45-S002

ディスプレイ・システムの研究開発

――オンラインシステムにおけるディスプレイ装置の応用――

昭和46年5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45年度情報処理に関する調査・研究補助事業」の一部として実施したものであります。

序に代えて

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，この報告書はグラフィック・ディスプレイおよびキャラクタ　ディスプレイ装置を用いた各種プログラムの開発成果を報告するものであります。とくに，この研究開発は，これらのディスプレイ装置を人間と機械のインタラクティブな情報交換のよき道具として生かすためには，ソフトウェア上にどのような問題点があるかを検討することを一つの目標としたものであります。

本報告が各方面に利用され，わが国情報処理産業発展の一助として寄与できるよう念願いたす次第であります。

昭和４６年５月

財団法人　日本情報処理開発センター

会長　　難　波　捷　吾

はじめに

本報告書は、４編に分かれている。

第１編　グラフィック　ディスプレイのプログラム　システムは，大型計算機とグラフィック　コンピュータを連結し，リアルタイムにグラフィック　ジョブをおこなうためのオペレーティング　システムを中心にして，三次元物体の隠れ線消去のプログラム，回路解析プログラムなどのアプリケーションも含めた開発報告である。

第２編　ＪＵＭＰＳ（ＪＩＰＤＥＣ　Universal　Mathematical　Programming　System　）は，統計解析，経量モデル作成等の機能を持つ汎用プログラムで，特にグラフィック　ディスプレイを用いインタラクティブに使用することができるのが特長である。ＪＵＭＰＳは４４年度にシステムの基本的な構想をまとめ，４５年度は具体的なシステムを作成した。このシステムは第１編で述べたグラフィック　オペレーティング　システムのもとで使用することができる。

第３編　オンライン　シミュレーション言語は，オンラインでインタラクティブにシミュレーションをおこなうための特殊問題向言語である。特にタイムシェアリング　システムの下で，キャラクタ　ディスプレイ端末を使用して会話的に作業できるところに特長がある。従来，バッチ用のシミュレーション言語はすでにいくつか利用されているが，オンライン用のものは極めて少ない。しかし，シミュレーションのモデル作成のような仕事は，コンピュータと密接に相互連絡をとりながら人間と機械が協同しておこなえばより効率があがるであろうということは多くの人々が認めるところである。

本年度は，こういった特長をふまえて，従来のシミュレーション言語の比較検討をおこなった上で，インタラクティブな機能，ディスプレイの効果を生かす機能等を盛り込んだ新しい言語の基礎研究をおこなった。

なお４６年度は具体的に言語プロセッサを作成する。

第４編　インタラクティブ学習システム（Interactive　Learning System略してＩＬＳ）は，コンピュータと人間とが相互に密接な連絡をとりながら効率よく学習をおこなうシステムである。

これは一般にＣＡＩと呼ばれている分野に属しているが，当センターで開発しつゝあるシステム

はＣＬＡＳＳ（Conversational Language ASsisted System） と呼び，タイムシェアリングシステムの下で稼動する会話型言語を教育するシステムである。したがって教育そのものも，タイムシェアリングの下で端末装置を使用しておこなわれることとなる。

なお４６年度には引きつづき，スライドプロジェクタ，ＣＲＴディスプレイ等を利用した場合の学習プログラムのあり方および教育効果等の面での比較実験をおこなう予定である。

なお，以上４つのプロジェクトの他，ディスプレイ システムの研究開発の事業の一環として，米国におけるインタラクティブ ディスプレイシステムのソフトウェアに関する調査をおこなった。この調査報告は

"米国におけるインタラクティブ・グラフィックス および オンライン・データ・マネージメントに関するソフトウェア調査報告書"

として，内部資料の形ですでに作成されている。

第1編　グラフィックディスプレイのプログラムシステム

目　次

第Ⅰ部　オペレーティング・システムCGOS

第1章　システムの概要

ＣＧＯＳ（Comprehencive Graphic Operating System）は，汎用大型計算機に，小型のグラフィック・ディスプレイ用計算機を連結し，グラフィック・ディスプレイ装置を用いる多種のアプリケーション・プログラムの作成と，その多目的な利用を容易ならしめるオペレーティング・システムである。

ＣＧＯＳの下において，グラフィック・ディスプレイ装置はアプリケーション・プログラムにとって図形を入出力する装置であると同時に，システムのオペレーション・コンソールともなっている。

ＣＧＯＳでは，ＨＩＴＡＣ８４００処理装置がシステムのホスト・コンピュータとして，ＨＩＴＡＣ－８８１１グラフィック・コンピュータがサテライト・コンピュータとして位置しており，図形処理のアプリケーション・プログラムの実行中も，Ｈ－８４００処理装置内においては，従前のバッチ，及びリアルタイム・ジョブが，マルチ処理可能である。また，これら全てのジョブを，ＣＲＴに向ったオペレータが制御することができる。

ＣＧＯＳでは，グラフィック・アプリケーション・プログラムにおける図形処理，及びインタラクション処理のかなりの部分をＨ－８８１１サテライト・コンピュータに負わせるロード・シエアリングによって，システム全体のスルー・プット向上を図っている。

このシステムによって，ユーザーは，ＦＯＲＴＲＡＮ，その他のプログラミング言語で，グラフィック・ＣＲＴを用いた，インタラクティブ・プログラムを容易に作製することができる。

第2章　システムの構成

2.1　機器構成と配置

〔ホスト・コンピュータ〕

○H－8400処理装置（262K byte）	
○マルチプレクサー・チャネル	1
○セレクター・チャネル	3
○ディスク・パック装置（7.2M byte）	4
○磁気テープ装置	8
○カード・リーダ	1
○カード・パンチ	1
○ライン・プリンタ	2
○コンソール・タイプライタ	1

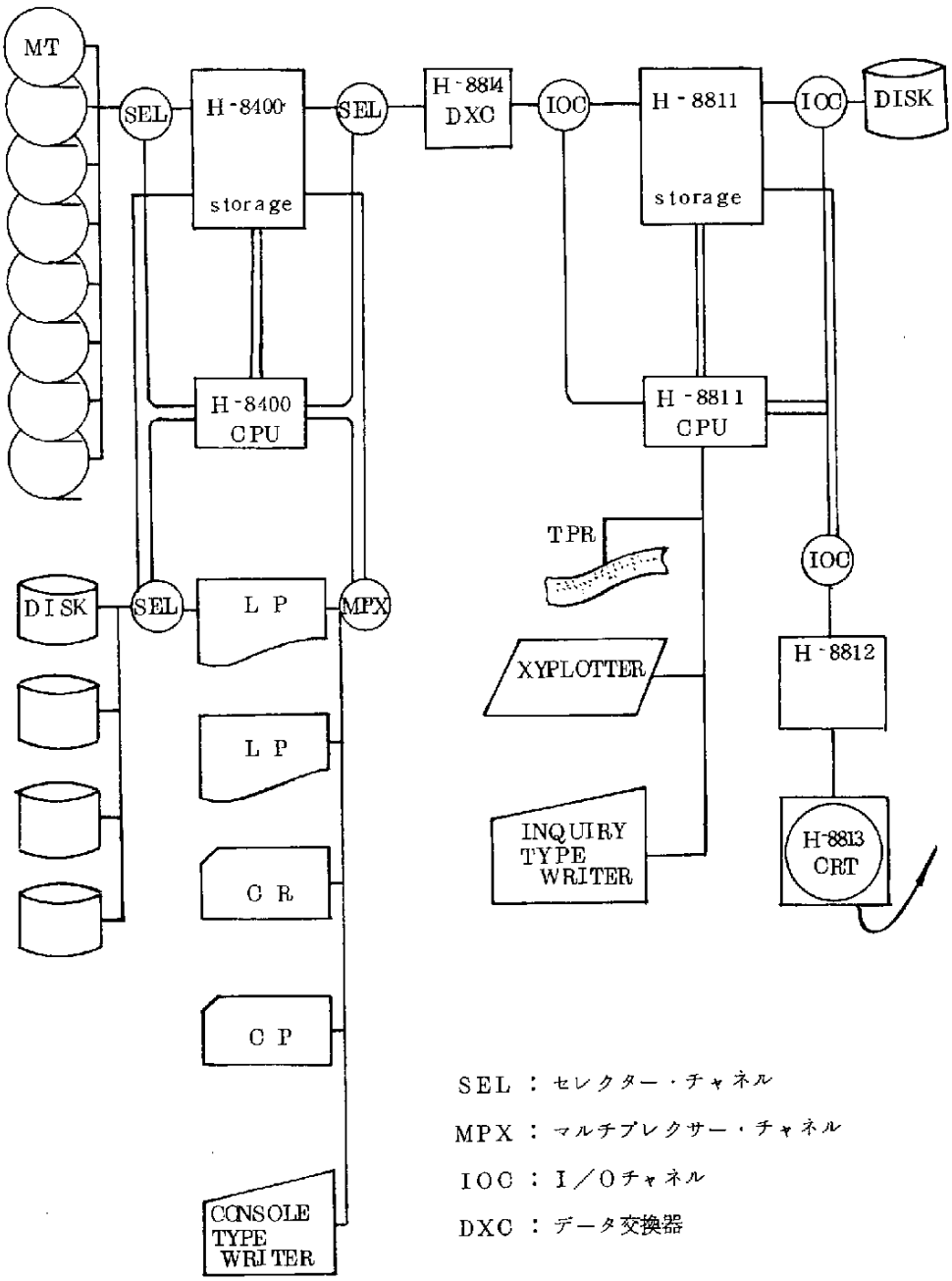
〔コミュニケーション・アダプタ〕

○H－8814データ交換装置（DXC）	1

〔サテライト・コンピュータ〕

○H－8811処理装置（16K word）	1
○I／Oチャネル	3
○H－8812ディスプレイ制御装置	1
○H－8813グラフィック・CRT ライト・ペン ファンクション・キイ　付き ダイアル	1
○紙テープ・リーダ	1
○XYプロッタ	1
○問い合わせタイプライタ	1
○ディスク（5.25M byte）	1

図 1.1 に構成図を示す。



SEL ： セレクター・チャネル

MPX ： マルチプレクサー・チャネル

IOC ： I／Oチャネル

DXC ： データ交換器

図 1.1　CGOSにおける機器構成

2.2 プログラム構成

ＣＧＯＳを構成するプログラム体系を表１－１に示す。ＣＧＯＳはＨＩＴＡＣ８４００のオペレーティング　システムＤＯＳ３０４を包含している。

表１－１　ＣＧＯＳを構成するプログラム群１

コントロール・プログラム

ＨＩＴＡＣ－８４００

ＤＯＳ３０４エグゼクティブ

グラフィック・ジョブ・モニター（ＧＪＭＯＮ）

Ｈ－８８１１エグゼクティブ＊

グラフィック・コミニュケーション・ハンドラー（ＧＣＨ）＊

プロセッシング・プログラム

ランゲッジ・プロセッサー

ＦＯＲＴＲＡＮ

ＣＯＢＯＬ

ＡＳＳＥＭＢＬＥＲ

グラフィック・ランゲージ・プロセッサー

システム・ユーティリテイ

リンケージ・エディター

ライブラリアン

ソート・マージ

ｅｔｃ.

アプリケーション・プログラム

グラフィック・アプリケーション・プログラム

□□で囲まれたものが，グラフィックに特有なプログラムである。

このうちグラフィック・ランゲージに関しては来年度以降，開発の予定である。なお＊はＨ－８８１１側のプログラムである。

表１－１に示したこれらのプログラム群はすべてコアイメージ・ライブラリイの形となっ

ている。

このほか，ＣＧＯＳでは，各種グラフィック・アプリケーション・プログラムの幅広い用途に応じ表１－２に示すサブルーチン・パッケージをライブラリイとして提供する。これらのサブルーチン・パッケージはオブジェクト・モジュール・ライブラリの形でシステムに登録されており必要に応じてユーザーのプログラムにリンケージ・バインドされて，その機能を果たす。

表１－２　ＣＧＯＳを構成するプログラム群２

グラフィカル・インタラクション
　　アナリシス・サブルーチン・パッケージ（ＧＩＡＳＰ）
グラフィカル・サブルーチン・パッケージ（ＧＳＰ）
データ・ベース・マネジャー（ＤＢＭ）
リング・ストラクチュア・プロセッサー（ＲＳＰ）

これらのプログラムのうち，既に開発したものについて，その大きさを示す。

1. ＧＪＭＯＮ　　７７０ステップ
　　８Ｋバイト
2. Ｇ　Ｃ　Ｈ　　4,５００ステップ
　　１６Ｋワード（一部オーバー・レイ）
3. ＧＩＡＳＰ　　1,３００ステップ
　　６Ｋバイト
4. Ｇ　Ｓ　Ｐ　　3,５００ステップ
　　３０Ｋバイト

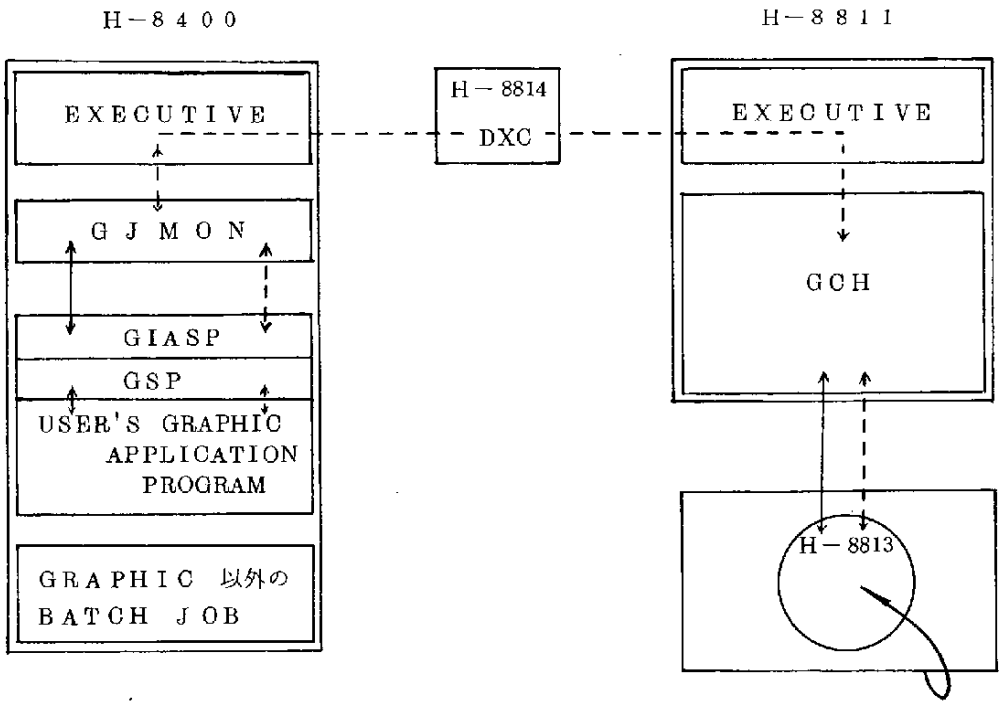
第3章　コントロール・システム

3.1　CGOS・コントロール・システムの概要

ＣＧＯＳにおけるコントロール・プログラム・システムの特徴は，Ｈ－８４００プロセッサー内のＧＪＭＯＮ（Graphic　Job　Monitor）と，Ｈ－８８１１プロセッサー内のＧＣＨ（Graphic　Communication　Handler）の両プログラムに現われている。

ＧＪＭＯＮ及びＧＣＨはＣＧＯＳのシステム・イニシェーション時にそれぞれの主記憶装置にローディングされ，ＤＯＳエグゼクティブ，Ｈ－８８１１エグゼクティブ等の制御プログラムと協業して，ＣＧＯＳの運行を管理する。

ＣＧＯＳのコントロール・プログラム・システムは，従来Ｈ－８４００付属のコンソール・タイプライタで行なっていたＤＯＳオペレーションのほゞ全てを，グラフィック・ＣＲＴからのオペレーションで代行できるようにし，同時に，幅広い分野のグラフィック・アプリケーション・プログラムにグラフィック・ＣＲＴを介した図形入出力を可能にする。すなわち，ＣＧＯＳの下では，グラフィック・ＣＲＴは図形入出力用のデバイスであるとともに，システム・オペレーション・コンソールである。



実　　線　：　コントロール・フロー

破　　線　：　データ・フロー

GJMON　：　Graphic Job Monitor

GCH　：　Graphic Communication Handler

GIASP　：　Graphical Interaction Analysis Subroutine Package

GSP　：　Graphic Subroutine Package

図1.2　GJMON，GCH，及びグラフィック・アプリケーション・プログラムの関係

3.2 GJMON（Graphic Job Monitor）

〔GJMONの機能〕

GJMONは，H－８４００プロセッサー内に，DOSエグゼクティブの下でスレーブ・モード・タスクとして常駐し，次の機能を果たす。

① DXC経由によるH－８８１１プロセッサーとの間の相互データ転送

② DXC経由で受け取ったメッセージ・データの解析

③ H－８８１３グラフィック・CRTのオペレータが要求したプログラムの起動，及びモニタリング（同時に５つまでのマチチ・プログラミングが可能）

④ グラフィック・コマンド・リクエスト（後述）の処理

⑤ グラフィック・アプリケーション・プログラムとの相互データ転送

〔ＧＪＭＯＮの処理アルゴリズム〕

ＧＪＭＯＮでは，次の４種の割り込みを制御する。

① ＤＸＣのデータ転送に伴なう割り込み

② プログラム制御に伴なう，キイ・イン，アウト・シミュレーション・マクロの割り込み

③ プログラム間データ転送に伴なうＩＰＣ割り込み

④ コンソール・リクエスト割り込み

ＧＪＭＯＮは，グラフィック・ＣＲＴオペレーターの要求で起動したプログラムを制御するために，表１－３のような制御表を持っている。この制御表は，3.5 システム・コマンド処理と重要な関係を持っている。

表 1-3　ＧＪＭＯＮプログラム制御表

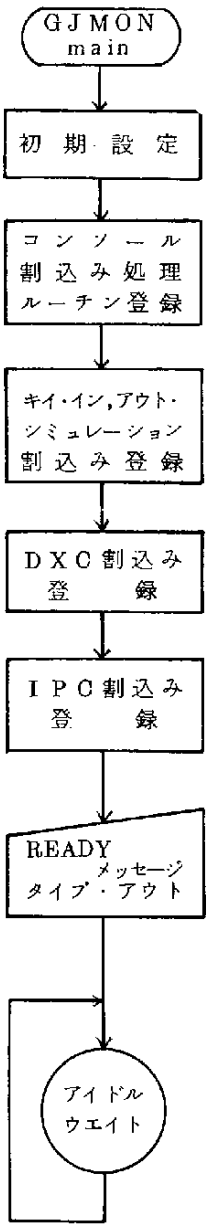
Pid.		Pname	PS1	PS2	PS3		システム・メッセージ・プール
1							
2							
3							
4							
5							
6							
7							
8							

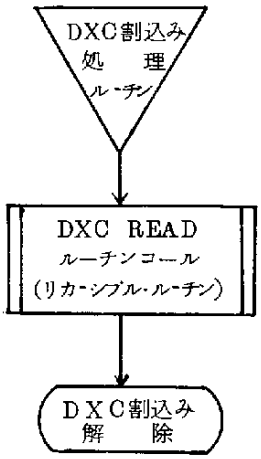
Ｐｉｄ　：　プログラムｉｄ．ナンバー
Ｐｎａｍｅ：　プログラム名
ＰＳ１　：　プログラム・ステータス１
　▼Ｒ▼ ── ランニング中
　▼Ｅ▼ ── 終了
ＰＳ２　：　プログラム・ステータス２
　▼Ｉ▼ ── 初期状態
　▼Ｒ▼ ── グラフィック・プログラム・ランニング中
　▼Ｑ▼ ──　〃　〃　クィット中
ＰＳ３　：　プログラム・ステータス３
　▼Ｇ▼ ── グラフィック・プログラム
　▼Ｎ▼ ── ノン・グラフィック・プログラム

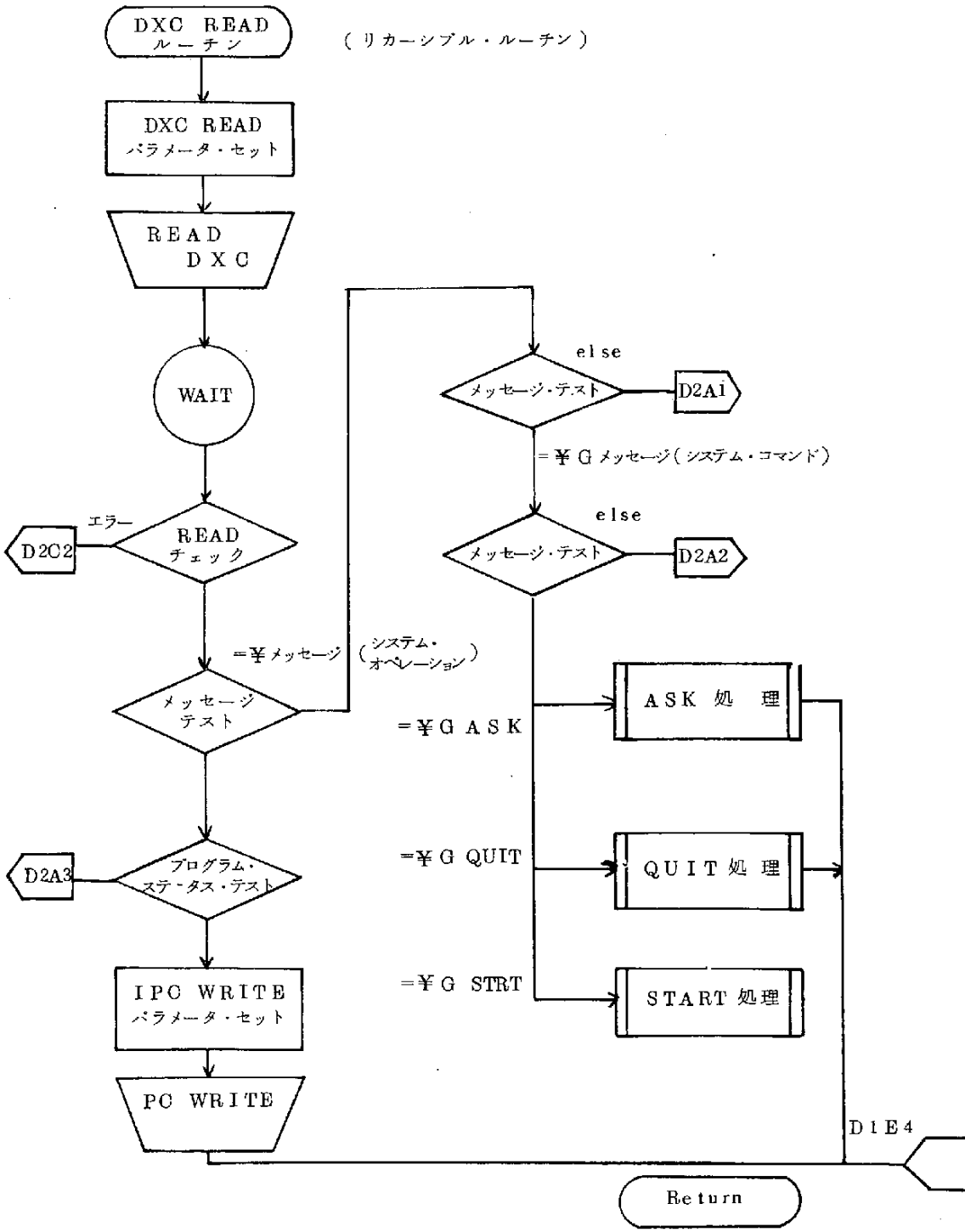
ＧＪＭＯＮの構造上の特徴は，ＤＸＣ入出力のルーチンが，リエントラント・リカーシブルになっている点である。Ｈ－８４００，Ｈ－８８１１の両プロセッサーが，非同期並列に稼動するため，タイミングによって両者の出力要求が衝突し合う場合が生ずる。ＤＸＣのデータ転送はシンプレックス方式で，一時には，一方向にしかデータ転送ができず，この要求は誤応答となって終る。コンソール・オペレーションに優先を与えるためGJMON側は自分の出力要求を待ち合わせ，一たんＨ－８８１１（つまりＧＣＨ）からのメッセージを読み取り，その処理を行なった後に，自分の出力要求を遂行する必要がある。Ｈ－8811側からのデータ転送が連続的に生じている場合には，この衝突が次々と生じ，ＤＸＣの入出力ルーチンは，次々に再帰的に呼ばれることが想定される。

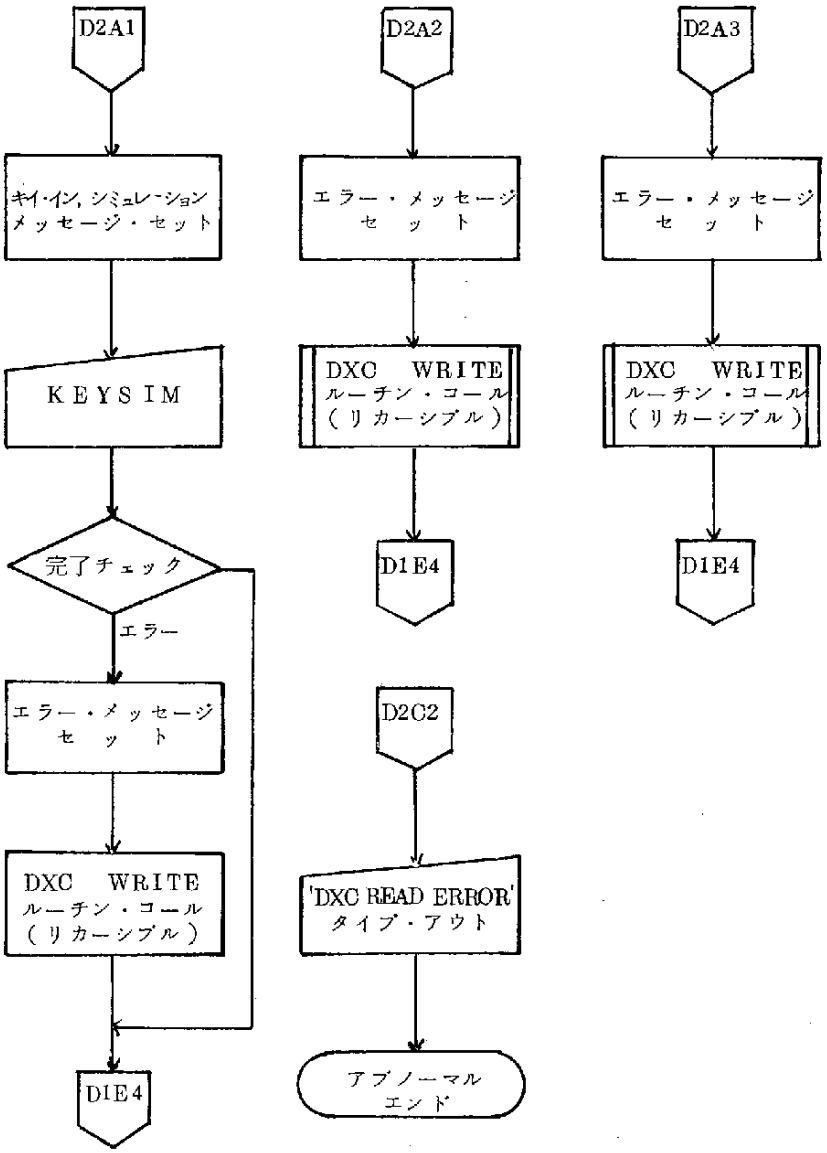
以下にＧＪＭＯＮのブロックチャートを示す。

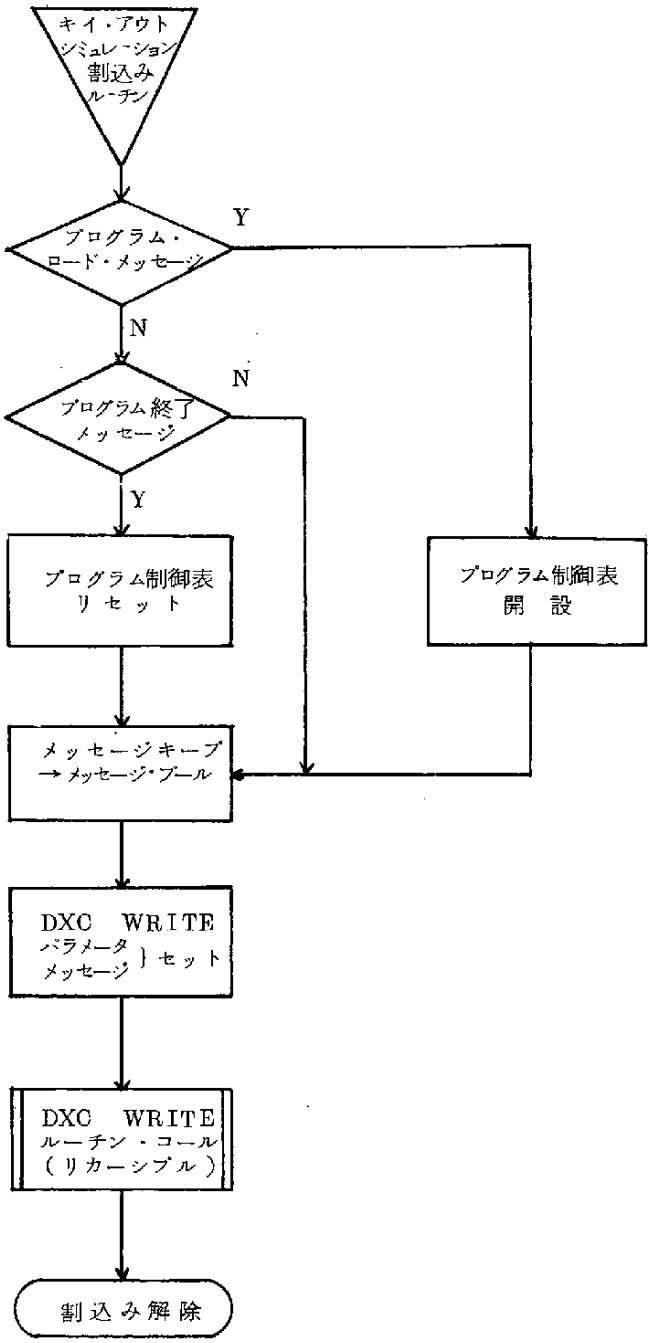
GJMONのブロック・チャート

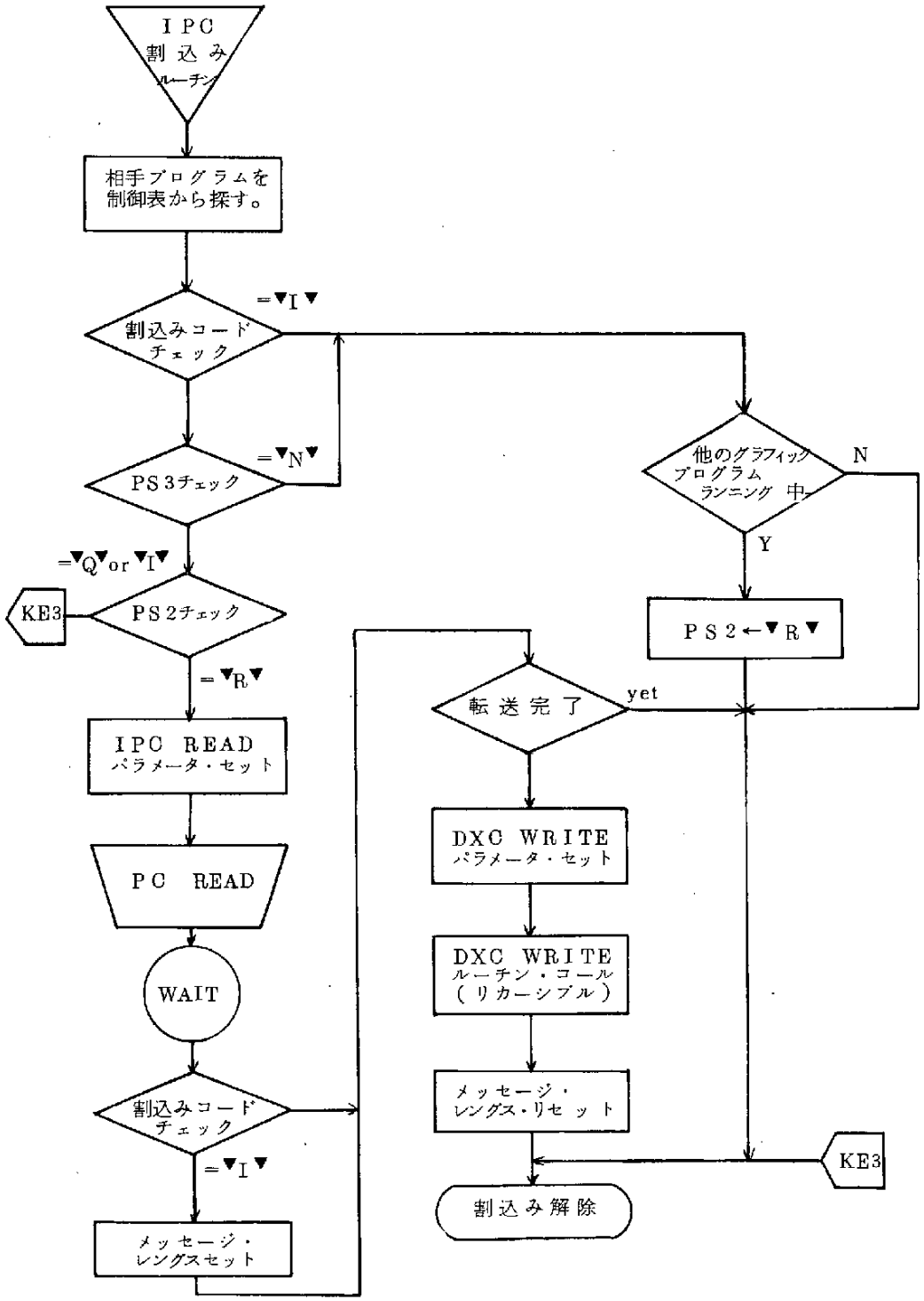


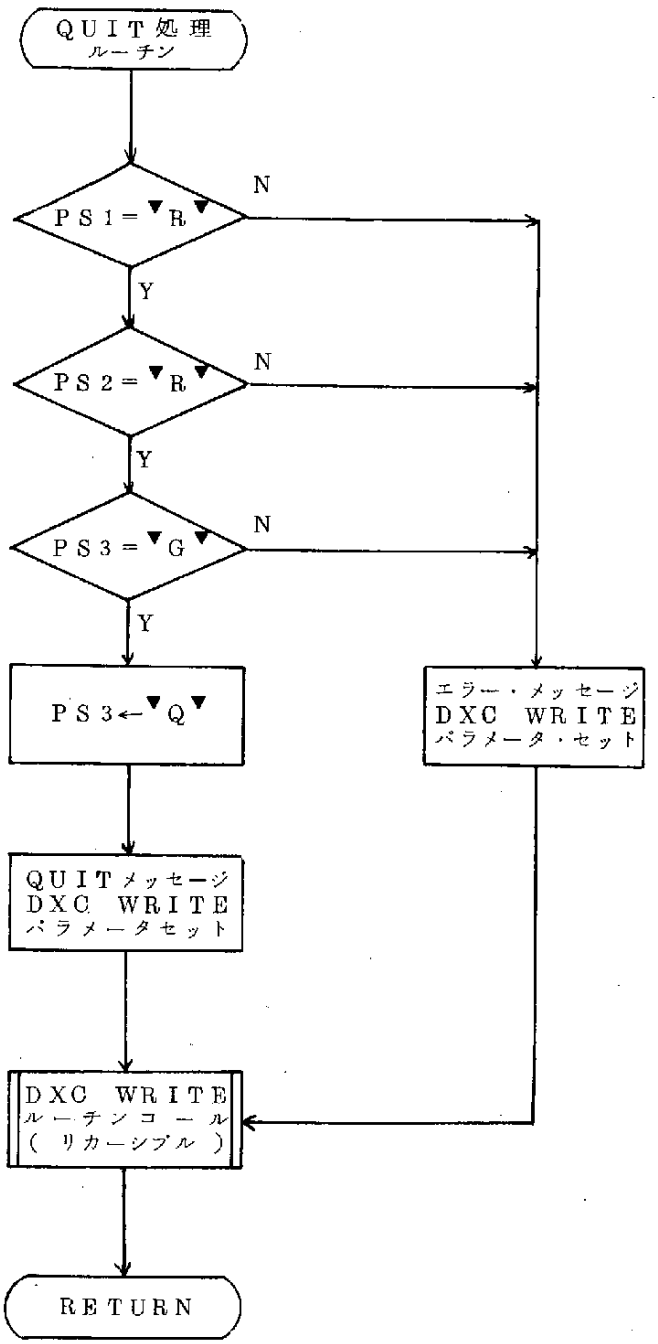


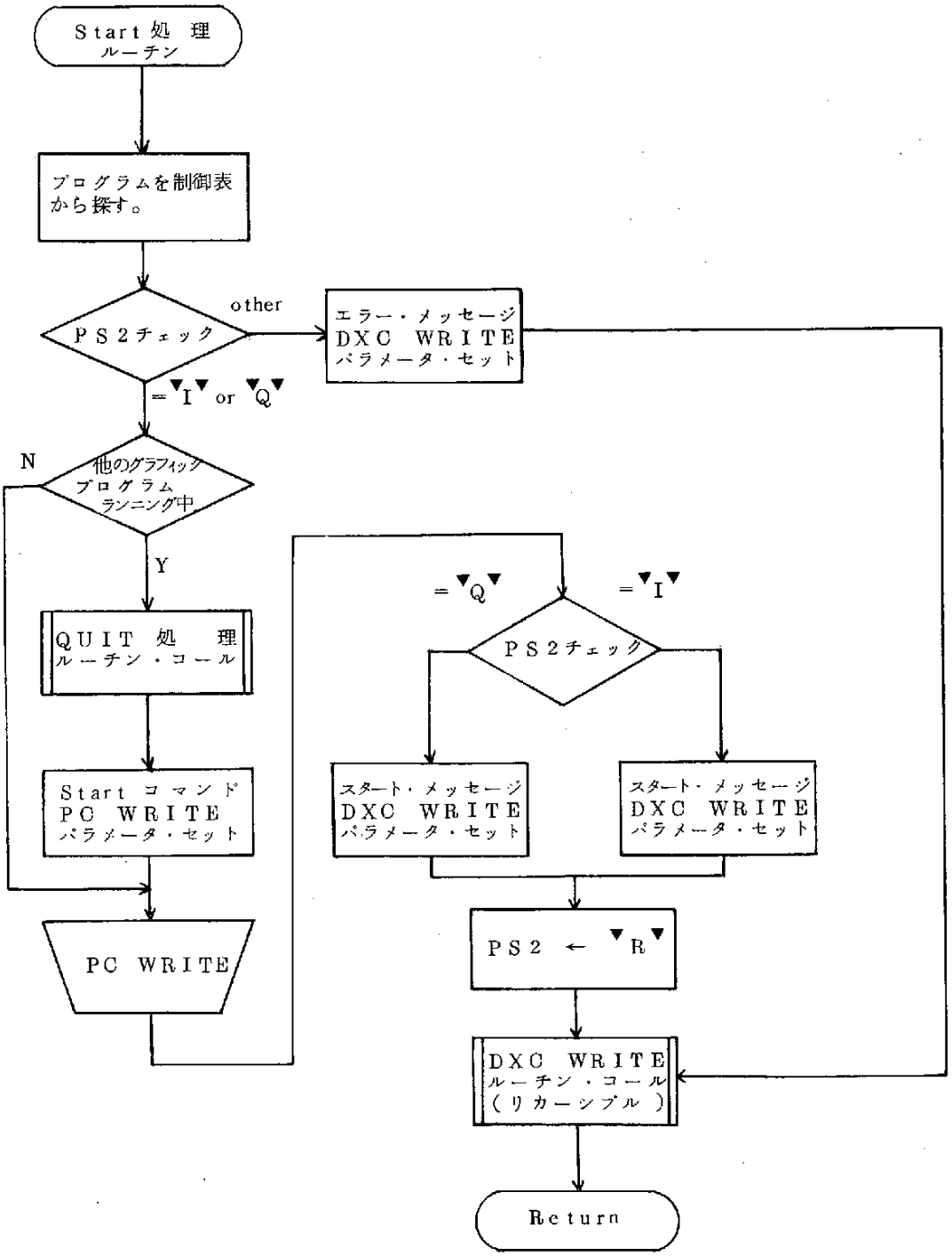

DXC READ
ルーチン
（リカーシブル・ルーチン）
DXC READ
パラメータ・セット
READ
DXC
WAIT
エラー
D2C2
READ
チェック
=¥メッセージ
（システム・
オペレーション）
メッセージ
テスト
D2A3
プログラム・
ステータス・テスト
IPC WRITE
パラメータ・セット
PC WRITE
else
メッセージ・テスト
D2A1
=¥G メッセージ（システム・コマンド）
else
メッセージ・テスト
D2A2
=¥G ASK
ASK 処 理
=¥G QUIT
QUIT 処 理
=¥G STRT
START 処理
D1E4
Return

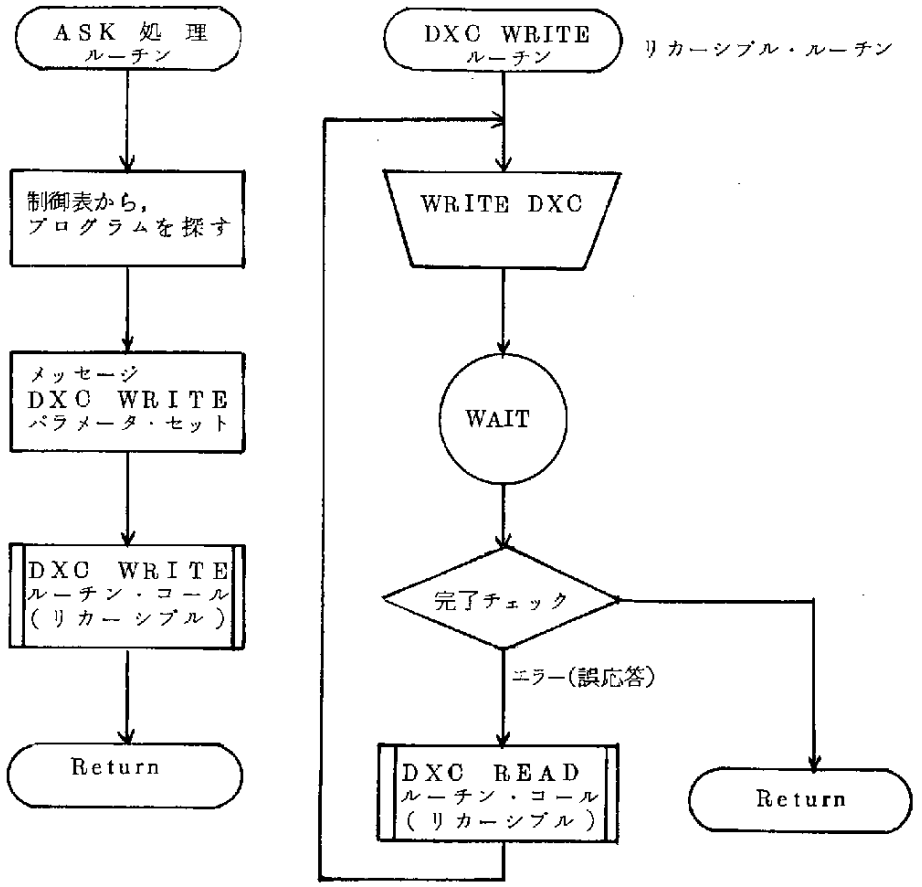

D2A1
キイ・イン, シミュレーション
メッセージ・セット
KEYSIM
完了チェック
エラー
エラー・メッセージ
セット
DXC WRITE
ルーチン・コール
(リカーシブル)
D1E4
D2A2
エラー・メッセージ
セット
DXC WRITE
ルーチン・コール
(リカーシブル)
D1E4
D2C2
'DXC READ ERROR'
タイプ・アウト
アブノーマル
エンド
D2A3
エラー・メッセージ
セット
DXC WRITE
ルーチン・コール
(リカーシブル)
D1E4


キイ・アウト
シミュレーション
割込み
ルーチン
プログラム・
ロード・メッセージ
Y
N
プログラム終了
メッセージ
N
Y
プログラム制御表
リセット
プログラム制御表
開設
メッセージキープ
→メッセージ・プール
DXC WRITE
パラメータ
メッセージ
セット
DXC WRITE
ルーチン・コール
(リカーシブル)
割込み解除


IPC
割込み
ルーチン
相手プログラムを
制御表から探す。
割込みコード
チェック
=▼I▼
PS3チェック
=▼N▼
=▼Q▼or ▼I▼
KE3
PS2チェック
=▼R▼
IPC READ
パラメータ・セット
PC READ
WAIT
割込みコード
チェック
=▼I▼
メッセージ・
レングスセット
他のグラフィック
プログラム
ランニング 中
N
Y
PS2←▼R▼
転送完了
yet
DXC WRITE
パラメータ・セット
DXC WRITE
ルーチン・コール
(リカーシブル)
メッセージ・
レングス・リセット
KE3
割込み解除


QUIT処理
ルーチン
PS1=▼R▼
N
Y
PS2=▼R▼
N
Y
PS3=▼G▼
N
Y
PS3←▼Q▼
エラー・メッセージ
DXC WRITE
パラメータ・セット
QUITメッセージ
DXC WRITE
パラメータセット
DXC WRITE
ルーチンコール
(リカーシブル)
RETURN


Start処 理
ルーチン
プログラムを制御表
から探す。
PS2チェック
other
エラー・メッセージ
DXC WRITE
パラメータ・セット
=▼I▼ or ▼Q▼
N
他のグラフィック
プログラム
ランニング中
Y
QUIT 処 理
ルーチン・コール
Start コマンド
PC WRITE
パラメータ・セット
PC WRITE
=▼Q▼
=▼I▼
PS2チェック
スタート・メッセージ
DXC WRITE
パラメータ・セット
スタート・メッセージ
DXC WRITE
パラメータ・セット
PS2 ← ▼R▼
DXC WRITE
ルーチン・コール
(リカーシブル)
Return


ASK 処 理
ルーチン
制御表から,
プログラムを探す
メッセージ
DXC WRITE
パラメータ・セット
DXC WRITE
ルーチン・コール
(リカーシブル)
Return
DXC WRITE
ルーチン
リカーシブル・ルーチン
WRITE DXC
WAIT
完了チェック
エラー(誤応答)
DXC READ
ルーチン・コール
(リカーシブル)
Return

3.3 GCH（Graphic Communication Handler）

GCHは，H－８８１１プロセッサー内のGOS－Ⅲエグゼクティブの下に常駐し，次の機能を果している。

① DXC経由による，H－８４００ホスト・コンピュータとの間の相互データ転送。

② DXC経由で受け取った，キャラクタ・メッセージの表示。〔システム・レスポンス情報のディスプレイ〕

③ DXC経由で転送された，図形データ（Display Data）からディスプレイ・コマンド列への翻訳，表示。〔ディスプレイ・コマンド・コンパイリング〕

④ 表示図形に対する，ライトペン・ピッキングの受け つけとブリンキング処理。

⑤ 表示図形のエッジ・シザリング処理。

⑥ メッセージ・インジケータの表示と，オペレーション・メッセージ・ストリングの編集と転送。

⑦ トラッキング・クロスの表示と，ライトペンによる入力データの処理。

⑧ ファンクション・キー（NO.０～３１）の割込み制御。

⑨ ダイアル（１，２）の割込み制御。

⑩ CRTに表示されている図形のXYプロッタ上へのハード・コピー。

3.3.1 GCHにおける割込み制御

GCHでは次の６レベルの割込みを制御する。

① ライトペン・ピッキング	チャネル 0
② コマンド割込み	
③ エッジ割込み	
④ ファンクション・キー割込み	
⑤ ダイヤル割込み	
⑥ DXC割込み	チャネル 1

H－８８１１のコントロール・システムでは，これらの割込みに対し，チャネル０（①～⑤）とチャネル１（⑥）に優先順位をつけてはいるが，連続した割込みに対しては，５レベルまでの重複割込みを許し，last－in，first out としてコントロールしている。

GCHでは，DXC割込み処理中に，DXCに対して出力命令を出す可能性のある，ファンクション・キー，ライトペン割込みの一部，およびダイヤル割込みなどを禁止し

ておく必要がある。この目的で，ＤＸＣへのアクセスをロックするスイッチを設けている。

またライトペン，ファンクション・キー割込みルーチンで，ＤＸＣに対する出力命令を準備している間に，ＤＸＣ割込みが生じ，ＤＸＣ割込みルーチンが，ＤＸＣのパラメータ・エリアを入力用に書き換えることが想定し得るので，ＤＸＣ割込みルーチンは，ＤＸＣのパラメータ・エリアを先ずセーブしておいて，入力の処理を終了した後に復元しておく必要がある。

ファンクション・キー（ＮＯ.４～ＮＯ.３１）の割込みと，ライトペンのＳＥＮＤピッキングが重複した時には，ＤＸＣに２重の出力要求を出す可能性があるため，これも防がなければならない。ＤＸＣのロックのｏｎ，ｏｆｆでこれも制御する。

以上，ＧＣＨにおけるＤＸＣアクセス部分は図3.3.0のフローチャートのようになる。

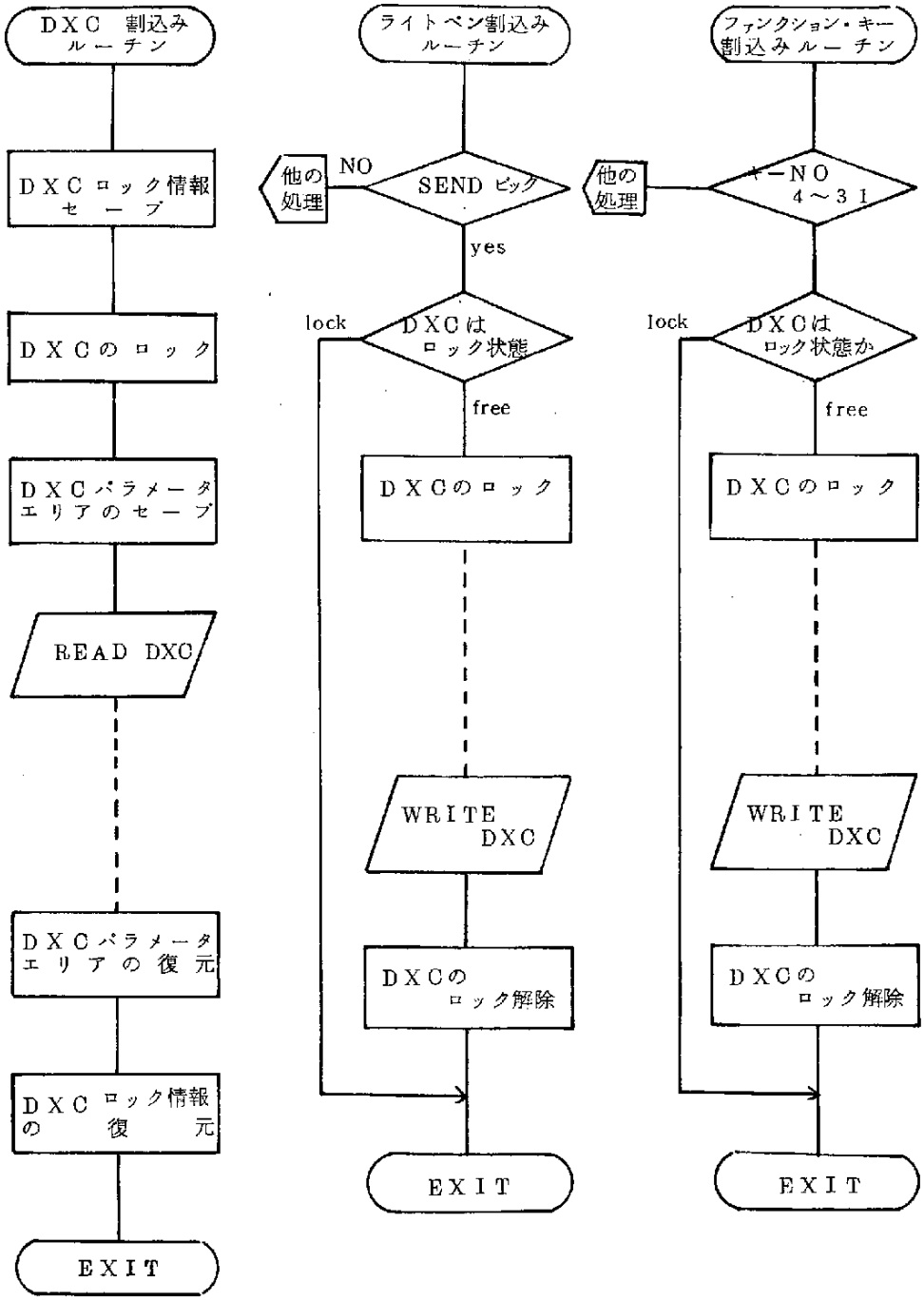


図 3.3.0　GCHにおける割込み処理フロー

3.3.2　ディスプレイ・コマンド・コンパイリング

前述したように，ＧＣＨの機能の中核に「ディスプレイ・コマンド・コンパイリング」がある。

ＧＣＨのこの機能は，Ｈ－８４００ホスト・コンピュータ内にあるユーザーのグラフィック・プログラムが編集し，ＧＪＭＯＮ→（ＤＸＣ）→ＧＣＨと経由して送ってきた図形データを，Ｈ－８８１３のＣＲＴ上に，ディスプレイ・ビームの軌跡として表示できるように，適切なディスプレイ・コマンド列に翻訳し，これを表示するものである。

ユーザーのグラフィック・プログラムから送られてくる図形データは，一定の様式に従っており，このフォーマッティングされた図形データをＤｉｓｔｉｌｌａｔｅｄ Ｄｉｓｐｌａｙ Ｄａｔａ（ＤＤＤ）と呼ぶ。

(1) ディスプレイ・コマンド

グラフィックＣＲＴへの図形表示（文字も含む）は，Ｈ－８８１１プロセッサの主記憶装置内につくられた，一連のディスプレイ・コマンド列を指定して，Ｈ－8813グラフィックＣＲＴの制御装置（Ｈ－８８１２）に対する出力命令をプログラムが発すると，ＣＲＴ制御装置が，ＣＰＵとは独自にディスプレイ・コマンドを先頭から読み出し，コマンドを解析して，ＣＲＴのビームの動きを制御することによって行われる。この様子を図3.3.1に示す。

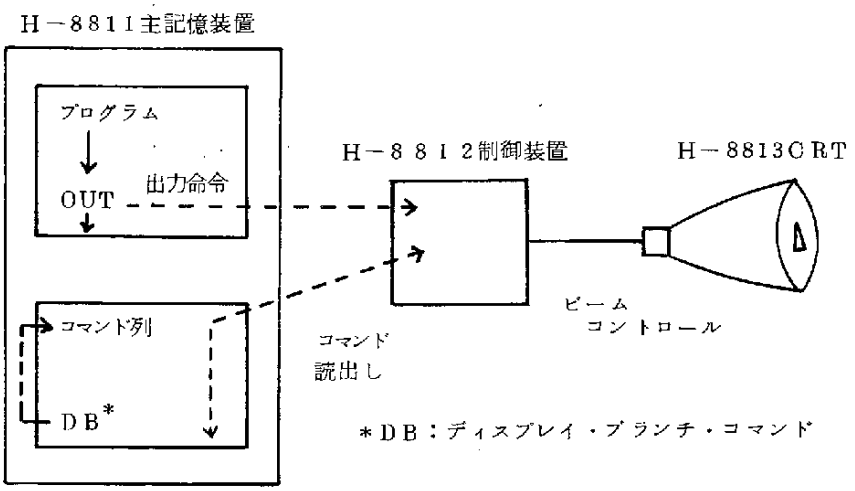


図3.3.1　ディスプレイの動作図

ディスプレイ・コマンドの種類には次のようなものがある。

① ビームの位置を動かすもの

（ブランク・ベクトルDAVB，DLVB，DSVB＊）

② ベクトルを表示するもの

（ベクトル　DAV，DLV，DSV）

③ 文字を表示するもの

（キャラクタ　DSC）

④ ディスプレイ制御装置のコマンド読出しの順序をかえるもの

（ディスプレイ・ブランチ等　DB　etc）

⑤ その他の制御コマンド

これらの図形コマンド列を一度実行させるだけでは，CRT画面上のビームの軌跡は一瞬光るだけで，短かい時間で消えてしまう。したがって，画像を表示し続けるためには，ディスプレイ制御装置が同じコマンド列を繰り返しトレースしていなければならない。（この機能をリフレッシュと呼ぶ）

この目的で，コマンド列の最後には，制御装置のコマンド読出しを，コマンド列の先頭に戻すコマンド（ディスプレイ・ブランチ・コマンド）が置かれていなければならない。

また，一連のコマンドを，サブルーチンのように定義して，同形の図形をいくつも表示することもできる。

これをディスプレイ・サブルーチンという。

ディスプレイ・サブルーチン制御の例を図3.3.2に示す。

＊これらのコードの詳細は3.3.2.3のエレメントに対応するコマンド参照

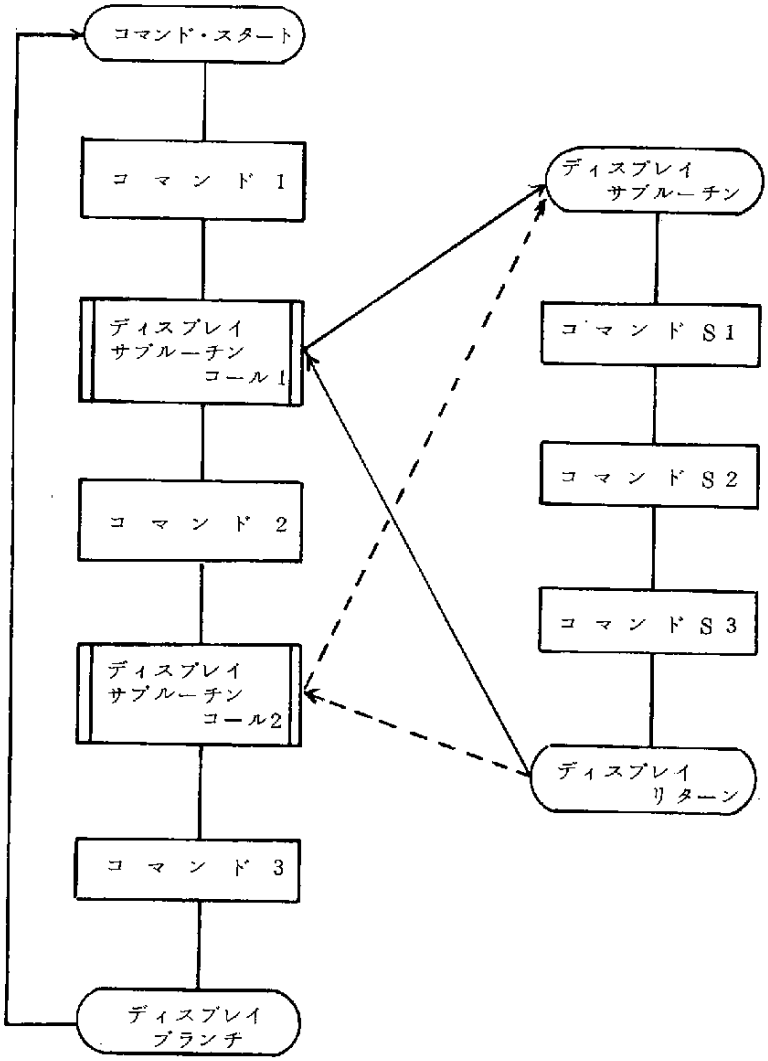


図 3.3.2　ディスプレイ・サブルーチン

〔2〕 図形データの形式（ＤＤＤ）

ＧＪＭＯＮ（Ｈ－８４００）から，ＧＣＨ（Ｈ－８８１１）へ，ＤＸＣ経由で転送されるデータのうち，ＴＦｉｄ（Ｔｅｘｔ Ｆｏｒｍ ｉｄｅｎｔｉｆｉｅｒ：ＤＸＣ転送データの第３バイト目）が０のものは，グラフィックＣＲＴの画面に表示すべき図形を，できるだけ少ない情報量でまとめた，特定の形式を持つ図形データ（ＤＤＤ：Ｄｉｓｔｉ－

lated Display Data）と定めている。

このＤＤＤは次の３つのレベルで構造化されている。

① フレーム

② エンティティ

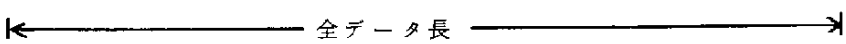
③ エレメント

このうち，フレームはＨ－８４００からＨ－８８１１への図形データ転送の単位であり，１フレームは１つのＤＤＤに対応している。

エンティティは，表示図形の論理的な単位でライトペンによるピッキング，ムービング，ディスプレイ・サブルーチンなどの単位になる。

エレメントは，一連の物理的な図形単位であり，点とか線などに対応する。

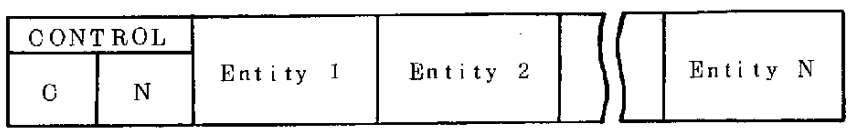
(1) ＤＤＤフォーマット



mℓ ： 全データ長（バイト数）で８～２０４０

ＴＦid ： ０の場合はＤＤＤであることを示す（詳細は3.4参照）
（ｎ）はバイト数を示す。

(2) Frameフォーマット

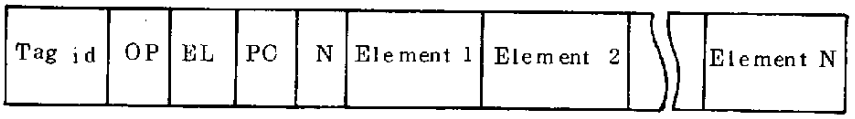


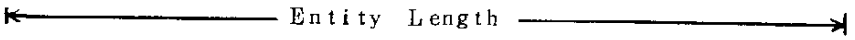
C ： フレームのモードを示す*
0：創成モード
1：更新モード
2：半創成モード

N ： フレーム中に含まれるエンティティ数

＊フレームのモードについては，次項のコマンド・コイバイリングを参照のこと。

(3) エンティティ・フォーマット





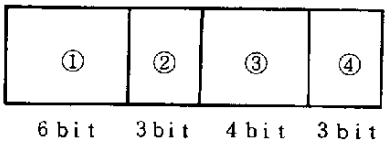
Tag id ： エンティティ・ネーム

OP ： オプション・ポインタ（任意使用）

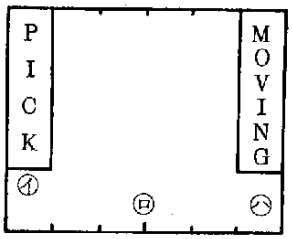
EL ： エンティティ長（ワード数 1W／2バイト）

PC ： エンティティの制御情報（picture control）

Picture Controlの内容



① エンティティの制御部分（MODE）*



㋑ PICK：ピッキングの指定

0：ピック不可

1：ピック可

㋩ MOVING：ムービング・モードの指定

0：非ムービング

1：ムービング

㋺ 000000 エンティティの削除

?1110? ノーマル・エンティティ

?11110 CRT制御変更指定

111111 サブ・ルーティン・エンティティ

*エンティティのモードについては次項コマンド・コンパイリングを参照のこと。

② 図形の方向情報を示す。（ＤＩＲＥＣＴＩＯＮ）

１００　標準方向（△X，△Yそのまま）
１０１　△X符号反転
１１０　△Y符号反転
１１１　△X，△Y符号反転

③ 図形の倍率を示す。（ＳＣＡＬＥ）

１０００　標準値（１倍）
１００１　２倍
１０１０　３倍
１０１１　４倍
１１００　５倍
１１０１　６倍
１１１０　７倍
１１１１　８倍

④ 図形の輝度を示す。（ＢＲＩＧＨＴ）

１１１　高輝度（標準値）
１１０　中輝度
１０１　低輝度
１００　ブランク

Ｎ　：　エンティティ中に含まれているエレメント数

(4) エレメント・フォーマット

Function Indicator	Component

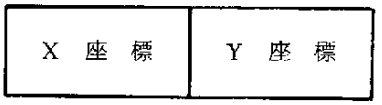
(2)

Function Indicator ： エレメントの種類を示す０～８までの数字

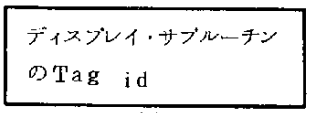
0　ビーム・アドレス・セット（absolute mode）
1　〃（relative mode）
2　ベクトル表示（absolute mode）
3　〃（relative mode）
4　ディスプレイ・サブルーチン・コール
5　キャラクタ表示（大文字）
6　〃（小文字）
7　任意曲線（incremental mode）
8　アキ
9　３次関数近似曲線

Component ： Function Indicator（F.I.）により異なる。

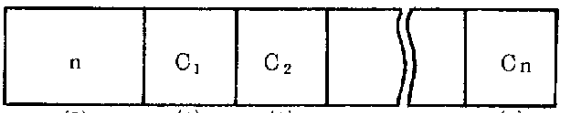
F.I.＝0～3の時　（アドレス・セット，ベクトル）



F.I.＝4の時　（ディスプレイ・サブルーチン・コール）



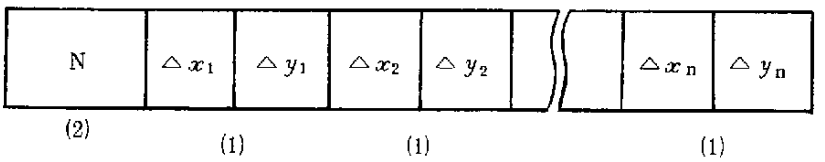
F.I.＝5，6　（キャラクタ表示）



n ： 文 字 数（2の整数倍）

c ： 文字コード

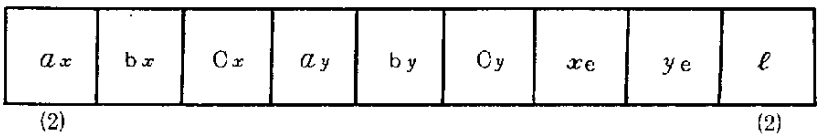
F.I.＝7の時　（任意曲線）



N ： △x，△yの組の数の4倍　（N＝4n）

△x，△y ： 符号を含んだ変位情報

F.I.＝9の時　（3次関数曲線）



3次関数は次式で与えられるもので

$$x = \frac{1}{6}a_y\ell^3 + \frac{1}{2}b_x\ell^2 + C_x\ell$$

$$y = \frac{1}{6}a_y\ell^3 + \frac{1}{2}b_y\ell^2 + C_y\ell$$

このエレメントの各係数の決め方については3.3.3.のDDAによるフィッティングの項参照

ユーザーのグラフィック・プログラム（H－８４００内）で編成され，以上の構成を持つ図形データはDXCを介してH－８８１１に転送され，GCH中のコマンド・コンパイラが，これを解析し，適当なディスプレイ・コマンドに翻訳して，CRTに表示する。

この図形データを編成するのは，グラフィック・プログラム中のユーザー・ルーチン，又はユーザー・プログラムに結合されたGSP（Graphic Subroutine Package）のサブルーチン群で，これをGJMONに転送するのは，GIASP（インタラクティブ・プログラミング・サポート・サブルーチン）の役目である。

〔3〕 コンパイリング・アルゴリズム

ディスプレイ・コマンド・コンパイラは，前項で述べたDDDを，フレームおよびエンティティのモードにしたがって，エンティティ単位に一連のディスプレイ・コマンドに翻訳していく。

ディスプレイ・コマンド用のエリアとしては，４Kワード分を用意しているが，この限られたスペースを有効に生かすために，空領域の管理を行っている。またピッキングやムービング，それに表示図形の更新のために，エンティティの登録テーブル，ディスプレイ・サブルーチンの登録テーブル，ムービング・コントロール・テーブルなどがある。

● フレームのモードについて

H－８４００のグラフィック・プログラムから送られてくるDDDのフレームは，次のような三つの状態のモードを有している。

1） 創成モード

2） 更新モード

3） 半創成モード

１）の創成モードは，まったく新しい画面をつくり作すものであり，前に表示されていた図形があっても，その図形はこわされて，新しい図形が表示される。

２）の更新モードは，すでに表示されている図形に対し，エンティティの追加，置換，削除およびピクチュア・コントロール情報の変更などを行うものである。この更新機能により表示図形を，エンティティ単位に自由にモディファイすることができる。

３）の半創成モードは，サブルーチン・エンティティのディスプレイ・コマンド

だけを残して，他はすべてつくり直すものである。

すなわち，ライト・ボタンや，よく使用する図形要素をサブルーチン・エンティティとして，あらかじめ登録しておいて（創成あるいは更新モードで），後でそれらをコールするメインの画面のみをつくり直すような場合に利用される。

2），3）の機能によりH－8400から転送する情報の量をかなり少なくすることができる。

● エンティティのモードについて

各モードのフレーム中に含まれるエンティティは次のような種類のエンティティ・モードを有する。

1） 創成モード，半創成モードのフレーム

① ノーマル・モード・エンティティ （normal）

2） 更新モードのフレーム

② 追加モード・エンティティ （Add）

③ 削除モード・エンティティ （remove）

④ 置換モード・エンティティ （replace）

⑤ 変更モード・エンティティ （change CRT）

さらに①，②，④のエンティティには，そのエンティティがサブルーチン・エンティティであるか否か，それにムービング*1 指定があるかないかが加わり，①，②，④，⑤にはピッキング*2 を許すかどうかの指定が加わる。

次に②～⑤の更新モードのエンティティについて説明を加える。

②の追加モード・エンティティは，すでに表示されている図形に対して，エンティティの追加を行うものである。サブルーチン・エンティティに対しての更新は，この追加モードのみを許している。

③の削除モード・エンティティは，すでに表示されている図形のある一つのエンティティを取り去るものである。

④の置換モード・エンティティは，すでに表示されている図形のある一つのエンティティを新らしく送られてきたエンティティと交換するものである。

*1 ムービングは次項および次節のムービング処理参照。

*2 ピッキングは次項および次節のピッキングとプリキング参照

⑤の変更モード・エンティティは，すでに表示されている図形のある一つのエンティティのピクチュア・コントロールの内容を変更するもので，たとえばピッキング指定の変更や，図形の方向，倍率，輝度の変更等を行う。

● コマンド・コンパイラ

ディスプレイ・コマンドをジェネレートする過程で必要となる各種領域およびテーブルにつき，まず説明する。相互の関係を図3.3.3に示す。

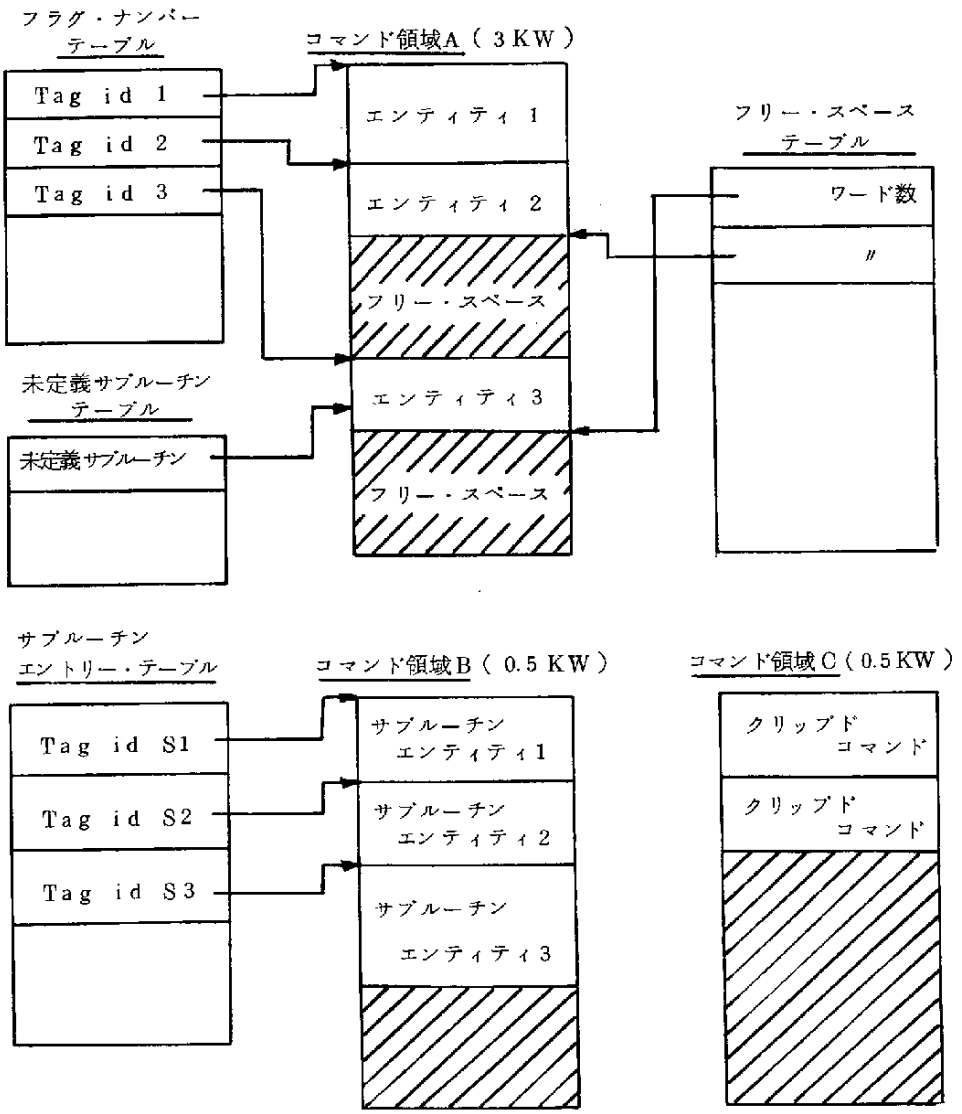


図3.3.3　各種テーブルとコマンド領域の関係

コマンド領域Aは，ディスプレイ・サブルーチン以外のエンティティのコマンド列をつくり出していくエリアで，コマンド領域Bはディスプレイ・サブルーチン・エンティティ用のエリア，コマンド領域Cは，エッジ・インタラプトをおこした時の修正コマンド（クリップド・コマンド）用のエリアである。

これらのエンティティがつくり出されるとそのコマンド・アドレスと使用語数などが，フラグ・ナンバー・テーブル（図3.3.4）に登録される。

FTBL

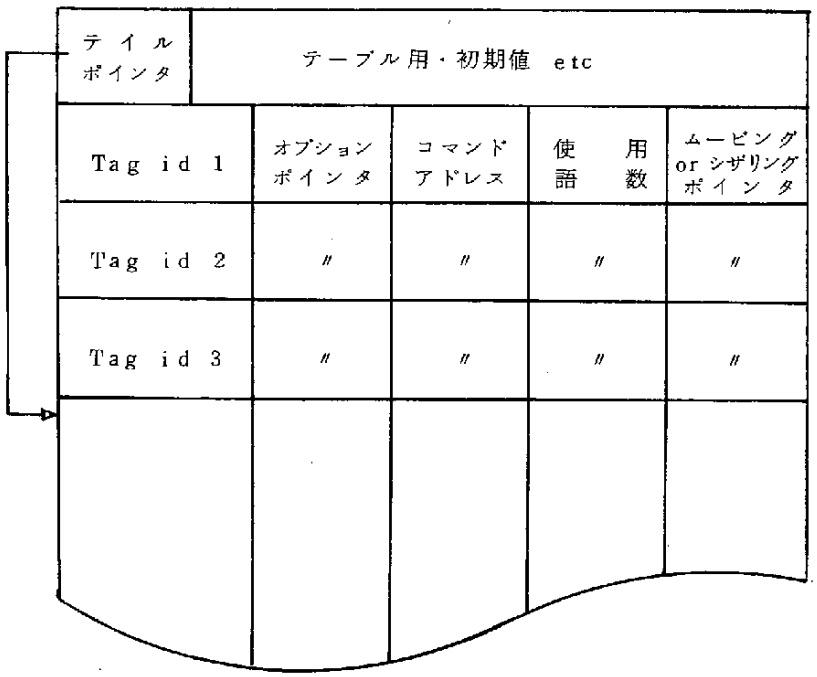


図3.3.4 フラグナンバー・テーブル・フォーマット

フラグ・ナンバー・テーブル中のムービング・ポインタは，そのエンティティがムービング指定されている時のムービング・コントロール・テーブル（ムービング処理参照）へのポインタであり，シザリング・ポインタはシザリングされたクリップド・コマンドへのポインタ（エッジ・シザリング参照）である。

ディスプレイ・サブルーチン・エンティティも同じようにコマンド・アドレスと使用語数が，ディスプレイ・サブルーチン・エントリーテーブル（図3.3.5）に登録される。

CSUB

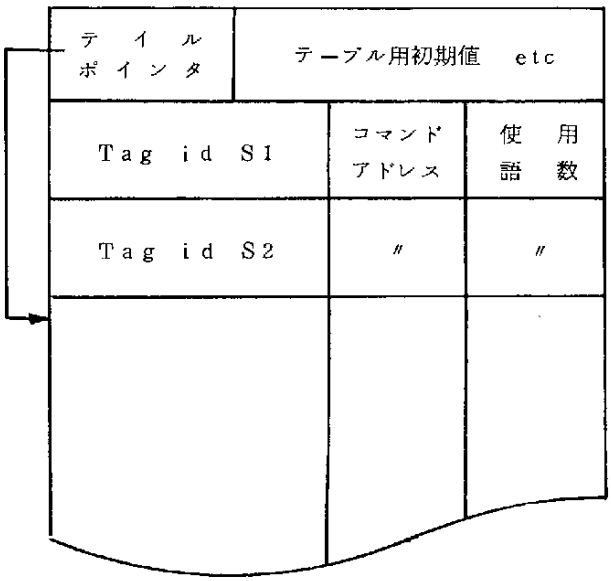
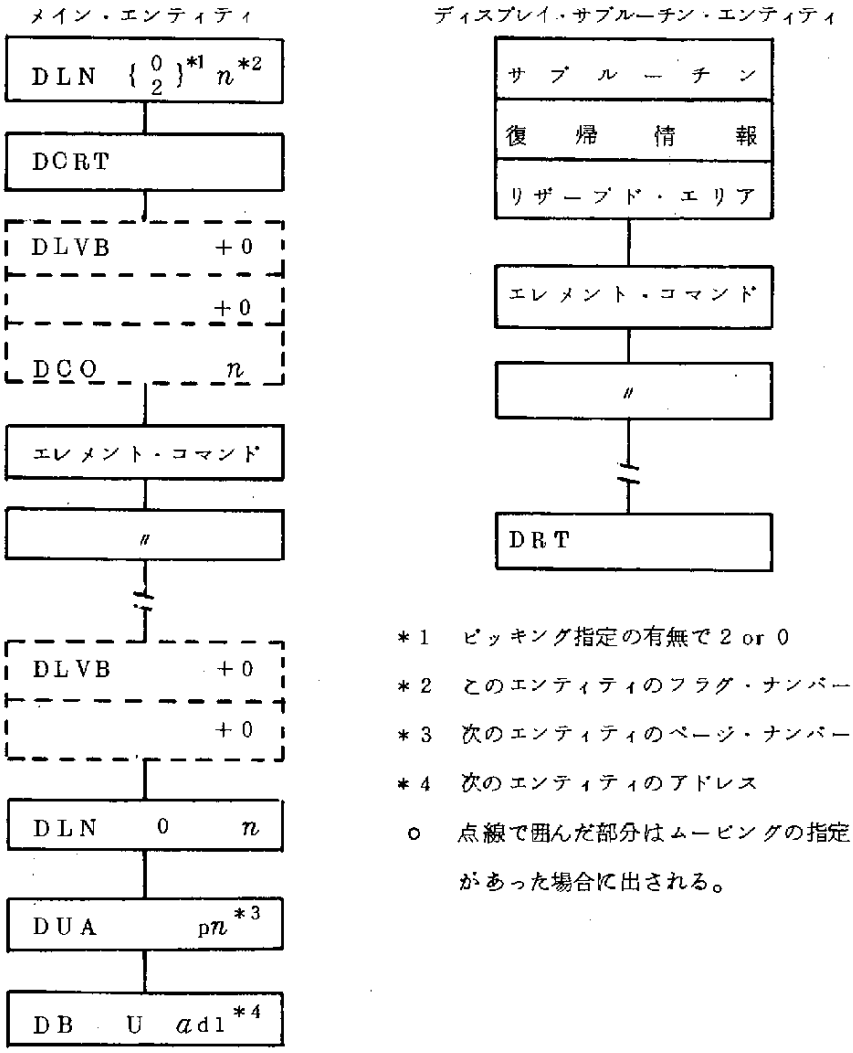


図3.3.5 ディスプレイ・サブルーチン・エントリー・テーブル

この時，未定義のディスプレイ・サブルーチン・エンティティに対してサブルーチン・コールが行なわれた時には，その未定義のディスプレイ・サブルーチン・エンティティをペンディング・サブルーチン・テーブルに登録しておき，そのサブルーチンが定義された時点で，サブルーチン・コールが行なわれるようにしている。（未定数サブルーチン処理参照）

コマンド・コンパイラは前記のフレーム・モードにしたがって，更新モードならC領域を初期化し，半創成モードならA領域，フラグナンバー・テーブル，フリー・スペース・テーブルとC領域を初期化する。また創成モードの場合は，B領域とコマンド・サブルーチン・エントリー・テーブルとA，C領域関係の諸情報を初期化する。その後フレームからエンティティを一つづつとり出してエンティティの各モードにしたがって，コマンド領域A，Bにエレメントの各ファンクションに対応したディスプレイ・コマンドをつくり出していく。

このようにして，エンティティごとにジェネレートされたコマンド列を図3.3.6に示す。



＊1　ピッキング指定の有無で 2 or 0

＊2　このエンティティのフラグ・ナンバー

＊3　次のエンティティのページ・ナンバー

＊4　次のエンティティのアドレス

o　点線で囲んだ部分はムービングの指定があった場合に出される。

図 3.3.6　1エンティティに対応するコマンド列

このコマンド列の中で，〔DLN　　n〕というコマンドは，このエンティティに対して一意のフラグ・ナンバーを与えるもので，例えばライトペンによるピッキングの時等はこのフラグ・ナンバーをもとにして，各テーブルが参照できるようになっている。

〔DCRT　　〕コマンドは，このエンティティに対するCRTの制御コマンドで，各エンティティの図形の方向，倍率，輝度を制御する。

（Ｐ２６(3)②～④参照）

次の〔ＤＬＶＢ，ＤＣＯ〕コマンドはムービング処理のためのものであり詳細はムービング処理で述べている。

〔ＤＵＡ，ＤＢ〕コマンドは，次のエンティティに対するリンクのためのコマンドである。

また，ディスプレイ・サブルーチンに対しては，最初の３ワードに復帰の為の情報が一時セットされるため，そのエリアをとっておき，最後に〔ＤＲＴ〕コマンド（サブルーチン・リターン・コマンド）を出している。

