46 S — 003

インタラクティブ学習システムの開発

— ディスプレイシステムの研究開発 —

昭和４７年３月

財団法人　日本情報処理開発センター

この事業は，日本自転車振興会の機械工業振興資金による「昭和４６年度情報処理に関する調査・研究補助事業」のうち「ディスプレイシステムの研究開発」の一部として実施したものであります。

序

当財団は，情報処理に関する調査および研究開発の一環として情報処理システムにおけるディスプレイ・システムの利用に関する各種ソフトウェアの研究開発を進めておりますが，この報告書は当財団で開発したCRTキャラクタ・ディスプレイ装置を用いた「インタラクティブ学習システム」の研究成果を述べたものであります。

コンピュータを利用した学習は，ここ数年来かなりクローズ・アップされては来ましたが，まだ多くの研究や体験を必要とする段階であると思われます。

この研究開発はそれらの事情を考慮して，もっとも身近なプログラミング言語の学習を目的としたシステムをとりあげ，問題点の解明の一助となることを念願致しております。

昭和４７年３月

財団法人　日本情報処理開発センター

会長　難　波　捷　吾

まえがき

Computer Assisted Instruction（ＣＡＩ）が世間の注目をあびる様になってからすでに久しいが，現在まだその実用効果の確証が得られていないというのが実情であろう。特に価格に見合う成果という点ではかなり悲観的見方も多い。

この様な実情の中で，比較的その効果を評価されつゝあるのがコンピュータのプログラミングの学習であるという。これはその目的と道具が一致したという強味と同時に非常に問題が大きいといわれる学習プログラムそのものに専門的知識の介入が可能であるからであろう。即ち学習の内容を１００％理解しているものがシステムを作る強味である。

ＣＬＡＳＳ（Conversational Language ASisted System）は，これに更に次のような２つの要素をつけ加えてシステムを設計した。

第一は，ＴＳＳの会話型言語ＣＰＬ（Conversational PL／Ｉ）の学習をおこなうシステムであるため，プログラミング言語に不可欠な練習問題のプログラムの多くは実際にＣＰＬのプロセッサーで処理して，エラーの指摘や結果の表示をおこなわせることができる。

第二は方式の比較という目的から２つのシステムを作成したことである。一つは会話型言語のガイダンス機能の拡張というニュアンスを持ったシステムで，ＣＲＴキャラクタ・ディスプレイ装置とＴＳＳ端末を学習端末とし，ＣＰＬによるプログラムの練習問題を中心に学習者のプログラムの正否および誤りの指摘，ＣＰＬシステムそのものへのガイダンスと同時に，ディバッグ方法のガイダンスの要素も含めたユニークなものであるが，もう一つはスライドプロジェクターを学習端末としたオーソドックスなものであるが，こちらも各セクションのまとめとして，併用されているＴＳＳ端末からＣＰＬの実習を実際におこなうことができる。この２つのシステムは対象とするコンピュータの名称から夫々ＣＬＡＳＳ－Ｆ システムおよびＣＬＡＳＳ－Ｎ システムと呼ぶ。

これらは，今後ＴＳＳ端末によるＣＡＩの可能性をさぐる一つの実験的試みとし

ておこなったもので，ガイダンスシステムの発展した形としての学習システムのあり方について検討したいと願っている。

この報告書では，第1部にCLASS－Nシステムを，第2部にCLASS－Fシステムを紹介する。

第 1 部　CLASS－N

目　　次

第１章　ＣＬＡＳＳ－Ｎシステムの概要

現在開発されているＣＡＩ（Computer Assisted Instruction）システムはその学習方式が個別指導型（Tutorial Instruction）かドリル演習型（Drill and Practice）か又は対話型（Inquiry）かのどれかである。このＣＬＡＳＳシステムの目的である会話型言語を教育する場合にはそのすべての方式を含むことが望ましい。一般にプログラミング言語の教育ではその言語の文法的知識を覚えることと同時にその言語を実際に使いこなすことが必要である。特に会話型言語は実際には計算機センターと離れたところから使われるのが普通であるから何か疑問が生じた場合は，その解決の糸口を見つけてくれるガイダンスの機能が必要である。更に，今後のＣＡＩシステムにとって特に重要な機能として診断機能がある。一般にプログラミング言語の教育では，学習の誤りを的確に指摘するのはきわめてむづかしいが，それがまたこの学習システムのポイントでもあろう。
このＣＬＡＳＳ－Ｎシステムは以上のことを考慮に入れて設計されたシステムである。

ＣＬＡＳＳシステムの特長とするところは第１に会話型言語に対する初心者はまずこのＣＬＡＳＳ－Ｎシステムにより言語の文法的知識やその使い方についての基礎的内容を学習し，このシステムの学習を終えた人や経験者は第２部で述べるＣＬＡＳＳ－Ｆシステムによりプログラム・テクニックを学習できるようになっている。第２に全体は，個別指導型で学習するようになっているが，練習問題は実習形式になっておりＴＳＳ端末よりプログラムを作成してすぐに実行し，その結果を解答として入力するようになっている。なおプログラミングの過程又は入力された結果が正しくないときにはその過程を分析し適切なデバッグのヒントを与えることができる。

第２章　システムの構成

２.１　ＮＥＡＣ２２００－５００の機器構成

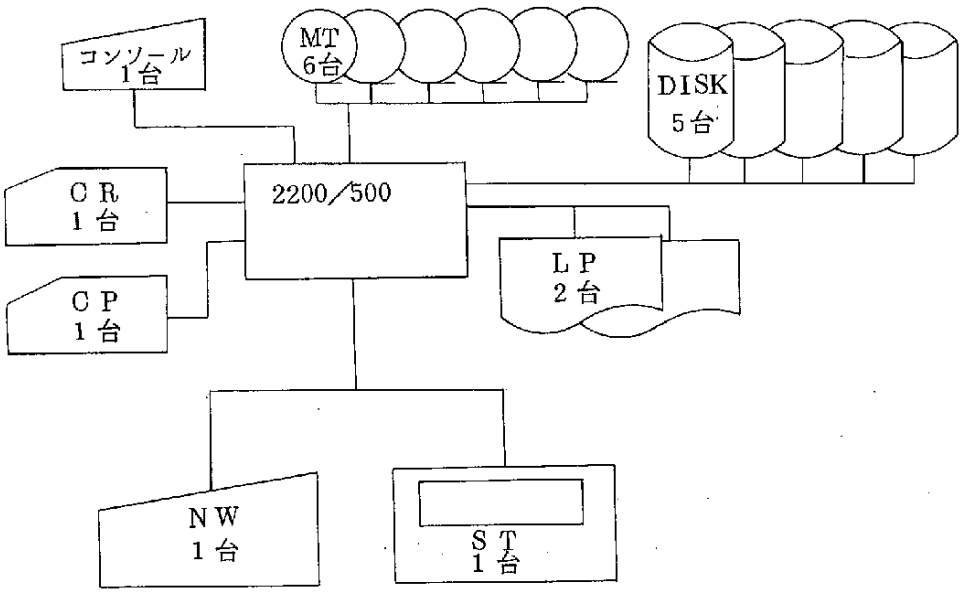


図２－１

(1)　中央処理装置

記憶容量	５２４Ｋ字
サイクルタイム	1.5 μs

(2)　磁気ディスク装置

記憶容量	9.6Ｍ字／台
平均アクセスタイム	８５ｍｓ

(3)　周辺入出力装置

カードリーダ	８００枚／分
カードパンチ	１００枚／分
ラインプリンタ	４２０行／分
磁気テープ	６４／４４.４ｋｃ／秒

(4)　学習端末装置

ＳＴ	スライド80コマ，ランプ10種，キーセット（応答キー10個，データキー100個）
タイプライタ	５６０字／分

2.2 学習端末の構成

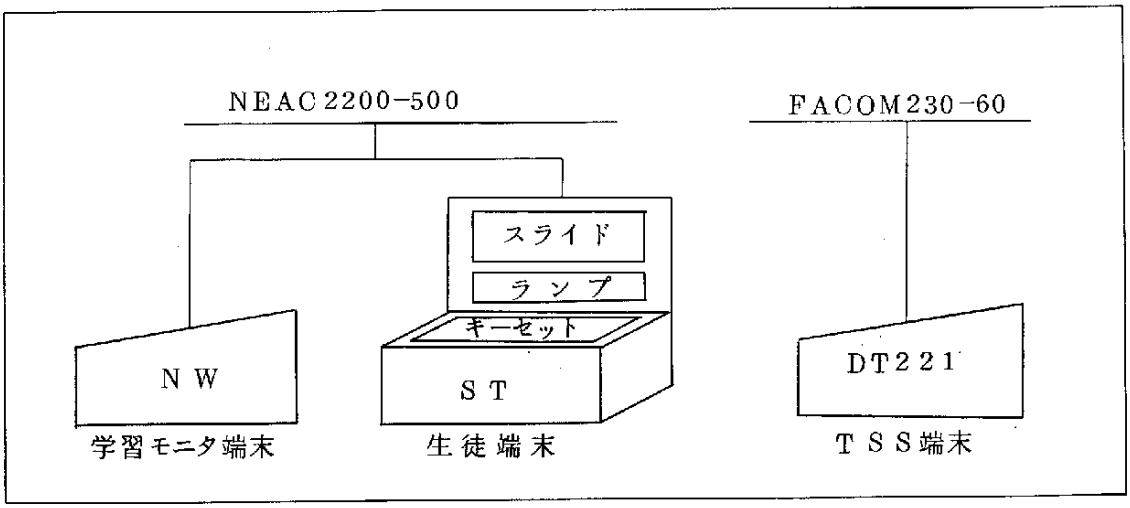
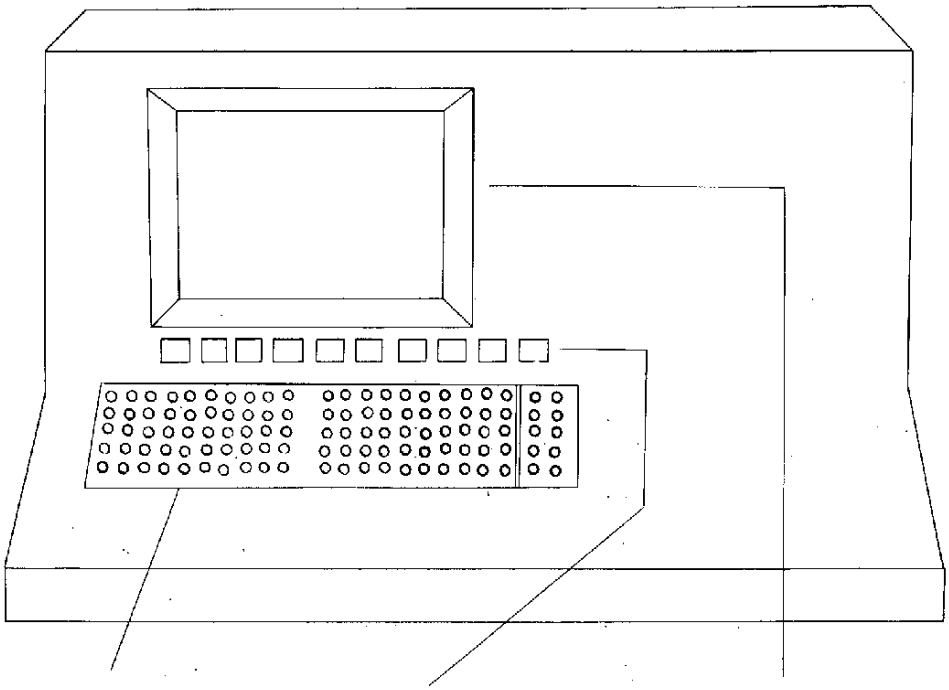


図２－２　学習端末全体構成図

学習端末は全体として３つの装置から構成されている。１つは生徒端末（ＳＴ）であるが，この生徒端末はまた次のような３つの機能をもった装置より構成されている。即ち説明や問題の提示装置としてランダム・アクセス・スライド，反応入力装置としてキーセット，反応の補助確認装置として１０個のランプがある（図２－３参照）。ＳＴの他学習者の学習モニタ装置として，NEAC WRITERと練習問題の実習用としてＦＡＣＯＭ２３０－６０のＴＳＳ端末を使用している。



・キーボード部
キーマット方式

このキーボード部は押しボタン式でキーのそれぞれには意味をもたせず鍵盤部全体にかぶせるキーマットにより，それぞれのキーに意味をもたせたもので，キーマットを取り替えることで任意の形式のデータをセットすることができる。

・指示ランプ部
パネル式

この指示ランプ部はコンピュータの状態を知らせ学習者が何を行なえばよいか，またどのような状態にあるかを表示する。

・スライド提示部
ランダム・アクセス方式

このスライド提示部はスライドプロジェクタとスクリーンより構成されている。スライドプロジェクタは８０枚のスライドを収納できるマガジンよりランダムにスライドを取り出し，スクリーンに映写し，このマガジン室に学習プログラムをファイルしておくことで学習時のプログラムを即時変更できる。

図２－３　生徒端末構成図

生徒端末のランプは図２－４のようになっており左から４個のランプは応答キーによりハード的に点灯し，ハード的に消灯される。右から６個のランプは学習者の応答内容により制御プログラムで点灯されたり消灯されたりする。

生徒端末のキーセットは図２－５のような学習用キーマットと図２－６のような応答キーマットをかぶせて使用される。

電源	学習中 学習終了	メモ (デバッグ)	どうぞ	やり直し	ヒント なし	正答 なし	正答	誤答	時間切れ

図２－４　生徒端末のランプの意味

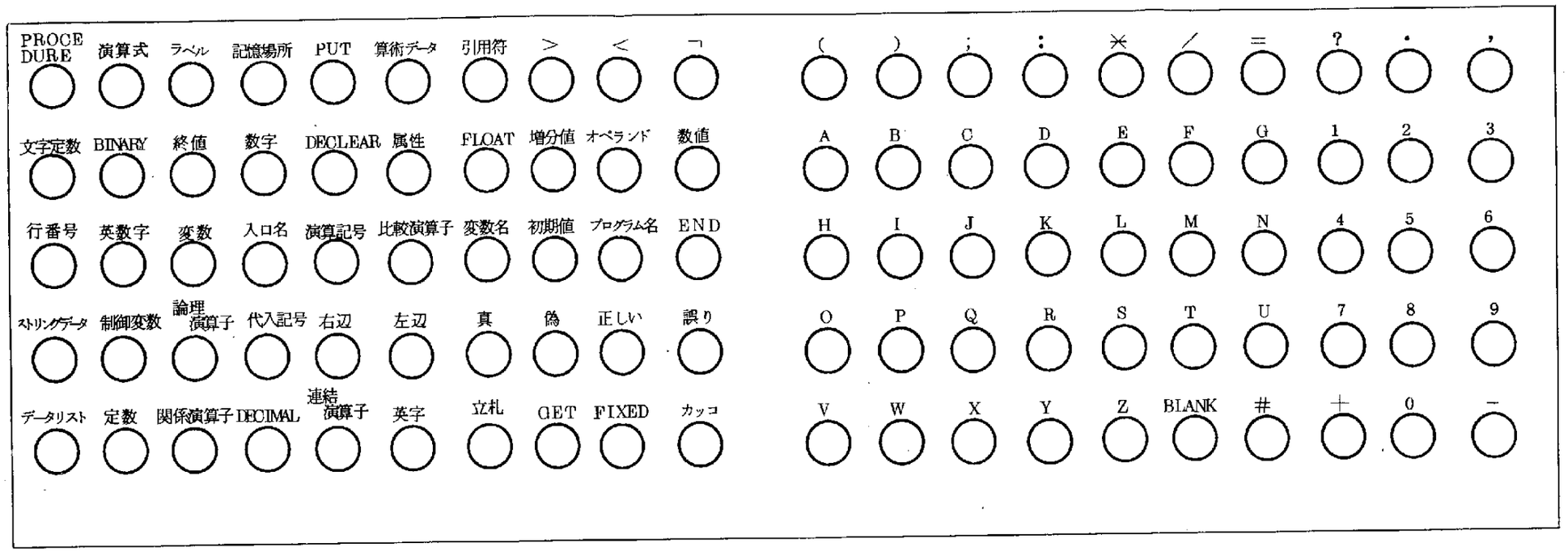


図２－５　　ＣＰＬ学習用キーマット

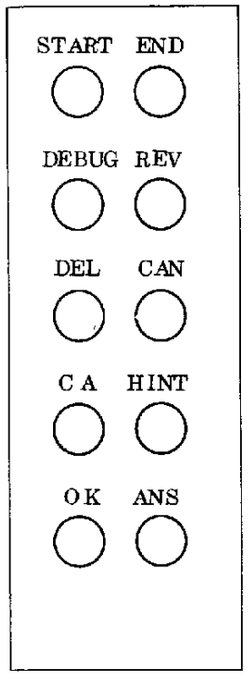
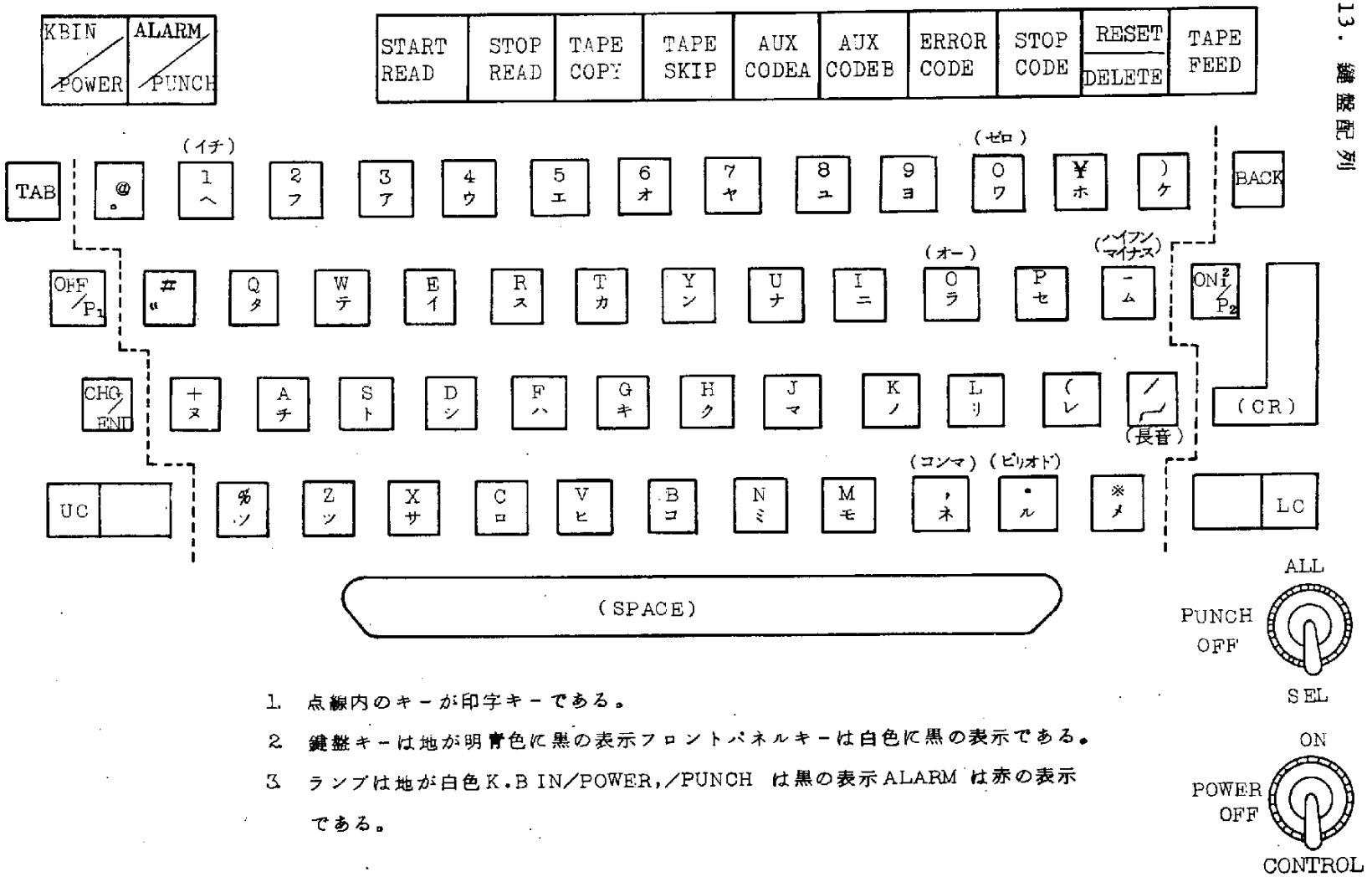
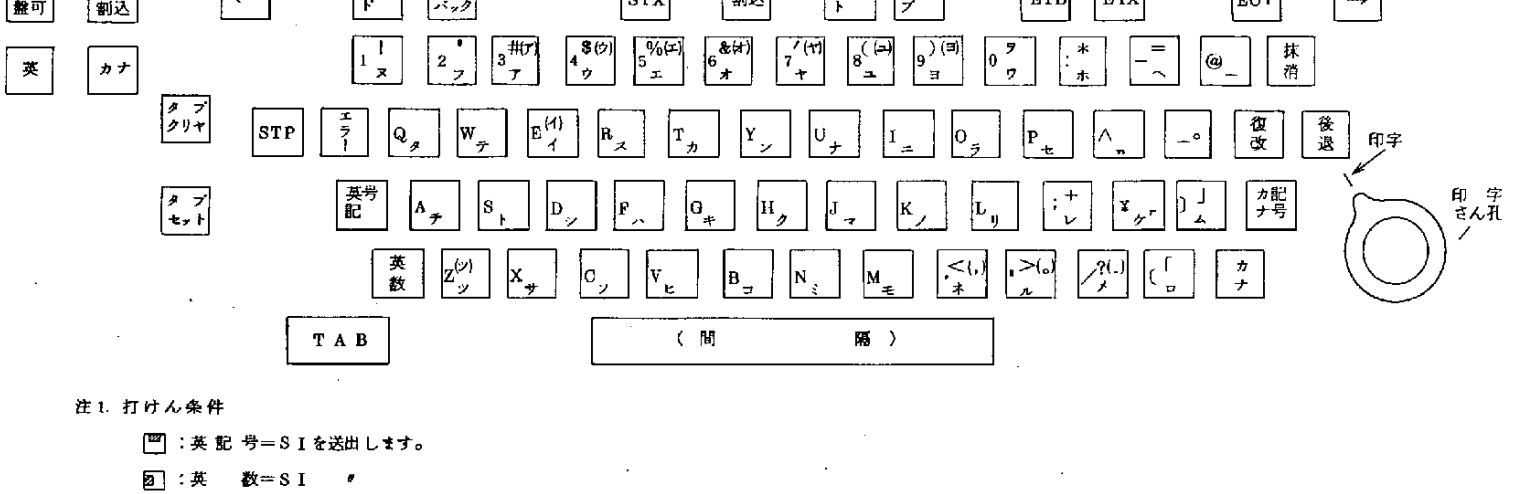


図２－６　応答用キーマット



1. 点線内のキーが印字キーである。
2. 鍵盤キーは地が明青色に黒の表示フロントパネルキーは白色に黒の表示である。
3. ランプは地が白色K.B IN/POWER,/PUNCH は黒の表示ALARM は赤の表示である。



注 1. 打けん条件

- ：英 記 号＝ＳＩを送出します。
- ：英　数＝ＳＩ　〃
- ：カ　ナ＝ＳＯ　〃
- ：カナ記号＝ＳＯ　〃

2. エラーおよび抹消は ☒ と印字されます。
3. ← および → キーは符号を送出せず、プリンタの印字ヘッドをそれぞれ、左および右に移動するボタンです。
4. 左側の「けん盤可」「相手割込」「英」「カナ」はランプです。
5. ( )内はキー・トップに刻印されていません。

200Ｂ型データ宅内装置のけん盤配列図

学習モニタ装置としてのNEAC WRITERはＳＴのキーセットの各キーの意味に応じて必要な文字に変換され印字される。

ＴＳＳ端末装置はFACOM2３０－６０に接続されている端末装置の１つでありＣLASS－Ｎシステムの実習用端末として使われる。

どちらの端末とも英文字，カナ文字，数字，記号の文字が使用できる。なおこの端末の鍵盤配列は図２－６と図２－７のようになっている。

2.3 ソフトウエアの構成

ＣＬＡＳＳ－ＮのソフトウェアはＮＥＡＣ２２００／５００ ＯＳ－ＭＯＤⅣの下で動作可能であり，それらは次のような部分から構成されている。

1. ＣＬＡＳＳ－Ｎモニター
2. 制御プログラム
3. 学習プログラム
4. ＣＬＡＳＳ－Ｎガイド
5. 学習資料

(1) ＣＬＡＳＳ－Ｎモニター

ＣＬＡＳＳ－Ｎモニターは学習時における全体の制御を行うもので次のような機能をもっている。

- システム全体のイニシャライズ
- 学習端末及びファイルのＯＰＥＮとＣＬＯＳＥ
- 学習者との会話
- 学習プログラムのロード

(2) 制御プログラム

学習プログラムの提示ステートメントを制御するもので次のようなルーチンから構成されている。

- ＳＬＩＤＥ

学習プログラムで与えられたステップ番号，スライド番号，ステップの種類等の各情報をセットし，前のステップで作られた学習記録を出力する。

・ＲＥＱ

学習プログラムで与えられたヒント，正答等の要求許要の各情報をセットする。

・ＡＮＳ

学習プログラムで与えられた予想解答の情報から予想解答テーブルを作成する。

・ＬＡＭＰ

学習プログラムで与えられたランプの種類により実際にランプを提示する。

・ＴＹＰＥ

学習プログラムで与えられた文字情報を実際にNEAC WRITERに印字する。

・ＴＩＭＥ

学習プログラムで与えられた応答制限時間の情報をセットし，ＳＬＩＤＥで与えられたスライド番号により実際にスライドを提示する。

・その他

これらに付随するユーティリティルーチンからなっている。

(3) 学習プログラム

学習プログラムはいくつかの教科からなり，それぞれはいくつかのセクションから作られている。各セクションはいくつかのステップより構成されている。ステップはいわば連続した知識の単位で，その１つ１つが学習者に提示され学習が行なわれる。

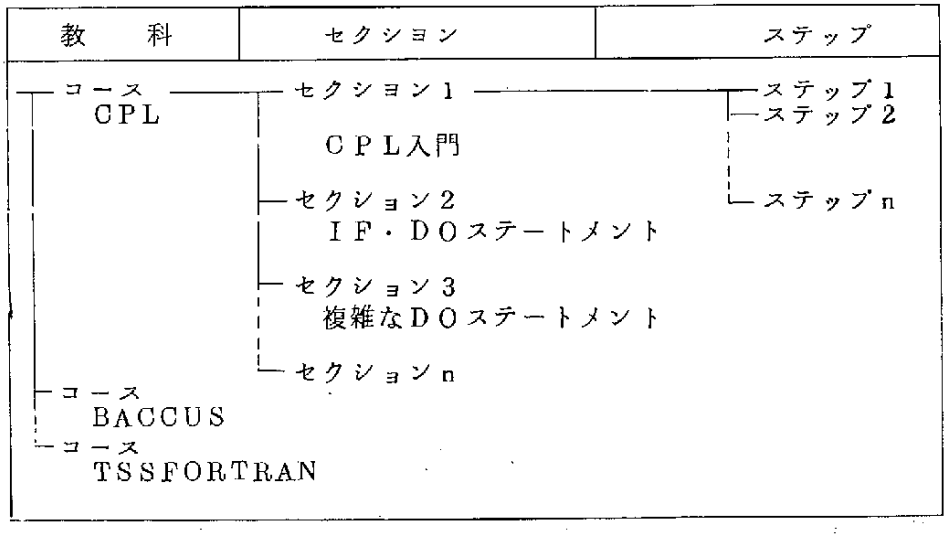


図２－８

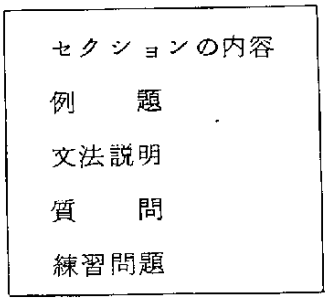


図２－９

(4) ＣＬＡＳＳ－Ｎガイド

ＣＬＡＳＳ－Ｎガイドは学習者へのサービスのプログラムであり，次のようなものから構成されている。

・学習端末の使い方（電源のＯＮ，ＯＦＦ，スライドのセット，キーマットのセット，キーセットの意味）

・ＴＳＳ端末の使い方（電源のＯＮ，ＯＦＦ，システムとの会話，ＤＴ端末のキーの意味）

・ＣＰＬの操作（エラーメッセージの説明，ステートメントの修正の仕方）

・カタログ（教科のセクションの内容と番号リスト）

このＣＬＡＳＳ－Ｎガイドは学習端末のところに置れているので学習者は必要に応じていつでも参照することができる。

(5) 学習資料

各教科の各セクションで必要な資料，即ち練習問題のときのフローチャート，コーディング用紙，組込み関数のリスト等や，どうしてもその問題がわからない等のときの処置は必要に応じて学習資料として用意されている。これは学習者がこのＣＬＡＳＳ－Ｎシステムでの学習を申込んだときに渡される。

２４　ファイルの構成

ファイルは大別してプログラム・ファイルとデータ・ファイルからなりそれぞれ別々のディスクパックに作られる（図２－１１参照）。

(1) プログラム・ファイル

プログラム・ファイルは次のようなファイルから構成されている。

・ＢＰＬプロセッサー

これは学習プログラムをコンパイルして相対形式のプログラムを作るプロセッサーである。

・ＢＰＬライブラリー

これはＢＰＬプロセッサーによりコンパイルされ作られたオブジェクトプログラム（学習プログラム）で使用されるランタイムルーチンのライブラリーである。

・ＫＥＹＭＡＴファイル

キーセットの各キーに意味をもたせるためにはそれに対応する変換テーブルが必要である。その変換テーブルは各教科毎にＫＥＹＭＡＴファイル作成ルーチンにより作られ実行時にＣＬＡＳＳ－Ｎモニターから呼ばれ使用される。

・教材ファイル

教材ファイルは１教科毎に作られ，制御プログラムと学習プログラム・セクションの集合からなっている。１教科は普通複数セクションからなっているので，その中の必要なセクションがＣＬＡＳＳ－Ｎモニターからロードされ学習に使用される。

(2)　データ・ファイル

データ・ファイルは学習記録ファイル専用に使われる。学習記録は学習者毎に別なファイルが作られ，必要な時点で学習記録出力ルーチンにより記録内容を出力できる。

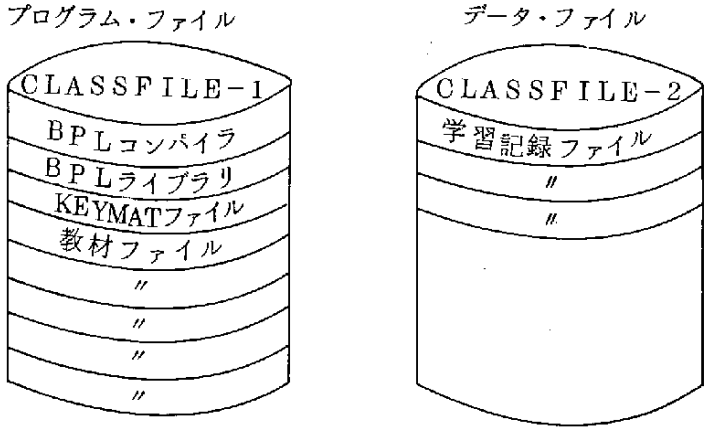


図２－１１　ファイル構成

2.5 CLASS—Nのメモリーマップ

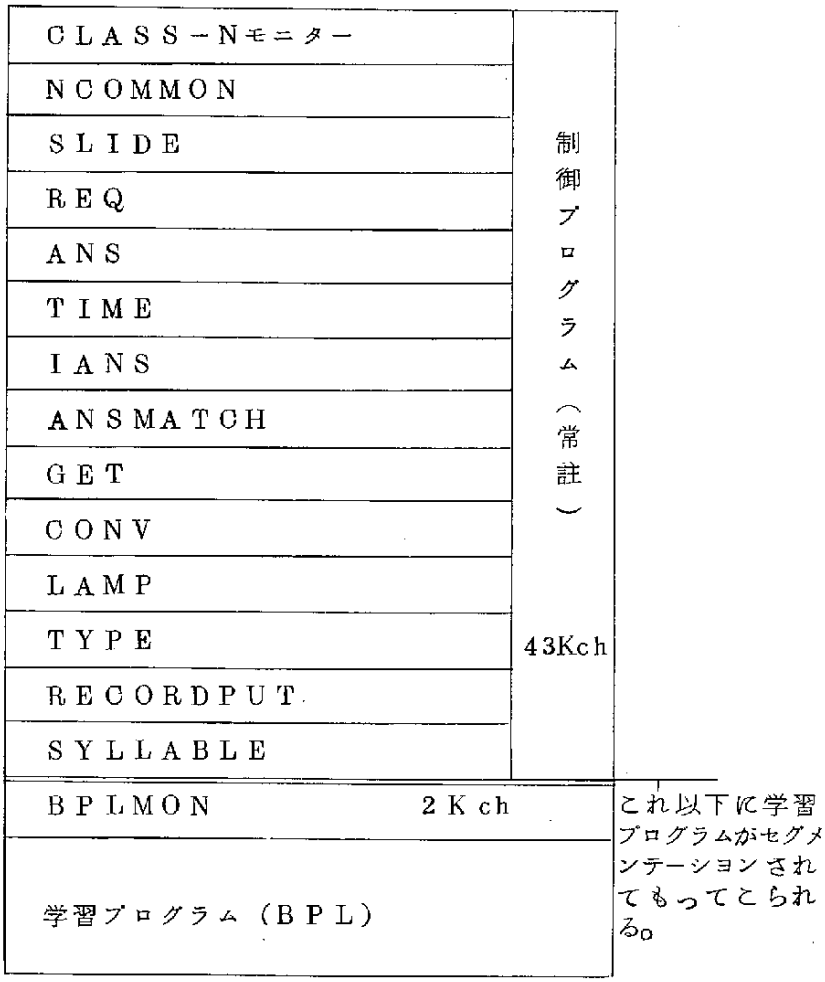


図2－12

第3章　学習プログラム記述言語

3.1　記述言語に対する考え方

学習プログラム記述言語としては標準ＣＡＩ言語が多く使われているようであるが，ファイル処理，演算処理，学習端末の操作（特にＣＬＡＳＳ－Ｎではランプ等の提示装置もある）等の機能的な面で不充分な部分も多い。そこで今回は標準ＣＡＩ言語的な処理系を作ることを止めて，ＣＡＩ機能の部分をアセンブリ言語で作りそれをＰＬ／ＩのサブセットであるＢＰＬ（Basic Programming Language）のＣＡＬＬタイプのサブプロシーデュアとして呼ぶことにより処理することとした。

3.2　CLASS—N 言語の機能

3.1 に述べた理由から学習プログラム記述言語はＢＰＬの特定ステートメントの集合であり，これをＣＬＡＳＳ－Ｎ言語と定義する。以下にそれらのステートメントの機能を示す。

ラベル指定	ステートメントの記述	機　　能
教科名： 〔ラベル：〕	ＰＲＯＣＥＤＵＲＥ； ＲＥＴＵＲＮ； ＥＮＤ；	教科名の定義 サブルーチンの終り 学習プログラムの終り
〔ラベル：〕	ＣＡＬＬ　ＳＬＩＤＥ （ｓ，ｊ，'ｋ'）； ＣＡＬＬ　ＲＥＱ （｛'ｵﾌﾟｼｮﾝ'〔，'ｵﾌﾟｼｮﾝ'〕…｝）； ＣＡＬＬ　ＡＮＳ （'指定：予想解答'〔，'指定： 予想解答'〕…）；	スライド第ｊコマ提示 ｓ：ステップ番号 ｊ：スライド番号 ｋ：ステップの種類 ヒント，正答要求等を許す オプション：ＨＩＮＴ，ＣＡＮＳ， ＲＥＶ，ＯＫ 予想解答の記述 指定：(1)解答指定　Ｃ…正答 Ｗ…誤答 …指定なし

		(2)型指定　A…文字 F…固定小数点数 E…浮動小数点数 (3)空白指定（型がAのとき） 1…そのまゝのパターン 2…空白を無視 3…1個以上の空白は無視
	CALL TIME (m) ;	応答制限時間m秒
	CALL LAMP (n) ;	ランプ第n番を提示
	CALL TYPE ('文字') ;	タイプライタに'で囲まれた文章を提示
〔ラベル：〕	SW='11111111111'B ;	すべてのスイッチをオンにする
	SW='0'B ;	すべてのスイッチをオフにする
	SUBSTR (SW, n, 1) = '1'B ;	スイッチnをオンにする
	SUBSTR (SW, n, 1) = '0'B ;	スイッチnをオフにする
	A=±B±C ;	AはACCn, CCOUNT, WCOUNT, TCOUNT BとCはACCn, CCOUNT, WCOUNT, TCOUNT, 又は整数
〔ラベル：〕	GO TO ラベル ;	ラベルで指定されたステートメントへ分岐
〔ラベル：〕	CALL　入口名 ;	入口名で指定されたサブルーチンへ分岐
〔ラベル：〕	IF～THENユニット1 〔ELSE ユニット2〕	～で表わされる条件が成立するとユニット1が実行され，それからIFの次のステートメントが実行される。 条件が成立しないときはユニット2が実行されそれからIFの次のステートメントが実行される。
	IF TIMEUP THEN	指定された応答制限時間が切れている時

	IF SUBSTR (ANS, n, 1) THEN	学習者解答が予想解答のn番目と一致したとき
	IF SUBSTR (ANS, 1, 1) THEN	学習者解答がどの予想解答とも一致しなかったとき
	IF HINT THEN	ヒント要求があったとき
	IF CANS THEN	正答要求があったとき
	IF REV THEN	前のステップに戻りたい要求があったとき
	IF OK THEN	ヒントの了解要求があったとき
	IF CORRECT THEN	学習者解答が予想解答のすべてと一致したとき
	IF WRONG THEN	学習者解答が予想解答のどれかと一致しないとき
	IF SUBSTR (SW, n, 1) THEN	スイッチnがオンのとき

各ブロックは上からプログラム構成ステートメント，提示ステートメント，代入ステートメント，GOTOステートメント，CALLステートメント，IFステートメントを表わす。しかしこれ以外でもBPLで許されるステートメントなら使用可能であり，よりきめ細かい学習プログラムの記述ができる。

(1) 学習プログラムで記述できるシステム変数

1. CORRECT　学習者の解答が予想解答のすべてと一致したときONになる。
2. WRONG　学習者の解答が予想解答のどれかと一致しなかったときONになる。
3. ANS0　学習者の解答が予想解答のどれとも一致しなかったときONになる。
4. ANSn　学習者の解答が予想解答のn番目と一致したときONになる。
5. HINT　ヒントキーが押されたときONになる。
6. CANS　正答キーが押されたときONになる。

7. REV　　戻りキーが押されたときＯＮになる。

8. ＯＫ　　了解キーが押されたときＯＮになる。

9. TMUP　　応答制限時間が切れたときＯＮになる。

10. CCOUNT　　正答回数のカウンター。

11. WCOUNT　　誤答回数のカウンター。

12. TCOUNT　　タイムアップ回数のカウンター

13. SWn　　$0 \leqq n \leqq 7$ で８個のスイッチがある。

14. ACCn　　$0 \leqq n \leqq 9$ で１０個の演算レジスタがある。

3.3 CLASS—N言語による学習プログラム記述例

```
NMOD4*0174    BPL - 021     DATE: 11/11/11   TIME: 06:15'52"      PROGRAM:                                    PAGE:   1

          SEQNO   PGLIN

              1           TEST:   PROC;                                                                    TST01010
              2                   DCL     CRCT BIN LINK,WRNG BIN LINK,ANS0 BIT(12) LINK,                   TST01020
              3                           ANS1 BIT(12)  LINK,ANS2 BIT(12) LINK,ANS3 BIT(12)                TST01030
              4                           LINK,ANS4 BIT(12) LINK,ANS5 BIT(12) LINK,ANS6                    TST01040
              5                           BIT(12) LINK,ANS7 BIT(12) LINK,ANS8 BIT(12) LINK,                TST01050
              6                           ANS9 BIT(12) LINK,HINT BIN LINK,CANS BIN LINK,                   TST01060
              7                           SWDS BIN LINK,TMUP BIN LINK,ACC0 BIN LINK,ACC1 BIN               TST01070
              8                           LINK,ACC2 BIN LINK,ACC3 BIN LINK,ACC4 BIN LINK,                  TST01080
              9                           ACC5 BIN LINK,ACC6 BIN LINK,ACC7 BIN LINK,ACC8 BIN               TST01090
             10                           LINK,ACC9 BIN LINK,CCNT BIN LINK,WCNT BIN LINK,                  TST01100
             11                           TCNT BIN LINK,REVK BIN LINK,OKKY BIN LINK,                       TST01110
             12                           ANSP BIT(12),SOR1 BIN LINK;                                      TST01111
             13                   ACC9=0;                                                                  TST01112
             14           STEP2:  CALL SLIDE(2,2,'C');        /*STEP 2C(2)*/                            TST01120
             15                   CALL TIME(0);                                                            TST01130
             16           STEP3:  CALL SLIDE(3,3,'C');        /*STEP 3C(3)*/                            TST01140
             17                   CALL TIME(0);                                                            TST01150
             18           STEP4:  CALL SLIDE(4,4,'C');        /*STEP 4C(4)*/                            TST01160
             19                   CALL TIME(0);                                                            TST01170
             20           STEP5:  CALL SLIDE(5,5,'O');        /*STEP 5O(5)*/                            TST01180
             21                   CALL ANS('CA1;2');                                                       TST01190
             22                   CALL TIME(0);                                                            TST01200
             23                   ACC9=ACC9+1;                                                             TST02010
             24                   IF CRCT=1 THEN GO TO S503;                                               TST02020
             25                   CALL LAMP(4);                                                            TST02030
             26                   IF ACC9=1 THEN GO TO S502;                                               TST02040
             27           S500:   CALL SLIDE(6,7,'O');        /*STEP 6,1O(7)*/                          TST02050
             28                   CALL ANS('CA1;0');                                                       TST02060
             29                   CALL TIME(0);                                                            TST02070
             30                   IF CRCT=1 THEN GO TO S501;                                               TST02080
             31                   CALL SLIDE(5,19,'C');       /*STEP 5C(19)*/                           TST02090
             32                   CALL TIME(0);                                                            TST02100
             33                   RETURN;                                                                  TST02110
             34           S501:   ACC9=0;                                                                  TST02120
             35                   GO TO STEP5;                                                             TST02130
             36           S502:   CALL SLIDE(5,6,'C');        /*STEP 5T(6)*/                            TST02140
             37                   CALL TIME(0);                                                            TST02150
             38                   GO TO STEP5;                                                             TST02160
             39           S503:   CALL LAMP(8);                                                            TST02170
             40                   ACC9=0;                                                                  TST02180
             41           STEP6:  CALL SLIDE(6,8,'C');        /*STEP 6C(8)*/                            TST02190
             42                   CALL TIME(0);                                                            TST02200
             43           STEP7:  CALL SLIDE(7,9,'O');        /*STEP 7O(9)*/                            TST03010
             44                   CALL ANS('CA1;2');                                                       TST03020
             45                   CALL TIME(0);                                                            TST03030
             46                   ACC9=ACC9+1;                                                             TST03040
             47                   IF WRNG=1 THEN GO TO S703;                                               TST03050
             48                   IF ACC9=1 THEN GO TO S702;                                               TST03060
```

