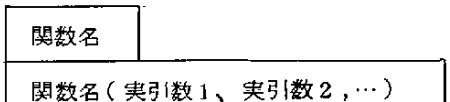
因子



符号のない定数

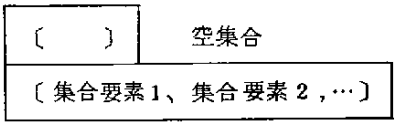


関数呼出し



引数を持つ関数には実引数を与える

集合



集合要素

式　　式の値で表わされる一つの要素

式１……式２　　式１の値以上で、式２の値以下のすべての要素

関係演算子

	整数型（の範囲型）	実数型	文字型（の範囲型）	論理型	列挙型（の範囲型）	文字列の型	集合型	参照型
＝	○	○	○	○	○	○	○	○
≠	○	○	○	○	○	○	○	○
＜	○	○	○	○	○	○	×	×
≦	○	○	○	○	○	○	○	×
＞	○	○	○	○	○	○	×	×
≧	○	○	○	○	○	○	○	×

上の表の○印以外の型には関係演算子は使えない

IN	「集合の要素の型の式」ｉＮ「集合型の式」として用いる

加法演算子

＋	整数型（の範囲型）・実数型の和 集合型の和集合
－	整数型（の範囲型）・実数型の差 集合型の差集合
V	（整数型と実数型の混合計算も許される） 論理型の論理和

乗法演算子

＊	整数型（の範囲型）、実数型の積 集合型の構集合
／	整数型（の範囲型）・実数型の商（結果は実数型） （整数型、実数型の混合計算も許される
DIV	整数型（の範囲型）の商（結果は整数型）
MOD	整数型（の範囲型）の剰余
∧	論理型の論理積

標準要素

標準・型名

INTEGER	整数型
REAL	実数型
CHAR	文字型
BOOLEAN	論理型
TEXT	テキストファイル型

標準・定数名

FALSE	論理型の値　「偽」
TRUE	〃　「真」
MAXINT	整数の最大数（＝32,767）

標準・変数名

INPUT	入力ファイル
OUTPUT	出力ファイル

標準手続き

GET	入力
PUT	出力
RESET	入力準備
REWRITE	出力準備
READ	テキストファイルからデータを読み込む
READLN	〃　の1行の終りまで読み込む
WRITE	〃　にデータを書き出す

WRITELN	テキストファイルの１行の出力を行なう
PAGE	〃　　で改頁を行なう
NEW	動的な変数を生成し、参照型の変数に値を与える
RELEASE	〃　を消去する

標準関数

ABS	絶対値関数
SQR	平方関数
SIN	正弦　〃
COS	余弦　〃
EXP	指数　〃
LN	自然対数
LOG	常用対数
SQRT	平方根関数
ARCTAN	逆正接　〃
ODD	奇のとき「true」の値をとる
EOF	ファイルが終了のときに「true」の値をとる
EOLN	テキストファイルが行の終りのときに「true」の値をとる
TRUNC	実数型データを切り捨てて整数値に変換する関数
ROUND	〃　を四捨五入して　〃
ORD	順序数を与える関数
CHR	整数値に対応する文字型の値を与える関数
SUCC	列挙型データで一つ後の値をとる関数
PRED	〃　前　〃

特殊要素

特殊手続き

SEEK	ランダム・ファイル　アクセス

特殊関係

BLKREAD	ディスク・ファイル　N〔BLOCK〕の読み込み、結果正常＝N(：INTEGER；)
BLKWRITE	〃　の書き出し　〃
IORESULT	I／O機器アクセスが正常に行なわれたか、結果正常＝0　(：INTEGER)
MEMORY	現時点でのメモリー使用可能な量〔WORD〕　(：INTEGER；)

example

```
FUNCTION MEMORY：INTEGER；
```

```
BEOIN
   ?
END:
```

昭和56年 3 月 発行

発行所 財団法人 日本情報処理開発協会
東京都港区芝公園3－5－8
機 械 振 興 会 館 内
TEL（434）8211（代表）

印刷所 株式会社 昌 文 社
住所 東京都港区芝5－26－30
TEL （452）4931