各エレメントに対するコマンドは，そのエレメントのファンクション（Ｐ２６）にしたがって図3.3.7に示されるようなコマンド群をジェネレートしていく。

Ｆ.Ｉ.＝０　（ビーム・アドレス・セット：アブソリュート・モード）

ＤＡＶＢ ０１００	1	0	X
０１００	1	1	Y

Ｆ.Ｉ.＝１　（ビーム・アドレス・セット：リラティブ・モード）

$\triangle x$，$\triangle y$の絶対値が共に＜３２のとき

ＤＳＶＢ ０００１	SX	$\triangle x$	SY	$\triangle y$

ＳＸ，ＳＹは符号ビット

$\triangle x$，$\triangle y$の絶対値が共に≧３２のとき

ＤＬＶＢ ００１１	0	SX	$\triangle x$
００１１	1	SY	$\triangle y$

F．I．＝2　（ベクトル表示　：　アブソリュート・モード）

DAV 0100	0	0	X
0100	0	1	Y

F．I．＝3　（ベクトル表示　：　リラティブ・モード）

△x，△yの絶対値が共に＜32のとき

DSV 0000	SX	△x	SY	△y

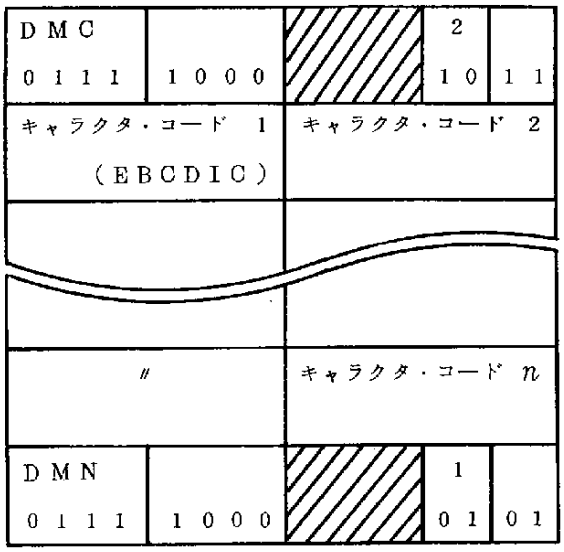
△x，△yの絶対値が共に≧32のとき

DLV 0100	0	SX	△x
0010	1	SY	△y

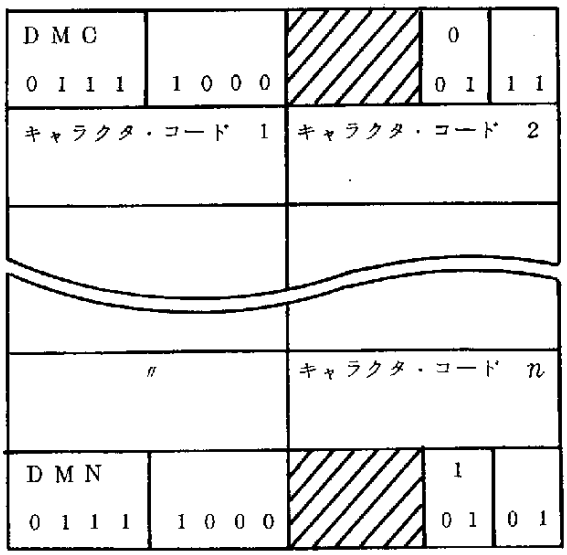
F．I．＝4　（ディスプレイ・サブルーチン・コール）

DUA			サブルーチン ページ
DRS 1010		1	サブルーチン アドレス

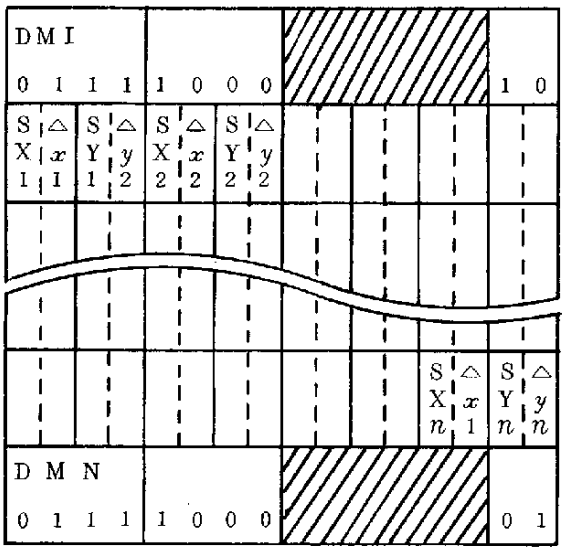
F.I.＝5　（キャラクタ表示　：　大文字）



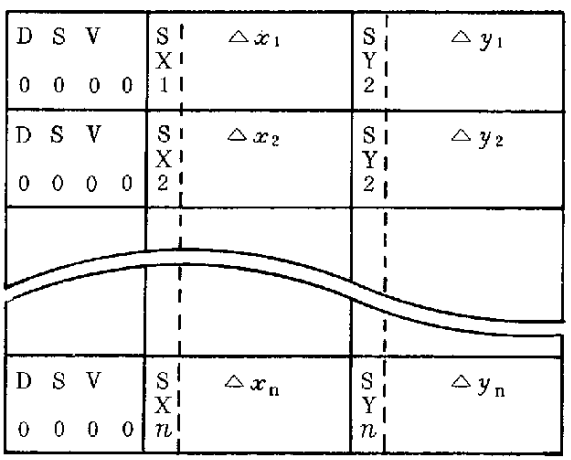
F.I.＝6　（キャラクタ表示　：　小文字）



F．I．＝7　　（任意曲線，インクリメンタル・モード）



F．I．＝8　　（3次関数近似曲線　：　ショート・ベクトル）



$\triangle x_n$，$\triangle y_n$ の値は曲率によって異なる。

図3.3.7　エレメントに対応するコマンド・ストリングのパターン

コマンド・コンパイラのフローチャートを図3.3.8，図3.3.9，図3.3.10に示す。

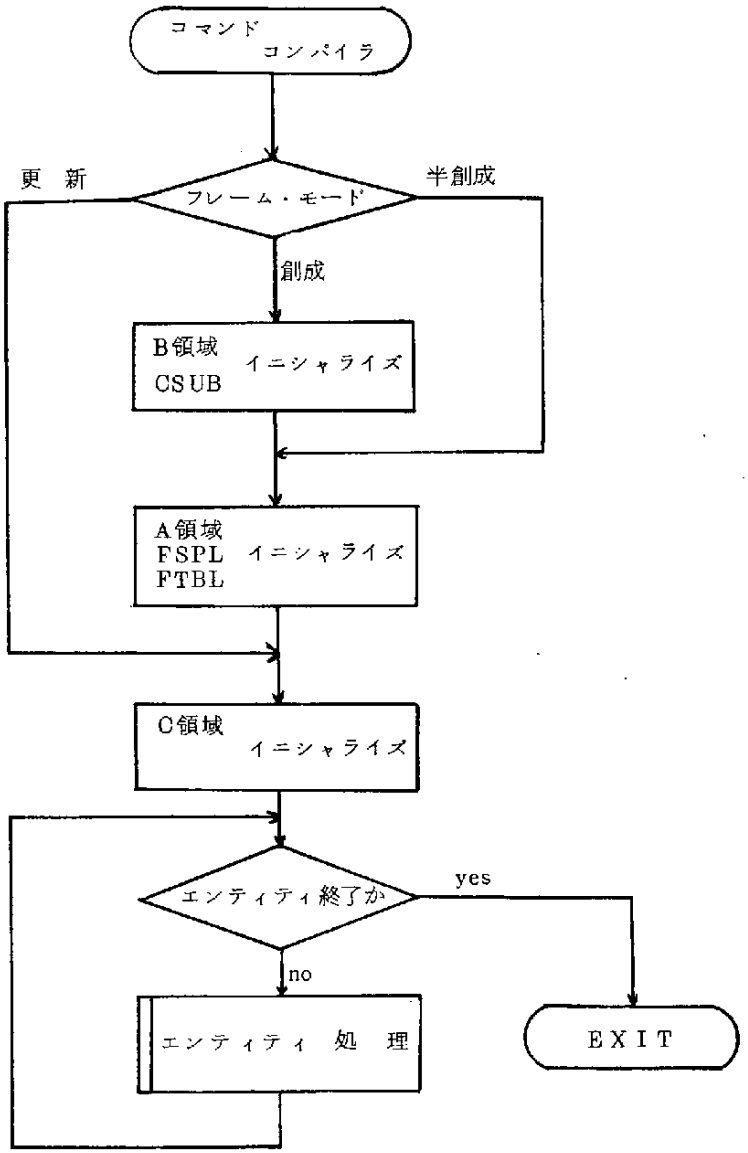


図3.3.8 コマンド・コンパイラ・メインルーチン

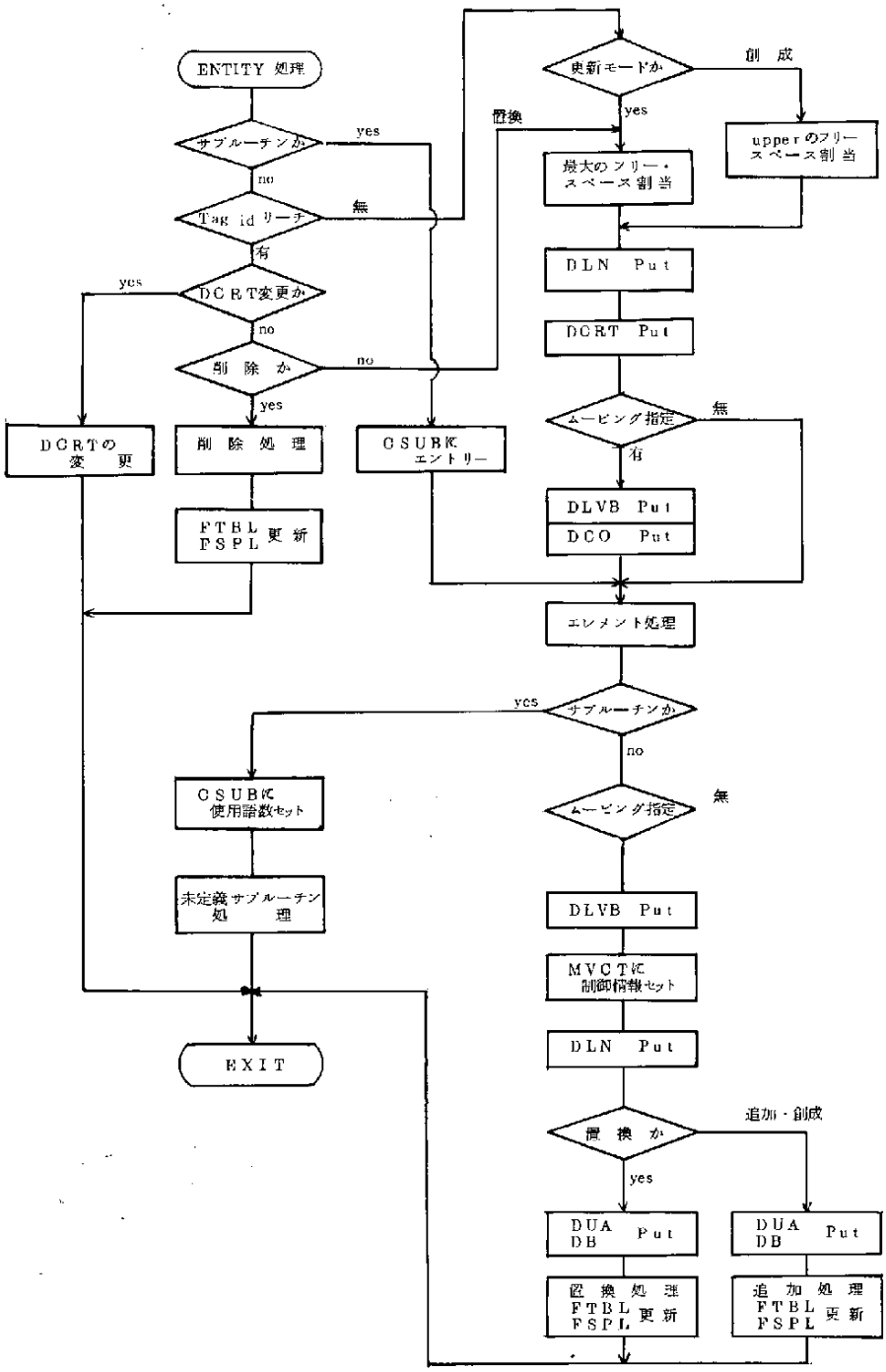
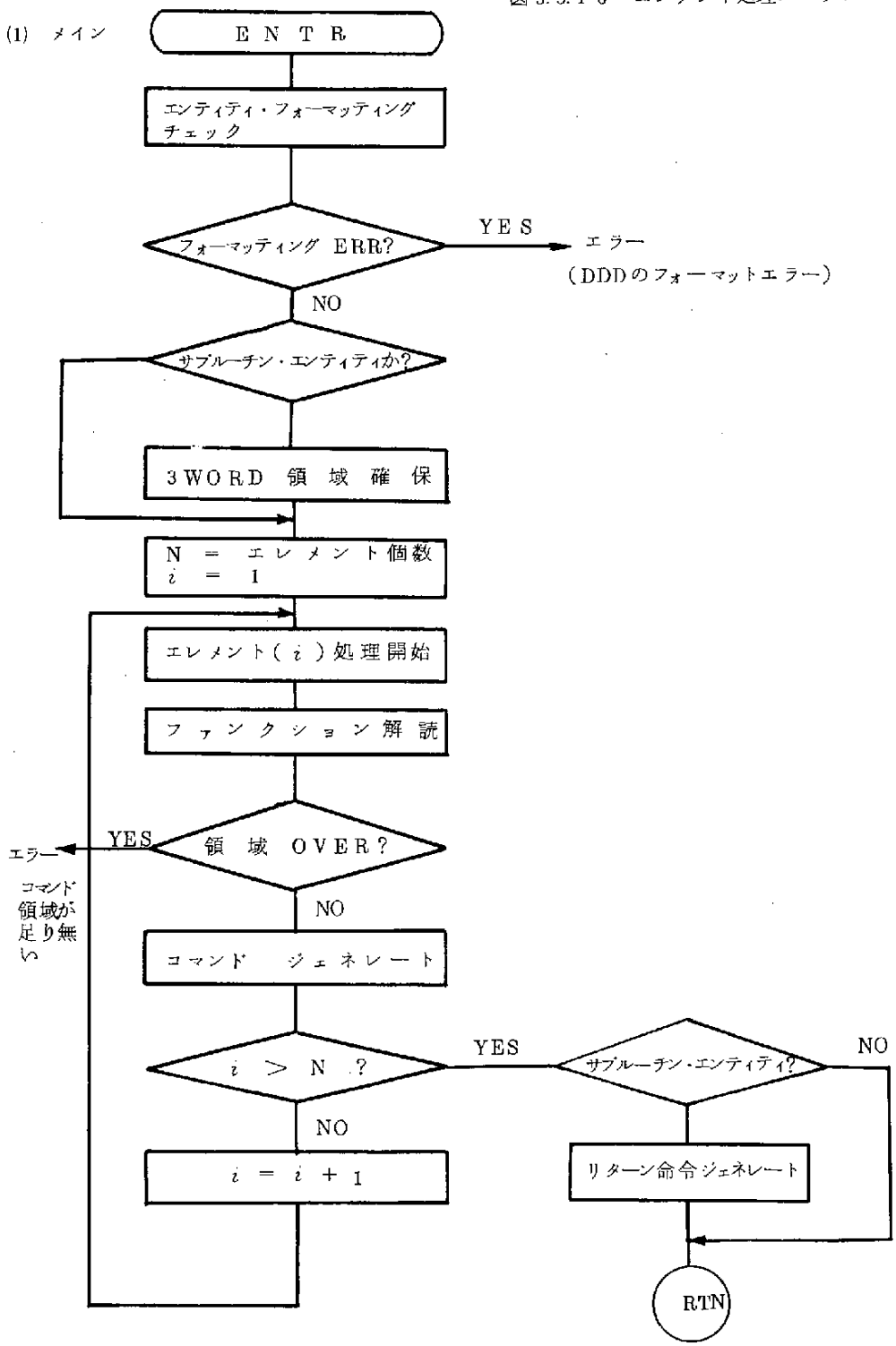


図 3.3.9　エンティティ処理ルーチン

図 3.3.10　エレメント処理ルーチン

(1)　メイン



(2) ファンクション解読とコマンド・ジェネレーションの過程

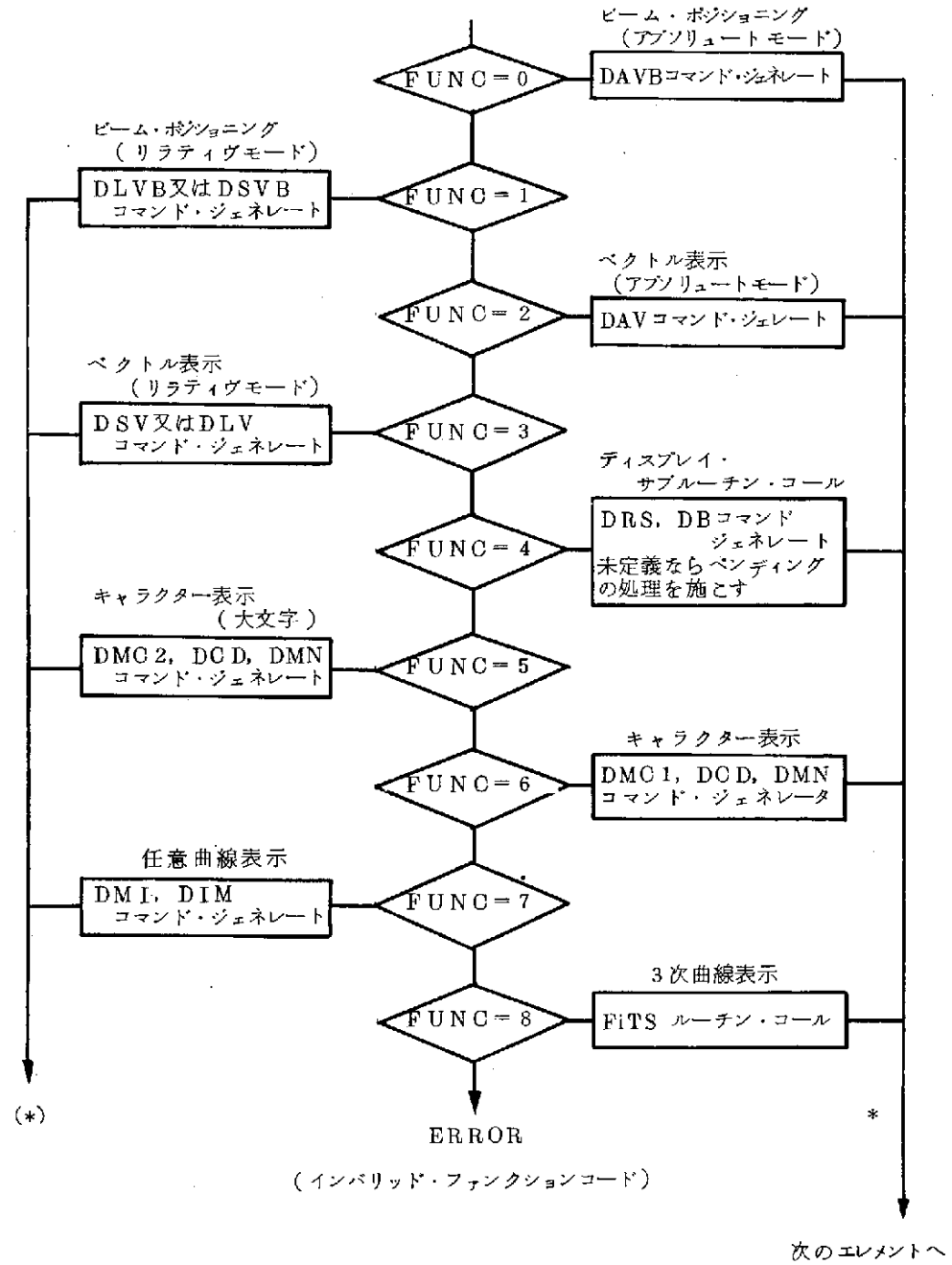


図3.3.10　エレメント処理ルーチン

● 空領域の管理

コマンド・コンパイラでは，限られたコマンド・エリアを有効に利用するため，次のような方法で，空領域の管理を行っている。

コマンド・エリアはA，B，Cの３つの領域に分割されている。C領域は，クリップド・コマンド用のエリアであり，新しいフレームが送られてくるたびに初期化するので，シークェンシャルに増えるカレントなポインタと，C領域の最終アドレスのみチェックしていれば良い。またB領域は，ディスプレイ・サブルーチンに対しては追加のみ認めているので，これもC領域と同じ管理法で良い。

しかしA領域は，エンティティの更新を受けるので，途中にこま切れの空領域がいくつもできる可能性がある。従って，A領域に関しては，フリー・スペース・テーブル（図3.3.11）を設け，そのこま切れのフリー・スペースと，まだ未使用の連続しているフリー・スペース（upper free space）を登録しておき，置換エンティティ，削除エンティティ等により新しくできたフリー・スペースが，既に登録してあるフリー・スペースと隣接している時には，つなぎ合わせて一つの連続したフリー・スペースとする等の管理を行っている。

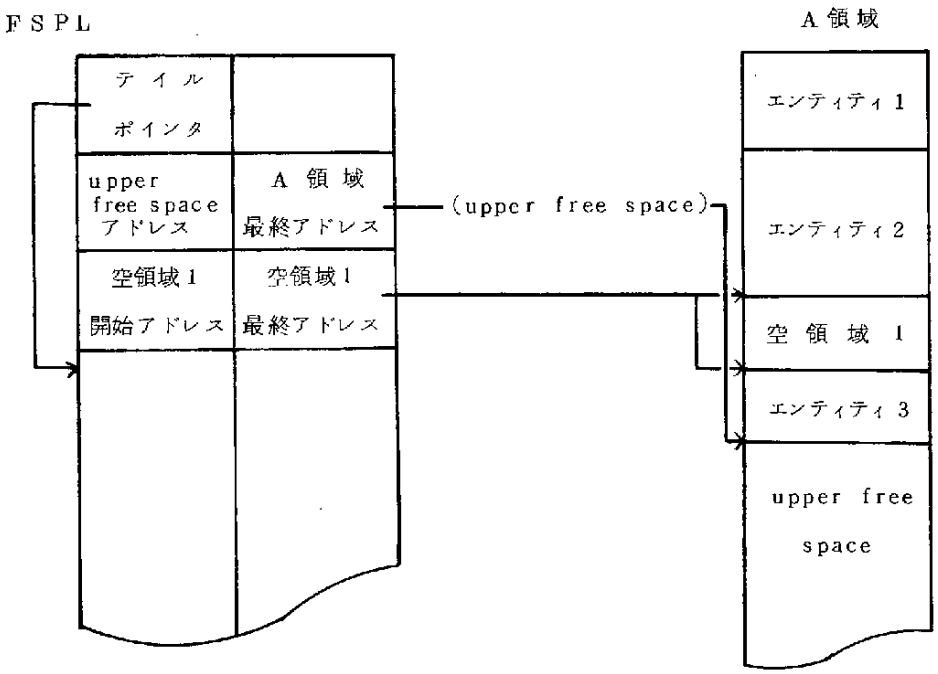


図3.3.11　フリー・スペース・テーブル

● ペンディング・サブルーチンの処理

未定義のディスプレイ・サブルーチンがコールされた時は，ペンディング・サブルーチン・テーブル（図3.3.12）に，その未定義サブルーチンの名前と，参照しているアドレスを次々と登録しておき，A領域のサブルーチン・コールのコマンドの替わりに，ダミーのコマンドをジェネレートしておく。

そしてそのサブルーチンが定義された時点で，ペンディング・サブルーチン・テーブルを参照して，その未定義サブルーチンをコールしているコマンドに対して，定義されたサブルーチンのアドレスを埋め直すという方法をとっている。

但し，この場合サブルーチンのリカーシブル・コールを防ぐため，自分自身のコールと未定義サブルーチン中での未定義サブルーチン・コールは禁じている。

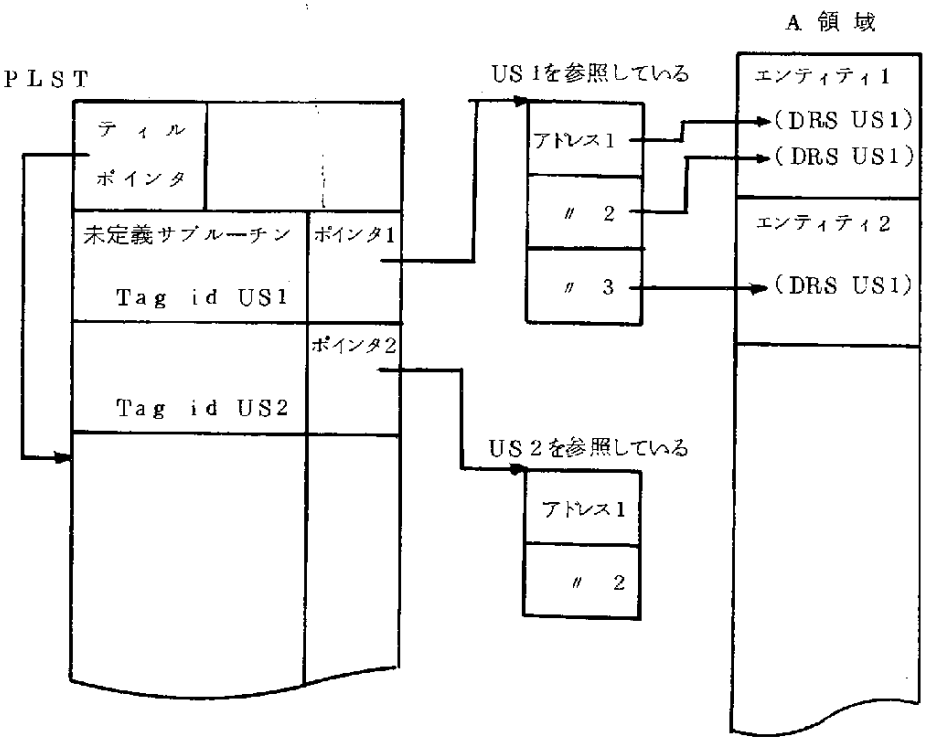


図 3.3.12 ペンディング・サブルーチン・テーブルとA領域の関係

3.3.3　GCHにおけるその他の処理

● ムービング処理

あるエンティティに対して，ムービングを指定すると，そのエンティティ内に含まれるムービング・コントロール情報（△X，△Yの移動点列と，ループ情報で構成される）にしたがって，CRT画面上を，平行移動させることができる。

このコントロール情報は任意個のループ・ブロックを含むことができ（最大254語），1ループ・ブロックは，移動点列と，ループ情報を含み，入れ子になったループ・ブロックが許される。

ムービング・コントロール情報の形式を図3.3.13，ループ・ブロックの基本形式を図3.3.14のように定めている。

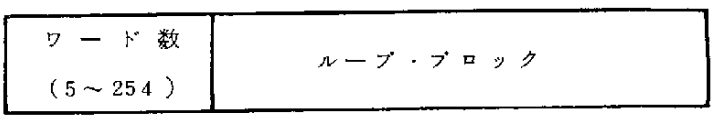
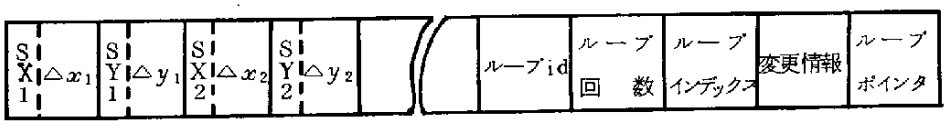


図3.3.13　ムービング・コントロール情報の形式



SXi，SYiは△x，△yの符号を示す

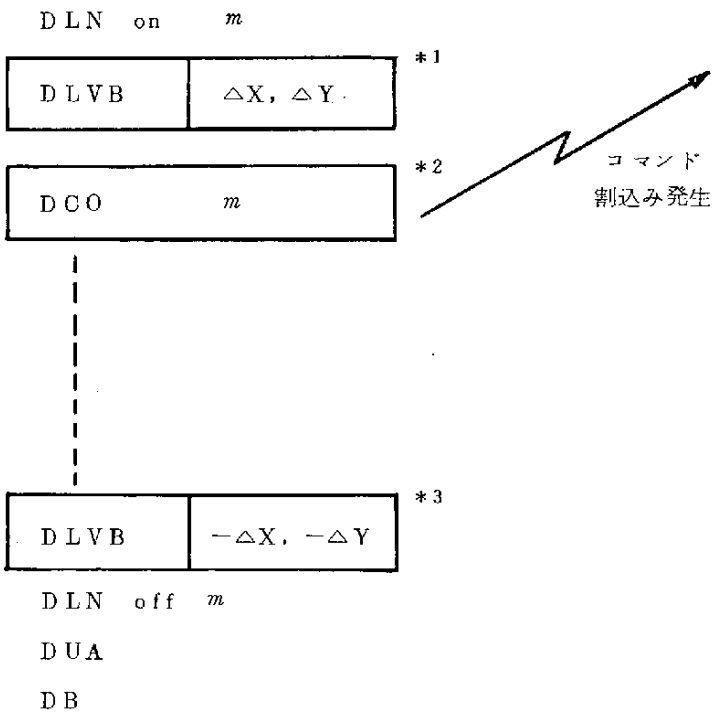
△xi，△yiは移動の絶対量（ラスター値：最大126）

変更情報：ループ時のSXi，SYiの変更情報と，

△x，△yの交換情報を含む。

図3.3.14　ループ・ブロックの基本形式

ムービングを指定されたエンティティは，コマンド・コンパイルの時点で，コマンド列中に2つのＤＬＶＢ（ロング・ベクトル・ブランク・コマンド）と1つのＤＣＯ（コマンド・インタラプション・コマンド）をつくり出しておく。（図3.3.15参照）さらに，ムービング・コントロール情報をムービング・コントロール・テーブルに登録し，そのポインタをフラグ・ナンバー・テーブルに登録しておく。これらの関係を図3.3.16に示す。



＊1　△Ｘ，△Ｙ　：　1ステップ分の変位

＊2　*m*　：　エンティティのフラグ・ナンバー

＊3　－△Ｘ，－△Ｙ　：　変位分を戻しておく

（他のエンティティに影響を与えないように）

図3.3.15　ムービング・エンティティのコマンド列

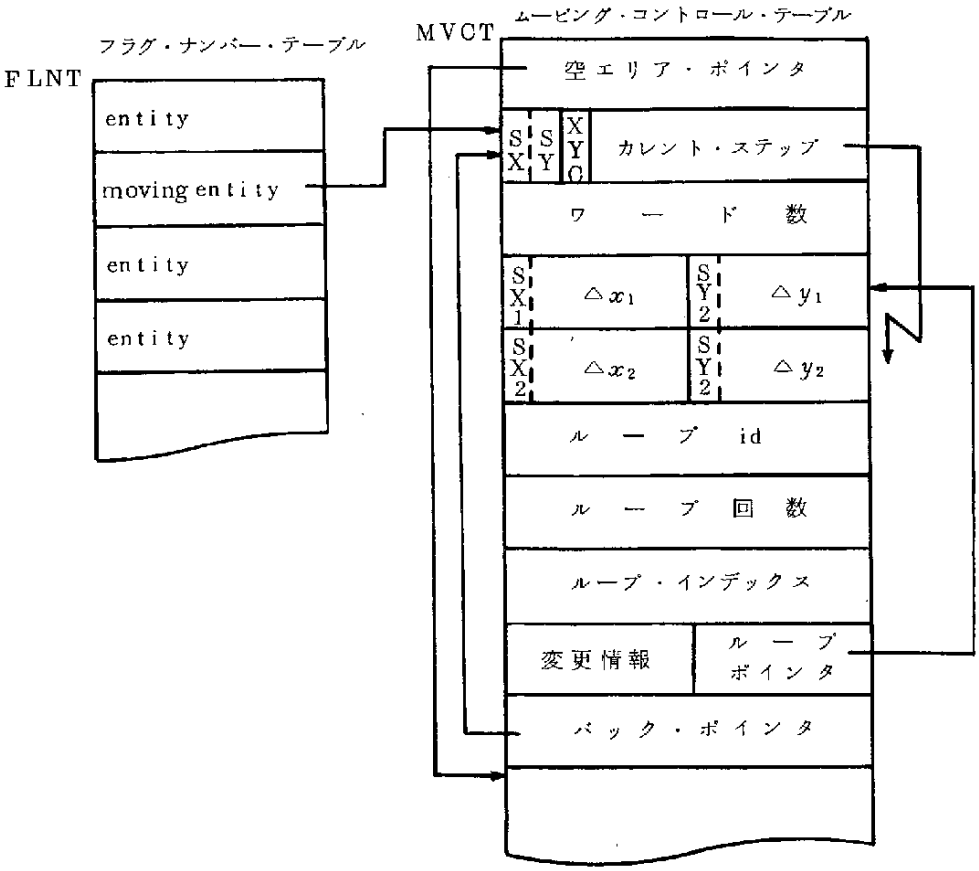


図3.3.16　ムービング・コントロール・テーブル

これらの情報をもとにして，ムービングの処理は次のような手順で行っている。

① CRT制御装置のコマンド・フェッチ中にDCOコマンドがみつかると，コマンド割込みがかかる。

② 割込み発生時のコマンドのフラグ・ナンバーから，このエンティティ中のDLVBコマンドの位置と，ムービング・エンティティのコントロール情報（MVCT中の）のアドレスを得る。

③ このエンティティに対するコントロール情報のカレント・ステップ，SX_i $\triangle x_i$，SY_i $\triangle y_i$ とSX，SY，XYC などをもとにして，コマンド列中のDLVBの△X，△Y情報を書きかえる。

④ これら①～③の処理を規定のループ回数だけ繰り返し，ループが終了したらムービングは終り，図3.3.17のようなムービング終了情報をH－8400側のユーザー・プログラムに連送する。

メッセージ長	TFid		エンティティ名	オプション	完了コード
12	04			ポインタ	(0 or 1)
(2)	(1)	(1)	(4)	(2)	(2)

図3.3.17　ムービング終了情報

このムービングの機能を利用して，ユーザーは，H－8811側のみで，あるエンティティのリアル・タイム・ムービングを行うことができ，たとえば，シミュレーションにおけるトランザクションの動きなどを表示するには有効である。

● ライトペン・ピッキングとブリンキング処理

表示図形に対するライトペンのピッキングは，ピックされた図形（エンティティ）のコマンド・アドレスとそのエンティティのTag id. を捕捉するとともに，ピッキングが適切に行われたかどうかの確認する必要がある。

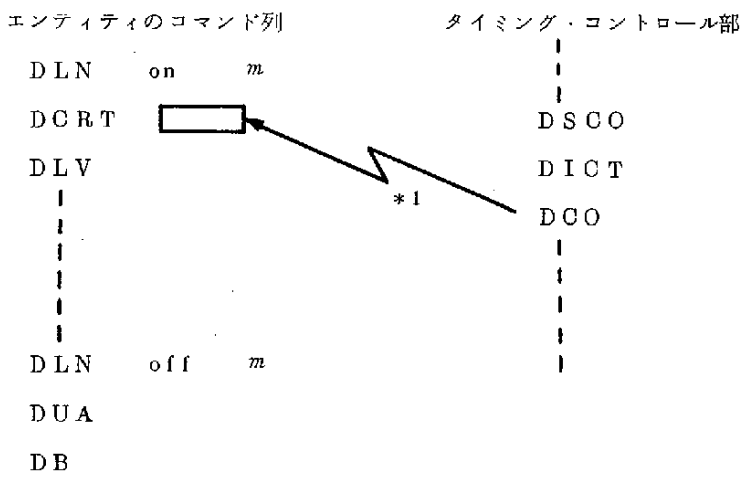
その確認の手段としては，適当な時間々隔で（0.5～1 sec位）ピックされた図形を明滅させるのが良い方法であろう。（これをブリンキングという）

ライトペンで，ある図形がピックされると，ディスプレイ制御装置に割込みがかかり，その図形コマンド列に対応するフラグ・ナンバー（DLNコマンドのフラグ）を返してくれる。そのフラグ・ナンバーによって，フラグ・ナンバー・テーブルを参照すれば，その割込みが，どのエンティティで起ったかがわかる。ユーザーが，本当にそのエンティティのピックを望んだかどうかを確認するためのブリンキング処理は次のように行っている。

ディスプレイ制御装置が，ディスプレイ・コマンド列をフェッチする際，何回かの周期で，一つのエンティティの図形コマンドのピクチュア・コントロール・コマンド（DCRT）をブランク表示に置き換えてやる。

この周期はDSCO（セット・カウンタ：ある数をセット）と，DICT（インクリーズカウンタ・アンド・テスト：カウンタを増してテストする）それにDCO（セット・コマンド・インタラプション）コマンドを組み合わせてタイミングをとってい

る。



＊1　コマンド割込みが起ったらエンティティ・コマンド列の
DCRTコマンドを書きかえる。

図 3.3.18　プリンキング処理

また，一つのフラグ・ナンバーに対応するエンティティは，そのエンティティの一部が画面のエッジからとび出ていて，コマンド領域Cに修正されたコマンドが登録されている場合もある得る。（次項エッジ・シザリングを参照）この場合はフラグ・ナンバー・テーブルのクリップド・コマンドのアドレスを参照して，クリップド・コマンドもプリキングさせる必要がある。

プリンキングが行なわれて，グラフィック・コンソールのオペレータが確認したピッキング情報は，オペレータがファンクション・キーNO.4を押すことにより，H－8400側のユーザー・プログラムに転送される。

そのフォーマットを図3.3.19に示す。

メッセージ長 20	TFid 01	00	エンティティ名	オプション ポインタ	フラグ ナンバー	XPR	YPR	DXR	DYR
(2)	(2)		(4)	(2)	(2)	(2)	(2)	(2)	(2)

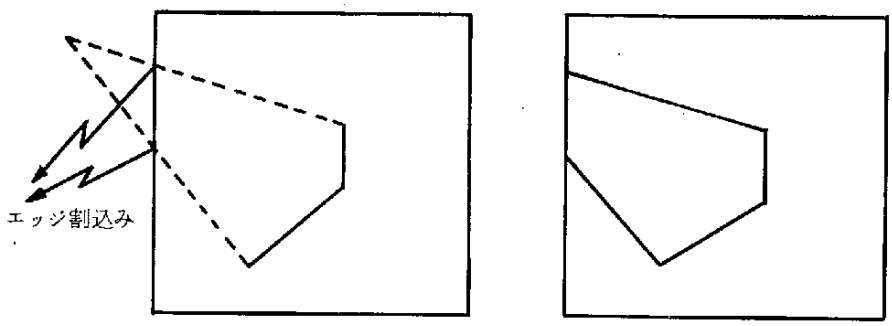
図 3.3.19　ピッキング情報の形式

● エッジ・シザリング処理

ＣＲＴの画面は，一辺が２５cmの正方形で，その平面内を０～１０２３のメッシュに区切り，ビームのアドレスとしては，その格子点の座標を指定することになっている。

しかし，ユーザー・プログラムの不注意で，その画面内から，はみ出すようなコマンドが実行されることがある。このような場合，いわゆる"エッジ割込み"がかかり，エッジにかかったベクトルは表示されて，消えてしまう。この時，エッジにかかったベクトルの，画面内にある部分だけを表示させてやるような処理を，エッジ・シザリングあるいはクリッピングなどと呼んでいる。（図3.3.２０参照）

ＧＣＨでは次のような方法でエッジ・シザリング処理を行っている。



（a）エッジ割込を起した図形　　（b）クリッピングした図形

図 3.3.２０　エッジ・シザリング処理

① コマンド・コンパイラがコマンドをジェネレートして，そのフレームが表示されたら，その図形がエッジ割込みを起したかどうかを覚えておく。

② Ｈ－８４００側のユーザー・プログラムから，今送ったフレームに対してエッジ・シザリングの処理を施こして欲しいという指令があったら，そのフレームがエッジ割込みを起したかどうか調べ，もし起していなければ何もしないで処理を終了する。

③ エッジ割込みがあれば，すでにつくり出した図形コマンドをシークエンス順に，すべてトレースして，エッジにかかったコマンドの修正コマンド（画面内の部分）をＣ領域につくり出していく。

この時，この修正コマンドは先頭にエッジ割込みを起したエンティティと同じＤＬＮ，ＤＣＲＴコマンドをつけておく。

④ 修正コマンドのアドレスを，フラグ・ナンバー・テーブルの該当エンティティに登録しておく。

このポインタは，ピックされたエンティティのブリンキング処理の時必要になる。（前項参照）

②の，エッジ・シザリング指令は，ファンクション・キーNO.8でも代行することができる。

これらの処理を施こした後は，エッジ割込みを禁止状態にしてコンティニュー・ディスプレイ・コマンドを実行させておく。

この禁止状態は次のフレームが送られてきた時に解除される。

● ＤＤＡによるカーブ・フィッティング

ＣＧＯＳの下では，グラフィック・プログラムが，ＣＲＴに表示するフィッティング・カーブ（複数個の点列を結ぶ，なめらかな曲線）を，Ｈ－８８１１サテライト・コンピュータに備わったＤＤＡ（Digital Differential Analyzer）を利用して表示する方法を採用している。

このためには，先づ，グラフィック・プログラムにバインドされるＧＳＰの１モジュールであるＧＣＲＶＦＴルーチンで，与えられた点列を，多項式で補間し，各点列間の曲線を，その距離の３次関数に置き換えて，その係数を定める。

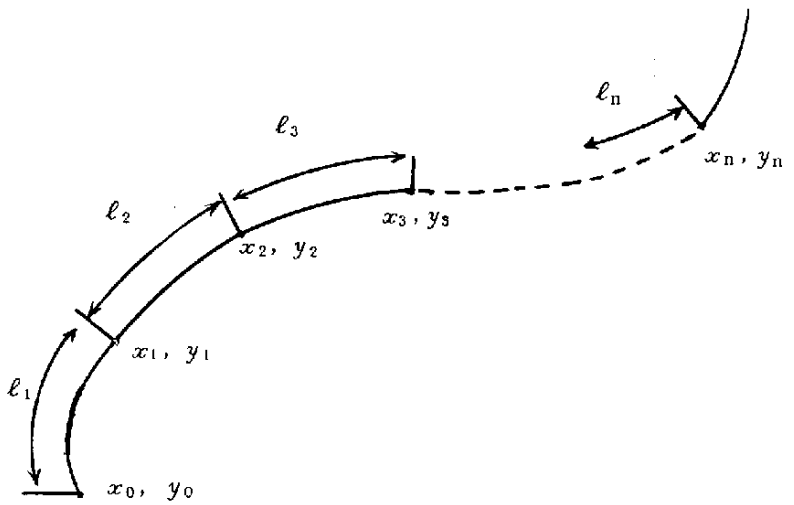
すなわち，

$$x = \frac{1}{6} a_x \ell^3 + \frac{1}{2} b_x \ell^2 + C_x \ell \qquad ①$$

$$y = \frac{1}{6} a_y \ell^3 + \frac{1}{2} b_y \ell^2 + C_y \ell \qquad ②$$

①，②式を満たす，a_x，b_x，C_x，a_y，b_y，C_y，ℓを定める。

ＧＳＰは，２点毎に，これらの係数と，曲線の長さ，点座標を，ＧＣＨ側に転送する。（図3.3.21参照）



9*	ax_1	bx_1	Cx_1	ay_1	by_1	Cy_1	x_1	y_1	ℓ_1
9	ax_2	bx_2	Cx_2	ay_2	by_2	Cy_2	x_2	y_2	ℓ_2
9	ax_3	bx_3	Cx_3	ay_3	by_3	Cy_3	x_3	y_3	ℓ_3
9	ax_n	bx_n	Cx_n	ay_n	by_n	Cy_n	x_n	y_n	ℓ_n

＊9は，フィッティング・カードを示す標識

図3.3.21　DDAによるカーブ・フィッティング
── GSPによって作られる係数群

この情報を受け取ったＧＣＨは，図 3.3.2 2のマッピングに従った関数発生回路をプログラムに内蔵しており，それぞれ，に初期値

$f'(x) = Cx_i$　　$\because$ $f'(x) = 2a_x\ell^2 + b_x\ell + C_x$

$f''(x) = bx_i$　　で，$\ell = 0$ の時 $f'(x) = C_x$

$f'''(x) = ax_i$　　以下同様に

$g'(y) = Cy_i$　　$f''(x) = a_x\ell + b_x = b_x$

$g''(y) = by_i$　　$f'''(x) = a_x$

$g'''(y) = ay_i$

――を代入し，ℓ_i を満足するまで，$d\ell$ の入力を繰り返し，独立に dx，dy を取り出す。

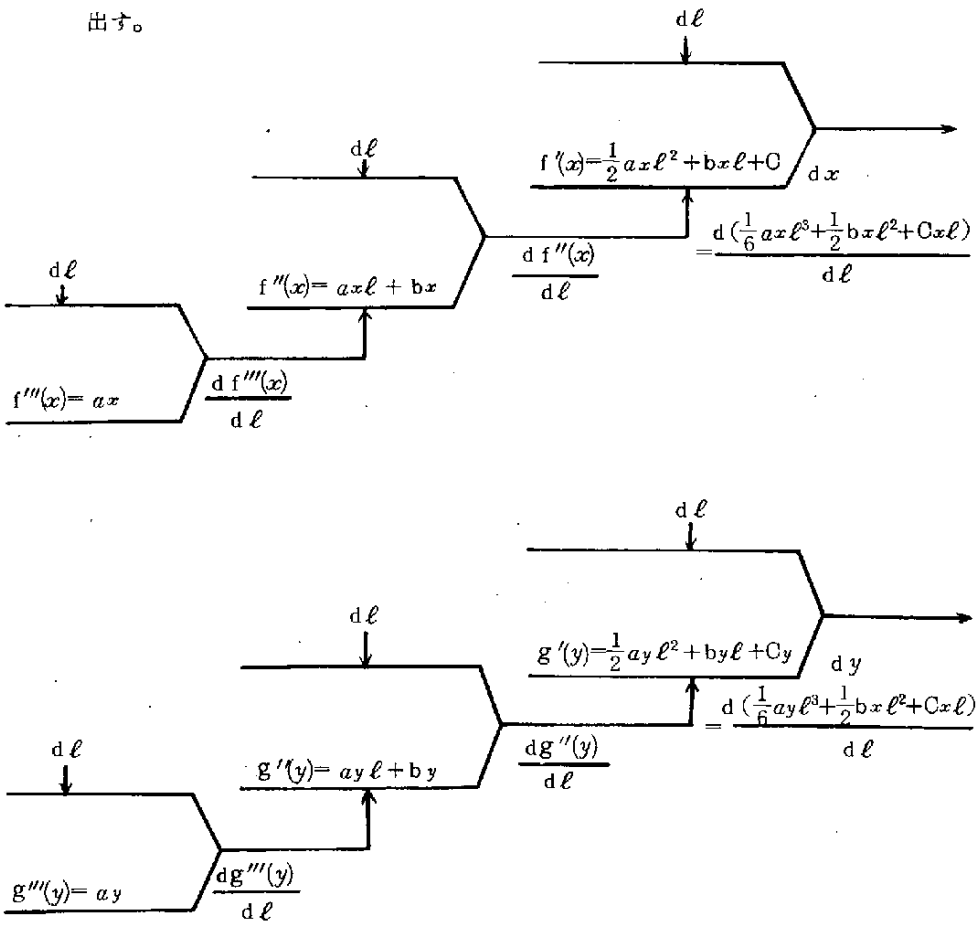


図 3.3.2 2　3次関数曲線のためのＤＤＡマッピング

dx，及び dy の一回分の出力は，スケーリングによって，CRT（１０２４＊１０２４mesh）の¼mesh 分に量子化して置き，実際のコマンド出力は，これに丸めをかけて行なう。

o　カーブ・フィッティングと，セグメント・レングス・コントロール

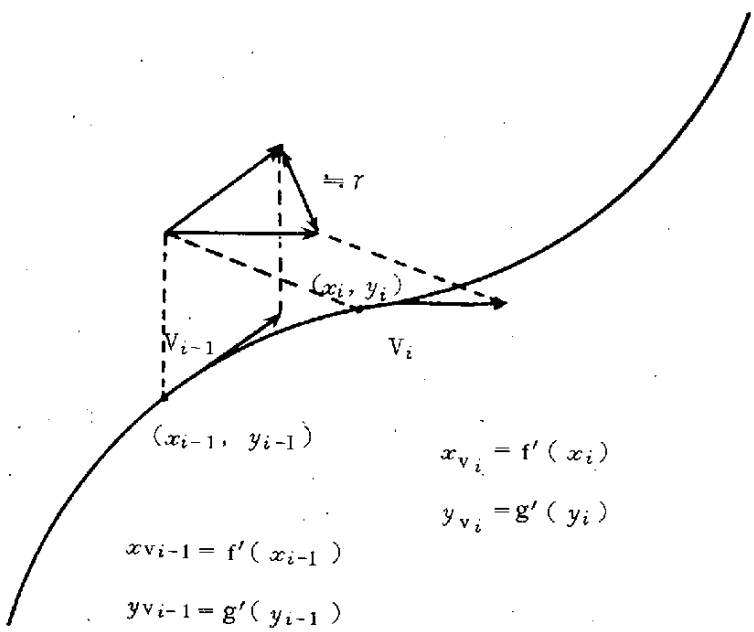
GCHは，上述の方法で取り出した dx，dy を累積し，x，y の短かいベクトル・コマンドに分けて曲線を表示するが，その際，一つ一つのベクトルの長さは，曲線の曲率によって長さを可変にした方が望ましい。その理由は２つあり，

① 全体のコマンド数を減らして記憶スペースの利用効率をあげる。

② 曲率の大きな所は短かいベクトル，小さな所は長いベクトルにして，曲線のなめらかさを生かす。

この目的を果たすため，GCHは，ベクトル・コマンドの出力を，曲線の曲率によってコントロールしている。曲率 r は，次式によって近似する。

$$r \fallingdotseq \sqrt{\left\{ f'(x_i) - f'(x_{i-1}) \right\}^2 + \left\{ g'(y_i) - g'(y_{i-1}) \right\}^2}$$



―つまり，各点の接線ベクトルの開きの角度に応じて，たゞちに短かいベクトルとして出力するか，さらにｄℓを加えて長いベクトルになるまで，ベクトル出力を遅らせるかを選択する。（図3.3.2 3参照）

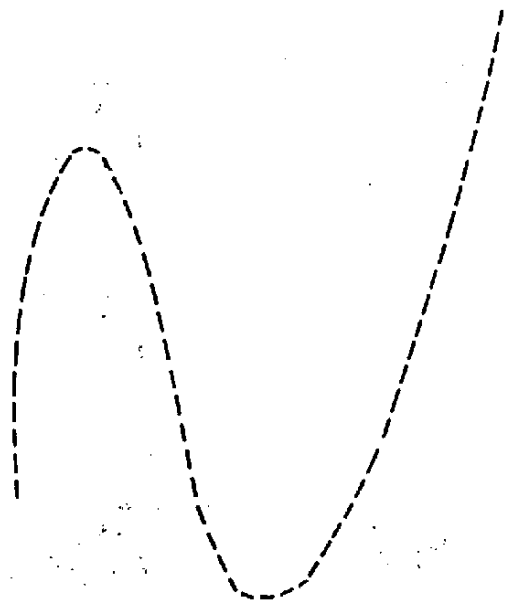


図3.3.2 3　曲率に応じたセグメント長の選択

● ＸＹプロッタ・コピー

ディスプレイの画面に表示した図形は，新しい画面を創成すれば当然消えてしまう。ユーザーが表示した図形のハード・コピーをとりたい場合は，このＸＹプロッタ・コピー・ルーチンが用いられる。

このルーチンはコマンド・コンパイラとオーバーレイして動作し，起動はオペレータがファンクション・キー・ＮＯ.6を押すことにより行なわれ，コピーの途中に図形コマンドに変更を与えるようなＤＤＤがＨ－８４００側から送られてきた場合は，コピーを中止する。

しかし，図形情報に関係のないメッセージ交換のような場合は，そのオペレーションと並行してコピーが行われる。

● メッセージ・インジケータとシステム・オペレーション・メッセージの表示

H－８８１１ディスプレイ装置には，画面の前にキーボード・タイプライタが備えられていないので，キャラクタ・ストリングの入力には次のようなメッセージ・インジケータを表示しておき，キャラクタのピッキングによりキャラクタ・ストリングをつくり出し，ＳＥＮＤというライト・ボタンのピックによりそのメッセージをホスト・コンピュータに送出するという方法をとっている。

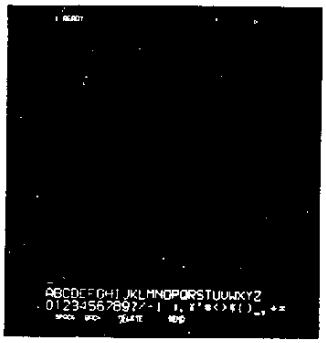


図 3.3.24　メッセージ・インジケータ

そのメッセージ・インジケータの機能とオペレーションについては５章のシステム・オペレーションで詳説している。

3.4 メッセージ転送のコントロール

1. 〔プロセッサー間のメッセージ転送〕

ＣＧＯＳの下では，ホスト・コンピュータ（Ｈ－８４００）とサテライト・コンピュータ（Ｈ－８８１１）との間に，必要に応じてメッセージの転送が行なわれる。両プロセッサー間のメッセージ・データ転送は，コントロール・プログラム（エグゼクティブ）のマクロ・サポートの下に，Ｈ－８８１４ＤＸＣ（データ交換器）を介して行なわれるが，このためには，両プロセッサーに各々ＤＸＣに対する入出力アクセスを行なうプログラムが，たゞ一つだけ必要とされる。ＣＧＯＳではＨ－８４００側ではＧＪＭＯＮ，Ｈ－８８１１側ではＧＣＨがこれに当たる。

ＤＸＣ経由のデータ転送は，一方のプロセッサー内のプログラムが出力マクロ命令（Ｗｒｉｔｅ　ＤＸＣ）を発すると，ＤＸＣ内のフリップ・フロップにデータの転送方向がセットされ，他方のプロセッサーに外部信号割込みがかけられる。割込みをかけられたプロセッサーのコントロール・プログラム（エグゼクティブ）は，前もってＤＸＣの割込み処理を登録したプログラムに割込みをかけて，その割込み処理ルーチンに制御を渡す。このルーチンでＤＸＣに対する入力マクロ命令（Ｒｅａｄ　ＤＸＣ）が出されると，実際のデータ転送が行なわれる。 ― つまり，Ｈ－８４００プロセッサーとＨ－８８１１プロセッサー間のメッセージ・データ転送は，ＧＪＭＯＮとＧＣＨの間での，相互割込み転送方式で行なわれる。（図3.4.1参照）

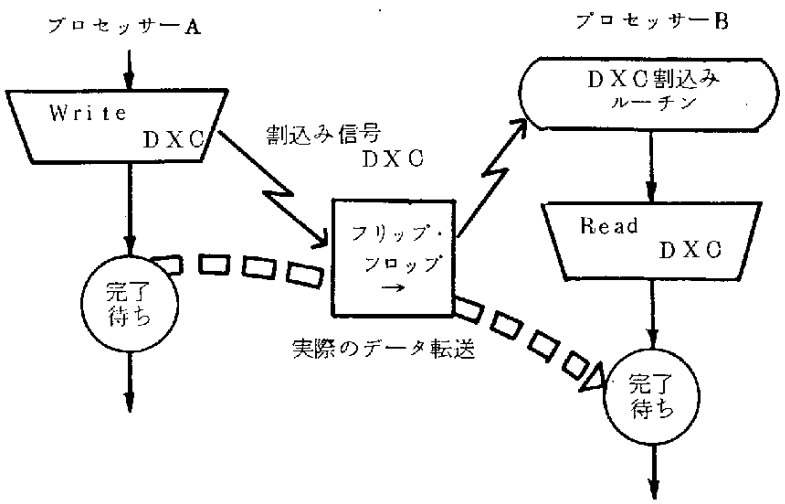


図3.4.1　ＤＸＣ経由の，プロセッサーＡから，プロセッサーＢへのデータ転送

こゝで一つやっかいな問題が生じる。両プロセッサー内のプログラムは，非同期並行に進行するため（ＤＸＣ経由の割込み以外に，この同期をとる手段はない），両方でほゞ同時に相手側に対する出力要求を出す場合もある。両方からの出力要求が衝突した時は，誤応答の完了情報がプログラムに返されるので，プログラムはこれを見て，どちらかゞ自分の側の出力を一たん見合わせ，先づ相手側の要求を満たした後に，自分の要求を果たすことが必要である。他方の側のプログラムは，再度ＤＸＣへの出力要求を発する。この際，できれば，相方の転送メッセージの内容によって，ＧＪＭＯＮとＧＣＨのどちらの要求を優先させるかを決定したいところだが，相手のデータの意味は，読んでみなければわからない。従って，どちらかのプログラムの要求に，固定的な優先権を与えることになる。

どちらの要求を優先させるかを選ぶとすれば，システムに対する最終的な制御権をグラフィック・ＣＲＴのオペレータに持たせるために，オペレータとの直接のインターフェイスをつかさどるＧＣＨを取るべきである。　逆にＧＪＭＯＮに優先権を与えるとすると，ＧＪＭＯＮの監視下で進行ずるグラフィック・アプリケーション・プログラムにバッグがあり，プログラムがグラフィック・ＣＲＴへの出力をひっきりなしにいつまでも続けた時，ＣＧＯＳはこのプログラムの"おしゃべり"のため身動きならなくなってしまうであろう。

ＧＪＭＯＮ，ＧＣＨ両プログラムのＤＸＣ出力ルーチンは，図 3.4.2 のようになる。

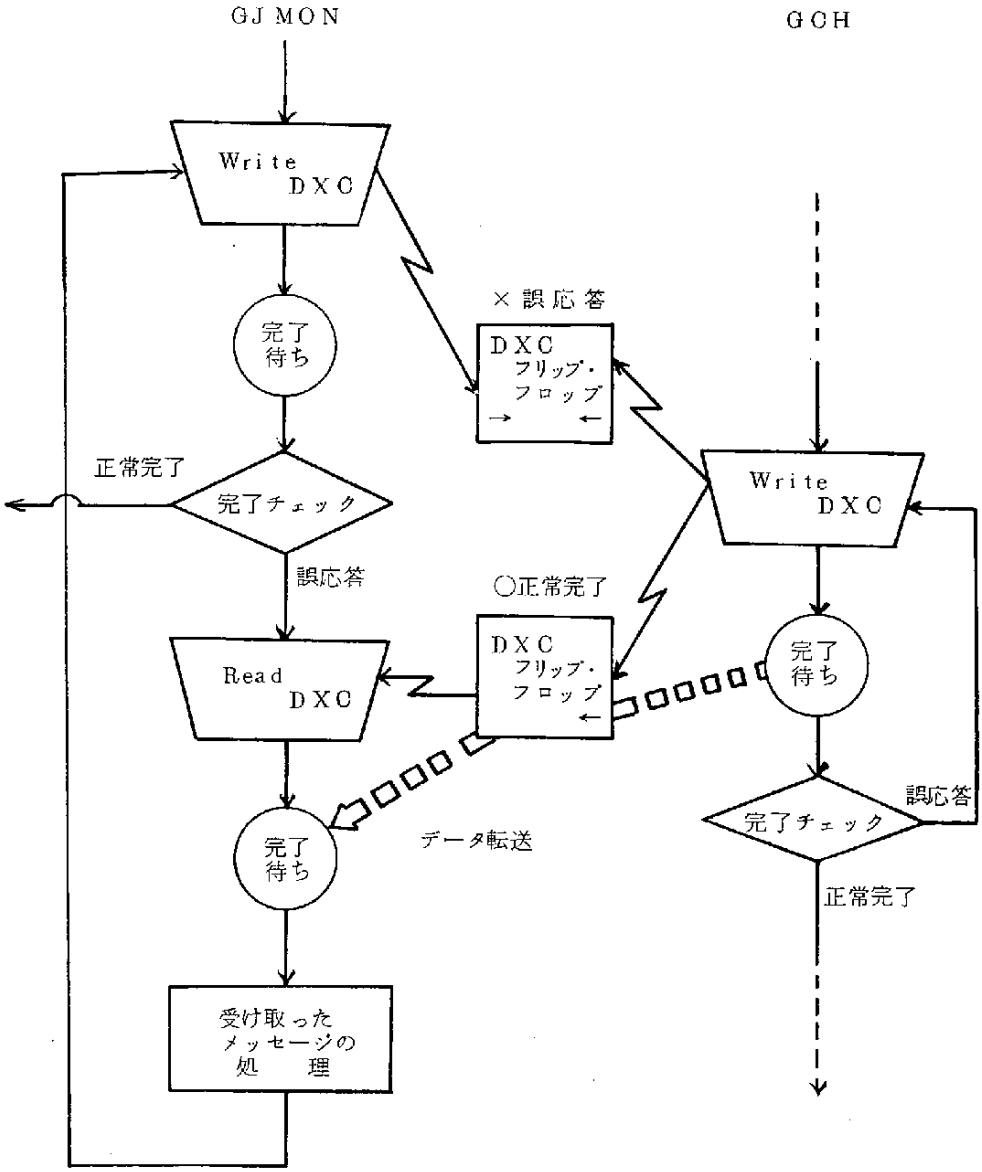
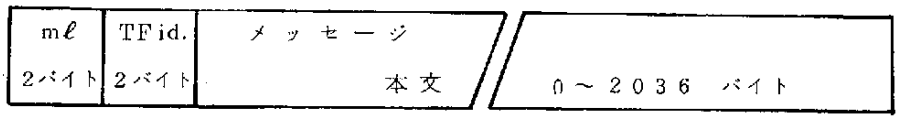


図 3.4.2　DXC経由のGJMON，GCHメッセージ・データ転送フロー

(1) プロセッサー間転送メッセージ・フォーマット

GJMON, GCH間をDXC経由で転送されるメッセージ・データは, 図3.4.3の形式に従う。



mℓ : message length

4～2040(バイナリイ)で, メッセージ全体の長さを表わす。

TFid : Text Form Identifier

転送されるメッセージの種類を示すコードで, GJMON→GCHの転送時と, GCH→GJMON方向の転送時では, 意味が異なる。

GJMON → GCH方向のTFid

C▼¥X▼	:	システム・メッセージ
X▼0000▼	:	ディスティレイティッド・ディスプレイ・データ (DDD)
X▼0100▼	:	ダイレクト・ディスプレイ・オーダー
X▼0200▼	:	ビュアー・オン指令
X▼0300▼	:	ビュアー・オフ指令
X▼0400▼	:	リード・データ指令
X▼0500▼	:	シザリング指令
other	:	プログラムからのキャラクター・メッセージ

GCH → GJMON方向のTFid.

X▼0000▼	:	ファンクション・キー割込み情報
X▼0100▼	:	ライトペン・ピッキング情報
X▼0200▼	:	トラッキング・クロス 〃
X▼0300▼	:	ダイアル 〃
X▼0400▼	:	ムーピング・エンド 〃
X▼0500▼	:	キャラクタ・メッセージ 〃

図3.4.3 GJMON－GCH間のメッセージ・フォーマット

(2) プログラム間のメッセージ転送

前項に述べたように，H－８４００，H－８８１１両プロセッサー間のメッセージ転送は，ＧＪＭＯＮ，ＧＣＨ間でＤＸＣを介した割込みデータ転送で実現されるが，H－８４００プロセッサー内で，マルチ・プログラミングで進行するグラフィック・アプリケーション・プログラムから，グラフィック・ＣＲＴに送られるべき図形データ，及びグラフィック・ＣＲＴに向ったオペレータからグラフィック・アプリケーション・プログラムに対して出された指令や情報も，全て，ＧＣＨ ― ＧＪＭＯＮ間のデータ転送の下に集中管理される。

このことは，H－８４００プロセッサー内では，ＧＪＭＯＮと，グラフィック・アプリケーション・プログラムとの間に，プログラム間の相互データ転送が必要であることを意味する。

H－８４００のコントロール・プログラムは，『プログラム間の通信連絡マクロ・サポート（Inter Program Communication Macro）』を持っている。"インタラクティブ・モード"が重要な課題であるグラフィック・システムにとっては都合の良いマクロで，ＣＧＯＳでは，ＧＪＭＯＮとグラフィック・アプリケーション・プログラム間のデータ転送に，これを採用している。

ＩＰＣマクロによるデータ転送は，ＤＸＣ経由のプロセッサー間データ転送の場合と一見良く似ている。H－８４００内でマルチ・プログラミング・モードで進行している２つのプログラムの一方が，相手プログラムの名称を指定してＰＣ Write 命令を発すると，相手プログラムは割込まれて，あらかじめ登録してある自分のＰＣ割込みルーチンに制御を渡される。割込み処理ルーチンの中でＰＣ Read 命令を発すると，実際のデータ転送が行なわれる。これを図3.4.4に示す。

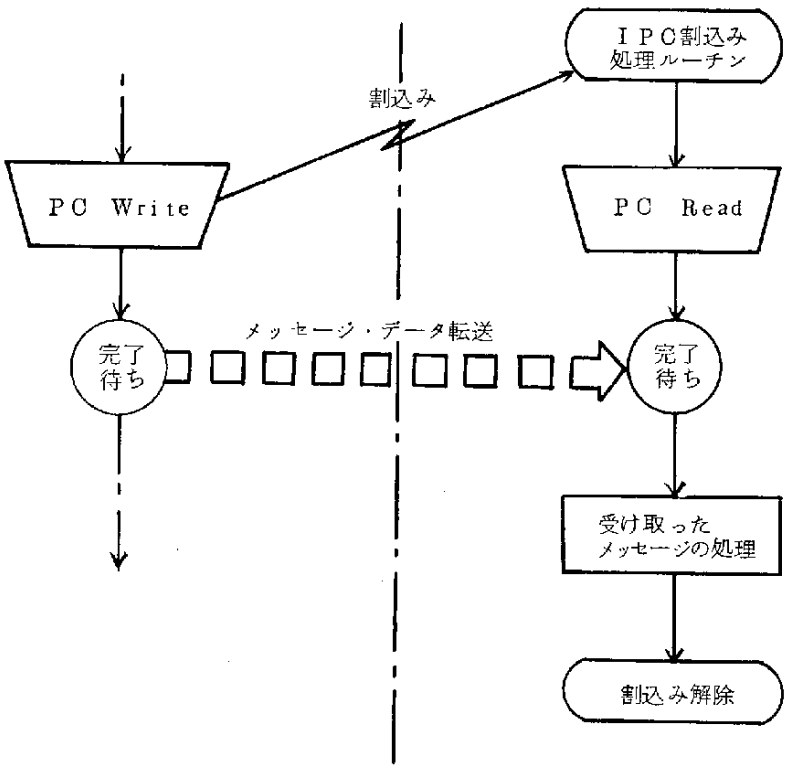


図3.4.4　H－８４００内のプログラム間メッセージ・データ転送
ＧＪＭＯＮ → グラフィック・アプリケーション・プログラム
（逆も同様）

3.5 グラフィック・オペレーション・コマンド

ＣＧＯＳでは，グラフィック・ＣＲＴからのオペレーションを容易にするべく，いくつかのコマンドを用意している。

コマンドには，

① ロギング・デバイスの変更
（ alter loging device ）

② プログラムの進行状態の問い合わせ

（ask program's current status）

③ グラフィック・プログラムの処理中断と，ＣＲＴ入出力の解放

（quit to graphic program）

④ グラフィック・プログラムの進行開始（または再開）とＣＲＴ入出力の許可

（start graphic program）

3.5.1 alterコマンド

alterは，Ｈ－８８１３グラフィック・ＣＲＴの上部に表示されるシステム・レスポンス・メッセージを，Ｈ－８８１１システムに付属した問い合わせタイプライタに切換えて出力させる指令である。

一度出力を切換えた後に再度alterコマンドを出すと，システム・レスポンス・メッセージは再びグラフィック・ＣＲＴに表示されるようになる。

3.5.2 askコマンド

askコマンドは，Ｈ－８４００内にＧＪＭＯＮの監視下で進行しているプログラムの一つについて，現状を問い合わせるコマンドである。

askコマンドを受けたGJMONは，プログラム制御表（表1.3参照）から，指定されたプログラムについての情報を編集して，グラフィック・ＣＲＴに表示するべく，そのデータをＤＸＣ経由でＧＣＨに送り出す。

3.5.3 Quitコマンド

Quitコマンドは，現在進行中のグラフィック・プログラムに関して，そのプログラムを一担停止させ，他のグラフィック・プログラムにＣＲＴ入出力の権利をあけ渡すようにする。

Quitコマンドは，場合によっては緊急性を要するコマンドであるため，他のシステム・コマンド同様のライトペンによるメッセージ入力の他に，ファンクション・キーNo.3で代行できるようになっている（第５章システム・オペレーション参照）

Quitされたグラフィック・プログラムは，後の処理再開にそなえて，グラフィック・ＣＲＴに表示中の図形情報を保存して置く必要がある。このため，Quitコマンドを受け取ったＧＪＭＯＮは，該当のプログラムに対する。Quit処理を行なった後に，図形情報の保存指令をＧＣＨに対して返す。これを"コマンド・セーブ指令"と呼び，図3.5.1に示す。

mℓ 20	¥	0	プロ id. グラ No. ム	△	プログラム名	△	QUITED

図3.5.1　コマンド・セーブ指令

コマンド・セーブ指令を受け取ると，GCHは，あらかじめディスク上に確保したセーブ領域*の，該当のプログラムid. No.に割り当てられたスペース上に，そのプログラムの全図形情報（コマンド，各種テーブル，スウィッチ類を含む）を書き出す。

GJMON側のQuit処理は次の手順をとる。まずプログラム制御表から，現在進行中のグラフィック・プログラム（PS3＝▼G▼）を見つけ，（なければエラー・メッセージをGCHに送る）そのPS2に▼Q▼をセットする。以後，このプログラムがGJMONに対してPC Write割込みをかけて来ても，GJMONは，このプログラムからのメッセージを受け取らず（PC readを出さない）に抑制しておく。グラフィック・プログラム側は，自分がGJMONに対して発したPC Writeが抑制された場合には，自らを割込み待ちのアイドル状態にしておき，後にGJMONからのStart指令を受け取った時に再度PC Writeを発するようにする（4.1インタラクティブ・プログラミング・サポート参照）。

3.5.4　Startコマンド

Start コマンドは，H－8400にGJMONの監視下でロードされたグラフィック・プログラムに，グラフィック・CRTを介した入出力の権限を与え，プログラムの処理開始を許す，もしくは，かって，Quitコマンドで処理を中断していたグラフィック・プログラムの処理再開を命ずるコマンドである。

GCHからStantコマンドを受け取るとGJMONは，指定されたプログラムが現在Quit状態，もしくは，ロードされたまゝの状態であることを，プログラム制御表上で確認する（PS2が▼Q▼もしくは▼I▼）。指定に誤まりがある時はエラー・メッセージをGCHに返す。

次に，現在処理進行中のグラフィック・プログラムが他にあれば（PS2が▼R▼か

＊　CGOSでは，同時並行処理可能なグラフィック・プログラムは5つまで（プログラムid. No.：1～5）なので，セーブ領域は5つのセッションに分ける。

つ，ＰＳ３が▼Ｇ▼），これに対するQuit処理をほどこす。これは，ＣＲＴを同時に複数のグラフィック・プログラムに使用させるのを防ぐためである。

次にStartコマンドを受けたグラフィック・プログラムが，プログラム・ロード後，最初の処理開始であるか否かをチェックする。（ＰＳ２が▼Ｉ▼の時は最初）もし，最初の処理開始でない場合は，グラフィック・ＣＲＴの表示図形を，このプログラムがQuitされた時の状態に復帰する必要があるので，ＧＪＭＯＮはＧＣＨに対して，"コマンド・リロード指令"を発する。コマンド・リロード指令の形式を図 3.5.2 に示す。

mℓ ２０	¥	1	プロid. グラNo. ム	△	プログラム名	△	ＲＥＳＴＡＲＴ

図 3.5.2. コマンド・リロード指令

コマンド・リロード指令をＧＪＭＯＮより受け取ったＧＣＨは，前にコマンド・セーブ指令に従って，ディスク上にセーブした，このプログラムの図形情報を読み出して，ＣＲＴ画面をQuit時の状態に復元する。

一方，ＧＪＭＯＮは，Start コマンドに指定されたグラフィック・プログラムのＰＳ２を▼Ｒ▼にセットして，このプログラムに対し，ＰＣ Writeマクロによって Start信号を送る。ＧＪＭＯＮからのStart信号を受け取ったグラィック・プログラムは，元通り処理を再開する。

第4章　グラフィック
プログラミング・システム

ＣＧＯＳの下では，Ｈ－８８１３グラフィック・ＣＲＴを介した入出力を行なうプログラムを"グラフィック・プログラム"と呼ぶが，ＣＧＯＳは，ユーザーのプログラムにＣＲＴ入出力を容易に行なわしめるべく，『グラフィック・プログラミング・システム』を提供する。

グラフィック・プログラミング・システムを便宜的に区分すると，

① インタラクティブ・プログラミング・サポート

② グラフィック・プログラミング・サポート

③ アソシェーティブ・データ・マネージメント

④ グラフィック・ランゲージ・ファシリティ

の４項からなる。

このうち③，④は来年度以降に開発が予定されているので，ここでは①，②につき記述する。

4.1 インタラクティブ・プログラミング・サポート

"インタラクティブ"という言葉は，しばしばグラフィック・プログラムの形容詞として用いられている。グラフィック・ＣＲＴへの入出力を利用することが，機械による問題解決のターン・アラウンド・タイムを極小にし，瞬時内に人間の理解しやすい図形的情報として回答を表示し，人間の即断による試行錯誤のプロセスを暗示的に含んでいるためであろう。グラフィック・システムでは，プログラムの"インタラクティブ・モード"の実現に，充分な努力を払う必要がある。

ＣＧＯＳは，ユーザーのプログラムに，グラフィック・ＣＲＴを介したインタラクションを容易にする機能を持ったサブルーチン・パッケージを提供する。これをGraphic Interaction Analysis Subroutine Package（ＧＩＡＳＰ）と呼ぶ。

4.4.1 ＧＩＡＳＰの概要

ＧＩＡＳＰは，ユーザーの作成したグラフィック・プログラム（COBOL, FORTRAN, ＡＳＳＥＭＢＬＥＲ，またはグラフィック・ランゲッジでプログラミング可能）と，ＧＪＭＯＮとのインターフェイスを受け持つ。このサブルーチン・パッケージは，リンケ

ージ・エディターによって，ユーザー・プログラムにバインドされる。

ＧＩＡＳＰは，次の機能を持つ。

① ユーザー・プログラムと，ＧＪＭＯＮとの間のデータ転送

② ユーザー・プログラムの割込み処理登録と変更

③ ＧＪＭＯＮから受け取ったデータの解析と，数値データの変換，割込み処理の制御

④ ＧＪＭＯＮからのＱｕｉｔ，Ｓｔａｒｔ指令に基づく処理

4.1.2 ＧＩＡＳＰに含まれる各種のモジュール

ＧＩＡＳＰには，次の１１個のモジュールがある。

ＧＲＩＮＩＴ	Graphic Program Initiate
ＧＲＡＴＭＫ	Graphic Attention Mask
ＧＲＡＴＦＲ	Graphic Attention Free
ＧＲＡＴＣＶ	Graphic Attention Table Convert
ＧＲＡＴＷＴ	Graphic Attention Wait
ＧＲＡＴＥＸ	Exit From Graphic Attention
ＧＲＣＨＤＰ	Display Character mode Data
ＧＲＤＰＬＹ	Display Graphical Data
ＧＥＴＣＨＲ	Get Character Data From CRT
ＧＥＴＮＵＭ	Get Numeric Data From CRT

以上は，ユーザー・プログラムから呼ばれるサブルーチン・モジュールで，この他に，ＩＰＣ割込み発生時に，その処理を行なう。

ＧＲＩＮＴ Graphic Iinterruption Handlerのルーチンがある。

これらのモジュールのそれぞれの機能を以下に説明する。

(1) ＧＲＩＮＩＴ（n, Code$_1$, ad_{11}, ad_{12}〔------, Code$_n$, ad_{n1}, ad_{n2}〕）

グラフィック・プログラムの初期設定を行ない，割込み処理制御表（図4.1）を作る。

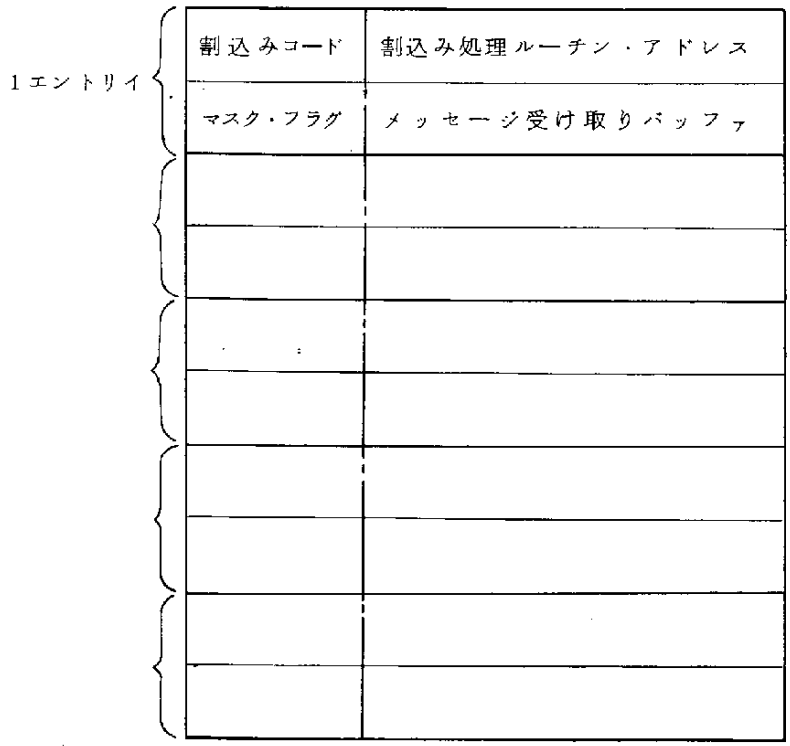


図4.1　ＧＩＡＳＰ割込み処理制御表

n　：　登録する割込みの種類

$code_i$　：　割込みメッセージ・コード

0　ファンクション・キイ割込み（No. 11～31）

1　ライトペン・ピッキング割込み

2　ダイアル割込み

3　トラッキング・マーク割込み

4　ムービング終了割込み

5　キャラクター・メッセージ割込み

▼¥▼ 非同期制御変更割込み

▼¥▼ メッセージの割込みに対しては，指定のルーチンへ，非割込みモードで制御を渡す。従って，元の状態への復帰はできない。

▼*▼　▼¥▼ 以外の，登録されていない全ての割込みを受けつける。

① 一 般 形 式

メッセージ レングス	割込みコード	メ ッ セ ー ジ 本 文

2byte　1byte

② ファンクション・キイ割込みメッセージ（FK No. 11～31）

12	X'00'	X'00'	ファンクション・キイ・ナンバー（11～31）	オーバレイ コード（0～7）	キイ・ランプ 1	キイ・ランプ 2	

1　1　2 byte　2 byte　2 byte　2 byte

③ ライト・ペン割込みメッセージ

20	X'01'	X'00'	エンティティ id.	オプション・ポインター（フレーム id.）	フラグ ナンバー（3～170）	XPR	YPR	DXR	DYR

1　1　4 byte　2 byte　2 byte　2 byte　2 byte　2 byte　2 byte

③ ダイアル割込み情報

8	X'02'	X'00'	ダイアルNo. (0=No1. 1=No2.)	ダイアル・コード	

④ トラッキング・クロス割込み情報

10	X'03'	X'00'	ライト・ペン トップ・ スウィッチ 0＝off 1＝on	XPR	YPR	

⑤ ムービング終了割込み情報

12	X'04'	X'00'	エンティティ id.	オプション・ ポインター (フレーム id.)	完了コード 0＝正常終了 1＝エッジ	

⑤ キャラクタ・メッセージ割込み

メッセージ・ レングス	X'05'	X'00'	キャラクタ・メッセージ

図 4.2　割込みメッセージ・フォーマット

(2) GRATMK (code$_1$〔, code$_2$, -------〔, code n 〕〕)

code$_n$ 指定した種類の割込みを一時的に抑制する。

(3) GRATFR (code$_1$ 〔, code$_2$, --------〔, code n 〕〕)

GRATMKの対で，禁止されていた割込みを解除する。

(4) GRATCV (i, code$_1$, ad_{11}, ad_{12} 〔, -------, code n, ad_{n1}, ad_{n2} 〕)

指定された種類の割込み処理を変更，もしくは追加する。

(5) GRATWT

プログラムを割込み待ち状態にして，停止させる。この状態から何らかの割込みが発生し，その割込み処理ルーチンが，次のGRATEXを呼んで割込みを解除すると，プログラムは，この命令の次に進む。

(6) GRATEX

割込み状態を解除して，元の処理へ復帰する。

このルーチンは割込み処理ルーチン内のみで使用可

(7) GRCHDP (〔n, 〕ad)

CRTに文字列を表示する。

n ： キャラクタ数

ad ： キャラクタ・ストリング

＊ adのみ指定の場合は，メッセージの先頭に，キャラクタ数n＋2を内容とする2バイト分を設けておかねばならない。

(8) GRDPLY (ad)

DDD (Distillated Display Data)の形式になったデータを，CRT側へ転送，表示する。

ad ： DDDアドレス

(9) GETCHR (ad, X, Y〔 j 〕)

キャラクタ・ストリングをCRTより読み込む。プログラムは，オペレータからキャラクタ・ストリングを送ってくるまで停止して待つ。

ad ： キャラクタ・ストリングの入るバッファー (20ワード配列)

X ： CRT上のX座標 (0～1000)

Y ： CRT上のY座標 (0～1000)

j ： キャラクタの大きさ

0 ―小文字， 1 ―大文字

⑽ GETNUM(i , ad, X, Y〔,j〕)

数値データをCRTより読み込み，内部形式に変換する。プログラムは，CRTオペレータが数値データを入力するまで停止する。

i ： 数値のタイプ

0 ― 整数型で入力型式は，

〔±〕$n\,n\,n$ - - - - - - - n

1 ― 実数型で入力型式は，

〔±〕〔$n\,n$ - - - - - - n〕・〔$m\,m$ - - - - - - m〕

又は，

〔±〕〔$n\,n\,n$〕・$m\,m\,m$ - - - - - mE〔±〕$\ell\,\ell$

他のアーギュメントはGETCHRと同形式

4.1.3 GIASPの割込み制御とデータ転送

GIASPは，インタラクティブ・プログラミングを実現可能にするため，GJMONとの間にIPCマクロ〔Inter Program Communication Macro:前出〕を利用した割込みデータ転送を行なっている。

すなわち，グラフィック・CRTのオペレーターが，プログラムに対して発した何らかの指令（ライトペンによるメッセージのSENDボタン・ピック，表示図形のピック，ファンクション・キイ，ダイアル，etcの操作によって行なわれる。）は，GCHからGJMONを経由し，GJMONから，ユーザー・プログラムにリンケージ・バインドされたGIASPに非同期割込みを伴って知らされる。つまり，CRTから発生したメッセージは，ユーザー・プログラムがどの様な処理中でも，待たされることなく，直ちにそのプログラムに渡される。（図4.3参照）

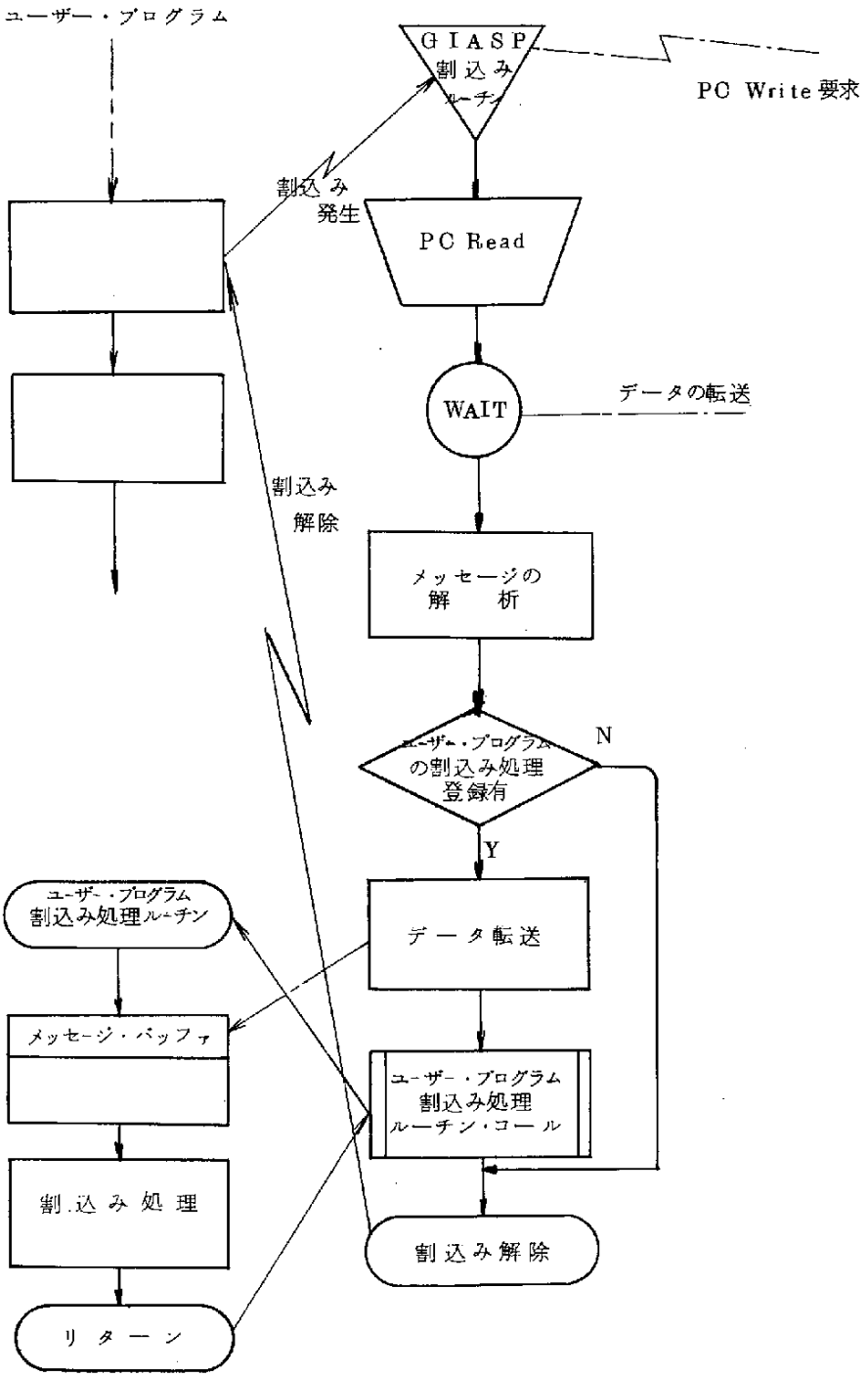


図 4.3　GIASPの割込み制御

このようにして受け取った，ＣＲＴ・オペレータからのメッセージを，ＧＩＡＳＰは，そのメッセージ・コード（メッセージの種類を表わす ― 図4.2参照）を，割込み制御表（図4.1参照）から探す。割込み制御表は，ユーザー・プログラムがあらかじめ，自分の処理する割込みの種類と，メッセージ受け取りバッファ・アドレス，割込み処理ルーチンを，ＧＩＡＳＰに登録して置く（ＧＲＩＮＩＴによる）もので，ＧＩＡＳＰは，こゝに登録されている種類のメッセージに関しては，メッセージをユーザー・プログラムのメッセージ受け取りバッファに転送し，さらに，割込み処理ルーチンを，サブルーチン・コールする。

こうして制御を渡された，ユーザー・プログラムの割込み処理ルーチンは，受け取ったメッセージの内容に従って必要な処理を行ない，元の処理（割込まれる前の）に戻りたい時は，サブルーチン・リターンを行なえばよい。割込みの解除はＧＲＡＴＥＸを呼ぶことでも代行できる。― この場合プログラムがＧＲＡＴＷＴで割込み待ちであった時は，そのWait ループを脱出する。

4.1.4　同期をとったデータ転送

ＧＩＡＳＰが扱うＣＲＴ・オペレータから，ユーザー・プログラムへのデータ転送は，全て非同期割込みを伴っているが，メッセージ送受の機能としては，この方式のみでは不便な場合も多い。その一例は，プログラムの側が意識的に，ＣＲＴ・オペレータの指示を求め，その指示に基いて，次の処理を決定して行おうとする場合である。換言すると，ＧＩＡＳＰに，"Read from ＣＲＴ"に相当するメッセージ転送の機能が望まれる。

この目的で，ＧＩＡＳＰは，3つのサブルーチン・パッケージＧＲＡＴＷＴ，ＧＥＴＣＨＲ，ＧＥＴＮＵＭを設けている。

ＧＥＴＣＨＲは，キャラクタ・メッセージ，ＧＥＴＮＵＭは数値データ，またＧＲＡＴＷＴは特に種類を問わないＣＲＴ・オペレータの指示をうながすもので，これらのサブルーチンは，ＣＲＴ・オペレータが，それぞれに対応した指示を与えるまで，プログラムをＷＡＩＴ状態にして止めておく機能を有している。

ＧＲＡＴＷＴ，ＧＥＴＣＨＲ，ＣＥＴＮＵＭ等のサブルーチンが，割込み処理内から呼ばれることもあり得る。これは，プログラミング上のミスでも生じ得るし，また，特殊なデータ送受を必要とするプログラム中でも生じ得る。