第4章　制御プログラム

4.1　制御プログラムの概要

学習プログラムは学習端末に教材を提示すること，これに対する学習者の反応を分析すること，その結果に基づいて所定の論理演算や評価計算を行なうこと等の組合わせを単位として記述されている。それら学習プログラムの特に提示ステートメントによりこの制御プログラムが呼ばれる。制御プログラムはすべてアセンブリ言語で作られており，それぞれ機能によってサブルーチン化されている。

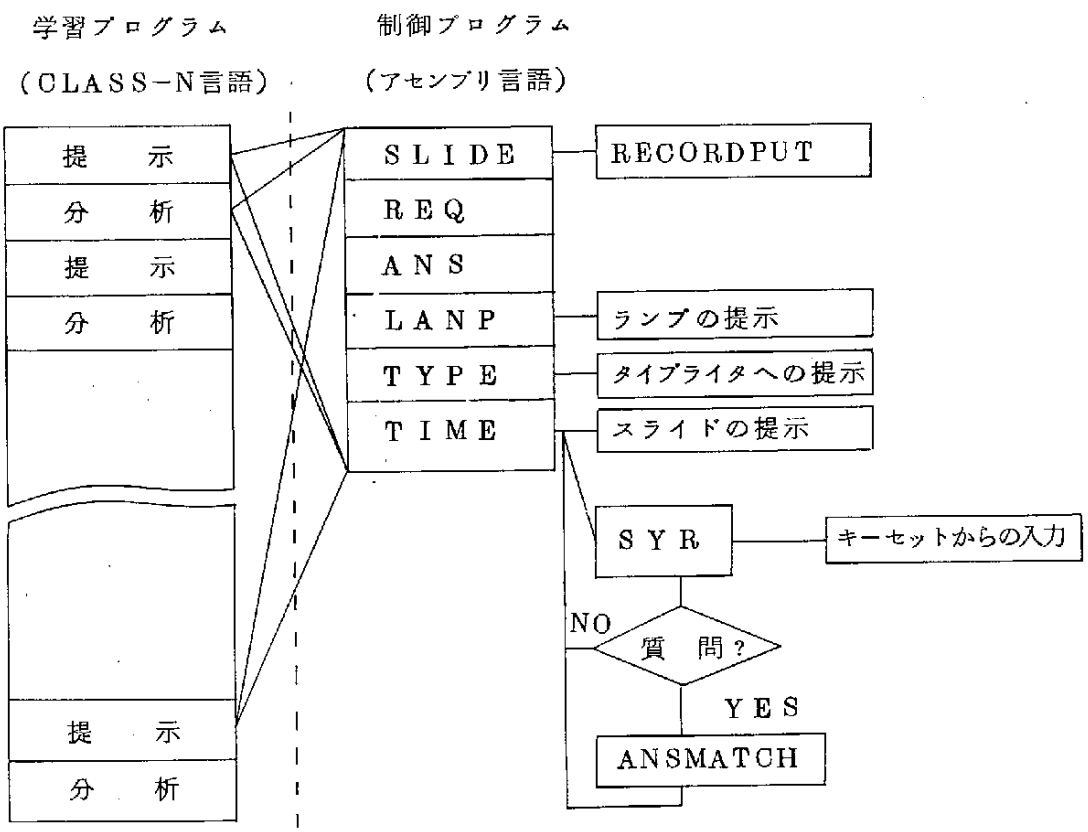


図4－1　学習プログラムと制御プログラムの関係

4.2　制御プログラムの処理

以下に制御プログラムの機能，ブロックチャート及びその処理内容を示す。

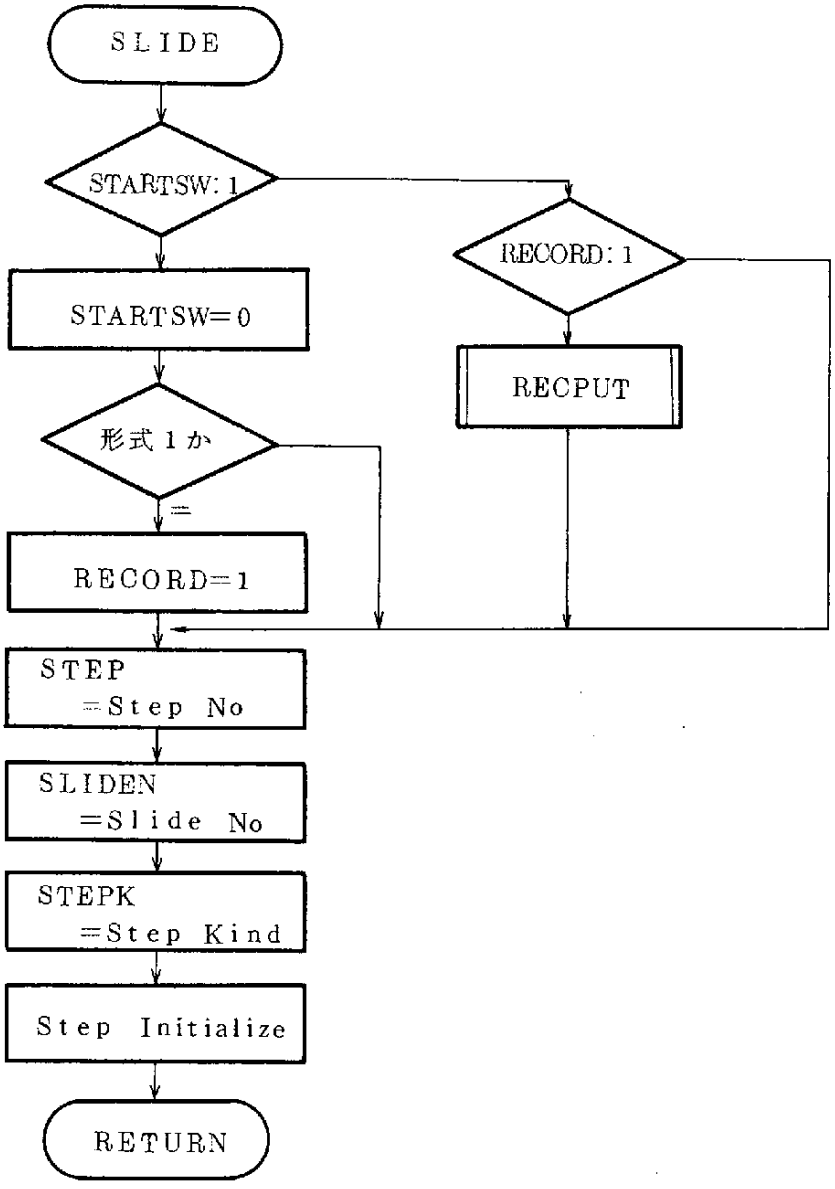
(1)　ＳＬＩＤＥルーチン

1.　機能

① 学習プログラムからＣＡＬＬ　ＳＬＩＤＥ（２，２，'Ｃ'）；の形式で送られてくるアーギュメントを受けとる。

② メインルーチンから送られてくるレコードスイッチの内容によって学習記録をとるかどうかの判断をする。

2. ブロックチャート


SLIDE
STARTSW: 1
RECORD: 1
STARTSW=0
RECPUT
形式 1 か
=
RECORD=1
STEP
=Step No
SLIDEN
=Slide No
STEPK
=Step Kind
Step Initialize
RETURN

3. 処理内容

① CALL SLIDE（2，2，'C'）；の各アーギュメントは，ステップナンバー，スライドナンバー，ステップカインドで，それぞれをSTEPN，SLIDEN，STEPKのエリアにSAVEする。

② RECORD SWITCH・・・1　学習記録をとる。

・・・0　学習記録をとらない。

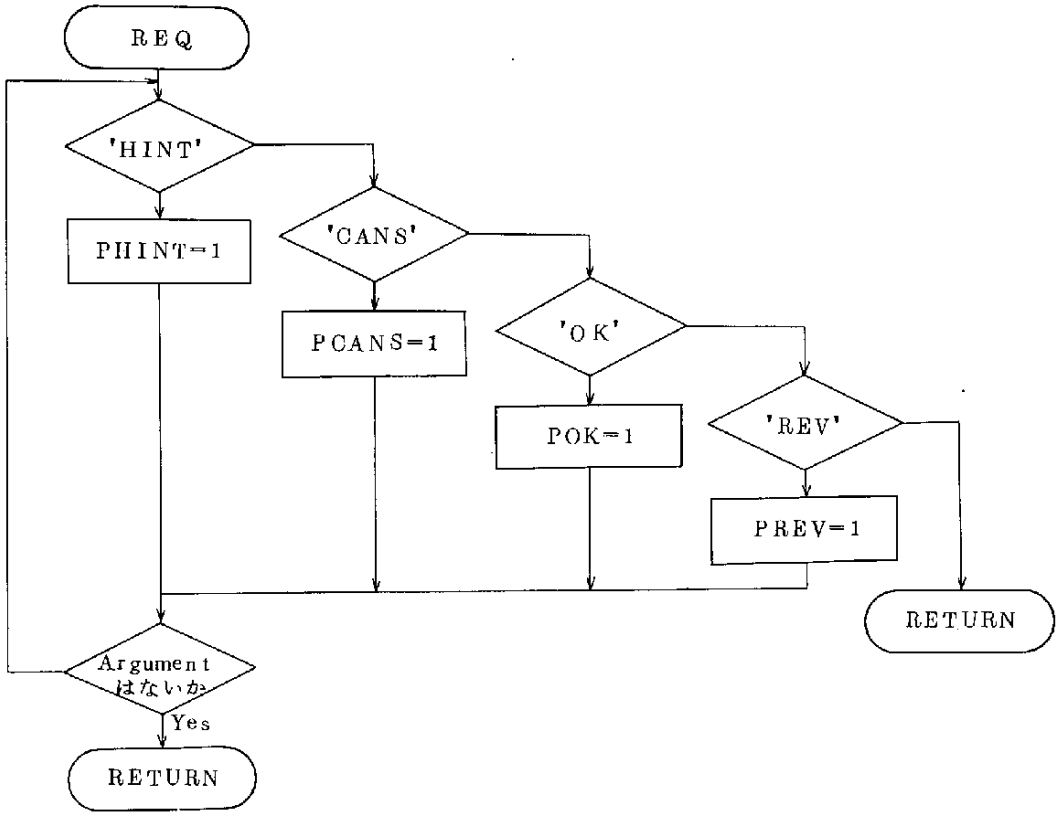
③ 各SWITCH，ANSSWITCHのINITIALIZEを行う。

(2) REQルーチン

1. 機能

学習プログラムからCALL REQ（'HINT'，'CANS'，'OK'，'REV'）；の形で送られてくるアーギュメントの有無を知らべて，PHINT，PCANS，POK，PREVのスイッチをセットする。

2. ブロックチャート



3. 処理内容

送られてきたアーギュメントが 'ＨＩＮＴ' , 'ＣＡＮＳ','ＯＫ','ＲＥＶ'のいずれにマッチするか知らべ,'ＨＩＮＴ' があった場合にはＰＨＩＮＴを１に,'ＣＡＮＳ' の場合にはＰＣＡＮＳを１に, 'ＯＫ' の場合にはＰＯＫを１に, 'ＲＥＶ' の場合にはＰＲＥＶを１にセットする。

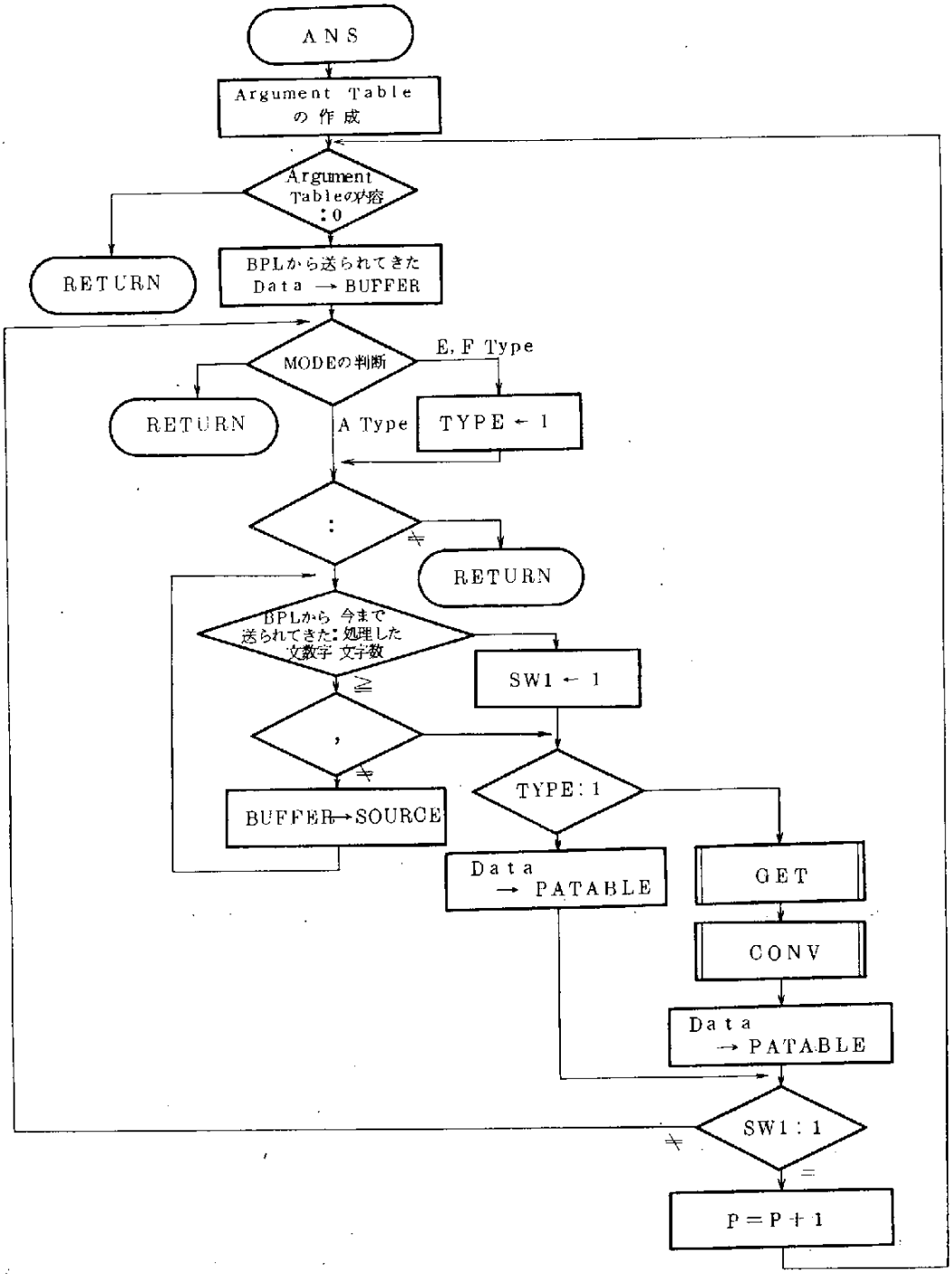
各スイッチは後にＴＩＭＥルーチンで使用する。

(3) ＡＮＳルーチン

1. 機能

学習プログラムからＣＡＬＬ　ＡＮＳ（'予想解答'）;の形で送られてくる,予想解答の解答テーブルを作成する。

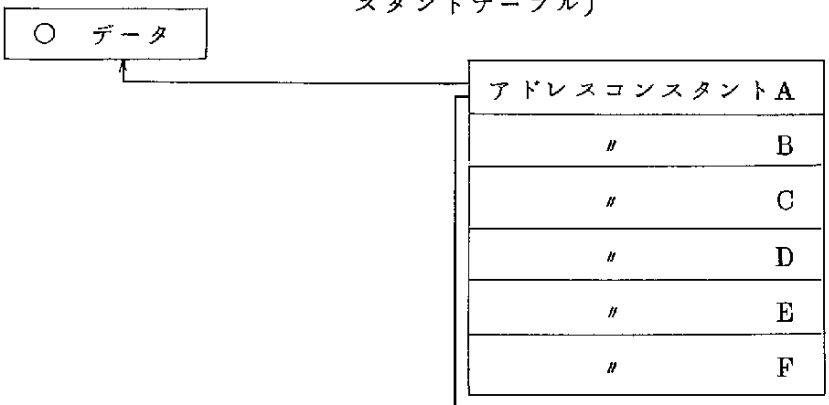
2. ブロックチャート


ANS
Argument Table の作成
Argument Tableの内容 :0
RETURN
BPLから送られてきた Data → BUFFER
MODEの判断
E, F Type
RETURN
A Type
TYPE ← 1
:
RETURN
BPLから 今まで 送られてきた:処理した 文数字 文字数
SW1 ← 1
,
BUFFER→SOURCE
TYPE: 1
Data → PATABLE
GET
CONV
Data → PATABLE
SW1 : 1
P = P + 1

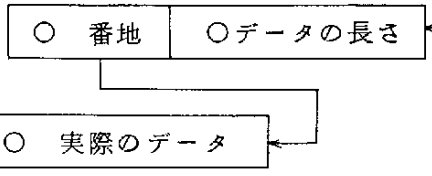
3. 処理内容

① 学習プログラムで各アセンブラモジュールをＣＡＬＬした場合，学習プログラムから送られてくる送り数（データ）は下図の方法でアセンブラーで受けとる。

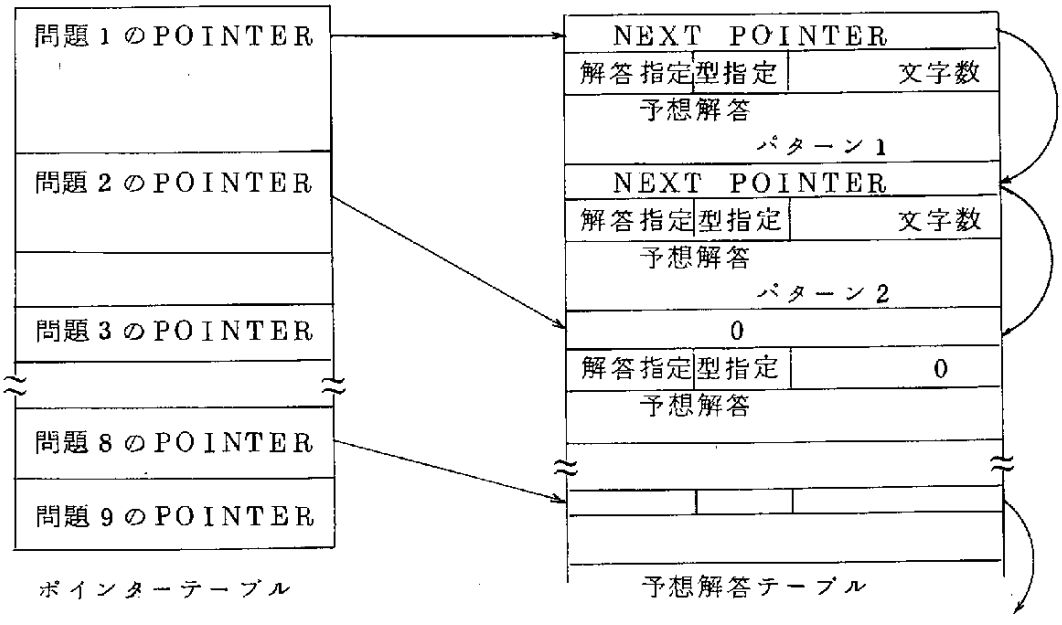
a） 算術型　　（ＢＰＬから送られてくる送り数のアドレスコンスタントテーブル）



b） 文字列型



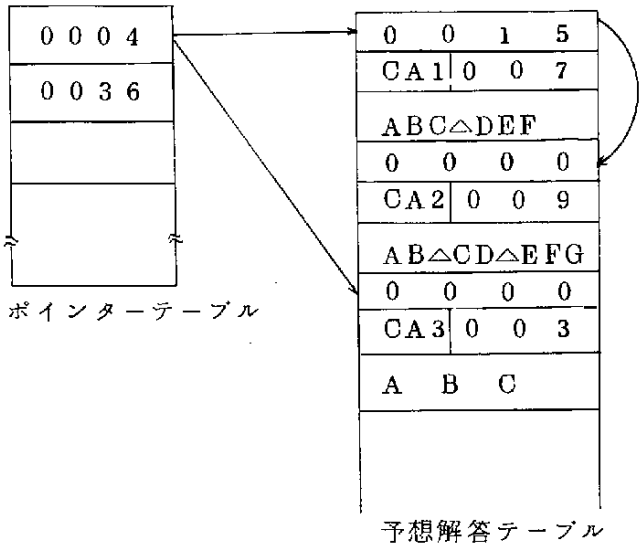
② 予想解答テーブル



例：

CALL ANS ('CA1：ABC△DEF, CA2：AB△CD△EFG','CA3：ABC');

の場合の予想解答テーブルとポインターテーブルを下に図示する。



(4) ＴＩＭＥルーチン

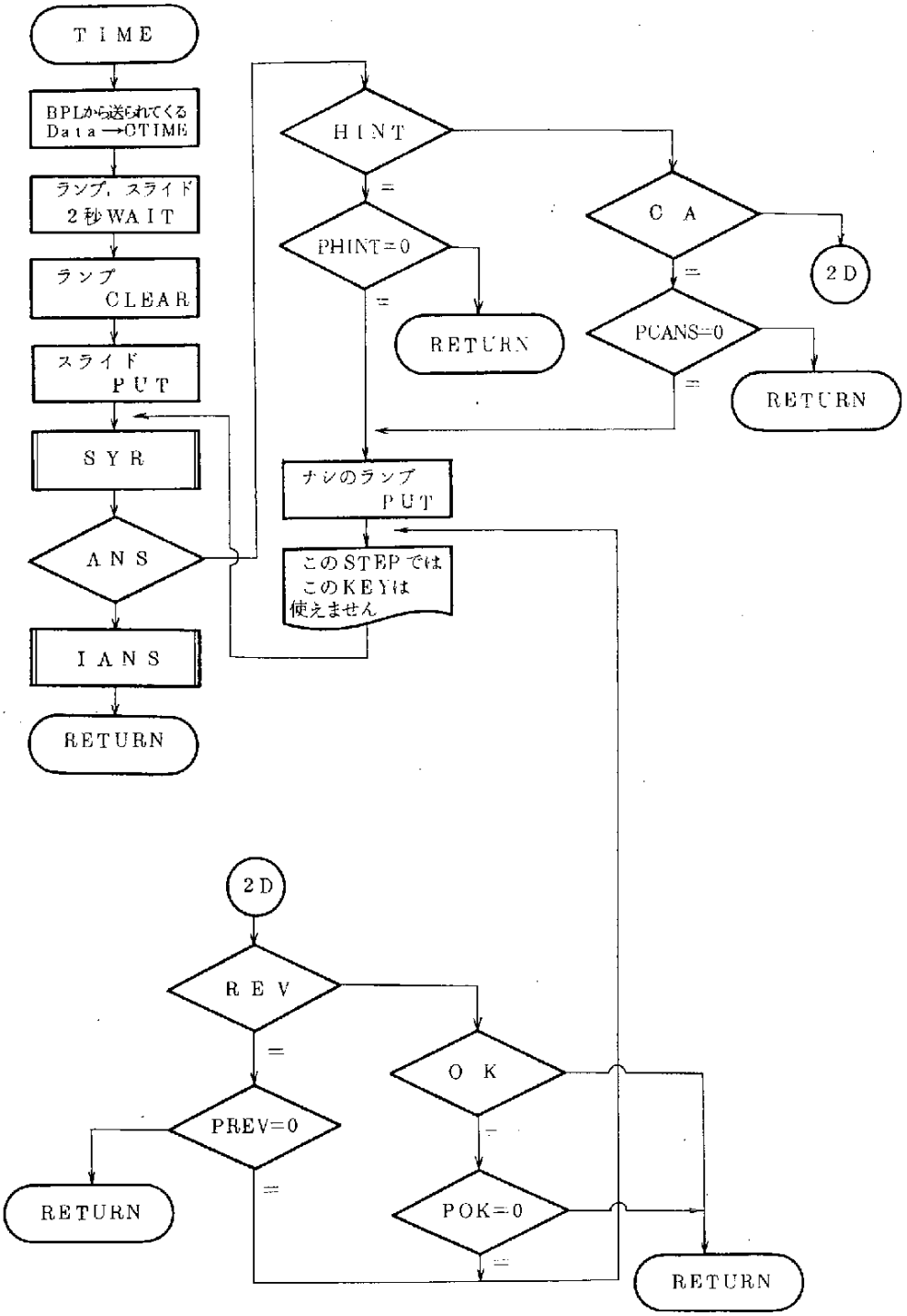
1. 機能

① 学習プログラムからＣＡＬＬ ＴＩＭＥ (m);の形式で送られてくる応答時間をアセンブラのエリアにセットをする。

② ＳＬＩＤＥ，ＬＡＭＰの提示

③ ＩＡＮＳルーチンへ飛んでＡＮＳＷＥＲ ＭＡＴＣＨを行なうか否かの判断をＳＯＲＴ１の内容によって判断する。

2. ブロックチャート


TIME
BPLから送られてくる Data→CTIME
ランプ，スライド 2秒WAIT
ランプ CLEAR
スライド PUT
SYR
ANS
IANS
RETURN
HINT
=
PHINT=0
=
RETURN
CA
=
2D
PCANS=0
=
RETURN
ナゾのランプ PUT
このSTEPでは このKEYは 使えません
2D
REV
=
PREV=0
=
RETURN
OK
=
POK=0
=
RETURN

3. 処理内容

① ＳＬＩＤＥを提示する前に，ＳＬＩＤＥルーチンから送られてくるスライドナンバーを９進数に変換する。

② ＳＬＩＤＥ提示，ＬＡＭＰ提示

提示をするのはＰＵＴマクロを使用する。

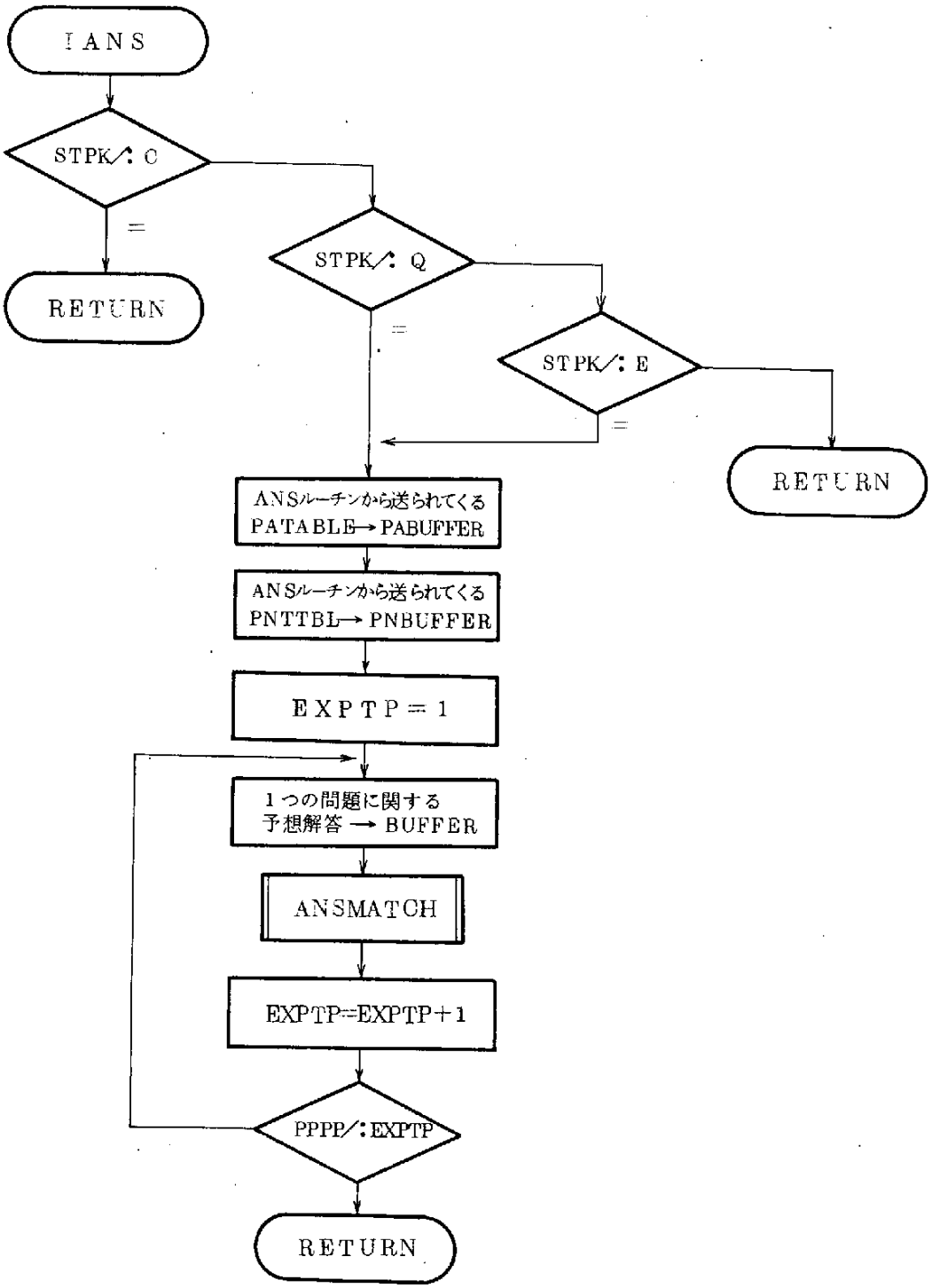
③ キーの使用あやまりや，答えの入力方法のあやまりなどに関するERROR MESSAGEなどを出力する。

(5) ＩＡＮＳルーチン

1. 機能

ＳＬＩＤＥルーチンから送られてくるステップカインドの内容によって，ＡＮＳＭＡＴＣＨルーチンへ飛ぶか否かの判断をする。

2. フロックチャート


!ANS
STPK/: C
=
RETURN
STPK/: Q
=
STPK/: E
=
RETURN
ANSルーチンから送られてくる
PATABLE→PABUFFER
ANSルーチンから送られてくる
PNTTBL→PNBUFFER
EXPTP=1
1つの問題に関する
予想解答 → BUFFER
ANSMATCH
EXPTP=EXPTP+1
PPPP/: EXPTP
RETURN

3. 処理内容

ステップカインド・・・C　TIMEルーチンへ戻る。

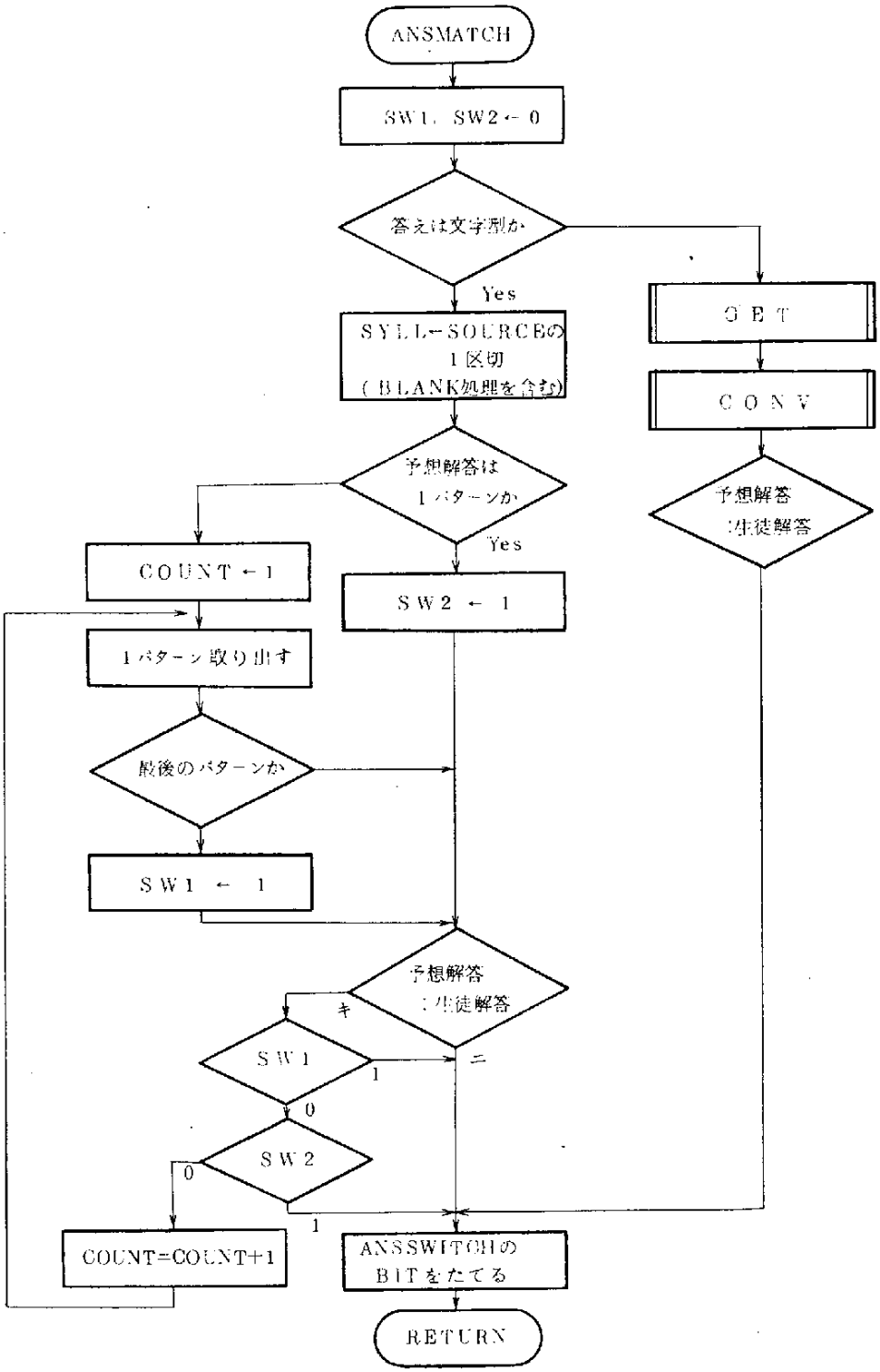
・・・E，Q　ANSMATCHルーチンへ飛ぶ。

(6) ANSWER　MATCHルーチン

1. 機能

ANSルーチンによって作成した予想解答テーブルと学習端末より学習者が入力した答えとを比較し，その結果を，bitのON／OFF によって表現し，ANSSWITCH テーブルを作成する。

2. ブロックチャート


ANSMATCH
SW1, SW2←0
答えは文字列か
Yes
SYLL←SOURCEの
1区切
(BLANK処理を含む)
予想解答は
1パターンか
Yes
COUNT←1
SW2 ← 1
1パターン取り出す
最後のパターンか
SW1 ← 1
予想解答
=生徒解答
キ
SW1
1
ニ
0
SW2
0
1
COUNT=COUNT+1
ANSSWITCHの
BITをたてる
RETURN
SET
CONV
予想解答
=生徒解答

3. 処理内容

ANSSWITCHテーブル

A．概略図

ANS0	ANS1～nまですべてOFFの時ONになる										
ANS1	RPLY	C	W	ANP1	ANP2	ANP3	ANP4	ANP5	ANP6	ANP7	ANP8
ANS2											
ANS3											
ANS4											
ANS5											
ANS6											
ANS7											
ANS8											
ANS9											

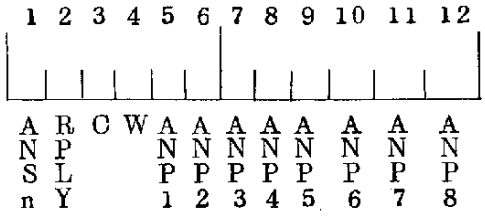
文字型でいずれかのパターンに一致した時ON
A，E，Fの各解答でWのついた解答と一致した時ON
A，E，Fの各解答でCのついた解答と一致した時ON
学習者からの解答があった時ON
ANPn のどれか又はE，Fの場合は解答と一致した時ON

CORRECT SWITCH [0 or 1] ANS1～nのすべてがONの時1

WRONG SWITCH [0 or 1] ANS1～nのとれからONでないものがあった時1

ANSSWITCH テーブルの１０個の要素は各２キャラクター（１２ｂｉｔ）から構成される。

2ch (12bit)



各ｂｉｔには上記のような意味がある。

Ｅ，Ｆタイプ，文字タイプで１パターンの時は１～４ｂｉｔ を使い，文字タイプで複数パターンの時は全ｂｉｔ 使用する。

ＡＮＳｎ：学習者の答が，予想解答と一致した時ＯＮ

ＲＰＬＹ：学習者から，なんらかの解答があった時ＯＮ

Ｃ　　：解答指定Ｃのつく予想解答と一致した時ＯＮ

Ｗ　　：解答指定Ｗのつく予想解答と一致した時ＯＮ

ＡＮＰ１～ＡＮＰ９：文字タイプで複数パターンの時，学習者の答が，予想解答の１番目のパターンと一致したらＡＮＰ１がＯＮ，９番目のパターンと一致したらＡＮＰ９がＯＮ，というように，何番目のパターンと一致したかによってどのｂｉｔ をＯＮにするかがきまる。

B．ANSSWITCH各パターン，8進コード，対応文字

ANSn	RPLY	C	W	ANP1	ANP2	ANP3	ANP4	ANP5	ANP6	ANP7	ANP8	8進コード	対応文字		
												0000	00	解答ナシ	
■												4000	-0	ANSOがON	
	■											2000	+0	予想解答と不一致	
■	■	■										7000	Y0		E,F Type
■	■		■									6400	U0		A Type
■	■											6000	＜0		1 Pattern
■	■	■		■								7200	@0		
■	■		■	■								6600	W0		
■	■			■								6200	S0		
■	■	■			■							7100	Z0		
■	■		■		■							6500	V0		
■	■				■							6100	／0		
■	■	■				■						7040	Y-		
■	■		■			■						6440	U-		
■	■					■						6040	＜-		A Type
■	■	■					■					7020	Y+		
■	■		■				■					6420	U+		
■	■						■					6020	＜+		
■	■	■						■				7010	Y8		
■	■		■					■				6410	U8		
■	■							■				6010	＜8		
■	■	■							■			7004	Y4		
■	■		■						■			6404	U4		
■	■								■			6004	＜4		
■	■	■								■		7002	＜2		
■	■		■							■		6402	U2		
■	■									■		6002	＜2		
■	■	■									■	7001	Y1		
■	■		■								■	6401	U1		
■	■										■	6001	＜1		

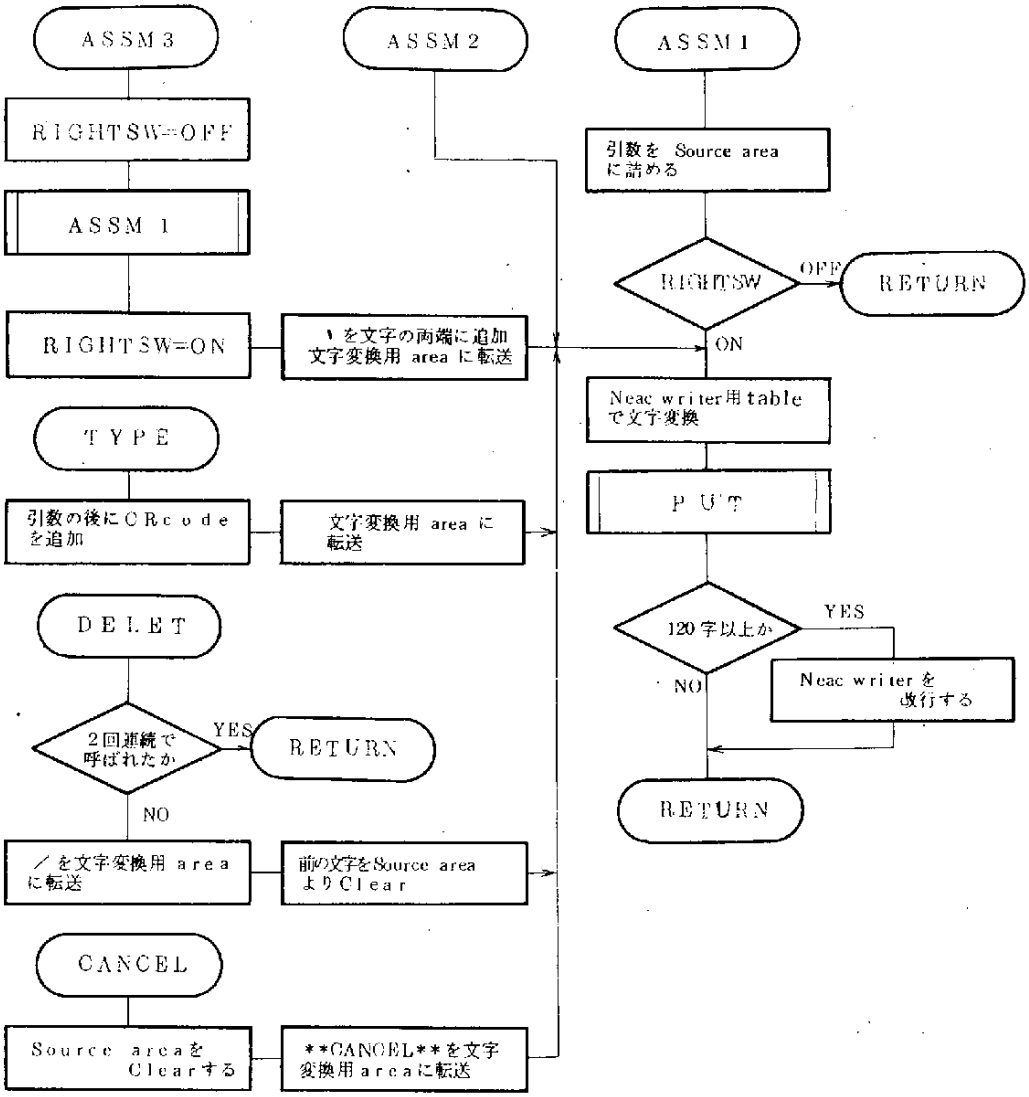
(7) TYPEルーチン

1. 機能

① 学習プログラムからCALL TYPE ('文字列');の形で送られてくる文字列の内容をNEAC WRITERに出力する。

② 他の制御プログラムからNEAC WRITERに出力したい場合もその文字列のある場所と文字数を送れば同じように出力される。

2. ブロックチャート


ASSM3
RIGHTSW=OFF
ASSM 1
RIGHTSW=ON
TYPE
引数の後にCRcodeを追加
DELET
2回連続で呼ばれたか
YES
RETURN
NO
／を文字変換用 area に転送
CANCEL
Source areaをClearする
ASSM2
＼を文字の両端に追加 文字変換用 area に転送
文字変換用 area に転送
前の文字をSource areaよりClear
CANCELを文字変換用 areaに転送
ASSM1
引数を Source area に詰める
RIGHTSW
OFF
RETURN
ON
Neac writer用table で文字変換
PUT
120字以上か
YES
Neac writer を改行する
NO
RETURN