場合によっては，この関係が，リカーシブルにもなり得る。（図4.4参照）ＧＩＡＳ

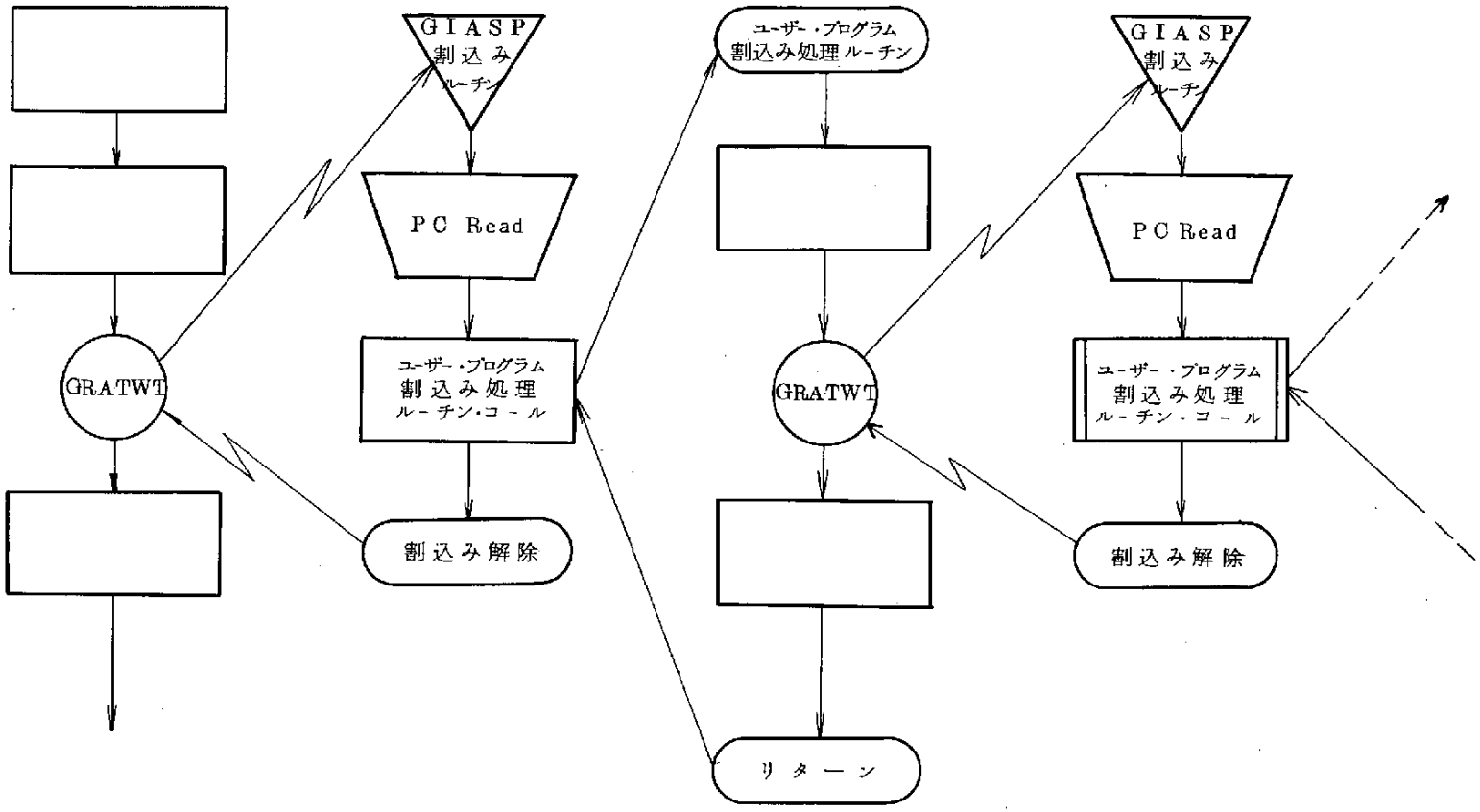
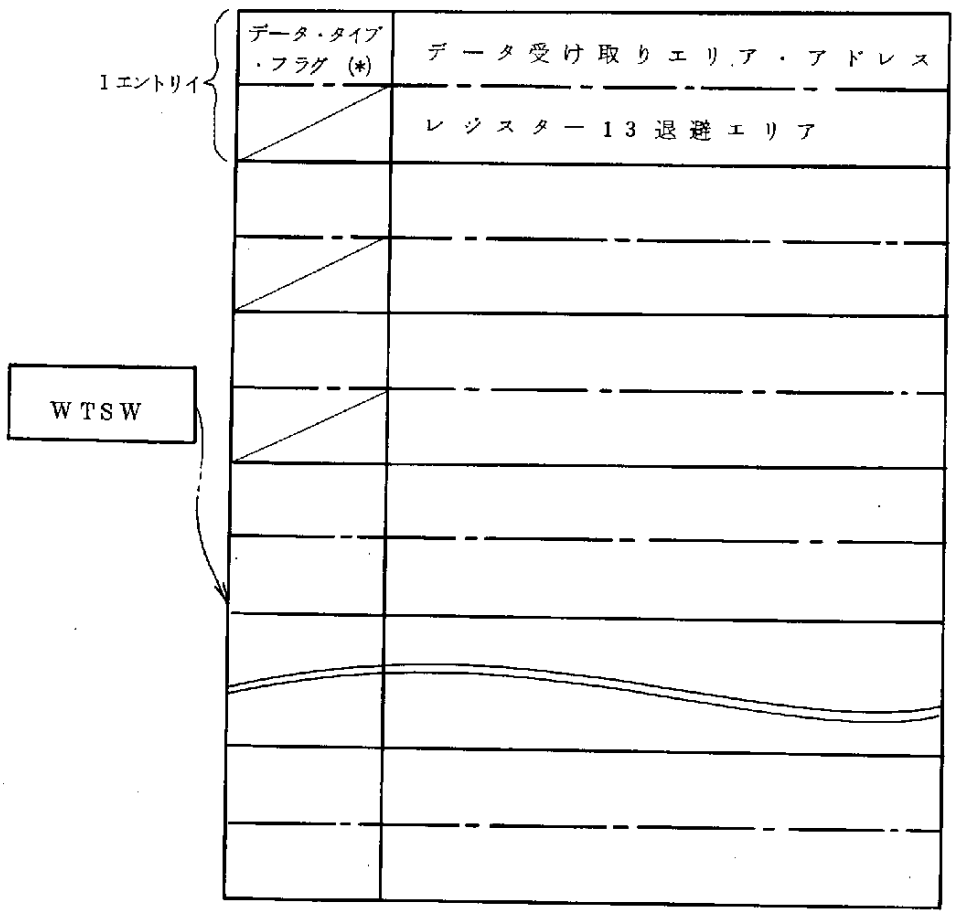


図 4.4　リカーシブル・コールを生ずるプログラム

Pでは，このような割込み待ちの状態を，８重まで許し，その制御に，RDTBL（Read Data Pending Table 表4.1参照）と，割込みの多重度を表わすのにWTSWを用いている。

表4.1　RDTBLフォーマット



＊　データ・タイプ・フラグ

0 ： キャラクター・ストリング

1 ： Iタイプ・データ

2 ： Rタイプ・データ

4.1.5　インタラクティブ・モードの問題点と，ＧＩＡＳＰ使用上の注意事項

ＣＧＯＳにおけるマン・マシン・インタラクションは，Ｈ－８８１３グラフィック・ＣＲＴと，ＣＲＴ・オペレータとの間に非同期に生ずる。かなり，きめの細かい割込み制御が行なわれないと，システムの運行は混乱に落ちいる恐れがある。

ＣＧＯＳではオペレータとシステムとのインタラクションは，ＧＣＨ，ＧＪＭＯＮ等のコントロール・プログラムが制御し，グラフィック・プログラムにとってのインタラクションは，直接は対ＧＪＭＯＮとの間に生ずる。この制御は，ＧＪＭＯＮと，グラフィック・プログラムにリンケージ・バインドされたＧＩＡＳＰとが協力して行なうが，こゝに次の問題がある。

① ＧＩＡＳＰ（すなわち，グラフィック・プログラム）側によって，ＧＪＭＯＮからのデータ転送は，全て非同期割込みを伴って発生するが，これをプログラムの進行と，どう同期を取るか。

② ユーザー・プログラム側のエラーが，ＧＪＭＯＮに波及して，ＧＪＭＯＮが運行不能になる（すなわちシステム・ダウン）事態をどう防ぐか。

完全な意味での“インタラクティブ・モード”とは，何らかの外部割込みに対して，プログラムが何かの処理遂行中でも，それを一担中断し，要求のあった処理を先に済ましてから，元の処理を再開し，その間に混乱が生じないことが保証されていることであろう。すなわち，“インタラクティブ・モード”は，いわゆる“会話形”のRead and Replyとは異っている。

“インタラクティブ・モード・プログラム”の構造を，図4.5に示す。

このようなプログラムは，非同期に外部から与えられる指令（たとえば，オペレータのinqniary）に対して，柔軟に対処できる。しかし，こうしたプログラムは特殊な構造を持っていなければならない。たとえば，図4.6のような場合も，正しく処理を行なえる構造が要求される。

図4.6では，メイン・プログラムから，あるサブルーチンが呼ばれ，このサブルーチン処理中に割込みが発生し，その割込み処理で，同じサブルーチンの機能が必要とされて，同一サブルーチンが呼ばれる場合を想定している。この場合，同じサブルーチンを通過する２つの処理の流れ ── 後のものが前のものを追い越す ── が，いずれも正しく処理されねばならない。つまり，このサブルーチンは，リエントラント構造でなければならない。

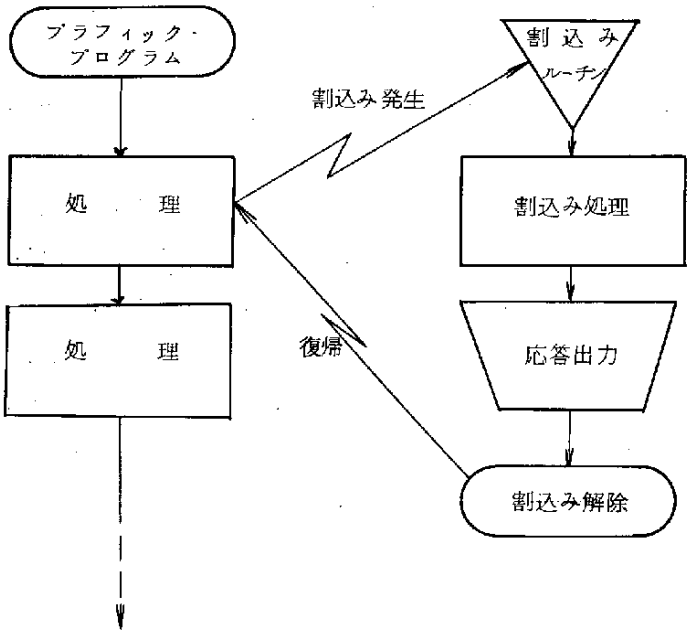


図 4.5　インタラクティブなプログラム

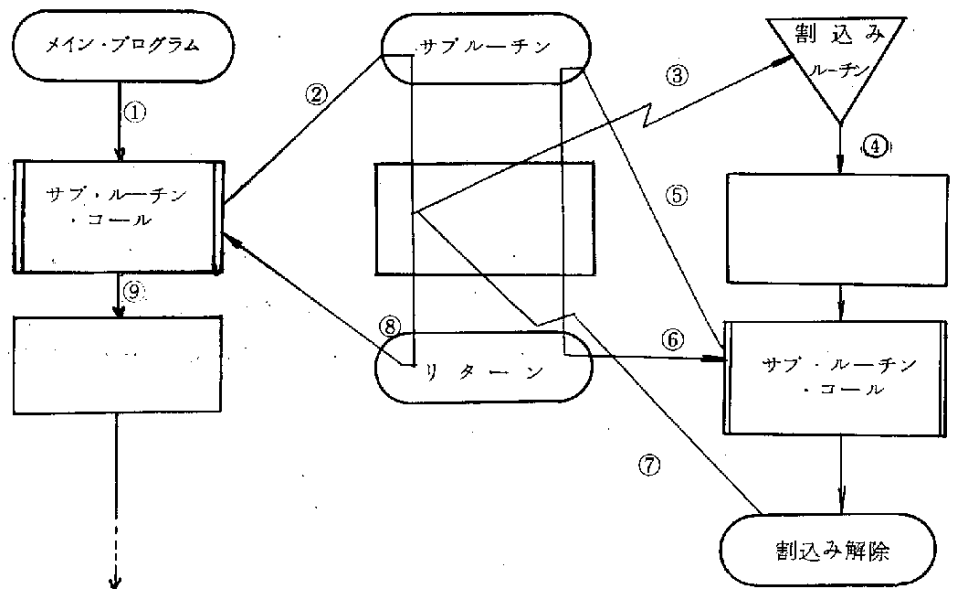


図 4.6　インタラクティブなプログラムにおける
ルーチンのリカーシブル・リエントラント・コール

しかし，既製のプログラミング言語の多くは，リエントラント構造のプログラムを編集する機能を有していない，既製の言語を用いた場合の一つの抜け道は，非割込み状態で呼ばれるサブルーチンとの共用を禁ずることであるが，ＣＧＯＳの場合は，両者ともに，ユーザーが，コンパイラー・レベルの言語で記述したプログラムが加わるので，具合が悪い。（たとえユーザーが意識的にサブルーチンの共用をさけたところで，そのプログラム言語にそなわった，ランタイム・ルーチンの共用まではさけ難い。）

従って，ＦＯＲＴＲＡＮ，ＣＯＢＯＬ等，既存のプログラミング言語での，完璧な非同期割込み処理は，あきらめねばならない。そうしたプログラムと，グラフィック・ＣＲＴ・オペレータとのインタラクションは，基本的にRead and Reply，―― つまり，自分の処理を行ない，結果を表示し，その応答を待って次の処理に進むといったやり方，――のサイクルに限定され，プログラムの内部処理，図形データ表示の為の処理は，インタラクションが禁止，もしくは待たされなければならない。

ＧＩＡＳＰでは，既存のプログラミング言語で記述されたプログラム（すなわち，ルーチンがリエントラントでないプログラム）の実行時を，割込み許可の状態 ―― enable mode ―― と，割込み禁止状態 ―― disable mode ―― の２状態に分けて制御している。

ＧＩＡＳＰのＧＲＡＴＷＴとＧＲＡＴＥＸは各々割込み制御のモードを変更するルーチンである。図 4.7 にその流れを示す。

リエントラント構造，もしくは，非割込み状態と，割込み状態で使用するルーチンが完全に別のモジュールであるようなプログラム，すなわち，完全なインタラクティブ・モードを実現できるプログラムの場合は，充分にその機能を生かせるように，ＧＩＡＳＰ自身をリエントラント構造にしておく必要がある。ＧＩＡＳＰは，最大８重まで，リエントランス可能で，これに基づき，ＧＳＰを構成する殆んどのモジュールも８重まで，リエントランス可能に作成されている。

以下にＧＩＡＳＰのブロック・チャートを示す。

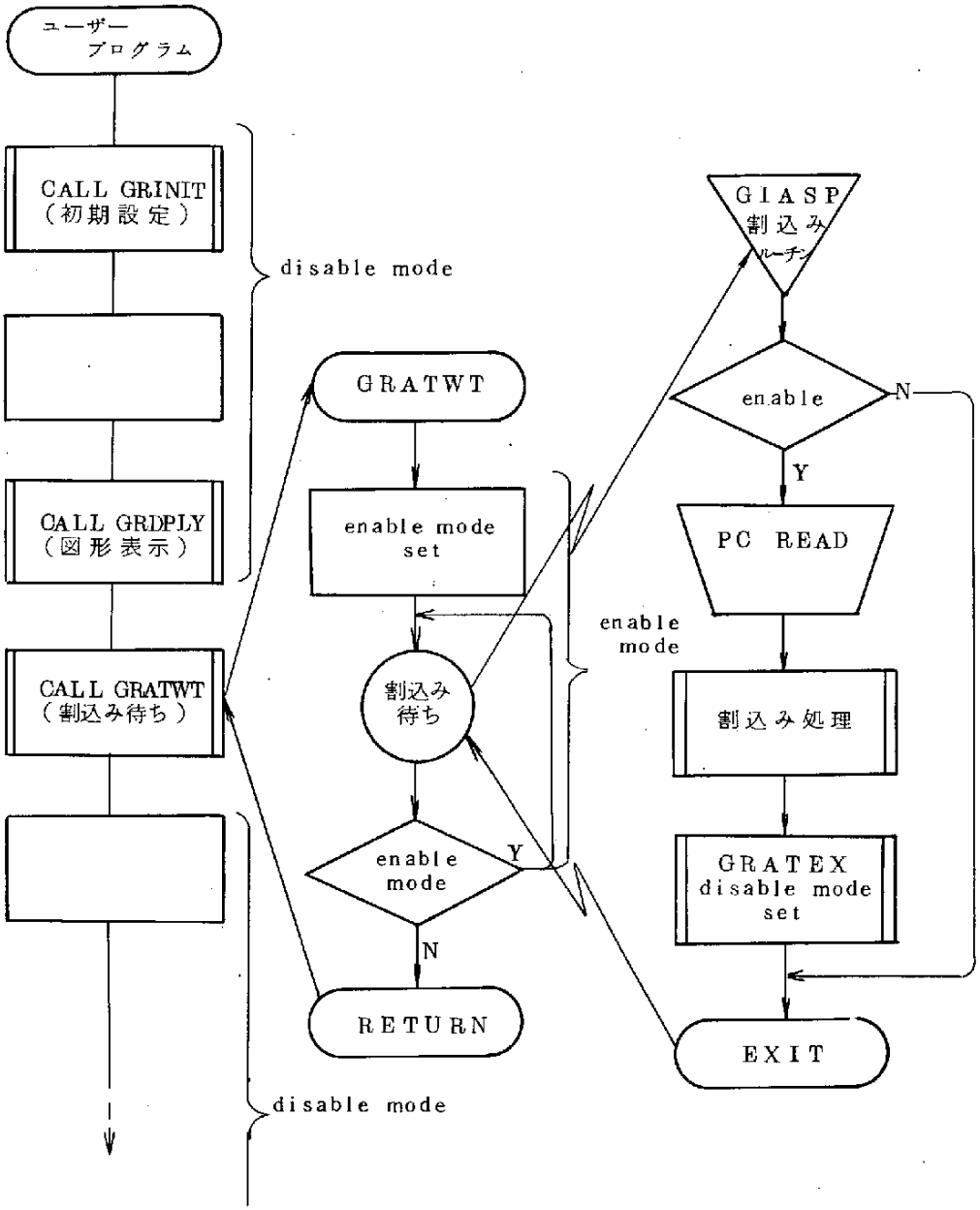
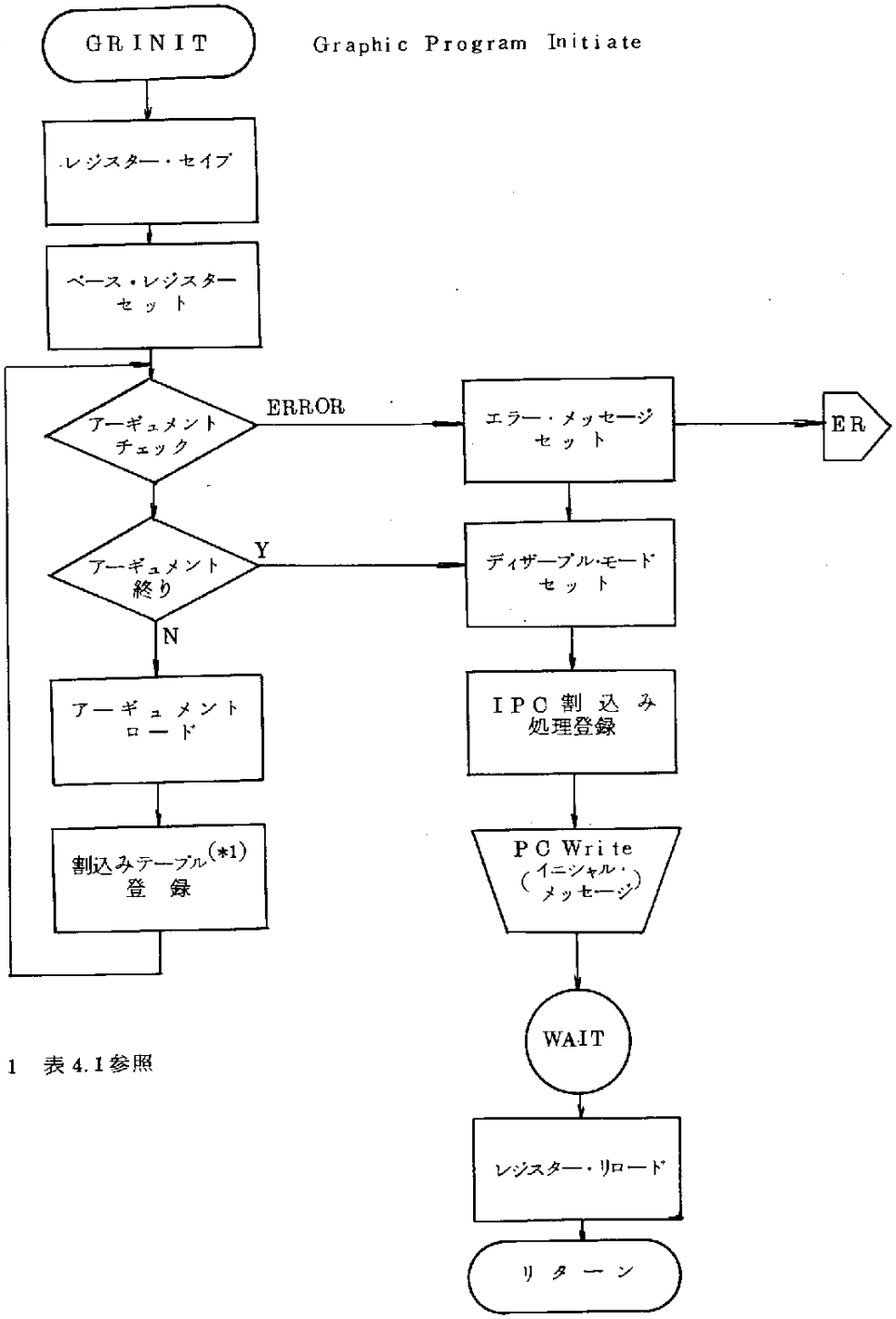
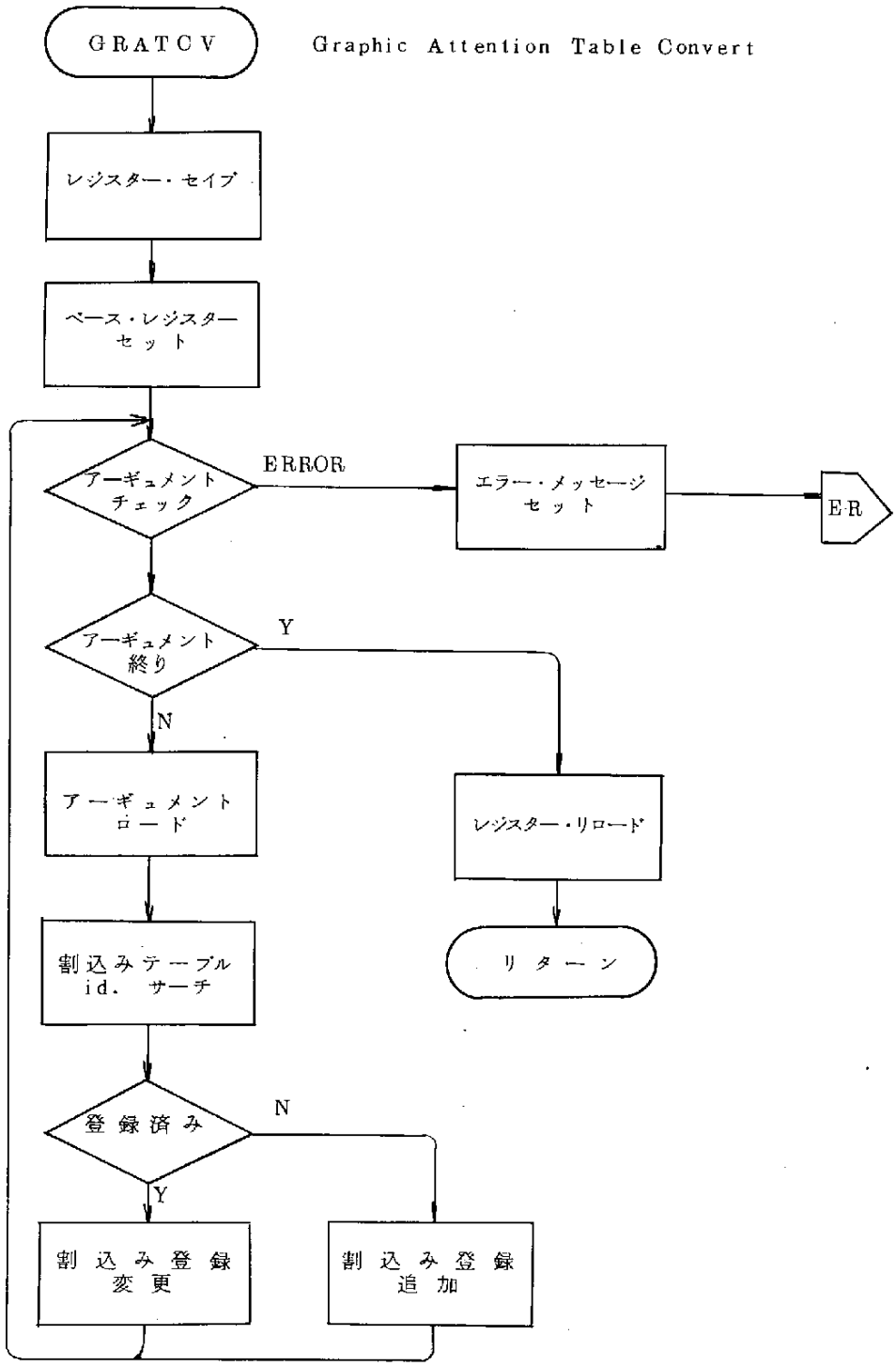


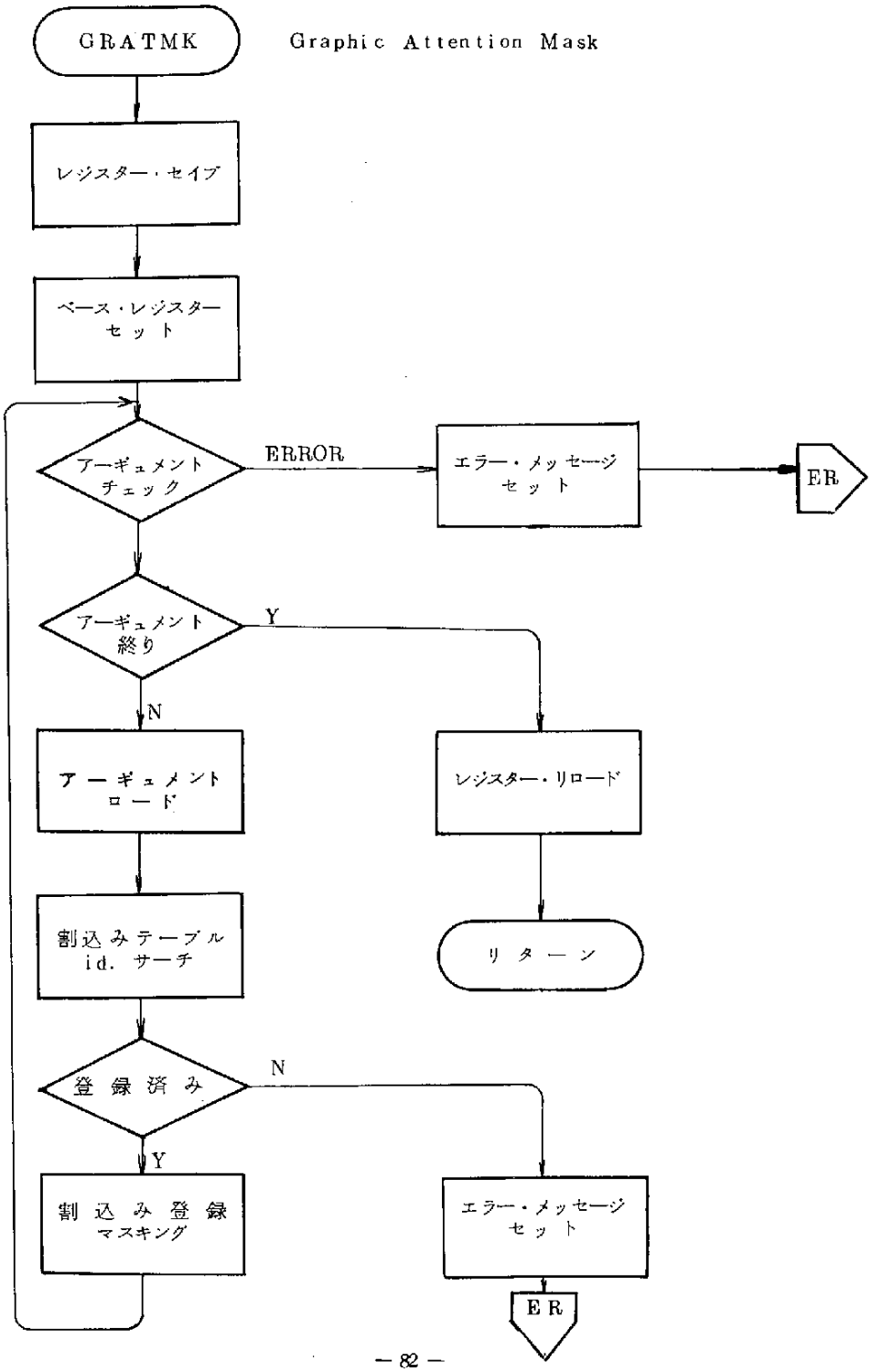
図 4.7　リエントラント構造でないプログラムの割込み制御

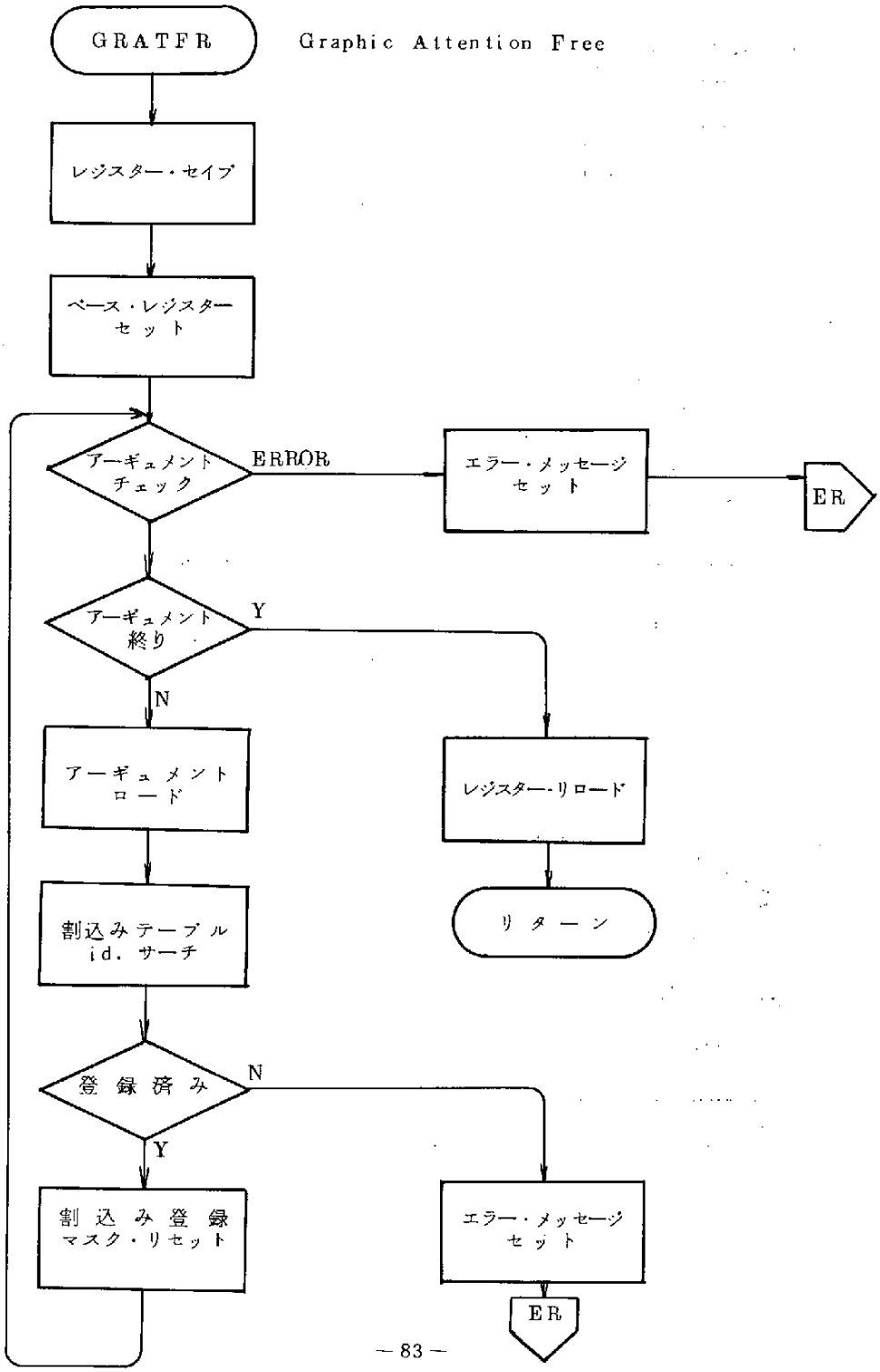
GIASPのブロック・チャート

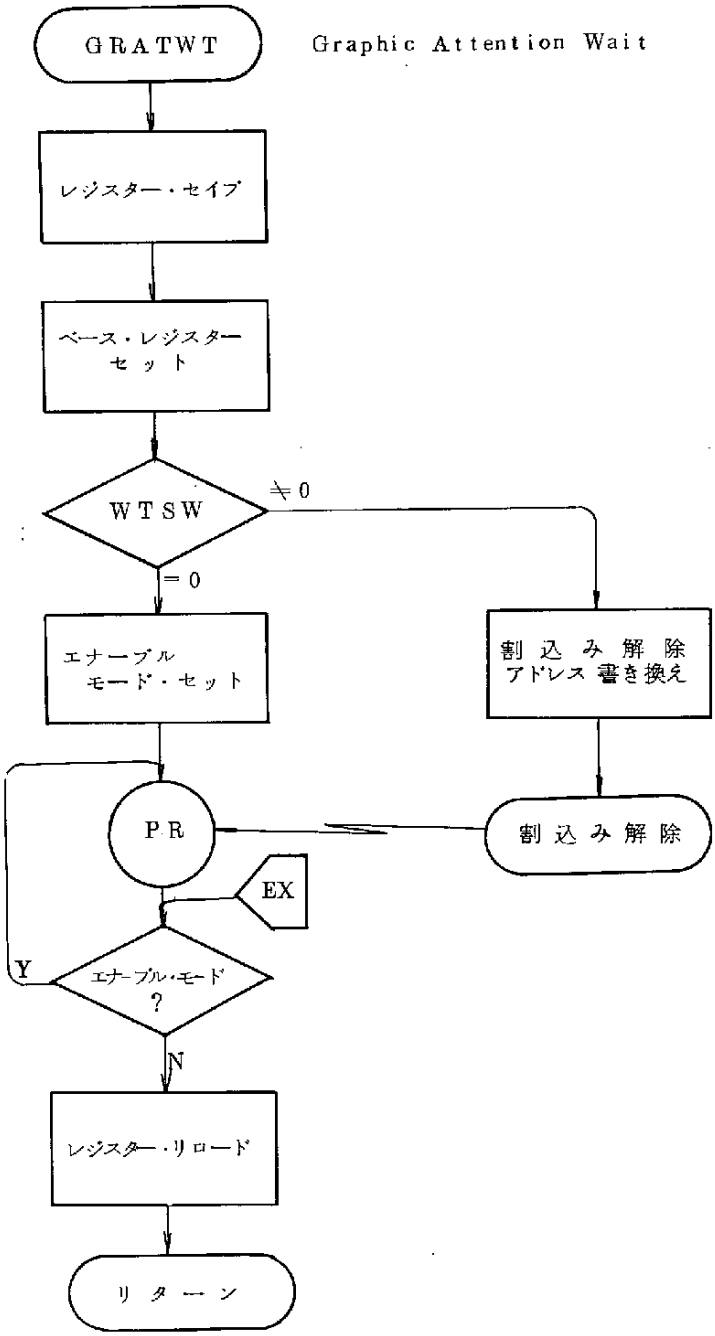


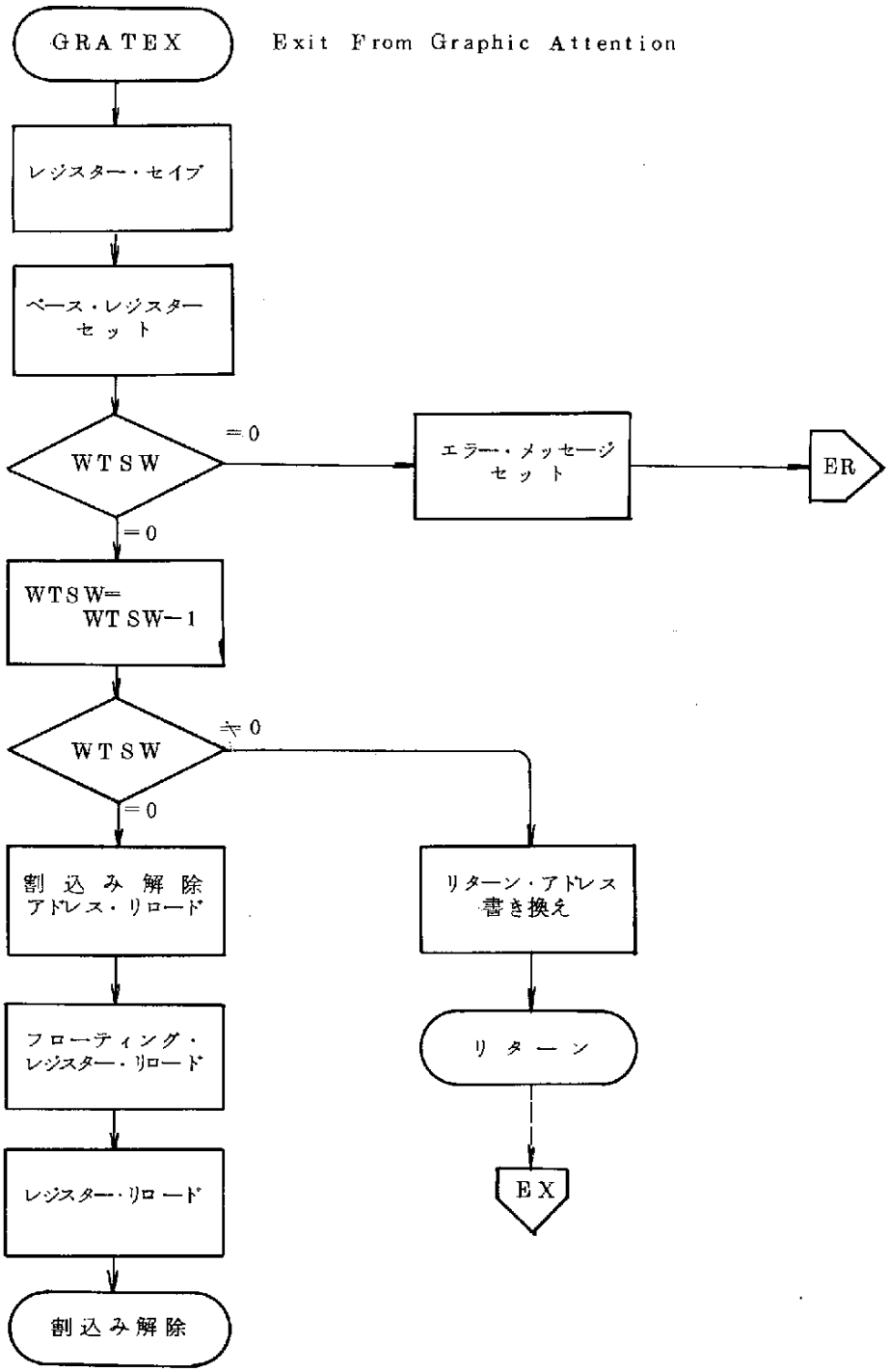
＊1　表4.1参照

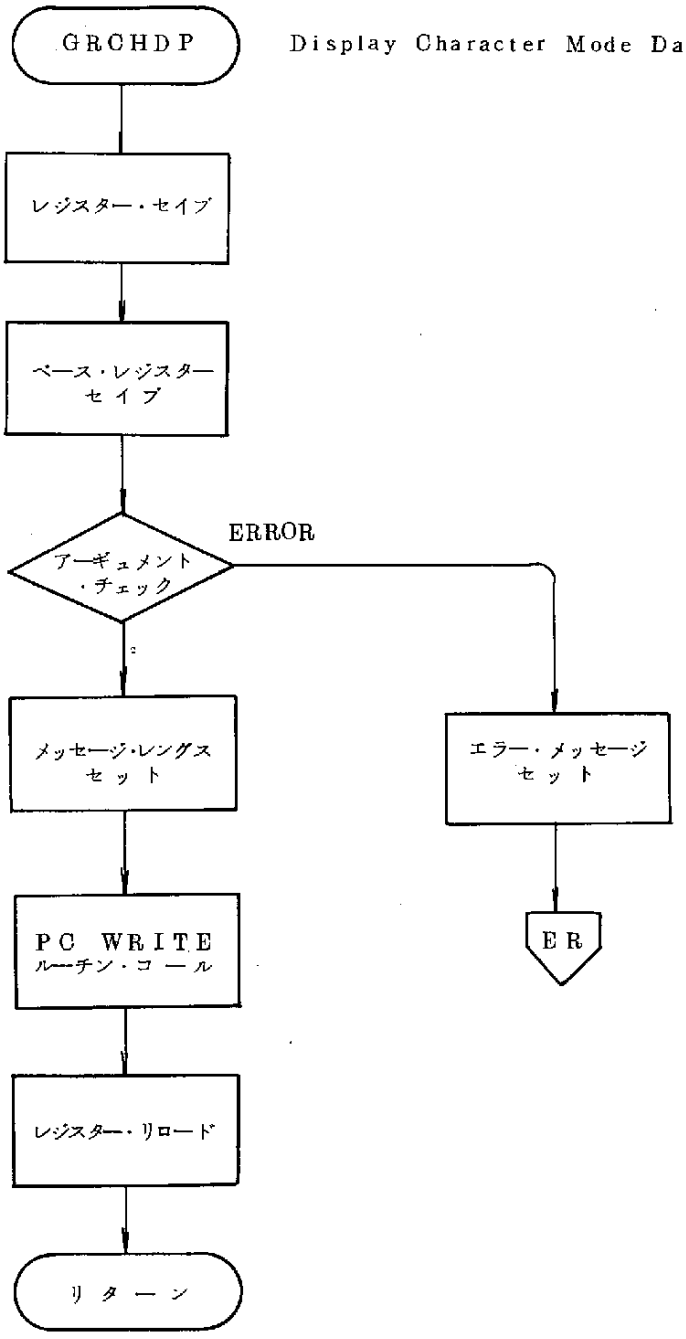

GRATCV
Graphic Attention Table Convert
レジスター・セイブ
ベース・レジスター
セット
アーギュメント
チェック
ERROR
エラー・メッセージ
セット
ER
アーギュメント
終り
Y
N
アーギュメント
ロード
レジスター・リロード
リターン
割込みテーブル
id. サーチ
登録済み
N
Y
割込み登録
変更
割込み登録
追加

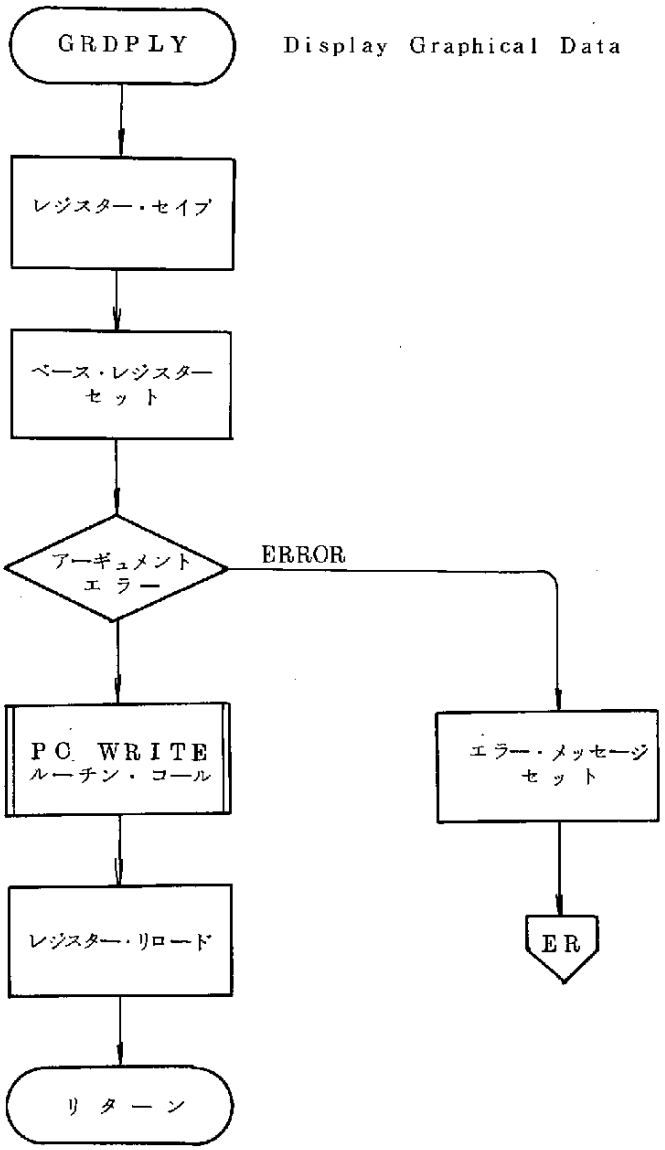

GRATMK
Graphic Attention Mask
レジスター・セイブ
ベース・レジスター セット
アーギュメント チェック
ERROR
エラー・メッセージ セット
ER
アーギュメント 終り
Y
N
アーギュメント ロード
レジスター・リロード
割込みテーブル id. サーチ
リターン
登録済み
N
Y
割込み登録 マスキング
エラー・メッセージ セット
ER

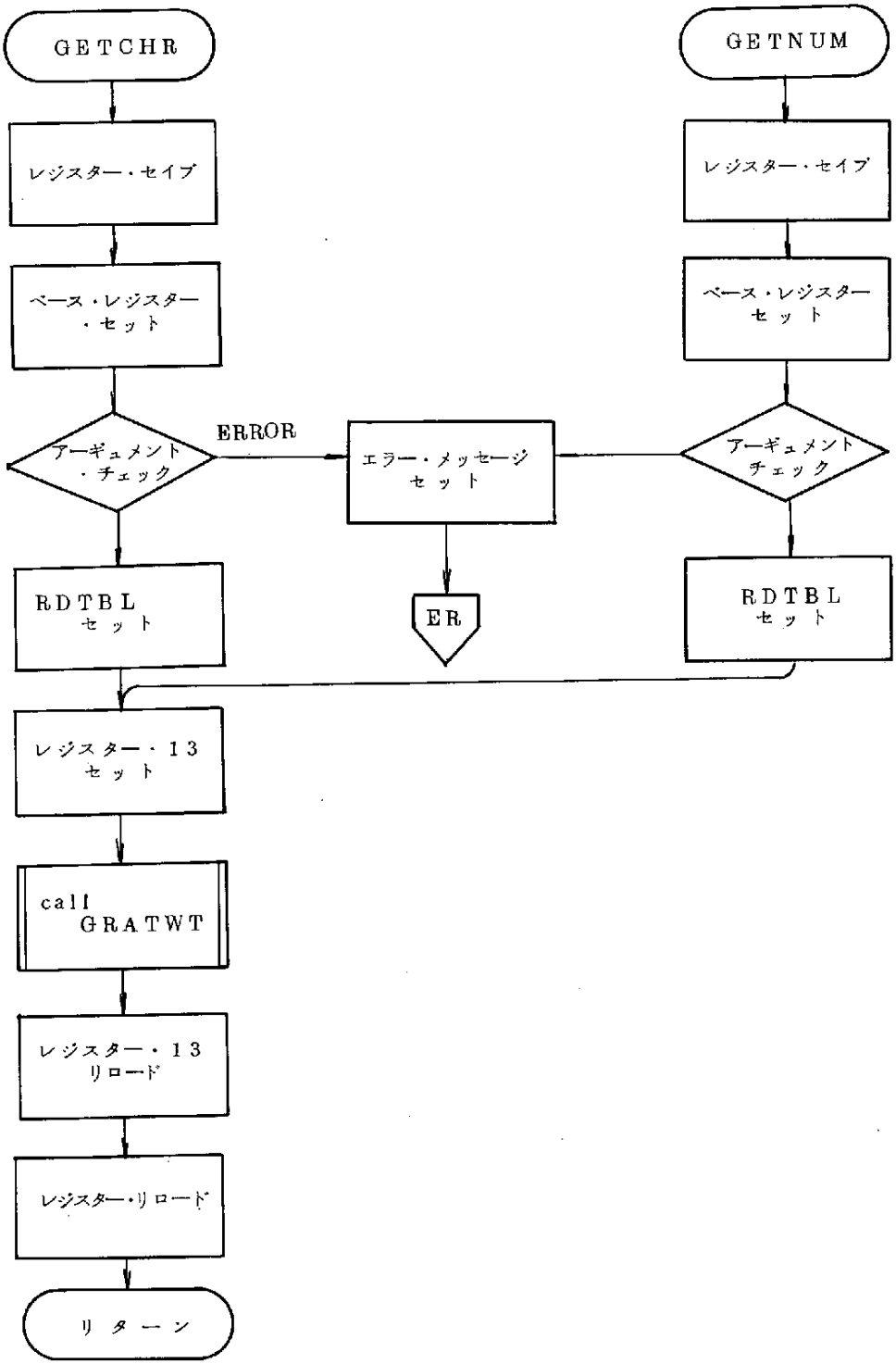

GRATFR
Graphic Attention Free
レジスター・セイブ
ベース・レジスター
セット
アーギュメント
チェック
ERROR
エラー・メッセージ
セット
ER
アーギュメント
終り
Y
N
アーギュメント
ロード
レジスター・リロード
リターン
割込みテーブル
id. サーチ
登録済み
N
Y
割込み登録
マスク・リセット
エラー・メッセージ
セット
ER

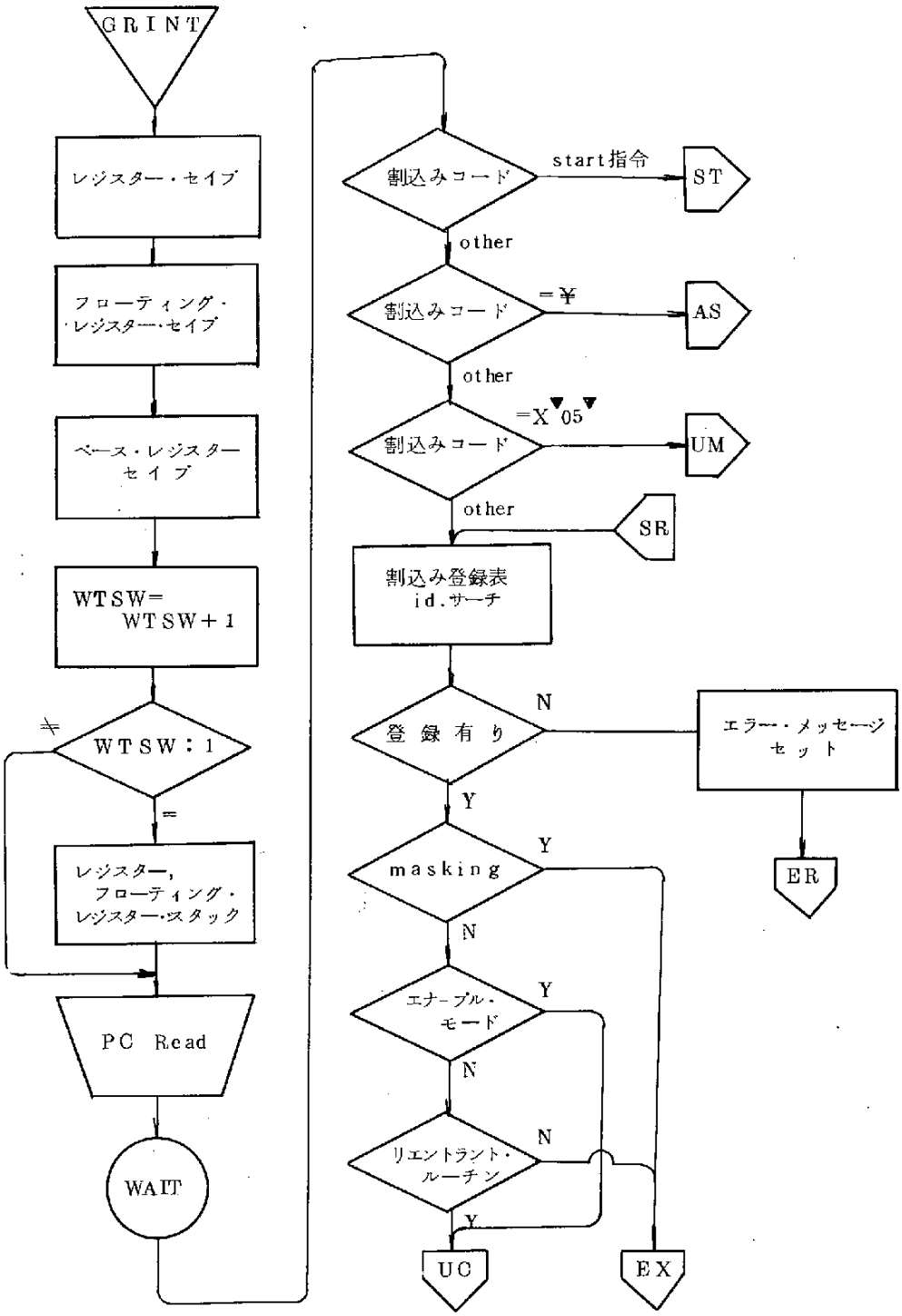

GRATWT
Graphic Attention Wait
レジスター・セイブ
ベース・レジスター
セット
WTSW
≒0
=0
エナーブル
モード・セット
割込み解除
アドレス書き換え
PR
割込み解除
EX
Y
エナーブル・モード
?
N
レジスター・リロード
リターン

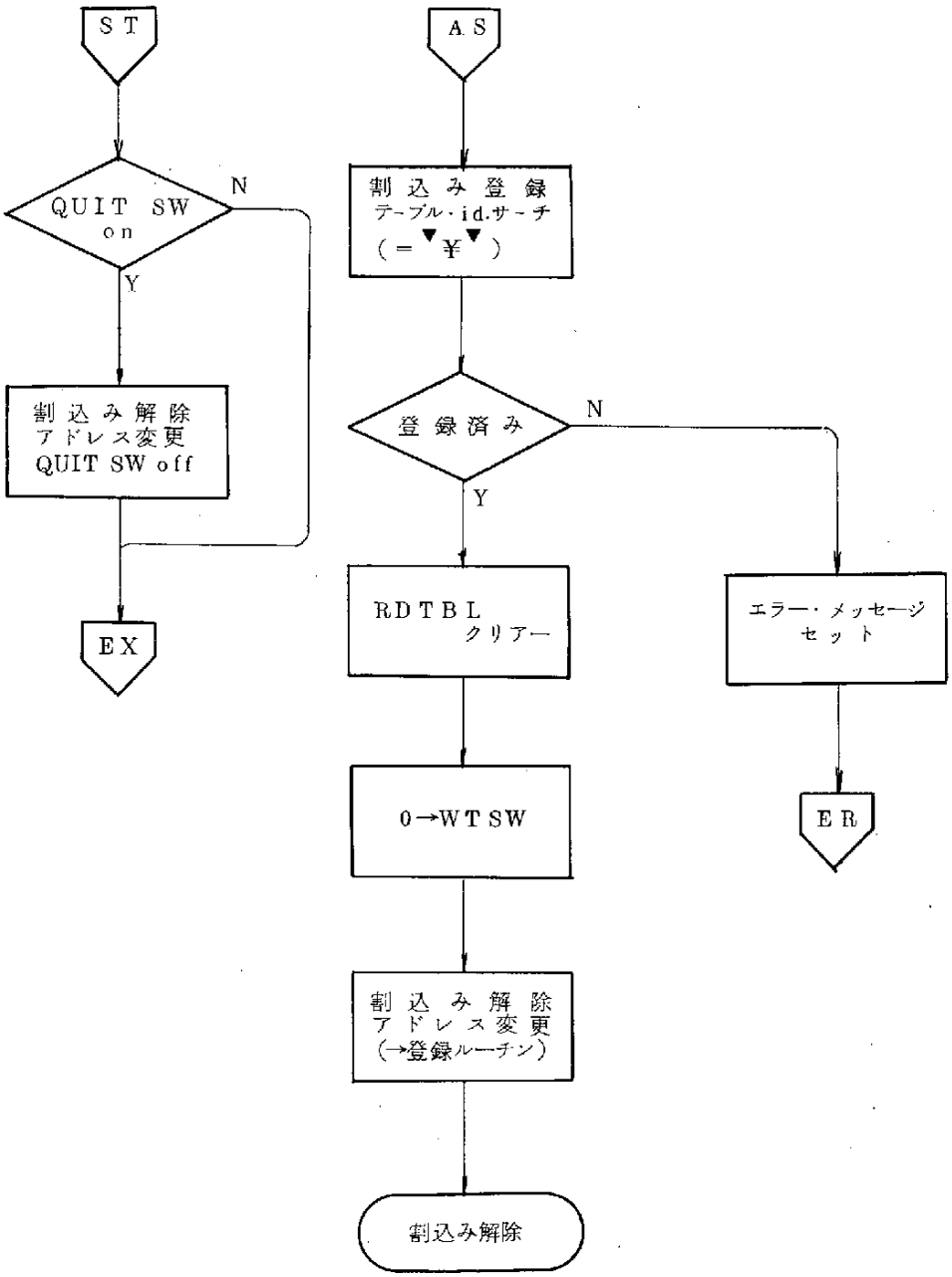

GRATEX
Exit From Graphic Attention
レジスター・セイブ
ベース・レジスター
セット
WTSW
=0
エラー・メッセージ
セット
ER
=0
WTSW=
WTSW−1
WTSW
≠0
=0
割込み解除
アドレス・リロード
リターン・アドレス
書き換え
フローティング・
レジスター・リロード
リターン
レジスター・リロード
EX
割込み解除

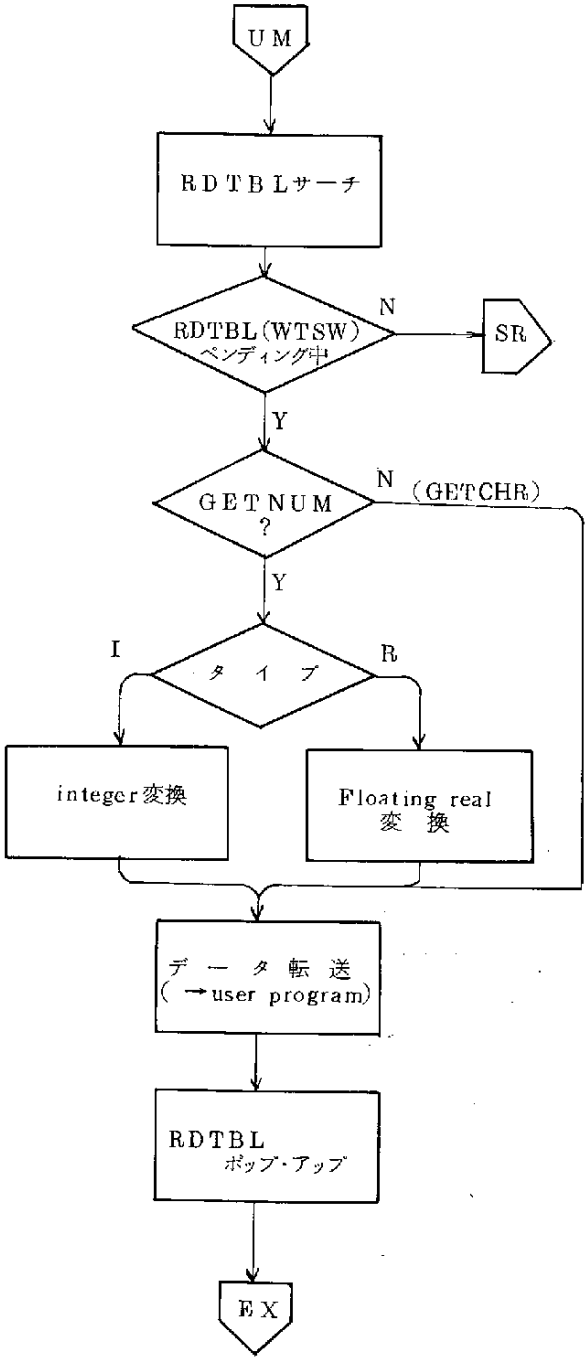

GRCHDP
Display Character Mode Data
レジスター・セイブ
ベース・レジスター
セイブ
ERROR
アーギュメント
・チェック
メッセージ・レングス
セット
エラー・メッセージ
セット
PC WRITE
ルーチン・コール
ER
レジスター・リロード
リターン

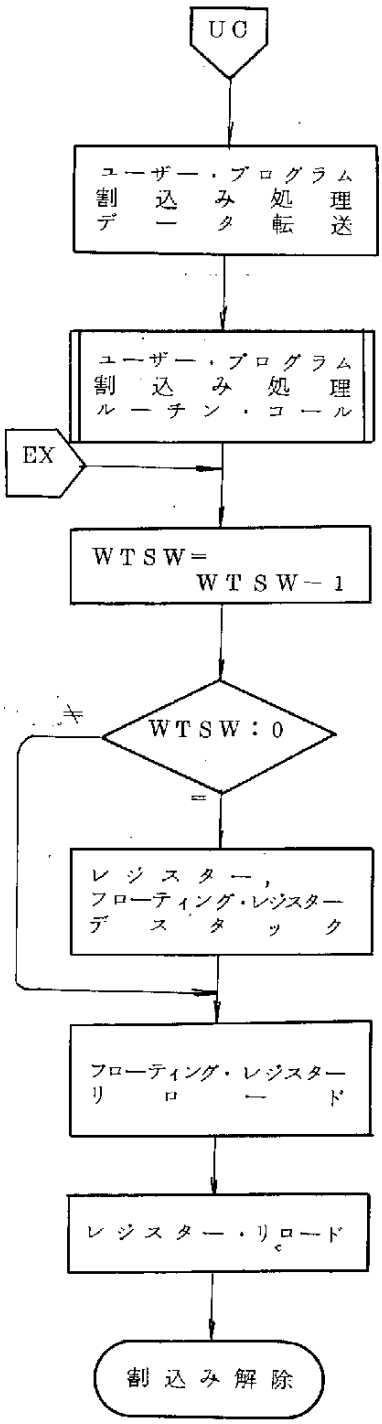

GRDPLY
Display Graphical Data
レジスター・セイブ
ベース・レジスター
セット
アーギュメント
エラー
ERROR
PC WRITE
ルーチン・コール
エラー・メッセージ
セット
レジスター・リロード
ER
リターン

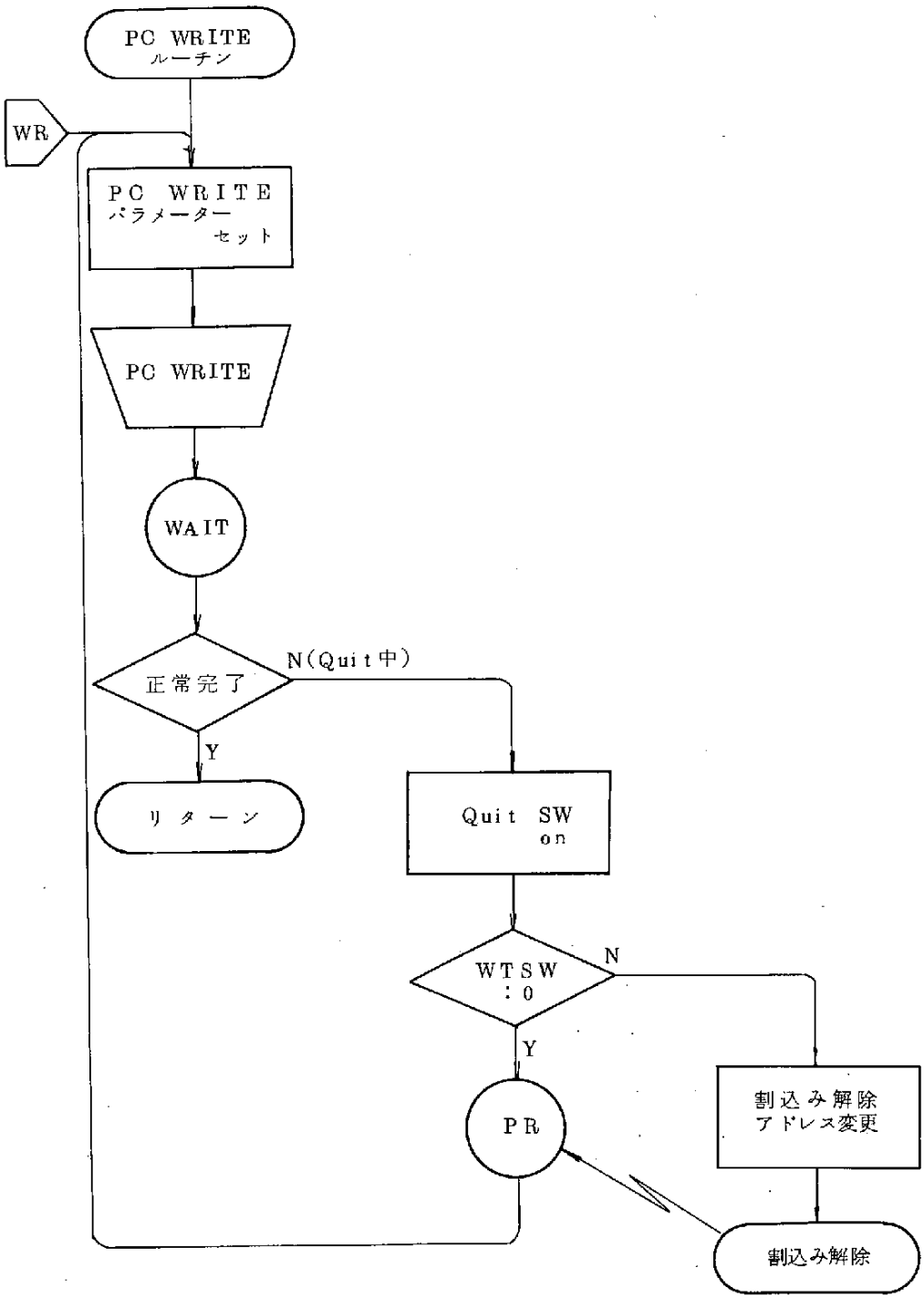

GETCHR
レジスター・セイブ
ベース・レジスター・セット
アーギュメント・チェック
ERROR
エラー・メッセージ・セット
ER
RDTBL セット
レジスター・13 セット
call GRATWT
レジスター・13 リロード
レジスター・リロード
リ タ ー ン
GETNUM
レジスター・セイブ
ベース・レジスター セット
アーギュメント チェック
RDTBL セット


GRINT
レジスター・セイブ
フローティング・
レジスター・セイブ
ベース・レジスター
セイブ
WTSW=
WTSW＋1
≠
WTSW：1
=
レジスター,
フローティング・
レジスター・スタック
PC Read
WAIT
割込みコード
start指令
ST
other
割込みコード
=¥
AS
other
割込みコード
=X'05'
UM
other
SR
割込み登録表
id.サーチ
登録有り
N
エラー・メッセージ
セット
ER
Y
masking
Y
N
エナーブル・
モード
Y
N
リエントラント・
ルーチン
N
Y
UC
EX


ST
QUIT SW
on
N
Y
割込み解除
アドレス変更
QUIT SW off
EX
AS
割込み登録
テーブル・id.サーチ
(=▼¥▼)
登録済み
N
Y
RDTBL
クリアー
0→WT SW
割込み解除
アドレス変更
(→登録ルーチン)
割込み解除
エラー・メッセージ
セット
ER


UM
RDTBLサーチ
RDTBL(WTSW)
ペンディング中
N
SR
Y
GETNUM
?
N
(GETCHR)
Y
I
タイプ
R
integer変換
Floating real
変換
データ転送
(→user program)
RDTBL
ポップ・アップ
EX


UC
ユーザー・プログラム
割込み処理
データ転送
ユーザー・プログラム
割込み処理
ルーチン・コール
EX
WTSW=
WTSW−1
WTSW：0
レジスター，
フローティング・レジスター
データスタック
フローティング・レジスター
リロード
レジスター・リロード
割込み解除


PC WRITE
ルーチン
WR
PC WRITE
パラメーター
セット
PC WRITE
WAIT
正常完了
N(Quit中)
Y
リターン
Quit SW
on
WTSW
:0
N
Y
PR
割込み解除
アドレス変更
割込み解除

4.2 グラフィック・プログラミング・サポート

グラフィック・プログラミング・サポートは，ＣＧＯＳの下で，グラフィック・ＣＲＴへの図形表示を志すユーザ・プログラムに対して，表示図形データ編集の煩雑な処理を代行するサブ・ルーチン・パッケージ，ＧＳＰ（Graphic Subroutine Package）を提供する。

4.2.1 グラフィック・サブルーチン・パッケージ（以下略してＧＳＰと呼ぶ）はＣＧＯＳの下でグラフィック・ＣＲＴを利用するユーザーに対し，図形データーを記述する上での便宜をはかるために用意されたプログラム・パッケージである。

通常それは，ＦＯＲＴＲＡＮのサブプログラム呼び出し形式でユーザー・プログラムにリンケージ・バインドされ，ユーザー・レベルで記述された図形データを所定の内部形式データに変換する。又これをＧＩＡＳＰ，ＧＪＭＯＮのもとにHITAC-8811処理装置に転送しＧＣＨによってＣＲＴに表示する。

ＧＳＰの提供する主な機能を挙げると次のようになる。

1. 図形データを高いレベルの言語で記述出来ること。

ユーザーは複雑な内部形式データを意識することなく，ＦＯＲＴＲＡＮのサブルーチン・コールの形で容易にその意図する図形を表現することが出来る。

2. 図形データを構造化して扱えること。

ＧＳＰでは図形データをフレーム・エンティティ・エレメントという３つのレベルで構造化して扱うことが可能である。

3. 図形データ集合の間で修正，追加，コピー等が可能であること。

既に作成された図形データを他のフレームに複写したり，一部を修正したり，別の図形要素を附加したりすることがプログラムで容易に出来る。

4. 図形データのサブルーチン化が可能である。

特定の図形データをサブルーチンとして登録し随時それを呼び出して使うことが可能である。

5. 図形データの記録，再成が可能である。

一時的に図形データを他の２次記憶に保存しておき，必要な時にそれを呼び出し表示することが出来る。

6. Ｈ－８８１１の諸機能が十分発揮出来るように工夫されていること。

Ｈ－８８１１の持つ豊富なハードウェア機能を出来るだけ生かせるように配慮され

ている。

例　Curve Fitting

7. インタラクティブな使用に耐えるよう，その多くのモジュールは，リエンタラブルに作られている。

4.2.2　GSPの扱うデータ構造

GSPでは，図形データをフレーム，エンティティ，エレメントと呼ぶ三つのレベルで構造化して扱う。従ってユーザーは図形データを記述する時に，自らの責任でこの階層関係を明示しなくてはならない。

ここではフレーム，エンティティ，エレメントが具体的にどのような関係にあるかと云うこと，又，それに対応する内部形式データがどのような構造をとるかについて簡単に説明する。

フレーム，エンティティ，エレメントは

フレーム⊐エンティティ⊐エレメント

の関係にあり，それぞれは次の様に定義される。

1. フレーム

いくつかのエンティティから構成され，論理的にはCRTに表示する一画面に相当し，表示の最小単位である。また内部形式データとしては，一回の転送単位であるDDDに相当する。表示図形を作成しようとするユーザは必ずフレームを定義しこの中に個々の図形要素を定義していかなくてはならない。

また後で述べるように，フレーム・アップデイトの対象となる。

2. エンティティ

いくつかのエレメントから構成され，論理的に識別可能な最小の図形単位である。即ちCRTに表示された段階ではピッキングの対象となり又内部データに於ては修正，コピーの対象となり得る。

3. エレメント

図形を構成する物理的な図形要素であると云える。単に図形要素を定義するにとどまる。

図4.8はフレーム，エンティティ，エレメントと実際の表示図形，内部データの関係を図示したものである。

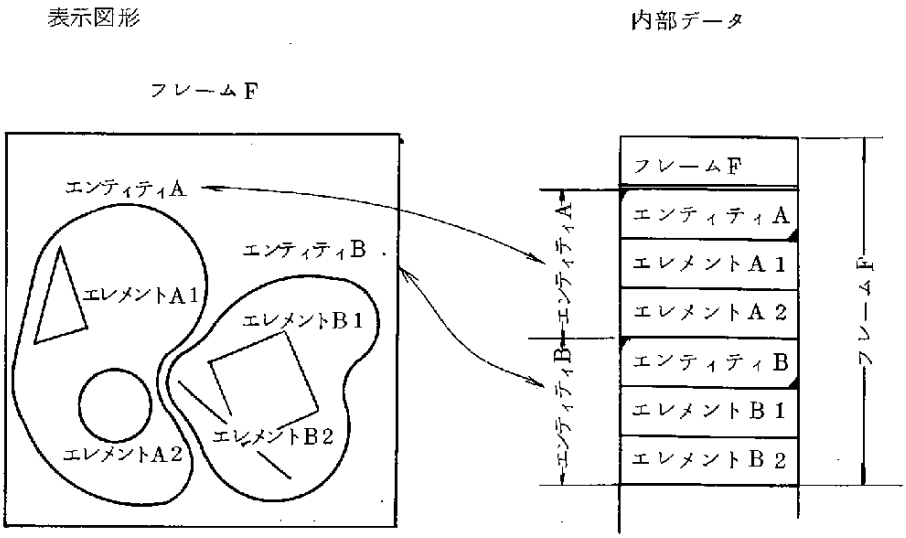


図4.8　フレーム，エンティティ，エレメントの関係

図4.8では，一画面は2つのエンティティA，Bより構成されている。またエンティティAは2つのエレメントA1，A2から成っている。従って図形A1とB1は論理的に区別できるが，A1とA2またはB1とB2は同一エンティティであるため論理的に区別出来ない。

図4.9は画面が2つのフレームから構成される様子を示している。このように，複数個のフレームを同時に表示することも可能である。

またこの時画面上では図形が合成されて表示されるが各フレーム自体は何等作用を受けない。つまりフレーム情報はそのまま保存される。フレーム自体を書き変えるのは後で述べるアップデート・フレームの機能による。

画面が２つのフレームＦ１，Ｆ２から構成される場合

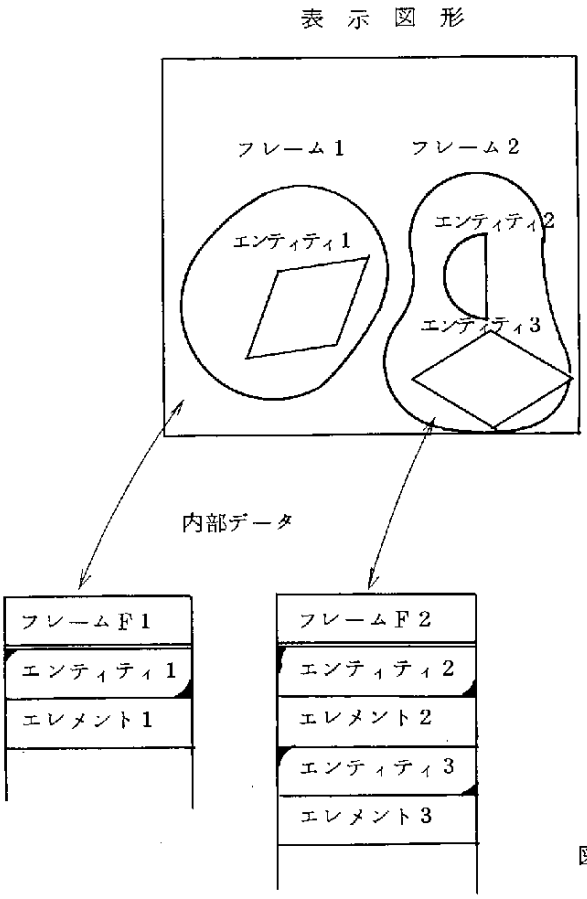


図 4.9

4.2.3　ＧＳＰのプログラム構造

ＧＳＰは全部で２３個のルーチンから構成されている。

ＵＰＤＦＲＭ，ＧＣＲＶＦＴ，ＧＡＲＣを除く各ルーチンは，独立したプログラム単位としては存在せず全体で一つのモジュール形式をとつている。

従ってＧＳＰ内の特定のルーチンだけを取り出して自己のプログラムと結合し利用する事は出来無い。常にＧＳＰ全体がバインドされる形になる。

各ルーチンは性格上大きく次の様に分類される。

1. 図形構造を記述するもの。
2. 図形要素を記述するもの。

3. 図形データ集合間での修正

コピー等の動作を指示するもの。

4. 図形が表示される時のＣＲＴ

制御情報を定義するもの。

5. 定義された図形データをＣＲＴに表示する指示を与えるもの。

6. その他

以下に個々のルーチンとその機能を簡単に記す。

1.に属するもの

ＯＰＮＦＲＭ ： フレーム領域を開設する。

ＣＬＳＦＲＭ ： フレーム領域を開放する。

ＢＧＮＥＴＹ ： エンティティの開設を宣言する。

ＥＮＤＥＴＹ ： 一つのエンティティの終りを指示する。

2.に属するもの

ＧＰＯＳＩＴ ： ビームのポジショニングを指示する。

ＧＶＥＣＴＲ ： 線分を定義する。

ＧＳＴＲＮＧ ： 文字例を定義する。

ＧＮＵＭＢＲ ： 数値データを表示パターンに変換する。

ＧＳＵＢＥＴ ： 図形サブルーティンを呼び出す。

ＧＣＲＶＦＴ ： 点列をフィッティングする。

ＧＡＲＣ ： 円弧を定義する。

ＧＸＡＸＩＳ
ＧＹＡＸＴＳ } ： 座標軸を定義する。

ＣＰＹＥＴＹ ： 指定されたエンティティをコピーする。

3.に属するもの

ＵＰＤＦＲＭ ： フレーム間での更新を指示する。

ＲＭＶＥＴＹ ： 指定したフレームから指定したエンティティの削除を指示する。

4.に属するもの

ＭＯＤＢＬＫ ： エンティティのモード・コントロール情報を定義する。

ＭＶＧＢＬＫ ： ムーヴィング・エンティティのコントロール情報を定義する。

ＣＮＴＥＴＹ ： エンティティの制御情報の内容を変更する。

5.に属するもの

DRWFRM　：　指定されたフレームをCRTに表示する。

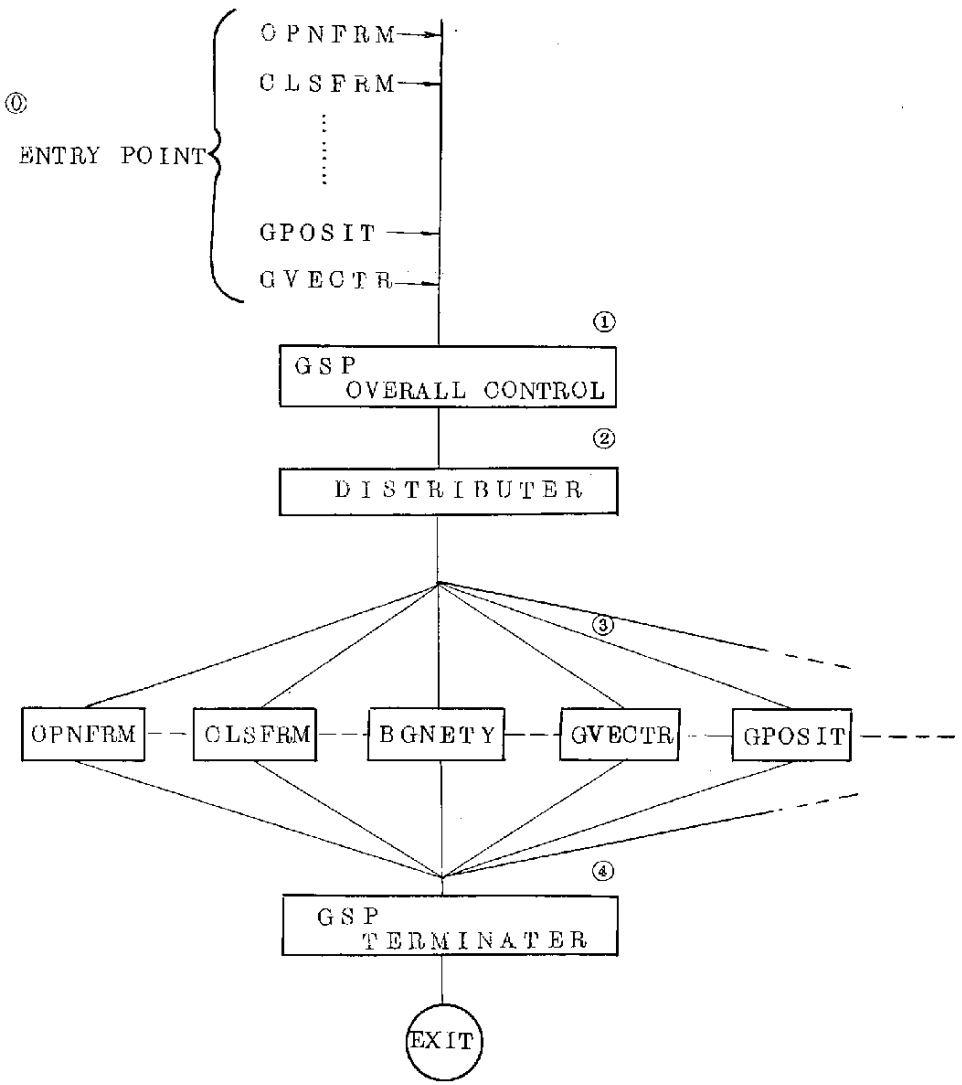
6.に属するもの

SCLFRM　：　フレームにかかるスケール・ファクターを変更する。

VIEWOF　：　CRT上に表示された図形の表示を一時的に中断する。

VIEWON　：　VIEWOFで中断された表示を再開する。

GSP全体の構成を図示すると図4.10のようになる。



各ブロックは主として次の様な機能を果す。

ブロック⓪：多くのエントリー・ポイントを示している。ユーザー・プログラムが各モジュールを参照するとこの中の対応する点とリンケージがとられる。

ブロック⓪に制御がくると各々はそのＥＮＴＲＹ情報を持って，一旦ブロック①に制御を移す。

ブロック①：ＧＳＰプログラム全体を制御する部分である。

その主な機能は次の通りである。

① レジスター類の退避

② プログラム・ステータスのチエック

③ システム作業域に使用するベースレジスタの値を設定する。

ブロック②：ブロック⓪のエントリー情報から対応するルーティンに制御を移す。

ブロック③：２３ケのモジュールに対応する。

ブロック④：レジスターのリロード等終了処理をする。

4.2.4 システムの動作概要

ＧＳＰ内部で図形データが作成され転送される過程をユーザー・プログラムの流れに沿って説明する。図4.11を参照されたい。

ユーザー・プログラムの例

```
(1) DIMENSION AREA(512)
(2) CALL OPNFRM(▼F2▼, 0, AREA)
(3) CALL MODBLK(MCB,0,0,1,0,0)
(4) CALL BGNETY(▼F2▼,▼PICK▼,MCB)
(5) CALL GPOSIT(0,50,50)
(6) CALL GVECTR(0,1,150,150)
(7) CALL GSTRNG(1,6,▼ADIEU!▼)
(8) CALL ENDETY
(9) CALL DRWFRM(▼F2▼)
      ⋮          ⋮
```

シーケンス(1)

フレーム用領域を確保しておく。

シーケンス(2)　ＯＰＮＦＲＭ

フレームの開設を宣言する。

(1)で定義された領域に▼Ｆ２▼のフレームＩ．Ｄを与えそのアドレスと共にフレーム管理テーブル(図4.11（Ｃ）)に登録する。またフレーム領域の先頭にフレーム情報をセットする。以後この領域はフレーム領域として使用可能となる。(図4.11（Ｃ），（Ｄ）)

シーケンス(3)　ＭＯＤＢＬＫ

モード．コントロールブロックを定義する。

アーギュメント・リストからＭＣＢで示された場所に，ＣＲＴコントロールのための制御ブロックＭＣＢ(図4.11（Ｅ）)を作成する。以後ＭＣＢを指定することによりこの制御情報を参照することが出来る。

シーケンス(4)　ＢＧＮＥＴＹ

エンティティの開設を宣言する。

フレーム管理テーブルをサーチしフレームＩＤ▼Ｆ２▼のフレームアドレスを知る。

またエンティティ・ポインター(図4.11（Ａ）)をこの点にリンクしておく。

フレームＦ２に，エンティティＩＤ，先に定義されたモード・コントロール・ブロック情報等エンティティ情報を設定する。(図4.11（Ｄ）エンティティ情報)

シーケンス(5)　ＧＰＯＳＩＴ

エンティティ・ポインター，フレーム管理テーブルより現在アクセス中のフレームＦ２のアドレスを知る。アーギュメント・リストをデータとしてビーム・ポジショニングのエレメント情報を作成しそのフレームに納める。(図4.11（Ｄ）エレメント１)

シーケンス(6)　ＧＶＥＣＴＲ

(5)と同様にしてベクトル．エレメントを作成する。(図4.11（Ｄ）エレメント２)

シーケンス(7)　ＧＳＴＲＮＧ

(5)～(6)と同様にして文字ストリング．エレメントを作成する。(図4.11（Ｄ）エレメント３)