3. 処理内容

① ＡＳＳＭ１

1）引数をｓｏｕｒｃｅ　ａｒｅａに詰める

2）引数を Neac writer 用ｔａｂｌｅで変換し，その結果をNeac writer に type outする

② ＡＳＳＭ２

1） Neac writer用 table で変換し typeout する。

③ ＡＳＳＭ３

1）引数の両側に＇を追加し引数の文字数に２を加える。

2）＇を除いた部分を source area に詰める。

3） Neac writer 用 table で変換し typeout する。

④ CALL TYPE（＇××××…×＇）で呼ばれた場合の処理

1）引数を typeout用の work area に転送する。

2） typeout する。

⑤ ＤＥＬＥＴ

1）／をNeac writerに typeout する。

2）前にＧＥＴされた keyに対応する文字を source area から clearする。

⑥ CANCEL

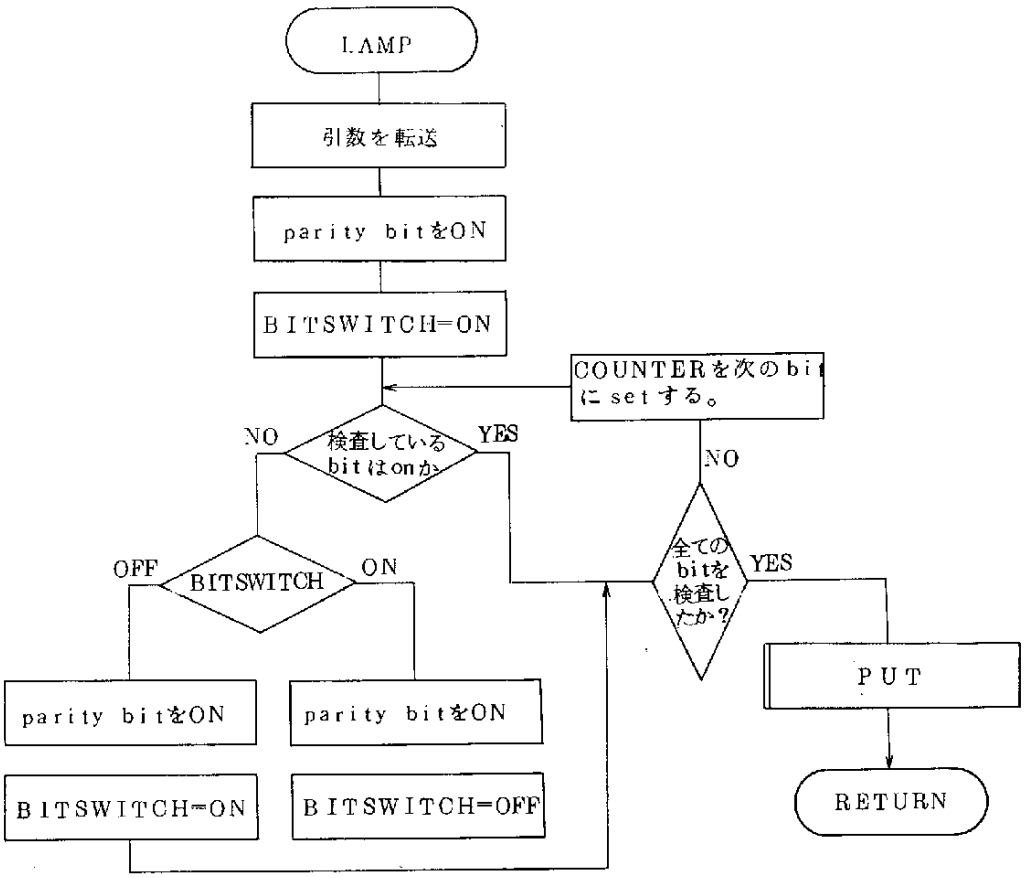
1）＊＊CANCEL＊＊を Neac writer に typeout する。

2） source area の内容を全て clear する。

(8) ＬＡＭＰルーチン

1. 機能

学習プログラムからＣＡＬＬ　ＬＡＭＰ（m）;の形で送られてくる，mの番号に相当するランプを点灯する。



3. 処理内容

学習ProgramからCALL LAMP （××）；で呼ばれた場合下記の処理を行う。

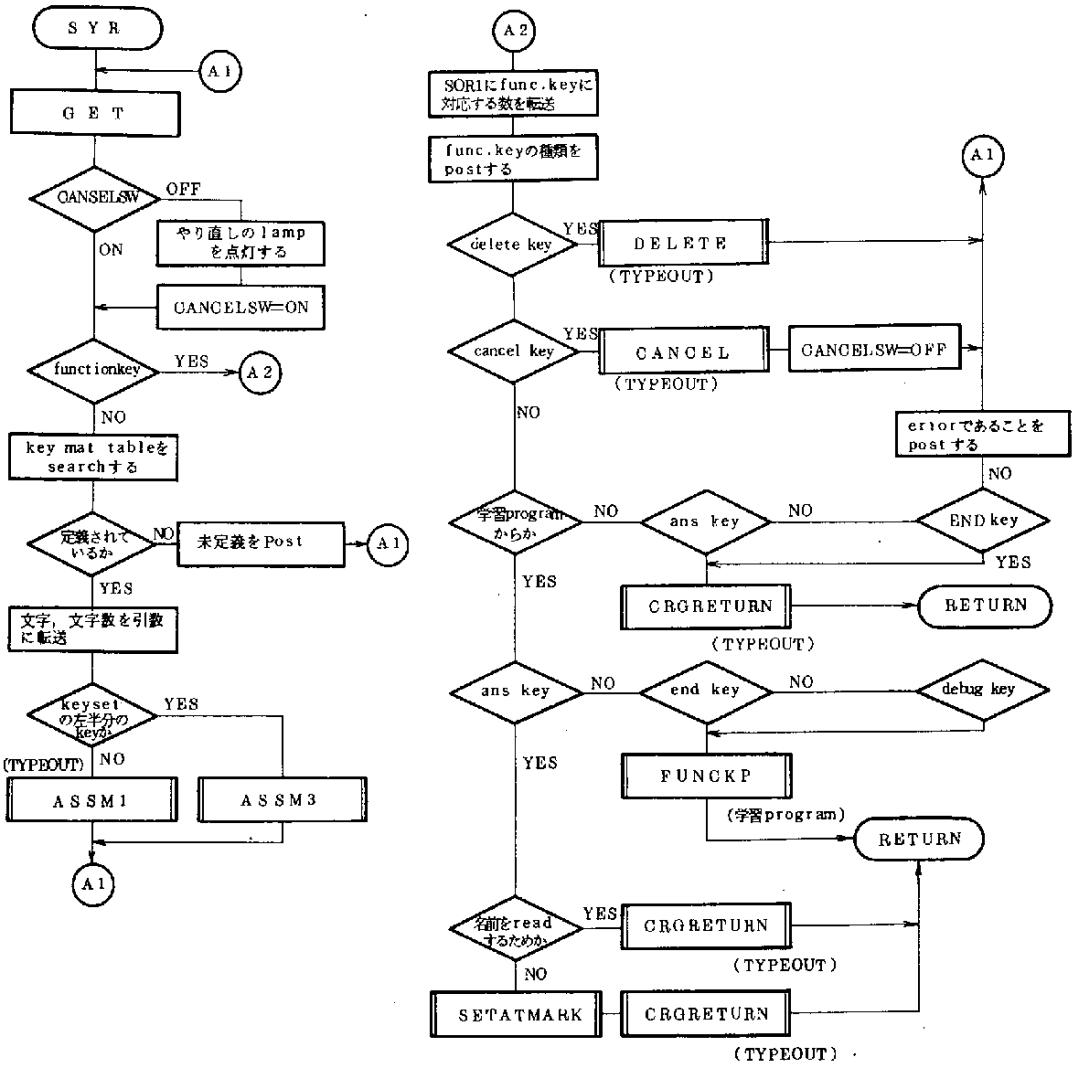
1）××の値が1，2，4，8，16．36，の値のときそれぞれ lamp の右から1，2，3，4，5，6番目の lamp が点灯する。

(9) SYRルーチン

1. 機能

KEYSETより1keyづつGETしfunction key に対してはそれに応じた処理を行い，それ以外のkeyに対してはkeymat table により変換しその結果を source area にセットする。

2. ブロックチャート


SYR
A1
GET
CANSELSW
OFF
やり直しのlampを点灯する
ON
CANCELSW=ON
functionkey
YES
A2
NO
key mat tableをsearchする
定義されているか
NO
未定義をPost
A1
YES
文字，文字数を引数に転送
keysetの左半分のkeyか
YES
(TYPEOUT)
NO
ASSM1
ASSM3
A1
A2
SOR1にfunc.keyに対応する数を転送
func.keyの種類をpostする
A1
delete key
YES
DELETE
(TYPEOUT)
cancel key
YES
CANCEL
CANCELSW=OFF
(TYPEOUT)
NO
errorであることをpostする
NO
学習programからか
NO
ans key
NO
END key
YES
YES
CRGRETURN
RETURN
(TYPEOUT)
ans key
NO
end key
NO
debug key
YES
FUNCKP
(学習program)
RETURN
名前をreadするためか
YES
CRGRETURN
(TYPEOUT)
NO
SETATMARK
CRGRETURN
(TYPEOUT)

3. 処理内容

① Keysetより1keyづつGETしfunction keyなら下記の処理を行う。

1）いかなるfunction keyかをNeac writerに表示するため,function keyに対応させてあるmessage及びその文字数をTYPEOUT routineに渡す。Neac writerへの表示はTYPEOUTroutineの各entry（この場合ASSM2）が処理する。

2）DELET keyならTYPEOUTroutineのDELETにbranchする。

3）CANCEL keyならTYPEOUTroutineのCANCELにbranchする

4）ANS keyならSYRよりreturnする。

5）押されたfunction keyに対し，あらかじめ定められた値をSORT1/に転送する。

6）押されたfunction key に対し，NCOMMON 内に定義してあるareaに1を転送。

その他のfunction keyに対応するareaには0を転送する。

7）DELET, CANCEL, ANSkey以外のfunction keyが押された場合学習Program内のFUNCKPにbranchする。

② data keyなら下記の処理を行う。

1）keyに対応する文字，文字数をkeymat tableよりsearchし，文字と文字数をTYPEOUTroutineに渡す。

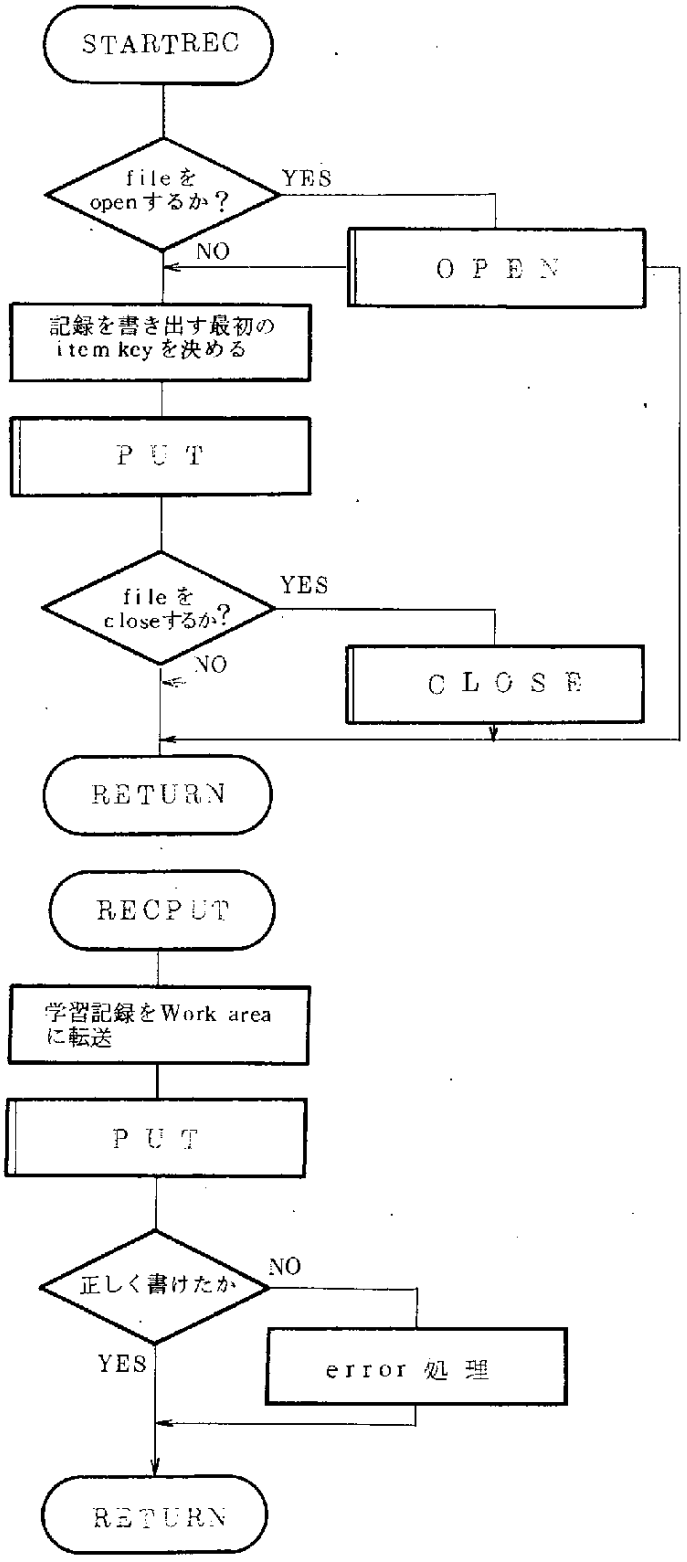
2）keyがkeysetの左半分のdata keyならTYPEOUT routineのASSM3右半分ならASSM1にbranchする。

⑩ RECORDPUTルーチン

1. 機能

各学習者に対して各学習ステップ毎に学習記録を出力する。

2. ブロックチャート


STARTREC
file を open するか？
YES
NO
O P E N
記録を書き出す最初の item key を決める
P U T
file を close するか？
YES
NO
C L O S E
RETURN
RECPUT
学習記録を Work area に転送
P U T
正しく書けたか
NO
YES
error 処 理
RETURN

3. 処理内容

① 各学習者毎の学習記録はIndex sequential file形式でデータ，ファイル上に書かれる。

② STARTREC が呼ばれたとき下記の処理をする。

1）記録を書き出す最初のItem keyを決める。

2）決定した最初のitemに学習program の名前，使用者の名前を書き出す。

③ RECPUTが呼ばれたとき下記のような情報を書き出す。

（学習記録の内容）

STEP-NO	step番号	2ch
SLIDEN	slide番号	2ch
STEPK	stepの種類	1ch
CRCT	answer matchの結果setされた情報	1ch
WRNG	answer matchの結果setされた情報	1ch
SORT1	学習者が押したfunction keyの種類	2ch
ANS0～ANS9	answer matchの結果setされた情報	2ch×10
SW	学習programでsetされた値	2ch
CCNT	〃	4ch
WCNT	〃	4ch
TCNT	〃	4ch
KINTIME	学習のkey inに要した時間	4ch
TTIME	全学習時間	4ch
SOURCE	学習者がkey inした情報	75ch
ACC0～ACC9	学習programでsetされた値	4ch×10

第5章　学習プログラム

5.1　目標行動とレディネスの設定

1）目標行動

このＣＬＡＳＳ－Ｎシステムで学習することにより，ＣＰＬの文法的知識が得られ実際のＴＳＳ端末からＣＰＬを使ってプログラミングできるようになることを目的としている。

2）レディネス

① コンピュータの基礎知識がある人

② 他の言語（例えばFORTRAN）をある程度知っている人

③ ＣＰＬ言語に関して初心者

5.2　学習方法

ＣＰＬのようなコンピュータ言語の教育においては文法的知識と実際に端末からプログラミングして実行させてみる，その結果に対して適切な診断を行ない学習した知識をより完全なものにしてやる，それが学習効果をあげることになる。

以上の方針から各セクションの大部分は個別指導型(tutorial instruction)を主とし，練習問題等はドリル演習型（drill and practice）により学習を進めるようになっている。

5.3　学習機能

1）問題に対する回答形式は，多枝選択式だけでなく，プログラムの一部訂正とかステートメントの入力による回答とか，実際にプログラムを実行させてその結果を入力させる等の方法がある。

2）学習の流れはブランチ形式を基本として，その条件としては，学習者の回答内容，過去の学習経過（得点，誤答回数），応答時間の制限，学習者からの要求（ヒント，正答，前のステップ提示）等により決定される。

3）学習者からは，ヒント要求，正答要求，前のステップに戻ることの要求等を出すことができる。

4）学習者の応答時間を制限するようにプログラムすることができる。

5）学習者毎の学習経過記録は学習プログラムの各ステップ毎にとられ，これは学習終了後出力される。

6）練習問題等は実際のＴＳＳ端末よりプログラミングし実行させてみることができる。

5.4 学習の流れ

学習者の応答と学習の進み方は次のようになっている。

提示されたスライドが説明のステップならば，問題の場合と異なり，学習者は解答を入れる必要はなく，その説明の内容をよく理解したらＡＮＳキーを押せば次のステップのスライドが提示される。

提示されたスライドが問題の場合には学習者はそれに対して解答を行なう。解答はキーセットより解答メッセージを入力し，最後にＡＮＳキーを押せばよい。ＡＮＳキーを押す前なら学習者は解答内容を任意に修正することができる。システムは現在のステップが問題のステップならば学習者の解答内容を判定しそれによって適当な次のステップのスライドを提示する。

提示されたスライドが練習問題の場合には学習者はその内容をよく理解してから隣りのＴＳＳ端末で実際にプログラムを作成する。そして学習端末に戻りＡＮＳキーを押す。次のステップではこのデータを使って実行してみなさいというスライドが提示されるので，またＴＳＳ端末より実行させる。それが終ったなら同様にＡＮＳキーを押す。次のステップでは実行して出力された値をキーセットから入力して下さいという，さきほどの問題提示のときと同じようなステップのスライドが提示されるのでＴＳＳ端末の出力結果を見ながらキーセットより解答メッセージを入力し，最後にＡＮＳキーを押せば，問題のときと同様に解答内容が判定され適当な次のステップが選択される。

学習者が説明，問題，練習問題等のステップで解答が見出せないとか，どうしてよいかわからないという場合に備えてヒントシーケンスを用意することができ，学習者はＨＩＮＴキーによりそれを要求できる。

ヒントシーケンスの学習途中で問題の解答，説明の内容がわかり，また元の問題

や，説明ステップに戻りたいとき，学習者がＯＫキーを押せば学習は自動的に元のステップに戻る。

問題や練習問題のステップでどうしても答を見出せないときは正答を要求するようなステップもできている。そのときには学習者はＣＡキーによりそれを要求する。プログラムは正答を提示し適当な次のステップに進む。

学習者が前のステップでだされた説明をもう一度見たい場合，その要求を受け入れられるようにプログラムされていて，ＲＥＶキーを押せば１つ前のステップに戻ることができる。

さらに学習プログラム作成者は学習者の応答時間を制限することができる。この場合，学習者が制限時間内に答えられなかった場合はタイムアップの条件となり適当なステップに分岐される。

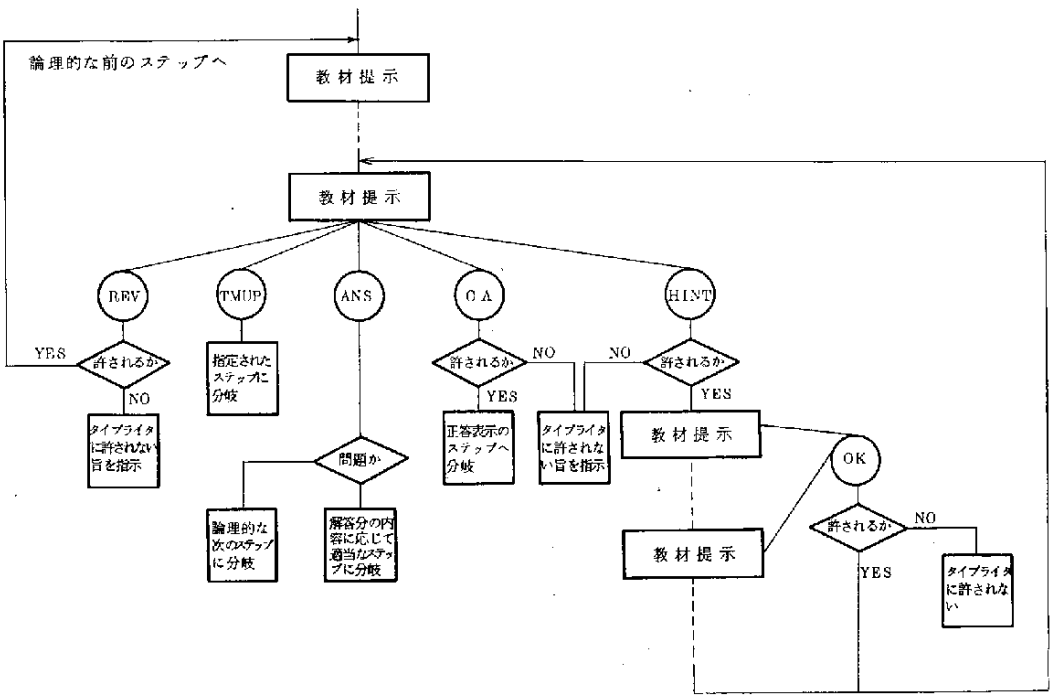


図５－１　学習者応答と学習の進み方

5.5 学習シーケンス

学習シーケンスは教材を提示してそれに対する学習者の反応により次にどの教材を提示するかを決める大まかな流れ図であり，各記号や数字は次のような意味をもっている。

一般形：　m〔．n〕X（j）

m：ステップ番号

n：サブステップ番号

X：ステップの種類であり次のものがある

C：　説明

Q：　質問

E：　練習問題

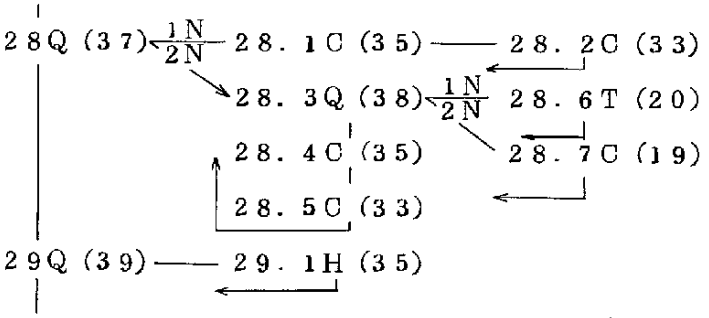
T：　治療

H：　ヒント

A：　正答

j：スライド番号

例：



5.6 学習プログラムの実例

CLASS システムは，FACOM230−60で使用できる会話形言語を教育するものである。その会話形言語としてCPL，BACCUS，FORTRAN，LINED等があり，今回は特に当センターで開発したCPL（会話形PL／I）を学習教材としている。

CPLの学習プログラムとしては全部で15セクション位になる予定であるが現在セクション3まで完成している。

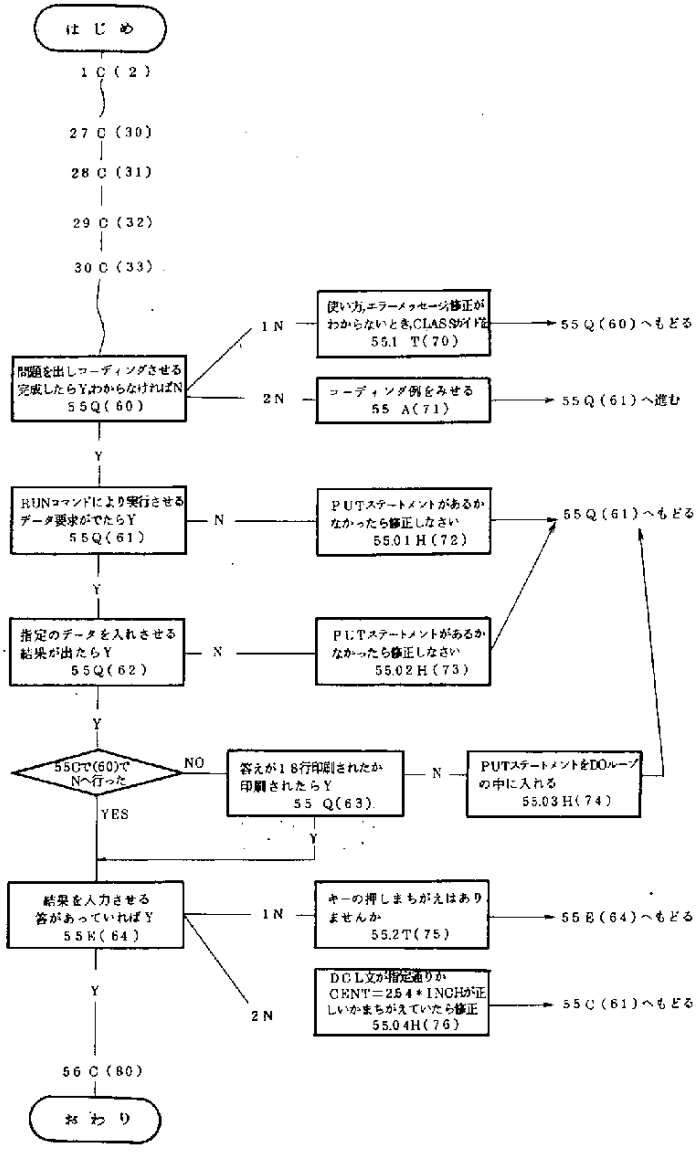
内容：　セクション１　ＣＰＬ入門（プログラムの構成，演算処理１）

セクション２　ＩＦとＤＯステートメント

セクション３　複雑なＤＯステートメント

このＣＬＡＳＳ－Ｎシステムでは特に学習端末とＴＳＳ端末とのインターフエイスのところを特徴としているので，学習プログラムの練習問題の部分を実例として次に説明する。

（１）　学習フロー（ＣＰＬセクション２）



(2) 学習プログラムリスト（ＣＰＬセクション２）

```
NMOD4*074     BPL - 021     DATE: 03/13/72   TIME: 12:31'45"      PROGRAM:

                    SEQNO   PGLIN

                      1     TEST:    PROC;
                      2              DCL   CRCT BIN LINK,WRNG BIN LINK,ANS0 BIT(12) LINK,
                      3                    ANS1 BIT(12)  LINK,ANS2 BIT(12) LINK,ANS3 BIT(12)
                      4                    LINK,ANS4 BIT(12) LINK,ANS5 BIT(12) LINK,ANS6
                      5                    BIT(12) LINK,ANS7 BIT(12) LINK,ANS8 BIT(12) LINK,
                      6                    ANS9 BIT(12) LINK,HINT BIN LINK,CANS BIN LINK,
                      7                    SWD5 BIN LINK,TMUP BIN LINK,ACC0 BIN LINK,ACC1 BIN
                      8                    LINK,ACC2 BIN LINK,ACC3 BIN LINK,ACC4 BIN LINK,
                      9                    ACC5 BIN LINK,ACC6 BIN LINK,ACC7 BIN LINK,ACC8 BIN
                     10                    LINK,ACC9 BIN LINK,CCNT BIN LINK,WCNT BIN LINK,
                     11                    TCNT BIN LINK,REVK BIN LINK,OKKY BIN LINK,
                     12                    ANSP BIT(12),SOR1 BIN LINK;
                     13                ACC8=0;
                     14                ACC9=0;
                     15           STEP1:   CALL SLIDE(1,2,'C');                  /*STEP 1C(2)*/
                     16                    CALL TIME(0);
                     17               /*STEP2 KARA STEP26*/
                     18           STEP27:  CALL SLIDE(27,30,'C');                /*STEP 27C(30)*/
                     19                    CALL TIME(0);
                     20           STEP28:  CALL SLIDE(28,31,'C');                /*STEP 28C(31)*/
                     21                    CALL TIME(0);
                     22           STEP29:  CALL SLIDE(29,32,'C');                /*STEP 29C(32)*/
                     23                    CALL TIME(0);
                     24           STEP30:  CALL SLIDE(30,33,'C');                /*STEP 30C(33)*/
                     25                    CALL TIME(0);
                     26               /*STEP31 KARA STEP54*/
                     27           STEP551: CALL SLIDE(55,60,'Q');                /*STEP 55Q(60)*/
                     28                    CALL ANS('CA1:Y');
                     29                    CALL TIME(0);
                     30                    ACC9=ACC9+1;
                     31                    IF CRCT=1 THEN GO TO S5512;
                     32                    IF ACC9=1 THEN GO TO S5511;
                     33                    ACC8=ACC8+1;
                     34                    CALL SLIDE(55,71,'C');                /*STEP 55A(71)*/
                     35                    CALL TIME(0);
                     36                    ACC9=0;
                     37                    GO TO STEP552;
                     38           S5511:   ACC8=ACC8+1;
                     39                    CALL SLIDE(55,70,'C');                /*STEP 55.1T(70)*/
                     40                    CALL TIME(0);
                     41                    GO TO STEP551;
                     42           S5512:   ACC9=0;
                     43           STEP552: CALL SLIDE(55,61,'Q');                /*STEP 55Q(61)*/
                     44                    CALL ANS('CA1:Y');
                     45                    CALL TIME(0);
                     46                    IF CRCT=1 THEN GO TO STEP553;
                     47                    CALL SLIDE(55,72,'Q');                /*STEP 55.01H(72)*/
                     48                    CALL TIME(0);
```

```
SEQNO  PGLIN

  49                GO TO STEP552;
  50        STEP553: CALL SLIDE(55,62,'Q');            /*STEP 55Q(62)*/
  51                 CALL ANS('CA1:Y');
  52                 CALL TIME(0);
  53                 IF CRCT=1 THEN GO TO S5531;
  54                 CALL SLIDE(55,73,'C');            /*STEP 55.02H(73)*/
  55                 CALL TIME(0);
  56                 GO TO STEP552;
  57        S5531:   IF ACC8>0 THEN GO TO STEP555;
  58        STEP554: CALL SLIDE(55,63,'Q');            /*STEP 55Q(63)*/
  59                 CALL ANS('CA1:Y');
  60                 CALL TIME(0);
  61                 IF CRCT=1 THEN GO TO STEP555;
  62                 CALL SLIDE(55,74,'C');            /*STEP 55.03H(74)*/
  63                 CALL TIME(0);
  64                 GO TO STEP552;
  65        STEP555: CALL SLIDE(55,64,'E');            /*STEP 55E(64)*/
  66                 CALL ANS('CF:7.62','CF:17.78','CF:12','CF:14','CF:43.18');
  67                 CALL TIME(0);
  68                 ACC9=ACC9+1;
  69                 IF CRCT=1 THEN GO TO S5552;
  70                 IF ACC9=1 THEN GO TO S5551;
  71                 CALL SLIDE(55,76,'C');            /*STEP 55.04H(76)*/
  72                 CALL TIME(0);
  73                 ACC9=0;
  74                 GO TO STEP552;
  75        S5551:   CALL SLIDE(55,75,'C');            /*STEP 55.2T(75)*/
  76                 CALL TIME(0);
  77                 GO TO STEP555;
  78        S5552:   ACC9=0;
  79        STEP56:  CALL SLIDE(56,80,'C');            /*STEP 56C(80)*/
  80                 CALL TIME(0);
  81                RETURN;
  82                 END;
```

(3) スライド（ＣＰＬセクション２）

```
**********************
*セクション２          *
*                     *
*  ＩＦ文，ＤＯ文の学習*
*                     *
**********************
```

前のセクションでは，ＣＰＬの基礎的なことについて学習しましたが，このセクションでは「比較，選択，分類等の判断をする」とか「演算を繰返す」場合はどうしたらよいかを学習しましょう。

特に指定のない限りは，次に進むにはＡＮＳキーを押して下さい。

1－C－2

すべてのステートメントを，たった一回だけ行なって終るというプログラムは，きわめて稀です。

ふつうは，あるステートメントのグループを何回か繰返すという処理が入ります。

そこで，その繰返しということについて，これから学習していきましょう。

27－C－30

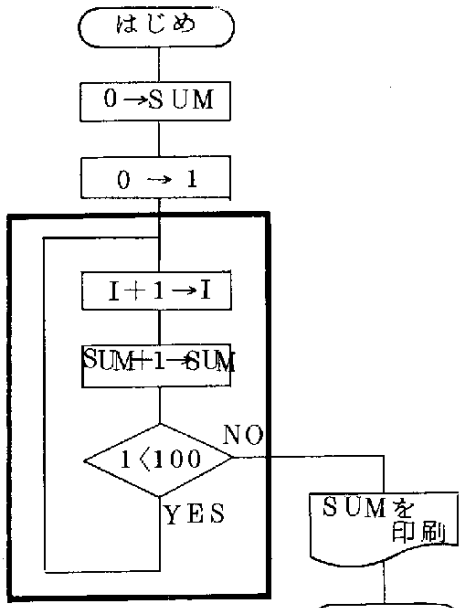
また，例によって次のような例題を使って考えていきます。

例題

1から100までの数を，1＋2＋……＋99＋100というように，すべて加え合わせて，その結果を印刷しなさい。

28－C－31

これは簡単な例ですがフローチャートを書いてみましょう。すべての数の合計はSUMにそれぞれの数はIに入れることにします。



くり返しの部分です。

29－C－32

一組のステートメント群を何回も繰返し実行することをループといいます。

CPLでは，ステートメント群の

最初に　DOステートメントを

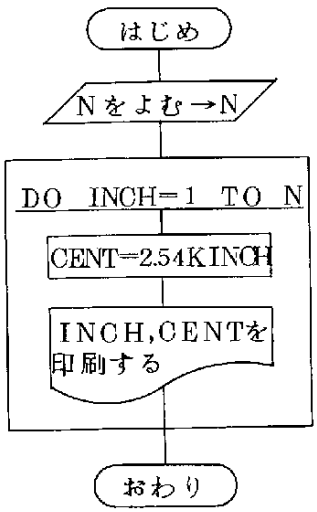
最後に　ENDステートメントを

使ってループを作ります。

30－C－33

では，練習問題を右のフローチャートに従ってコーディングして下さい。
会話の始め方が解らない時はCLASSガイドを見て下さい。ただし，宣言ステートメントは次のように指定して下さい。

```
DECLARE N     DECIMAL FIXED(3)
        INCH  DECIMAL FIXED(3)
        CENT  DECIMAL FIXED
                      (10, 2);
```



コーディングが完成したらYのキーをわからなければNのキーを押して下さい。

55－C－60

それでは，N＝18として実行してみましょう。？に後に，RUNコマンドを入力します。
データ要求が出たらYのキーを出なかったらNのキーを押して下さい。

55－C－61

N＝18として実行しますからDATA 0030. 00 N?
に続けて [STX] 18 [復改] [ETX] と入力しなさい。
そこで結果が出力されたでしょうか出力されたらYのキーを出力されなかったらNのキーを押して下さい。

55－C－62

答が出力されたけど1行しか印刷されなかったならばNのキーを18行印刷されたならばYのキーを押して下さい。

55−C−63

実行した結果，出力された値のうち次の値を答えなさい。

3 インチ ＝ (1) センチメートル

7 インチ ＝ (2) センチメートル

(3) インチ ＝30.48センチメートル

(4) インチ ＝35.56センチメートル

17 インチ ＝ (5) センチメートル

(1), (2), (3), (4), (5)の順に答え，最後にANSキーを押して下さい。

55−E−64

端末の使い方，エラーメッセージ，修正の仕方等がわからないときは，CLASSガイドを参考にしてコーディングしてみましょう。

55.1−T−70

プログラムは，次のようになります。違っていたら直しましょう。
修正の仕方は　ＣＬＡＳＳガイドを見て下さい。

```
10 HENKAN:PROCEDURE OPTIONS (MAIN) ;
20 DECLARE N     DECIMAL FIXED (3) ,
           INCH  DECIMAL FIXED (3) ,
           CENT  DECIMAL FIXED (10, 2) ;
30 GET LIST (N) ;
40 DO INCH=1 TO N;
50 CENT=2.54*INCH;
60 PUT LIST (INCH, CENT) ;
70 END;
80 END HENKAN;
```

55－A－71

あなたのコーディングしたプログラムに

GET LIST (N) ;

というステートメントがありますか？　なければ，DECLAREステートメントの次に入れ，ＡＮＳキーを押して下さい。

ステートメントの挿入を行なう場合やり方がわからないときには，ＣＬＡＳＳガイドを参照して下さい。

55.01－H－72

あなたのコーディングしたプログラムに

PUT LIST (INCH, CENT) ;

というステートメントがありますか？ なければ，ＤＯループの中に入れましょう。 そして，もう一度実行してみましょう。

ステートメントの消去，挿入を行なう場合やり方がわからないときにはＣＬＡＳＳガイドを参照して下さい。

55.02－H－73

ＤＯループ内に，ＰＵＴステートメントがありますか？ なかったら，ＤＯループの中にＰＵＴステートメントを置き，もう一度実行してみましょう。

ステートメントの消去，挿入を行なう場合やり方がわからないときは，ＣＬＡＳＳガイドを参照して下さい。

55.03－H－74

キーの押しまちがえではありませんか。もう一度，今の問題を出しますから，今度はよく問題を読んで答えなさい。

55.2－T－75

それでは，ＤＥＣＬＡＲＥステートメントが指定通りに入力されていますか。次と比べてごらんなさい。

```
DECLARE N     DECIMAL FIXED (3),
        INCH  DECIMAL FIXED (3),
        CENT  DECIMAL FIXED (10, 2) ;
```

あるいは，CENT=2.54*INCH; というステートメントが間違っていないでしょうか。

２つのことに注意して，もう一度あなたのプログラムを見直してごらんなさい。

修正が必要なときは，ＣＬＡＳＳガイドを参照して修正してからＡＮＳキーを押して下さい。

55.03−H−76

これで，このセクションの学習は終ります。　ご苦労様でした。

次のセクションでは複雑なＤＯステートメントを学習しましょう。

56−C−80

(4) 学習リスト

```
※※ STARTKEY 0 OSHINASAI
※※ GAKUSHIYU KAISHI
※※ KYOKAMEI ....CPLS2  ※※ ANS ※※
※※ ANATANO NAMAE ....JIPDEC    ※※ ANS ※※
          (
          )
       ※※ ANS ※※
       ※※ ANS ※※
       ※※ ANS ※※
       ※※ ANS ※※
          (
          )
       ※※ ANS ※※
N      ※※ ANS ※※
       ※※ ANS ※※
N      ※※ ANS ※※
       ※※ ANS ※※
N      ※※ ANS ※※
Y      ※※ ANS ※※
N      ※※ ANS ※※
Y      ※※ ANS ※※
Y      ※※ ANS ※※
7.62#17.89#12#14#42.18    ※※ ANS ※※

※※※ ｺﾀｴ ﾉ ｶｽﾞ ｶﾞ ｵｵｽｷﾞﾏｽ    ﾓｳｲﾁﾄﾞ ﾄﾞｵﾝﾞ ※※※
Y      ※※ ANS ※※

※※※ ｺﾀｴ ﾉ ﾆｭｳﾘｮｸ ﾉ ﾋﾂﾖｵ ﾊ ｱﾘﾏｾﾝ ※※※
       ※※ ANS ※※
Y      ※※ ANS ※※
Y      ※※ ANS ※※
7.63#17.87#12#14#43.18    ※※ ANS ※※
       ※※ ANS ※※
7.62#17.78#12#14#43.18    ※※ ANS ※※
       ※※ ANS ※※
※※ END KEY 0 OSHINASAI
※※ RECORD IN 0023※※ GAKUSHIYU OWARI
```

(6) 学習記録リスト

***** CLASS SYSTEM EXECUTION LIST ***** LEARNING PROGRAM NAME (TEST) STUDENT NAME (JIPDEC) PAGE (1)

STEP-NO	SLIDEN	STEPK	CRCT	WRNG	SORT1	ANS0	ANS1	ANS2	ANS3	ANS4	ANS5	ANS6	ANS7	ANS8	ANS9	SW	CCNT	WCNT	TCNT	KINTIME	TTIME
	SOURCE																				
																ACC0	ACC1	ACC2	ACC3	ACC4	ACC5	ACC6	ACC7	ACC8	ACC9
* 45	45		0	0	1	15	15	15	15	15	15	15	15	15	15	00	0000	0000	0000	0000	0000				
																0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
* 01	02	C	0	0	1	00	00	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	@															0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
* 27	30	C	0	0	1	00	00	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	@															0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
* 02	31	C	0	0	1	00	00	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	@															0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
* 29	32	C	0	0	1	00	00	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	@															0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
* 30	33	C	0	0	1	00	00	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	@															0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
* 55	60	Q	0	1	1	40	20	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	N@															0000	0000	0000	0000	0000	0000	0000	0000	0001	0001
* 55	70	C	0	0	1	00	00	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	@															0000	0000	0000	0000	0000	0000	0000	0000	0001	0001
* 55	60	Q	0	1	1	40	20	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	N@															0000	0000	0000	0000	0000	0000	0000	0000	0002	0002
* 55	71	C	0	0	1	00	00	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	@															0000	0000	0000	0000	0000	0000	0000	0000	0002	0000
* 55	61	Q	0	1	1	40	20	00	00	00	00	00	00	00	00	00	0000	0000	0000	0000	0000				
	N@															0000	0000	0000	0000	0000	0000	0000	0000	0002	0000

***** CLASS SYSTEM EXECUTION LIST ***** LEARNING PROGRAM NAME (TEST) STUDENT NAME (JIPDEC) PAGE (2)

STEP-NO SLIDEN STEPK CRCT WRNG SORT1 ANS0 ANS1 ANS2 ANS3 ANS4 ANS5 ANS6 ANS7 ANS8 ANS9 SW CCNT WCNT TCNT KINTIME TTIME

SOURCE ACC0 ACC1 ACC2 ACC3 ACC4 ACC5 ACC6 ACC7 ACC8 ACC9

* 55 72 0 1 0 1 00 70 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 0000

Y@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0000

* 55 61 0 0 1 1 40 20 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 0000

N@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0000

* 55 72 0 1 0 1 00 70 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 0000

Y@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0000

* 55 61 0 1 0 1 00 70 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 0000

Y@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0000

* 55 62 0 0 1 1 40 20 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 0000

7.52#17.89#12#14#42.18@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0000

* 55 73 C 0 0 1 00 00 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 0000

@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0000

* 55 61 0 1 0 1 00 70 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 0000

Y@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0000

* 55 62 0 1 0 1 00 70 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 0000

Y@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0000

* 55 64 E 0 1 1 00 20 20 70 70 70 00 00 00 00 00 0000 0000 0000 0000 0000

7.63#17.87#12#14#43.18@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0001

* 55 75 C 0 0 1 00 00 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 0000

@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0001

* 55 64 E 1 0 1 00 70 70 70 70 70 00 00 00 00 00 0000 0000 0000 0000 0000

7.62#17.78#12#14#43.18@ 0000 0000 0000 0000 0000 0000 0000 0000 0002 0000

(5) ＴＳＳ端末での学習リスト

```
? USER-ID...S*TAKE
/YOUR JOB-N....SJIA055
MACRO SUN NYUURYOKU
*#OPL
JOB KAISI.
? 10 HENKAN:PROCEDURE OPTIONS(MAIN);
? 20         DECLARE N     DECIMAL FIXED(3),
?                  INCH    DECIMAL FIXED(3),
?                  CENT    DECIMAL FIXED(10,2);
? 30         GET LIST(N);
? 40         DO INCH=1 TO N;
? 50         CENT=2.54*INCH;
? 60         PUT LIST(INCH,CENT);
? 70         END;
? 80         END HENKAN;
 LAST LINE(END ST)   0080.00
? *RUN;
DATA 0030.00    N? 18
         1              2.54
         2              5.08
         3              7.62
         4             10.16
         5             12.70
         6             15.24
         7             17.78
         8             20.32
         9             22.86
        10             25.40
        11             27.94
        12             30.48
        13             33.02
        14             35.56
        15             38.10
        16             40.64
        17             43.18
        18             45.72
? *STOP;
JOB OWARI.
MACRO SUN NYUURYOKU.
*#BYE
//CPU-TIME.  00:00:01.975
ANATA WA DENWA NO SIYOO O YAMETA.
ZIKOKU 14:42 NI OFF-LINE NI NATTA. SIYOOZIKAN 00:09:00
```

第 6 章　システム・オペレーション

6.1　CLASS—N システム作成における操作

ＣＬＡＳＳ－Ｎシステムは制御プログラムと学習プログラムとからなっている。更に制御プログラムはアセンブラ言語で学習プログラムはＢＰＬ言語で作られている，従って両者の作成手順は異なる。制御プログラムはアセンブラでアセンブルされ相対形式プログラムが作られて，学習プログラムはＢＰＬプロセッサーによりコンパイルされ相対形式プログラムが作られる，両者の相対形式プログラムはUPDATEの処理プログラムにより１つの相対形式プログラムに直される。１つになった相対形式プログラムはＬＩＮＫＬＯＡＤ 処理プログラムにより実行形式プログラムが作られる。それらの関係を図示すると図６－１のようになる。

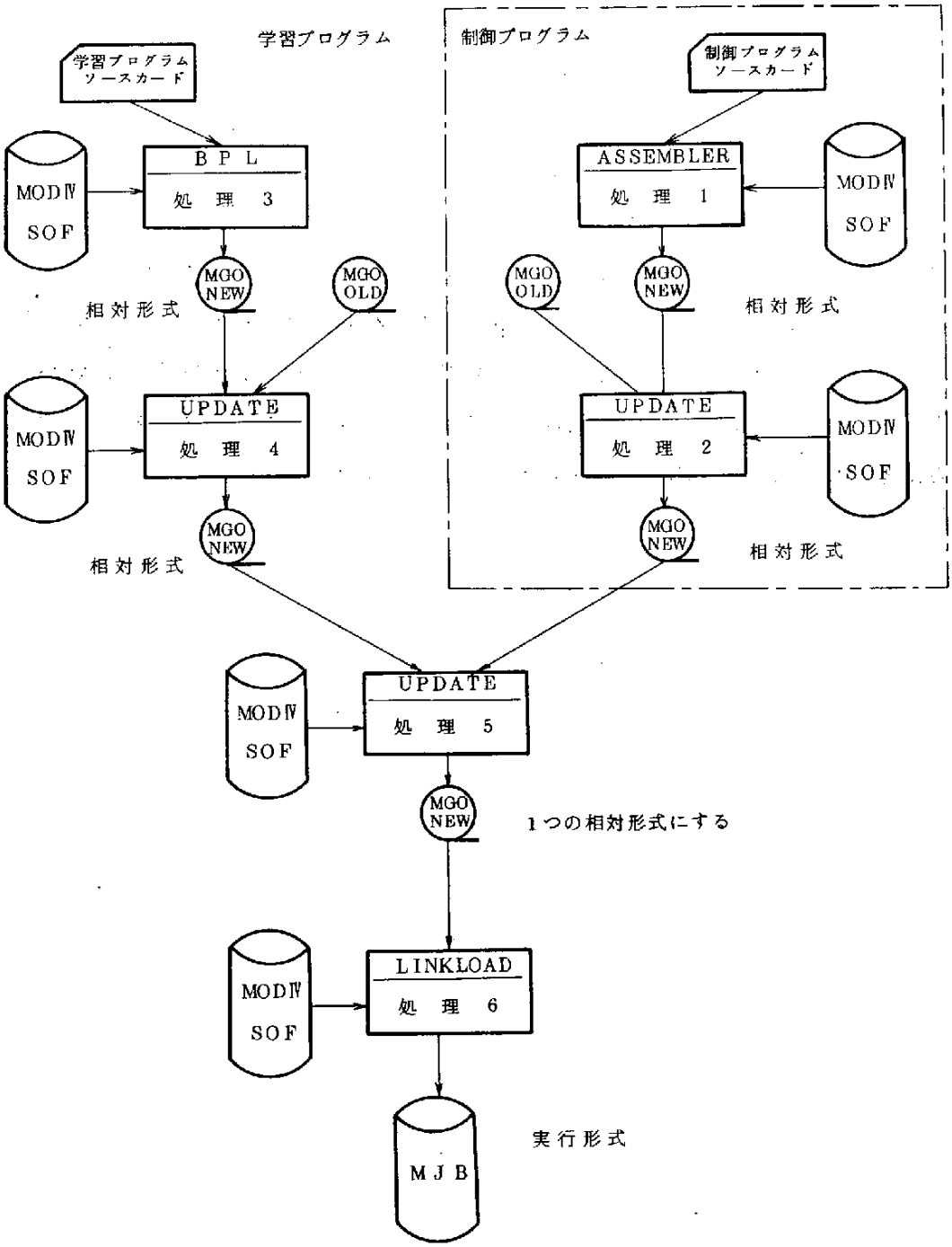

学習プログラム
制御プログラム
学習プログラム
ソースカード
制御プログラム
ソースカード
BPL
処理 3
ASSEMBLER
処理 1
MODⅣ
SOF
MODⅣ
SOF
MODⅣ
SOF
MGO
NEW
MGO
OLD
MGO
OLD
MGO
NEW
相対形式
相対形式
UPDATE
処理 4
UPDATE
処理 2
MODⅣ
SOF
MODⅣ
SOF
MGO
NEW
MGO
NEW
相対形式
相対形式
UPDATE
処理 5
MODⅣ
SOF
MGO
NEW
1つの相対形式にする
LINKLOAD
処理 6
MODⅣ
SOF
MJB
実行形式

図　6 － 1

システム作成における操作手順は次のようになっている。

(1) 制御プログラムの作成

制御プログラムのソースカードは処理1でアセンブルされ相対形式のＭＧＯファイルが作られる。もしそれがupdateならば処理2で古い相対形式のＭＧＯファイルを処理1で作った新しいＭＧＯファイルで修正することにより新しい相対形式のＭＧＯファイルが出来る。

(2) 学習プログラムの作成

学習プログラムのソースカードは処理3でＢＰＬコンパイラによりコンパイルされ相対形式のＭＧＯファイルが作られる。もしそれが学習プログラムの追加やupdateならば処理4で古い相対形式のＭＧＯファイルを処理3で作った新しいＭＧＯファイルで修正することにより新しい相対形式のＭＧＯファイルが出来る。

(3) 実行プログラムの作成

実行形式プログラムを作成ためには処理5で制御プログラムの相対形式と，学習プログラムの相対形式ファイルを1つの相対形式のＭＧＯファイルにする必要がある。そしてそのＭＧＯファイルを処理6でリンクロードすることにより実行形式プログラムファイルがディスク上に作られる。

6.2 CLASS—Nシステム実行における操作

ＣＬＡＳＳ－Ｎシステムは学習者の学習申し込みによりＣＬＡＳＳ－Ｎシステムで使用するファイル（プログラム・ファイルとデータ・ファイル）をディスク装置にセットし，各学習端末装置をオンラインにする，それからカードリーダにこのシステムを起動するためのコントロールカードをセットし読ませることにより実行は開始される。実行における各装置の構成を図６－２のようになっている。

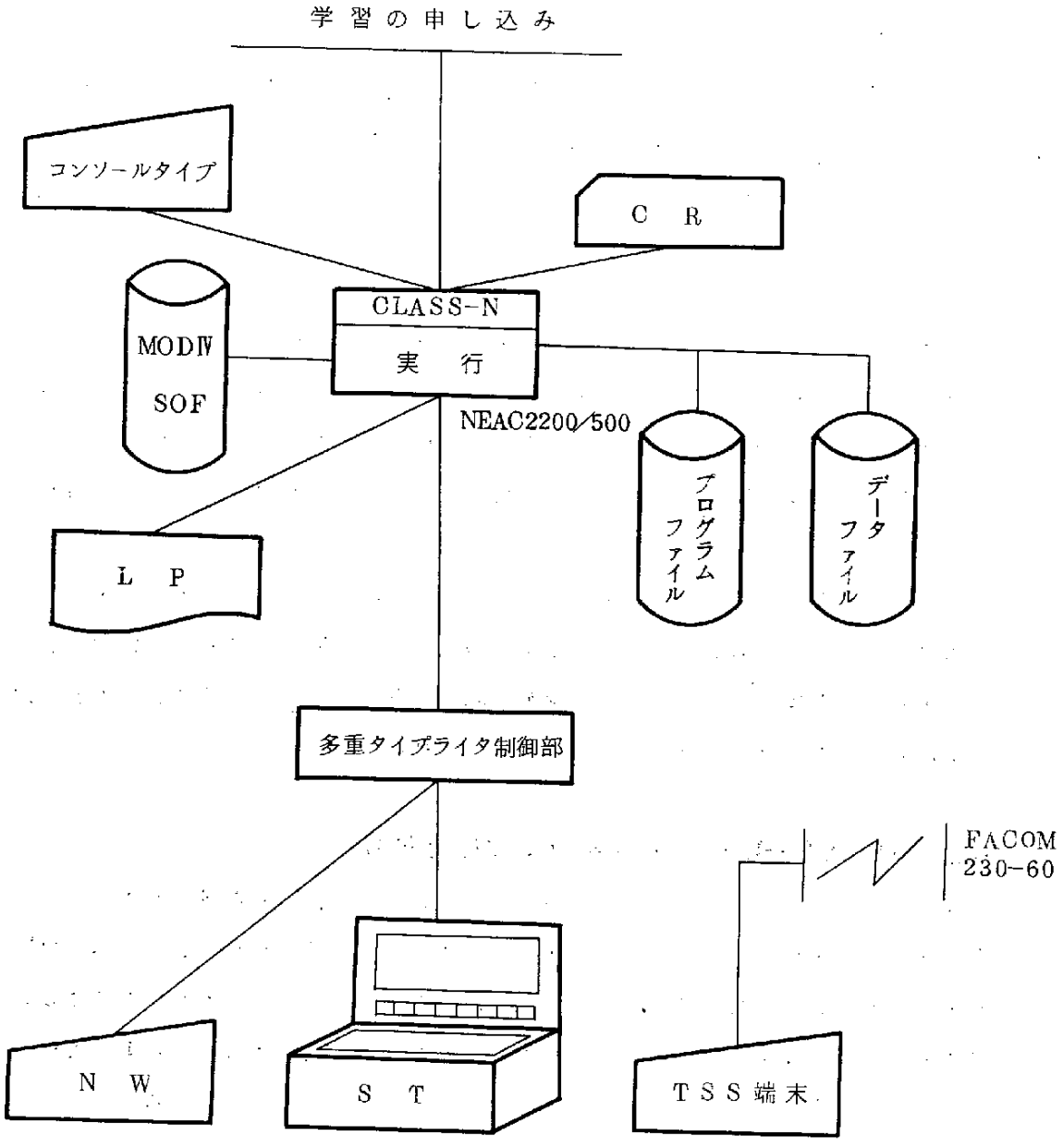

学習の申し込み
コンソールタイプ
C R
CLASS-N
実 行
MODⅣ
SOF
NEAC2200/500
プログラムファイル
データファイル
L P
多重タイプライタ制御部
FACOM
230-60
N W
S T
TSS端末

図 6 － 2

このシステムの学習における操作手順は次のようになっている。

まず学習者はこのシステムで学習したい旨，計算機室のオペレータに申し込む。オペレータはＣＬＡＳＳ－Ｎのファイルをセットし，数枚のコントロールカードを読ませればシステムは動作状態となる。次に学習者は希望する教科のスライドトーレをセットして学習端末の電源を入れれば学習開始の状態になり，スライド及びタイプライタに〝ＳＴＡＲＴキーを押しなさい〟という指示がでるのでＳＴＡＲＡキーを押す。　次に〝教科名をどうぞ〟とタイプライタに印字されるので，希望する教科名をキーセットから入力しＡＮＳキーを押せばよい。それから次に〝あなたの名前をどうぞ〟と印字してくるので，自分の名前を入力しＡＮＳキーを押せばよい。

後は学習プログラムによって指示される応答を行なうことにより学習は進められる。学習中は質問以外であればＡＮＳキーの応答をまた質問の応答であれば回答を入力しＡＮＳキーを押せばよい。またその回答入力中，間違いに気がつき修正したいならば１字修正はＤＥＬキー（delete）を全部の修正はＣＡＮキー(Canncel)を押し正しく入力されるまで繰返せばよい。

学習がセクションの終りまでいくと〝ＥＮＤキーを押しなさい〟と印字されるのでＥＮＤキーを押すことにより学習が終了する。また続けて次のセクションの学習を行うときには次に自分の学習したいセクションのスライドトーレをセットしなおして最初のＳＴＡＲＴキーを押すところから始めればよい。

学習は原則としてセクション単位になっているがどうしてもそのセクションの途中で打切りたいときにはその時点でＥＮＤキーを押せば学習は終る。

なお１つの問題で１個以上の回答を要求している時にはその区切り記号として'#'記号を使用するようになっている。

第７章　今後の課題

今後の課題として次のようなことを予定している。

① 今年度作成したＣＰＬ学習プログラムの完成及びその評価

② ＣＬＡＳＳ－ＮとＣＬＡＳＳ－Ｆでの学習端末による学習効果の比較

③ 今回はガイダンス機能としてＣＬＡＳＳガイドを作成した。これはプログラムテキスト形式に作られているが，これもＣＡＩ的にコンピュータを利用してその機能が生かせることが望ましい。しかもガイダンス要求は自然言語による入出力ができればその効果は大きい。そうしたことについても今後研究していく予定である。

第 2 部　CLASS－F

目　　次

第1章　システム概要

1.1　概　　要

現在開発されているプログラミング教育用のＣＡＩ（Computor Assisted Instrction）システムは，プログラム学習法にもとずいて作成され，プログラムの個々のステートメントのみを対象にして，対話的に学習させているものが多い。

電子計算機のプログラミング教育においては，この種のＣＡＩシステムでは不十分で，学習者自身が，自分でプログラムを組み，直接計算機にかけ，その結果に対して，適切な診断（エラー・メッセージ）や治療が行なわれることが望ましい。

CLASS－FシステムはFACOM230－60タイム・シェアリング・システムの元で使用される会話型言語ＣＰＬ（Conversational Programming Language）の学習をおこなうシステムである。そしてまた学習自身もＴＳＳ端末を使用して会話型式でおこなうことができる。すなわち会話型言語ＣＰＬによるプログラムの作成，実行あるいは修正の各段階に於て，充分なガイダンス機能を与えると共に，シンタックスの誤り指摘は当然のこと，ロジカルな誤りに対しても診断をおこない学習者のディバッグ技術も併せて習得させる。

一般に，学習者の作成したプログラムは，バラエティに富み，学習者の犯す誤りも千差万別である。したがって，考えられるすべての誤りに対して適切な診断や治療を与えることは，きわめて難かしい。

しかし一面ある問題に対して，学習者に自由にプログラムを作成させ，分析して統計とってみると，アルゴリズムのパターンはある程度限られており，そのアルゴリズムの誤りやすい点がわかる。

そこで，提示された問題に対して，学習者に自由にプログラミングさせ，そのプログラムの文法エラー（シンタックス・エラー）は即座に指摘し，すぐ訂正させるとともに，実際のデータを使ってそのプログラムを実行させ，実行結果を診断することを行っている。この場合最終結果だけでなく中間結果もシス

テム側でチェックし，どのあたりに誤りがあるかの見当をつけ，学習者との会話によって誤りを発見させるという方法を採用した。

プログラム診断の学習アルゴリズムは次のようになる。

(1) まず，学習者が演習すべき問題とその問題に対する標準プログラム(Sample Program）と標準データ（チェック・データ）をファイルから読み込む。

(2) 標準プログラムをコンパイルし，実行させ，その実行結果を保存する。

(3) 問題をディスプレイ装置に表示する。

(4) 学習者は，その問題に対して，ディスプレイ装置を経由して会話型言語CPL（Conversational Programming Language）で会話しながらプログラムを作成する。

入力されたステートメントに文法エラーがあるとすぐさま指摘され，直ちに修正することができる。途中でプログラム上の疑問が生じた場合は，随時，計算機に質問をすることができる。

(5) 学習者のプログラムが完成すると，それを標準データを使って実行させる。その実行結果とすでに作成された標準プログラムの実行結果とくらべ，評価し，適切な診断メッセージをディスプレイ装置に表示する。

(6) 正しければ，ここで中断するか，又は次の問題に移るために(1)に戻る。誤りならば，(4)からやり直す。何回も誤りをくり返している場合は，適切なメッセージやヒントを与え，場合によっては，標準プログラムを表示し，学習を打切ることもある。

本システムは，FACOM230−60モニターVの下で使用される。

モニターVは，ローカル・バッチ処理，リモート・バッチ処理，会話型処理の3つの処理形態が同時に実行される3次元のオペレーティング・システムであり，本システムは，そのTSS管理の下で働く会話型処理プログラムとして作成されている。

1.2 システムの機器構成

本システムの機器構成を図1.1に示す。

計算機本体は富士通の大型計算機FACOM230−60で，主記憶装置の容量は，高速磁心記憶装置が１６２K語（１語３６ビット），大容量記憶装置が２５６K語である。補助記憶装置としては，集団ディスク・パック装置，磁気ドラム装置及び磁気テープ装置がある。又通信制御装置（２台）を経由してＴＳＳ用端末装置（１０台）が連結され，常時ＴＳＳサービスを行なっている。CLASS−Ｆ用の学習端末であるタイプライター装置及びキャラクタ・ディスプレイ装置もまた通信制御装置を経由して連結されている。

CLASS−Ｆのシステムや学習経過記録は原則として集団ディスクパック装置に保存される。

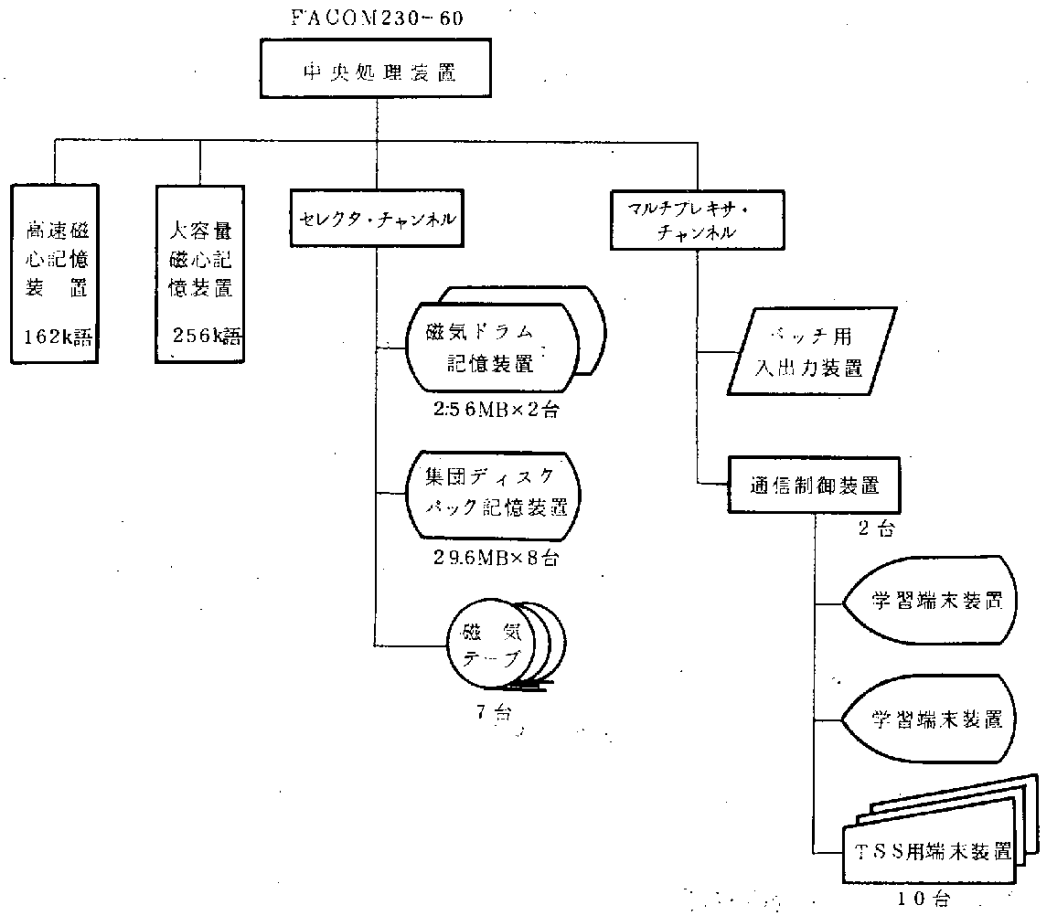


図 1.1　システムの機器構成

第2章　システム構成

2.1 概　　要

システムは，大きく次のような部分に分けることができる。

(1) 全体を管理する制御プログラム……モニター

モニターは，システムの初期設定を行ない，学習者の入力したシステム制御コマンドに従い，各々のサブ・システムを主記憶装置にローデングし，制御を渡し，各サブ・システムの状態（モード）に遷移する。

(2) ＣＰＬ言語で書かれたプログラムを会話的に実行するプロセッサ……CPL

ＣＰＬは，学習者が会話しながら作成したプログラムをインタプリートしながら実行し，実行結果を保存したり，すでにシステムで準備されているその問題に対する標準プログラムをコンパイル，実行し，その実行結果を保存する。

(3) 問題に対する標準プログラム……サンプル・プログラム

サンプル・プログラムは，ＣＰＬ言語で書かれ，問題に対する標準的なアルゴリズムによるコーデングの他に，学習者の誤り易い部分のコーデングも含まれている。プログラムはＣＰＬプロセッサによってコンパイルし，実行される。

(4) 学習者の作成したＣＰＬプログラムを診断するプログラム及びその言語……診断プログラムとＤＩＡＬ

診断プログラムは，学習者が作成したＣＰＬプログラムとサンプル・プログラムの各々の実行結果を診断する手順が詳細にＤＩＡＬ言語で書かれている。

(5) 診断プログラムを実行させるインタプリタ……ＣＡＬＬ

ＣＡＬＬは，コンパイルされた診断プログラムを主記憶にローデングし，診断プログラム，学習者が作成したＣＰＬプログラムとサンプル・プログラムの３者を結合し，診断プログラムをインタプリートとしながら実行させ，プログラム診断を行なう。

(6) 学習者と計算機とが会話するに必要な学習端末……キャラクタ・ディスプ

レイ装置と補助的なタイプライター装置

本システムの主要な学習端末は，キャラクタ・ディスプレイ装置である。キャラクタ・ディスプレイ装置はタイプライター装置とくらべて，騒音もなく，表示速度も非常に早く，特に入力の修正や変更には偉力を発揮する。しかし，文字がブラウン管上に表示されるので，ハード・コピー（印刷物）がとれない。この欠点を補うため，本システムでは学習者が要求すれば，補助のタイプライター装置にプログラムのリストをタイプ・アウトするなどができる。

(7) 学習端末を管理するプログラム……SCROLL

学習者がキャラクタ・ディスプレイ装置を使用してプログラムを作成している場合，キャラクタ・ディスプレイ装置の画面は５０字×２０行しかないので，作成されたプログラムを全部一度に表示することはできない。そこで，学習者の希望するプログラムの任意の部分を自由に画面に表示する機能を有する画面の制御を行なうプログラムがSCROLLである。

(8) その他にプログラムの作成とか，学習経過の整理のユーティリティ・プログラムとしてLIBEがある。

CPLはConversational Programming Langnage，DIALはDIAgnostic Language，CALLはCApable Linkage Loader，LIBEはLIBrary Editorの略である。

このうち，CPLは，このシステムの学習対象であると同時に学習をaidするシステムでもある。

図2.1に本システムの構成図を示す。

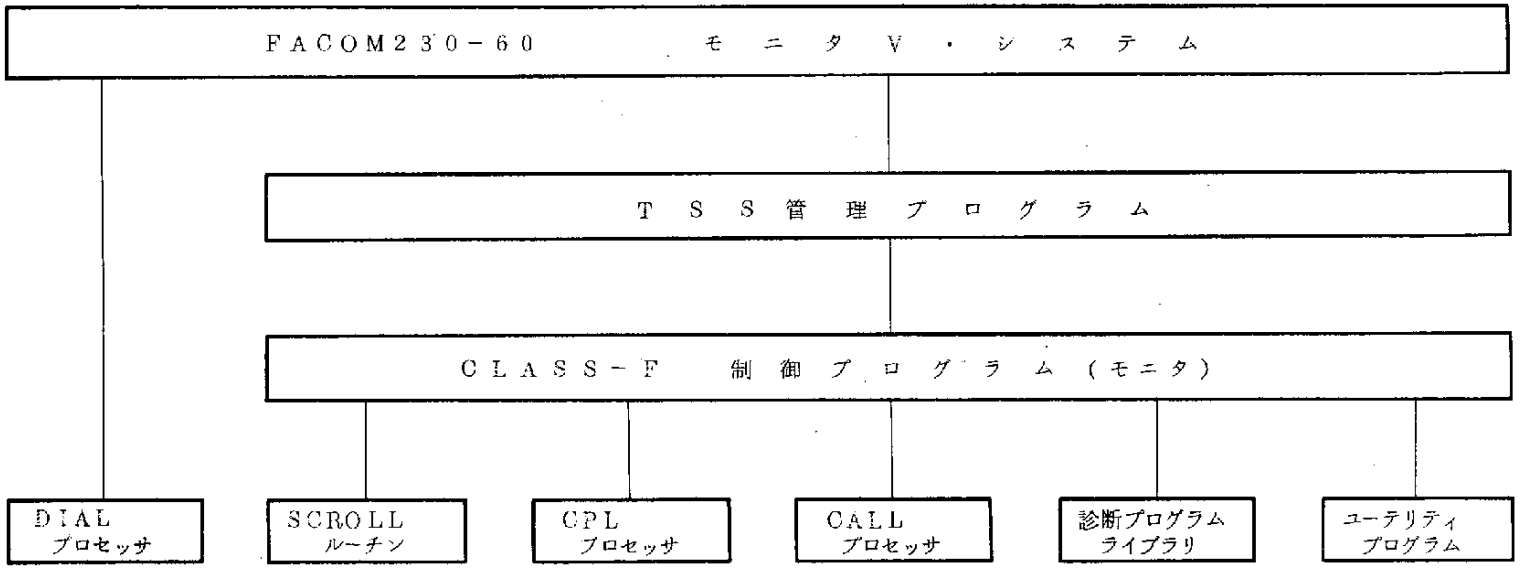


図 2.1　CLASS－F システム構成図

2.2 モニター

2.2.1 モニターの概要

モニターは，システム全体の管理を行なう。

モニターは，次の機能を有する。

(1) システムの初期設定

学習者と会話し，氏名，学習者に与えられたコード名及び問題名を問合わせる。

もし，学習者が昨日まで行っていた問題を途中からつづけてプログラムの作成を行うことを指定したならば，すでに学習経過記録として保存されているプログラムを主記憶にローデングする。

(2) 処理モードの遷移

学習者が入力したシステム制御コマンドに従って，指定された処理プログラムをローデングし，制御を処理プログラムに渡し，処理モードを遷移する。

(3) システムの終了処理

学習者とのすべての学習を終ったならば，学習経過記録を整理し，大記憶装置に書きだす。学習時に使用したすべてのファイルをクローズし，資材をFACOM230−60モニターVに返却し，処理を終了する。

2.2.2 処理モードとシステム制御コマンド

処理モードには次の8種類あり，それぞれ入力できるシステム制御コマンドが異なる。

① ＯＦＦモード
② 初期モード
③ 学習モード
④ サンプル・モード
⑤ 問答モード
⑥ 診断モード
⑦ 治療モード

⑧ 終了モード

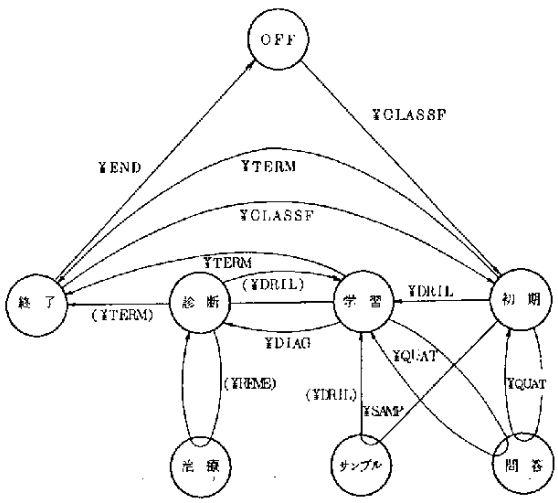
システム制御コマンドは，処理モードの遷移を指令する。システム制御コマンドとしては次の９種類がある。

① ¥CLASSFシステム制御コマンド
② ¥DRIL　システム制御コマンド
③ ¥SAMP　システム制御コマンド
④ ¥QUAT　システム制御コマンド
⑤ ¥DIAG　システム制御コマンド
⑥ ¥REME　システム制御コマンド
⑦ ¥TERM　システム制御コマンド
⑧ ¥END　システム制御コマンド

この内，①はFACOM230−60モニターVに対するシステム制御コマンドでもあり，⑦はシステム内部で発生するシステム制御コマンドで，学習者が入力するものではない。

これらのモードおよびコマンドの関連につき簡単に説明する。

図2.2は，これらの関連を図示したものである。



(　)はシステム内部で発生

図2.2　システムの処理モードとシステム制御コマンドの関係（状態遷移図）

(1) ＯＦＦモード

ＯＦＦモードとは，システムが主記憶装置に存在しない状態のことである。この状態でシステム制御コマンド¥CLASSF を入力すると FACOM 230−60 モニターVによってCLASS−F システムが主記憶装置にローデングされ，初期モードに遷移する。

終了モードにおいて¥ENDシステム制御コマンドが入力されると，この状態に遷移する。

(2) 初期モード

初期モードとは，FACOM230−60のモニターVによってCLASS−Fシステムが主記憶装置にローデングされ，システムの初期化を行っている状態である。すべての初期化を終ると次の状態に遷移するため，システム制御コマンドの入力を要求する。ここで例えば，ＣＰＬのＤＯステートメントについて問答（質問）をしたければ，¥QUATシステム制御コマンドを入力する。このコマンドが入力されると状態は問答モードに遷移するが，問答が終了すると自動的に初期モードに遷移する。

新しい問題に対してプログラムを作成する場合は¥SAMPシステム制御コマンドを入力する。サンプル・モードでサンプル・プログラムが実行され，実行結果が保存され，自動的に学習モードに遷移し，学習モードに入る。

過去におこなったもののつづきとしてプログラムを作成する場合は，¥DRILシステム制御コマンドを入力すると，サンプル・モードの状態を経由せず，直接学習モードに遷移する。

このモード状態でシステムとの会話を終了したい場合は，¥TERMシステム制御コマンドを入力する。

(3) 学習モード

学習モードとは，提示された問題に対して学習者がディスプレイ装置を経由してＣＰＬプロセッサと会話しながらＣＰＬプログラムを作成している状態である。

学習者はプログラムを作成中，随時，ＣＰＬの文法についてシステムに問い合わせることができる。この場合は，システム制御コマンド¥QUATを入力する。

学習者は，ＣＰＬのプログラムの作成が終わると，システムに診断してもらうため，システム制御コマンド¥DIAGを入力する。このコマンドが入力されると，診断モードへ状態は遷移する。

都合によりプログラムの作成を中断したい場合は，¥TERMシステム制御コマンドを入力する。

(4) サンプル・モード

サンプル・モードとは，提示された問題に対するサンプル・プログラムをローデングし，ＣＰＬプロセッサでコンパイル，実行し，実行結果を主記憶装置の一部に保存する処理過程である。

(5) 問答モード

問答モードとは，学習の時点で学習者がＣＰＬの文法についての質問が生じた場合に，それに対して会話型で応答する処理過程である。

(6) 診断モード

診断モードとは，学習者の作成したＣＰＬプログラムの実行結果とサンプル・プログラムの実行結果を診断プログラムの手順に従って診断している状態である。診断結果が重傷であると，治療モードに自動的に遷移し，治療を行なう。軽傷の場合は，プログラムを修正させるため自動的に学習モードへ状態を遷移させる。正しい場合は自動的に終了モードに遷移する。

(7) 治療モード

治療モードとは，システムで準備されているヒントを表示したり，やゝ簡単な問題に戻って学習を経過させている状態である。

本モードが終了すると自動的に診断モードに遷移する。

(8) 終了モード

終了モードとは，学習者が指示された問題に対する正しいＣＰＬプログラムを作成したので，学習者と会話して，次の問題をやるかどうかを問答

させたり，システムの後始末をやっている状態である。

学習者は次の問題をやりたければ，¥CLASSF システム制御コマンド，ここで完全に終了したければ¥ENDシステム制御コマンドをそれぞれ入力する。

図2.3に本システムのコアー・マップを示す。

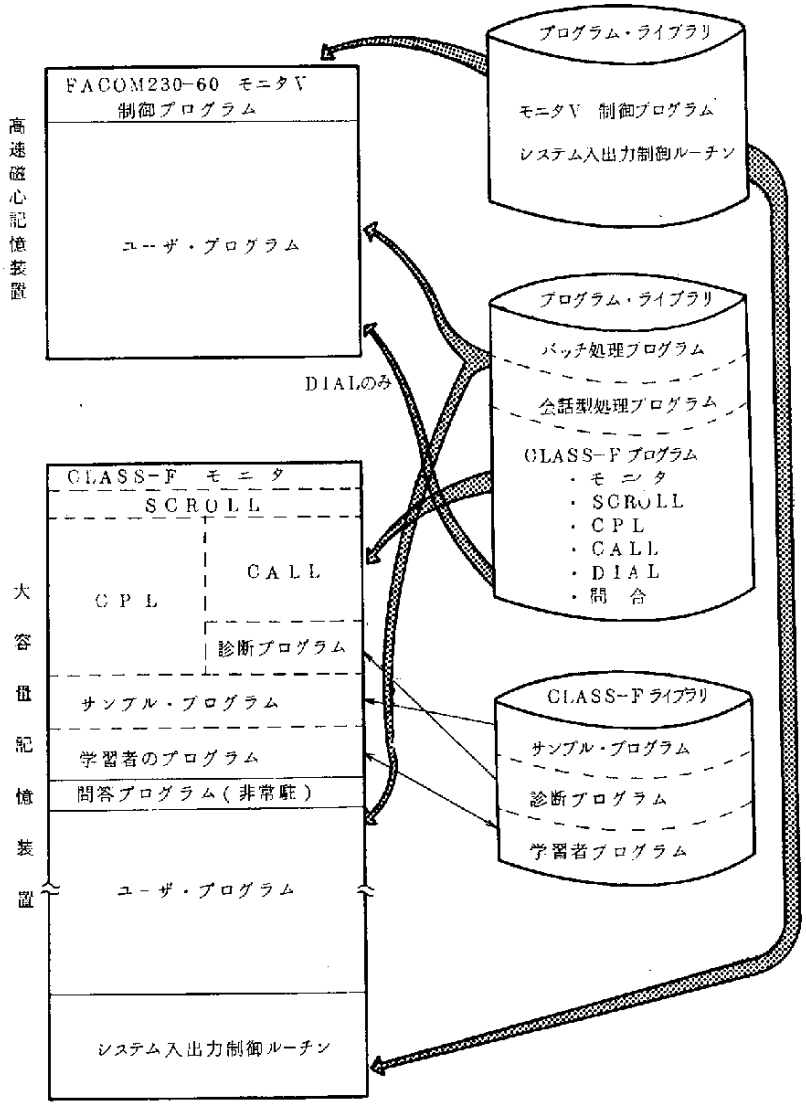


図 2.3 CLASS—Fシステムのコマー・マップ

第3章　C　P　L

3.1　会話型言語とプロセッサ

CLASSは会話型言語ＣＰＬによるプログラミングの学習を目的としている。

ＣＰＬは会話型のＰＬ／Ｉである。PL／Iは技術計算，事務計算両用の汎用言語であると同時にリスト処理，ビット処理等の機能を持ち，システム・プログラムの記述も可能と言われている。

ＣＰＬは会話型のＰＬ／Ｉであり，タイム・シェアリング・オペレーティング・システムの元で稼働するため，本来のＰＬ／Ｉの持つ，メモリー・アロケーションの多種性や，マルチタスクの機能等には若干の制限を加えねばならなかったが，その他の機能は出来るだけ落さずにサブセットを設計した。

人間とコンピュータがあたかも会話をしている如く相互に密接な連絡をとりながら，協力してより効率よく仕事を進める道具として，最近は会話型言語が開発され，同時にそのためのプロセッサが開発されている。

一般に会話型プロセッサは次のような点が特徴とされている。

① プログラムの作成，実行，変更が随時交互に切換えられること。

② プログラムのエラーが即時に検出できること。

③ プログラムの修正および編集が任意に行なえること。

④ 任意のステートメントの実行や，プログラムの部分実行が自由に行なえること。

⑤ 入力すると直ちに実行できるダイレクト・ステートメントが使用できること。

⑥ プログラムを自動的に保存することができ，いつでも呼び出して使用できること。

⑦ 人間と機械とのインタラクションが会話的に行なわれ，ガイダンスの要素があること。

3.2 ＣＰＬの概要

ＣＰＬは，これらの機能をできるだけ満たすように設計されたＰＬ／Ｉの会話型プロセッサである。

次のような特長を有する。

① プログラムが完成しないでも入力されているプログラムがいつでも実行できなければならないので逐次翻訳を行なう。

② 会話型の本質を生かすには，一般にはインタプリタ方式をとる必要があるといわれているが，完全なインタプリタ方式ではあまりにも実行時間がかかりすぎるので，インタプリタ方式とコンパイル方式の中間をとり，できるだけ機械語のオブジェクトを出すようにした。

③ ソース・プログラムのリストの要求に応じられるようにソース・イメージもそのまま保存しておく。

④ 文法的なエラーはできる限り即時に検出し，わかりやすい適切なメッセージを出す。

⑤ セーブしてあるプログラムをCALLステートメントや関数として呼び出せる。

⑥ ダイレクト・ステートメントを導入したこと。

⑦ インプットされたプログラムの大きさ（ソース・プログラム）によって，オブジェクト・エリアや記号表エリアを実行時に伸長できること。またオブジェクトの実行時におけるストレイジは，ブロック構造やAUTOMATICストレイジ・クラスを許すので効力に使用することができる。

⑧ ステートメントが挿入あるいは削除された場合，原則としてプロセッサはリコンパイル（recompile）をするが，端末利用者が意識的にリコンパイルをさせなくすることもできるし，任意の時点で強制的にリコンパイルさせることもできる。

⑨ システムはリエントラントに作られている。

⑩ FACOM230－60バッチ用ＰＬ／Ｉと言語仕様上の互換性を持たせてある。

ＣＰＬにおけるステートメントは，コレクト（collect）ステートメントとダイレクト（direct）ステートメントとに分けられる。コレクト・ステートメントは通常のプログラムで，プロセッサによって，オブジェクトが作成され，ソース・ステートメントと共に保存される。ダイレクト・ステートメントは先頭に％マークを付したもので入力されると直ちに実行され，ソース・ステートメントとオブジェクト・ステートメントは共に保存されない。なお，ある種のステートメント（代入，GET，PUT，STOP 等）の先頭に％マークをつけることによってダイレクト・ステートメントとして使用することもできる。これにより，初期値の設定，アーギュメントの値の変更，中間結果の印刷等を任意に行なうことができ，デバックの際有効な手段となる。