シーケンス(8)　ＥＮＤＥＴＹ

これまで作成した情報を１エンティティとして完成する。

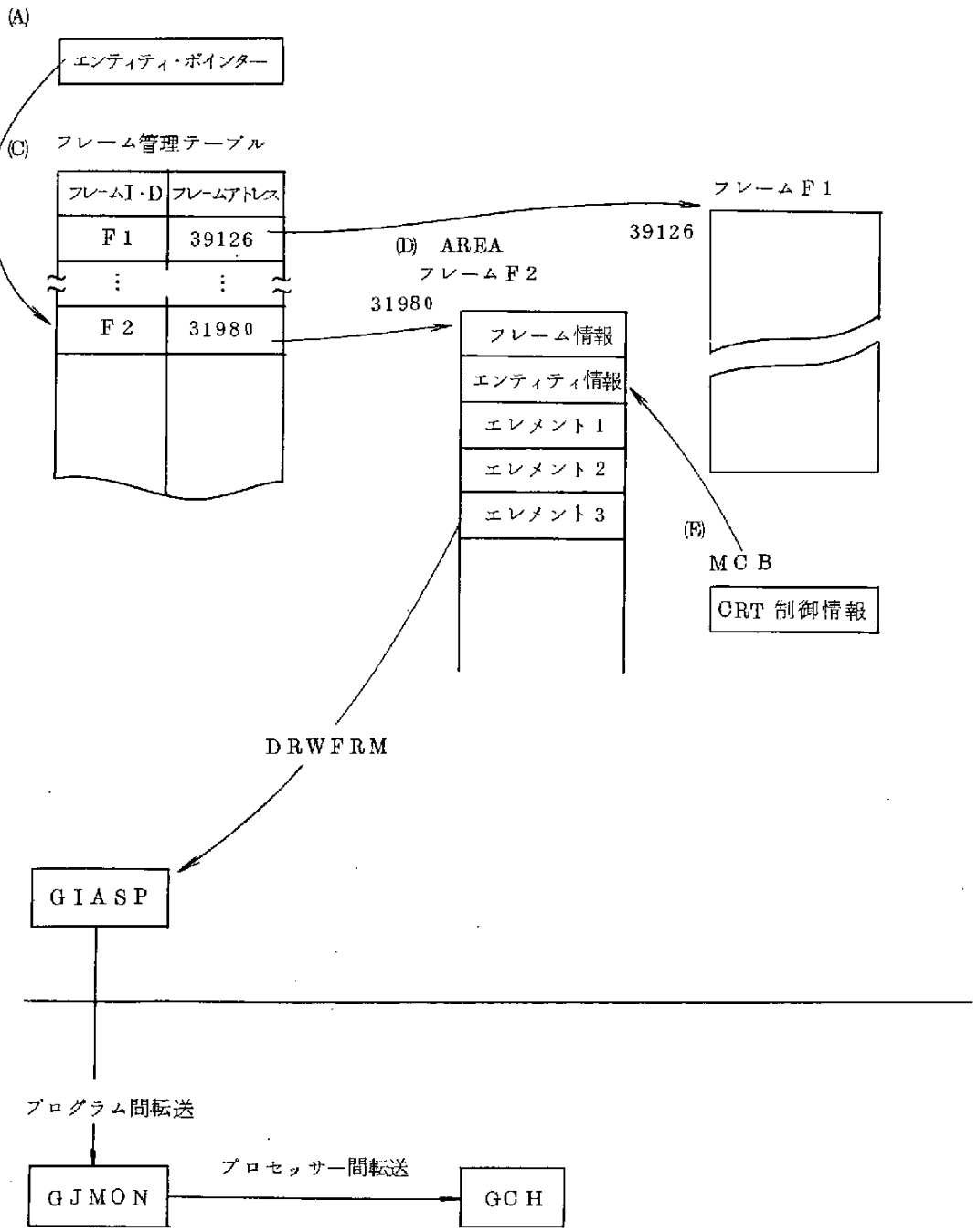


図 4.1 1　ＧＳＰの動作図

エンティティ・ポインターをクリアしておく。

シーケンス(9)　DRWFRM

フレームＩＤ▼Ｆ２▼のフレームをフレーム管理テーブルより探しそのアドレスを知る。そのフレーム内のデータをＧＩＡＳＰによりＧＪＭＯＮに転送する。以下ＧＪＭＯＮ→ＧＣＨの転送が行なわれＣＲＴに表示される。

4.2.5　ＧＳＰ各ルーチンの機能

1. ＯＰＮＦＲＭ　（Open Frame）

〔機能〕

前もって確保しておいた領域をフレーム用領域として登録する。

〔呼び出し形式〕

ＣＡＬＬ　ＯＰＮＦＲＭ（▼ＩＤ▼，ＭＯＤＥ，ＡＤ〔，ＳＣＡＬＥ〕）

〔アーギュメント〕

▼ＩＤ▼　：　フレームを識別する文字列で最初の２文字が有効である。

ＡＤ　：　フレーム用領域の先頭アドレンを記す。５１２ワードの配列

ＭＯＤＥ　：　このフレームのモードを記す。

0　創成フレーム

1　更新フレーム

2　半創成フレーム

3　再成フレーム

(注)　フレームのモードについて，フレームには次の４つのモードがある。

創成フレーム

このフレームが転送されると（ＤＲＷＦＲＭ）それまでＣＲＴ上に表示されていた図形情報をすべてクリアーし新たに図形表示をする。

更新フレーム

このフレームが転送されてもそれまでにＣＲＴ上に表示されていた図形情報は保存され，それに対しエンティティの追加，変更，削除を行うことが出来る。

半創成フレーム

サブルーティン・エンティティが保存されることを除いては創成フ

レームと同じ。

再成フレーム

一旦閉じられたフームやＧＳＰを使わずにユーザープログラムが直接フォーマッティングしたフレーム，あるいは過去に２次記憶上に保存しておいて再びロードしてきたフレーム等，既にフォーマッティングが完成されているものを生かしたい時に指示する。実際のフレームのモードはそのフレームが以前に保持していたモードがそのまま用いられる。

ＳＣＡＬＥ　：　ここで示された値だけフレーム内の座標情報をスケール倍する。デフォルト値は１.０とする。

〔内部処理〕

ＡＤで記されたアドレスをフレームＩＤと共にＧＳＰ内部のフレーム管理テーブルに登録する。（図4.１２）

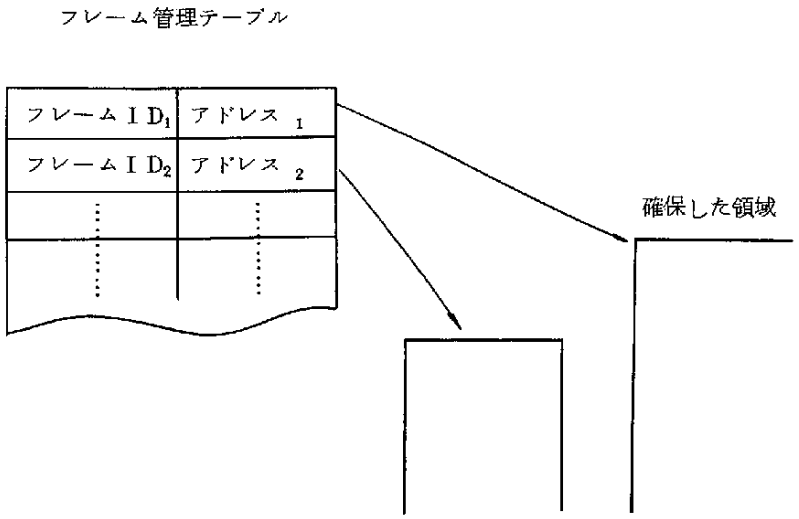


図 4.１２　フレームとフレーム管理テーブル

また開設したフレーム領域にフレーム情報を設定する。（図4.１３）

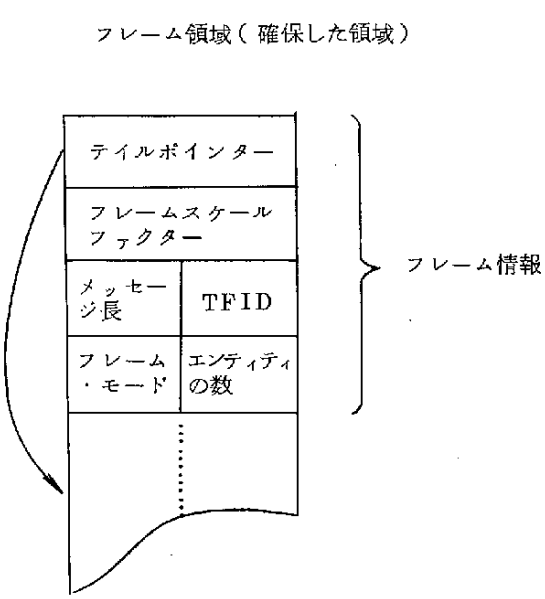


図4.１３　フレーム情報

（例）　配列Aをフレーム用領域として登録しこれにフレームＩＤ▼Ｆ１▼を与える。
またモードは創成モードで，スケールは２倍とする。

```
DIMENSION A(512)
CALL OPNFRM(▼F1▼,0,A,2.)
```

2. ＣＬＳＦＲＭ （Ｃｌｏｓｅ Ｆｒａｍｅ）

〔機能〕

前に定義されたフレーム領域を開放する。

〔呼び出し形式〕

```
CALL CLSFRM(▼ID▼)
```

〔アーギュメント〕

ＩＤ　：　閉じようとするフレームのフレームＩＤを与える。

〔内部処理〕

フレーム管理テーブルからそのＩＤで示されたエントリーを取り除く。

（例）　前の例で示されたフレーム▼Ｆ１▼をクローズする。

ＣＡＬＬ　ＣＬＳＦＲＭ(▼Ｆ１▼)

この結果▼Ｆ１▼で示されるフレーム領域ＡとＧＳＰとの関係が断たれる。

3. ＤＲＷＦＲＭ　（Ｄｒａｗ　Ｆｒａｍｅ）

〔機能〕

指示したフレームをＣＲＴに表示する。

〔呼び出し形式〕

ＣＡＬＬ　ＤＲＷＦＲＭ(▼ＩＤ▼〔，ＩＳＣ〕)

〔アーギュメント〕

▼ＩＤ▼　：　表示しようとするフレームのフレームＩ.Ｄ.を記す。

ＩＳＣ　：　シザリング・パラメータ ≒ ０ の時，画面のシザリングを行なう。デフォルト値は０

〔内部処理〕

ＧＩＡＳＰのＧＲＤＰＬＹを介してフレームをＧＪＭＯＮ，ＧＣＨへ転送する。（詳しくはＧＩＡＳＰを参照）

（例）　フレームＩＤ　ＥＸ　のフレームを表示する。

ＣＡＬＬ　ＤＲＷＦＲＭ(▼ＥＸ▼)

(注)　フレーム▼ＥＸ▼は開設中でなくてはならないし，少くとも一つのエンティティーは定義されていなくてはならない。

4. ＵＰＤＦＲＭ　（Ｕｐｄａｔｅ　Ｆｒａｍｅ）

〔機能〕

▼ＩＤ１▼で示されるフレームが▼ＩＤ２▼で示されるフレームにより更新される。その結果更新されたフレームが▼ＩＤ１▼フレームに残る。フレームのモードは更新前と同じである。

更新はエンティティ単位で行われる。

以下にその機能を記す。

〔呼び出し形式〕

ＣＡＬＬ　ＵＰＤＦＲＭ(▼ＩＤ１▼,▼ＩＤ２▼,Ｋ)

〔アギュメント〕

▼ＩＤ１▼　：　更新の対象となるフレームのフレームＩＤを与える。

▼ＩＤ２▼　：　更新情報を蓄えたフレームのフレームＩＤを与える。

Ｋ　　：　更新が完了した時の完了状態をシステムがセットする。

0　正常終了

0以外　異常終了

1. 置　換　：　ＩＤ１のフレームに▼ＩＤ２▼フレームのエンティティと同じエンティティＩＤを有するエンティティが存在する時▼ＩＤ１▼フレーム中のそのエンティティは▼ＩＤ２▼フレームのエンティティによって置き換えられる。

2. 追　加　：　▼ＩＤ１▼フレーム中に同じエンティティＩＤを有するエンティティが存在しない時は新たに▼ＩＤ１▼フレームにそのエンティティが追加される。

3. 削　除　：　▼ＩＤ１▼フレーム内の特定のエンティティを取り除く。

4. コントロール情報変更　：

▼ＩＤ１▼フレーム内の特定のエンティティのエンティティ制御情報を変更する。

(注)　フレームのモードとの関係

フレームに３つのモードがあることは前に説明した。（ＯＰＮＦＲＭの項参照）この分類は，フレームを受け入れる側（ＧＣＨ）の立場からなされたものとみることが出来る。

従ってＵＰＤＦＲＭにあっては▼ＩＤ１▼フレームと▼ＩＤ２▼フレームのモードの組み合わせに於て全く意味をなさない様な場合が存在し得る。

以上の様な理由から▼ＩＤ１▼フレームと▼ＩＤ２▼フレームのモードの組み合わせに関して次の様な制限を設けている。

下の表に於て○印は許される組み合わせ，×印は許されない組み合わせを意味する。

ID1＼ID2	創成	更新	半創成
創　成	○	○	○
更　新	×	○	×
半創成	×	○	○

5. ＶＩＥＷＯＦ （Display viewer off）

〔機能〕

ＣＲＴの表示を一時中断する。

〔呼び出し形式〕

ＣＡＬＬ ＶＩＥＷＯＦ

〔アーギュメント〕

無し

〔内部処理〕

ＧＣＨに対し表示コマンドの流れを変える指令を送る（ＧＩＡＳＰ参照）

6. ＶＩＥＷＯＮ （Display viewer on）

〔機能〕

前述のＶＩＥＷＯＦで中断された表示を再開する。

〔呼び出し形式〕

ＣＡＬＬ ＶＩＥＷＯＮ

〔アーギュメント〕

無し

〔内部処理〕

ＶＩＥＷＯＦに同じ

7. ＢＧＮＥＴＹ （Begin Entity Definition）

〔機能〕

指定したフレームにエンティティを開設する。これより後でＥＮＤＥＴＹより前に作成されるエレメントはこのエンティティに含まれる。常にＥＮＤＥＴＹと対になって使用される。一つのエンティティの記述が完結する前に別のエンティティの定義を行ってはならない。

〔呼び出し形式〕

ＣＡＬＬ ＢＧＮＥＴＹ（▼ＩＤ１▼,{▼ＩＤ２▼}〔, ＡＤＲ〕）

〔アーギュメント〕

▼ＩＤ１▼ ： エンティティを定義するフレームのフレームＩＤを記す。

▼ＩＤ２▼ ： 定義したエンティティに与えるエンティティＩＤを記す。整数値または４文字からなる文字列で与える。エンティティごとにユニークな

ものである。

I　　：　エンティティ番号（I≠0）

ADR　：　モード・コントロール・ブロックのアドレスを記す。
省略すると標準値が設定される。モード・コントロール・ブロックについては後述する。

〔内部処理〕

エンティティ・ポインタ（現在アクセス中のフレームを示す）を該当するフレームにリンクし，フレームにエンティティ定義情報を設定する。（図4.14）

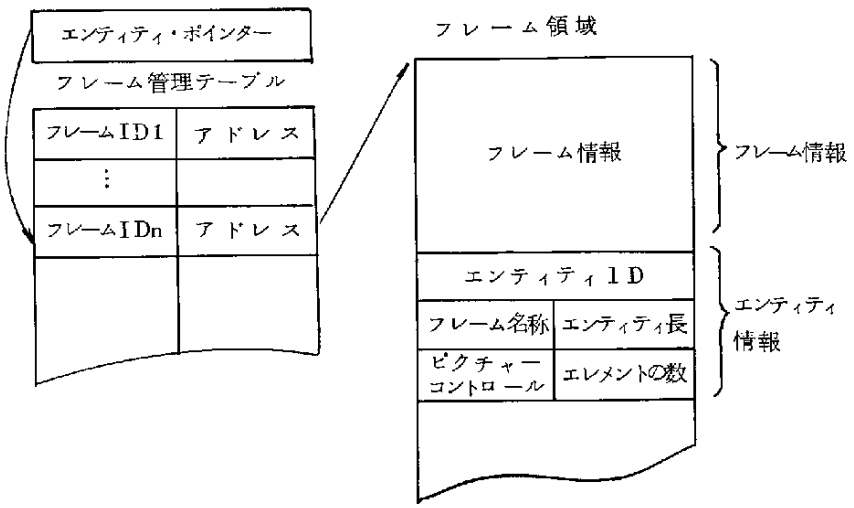


図4.14

(例)　フレームID▼F1▼で示されるフレームにエンティティID▼EXP1▼のエンティティを開設する。モード・コントロールブロックのアドレスはMCBとする。

```
CALL BGNETY(▼F1▼,▼EXP1▼,MCB)
      ⌇
  エレメント記述
      ⌇
CALL ENDETY
```

㊟ MCBで示されたモードコントロールブロックはこれより以前にMODBLKで定義されていなければならない。

8. ENDETY (End Entity Definition)

〔機能〕

BGNETYで開設されたエンティティの完了作業を行う。

〔呼び出し形式〕

CALL ENDETY〔(MVBLK)〕

〔アーギュメント〕

MVBLK : このアーギュメントを指定するとムービング・エンティティとして処理する。ムービング情報はMVBLKで示された所に定義されていること。

〔内部処理〕

開設されていたエンティティを1エンティティとして完成させ，フレームとの結合をおこなう。またムービング指定のある時はムービング情報をそのエンティティに付加する。

9. MODBLK (Entity Mode Control Block Build)

〔機能〕

アーギュメント・リストからモード・コントロール・ブロックを作成する。

〔呼び出し形式〕

CALL MODBLK(ADR〔,MODE〔,DIR〔,SC〔,BR〔,PIC〕〕〕〕〕)

〔アーギュメント〕

ADR : モードコントロールブロックに割り当てる領域アドレスを記す。

＊モードコントロールブロックは，4バイト長

MODE : エンティティの性格を指定する。

0 ノーマルエンティティ

1 サブルーティンエンティティ

デフォルト値は0とする。

DIR : 表示ベクトルの方向を指定する。

0 △x，△y標準方向

1 △x符号反転

2　△y符号反転

3　△x，△y符号反転

デフォルト値は0とする。

SC　：図形に与える倍率を指定する。

1　1倍

2　2倍

3　3倍

4　4倍

5　5倍

6　6倍

7　7倍

8　8倍

デフォルト値は1とする。

BR　：表示された時の輝度を指定する。

0　高輝度

1　中輝度

2　低輝度

3　ブランク

デフォルト値は0とする。

PIC　：ライトペンによるピッキングの可否を指示する。

0　ピッキング許容

1　ピッキング禁止

デフォルト値は0とする。

〔内部処理〕

指定されたアドレスにモード・コントロール・ブロックを作成する。

10. CNTETY （Change Entity Mode Control）

〔機能〕

更新モード・フレーム中に1エンティティとして設置する。DRWFRMで転送されると該当するエンティティのCRT制御情報を変更する。

またUPDFRMで他のフレームを更新するとそのフレーム内の対応するエンティテ

ィのＣＲＴ制御情報を書き替える。

〔呼び出し形式〕

ＣＡＬＬ　ＣＮＴＥＴＹ(▼ＩＤ１▼,▼ＩＤ２▼,〔ＭＣＢ〕)

〔アーギュメント〕

▼ＩＤ１▼：　開設されている更新モードのフレームを指示する。

▼ＩＤ２▼：　対象とするエンティティのエンティティＩＤを記す。

ＭＣＢ　：　変更しようとする内容を備えたモード・コントロール・ブロックのアドレスを記す。

〔内部処理〕

指定された更新モードフレームにＣＲＴ制御情報だけを備えたエンティティを作成する。

11. ＲＭＶＥＴＹ　(Remove Entity)

〔機能〕

更新モード・フレーム中に１エンティティとして配置する。

ＤＲＷＦＲＭによって転送されると表示中のエンティティの内から同じエンティティＩＤを持つものを探し削除する。

ＵＰＤＦＲＭで他のフレームを更新すると対応するエンティティをそのフレームから取り除く。

〔呼び出し形式〕

ＣＡＬＬ　ＲＭＶＥＴＹ(▼ＩＤ１▼,▼ＩＤ２▼)

〔アーギュメント〕

ＩＤ１　：　開設されている更新モードのフレームを指定する。

ＩＤ２　：　削除しようとするエンティティのエンティティＩＤを記す。

〔内部処理〕

指定された更新モード・フレームにエンティティ削除の指令情報を作成する。

12. ＳＣＬＦＲＭ　(Change Frame Scaling Facton)

〔機能〕

すでに開設されているフレームのスケーリング・ファクターを変更する。

〔呼び出し形式〕

ＣＡＬＬ　ＳＣＬＦＲＭ(▼ＩＤ▼,ＳＣＡＬＥ)

〔アーギュメント〕

ＩＤ　：　スケール値を変更しようとするフレームを指定する。

SCALE　：　スケーリング・ファクターを与える。

〔内部処理〕

ＩＤ　で記されたフレームのスケーリング・ファクターを書き替える。

13. MVGBLK (Moving Control Block Build)

〔機能〕

ムービング・エンティティのための制御ブロックを作成する。

ムービング・エンティティとは，グラフイック・ＣＲＴ上で，プログラム上でいちいち更新情報を作成しなくとも，リアル・タイムに連続的な平行移動ができる特質を持ったエンティティである。一つのムービング・ブロック内に，移動の繰り返し，繰り返しに伴った$\triangle x$，$\triangle y$の方向変換，$\triangle x$，$\triangle y$の入れ替え等の指示を組み合わせて，かなり，複雑な移動まで実現できる。

〔呼び出し形式〕

ＣＡＬＬ　ＭＶＧＢＬＫ(ＡＤ,i1,j1,k1,l1,XAR1,YAR1,in,jn,……

……kn,ln〔,XARn,YARn〕)

〔アーギュメント〕

ＡＤ　：　コントロールブロックを作成する領域の先頭アドレスを記す。

（最大６２ワードの配列）

i　：　ループのレベルを与える。０～９の範囲で記す。０は最小レベルで，ループの入れ子の一番内側となる。

j　：　ループ回数を与える。１～∞（∞の時は０を与える）

k　：　ループ時の$\triangle x$，$\triangle y$符号反転，及び$\triangle x$，$\triangle y$の入れ替え指定

0　変化させず

1　$\triangle y$符号反転

2　$\triangle x$符号反転

3　$\triangle x$，$\triangle y$の入れ替え

4　１と２の組み合わせ

5　１と４の組み合わせ

6　２と４の組み合わせ

7　1，2，4の組み合わせ

1　：このループに含まれる$\triangle x$，$\triangle y$の組数（より小さいループに含まれる組は除く）

XAR：ℓで指定した数の$\triangle x$の要素を含む配列名または変数名で，ℓが1の場合は単一変数名であるが，この場合は$\triangle x$の値そのものでもよい。この値はCRT上の実際の移動量を示す。

また$\ell=0$の場合はXARの指定を省略してよい。

YAR：XARの$\triangle x$が$\triangle y$に変ったもので他の条件は同じ。

〔内部処理〕

アーギュメントに従って，ADで記された場所にムービング・コントロール・ブロックを作り出す。

〔例〕

次のコーリング・シーケンスによるムービング・コントロール・ブロックは，図4.15のような動きを制御する。

```
DIMENSION  MVDIM(62)                              ①
CALL  MVGBLK(MVDIM,0,32,0,1,0,-16                 ②
                              1,2,4,0,            ③
                              2,0,3,0)            ④
        ⋮
CALL  BGNETY(▼FR▼,▼MVET▼)                         ⑤
CALL  GSTRNG(1,▼*▼)                               ⑥
CALL  ENDETY(MVDIM)                               ⑦
        ⋮
```

説　明

①　：ムービング・コントロール・ブロックの配列宣言

②　：ムービング・コントロール・ブロックの作成

MVDIM　：コントロール・ブロック名

0　：ループ・レベル0

３２ ： ループ・レベル０の繰り返し回数

０ ： ループ時の△x，△yは変化させず。

１ ： ループ・レベル０に含まれる△x，△y組数１

０ ： △xの移動量０

－１６ ： △yの移動量－１６

③ ： ②の続き

１ ： ループ・レベル１

２ ： ループ・レベル１の繰り返し回数

４ ： ループ時に△xと△y入れ替え

０ ： ループ・レベル１に含まれる△x，△y組数０

④ ： ②③の続き

２ ： ループ・レベル２

０ ： ループ・レベル２の繰り返し回数（０は∞）

３ ： ループ時に△x，△yの方向反転

０ ： ループ・レベル２に含まれる△x，△y組数０

⑤ ： エンティティ開始宣言

▼ＦＲ▼ ： フレーム名

▼ＭＶＥＴ▼ ： エンティティ名

⑥ ： ＊記号をエレメントとして組み込む。

⑦ ： エンティティの終了宣言

ＭＶＤＩＭ ： このエンティティが，コントロール，ブロック，ＭＶＤＩＭによって制御される。ムービング・エンティティであることを示す。

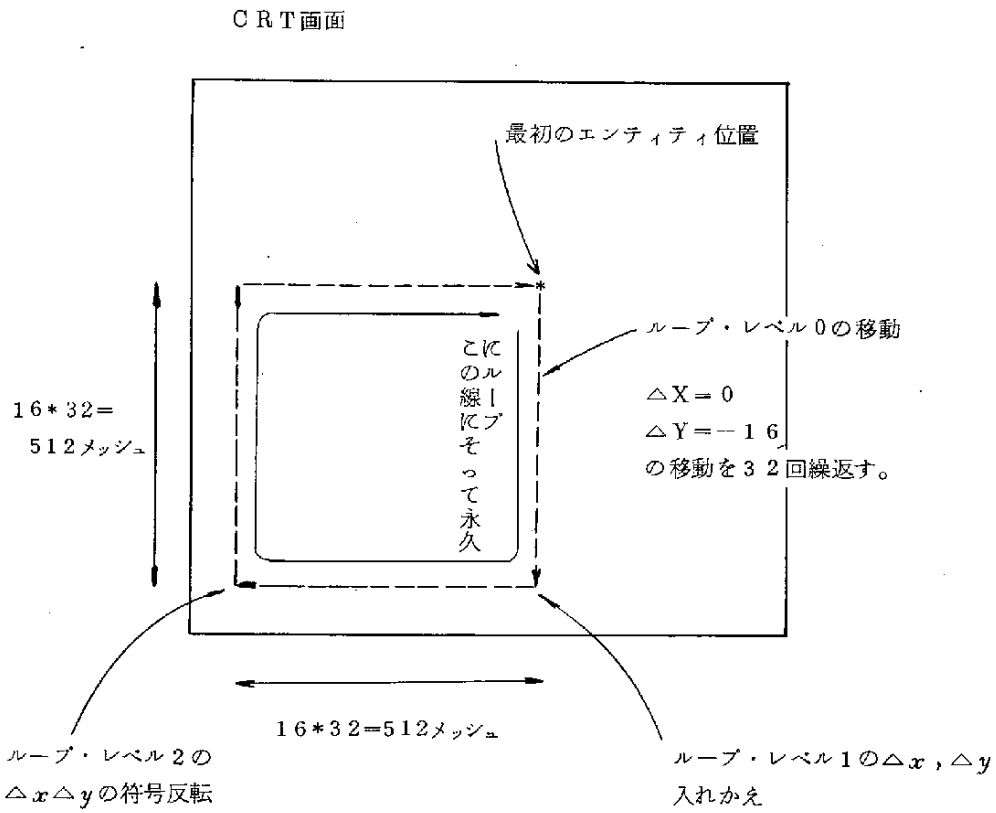


図4.15　ムービング・エンティティの例

14. CPYETY （Copy Entity）

〔機能〕

指定されたエンティティと全く同じ内容を持ったエンティティを指定されたフレームに作り出す。

この時コピーの対象となるエンティティはすでに作成されていなくてはならない。

〔呼び出し形式〕

CALL CPYETY(▼ID1▼,▼ID2▼,▼ID3▼,▼ID4▼)

〔アーギュメント〕

ID1　：　コピーの対象となるエンティティの属するフレームを指定する。

ＩＤ２　：　コピーの対象となるエンティティを指定する。

ＩＤ３　：　コピーしたエンティティを容れるフレームを指定する。

ＩＤ４　：　コピーしたエンティティに与えるエンティティＩＤを記す。

〔内部処理〕

指定されたフレーム領域から該当するエンティティを探し指定されたフレーム領域へ転送する。

(例)　フレームＩＤ▼Ｆ１▼に存在するエンティティと全く同じエンティティを別のフレーム▼Ｆ２▼に作りそれにエンティティＩＤ▼ＥＮＴＸ▼を与える。

```
CALL CPYETY(▼F1▼,▼ENT1▼,▼F2▼,▼ENTX▼)
```

15. GPOST (Set Beam Position)

〔機能〕

指示された点にビームをブランク移動する。

〔呼び出し形式〕

```
CALL GPOSIT(ITYPE,ARGX,ARGY)
```

〔アーギュメント〕

ＩＴＹＰＥ　：　表示ベクトルのモードを与える。

0　指定された値を絶対座標系の点とみなしてビームを位置ずける。

1　現在，ビームのある位置から指示された値だけビーム位置をずらす。

ＡＲＧＸ　：　Ｘまたは$\triangle x$を与える。

ＡＲＧＹ　：　Ｙまたは$\triangle y$を与える。

〔内部処理〕

開設中のエンティティにビームをブランク移動させるエレメント情報を作成する。

(例)　ビーム位置をＣＲＴ座標で(５１２，５１２)へ移す。

```
CALL GPOSIT(0,512,512)
```

ビーム位置を現在点からx方向へ50，Ｙ方向へ－７０だけ移動する。

```
CALL GPOSIT(1,50,-70)
```

(注)　$\triangle x$，$\triangle y$の値は，実際にはそのエレメントの属するフレームのフレーム・スケリング・ファクターをかけた値がとられる。

つまり，

$\triangle x' = \triangle x * SCALE$ となる。

(注) ブランク移動とは，ビームを表示することなくCRT上でビームの移動を行なうことを云う。

(注) CRT座標（又は絶対座標）とは，VIEWER上で図4.16の様に定義されている。

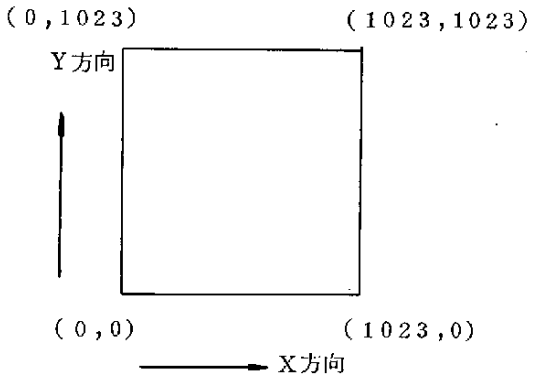


図4.16

16. GSTRNG (Draw Character String)

〔機能〕

現在，ビームのある位置から指示された文字列を作り出す。

〔呼び出し形式〕

CALL GSTRNG (IS,N,▼STRING▼)

〔アーギュメント〕

IS : 大文字，小文字の区別を与える。

0 小文字

1 大文字

N : 文字数を与える。

▼STRING▼ : 文字列を与える。

〔内部処理〕

開設中のエンティティ内に文字列を発生させるエレメント情報を作る。

(例) 点(50,50)から BRAVO! と書く

CALL GPOSIT(0,50,50)

CALL GSTRNG(1 , 6 , ▼BRAVO!▼)

(注) 文字のＳＩＺＥは大文字，小文字の２種類に限られている。

(注) フレーム・スケーリング・ファクターの作用は受け無い。

(注) 文字の書き出し点，書き終りの点は図4.17のように決められている。

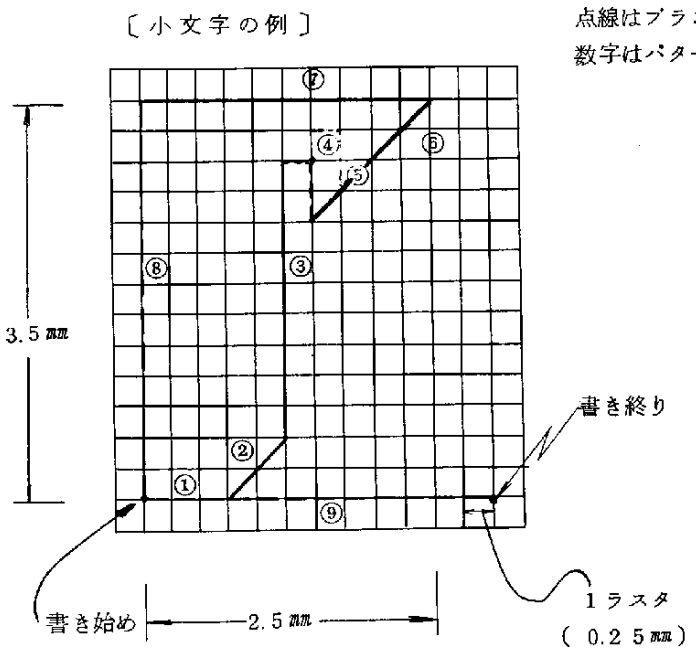


図4.17

17. GSUBET (Call Subroutine Entity)

〔機納〕

指定されたエンティティＩＤを持つサブルーティン・エンティティを呼び出す。

〔呼び出し形式〕

CALL GSUBET(▼ID▼)

〔アーギュメント〕

▼ID▼ ： 呼び出すべきサブルーティン・エンティティのエンティティＩＤを記す。

〔内部処理〕

開設中のエンティティ内にサブルーティン・エンティティ呼び出しのためのエレメント情報を作成する。

(例)　エンティティＩＤ▼ＳＵＢ１▼で登録されたサブルーティン・エンティティを呼び出す。

ＣＡＬＬ　ＧＳＵＢＥＴ(▼ＳＵＢ１▼)

18. ＧＶＥＣＴＲ　(Draw Vector)

〔機能〕

与れられた点列をベクトルで表示する。

〔呼び出し形成〕

ＣＡＬＬ　ＧＶＥＣＴＲ(ＩＭＯＤＥ,Ｎ,ＡＲＸ,ＡＲＹ)

〔アーギュメント〕

ＩＭＯＤＥ　：　ベクトルのモードを与える。

0　アブソリュート・モード与えられた点列を絶対座標系の点とみなして表示する。

1　リラティブ・モード1(原点固定)
現在ビームがある位置を原点とみなし与えられた点列をその点からの変位として表示する。

2　リラティブ・モード2(原点移動)
一つ前の点からの変位として与えられた点を表示する。

Ｎ　：　点列の個数を記す。

ＡＲＸ　：　Ｘまたは$\triangle x$の格納されている配列名，変数名を記す。

ＡＲＹ　：　Ｙまたは$\triangle y$の格納されている配列名，変数名を記す。

〔内部処理〕

開設中のエンティティにベクトルを発生させるエレメント情報を作り出す。

19. ＧＮＵＭＢＲ　(Draw Number)

〔機能〕

指定された変数を数値表示する。

〔呼び出し形式〕

ＣＡＬＬ　ＧＮＵＭＢＲ(ＩＰ,ＡＲ〔,Ｍ〔,Ｊ〔,Ｋ〕〕)

〔アーギュメント〕

IP ： 表示する時の形式を記述する。

0 Iタイプ

1 Fタイプ

2 Eタイプ

AR ： 対象とする数値が格納されている変数名を書く。

M ： 表示文字のサイズを指定する。

0 小文字

1 大文字

デフォルト値は0とする。

J ： 表示する桁数を記す。（1≦J≦12）

デフォルト値はJ＝12とする。

K ： 少数点以下の桁数を指定する。（0≦K≦6）

デフォルト値はEタイプの時は6，I，Fタイプの時は0とする。

〔内部処理〕

数値を表示パターンに変換する以外は，GSTRNGと同様な扱いをする。

（例） 変数Xの値をFタイプで下のように大文字で表示する。

nnn・nnnnn

CALL GNUMBR(1,X,1,9,5)

20. GXAXIS (Draw X-Axis)
GYAXIS (Draw Y-Axis)

〔機能〕

それぞれX軸，Y軸を表示する。

〔呼び出し形式〕

CALL GXAXIS(X0,Y0,N,DX, V0, DV, IP)

〔アーギュメント〕

X0,Y0 ： X軸の左端の座標を与える。

N ： 目盛によって区切られてる区間の数

DX ： 目盛の間隔

V0 ： X軸の左端に表示する数値

ＤＶ　：　表示する数値の刻み。

ＩＰ　：　数値を表示する時の形式を指する。

ＩＰ＝２　数値を軸の上に整数として表示

ＩＰ＝１　数値を軸の上に実数のまま表示

ＩＰ＝０　数値を表示しない。

ＩＰ＝－１　数値を軸の下に実数のまま表示

ＩＰ＝－２　数値を軸の下に整数として表示

２１．ＧＣＲＶＦＴ（Ｄｒａｗ ｃｕｒｖｅ ｗｉｔｈ ｆｉｔｔｉｎｇ）

〔機能〕

与えられた点列をなめらかな曲線で結ぶ。

〔呼び出し形式〕

ＣＡＬＬ　ＧＣＲＶＦＴ（ＩＭＯＤＥ，Ｍ，ＡＲＸ，ＡＲＹ）

〔アーギュメント〕

ＩＭＯＤＥ　：　点列座標のモードを示す。

０　アブソリュート・モード

１　リラティブ・モード１（原点固定）

２　リラティブ・モード２（原点移動）

（ＧＶＥＣＴＲ　参照）

Ｍ　：　点列の個数を示す。

ＡＲＸ　：　ＸまたはΔxの格納されている配列名

ＡＲＹ　：　ＹまたはΔyの格納されている配列名

〔内部処理〕

開説中のエンティティ中にＧＣＨカーブ・フィット・ルーチンに渡すための諸パラメータをエレメントとして生成する。

（パラメータについては3.3 ＧＣＨ参照）

第5章　システムオペレーション

ＣＧＯＳを利用して，グラフィック・プログラムによる各種の図形を，Ｈ－８８１１グラフィックＣＲＴ上に表示したり，インタラクティブなマン・マシン・コミュニケーションをおこなおうとするユーザは，ＣＧＯＳのシステム・オペレーションを知る必要がある。そのためには，Ｈ－８４００ＤＯＳオペレータガイド，Ｈ－８８１１ディスクベースオペレータガイド等を参照する事が必要となる。ここでは一応Ｈ－８４００ＤＯＳオペレーションが可能なユーザを対象とした，ＣＧＯＳのシステム・オペレーションについて記述する。

5.1　ＣＧＯＳシステムオペレーション

ＣＧＯＳのシステムオペレーションは，次の様な特徴を持つ。

(1) ＣＧＯＳコントロール・システムのもとでは，同時に５個迄のマルチ・プログラミング・オペレーションが可能である。

(2) プログラムのロードは，Ｈ－８４００側のコンソール・タイプライタ，Ｈ－８８１１側のグラフィックＣＲＴコンソールの両者から可能であるが，Ｈ－８４００側のコンソール・タイプライタから呼び出されたプログラムは，ＧＪＭＯＮの管理外に置かれる。

(3) ユーザのグラフィック・プログラムは，Ｈ－８８１３グラフィックＣＲＴコンソールから呼び出す必要がある。つまり，ＧＪＭＯＮの管理下でなければ，グラフィック・プログラムは働けない。

(4) グラフィック・プログラムは，同時に４つ迄オペレーション可能であるが，一時にＨ－８８１３グラフィックＣＲＴコンソールを介した入出力の権限を持つのは，ただ１つである。

(5) ユーザ側で一般に使用される，ＦＯＲＴＲＡＮ，ＣＯＢＯＬ等の汎用言語や，アセンブラに容易にリンケージ・バインド可能な，ＧＩＡＳＰ，ＧＳＰ等のサブルーチン・パッケージを提供して，グラフィック・プログラムを簡単に作成する事を可能とした。

5.2　コントロール・カード

ユーザの・グラフィック・プログラムにＧＳＰ，ＧＩＡＳＰ，その他のインタラクティブ

・グラフィック・プログラミング・サポートのモジュールをリンケージ・バインドするには，DOS・SYSOMLの，インプリシット・コール機能により，特別のコントロール・カードは，必要としない。また，FORTRANのエグゼクティブ・ランで，データセットを必要とする時，GRDSET（DATADで作られた）を利用できるが，このGRDSETは，テープ上に作成してあるので，リンケージのインクルード・カードを用いる。

以下に，FORTRANを例にとって，ロード・モジュールを作成するための，コントロール・カードを示す。

なおモニター・ランで，コンパイル・アンド・ゴーの場合には，H－8813グラフィックCRTコンソールより，モニターを，ロードしなければならない。エグゼクティブ・ランの場合は，ロード・モジュール（ユーザ・プログラムとGSP，GIASPのリンケージ・バインドされたもの）をH－8813グラフィックCRTコンソールよりロードすれば充分である。

〔1〕

エグゼクティブ・ランで，データセットを使用しないユーザ・プログラムのコントロール・カードと，プログラム起動方法

(Ⅰ) コントロール・カード

```
// STARTM
// ASSGN SYSLST,L0
// VDC SYSOML,,DOS・SYSOML,000001
// VDC SYSUT1,,DOS・SYSUT1,000002
// VDC SYSUT2,,DOS・SYSUT2,000002  **
// VDC SYSUT3,,DOS・SYSUT3,000002
// FORTRN
// LNKEDT
// ENDMON
```

(Ⅱ) 起動方法

H－8813グラフィックCRTコンソールからSYSUT2のロード・モジュールを呼び出す。

<例> ¥E LOD PROG1,A1U

** SYSUT2はテープにアサインしてもよい。

〔2〕

エグゼクティブ・ランで，データセットを使用するユーザ・プログラムのコントロール・カードと，プログラムの起動方法

(i) コントロール・カード

前記〔１〕のパラメータに，リンケージ・パラメータを加える（ＬＮＫＥＤＴの次）

＿ＰＲＯＧプログラム名

＿ＩＮＣＬＵＤＥ　ＳＹＳＵＴ１

＿ＩＮＣＬＵＤＥ　ＳＹＳＵＴ４（ＧＲＤＳＥＴ）　　＊＊＊

(ii) 起動方法

〔1〕と同様

＊＊＊ ＳＹＳＵＴ４はテープにアサイン。ＣＧＯＳで提供する。

〔3〕

モニター・ランでコンパイル・アンド・ゴーの場合のコントロール・カードと，プログラムの起動方法

(i) コントロール・カード

前記〔1〕のパラメータに，エグゼク・パラメータを加える（ＬＮＫＥＤＴの次）

／／ ＥＸＥＣ

(ii) 起動方法

Ｈ－８８１３グラフィックＣＲＴコンソールから，モニターを呼び出す。

＜例＞　¥Ｅ　ＬＯＤ　ＭＯＮ，，，Ｒ０，，１００００

5.3 操作手順

グラフィック・プログラムはＣＧＯＳのコントロール・システムの下で実行されなければならない。そのためにユーザのグラフィック・プログラムは，Ｈ－８８１３グラフィックＣＲＴコンソールから呼び出す必要がある。もしＨ－８４００側のコンソール・タイプライタより呼び出すと，プログラムはＧＪＭＯＮの管理外に置かれる。

ＣＧＯＳの下では，プロセッシング・プログラムは，グラフィック・プログラムと，ノングラフィック・プログラムに分けられ，グラフィック・プログラムは，Ｈ－８８１３グラフィックＣＲＴコンソールを介した入出力を行ない得るが，ノングラフィック・プログラムは，これを行なえない。

次にＣＧＯＳ操作手順のブロック・チャートを示す〔図5.2〕

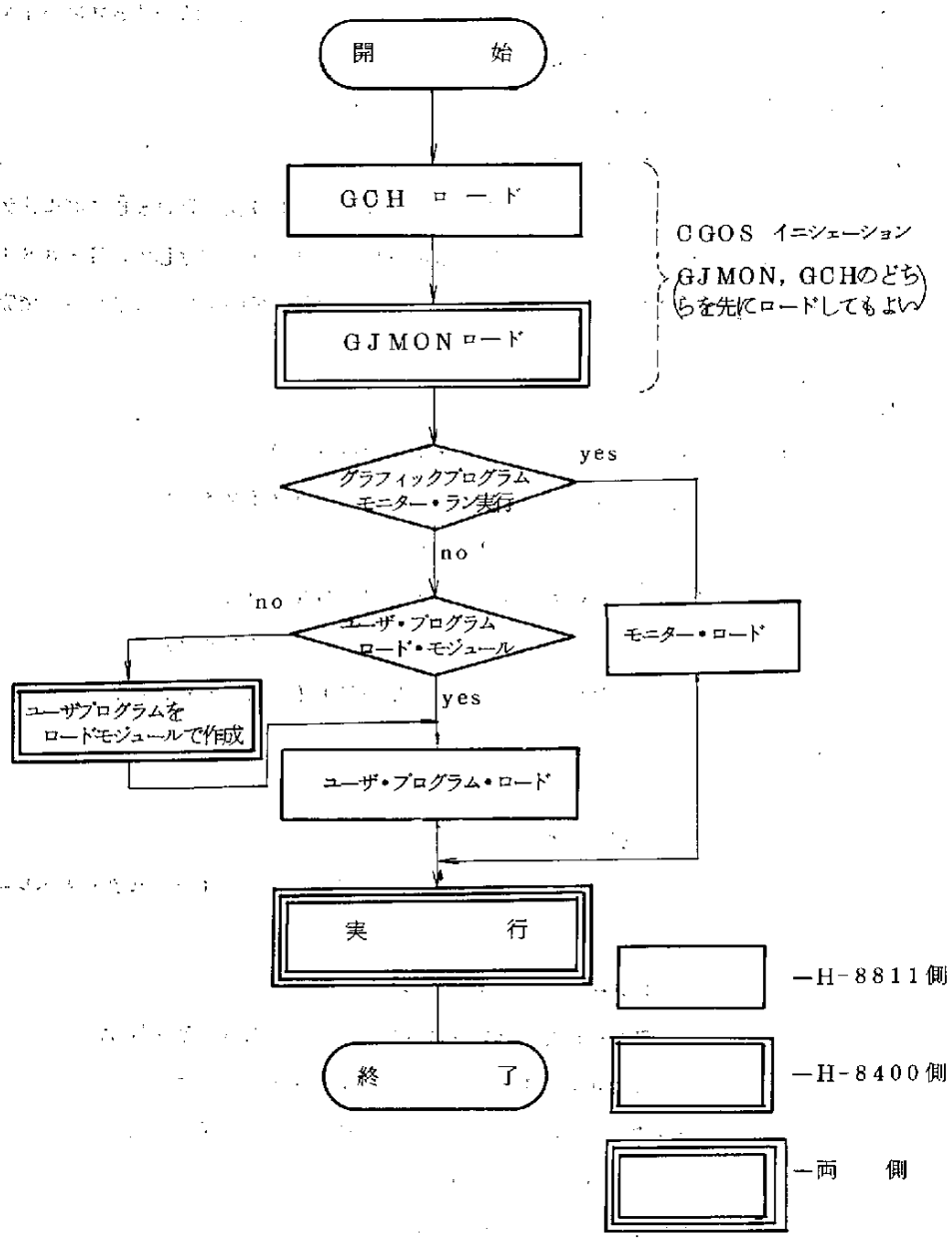


図5.1　ＣＧＯＳ操作手順

なお，グラフィック・プログラムの実行には，その前にあらかじめ，ＧＳＰ，ＧＩＡＳＰ，その他のモジュールと，リンケージバインドされた，グラフィック・プログラムの，ロード・モジュールを作成しておく事が望ましい。ただし，モニター・ランにおけるコンパイル・アンド・ゴーの場合には，この限りでない。

次に各操作項目につき説明する。

5.3.1　ＣＧＯＳシステム・イニシェーション

ＣＧＯＳのシステム・イニシェーションは，Ｈ－８４００側にＤＯＳⅢエグゼクティブ・イニシャル・ローディング，続いてＧＪＭＯＮのロード，並行して，Ｈ－８８１１側にＧＯＳⅢエグゼクティブ・イニシャルローディング，続いてＧＣＨのロードで完了する。

〔1〕

Ｈ－８４００側のロード及びコンソール・メッセージの説明

Ｅ　ＬＯＤ　ＧＪＭＯＮ　　（タイプイン）

Ｖ　ＧＪＭＯＮ　０２Ｌ６　２２３４０９

Ｙ　ＧＪＭＯＮ　０１１ＧＡ　６　ＡＳＳＧＮ　ＤＸＣＤＸＣ

Ｙ　Ｘ０　　（タイプイン）

６　ＧＪＭＯＮ　ＧＲＰＨＩＣ　ＪＯＢ　ＭＯＮＩＴＯＲ　ＲＥＡＤＹ

以上でＧＪＭＯＮがロードされる。

〔2〕

Ｈ－８８１１側のローディング

Ａ　コントロール・プログラムのイニシャライズ（詳細は８８１１ディスク・オペレータガイド参照）

≪ Ｈ－８８１１コンソール・パネル操作≫

(i)　ＧＥＮＲＥＳＥＴを押してＩＣレジスタを選択し，ＲＥＡＤＹを押す。

(ii)　インディケータに $(0201)_{16}$ をセットしＴＲＡＮＳを押す。

(iii)　ＳＴＡＲＴを押すと，イニシャライズされ割込み待ちの状態になる。

Ｂ　コミュニケーション・ハンドラの呼び出し。

≪ Ｈ－８８１１リモート・ターミナル操作≫

(i)　ＲＥＱ１を押すと，図5.3のような画面が，ＣＲＴ上に表示される。これは，コミュニケーション・ハンドラの第１画面である。

≪ H－８８１３ＣＲＴの操作 ≫

(ii) ＊ＳＰＥＣＩＦＹ　ＰＲＯＧＲＡＭをピックする。（ライトペンによる）

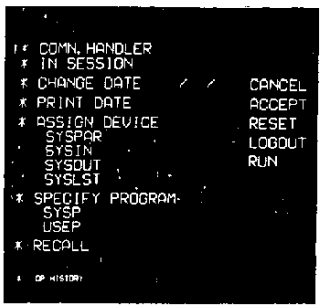
(iii) ＵＳＥＰをピックする。

(iv) ＡＣＣＥＰＴをピックする。

(V) ＲＵＮをピックすると，図5.4のような画面が，ＣＲＴ上に表示される。これは，コミュニケーション・ハンドラの第２画面である。

```
＊＊ COMN HANDLER
 ＊ IN SESSION
 ＊ CHANGE DATE   ／ ／ CANCEL
 ＊ PRINT DATE          ACCEPT③      B(iv)
 ＊ ASSIN DEVICE        RESET
    SYSPAR              LOGOUT
    SYSIN               RUN④         B(V)
    SYSOUT
    SYSLST
 ＊ SPECIFY PROGRAM①                  B(ii)
    SYSP
    USEP②                            B(iii)
 ＊ RECALL
 ＊ OP HISTORY
```

図5.3 コミュニケーション・ハンドラ　第１画面



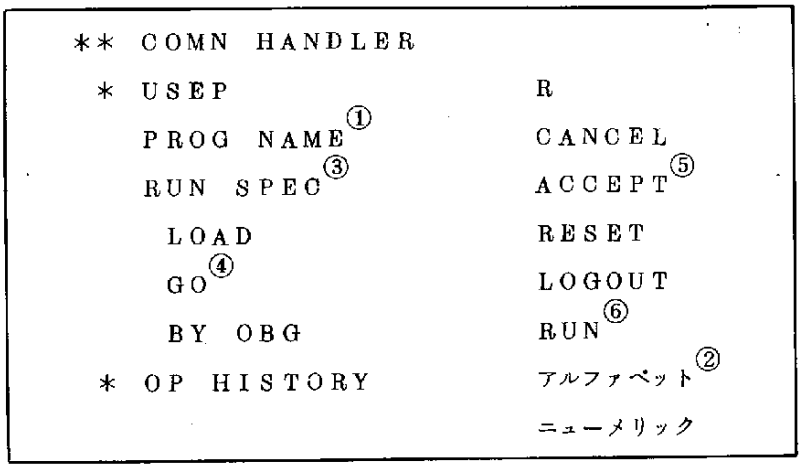


図5.4 コミュニケーション・ハンドラ　第2画面

C　GCHのロード

≪ H－8813CRTの操作 ≫

(i) PROGNAMEをピックすると，RUNの下に英数字が表示される。

(ii) プログラム名GCHを英数字よりピックする。

(iii) RUNSPECをピックする。

(iv) GOをピックする。

(v) ACCEPTをピックする。

(vi) RUNをピックすると，GCHがロードされ，メッセージ・インディケータが画面に表示される。

以上のA，B，C操作で，GCHがロードされない時には，H－8811コントロール・プログラムがこわされているので，イニシャル・ローダのセット，ディスク・システムローダの読み込みを行なった後，B，Cの操作を行なわなければならない。（詳しくは，H－8811ディスク・オペレータ・ガイド参照）

また，H－8811に付属するディスクが駆動可能になっていなかったり，システムディスクがかかってなかったりすると，コミュニケーション・ハンドラの呼び出しができな

いので，注意を要する。

D　GCHのリロード

GCHのリロードは，H－８８１１のコントロール・プログラムが壊れていなくて，何らかの理由により，GCHが終了，停止した時に，次の操作を行なう。

(i)　B(i)の操作。

(ii)　RECALLをピックする。

(iii)　ACCEPTをピックする。

(iv)　RUNをピックすると，GCHがリロードされ，メッセージ・インディケータが画面に表示される。

以上の操作でリロードされない時には，A～Cの操作を行なう。

5.3.2　CGOSコンソール・リクエスト

CGOSのコンソール・リクエストは，H－８４００付属コンソール・タイプライタ，H－８８１３グラフィックCRTコンソールのどちら側からでも可能であるが，若干のコマンド・リクエストと，グラフィック・プログラムにメッセージ・データを渡す場合には，H－８８１３グラフィックCRTコンソールによらなければならない。また，H－８８１３グラフィックCRTコンソールからのDOS・コンソール・リクエストは，ほぼすべてが可能であるが，GJMON自身に対するコントロールは，不可能である。

H－８８１３グラフィックCRTコンソール・リクエストの方法は，メッセージのライトペン・ピッキングによって行なわれる。このため，CGOSでは，メッセージ・インジケータを用意する。メッセージ・インジケータについては後述する。

CGOSコンソール・リクエストは，次の三種類に分けられる。

(i)　システム・コンソール・リクエスト

(ii)　コマンド・リクエスト

(iii)　プログラム・リクエスト

以下にメッセージ・インジケータ，コンソール・リクエストの詳細について記述する。

〔1〕

メッセージ・インジケータ

メッセージ・インジケータは図5.5の様式であり，GCHのロードによって，CRT画面上に表われる。また，H－８８１１システムには問い合せタイプライタがあるが，これはH－８８１３グラフィックCRTコンソールそのものに付属している訳ではないので，

オペレータが，インクァイアリーを行なうのに不便である。従って，H－８８１３グラフィックＣＲＴコンソールに，メッセージ・インジケータを表示し，これに対するライトペン・ピッキングによってメッセージを作り出し，ＣＧＯＳコントロール・システムとインタラクションをとる。

メッセージ・インジケータ

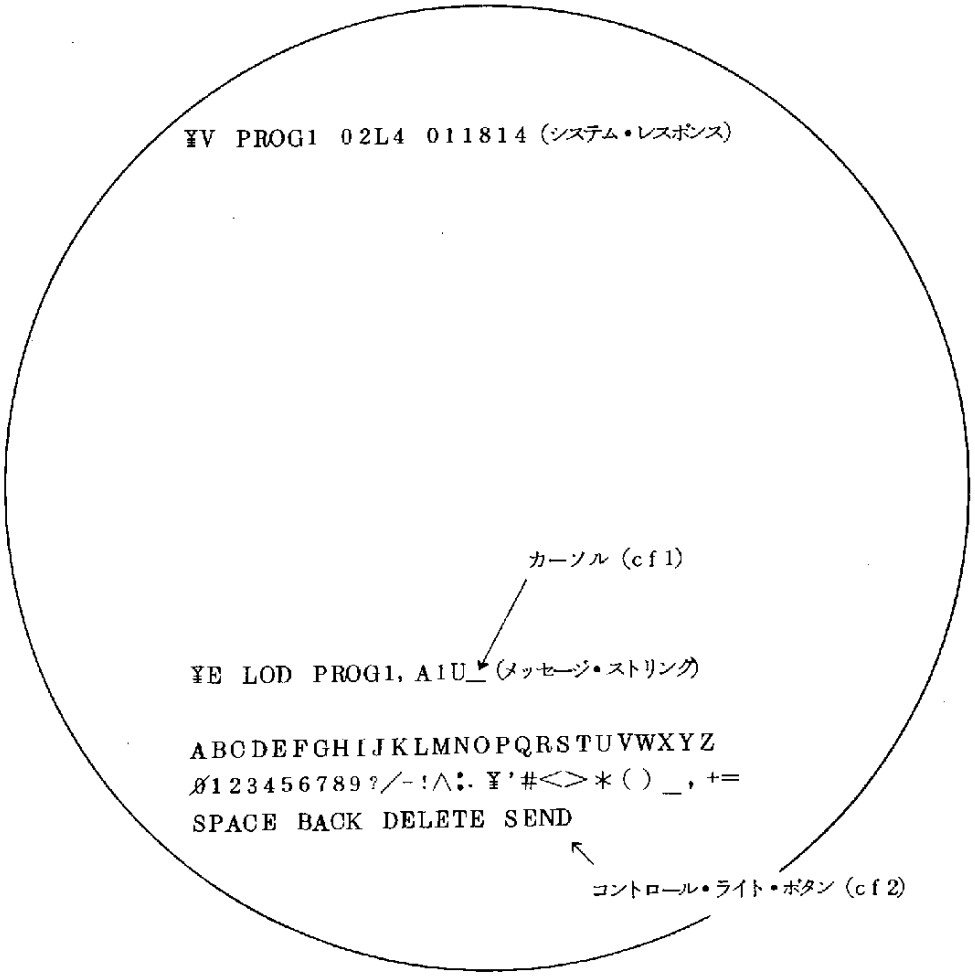


図 5.5

cf 1. ライトペンでピックされた文字は，メッセージ・インジケータのカーソル位置に１文字づつ表われる。

cf 2. ライト・ボタンは次の意味を持つ。

ＳＰＡＣＥ：１文字分スペースをあける。

ＢＡＣＫ　：１文字分消す。

ＤＥＬＥＴＥ：メッセージ列を消す。

ＳＥＮＤ　：メッセージ列をＣＧＯＳコントロール・システムに送る。

cf 3. メッセージ・インジケータと，メッセージ・ストリングは，ファンクション・キー０によって表示，消去ができる。

〔2〕

システム・コンソール・リクエスト

従来Ｈ－８４００ＤＯＳで行なっていた，コンソール・リクエストを，ＣＧＯＳでは，システム・コンソール・リクエストと定める。このシステム・コンソール・リクエストは，Ｈ－８４００ＤＯＳコンソール・リクエストの頭に▼¥▼の記号をつける。

ユーザのグラフィック・プログラムをロードする時は，システム・コンソール・リクエストによらなければならない。

例

```
¥E  LOD  PROG1, A1U
¥E  LOD  MON, , , R0, , 100000
¥Y  L0
¥U  L1
¥X  0
¥E  DEM  000001
```

〔3〕

グラフィックオペレーション・コマンド

1. コマンド・リクエスト

Ｈ－８８１３グラフィックＣＲＴコンソール・オペレータは，ＣＧＯＳのグラフィック・オペレーションのコマンド・リクエストの機能により，次のサポートを受けることができる。

(i) Alter loging device

システム・メッセージ出力装置の交替。（H－８８１３ＣＲＴとH－８８１１付属タイプライタの交替）

(ii) Ask current status of program

プログラムの進行現状の問い合わせ。（このプログラムに関する，最新のシステム・メッセージを表示する）

(iii) Quit to graphic program

グラフィック・プログラムの進行の一時停止と，H－８８１３ＣＲＴ入出力装置のあけわたし。

(iv) Start program

グラフィック・プログラムの進行の開始，再開と，ＣＲＴ入出力権を与える。

2. コマンド・リクエストの型式

グラフィック・オペレーション・コマンドの一般形は，¥Ｇ△コマンド・コード△〔｛プログラム名^{n}｝〕である。このコマンドの入力後，ライト・ボタンのＳＥＮＤをピックする必要がある。

コマンド・コード

(i) ＡＬＴＲ

(ii) ＡＳＫ

(iii) ＱＵＩＴ

(iv) ＳＴＲＴ

ｎ：プログラム・ナンバー

¥Ｇ：コマンド・リクエストである事を示す。

cf 1. ＡＬＴＲとＱＵＩＴには，プログラム名またはｎをつけてはならない。

cf 2. ＡＳＫコマンドのプログラム進行現状は，ＧＪＭＯＮ中のプログラム制御表によって確保される。

[4]

プログラム・リクエスト

第４章でも述べたようにH－８８１３グラフィックＣＲＴコンソールから，現在進行中のグラフィック・プログラムに，メッセージ・データを与える時に使用され，次の二種類がある。

(1) 非同期な割込み転送

与えたいメッセージを，メッセージ・インジケータよりピックし，メッセージ・ストリングを作り出し，ライト・ボタンのＳＥＮＤをピックする。この時のストリングは，画面上では図5.5に示すように，　画面上では図5.5に示すように，画面の下部に表示される。転送を行なった際データを受け取るべきプログラムがなかったり，グラフィックプログラムが何らかの処理中で，データを受け取れない時は，エラ・メッセージが帰ってくる。

(2)　同期をとった転送

グラフィック・プログラムから意識的に，グラフィックＣＲＴコンソールの指示を求め，その指示に基づいて次の処理を決定しようとする場合には，同期をとった割込みが可能であり，この機能は，ユーザのグラフィック・プログラムからのデータの入力命令による。

データの種類としては，キャラクタ・メッセージと，数値データを許している。キャラクタ・メッセージは，メッセージ・インジケータよりピックして，メッセージ・ストリングを作り出し，数値データの場合は，以下に述べるフォーマットに従って，ストリングを作り出す。その後ライト・ボタンのＳＥＮＤをピックする事によって，データを転送する。

なお，ユーザがピックしたストリングの画面における表示位置は，ユーザ・プログラムからの，ポジショニング指定による。

数値データのフォーマット

(i)　整数型の数値データ

〔±〕ｎｎｎ……

(ii)　実数型の数値データ

〔±〕〔ｎｎｎ……ｎ〕．〔ｍｍｍ……ｍ〕

または

〔±〕〔ｎｎｎ〕．ｍｍｍ……ｍＥ〔±〕$\ell\ell$

但し，ｎ，ｍ，ℓは数値のディスプレイコード。

5.3.3　ファンクション・キー・オペレーション

ユーザのグラフィック・プログラムは，ファンクション・キーの割込みコードを持った割込処理ルーチンをＧＩＡＳＰを使用して登録しておけば，ファンクション・キーの割込みをかける事により，メッセージを自分のプログラムのデータとして受け取り，処理する事ができる。ファンクション・キー・ナンバーと，割込みコードの対応は次の通

りである。

割込みコード	キー・ナンバー
0	11～31
1	4（ピッキング）
3	5（トラッキング）

なお，H－8813に付加されている，ファンクション・キーは，CGOSの下では次のように定義されていて，CGOSのシステム・オペレーションに使用される。

〔表5.1〕

表5.1　ファンクション・キー・ナンバ対応表

キーナンバ	ファンクション	説　　明
0	メッセージ・インジケータ・オン・オフ	キーナンバ・0，1，2は，H－8811プロセッサ内だけのトラフィックコントロール
1	トラッキングマーク・　オン・オフ	
2	ビューアー　オン・オフ	ユーザのグラフィック・プログラムが作り出した表示図形をCRT上に出したり，消したりする。
3	クィット	現在進行中のグラフィック・プログラムに割込みをかけて，一時停止させる。
4	ライトペン・ピッキング・転送	ライトペン・ピッキング時の情報を転送する。
5	トラッキング・マーク位置転送	トラッキング・マークの位置情報を転送する。
6	X－Yプロッタへのハードコピー	CRT上の情報を，X－Yプロッタにコピーする。
7	プリンキングのリセット	ピッキング時のプリンキングをリセットして，もとにもどす。
8	シザリング処理	エッジ・シザリング処理を行なう事を指令する。
9 10	予　　備	
11 ～ 31	無定義（ユーザ・使用可能）	ユーザが使用する場合，ファンクションキーナンバと，オーバレイ・コードが転送される。

（cf）　キーナンバ，0～10はシステム予約である。

5.4 メッセージの説明

システム・レスポンスの欄に，表示されるメッセージには，次の様なものがある。

(i) ¥READY

割込み受付け可能な状態を示す。

(ii) ¥WAIT

システムのレスポンスを待っている状態を示す。

(iii) ¥MIS OP

コンソール・リクエストのミスオペレーションを示す。

(iv) ¥BUSY NOW

多重割込み中で，待ち合せキューができている状態で，この場合は受けつけられるまで再度試みる必要がある。

(v) ¥ERR Ennn

GCHの出すエラー・メッセージで，nnnはエラーの種類によって異なる。詳細は，エラー・メッセージ表を参照〔表5.2〕。

(vi) H-8400DOSシステム・レスポンスの頭に▼¥▼のついたもので，システム・コンソール・リクエストのレスポンス。

(vii) コマンド・リクエストのレスポンス

グラフィック・オペレーション・コマンド・リクエストの種類によって異なる。詳細は3章5節に述べている。

表5.2　エラー・メッセージ表　（Ｅｎｎｎ）

メッセージ	ファンクション
＊Ｅ００１	フリースペース・テーブル・オーバーフロー
Ｅ００４	創成モードなのに，更新エンティティがきた。
＊Ｅ００５	ムービング・コントロール・テーブル・オーバーフロー
＊Ｅ００６	エンティティの登録テーブル・オーバーフロー
Ｅ００７	フレーム・フォーマット・エラー（エレメント数０より小）
Ｅ００８	ファンクション・ナンバーの指定にあやまりがある。(エレメントレベル)
Ｅ００９	フレーム・フォーマット・エラー（フレームレベル）
＊Ｅ０１０	コマンド・エリアがなくなった。
Ｅ０１１	フレーム・フォーマット・エラー（エンティティレベル）
＊Ｅ０１２	ペンディング・サブルーチンの数が，１５を越えた。
ユーザの使用に際してのエラー表示が出るのは，＊のものである。	

第Ⅱ部では，当センターが開発した，隠れ線消去のプログラムについて，その内容を紹介する。

この隠れ線処理の問題については３，４年ほど前から各国で種々の解決法が提唱されてはいるが，いづれも，一般性，計算時間などの点で，問題があり決定的なアルゴリズムはまだ確立されていない状態である。

昨年度，当センターの『図形表示用プログラムの基礎研究』〔４４－Ｓ－００４〕と題する報告書の中で，この隠れ線処理の問題について，基礎的な調査を行ったが，今年度は，その研究を発展させ，実用プログラムを開発した。この第２章ではGalimbertiの方法にもとづいたＸＹプロッタおよびディスプレイ用のプログラムについて説明し，第３章は，通常，数学および工学の分野で，よく用いられる２変数関数について，その立体的な姿を視覚的に捕えられるように，隠れ線を消去して表示するプログラムにつき説明してある。

第II部　アプリケーションプログラム　1

第１章　隠れ線処理

最近，グラフィック　ディスプレイの応用は従来の２次元的なものから，次第に３次元的なものへと広がりを見せて来た。

ディスプレイ画面はしょせん２次元平面でしかないから，３次元の立体図形を表示する場合にも何等かの手段で２次元画面に変換して表示しなければならない。

この場合，それができるだけ実物に近く，すなわちリアリスティック（ realistic ）に人間の眼に映ずるためには，何等かの処理が必要となる。

その方法として，次のようないくつかの手段がある。

1　投象変換

３次元における物体の各点をある投象面に投象し，２次元座標に変換する。方法としてはいろいろあるが，例えば $z=0$ 平面に投象する場合には，各点を同次座標で表わし，４×４の変換マトリックスをかけることによって簡単に求まる。単純な物体の場合は辺をなす点を結ぶことによって物体は投象された形を表わすが，複雑な物体はｗｉｒｅ ｆｒａｍｅ（針金細工のような形）になってしまい，形をよく判断できないこともおこる。

2　立体感をもたせるために輝度を調整する

３次元物体の投象，透視を行う際に応用する。遠くにある ｌｉｎｅ より，手前にある ｌｉｎｅ を明かるくすることによって立体的に見せる。単純な物体には効果的である。

3　回　　転

物体をリアルタイムで回転させる。線の動きの変化によって，見ている人間は３次元的に認識できる。例えば，六角形を底から見て，果して六角柱なのか，六角すいなのか判別できない。そこで回転することによって初めて形を知ることができる。

4　ステレオスコープ

人間の両眼で見るように，物体の二カ所からの光景を合成する。方法のひとつとして図１で

示すように，鏡によって二つの光景を一つのスクリーンに分けて写しだす。両目の光景は重ってしまうため，左目での光景は反射させて別の位置に表わす。

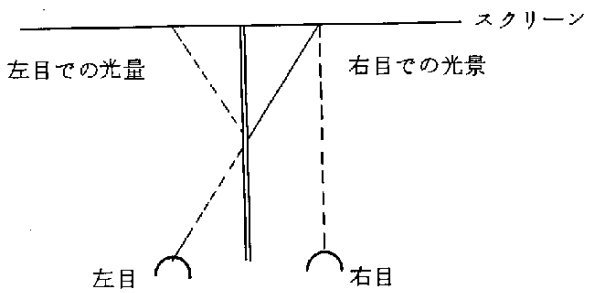


図　1

そのほかに，Sutherland colorwheelのように同期シャッタア装置を使う方法や，Sutherland headmounted displayのように2つのスクリーンを用いる方法がある。

5　隠れ線消去

前にも述べたように複雑な物体の場合はその物体を判別するために，lineの輝度に変化をもたせたり，回転させてもlineの数が多いので効果は少ない。そこで見えない線を消すことによってその物体をより立体的に見せようとするのが隠れ線消去である。

以上のような方法がこれまで各所で研究されてきているが，まだ欠点をかかえている。輝度に変化を持たせる方法は簡単な方法ではあるが，単純な物体でないと効果的でない。又回転する方法は反復による計算量が莫大なものとなる。ステレオスコープは非常に効果が高いが特別な装置を用意するか，そうでなければ情報量が2倍になるという欠点をもっている。

隠れ線消去の方法は，実際に人間が見た場合と同じ画像を作りだすことができるがこの方法も非常に計算時間がかかるという欠点がある。そこで計算時間を短縮しようと多くの努力と種々な工夫がなされてきた。

ここではその中でRobertの隠れ線消去の方法を説明する。

〔Robertの方法〕

この方法では，平面からなる凸物体を対象としている。そのために凹物体を取り扱う場合にはいくつかの凸物体に分けて考えなければならない。凸物体の場合には物体をつくっている面

だけで物体を定義できるが凹物体の場合には，各々の面の範囲に関する情報がさらに必要である。

ここで平面P_0の方程式を

$$ax + by + cz + d = 0 \qquad \text{とする}$$

係数は列ベクトルで

$$P_0 = \begin{Bmatrix} a \\ b \\ c \\ d \end{Bmatrix} \qquad \text{と表わす}$$

同次座標で点 $\overline{S}_0$ を表わすと

$$\overline{S}_0 = [\,WX \quad WY \quad WZ \quad W\,] \qquad \text{となる}$$

もし $\overline{S}_0 \; P_0 = 0$ であるならば点S_0が平面P_0上にあることを示している。$\overline{S}_0 \; P_0 \neq 0$ の時には$\overline{S}_0 P_0$の符号により平面のどちら側にあるかを判別できる。Robertは物体内の点と物体をつくる平面のベクトルとの積が正となるようにすべての面を選んだ。

点$\overline{S}_0$を投象した点$\overline{S}$を求めるには変換マトリックスMをかける。

$$\overline{S} = \overline{S}_0 M$$

同様に平面P_0を投象した面Pを求めるには

$$P = K P_0 \qquad \text{ここで} K = M^{-1}$$

このようにして物体をつくる平面は投象される。これらの面のベクトルは4×Nのマトリックスをつくる。このマトリックスをvolume matrixと呼ぶ。

$$V = \begin{pmatrix} a_1 & a_2 & \cdots & a_k & \cdots & a_n \\ b_1 & b_2 & \cdots & b_k & \cdots & b_n \\ c_1 & c_2 & \cdots & c_k & \cdots & c_n \\ d_1 & d_2 & \cdots & d_k & \cdots & d_n \end{pmatrix}$$

ここでa_k, b_k, c_k, d_k は投象した面の係数でNは物体を構成する面の数である。投象した点は投象した物体内にある。そこでもし原点が物体内にあるならば，各点のベクトルとvolume matrixとの積の係数が全て正である時，その点は物体の内側にあるといえる。

$(\overline{S}\,V \geq 0)$のとき $\overline{S}$ はVの内側にある。

かくれ線を消去する手順として3段階に分けられる。

1) screen boundaryを出る線を消去する。

2） back line を消去する。

3） 他の物体に対してテストする。

line とは物体をつくる二平面の交線であり両端点によりパラメータ表示をする。

$$\bar{V} = \bar{S} + t(\bar{r} - \bar{S}) \qquad 0 \leq t \leq 1$$

ここで $\bar{S}$, $\bar{r}$ は両端点である。どの平面と交わるかは volume matrix で表わされる。これらの点は平面の方程式を解くことによって得られる。例えば，平面 P_1, P_2, P_3 が交わる点は次のようなマトリックスQの逆行列 Q^{-1} の最後の列によって得られる。

$$Q = [P_1, P_2, P_3, I] \qquad I = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

1） クリッピング（Clipping）

スクリーンをはみだす line を消去する。投象面の負の側にある line も消去する。ここで投象面は $z = 0$ 平面とする。スクリーンの edge $x = -1$, $x = 1$, $y = -1$, $y = 1$ と，投象面 $z = 0$ を表わす volume matrix V_0 に対して各々の line を検査する。

$$V_0 = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

この volume matrix内にある line $\bar{V}$はみえる。部分的に見える時，その境点はベクトル $\bar{V}$ と V_0 の列ベクトルを順にかけあをせることによって求められる。line は $\bar{V} = S + t\bar{d}$ （$\bar{d} = \bar{r} - \bar{S}$, $0 \leq t \leq 1$）で表わせるから，$\bar{S}V_0 + t\bar{d}V_0 \geq 0$ を調べればよい。ここで $\bar{P} = \bar{S}V_0$, $\bar{q} = \bar{d}V_0$ とすると V_0 の j 番目の列ベクトルに於て $P_j + t q_i \geq 0$ となる。この式の係数 P_j, q_i から t の maximum と minimum がわかる。

もし $q_j > 0$ ならば $t \geq -P_j / q_j$ であり

もし $q_j < 0$ ならば $t \leq -P_j / q_j$ となる。

t の範囲によってスクリーン内に存在する line の部分を知ることができる。

2） Back line の消去

line がスクリーン内にあっても全体が見えなかったり，部分的に見えなかったりするという問題が残っている。そのひとつにその物体のある edge が自分自身の他の面で隠されて

しまう場合がある。このようなlineをback lineと呼ぶ。

edgeが見えるか否かを判定するのに簡単な方法がある。edgeをなす二平面，あるいはそのうちの一平面が視点の方に表側を見せていれば，そのedgeは見える。平面が視点に表側を見せているということを判別するには，面の方程式のzの係数を調べればよい。zの係数が正で零でなければ表側を視点に向けている。

3) 他の物体に対してのテスト

2)でback lineを消去した後でもまだ残っているline はもはや自分自身の物体の一部で隠されることはないが他の物体によってlineが隠される可能性は残っている。それをテストするためにまず，line上のpointを考える。これらの点はlineの方程式でtの値を変化させることによって表わせる。

$$\bar{V} = \bar{S} + t\,\bar{d}$$

このような点に対して，zの負の方向にlineをひく。即ち投象面方向にひく。もしこのlineが他の物体を通らずに，投象面まで続いているならば，その点は見える。逆に物体を横切っているならば，その物体によって点は隠される。

line上の一点$\bar{V} = \bar{S} + t\,\bar{d}$から$-z$の方向にひいたlineはパラメータ表示により次のように表わせる。

$$\bar{U} = \bar{S} + t\,\bar{d} + \alpha\,\bar{z} \qquad \alpha \geq 0$$

ここで$\bar{z}$は〔0，0，−1，0〕なるベクトルである。点はline上にあるから$0 \leq t \leq 1$の範囲にある。このlineの方程式とvolume matrixを順次かけあわせていき，すべての係数が正であれば，そのlineは物体を通りぬけている。

$$\bar{U}\,V \geq 0$$

あるいは

$$\bar{S}\,V + t\,\bar{d}\,V + d\,\bar{z}\,V \geq 0$$

ここで$\bar{Z}\,V = -\bar{W}$とすると$\bar{W}$はVの3行目の要素を3行目にもち，他の要素は全て0となる。$\bar{P} = \bar{S}\,V$，$\bar{q} = \bar{d}\,V$とすればj列目は

$$P_j + t\,q_j - \alpha W_j > 0 \qquad (0 \leq t \leq 1,\ \alpha \geq 0)$$

αがいかなる値でもこの不等式から，tのmaximumとminimumを求められる。P_jとq_jとW_jの範囲は，わかっているから，例えば$q_j > 0$，$P_j \geq 0$，$P_j - W_j \geq 0$とすれば

$$(q_j > 0) \wedge (P_j \geq 0) \wedge (P_j - W_j > 0)$$

の領域内でt（$0 \leq t \leq 1$）とαのいかなる値に対しても上記の不等式が成りたてば，このline全体が物体にかくされていることを意味する。不等式が成り立たない時は，領域内ではlineは他の物体に隠されることはない。このようにして画面内の多くのlineが他の物体に隠されるか否かをテストするのにかかる時間を短縮する。

lineが部分的に見える場合には，その境線を求めるためにいくつかの連立不等式を解く。各々の不等式はt，α平面の領域を表わす。この領域はline $P_j + yq_j - \alpha W_j = 0$によって区切られる。これらの境界線は$t$，$\alpha$平面上にあるが，$t = 0$，$t = 1$，$\alpha = 0$の直線を加えることによって，この場合のlineと物体の不等式の交わる領域を決める。不等式を満足するtとαの値を含む交わりの領域がない時，すなわちline上の点がいずれも不等式を満足しない場合はlineは隠されることはない。

もし，交わる領域があるならば，不等式にαのかわりにtを代入し，tのminimumとmaximumを求める。その結果，lineの見える部分と見えない部分の境点がわかる。

第２章　一般物体の隠れ線消去プログラム

Robertのアルゴリズムは物体を凸物体（convex model）の複合体として扱い，基本となる考え方はpointがmodelの内側にあるか，否かを判別することであり，平行投影のみに関して行っていた。

ここではこの考え方を拡張し透視画法により，凸物体，凹物体，両方の処理に適用した。物体は次の条件を満たすものとする。

1）　平面で構成されている。

2）　閉じている物体である。（辺が2平面の交線である。）

透視法としては，視点と3次元での原点を結んだ直線に垂直な面を投象面とし，3次元での原点と投象面での原点を一致させてある。

物体を原点を中心に回転させた後に，さらにある点を中心にあるきざみの角度で回転して透視する機能も備えている。

このプログラムを用いる際には，次のデータを必要とする。

1）視点の座標

2）各点の座標

3）物体を外側から見て，各々の面について，その面を構成している点を時計まわりに列記する。およびその時の個数

4）回転角度，およびきざみの回転角度と回数

5）回転する時の中心座標

6）物体の個数

7）点の個数，面の個数

処理の手順

1）視点を決める。物体を，$x-y$平面，$y-z$平面，$z-x$平面に関して回転させる。

2）視点と3次元での原点を結ぶ直線と垂直な面を投象面として，各点を透視する。

3）各面が視点方向に表を向けている（on view face）か裏を向けている（hidden face）かを調べる。

4）edgeを構成する二平面が3）のどちらの面であるかを調べ，二平面がhidden

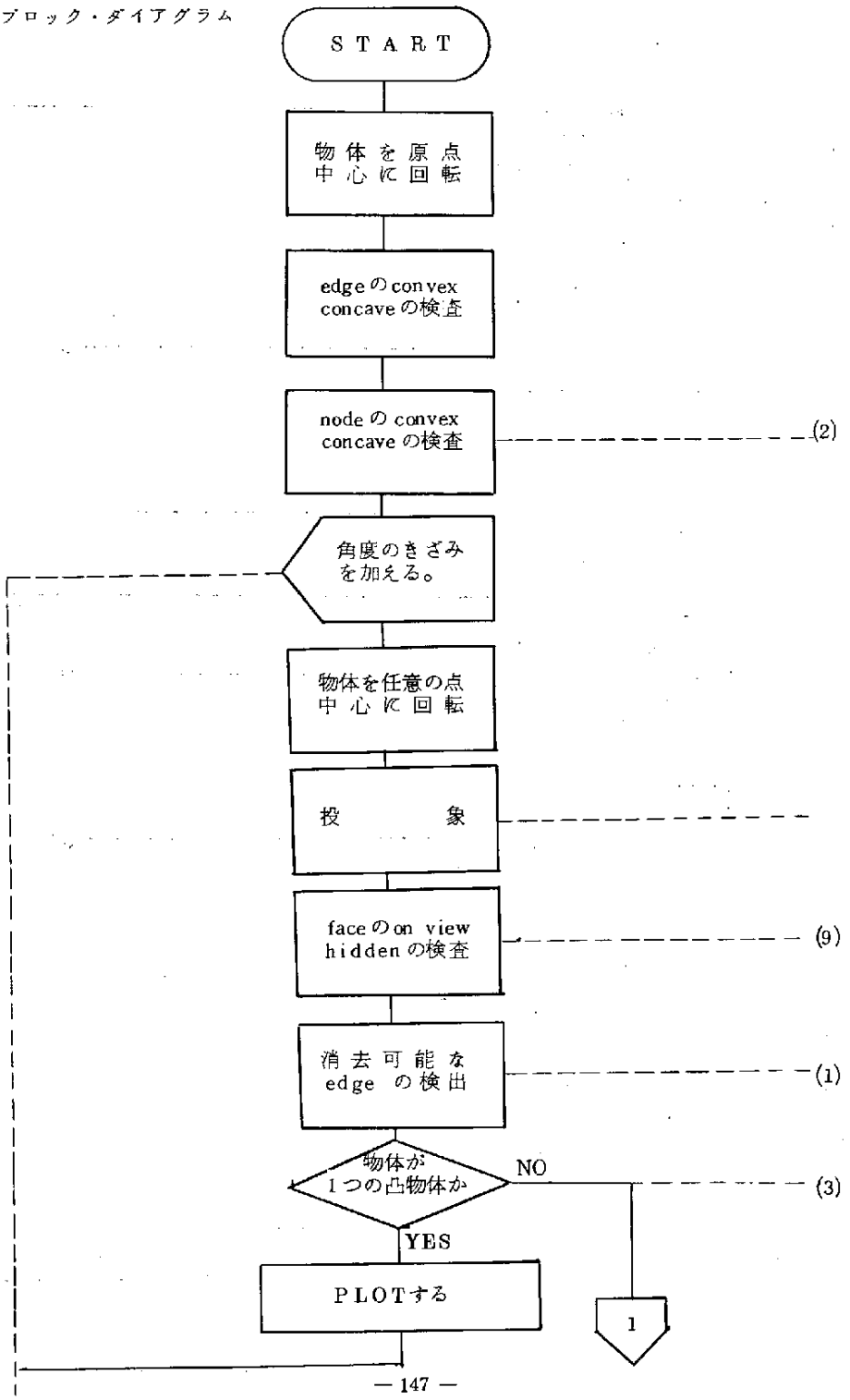
faceである時，又はon view face，hidden face同時に属し，edge が凹である時，edgeはhidden lineである。凸物体であれば他のedge は全て見える。そこで凸物体の時には，ここまでで問題は解決する。ところが凹物体の時には他のedgeについては，各々そのedgeを隠す面があるかどうか検査する。

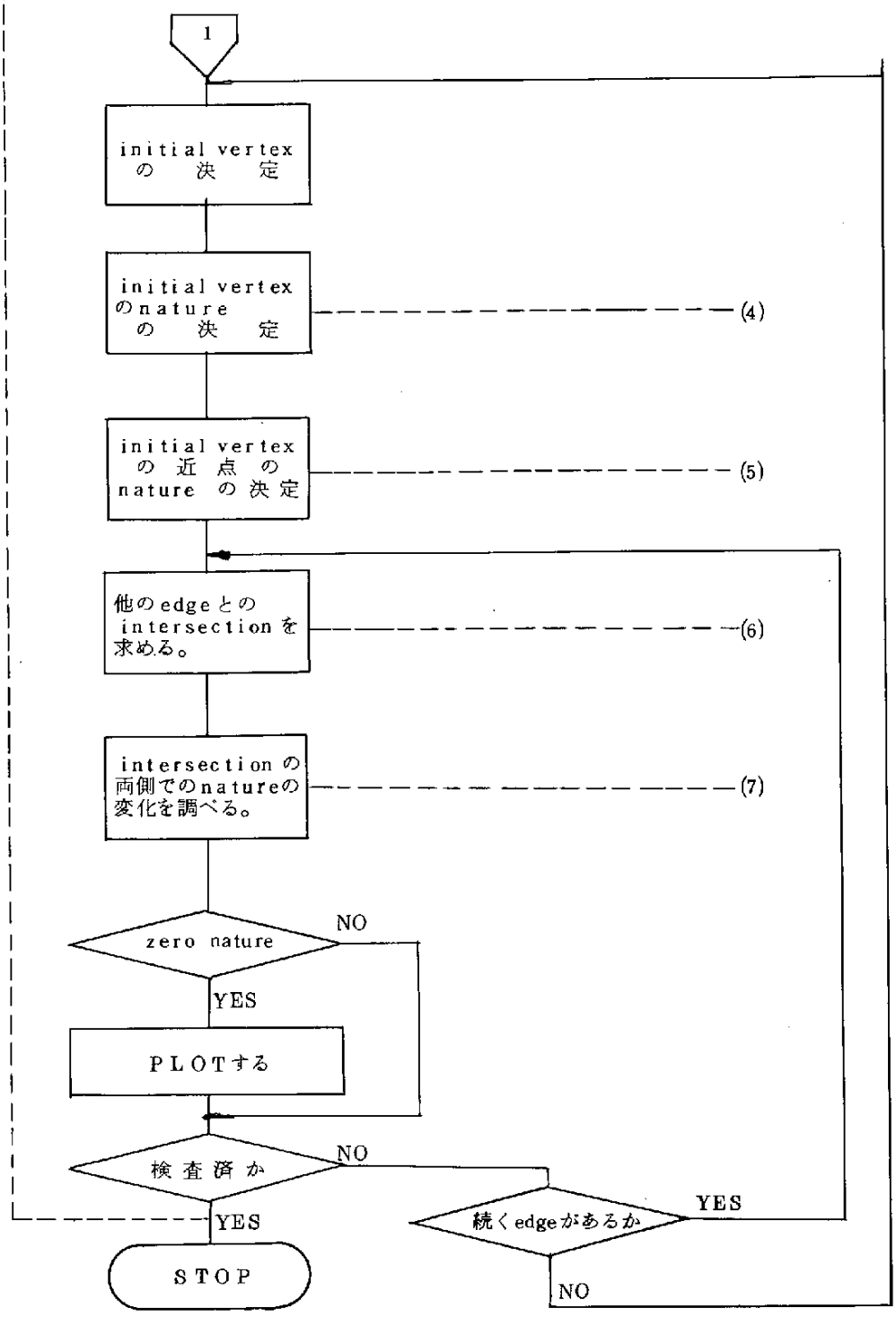
5）残ったedgeを調べ初める点（initial vertex）を探し，その点を隠す面を求める。残ったedgeに沿って順次，隠す面の変化を調べていくが，その時，edgeを透視した線分の端点，及び線分間の交点において，その点に非常に近い両側の点で調べるだけで良い。もしその点を隠す面がないのなら，その点は見えることになる。このようにして全てのedgeについて調べてみる。