3.3 コレクト・ステートメント

コレクト・ステートメントはＰＬ／Ｉ言語の規則に準拠しているので説明を省略する。

3.4 ダイレクト・ステートメント

ダイレクト・ステートメントとしてしか使えないステートメントについて説明する。

(1) ％ＲＵＮステートメント

∘機能：

指定された範囲のステートメントを実行する。

∘一般形：

％ＲＵＮ〔ライン番号１〔，ライン番号２〕〕；

∘パラメータ：

ライン番号１：実行を開始するステートメントのライン番号を指定する。

ライン番号２：実行を終了するステートメントのライン番号を指定する。

① ライン番号が全く指定されなかった場合は，

external block の procedure ステートメントから実行

を開始する。

② ライン番号1だけが指定された場合は，そのライン番号の付いたステートメントのみを実行する。

◦使用例：

%ＲＵＮ　１０,２００；

%ＲＵＮ；

%ＲＵＮ　５０；

(2) %LISTステートメント

◦機能：

指定された範囲のソース・プログラムをタイプライター上に印刷する。

◦形式：

%ＬＩＳＴ〔ライン番号1〔，ライン番号〕〕;

◦パラメータ：

ライン番号1：印刷をすべき最初のステートメントのライン番号を指定する。

ライン番号2：印刷をすべき最後のステートメントのライン番号を指定する。

① ライン番号が全く指定されなかった場合は，全部のリストをとる。

② ライン番号1のみが指定された場合は，ライン番号1を持つステートメントのみリストをとる。

◦使用例：

%ＬＩＳＴ；

%ＬＩＳＴ　１０,２００；

%ＬＩＳＴ　５０；

(3) %SAVEステートメント

◦機能：

現在入力されているプログラムを大記憶にセーブする。

プログラム名はexternal blockのprocedure nameが採用される。

◦形式：

%ＳＡＶＥ；

◦パラメータ；

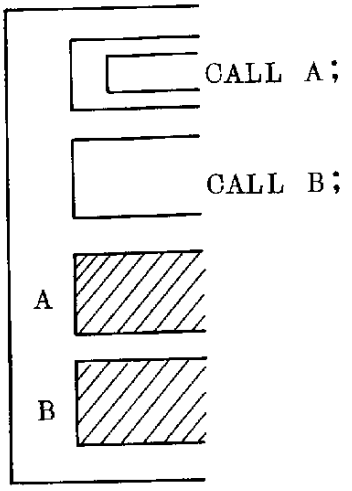
なし。

◦使用上の注意事項：

① 現在セーブしょうとするプログラム名がすでにセーブされているとエラーとなる。この場合プログラム名を変更するか%PURGEステートメントによってセーブされているものを消去する必要がある。

② 一旦セーブしたものは，後日ＣＡＬＬステートメントやFunction Call あるいは%ＬＯＡＤステートメントによって再使用することができる。

③ %ＬＯＡＤステートメントで呼び出された場合，そのProcedureはexternal procedure となり，ＣＡＬＬまたはFunction Callについて呼び出された場合はinternal block となる（図3.1参照）。



A，Bはセーブされていたもの。

図3.1　セーブされたプログラムの組込み例

(4) %LOADステートメント

◦機能：

セーブされているプログラムを主記憶装置にロードする。

◦形式：

%LOAD　プログラム名；

◦使用上の注意事項：

① ソース・プログラムの入力後，%LOADステートメントを入力するとそれ迄に入力されたプログラムは全て消去され，指定されたプログラムがロードされる。それゆえ，ロードされたプログラムはexternal blockを形成する。

◦使用例：

%LOAD　EXAMPLE；

(5) %ERASE ステートメント

◦機能：

指定したステートメントを削除する。

◦形式：

%ERASE　ライン番号1〔,ライン番号2〕；

◦パラメータ：

ライン番号1：削除すべき最初のステートメントのライン番号を指定する。

ライン番号2：削除すべき最後のステートメントのライン番号を指定する。

① ライン番号1だけ指定された場合はライン番号1を持つステートメントのみが削除される。

(6) %COM ステートメント

◦機能：

任意の時点で入力されたプログラムをリコンパイルする。

◦形式：

％ＣＯＭ；

◦使用上の注意事項：

① ステートメントの追加，置換，削除等が行なわれた場合，通常にリコンパイルされるが，％NORECOMステートメントが入力されていればこれらの場合であってもリコンパイルされない。この様な場合，再びコンパイルさせるのにこの％ＣＯＭステートメントが使用される。

(7) **％NORECOM ステートメント**

◦機能：

ステートメントの追加，置換等があった場合，コンパイラは通常リコンパイルをするが，それをさせないことをコンパイラに指示する。

◦形式：

％NORECOM；

◦使用上の注意事項：

① このステートメントの効力は，％ＣＯＭステートメントとによって強制的にリコンパイルされる迄つづくが，それゆえステートメントの追加等においてリコンパイルの必要があるかどうかの判断は利用者の責任となる。

(8) **％RESEQステートメント**

◦機能：

ライン番号のシーケンスの付け直しをする。

◦形式：

％RESEQ；

◦使用上の注意事項：

① 10.00から始まって１０きざみでシーケンスのつけ直しをする。

(9) **％NAMELIST ステートメント**

◦機能：

大記憶にセーブしてあるプログラム名の一覧表を印刷する。

◦形式：

%NAMELIST ;

(10) %NOMSS ステートメント

◦機能：

エラー・メッセージを印刷する際メッセージを全部印刷せず，エラー番号のみを印刷するよう指示する。

◦形式：

%NOMSS;

◦使用上の注意事項：

① このステートメントの効力は%MSSが指定されるまで続く。

(11) %PURGE ステートメント

◦機能：

大記憶にセーブしてあるプログラムの中から指定されたプログラムを消去する。

◦形式：

%PURGE プログラム名;

◦使用例：

%PURGE PROG1;

(12) %MSSステートメント

◦機能：

%NOMSSステートメントの効力を無効にする。

◦形式：

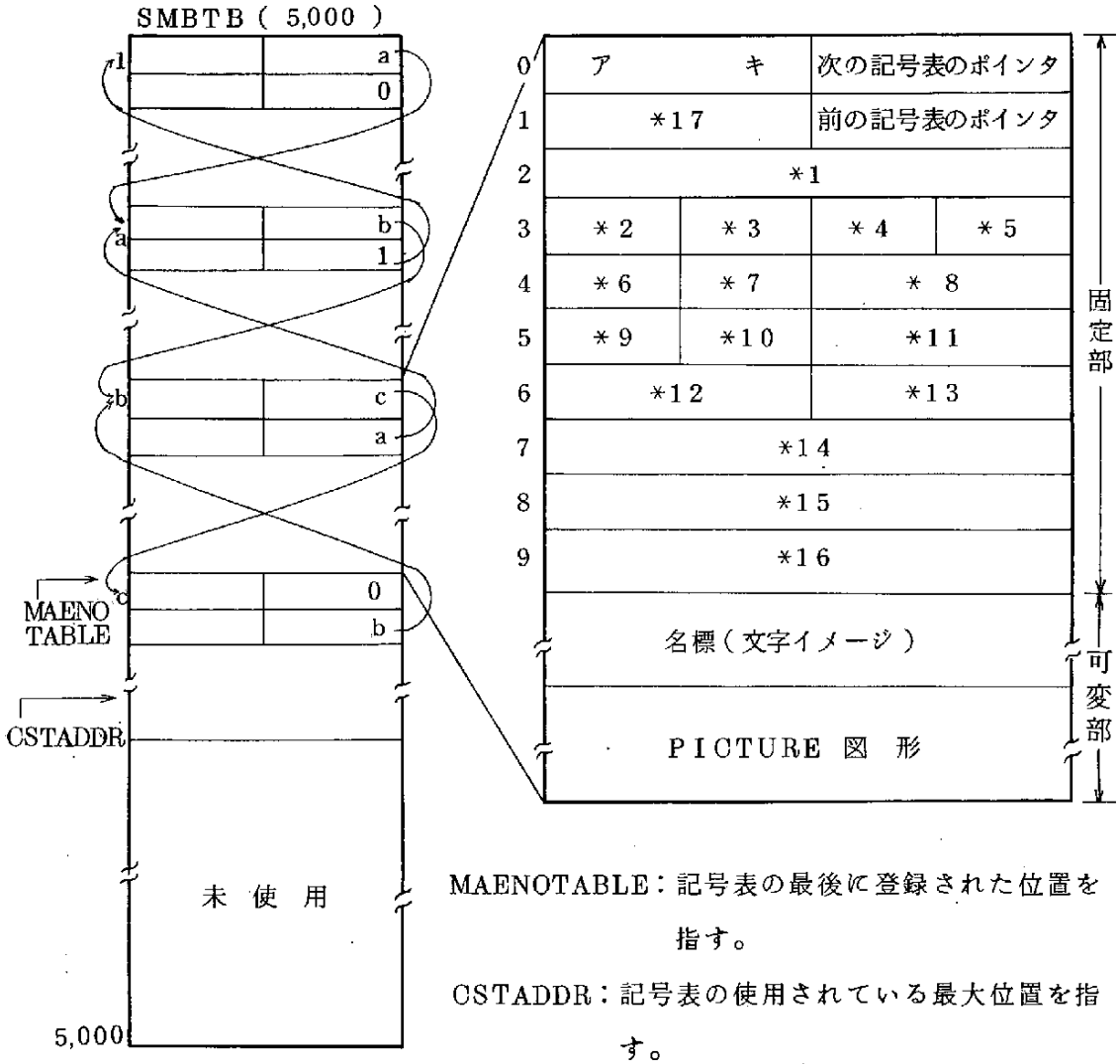
%MSS;

3.5 記号表

CPLのもとで実行されたサンプル・プログラムと学習者のプログラムの実行結果は記号表に保存されている。この2つの記号表を比較し，学習者のプログラムを診断する。

記号表の比較の手続は，診断プログラムに書かれている。

記号表の内容は次の通りである。



MAENOTABLE：記号表の最後に登録された位置を指す。

CSTADDR：記号表の使用されている最大位置を指す。

図3.2　ＣＰＬの記号表

＊１：名標の属性情報

ビット位置	オ　　ン	オ　　フ
0	DOステートメントにつけられたレーベル	
1	配　列	
2	構造体	単純変数
3	主構造体	
4	従構造体	
5	構造体要素	
6	PICTURE	
7	算術データ	
8	ストリング・データ	
9	FIXED	FLOAT
10	BINARY	DECIMAL
11	ビット・ストリング	文字ストリング
12	編集文字あり	
13	ＥＮＤステートメントでＤＯとPROCのラベルと対応した	
14	ア　キ	
15	構造体の最後である	
16	パラメータ	
17	ア　キ	
18	EXTERNAL	INTERNAL
19	確定，定義（変数又はレーベル）	
20	CONTROLLED	
21	STATIC	AUTOMATIC
22	BASED	
23	POINTER	
24	ENTRY 名	

ビット位置	オ　　ン	オ　　フ
25	FILE名	
26	ENTRY	
27	LABEL	
28	定　数	変　数
29	組込み関数	
30	CHECK コンディションの名標リストに指定あり	
31	DOステートメントのオプション1である	
32	明示宣言によって属性が付けられた	
33	省略時解釈によって属性が付けられた	
34	前後関係によるもの	
35	暗黙の宣言によるもの	

＊ 2：名標の文字数

＊ 3：データの場合はそのサイズ（桁数，文字数，ビット数），構造体の場合は大きさ（語数）

＊ 4：算術データの場合は，整数部の桁数

＊ 5：算術データの場合は，小数部の桁数

＊ 6：構造体の場合は，構造体レベル，組込み関数の場合は，その番号

＊ 7：構造体の場合は，主構造体からの相対番号

＊ 8：配列の場合は，1次元目の上限

＊ 9：構造体の場合は，その構造体中のデータが割付けられた先頭のバイト位置（0～3）

＊10：配列の場合は，その次元数

＊11：配列の場合は，2次元目の上限

＊10～11：FILE名の場合は，FCBを指すポインタ，

.ENTRY 名の場合は，その名標が定義されたインデックス部を指すポインタ

*12：配列の場合は，要素の合計数

*13：配列の場合は，３次元目の上限

*12～13：ENTRY 指定の場合は，もし，引数があるときは，その引数に関する情報の入っている記号表を指すポインタ，

LABEL の場合は，ＤＯのランクを示す

*14：基底付変数の場合は，ポインタ変数の記号表を指すポインタ，

LABELの場合は，PROCEDURE のランク

*15：名標に対して割け付けられた番地又はレーベルが定義されたステートメントのインデックス部を指すポインタ

*16：ブロック・レベル

*17：PICTURE指定の場合は，PICTURE図形の記憶場所を指すポインタ

3.6 オブジェクト・プログラムの形式

このコンパイラでは，ステートメント単位に挿入（Insert），削除（Delete）および置換え（Replace）が出来る機能をもっているので，ステートメント単位に Chained Sequential に主記憶装置内に保存されている。

各ステートメントには，Chainの情報とそのステートメントに関する情報とをもっている。

前者をインデックス部，前者をステートメント部と呼ぶことにする。

インデックス部は，次に示す１０個の情報から成っている。

① 次に実行すべきステートメントのインデックス部を指すポインタ

② 物理的な前のステートメントのインデックス部を指すポインタ

③ 物理的な後のステートメントのインデックス部を指すポインタ

④ ソース・ステートメントの占有行数

⑤ このステートメントの前に実行されるステートメントのインデックス部を指すポインタ

⑥ ステートメントの種類

⑦ コンディションに関する情報

⑧　ステートメントのライン番号

⑨　オブジェクト・ステートメントの大きさ（語数）

⑩　ソース・ステートメントの位置を示すポインタ

①，②，③，⑤は，ステートメントがChained Sequential になっているため，次のステートメントの場所及び前のステートメント場所を示すのに必要である。

①，⑤は実行フェーズでステートメントを実行するときに必要である。

②，③は，修正など編集するときに必要である。

④は，ソース・プログラム・リストをキャラクタ・ディスプレイ装置のフレーム内に表示するときに必要である（フレームの大きさは１５行に固定）。

⑥は，実行フェーズを効率的に行なうための情報である。

⑦は，コンディション接頭語を処理するための情報である。

⑧は，学習者が各ステートメント毎にタイプインするライン番号で0～999.99まで使用することができる。

⑨は，セーブされたステートメントを再コンパイルするときに必要である。

⑩は，ソース・ステートメントの入っている場所を示すために必要である。

しかし，オブジェクト・ステートメントはインデックス部の直後から入り，その場所は固定しているので場所に関する情報は必要ない。

ステートメント部は，ソース・ステートメント項目とオブジェクト・ステートメント項目の２つに分かれる。

１つのステートメントのソース・ステートメントおよびオブジェクト・ステートメントの量は可変である。

１つの領域にオブジェクト・ステートメントとソース・ステートメントが混って入って良いならば語単位に出力することができるが，オブジェクト・ステートメントは機械語になっている部分が多いために全部まとまっている必要がある。したがって，一般に記憶する情報の少い方を他方が出力されるまでためておき，他方の出力がすべて終った時に，ためておいた情報を出力する。

ＣＰＬでは，ソース・ステートメントよりそのオブジェクト・ステートメン

トの方がしめる語数が大であるから，ソース・ステートメントをためておいてオブジェクト・ステートメントが全て出力された後，ソース・ステートメントをその後に出力する。

これらを図示すると図3.3になる。

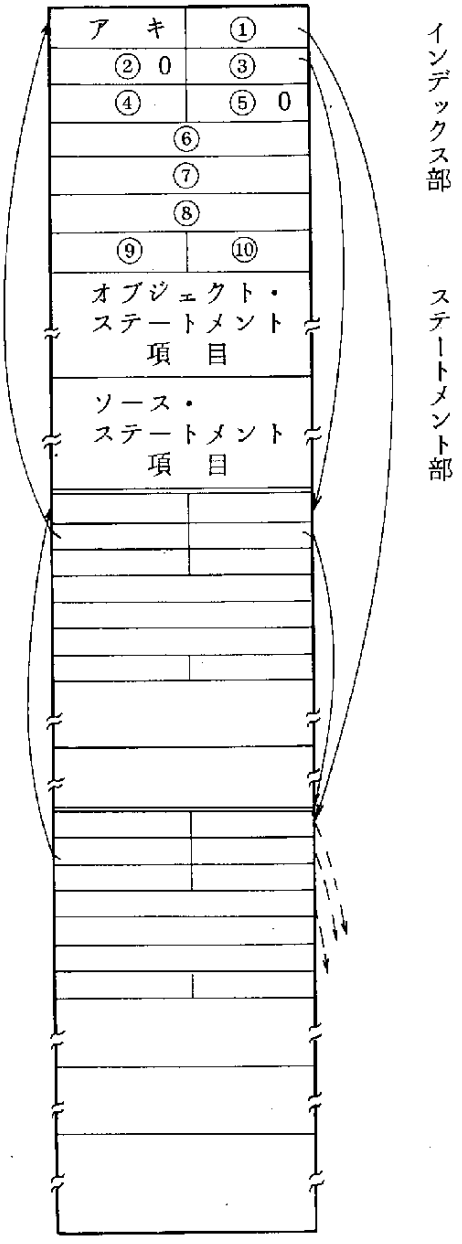


図 3.3　コンパイラの出力形式

第４章　サンプル・プログラム

4.1　サンプル・プログラムとは

学習者にプログラミングの問題を指示し，自由にプログラミングさせ，そのプログラムを診断するには，同じ問題を解く模範プログラムを同じデータを使って走らせてみて，実行結果を比較するのが最も容易な方式の一つである。

模範プログラムには，その問題に対するプログラムだけではなく，プログラム上，よくまちがいやすい所を診断するため，実際にまちがったプログラムも入れておく。このプログラムをサンプル・プログラムと呼ぶ。

サンプル・プログラムは，ＣＰＬ言語でコーディングし，ＣＰＬプロセッサで学習前に実行しておく。

4.2　サンプル・プログラムの例

（問題１）

整数データ a_1，a_2，……，a_n を読み込み，これらを印刷し，かつ，そのうちの最小値とデータの個数ｎとを印刷するプログラムを作成せよ。

ここに，データは１０桁以内の正の整数で，それぞれ入力され，データの尽きた印として，０または負の数を入力するとする。

ただし，データ読込みエリアの変数名をＡ，最小値を求めるエリアの変数名をAMIN，データの個数をカントするエリアの変数名をＮとせよ。

図4.1　指示問題の例

図4.1の問題1を学習者にプログラミングさせた場合，次の誤りを犯しやすい。

(1)　最小値を求めるかわりに最大値を求める。

(2)　データの個数のカウントが１だけ多いかすくない。

(3)　最初のデータ又は最後のデータを最小値とする。

これらの誤りを検出するため，サンプル・プログラムは診断プログラムへ適切な情報をあたえるようにコーディングされる。

図4.2に，この問題に対するサンプル・プログラムの例を示す。

ライン#100のステートメントで診断用の最大値を求め，ライン#120でデータを保存しておく。

FORTRAN CODING SHEET

PROGRAM _____ PAGE _____ OF

CODED BY _____ DATE _____

SEQUENCE NO.	STATEMENT NUMBER	FORTRAN STATEMENT	IDENTIFICATION
0010.	0	MIN: PROCEDURE OPTIONS(MAIN);	
0020.	0	DECLARE AMIN DECIMAL FIXED(10), /* STORAGE ASSIGNED */	
		N DECIMAL FIXED(10),	
		A DECIMAL FIXED(10);	
0030.	0	DECLARE B(100) DEC FIXED(10), /* FOR DIAGNOSIS */	
		AMAX DEC FIXED(10);	
0040.	0	AMIN=999999999;	
0050.	0	AMAX=0;	
0060.	0	N=0;	
0070.	0	LOOP: GET LIST(A);	
0080.	0	IF A <=0 THEN GO TO FINISH;	
0090.	0	IF A < AMIN THEN AMIN=A;	
0100.	0	IF A > AMAX THEN AMAX=A;	
0110.	0	N=N+1;	
0120.	0	B(N)=A; /* SAVE DATA FOR DIAGNOSIS */	
0130.	0	GO TO LOOP;	
0140.	0	FINISH: END MIN;	

図4.2 問題1のサンプル・プログラム例

JAPAN INFORMATION PROCESSING DEVELOPMENT CENTER

第5章　診断プログラムとDIAL

5.1　診断プログラムとは

提示された問題に対して学習者がコーディングしたプログラムの実行結果とサンプル・プログラムの実行結果の比較手順の手続きを表わしたプログラムを診断プログラムという。

診断プログラムはDIAL言語でコーディングされている。

5.2　診 断 方 法

プログラム診断の1つの方法として，ＣＰＬ言語で書かれた学習者のプログラムをデータをあたえて実行させ，その実行結果とすでにシステムで準備してあるサンプル・プログラムを同じデータで実行させた結果とを比較する。

実行結果を比較するにはまず，両者のプログラム中で重要な変数の名前の付け方を統一しておく。ＣＰＬ言語では変数に対してバラエティに富んだ属性を与えることができるため，まず，変数の属性を比較する。属性の中でも，例えばDECIMAL FLOATとBINARY FLOATの中どちらの属性を指定してもよい場合があるので，必ずしもすべての属性が一致する必要はない。

次に変数の内容の比較であるが，一致している場合は問題がないが，一致しない場合，何故一致しなかったかの診断を行わなければならない。サンプル・プログラム中には，誤りやすい部分（アルゴリズムの誤り）をわざと誤ったコーディングもされているので，その中のどの結果と一致するかを調べ，それによってアルゴリズムの誤りを指摘する。

例えば，プログラム・ループでよく回数のカウントが1回だけ多かったり，少なかったりすることがよくある。学習者のプログラム中の回数用の変数とサンプル・プログラム中の回数用の変数の内容をくらべ，1だけ異る場合は，プログラム・ミスとして考えられるのは，初期値の誤り，回数の終りの判定位置の誤りが考えられるので，それらを診断結果として学習者に知らせる。

5.3 DIAL言語の概要

DIALは診断プログラムを記述する言語である。一般のＣＡＩのＣＡＩ言語に相当する。

DIALはPL/Ｉとほゞ同じ機能を持つが，このCLASS－F システムにおけるプログラム診断に便利な機能が若干つけ加えられている。特にＣＰＬプロセッサの中の記号表を直接参照したり，変数の属性の宣言の正否を診断したりする特殊目的がある為，変数名のつけ方に工夫をしている。例えば診断すべき属性は，その属性の前に〝？〟記号を書けば，システムが自動的に診断し，適切な診断メッセージを出す。

〔例〕 DECLARE　@A　？FIXED(？10)；

この例は，変数@Aが固定小数点であるかどうか，その桁数が１０桁かどうかを診断すべきことを表わしている。

変数は４種類あり，次のように区別している。学習者のプログラム中で使われる変数を参照する変数には，頭に〝@〟記号を付け ― スチュデント変数と呼ぶ ―，サンプル・プログラム中で使われる変数を参照する変数には，頭に〝＃〟記号を付け ― サンプル変数と呼ぶ ―，診断プログラム中で特別な機能を有する変数には，頭に〝¥〟記号を付け ― システム変数と呼ぶ ―，診断プログラム中だけに有効な変数は頭に特殊文字を付けず ― ローカル変数と呼ぶ ―，表現する。

例えば，学習者のプログラム内の変数AMINに正しい結果が求まっているかどうかを診断するには，サンプル・プログラム内の変数AMINに正しい結果が入っているので，

IF @AMIN＝#AMIN THEN～； ELSE～；

というステートメントで診断することができる。

5.4 DIAL言語仕様

5.4.1 プログラム要素

1. 言語の基本的構造

【1】 使われる文字

プログラムを書くには，６０字の集合が使われる。

６０字集合は，英子，数字，特殊文字より構成される。

英　　字

英　　字　　A～Z

数　　字

数　　字　　0～9

特殊文字

等　号	＝	セミコロン	;
空　白		コロン	:
正符号（プラス）	＋	論理否定（Not 記号）	¬
負符号（マイナス）	－	論理積（And 記号）	&
アスタリスクまたは乗算記号	＊	論理和（Or記号）	|
スラッシュまたは除算記号	／	〝より大きい〟記号	＞
左括弧	（	〝より小さい〟記号	＜
右括弧	）	下　線	＿
コンマ	,	疑問符	?
ピリオドまたは小数点	．	通貨記号	¥
引用符号	'	単価記号	@
パーセント	％	番号記号	#

空白も文字である。英数字は英字と数字である。特殊文字はある場合，組合わせによって別の意味をもつほかの記号を作る。たとえば，＞＝は〝より大きいか等しい〟という意味である。

【2】 分離記号 (delimitor)

ある文字は，分離記号として使われて，次の三種類に分けられる。

演算子

括　弧

区切り記号（separator）と他の分離記号

(1) 演　算　子

言語で用いられる演算子は，４つの型に分けられる。

算術演算子

比較演算子

ビット・ストリング演算子

ストリング演算子

(2) 算術演算子

算術演算子には次のものがある。

＋　加法または，プラスの接頭語を示す。

－　減法または，マイナスの接頭語を示す。

＊　乗法を示す。

／　除法を示す。

＊＊　ベキ乗を示す。

(3) 比較演算子

比較演算子には次のものがある。

＞　より大であることを示す。

¬＞　より大でないことを示す。

＞＝　より大であるか，等しいことを示す。

＝　等しいことを示す。

¬＝　等しくないことを示す。

＜　より小であることを示す。

¬＜　より小でないことを示す。

(4) ビット・ストリング演算子

ビット・ストリング演算子には次のものがある。

¬　否定を示す。

&　　論理積を示す。

|　　論理和を示す。

(5) ストリング演算子

ストリング演算子には次のものがある。

||　　連結を示す。

(6) 括　弧

括弧は式中や，リストをかこむためや，いろいろなキー・ワードと結びついた情報を指定するのに用いられる。

(　　左括弧

)　　右括弧

(7) 区切り記号とその他の分離記号

コンマ	,	リストの構成要素を分離する。
セミコロン	;	ステートメントを終了させる。
代入記号	=	代入ステートメントおよびＤＯステートメントで使う。
コロン	:	ラベルの後につづく。
空　白		区切り記号として用いられる。
引用符	'	ストリング定数をかこむ。
ピリオド	.	修飾された名前の中の項目を分離する。 定数中で１０進または２進の小数点として用いられる。
通貨記号	¥	システム変数の頭に付けられる。
単価記号	@	スチューデント変数の頭に付けられる。
番号記号	#	サンプル変数の頭に付けられる。

【3】 名標 (identifier)

名標は，分離記号を前後に持ち，コメントまたは定数に含まれない英数字および下線の列である。ただし，はじめの文字は，いつも英文字でなくてはならない。

言語の中で，名標は次のように使われる。

スカラ変数の名前

配列の名前

構造体の名前

ステートメントのラベル

プログラム名

キーワード

(1) 名標の長さ

DIAL言語でプログラムを書く際に，8文字より長い名標を用いてはならない。

【4】 キーワード

キーワードは，言語の一部となっているような名標をいい，予約された語である。それらは次のように分類される。

ステートメント名標

属　性

区切り用キーワード

組込み関数の名前

システム変数の名前

【5】 空白の用法

名標，定数，合成演算子（たとえば¬=）等は空白を含んではいけない。空白は文字ストリング定数の中で使用が許される。

名標，定数等は，隣接して使用してはいけない。それらは，演算子，代入記号，括弧，コロン，セミコロン，コンマ，ピリオード，空白，コメント等を間に入れなければならない。さらに，空白やコメントを追加挿入することは常に許される。空白はステートメント名標（GO TOなど）の間には入れても入れなくてもよいが，レベル番号とそれに続く名標の間には，少くとも1個入れなければならない。

〔例〕

CALLAはCALL Aと等しくない。

K TO L BY MはKTOLBYMと等しくない。

A+BはA + Bと等しい。

【6】 コメント

一般形：

／＊ 文字ストリング ＊／

コメントは通常，ドキュメンテーションのために使用され，プログラムの実行には関係しない。また，空白が許される所なら，どこにでも挿入できる（文字ストリング内を除く）。

コメントに用いた文字ストリング中に，文字ストリング＊／をこの順序に入れてはならない。

〔例〕

LABEL: ／＊ THIS IS A COMMENT ＊／ A=B+C;

2. プログラムの基本的構造

DIALプログラムは，ステートメントという基本要素から構成される。ステートメントは，グループという，より大きなプログラム要素にまとめられる。ステートメントには，単純ステートメントと複合ステートメントの2つのタイプがある。

【1】 単純ステートメント

単純ステートメントは次のように定義される。

〔〔ステートメント名標〕ステートメント本体〕;

ステートメント名標はキーワードで，ステートメントの種類を特徴づける。これがなくてステートメント本体がくる場合は，代入ステートメントである。セミコロンのみのステートメントを空ステートメントと呼ぶ。

〔例〕

DO I=1 TO 10;（DOはキーワード）

A＝B＋C；　　（代入ステートメント）

；　　（空ステートメント）

【2】 複合ステートメント

複合ステートメントは他のプログラム要素を含むステートメントである。それは次の通りである。

ＩＦ　複合ステートメント

複合ステートメントに含まれる最後のステートメントは単純ステートメントで，従って終了するセミコロンを伴う。複合ステートメントはこのセミコロンで自動的に終了する。

〔例〕

ＩＦ　A>B　THEN　MAX=A；　ELSE　MAX=B；

【3】 ラベル接頭語

ステートメントはそれを呼びだすためにラベルをつけておくことができる。ラベルを付けたステートメントの形は，

名標：ステートメント

この名標をラベルと呼び，このステートメントを呼ぶ際に同じ意味に使うことができる。

PROCEDURE の前にくるラベルは特殊な場合で，プログラム名と呼ばれる。それ以外のラベルはステートメント・ラベルと呼ばれる。

【4】 グループ

グループは１つ以上のステートメントの集合で，制御のためにつかわれる。

グループは次の形式を持っていて，ＤＯグループと呼ばれる。

〔ラベル：〕　ＤＯステートメント

プログラム要素１

プログラム要素２

：

ＥＮＤ〔ラベル〕；

ＥＮＤに続くラベルはＤＯステートメントに付けられたラベルである。

ＤＯステートメントはＤＯグループの先頭のステートメントと呼ばれ，繰り返しを指定することができる。各プログラム要素は１つあるいはそれより多いステートメントを表わす。

〔例１〕

```
DO I=1 TO 10;
SX=SX+X(I);
SY=SY+Y(I);
END;
```

〔例２〕

```
IF X=Y THEN DO; ISW=1; A=0.0; END;
       ELSE GO TO LABEL;
```

【5】 プログラム

プログラムは，一つの外部手続きにより構成される。

〔例〕

```
DIA_EX:PROCEDURE OPTIONS(MAIN);
       DCL @A ?FIXED, #A FIXED;
       IF @A=#A THEN GO TO OK;
        TYPE('A ノ アタイ ガ チガウ.');
        STOP(0);
       OK:
        TYPE('A ノ アタイ ハ タダシク モトマッテイル
             .');
        STOP(100);
       END;
```

【6】 文の種類

ＤＩＡＬは次の５種類の文に分けられる。

① 定義文

DECLARE

② 入出力文

DISPLAY

TYPE

③ 計算の文

代入の＝

④ 制御文

GO TO

IF

DO

CALL

STOP

⑤ プログラム構造文

PROCEDURE

DO

END

5.4.2 データの要素

1. データの編成

【1】 スカラ項目

定数とスカラ変数はスカラ項目と呼ばれる。

(1) 定　数

定数は，それ自身を表わすデータ項目である。つまり，その表現が，そのまま名前であり，値であるようなものである。その値は，プログラムの実行中に変化しない。各定数は型を持っている。

定数としては，算術定数，文字やビットの列があるが，前者には符号付きと符号付きでないものがある。

定数 { 算術定数
　　 { 文字やビットの列定数

算術定数 { 符号付き定数
　　　　 { 符号なし定数

(2) スカラ変数

スカラ変数は，定数と同じようにデータ項目を表わす。このデータ項目をスカラ変数の値と呼ぶ。しかし，定数と異なる点はプログラムの実行の際に1つ以上の値をとりうるという点である。

スカラ変数を利用するときは，名前で呼び出し，その名前は，単純名，添字付きの名前，修飾された名前，添字付きの修飾された名前の場合がありうる。

【2】 データ集合体

変数データ項目は，配列あるいは構造体にまとめられる。

(1) 配　列

配列はn次元の順序ずけられた要素の集合で，それらの要素は，みなおなじデータの宣言をもつ。もし，算術型ならば配列要素はみな同じ基底，表示，モード，あるいは精度を持つ。文字ストリングあるいはビットストリングの場合は，みな長さが同じである。配列の次元数および各次元の上限は，次元属性をつかって指定される。

〔例〕

DECLARE A(3, 2);

このステートメントは，Aを3行2列の2次元の配列として定義する。

配列要素は次の順にとられる。

A(1, 1), A(1, 2), A(2, 1), A(2, 2), A(3, 1), A(3,2)

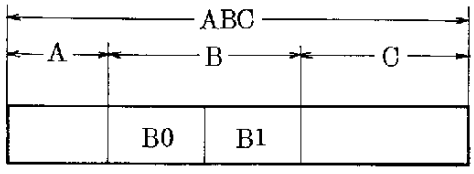
(2) 構造体

構造体はスカラ変数あるいは構造体の集りで，含まれるものにレベル番号がつけられている。それらは同じデータの型をもつ必要はなく，また同じ属性をもたなくてもよい。最も外側の構造体を主構造体(major structure)という。構造体に含まれる構造体を従構造体

(minor structure)という。主構造体は必らずレベル１でなければならない。含まれる構造体は常にそれをふくむ構造体より数値的に大きいレベル番号を持たなければならない。レベル番号がついていても構成要素を持たない名標は構造体とは考えない。レベル番号の後には，１つ以上の空白をおかなければならない。

〔例〕

```
DECLARE        レベル1
  1 ABC        レベル2
    2 A,       レベル3
    2 B,
      3 B0,
      3 B1,
    2 C;
```



上の例で，ＡＢＣはスカラ変数Ａ，Ｃおよび構造体Ｂを含む主構造体である。構造体Ｂはスカラ変数Ｂ０とＢ１を含んでいる。

2. 名　前

特定のデータ項目，項目グループ，配列および構造体を呼び出す場合の規則について述べる。データの名前の形としては，単純名，修飾された名前，添字つき名前が許される。

【1】 単純名

単純名は，スカラ，配列，構造体を呼び出す名標である。

【2】 添字つき名前

添字つき名前は配列のある要素を呼び出すのに用いられる。それは添字リストを従えた，配列名として宣言されている単純名のことである。添字はコンマで区切られ，全体がカッコでくくられている。添字は添字式であり，使われる前に計算され，整数に変換される。添字の数は配列の次元の数と等しくなければならない。また指定された添字の値は配列の次元として宣言された上限内に入っていなければならない。

添字つき名前は次の形式をとる。

名標（添字〔，添字〕……）

〔例〕

A（3）

ALPHA（I+7）

【3】 修飾された名前

同じ名前がついた項目が異なる構造体に含まれるときは，それらを混同せずに区切してよぶのに一意の名前（unique name）が必要である。これは修飾された名前を作ることによってなされる。このことは1つ以上の項目に共通な名前の前に，その名前が含まれている構造体の名前をつけることを意味する。この操作は修飾された名前が，求めている名前を一意的に呼び出すようになるまで，つぎつぎにほどこしてよい。

このように，修飾された名前は，レベル数のふえる順序に，左から右へならべられた名前の列である。名前はピリオドで区切られ，ピリオドの前後には必要ならば空白を入れてもよい。名前の列は，それを含んでいる構造体の全部を含む必要はないが，あいまいさを除去するには十分なだけ名前を含まなければならない。

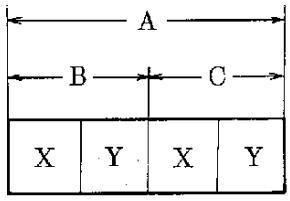
修飾された名前は，一度構成されると，それ自身一つの名前である。名前という語が用いられるときは，それは修飾された名前も含むものとする。

修飾された名前は，次の形をとる。

名標〔．名標〕……

〔例〕

```
DECLARE 1 A,
        2 B,
          3 X,
          3 Y,
        2 C,
```



3 X，

3 Y；

では，それぞれ2つのX，Yを

$$\begin{cases} B \cdot X \\ C \cdot X \end{cases} \quad や \quad \begin{cases} B \cdot Q \\ C \cdot Q \end{cases}$$

で区別する。

3. データの型

データの型は，普通のデータとラベル・データの2つに分類される。

【1】普通のデータ

普通のデータとは，算術データあるいは，ストリング・データとして類別されるデータである。

(1) 算術データ

算術データ項目は，基底（base），表示（scale），精度といった性質を値にもつものである。データ項目は，コード化された型，すなわち，計算機に依存する内部的表現であらわされる。コード化された型のデータ項目は，基底，表示，モード，精度を指定する属性によって与えられる性質をもっている。

基底（10進か，2進か），表示（固定小数点か，浮動小数点か）および精度は，記述されているデータの内部表現や用いられる内部の演算に関係する。

1) 基　底

算術データは10進か2進の基底をもつものとして指定される。

2) 表　示

算術データは固定小数点表示か，浮動小数点表示をもつものとして指定される。固定小数点データ項目は指定された桁数の10進または2進数字をもつ有理数である。10進または2進小数点を位取りによって，指定することができる。浮動小数点項目は，仮数部分と指数部分の形をもつ有理数である。

3) 精　度

固定小数点データの精度（w，d）は，wで２進または１０進数の保持すべき桁数を表わし，dで位取りを表わす。浮動小数点データの精度は，wで２進または１０進数の仮数部分に保持すべき有効桁数を表わす。

(2) 実数定数

実数定数は，２進数または１０進数である。

1） １０進固定小数点定数

１０進固定小数点定数は，１個以上の１０進数であらわされ，必要に応じて小数点をもつものである。もしも，小数点が指定されなければその定数は１０進整数定数である。

〔例〕　12.345　.678　314.　27.　100

2） ２進固定小数点定数

２進固定小数点定数は，１個以上の２進数であらわされ，必要に応じて小数点をもつものである。最後にＢをもつ。

〔例〕　1010B　100.1101B　.011B

3） １０進浮動小数点定数

１０進浮動小数点定数は，必要に応じて小数点のついた１桁以上の１０進数のあとにＥをつけて，その後に必要に応じて符号のついた指数をつけたものである。その指数は１０のベキ乗を指示する１０進数である。

〔例〕　12.E34　3.14E−1　.42E+5　56E7

4） ２進浮動小数点定数

２進浮動小数点定数は，必要に応じて小数点のついた２進数の後にＥをつけて，その後に必要に応じて符号のついた指数をつけ，最後にＢをつけたものである。その指数は２のベキ乗を指示する１０進数である。

〔例〕　1.101011E3B　.11001E−17B

5) 実数定数の精度

式の計算のために実数に対して精度が明確に定数される。

実数固定小数点は（w，d）という形の精度を持ち，wは定数内の数字の個数，dは小数点の右側にある数字の個数である。浮動小数点定数の精度は（p）である。ここでpはEの左側にある定数の桁数である。

〔例〕 3.14 の精度は（3，2）

0.012E5 の精度は（4）

0000001B の精度は（7，0）

(3) 算術変数

算術変数は算術データ項目の名前である。これらの名前は，基底，表示，精度の特性（すなわち属性）を与えられている。

(4) ストリング・データ

ストリング・データは，文字ストリングとビット・ストリングに分けられる。ストリング・データ項目の長さは，文字ストリングの場合はその文字数で，ビット・ストリングの場合はそのビット数に等しい。長さ0のストリング・データ項目を空ストリングという。

(5) 文字ストリング・データ

文字ストリング・データは，データ文字集合の文字が0個以上並んだものである。ストリングの長さは固定である。

① 文字ストリング定数

文字ストリング定数は，データ文字集合の文字の列を引用符でかこんだものである。引用符自身をこれで表わしたければ，となりあう2つの引用符であらわさなければならない。

文字ストリング定数は，構文上は，注釈となるような文字の例をふくんでもよい。しかしながら，これらの文字は注釈とはならず，ストリングの値の一部になる。

〔例〕 '$123.45' 'JAPAN' 'IT''S'

(6) ビット・ストリング・データ

ビット・ストリング・データは0個以上のビットの列である。ビット・ストリング・データは長さが固定で，その実際の長さを指定する。

① ビット・ストリング定数

ビット・ストリング定数は，0個以上のビットの列を引用符でかこみ，最後にBをつけたものである。

〔例〕 '0100'B ''B

(7) ストリング変数

ストリング変数は，ストリング・データ項目の名前である。これらの名前はストリング特性を与えられている。

【2】 ラベル・データ

ラベル・データは必ず，ステートメント・ラベルとむすびついてつかわれる。または，ステートメント・ラベル・データは，定数または変数でありうる。また，その変数は配列であってはいけない。

(1) ステートメント・ラベル定数

ステートメント・ラベル定数とは，プログラム中に，ステートメント・ラベルとしてあらわれている名標のことである。それによって，ステートメントを呼び出すことが可能である。

〔例〕

```
HERE: IF A > 0.0 GO TO THERE;
                :
          GO TO HERE;
THERE: X=Y+Z;
```

(2) ステートメント・ラベル変数

ステートメント・ラベル変数とは，値としてステートメント・ラベル定数をとり得る変数のことである。

〔例〕

```
DECLARE L LABEL;
```

```
    L=P1;
      ⋮
    L=P2;
    GO TO L;
P1: ...
P2: ...
```

ラベル変数LはP1またはP2を値とすることができる。上記の例では，GO TO L; はGO TO P2; と同じ意味である。

5.4.3 データの記述

1. 属 性

DIALのプログラムに現われる名標は，何種類かの対象の1つを表わすことができる。

名標によって表現される対象を規定する諸性質および名標自体の性質が一緒になって，名標にあたえることのできる属性の集合を形成する。

属性には多数の種類がある。これらの種類と各々の属性について述べる。

名標がプログラムの特定の文脈で使われているとき，その名標に一意的な意味を与えるためには，属性が知られていなければならない。例えば，名標がデータ変数として使用されているときはデータの型が知られていなければならない。データの型が算術データならば基底，表示および精度が知られていなければならない。

〔例1〕 CHARACTER(10)

この属性をもつ名標は長さ10の文字ストリングを示す変数を表わす。

〔例2〕 FLOAT

この属性をもつ名標は計算機内部で浮動小数点表示で表わされるデータを示す変数を表わす。

2. 宣 言

名標は宣言によっていくつかの属性を与えることができる。

宣言には，明示された宣言，前後関係による宣言，暗黙の宣言の3種類がある。

【1】 明示された宣言

明示された宣言は，DECLAREステートメント，ラベル接頭語に現われることによって行なわれる。これによって名標が名前として認められ，かつ属性の集合が与えられる。

(1) DECLAREステートメント

機能：

DECLAREステートメントは，単純名の属性を指定するために用いられ実行されないステートメントである。

一般型：

DECLARE 〔レベル〕名前〔属性〕・・・
〔,〔レベル〕名前〔属性〕・・・〕・・・;

構文規則：

1) 名標は，1つのDECLAREステートメントで名前としていくつでも宣言できる。その場合，1つの名標ごとにコンマで区切られる。

2) 属性は，それが関係する名前に続かなければならない。

3) 属性のくくり出しはできない。

一般規則：

1) 特定の名前に明示して与える属性は，すべて1つのDECLARE ステートメントで宣言しなければならない。

【2】 暗黙の宣言

プログラムに使われる名前が，明示にも宣言されてないときには，暗黙の宣言がされたという。

暗黙の宣言では，名前の最初の文字によって決まる省略時の属性が与えられる。

I,J,K,L,M,Nで始まるとき BINARY FIXED(25)

I～N以外の英字で始まるとき　BINARY FLOAT(8)

3. 属　性

属性は，それが関係する名標にある性格を与えるのに使われる。属性は次の種類に分けられる。

名前の種類

データの属性

診断属性

【1】 名前の種類

名前の種類には次の種類がある。

スチューデント変数名

サンプル変数名

システム変数

ローカル変数

名前の種類を宣言するには，その名前の最初の文字の前に特殊文字をつける。

名前の前に単価記号（@）を付けるとスチューデント変数名，番号記号（#）を付けるとサンプル変数名，通貨記号（¥）を付けるとシステム変数名，特殊文字をつけないとローカル変数名になる。

スチューデント変数は，学習者の作成したプログラム中の同じ名標の変数名を参照するのに使われ，サンプル変数は，サンプル・プログラム中の同じ名標の変数名を参照するのに使われ，システム変数は，DIALシステム内で定義されている関数を参照するのに使われ，ローカル変数はDIALプログラム中だけで有効なものである。

【2】 データの属性

(1) 算術データ

変数は，基底，表示の属性のいずれかが与えられた場合算術型であると宣言される。

(2) 基　底

機能：

基底属性は，データが２進か１０進のいづれの形であるかを宣言する。

一般形：

BINARY|DECIMAL

〔例〕 DECLARE A DECIMAL, B BINARY;

(3) 表　示

機能：

表示属性は，データが固定小数点表示が，浮動小数点表示かを指定する。

一般形：

FIXED|FLOAT

〔例〕 DECLARE X FIXED, Y FLOAT;

(4) 精　度

機能：

精度属性は，固定小数点または浮動小数点データに対し，保存されるべき２進または１０進の有効桁数を指定する。位取り(scale)を指定することもできる。

一般形：

（桁数〔，位取り〕）

一般規則：

1) 精度属性は表示あるいは基底属性の直後におかなければならない。

2) 桁数は保持されるべき２進または１０進の桁数を指定する正の１０進定数で，固定小数点，浮動小数点のいずれのデータでも使われる。

3) 位取りは指定された桁数の整数データ項目を基準にした点（小数点）の位置を定義する。

これは固定小数点データに対してだけつかうことができる。

4) 表示が固定小数点を示し，位取りが与えられていない場合には，位取りは0と仮定される。

5) 位取りは負であっても，また桁数より大きくてもかまわない。

6) 位取りは位取りの符号を逆にした数で基底をベキ乗したものと，整数データの積が実際の値であることを示す。

たとえば，精度が（5, 2）の10進データは，.04から999.99までか0の値をとる数を表わす。

(5) 算術データの省略時解釈

もし，基底，表示のいずれも指定されていないなら，算術データの省略時の属性は，名前の最初の文字に依存する。もし，名前の最初が，IからNまでのものであるなら，BINARY FIXED とみなされる。そうでないときは，BINARY FLOAT とみなされる。

もし，算術データの属性が部分的に指定されているなら，残りの属性は，次のようにみなされる。

基底：DECIMAL

表示：FLOAT

もし精度が指定されていないとき仮定される精度は，

DECIMAL FIXED ……（8, 0）

BINARY FIXED ……（25, 0）

DECIMAL FLOAT ……（8）

BINARY FLOAT ……（26）

によって定義される。

(6) ストリング属性

機能：

ストリング属性は，ストリング・データの長さと，それが，ビット・ストリングであるか，文字ストリングであるかを指定する。

一般形：

{ BIT / CHARACTER }（長さ）

一般規則：

1) ＢＩＴはビット・ストリング・データを，CHARACTER は文字ストリング・データを指定する。

2) 長さは固定長ストリングの長さを表わす。

3) 長さの記述は符号なし整数である。

(7) ラベル属性

機能：

ラベル属性は，関係する変数が，ステートメント・ラベルを値として持つことを表わす。

一般形：

LABEL

(8) 次元属性

機能：

次元属性は配列の上限を定義する。

一般形：

（上限〔，上限〔，上限〕〕）

構文規則：

上限は，符号なし正の整数である。

一般規則：

1) 上限の個数は配列の次元数を指定する。

〔例〕 DECLARE D(3,2);

Dは，３行２列の２次元配列である。１次元添字は１～３，２次元添字は１～２である。

(9) 診断属性

機能：

診断すべき属性を定義する。

一般形：

?

一般規則

1) 診断すべき属性の直前に？記号を書く。

2) スチューデント変数の属性に対してしか指定することができない。

〔例1〕 DECLARE @A ?FIXED;

スチューデント変数Aの属性が固定小数点かどうか診断することを指定する。

〔例2〕 DECLARE @ARRAY?(?20,30);

スチューデント変数ARRAYの属性が，2次元配列かどうか，一次元添字の上限が20かどうかを診断することを指定する。

5.4.4 データ処理

1. 式

式は，ある値を計算するのに使われる算法である。式は，定数，スカラ変数，関数の呼び出し，括弧でくくられたスカラ式，単項演算子の先行するスカラ式，2項演算子でむすばれた2つのスカラ式から構成される。式中のオペランドは必らずしも同じ属性のデータである必要はない。もし属性が異なれば，演算に先立って変換が行なわれる。特に添字式は固定小数点の精度（p,0）のデータから構成される式である。

【1】 スカラ式

スカラ式は，スカラ値を返す。値の型は式の型と同じである。式の型は演算子（すなわち，算術，比較，ビット・ストリングおよび連結演算子）のクラスに依存する。

A，Bが式であるとき，＋A，－Aの形の式で使われる演算子＋，－は単項演算子と呼ばれる。これらの演算子がA＋B，A－Bの型で使われる場合，それを2項演算子と呼ぶ。

(1) 算術演算子

どんな複雑な式も，単純算術演算を組合せたものである。

単純算術演算子は次の一般形を持つ。

{{+|−}オペランド}|{オペランド{+|−|*|/|**}オペランド}

この一般形は，正，負符号の単項演算および，加減乗徐，ベキ乗の2項演算をしめす。

演算は，符号化された算術データに対してのみ施こされる。必要なら演算を実行する前に，データは符号化された算術データに変換される。

(2) 特性が混合するとき

算術演算の2個のオペランドは，異なった型，基底，表示，精度をもつことを許される。それが異なるときは，以下の規則に従って，変換が行なわれる。

1) 型

算術演算の場合の数値フィールドのオペランドは，符号化された形に変換される。算術演算の結果は常に符号化された形である。

2) 基　底

基底が異るときは，10進オペランドが2進に変換される。

3) 表　示

2個のオペランドの表示が異るときは，固定小数点のオペランドが浮動小数点に変換される。ただし，第1オペランドが浮動小数点で第2オペランドが精度が精度（p,0）の固定小数点であるようなベキ乗の場合には，第2オペランドの変換は行われない。

4) 精　度

精度が異っても，変換は行なわれない。

(3) ビット・ストリング演算

ビット・ストリング演算は，次の一般形をもつ。

¬オペランド

オペランド&オペランド

オペランド1オペランド

単項演算〝否定〟と2項演算〝論理積〟と〝論理和〟が，上に指定されている。オペランドは演算が実行される前にビット・ストリング型に変換される。もし，オペランドが変換後で異なった長さになるなら，短い方は長い方の長さになるまで右の方へゼロをつめて拡張される。その結果の長さは，この拡張された長さである。

演算はビットごとに行われる。演算の結果として，各々のビット位置は次の表に定められた値を持つ。

A	B	NOT A	NOT B	A AND B	A OR B
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

〔例〕 もしAが'010111'Bであり，Bが'111111'B，
Cが'101'Bであるなら，
¬Aは'101000'B になり，
C&Bは'101000'B になり，
A|¬Cは'010111'B になり
¬(¬C|¬B)は'101111' Bとなる。

(4) 比較演算

比較演算は，次の一般形をもつ。

オペランド{<|¬<|<=|=|¬=|>=|>|¬>} オペランド

比較には，3つの型がある。

1) 代数的： これは，符号化された算術データの符号のついた数値の比較を意味する。数値フィールドは符号化された形に変換されて比較される。

2) 文字： これは，照合順序（EBCDICコード）に従って，左から

右の方へ対応する文字ごとの比較を意味する。もし，オペランドが異なった長さであるなら，短い方のものは右の方へ空白をつめて拡張される。

3) ビット： これは，２進数字の左から右への比較を意味する。もし，ストリングが異なった長さであるなら，短い方のストリングは０をつめて右の方へ拡張される。

比較の結果は，長さ１のビット・ストリングである。その値は，もし関係が真ならば'１'B，偽ならば'０'Bである。

もし，比較のオペランドが異った型のものであるなら，低い方の型のオペランドは高い型のオペランドに変換される。型の優先順位は，(1)算術オペランド，(2)文字ストリング・オペランド，(3)ビット・ストリング・オペランドである。

変換の結果，オペランドは両方共，数値または文字ストリングとなり，代数的比較または文字比較がなされる。

(5) 連結演算

連結演算は，次の一般形をもつ。

オペランド１｜｜オペランド

もし，両方のオペランドがビット・ストリング型であれば，変換は行なわれず，結果はビット型となる。他の全ての場合には，必要な場合には文字ストリングに変換して連結を行ない，結果は文字ストリング型となる。

〔例〕 Aが'０１０１１１'B， Bが'１０１'B，Cが'XY，Z'，Dが'AA／BB'であるとき，

A｜｜B　は'０１０１１１１０１'Bとなり，

A｜｜B｜｜Bは'０１０１１１１０１１０１'Bとなり，

C｜｜D　は'XY，ZAA／BB'となり，

D｜｜C　は'AA／BBXY，Z'となる。

【2】 添字式

添字は，次の一般形を持つ。

{オペランド} | {オペランド{＋|－}オペランド}

ここで，オペランドは，固定小数点データで精度（p，0）の定数か変数とする。

5.4.5 入 出 力

入力動作は，データを入力装置からプログラムに読み込む。出力動作は，データをプログラムから出力装置に書き出す。

DISPLAYおよびTYPEステートメントがデータ値を入力装置から読み込んだり，出力装置へ書き出すために使用される。

1. データの指定

データ指定はDISPLAYおよびTYPEステートメントにおいて入出力されるデータを指定するためにあたえられる。データ指定は，入出力の様式に対応している。

【1】 データ・リスト

データの指定には，入出力されるデータ項目を指定するデータ・リストが必要である。

一般形：

要素〔，要素〕・・・

構文規則：

1) 各データ・リストの要素はスカラ，配列，構造体，制御コードまたはこれらを含む反復指定である。

制御コードとしては'／'（スラッシュ）があり，改行を表わす。

【2】 反復指定

一般形：

（要素〔，要素〕・・・

DO 制御変数＝添字式1 TO 添字式2〔BY 添字式3〕）；

構文規則：

1) 反復指定の要素リスト中の各要素については，上のデータ・リストのところで述べてある。

2) 制御変数は，固定小数点で精度（p，0）の属性をもつこと。

添字式1は制御変数の出発値を表わす。添字式3は反復指定の中の要素を実行するたびに，制御変数に加算されるべき増分を表わす。添字式2は制御変数の終了値を表わす。この指定の意味はＤＯステートメントと全く同じである。最後の指定が完了すると制御はデータ・リストの次の要素に移る。

3) 反復指定は3重の深さまでの入れ子にできる。

〔例〕 DISPLY(((A(I,J) DO I=1 TO 2)
DO J=1 TO 3));

これは，配列Aの要素を次の順序で表示する。

A(1,1), A(2,1), A(1,2), A(2,2), A(1,3), A(2,3)

2. 入力の形式

データ項目は文字ストリング変数であり，入力された文字ストリングがそのまま代入される。

3. 出力の形式

データ・リストのスカラ変数の値は，そのデータ値の文字表現に変換されて出力される。

5.4.6 ステートメント

各ステートメントの記述はアルファベット順である。これらのステートメントの間の関係を示すために，論理的な類別を行なうことにする。

1. ステートメント

【1】 分　類

ステートメントは次の論理的なグループに分類される。すなわち，代入，制御，データ宣言，入出力およびプログラム構造である。

(1) 代入ステートメント

代入ステートメントは式を計算してスカラに対して値を代入するのに用いる。

(2) 制御ステートメント

制御ステートメントは，通常の逐次的な制御の流れに影響を及ぼすものである。制御ステートメントは，ＧＯ　ＴＯ，ＩＦ，ＤＯ，ＣＡＬＬ及びＳＴＯＰである。

(3) データ宣言ステートメント

データ宣言ステートメントDECLAREは名標の属性を指定する。このステートメントは，**5.4.3**のデータの記述の項でのべられている。

(4) 入出力ステートメント

入出力ステートメントには，ＤＩＳＰＬＡＹ及びTYPEステートメントがある。DISPLAYステートメントは，プログラムの中の変数とキャラクタ・ディスプレイ装置の間で１つのレコードを入出力する。TYPEステートメントは，プログラム中の変数と端末タイプライター装置の間で出力する。

(5) プログラム構造ステートメント

プログラム構造ステートメントは，PROCEDURE, DO 及びＥＮＤである。PROCEDURE ステートメントはプログラムの記述の最初を表わし，対応するＥＮＤステートメントはプログラムの記述の最後を示す。ＤＯステートメントはＤＯグループの始まりを表わし，対応するＥＮＤステートメントはＤＯグループの終りを表わす。

【2】 制御の順序

通常，制御は１つのステートメントから次のステートメントへと順次わたる。DECLAREステートメントにぶつかれば，制御は次のステートメントへ飛ぶ。また制御はTHENユニットの終りに達したときには，ＩＦステートメントの次へ飛ぶ。逐次制御は，次のステートメント，CALL，GO TO及びSTOPによってかえられる。

CALLステートメントは制御を指定入口へわたす。

GO TO ステートメントは，指定されたステートメント・ラベルへ制御を移すことを行なう。

STOPステートメントは，制御をプログラムから引き離す。

2. 擬変数

組込み関数は，代入ステートメントの等号の左側において擬変数として使用することができる。

1) SUBSTR(s,i[,k])

文字sはストリングを表わす。代入ステートメントの実行中，式が組込み関数SUBSTRで定義されるsの部分ストリングに代入される。この部分ストリングは，つねに，固定長ストリングとして取扱われる。

3. ステートメントのリスト

【1】 代入 (assigment) ステートメント

機能：

代入ステートメントは，式を計算しスカラへ値を代入するために用いられる。

一般形：

{ スカラ変数 / 擬変数 } [,スカラ変数 / ,擬変数] ...=スカラ式;

構文規則：

1) 等号の左にある各変数は，算術的変数でも，ビット，文字データ型でもよい。

2) 算術データと文字ストリング・データの混合演算は禁止する。

一般規則：

1) 代入ステートメントは次のように計算される。

a. 式が等号の左側の添字の中，あるいは擬変数のいずれかに現われると，正確に一度だけ，左から右へ計算される。等号の右

側の式が計算される。等号の右側の式の値が，等号の左の変数に左から右へ代入される。

2) 代入による編集

a．右辺の値が左辺に代入されるときに，左辺の属性に合わせて変換される。精度や長さが変わることもあり，その場合には，文字の付加，変更，削除が行なわれることがある。

b．ある値がストリングの変数に代入されるときは必要に次じて次の操作が行なわれる。

(イ) 受ける側のストリングと同じ型 — ビットまたは文字 — に変換される。

(ロ) 受ける側のストリングの長さにするため，切捨て，または付加，付加が必要のときは，文字ストリングには空白が，ビット・ストリングには0が加えられる。

【2】 CALLステートメント

機能：

CALLステートメントは，標準外部関数手続きを呼び込んで，手続きの指定された入口へ制御を渡す。

一般形：

CALL　手続き名〔（実引数〔，実引数〕．．．）〕；

構文規則：

1) 実引数は，スカラ式，配列名，構造体名のうちどれかである。

一般規則：

1) 実引数は仮引数と属性が完全に一致しなければならない。

【3】 DECLAREステートメント

DECLAREステートメントについては，**5.4.3**のデータの記述の項で述べられている。

【4】 DISPLAYステートメント

機能：

DISPLAYステートメントは，学習者に対して，キャラクタ・ディスプレイ装置にメッセージを表示するもので，また，学習者に応答を要求することもできる。

一般形：

オプション1

DISPLAY（データ・リスト）〔AT（添字式1〔，添字式2〕）〕；

オプション2

DISPLAY（データ・リスト）REPLY（文字変数）
〔AT（添字式1〔，添字式2〕）〕；

構文規則：

1) 添字式1はキャラクタ・ディスピレイ装置のY座標の位置を表わし，添字式2はX座標の位置を表わす。添字式2が省略されると1とみなされる。添字式1及び添字式2が省略されると，現在のカーソルの位置を指定したとみなされる。

2) オプション2では，文字変数に学習者から入力されるメッセージのストリングが格納される。

〔例1〕 DISPLAY (／,'WELCOM')；

改行して，ディスプレイ装置上に'WELCOM'が表示される。

〔例2〕 DISPLAY('セイセキ　ハ', TOKUTEN, '　テン')
AT(20)；

ディスプレイ装置上の第20行目の1字目から

セイセキ　ハ　×××　テン

が表示される。×××はTOKUTENのデータ値とする。

〔例3〕 DISPLAY (LINENO) REPLY (LINE_BUF)
AT (YCOORD)；

ディスプレイ装置上の第YCOOKD 行目の1字目から

××××▶

が表示される。××××はLINENO のデータ値で，▶はヘッデン

グ・マークである。学習者はヘッデング・マークの次から適切なメッセージのストリングを入力する。入力されたストリングは，LINE_BUF の文字ストリング変数に格納される。

【5】 ＤＯステートメント

機能：

DO ステートメントは，DO グループ（'グループ'の項参照）の開始を規定する。グループ内部のステートメントの繰り返しの実行も指定することができる。

一般形：

オプション１

DO；

オプション２

DO 制御変数＝添字式１ TO 添字式２〔BY 添字式３〕；

構文規則：

1) BY 添字式３が指定からはずされれば，添字式３は１であると仮定される。

一般規則：

1) オプション１では，DO ステートメントはDO グループの開始を規定する。

2) オプション２では，DO ステートメントはDO グループの開始を規定し，制御変数には初回に初期値（添字式１）が与えられ，DO グループの実行が１回終わると，制御変数には増分値（添字式３）が加えられて，DO グループが実行される。これが，制御変数の値が終値（添字式２）を越えるまでくり返される。

【6】 ＥＮＤステートメント

機能：

ENDステートメントはグループを終了させるか，又はプログラムの記述の終りを示す。

一般形：

END〔ラベル〕；

一般規則：

1) ラベルがＥＮＤの後にきた場合，そのラベルを持つグループを終了する。

2) ラベルがＥＮＤの後にこない場合，ＥＮＤステートメントは，手前にあり終了していない最も近くのＤＯステートメントで始まるグループを終了させる。もし，ＥＮＤステートメント対応するＤＯステートメントがなければ，プログラムの記述の終りを示す。

【7】 GO TO ステートメント

機能：

GO TO ステートメントは，制御を指定されたステートメントに移させる。

一般形：

{ GO TO / GOTO } { ラベル定数 / ラベル変数 } ；

一般規則：

1) ラベル変数が指定されているときは，GO TO ステートメントは多重路スイッチの作用をもつ。ラベル変数の値は，制御が移されるステートメントのラベルである。

ラベル変数は，GO TO ステートメントの実行のたびごとに異った値を持つことができるので，制御は必ずしも同じステートメントに移らなくてもよい。

2) GO TO ステートメントでDO グループの中へ制御を移してはならない。

【8】 ＩＦステートメント

機能：

ＩＦステートメントは，プログラムの流れを式の値によってかえる。

一般形：

IF スカラ式 THEN ユニット1〔ELSE ユニット2〕

構文規則：

1) 各々の〝ユニット〟はグループであって，セミコロンで終っている。

2) IFステートメントは，それ自身はセミコロンで終らない。

3) 各々のユニットはラベルをもつことができる。

一般規則：

1) 先ずスカラ式を計算し，必要ならばビット・ストリングに変換する。もし結果として生ずるストリングのどれかのビットが値1をもつならばユニット1を実行し，制御をIFステートメントに続くステートメントに移す。もし，結果のストリングの全ビットが0ならば，ELSE節（ELSEとその後に続くユニット）を指定してあればユニット2を実行し，ELSE節の指定がなければ，直ちに，制御を次のステートメントに移す。

2) IFステートメントの入れ子は禁止する。

[9] PROCEDURE ステートメント

機能：

PROCEDURE ステートメントは，手続きの先頭に立ち，プログラム名を定義する。

一般形：

プログラム名：PROCEDURE OPTIONS (MAIN);

一般規則：

1) プログラム名は1～8個の英数字からなり，その最初のものは英字でなければならない。

[10] STOPステートメント

機能：

STOPステートメントはSTOPステートメントを含む手続きを終了

させ，制御をモニタに返す。STOPステートメントは値をかえすこともできる。

一般形：

STOP〔（添字式）〕；

一般規則：

1) 添字式は，一般にスチューデント・プログラムの診断結果の得点をかえす場合に指定する。

添字式が省略された場合は，１００点満足であると見做される。

5.5 診断プログラムの例

4.2の問題1に対する診断プログラムの例を示す。

診断事項

1) 変数名が正しく宣言されているかどうか。

2) 最小値を正しく求めているかどうか。

3) データの個数nが正しく求まっているかどうか。

4) その他

(1) 変数名の属性診断

① 変数名A，AMIN，Nが固定小数点で位取りが0かどうか。

(2) 変数AMINの結果の診断

① 変数AMINに正しい最小値が求っているかどうか。

② 変数AMINに最大値を求めているかどうか。

③ 変数AMINに一番最初のデータが入っているかどうか。

④ 変数AMINに一番最後のデータが入っているかどうか。

(3) 変数Nの結果の診断

① 変数Nに正しいデータの個数が求まっているかどうか。

② 変数Nに1だけ多い個数を求めているかどうか。

③ 変数Nに1だけ少い個数を求めているかどうか。

FORTRAN CODING SHEET

PROGRAM _____ PAGE _____ OF _____

CODED BY _____ DATE _____

COMM	STATEMENT NUMBER (2–5)	CONT (6)	FORTRAN STATEMENT (7–72)	IDENTIFICATION (73–80)
			MIN: PROCEDURE OPTIONS(MAIN) ;	
			/*	
			** THE EXAMPLE OF DIAGNOSTIC PROGRAM **	
			THE RULE FOR IDENTIFIERS WITH FIRST LETTER	
			@ ... THE STUDENT VARIABLE	
			+ ... THE SAMPLE VARIABLE	
			¥ ... THE SYSTEM VARIABLE	
			ELSE.. THE LOCAL VARIABLE	
			*/	
			DECLARE @AMIN ?FIXED(10,?0), /* ? ... THE DIAGNOSTIC */	
			@N ?FIXED(10,?0), /* ATTRIBUTE */	
			@A ?FIXED(10,?0);	
			DECLARE +AMIN FIXED, +N FIXED, +A FIXED,	
			+B(100) FIXED, +AMAX FIXED;	
			DECLARE SCORE FIXED;	
			SCORE=100;	

JAPAN INFORMATION PROCESSING DEVELOPMENT CENTER

(H#2) 40000BRT

FORTRAN CODING SHEET

PROGRAM _____

CODED BY _____

PAGE _____ OF _____

DATE _____

COMM	STATEMENT NUMBER (2–5)	CONT (6)	FORTRAN STATEMENT (7–72)	IDENTIFICATION (73–80)
			/* THE DIAGNOSTIC FOR 'AMIN' */	
			IF ⊕AMIN = ✚AMIN THEN GO TO HERE;	
			SCORE=SCORE-50; /* ｹﾞﾝﾃﾝ 50 */	
			IF ⊕AMIN = ✚AMAX THEN	
			DISPLAY(''AMIN'' = ｻｲﾀﾞｲ ﾉ ﾃﾞｰﾀ ｶﾞ ﾍｲｯﾃｲﾙ.');	
			IF ⊕AMIN = ✚B(1) THEN	
			DISPLAY(''AMIN'' = ｻｲｼｮｳ ﾉ ﾃﾞｰﾀ ｶﾞ ﾍｲｯﾃｲﾙ.');	
			IF ⊕AMIN = ✚B(✚N) THEN	
			DISPLAY(''AMIN'' = ｻｲｺﾞ ﾉ ﾃﾞｰﾀ ｶﾞ ﾍｲｯﾃｲﾙ.');	
			DISPLAY(''AMIN'' ｵ ﾓﾄﾒﾙ ｱﾙｺﾞﾘｽﾞﾑ ｵ ﾐﾅｵｼﾃｸﾀﾞｻｲ.');	
	HERE			
			/* THE DIAGNOSTIC FOR 'N' */	
			IF ⊕N = ✚N THEN GO TO FINISH;	
			SCORE=SCORE-50; /* ｹﾞﾝﾃﾝ 50 */	
			IF ⊕N = ✚N+1 THEN	
			DISPLAY(''N'' ﾉ ｱﾀｲ ｶﾞ 1 ﾀﾞｹ ｵｵｲ.');	
			IF ⊕N = ✚N-1 THEN	

JAPAN INFORMATION PROCESSING DEVELOPMENT CENTER

IBM G. 40000BRT

FORTRAN CODING SHEET

PROGRAM _____ PAGE _____ OF _____

CODED BY _____ DATE _____

COMM	STATEMENT NUMBER (2–5)	CONT (6)	FORTRAN STATEMENT (7–72)	IDENTIFICATION (80)
			DISPLAY('''N'' ノ アタイ カ" 1 ダ" ケ スクナイ.');'	
			DISPLAY('''N'' オ モトメル アルゴ"リズ"ム オ ミナオシテクダ"サイ.');'	
	FINISH:			
			STOP (SCORE); /* SCORE ヘ サイテン クッカ. */	
			END MIN;	

図 5.1 診断プログラムの例

JAPAN INFORMATION PROCESSING DEVELOPMENT CENTER

5.6　ＤＩＡＬコンパイラ

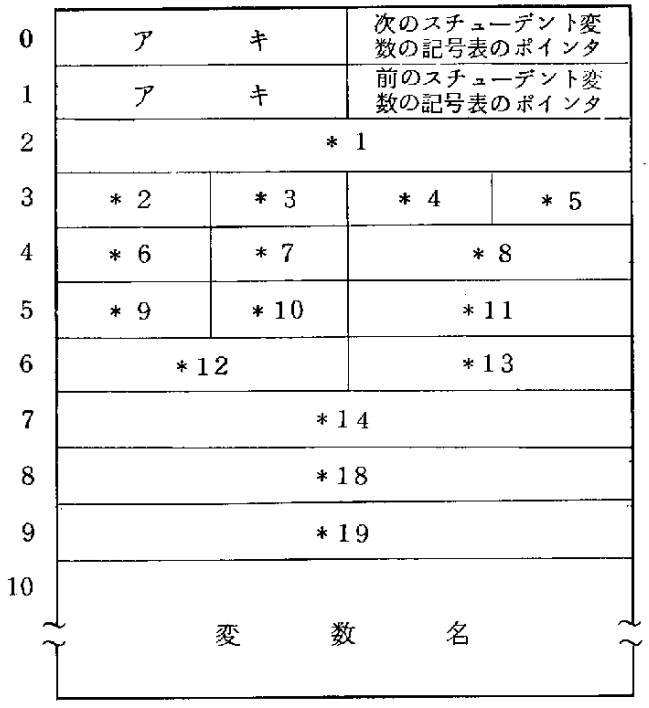
ＤＩＡＬコンパイラの特徴的な所のみ説明する。

ＤＩＡＬ言語の変数は，複雑な属性をもち，コンパイル時にすべての属性がわかっているとはかぎらない。ラン・タイムまで未定義の属性をもちこみ，ラン・タイムに次章で述べるＣＡＬＬプロセッサで未定義の属性を確定してもらう。

複雑な属性をもつ変数を含むステートメントをコンパイルするには，直接機械語におとすのではなく，インタプリタ形式のオブジェクトを作り出すのがよい。

5.6.1　記号表

各種の変数の記号表を図5.2，図5.3，図5.4，図5.5に示す。記号表は，ＣＰＬの記号表（図3.2）とほぼ同じであるので，異なる所のみ記す。



＊18　診断属性１

ビット位置	診断指定
0	＊2
1	＊3
2	＊4
3	＊5
4	＊6
5	＊7
6	＊8
7	＊9
8	＊10
9	＊11
10	＊12
11	＊13
12	＊14

＊19　診断属性２

ビット位置と診断指定は＊１の項目と一対一の対応とする。

図5.2　スチューデント変数の記号表

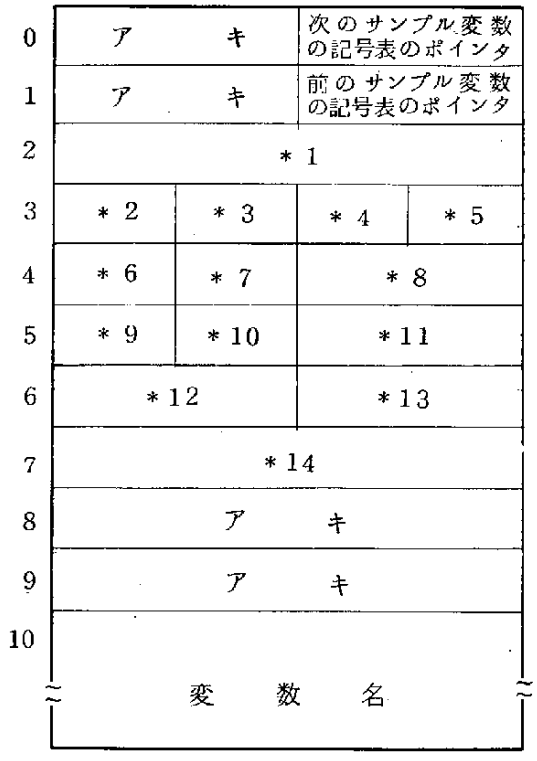


図 5.3　サンプル変数の記号表

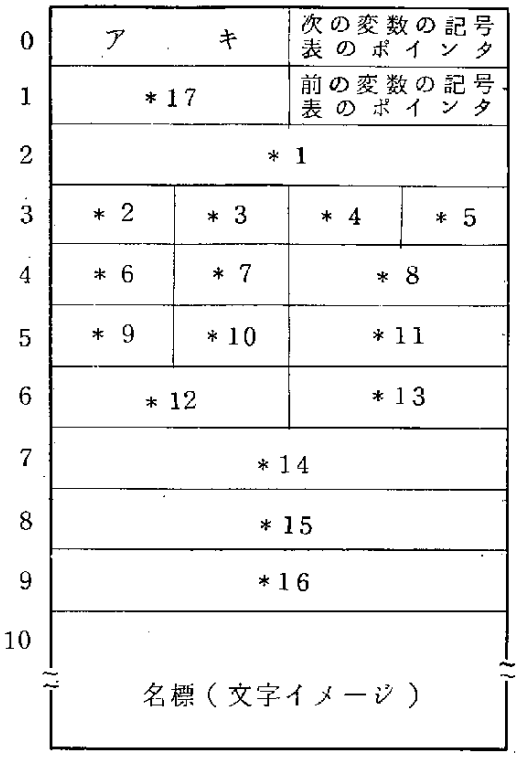


図 5.4　システム変数及びローカル変数の記号表

5.6.2　目的プログラムの構造

ソース・プログラムをコンパイラに通して翻訳された目的プログラムは，ライブラリの参照等がある場合，いちいち原プログラムに組み入れて翻訳されていない。そのため，実行可能なプログラムを作成するため，リンケージ・ローダに番地記号の特性等の情報を渡さなければならない。

【1】 番地記号の特性

目的プログラムから別のライブラリの定数文，変数文，実行文を参照したり，あるいはその逆の場合が生じる。したがって，番地特性としては，次に示す3種類のものがある。

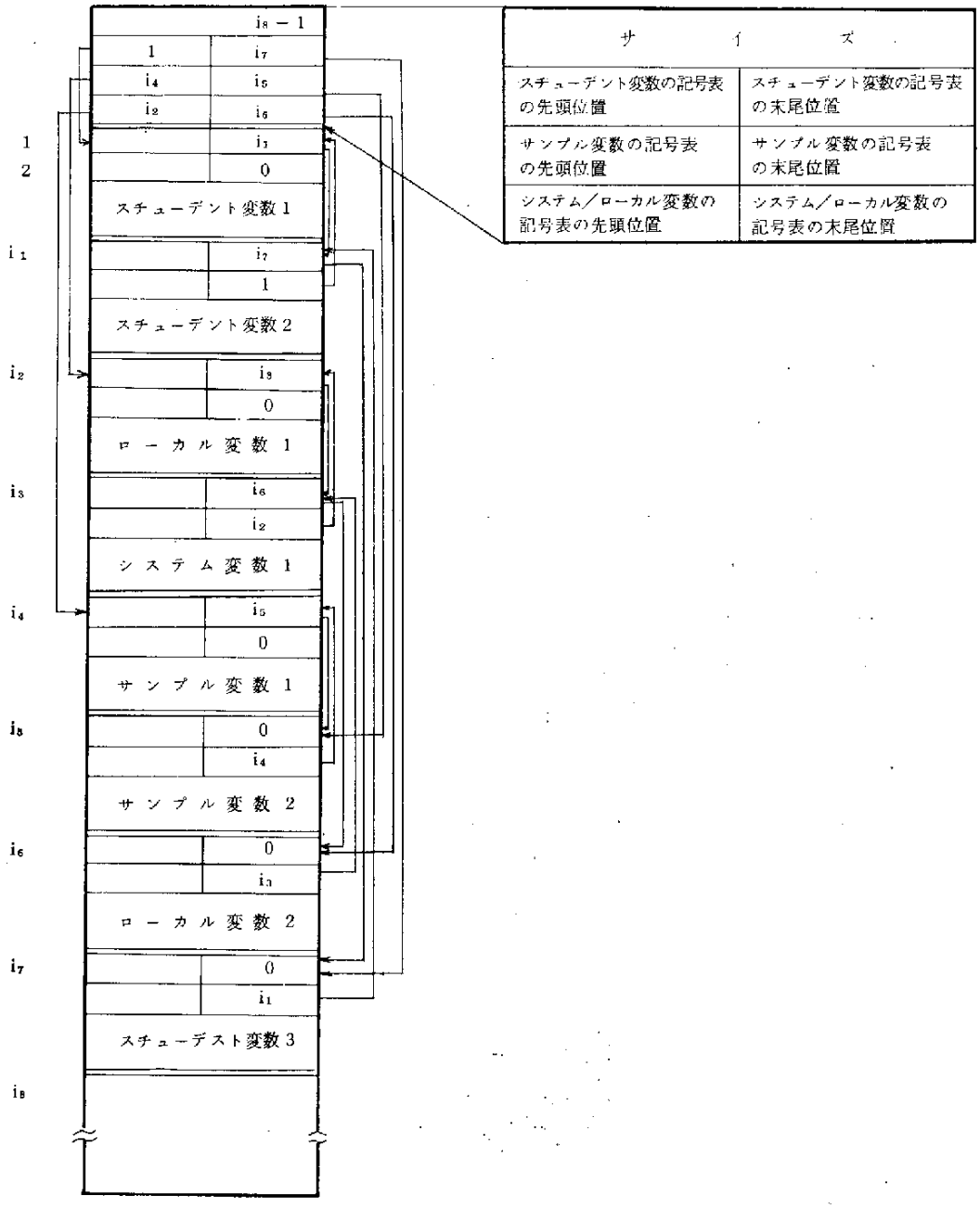


図 5.5　全体の記号表の構造

種　類	特　　性
A	プログラム内で定義され，値は絶対値をとるもの。
R	プログラム内で定義され，値はプログラムの先頭番地に相対的なもの。
C	別のプログラムで定義され，値は絶対値または，そのプログラムの先頭番地に相対的なもの。

【2】 オブジェクト・プログラムのファイル形式

オブジェクト・プログラムは大記憶上に分割型順編成ファイルとして登録される。

そのファイル形式を以下に記す。

オブジェクト・プログラムは，名前レコード，記号表レコード，辞書レコード，本文レコード，ＥＮＤレコード等から成り立つ。

各レコードは各々１８語のサイズである。又各々のレコードは同一のレコードの種類で１ブロック以上のデータを作る。ブロックのサイズは540語以下である。

(1) オブジェクト・プログラムの配列順序

図 5.6 に大記憶上の目的プログラムの配列順序を示す。

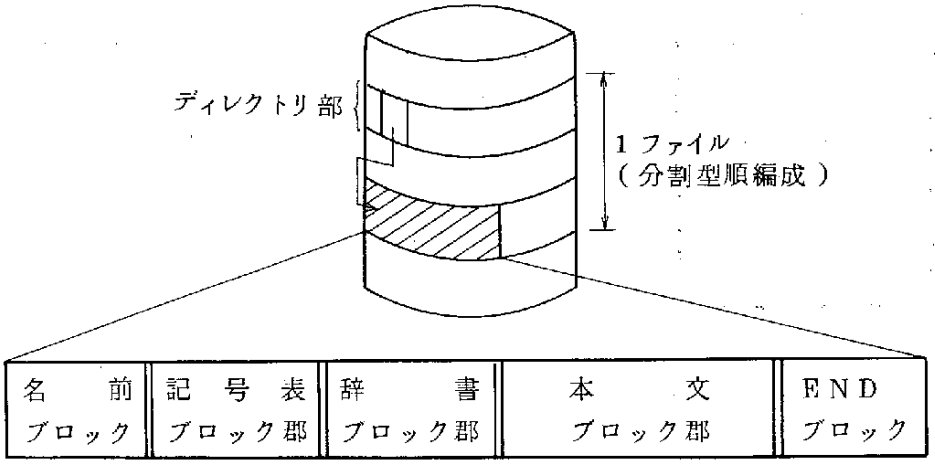
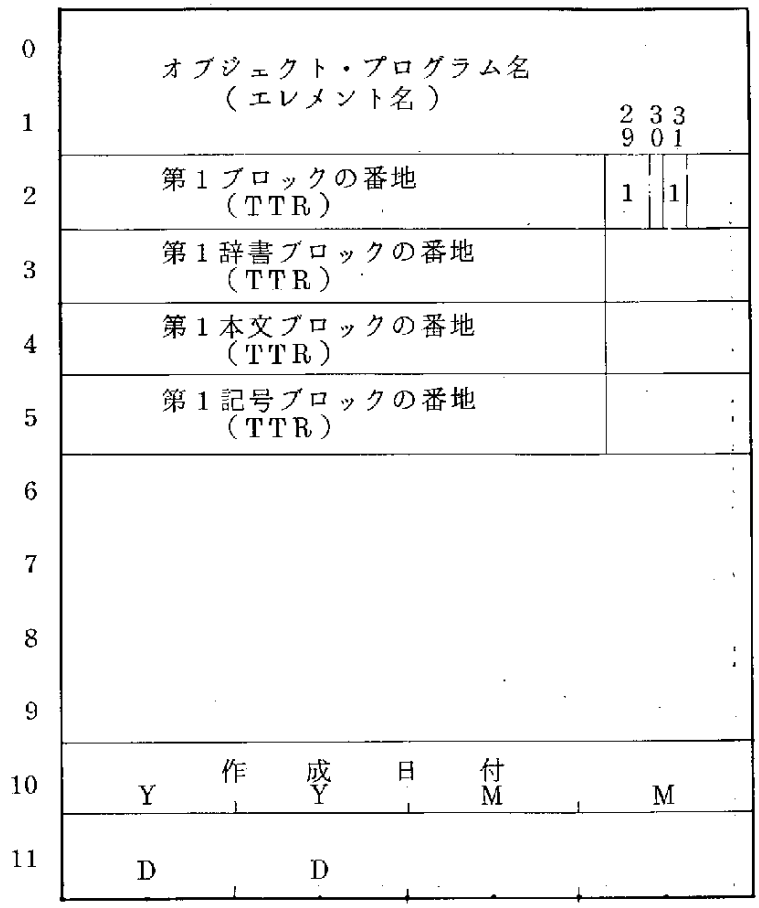