（注意） 凸物体であってもいくつかの物体が集っている時には，5）の手順もふまなければならない。そのためにデータとして物体の個数を与えるようにした。

以上の処理のグロックダイアグラムを次に示す。

処理のブロック・ダイアグラム




1
initial vertex の 決 定
initial vertex のnature の 決 定
(4)
initial vertex の 近 点 の nature の決定
(5)
他のedgeとの intersectionを 求める。
(6)
intersectionの 両側でのnatureの 変化を調べる。
(7)
zero nature
NO
YES
PLOTする
検 査 済 か
NO
YES
続くedgeがあるか
YES
NO
STOP

2.1 Algorithm

予備的な定義

FACE　3次元空間における物体を形成する部分平面。

常に閉じている。

EDGEによって囲まれる。

EDGE　二つのFACEによって作られる。

二平面が凸に交わっているとき　convex　edge

二平面が凹に交わっているとき　concave edge

と呼ぶ。

NODE　EDGEの両端の点

少なくとも一つのconcave edgeに属しているときconcave node，

その他をconvex nodeと呼ぶ。

投象面上に於ては下記のように定義する。

物体（3次元）	投象面上（2次元）
FACE	POLYGON
EDGE	SEGMENT
NODE	VERTEX

ON VIEW FACE

視点から見て面を構成する点列が時計まわりである。つまり，その面は外側を視点に向けている。

HIDDEN FACE

ON VIEW FACEとは逆の条件を満たす面

WINNING, LOSING

edge e_1 , e_2 のsegment S_1, S_2 の交点Qがあり，その点において，edge e_1が edge e_2 より視点に近い時，交点は segment S_1 に関してwinningであると言い，segment S_2に関してlosingであると言う。これを図2に示す。

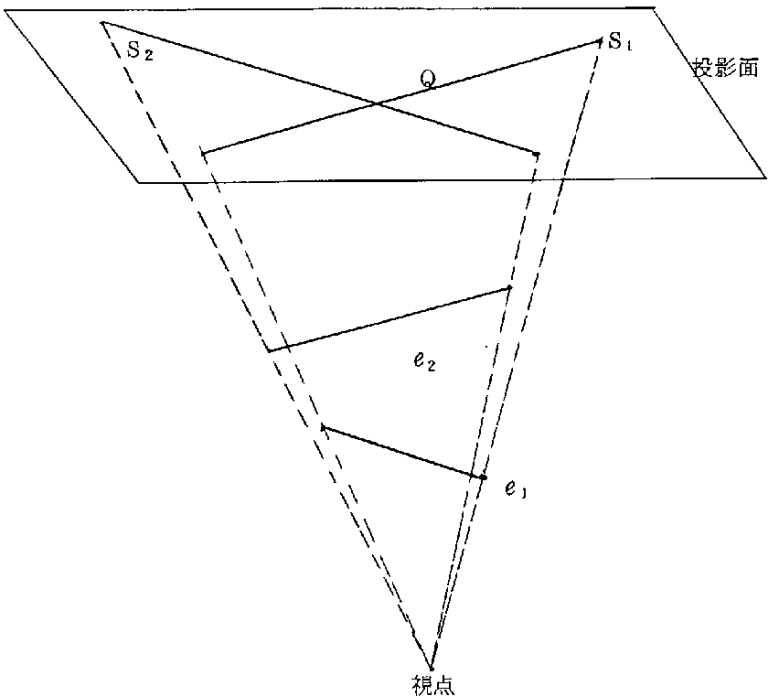
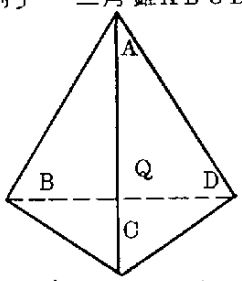


図　2

NATURE　点Pが投象面上の segment に属していて，点Pを隠す on view face の集合をNATUREと呼ぶ。

〔例〕　三角錐ABCDについて考えてみる。



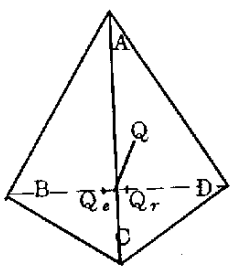
segment BDとACの交点QにおけるNATUREをH，Qの左近傍点 Q_e のNATUREをH_e，右近傍点Q_rのNATUREをH_rとする。

1) Q_e，Q，Q_r がBD上にある。すなわちQがACに関して losing である時

H_e ： △ABC

H　： △ABC，△ACD

H_r ： △ACD



そこで

$H = H_e \cup H_r$が成立する。

さらにＡＣをつくるfaceのうちon view faceの集合をＪとすると次の関係式が求まる。

$G_e = H - H_e$，　$G_r = H - H_r$

$J = G_e \cup G_r$　　$(G_r \cap G_e = \phi)$

ここでG_eはQは隠すがQ_eは隠さないfaceの集合，G_rはQは隠すがQ_rは隠さないfaceの集合を表わす。

2) Q_e，Q，Q_rがＡＣ上にある。すなわちQがＢＤに関してwinningである時

H_e ： ϕ

H ： ϕ

H_r ： ϕ　　（ϕは空集合）

そこで

$H = H_r = H_e$が成立する。

1. ＦＡＣＥのon view，hiddenの決定

自分自身によってかくされるedgeに対応するsegmentを消去するために，物体のすべての面，face jがon viewであるかhiddenであるかを決定する。

face jに対応するpolygonのnameをjとする。polygon jの法線の向きが視点の方向であればface jはhiddenであり，逆方向であればface jはon viewである。

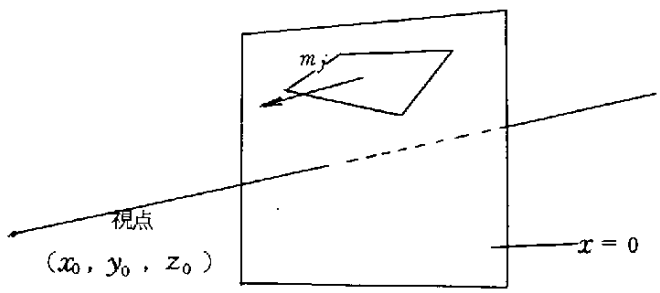


図　3

図３において

$x=0$平面上に投象するとすれば

$m_j \cdot x_0 < 0$　のとき　on view となる。

polygon jの法線m_jはx成分のみであり，物体の内側に向うベクトルである。

m_jの求め方は，ＥＤＧＥのconvexの検査の際のＦＡＣＥの法線の求め方と同様

2. ＥＤＧＥのconvex，concaveの決定

1) face jに対して内部に向う法線ベクトルの決定

face jのnodeの位置ベクトルは座標テーブルX，Y，Zから知ることができる。

点P_{jk}の位置ベクトルを$\mathbb{P}_{jk}$とする。

$\mathbb{P}_{jk}$　（$k=1, 2, 3, \cdots\cdots$）

$\mathbb{P}_{j1}$　（x_1, y_1, z_1）

$\mathbb{P}_{j2}$　（x_2, y_2, z_2）　とすれば

$\mathbb{P}_{j3}$　（x_3, y_3, z_3）

ここでP_{j1}，P_{j2}，P_{j3}はface jをつくる点の任意の３点。

$$A_j m_j = (\mathbb{P}_{j2} - \mathbb{P}_{j1}) \times (\mathbb{P}_{j3} - \mathbb{P}_{j1})$$

$A_j m_j$は大きさA_jの法線ベクトルである。

node i_r，i_sを端点とするedge iをつくる。

face j_1，j_2の法線ベクトルを求め，それぞれをm_{j1}，m_{j2}とする。

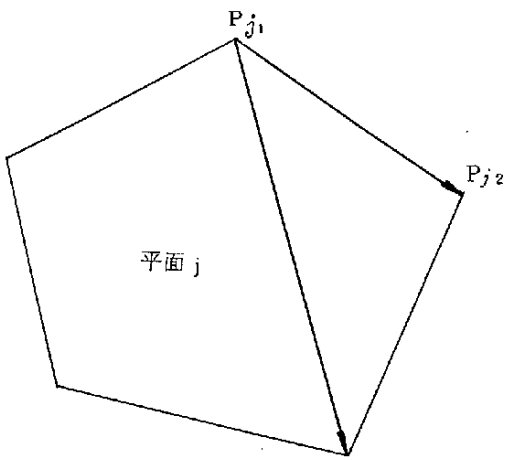


図　4

図４の５角形において任意の３点をP_{j1}，P_{j2}，P_{j3}とすると

法線ベクトルm_jの向きは，紙面から向う側へ向く

2) edgeをつくる二平面の法線を調べる。

$C = ({}^{l}P_{is} - {}^{l}P_{ir}) \cdot (m_{j1} \times m_{j2})$

に於て

$C > 0$ ならば edge は concave

$C < 0$ ならば edge は convex

ここで ${}^{l}P_{is}$ は edge i の端点

${}^{l}P_{ir}$ は edge i の他の端点

m_{j1} は edge i をつくる平面

m_{j2} は edge i をつくる他の平面

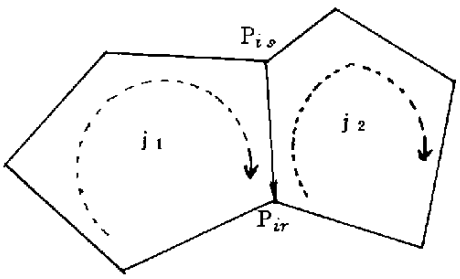
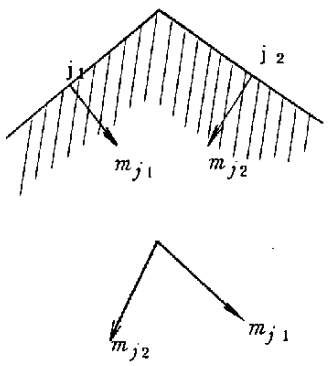


図5において $({}^{l}P_{is} - {}^{l}P_{ir})$ の向きは向うからこちらで，法線のベクトル積と $\pm \frac{\pi}{2}$ の範囲にある。

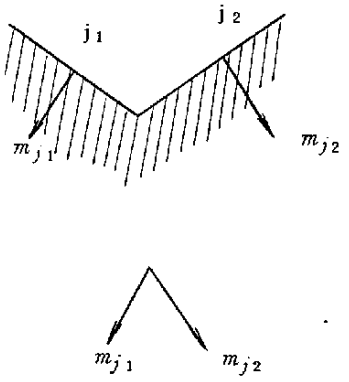
edgeが凸の場合



$(m_{j1} \times m_{j2})$ の向きは

こちらから向うに紙面に垂直

edgeが凹の場合



$(m_{j1} \times m_{j2})$ の向きは

向うからこちらに紙面に垂直

図 5

3. 消去可能なＥＤＧＥの検出

1) ２つのhidden faceにedgeが属している。

2) edgeがconcaveでhidden faceと同時にon view faceに属している。

1)，あるいは2)を満足するsegmentは消去できるが，他のsegmentについてはnature（その点をかくすon view faceの集合）を調べる。

4. initial vertexのnatureの決定

node P_i の投象V_iをinitial vertexとした時P_iを隠すon－view face j（nature）とすればface jは次のような条件を満す。

1) V_i はface jの投象であるpolygonに属す。

2) P_0をview pointとすればsegment $P_i P_0$がface jと対応する面と交わる。

1)を満たすpolygonの求め方

a) initial vertexから半直線をひく。

b) 交わるsegmentに対応するpolygonにmarkをつける。

ただし，この時のpolygonはon view faceを投象したもののみ。

c) markが奇数のpolygonが1)を満す。

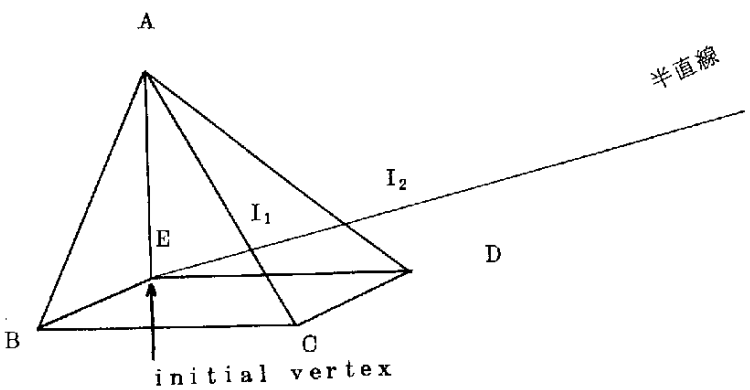


図 6

図6において

initial vertex Eから半直線をひいた時

I_1に対応するpolygonは△ＡＢＣ，△ＡＣＤ

I_2に対応するpolygonは△ＡＣＤ，△ＡＥＤ（hidden face）

よって1)を満たすpolygonは△ＡＢＣ

2) の判定

$x=0$ 平面に投象したものとすれば

P_i, P_0（P_0：視点）を結ぶ直線

$$\frac{x-x_0}{x_i-x_0}=\frac{y-y_0}{y_i-y_0}=\frac{z-z_0}{z_i-z_0}$$ と

3点（x_i, y_i, z_i） $i=1, 2, 3$ を通る平面

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0$$

との交点を求め，x 座標を比較する。

5. initial vertexの近点のnatureの決定

Case a.

initial vertexが凸でwinning intersectionの場合

initial vertexのnatureをH，近点のnatureをH_nとすれば

$H_n=H$ 〔図7 a〕

Case b.

initial vertexが凹でwinning intersectionの場合

initial vertexを含むon view faceの集合をGとすれば

$H_n=H+G'$ （$G'\subset G$）

ここでGはvertexを含むon view faceの集合

又G'はinitial vertexに無限に近い点のnatureでinitial vertexのnatureを求める方法で求められる。〔図7 b〕

Case c.

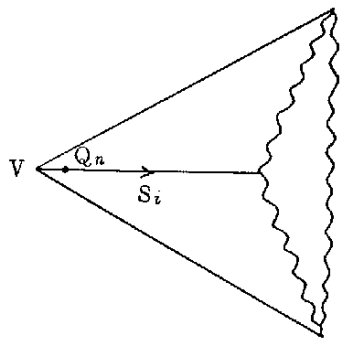
initial vertexが凸でlosing intersectionであるとき

$H=H_n\cup J$, $H_n=H-G_n$ （$G_n\subset J$）

ここでG_nはJに属するon view faceのpolygonでinitial vertexの近点でテストしたもの。〔図7 c〕

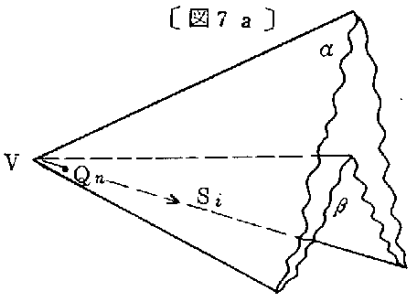
ここで1つあるいはそれ以上のlosing intersectionがinitial vertexに関して，検出されたか，又はこのvertexが凹であるなら，すべてのJ及びGについて考え

ある。



〔図7 a〕

$H = H_n = \phi$



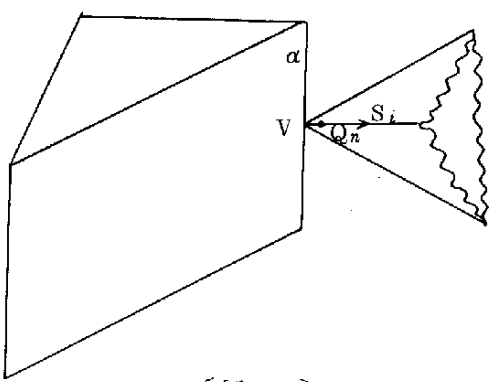
〔図7 b〕

$G = \{ \alpha, \beta \}$

$G' = \{ \alpha \}$

$H = \phi$

$H_n = H + G' = \{ \alpha \}$



〔図7 c〕

$H = \{ \alpha \}$

$J = \{ \alpha \}$

$G_n = \{ \alpha \}$

$H_n = H - G_n = \phi$

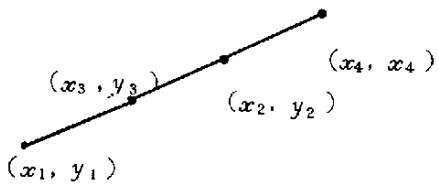
図 7

6. intersectionの決定

intersectionに於てnatureの変化が起きるため，initial vertexからそれを含むsegmentに沿って順次intersectionを求めて検査する。

線分のintersectionにおいては，図8に示すようないろいろな状態が生ずるので，natureを調べる上で注意せねばならない。

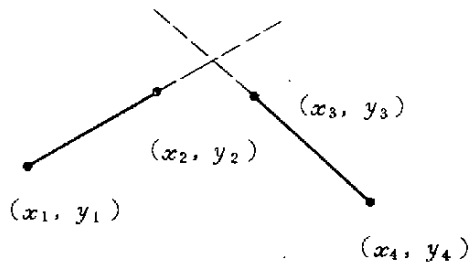
a)



2線分が

重った時

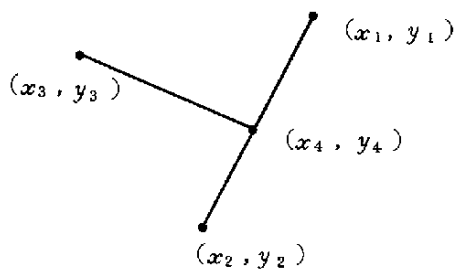
b)



2線分の延長上で

交っている時

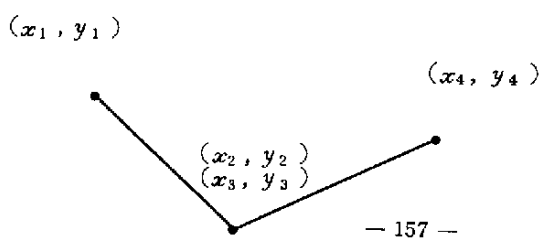
c)



交点が他の線分

の端点である時

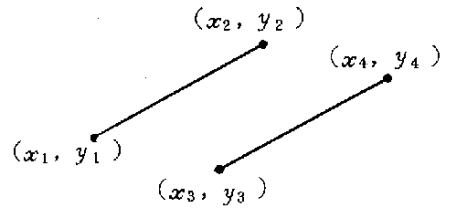
d)



線分の一端点が

一致している時

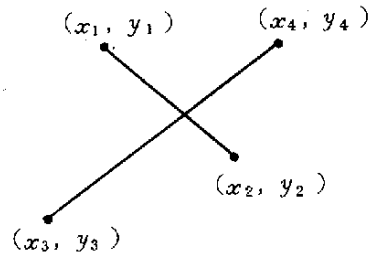
e)



線分が平行

である時

f)



線分が

交わっている時

図　8

交点を求めるには

(x_1, y_1), (x_2, y_2)を通る直線

$$(z_1 - z_2)y + (y_2 - y_1)z = y_2 z_1 - y_1 z_2 \quad \cdots\cdots (1)$$

(x_3, y_3), (x_4, y_4)を通る直線

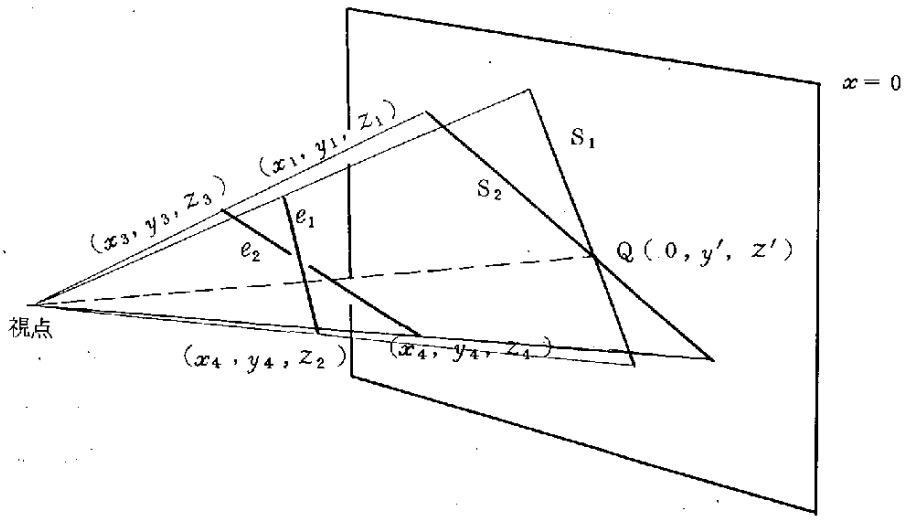
$$(z_3 - z_4)y + (z_4 - z_3)z = y_4 z_3 - y_3 - z_4 \quad \cdots\cdots (2)$$

(1), (2)の連立方程式を解くことによって得られる。

8. intersectionにおけるnatureの変化

1) 検査中のsegmentに関して，他のすべてのsegmentとのintersectionを求め，そのintersectionがlosingであるかwinningであるかを調べる。

$x = 0$平面に投象するとすれば，winning losingは次のように求められる。



e_1の端点の座標を(x_1, y_1, z_1) (x_2, y_2, z_2)

e_2の端点の座標を(x_3, y_3, z_3) (x_4, y_4, z_4)

S_1, S_2の交点Qの座標を，$(0, y', z')$

視点の座標を(x_0, y_0, z_0)とすれば

(x_0, y_0, z_0), $(0, y', z')$を通る直線は

$$\frac{x}{x_0} = \frac{y-y'}{y_0-y'} = \frac{z-z'}{z_0-z'} \quad \cdots\cdots (1)$$

となり

(x_1, y_1, z_1) (x_2, y_2, z_2)を通る直線は

$$\frac{x-x_1}{x_2-x_1} = \frac{y-y_1}{y_2-y_1} = \frac{z-z_1}{z_2-z_1} \quad \cdots\cdots (2)$$

となる。

(1)と(2)式から視点とQとを結ぶ直線とe_1との交点座標が求まる。同様にしてe_2との交点座標も求める。おのおのの座標のx座標と視点のx座標との距離を求め，比較することによって点QがS_1あるいはS_2に関してwinningであるか，あるいはlosingであるかがわかる。

2) intersectionの状態と1)の状態によって次のようなcaseに分けられる。

a. intersectionがwinningであれば

$H_r = H_e$

b.1. intersectionがlosingでQが交っているsegmentの端点でないとき

Jが1つの元しか含まず$J \subset H_e$のとき

$H_r = H_e - J$

Jが1つの元しか含まず$J \not\subset H_e$のとき

$H_r = H_e + J$

b.2. Jが2つの元を含んでいるとき，$J = \{ j_e, j_r \}$としj_e，j_rのうち必ず1つがH_eに属している，それをj_eとすれば

$H_r = (H_e - \{ j_e \}) + \{ j_r \}$ 〔図10a〕

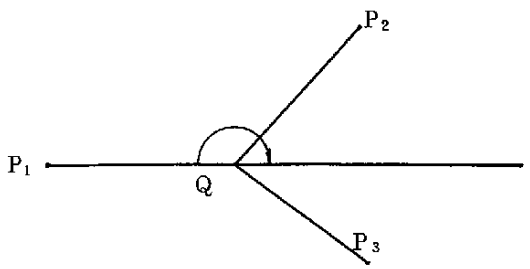
c.1. 交点がlosingでQが交わっているsegmentの端点であり，segment S_jとS_iが重っていないとき，Q_eからQ_rへ時計方向にまわることによってsegment S_jに会わなければ

$H_r = H_e$

もし，会ったならばb.1. b.2. の場合と同様である。

〔図10b〕

ここで時計方向にまわした時，ぶつかるsegmentはsegmentの外積を求め，符号を調べることによってわかる。



$(P_1 - Q) \times (P_2 - Q)$：向きはこちらから向う

$(P_1 - Q) \times (P_3 - Q)$：〃 向うからこちら

c.2. segment S_iとS_jが重っているとき

$J \subset H_e$ ならば

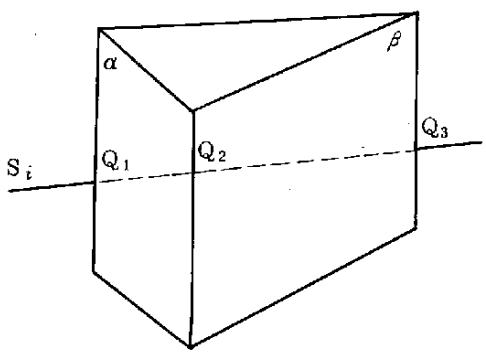
$H_r = H_e - J$

$J \supset H_e$ ならば 〔図10c〕

$H_r = H_e + J$

このようにしてintersectionの性格によりそれぞれのnatureが求められるが，同じ点において，1つ以上の交点が重っているときその点のnature の変化はすべての交点に対する条件が総合されたものとなる。

intersectionからintersectionまでnatureは変化しないので，segment上を左から右へ移動させた時，H_rは次のintersectionのH_e として用いることができる。このようにして順次intersection でのnatureの変化を調べていく。



〔図10a〕

Q_1において

$H_e = \phi$　　$J = \{\alpha\}$

$H_r = (H_e - \phi) + \{\alpha\} = \{\alpha\}$

Q_2において

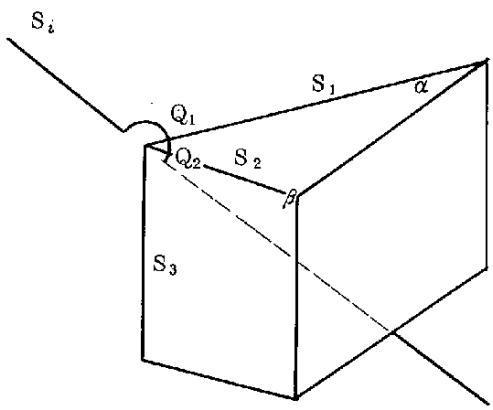
$H_e = \{\alpha\}$　　$J = \{\alpha, \beta\}$

$H_r = (H_e - \{\alpha\}) + \{\beta\} = \{\beta\}$

Q_3において

$H_e = \{\beta\}$　　$J = \{\beta\}$

$H_r = (H_e - \{\beta\}) + \phi = \phi$



〔図10b〕

Q_1において

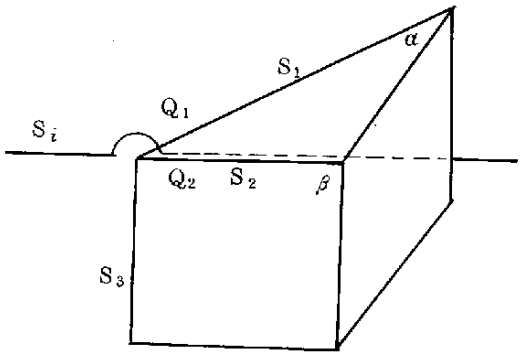
$H_e = \phi$　　$J = \{\alpha\}$

$H_r = \phi + \{\alpha\} = \{\alpha\}$

Q_2において

$H_e = \{\alpha\}$　　$J = \{\alpha, \beta\}$

$H_r = \{\alpha\} + \{\beta\} = \{\alpha, \beta\}$



〔図１０ｃ〕

Q_1において

$H_e = \phi \qquad J = \{\alpha\}$

$H_r = \phi + \{\alpha\} = \{\alpha\}$

Q_2において

$H_e = \{\alpha\} \quad J = \{\alpha, \beta\}$

$H_r = \{\alpha\} + \{\beta\} = \{\alpha, \beta\}$

図１０

9. 投　　象

与えられた視点をx軸上にもってくるように視点と各点を回転し，x軸上から$x=0$平面に投象する。視点の座標を（x_0，y_0，z_0），３次元座標を同次座標で表わし（x，y，z，１），投象した２次元の新座標を（X，Y，Z，１）とすれば次式が成り立つ。

$$(X, Y, Z, 1) = (x, y, z, 1) \begin{pmatrix} 0 & 0 & 0 & -1/x_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \left(0,\ y,\ z,\ -\frac{x}{x_1} + 1\right)$$

$$= \left(0, \frac{y x_0}{x_0 - x}, \frac{z x_0}{x_0 - x}, 1\right)$$

よって（x，y，z）の投象面上の座標は$\left(\frac{y x_0}{x_0 - x}, \frac{z x_0}{x_0 - x}\right)$となる。

22 DATA STRUCTURE

任意の３次元物体に対し，この隠れ線消去の処理をおこなうためには，次のようなデータを準備せねばならない。

1） ３次元座標を表示するX，Y，Z （１次元）

2） 投象面上の座標 YP，ZP （１次元）

3） nodeのconvex，concaveの表示 INOD （１次元）

4） faceの情報 IFACE （３次元）

5） faceのテーブルを作成するためのワークテーブル IWORK （１次元）

6） edgeのnodeに対する情報 IEDN （３次元）

7） edgeのfaceに関する情報 IEDF （２次元）

これらのデータを元にして各種のテーブルを作成する。

1. データの与え方

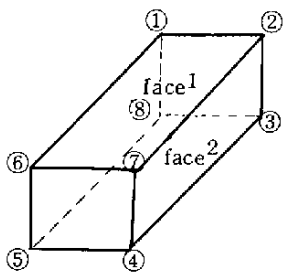
1） テーブル X，Y，Z

各点に１から順に番号をつけ（internal name),番号順にx座標はテーブルX，y座標はテーブルY，z座標はテーブルZに登録する。

2） テーブル IWORK

各面に１から順に番号をつけ，番号順にその面を物体の外側から見て，その面をつくるnodeのinternal nameを時計まわりに登録する。

例えば



IWORK

①	②	⑦	⑥	⑦	②	③	④	⑦	④	⑤……

face 1　　face 2

3) テーブル　ＩＦＡＣＥ

各面をつくるnodeの状態を示すのにＩＷＯＲＫの何番目から初まり，何点（何角形）であるかを示す。

2）の例についてＩＦＡＣＥテーブルをつくると次のようになる。

ＩＦＡＣＥ

face No.	ＩＷＯＲＫのポインター	nodeの個数
1	1	4
2	5	4
	9	4
	⋮	⋮

これらのテーブル間の関係を図１１に示す。

2. その他に与えるデータ

1）視点の３次元での座標（x_c，y_c，z_c）

投象面が視点と３次元の原点とを結んだ直線に垂直になるということを考慮して視点を配置する。

2）原点中心の回転角度

$x-y$平面におけるx軸からy軸への回転角度と

$y-z$平面　〃　y軸からｚ軸　〃

$z-x$平面　〃　ｚ軸からx軸　〃　を

与える。実際には，このうちの２つの角度で回転したい角度は決まる。

3）連続的に回転させるための回転角度のきざみ，回数，中心点

回転角度のきざみは，2）と同様に与える。

中心点は３次元座標（x_0，y_0，z_0）で与える。

4）物体の個数

5）点，面の個数

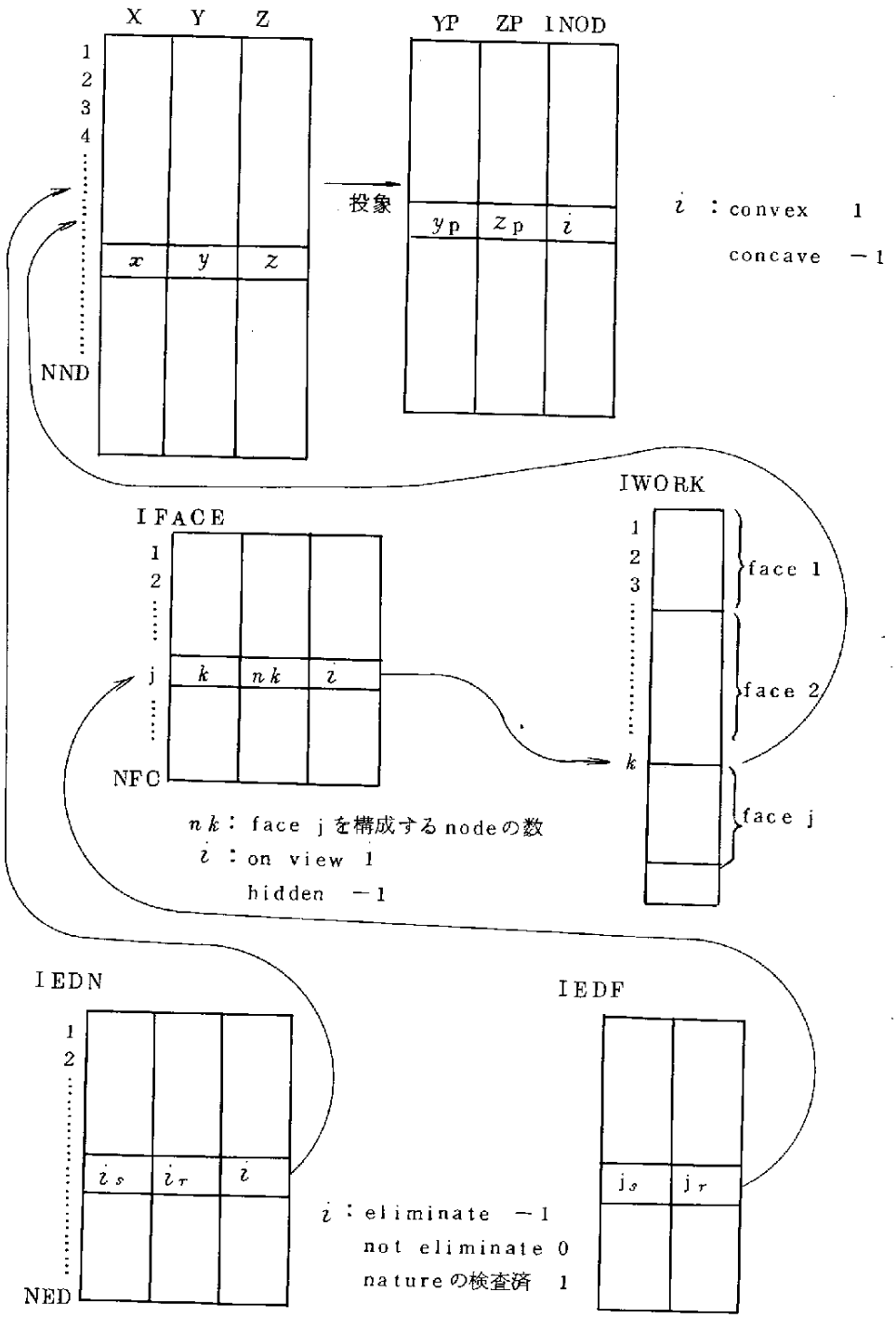


図11 各種テーブル間の引用手順

テーブル作成

ユーザーが与えたX，Y，Z，IWORK，IFACEのデータを用いて，プログラムの中でIEDF，IEDNのテーブルを作成する。

X	……	NODEの internal name順の x 座標
Y	……	〃 y 座標
Z	……	〃 z 座標
IWORK	……	FACEを右まわりプロットしたNODEの internal name
IFACE	……	IFACE(1, i)：face i のIWORKの始めの番地 IFACE(2, i)：face i のnodeの個数
IEDN	……	IEDN(1, i) edge i を構成するnodeの IEDN(2, i) internal name IEDN(3, i) 消去可能，natureの検査済
IEDF	……	IEDF(1, i) edge i を構成するfaceの IEDF(2, i) internal name

ＩＥＤＮ，ＩＥＤＦの作成

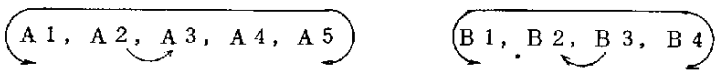
edge Ｉが face Ａと face Ｂから構成されているとき，

face Ａは　右まわりに　node　Ａ１，Ａ２，Ａ３，Ａ４，Ａ５

face Ｂは　右まわりに　node　Ｂ１，Ｂ２，Ｂ３，Ｂ４

で構成されているときＩＥＤＮ，ＩＥＤＦは次のように作成する。

face Ａ　　face Ｂ



○　ＥＤＧＥの端点が　Ａ２，Ａ３のとき

Ａ２＝Ｂ３　かつ　Ａ３＝Ｂ２ならば

ＩＥＤＮ（１，Ｉ）＝Ａ２　（Ｂ３）

ＩＥＤＮ（２，Ｉ）＝Ａ３　（Ｂ２）

ＩＥＤＦ（１，Ｉ）＝Ａ　（Ｂ　）

ＩＥＤＦ（２，Ｉ）＝Ｂ　（Ａ　）

○　ＥＤＧＥの端点が　Ａ５，Ａ１のとき

Ａ５＝Ｂ１　かつ　Ａ１＝Ｂ４ならば

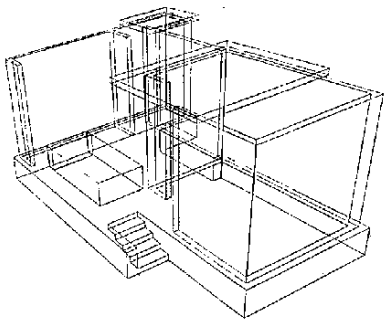
ＩＥＤＮ（１，Ｉ）＝Ａ５　（Ｂ１）

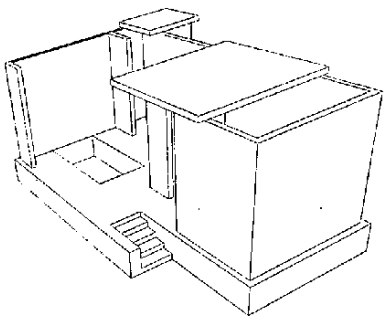
ＩＥＤＮ（２，Ｉ）＝Ａ１　（Ｂ４）

ＩＥＤＦ（１，Ｉ）＝Ａ　（Ｂ　）

ＩＥＤＦ（２，Ｉ）＝Ｂ　（Ａ　）

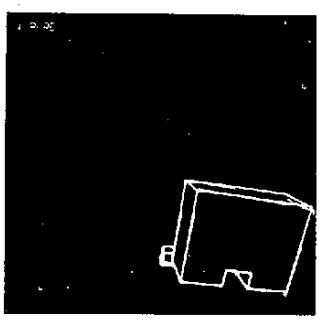
2.3 実　　例

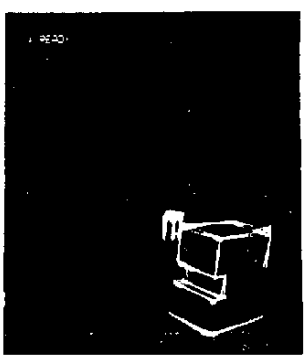




XYプロッタへの出力

図１２





回転を施したもの

図１３

図１４

1)　ステップ数　　８６０　FORTRAN　ステートメント

2)　ＳＩＺＥ　　２９Ｋ　ＷＯＲＤ　（３６bit／ＷＯＲＤ）

2.4 結果および考察

1) 計算時間

以上のようなalgorithm で実際に動かしてみると，まだ計算時間が非常にかかることがわかる。

2) X，Y，Z，IWORK，IFACEのデータを作成するのに手作業で行っているため，正確なデータを作るのが大変である。そこで3次元空間における物体の各点の座標をよみとれる装置を用いれば，便利である。

3) 連続的に回転させる機能は持たせてあるが，単純な物体の場合には効果的であるが，複雑な物体の場合には適さない。

edgeの数をnとした時，計算値がほぼ $\frac{n(n-1)}{2}$ 倍になる。

第３章　２変数関数の立体表示プログラム

今日の物理学，工学の分野に於ては，図で表わすことによって明解になることが多い。図は，ほとんどが１変数関数か２変数関数である。１変数関数は２次元であるため，プロットするのは容易であるが，２変数関数は３次元であるため，２次元に投象して表わす。ここではより立体的に見えるよう，関数の曲面自身によって視点から見えない線は消去してある。曲面はメッシュによって表わされメッシュ間は直線で結ばれる。

ここで表わせる関数は次の条件を満さなければならない。

1) 関数ｚ（x，y）は，表示しない領域内において（x，y）に対してただ１つのｚが定まる一価関数であること。

2) ｚの絶対値が無限大になる時にはｚの表示上の極限を持たせておく。

3.1 Algorithm

消去手順は次のようである。

1) 領域をあるきざみ幅でメッシュに区切る。

2) メッシュ上の点Ｐが視点から見えるか，見えないか，また見える時には面の下側から見えるか，上側から見えるかを判定する関数 $\phi(P)$ を考える。関数 $\phi(P)$ を求めるために，視点と点Ｐを結ぶ直線と関数（実際には直線近似されている）の上下の状態を示す関数 δ を導入する。

3) たて，よこのメッシュで隣りあう点Ｐ，Ｑの $\phi(P)$ を調べることにより，点Ｐ，Ｑ間の線の状態が決定できる。

4) ３次元における点を２次元の投象面に透視する。

(I) 関数 δ（Ｐ，$\tilde{P}$）の導入

点Ｐ（function 上の一点）と視点を結ぶ直線が $\tilde{P}$ において function の上側にあるか下側にあるか示す関数を $\delta(P, \tilde{P}) = \tilde{z} - z^{*}$ とする。ここで $\tilde{P}$ は視点とＰを結ぶ直線を xy 平面に平行投影し，その直線上を与えられた領域内で点Ｐまで動く。この様子を図１５に示す。

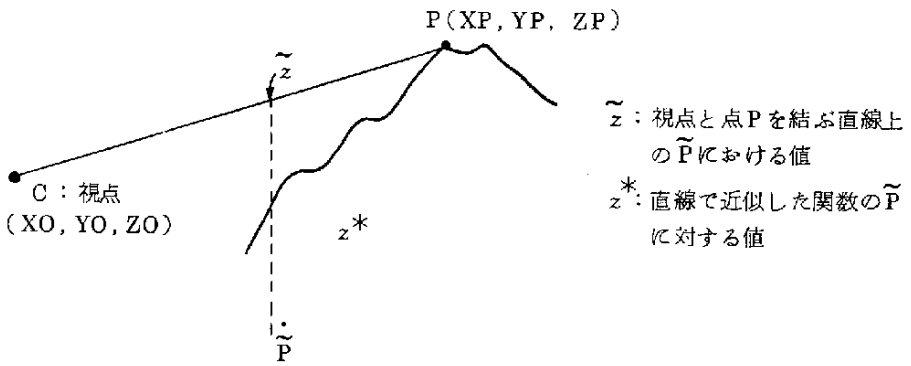
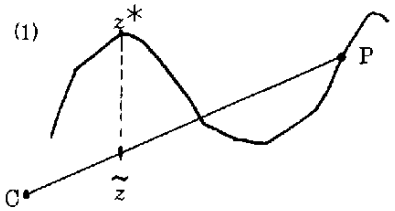
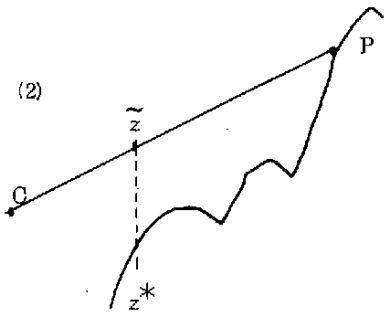


図 15

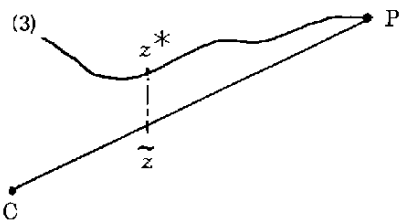
これは図16の示すような3つの場合に分けられる。



点Pが関数に隠されてみえない。
Pはhidden point



関数を上から見る。
Pはvisible point



関数を下から見る。
Pはvisible point

図 16

点Pが視点に対してUisible Point((2)または(3))であるかhidden point((1))であるか半断するために$\tilde{P}$を視点から点Pまで動かし，$\delta(P, \tilde{P})$の変化を調べる。

これを図示すると図17のようになる。

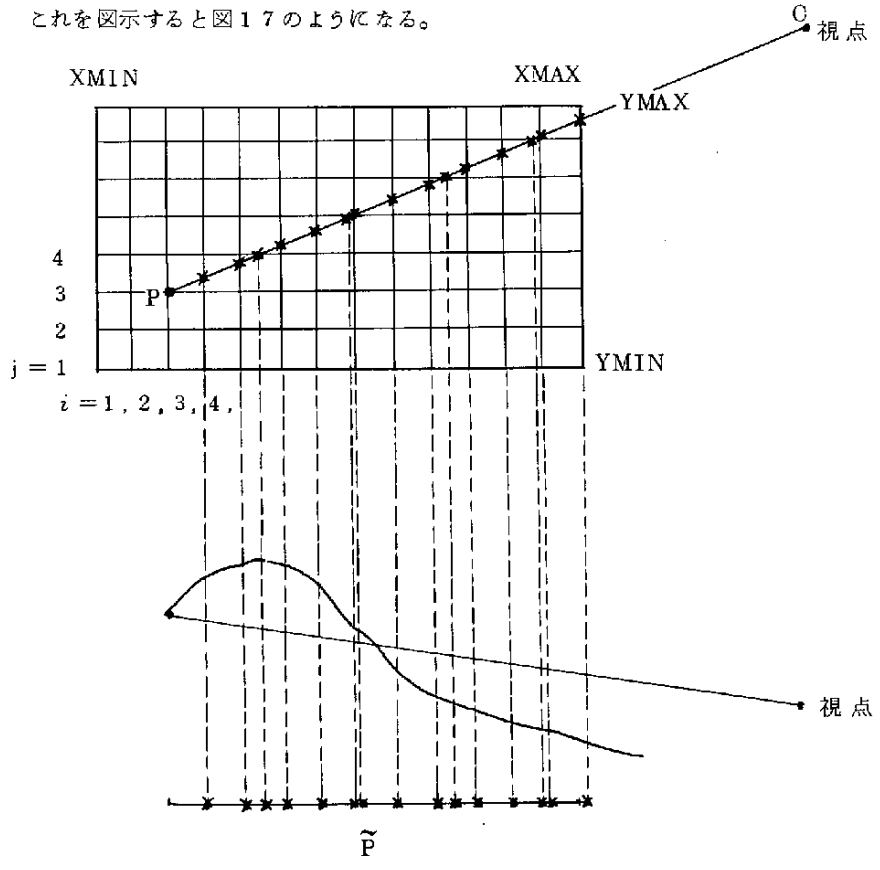


図 17

点Pから視点を結ぶ直線をxy平面に平行投影した直線とx＝XMIN＋(i－1)＊DXあるいはy＝YMIN＋(j－1)＊DYとの交点を$\tilde{P}$とする。ここでDXはxのきざみ幅，DYはyのきざみ幅，iはX方向へのカウンター，jはY方向へのカウンターを表わす。

図17のすべての$\tilde{P}$に対して，$\delta(P, \tilde{P})$を求める。

その結果

(1)の場合

$\delta(P, \tilde{P})$は途中で符号が変わる。

(2)の場合

$\delta(P, \tilde{P})$は常に正

(3)の場合

δ (P, $\tilde{P}$) は常に負

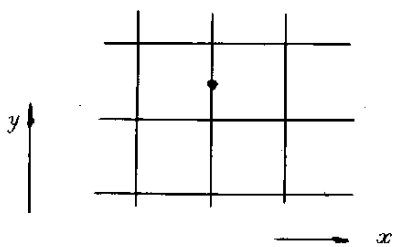
ここで $\tilde{z}$, z^* は次のように求められる。

Pの座標を (x, y, z), $\tilde{P}$ の座標を ($\tilde{x}$, $\tilde{y}$, $\tilde{z}$)

視点の座標を (x_0, y_0, z_0) とすれば

$$\frac{\tilde{x}-x_0}{x-x_0}=\frac{\tilde{y}-y_0}{y-y_0}=\frac{\tilde{z}-z_0}{z-z_0}$$

$\tilde{x}$が constant の時 ($\tilde{x}$がメッシュ上にある)



$$\tilde{z}=\frac{(\tilde{x}-x_0)(z-z_0)}{x-x_0}+z_0$$

$\tilde{y}$が constant の時 ($\tilde{y}$がメッシュ上にある)

$$\tilde{z}=\frac{(\tilde{y}-y_0)(z-z_0)}{y-y_0}+z_0$$

関数zを直線で近似したものが z^* であるから

$\tilde{x}$が constant の時

$$\frac{z^*-z(\tilde{x},y_j)}{z(\tilde{x},y_{j+1})-z(\tilde{x},y_j)}=\frac{\tilde{y}-y_j}{y_{j+1}-y_j} \qquad y_j \leqq \tilde{y} \leqq y_{j+1}$$

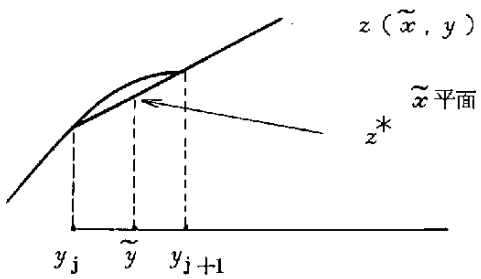


図 18

同様に

$\tilde{y}$ が constant の時

$$\frac{z^{*}-z(x_i,\tilde{y})}{z(x_{i+1},\tilde{y})-z(x_i,\tilde{y})}=\frac{\tilde{x}-x_i}{x_{i+1}-x_i} \qquad x_i\leqq\tilde{x}\leqq x_{i+1}$$

上記の式より z^{*} は求められる。

(Ⅲ) $x_i\leqq\tilde{x}\leqq x_{i+1}$ なる x_i , x_{i+1} の求め方

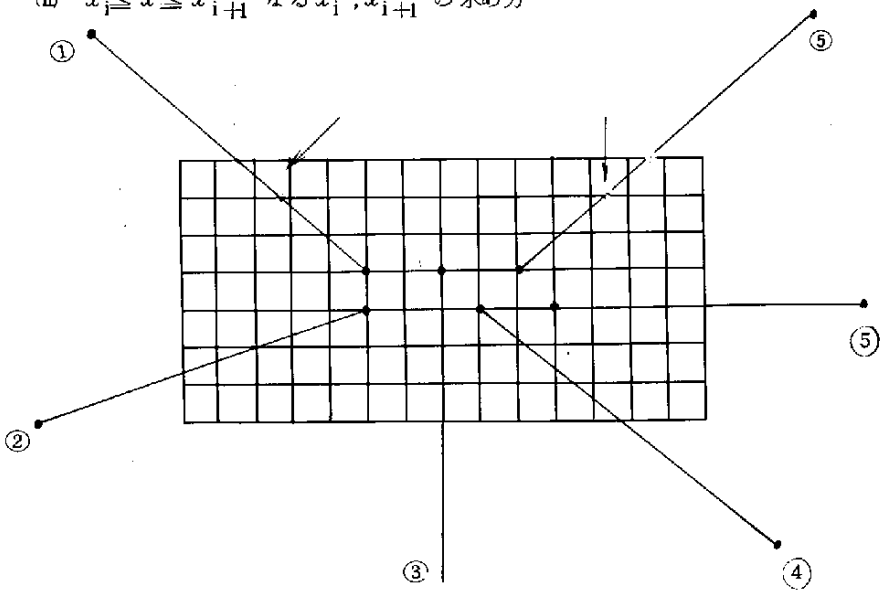


図 19

1) x_i , x_{i+1} を求める。

①, ②の場合

XMIN から順次調べていく。1番目の $\tilde{x}_1$ が

$x_i\leqq\tilde{x}_1\leqq x_{i+1}$ とすれば2番目の $\tilde{x}_2$ は

$x_i\leqq\tilde{x}_2\leqq x_{i+1}$, あるいは $x_{i+1}\leqq\tilde{x}_2\leqq x_{i+2}$ が成りたつ。

④, ⑥の場合

XMAX から順次調べていく。1番目の $\tilde{x}_1$ が $x_i\leqq\tilde{x}_1\leqq x_{i+1}$ とすれば2番目の $\tilde{x}_2$ は $x_i\leqq\tilde{x}_2\leqq x_{i+1}$, あるいは $x_{i-1}\leqq\tilde{x}_2\leqq x_i$ が成りたつ。

③の場合

$\tilde{x}$ は考える必要がない。

⑤の場合

$\tilde{x} = x_i$ ，あるいは $\tilde{x} = x_{i+1}$ に相当する。

即ち，$z^* = z(x_i, \tilde{y})$ ，あるいは $z^* = z(x_{i+1}, \tilde{y})$ である。

2) y_i ・ y_{i+1} も 1) と同様に求められる。

(III) 関数 $\phi(P)$ の導入及び比較法

visible point 間を直線で結ぶ際に次のような関数 $\phi(P)$ を考える。

$$\begin{cases} \phi(P) = +1 & P\text{がvisibleで}\delta(P, \tilde{P}) > 0 \\ \phi(P) = -1 & P\text{がvisibleで}\delta(P, \tilde{P}) < 0 \\ \phi(P) = 0 & P\text{がhidden} \end{cases}$$

メッシュ上のそれぞれの点で $\phi(P)$ を求め，となり合う点（同じ行のとなり合う点，あるいは同じ列のとなり合う点）と比較することによってどこまで直線をひくか決定する。

任意のとなり合う点をP，Qとした時，次のような①②③④の4つのcaseに分けられる。

$\phi(P)$ / $\phi(Q)$	−1	0	1
−1	①	③	④
0	③	②	③
1	④	③	①

CASE ①

$$\phi(P) = \phi(Q) = \pm 1$$

この場合，点P，Qは共にvisible であるので直線で結ぶ。実際にはP，Q間にhidden pointがある場合もあるが，これはきざみ幅を適当に小さくすることによって防ぐことができる。

CASE ②

$$\phi(P) = \phi(Q) = 0$$

この場合，点P，Qは共にhiddenである。実際には，P，Q間にvisible pointがある場合もあるが，CASE ①と同様にきざみ幅のえらび方で防ぐことができる。

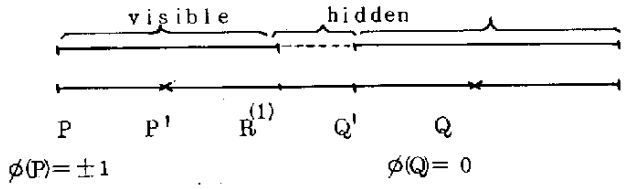
P，Q間の直線は表示しない。

CASE ③

$$\phi(P) = \pm 1 \ , \ \phi(Q) = 0$$

この場合は，点Pはvisibleで Qはhiddenである。この場合PQ間のある点Rを考え，PR間のすべての点はvisibleで，かつRQ間のすべての点がhiddenとなるような点Rを求め，P，R間を直線で結ぶ。

Rは次のようにして求められる。（binary search）



1) PQの中点$R^{(1)}$に於ける$\phi(R^{(1)})$を求める。

2) $\phi(R^{(1)}) = \phi(P)$なら

$R^{(1)}$Qの中点Q'に於ける$\phi(Q')$を求め，$R^{(2)} = Q'$とする。

$\phi(R) = \phi(Q)$なら

P $R^{(1)}$の中点P'に於ける$\phi(P')$を求め，$R^{(2)} = P'$とする。

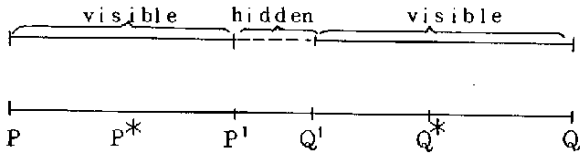
上記2)を繰り返し，$|R^{(i)} - R^{(i+1)}| < \varepsilon$となった時の（ここで$\varepsilon$はきざみの$\frac{1}{100}$とした）

$R^{(i+1)}$をRとする。

CASE ④

$$\phi(P) = -\phi(Q) = 1$$

この場合，点P，Qは共にvisibleであるが，PQ間に，hiddenな部分がある。そこで，PP'間のすべての点P^*が$\phi(P) = \phi(P^*)$となるようなP'と，Q' Q間のすべての点Q^*が$\phi(Q) = \phi(Q^*)$となるようなQ'を求める。



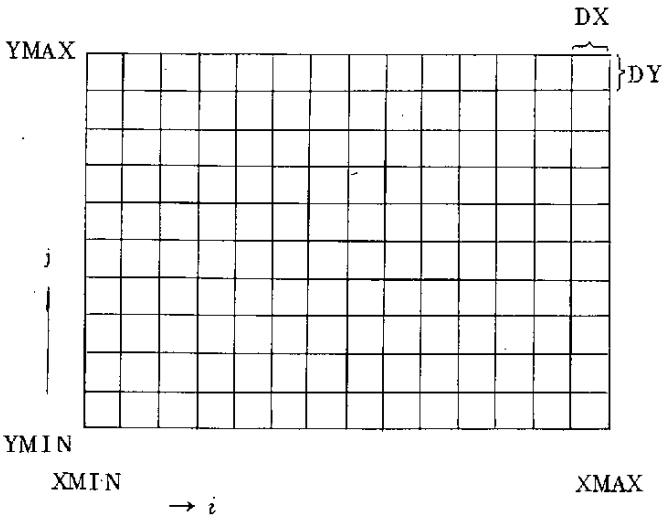
$\phi(P) = 1$ $\phi(Q) = -1$

P P'と Q'Q 間を直線で結ぶ。

尚，P'と Q' はCASE ③のRを求める方法で求められる。

(Ⅳ) 描く手順

XをconstantとしてYMINからYMAXへきざみDYで移動しXはXMINからきざみDXでXMAXまで移動する。



$$NX = \frac{XMAX - XMIN}{DX} + 1.0 + \frac{DX}{2}$$

$$NY = \frac{YMAX - YMIN}{DY} + 1.0 + \frac{DY}{2}$$

i はX方向へのカウンター，j はY方向へのカウンターとしたとき

1) $i = 1$ ，$j = 1$ ならば

$\phi(P)$ を $\phi(1)$ に登録

2) $i = 1$ ，$j \neq 1$ （$j = 2$ ，……NY）ならば

$\phi(P)$ と $\phi(j-1)$ を比較し，プロットした後に

ϕ(P)を$\phi(j)$に登録

3) $i \neq 1$，$j = 1$（$i = 2$，…… NX）ならば

ϕ(P)と$\phi(j)$を比較し，プロットした後に

ϕ(P)を$\phi(j)$に登録

4) $i \neq 1$，$j \neq 1$（$i = 2$，…… NX，$j = 2$，……NY）ならば

ϕ(P)とϕ（$j-1$）を比較し，プロット

ϕ(P)と$\phi(j)$を比較し，プロットした後に

ϕ(P)を$\phi(j)$に登録

(V) 透視法

3次元の物体を投象し，2次元の投象面に表わした時，人間の目で見た形に最も近い投象法が透視法である。

3次元における任意の点をP，平面π上の点をP'とした時，すべてのPP'は点Cで交わる。この点Cのことを視点と呼ぶ。平面πは投象面と呼び，この面の法線は視線と平行になる。

関数$z = f(x, y)$に対して，直交座標(x, y, z)が対応する。

視点Cの座標を(C_x, C_y, C_z)，視線とx軸となす角度，y軸となす角度，z軸となす角度をα，β，γとする。距離をdとし，$|CQ| = d$となるように視線上にQをとる。Qと法線CQによって投象面πが決まる。任意の点P(x, y, z)と視点Cとの直線が投象面πと交わる点をP'(ξ, η, ζ)とする。P'は投象面π上のPの投象である。この様子を図20に示す。

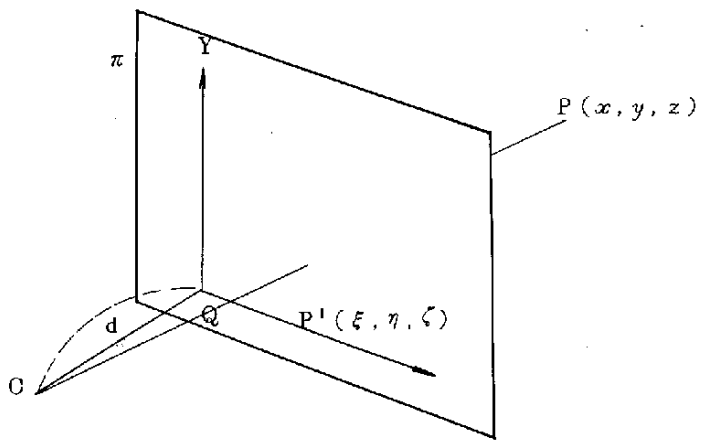


図 20

〔図20〕よりQの座標（q_x，q_y，q_z）は次のように表わせる。

$$q_x = C_x + d\cos\alpha$$

$$q_y = C_y + d\cos\beta$$

$$q_z = C_z + d\cos\gamma$$

$P'(\xi, \eta, \zeta)$，$P(x, y, z)$，$C(C_x, C_y, C_z)$が一直線上にあることから

$$\frac{\xi - C_x}{x - C_x} = \frac{\eta - C_y}{y - C_y} = \frac{\zeta - C_z}{z - C_z} = K$$

と表わせる。ここでKは次のようになる。

$$K = d / [(x - C_x)\cos\alpha + (y - C_y)\cos\beta + (z - C_z)\cos\gamma]$$

またξ，η，ζは上の式より

$$\xi = C_x + K(x - C_x)$$

$$\eta = C_y + K(y - C_y)$$

$$\zeta = C_z + K(z - C_z)$$

となる。

一方投象面上のX軸は$\zeta - q_z = 0$なる平面と投象面πとの交線で決められる。この座標系も直交座標であるからX軸の単位ベクトルをU_x，Y軸の単位ベクトルをU_yとすれば

$$U_x \cdot U_y = 0$$

が成りたつ。ここでQを原点とすればP'は新しい座標系では$(QP' \cdot U_x, QP' U_y)$で表わされる。

そこでP'の新座標（X，Y）は次のように求められる。

$$X = [(\xi - q_x)\cos\beta - (\eta - q_y)\cos\alpha] / \sin\gamma$$

$$Y = (\zeta - q_z) / \sin\gamma$$

X軸を決定する二平面が同一平面となった時$\sin\gamma = 0$となり，上の変換式は使えない。この場合にはX軸は投象面πとπに垂直な面$\eta - q_y = 0$との交線とする。y成分の正方向を上にとればX軸の正方向は右向き，Y軸の正方向は上向きとなり，変換式は次のように表わせる。

$$X = [-(\xi - q_x)\cos\gamma + (\zeta - q_z)\cos\alpha] / \sin\beta$$

$$Y = (\eta - q_y)\sin\beta$$

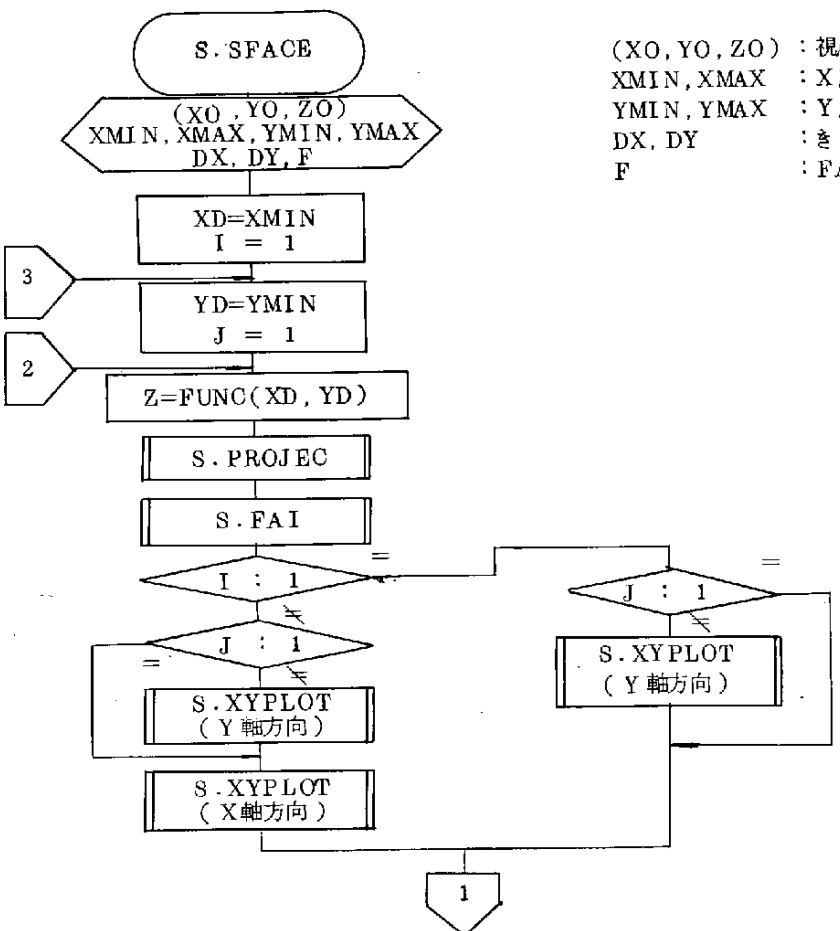
$\sin\gamma \neq 0$の時には，3次元でのz軸は投象面でのY軸と一致する。

3.2 Subroutineの構成

2変数関数の透視図を画くルーチンは，サブルーチン形式をとりサブルーチン名はSFACEである。ブロックチャートを図 21に示す。この中で更に次の4つのsubroutineを使用している。

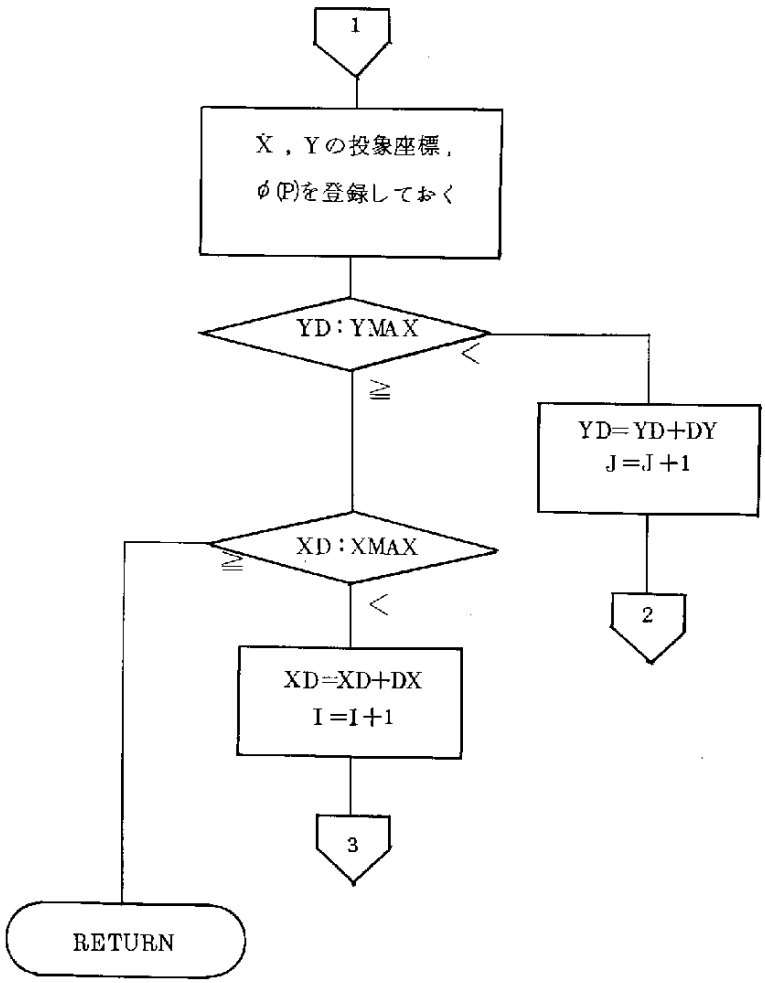
1.	XYPLOT	2点間のPLOT状態を判断する。
2.	FAI	関数φ(P)を求める。
3.	PROJEC	投象面への透視
4.	STPJT	投象面を決定する。

それぞれのルーチンの簡単な説明とブロックチャートを以下に示す。



(XO, YO, ZO) ：視点
XMIN, XMAX ：X座標の範囲
YMIN, YMAX ：Y座標の範囲
DX, DY ：きざみ幅
F ：FACTOR

図 21　SFACEのブロックチャート



1. XYPLOT

○機　能　二点間，PQにおいてφ(P)，φ(Q)を比較することにより，visibleな部分を調べ，pen downでPLOTする。

○使用法　CALL XYPLOT（IP，IQ，J，N）

IP　：　φ(P)

IQ　：　φ(Q)

J　：　y方向へのカウンター　j

N　：　1　P，Qのy座標が等しい時

－1　P，Qのx座標が等しい時

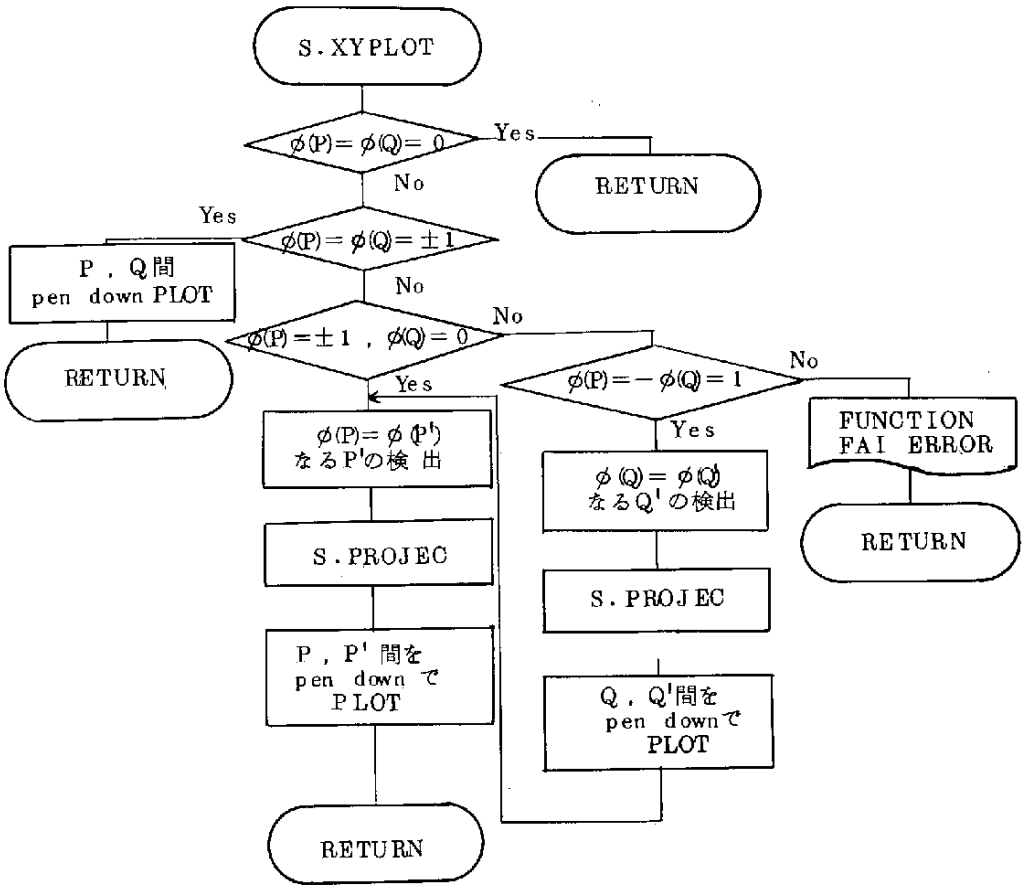


図 22 XYPLOT ブロックチャート

2. FAI

○機　能　点Pにおける関数φ(P)を求める。

○使用法　CALL FAI(XP, YP, ZP, IFAI)

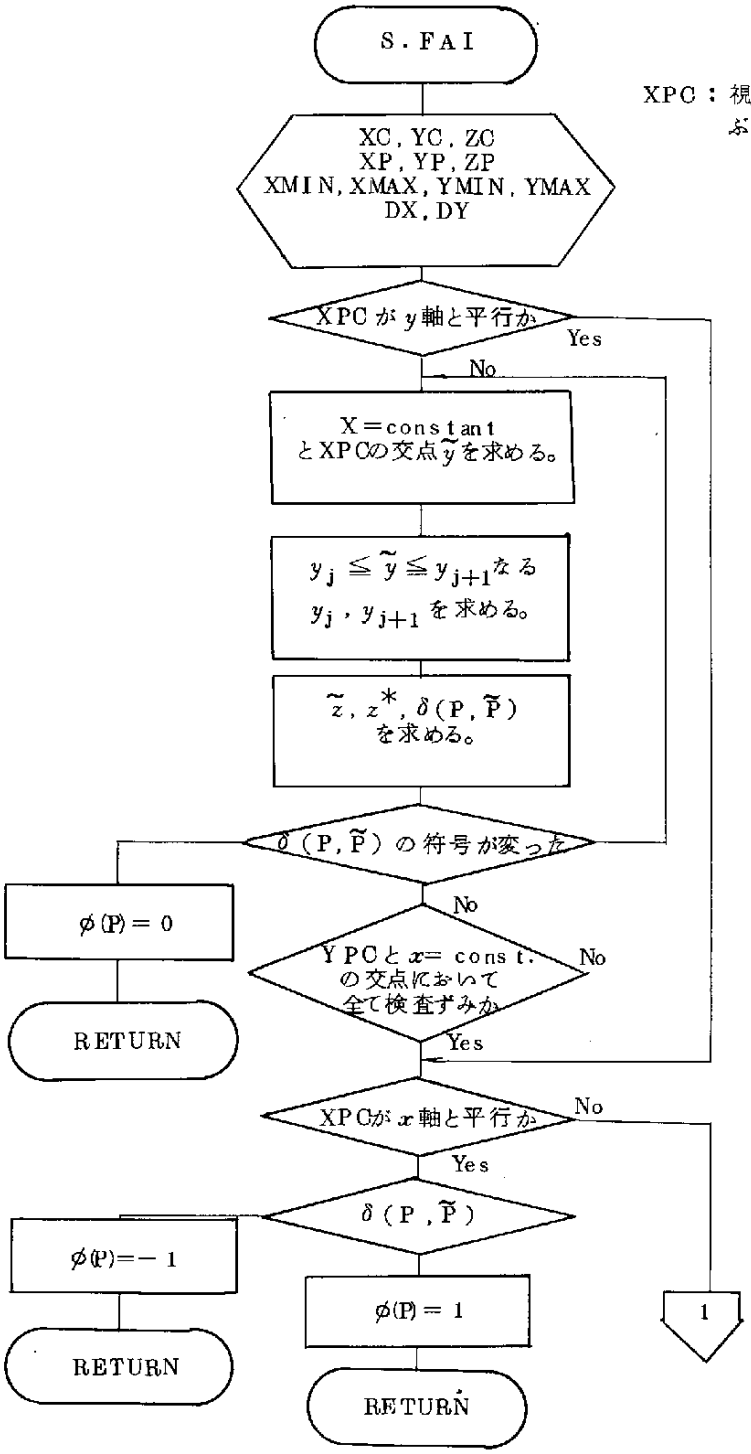
XP　：点Pのx座標

YP　：点Pのy座標

ZP　：点Pのz座標

IFAI：点Pにおけるφ(P)

(−1，0，1)



XPC：視点と点Pを結ぶ線分

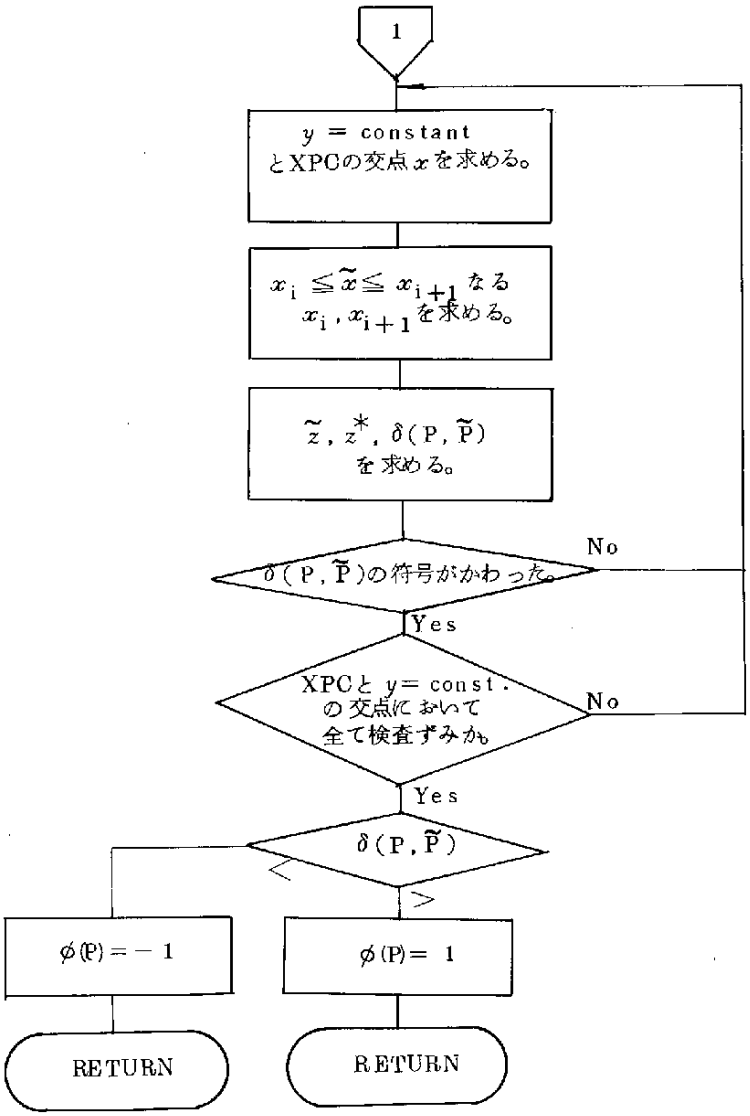


図23 FAIのブロックチャート

3. PROJEC

○機　能　3次元座標を投象し，2次元座標に変換する。投象面はSTPJTで決められる。

○使用法　CALL PROJEC(X, Y, Z, XX, YY)
X, Y, Z 3次元での点の座標　　XX, YY 投象面上の2次元座標

○注　意　SUBROUTINE PROJEC の中に投象面を決めるためのENTRY STPJTをもつ。

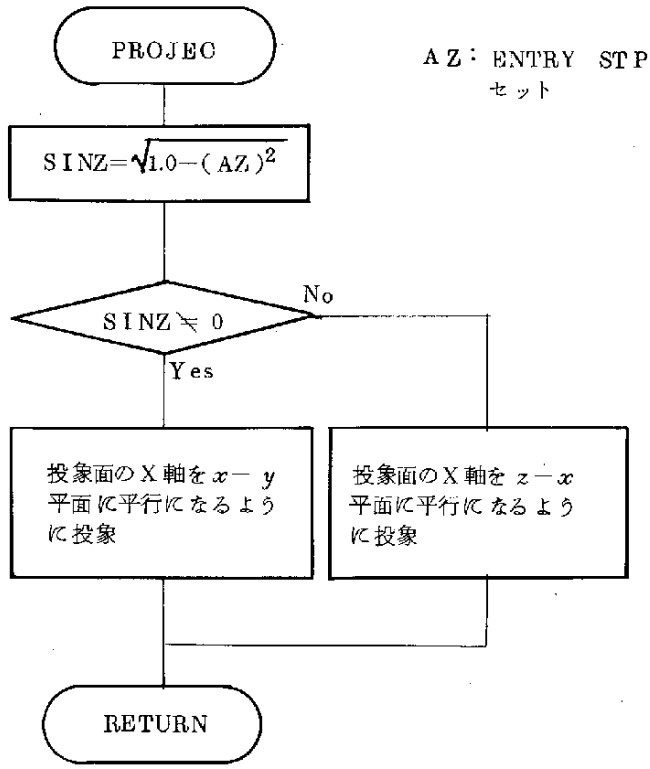


図24　PROJECのブロックチャート

4. STPJT

○機　能　視線に垂直に投象面をとる。

○使用法　CALL STPJT (CX, CY, CZ, AX, AY, AZ, D)

CX：視点のx座標

CY：視点のy座標

CZ：視点のz座標

AX：視線のx方向への方向余弦

AY：視線のy方向への方向余弦

AZ：視線のz方向への方向余弦

D　：視点と投象面との距離

○注　意　SUBROUTINE PROJEC の中でENTRY STPJTをもっている。

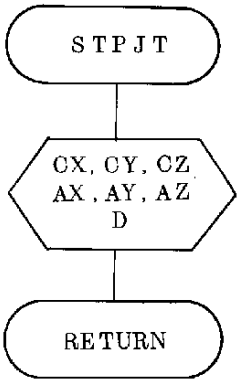


図25　STPJTのブロックチャート

3.3 使用法

描きたい関数をFORTRANの関数サブプログラムとして定義し，サブルーチンSFACEをFORTRANのCALLで呼びだす。

1) 関数サブプログラム

1. FUNCTION名はFUNCとする。
2. 関数は1価関数であること。
3. 関数の絶対値が無限大になる場合にはupper limit, lower limitを定め，limitを越えた時にはFUNCの値をupper limitあるいは，lower limitとする。

```
FUNCTION FUNC(X,Y)
ULIM= ZU
DLIM= ZD

IF(FUNC.GE.ULIM) FUNC=ULIM
IF(FUNC.LE.DLIM) FUNC=DLIM
RETURN
END
```

2) サブルーチン　SFACE

```
CALL SFACE (XMIN,XMAX,YMIN,YMAX,DX,DY,XO,YO,ZO,AX,AY,
```

AZ, D, F)

XMIN, XMAX 描きたい x 座標の領域

YMIN, YMAX 描きたい y 座標の領域

DX, DY x 方向, y 方向のきざみ幅

XO, YO, ZO 視点の座標, ただし XO $\neq$ 0.0

AX, AY, AZ 視線の方向余弦

D 視点と投象面との距離

F factor

注意 1) 領域, きざみ幅は関数の複雑さを考慮して与える。

2) factor は視点の位置, 視点と投象面との距離から投象面に投象される大きさを予測し, factor を与える。

3.4 実 例

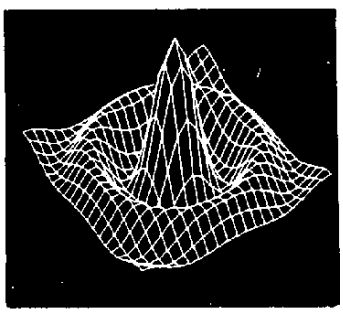


図 26

1) ステップ数 300 FORTRAN ステートメント

2) SIZE 15K WORD (36 bit/WORD)

一般的に計算時間は, きざみ幅に逆比例する。すなわち, きざみを 1/n にすると計算時間は約 n 倍となる。図形の複雑さと計算時間の兼ね合いを考慮した上で, できればきざみは細かい方が正確な陰れ線消去がおこなわれる。

参 照 文 献

第1章に関しては

(1) ユタ大学 computer science学部の Note CS551 No. 7/December, 12, 1969

を参照しており,

第2章は

(2) R. Galimberti and U. Montarari

An Algorithm for Hidden line Elimination

ACM vol 12／Number 4／April, 1969

(3) Arther Appel

The notion of quantitative invisibility and the machine rendering of solids Proceedings A.C.M. National Meeting, 1967.

(4) Philippe P. Loutrel

A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedera IEEE Transactions on Computers vol C-19／Number 3／March, 1970.

第3章は

(6) B. Kubert, J. Szabo and S. Giuliert

The Perspectine Representation of Functions of Two Variables

ACM vol 15／Number 2／April, 1968.