図 5.6　オブジェクト・プログラムの配列順序

(2) オブジェクト・プログラム・フアイルのデイレクトリ形式

図5.7に大記憶上のオブジェクト・プログラム・ファイル内のディレクトリ部の形式を示す。

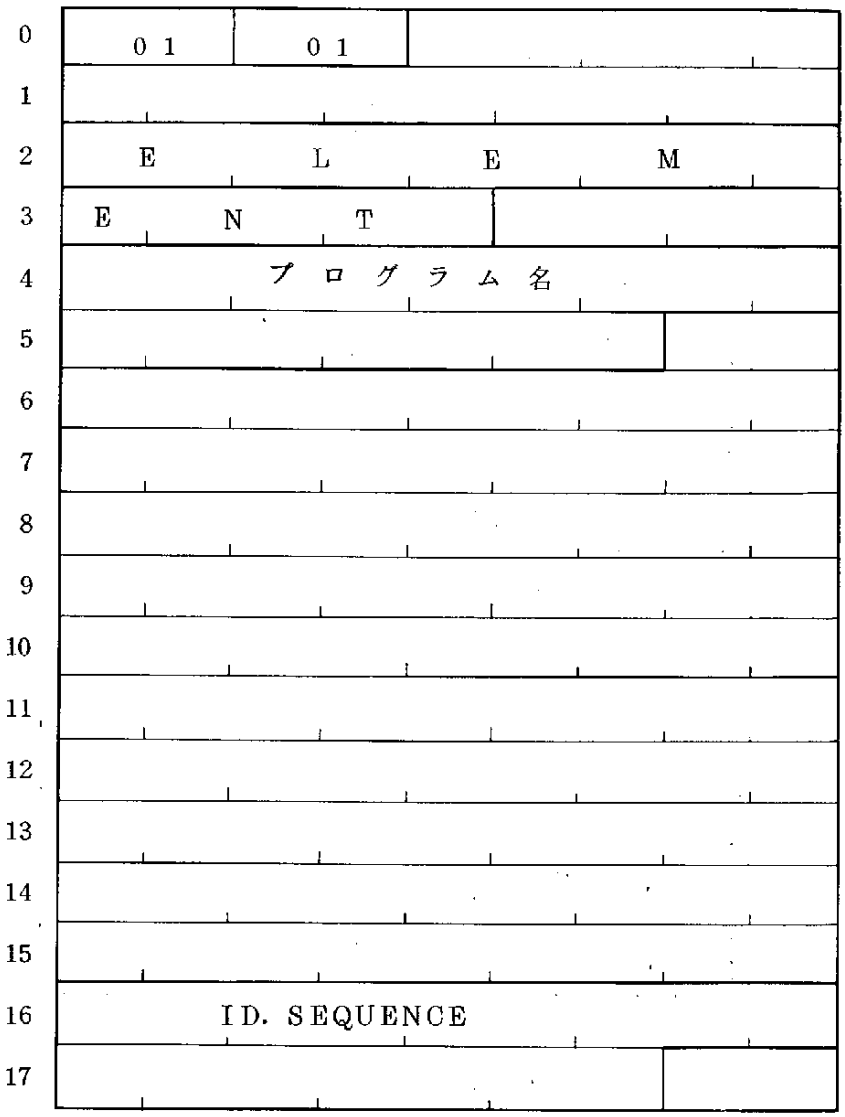


TT：ファイル上の相対トラック番地（18ビット・バイナリ表現）

R　：トラック上の相対ブロック番地（9ビット・バイナリ表現）

図5.7　オブジェクト・プログラム・フアイルのデイレクトリ形式

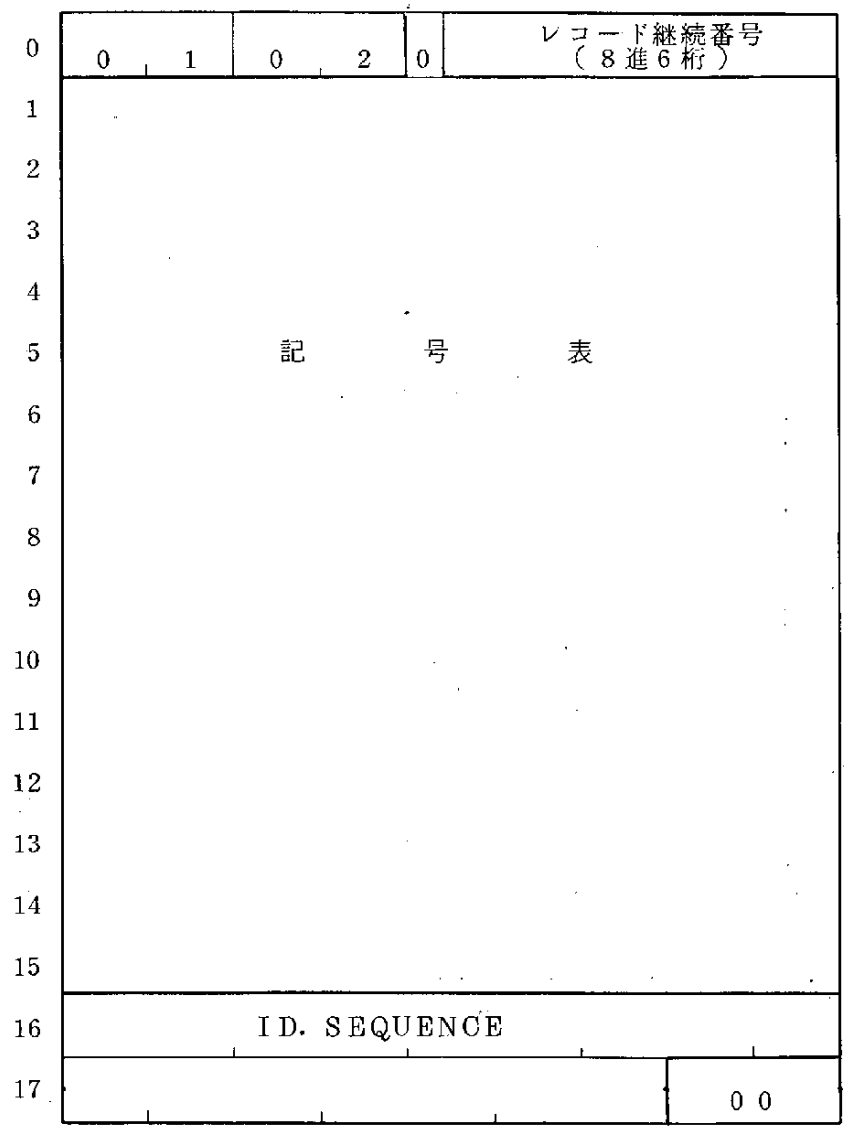
(3) 名前レコードの形式



空白はすべてEBCDICコードのブランク（40_{16}）で埋められていること。

図5.8 名前レコードの形式

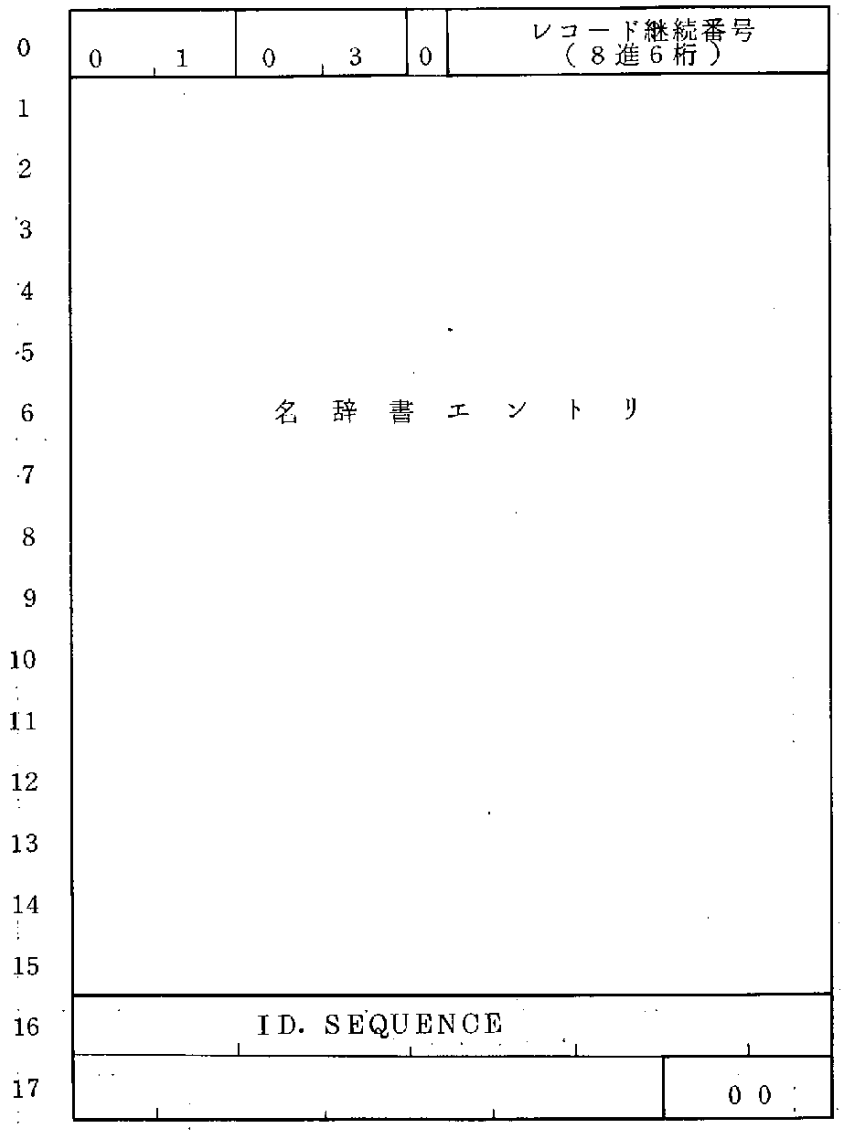
(4) 記号表レコードの形式



レコード継続番号は０からの一連番号である。

図 5.9　記号表レコードの形式

(5)　辞書レコードの形式



レコード継続番号は０からの一連番号である。

図 5.１０　辞書レコードの形式

① 辞書部分の各エントリーの詳細

1) ＥＮＴエントリ

1　0		r

このプログラムの入口を指定する。

r：実行開始入口（プログラム内相対番地）

2) ＥＸＴエントリ（システム・ルーチン名）

0　1	B	
記　号 （EBCDIC コード）		

外部参照記号を定義する（システム・ルーチン，システム変数名）。

B：割り当てられたベース・レジスタの番号を示す（Ｂ１に固定）。

3) DUMYエントリ

6　6	1　1	0　0	
記号表のサイズ			
スチュデント変数の記号表の先頭位置		スチューデント変数の記号表の末尾位置	
サンプル変数の記号表の先頭位置		サンプル変数の記号表の末尾位置	
システム／ローカル変数の記号表の先頭位置		システム／ローカル変数の記号表の末尾位置	

4) ＳＹＭエントリ

s ：記号表ブロック中の記号の数を示す。

② 辞書名エントリの順序

次の順序で出力されること。

ＥＮＴエントリ

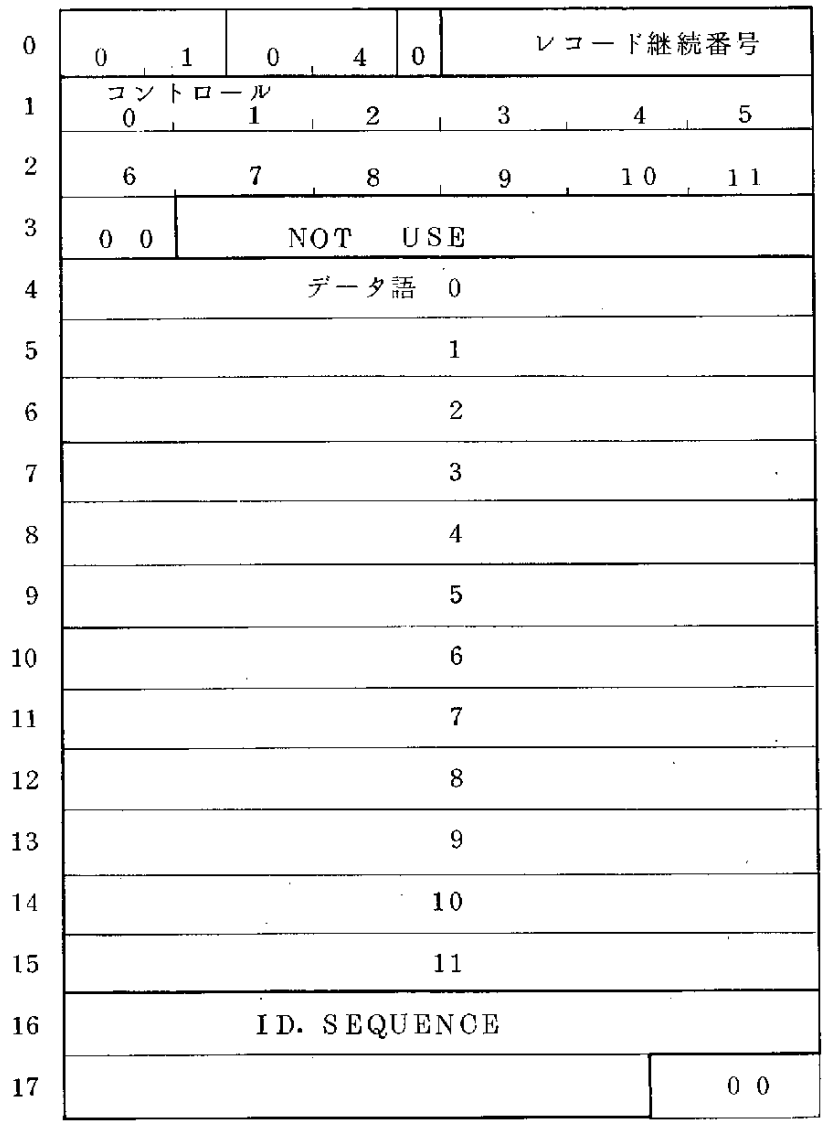
ＥＸＴエントリ

ＤＵＭエントリ

ＳＹＭエントリ

辞書索引番号はＥＮＴエントリを０として以下１，２，……と辞書全体の通し番号をふる。

(6) 本文レコードの形式



レコード継続番号は0からの一連番号である。

図5.11 本文レコードの形式

本 文 エ ン ト リ ー 覧

エントリ名	コントロール	データ語				意味
EOC	00	無 し				END OF RECORD
ORG	01	NOT USE			r	ORIGIN文に対して作られる。
SEC	02	セクション番号			r	SECTION，COMMON文に対して作られる。
RES	03	NOT USE			d	RESERVE文に対して作られる。
VA	10	t	b	a		VFD（絶対値）
VR	11	t	b		r	VFD（相対値）
VD	12	t	b		d	VFD（辞書索引）
VC	13	t	b		c	VFD（記号表索引）
A	40	a				1語データ（絶対値）
R	41	a			r	1語データ（相対値）
D	42	a			d	1語データ（辞書索引）
C	43	a			c	1語データ（記号表索引）

(7) ＥＮＤレコードの形式

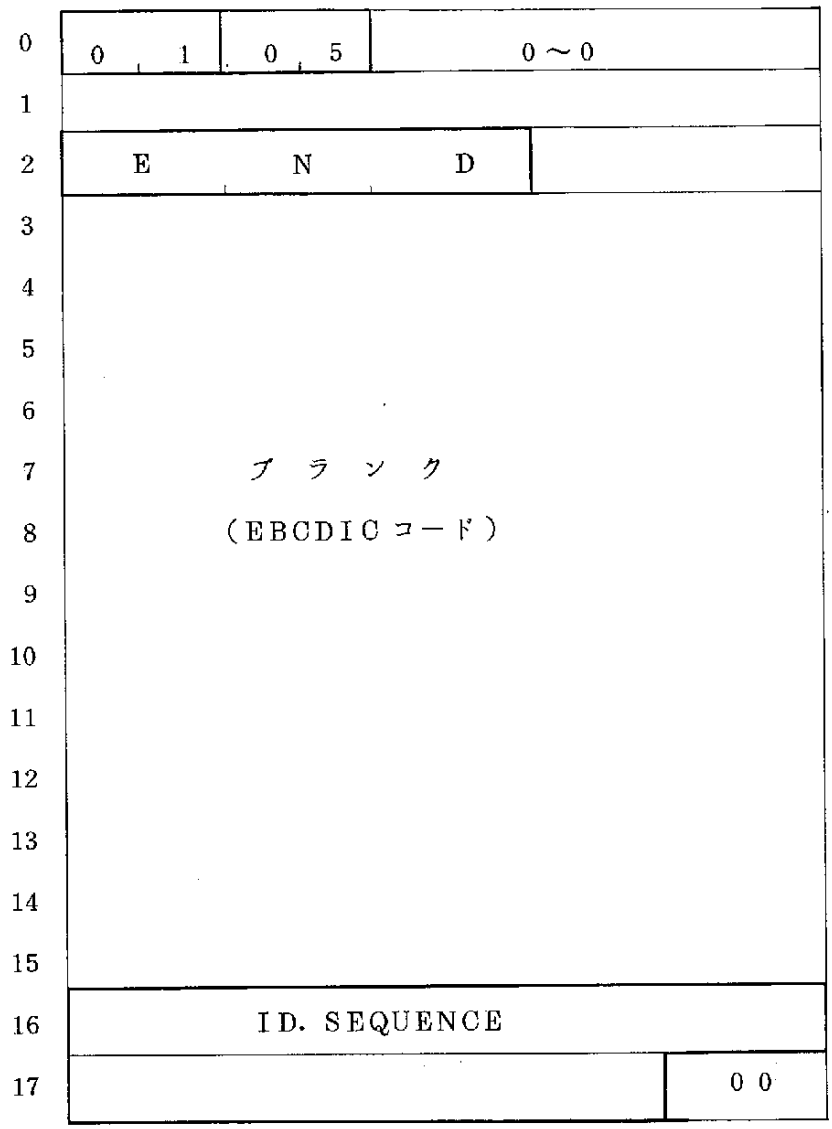


図5.12 ＥＮＤレコードの形式

第6章　ＣＡＬＬ

6.1　概　　要

CALLは，学習者の作成したプログラムの実行結果とサンプル・プログラムの実行結果（どちらも主記憶装置内に保存されている）とライブラリに登録されている診断プログラムの目的プログラムを読み込み，３つのプログラムを結合し，実行形式プログラムを作成し，インタプリートしながら実行し，プログラム診断を行なう。

診断結果によって次のモードを決定して処理を終了する。

学習者の作成したプログラムが正しければ，終了モードへ遷移するため，内部的に¥TERMシステム制御コマンドをジェネレートし，誤まりがあれば，試行回数によって学習モードに遷移したり，治療モードに遷移するため，内部的に，¥DRIL又は¥REMEシステム制御コマンドをジェネレートする。

6.2　CALLサブシステムの構成とその機能

CALLサブシステムは，大きく次の３つに分けることができる。

1)　CALLサブシステム全体を管理するルーチン……CALL制御ルーチン

2)　診断の目的プログラムの外部記号（スチューデント変数，サンプル変数，システム変数）の未定義部分を結合して，実行形式プログラムを主記憶装置上に作成するルーチン……リンケージ・ローダ・ルーチン

3)　主記憶装置上に作成された実行形式プログラムをインタプリートしながら実行するルーチン……インタプリタ・ルーチン

CALL制御ルーチンは，まず，リンケージ・ローダ・ルーチンの部分を主記憶装置にローデングし，制御権をリンケージ・ローダ・ルーチンに受渡す。リンケージ・ローダ・ルーチンは，CLASS－F 制御ルーチンから引渡された制御情報を分析，主記憶装置の初期化（入力バッファの設定や出力ロード・モジュールの収容個所の設定），リンケージ・ローダが使うファイル（診断プログラム・ライブラリ）をオープンする。

そのあとで，診断プログラム・ライブラリから相対形式プログラム（モジュール）を読み込んで処理する。本文レコードのリロケーションが終ったあとで（DIALの）記号表の外部参照記号に等しい記号をもつCPLのスチューデント・プログラムと学習者の作成したプログラムの記号表を探し出し結合する。すべての未解決外部参照記号の処理が終ると最後の処理ステップに入る。このステップで，学習者の作成したプログラム中に診断プログラム内で参照する変数がない場合，診断メッセージをディスプレイ装置に表示し，完了コードをセットする。その後で，制御権をCALL制御ルーチンに返却する。

CALL制御ルーチンは，リンケージ・ローダ・ルーチンの完了コードを検査し，正常終了ならば，リンケージ・ローダ・ルーチンが占有していた主記憶装置にオーバレイして，インタプリータ・ルーチンをロードし，完成した実行形式プログラムを実行するため，制御権をインタプリタ・ルーチンに渡す。この結果，診断プログラムの実行が開始され，診断が行なわれる。診断プログラムの実行が終了すると，制御権は，再度CALL制御ルーチンに使される。

もし，完了コードが異常終了ならば，そのむねメッセージを表示し，CALLの処理を終了する。

図6.1にCALLによる制御権の論理的な流れを示す。

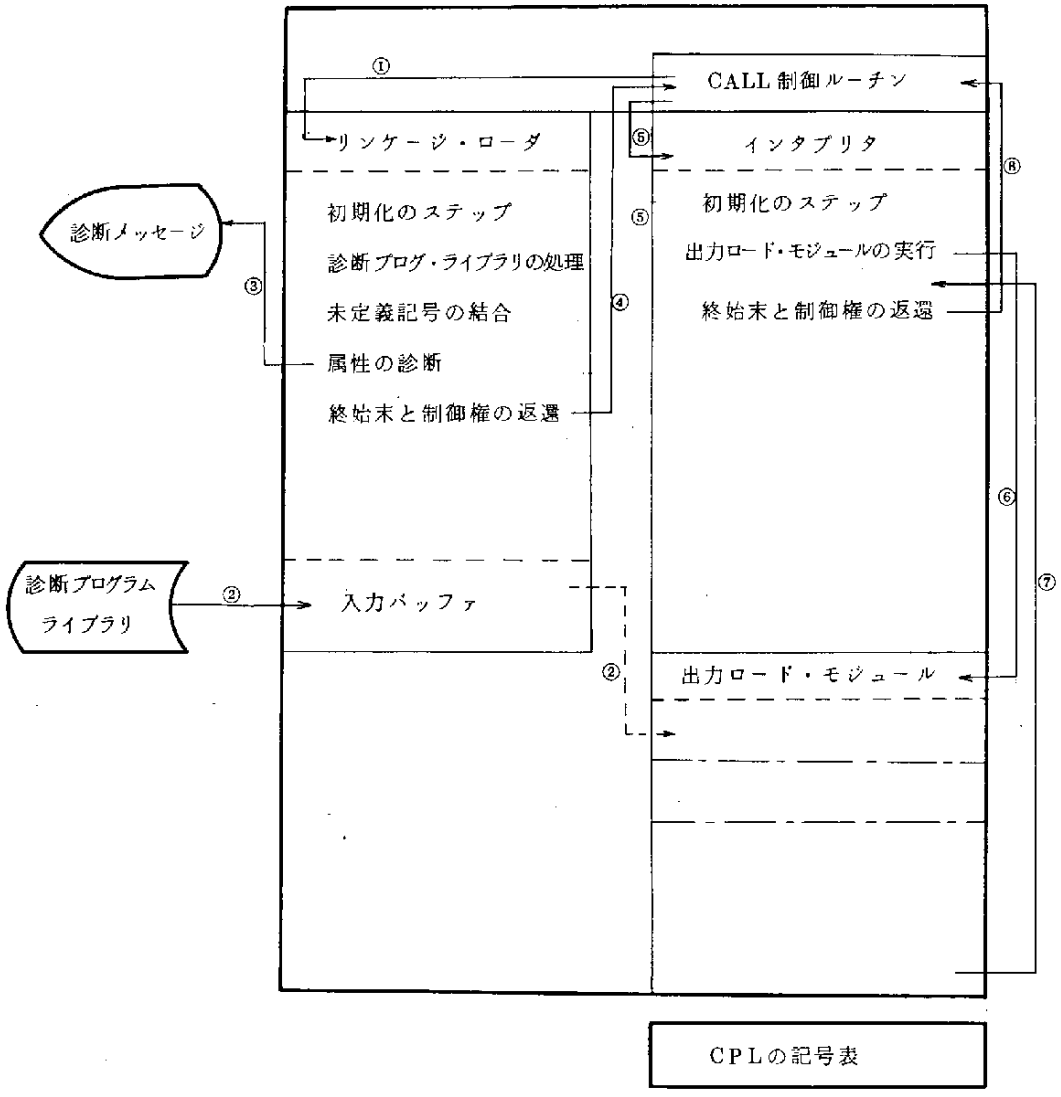


図 6.1　ＣＡＬＬによる制御権の論理的な流れ

6.3 外部記号の結合方法

例を用いて説明する。

〔例〕 IF @A=#B THEN GO TO EQUAL;

このステートメントがDIALコンパイラによってコンパイルされると図6.2に示すように@Aの番地はDIALの記号表の i_d 番地であり，CALLによって

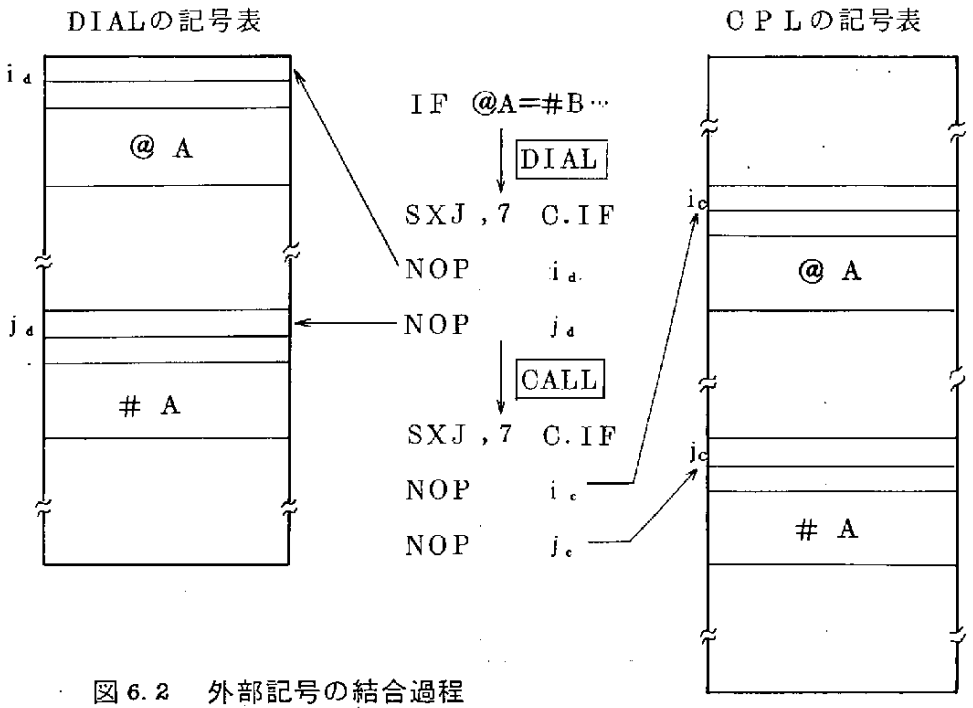


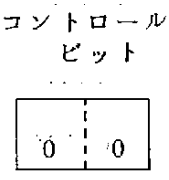
図6.2 外部記号の結合過程

実行形式プログラムに変換されるとCPLの記号表の i_c 番地になる。

6.4 リロケーション処理の詳細

以下に目的プログラム中の本文エントリに対するCALLの処理を説明する。

(1) EOCエントリ



意　味

これで，このレコードの処理を終りにする。

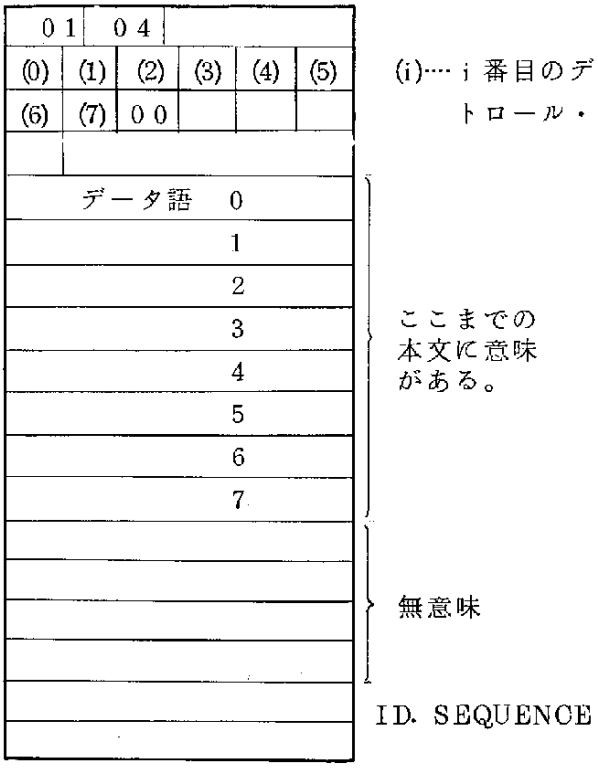
注

これには，データ語はともなわない。

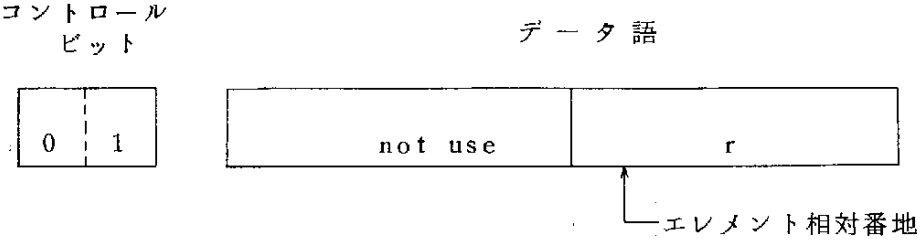
CALLの処理

このレコードに対する処理を終了する。

例）



(2)　**ORG**エントリ



意　味

ORIGN文に対して作り出されたものである。

CALLの処理

番地割当てカウンタをｒに対応する実際の値（記憶領域内）の相対番地に合わせる。

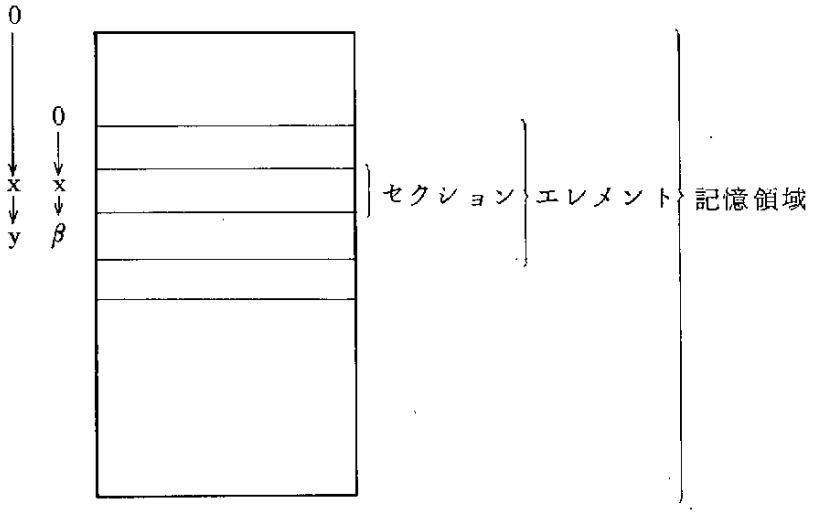
ｒに対応する実際の値とは，ｒをｒの属するセクションに相対的な値になおし，それにそのセクションの先頭番地をたしたものである。

すなわち，ｒがＸセクションに属し，Ｘに割当てられた領域が，エレメント内では〔α,β〕とあたえられ，Ｘセクションの先頭が記憶領域内では，χ番地に割当てられたとき，

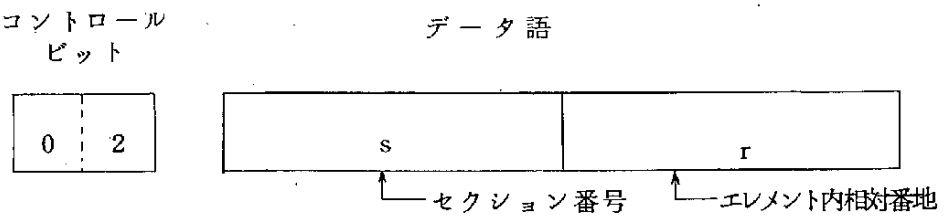
ｒは，

$$\chi + (r - \alpha)$$

に変換される。



(3)　**ＳＥＣエントリ**



意　味

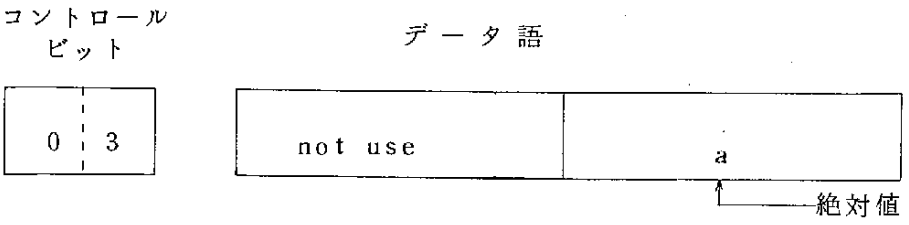
SECTION文に対して作り出されたものである。

CALLの処理

このセクションが組み込まれるのであれば，番地割当てカウンタをｒに対応する領域内相対番地に変換する。

もし，組み込まれないのであれば，このエントリから，つぎのＳＥＣエントリ，またはＣＳエントリまでのすべてのエントリが無視される。

(4)　**ＲＥＳエントリ**



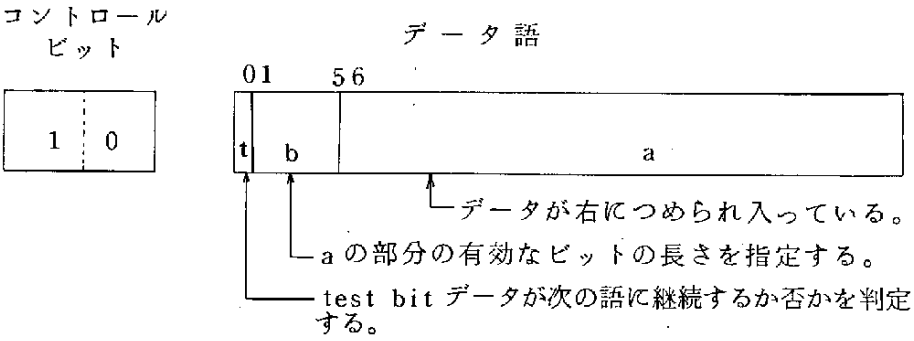
意　味

RESERVE文に対してつくり出されるものである。

CALLの処理

出力ブロックをここで切り，次の出力ブロックのORIGIN を現在の番地割当てカウンタ＋ａとする。

(5)　**ＶＡエントリ**



t＝0 …継続する。

t＝1 …継続しない。この語で終り。

0 ≦ b ≦ 30 （10進）

意　味

ＶＦＤ文に対してつくり出されるものである。

〔例〕　VFD　1／1，12／A，15／C－3，8／R－S

但し，オペランド欄の分母のS.E（シンボリック・エクスプレッション）のV.P（Value Property）がアブソリュート（絶対値）のものに対してつくり出される。

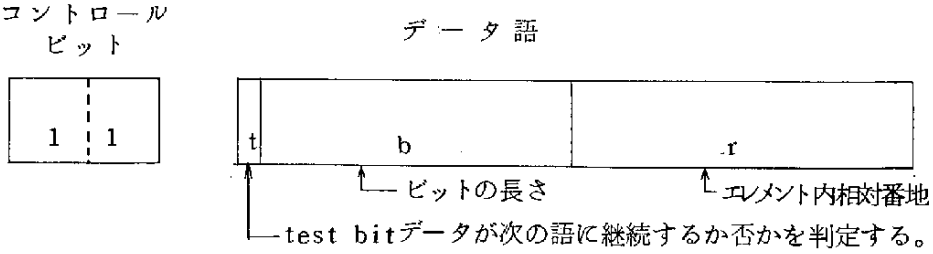
CALLの処理

ＶＦＤ文によって指定された通りに語のビット構成を完成させる。つまり，aにあるデータの右ビットを現在構成中の語につけ加える。

ｔ＝１で，現在構成中の語にｂビットをつけ加えてもなお１語が完成されていない場合は，残りの右ビットに０がつめられる。

次頁以下に述べるVR，VD，VC エントリとともに，１語または，数語を構成してもよい。

(6)　**ＶＲエントリ**



意　味

ＶＦＤ文に対してつくり出されるものである。

〔例〕　VFD　18／R，18／S－3

但し，オペランド欄の分母のシンボリック・エクスプレッションのヴァリュー・プロパティが，リラティブ（相対値）のものに対してつくり出される。

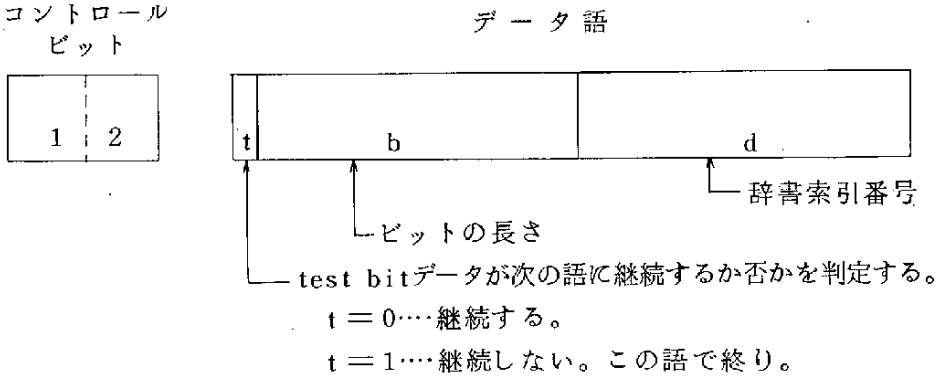
CALLの処理

ＶＦＤ文によって指定された通りに，語のビット構成を完成させる。

rを領域内相対番地に変換し，bビットの長さのデータとする。データは右につめられるものとする。

bの指定が１８より小ならば，データの上位ビットがなくなり，大ならば，上位に０がおぎなわれる。

(7) **ＶＤエントリ**



意　味

ＶＦＤ文に対してつくり出されるものである。

〔例〕　ＶＦＤ　１８／Ｅ１，１８／Ｅ２

但し，オペランド欄の分母の記号が，他のエレメントで定義されているもの，つまり，外部記号であるものに対してつくり出される。

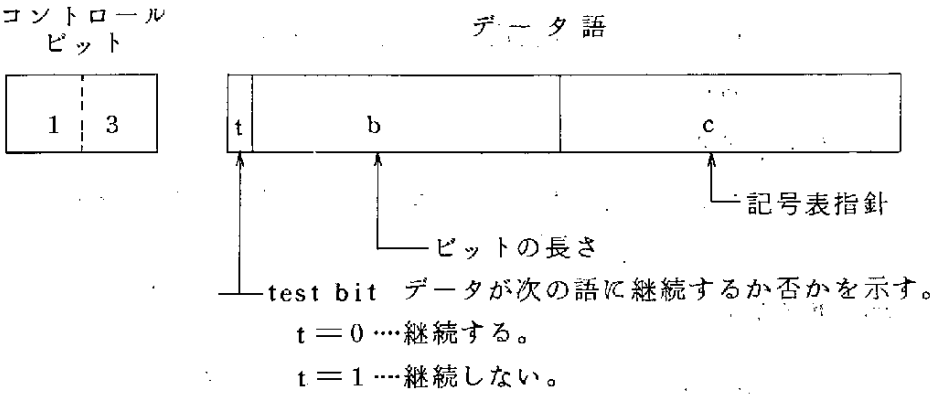
CALLの処理

ＶＦＤ文によって指定された通りに，語のビット構成を完成させる。

dは辞書索引番号である。この番号に対応する外部記号を，それが定義されている領域の領域内相対番地に変換し，bビットの長さのデータとする。

bの指定が１８ビットより小さいならば，データの上位ビットがなくなり，大ならば上位に０がおぎなわれる。

(8) **ＶＣエントリ**



意　味

ＶＦＤ文に対してつくり出されるものである。

〔例〕　VFD　18／SAMPLE，18／STUDENT

但し，オペランド欄の分母の記号が他のサンプル・プログラム又は，スチューデント・プログラムで定義されているもの。つまり，サンプル変数名，スチューデント変数名であるものに対してつくり出される。

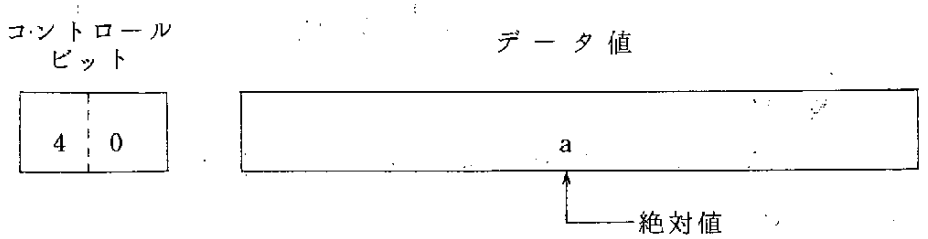
CALLの処理

ＶＦＤ文によって指定された通りに語のビット構成を完成させる。

ｃは記号表指針である。この指針に対応する記号（サンプル変数名，スチューデント変数名）をそれが定義されている領域の領域内相対番地に変換し，ｂビットの長さのデータとする。

ｂの指定が１８ビットより小さいならば，データの上位ビットがなくなり，大ならば上位に０がおぎなわれる。

(9) **Ａエントリ**



意　味

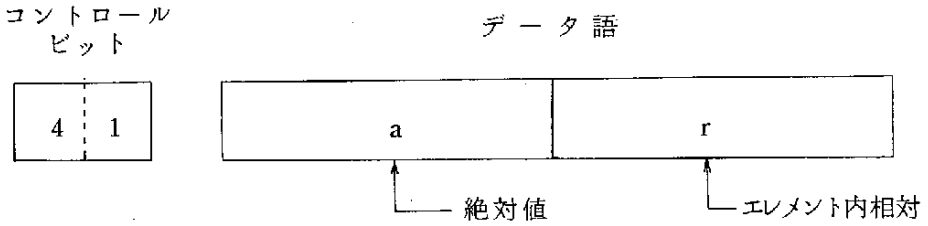
語のビット構成が完成していることを示す。

〔例〕　LXI，4　　5

CALLの処理

何ら処理を施さず，そのまま，実行形式プログラムを構成する1語として出力する。

(10)　Rエントリ



意　味

同じエレメント内を参照している文であることを示す。

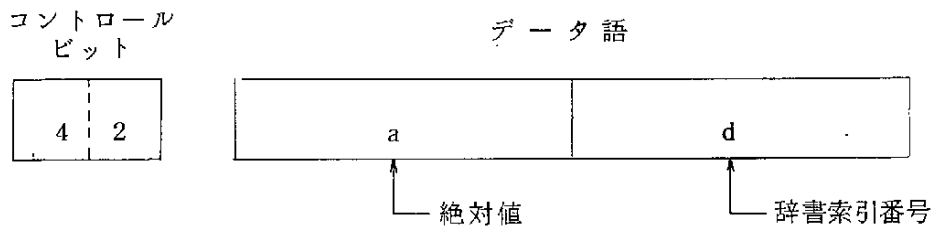
〔例〕　LA　　R
　　J　　＊＋6

Rはエレメント内で定義されている記号

CALLの処理

rを領域内相対番地に変換する。

(11)　Dエントリ



意　味

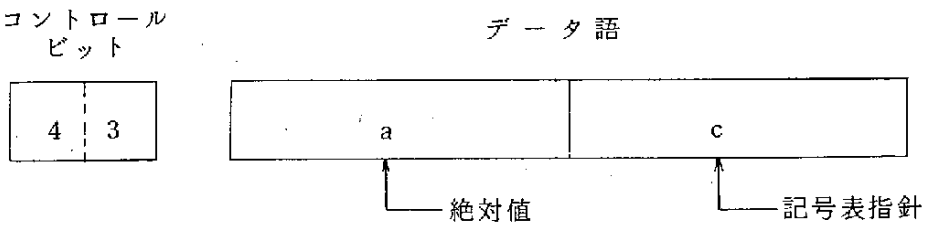
他のエレメントで定義されている記号を参照している文であることを示す。

〔例〕　EXTERNAL　　SUB

CALLの処理

dをそれが定義されている，領域の領域内相対番地に変換する。

⑿ **Cエントリ**



意　味

サンプル・プログラム又は，スチューデント・プログラム内で定義されている記号を参照している文であることを示す。

〔例〕　TE　#SAMPLE

SAMPLE はサンプル変数である。

CALLの処理

Cをそれが定義されている領域内相対番地に変換する。

第7章　学習端末装置

7.1 概　要

本システムの学習端末としては，キャラクタ・ディスプレイ装置を使用している。

この装置は，タイプライターとくらべ，印字（表示）速度は約４００倍であり，学習効果をさまたげる騒音もなく，特に電子計算機のプログラミング教育の場合，文字修正が簡単におこなえ便利である。

欠点としては，ブラウン管に文字が表示されるので，学習経過を保存することができない。本システムでは，生徒が希望すれば，学習経過記録をタイプライター装置に印刷することができる。

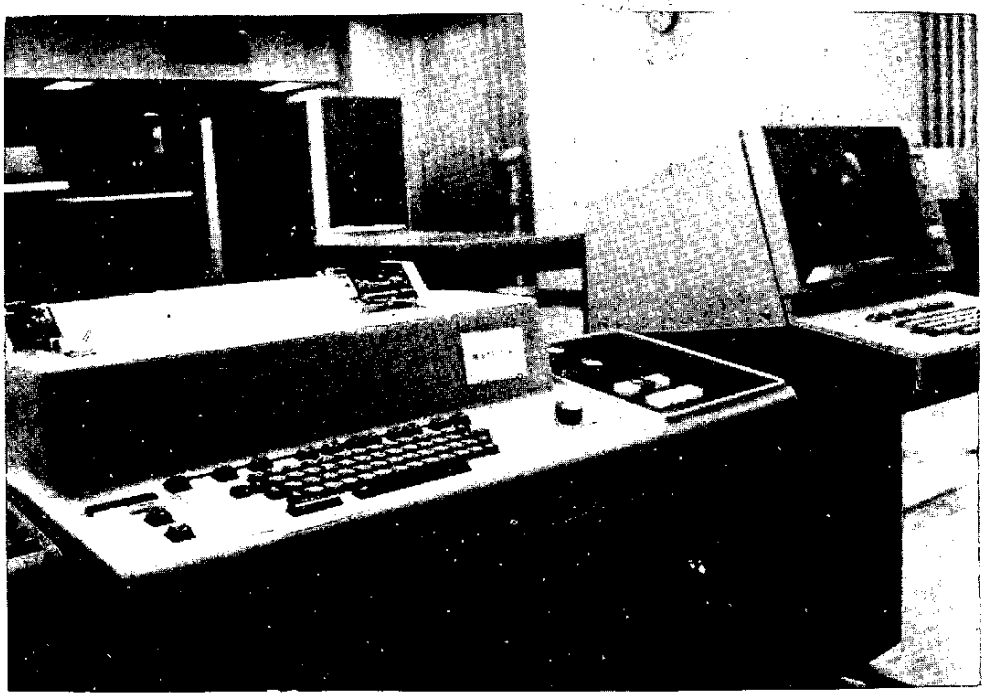


図7.1　学習端末装置

（右：キャラクタ・ディスプレイ装置，左：タイプライター装置）

7.2 仕　　様

本装置の規格は次の通りである。

ブラウン管	１２インチ形
表示面積	１９０×１４０（mm^2）
表示字数	５０字×２０行（１００字）
文字の大きさ	４㎜（高）×２.８㎜（幅）
表示フレーム数	毎秒約４０フレーム
表示色	緑色
字種	１２８種（カタカナ，アルファベット，数字，記号，スペース） 他特殊記号　４種
コード	ＩＳＯコード
情報転送速度	ＣＰＵ－ディスプレイ間５０ｍ以内： 約４Ｋ字／秒または約1.５Ｋ字／秒 ＣＰＵ－ディスプレイ間５００ｍ以内： 約1.５Ｋ字／秒
誤り検出方式	垂直パリティチエック
誤り訂正方式	自動再送（計算機からの命令による）
電源	ＡＣ単相，１００Ｖ±１０Ｖ

7.3 機　　能

本装置は次の機能を有する。

(a) 打鍵表示
(b) 編集機能
(c) 受信データの表示
(d) 表示データの送出
(e) スプリット・スクリーン
(f) アドレッシング
(g) 部分転送

7.4　操作釦および基本操作手順

7.4.1　操作釦およびランプ

表示部の操作釦およびランプの配置は，図7.2の通りである。

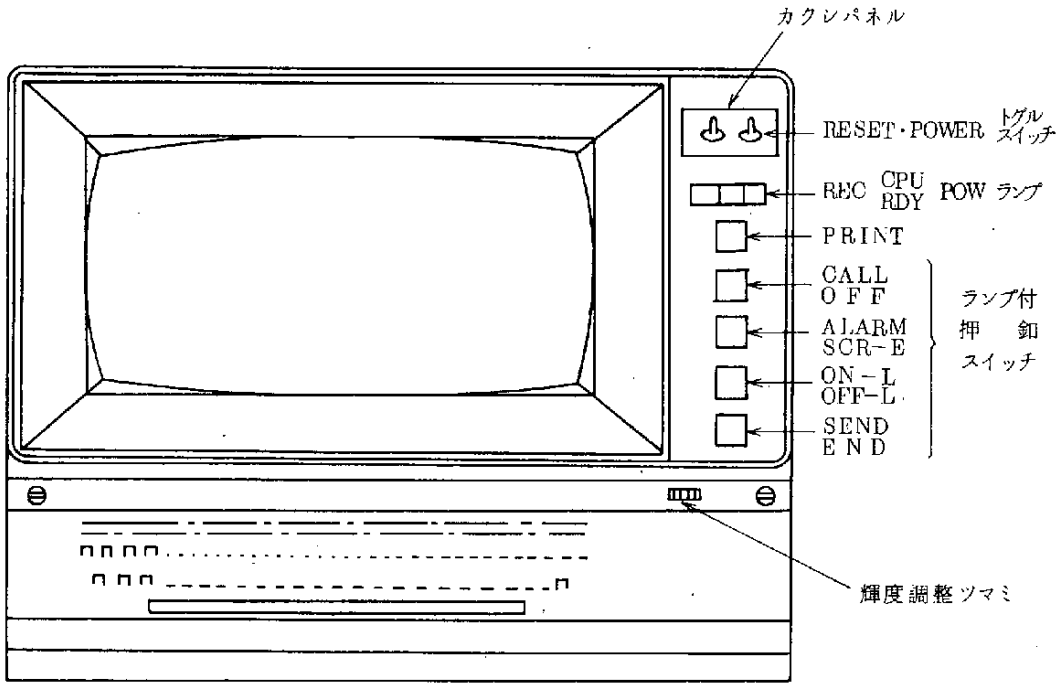


図7.2　操作釦およびランプ配置図

7.4.2　電　源　投　入

本装置の電源釦は，表示部カクシパネル内にあり，表示部の電源をオンにすることにより，電源が入る。この時のランプ点灯状態は次の通り。

- ＰＯＷ……電源投入直後に点灯する。
- ON−L……準備完了で点灯する。
- CPU−RDY……ＣＰＵ側が準備完了状態の場合点灯する。

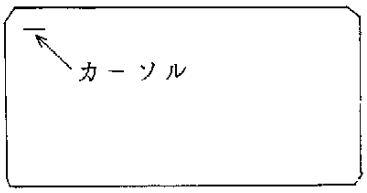


図 7.3　電源投入直後の表示面

7.4.3　打鍵操作

オフライン・モード（OFF−Lランプ点灯中）のとき，学習者は鍵盤を操作できる。オンライン中はキーボードはロックされる。

鍵盤は，文字キー，シフト・キー，後退キー，復改キー，およびエディト用押釦を用する（図7.4参照）。

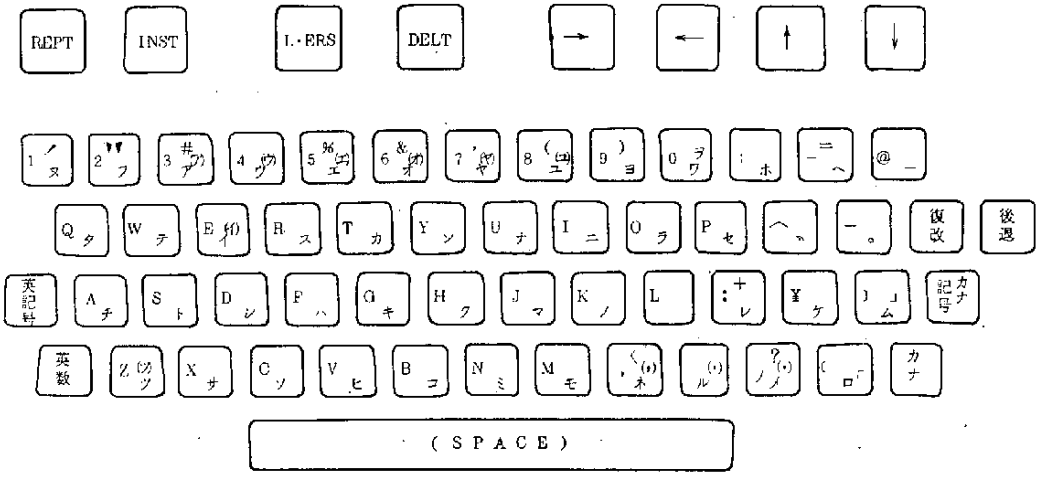


図 7.4　鍵 盤 配 列 図

(1) 文字キー

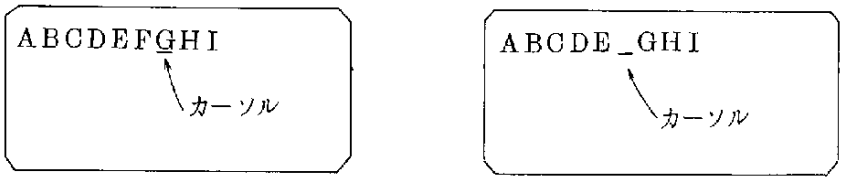
打鍵可（キーロックされていない状態）の状態において，文字キーを打鍵すると，そのシフト状態において打鍵内容が，表示面のカーソルの位置に表示され，カーソルは次の文字の位置に移動する。

(2) シフト・キー

キー・ボードには，英記号，英数，カナ，カナ記号の4つのシフト・キーを有する。

(3) 後退キー

後退キーを打鍵することにより，カーソルのある位置より1文字前の文字を消去し，カーソルは消去した文字の下に移動する。この時，前にあったカーソル上の文字は何等変化しない（図7.5参照）

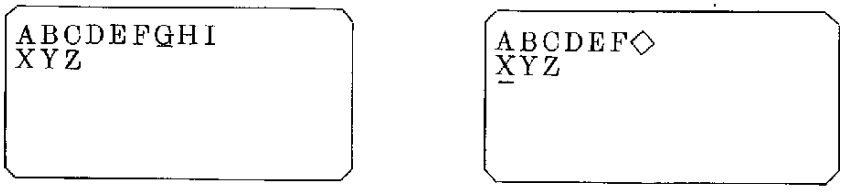


(a) 後退キー打鍵前　　(b) 後退キー打鍵後

図7.5 後退キー打鍵

(4) 復改キー

復改キーを打鍵することにより，カーソルの位置に「◇」の記号を表示し，その後の終りまで文字を消去し，カーソルは次の行の第1文字目に移動する（図7.6参照）。



(a) 復改キー打鍵前　　(b) 復改キー打鍵後

図7.6 復退キー打鍵

7.4.4　ファンクション釦操作

鍵盤部には打鍵キーの他にカーソル釦（→，←，↑，↓），レピート釦（REPT），インサート釦（INST），デリート釦（DELT），行消去釦（LERS）を有する。

(1)　カーソル釦

カーソル移動用釦により，カーソルを移動させる場合，ブラウン管に表示されている内容に関係なく移動させることができる。

(a)　スプリット・スクリーンがない場合

画面上どの位置にでも自由に移動させることができ，その表示内容は変らない。

(b)　スプリット・スクリーンがある場合

カーソルはEDIT領域のみ移動することができ，ＣＰＵ領域に入る事はない。

(c)　カーソルの連続移動

カーソル移動用釦を押下し，その後連続していると約1000字／分のスピードでカーソルを移動させることができる。

カーソルの連続移動の開始時間はカーソル移動用釦押下後約1秒たってからである。

（注）　レピート釦は，カーソルの移動には関係しない。

(イ)　カーソル・アドバンス……→

本釦押下により，カーソルは1ステップ（1文字）右に移動する。連続して押下している事により連続移動する。

(ロ)　カーソル・バック……←

本釦押下により，カーソルは1ステップ左に移動する。連続して押下している事により連続移動する。

(ハ)　カーソル・ライン・フィード……↓

本釦押下により，カーソルは1行下に移動する。連続して押下している事により連続移動する。

(ニ) カーソル・リバース・ライン・フィード…… ↑

本釦押下により，カーソルは１行上に移動する。連続して押下している事により連続移動する。

(2) レピート釦 (REPT)

鍵盤上から同一文字を連続して入れる場合，レピート釦（REPT 釦）を押下し，文字キーを打鍵する事により行なうことができる。

レピート釦は文字キーの時のみ有効であり，復改キー，シフトキー，後退キーに対しては，何等作用しない。

(イ) レピート開始および速度

打鍵可の状態において，レピート釦を押下し，文字キーを打鍵することにより，レピートを開始する。

連続して文字を書込む速度は約1000字／分である。

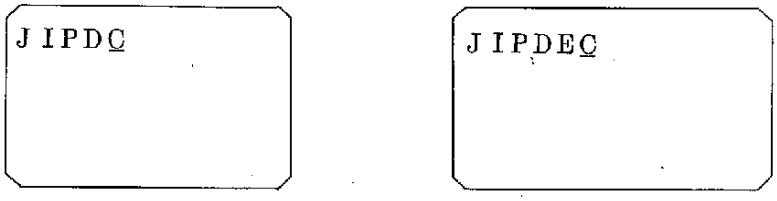
レピート作動中は，キーロックを行なう。

(ロ) レピートの終了

1000字までレピートを行なった時，カーソルは1000字目に残りキーロックを解除する。

(3) １字挿入釦 (INST)

打鍵可の状態において，挿入釦（INST 釦）を押下し，文字キーを打鍵することにより，カーソルの位置に打鍵された内容が入りカーソルは次の位置に移動する（図7.7）。



(a) 挿 入 前　　(b) 挿 入 後

図7.7 〝E〟の挿入

(イ) ＬＦ（◇）記号がある場合（図7.8）

カーソルの位置に，ＬＦ（◇）記号がある場合，挿入動作はカーソ

ルの位置によりＬＦまでである。

ただしＬＦ記号も１文字移動する。

(a) 挿　入　前　　(b) 挿　入　後

図7.8　〝Ｅ〟の挿入

ＬＦ記号が５０文字目（行の最後の位置）にある場合，ＬＦの位置に１文字前の内容が入り，ＬＦ記号は消える。

ただし，挿入は５０文字目で終了するが，続いて挿入を行なった場合は，ＬＦのあった次の行に対しても挿入動作を行なう。

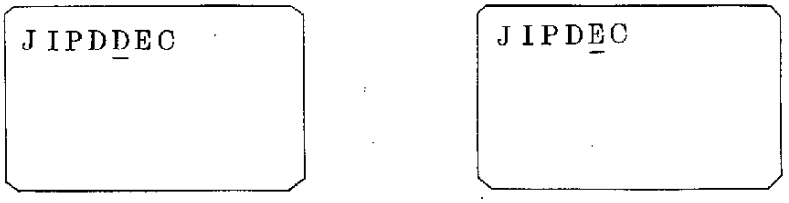
(ロ) カーソル以後にＬＦがない場合

この場合は，１０００文字目まで挿入動作を行ない，１０００文字目の文字はオーバーフローする。

(4) １字削除釦（**ＤＥＬＴ**）

打鍵可の状態において，削除（ＤＥＬＴ）釦を押下する事により，カーソルの位置の文字が消去され，カーソル以下の文字が，１文字分前に移る（図7.9）。

削除した後のカーソルの位置は変らず，次々と押下することにより，続けて削除することができる。



(a) 削　除　前　　(b) 削　除　後

図7.9　削除釦押下

DELT釦押下により，カーソル上の文字が消去され，カーソル上の文字が消去され，カーソル以後の文字は１文字分前に移動するが，削除動作の行なう範囲は次の通りである。

(イ) ＬＦ（◇）起号がある場合（図7.１０）

カーソルの位置以後にＬＦ（◇）記号がある場合，削除動作はカーソルの位置よりＬＦまでである。

ただし，ＬＦも１文字分前に移動する。

(a) 削　除　前　　(b) 削　除　後

図7.１０　削除釦押下（ＬＦがある場合）

(ロ) カーソル以後にＬＦがない場合

この場合，１０００文字まで削除動作を行ない１０００文字目にはスペース・コードが入る。

(5) 行消去釦 (L'ERS)

打鍵可の状態において，行消去釦を押下する事により，カーソルのある位置よりその行の終りまで消去し，カーソルはもとの位置に残る（カーソル上の文字も消去する）。

7.4.5 制　御

(1) ALARM/SCR.E釦

説明省略

(2) ON-L/OFF-L (ON-LINE/OFF-LINE)

オンライン・オフライン切換釦であり，オンライン状態（ON-L ランプ点灯中）のとき，ＣＰＵからのデータをいつでも受信表示することができる。

ディスプレイ装置をＣＰＵから切離して，打鍵表示するときは，オフラ

イン状態（OFF—L 点灯）にする。

この時からデータはすべて受付けられない。

(3) **SEND／END釦**

表示画面上に表示されている内容をＣＰＵに送出するとき使用する。本釦は，オフライン状態のときのみ押下でき，オフライン状態で押下した場合，押下したと同時にオンライン状態となる。

SEND釦押下により，SENDランプ点灯中は，ＣＰＵからデータを受付けることも，打鍵表示することもできない。

(4) **CALL OFF釦**

説明省略

(5) 表示ランプ

表示部押釦上部にPOW，CPU，RDY，RECの三つのランプを存する。

(a) ＰＯＷ……電源，ＯＮ状態である事を示す。

(b) CPU RDY……ＣＰＵ側でディスプレイとのデータの送受信が可能なとき点灯する。

(c) ＲＥＣ……オンライン状態でＣＰＵよりデータを受信中点灯する。本ランプ点灯中はキーロックされる。

7.5 制御コード及びメッセージ形式

7.5.1 制御コード

ディスプレイ装置とＣＰＵとのデータ転送およびキーインによるデータ作成中等におけるプログラム制御としては，ＣＰＵからの画面消去，アドレッシング，スプリット・スクリーン等がある。

(1) 画面消去 (**FE**) ……ＣＰＵからのみ制御可能

ＣＰＵからＦＥコードを受信すると，今まで表示画面上に表示されていたＣＰＵ領域の内容をすべて消去し，カーソルは原点に復帰する。

ただし，表示画面上，スプリット・スクリーンで分割されていない場合は，表示内容はすべて消去する。

(2) スプリット・スクリーン (SS) ……ＣＰＵからのみ制御可能

ＣＰＵより任意の行位置にＳＳを書込むことができ，表示画面を２つに分割することができる。またＣＰＵよりＳＳを消すことができる。

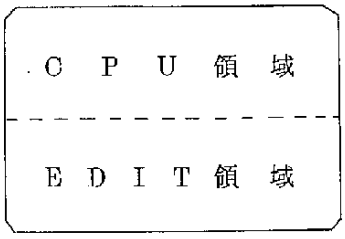


図 7.11　スプリット・スクリーンによる分割

ＳＳを書込む事により次の機能を有する。

(a) ＳＳより上をＣＰＵ領域，下をEDIT領域とし分割する。

(b) カーソルがEDIT領域にある場合，学習者はＣＰＵ領域にカーソルを移すことはできない。

この機能により，ＣＰＵ領域は学習者によって書き替えられることはない。

(c) SEND釦押下によるＣＰＵへの送信は，カーソルがEDIT領域にある場合，EDIT領域内のデータが送信される。

第8章　SCROLL エディテング

8.1 概　　要

学習端末を経由してプログラムを作成している場合，プログラムの任意の部分を見たくなるときがある。学習端末としてキャラクタ・ディスプレイ装置を使用しているので，タイプライターのようにプリント用紙をたぐって前の方の印刷内容を見ることが簡単にできない。そこでこのシステムでは，学習者が希望するプログラムの任意の部分を学習端末に表示する機能を有する。

8.2 SCROLL

ディスプレイ画面の大きさは２０行分しかないので，いつも全部のプログラムを表示することができない。プログラムが一種の絵巻物に書かれていると見做して，いつでも，２つの心棒（リール）に巻かれた絵巻物の視野の部分のみディスプレイ画面に表示する。絵巻物のことをSCROLL，視野の部分をフレームと呼ぶことにする。

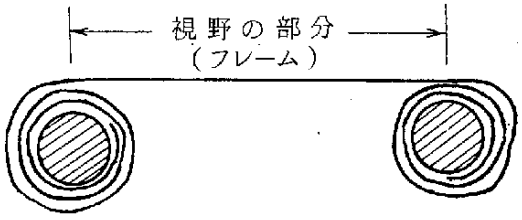


図8.1　SCROLLとフレーム

8.3 SCROLLの位置づけ

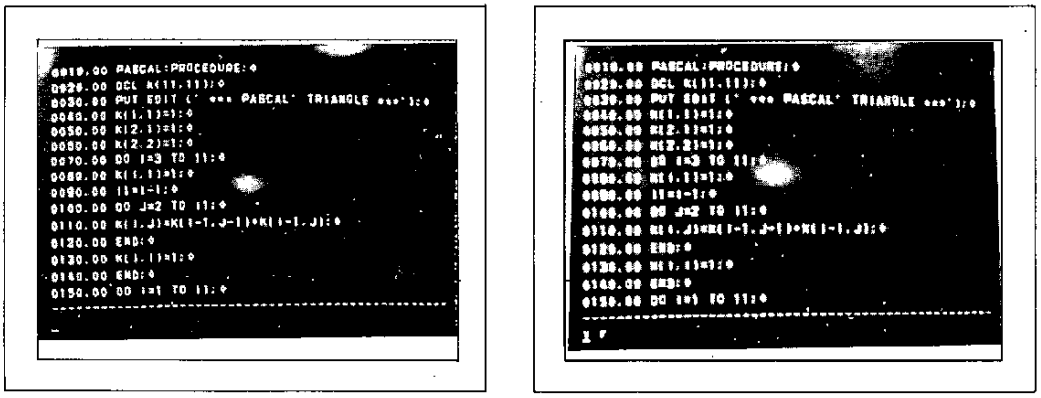
学習者はSCROLLをどちらの方向にも動かすことができ，異なったフレームをディスプレイ画面に表示することができる。その動作は，SCROLLコマンドによって指令され，SCROLLを前進させたり，後進させたり，あるいは１フレーム又は１行だけ動かすこともできる。

一般に，画面に表示されるフレームのおおまかな調整には，ライン番号が使

われ，微妙な調整にはSCROLLコマンドが使われる。

8.3.1 連続的なSCROLLの移動

SCROLLコマンド'∧'又は'V'をタイプインすると，現在表示されているSCROLLのフレームが1ステートメントだけ前後進する。一般に1ステートメントは，1行からなるので，1行だけ前後進する。

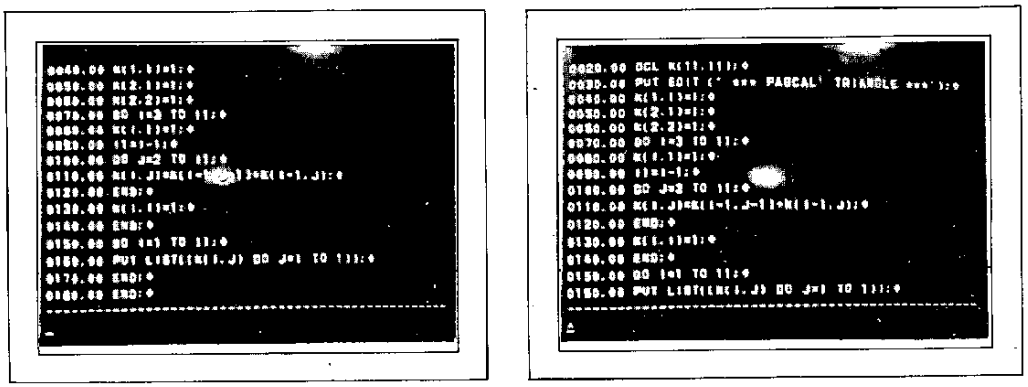


(1) 現在のフレーム　　(2)'∧'入力後のフレーム

図8.2　1ステートメント（1行）前進の例

図8.2の(1)で'∧'を入力すると，現在のフレームの末尾行のライン番号は150であるので，次のライン番号160が新たに表示され，フレームの先頭行のライン番号10のステートメントがフレームから消える。

SCROLLコマンド'∧△F'又は'V△F'をタイプインすると，SCROLLが1フレーム分だけ前後進する（△は0個以上の空白を表わす）。



(1) 現在のフレーム　　(2)'ＶＦ'入力後のフレーム

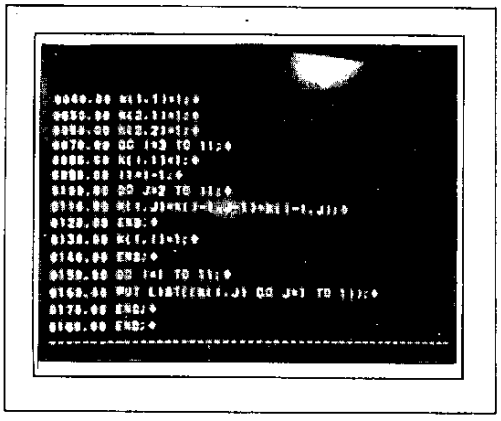
図8.3　１フレーム後退の例

例えば，図8.3の(1)で'ＶＦ'をタイプインすると，ライン番号４０より前のＳＣＲＯＬＬの１フレーム分が画面に表示される。

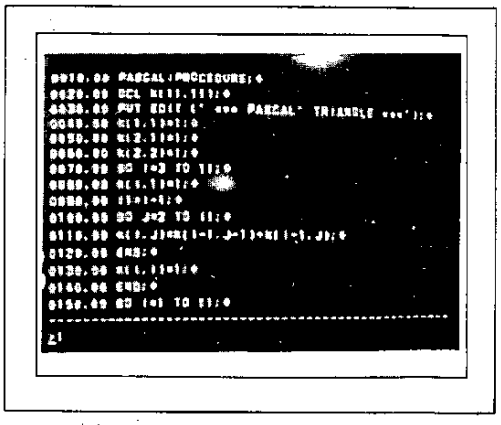
8.3.2　ランダムなＳＣＲＯＬＬの移動

画面上に表示されていないステートメントを見たい場合はＳＣＲＯＬＬコマンド'＞'又は'＜'を使用すればよい。例えば，'＞４０'をタイプインすると，ステートメント番号４０以上の１フレーム分のステートメント郡が画面に表示される。

ＳＣＲＯＬＬコマンド'＞△ライン番号'は，指定されたライン番号以上の１フレーム分のステートメント郡を,'＜△ライン番号'は，指定されたライン番号以下の１フレーム分のステートメント郡を画面に表示することを指令する。



(1) 現在のフレーム



(2) '＞1'入力後のフレーム

図8.4 '＞'使用の例

例えば，図8.4の(1)で'＞1'をタイプインすると，ライン番号1以上の1フレーム分のステートメントが画面に表示される。

8.4 編　　集

SCROLLの編集においては，新しいステートメントの入力と追加，すでに入力したステートメントの修正との間の区別はない。そして，編集用のコマンドも又ない。

8.4.1 新しいステートメント入力と追加

入力すべきステートメントをライン番号を付けて，単に入力するだけでよい。SCROLLは，入力されたステートメントが表示されるフレームの末尾行にくるようにコントロールされる。

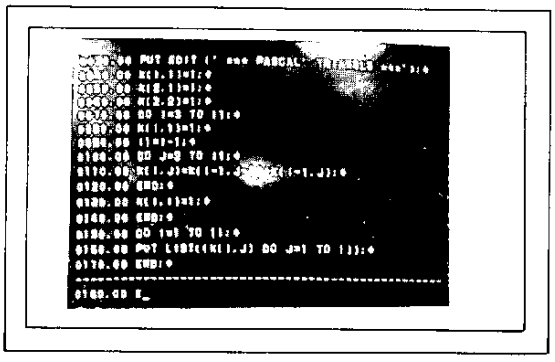


図8.5 ライン番号180の入力中

すでに入力されたステートメント郡の途中にステートメントを追加すると，そのステートメントが表示されるフレームの末尾行になるようにSCROLLをコントロールする。

8.4.2　ステートメントの修正

すでに入力されたステートメントを修正する場合，２通りの方法がある。

１つの方法は，１ステートメント全部を修正する場合に使われる。すでに入力されたステートメントと同じライン番号をもつ新しいステートメントを入力すると，古いステートメントと置きかわる。

例えば，図8.6の(1)で，ライン番号４０がまちがっていることに気が付いたので，(2)で，直接，ライン番号４０のステートメントをタイプインして修正した。

図8.6の(2)のフレームでもわかるように，SCROLLは修正されたステートメントが画面に表示されるフレームの末尾行にくるようにコントロールされる。ただし，修正されたステートメントより前にあるステートメント郡が１フレーム分に満たない場合は，修正されたステートメントより後にあるステートメントが追加表示される。

```
0010.00 PASCAL:PROCEDURE;◇
0020.00 DCL K(11,11);◇
0030.00 PUT EDIT('***……***');◇
  (SKIP(5),X(23),A,SKIP(2));◇
0040.00 K(I,1)=1;◇
0050.00 K(2,1)=1;◇
0040.00 K(1,1)=1;_
```

(1)　ライン番号４０を修正中の画面

```
0010.00 PASCAL:PROCEDURE;◇
0020.00 DCL K(11,11);◇
0030.00 PUT EDIT('***……***');◇
  (SKIP(5),×(23),A,SKIP(2));◇
0040.00 K(1,1)=1;◇
0050.00 K(2,1)=1;◇
_
```

(2)　ライン番号４０を修正後の画面

図8.6　直接ステートメントを修正した例

別の方法は，修正すべきステートメントが文字単位の修正ですむ場合に使われる。まず，SCROLLコマンドの'＝△ライン番号'をタイプインし，修正すべきステートメントをEDIT領域に表示させる。そして，ディスプレイ装置のハードウェアの修正機能を使い，ステートメントを修正し，修正されたステートメントをタイプインする。

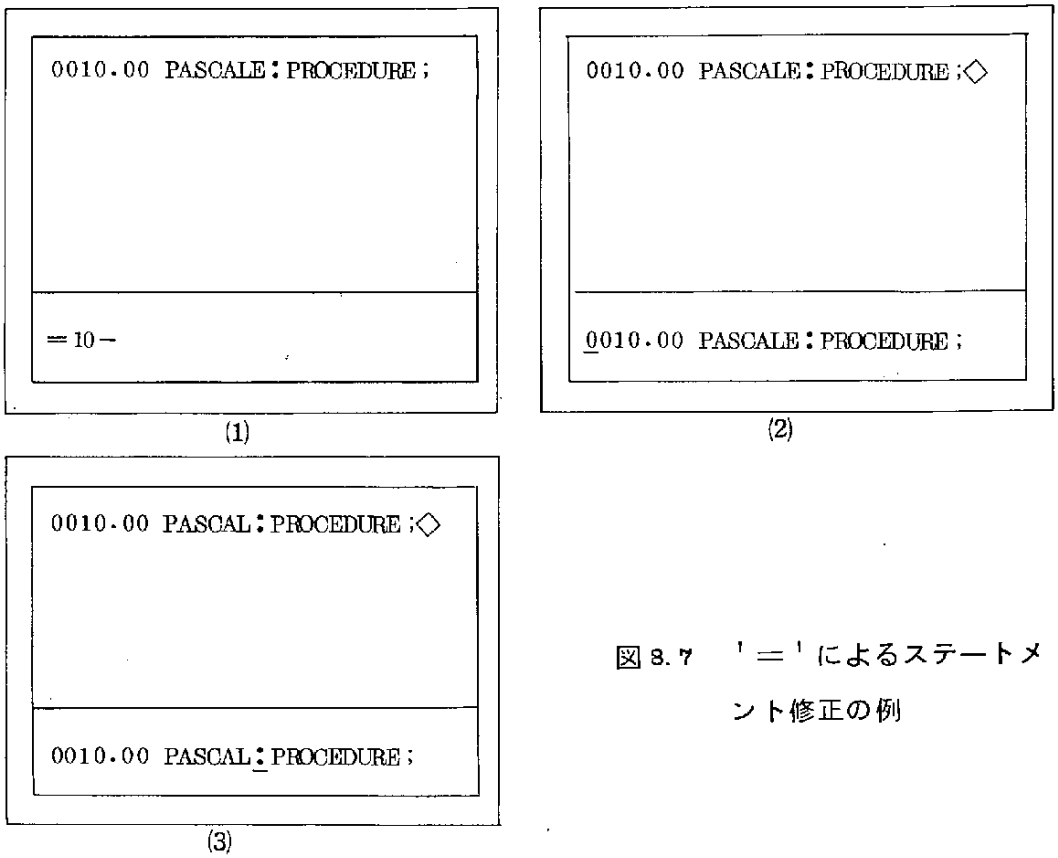


図8.7 '＝'によるステートメント修正の例

例えば，図8.7において，ライン番号１０を入力した直後，'PASCALE'の'E'が余分であることに気が付いた。'E'の文字を消去するだけでよいので'＝10'をタイプインして（図3.4の(1)），ライン番号１０のステートメントをEDIT領域に表示する。カーソルは１文字目にセットされる（(2)）。ここで，学習者は，カーソルを１５文字目の'E'の下にもってきて，ディスプレイ装置上のファンクション釦 DEL を押下すると，カーソルの位置にある文字'E'が消去され，カーソル以下の文字が，１文字分前に移る（(3)）。これで，ステートメントの修正が終ったので，カーソルを';'の次にもってきて，タイプインする。

修正すべきステートメントが複数行から成っている場合，まず計算機から１行目がEDIT領域に表示されるので，１行目を修正しタイプインする。タイプインし終ると次の行がEDIT領域に表示される。同様に修正し，タイプ

インする。この処理を最後の行までくり返す。途中の行で修正されたステートメントが終った場合，以下に表示される行に対しては，０文字だけ（カーソルを１字目にセット）タイプインすればよい。

以上をまとめると，１ステートメント全部を修正する場合は，

ライン番号　ステートメント

の形式でタイプインし，文字単位でステートメントを修正する場合は，

＝ライン番号

の形式でタイプインし，修正すべきステートメントをEDIT領域に表示させ，修正してタイプインする。この場合，直前に入力したステートメントを修正する時はライン番号を省略してもよい。

修正すべきステートメントのライン番号の変りに，そのステートメントに含まれる文字列を指定してもよい。

形式は

＝△：文字列：　又は　＝△' 文字列 '

である。

8.4.3　ステートメントの消去

すでに入力されたステートメントを消去する場合は，ＣＰＬの**％ERASE**ステートメントを使用する。

例えば，ライン番号５０のステートメントを消去するには，

％ERASE　50；

をタイプインし，ライン番号１００から２００までのステートメントを消去するには

％ERASE　100，200；

をタイプインすればよい。

％ERASEステートメントの形式は，

％ERASE　ライン番号1〔，ライン番号2〕;

である。

％ERASEステートメントがタイプインされると，ライン番号１の直前

（1つ前）のライン番号をもつステートメントが，フレームの中央にくるようにSCROLLをコントロールする。

8.5 ステートメントの印刷

すでに入力されたステートメントのリストをタイプライターに印刷するのには，CPLの%LISTステートメントを使用する。

例えば，ステートメント全部を印刷するには

```
%LIST;
```

をタイプインし，ライン番号10だけのステートメントを印刷するには，

```
%LIST  10;
```

をタイプインし，ライン番号100から200までのステートメントを印刷するには

```
%LIST  100, 200;
```

をタイプインすればよい。

%LISTの形式は，

```
%LIST 〔ライン番号1〔 ,ライン番号〕〕;
```

である。

8.6 SCROLL用コマンド

ここで，SCROLL用コマンドについてまとめておく。

以下の説明は，次の約束に基づく。

△は0個以上の空白を表わす。

␣は1個以上の空白を表わす。

〔　〕のついた部分は，省略可能を表わす。

{　}のついた部分は，その中のひとつを選択することを表わす。

〔{　}〕のついた部分では，省略した場合には，下線部を指定したとみなされる。

(1) 前進コマンド

(a) 機能

SCROLLを1ステートメント又は1フレーム前進させる。

(b) 形式

$$\left\{ \begin{matrix} \wedge \\ F \end{matrix} \right\} [\ \triangle F\ ]$$

(c) パラメータ

F　：1フレーム前進することを指定する。

省略：1ステートメント前進することを指定する。

(2) 後進コマンド

(a) 機能

SCROLLを1ステートメント又は1フレーム後進させる。

(b) 形式

$$\left\{ \begin{matrix} V \\ B \end{matrix} \right\} [\ \triangle F\ ]$$

(c) パラメータ

F　：1フレーム後進することを指定する。

省略：1ステートメント後進することを指定する。

(3) 以上コマンド

(a) 機能

指定したライン番号をもつステートメントがフレームの先頭行にくるようにSCROLLをコントロールする。

(b) 形式

$$\left\{ \begin{matrix} > \\ G \end{matrix} \right\} \triangle \text{ライン番号}$$

(c) パラメータ

ライン番号：フレームの先頭行のライン番号を指定する。

(4) 以下コマンド

(a) 機能

指定したライン番号をもつステートメントがフレームの末尾行にくるようにSCROLLをコントロールする。

(b) 形式

$$\left\{\begin{matrix}< \\ \mathrm{L}\end{matrix}\right\}\triangle\text{ライン番号}$$

(c) パラメータ

ライン番号：フレームの末尾行のライン番号を指定する。

(5) 等号コマンド

(a) 機能

指定したライン番号をもつステートメントをEDIT領域に表示する。

(b) 形式

$$\left\{\begin{matrix}= \\ :\end{matrix}\right\}\left[\triangle\left\{\begin{matrix}\text{ライン番号} \\ \left\{\begin{matrix}' \\ :\end{matrix}\right\}\text{文字列}\left\{\begin{matrix}' \\ :\end{matrix}\right\}\end{matrix}\right\}\right]$$

(c) パラメータ

ライン番号：EDIT領域に表示すべきステートメントのライン番号を指定する。

$\left\{\begin{matrix}'\text{文字列}' \\ :\text{文字列}:\end{matrix}\right\}$：EDIT領域に表示すべきステートメントが含む文字列を指定する。文字列は8字以内とする。

省略：EDIT領域に直前にタイプインしたステートメントを表示することを指定する。

(d) 使用上の注意事項

(ア) パラメータに文字列を指定した場合は，ステートメント（プログラム）の先頭から探して，最初にみつかったステートメントをEDIT領域に表示して，このコマンドは終了する。もし，つづけて探してもらいたい場合は，次の反復コマンドをタイプインする。

(6) 反復コマンド

(a) 機能

直前にタイプインした等号コマンドを実行する。

(b) 形式

{ : / R }

(c) パラメータ

なし。

(d) 使用上の注意事項

(ア) 直前に入力したSCROLL用のコマンドは必ず等号コマンドで，パラメータは文字列でなければならない。

(7) 質問コマンド

(a) 機能

指定されたエラー番号の意味を詳細に説明する。

(b) 形式

{ ? / Q } 〔△エラー番号〕

(c) パラメータ

エラー番号：質問すべきエラー番号を指定する。

省　　略：直前に表示されたエラー・メッセージのエラー番号を指定したと見做される。

8.7　メッセージ一覧

エラー番号	メ　ッ　セ　ー　ジ
SCR0	パラメータ　ガ　アヤマッテイル。
SCR1	シテイサレタ　ラインバンゴウ　オ　モツ　ステートメント　ガ　ナイ。
SCR2	コマンド　ノ　ツカイカタ　ガ　アヤマッテイル。
SCR3	オブジェクト　エリア　ニ　ステートメント　ガ　ナイ。
SCR4	シテイサレタ　ストリング　オ　モツ　ステートメント　ガ　ナイ。
SCR5	ストリング　ノ　ナガサ　ガ　8　モジ　オ　コエテイル。
SCR6	ワリコミ　オ　ウエツケマシタ。
SCR7	ニュリョク　ノ　モジスウ　ガ　49　ジ　オ　コエテイル。

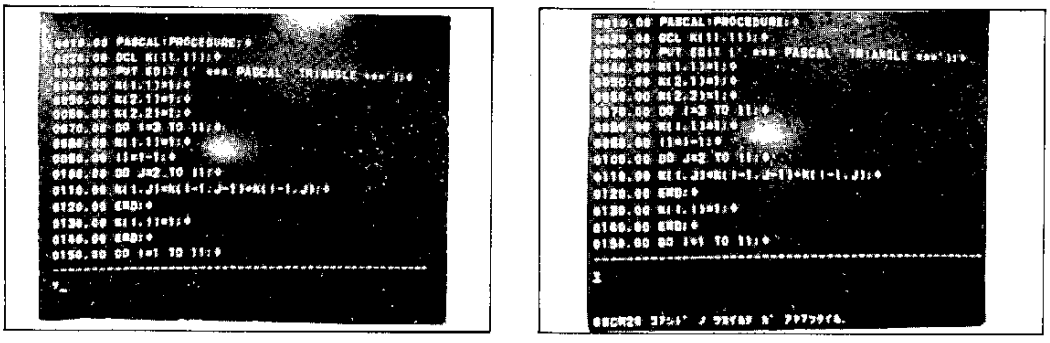


図8.8　エラー・メツセージの例

昭和４７年 ３ 月発行

発行所　財団法人　日本情報処理開発センター

東京都港区芝公園３丁目５番８号

機 械 振 興 会 館 内

ＴＥＬ （４３４）８２１１（代表）

印刷所　株式会社　チャンスメーカー

東京都千代田区神田錦町３－１９

ＴＥＬ （２９５）１１７７（代表）