などの論文を参照している。

第III部　アプリケーションプログラム　2

今日の高度な能力を持つ計算機の応用分野の一つとしてＣＡＤ（Computer Aided Design）があげられる。

その中の一つである電気回路設計の分野においては，回路図，動作特性表示曲線，配線図，部品配置図などが主な処理対象である。これらのほとんどは図形情報であり，従来の方法で，パンチ・カードとラインプリンタを用いて入出力していたのでは，あまりにも情報量がぼう大となり，かつマン・マシン間のインタラクションに円滑さを欠いてしまう。

このような意味から設計者と計算機との協同作業の接点に，グラフィック・ディスプレイを用いて，図形情報のやり取りをおこなおうとするのは当然のことであり，今後もますますこの方法が発展していくことであろう。

電気回路の設計でディスプレイ装置を利用する場合は，大別して，

1) 回路の電気的設計

2) 部品，配線の適正配置

の二つに分けられる。

いづれの場合にも，そのプロセスとしては図1のような手順をとり，それらのシステムをより効果的に運用するには，

1) 使いやすく，わかりやすい図形処理言語
2) 複雑な情報間の関連をまとめるデータ構造
3) 目的にあった適切なアルゴリズム

} が不可欠である。

この第Ⅲ部は

これらの電気回路におけるＣＡＤへの応用研究の一部として，簡単な電気回路の過渡現象解析を行う試作プログラムについての報告である。

なお，このプログラムは当センター職員の指導のもとに，東京電機大学，大学院学生等の実習として作成されたものである。

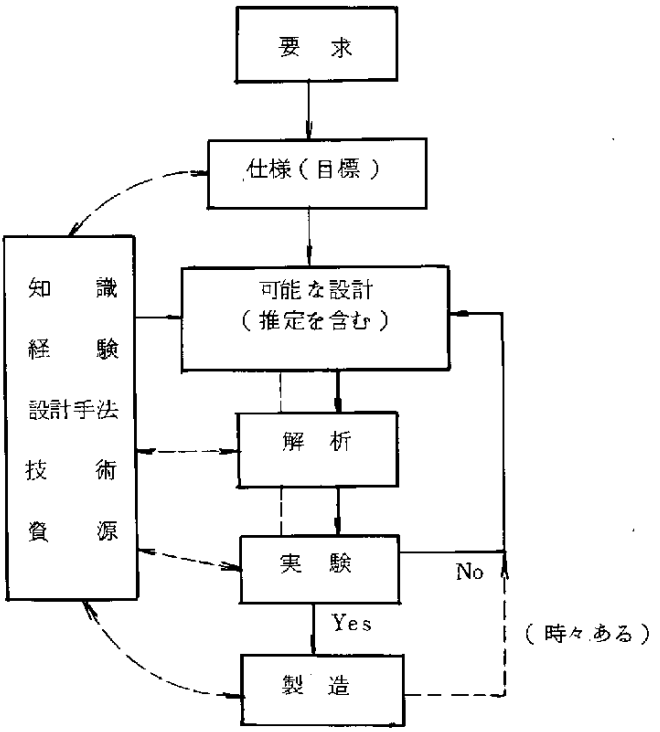


図　1

第1章　回路解析プログラム

回路解析プログラムは数多く開発されているが国内では，まだ試験段階である。代表的なものを，表1に示す。このようなプログラムで処理できる回路網は，節点数50，枝の数200，トランジスタ数40，ダイオート数80程度である。こういったものはすべて，結線関係，素子種類や定数，電源などを入力し，直流特性，過渡特性，交流特性，伝達関数などを求めるものである。しかし最近では，回路を合成したり，感度最大の回路網を求めるもの等も現われている。

回路設計における技術者の自然言語というものは図形である。したがって，技術者と計算機の伝達手段はgraphicであり，表1に掲げたプログラムはすべてその考えにもとずいている。

表　1

名	CALAHAN	POTTLE	ECAP	CIRCUS*	SCEPTER**
開発者	D.A. Calahan（ミシガン大学）	C.Pottle（コーネル大学）	I.B.M	Dickhant 他 ボーイング社	Mathers. 他 I.B.M
対　象	線形回路	線形回路	線形回路（折線近似非線形含）	非線形回路	
入　力	トポロジーまたは回路の多項式の比	回路のトポロジー	回路のトポロジー	回路のトポロジー	
出　力	伝達関数，極と零周波数特性，過渡特性	伝達関数，極と零周波数特性，過渡特性	直流解（感度を含む）周波数特性，過渡特性	直流解，過渡特性	
言　語	FORTRAN Ⅱ, Ⅳ	FORTRAN Ⅱ,63	FORTRAN Ⅱ, Ⅳ	FORTRAN Ⅳ	
特　徴	回路のトポロジーから多項式の比の形で回路関数を求める。トポロジカルな手法による。	同　左 状態変数手法による。	数値的な回路関数を求める。節点方程式解法が中心	状態変数法によるトランジスタ，他各種の半導体素子モデルを入れることができる。	
備　考		ハードコピー可能	IBM 2250 グラフィックコンソール使用	＊　NET−1 から進化 ＊＊ PREDICT から進化	

第2章　DDAを用いた回路解析プログラム

2.1 DDAの原理及び使用法

DDA(Digital Differential Analyzer)は，微分方程式を能率よく解くのに適したディジタル演算回路で，変数の入出力や，やり取りがすべてインクリメンタル(増分)の形で行なわれるため演算の速度が，速いこと，演算結果の図式表示が容易であることなど，グラフィック装置用の演算方式としては特に優れた性質をもっている。

DDAは，積分器，定数掛算器の線形要素，サーボ要素，ディシジョン要素等の非線形演算要素からなる。これ等を組み合わせて与えられた微分方程式と等価な回路を記述し(これをMappingという)数式上の数値と計算機内の数値との関係を対応づけた上で(Scalingという)Mapと等価なルーチンをプログラムして計算機にかける。各要素の数値は与えられた方程式に拘束された変化をたどり，適当な場所から出力を取り出してタイプアウトまたはプロットすれば方程式が解けたことになる。

演算要素の中でも基本となるのは積分器であって，他はその変形とみることができる。HITAC 8811 システムでは演算要素をハードウエアとして組み込むことはしていなく，一般のH－8811処理命令(GPC 命令と称す)の他に，DDA 演算を行なうに適した特殊命令(DDA 命令と称する)をいくつか用意している。これらの命令は，DDA 演算を単純な基本動作に分解した個々の機能を受持っており，したがって，演算要素は必らず数個の命令の一定の順序の組み合せによって表わされ，さらに一般命令の併用により，従来の DDA より一層フレキシブルな応用が可能となっている。

2.2 DDAによる微分方程式の解析

DDA を用いて微分方程式の解を求める時，通常次のような過程をたどる。

(1) 与えられた微分方程式を微分形による表現(インクリメンタル表現)に変換する。

(2) 微分形からマッピングに変換する。

(3) 各インクリメントとレジスターのスケールの関係を決定する。

(4) 解の出力を表示する方法を考える。

(5) (2)，(3)から対応するハードウェア命令を作り出す。

(6) 初期値を与えて実行する。

2.3 回路解析への応用

前記のようなＤＤＡの機能をグラフィック・システムに加えることにより次の様な回路解析のためのＣＡＤシステムが可能である。

(1) ＣＲＴ画面に於て任意の構成の回路を組み立てる。

(2) 入力端，出力端を指示する。

(3) (1)の回路構成を微分方程式に変換する。

(4) (3)で表わされた微分方程式をマッピングして変換しＣＲＴに表示する。

(5) 入力端に加える信号を定義する。

(6) マッピングを命令群に内部変換する。

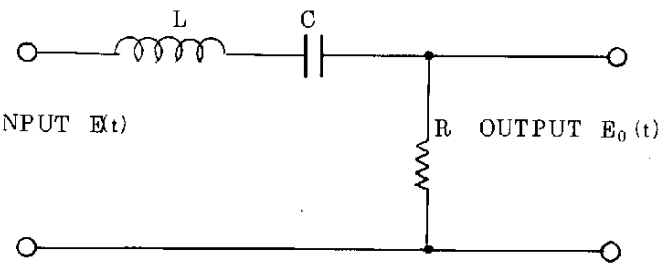
(7) 回路定数，初期値を代入する。

(8) 実行を指示し出力をＣＲＴ画面に表示する。

ここでは，上記のシステムの基本的な調査という意味で，限られた回路構成（ＲＬＣ直列回路）を対象に実験的試みを行った。

対象とした回路はＲＬＣ直列回路で２階の線型微分方程式として表わされるものである。

入力端，出力端は図の様に定める。



今入力端にＥ(t)なる電圧を印加した時出力端に得られる電圧$E_0(t)$は次の様に表わされる。

$$E_0(t) = R\dot{i}(t) = E(t) - L\frac{di}{dt} - \frac{1}{C}\int i\,dt$$

この実験プログラムは入力電圧波形，回路定数，初期値等を変化した時の系の振舞いを解析することを目的とする。

Ｒ，Ｌ，Ｃの初期値は，ＣＲＴ画面より与えることが出来る。また入力電圧波形としては線型要素として階段波，ＳＩＮＥ波，非線型要素として三角波，矩形波を用意している。

回路構成をマッピングしたものを図２に示す。

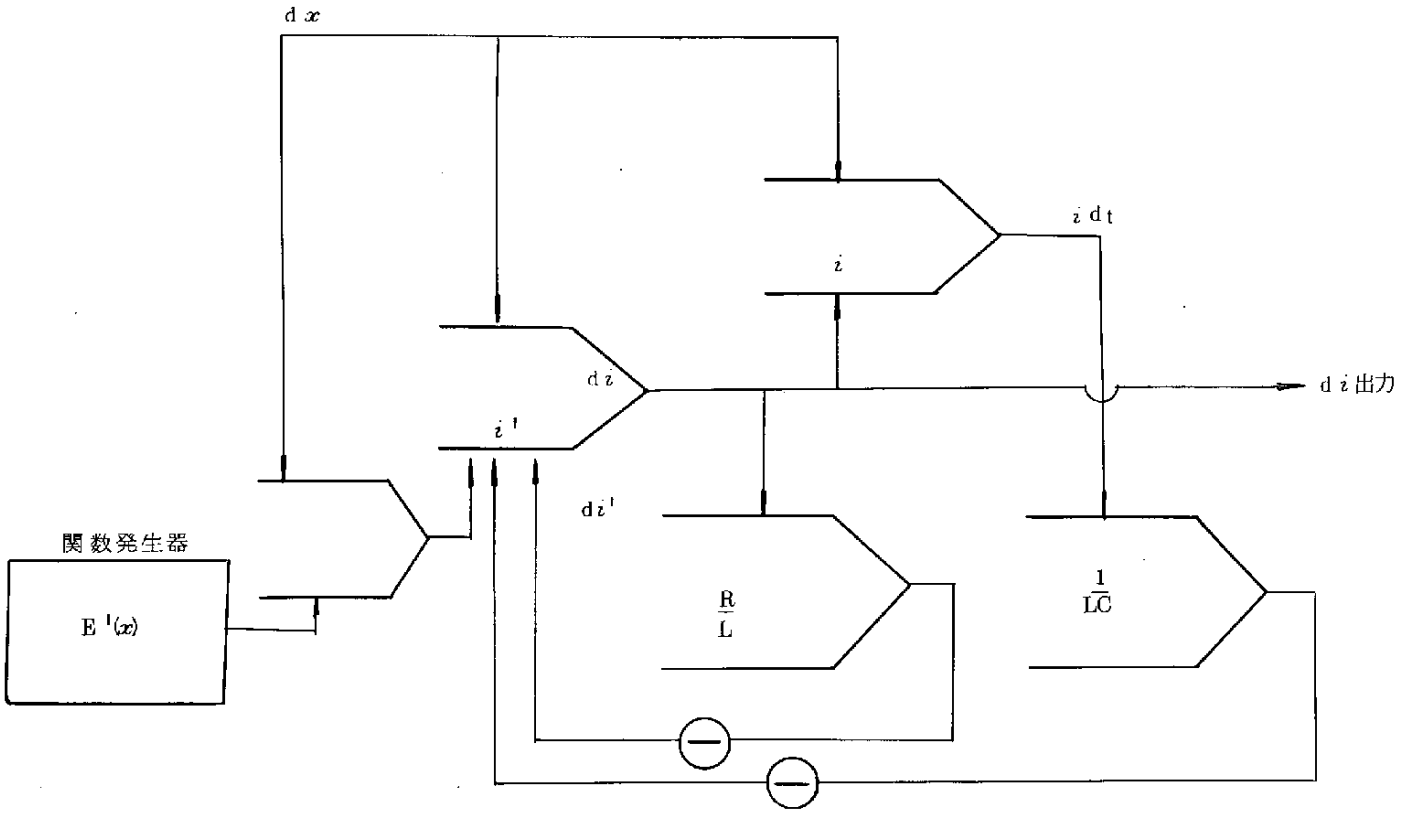


図２　R－L－C回路のDDAへのマッピング

2.4 試作プログラム

〔入力法〕

素子やその値，あるいはその他の機能の大部分は，ライトボタンにより入力する。また，一部の機能については，ファンクション・キーを使用する。また，CRTの表示面は表示領域とコントロール領域とに区分する。（図3）

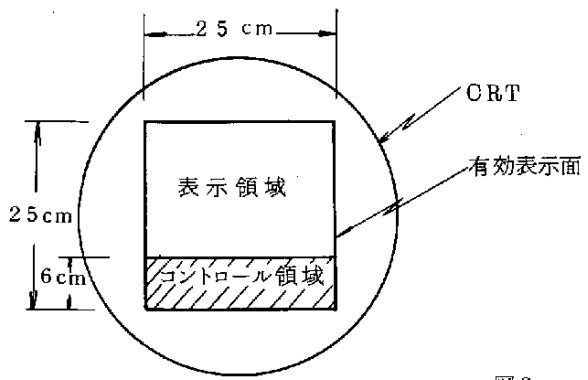
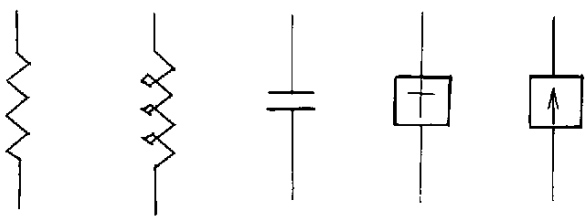


図3

ライトボタンの種類は次のようなものを用いる。



POSITION VALUE

NODE-DEF

ここで"VALUE"をポインティングすると

1 2 3 4 5 6 7 8 9 0

SINE PULSE DC

RESET ANALYZE RETURN

なるライトボタンが現われる。また，出力表示から回路を構成する画面にもどすために

CIRCUIT

を使用する。

〔素子の種類〕

簡単なR，L，Cの直列回路表示に必要なR（抵抗），L（インダクタンス），C（キャパシタンス），E（電圧源），の四種類に加え，将来のためI（電流源）をも用意しておく。

また，素子と素子を結合するために必要なリード線も，素子と同じ長さのものと，半分の長さのものとを用意しておく。（表２）これらの素子をディスプレイ画面に描く場合，ストローク数が出来るだけ少ないものが望ましく，また，普通用いられている素子のイメージであることが好ましい。

表 2

種　　類	記　　号	シ ン ボ ル
リード線（短）	ＬＳ	
リード線（長）	ＬＬ	
抵　　抗	Ｒ	
インダクタンス	Ｌ	
キャパシタンス	Ｃ	
電　圧　源	Ｅ	
電　流　源	Ｉ	

〔節点の登録について〕

節点（ NODE ）を決定するのに使用するトラッキング・マーク（ TRM ）は，図４に示すように中心の点と正八角形の頂点に分布する８個の点からなる。トラッキング操作をすると，ＴＲＭの中心がライトペンの中心と一致するようにＴＲＭは移動するが０はライトペンの影になるため，TRM の現在位置としてユーザプログラムに知らされるのは，０以外の点，例えば左上の＃４の位置である。

将来データ構造を考えたり，素子を結合した時点で何等かの機能をもたせたり，何等かの処理をほどこすためには，素子を結合したという事をシステムに認識させる必要がある。又，最初の考えでは，図５に示すように，TRM の基準点と素子の端点とを手で一致させるつもりであったが，これを完全に一致させる事は非常にめんどうであり，完全に一致させないと図６の

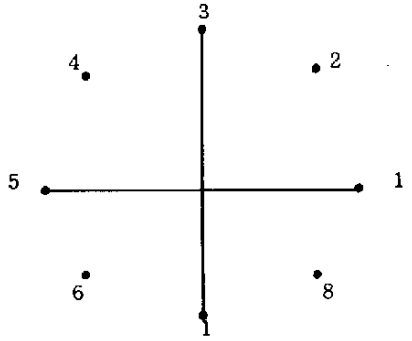


図　4

ようになり見た目にも悪くなる。したがって，次のような処置をとっている。

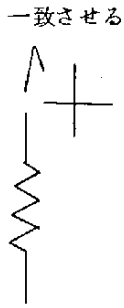


図 5

図6

〔1〕 先ず，最初の素子をＴＲＭ 表示する。〔図７―(a)〕

〔2〕 次の素子を表示するために，素子の端点を中心にして半径Ｒの円内部にＴＲＭの基準点をもっていくと，素子は，その端点に接続されたことを認識し，表示を可能にする。〔図７―(b)〕もしこの円の外部であると，これは接続されたことにならず，素子の表示は不可能となる。〔図７―(c)〕この場合，再度ＴＲＭのポジショニングを行なう。

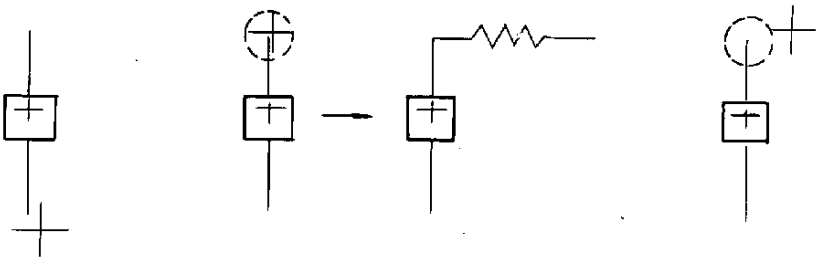
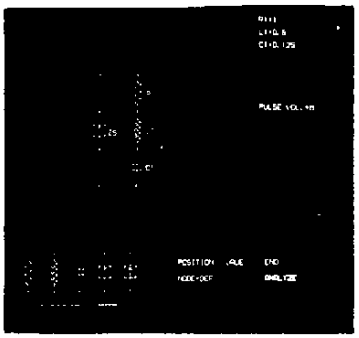
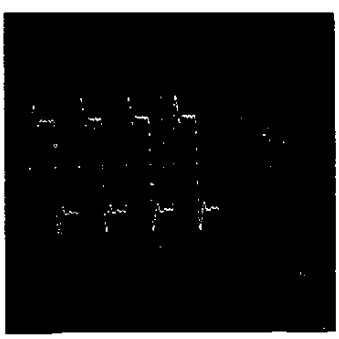


図 7

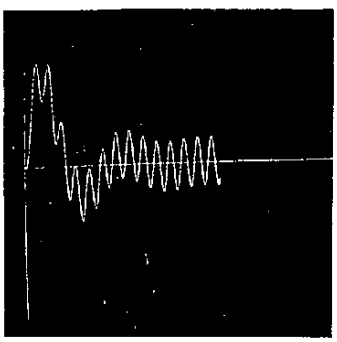
(1) 回路の組立



(2) 回路構成終了



(3) 三角波を入力とした出力電流波形



(4) サイン波を入力とした出力電流波形

2.5 DDAの限界と問題点

DDA（Digital Differential Analyzer）は，手続きが非常に複雑となる微分方程式の解析を，簡単な式の変形によって，容易にこれを解くことができ，かつ，入出力が量子化されているため，その出力を直接図形表示に利用し易いなどの利点を持っているが，反面，原理上の制約から，実際の応用時に問題を残しやすい。

最大の障壁はDDAの演算が，16 bit固定小数点で行なわれるために，数値の有効桁数が非常に小さいことからくる制約である。

第1に数値の取り得る範囲が $2^{15}-1 \sim -2^{15}$ なので，積分の精度を上げるためには，関数毎にその最大最少の値を考慮した上でのスケーリングをほどこさねばならない。しかし，いくつかの積分器を直列に連ねた場合── 云い換えると，変化の範囲が大きい関数を扱うような場合は，微少な変化を生かすスクーリングはほどこせない。

第2にマッピングする関数が，あらかじめ予測がつかない場合 ── 例えば，回路解析システムのように，ユーザーによって，外からさまざまな回路を与えられて，それをマッピングするような場合 ── には，変化途上の数値の範囲を予測して，自動的に最適のスクーリングをほどこすことは，至難である。

第3に，関数毎に最適のスクリーングがほどこせたとしても，今度は，その出力を逆スクーリングしなければならなくなってしまう。

第4に，精度を上げるためにスケールを下げて演算を行なうと，計算機による一般の数値計算の概念を破る程の処理時間がかゝる。経験的に結論づけると，DDAのスピードは，プロッターのペンを動かすには，ほゞ満足の行くものであるが，CRTのビームを振らせるような，リアル・タイム性の要求される場合には，我慢のならない遅さである。我々の試作したプログラムの，RCの直列回路では，出力の取り方に問題が生ずる。又，RLCの直列回路を考えて見ると，電源が正弦波や直線に一定変化するといった，時間に比して，ある程度ゆるやかな変化をするものであれば，解析出来るのだが，パルスのような，急激な変化分のある電源の入った回路の解析は難かしくなる。

例えば，RLC直列回路にf(t)なる電源が印加された場合，DDAで解析するためには，回路に流れる電流を i とすると回路方程式

$$L\frac{di}{dt} + Ri + \frac{1}{C}\int i\,dt = f(t)$$

を微分して，

$$L\frac{d^2 i}{dt^2} + R\frac{di}{dt} + \frac{i}{C} = f'(t)$$

とし，両辺にdtを掛け

$$(di)' = \frac{1}{L}\left(f'(t) - \frac{i}{C}\right)dt - Rdi$$

という形に変形して，積分器でマッピングをして，解くのであるが，上式を見ても分る通り，電源f(t)を一度微分しているため，パルスのような関数がf(t)であると，ある時点でf'(t)は∞となるが，16ビットのレジスタには到底∞なる数値は入れられないので，DDAでは解析できなくなる。

このようなことから，図形でRLC等により適当に回路を作り，これを自動的にDDAのマッピングをして演算をさせ，解析させるような，完全な回路解析は，非常に難かしい。

DDAを利用してみて，以上のような経験を得たが，やはり数値の有効精度などの問題から現時点では汎用的な利用は困難と思われる。

第2編　JUMPS(JIPDEC UNIVERSAL MATHEMATICAL PROGRAMMING)

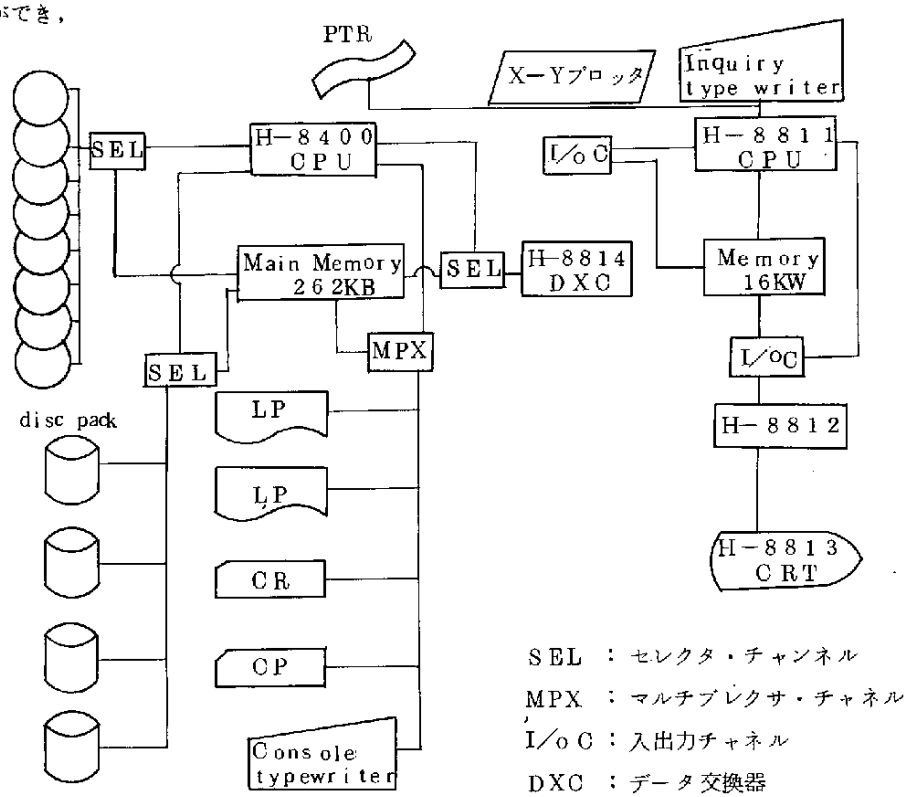
目　次

第1章　JUMPSの概要

1.1　JUMPSの構成

ＪUMPＳは統計解析，模型作成，シミュレーション等の機能をもつ会話形のシステムであって，利用者はＪUMPＳが用意しているコマンドを適当に組合せて使うことにより求める結果を得ることができる。

更に大きな特長としてはグラフィックディスプレイ（ＣＲＴ）を入出力として利用することができ，



各種の情報や計算結果をグラフで表示することによって，人間の理解をより助け，ライトペン，ファンクションキーなどを使用して，よりインタラクティブに模型作成や処理の指示を与えることができる。

ＪUMPＳ を実現するためのコンピュータのシステム構成は，図1.1で示すようにHITAC－8400とHITAC－8811グラフィックコンピュータを接続したものである。

この機器構成のコンピュータに対して，日本情報処理開発センターにおいては，45年度事業として「ディスプレイシステムの研究開発」の一環として，CGOS（Comprehensive Graphic Operating System）と呼ぶ汎用グラフィックオペレーティングシステムを開発したので，このCGOSのコントロールの下でＪUMPＳ を動かすこととした。

CGOSの詳細については「ディスプレイシステムの研究開発」の報告書にゆづることとして，こゝでは概説のみにとゞめる。

CGOSにおけるコントロールプログラムシステムの特徴はH－8400プロセッサ内のGJMON（Graphic Job Monitor）とH－8811プロセッサ内のGCH（Graphic Communication Handler）の両プログラムに代表される。

GJMON及びGCHは，CGOSのシステムイニシエーション時にそれぞれ主記憶装置にローディングされ，DOS Ⅲ エグゼクティブ，H－8811 エグゼクティブ等の制御プログラムと協業して，CGOSの運用を管理する。

CGOS のコントロール・プログラムシステムは従来H－8400付属のコンソールタイプライタで行っていた，DOS オペレーションのほゞすべてを，H－8813グラフィック・CRT・コンソールからのオペレーションで代行することができる。

GJMON は，H－8400プロセッサ内で，DOS Ⅲ エグゼクティブの下で，スレーブモードタスクとして常駐し，次の機能を果たす。

1. ＤXC経由による，H－8811プロセッサとの間の相互データ転送
2. ＤXC経由で受け取ったメッセージデータの解析
3. H－8813グラフィック・CRT・コンソールから要求のあったプログラムの起動，及びモニタリング
4. グラフィック・コマンドリクエスト
5. グラフィック・アプリケーションプログラムとの間の相互データ移送

GCHは H－8811プロセッサ内に常駐し，次の機能を果たす。

1. ＤXC経由によるH－8400 host computer との間の相互データ転送
2. ＤXC経由で受取ったキャラクタ・メッセージの表示
3. ＤXC経由で受け取った濃縮図形データからディスプレイコマンド列への翻訳，表示

4. メッセージ・インディケータの表示と，オペレーション・メッセージ・ストリングの編集と転送

5. コマンド・コンパイルされた図形データに対するライトペンのピッキングの受付けと，ブリンキング処理

6. コマンド・コンパイルされた図形データのエッヂ・シザリング処理

7. ファンクションキー（０～３１）の割込み制御

8. ダイアル（１，２）の割込み制御

9. トラッキング・クロスの表示

10. ＣＲＴ表示データのＸ－Ｙプロッタ上へのハード・コピー

ＣＧＯＳとＪＵＭＰＳの関係

ＣＧＯＳは，概説に述べたように各種機能を持っているが，ＪＵＭＰＳはＣＧＯＳの下にあって，その機能を使いながら動くのである。使用する主な機能は次のとおりである。

1. ＪＵＭＰＳシステムのローディング

2. ＪＵＭＰＳシステムをオーバーレイして使用

3. 文字及びグラフの表示

4. 文字入力及び，文字列のピッキング

5. Quit 機能，Restart 機能

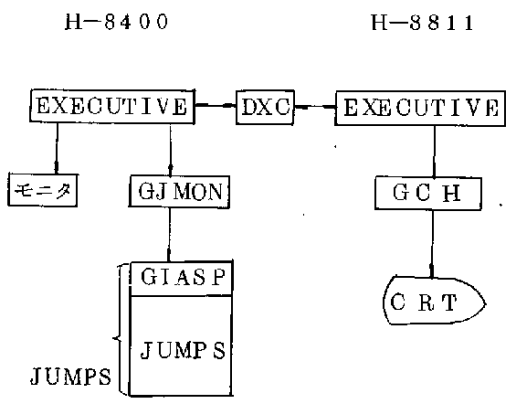


図１－２　ＣＧＯＳとＪＵＭＰＳの関係図

これらの機能は図１－２で示されるような形でＪＵＭＰＳの各コマンドを通じて使用されることになる。

ＧＪＭＯＮの下で動くＪＵＭＰＳは，ステージとステップという２つの概念からなりたっている。ステージは，一連の手法をまとめたものとして

1. データ定義ステージ

2. データ加工ステージ

3. データ分析ステージ

4. 変数間の分析ステージ

5. 模型作成ステージ

6. 推定と評価ステージ

7. シミュレーションステージ

の７つのステージがあり，利用者の目的によりどのステージから処理を始めてもよいが模型作成やシミュレーションのステージにおいては先行するステージを要求することがある。この場合にはその要求を満足するような処理をおこなってから再び処理をおこなえばよい。

ステップは，各ステージに１個以上あり各手法と対応した実行の最小単位ということができる。

これらの各ステージもまたステップもともにモジュール化されており，利用者が不必要と思うステージあるいはステップはシステムジェネレーション時にはぶくことができる。また必要となった時点で，いくつかのモジュールを追加することも可能である。この方法に関しては第四章を参照されたい。

1.2 コマンドの概念

ＪＵＭＰＳ において，システムとその利用者との情報の授受は，利用者側からのコマンドという形によっておこなわれる。コマンドは，使用者がＪＵＭＰＳ に対してサーヴィスを要求する単位であって，２つの種類がある。すなわち，どのステージを選択するかというＪＵＭＰコマンドと選択されたステージ内のどのステップを選ぶかという各ステップに対応するコマンドである。

ＪＵＭＰ コマンドによるステージの選択は次のとおりである。

ＪＵＭＰ　ＤＤＥＦ；	データ定義ステージにはいる。
ＪＵＭＰ　ＥＤＩＴ；	データ加工ステージにはいる。
ＪＵＭＰ　ＤＡＴＡ；	データ分析ステージにはいる。
ＪＵＭＰ　ＣＯＲ；	変数間の分析ステージにはいる。
ＪＵＭＰ　ＡＵＴＯ；	模型作成ステージにはいる。
ＪＵＭＰ　ＥＳＴＩＭＡＴＥ；	推定と評価ステージにはいる。
ＪＵＭＰ　ＳＩＭＵＬ；	シミュレーションステージにはいる。
ＪＵＭＰ　ＥＮＤ；	ＪＯＢ の終了あるいは中断時におけるあと処理をおこなう。

ＪＵＭＰ コマンドは，ステージに対して以上８つの分枝を持っている。

ステップに対応するコマンドについては第二章を参照されたい。

1.3 ファイルの概要

JUMPS において，必要となるデータは，必要な時点で即座に提供されねばならない。そのため，諸データは，ダイレクト・アクセス装置上のデイスク・パックに格納されている。これらのデータで構成されているファイルには，つぎのようなものがある。

1.3.1 オリジナル・ファイル：

原始データのファイルであり，ファイル形式は，キーワードによるアクセス，アクセス時間の短縮，ファイルメンテナンスの容易さなどの理由からインデックスド・シーケンシャル（ Indexed Sequential ）ファイル形式をとっている。

1.3.2 セレクト・ファイル：

オリジナル・ファイル上から抽出されたデータや，加工を施したデータのファイルであり，ファイル形式は，更新（アップデイト）作業が簡単に行えるランダム・アクセス（ random access ）ファイル形式をとっている。またこのファイルは，利用者が消去を希望するまで保存される。

1.3.3 セーブ・ファイル：

グラフィック・ディスプレイ上に表示した図形データのファイルであり，ランダム・アクセス・ファイル形式をとっている。

1.3.4 ワーク・ファイル：

各ステージで作成された各種テーブル用データのファイルであり，ランダム・アクセス・ファイル形式をとっている。

各ファイルともキーによってアクセスできるようになっており，実際の方法については，後述のユーティリティ・ルーティンの項を参照のこと。

つぎに，オリジナル・ファイルとセレクト・ファイルとの相違であるが，前者が，特定ジョブの目的だけに限定されないファイルであり，あらかじめ作成されているのに比べて，後者は，特定ジョブの目的だけに限定されたデータのファイルであり，ジョブの進行とともに作成されるものである。これらのファイルの性質は，JUMPS システムの効率のよい運用を図るために考えられたものである。

また，JUMPS においては，システムと利用者との会話媒体にグラフィク・ディスプレイを用いているが，グラフィク・デイスプレイの欠点は会話内容の履歴が残らないこと

である。このため，全ての会話内容を，ＨＩＴＡＣ－８４００ のラインプリンターに出力するようにしてある。詳細については，個々のステップで説明してあるので参照されたい。

第2章　コマンドの説明

この章では，JUMPSにある7つのステージと，そのステージ内で使用できるコマンドについて述べる。

2.1 データの定義

このステージにおいては，のちのステージにおいて式の自動作成をおこなう場合の前準備をおこなう。すなわち，使用する変数が内生変数であるか外生変数であるか，あるいは自動選択のさいに選択する変数に高いプライオリティをつけるかどうか等を指定する。

JUMPSにおいては個々の変数は

セクター　→　グループ　→　個々の変数

という階層関係を持っている。

グループというのは，例えば国民所得と可処分所得と個人業種所得などは，その動向がきわめて類似しているので，これらのものをモデルの変数として含む場合にはパラメータ推定においてmulti collinearity が発生する可能性がありこれをのがれるためのものである。すなわち，その動向がきわめて類似しているものは同一グループの指定をおこなうことにより，変数の自動選択をおこなう場合には，同一グループ内の変数は一本のモデル式には1個しか選択されないことが保障される。

セクターというのは，グループのあつまったもので種々の変数に対して同一処理をおこなう場合に，それらの変数を同一セクターとして登録しておけば，セクター番号を指定することによりそのセクターに属する全変数を指定したのと同じになる。

このステージにとびこむためには

JUMP　DDEF；

コマンドを使用すればよい。

2.1.1 DEFINEコマンド

DEFINE　〔セクター番号〕；

セクター番号：0～9までの整数で指定のない時は0とみなす。

このコマンドによってセクター番号を指定し，以降に次の3つのサブコマンドを適当

に使用する。

ENDO サブコマンド

ENDO（Var 1, Var 2, …,（Var 8, Var 9, …）, …, Var n, …）;

このサブコマンドにより，カッコ内の中で指定された変数が内生変数であることを指定する。もしグループを指定したい時には，さらにカッコでくくって同一グループであることを指定すればよい。

- Var n ：変数名あるいは，変数名のあとにカッコでくくったプライオリティ指定を付加する。このプライオリティは0～2までで，数字が大きい程プライオリティは高い。指定されていなければプライオリティは0とみなされる。このプライオリティは模型の作成において，モデル式の中にとり入れる優先順位を決定するものである。

- EXOサブコマンド

EXO（Var 1,（Var 2, Var 3,…）, ……, Var n, …）;このサブコマンドにより，カッコの中で指定された変数が外生変数であることを指定する。他はENDOサブコマンドと同じ。

- UNDサブコマンド

UND（Var 1,（Var 2, Var 3）,（Var m, Var n,…）,…）;カッコ内の変数が内生変数であるか外生変数であるか使用者側によくわからない場合，このコマンドでシステム側にそのどちらにするかの決定を一任することを指定する。他はENDOサブコマンドと同じ。

例

```
DEFINE 2 ;
ENDO (X1, X2, (X3, X5));
EXO  (X6, (X7, X8, X9));
DEFINE 3;
ENDO (A1, A2(2), A3, (A4, A5));
ENDO (B1, B2, (B4(2), B5(2)));
```

2.1.2 DISPLAYコマンド

データをグラフィックディスプレイに表示する。

DISPLAY Var;

Var：変数名あるいは，変数名のあとにデータの表示開始時点と終了時点をカッコでくくったもの。

出力

○CRTにグラフを表示する。

○ラインプリンターに原データをプリントする。

例

```
DISPLAY CRUDESTEEL;
DISPLAY SMALLS(1965/1, 1970/1);
```

注　このコマンドは，いずれのステージにおいても使用可能である。

2.1.3 DATA コマンド

JUMPS オリジナルファイル中にもセレクトファイル上にもデータが存在しない時に，セレクトファイルにデータを登録するコマンドである。

DATA key , attribute,データ, データ, データ… ;

key ： データをサーチする時に使用するキイ

attribute ：

データ開始年

データ開始月日

データ終了年

データ終了月日

等をカンマで区切って書く。くわしくは，3.1ファイルの作成および保守のスイッチ情報についてを参照のこと。

データ：実定数を必要な個数だけカンマで区切って書く。

例

1960年から1963年までの年次データAAを登録する場合を考える。

```
DATE AA, 1960, 0, 1963, 0, 7, 0, 0, 1, 120, 130, 150, 170;
```

この場合

1960 → データ開始年を示す。

0 → データ開始月日であるが年データであるので0とする。

1963 → データ終了年を示す。

と算術要素から成るものである。算術演算子は＋，－，＊，／，＊＊の5種類である。

算術要素の一般形をバッカス記法で記す。

＜定　数＞　::＝　＜整定数＞｜＜実定数＞

＜符　号＞　::＝　＋｜－

＜変数の引用＞　::＝　＜変数名＞｜＜変数名＞(＜整定数＞)｜＜変数名＞(＜符号＞＜整定数＞)

＜関数名＞　::＝　ABS｜EXP｜ALOG｜ALOG10｜SIN｜COS｜TANH｜SQRT｜ATAN

＜関数の引用1＞　::＝　＜関数名＞(＜変数の引用＞)

＜関数の引用2＞　::＝　＜関数名＞(＜定数＞)｜＜関数名＞(＜符号＞＜定数＞)

＜1次子　1＞　::＝　＜変数の引用＞｜＜関数の引用1＞｜(＜算術式＞)

＜1次子　2＞　::＝　＜定数＞｜＜関数の引用2＞｜TIME｜(＜算術式 2＞)

＜因　子　1＞　::＝　＜1次子1＞｜＜1次子1＞＊＊＜1次子1＞｜＜1次子1＞＊＊＜1次子2＞｜＜1次子2＞＊＊＜1次子1＞

＜因　子　2＞　::＝　＜1次子2＞｜＜1次子2＞＊＊＜1次子2＞

＜項　1＞　::＝　＜因子1＞｜＜項1＞／＜因子1＞｜＜項1＞／＜因子2＞｜＜項2＞／＜因子1＞｜＜項1＞＊＜項1＞｜＜項1＞＊＜項2＞｜＜項2＞＊＜項1＞

＜項　2＞　::＝　＜因子2＞｜＜項2＞／＜因子2＞｜＜項2＞＊＜項2＞

＜算術式　2＞　::＝　＜項2＞｜＜項2＞＜符号＞＜項2＞｜＜項2＞＜符号＞＜算術式2＞

＜算術式＞　::＝　＜項1＞｜＜符号＞＜項1＞｜＜算術式＞＜符号＞＜項2＞｜＜算術式2＞＜符号＞＜項1＞

COMP コマンドにおける算術式の中には，必らず1コ以上の変数名があって，且つ，この変数名の属性は，左辺のVar 1の属性と矛盾しないものでなければならない。変数名は，すでにファイルの中にあるデータの名前で，これにタイム・ラグを( )でくくったものをつけることができる。

0 → データ終了月日であるが年データであるので0とする。

7 → データの単位を示す。この場合はその他とした。

0 → データが10^nのスケールである時にnを示す。この場合はもとの値とした。

0 → ストックデータかフローデータを示す。この場合はストックデータ

1 → データ属性を示す。年次データであるので1とした。

12.0, 13.0, 15.0, 17.0 → データ

2.2 データ加工

このステージにおいては，オリジナルファイルにあるデータ，あるいは既にセレクトファイルに登録されているデータの加工をおこない，その結果をセレクトファイルに登録する役目をもつ。

このステージに入るためには

JUMP EDIT ;

コマンドを使用すればよい。

2.2.1 COMP コマンド

算術式の演算の実行を行なうコマンドである。

COMP Var1〔(P$_1$, P$_2$〔, 属性〕)〕=算術式 ;

Var 1：指定変数であって右辺の算術式の演算結果がこの名前でセレクト・ファイルに登録される。

P$_1$ ：Var 1のデータの開始時点を示すもので，年次データ及び時系列以外のデータに対しては，整数値1コから成り，半期，四半期，月次データに対しては，α_1/α_2 の形である。ここでα_1 は指定しようとする年次を示し，スラッシュ(／)のあとのα_2 は指定しようとしている期又は月を示す。

P$_2$ ：Var 1のデータの終了時点を示すもので形式はP$_1$ に同じである。

属性 ：年(YR)，半期(BI)，四半期(QT)，月(MT)，時系列以外のデータ(AN)のいずれのデータであるかを指定する。

算術式 ：ここでいう算術式は，FORTRAN の算術式に類似したものであるが演算は，すべて実数型のものとする。すなわち，算術式は，算術演算子

例　AA(1960, 1970)＝A1(－1)＋A2(＋5)

COMP コマンド中に書くことのできる定数は整定数であっても，実定数であってもかまわないが，内部的には，いずれの場合も実定数としてとり扱う。したがって，たとえば整定数1は実定数1.0に変換してから演算に用いる。

COMPコマンドにおいて使用可能な関数はABS，EXP，ALOG10，SIN，COS，TANH，SQRT，ATANの9個のみで，これらの引数として使用できるのは，1個の変数名(タイム・ラグがついていてもよい)，あるいは，1個の符号つき，または符号のない定数にかぎる。引数に式を書くことはできない。

キーワードTIMEというのはJUMPSの算術式に特有のものでこれは演算の開始から終了までの期間の各々に対して，開始時点を1.0とし1.0ずつ増えていく数列を意味するものである。

例　AA(1960, 1970)＝A1＋TIME

この例では，AAの1960年のデータは，A1の1960年のデータに1.0を加えたもの，AAの1961年のデータはA1の1961年のデータに2.0を加えたもの，……，1970年のデータはA1の1970年のデータに11.0を加えたものとなる。

（注意）

(1) 左辺の変数Var 1 に期間の指定がない場合には，右辺の変数に共通する最大の期間とその属性とが与えられる。

(2) 左辺の変数Var 1 に期間の指定があって，属性の指定がない場合には，右辺に出現した変数と同じ属性がとられる。

(3) COMPコマンド中に出現した変数どうしの間の属性に矛盾がある場合は演算は行なわない。

(4) 算術式の中のベキ乗の演算で2つ以上のベキ乗演算を続けて書くことは禁止する。

（例1）

COMP　RATIO　＝　CAR／CAR(－1)；

ここで，変数CARが年次データであって1960年から1971年までのデータを持つとすると，このコマンドの実行の結果RATIOとしては，1961年から1971年までの年次データが登録される。

（例２）

COMP　ABCDE（1970／1，1972／12，MT）＝DATAM＊10；

このコマンドの実行の結果，月次データABCDEに対して1970年1月から1972年12月までのデータが得られる。ここでDATAMは少なくとも1970年1月から1972年12月までの内容を持った月次データでなければならない。

2.2.2　EDITコマンド

指定された属性に，データを編集する。

命令形式

EDIT（Var 1，属性）＝Var 2；

Var 1　：編集されたデータにつけられる名前，編集すべき期間の開始時点，及び終了時点の三要素から構成される。又各々の要素はコンマ(,)で区切られる。

属　性　：年データに変換する場合　　YR

半期データに変換する場合　　BI

四半期データに変換する場合　　QT

月次データに変換する場合　　MT

上記四種類の属性に従ったデータ編集が行なわれる。

Var 2　：編集すべきデータの名前。

例

1) EDIT（NAME 1，1950／1，1970／4，QT）＝NAME 2；

（Var 1　／　属性　／　Var 2）

この例は，NAME 2で示されるデータを，1950年第1四半期から，1970年第4四半期まで四半期データに編集し，編集されたデータをNAME 1と名づける。

2) EDIT（NAME 3，1961／1，1971／12，MT）＝NAME 4；

3) EDIT（NAME 5，1955，1965，YR）＝NAME 6；

4) EDIT（NAME 5，1955／0，1965／0，YR）＝NAME 6；

尚 属性と年，期，月との関係は次に示す通りである。

YR　：　0

BI　：　1（上期），2（下期）

QT　：　1（第1四半期），2（第2四半期），3（第3四半期），4（第4

COMPコマンドの例

```
COMP  DDDDD(1958,1965)=POPULATION/POPULATION(-1) ;

   ***  RESULT  ***

THE OPERATED DATA NAME       DDDDD

     INITIAL TIME   ...     YEAR = 1958      MONTH/TERM =  0
     TERMINAL TIME  ...     YEAR = 1965      MONTH/TERM =  0

     DATA ATTRIBUTE                1
               (N.B.  YEAR=1, HALF YEAR=2, A QUATER YEAR=3, MONTH=4, ANOTHER=6 )

     OPERATED DATA NUMBER          8

THE OPERATED DATA

 0.10083838E 01      0.10188389E 01      0.10083971E 01      0.10092907E 01      0.10094814E 01
 0.10102434E 01      0.10106697E 01      0.10112467E 01
```

四半期）

MT ： 1，2，3，……，10，11，12（月）

EDITコマンドによる出力形式

EDIT コマンドを用いた場合，次の様な形式でライン・プリンターに出力される。

＊＊＊＊JUMPS DATA CONVERSION CHECK LIST＊＊＊＊

SOUCE DATA NAME＝

INITIAL TIME＝ TERMINAL TIME＝ UNIT＝

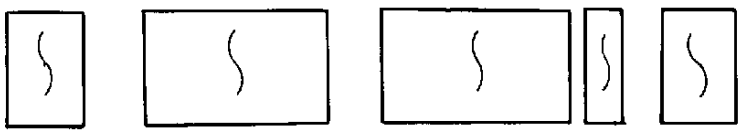
EXPONENT＝ CHARACTER＝ ATTRIBUTE＝ CONVERTED

CONVERTED DATA NAME＝

INITIAL TIME＝ TERMINAL TIME＝ UNIT＝

EXPONENT＝ CHARACTER＝ ATTRIBUTE＝

TIME SOURCE DATA CONVERTED NO. TIME



END OF CONVERSION

注意

INITIAL TIME	：	編集開始時
TERMINAL TIME	：	編集終了時
UNIT	：	単位（数字コード）
		1＝km，2＝km^2，3＝m^2，4＝Ton
		5＝円，6＝ドル，7＝その他
EXPONENT	：	指数（データが$1\cdot0^n$のSCALEではいっているときのn）
CHARACTER	：	STOCK DATAのとき0，FLOW DATAのとき1
ATTRIBUTE	：	属性 1＝年データ，2＝半期データ，3＝四半期データ，4＝月次データ

注意(1) データの名前は，英数文字で構成され，名前の先頭は英文字でなくてはなら

ない。

(2) 例3), 4)に於いて両コマンドは，同じ働きをする。

JUMPS で取り扱えるデータ個数は最大181個であるから，編集した後，データが181 を越える場合，182 個目以降は全て無視される。
そのとき，次のメッセージが現われる。

＊EDIT WARING TOO MANY DATAS＊

例 1951年から1970年のデータを月データに変換する場合，変換されたデータは全部で240個となる。そのうち181個目までのデータは有効となるが182個目から240個目までのデータは無視される。

EDIT コマンドによる編集方法

時系列データを，ストックデータ（STOCK DATA）とフローデータ（FLOW DATA）に大きく分ける。

編集は次に示す2つの場合が考え得る。

(1) 属性番号の大きい属性から，小さい属性への編集。

(2) 属性番号の小さい属性から，大きい属性への編集。

尚属性番号とは属性に与えられた数字とする。

上記(1), (2)の場合とストックデータ，フローデータとを考え合せれば，編集には4つの場合がある。

以下に，4つの場合についての編集手法を述べる。

(1) ストックデータを属性番号の大きい属性から，小さい属性への編集。

時系列n，n＋1，……，n＋w－1のw個のデータ

S(n)，S（n＋1），……，S（n＋w－1）を，属性番号の小さい属性に編集し，その結果をS_{ED}とすると，

$$S_{ED}(j) = S(n)+S(n+1)+S(n+2)+\cdots\cdots+S(n+w-1) \quad -①$$

但し，wは相互のデータによって定まる。又jは編集後の時系列とする。

(2) フローデータを属性番号の大きい属性から，小さい属性への編集。

(1)の場合とは異なりフローデータの場合は編集値は算術平均とする。

F(n)，F（n＋1），……，F（n＋w－1）のw個のデータによって，編集値

$$F_{ED}(j) = \frac{F(n)+F(n+1)+\cdots\cdots+F(n+w-1)}{w} \quad -②$$

を求める。

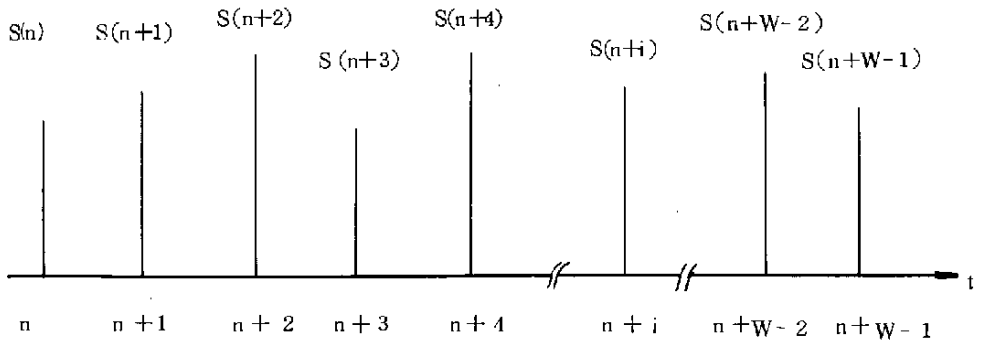


図 2.1

(3) ストックデータを属性番号の小さい属性から，大きい属性への編集。

ストックデータを属性番号の大きいデータに編集する場合，データ $S_{ED}(j)$ 注1) については $S_{ED}(j+1)$ のデータを考慮に入れて，次式の如く編集されたデータ S(n)，S(n+1)，……，S(n+w−1) の増加率を決定する。

$$S_{ED}(j+1)-S_{ED}(j)=(\triangle y\cdot w)w$$

$$\therefore \triangle y=\frac{S_{ED}(j+1)-S_{ED}(j)}{w^2} \quad -③$$

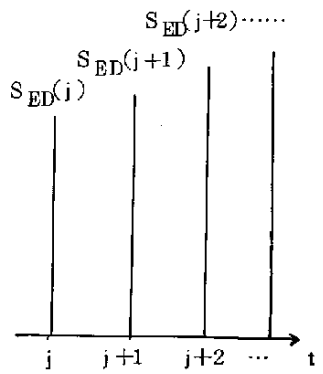


図 2.2

又 S(n) は次の如く決定する。

$$S_{ED}(j)=w\cdot S(n)+1/2\cdot w(\triangle y\cdot(w-1))$$

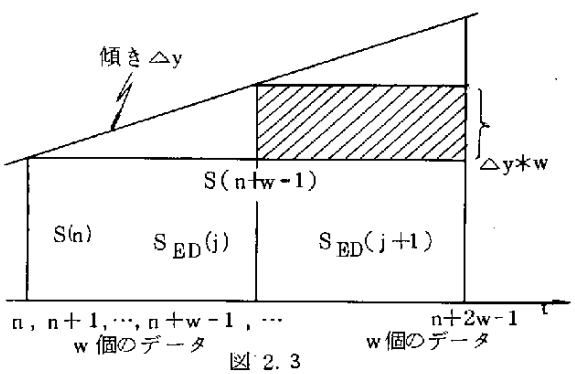


図 2.3

$$\therefore S(n)=\frac{(3w-1)S_{ED}(j)-(w-1)S_{ED}(j+1)}{2\cdot w^2} \quad -④$$

注 1) j が最大で，j+1 がデータとしてない場合，仮想データ $S_{ED}(j+1)$ を考える。

$$S_{ED}(j+1)=2S_{ED}(j)-S_{ED}(j-1) \quad -⑤$$

(4) フローデータを属性番号の小さい属性から，大きい属性に編集。

フローデータ$F_{ED}(j)$は式②から算術平均によって求められたので図2.3の$S_{ED}(j)$との間に次の関係が成立つ。

$$S_{ED}(j) = F_{ED}(j) \cdot w \qquad -⑥$$

従って式③，④，⑤，⑥から

$$\triangle y = \frac{F_{ED}(j+1) - F_{ED}(j)}{w} \qquad -⑦$$

$$F(n) = \frac{(3 \cdot w-1) * F_{ED}(j) - (w-1) * F_{ED}(j+1)}{2w} \qquad -⑧$$

又，フローデータの場合もストックデータ同様式⑤を最終のデータに用いる。

2.3 データ分析

このステージにおいては，傾向分析，周期解析，季節調整等の時系列分析をおこなう。またGPFS（General Purpose Forecasting Simulator）を使うことにより，予測をおこなうことができる。

このステージに入るためには

JUMP DATA;

コマンドを使用すればよい。

このステージで用意されている手法は以下のとうりである。

傾向分析

a．多項式（1～3次まで）

b．指数（1～3次まで）

c．修正指数

d．生長曲線

e．ゴンペルツ曲線

f．ロジスティック曲線

周期解析

a．ペリオドグラム

季節調整

a．連環比率法

EDITコマンドの例

```
EDIT(CONVERT0008,1967/2,1967/3,QT)=POPULATION;

        * * * *  J U M P S   D A T A   C O N V E R S I O N   C H E C K   L I S T  * * * *

SOUCE DATA NAME = POPULATION
        INITIAL TIME = 1951/ 0   TERMINAL TIME = 1967/ 0   UNIT =  7   EXPONENT =  3   CHARACTER = 1   ATTRIBUTE = 1

CONVERTED DATA NAME = CONVERT0008
        INITIAL TIME = 1967/ 2   TERMINAL TIME = 1967/ 3   UNIT =  7   EXPONENT =  3   CHARACTER = 1   ATTRIBUTE = 3


         TIME        SOUCE DATA          CONVERTED DATA      NO.     TIME

        1967/ 1     0.1002430E 06
                                        0.1000944E 06        1     1967/ 2
                                        0.1003916E 06        2     1967/ 3

                                                                    END OF CONVERSION
```

b．平均移動法

センサス局法

GPFS

a．単純指数平滑
b．二次指数平滑
c．二重指数平滑
d．三重指数平滑

このステージ内で使用できるコマンドは次のとおりである。

2.3.1 POLコマンド

$X = a_0 + a_1 t + a_2 t^2 + a_3 t^3$ という多項式の $a_0 \sim a_3$ を推定する。

POL Var〔, 予測期間〔, 次数〕〕;

Var　：指定変数である。変数名のみか，あるいはそのあとにデータの期間指定をつけたものである。これは開始時点と終了時点をカンマでくぎり，両者をカッコでくくる。

データの開始時点および終了時点が省略された場合はデータのある全期間が指定されたものとする。

予測期間：非負の整数で何期分の予測をするのかを示す。

指定のない時は0とみなし，予測はおこなわない。

次　数：1～3まで整数であって多項式の次数を指定する。

指定のないときは3とみなす。

出　力

○CRTに原データおよび予測データのグラフを表示する。

○ラインプリンターに原データ，予測データ，標準偏差，推定された $a_0 \sim a_3$ をプリントする。

例

```
POL CAR;
POL CRUDESTEEL, 5, 2;
POL COMPUTER (1960/1, 1970/1), 5, 3;
```

2.3.2 EXPOコマンド

$X = a_0 a_1^t a_2^{t^2} a_3^{t^3}$ の $a_0 \sim a_3$ を推定する。

EXPO Var〔, 予測期間〔, 次数〕〕;

Var　　: POLコマンドと同じ

予測期間 : POLコマンドと同じ

次　数 : 1～3までの整数であって指数の次数を指定。

指定のない時は3次とみなす。

出 力

POLコマンドと全く同じである。

例

EXPO POPULATION;

2.3.3 MEXPOコマンド

$X = K - a_0 a_1^t$ の a_0, a_1 を推定する。

MEXPO Var〔, 予測期間〔, K〕〕;

Var　　: POLコマンドと同じ

予測期間 :　同　上

K　　: 極限値Kの値を指定(実数で)する。

指定のない場合はシステム側で推定する。

出 力

○CRTに原データおよび予測データのグラフを表示する。

○ラインプリンターに原データ, 予測データ, 標準偏差, 推定された a_0, a_1, Kをプリントする。

例

MEXPO DEATH, 5;

2.3.4 GRWコマンド

$X = Ke^{-\frac{a}{t}}$ のKとaを推定する。

GRW Var〔, 予測期間〕;

Var　　: POLコマンドと同じ

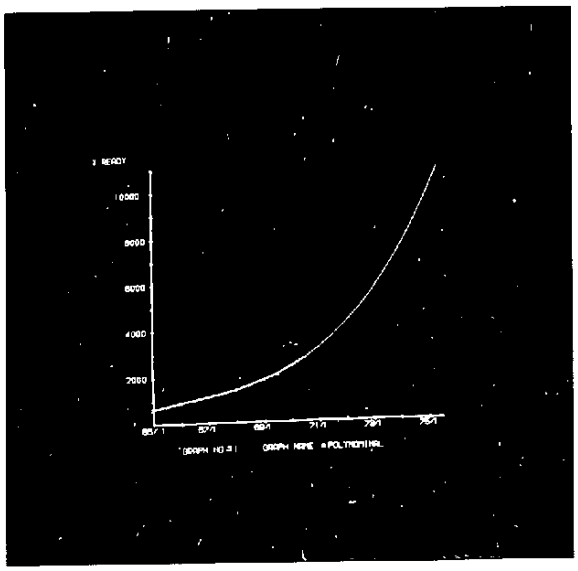
予測期間 :　同　じ

出 力

○CRTに原データおよび予測データのグラフを表示する。

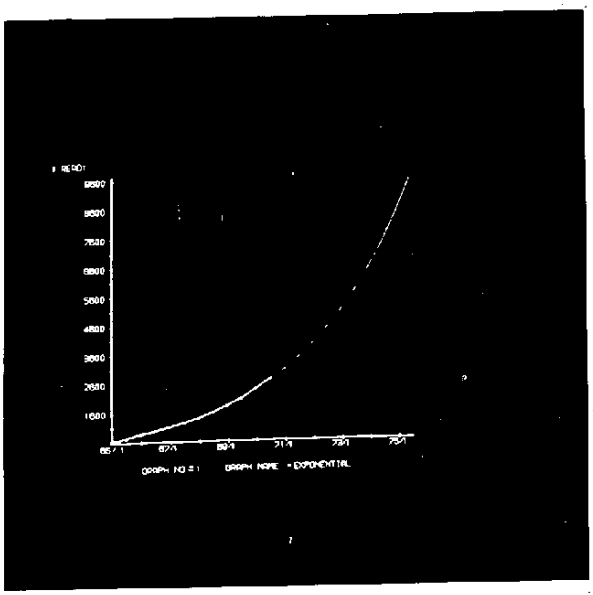
○ラインプリンターに原データ, 予測データ, 標準偏差, 推定されたK, aをプリン

POL MEDIUMS, 10; の実例




```
DATA=
   0.631000E 03   0.743000E 03   0.869000E 03   0.988000E 03   0.111200E 04
   0.124500E 04   0.136200E 04   0.159100E 04   0.181200E 04   0.203500E 04
   0.237600E 04   0.271400E 04

POL   Y=A0+A1*T+A2*T**2+A3*T**3

SD=    0.113106E 02

A0=    0.487364E 03
A1=    0.151699E 03
A2=   -0.112376E 02
A3=    0.117402E 01

FORECAST DATA=
   0.629000E 03   0.755209E 03   0.873042E 03   0.989545E 03   0.111177E 04
   0.124676E 04   0.140197E 04   0.156525E 04   0.179885E 04   0.205541E 04
   0.235998E 04   0.271962E 04   0.314146E 04   0.363227E 04   0.419939E 04
   0.484977E 04   0.559544E 04   0.645284E 04   0.737093E 04   0.842482E 04
   0.959723E 04   0.106952E 05
```

EXPŌ MDIUMS, 10, 1; の実例




```
 DATA=
      0.631000E 03    0.743000E 03    0.889000E 03    0.988000E 03    0.111200E 04
      0.124500E 04    0.138800E 04    0.159100E 04    0.181200E 04    0.203500E 04
      0.237600E 04    0.271400E 04

 EXPO  Y=A0+A1**T*A2**(T**2)*A3**(T**3)

 SD=    0.250011E 02

 A0=    0.578788E 03
 A1=    0.113621E 01
 A2=    0.000000E 00
 A3=    0.000000E 00

 FORECAST DATA=
      0.657627E 03    0.747206E 03    0.848956E 03    0.964631E 03    0.109603E 04
      0.124532E 04    0.141496E 04    0.160769E 04    0.182668E 04    0.207551E 04
      0.235822E 04    0.267945E 04    0.304443E 04    0.345912E 04    0.393031E 04
      0.446567E 04    0.507396E 04    0.576511E 04    0.655041E 04    0.744267E 04
      0.845647E 04    0.960837E 04
```

トする。

2.3.5 GOMコマンド

$X = K a_0 a_1^t$ のK，a_0，a_1を推定する。

GOM Var〔，予測期間〕；

Var　：POLコマンドと同じ

予測期間：　同　上

出力

○CRTに原データおよび予測データのグラフを表示する。

○ラインプリンターに原データ，予測データ，標準偏差，推定されたK，a_0，a_1をプリントする。

例

GOM OIL, 3;

2.3.6 LOGコマンド

$X = \frac{K_0}{1 + m e^{-at}} + K_1$ の K_0，m，aを推定する。

LOG Var〔，予測期間〔，K_0〔，K_1〕〕〕；

Var　：POLコマンドと同じ

予測期間：　同　上

K_0　：K_0の値（実数）を指定する。

指定しなければシステム側が推定する。

K_1　：K_1の値（実数）を指定する。

指定しなければ0.0とみなす。

出力

○CRTに原データおよび予測データのグラフを表示する。

○ラインプリンターに原データ，予測データ，標準偏差，推定されたK_0，m，aをプリントする。

例

LOG FLIGHT, 5;

LOG FLIGHT（1960，1968），6，3.0;

2.3.7 TRENコマンド

POL, EXPO, MEXPO, GRW, GOM, LOGコマンドの中で指定されたものすべてを計算し，その標準偏差の一番小さいものをとりだす。

TREN(手法〔, 手法… 〕), Var〔, 予測期間〕;

手　法：ALL　　傾向分析すべて

POL(n)　n 次の多項式

EXPO(n)　n 次の指数

MEXPO　修正指数

GRW　生長曲線

GOM　ゴンペルツ曲線

LOG(K_0, K_1) ロジスティック曲線

Var　：POL コマンドと同じ

予測期間：　同　上

出　力

○標準偏差の一番小さかった曲線のあてはめのグラフがCRTに表示される。

○ ラインプリンターに，指定された手法によって推定されたパラメータおよび標準偏差がプリントされる。また，一番あてはまりのよかった曲線に関しておこなった予測結果がプリントされる。

例

TREN(ALL), COMPUTER(1965, 1970), 5;

TREN(POL(3), EXPO(2), GRW), CAR;

2.3.8　PGM コマンド

ペリオドグラムによって最小分散を持つ周期を得る。

PGM Var, m, n;

Var　：POL コマンドと同じ

m　：計算をする最小周期

n　：計算をする最大周期

m≦Cycle≦n の周期が得られる。

例

PGM TVSET, 2, 5;

2.3.9　LRV コマンド

連環比率法により，季節指数，季節調整済データを作り上げる。

LRV Var ;

Var : POLコマンドと同じ。たゞし，この変数は，四半期あるいは，月次データでなければならない。

例 LRV DEPART;

LRV DEPART(1960/1, 1968/12);

2.3.10 MVAコマンド

平均移動法により，季節指数，季節調整済データを作り上げる。

MVA Var ;

Var LRVコマンドと同じ

例 MVA BEER;

MVA BEER(1960/1, 1968/12);

2.3.11

CENSUS Var[, G];

Var : POLコマンドと同じ，但し，この変数は60ケ月以上の月次データでなければならない。又，データ数が12の倍数でない時は，余り数だけのデータを古い順に切り捨てる。

G : 計算結果をグラフィック表示する場合の指示を与える。

G=NO : グラフィック表示しない。

=S : 季節指数及びむこう1年間の予測を表示する。

=I : 不規則変動を表示する。

=TC : トレンドサイクル(スペンサー15ケ月加重による)及び原データを表示する。

=TCM:トレンドサイクル(MCDスパンによる)及び原データを表示する。

注1) Gの指定を省略した場合は，S，I，TC，TCM全てが指定されたものとする。

注2) Gの指定は，で区切る事によって複数個指定出来る。

注3) i) G=Sの場合，予測期間は淡い線で示す。

ii) G=TC or TCMの場合，原データは濃い線，トレンドサイクルは淡い線で示す。

出 力

1）Gで指定したグラフィック表示

2）ラインプリンタ出力

(A) 原データ

(B) 季節指数（向こう1年の予測を含む）

(C) 不規則変動

(D) トレンド・サイクル（スペンサー加重15ケ月調整）

(E) MCDスパン

(F) トレンド・サイクル（MCDスパン調整）

(G) 季節調整系列のランダム性の検定（TCI）

(H) トレンド・サイクルのランダム性検定（TC）

(I) 不規則変動のランダム性検定(I)

例

```
CENSUS  A;
CENSUS  A(1958, 1965), S, TC;
```

2.312 SESコマンド

単純指数平滑法により予測をする。

SES Var〔, 予測期間〔, α〔, S〕〕〕;

Var : POLコマンドの同じ

予測期間 : GPFSで予測する期間を指定する。
指定されていなければ1期とみなす。

α : 指数平滑化定数
指定されていなければα＝0.1とみなす。

S : 指定してあればステップワイズに予測する。すなわち，まず最初に一期先を予測し，その予測値を実際のデータとみなして順次予測してゆく。

出 力

○CRTに，原データおよび予測データのグラフを表示する。

○ラインプリンターに，原データ，予測データおよび標準偏差のプリントをする。

例

```
SES  COMPUTER, 10, 0.9, S;
SES  POPULATION, 5, 0.3;
```

****

*** CENSUS METHOD JUMPS ***

****

DATA NAME ;DEPTSELLING

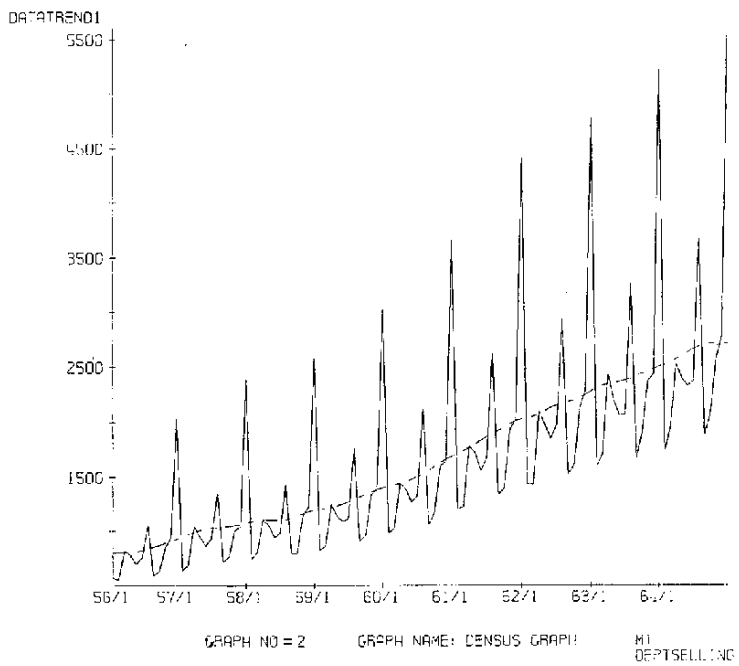
PERIOD COVERED ; 1/1956 - 12/1964

SAMPLE SIZE ;108

** DEPTSELLING **

*** ORIGINAL DATA (SHUSEI) ***(1/1956 - 12/1964)

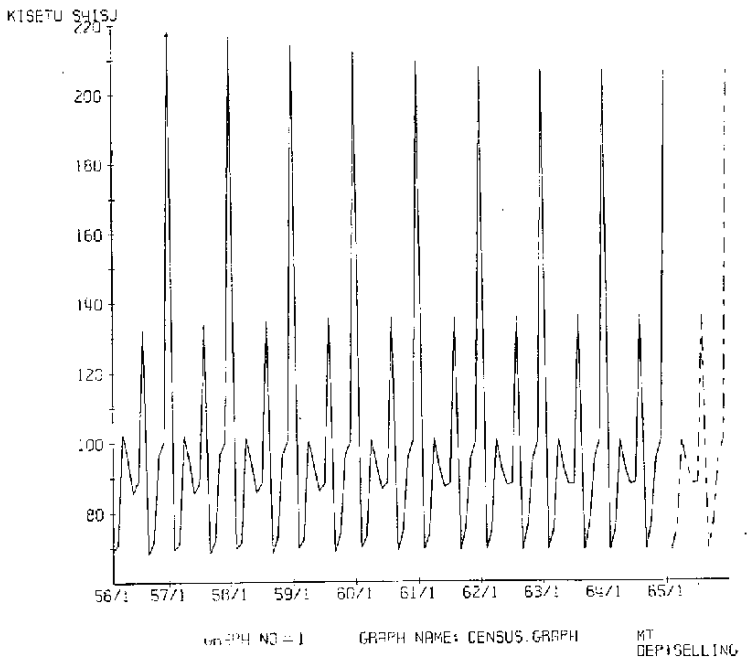
TIME	1	2	3	4	5	6	7	8	9	10	11	12
1	571.00	544.00	819.00	776.00	696.00	787.00	1051.00	593.00	630.00	847.00	943.00	2039.00
2	634.00	686.00	1046.00	946.00	868.00	932.00	1356.00	718.00	768.00	1007.00	1048.00	2393.00
3	746.00	811.00	1100.00	1058.00	942.00	985.00	1439.00	796.00	792.00	1140.00	1228.00	2561.00
4	828.00	862.00	1248.00	1139.00	1089.00	1120.00	1764.00	939.00	964.00	1341.00	1383.00	3027.00
5	985.00	1026.00	1441.00	1389.00	1265.00	1327.00	2122.00	1066.00	1164.00	1602.00	1648.00	3661.00
6	1204.00	1227.00	1781.00	1719.00	1559.00	1671.00	2639.00	1338.00	1396.00	1933.00	2025.00	4412.00
7	1434.00	1427.00	2106.00	1974.00	1851.00	1979.00	2938.00	1522.00	1607.00	2130.00	2293.00	4785.00
8	1594.00	1702.00	2437.00	2197.00	2064.00	2071.00	3274.00	1662.00	1906.00	2377.00	2436.00	5216.00
9	1739.00	1952.00	2530.00	2396.00	2325.00	2372.00	3662.00	1885.00	2069.00	2568.00	2784.00	5840.00



** DEPTSELLING **

*** TREND CYCLE 15 MT [DOME] ***(1/1956 - 12/1964)

TIME	1	2	3	4	5	6	7	8	9	10	11	12
1	798.51	798.53	800.74	805.56	813.22	824.17	838.42	854.66	870.21	892.45	910.99	930.07
2	950.12	969.92	989.29	1007.38	1020.66	1029.86	1036.48	1041.28	1046.85	1055.38	1066.22	1078.74
3	1091.12	1100.43	1105.27	1105.05	1102.72	1102.54	1108.43	1121.50	1141.69	1163.23	1181.75	1195.64
4	1204.56	1211.82	1221.17	1234.90	1251.78	1271.82	1293.61	1316.81	1340.66	1363.82	1384.93	1401.68
5	1415.48	1428.66	1442.72	1459.86	1480.86	1504.30	1531.28	1562.90	1596.71	1631.46	1664.65	1692.00
6	1715.78	1741.37	1769.37	1801.70	1836.84	1868.53	1897.78	1927.45	1957.52	1987.04	2013.24	2031.16
7	2043.54	2058.05	2077.87	2105.26	2135.00	2158.60	2174.70	2186.78	2201.27	2222.91	2252.64	2283.89
8	2312.22	2333.98	2347.75	2358.93	2369.65	2383.27	2400.84	2422.55	2444.63	2466.32	2486.51	2504.21
9	2522.57	2544.33	2571.58	2602.06	2634.85	2665.76	2690.25	2706.46	2715.01	2717.25	2715.50	2713.63



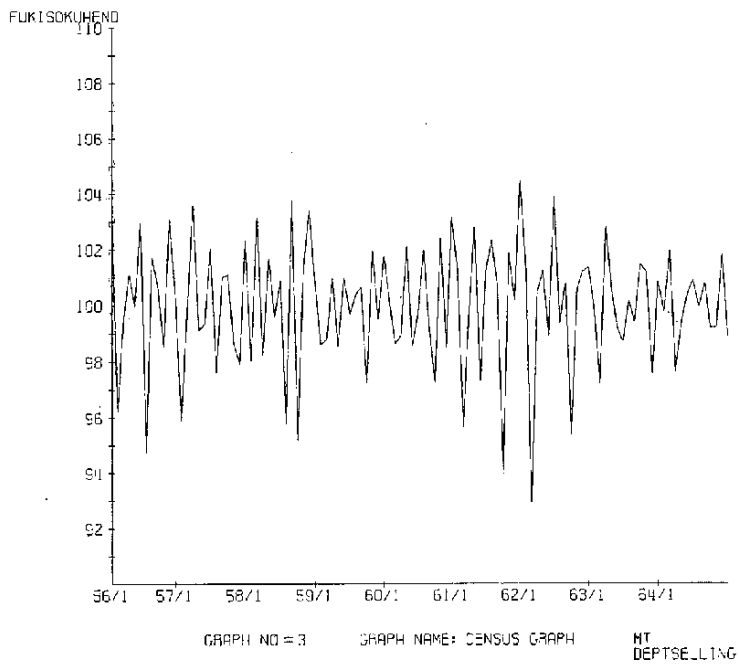
** DEPTSELLING **

*** KISETSU SHISU ***(1/1956 - 12/1964)

TIME	1	2	3	4	5	6	7	8	9	10	11	12
1	69.25	70.85	102.82	95.27	85.62	89.17	132.32	68.20	71.67	96.53	100.40	218.45
2	69.56	70.99	102.05	94.73	85.59	88.70	134.04	68.26	72.09	96.73	100.39	216.77
3	69.75	71.44	101.33	94.13	85.79	88.53	135.19	68.40	72.88	96.71	100.49	214.40
4	69.71	72.00	101.14	93.61	86.13	88.36	135.86	68.57	73.95	96.46	100.38	212.10
5	69.53	72.81	100.96	93.16	86.69	88.51	135.87	68.80	74.97	95.87	100.53	209.70
6	69.29	73.66	101.10	92.82	87.23	88.32	135.88	68.91	75.94	95.51	100.40	207.93
7	69.13	74.61	100.85	92.61	87.67	88.24	135.95	69.01	76.53	95.24	100.56	206.67
8	69.08	75.02	100.96	92.59	87.79	88.05	136.11	69.02	76.84	95.25	100.47	206.50
9	69.11	75.18	100.77	92.61	87.89	88.18	136.19	69.10	76.82	95.24	100.48	206.46

** KISETSU SHISU YOSOKU **

TIME	1	2	3	4	5	6	7	8	9	10	11	12
	69.13	75.27	100.71	92.62	87.93	88.24	136.23	69.13	76.82	95.24	100.78	206.44



** DEPTSELLING **

*** FUKISOKU HENDO ***(1/1956 - 12/1964)

TIME	1	2	3	4	5	6	7	8	9	10	11	12
1	103.26	96.15	99.47	101.11	99.96	103.00	94.73	101.74	100.66	98.53	103.10	100.36
2	95.90	99.63	103.61	99.13	99.36	102.03	97.60	101.02	101.11	98.64	97.91	102.34
3	98.02	103.16	98.22	101.71	99.57	100.92	95.76	103.76	95.19	101.34	103.41	100.69
4	98.60	98.79	101.04	98.53	101.00	99.68	100.37	100.67	97.23	101.95	99.48	101.82
5	100.08	98.64	98.93	102.14	98.54	99.67	102.00	99.14	97.24	102.42	98.48	103.18
6	101.27	95.66	99.56	102.79	97.30	101.25	102.34	100.74	93.91	101.86	100.18	104.47
7	101.51	92.93	100.50	101.25	98.89	103.89	99.38	100.86	95.40	100.61	101.23	101.38
8	99.80	97.21	102.88	100.59	99.21	98.69	100.19	99.40	101.46	101.18	97.51	100.87
9	99.75	102.04	97.63	99.42	100.40	100.91	99.95	100.80	99.20	99.23	101.83	98.88

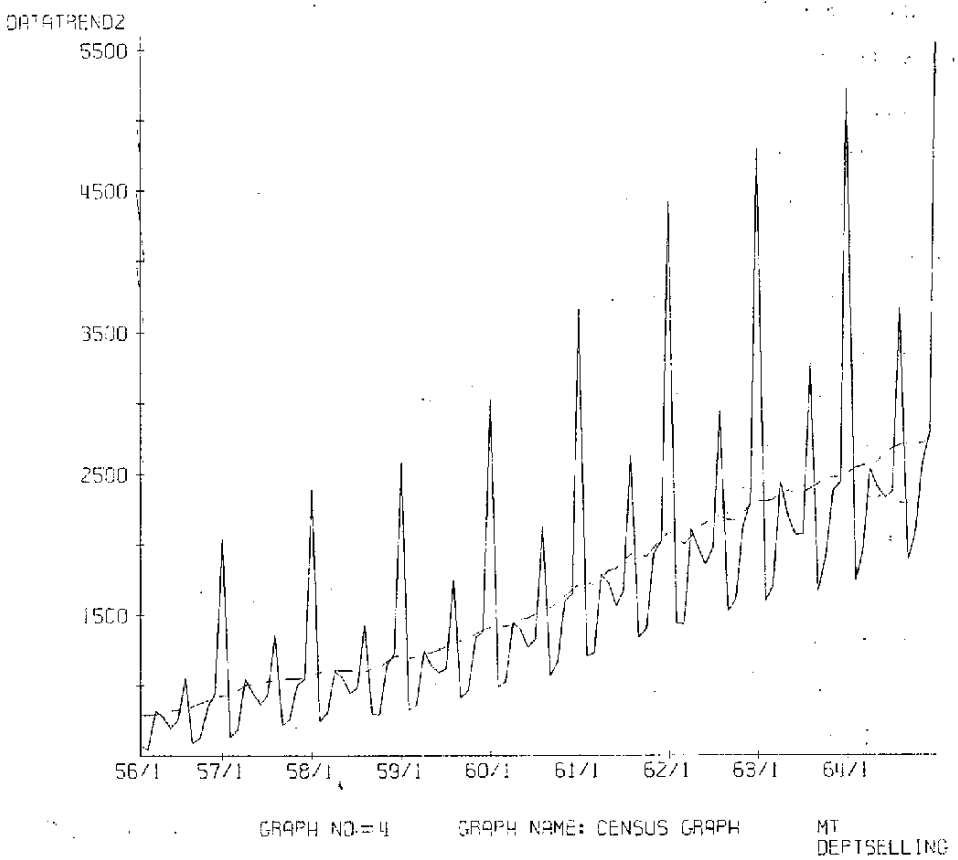
RANDOM SEI KENTEI DCI= 1 DC= 26 DI= 1

** DEPTSELLING **

*** TREND CYCLE BY MCD SPAN ***(1/1956 - 12/1964)

TIME	1	2	3	4	5	6	7	8	9	10	11	12
1	789.15	789.15	793.84	809.61	822.30	826.25	826.75	853.08	876.70	894.21	922.79	929.29
2	938.51	967.22	1003.74	1009.09	1019.42	1031.83	1031.48	1043.46	1052.44	1046.10	1058.22	1080.34
3	1094.55	1106.40	1107.60	1107.89	1108.15	1098.18	1099.80	1118.89	1129.00	1166.59	1206.66	1204.35
4	1194.11	1204.01	1220.44	1232.92	1253.28	1274.54	1297.54	1313.31	1330.77	1369.51	1393.25	1412.15
5	1417.37	1415.57	1438.70	1467.15	1477.21	1504.92	1543.10	1553.33	1581.39	1633.46	1673.85	1717.13
6	1721.71	1707.71	1760.25	1813.17	1829.59	1878.33	1929.90	1915.96	1910.53	1975.75	2044.91	2083.75
7	2045.79	1996.97	2055.22	2115.72	2149.22	2189.45	2192.64	2168.06	2160.45	2213.25	2278.09	2304.64
8	2299.85	2315.17	2368.09	2377.97	2356.66	2365.04	2392.67	2425.92	2466.20	2474.03	2467.60	2498.15
9	2538.69	2554.85	2551.12	2552.55	2642.01	2678.96	2698.92	2709.55	2702.70	2712.79	2727.55	2727.55

RANDOM SEI KENTEI DCI= 1 DC= 26 DI= 1



GRAPH NO.= 4 GRAPH NAME: CENSUS GRAPH MT DEPTSELLING

SES SALES;

2.3.13 SEOS コマンド

二次指数平滑法により予測する。

SEOS Var〔, 予測期間〔, α〔, S〕〕〕;

コマンドパラメータおよび出力はすべてSESコマンドと同じである。

2.3.14 DES コマンド

二重指数平滑法により予測する。

DES Var 〔, 予測期間〔, α〔, S〕〕〕;

コマンドパラメータおよび出力はすべてSESコマンドと同じである。

2.3.15 TES コマンド

三重指数平滑法により予測する。

TES Var 〔, 予測期間〔, α〔, S〕〕〕;

コマンドパラメータおよび出力はすべてSESコマンドと同じである。

2.3.16 SETコマンド

データ分析した結果をセレクトファイルに登録する。

SET key ;

key : 調整済データにつけるキー

注意 このコマンドを使用する場合は，直前にデータ分析コマンドを使用して分析するか，または予測をおこなった後でなければならない。

2.4 変数間の分析

このステージにおいては，変数相互間の相関分析および回帰分析をおこない，つぎの模型作成ステージに対する情報を提供する。

このステージに入るためには

JUMP COR;

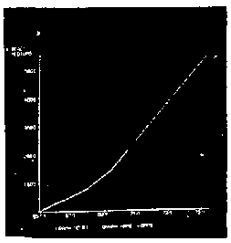
コマンドを使用すればよい。

このステージ内で使用できるコマンドには，以下のものがある。

2.4.1 CDR コマンド

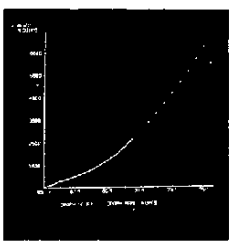
CDRコマンドは，指定された変数相互間の共通期間について相関分析，回帰分析をおこなうコマンドであり，変数を直接に変数名で指定するものと，セクター番号で指定

TES MEDIUMS, 10, 0.9; の実例



```
 ALF=    0.900000E 00

 SD=    0.975393E 02

 ALF=    0.900000E 00

 ORIGINAL DATA
     0.631000E 03   0.743000E 03   0.889000E 03   0.988000E 03   0.111200E 04
     0.124500E 04   0.138800E 04   0.159100E 04   0.181200E 04   0.203500E 04
     0.237600E 04   0.271400E 04

 FORECASTED DATA
     0.307882E 04   0.346768E 04   0.388074E 04   0.431804E 04   0.477969E 04
     0.526559E 04   0.577559E 04   0.630947E 04   0.686766E 04   0.744974E 04
```

DES MEDIUMS, 10, 0.9; の実例



```
 ALF=    0.900000E 00

 SD=    0.514373E 02

 ALF=    0.900000E 00

 ORIGINAL DATA
     0.631000E 03   0.743000E 03   0.889000E 03   0.988000E 03   0.111200E 04
     0.124500E 04   0.138800E 04   0.159100E 04   0.181200E 04   0.203500E 04
     0.237600E 04   0.271400E 04

 FORECASTED DATA
     0.304904E 04   0.338428E 04   0.371953E 04   0.405477E 04   0.439002E 04
     0.472526E 04   0.506050E 04   0.539574E 04   0.573121E 04   0.606660E 04
```

するものとがある。

COR 変数名，変数名〔，G〕；

変数名：相関分析および回帰分析をおこないたい変数の名前

G　：文字ストリング'G'で，グラフィク・ディスプレイ上に相関図を表示することを指示

COR セクター番号〔，個数，相関係数〕；

セクター番号 ： データ定義ステージで指定したセクター番号で，このセクター番号に属する変数について，相関分析，回帰分析をおこなう。

個　　数 ： 相関係数の値によって選択される変数の組合せの最低個数を指示。指定のない場合は5個と仮定される。

相関係数 ： 指定された変数の組合せを選択する場合，この相関係数の値以上の相関係数を有する組合せが選択される。この選択基準の相関係数の値を指示する。指示のない場合は0.95と仮定される。

このコマンドによって，2変数の共通期間のデータを用いて，相関分析および回帰分析をおこない，相関係数，誤差の標準偏差，ダービイン・ワトソン比の計算，パラメータの推定をやり，結果を表示するとともに変数分析表および相関表に登録する。

例

```
COR CRUDESTEEL, DILSELLING, G;
COR 1;
COR 1, 5, 0.98;
```

<注> 変数分析表および相関表の登録について：

変数名を指定して変数間の分析をおこなった場合は，その結果は無条件に登録される。また，セクター番号指定による場合は，個々の組合せによる相関係数値が選択基準の相関係数値以上であれば，指定個数に関係なくその結果は登録される。このようにして登録された個数が指定個数に満たないならば，未登録の組合せの中から相関係数値の高い順に指定個数になるまで登録される。

2.5 模型作成

このステージにおいては，連立方程式模型を想定して，変数分析表および相関表を参照しながら，単一方程式のパラメータの推定と方程式の自動選択をおこない，模型を作成する。

COR コマンドの例

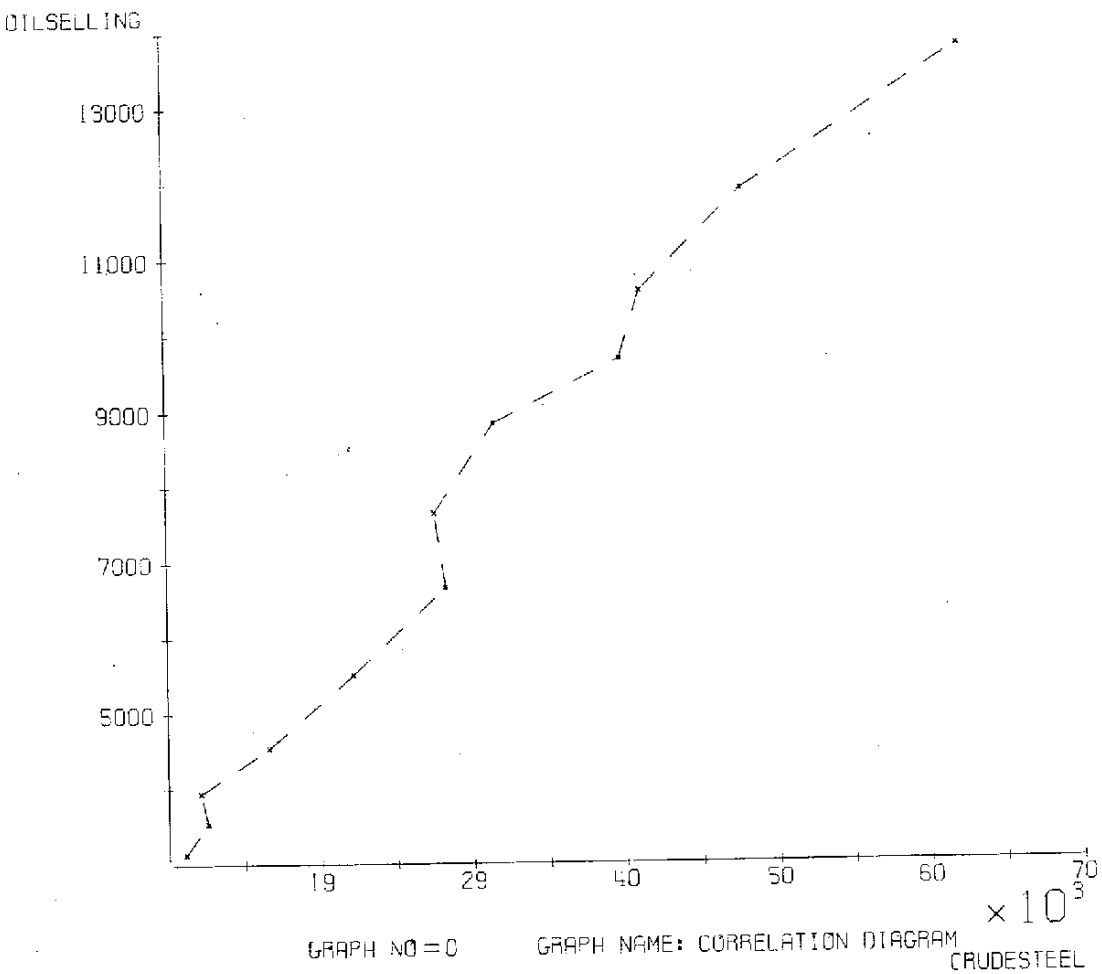
1.2 変数指定の場合

```
COR PASENGCARPR,POPULATION;
```

```
COR COMMAND;
DATAX:
  0.342000E 04   0.467700E 04   0.848900E 04   0.144720E 05   0.202610E 05
  0.320560E 05   0.171210E 05   0.506430E 05   0.785980E 05   0.165094E 06
  0.249508E 06   0.268784E 06   0.407830E 06   0.579660E 06   0.696175E 06
  0.877650E 06   0.137576E 07

DATAY:
  0.832000E 05   0.845410E 05   0.858080E 05   0.869810E 05   0.882390E 05
  0.892760E 05   0.901720E 05   0.909280E 05   0.926410E 05   0.934190E 05
  0.942870E 05   0.951810E 05   0.961560E 05   0.971820E 05   0.982750E 05
  0.990540E 05   0.100243E 06

   (X)  ,VARIABLE, (Y)       ,NO OF DATA,   ,CORRELATION,  ,D-W RATIO,     ,SIGMAH,        ,ALPHA,         ,BETA,
PASENGCARPR ,POPULATION  ;      17          0.835685E 00   0.245559E 00   0.298044E 04   0.888854E 05   0.112427E-01
POPULATION  ,PASENGCARPR ;      17          0.835685E 00   0.350188E 00   0.221512E 06  -0.543530E 07   0.621176E 02
```

2.2 変数指定で相関図表示の場合

```
COR CRUDESTEEL,OILSELLING,G;
```




```
COR COMMAND;
DATAX:
  0.111160E 05   0.125700E 05   0.121180E 05   0.166290E 05   0.221380E 05
  0.282680E 05   0.275460E 05   0.315010E 05   0.397990E 05   0.411610E 05
  0.477040E 05   0.621540E 05

DATAY:
  0.313600E 04   0.354500E 04   0.394200E 04   0.454300E 04   0.551600E 04
  0.662200E 04   0.765400E 04   0.883700E 04   0.968900E 04   0.105770E 05
  0.119230E 05   0.138410E 05

 (X) ,VARIABLE, (Y)      ,NO OF DATA,  ,CORRELATION,  ,D-W RATIO,     ,SIGMAH,      ,ALPHA,       ,BETA,
 CRUDESTEEL ,OILSELLING  ;     12        0.989108E 00   0.157621E 01   0.543931E 03   0.100682E 04   0.218477E 00
 OILSELLING ,CRUDESTEEL  ;     12        0.989108E 00   0.163873E 01   0.246251E 04  -0.414024E 04   0.447796E 01
                                                    JIPDC
```

3. セクター番号指定の場合

```
COR 1,5,0.98;
```

```
COR COMMAND;
SECTOR PARA; SECTOR NO=   1 LIMIT NO=   5 LIMIT CORR= 0.980000E 00
```

(X) .VARIABLE, (Y)	.NO OF DATA,	.CORRELATION,	.D-W RATIO,	.SIGMAH,	.ALPHA,	.BETA,
POPULATION ,OILSELLING ;	13	0,983500E 00	0.524198E 00	0.691513E 03	-0.799869E 05	0.924258E 00
POPULATION ,CRUDESTEEL ;	12	0,967983E 00	0.102293E 01	0.421207E 04	-0.379015E 06	0.431114E 01
CRUDESTEEL ,PASENGCARPR ;	12	0,980426E 00	0.664192E 00	0.858841E 05	-0.351589E 06	0.255629E 02
OILSELLING ,CRUDESTEEL ;	12	0,989108E 00	0.163873E 01	0.246251E 04	-0.414024E 04	0.447796E 01
OILSELLING ,PASENGCARPR ;	13	0,957533E 00	0.647413E 00	0.124035E 06	-0.399692E 06	0.108349E 03

4. セクター番号指定の省略形の場合

```
COR 1;
```

```
COR COMMAND;
SECTOR PARA; SECTOR NO=   1 LIMIT NO=   5 LIMIT CORR= 0.950000E 00
```

(X) .VARIABLE, (Y)	.NO OF DATA,	.CORRELATION,	.D-W RATIO,	.SIGMAH,	.ALPHA,	.BETA,
POPULATION ,OILSELLING ;	13	0.983500E 00	0.524198E 00	0.691513E 03	-0.799869E 05	0.924258E 00
POPULATION ,CRUDESTEEL ;	12	0.967983E 00	0.102293E 01	0.421207E 04	-0.379015E 06	0.431114E 01
CRUDESTEEL ,PASENGCARPR ;	12	0.980426E 00	0.664192E 00	0.858841E 05	-0.351589E 06	0.255629E 02
OILSELLING ,CRUDESTEEL ;	12	0.989108E 00	0.163873E 01	0.246251E 04	-0.414024E 04	0.447796E 01
OILSELLING ,PASENGCARPR ;	13	0.957533E 00	0.647413E 00	0.124035E 06	-0.399692E 06	0.108349E 03

このステージに入るためには

JUMP AUTO;

コマンドを使用すればよい。

このステージ内で使用できるコマンドには，以下のものがある。

2.5.1 SHOW コマンド

SHOW コマンドは，これまでの分析結果を表わす変数分析表および相関表を，グラフィク・ディスプレイ上に表示するコマンドである。

SHOW 〔変数名〔,変数名〕…〕;

変数名：被説明変数であり，任意個指定することができる。

指定されない場合は，これまで作成された変数分析表および相関表の全変数を被説明変数とする。

このコマンドによって，指定された被説明変数と変数分析表および相関表に存在する他の変数を説明変数として，プライオリテイ，相関係数，誤差の標準偏差，ダービン・ワトソン比，回帰係数を表示する。

例

SHOW POPULATION, GNP ;

SHOW;

2.5.2 MCON コマンド

MCON コマンドは，模型の自動作成に際して，設定すべき条件を与えるコマンドである。

MCON 本数，係数符号;

本　　数 ： 選択される基本方程式の本数

係数符号 ： 説明変数項係数の符号がマイナスの場合，その方程式を基本方程式候補とするには，＋－，候補としない場合は，＋，と指定する。

基本方程式 ： 変数間の分析ステージで作成された変数分析表および相関表から内生変数による被説明変数，説明変数の２変数模型として方程式群をつくる。これらの方程式群から諸条件によって選択される方程式のことである。

被説明変数 ： ここで被説明変数としては，データ定義ステージで指定した内生変数のなかでプライオリティのもっとも高いものが優先される。また，被

説明変数が2変数以上のグループに属している場合は，そのグループ内の各内生変数が被説明変数となり，同一方程式に，同一グループの変数が併存することはない。

例

MCON　7，＋；

MCON　5，＋－；

2.5.3　MODEL コマンドは，模型の自動作成開始を指示するコマンドである。

MODEL；

このコマンドの入力によって，模型の自動作成が開始される。この時点で，MCON コマンドによって基本方程式の本数と係数符号が指定されていなかった場合は，（本数：5）（係数符号：＋）にセットされる。

例

MODEL；

2.5.4　模型作成中の応答

模型の自動作成中において，係数符号が（＋－）と指定されている場合で，説明変数項の係数符号がマイナスのとき，その方程式を基本方程式候補として残したいか否かを問合せるメッセージが，グラフィク・ディスプレイ上に出力される。このとき利用者は，ライトペンによって答えを選択（ピッキング）しなければならない。

メッセージの形式：

COEFFICIENT SIGN IS MINUS；

対象となっている方程式
$Y = \beta_0 + \beta_1 X_1$

RESPONSE；

＊REMAIN＊ ←--- 残したいときにピック

＊DELETE＊ ←--- 残さないときにピック

2.5.5　模型の表示

模型の自動作成の結果，作成不可能のときは，その旨メッセージが，また作成完了のときは，当該模型が，グラフィク・ディスプレイ上に表示される。

＜注＞　作成された模型は，後述（付録参照）の式テーブルの形式に変換されて格納される。そのため，当該模型は，式番号によって参照することができる。

2.6 推定と評価

このステージにおいては，模型のパラメータ推定および評価をおこなう。

このステージで用意されている手法は，直接最小二乗法，間接最小二乗法および二段階最小二乗法である。

このステージに入るためには，

JUMP ESTIMATE;

コマンドを使用すればよい。

2.6.1 PERIOD コマンド

PERIOD コマンドは，このステージにはいった時に最初に実行しなければならないコマンドであって使用する変数のデータ期間をそろえるためのものである。

PERIOD 開始時点, 終了時点〔, 属性〕;

開始時点：推定に使用するデータの開始時点を指定する。

終了時点：推定に使用するデータの終了時点を指定する。

〔注意〕 変数の中でタイムラグを使用する場合には注意が必要である。例えば1960年から1969年までの年データを使用する場合に，モデル式の中で一年前というタイムラグを使用している変数が1個でも存在する場合は，そのタイムラグ分だけ start point を前にしたものを指定しなければならない。すなわち1959年から1969年までのデータを使うことを宣言しておかなければならない。

属　　性：年（YR），半期（BI），四半期（QT），月（MT）のいずれのデータであるかを指定する。

指定のない場合は年データであることを示す。

例 PERIOD 1959, 1969;

PERIOD 1960/1, 1968/2, BI;

PERIOD 1962/1, 1968/4, QT;

2.6.2 EQコマンド

EQコマンドは，推定すべき式を入力するコマンドであって，同時に式のアップデートの機能も持っている。

EQ 〔n〕 左辺=（同時従属変数群（先決変数群））;

n　　：式番号である。

式番号が指定されていなければ登録された順に1から連番がふられてゆく。

指定されていれば，すでに登録されている番号nの式がこのEQコマンドで入れた式におきかえられる。

左　　辺：同時従属変数の名前

同時従属変数群：同時従属変数の名前をカンマでくぎって書く。

先決変数群：先決変数をカンマでくぎって書く。

先決変数は，外生変数あるいは，変数名のあとにカッコでくくってタイムラグをつけたものである。

例

EQ　X1＝(X2, X4, (Z1, X1(－1)));

EQ　X2＝(X3, X4, (Z2, Z4));

EQ　X3＝(X1, (Z3));

EQ　2　XS＝(X3, X1, (Z2, Z3));

これ等4つのEQコマンドの使用によって

$X1=\alpha_1 X2+\alpha_2 X4+\beta_1 Z1+\beta_2 X1(-1)$

$X2=\alpha_3 X3+\alpha_4 X1+\beta_3 Z2+\beta_4 Z3$

$X3=\alpha_5 X1+\beta_5 Z3$

という式を入力したことになる。

2.6.3　DEFコマンド

DEFコマンドは定義式，あるいはすでに推定されている式の入力をするためのコマンドである。

DEF〔n〕　左辺＝(同時従属変数式,(先決変数式));

n　：式番式で指定があればその式のアップデートをおこなう。指定がなければ連番がつけられる。

左辺：変数名

同時従属変数式，先決変数式をバッカス記法で表現すれば次のようになる。

先決変数式は，同時従属変数式とほゞ同じであるが，変数名のあとカッコでくくってタイムラグをつけることができるのと，最初に定数項がつけられない点が異なる。

符号：＝＋｜－

符号付実定数 : : =＜実定数＞｜－＜実定数＞

同時従属変数項 : : =＜実定数＞＊＜同時従属変数名＞

同時従属変数並び : : =＜同時従属変数項＞｜＜同時従属変数項＞＜符号＞＜同時従属変数並び＞

同時従属変数式 : : =＜同時従属変数並び＞｜－＜同時従属変数並び＞｜＜符号付定数＞＜符号＞＜同時従属変数並び＞

先 決 変 数 : : =＜外生変数名＞｜＜変数名＞(＜タイムラグ＞)

先決変数項 : : =＜実定数＞＊＜先決変数＞

先決変数並び : : =＜先決変数項＞｜＜先決変数項＞＜符号＞＜先決変数並び＞

先決変数式 : : =＜先決変数並び＞｜－＜先決変数並び＞

例

```
DEF  A=(0.5+3.2*B+5.1*C,(0.35*Z1+3.0*Z2));
DEF 2  B=(-0.02*C+0.5*A,(-0.1*Z3));
```

2.6.4 DLSコマンド

単一方程式モデルのパラメータの推定をおこなう。

DLS 〔n〕;

n：推定すべき方程式の式番号である。

このコマンドの実行により，パラメータの推定値，標準偏差，ダービンワトソン比を知ることができる。

2.6.5 IDLS コマンド

間接最小二乗法によりパラメータの推定をする。

IDLS;

このコマンドの入力により，まず適度認定であるかどうかを調べ，適度に認定であれば間接最小二乗法によりパラメータの推定をおこない，同時に，部分テストおよび全体テストをおこなう。

2.6.6 TSLS コマンド

二段階最小二乗法により，パラメータの推定をおこなう。

TSLS;

このコマンドの入力により，二段階最小二乗法によりパラメータの推定をおこない，同時に部分テストおよび全体テストをおこなう。

2.6.7　REVIEW コマンド

REVIEW コマンドは，既に入力されている式をディスプレイ上に表示するコマンドである。

REVIEW 〔n〔, n〕… 〕;

n：式番号であって任意個指定することができる。もし指定しない場合には，これまで入力されているすべての式が表示される。

CRT 上の出力形式は次のとおりである。

n　変数名＝（同時従属変数式，（先決変数式））

n：式番号

同時従属変数式，先決変数式とは次の形で規定するものである。

＜同時従属変数項＞::＝＜実定数＞＊＜同時従属変数名＞|＜同時従属変数名＞

＜符号＞::＝＋|－

＜符号つき実定数＞::＝＜符号＞＜実定数＞

＜同時従属変数式＞::＝＜同時従属変数項＞|

＜符号つき実定数＞＜符号＞＜同時従属変数項＞|

＜同時従属変数式＞＜符号＞＜同時従属変数項＞

＜先決変数項＞::＝＜実定数＞＊＜先決変数名＞|＜先決変数名＞

＜先決変数式＞::＝＜先決変数項＞|

＜先決変数式＞＜符号＞＜先決変数項＞

但し，先決変数名にはタイム・ラグをつけることができる。

例　REVIEW　2;

REVIEW;

注　このコマンドはシミュレーションステージにおいても使用可能である。

2.7 シミュレーション

このステージにおいては，予測およびシミュレーションをおこなう。

このステージに入るための前提条件として，模型作成ステージにおいてモデル式を自動作成をするか，あるいは推定と評価ステージをとおしてモデル式の入力および推定をおこなっておかなければならない。

このステージに入るには，

JUMP　SIMUL;

コマンドを使用すればよい。

2.7.1　EXPOL コマンド

以前のステージにおいてできあがったモデルに対して，外挿テストをおこなう。

EXPOL 変数名（S1，F1，F2）；

変数名：予測したい変数名である。

S1　：グラフに表示したいデータの開始時点

F1　：予測開始時点

F2　：予測終了時点

出　力

○CRT に原データおよび予測データのグラフを表示する。

○ラインプリンターに原データ，および予測データをプリントする。

例

EXPOL　MACHINE（1960，1969，1972）；

EXPOL　CAR（1962／1，1970／1，1971／2）；

〔注意〕　このコマンドを使うにあたっては，すべての先決変数のデータ予測期間の値は，まえもってセットしておかなければならない。

2.7.2　EXCD コマンド

シミュレーションをおこなう場合に必要なデータをセットすることができる。たゞし，ここでセットしたデータは，このステージ内でのみ有効であり，他のステージにJUMP コマンドによってとびこんだ時には値は保持されない。

EXCD 変数名（開始時点，終了時点）＝data，data，…；

変 数 名：データをセットしたい変数

開始時点：データの開始時点をさす。

終了時点：データの終了時点をさす。

data　：実定数でデータの値を示す。

例

EXCD　PAY（1969，1971）＝1234.，1345.，1581.；

EXCD　POPUL（1969／1，1970／2）＝12.1，15.3，19.0，21.0；

2.7.3　CHANGE コマンド

シミュレーションステージにおいて，モデル式の係数あるいは定数項を変化させるためにもちいる。

CHANGE 式番号〔, Var〕, 実定数;

式番号：変更したい式の番号を指定する。

Var ：変数名あるいは，変数名のあとカッコでくくってタイムラグの指定をつけたものである。Varである変数が指定されていると，その変数にかかっている係数の変更とみなす。指定がなければ定数項の変更とみなす。

実定数：実数あるいは負符号付実数であって変更後の値を示す。

例

CHANGE 1, CAR, 0.85;

CHANGE 2, −0.03;

第３章　JUMPSの使用法

ＪUMPＳ においては，第一章，第二章で述べたような各種コマンドを使用すればよいが，その場合の注意事項およびＪUMPＳ使用に先だって作らなければならないオリジナルファイルの作成法を述べる。

3.1　ファイルの作成および保守

ＪUMPＳにおいて，使用されるデータは，利用目的によって異るので，ファイルの更新（アップデート）は頻繁に行われ，また，登録データの量も多量から少量と多岐にわたる。

このような理由から，ファイルの作成および保守は，図 3.1～図 3.4 に示すようにその都度，効率のよい方法が選択できるようになっている。

(1)　ファイルの創成

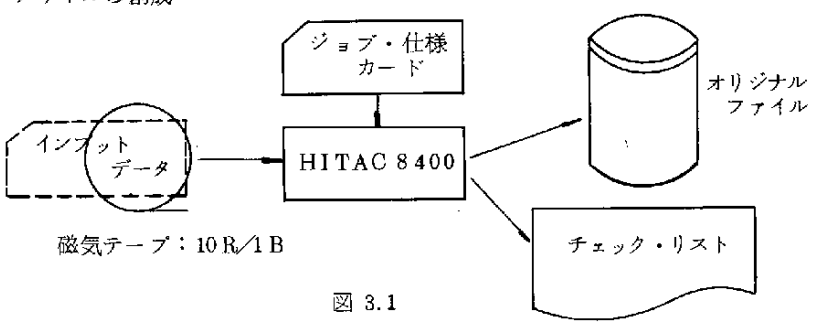


図 3.1

(2)　ファイルの更新

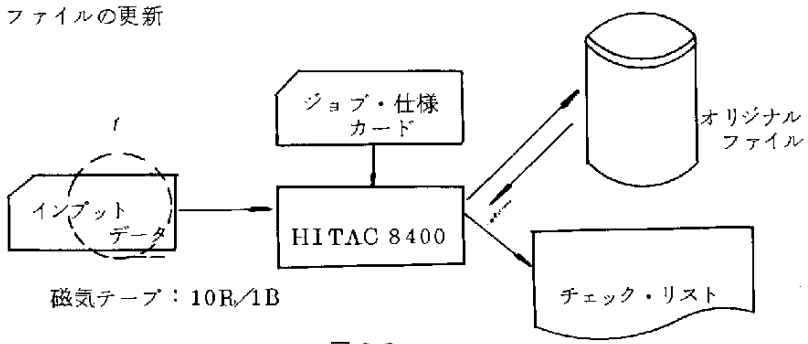


図 3.2

(3) ファイルの複写(保存)

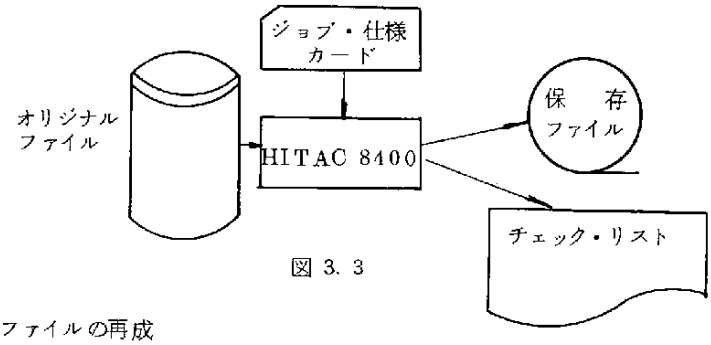


図 3. 3

(4) ファイルの再成

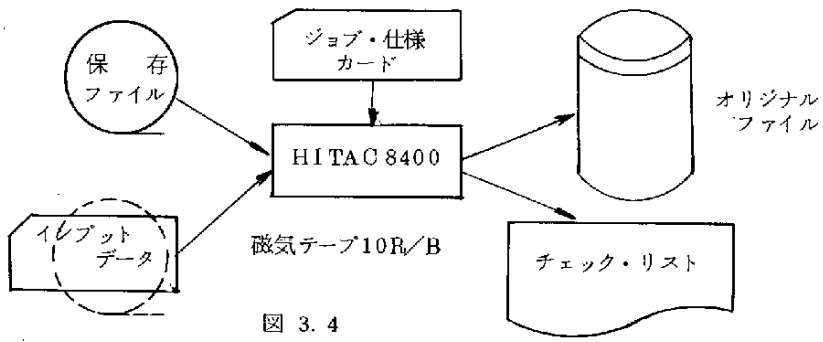


図 3. 4

3. 1. 1　入力データの作成

ジョブ仕様カードの書き方，インプット・データの記入方法について述べる。

1. ジョブ仕様カードについて

a. ジョブ仕様カードの様式を図3.5に示す。

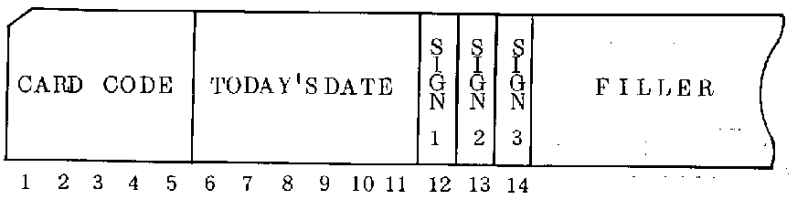


図 3. 5

b. 各項目の説明

CARD CODE：'JUMPS'の5文字

TODAY'S DATE：ラベル情報として，月一日一年の順(年は19××年××である)。

SIGN1：C→カードによるインプット（インプット・データ）。

M→磁気テープによるインプット（インプット・データ）。

ブランク→保存ファイルを作成する。

SIGN2：L→ファイルを創(再)成する。

A→ファイルを更新する。

ブランク→保存ファイルを作成する。

SIGN3：1→インプットはインプット・データのみで1個

2→インプットはインプット・データと保存ファイルの2個

ブランク→保存ファイルを作成する。

c．具体的なジョブ仕様カードを示すと

(1)の場合（ファイルの創成）

JUMPS　MMDDYYCL1　… インプット･データがカード

JUMPS　MMDDYYML1　… インプット･データがテープ

(2)の場合（ファイルの更新）

JUMPS　MMDDYYCA1　… インプット･データがカード

JUMPS　MMDDYYMA1　… インプット･データがテープ

(3)の場合（ファイルの複写）

JUMPS　MMDDYY

(4)の場合（ファイルの再成）

JUMPS　MMDDYYCL2　… インプット･データーがカード

JUMPS　MMDDYYML2　… インプット･データがテープ

2．インプット・データについて

a．インプット・データの様式を図3.6に示す。

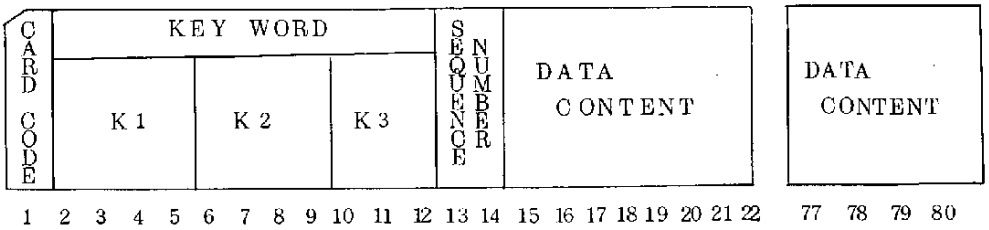


図 3.6

b．各項目の説明

CARD CODE ： 1→新規（新しくファイルに登録する）

2→修正（すでに登録されているデータに追加する）

3→消去（ファイルから当該データを消去する）

KEY WORD ： 英文字ではじまる英数字（ブランクは含まない）

SEQUENCE NUMBER ： ００→DATA CONTENTはスイッチ情報であることを示す。

０１→DATA CONTENTのデータを，データ・エリアの先頭から書き込むことを指示

０２～９９→DATA CONTENTのデータを，そのSEQUENCE-NO順に書き込むことを指示

注意．修正カードでSEQUENCE-NO．が０１であれば，データ・エリアの先頭から，０２以上であれば，すでに書き込まれているデータのつぎから書き込まれる。

DATA CONTENT ： 各データ項目の区切りはカンマ(,)かブランク（ ）で行うこと。また，データ項目の位置は，どこでもよいが，１データ項目が２枚のカードにわたってはならない。負値の表示はその先頭に，負符号(-)で行うこと。さらに，FORTRANでのEタイプ表示も可能である。

表3.1の記入例を参照のこと。

c．スイッチ情報について

i．スイッチ情報の構成：①データ開始年→1901～2099（時系列），1～32768（その他）

②データ開始月日→年 次0

半 期1，2

四半期1，2，3，4

月次　1，2，……11，12

月次　101，102，……1231

（1月1日），……（12月31日）

その他0

③データ終了年→データ開始年と同じ。

④データ終了月日→データ開始月日と同じ

⑤データの単位→1　kM

2　kM^2

3　M^3

4　TON

5　円

6　ドル

7　その他

⑥スケール→データが10^nのスケールで表示されているとき，そのnをセットする。

⑦データの種別→0　ストック・データ

1　フロー・データ

⑧データの属性→1　年次データ

2　半期データ

3　四半期データ

4　月次データ

5　日次データ

6　その他のデータ

ii. スイッチ情報の変更：スイッチ情報の変更は，たとえ1項目を変更するだけの場合でも全部記入しなければならない。

d. インプット・データは，大分類として，KEY WORD，小分類として SEQUENCE NO.の順に分類されていなければならない。

3.1.2　ファイルの作成および保守を行うための手順

オリジナル・ファイルの作成および保守は'JFILE'というプログラムによって行われる。このプログラムを作動させるためには，つぎのような手順によって行わなけれ

ばならない。

1. プログラム・ライブラリー用ディスク・パックを，機番A2にセットし，データ・ファイル用ディスク・パックを，機番A3にセットする。
2. 図3.7のような，ランタイム・パラメータ，ジョブ仕様カード，インプット・データ・カードをセットして，コンソール・タイプライターから'E LOD JFILE, A2, , R0'とタイプインすればよい。
3. ジョブの終了時に，コンソール・タイプライターに，ジョブの終了のメッセージがタイプアウトされる。正常終了であるかどうか確認する事。

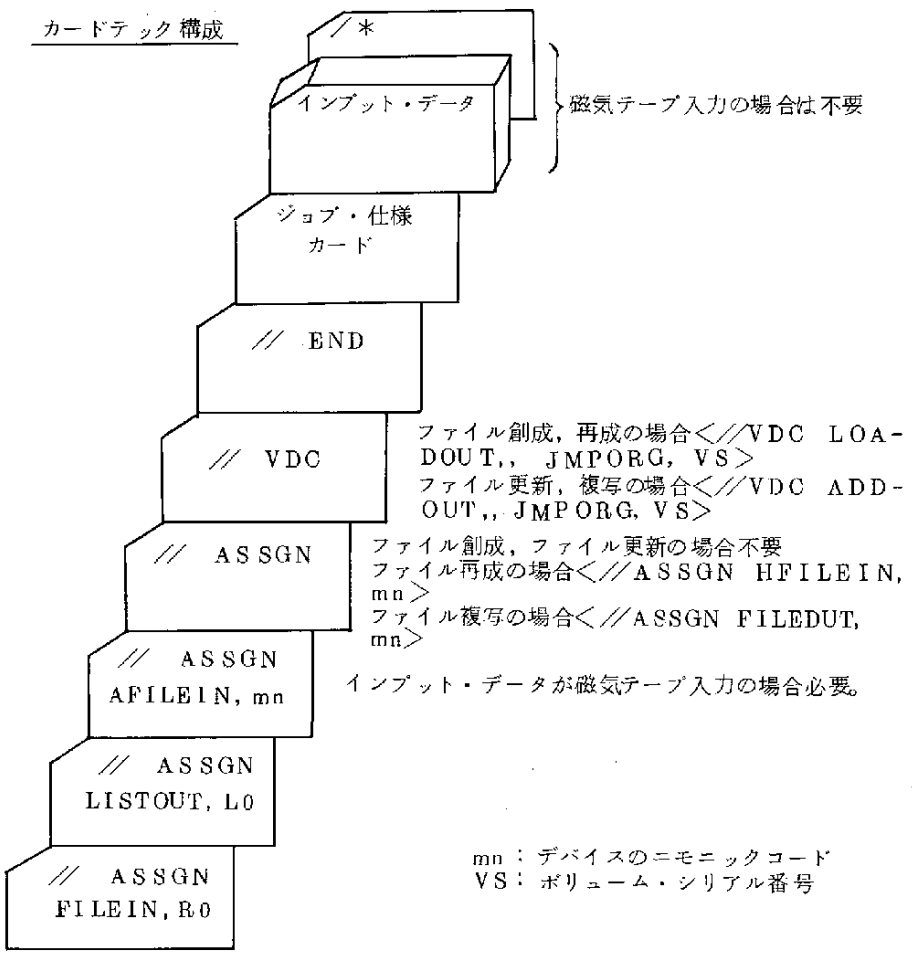


図 3.7

JUMPS インプットデータ記入例

```
1POPULATION  00        1951.0,1967.0,7,3,1,1
1POPULATION  01  83200,84541,85808,86981,88239,89276,90172
1POPULATION  02  90928,92641,93419,94287,95181,96156,97182,98275
1POPULATION  03  99054,100243

1OILSELLING  00  1953.0,1967.0,3,3,0,1
1OILSELLING  010.2559E+04,0.3136E+04,0.3545E+04,0.3942E+04,0.4543E+04,0.5516E+04
1OILSELLING  020.6672E+04,0.7654E+04,0.8837E+04,0.9689E+04,0.10577E+05
1OILSELLING  030.11923E+05,0.13841E+05
```

※ 3. 1

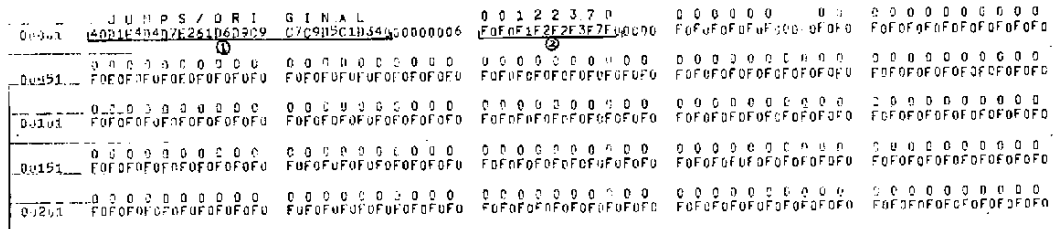
オリジナル・ファイル　プルーフ・リストー部

1. ラベル・レコード

COUNT: (C1) 000 (C2) 016 (H1) 00 (H2) 00 (R) 018 (KL) 012 (DL) 00788

KEY:
404040404040404040404040

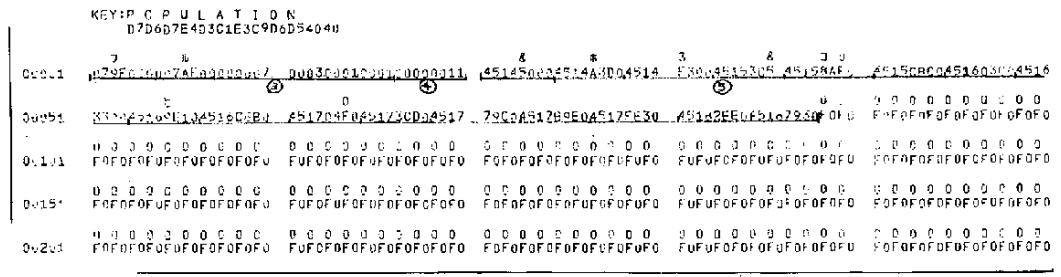
JUMPS/ORIGINAL



2. データ・レコード

COUNT: (C1) 000 (C2) 016 (H1) 00 (H2) 07 (R) 002 (KL) 012 (DL) 00788

KEY:POPULATION
D7D6D7E4D3C1E3C9D6D5404040



① ラベル
② 作成年月日
③ スイッチ情報
④ データ個数
⑤ データ

3.2 会話の開始

HITAC-8400側にGJMON, HITAC-8811側にGCHがロードされている場合，ライトペンによる文字ピッキングにより

¥E LOD JUMPS, An,, Am,,,, PPP

An ：JUMPSがはいっているディスクパックの指定

Am ：ランタイムパラメータがはいっているディスクパックの指定

PPP：ランタイムパラメータの名称

とインプットすればよい。（モニターローディングに関しては第一篇「ディスプレイシステムの研究開発」第5章システム・オペレーションを参照されたい。）現在JIPDECでサーヴィスしているシステムをそのまま使用する場合には，プログラムパックをA3，ファイル用のパックをA2にセットして

SE LOD JUMPS, A3,, A3,,,, JMPRUN

とすればよい。

これによって，JUMPS がコア内にローディングされ，同時にオリジナル・ファイル，セレクトファイル，ワークファイルおよびプリンターのアサインがなされる。

以上のことがおこなわれると，JUMPS 側から，新規のJOBの開始かあるいは中断JOB の再開かを聞いてくるので，このいずれかをライトペンピッキングにより指定すればよい。

新規のJOB の場合には，自分の要求するステージにJUMP コマンドによってジャンプすればよいし，中断JOBの再開の場合には，JUMPSが使用しているテーブル類はすべて，中断した時の状態に再現し，そのステージ名を表示するので中断時に使用していたステージにJUMP コマンドでジャンプして，JOB を続行すればよい。

メッセージの選択をする場合には，選択する文字列のどの部分でもよいから，ライトペンでピッキングすると，ピックした文字列がブリンキングする（点滅する）。これが自分のピックしたものであればSend Picking message キー（ファンクションキーNO 4）をおす。間違っていれば，Reset Picking message キー（ファンクションキーNO 7）をおし，再びピッキングしなおす。

3.3 ファンクションキー

CRTにはファンクションキーが0～31 番までありこのうちCGOS 側で0～10番ま

で予約して使用している。ＪＵＭＰＳ では以下に述べるキーだけを使用している。

キー番号

0番　メッセージインディケータのオンオフ。

ピッキングすべき文字列がＣＲＴの画面上に現われたり消えたりする。

2番　ビューオン・オフ

現在現われている画面を消すことができる。

3番　クイット

このキーをおすことにより，ＪＵＭＰＳ は一時停止する。この状態からぬけだすには次の2つの方法がある。

a. ￥Ｇ　ＳＴＲＴ　実行の再開

b. ￥Ｅ　ＨＬＴ　ｎ　ＪＵＭＰＳのＪＯＢの打切り。ｎはプログラム番号

4番　ライトペンピッキングの確認

ピッキング指定があった時に，ライトペンによってピッキングすると，文字列がブリンキングするから，このキーをおすことにより，ピッキングされたことをシステム側が確認する。

6番　Ｘ－Ｙプロッタへのコピー

現在画面に現われている図をＸ－Ｙプロッタにコピーをとる。

7番　ライトペンピッキング情報のリセット

ピッキングする文字列を間違えた時に，このキーをおし再度ピッキングしなおす。

11番　ＪＵＭＰＳ クイット

このキーをおすことによりＪＵＭＰＳ は一時停止する。この状態からぬけだすには次の2つの方法がある。

a. ＲＥＳＴＡＲＴ　実行の再開

b. ＥＮＤ　ＪＵＭＰＳの中のトータルディストリビュータ（次章参照）にもどるのでＪＵＭＰ コマンドから入れなおす。

〔注〕クイットとＪＵＭＰＳクイットの違いは，後者は ＪＵＭＰＳ システムの中にまだコントロールが存在しており，別のあるいは再び同じＪＵＭＰ コマンドの入力から続行が可能であるが，前者はＪＵＭＰＳのＪｏｂ 自身が打ち切られるので，通常は後者のＪＵＭＰＳ クイットを利用する方が続行が容易である。

12番　ｗａｉｔ 状態からｎｏｒｍａｌ 状態にもどす。

JUMPS において，ピッキング指定のあるものをのぞいたメッセージ，あるいはグラフをディスプレイすると，wait 状態になるので，実行を続けるためにはこのキーをおさねばならない。

第4章　利用者によるシステムの拡張

4.1　システムの内部構成

ＪUMPS は，処理手続のまとまりとしてのステージと，個々の手法に対応するステップという概念で構成されている。このことは第一章においても述べたとおりであるが，これがシステムのプログラム上には，ＪUMPコマンドによって各ステージにコントロールをわたすトータルディストリビューター（以降はこれを略してＴＤＩＳとよぶ）と，ＴＤＩＳによって起動されてそのステージ内のコマンドを識別して各コマンド処理ルーティンにコントロールをわたすステージディストリビューター（以降はこれを略してＳＴＤＩＳとよぶ）に対応している。ＳＴＤＩＳはステージの数だけ存在し，それぞれに別々の名前がついてＪＵＭＰコマンドの名前に対応するＳＴＤＩＳが選択されるわけである。ＪＵＭＰＳはＨＩＴＡＣ８４００のＣＧＯＳの管理下にあり，ＪＵＭＰＳで使用可能のエリアは約１７０Ｋバイトであるため，ＳＴＤＩＳの段階でオーバーレイをおこなっている。そこでＴＤＩＳがＪＵＭＰコマンドをうけとるとＳＴＤＩＳをローディングしてコントロールをわたし，ＳＴＤＩＳが次のＪＵＭＰコマンドをうけとっても一応ＴＤＩＳへリターンしてコマンド解読がおこなわれる。

図4.1に各モジュールの構成を示す。

4.1.1　イニシャライズルーティン

ＪＵＭＰＳ がローディングされた時に最初にコントロールがわたされるルーティンであって初期設定と最終処理をおこなう。

初期設定としては，

1. オリジナルファイル，セレクトファイル，ワークファイルのオープン
2. 各種テーブル等の初期設定
3. 中断ＪＯＢ の再開の場合には，テーブル等を中断時の状態に復元する。
4. 割込み処理ルーティンをＧＩＡＳＰに登録する。
5. トータルディストリビュータにコントロールをわたす。

ＪＵＭＰ ＥＮＤ ；コマンドが入力されると，コントロールがトータルディストリビュータから帰り次のような最終処理をおこなう。

6. ＪＯＢ の中断であるかどうかを使用者に聞いて必要とあれば中断処理をおこなう。

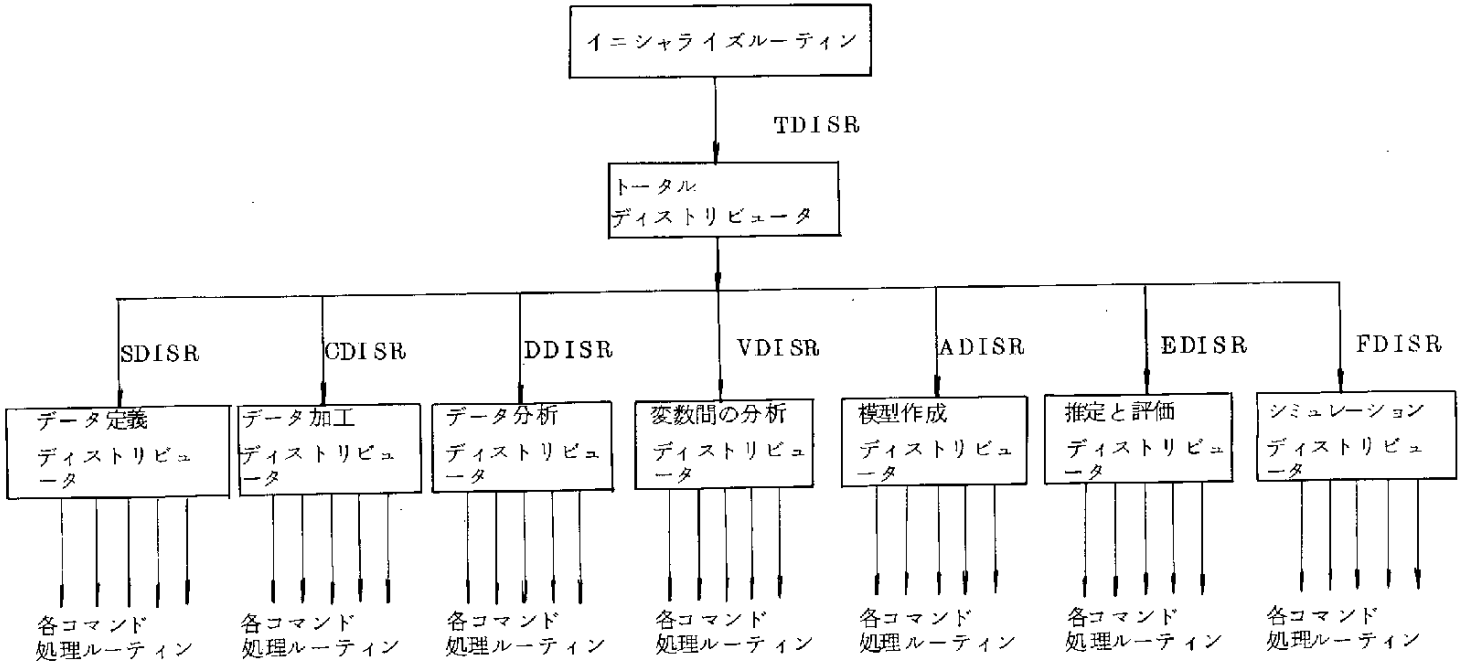


図4.1　JUMPS のモジュール

実際には各テーブルをワークファイルに書きだす。

7. 各ファイルをクローズする。

4.1.2 トータルディストリビュータ

JUMPコマンドを解読することにより，そのコマンドで指定されたSTDIS をローディングしてそれにコントロールをわたす役目を持ち，同時にCOMMONエリアにあるISTAGEに,どのステージにはいったかという情報をセットする。トータルディストリビュータ中で処理が行われている場合はISTAGEは1になっている。

JUMP DDEF；の場合

ISTAGEを2にして，SDISRにコントロールをわたす。

JUMP EDIT；の場合

ISTAGEを3にして，CDISRにコントロールをわたす。

JUMP DATA；の場合

ISTAGEを4にして，DDISRにコントロールをわたす。

JUMP COR；の場合

ISTAGEを5にして，VDISRにコントロールをわたす。

JUMP AUTO；の場合

ISTAGEを6にして，ADISRにコントロールをわたす。

JUMP ESTIMATE；の場合

ISTAGEを7にしてEDISTRにコントロールをわたす。

JUMP SIMUL；の場合

ISTAGEを8にしてFDISRにコントロールをわたす。

JUMP END；の場合

イニシャライズルーティンにリターンする。

トータルディストリビュータは，TDISRとTDISという2個の入口名を持っている。TDISRは，イニシャライズルーティンからの入口であり，リターン番地をセットする。TDISはリターン番地のセットをしない入口である。これは，トータルディストリビュータがユーティリテイルーティンを使用した場合にエラーが起ったとすると，エラーメッセージをディスプレイしてからトータルディストリビュータに帰ってこなければならないが，この時の入口である。

4.1.3 ステージディストリビュータ

JUMPコマンドによってコントロールがわたされると，利用者に対してコマンドを

要求し，指定されたコマンド処理ルーティンにコントロールをわたす役目を持っている。コマンドの処理が終ると再び別の入口名で呼ばれて次のコマンドの処理をしてゆく。このためにSTDISは2つの入口を持っており，一つはTDISからの入口，もう一つは各コマンドあるいはユーティリティルーティンからの入口である。

STDIS はステージの数だけあるが，その構造は全く同じであって，たゞ登録してあるコマンドが異るだけである。

4.1.4 利用者によるステージあるいはステップの追加

利用者が新しくステージあるいはステップを追加する場合には，TDIS あるいはSTDIS の修正をおこなわなければならない。

まずステップの追加を考えてみよう。ステップの追加は，追加したいコマンドを対応するステージのSTDISに追加登録して，そのコマンドの処理ルーティンを作り，それにコントロールをわたすようにすればよい。

① 対応するSTDIS のTBLCNT（そのステージにあるコマンドの数がはいる）の内容を追加したいコマンド数だけ増す。

② TABLE にコマンド名を登録する。

③ TOBIBANにコマンドを処理するルーティンの入口名を登録する。

〔注意〕

TOBIBANとTABLE ではコマンド処理ルーティンの入口と，コマンド名が逆順になっている。

④ コマンド処理ルーティンの最后で，STDISにリターンする。

ステージの追加の場合には

① 利用者が追加したいステージ用のSTDISを作る。

② ステップの追加と同じようにして，各処理コマンドを作る。

③ TDISの中のTBLCNT，TABLE，TOBIBANを修正する。

④ ERROR ルーティンのジャンプテーブルに追加する。（4.2.3表参照）

4.2 ユーティリティ ルーティン

使用者がJUMPS の機能を拡張する場合に必要と思われる各種ユーティリティとして次のようなルーチンが用意されている。

(1) シラブルリード ルーティン（Syllable read routine）

(2) サーチ ルーティン (Search routine)

(3) エラー ルーティン (ERROR routine)

(4) ディスク ファイル アクセス ルーティン (Disk file access routine)

(5) グラフ ディスプレイ ルーティン (Graph display routine)

(6) パラメータ推定ルーティン

以下に各ルーティンの説明を述べる。

4.2.1 シラブル リード ルーティン

呼び出し方 CALL SYLRED

機 能：下記のlabeld Commonエリアに情報をセットする。

COMMON／SYLCOM／ISB(3), IEM, ISORT, ICC, ICCW／

ISB(3) ： セパレータが来る迄のデータの内容がセットされる。

IEM ： 区切り記号が数字コードで入る。

ISORT： ISBに入っている内容の属性を示す。

ICC ： ISBの内容の属性が名前の時にその文字数が入る。

ICCW ： ISBの内容の属性が名前の時にそのWORD数が入る。

ISBの識別コード(ISORTの内容)

1 ： ISBに何も入っていない。ISB(1)～ISB(3)はブランクがつまっている。

2 ： 名前 この場合ISB(1)～ISB(3)に左づめで名前が入っている。なお，名前にはブランクを含む物は許されない。

3 ： 整定数 この場合ISB(1)に値が整数タイプでは入っている。

4 ： 実定数 この場合ISB(1)に値が実数型で入っている。

IEMの内容

0	**	べき
64	␣	ブランク
77	(	左かっこ
78	+	プラス
92	*	乗算記号
93	)	右かっこ
94	;	セミコロン
96	−	マイナス

97	／	除算記号
107	,	カンマ
122	：	コロン
126	＝	等号

上記のいずれにも該当しない区切り記号の場合，IEMは常に192という数字が入っている。

注　シラブルリードルーティンのイニシャライズ

シラブルリードルーティンを使用する場合，下記の方法でこのルーティンをイニシャライズしておかねばならない（JUMPSにおいてはイニシャライズルーティンが行っている）。

```
COMMON/SYLCOM/ISB(3), IEM, ISORT, ICC,
1 ICCW/BLK/ICHAR/BEBLK/IBUFN, IBUFC, IB
2 UF(m)/BLKIN/ISBL, ISBC/IOSEI/IOGSW

IOGSW=0                                   m: バッファ数/4
IBUFN=4m+1
IBUFC=4m
   ↓
CALL SYLRED
```

(例)

CENSUS△DEPTSELLING△△(1961, 1962)△;

上記のデータをシラブルリードルーティンで処理すると次の様になる。

回数	ISB(1)	ISB(2)	ISB(3)	IEM	ISORT	ICC	ICCW
1	CENS	US△△	△△△△	64	2	6	2
2	DEPT	SELL	ING△	77	2	11	3
3	1961			107	3		
4	1962			93	3		
5				94	1		

4.2.2　サーチルーティン

セレクトファイルのname table をサーチして，もしあればセレクトファイル上か

らデータを読みだして，セットしなければ，オリジナルファイルからデータを読み出してセレクトファイルに登録してからデータをセットする。

呼び出し方

CALL SEARCH(KEY, IATR, DATA)

KEY ：データのキーがはいっている場所を示す。一次元で要素が3のアレイでなければならない。

IATR：データの属性がはいる。半語長の整数で，28の要素をもつ一次元アレイでなければならない。

DATA：読み出したデータがはいる。190の要素をもつ一次元アレイで実数でなければならない。

注 指定したキーを持つデータが，オリジナルファイルにもセレクトファイルにもない場合には，エラーメッセージを出して，ステージディストリビュータにかえる。

4.2.3. エラールーティン

エラーメッセージをラインプリンタとグラフィックディスプレイに出力して，ディストリビュータにとぶ。

呼び出し方

CALL ERROR(ERMES, N)

ERMES：エラーメッセージが文字ストリングとしてはいっている配列である。

N ：エラーメッセージの文字数

表 エラールーティンリスト

```
SEQ#  LABEL  FORTRAN  STATMENT
0001         SUBROUTINE  ERROR(ERMES, N)
0002         COMMON/STCOM/, ISTAGE
0003         COMMON/SYLCOM/ISB(3) IEM, ISORT, ICC, ICCW
0004         DIMENSION ERMES(N)
0005         NN=(N+3)/4
0006         WRITE(1HO, 30A4)
0007   100   FORMAT(1HO, 30A4)
0008         CALL GRCHDP(N, ERMES)
0009    10   IF(IEM.EQ.94)GO TO 20
```

```
0010        CALL SYLRED
0011        GO TO 10
      C*
      C***** IF YOU WANT TO ADD USER'S STAGE
      C***** YOU MUST ADD TO THE FOLLOWING JUMP-
            TABLE.
      C*
0012     20 GO TO(1, 2, 3, 4, 5, 6, 7, 8), ISTAGE
0013      1 CALL TDIS
0014      2 CALL SDIS
0015      3 CALL CDIS
0016      4 CALL DDIS
0017      5 CALL VDIS
0018      6 CALL ADIS
0019      7 CALL EDIS
0020      8 CALL FDIS
0021        STOP
0022        END
```

4.2.4 ディスク・ファイル アクセス・ルーティン

一 般 形

(1) CALL FOPEN(USER'S NAME & DATE, FILENAME, RESULT SIGN)

(2) CALL FCLOSE(FILE NAME, RESULT SIGN)

(3) CALL FPURGE(FILE NAME, RESULT SIGN)

(4) CALL FREG(KEY, DATA, KEY & SWITCH, POINTER, RESULT SIGN)

(5) CALL FSERCH(KEY OR POINTER, DATA, KEY & SWITCH, FILE NAME, RESULT SIGN)

(6) CALL RSERCH(KEY, DATA, KEY & SWITCH, RESULT SIGN)

(7) CALL DWR (KEY, DATA, FILE NAME, DATA SIZE, RESULT SIGN)

(8) CALL DFR (KEY, DATA, FILE NAME, DATA SIZE, RESULT SIGN)

このユーティリティ・ルーティンは，HITAC 8400のアセンブリ言語によって記述されており，そのモジュール名は，'JDATA'としてまとめてある。また，ボリューム・シリアル・ナンバー'JUMPOO'のSYSOML上に収録されており，ユーザは他のユーティリティ・ルーティンと同様に使用することができる。以下に各ルーティンの説明をおこなう。

(1) FOPEN：FILE OPEN

このサブルーティンによって，JUMPSに必要なファイルのオープン作業が行われる。このサブルーティンでは，ファイルをオープンするのみでなく，利用者水準のラベル情報を与えたり，チェックしたりする。

コーリング・シーケンス：

CALL FOPEN(USER, I, ISW)

USER：DIMENSION USER(7)で予め領域を確保する必要がある。このうち5ワードは利用者で2ワードは作成年月日である。

I　：1ワードでファイルの種別を指定する。

1→オリジナル・ファイル

2→セレクト・ファイル

3→セーブ・ファイル

4→ワーク・ファイル

ISW ：1ワードでオープン作業の結果を示す。

0→正常オープン

9→異常オープン（ラベル情報に誤りがあるかデバイス・エラーその他）

使用上の注意：

USER 領域は，Iの内容によって以下のラベル情報が授受される。

I＝1のとき

既成のラベル情報（'JUMPS／ORIGINAL-FILE：'の文字列と作成年月日<月，日，年の順>）が格納される。

I＝2のとき

ファイルが，すでに消去（CALL PURGE によって）されている場合，ファイルのラベル情報（'利用者名'＋'作成年月日'をセットしなければ

ならない。

ファイルが消去されてない場合，既成のラベル情報が格納される。

I＝3，I＝4のとき

I＝2に準じた作業を行う。

(2) FCLOSE：FILE CLOSE

このサブルーティンによって，JUMPS に必要なファイルのクローズ作業が行われる。

コーリング・シーケンス：

CALL FCLOSE(I, ISW)

I ： 1ワードでファイルの種別を指定する。

1→オリジナル・ファイル

2→セレクト・ファイル

3→セーブ・ファイル

4→ワーク・ファイル

ISW： 1ワードでクローズ作業の結果を示す。

0→正常クローズ

9→異常クローズ(デバイス・エラー，その他)

(3) FPURGE：FILE PURGE

このサブルーティンによって，JUMPS に必要なファイルの消去作業が行われる。

コーリング・シーケンス：

CALL FPURGE(I, ISW)

I ： 1ワードでファイルの種別を指定する。

2→セレクト・ファイル

3→セーブ・ファイル

4→ワーク・ファイル

ISW： 1ワードで消去作業の結果を示す。

0→正常消去

9→異常消去(デバイス・エラー，その他)

(4) FREG：FILE REGISTER (RECORD WRITE IN)

このサブルーティンによって，JUMPS におけるセレクト・ファイルに所定のフ

ォーマットで書き込みが行われる。

コーリング・シーケンス：

CALL FREG(KEY, DATA, IAIR, N, ISW)

KEY ： DIMENSION KEY(3)で予め領域の確保が必要であり，書き込みレコードのキーワードとなる。

DATA： DIMENSION DATA(181)で予め領域の確保が必要であり，データの存在しない部分は，文字ストリング'0'でパディングしておかなければならない。

IAIR： キーワードとスイッチ情報の領域であり，そのフォーマットは添付資料を参照のこと。

N ： COMMON領域SLTTBLのセレクト・ファイル・キーテーブル上の当該キーワードのロケーションを示すポインター（1ワード）である。

ISW ： 1ワードで書き込み作業の結果を示す。

0→正常書き込み

1→書き込みファイル領域がオーバーフロ

9→異常書き込（KEYのキーとIAIRのキーとが等しくないとき，デバイスエラー，その他）

(5) FSERCH：FILE SEARCH(RECORD READ OUT)

このサブルーティンによって，JUMPSにおけるオリジナル・ファイル，セレクト・ファイルから当該レコードを所定のフォーマットで読み出しが行われる。

コーリング・シーケンス：

CALL FSERCH(KEY, DATA, IAIR, I, ISW)

KEY ： DIMENSION KEY(3) 予め領域の確保が必要であり，オリジナル・ファイルの場合，読み出し情報のキーワードであり，セレクト・ファイルの場合，当該キーワードのキーテーブル上のポインターを，KEY(1)のところに入れておかねばならない。

DATA： DIMENSION DATA(190)で予め領域の確保が必要であり，この領域にデータ部分が格納される。また，データの存在していない部分は，文字ストリング'0'でパディングしてある。

IAIR： キーワードとスイッチ情報の領域であり，そのフォーマットは，添付資料を参照のこと。

I　　： 1ワードでファイルの種別を指定

1→オリジナル・ファイル

2→セレクト・ファイル

ISW ： 1ワードで読み出し作業の結果を示す。

0→正常読み出し

1→該当レコードなし

9→異常読み出し（デバイスエラー，その他）

使用上の注意：

DATA 領域は，I＝1とI＝2とでは内容が異なるので注意すること。すなわち，I＝1のときは，最大190ワードのデータが含まれるが，I＝2のときは，最大181ワードで残余9ワードは，文字ストリング'0'でパディングされている。

(6) RSERCH：FILE RESEARCH（ANALOGOUS RECORD READ IN）

このサブルーティンによって，JUMPSにおけるオリジナル・ファイルから当該キーワードに最も近いレコードを所定のフォーマットで読み出しが行われる。

コーリング・シーケンス：

CALL RSERCH(KEY, DATA, IAIR, ISW)

KEY ： DIMENSION KEY(3) で予め領域の確保が必要であり，このキーワードに最も近似したキーワードが読み出される。

DATA： DIMENSION DATA(190)で予め領域の確保が必要であり，KEYで示された，キーワードに近似したキーワード＊のデータが入る。

IAIR： 読み出されたレコードのキーワードとスイッチ情報の領域であり，そのフォーマットは，添付資料を参照のこと。

ISW ： 1ワードで読み出し作業の結果を示す。

0→正常読み出し

＊近似したキーワードについて

近似したキーワードは，当該サブルーティンの直前でコールされたFSERCH サブルーティンのキーワードとする。そして，そのキーワードの第1シラブル（1ワード）が当該ルーティンのKEYにおける第1シラブルと等しいものである。

1→該当レコードなし

9→異常読み出し（デバイスエラー，その他）

(7) DWR : DISC FILE WRITE

このサブルーティンによって，JUMPSにおけるセーブ・ファイル，ワーク・ファイルに対して所定のフォーマットで書き込みが行われる。

コーリング・シーケンス：

CALL DWR(KEY, DATA, I, N, ISW)

KEY : DIMENSION KEY(3) で予め領域の確保が必要であり，当該データの指標となる。

DATA: DIMENSION DATA(N)で予め領域の確保が必要であり，この領域が当該ファイルのレコードサイズに従って分割格納される。

I : 1ワードでファイルの種別を指定

3→セーブ・ファイル

4→ワーク・ファイル

N : 1ワードでDATA 領域の大きさをワード数で指示

ISW : 書き込み作業の結果を表示

0→正常書き込み

1→書き込みファイル領域がオーバーフロー

9→異常書き込み（デバイスエラー，その他）

(8) DFR: DISC FILE READ

このサブルーティンによって，JUMPSにおけるセーブ・ファイル，ワーク・ファイルに対して所定のフォーマットで読み出しが行われる。

コーリング・シーケンス：

CALL DFR(KEY, DATA, I, N, ISW)

KEY : DIMENSION KEY(3)で予め領域の確保が必要であり，読み出すべきデータの指標である。

DATA: DIMENSION DATA(N)で予め領域の確保が必要であり，この領域に当該データが格納される。

I : 1ワードでファイルの種別を指定

3→セーブ・ファイル

4→ワーク・ファイル

N　：　1ワードでDATA 領域の大きさをワード数で表示される。

ISW　：　読み出し作業の結果を表示

0→正常読み出し

1→該当データなし

9→異常読み出し（デバイスエラー，その他）

4.2.5　グラフ・ディスプレー　ルーティン

グラフディスプレー　ルーティンは以下に述べるものから構成される。

(1)　GJOPEN　ROUTINE

機　能：グラフのディスプレイのイニシャライズを行う。

形　式：CALL　GJOPEN(IFIG, ID, N)

IFIG：　グラフの番号（整数型）

ID　：　グラフの名前（EBCDIKコード）

N　：　名前の文字数

例

CALL　GJOPEN(100,　5HGRAPH, 5)

出力形式

GRAPH　NO=100　　GRAPH　NAME：GRAPH

注) グラフ番号及びグラフの名前を図面下部に表示する。又ディスプレイのイニシャライズを行う。

名前の文字数N，又名前にブランクがある場合その数をBとするとき次の条件を満足すること。

$$N+B \leq 17$$

又グラフの番号は次の条件を満足すること。

$$IFIG < 10^7$$

(2)　GJAXIS　ROUTINE

機　能：グラフの座標軸を描く。

形　式：CALL　GJAXIS(Xmax, Xmin, Ymax, Ymin, ISW1, ISW2, UNIT)

Xmax　：　X軸データの最大値

Xmin　：　X軸データの最小値

Ymax ： Y軸データの最大値

Ymin ： Y軸データの最小値

ISW 1： 1　時系列データでない時

2　時系列データの時

ISW 2： 1　X軸－Y軸ともに等間隔座標

2　X軸だけが対数

3　Y軸だけが対数

4　X軸－Y軸ともに対数

UNIT

UNIT (2) ： X軸の単位

UNIT (3)
UNIT (4)　X軸の名前（12文字以内）
UNIT (5)

UNIT (7) ： Y軸の単位

UNIT (8)
UNIT (9)　Y軸の名前（12字以内）
UNIT (10)

但し UNIT(1), UNIT(6) は将来の機能拡張の為のエリアで，現在は，意味をもっていない。

Xmax, Xmin, Ymax, Ymin は次の条件を満足していれば，必ずしもデータ中の最大，最小値でなくてもよい。データ中のX，Y軸の最大，最小値を各々DXmax, DXmin, DYmax, DYmin とすると，

$$Xmax \geq DXmax$$
$$Xmin \leq DXmin$$
$$Ymax \geq DYmax$$
$$Ymin \leq DYmin$$

Xmax, Xmin, Ymax, Yminによって，X，Y軸は適当にスケーリングされる。

データ形式

全てのデータは実数を用いる。

(i) 時系列データでない場合

データは全て実数を用いる。

(ii) 時系列データの場合

(1) GJAXIS ROUTINE で扱う時系列は次の4種類とする。

年データ

半期データ

四半期データ

月次データ

(2) データ構造

(イ) 年データ

データとして，年の下2桁のみを記述すること。

例 1950 → 50.

1971 → 71.

(ロ) 半期データ

年については，(イ)と同様に取扱い，10^4 桁を2とし，10^3 桁で上期(1)，下期(2)を指示する。（以下年は下2桁指示とする。）

例 1950年，上期 → 21050.

1971年，下期 → 22071.

(ハ) 四半期データ

10^4 の桁を4とし，10^3 の桁で，1期(1)，2期(2)，3期(3)，4期(4)，を指示する。

例 1960年1期 → 41060.

1965年4期 → 44065.

(ニ) 月次データ

月は，10^3，10^4 の桁に右詰で指示する。

例 1955年 1月 → 1055.

1968年12月 → 12068.

(3) GJPUT ROUTINE

機 能：グラフを描く

形 式：CALL GJPUT(X, Y, N, IDUMMY, INT, K)

X ： データのX座標

Y　　：データのY座標

N　　：データの個数

IDUMMY：注1)

INT　：1　高輝度
　　2　中輝度
　　3　低輝度
　　4　ブランク
　　} グラフの輝度

K　　：1　(X，Y)で与えられた点に印(×)を記す。
　　2　(X，Y)に印(×)を記さない。

データ(X，Y)が配列で与えられる場合，つまりNが2以上の場合，INTで指示される輝度に従って表示されるが，(X，Y)が配列で与えられない場合(N＝1)，INTの指示には従わないで，グラフの輝度はブランクとなり，又(X，Y)はKの指示によって印される。又，INT＝4の場合，必ず(X,Y)点は印(×)が記される。

(4) GJCLOS ROUTINE

機　能：グラフ処理のクローズルーチン(CLOSE ROUTINE)

形　式：CALL GJCLOS

(5) グラフのディスプレー

上記(1)～(4)のルーチンを次の順序で実行することによってグラフは表示される。

```
CALL GJOPEN    ：グラフのディスプレイのイニシャライズ
CALL GJAXIS
CALL GJPUT
                  X，Y軸を描く。
CALL GJAXIS       グラフを描く。
CALL GJPUT
      ⋮
CALL GJCLOS    ：グラフ処理のクローズ
```

注1) IDUMMY

IDUMMYは将来の機能拡張の為のエリアで現在は意味をもっていないが，必ず整数を入れること。

4.2.6　パラメータ推定ルーティン

(1)　DLSMルーティン

モデル式のパラメータ推定を直接最小二乗法において行う呼び出し方

CALL DLSM(A, N, IP, M, BETA, SD, DW, IS, ISW)

入　力　A　：　データエリア(1次元配列)

N　：　パラメータ1個当りのデータ数を示す。

IP　：　A中各パラメータ及び従属変数のデータがどこから始まるかの情報が入っている(1次元配列)の配列

M　：　IPの大きさ

IS　：　定数項推定を行うか否かを示す。

0：行わない

1：行う

出　力

BETA：　推定された係数(1次元配列)

ISW　：　解の状態を示す

0　：　NORMAL END

1　：　解が求まらなかった

SD　：　残差平行和

DW　：　ダービンワトソン比

例　右式の係数を推定するとする

$$Y_1 = \beta_0 + \beta_1 X_{11} + \beta_2 X_{12} + \beta_3 X_{13} \cdots + \beta_{M-1} X_{1,M-1}$$

$$Y_2 = \beta_0 + \beta_1 X_{21} + \beta_2 X_{22} + \beta_3 X_{23} \cdots + \beta_{M-1} X_{2,M-1}$$

$$Y_3 = \beta_0 + \beta_1 X_{31} + \beta_2 X_{32} + \beta_3 X_{33} \cdots + \beta_{M-1} X_{3,M-1}$$

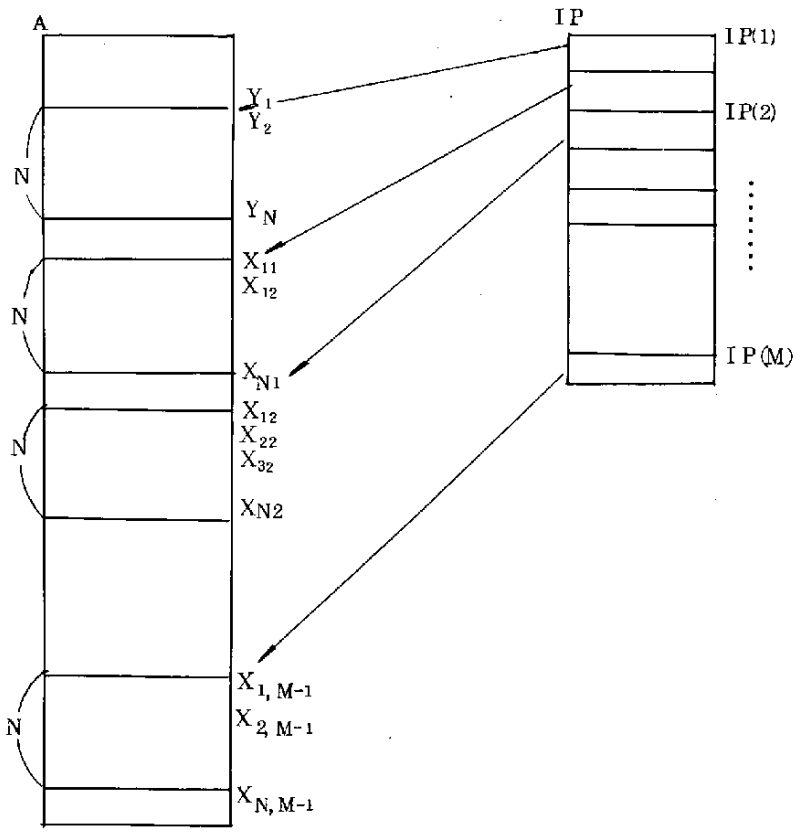
$$\vdots$$

$$Y_N = \beta_0 + \beta_1 X_{N,1} + \beta_2 X_{N,2} + \beta_3 X_{N,3} + \cdots \beta_{M-1} X_{N,M-1}$$

IS　：　1　　β_0 は指定する。

IS　：　0　　β_0 は指定しない。

入　力　TABLE



(2) IDLSM ルーティン

適度認定であるモデル式のパラメータ推定を間接最小二乗法によりおこなう。

呼び出し方

CALL IDLSM (SD, DW)

SD ： 推定した式に対し全体テストを行なった結果の標準偏差がはいる。一次元配列で式の個数の要素をもたねばならない。

DW ： 同じくダービンワトソン比がはいる。一次元配列で式の個数の要素をもたねばならない。

メインルーティン側からわたす情報は式テーブル，同時従属変数テーブル，先決変数テーブル，データテーブルであってコモンでわたされる。

(3) TSLSM ルーティン

過剰認定であるモデル式のパラメータ推定を二段階最小二乗法によりおこなう。

呼び出し方

CALL TSLSM(SD, DW)

わたされるパラメータおよび情報はすべてIDSLMルーティンと同じである。

付 録

1. セレクトファイル テーブル
2. 変数分析表および相関表
3. 式を表現するための各種テーブル
   a 式テーブル
   b 同時従属変数テーブル
   c 先決変数テーブル
   d データテーブル
   e テーブル間の関係
4. キーワードとスイッチ情報について
5. ファイルの仕様について
6. エラーメッセージ一覧

1. JUMPS で使用しているテーブル

JUMPS のセレクトファイル キーテーブル

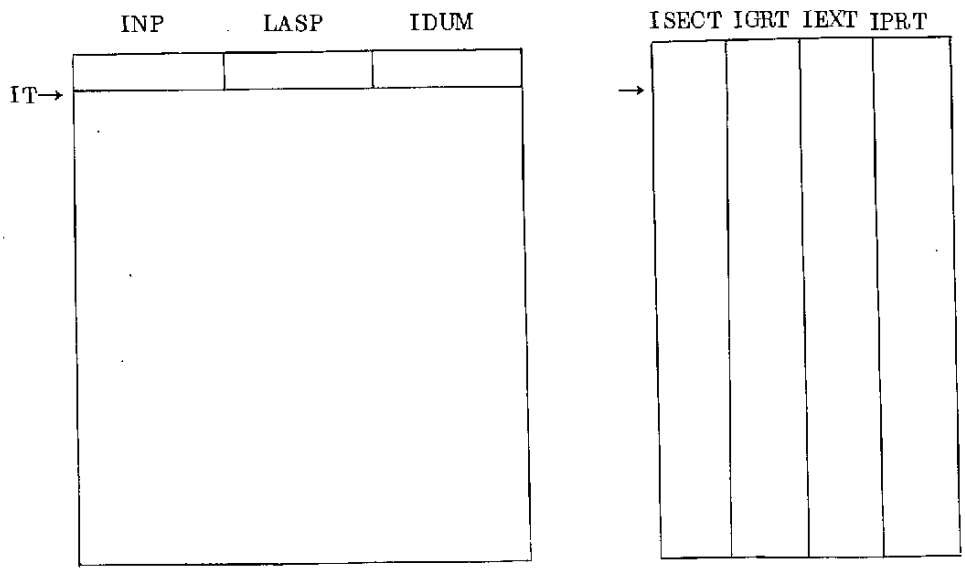


表 5. 1

INP ： 今度使用できるテーブルエリアのポインター。
FOPENをかけた時点では１にセットされる。

LASP ： テーブルの大きさ＋１がはいる。

IDUM ： 使用せず。初期状態は０になっている。

I T ： テーブルのキー部分がはいる場所で１２バイト分ある。

ISECT: その変数のセクター番号がはいる。

IGRT ： その変数のグループ番号がはいる。

IEXT ： その変数の属性
０：undefine　１：内生変数　２：外生変数

IPRI ： その変数のプライオリティがはいる。

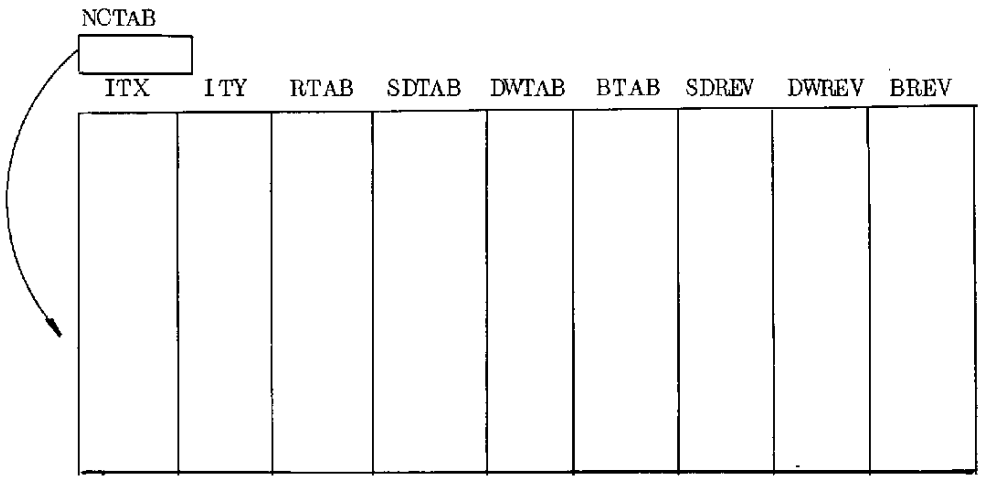
IGC ： グループコードカウンターで現在までに使用されたグループの個数がはいる。

このテーブルは，ＳＬＴＴＢＬという名前のCOMMONにとられていて，その順序は次のとおりである。

```
COMMON／SLTTBL／INP，IASP，IDUM
  IT(900)，IGC，ISECT(300)，IGRT(300)，IEXT(300)
  IPRI(300)
INTEGER ISECT*2，IGRT*2，IPRI*2
```

2. 変数分析表および相関表

表 5. 2



NCTAB ： 現在使用可能なテーブル領域の先頭が示されている。
新規ジョブ開始のとき1がセットされる。

ITX ： $Y=\alpha+\beta X$という回帰式でのXにあたる変数のセレクト・ファイル・キーテーブル上の位置を示している。

ITY ： $Y=\alpha+\beta X$という回帰式でのYにあたる変数のセレクト・ファイル・キーテーブル上の位置を示している。

RTAB ： ITX と ITY の両変数間の相関係数である。

SDTAB ： ITX と ITY の両変数での$Y=\alpha+\beta X$という回帰式の誤差の標準偏差である。

DWTAB ： ITX と ITY の両変数での$Y=\alpha+\beta X$という回帰式のダービィン・ワトソン比である。

BTAB ： ITX と ITY の両変数での$Y=\alpha+\beta X$という回帰式のβである。

SDREW ： ITX と ITY の両変数での$X=\alpha+\beta Y$という回帰式の誤差の標準偏差である。

DWREV ： ITX と ITY の両変数での$X=\alpha+\beta Y$という回帰式のダービィン・ワトソン比である。

BREV ： ITX と ITY の両変数での$X=\alpha+\beta Y$という回帰式のβである。

このテーブルは，CORTAB という名前のCOMMONエリアにとられており，その大きさはつぎのとおりである。

COMMON／CORTAB／NCTAB，ITX(1000)，ITY(1000)，RTAB(1000)，SDTAB(1000)，DWTAB(1000)，BTAB(1000)，SDREV(1000)，DWREV(1000)，BREV(1000)

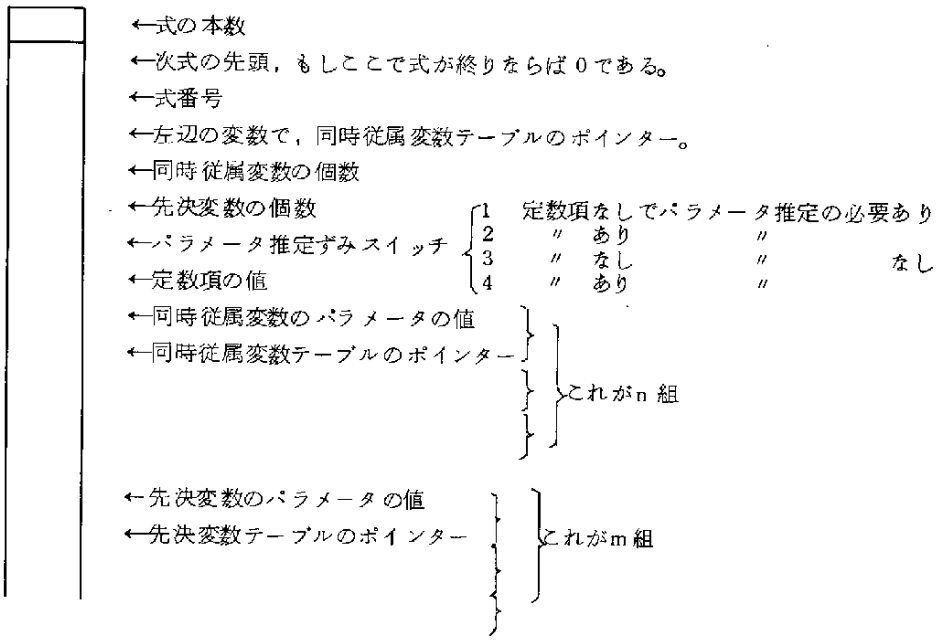
3. 式を表現するための各種テーブル

a. 式テーブル

式テーブルは，モデル作成，予測，シミュレーション等で利用する式のパターンを憶えておくものである。式を憶えておくさいに式の本数，次の式のある場所，式番号，左辺の変数のあり場所，同時従属変数の数(n)，先決変数の数(m)，パラメータと変数のあり場所をn組，先決変数に対するパラメータと変数のあり場所を

m組という順序でテーブルにのせておく。

表 5. 3



このテーブルは，EXPTBLという名前のCOMMONにあり，これをFORTRANで書けば

COMMON／EXPTBL／IEXPT(500)

DIMENSION AIEXPT(500)

EQUIVALENCECAIEXPT(1), IEXPT(1))

たゞし，この場合テーブルサイズは500で，パラメーターを憶えておくためのエリアとして使う場合は，AIEXPT の名前を使用し，他の場合にはIEXPTという名前で使用する。

b. 同時従属変数テーブル

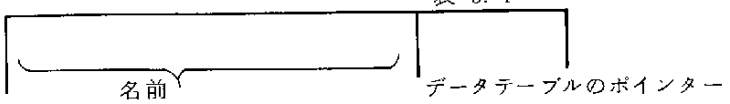
同時従属変数の名前とデータのあり場所を示すテーブルである。

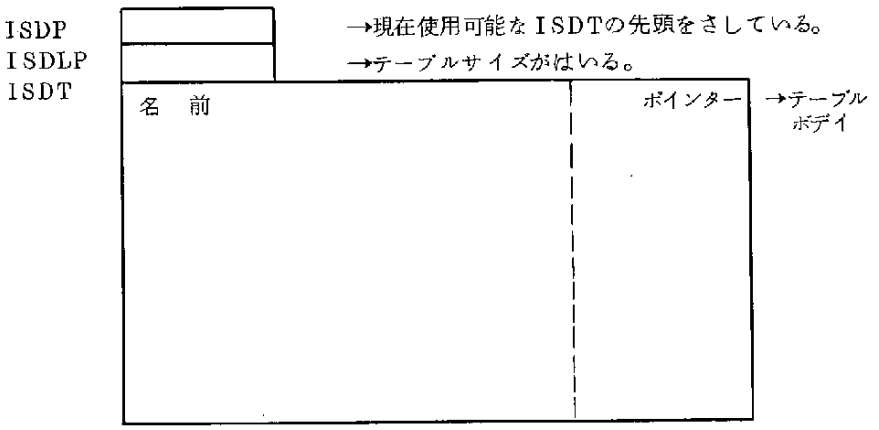
このテーブルは，ISDTBLという名前のCOMMONにあり，これをFORTRANで書けば

COMMON／ISDTBL／ISDP，ISDLP，ISDT(4,200)となる。

たゞしこの場合テーブルサイズは200個である。

表 5.4

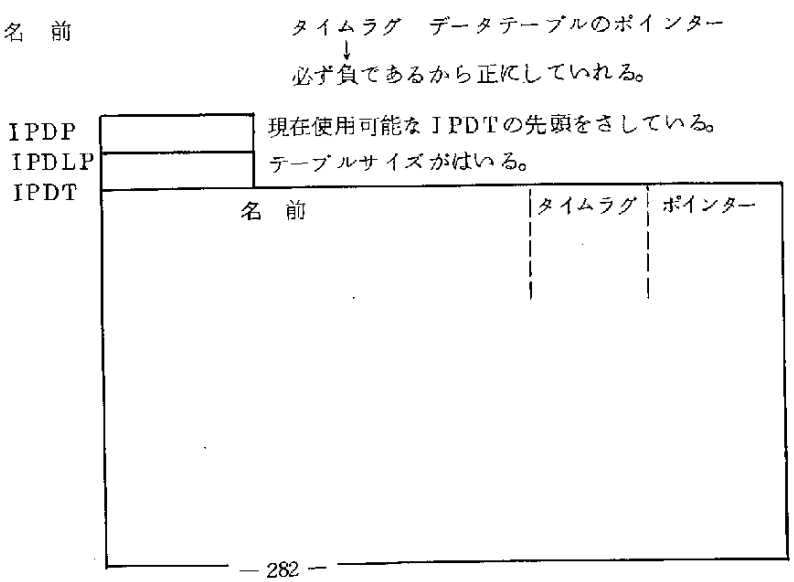




c．先決変数テーブル

外生変数名あるいはタイムラグのある内生変数名，タイムラグ，データのあり場所を示すテーブルである。

表 5.5



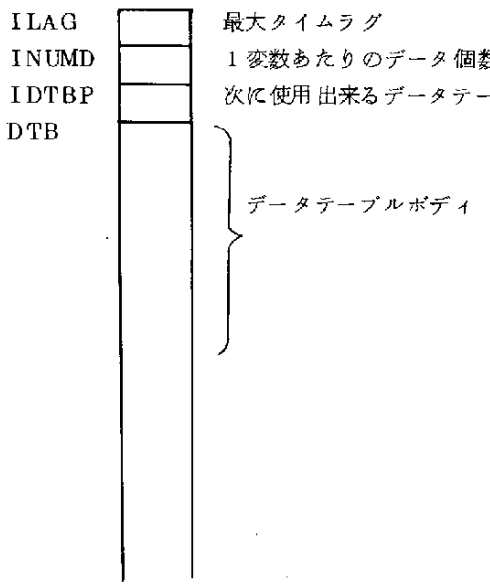
このテーブルは，IPDTBLという名前のCOMMONあり，これをFORTRAN で書けば，

COMMON／IPDTBL／IPDP，IPDLP，IPDT(5,200)となる。たゞし，この場合テーブルサイズは200個である。

d．データテーブル

データテーブルは方程式のパラメーター推定に必要なデータを入れておくためのものである。

表 5. 6



このテーブルは，DTBLCという名前のCOMMONにあり，これをFORTRAN で書けば

COMMON／DTBLC／ILAG，INUMD，IDTBP，DTB(2000)となる。たゞしこの場合テーブルサイズは2000である。

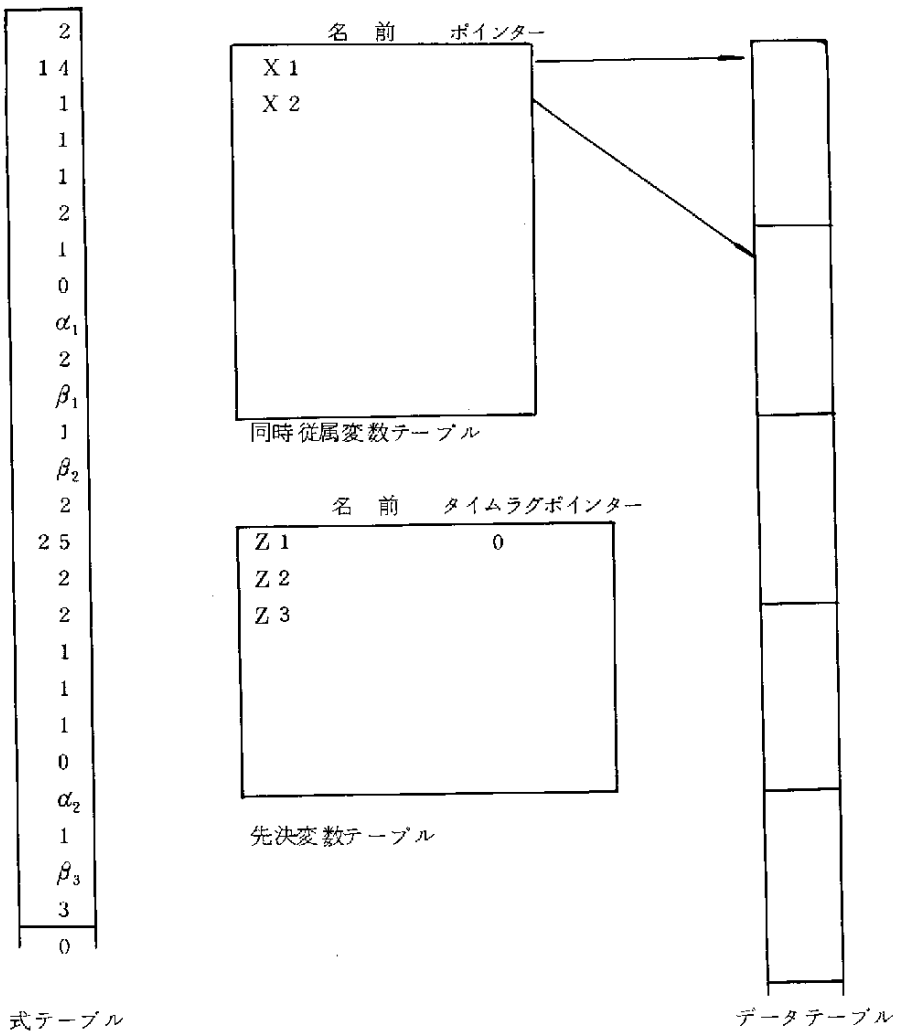
e．テーブル間の関係

$X1 = \alpha_1 X_2 + \beta_1 Z_1 + \beta_2 Z_2$

$X2 = \alpha_2 X_1 + \beta_3 Z_3$

を例としてテーブル間の関係を図示してみよう。

表 5.7



4. キーワードとスイッチ情報（IAIR）について

ここでいうスイッチ情報とは，オリジナル・ファイル，セレクト・ファイルに存在する各データの仕様を表示している情報のことであり，各データの一部分として含まれている。

4.1 IAIR 領域の仕様：

INTEGER IAIR*2, JSW*2

```
DIMENSION IAIR(34), JSW(8), LSW(2), ASW(8)
EQUIVALENCE(IAIR(7), JSW(1)), (IAIR(15),
                LSW(1)), (IAIR(19), ASW(8))
```

4.2 IAIR 領域図：

表 5. 8

IAIR(1)	キーワード
IAIR(7)	JSW(1) JSW(2)
	JSW(3) JSW(4)
	JSW(5) JSW(6)
	JSW(7) JSW(8)
IAIR(15)	LSW (1)
	LSW (2)
IAIR(19)	ASW (1)
	ASW (2)
	ASW (3)
	ASW (4)
	ASW (5)
	ASW (6)
	ASW (7)
	ASW (8)

4.3 各領域の説明：

ISW(1) ： データ開始年

時系列データ　1901 ～ 2099の範囲

その他データ　　1 ～ 32768の範囲

ISW(2) ： データ開始月日

年次　0

半期　1, 2

四半期　1, 2, 3, 4

日次　1, 2, ……, 11, 12

日次　101, 102, ……, 1231

その他　0

ISW(3)　：データ終了年

データ開始年と同じ

ISW(4)　：データ終了月日

データ開始月日と同じ

ISW(5)　：データの単位を数値で表示

1　KM

2　KM^2

3　M^3

4　TON

5　円

6　ドル

7　その他

ISW(6)　：データが10^nのスケールで表示されているとき，そのnが表示されている。

ISW(7)　：データの種別を数値で表示

0　ストック・データ

1　フロー・データ

ISW(8)　：データの属性を数値で表示

1　年次データ

2　半期データ

3　四半期データ

4　月次データ

5　日次データ

6　その他データ

LSW(1)　：データの個数を表示

LSW(2)　：データ分析結果を表示

ASW(1)　：

ASW(2)　：　データ分析結果の各種情報

ASW(3)　：

ASW(4) ：

ASW(5) ：

ASW(6) ： データ分析結果の各種情報

ASW(7) ：

ASW(8) ：

（注） LSW(2)，ASW(1)～ASW(8)は，データ分析以前では，オールビット・オフの状態にセットされている。

5. ファイルの仕様について

5.1 ディスク・パックにおける各ファイルの割当て：

表 5.9 割 当 図

CHC / HO	1	6	11	16	141	146	201
0	VTOC	JMPSVE	JMPWRK	JMPORG／INDEX	JMPORG DATA AREA OVER FLOW AREA	JMPORG／OVERFLOW	JMPSLS	ALTANATE／TRACK
1								
2								
3								
4								
5								
6								
7								
8								
9								
	1	5	5	5	125	5	55	2

ボリューム：シリアル・ナンバー：J00000

所有者ID：JUMPS／FILE

ファイル名：

JMPSVE ： セーブ・ファイル

JMPWRK ： ワーク・ファイル

JMPORG ： オリジナル・ファイル

JMPSLT ： セレクト・ファイル

（注）上述の例は，JIPDEC で用意しているディスク・パックの場合であるが，利用者は，ボリューム・シリアル・ナンバーを変更することによって利用者専用のファイルを保有することができる。

5.2　ファイルの構成

1. オリジナル・ファイル

1) ラベル・レコード

表 5.10

キーワード (スペース)	'JUMPS/ORIGINAL'	6.0	創成期日	更新アクセス		再成アクセス			最新アクセス	
				回数	最終日	回数	最終日		回数	期日
3語	4語	1語	2語	半語	3半語	半語	3半語		半語	3半語

200語

2) データ・レコード

表 5.11

キーワード	スイッチ情報	データの長さ	データ	本体	最新アクセス	
					回数	期日
3語	8半語	1語	190語		半語	3半語

200語

2. セレクトファイル

1) ラベル・レコード

表 5.12

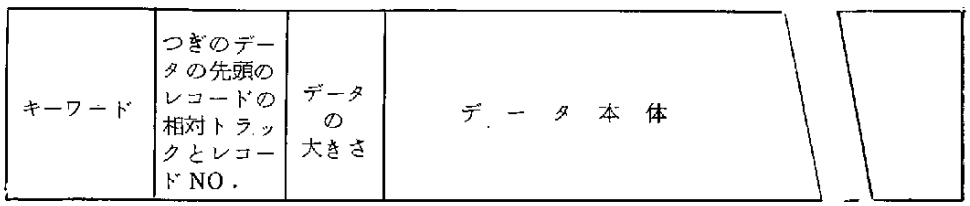
利用者名	作成期日	リザーブエリア	テーブル指標			キーワード1	キーワード2
			ポインター	範囲	ダミー		
5語	2語	2語	3語			3語	3語

1レコード＝198語

3. セーブ・ファイル

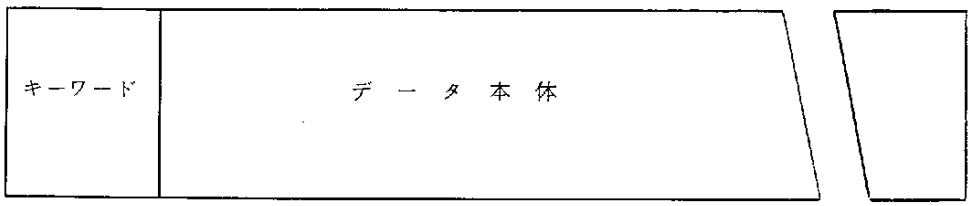
1) データ・レコード

表 5.14



3 語　　2 語　　1 語　　397 語

403 語



3 語　　400 語

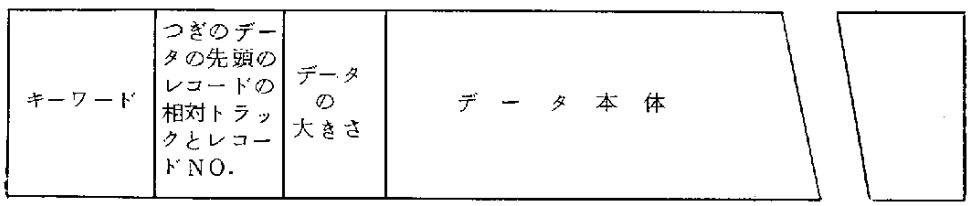
403 語

最後のデータであれば，つぎのデータの先頭レコードの位置を示す相対トラックとレコードNOがともにゼロ となる。

4. ワーク・ファイル

1) データ・レコード

表 5.15



3 語　　2 語　　1 語　　397 語

403 語

キーワード	データ本体
3 語	400 語
	403 語

最後のデータであれば，つぎのデータの先頭レコードの位置を示す相対トラックとレコードNOがともにゼロとなる。

6. エラーメッセージ一覧表

6.1 データ定義ステージのエラーメッセージ

1. DEFINE COMMAND ERROR. DEFINE コマンドの間違い。
2. SUB-COMMAND ERROR. EXO, ENDO, UND コマンドの間違い。
3. PRIORITY MUST BE LESS THAN 4. プライオリティは0～3までしか使えない。

6.2 データカロエステージのエラーメッセージ

1. DATA NAME IS INVALID. 左辺の変数名に当るものが英字名でない。
2. EQUAL SIGN IS NOT FOUND. 等号がみつからない。
3. PERIOD SPEC. IS INVALID. 期間の指定に誤りがある。
4. DELIMITER ERROR IS OCCURRED. 等号のあるべき位置に変数名もしくは数値がある。
5. NOW DAY-DATA IS NOT TREATED. 現バージョンでは，ロ次データは使用できないのに，この種のデータが入力された。
6. FUNCTION ARGUMENT IS INVALID. 関数の実引数に誤りがある。
7. OPERATOR ERROR IS OCCRRED. 演算記号に誤りがある。
8. ERROR IN RIGHTSIDE OF EXPRESSION. 右辺に1つも変数名が出現していない。

9.	TOO MANY OPERANDS IN EXPRESSION.	演算記号と演算要素の対応がおかしい。
10.	WRITE ERROR IS OCCURRED.	セレクト・ファイルにデータを書く際，WRITE ERROR が生じた。
11.	TIME LAG IS INVALID.	タイム・ラグに誤りがある。
12.	DATA RANGE IS NOT SATISFIED.	右辺の変数に指定期間のデータが足りない。
13.	DATA RANGE ERROR.	変数の期間がおかしい。
14.	TOO MANY ITEMS IN VARIABLE.	期間の指定が長すぎて，1変数名に関して，そのデータ個数が180個を越えた。
15.	THE 2ND RANGE ERROR.	半期，四半期，月次データにおいて期または月の指定に誤りがある。
16.	EDIT COMMAND FORMAT ERROR.	命令形式に誤りがある。
17.	EDIT COMMAND VAR1 ERROR.	VAR1の名前，又はVAR1での期間指定に誤りがある。
18.	EDIT COMMAND VAR2 ERROR.	VAR 2 の指定に誤りがある。
19.	EDIT COMMAND ATTRIBUTE ERROR.	属性の指定に誤りがある。

6.3 データ分析ステージのエラーメッセージ

1.	TREND PARAMETER ERROR.	傾向分析のコマンドパラメタの間違い。
2.	POLYNOMIAL TREND CANNOT SOLVE.	多項式近似した場合にその係数を求めることができない。
3.	EXPO TREND CANNOT SOLVE.	指数近似した場合に，その係数を求めることができなかった。
4.	MEXPO TREND CANNOT SOLVE.	修正指数近似をした場合に，その係数を求めることができなかった。
5.	GRWTH TREND CANNOT SOLVE.	生長曲線近似をした場合に，その係数を求めることができなかった。

6.	GOMP TREND CANNOT SOLVE.	ゴンペルツ曲線近似をした場合に，その係数を求めることができなかった。
7.	LOGISTIC TREND CANNOT SOLVE.	ロジスティック曲線近似した場合に，その係数を求めることができなかった。
8.	CYCLE PARAMETER ERROR.	周期解析のコマンドパラメタの間違い。
9.	SEASON PARAMETER ERROR.	季節調整のコマンドパラメタの間違い。
10.	DATA ATTRIBUTE ERROR.	季節調整するデータが年，あるいはその他の属性をもつデータである。
11.	CENSUS GRAPH COMMAND ERROR.	センサスにおいて，グラフィック表示コマンドにS，I，TCM，NO以外のコマンドが指示された。
12.	CENSUS DATA ATTRIBUTE ERROR	センサスの対象となるデータが月次データでない。
13.	SHORT OF CENSUS DATA.	センサスの対象となるデータが60個に満たない。
14.	ILLEGAL GPFS COMMAND.	GPFSのコマンドパラメタの間違い。
15.	ILLEGAL VALUE OF ALPHA IN GPFS.	GPFSの予滑定数が $0<\alpha<1$ ではない。

6.4 変数間の分析の分析ステージのエラーメッセージ

1.	COR COMMAND ERROR.	CORコマンドの形式および内容の間違い。
2.	COR COMMAND TOO MANY ARG..	CORコマンドの指定項目が多過ぎる。
3.	COR COMMAND COR-ITEM OVER 50.	セクター番号指定で表示すべき組合せが50以上。
4.	COR COMMAND ILLEGAL SECTOR NUMBER.	登録されていないセクター番号の指定があった。
5.	COR COMMAND ONLY ONE VARIABLE.	変数が1個のため変数間の分析が不可能。
6.	COR COMMAND ATTRIBUTE UNMATCHED.	変数間の属性一致していない。

7. COR COMMAND NO COMMON DATA IN PERIOD. 変数間で共通期間のデータが存在しない。
8. CORRELATION TABLE OVER-FLOW. 変数分析表および相関表に登録余地がない。

6.5 模型作成ステージのエラーメッセージ

1. SHOW COMMAND ERROR. SHOW コマンドの形式および内容の間違い。
2. VAR IS NOT FOUND ON KEY-TABLE. 指定変数名がセレクトファイルキーテーブルにない。
3. VAR IS NOT FOUND ON COR-TABLE. 指定変数名が変数分析表および相関表にない。
4. MCON COMMAND ERROR. MCON コマンドの形式および内容の間違い。
5. MCON COMMAND TOO MANY ARG.. MCON コマンドの指定項目が多過ぎる。

6.6 推定と評価ステージのエラーメッセージ

1. PERIOD COMMAND ERROR. PERIOD コマンドの間違い。
2. EQ COMMAND ERROR. EQ コマンドの間違い。
3. EXPRESSION TABLE OVER. 式テーブルのオーバーフロー。
4. EXPRESSION NUMBER IS NOT FOUND. 指定された式番号が見つからない。
5. DEFINE COMMAND ERROR. DEFINE コマンドの間違い。
6. EXPRESSION IN DEF COMMAND IS INVALID. DEF コマンドの中の式がおかしい。
7. DLS COMMAND ERROR. DLS コマンドの間違い。
8. IDLS COMMAND ERROR. IDLS コマンドの間違い。
9. MODEL IS NOT JUST IDENTIFIABLE. モデルが適度認定でない。
10. TSLS COMMAND ERROR. TSLS コマンドの間違い。

6.7 シミュレーション ステージのエラーメッセージ

1. EXPOL COMMAND DATA IS NOT SATISFIED. EXPOL コマンドにおいて予測のためのデータが不足している。

2. MODEL IS NOT FOUND　　モデル式が存在しない。

3. EXPOL COMMAND ERROR.　　EXPOLコマンドのパラメタの間違い。

4. EXCD COMMAND ERROR.　　EXCDコマンドのパラメタの間違い。

5. DATA COUNT UNMATCH.　　データ期間の指定のデータ個数があわない。

6. EXPRESSION NUMBER IS NOT FOUND.　　指定した式番号式が存在しない。

7. SPECIFIED VARIABLE IS NOT FOUND.　　指定した変数が式の中に存在しない。

8. CHANGE COMMAND ERROR.　　CHANGE コマンドのパラメタの間違い。

6.8 その他のエラーメッセージ

1. ILLEAGAL COMMAND DESCRIPTION.　　コマンド記述のあやまり。

2. ILLEAGAL COMMAND NAME.　　コマンド名の間違い。

3. ILLEAGAL STAGE NAME.　　ステージ名の間違い。

4. SPECIFIED NAME IS NOT FOUND ON FILE.　　指定した変数がファイルにない。

5. SELECT-FILE OVER.　　セレクトファイル　テーブルのオーバーフロー。

6. VARIABLE NAME MUST BE THERE.　　変数名がなければならない所に変数名以外のものがある。

7. INVALID TIME SPECIFICATION.　　データの持っている属性と変数のあとカッコでくくって指定した開始時点あるいは終了時点の書き間違い。

8. REVIEW COMMAND ERROR.　　REVIEWコマンドの間違い。

第3編
オンラインシミュレーション言語

目　次

まえがき

現在我々が計画中のオンラインシミュレーション言語ＳＩMBOL 設計に当り，離散系シミュレーション言語が持つ一般的特性について知る為に，各種シミュレーション言語の比較調査を行った。GPSS，SIMSCRIPT 1.5，SIMSCRIPTⅡ，SIMULA,OPS－4などを主として取り上げ，特にインプリメンターの側に立って調査を行った。

この報告書は，これら調査の一端を紹介したものである。

現在，オンラインシミュレーション言語の代表として挙げられるOPS－4 が，日本でまだあまり紹介されていない事，プロセスタイプの言語として興味あるＳIMULA についても詳しい内容が紹介されていない点などを考え，これら言語については特に詳しく解説しておいた。

しかし，ＳIMBOL の具体的な仕様に関しては次の機会に報告する事とし，ここでは序としてその特徴を述べるにとどめて置く。

ＳIMBOL（Simulation Model Builder for On-Line Usage）はその名の通りオンラインで，インタラクティブにシミュレーション・モデルの作成，テスティング，デバッキングが出来る様に設計された，シミュレーション言語である。離散系モデルを主な対象とし，どちらかと言えばＳIMULA と同様，プロセスタイプの言語体系をしている。ＳIMBOLは従来のバッチ処理用シミュレーション言語と比べて，モデルランニング時にモデルとのインタラクションが自由に行えたり，数回に渡るモデルランの総合的統計データの処理を行える様に考慮してある。又，モデルビルディングに関しても任意時点でモデルの修正が可能で，その融通性を増している。特にタイムシエアリングのもとで会話型でシミュレーションを行なうことを目標とし，端末にＣＲＴディスプレイを使用する。

第1章　オンラインシミュレータ

シミュレーション言語に要求される機能はその目的によってやや異なった面もあるが，一般には次のような項目があげられる。

(1)　モデル作成をすること。

(2)　モデルの正当性を確かめること。

(3)　モデルを実行すること。

(4)　統計データーを集めること。

(5)　そのデータを解析すること。

(6)　デバッキング

現在使われているバッチ処理用シミュレータ，あるいはシミュレーション言語では，他はともかく，(3)と(6)の点で不満な所が多い。特に(3)については，モデルの実行中，ユーザーがその実行にほとんど関与できないという欠点がある。シミュレーションの実行中に，その状態を即座に把握し臨機応変に適切な指示を与えることができれば，非常にシミュレーションの効率があがる。現在のバッチ処理用シミュレータでは，この点が不充分である。また，デバッキングに関しても，バッチ用のものでは，トレース機能の自由な使い分けができないので，無駄な情報まで出てしまうことになるが，インタラクティブにこのような操作ができるならば，適切な要領のよいデバッキングが可能になるであろう。バッチ用のコンパイラ形式の言語では，エラーが見つかった場合一部の修正で済むところでも全体を再コンパイルしなければならないのが普通である。

統計データを収集したり，それを解析することも，現在のバッチ用シミュレータの場合，予めモデルの中に処理用のステートメントを挿入しておかなければならないが，特にこのようなデータ収集については，モデルを実行してみて，初めて，取るべきデータの種類がはっきりする場合が多い。またデータを解析する場合にも，数回にわたるモデルの実行により得られたデータを解析する手段が用意されていない。

モデルを作成する場合，本来シミュレーションの性格から言って，一回で最適のモデルを完成することは概して不可能であるから，どうしても，部分部分を確かめながらモデルを改良して行く機能が必要である。

以上のように，現在のバッチ用シミュレータでは，まだ不充分な点が多く残されており，この

ような問題を解決するために，処理方法をオンライン化し，人間とコンピュータが，よりインタラクティブにシミュレーションをおこなう必要性と期待が高まっているのである。

第2章　オンラインシミュレータの現状

ＭＩＴではオンラインシミュレータの試みとして１９６４年来ＯＰＳ－１に始まるＯＰＳシリーズが開発されて来た。このうちＯＰＳ－３（これについては後の章で特徴を述べてある。）を拡張したＯＰＳ－４は従来のバッチ処理シミュレーション言語に比べ，遙かに機能的にも思想的にも進んだ言語として注目に値する言語である。

ＯＰＳ－４に関する詳細は後述するが，ここではオンライン使用と云う面から既に発表された他のシステムとあわせて，その特長を述べることにする。ディジタルシミュレータ（Digital Simulator）の分野で，現在迄に発表されているオンラインシミュレータは，そのほとんどが従来のバッチ処理言語にグラフィックディスプレイ（Graphic display）の使用を含ませた形のものである。ＧＰＳＳにグラフィックディスプレイを結合したシステムが，その主流を占めている。初期のものはマンマシンシステム（Man-Machine System）と呼ぶにはふさわしくない程度のもので，シミュレーション結果をディスプレイ装置上に表示させるという，出力用の使用が主であった。この様な目的を持って設計された言語としてRAND Corp.のＧＡＰＳＳ（Graphical Analysis Procedures for System Simulation）を挙げる事が出来る。ＧＡＰＳＳには，２つのフエーズがあり，第１のフエーズでは，シミュレーションの実行と，その結果の周期的なディスクストレージ（Disk Storage）上への蓄積とを行ない，第２のフエーズで蓄積結果をグラフィカルに解析すると云う方法が取られている。ユーザは必要に応じ Bar chart や gant chart を得る事も出来る。又グラフィック オン ライン モデリング（Graphic On-line Modeling）を志向するものとしてRAND の ＧＲＡＩＬ（GRAphic Input language）がある。

フローダイアグラム（flow-diagram）的なモデルビルディング（model Building）を特徴としファイルハンドリング（file handling），パターンコグニション（pattern cognition）に工夫がこらされている。ここでは主として，入力媒体としてグラフィックディスプレイを使用する事に力点がおかれている。

United Aircraft 社のＧＰＳＳ／３６０－ＮＯＲＤＥＮは，ＧＰＳＳ／３６０を一部会話形に修正したもので，マルチプログラミング（multi-programming）の下での操作をも可能にしている。ライトペン（light pen）やファンクションキー（function

key）などを持つディスプレイ装置はオフラインのプリンタをバックアップするために用いられる。このシステムではモデル作成の段階でもディスプレイが使用され，モデルの修正や再構成も従来のバッチ用のＧＰＳＳよりも簡単に行なえるようになっている。しかしディスプレイはモデルの窓口であるべきで，動的なシミュレーション結果を見る事ができ，またモデルやシミュレーション過程を試行錯誤的に変更することが可能でなければならないという観点からするならば，まだ不充分でありそれにはＧＰＳＳ構造の基本的な変更が必要である。もう一歩進んだ段階ではモデル構造をトポロジカル（topological）乃至は，階層的に記述し，実時間でのモデルの正当性のチェックを提供しようとする試みがある。この目的のために設計された言語としてproject MACのＳIMPLE（SIMulation programming language）が挙げられる。バッチ処理用シミュレーション言語の最大の欠点である，モデルとのインタラクトの不備をディスプレイの使用で補ない，ＧＰＳＳのTransaction-oriented なアプローチ及びSimula 流のActivity orientedなアプローチをも導入し得る様に拡張されている。これはＴＳＳの下で操作され，経済的であり，リアルタイムクロック（Real Time clock）との同期が取れる。またexogeneous eventはリアルタイムで受け付けられ，ディスプレイ装置の視覚機能が広範に利用出来て，数値及び時系列データの表示も可能となっている。これはＯＰＳ－４設計者Ｍ．Ｍ．Jonesの構想によるもので，ＯＰＳ－４の縮少版とも云えるものである。どの様な形でマンマシーン シミュレーションシステム(man-machine Simulation System）を展開させていくかについて，確かなフィロソフィは未だ存在しない。マン マシーン システムに於ける人間の位置づけと云う困難な問題は，これからの研究に俟たなければならないが，その重要性は現在に於ても十分に認識され，除々にではあるが，解決への道を歩み出しているのである。

第3章　バッチ用シミュレータのオンライン化

シミュレーション言語をオンライン化するに当っては単に表現上の問題だけでなく他にも種々の問題が起ってくる。

どの様な言語がオンライン用に適しているかを調べる為にも，まず既存のバッチ処理用シミュレータをそのままオンライン化するとどの様な問題があるのか明らかにしておくべきであろう。

以下にGPSS，SIMSCRIPT，SIMULA について，これらの言語のオンライン化の難易について述べる。

3.1　SIMSCRIPT

SIMSCRIPTは最も広く用いられているシミュレーション言語の1つであるが，可成りの修正を加えない限りオンライン言語として不適当である。それは何故か？

1. データの仕様と初期設定に複雑な固定欄形式が用いられている。それはもっと融通性のあるもの（例えば，SIMSCRIPTⅡのために考え出された日常の英語に近いステートメント）に代えられなければならない。
2. SIMSCRIPTには全体的にデバッギング機能が欠けている。これはオフライン・ユーザーにとってさえハンディキャップになっている。
3. ユーザーはマスター・スケジューリング・システム（イベント・リストと呼ばれている）を簡単に参照したり修正したりできない。
4. ユーザーは，プログラムをコンパイルし直したりローディングし直したりしない限り，モデルのストラクチャを変更できない。
5. イベントの条件付き実行を指定する方法がない。また，前にスケジュールしたイベントと関連づけてイベントをスケジュールすることができない。
6. モデルに関する統計を収集したり作ったりする機能が極めて限定されている。
7. SIMSCRIPTはアクティビティーよりもむしろイベント中心の言語であるため，大量のローカル・データーを1つのイベントから他のイベントへパラメーターとして受け渡す必要がしばしば生ずる。

SIMSCRIPTの利点は，セット処理機能が非常に強力である点にある。またFORTRANがSIMSCRIPTのサブセットになっているということも重要な特長である。

3.2 GPSS

GPSSも大変ポピュラーなシミュレーション言語である。オンライン利用の為に改訂されたGPSSII版は，1964年からMITタイムシエアリング・システム（CTSS）で利用されている。これを利用している人は極く限られているが，GPSS言語のこの拡張版がオンライン利用に適していることを認めているユーザーはもっと多いといわれている。その理由としては，次のような点があげられる。

1. デバッギング機能が優れている。
2. イベントをスケジューリングするために用いられるイベント・チェーンを何時でも調べることができる。
3. 実行の最中であっても，さうでなくても，モデルのストラクチャを容易に修正することができる。
4. ブロック，キュー（queue），ファシリティ，ストーレッジ或いはテーブルに関して自動的に統計を収集できるし，その統計やセイブ・バリューを何時でもプリントできる。
5. ユーザーは，実行の最中にコンソールからパラメータやセイブ・バリューをインプットすることができる。
6. 数値アウトプットを識別したり，モデル内のコントロールの流れを示したりするために，コメントをプリントすることができる。

しかしながら，GPSSには重大な限界が幾つかあり，そのために汎用シミュレーション言語として利用できないのである。

1. GPSSは制約の多い言語である。一般の算術式が扱えず，演算は整数型に限られている。エンティティのタイプが極く限定されていて，各タイプの数も決まっている。
2. GPSSは閉じた言語である　即ち，他の言語で書かれたサブルーチンとコミュニケーションできない。（HELPブロックは十分なメカニズムをもつものとは考えられない，というのは，それを利用するためにはFAPとGPSSシステムの内部構造とをユーザーが知っていなければならないからである）。
3. 利用できるコアのスペースが限られているため，ファシリティやストーレッジやキューやセイブ・バリュー等のようなエンティティの数が限定されているから，大規模なモデル

にGPSSを用いることができない。

4. ラベル・エンティティにニーモニックを用いることができない。ラベルは全てシンボルではなく番号を用いる。
5. データの仕様と初期設定が極めて制約されている。
6. ユーザーはコンソールからイベント・チェーンを訂正することができない。

3.3 SIMULA

SIMULA は現在利用されているシミュレーション言語のうちで最上のものである。それはALGOL をサブセットとして含み，イベントやアクティビティ（プロセスと呼ばれる）をスケジューリングする柔軟な手段をもっている。ユーザーはエンティティのタイプや数がどんなであろうと定義することができる。デバッギングやトレイシングの機能（これは最初の版にはなかったが，極く最近指定された）も優れているし，ランニング・タイムの点でも優秀である。しかし，現在の版でもオンライン利用に適しているとは言い難い。

1. SIMULA は実行する前にモデルのコンパイルを必要とするから，どんなにコンパイラが早くても，モデルを変更するには時間がかかる。
2. SIMULA では，ユーザーが条件付きでプロセスをアクティブにすることは許されない。
3. 基本言語としてALGOL を採用しているため，SIMULA は COBOL や FORTRAN システムと同じ間違いを犯しやすい。SIMULA によるモデルは1つの大きなプログラムから成り立っていて，どこかで変化が生じた時はその全体をコンパイルしなおさなければならない。
4. 現在のところ，統計収集のためにSIMULA が備えている補助手段は自動ヒストグラム，プロッティング・ルーチンだけである。
5. SIMULA には，シミュレーションの最中にモデルの状態をセイブしたり，ストアし直したりするための備えがない。

第4章　オンラインシミュレータ，その実例

前にも述べた通り，ＭＩＴに於けるプロジェクトMAC のもとで行われた，オンラインシミュレータに関する試みのうち，ＯＰＳ シリーズは特に貴重なものである。ＯＰＳ－4 が完全にインプリメントされていない現在ＯＰＳ－３ がこのシリーズ中，最新のものであるが，制作者みづから認めている様に，実用システムとは言いきれずむしろ実験用システムというべきであろう。以下にその概略を記しておく。

4.1 ＯＰＳ－３

上述の３つの言語と比べると，ＯＰＳ－３ によるシミュレーション・システムは特にオンライン利用のために設計されたものと言えよう。次に述べるＯＰＳ－３ の特徴は重要である。

1. ＯＰＳ －３では，データ・ストラクチャの仕様とデータの初期設定とが動的に行なわれ，また指定も容易である。
2. モデルの初期設定のし直し（部分的であれ全体的であれ）が簡単であり，しかも完全にユーザーのコントロールに委ねられている。
3. モデルのストラクチャを何時でも容易に修正できるし，プログラムをコンパイルしたり，ローディングし直したりする必要がない。
4. デバッギングやトレイシングの機能が包括的で融通性に富み，利用し易い。
5. ユーザーはスケジューリングのメカニズム――エイジェンダ（Ａｇｅｎｄａ）と呼ばれている――を好きな時に調べたり修正したりすることができる。
6. アレイ・オペレーションを有する普通の汎用言語を利用することができる（但し，コントロール・ステートメントは限定されている）。
7. 他の言語で書かれたサブルーチンとのコミュニケーションが簡単であり，ユーザーの好みに合うようにＯＰＳ－３ の基本的特徴を変えることも容易にできる。

ＧＰＳＳ と同様，ＯＰＳ--３ も主としてインタープリティブな言語であり，従ってランニング・タイムも非常に長い。現在のＯＰＳ－３ の欠陥は統計収集ルーチンがないという点にある。即ち，データ・エンティティの種類が限られているし，プログラムとデータとに利用

できるコアのスペースも極く限られている。また，幾つかのステートメントに用いられるシンタックスも，時として扱いにくく調和のとれていないことがある。

4.2 OPS－4

このシステムはMALCOLM M, JONES らによりMITに於けるプロジェクトMACの一環として計画されたオンラインシミュレータである。

OPS－4 はそれまで，過去数年にわたるOPS－3 の経験をもとに計画されたもので，CTSS のもとで稼動するのに対して，これはMULTICSのもとで稼動する事に成っている。

ただこの計画は１９６７年に発表されてはいるが，実際にシステムがインプリメントされたか否かは解らない。しかしながら離散系モデルを主にしたオンラインシミュレータの試みが，他にあまり例がない現在，特にOPS－3 の実績をもとに計画された点,非常に貴重なものと言えよう。以下，先ずこのシステムの概要を紹介する。

OPS－4 の特色は柔軟性と変更の容易さとにあり，またその目標は，モデルを作る人がモデルを構成しながらそれと絶えずインターアクトできるようにすることにある。

OPS－4 によって，ユーザーはインクレメンタルにモデルを作ったり，テストしたりすることができるようになる。モデルを個別的に構成できるように，特殊な機能も幾つか備えられている。モデルのうちの，よりフォーマルに構成されている部分は，通常のプログラムとして書くことができるし，インタープリティブに実行することができる。これらのプログラムは，くり返し修正することができるし，コンパイルし直さなくとも実行することができる。またモデルのうちの完全な形で記述された部分は，コンパイルされたプログラムとなり，最も効率的にランされる。シミュレーション・モデルにおいては，これら３つのタイプのプロシージャをどのように組み合せることもできる。

OPS－4 のワールド・ビュー（world view）はどちらかと言えばプロセスタイプである。すなわちプロセスはSIMULA の場合と同様にアクティビティによって定義される。

これらプロセスの発生や実行をコントロールするスケジューリングステートメントには，直接アクティビティを指定して行うもの（イクスプリシットなもの）と指定しないで行うもの（インプリシットなもの）とがあり，それらの選択が自由に出来るので，アクティビティは条件付きでも無条件でも実行することができる。

また既にエイジェンダ[1]（Agenda）に登録されているものと関係づけてエイジェンダに置くこともできる。アクティビティをスケジュールし直したり，キャンセルしたり，インタラプトしたり，再開始したりするためのステートメントもOPS－4 に用意されている。ユーザーは，スケジュール乃至インタラプトされた全てのアクティビティのリストを含むエイジェンダを，任意の時に見ることができるし，エイジェンダ・スキャン・オペレーションをモニタリングすることもできる。或るアクティビティの種々の状態が定義され，その状態を変更させ得るシークェンシング・ステートメントが記述される。

多種多様なトレーシング・オプションが用意されて，それによってユーザーは，総括的な情報やコントロールの全体的な流れの情報を受け取ったり，実行中あらゆるステートメントの完全なトレースを得たり，OPS－4 プログラムで参照される全ての変数の値を見たりすることができるようになる。個々のプロシージャやプロシージャのコンパイルされていない部分や全シミュレーション・モデルを，終了条件を指定して，選択した或る開始点から実行できるようなコントロール機能も，OPS－4 に用意されている。

コンパイルされたステートメントもしくはインタープリティブに実行されたステートメントを実行する特殊なインタープリタによって，これら柔軟且つ包括的なトレース機能及びコントロール機能が効果を発揮する。特殊な時間属性は，通常，変数定義の変更やモデル・ストラクチャの変更やトレース・オプションの設定の変更等をシステムに警告する。警告が与えられると，直ちに広汎なチェッキングが行なわれて，修正が成される。

OPS－4 は，自動統計処理機能は備えていないけれども，システムに関連した或る重要なデータをユーザーに提供することはできる。更に，それは多数の特殊なステートメントや関数を備えていて，モデルの統計収集・処理が簡単に行なえるようになっている。また，モデルのシミュレーション時間全体に渡る動きや総括的な統計をダイナミックにとることができる。

以上がOPS－4 の概略であるが，その特徴を要約してまとめると次のようになる。

1. PL/1 言語のサブセットはOPS－4 の基本言語になっており，演算処理機能やデータ処理機能も備えている。
2. OPS－4 のワールド・ビューは細かく限定されていない。即ち，モデルが材料や流れ中心のものであっても，マシンやエンティティ中心のものであっても，或いはこの両視点

注. 1）エイジェンダは一種のスケジュールテーブルでこれについては後の章で詳しく述べてある。

を結びつけたものであっても，ユーザーはそれを無理なくOPS－4 で表現することができる。また，アクティビティティ中心のモデル記述もイベント中心のモデル記述も共に利用できるのである。

3. OPS－4 は，ユーザーがインクリメンタルにモデルをつくったり，全部が完了する前に一部のモデルを調べたりできるように設計されている。

4. OPS－4 は，特殊分布に従って乱数を発生させるステートメントを含んでいる。

5. PL/1 の普通のデータ・タイプのほかに，セットやキューやテーブルと呼ばれている特殊なデータ・タイプも利用できる。データ・タイプの数やサイズには何ら制限がない。

6. データ・ベースを再構成することが容易であり，デバッギング・モードでランしているプロシージャをコンパイルし直す必要がない。

7. 簡単なステートメントを実行することによって，任意の所で任意の時間に，モデルの状態をセイブすることができる。

8. 前にセイブした状態からモデルの実行をやり直すことも同様に簡単である。従って，前にセイブした時点に戻してシミュレーションの実行をし直すことも容易である。

9. シミュレーション中のどの時点でも，シミュレーション・システムのデータ・ベースとユーザーのデータ・ベースとを（部分的にであれ，全体的にであれ）容易に初期設定し直せるし，システム・タイムのリセットも簡単であるから，シミュレーションを再スタートさせることもできるし，一連のシミュレーション・ランも容易に実行することができる。

10. ユーザーは，シミュレーションの最中にイベントが実行される精確な順番を指定してやることができる。

11. シミュレーション・システムの重要な部分は全てユーザーに解かるようになっている。ユーザーは，シミュレーションの全エレメントを，コンソールから直接参照できるし，モディファイすることもできる。

12. デバッギングやトレイシングの機能も拡張して利用できるし，またその利用が容易である。

13. コンパイルし直さなくても，モデルのロジカル・ストラクチャを容易にモディファイできるし，新たにモディファイしたものを直ちにランさせることができる。

14. シミュレーション・ランの出発点と停止点或いはその期間を指定する柔軟な手段が備わっている。

15. モデルの個々の成分を（たとえ，それらがもっと大きなモジュールに組み込まれていて

も）それぞれ別々にテストすることができる。

16. ユーザーは実行中のどの時点でもモデルへ割込むことができるし，実行順を変えることも，変数の値を調べたり変えたりすることもでき，更には割込みのあった時点から直ちにシミュレーションを続けることもできる。

17. 時間の逆行のような異常な事態が生じた場合，ユーザーはこれを直ちに知ることができるしシミュレーションが中断されることもない。

18. 通常，プロシージャに必要な初期データ・インプットのストラクチャとモードだけが，ユーザーによって宣言される。計算の結果出てくる大部分のデータ・オブジェクトのストラクチャとモードは，計算のルールによって示される。

19. モデルのランニング最中にエラーが発見される場合，詳細なオンライン診断情報がユーザーに与えられる。

20. モデルのデバッグされた部分はコンパイルされたプログラムとなるから，それをフル・スピードでランすることができる。

プログラムのまだチェックされていない部分が，必要とする時だけインタープリティブな手法が用いられる。

第５章　スケジューリングメカニズムとその周辺

５１　スケジューリングメカニズム － シンプルなモデル

シミュレータに於けるスケジューリングメカニズムはタイミングルーチンとスケジュールテーブルの２つから成っている。

タイミングルーチンがシミュレーション・クロックに従ってサブ・プログラムの実行順序をコントロールする為にスケジュールテーブルを参照しながらこれを行う。

特にシミュレーションプログラムでは１つのサブプログラムを異った時点で繰り返し実行したり，時には同時に同一プログラムを実行したりする事さえ起り得る。この為に「プログラムそのもの」と「プログラムの実行」は，ハッキリと区別しておかねばならない。

丁度，これはプログラムとタスクの関係に似ている。ＳＩＭＵＬＡ ではこれらの関係をアクティビティ宣言とプロセスとに区別している。ＳＩＭＳＣＲＩＰＴでも，イベントルーチンとイベント，に区別されている。

この様なわけでタイミングルーチンが行うプログラム実行順序の管理は，言わばタスクの管理を行う事である。

タスクの管理をする為にタスクにはコントロールブロックが対応している。

タスクをいつ実行するかはシミュレーションプログラムの中でイクスプリシット（ｅｘｐｌｉｃｉｔ）に，或はインプリシット（ｉｍｐｌｉｃｉｔ）に指定されているが，しかし要求があったときかならずしも即座にタスクを発生させるとはかぎらない。これらタスクの実行指定があるとコントロールブロックが作成され，スケジュールテーブルに登録される。

このコントロールブロックには実行時間やタスクと対応するプログラムなどに関する情報が書き込まれていてスケジュールテーブルには実行すべき時間の順序に従って適当な場所にこのブロックが挿入される。

スケジュールテーブルへのブロック挿入は実行時にダイナミックに発生するが，いつもテーブル中のブロックは，実行すべき時間の順序に従って並べられている。

この為に，タイミングルーチンはテーブルの先頭にあるブロックを取り出し，対応するプログラムを実行することで，タスクの管理をする事が出来る。図１にこの関係を示しておく。

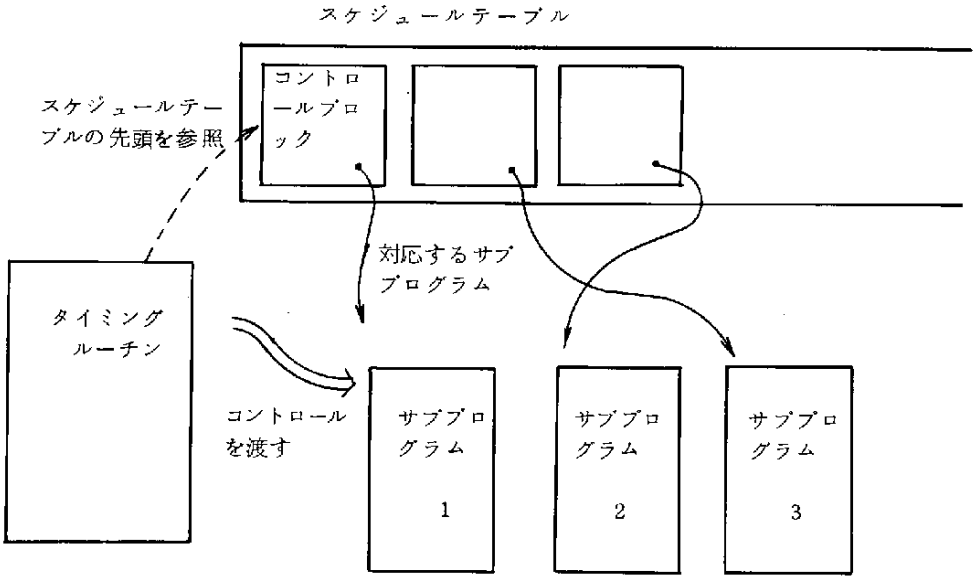


図1. スケジューリングメカニズムの構成

スケジュールメカニズムに関しては，もちろんシステム側で自動的に管理する機能があるにしろ，ユーザも，直接これをコントロール出来る様になっていなくてはならない。

一応，ここではスケジューリングメカニズムに対し，これをコントロールする事をスケジューリングと呼んでおこう。

以上で述べた，ごくシンプルなスケジューリングメカニズムではタスクにコントロールが渡る順序が単にスケジュールテーブルに登録されている順序に従って行われるので，スケジュール手段のほとんどがスケジュールテーブルのオペレーション手段に置き換えられてしまう。

しかし，さらに複雑なスケジューリングメカニズムを持ったシミュレータでは，単にスケジュールテーブルの順序に従って次に実行するタスクの選択が行われない事がある。これらの問題は後に紹介するＯＰＳ－４のスケジューリングメカニズムに見る事が出来る。

さて，ここでスケジューリングについて２つの区別をしておこう。

１つはイクスプリシットなスケジーリングで，ＳＩＭＳＣＲＩＰＴでは全てイクスプリシットなスケジーリング機能しか持っていない。

この言語でタスクに当るものはイベントと呼ばれ，これはイベントルーチンとして記述さ

れたプログラムの実行である。イクスプリシットなスケジューリングとはイベントを名ざしでスケジューリングする事で，これを行うには言語の中に個々のイベントを区別する手段が用意されていなくてはならない。

ＳＩＭＳＣＲＩＰＴの場合，イベントに対応するコントロールブロック（イベントノーティス）が，作成されると，そのアドレスを指定の変数に入れてくれるので，個々のイベントをこの変数によって指定する事が出来る。

ＳＩＭＵＬＡ やＯＰＳ－４ でも同様にプロセスを区別する手段が用意されていて，イクスプリシットなスケジューリングが可能に成っている。

このイクスプリシットなスケジューリング手段は後で説明するインプリシットなスケジューリングに比べて，他のタスクとの関係に於て，スケジューリングを行なわなくてはならないので，使う際に気を使う必要があるが，インプリシットなものにないパワーフルな機能を持っている。例えば，すでにスケジュールテーブルに登録されているコントロールブロックを削除する事が出来る。これは将来実行すべくタスクを登録したにもかかわらず，予定の変更があり，これを中止する様なケースに対応している。

次にインプリシットなスケジューリングについては，このスケジューリング手段しか持たないＧＰＳＳ がよい例となる。

ＧＰＳＳ はプロセス中心の言語と考える事も出来るが，直接プロセス単位にコントロールする手段は持っていない。つまり言語の中でプロセス単位で扱うブロック命令が存在していない。しかしタスクに相当するものはやはりプロセスと考える事が出来る。この言語でプロセスとは一連のブロック命令の実行で，それぞれに対応してトランザクションが発生される。トランザクションが，例のコントロールブロックの役割をはたしている。

インプリシットなスケジューリングでは，イクスプリシットな場合と比べて他のタスクに関係付けてスケジュールが行なわれない。

さらに詳しくこの違いを見るなら，イクスプリシットなスケジュールが，今実行されているタスク（プロセスとかイベント）が一般に他のタスクの実行計画を指示するのに比べ，インプリシットな場合には，今実行しているタスク自身の処置（例えば一定時刻後に実行する。）を指示している点にある。

イベントタイプの言語では，本来イベントの持つ性格として，その実行中にシミュレーションクロックを進めないから，一度イベントルーチンにコントロールが渡されると完全に終了してから別のプログラムにコントロールを渡すことになる。この為，このタイプの言語に

インプリシットなスケジューリング手段を持ち込む事が出来ない。事実ＳＩＭＳＣＲＩＰＴはこの様な手段は持っていない。

インプリシットなスケジューリングの特徴はイクスプリシットなものと比べて簡単に使える点にある。

以上，２つのスケジューリング手段については，ＳＩＭＵＬＡ やＯＰＳ－４ と言ったプロセスタイプの言語が両者の性格を兼ね備えているので，これらの例を見る事により明らかになるであろう。

5.2 プロセス

イベントを個々に記述して行くのがＳＩＭＳＣＲＩＰＴの立場なら，関連するイベントをまとめて，プロセスとして記述するのがＳＩＭＵＬＡ の立場である。プロセスによるモデル記述のアプローチは，プログラミングの容易さ，見易さ，などに於て特にイベントタイプの言語と比べて優れていると言えよう。

Ｋ.ＮｙｇａａｒｄらによるＳＩＭＵＬＡは 現在それほど使われていないにもかかわらず，常に引合に出される非常に注目すべき言語である。

この言語については付録としてＫ.Ｎｙｇａａｒｄらによるマニュアルを載せてあるので詳しい仕様についてはこれを参照されたい。

さて，このプロセスによるモデルの表現はＳＩＭＵＬＡ 以前にも，すでにＧＰＳＳ に於て見る事が出来る。ＧＰＳＳ はむしろトランザクションタイプの言語として分類される事が多いが，このトランザクションによる表現はプロセスと非常に多くの共通点を持っている事に気がつく。プロセスに於けるｒｅａｃｔｉｖａｔｉｏｎ ｐｏｉｎｔ の概念は，そっくりそのままトランザクションの位置に対応して考える事が出来るし，プロセスの持つアトリビュートはトランザクションのパラメータとして考える事が出来る。

もちろんＧＰＳＳ に於てはイクスプリシットなプロセスの概念はなく，プロセス単位にスケジュールする機能も持っていない事など相異点もある。しかし，ＧＰＳＳ の使い易さ，手軽さが，そのまま，プロセスタイプの言語に当てはまると言える様である。反面イベントタイプの言語と比べて小回りのきかないと言う欠点を持っている。

5.3 プロセスとアクティビティ

SIMULA に於てプロセスとアクティビティの概念は明確に区別されている。

SIMSCRIPTでイベントルーチンは，たとえ生起する時刻の異なるイベントであっても同じ様なパターンを持つイベントの代表として記述されている。

例えばガソリンスタンドに来た自動車が給油サービスを受けるモデルを考えた時，時刻 t_0 に来た自動車も，時刻 t_1 に来た自動車も，車の到着と言う記述でまとめてしまう様な方法である。プロセスとアクティビティの関係は丁度この様な関係と同じである。

前と同じモデルを考えるならば，時刻 t_0 にガソリンスタンドに来た車が s_0 時間だけサービスを受けたとか，時刻 t_1 に来たのが s_1 時間サービスを受けたと言うのが，1つ1つプロセスであり，ある時刻に来た車がサービスを受けた，とまとめて表わす時，これをアクティビティと言う。

この様にアクティビティは同じパターンを持ったプロセスが作る類の代表として定義される。

イベントルーチンがサブルーチンと同じ様なサブプログラムとして記述される様に，アクティビティも同様にプロシデュアーと同じ形式を持つ一種のサブプログラムとして記述される。

アクティビティにはその本来の性格から言って普通発生時刻が異なる，幾つかのイベントに関する記述を含んでいる。このイベント同士は，シミュレーションクロックによる時間経過を示すWAIT とかDELAY と言ったステートメントにより，関係付けられている。

5.4 プロセスの導入とプログラム構造

プロセスの代表パターンであるアクティビティ宣言の中で時間経過を示すステートメントが使える事は，このサブプログラムの構造にとって大きな意味を持っている。

タイミングルーチンによるプロセス実行のコントロールは，これらに対応するサブプログラム（アクティビティ）の実行順序をコントロールする。コントロールは各プロセスに含まれるイベント単位に行われるから，サブプログラムの実行は「時間経過を示すステートメントで区切られた単位」ごとに行なわれる。

従って1つのサブプログラムの実行が完全に終るのを待たずにタイミングルーチンを経由して別のプロセスの為にそのサブプログラムを始めから実行する様な事態が起り得る。

これはこのサブプログラムがリエントラントに使われることを意味している。

その為にアクティビティ宣言の中で使われる変数（ローカルな）は，発生するプロセス（このアクティビティから）ごとに別々のエリアが割当てられなくてはならない。図2にこの様

子を示す。

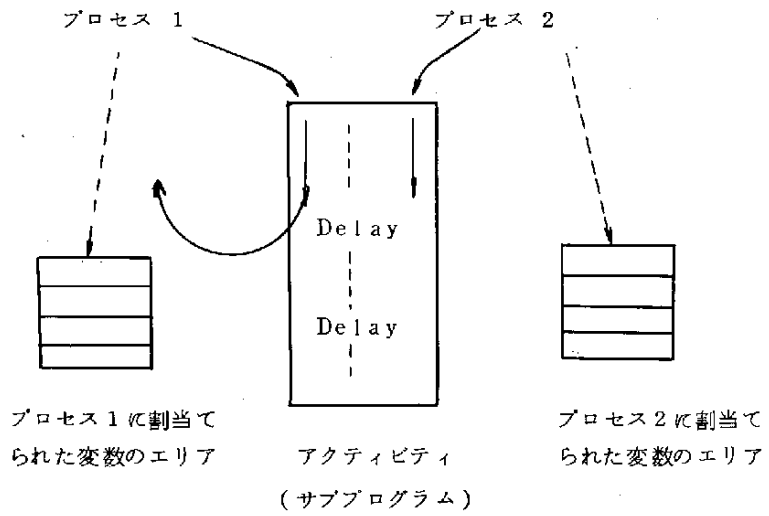


図2　プロセスとそこで使われている変数のエリア

ＳＩＭＳＣＲＩＰＴのようなイベント・タイプの言語の場合には，この場合のようにイベント・ルーチンで使われる変数に対して個別のエリアを割当てる必要がない。それは，タイミング・ルーチンによるコントロールが，イベントに対応するサブ・プログラムを完全に実行し終ってから，次に発生するイベントのサブプログラムを実行すると云うルールに従っているからである。従って，イベント・ルーチンはリエントラント構造をとる必要がない。以上のことから明らかなように，サブプログラムの中で時間経過を起すステートメントを使えるようにした場合には，プログラム構造に非常に大きな影響を与えることになる。

5.5　オペレーションルールとデータキャリヤー

プロセスの概念は関連したイベントをまとめて表すことだけでなく，もう一つの側面を持っている。それはプロセスの概念がオペレーションルールの記述とデータキャリヤーの性格を兼ね備えている点にある。

つまりＳＩＭＳＣＲＩＰＴに於けるイベントルーチンは，その時刻に起る変化がどの様なものかを記述しているが，そのイベントのアトリビュートを表わすデータがこれに付随して作成される様な事はない。(イベントノウティスを別に定義する必要がある。)一方ＳＩＭＵＬＡ

に於ては，プロセスのアトリビュートにあたるべきデータの定義がアクティビティ宣言の中に含まれている。これは前に説明したプログラム構造と密接に関係する問題でアクティビティ宣言を持つプログラムをreentrantに使わねばならない為，このサブプログラムの実行（つまりプロセスの発生）ごとにワークエリアが割付けられる。

このワークエリアはアクティビティ内でローカルに使われている変数に対して全て割当てられており，これはプロセスのアトリビュートとでも言うべきものである。

このデータブロック（work area）は個々のプロセスに対応して作成され，その内容は時々刻々に変るシステムのステータス情報を保持して，他のプロセスにその情報を受渡したりする，言わばインホメーション・キャリアー（Information carrier）或はデータ・キャリアー（Data carrier）としての性格を持っている。

一方アクティビティ宣言にはシステムステータスの変化過程，すなわち，個々のイベントによるステータス変化とか，その関係も記述されている。この様な記述は結局それぞれのプロセスに対応しており，プロセスが持つ「システム行動様式の記述」，或は「オペレーション・ルールの記述」といった性格を表わしている。

以上の様にプロセスはオペレーション・ルールの記述とデータ・キャリアーとしての性格と言う2つの要素を一体化した概念である。

ちなみに，SIMSCRIPTに於て，この2つの性格を見出すならば，オペレーション・ルールはイベントルーチンにより記述され，データ・キャリヤーの役目はテンポラリーエンティティが負っている。この様にイベントタイプの言語に於ては両者が完全に分離されている。

5.6 OPS－4のスケジューリングメカニズム

OPS－4はプロセスタイプの言語であり，システムの記述はプロセス中心に行われる。

OPS－4のスケジューリングメカニズムはエイジェンダと言われる一種のスケジュールテーブルをもとに構成されている。

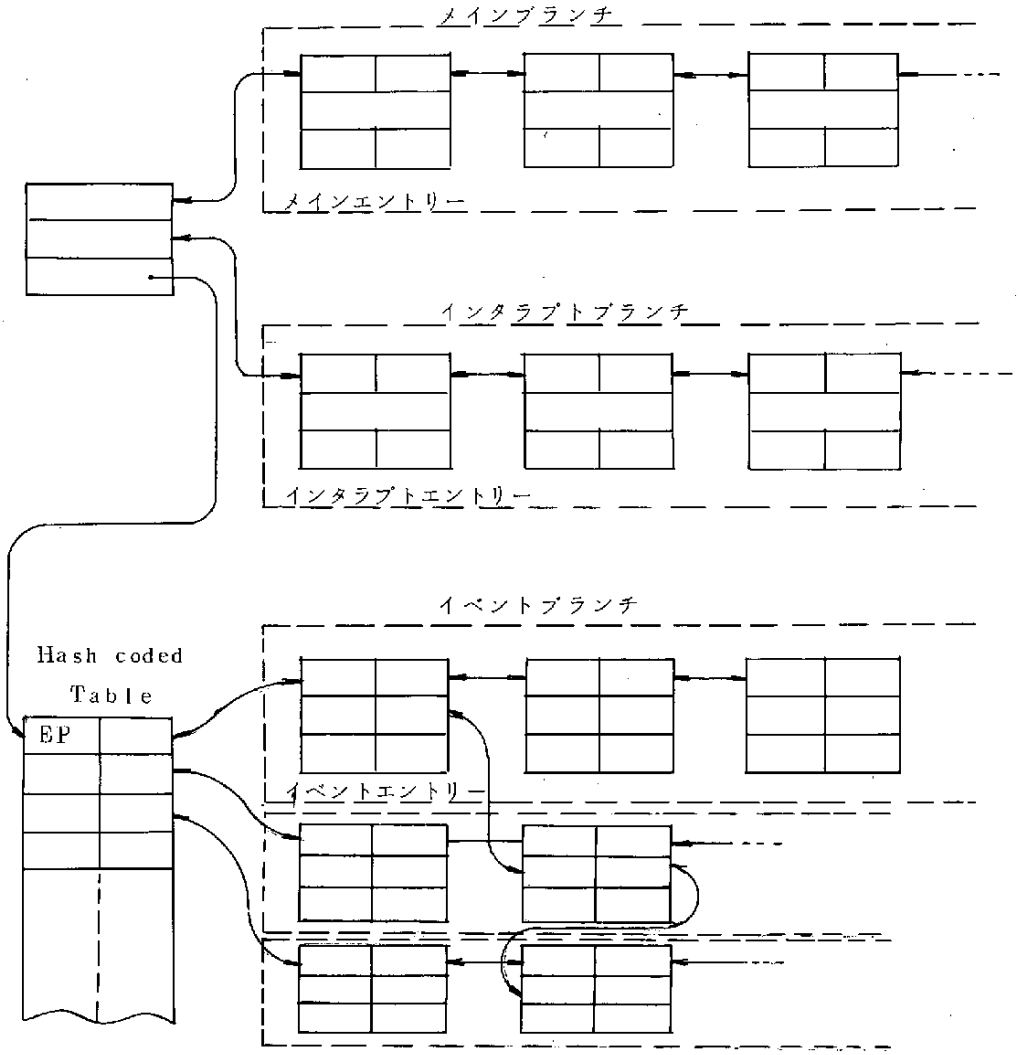
エイジェンダには各プロセスに対応するエントリーが登録されており，原則としてその順序がプロセス実行の順序を表わしている。ただOPS－4に於ては後にのべるアクティビティの条件付きスケジューリング機能を持っている為にこれらに関するエントリーがエイジェンダの先頭にある為，かならずしも実行順と一致しない事がある。

エイジェンダの詳しい構造については後で説明してあるが，スケジューリング手段につい

て説明するにはエイジェンダの概略を知っておく必要がある。

エイジェンダは図3に示すごとく大きく3つの部分に分ける事が出来る。

1. メインブランチ
2. インタラプトブランチ
3. イベントブランチ



EP： イベント名へのポインター

図3 エイジェンダの構造
（各エントリーの構造は図5を参照）

このうちメインブランチには条件なしでスケジュールされたプロセスに対応するエントリーが登録され，インタラプトブランチには割込みをかけられたプロセスのエントリーが登録される。

イベントブランチにはイベント発生の条件によりスケジュールされたプロセスのエントリーが登録される。

5.7 エイジェンダーのモディファイ

5.7.1 イクスプリシットなモディファイ

エイジェンダーに対して3種類の変更が可能である。

1. 新しいエントリーを付け加える事。
2. 既にエイジェンダーにあるエントリーを削除する事。
3. エイジェンダーに登録されているエントリーに対してモディファイする事。

エイジェンダーは特殊なセットと考えられるからセットに対する操作がエイジェンダーに対してもすべて可能である。即ち，エイジェンダーの先頭や最後にエントリーを挿入することや指定されたエントリーの直前や直後に挿入することも可能である。しかし，エイジェンダーに対しては，セットに対する事以外に特殊なステートメントが用意されており異なった扱いができるようになっている。モディファイのしかたは5通りあり以下それらのステートメントの機能について説明しよう。

5.7.2 新しいエントリーの付加

- スケジュール・ステートメント

a．無条件スケジュール

このステートメントにより新しいエントリーが作成される。もし1つのアクティビティから他のアクティビティの実行を指定したいときこのステートメントを使えば良い。これは丁度SIMSCRIPTにおけるCreate ステートメントとCauseステートメントを合せた機能を持っている。スケジュール・ステートメントによりエイジェンダーに挿入されるエントリーの位置はエイジェンダーの先頭，最後あるいは既にエイジェンダーにあるエントリーの直前，直後を指定することができる。

例えば

```
Schedule New-User after Finish
```

とか

Shedule New-User named Sam after Finish

はどちらもNew-Userというアクティビティを Finish というアクティビティの直後に挿入することを指定している。ただ後の例ではNew-Userというアクティビティに対して Sam という名前をつけている点だけが異っている。Sam というポインター変数はダイナミックに作られた個々のアクティビティを識別するためのものである。エイジェンダーのユニークな特徴の1つとして時間属性をエントリーに対して持たせることができる。従って前述の位置だけに依存するスケジュールに加えて時間属性を持たせたスケジュールも可能である。

Schedule New-User named Sam at 1225

これにより，Sam として識別される新しいアクティビティは1225 あるいは，それより小さい時間属性を持つ，先にスケジュールされたアクティビティの後に登録されることになる。

b．条件付きスケジュール

変数による条件式を付けて特定な条件が満されたらアクティビティを実行させることもできる。

Schedule New-User when X＝B

これは条件X＝BがTrueになったらアクティビティNew-Userを実行させることを意味する。更に，エイジェンダー上の位置に依存しない条件付きスケジュールも可能である。それは条件としてイベントが生起したらアクティビティを実行させる様なスケジュール方式である。この場合には，後に述べる様にエイジェンダー上のメイン・ブランチにではなくイベント・ブランチに登録される。イベントが生起されることにより条件が満たされると，その時点で現在実行されているアクティビティの直後に実行されるものとしてメイン・ブランチに移される。

5.7.3 アクティビティの削除

- キャンセル・ステートメント

実行を計画されたアクティビティを削除した場合にはエイジェンダーからそのエントリーを取り除けば良い。そのためにキャンセル・ステートメントが用意されている。キャンセルされたアクティビティはエイジェンダー上のエントリーを失うのみでディフィニション・ブロック（Definition block）は残っているので他のアクティビティから参照される場合もある。

5.7.4　モディファイ

- リスケジュール・ステートメント

既にエイジェンダーに登録されたアクティビティの実行をモディファイすることができる。このリスケジュール・ステートメントは先に述べたスケジュール・ステートメントの機能はすべて持っている。従ってコンディショナルなスケジュールをアンコンディショナルなスケジュールに，また，その逆にも変えることができる。

- インタラプト・ステートメント

アクティビティが実行を続行できない様な条件（クィットボタンがおされたり，優先度の高いアクティビティからのすでに使用されている設備に対する使用要求がある時）に遭遇した時，現在実行されているアクティビティは割り込まれる。インタラプトは，割り込みがいつ解除されて実行し得る様になるか指定できないのでリスケジュールとは異なったモディファイのしかたである。更に，インタラプトの場合はメインブランチからエントリーが除去されるのではあるがインタラプト・ブランチにエントリーが設けられるので割込み解除時点で必要に応じてメイン・ブランチへ登録しなおし実行させることができる。その為にキャンセルとも異なるモディファイのしかたである。この場合インタラプト・ブランチからメインブランチへエントリーの移動がなければキャンセル・ステートメントと実効的に等価となる。

- リシューム・ステートメント

割り込みがリシューム・ステートメントにより解除されると，割り込まれた時点での時間は現在時刻に更新されメインブランチの適当な場所にエントリーが設けられる。その時点でインタラプト・ブランチにつながれていたエントリーは除去される。

スケジュール・ステートメントを除く４つのステートメントは先に登録されてあるアクティビティに対してのみ有効なステートメントであるため，もし指定したアクティビティがエイジェンダーに存在しない場合にはエラーを示すフラグがセットされるだけでモディファイはなされない。

5.7.5　インプリシットなモディファイ

先に述べた５つのモディファイ・ステートメントは特定な名前のアクティビティをイクスプリシットに指定してモディファイするものであった。ここではアクティビティが実行された時それ自身の状態をモディファイするステートメントについて述べよう。ここでインプリシットというのはアクティビティがいつ実行されるか既知ではないが実行

された時点でモディファイし得る様になるという意味である。

- ディレイ・ステートメント

  インプリシット・スケジュール・ステートメントで最も単純な型で記述されるのはディレイ・ステートメントである。それは現在実行されているアクティビティの遅延期間を記述する。これにより現在実行されているアクティビティは遅延期間後に再び実行されるものとしてエイジェンダーに登録されるので，アンコンディショナルなリスケジュール・ステートメントと同じ働きをする。

- ウェイト・ステートメント

  この場合は時間間隔を記述するのではなく特定な条件を記述することになる。条件が満されない間は割り込み状態におかれる。これは特定な条件が記述された，リスケジュール・ステートメントと同様な働きをする。

  その条件として一般的な変数による条件式，及びイベント名が記述できる。またディレイステートメントとウェイトステートメントを組合せた様なモディファイのしかたもできる。

5.7.6 セルフ・インタラプションとセルフ・キャンセレーション

アクティビティは自分自身を割り込み状態に置く事も出来る。アーギュメントなしのイラタラプト・ステートメントを実行したアクティビティは自分自身を割り込み状態に置く。この状態になったアクティビティは他のアクティビティから割り込みを解除してもらわないと動き出す事は出来ない。更に，自分自身をキャンセルする事も出来る。これらをセルフ・インタラプション（self interruption）およびセルフ・キャンセレーション（self cancellation）という。

Return，Exit や End ステートメントを実行するとエイジェンダ上からエントリーが除去される。コンティニューステートメントは，エイジェンダ上のエントリーに変更を加える事はない。

5.8 アクティビティのステート

アクティビティ（activity）はそれが存在している任意の時点で次の6つのステートを取る。

1. アクティブ（active）―― アクティビティはプロセッサで現在実行されている。
2. アンコンディショナリー・スケジュールド（unconditonaly scheduled）

アクティビティがアクティブになる為には時間の経過だけが必要である。

3. コンディショナリー・スケジュールド（conditionaly scheduled）――指定された条件がTrueになるとすぐにアクティビティがアクティブになる。

4. 将来におけるコンディショナリー・スケジュールド――指示された時間が経過し指定された条件がTrueになるとアクティビティがアクティブになる。

5. インタラプティド（interrupted）――アクティビティはRESUME ステートメントによって参照されるまでアクティブになることはできない。

6. インアクティブ（inactive）――アクティビティは作られずEnd ステートメントを実行することによってキャンセル乃至は終了する。しかしながら，それはまだローカル・データ・ベースを持っていて他のアクティビティによって参照される。

外部シークエンシング・ステートメントがエイジェンダーに既に登録されているアクティビティを直接アクティブにすることができないということは大変興味のある点である。これは，エイジェンダーが常にコントロールされていることを示す。

しかしながら

Reschedule Z top

Delay ϕ

等のステートメントはアクティビティZのダイレクトコールと等価である。

ダイレクトコールは，現在定義されている，リアクティベェイション・ポイント（reactivation point）をアクティベイトする事である。

5.8 エイジェンダーのスキャン

エイジェンダーのスキャン

次にアクティブにすべきアクティビティを選びそのアクティビティにコントロールを移す事により順次シミュレーションを進行させる事が出来る。アクティビティのコントロールの戻しに特別の指定がないかぎりスキャンはエイジェンダの先頭に戻される。

スキャンが先頭から開始されると最初のエントリーが調べられる。それは前に述べた３つのステート２・３あるいは４に対応するアクティビティである。それが無条件のエントリー――例えばそれを指定するアクティビティが無条件にスケジュールされている場合（ステートの２）――であるならば即座に，アクティブになる。それが条件付きエントリー――例えばそれを指定するアクティビティが条件付きでスケジュールされている場合（ステート

の２・３)――であるならばその条件がテストされる。もし条件がＴｒｕｅならばアクティビティはただちにアクティブになる。条件がＴｒｕｅでない場合はこのエントリーが素通りさせられて次のエントリーがテストされる。

エイジェンダー上に条件付きエントリーが存在しているため適当なアクティビティが見いだされる前にいくつかのエントリーがテストされることになる。全エイジェンダーがスキャンされ適当なアクティビティが見つからない場合には，ユーザーにエラーが知らされる，コンディショナルなアクティビティに関してエイジェンダ上のメインブランチにエントリーをもつものは，一般の変数による条件式でスケジュールされたアクティビティのみであり，イベントによる条件でスケジュールされたアクティビティは，エイジェンダー上の異った場所（イベント・ブランチ）に置かれておりエイジェンダー・スキャンの対象とはならない。

5.9 エイジェンダーのストラクチャー

全エイジェンダーはツリーストラクチャー（ｔｒｅｃ ｓｔｒｕｃｔｕｒｅ）をなしている。メインブランチについては，前に簡単にふれておいた。イベントによる条件で指定された各イベントは，イベント・ブランチにおかれる。例えばユーザーが

```
wait for Event A and B
```

と指定した場合このステートメントを実行したアクティビティを指すメインエントリーは，イベントＡ及びＢのブランチにつなぎ換えられる事になる。

ブランチＡ及びＢにおける２つのエントリーがチェーンされカウント・フィールドが２にセットされる。その場合それはイベントが両方共条件を満たさなければならないことを示している。イベントＡもしくはＢを条件として指定した場合エイジェンダー上のエントリーは上と同様であるが，カウント・フィールドは１にセットされる。その場合には２つのイベントのどちらか一方が，アクティビティをアクティブにすることになる。ユーザーが「and」か「or」かをイックスプリシットに指定しない時にはデフォールト（ｄｅｆａｕｌｔ）は，常に「ａｎｄ」とみなされる。ＰＬ／１のカウント・オプションも用いられることもある。

例えば

```
wait for Event A, B, C, D, E, (3)
```

はイベントＡ，Ｂ，Ｃ，Ｄ，Ｅ，のうちのどれか３つが発生した時アクティビティは続行されなければならないということを表わしている。これは各イベント・ブランチのエントリーのカウント・フィールドに於て名前を付けられたイベントの総数よりも，指定したカウントをス

トアすることによって行なわれる。

これら3つのどの場合においても1つのイベントが完了すると，それに対応するエントリーをイベント・ブランチから取り除き，イベント・ブランチ上の別のエントリーのカウントを1ずつ減らす。カウントが0になると残っているエントリーはすべて取り除かれて，指定されたアクティビティのメイン・エントリーがメインブランチに挿入されることになる。

Set- Event (event name)

というPL/1 ステートメントはイベントの完了を示すためのものである。イベントの完了が知らされるとイベントブランチ上にそれに対応するイベントエントリーが連鎖されているか否か調べられ，そのブランチや他のブランチ上のエントリーに必要なモディフィケーションを行なう。アクティビティに関するイベント・コンディションがすべて満たされると指定されたアクティビティがエイジェンダーのメイン・ブランチに入れられる。

インタラプトされたブランチはすべてインタラプト・ブランチにおかれる。アクティビティがインタラプトされるとそれはメインブランチから取り除かれインタラプション時間を含むエントリーとインタラプトされたアクティビティに対するエントリーのポインターとがインタラプト・ブランチにおかれる。

この様に，エイジェンダーはリストのリストである。最初のサブリストはメインブランチであり，2番目のサブリストはインタラプト・ブランチで，3番目はハッシュ・コウディット・テーブル(Hash coded table)のポインター(これは特定のイベント・ブランチの各々のポインターを含む)である。サブリストの数は3と決められているから，これらのサブリストのポインターは3エレメント・ベクトルにストアされる。最初のエレメントはメインブランチの始まりの ポインターを含む。2番目のエレメントはインタラプト・ブランチの始まりのポインターを含み3番目のエレメントはハッシュ・コウディッド・テーブルのベースのポインターを含む。ハッシュ・コウディッド・テーブルにおける各エントリーは1ワードで，2つのポインターを含んでいる。その1つは，イベントの名前に対するものであり，もう1つはイベント・ブランチに対するものである。このテーブルは長すぎたり短かすぎたりする場合にはハッシュしなおされる。

MULTICS によって与えられるEvent table や別のインタラプト・リストを用いるよりも，むしろこれらのイベント・ブランチやインタラプト・ブランチをエイジェンダーの1部として含める理由は，エイジェンダーを調べることによって，アクティビティのステートに関するあらゆる情報を，異なった場所をいちいち見る事なく得られる様にするため

である。

5.9.1　アクティビティ定義ブロックのストラクチャ

ＯＰＳ－４ システムでスケジュールされているアクティビティは，常にそれと関連する識別データを幾つか持っている。これは，アクティビティのディフィニション・ブロックと呼ばれる。以下ＡＤＢ（Ａｃｔｉｖｉｔｙ　Ｄｅｆｉｎｉｔｉｏｎ　Ｂｌｏｃｋと略記）

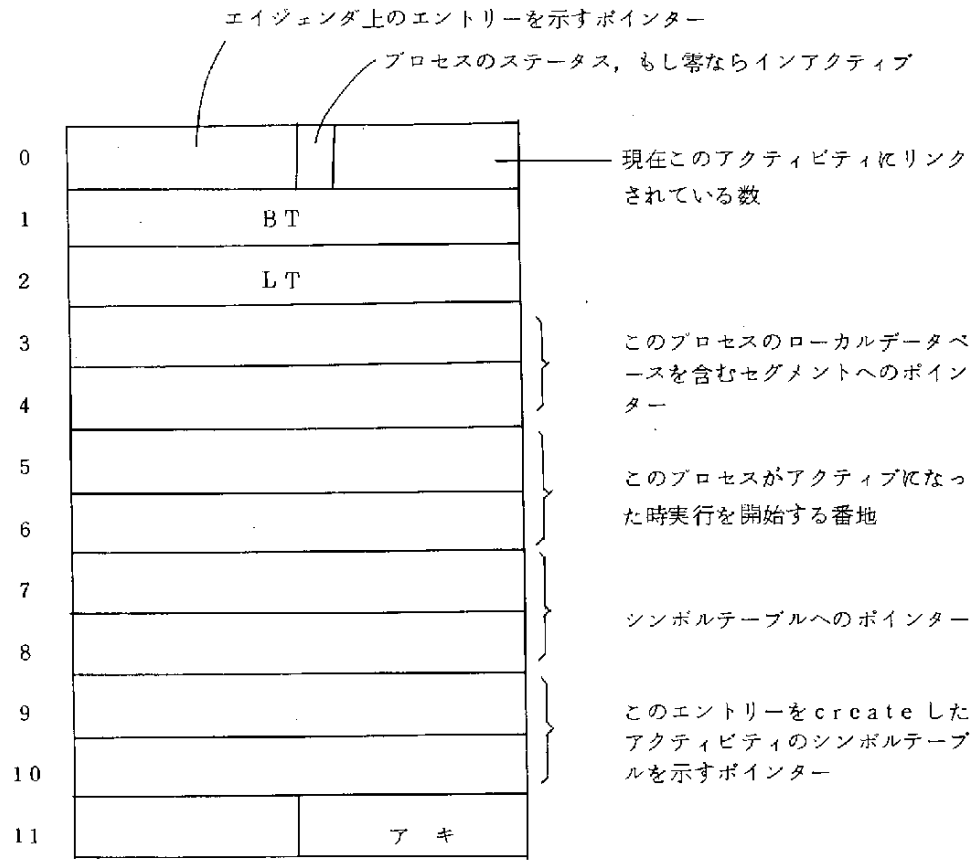
ＡＤＢは１２ ｗｏｒｄ から構成されアクティビティに関する種々の情報を含んでいる。ユーザーが記述したアクティビティはＡＤＢとして実体化される。アクティビティの実行はＡＤＢの参照を通して行なわれる訳でＡｇｅｎｄａ には，ＡＤＢを指すポインターを登録して置く事により，スキャンで実行が決まったアクティビティを混乱なく実行する事が出来る訳である。

ＡＤＢの構造を図４に示す。

5.9.2　各エントリーの構造

エイジェンダ上に置かれる３つのｂｒａｎｃｈ のエントリーは各々３ワードで構成されており，対称リストを構成している。

図５にこれらの構造を示しておく。

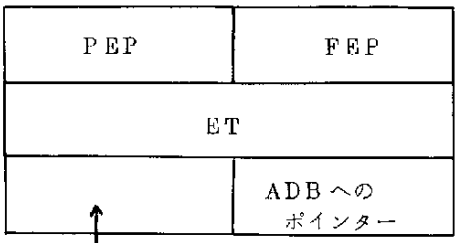


BT： プロセスが始めてアクティブになった時刻

LT： プロセスが最後にモディファイされた時刻

図4 ADB(Activity Definition Block)の構造

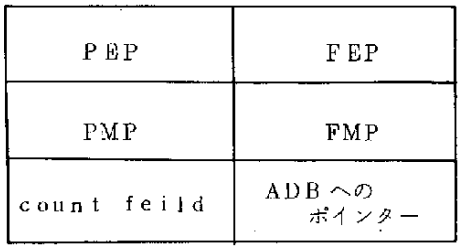
メインエントリー　（３words）



ET：Execution Time
実行時間

条件ナシでスケジュールされた場合　零
変数の条件による場合は変数を示すポインターが入る。

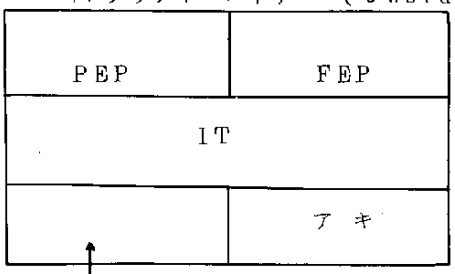
イベントエントリー　（３words）



PMP，FMP は同じWAIT ステートメントによって指定されたイベント同士をチェインする為に使われる。

cf＝0　：" or" condition
＞0　："and"　〃

インタラプトエントリー　（３words）



IT：割込まれた時刻

割込まれたプロセスのADBを示すポインター

PEP：　ブランチ内でのPrecceding エントリーを示すポインター

FEP：　〃　Following　〃

図５　各エントリーの構造

5.10 時刻の更新

エイジェンダ・スキャンのメカニズムは，シミュレーション時間を進めるのに用いられる。時間を進めるためのルールは以下の通りである：

1. 実行すべき無条件アクティビティが選ばれると，システム・タイムは，そのアクティビティのもつETに更新される。そのETが更新される前のシステム・タイムよりも小さいと，フラグをセットする。
2. 条件付きのアクティビティが選ばれると，そのETが現在のシステム・タイムより小さいかもしくは等しい時には，時間は更新されず，それ以外の時はETがシステム・タイムとして取られる。

ユーザーは

```
Set sys-time = expression
```

ステートメントを実行することにより，時間を更新することも可能である。この機能によって，アクティビティが互いに相対的な関係をもって計画されている場合，時間の更新が円滑に行なわれる。

5.11 エイジェンダへのコントロールの戻り

アクティビティが内部シークエンシング・ステートメント（Delay,WAIT等）を実行すると，コントロールは自動的にエイジェンダ・システムに返される。アクティビティが制御をエイジェンダに返す前に，スキャンを現在のエントリーから行なわせるか，それともエイジェンダの先頭から行なわせるかを指定することができる。

```
CONTINUE NEXT
```

ステートメントにより，エイジェンダ上の現在のエントリーからスキャンを行なわせることができる。何の指定もない場合は，ディフォールトとしてスキャンはエイジェンダの先頭から開始されるようにセットされる。スキャンにより実行すべきアクティビティが選ばれ，実行された後，GVC（ゼネラル・ヴァリアブル・コンディション）をテストして，条件に変更があったかどうかを調べることができる。もし多くのGVCがあると，スキャンが非常に遅くなるので，先に述べたCONTINUE NEXT ステートメントによってGVCのテストをスキップさせて次のエントリへスキャンを移動させることができる。

5.12 エイジェンダ・エントリーのモディファイ

先にユーザー側から見たエイジェンダのモディファイについて述べたが，ここではシステム・サイドに立ってもう一度述べることにする。

5.12.1 外部シークエンシング・ステートメント

- SCHEDULE によるモディファイ

SCHEDULE ステートメントは，常に新しいアクティビティに対するＡＤＢを作り出し，同時にエイジェンダ上のしかるべきブランチにエントリーを作り出す。この時，イベントが記述されていると，イベント名はハッシュ・テーブルと付き合わされ，必要ならば，テーブルに登録され，イベント・ブランチにエントリーが設けられる。しかしながら，SCHEDULE 以外 のモディファイ・ステートメントは既に登録済みのエントリーの位置を操作するのみで，新しくエントリーを追加したりＡＤＢを作ったりすることはない。

a．メイン・ブランチの先頭に最初にエントリーが設けられたとき。

その時点でのシミュレーション時間によりＥＴ（実行時間），ＢＴ（ＡＤＢが作成された時刻を示す），ＬＴ（最後にモディファイされた時刻を示す）をイニシャライズする。

b．ＢＥＦＯＲＥ，ＡＦＴＲＥ，ＩＮ ＰＬＡＣＥ ＯＦが指定されたとき。

参照したアクティビティのＥＴによりＥＴ，ＢＴ，ＬＴをイニシャライズする。

c．条件付きの場合。

現在のシミュレーション時間によりＥＴ，ＢＴ，ＬＴをイニシャライズする。

d．ＴＯＰ，ＢＯＴＴＯＭが指定されたとき。

現在のシミュレーション時間により，ＥＴ，ＢＴ，ＬＴがイニシャライズされる。

e．時間が指定されたとき。

指定された時間により，ＥＴ，ＢＴ，ＬＴがイニシャライズされる。

- ＲＥＳＣＨＥＤＵＬＥによるモディファイ

この場合は，エイジェンダ上のエントリーの位置のみをモディファイする。

a．メイン・ブランチ上で位置が動かされる場合。ＥＴが指定された時刻に更新され，このエントリーのＦＰ（フォーワード・ポインタ）及びＢＰ（バック・ポインタ）と変更前の前後のエントリーのポインタ（ＢＰとＦＰ）更に，新しく挿入

される位置の前後のエントリーのポインタ（ＢＰとＦＰ）が調整される。

b．他のブランチへモディファイされる場合。

その場合には，変更前の前後のエントリーのポインタ（ＢＰとＦＰ）が調整され，新しく別のブランチにエントリが移される。

- ＣＡＮＣＥＬ ステートメントによるモディファイ

この場合には，変更前の隣接する２つのエントリのポインタが調整されることにより，除去される。しかしながら，ＡＤＢは存在しているので，他のアクティビティがキャンセルされたアクティビティを参照することは可能である。

- ＩＮＴＥＲＲＵＰＴステートメントによるモディファイ

メイン・ブランチの隣接するエントリの各ポインタが調整され，インタラプト・ブランチの最後にエントリが挿入されることにより，メイン・ブランチに置かれていたエントリがインタラプト・ブランチに移される。

- ＲＥＳＵＭＥ ステートメントによるモディファイ

これは，インタラプト・ブランチに置かれたエントリを参照する点を除いては，ＲＥＳＣＨＥＤＵＬＥステートメントと同様の働きをする。そして，エントリが再びメイン・ブランチ或いはイベント・ブランチへ移されると，割り込まれていた時間が，現時刻からインタラプト・ブランチのＩＴ（割り込まれた時刻を示す）にストアされている時間を引いて計算され，ＥＴに加えられて，それが新しい実行時間となる。

5.12.2　内部シークエンシング・ステートメント

ＲＥＴＵＲＮ，ＥＮＤ，ＥＸＩＴ を除く全ての内部シークエンシング・ステートメントは，以下の３つのモディファイを行なう。

a．現時刻に変更する為にＬＴをモディファイする。

b，アクティビティのリアクティベイション・ポイントを再定義する。

c．アクティビティのステートを表わすモードをモディファイする。

- ＤＥＬＡＹ ステートメントによるモディファイ

これは，ＥＴもモディファイし，エイジェンダ上での位置を決めるのに用いられ，移動に伴なう４つのポインタを調整する。

- ＷＡＩＴ ステートメントによるモディファイ

これは，ＥＴを現時刻に更新し，コンディション・ポインタをセットする。新し

いエントリはメイン・ブランチ或いはイベント・ブランチへ移される。

- WAIT とDELAY ステートメントの組合せによるモディファイ

  この場合は，上記の1，2の場合を合わせた操作が行なわれる。

- INTERRUPTステートメントによるモディファイ

  この場合，外部シークエンシング・ステートメントの項で述べたINTERRUPTステートメントと同様の操作が行なわれる。コンディションポインタはゼロにリセットされる。

- CONTINUE ステートメントによるモディファイ

  この場合は，エントリの変更はない。

- RETURN, EXIT, END ステートメントによるモディファイ

  この場合は，隣接する2つのエントリの各ポインタを調整することにより，エイジェンダから除去される。

もしアクティビティが直接他のアクティビティをコールする場合は，エイジェンダ上のエントリはモディファイされないので，コントロールはエイジェンダに返されない。ここで述べた3つの時間属性ET，BT，LTは各々，システムで定義された関数を用いてその値を参照することはできるが，ユーザーが直接それをモディファイすることはできない。

第6章　データベースの扱い

シミュレーションモデルの実行時に於ける処理の大部分はリスト処理であると言って過言でない。それはタイミングルーチンによるイベント，或はプロセスなどのコントロールがスケジュールテーブルへの登録，変更，削除などによって行なわれる事を見ても明らかであろう。又，内部で行うリスト処理とは別に言語仕様として表に出されたリスト処理機能も重要である。本来，リスト処理は物の関係を表わすのに非常に便利な機能を持っているが，シミュレーションに於てもこれは不可欠のものである。

シミュレーション言語の持つリスト処理機能については，データ構造の問題と，その扱いに関して調べる必要があろう。

そこで，まづ，シミュレーション言語が行うリスト処理と似かよった機能を持つリスト処理用言語L^6を紹介する事から始めよう。

6.1　リスト処理言語L^6

L^6（Bell Telephone Laboratories Low-Level Linked List Language）は名の通りBell Telephone Laboratorieで開発された言語で，LISP，SNOBOLなどのストリング処理を得意とする言語と異りlinked blockの処理を主眼に設計された言語である。以下にその機能の一端をかいつまんで説明しておく。

6.1.1　データ単位

ブロック：　L^6で扱うデータ単位は1，2，4，8，……128，つまり2^n（n＝0，1，2，……7）ワードより成る連続したメモリーブロックである。これをブロックと言う。

フィールド：　図6に示すようにブロックは幾つかの区画に区切ってそこに名前を付

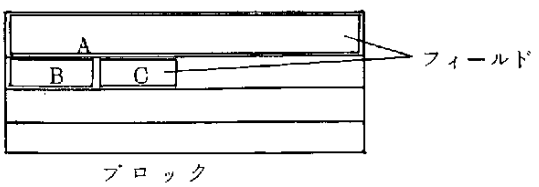


図6　L^6に於けるブロックとフィールド

ける事が出来る。

これをフィールドと呼び演算をしたり，内容を参照したりする最少単位となる。

6.1.2　フィールドの定義

ブロックに於けるフィールドの定義及び再定義は任意の時点で行う事が出来る。

フィールドの定義はフィールドの名前とブロックに於けるそのワード位置，先頭ビット位置，最終ビット位置を指定する事により行われる。

フィールドの定義は各ブロックについて共通であり，異ったブロックであっても同じ位置にあるフィールドは同じ名前で与えられる。フィールドはブロックをそれぞれ別個の区画に区分するものではなく，フィールド同志がオーバラップする事も一向にかまわない。もちろんブロックサイズが短か過ぎて，フィールド位置がそれより後になってしまう場合には，そのブロックにこの名前のフィールドは存在しない事になる。

6.1.3　ブロックのGet，Free

図7に示すように指定されたサイズのブロックをFree storage からGetすることとそのブロックをFree storage に返す事が出来る。

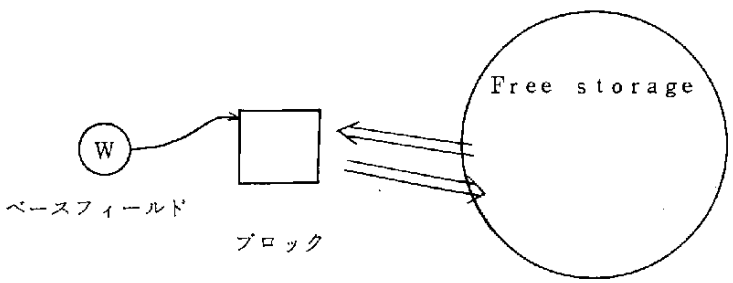


図7　ブロックのGet，Free

取り出されたブロックは"Bugs"と呼ばれるベースフィールド（AからZまでの名前で26個ある）に接続させる（ブロックのアドレスをベースフィールドにセットする）事が出来るので，このベースフィールドを経由して内容を参照する事が出来る。

又あるブロックのフィールドに別のブロックのアドレスをセット出来るので，ベースフィールドを基にしてブロックを適当な構造でつなぎ合せる事が出来る。図8にこの様子

を示す。

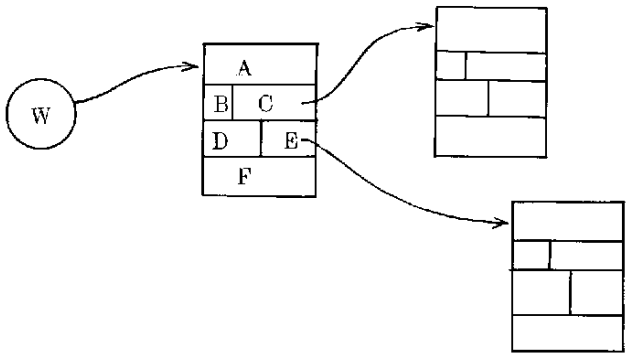


図8　ベースフィールドとブロックの関係

6.1.4　フィールドの参照

フィールドの内容は，ベースフィールドから順にそのブロックをさし示すポインターの入っているフィールドの名前を，連ねて参照する事が出来る。

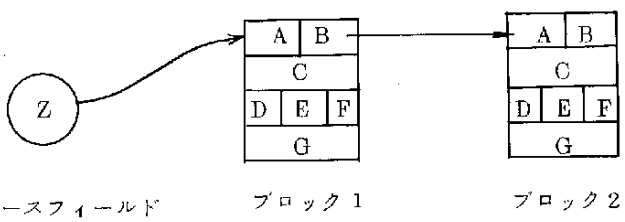


図9　フィールド参照の説明図

例えば図9でブロック2のGフィールドの内容はＺＢＧによって参照する事が出来，同じブロックのＥフィールドはＺＢＥで，又ブロック1のＣフィールドはＺＣによって参照する事が出来る。

6.1.5　オペレーション

ブロックやフィールドに対してL^6では次の様なオペレーションが可能である。

- ブロックをフリーストレッジより取り出す。
- ブロックをフリーストレッジに戻す。

- ブロックをコピーする。

以下フィールドに対して

- 四則演算
- 論理演算
- ビットカウンティング
- シフト
- フィールドの内容をコピーする。
- フィールドの内容を入れ換える。
- 入出力機能

などが行える。

6.2 シミュレーション言語に於けるブロックの構造とフィールド

シミュレーション言語で，ダイナミックに作成，消去されるブロックは，スケジューリングメカニズムと関連して扱われるものが代表的である。しかしSIMSCRIPTのテンポラリーエンティティに見られる様に，かならずしも直接関係ないものもある。

ここでは，各言語でユーザが使えるこれらのブロックとシステム側でコントロール用に作成されるものとを含めて，フィールドの扱いなどとの関係を考察しておきたい。

GPSS で作成されるブロックはトランザクションと呼ばれる。この言語でユーザから直接扱えるブロックはこれ以外にない。

トランザクションを構成するブロックは2つに分かれている。1つはどのトランザクションでも同じ大きさを持つブロックで，ここにはスケジュールテーブルへ登録する為のポインターなどが入っている。

他の一つはコモンエリアに作成されるもので，トランザクションが持つパラメータの数により，ブロックサイズはバリアブルとなる。

図10にGPSS／360に於けるトランザクションのフォーマットを例として示す。

この図からも明らかな様に，ユーザーから直接モディファイ出来るフィールドはパラメータに当る部分のみである。他の一部はGPSS のStandard Numeric Atributeとして参照する事が出来る様になっている。又，GPSS に於けるブロックのフィールドはユーザが任意に定義する事が出来ず，上記のパラメータについてのみその個数をブロック発生時に指定出来る。

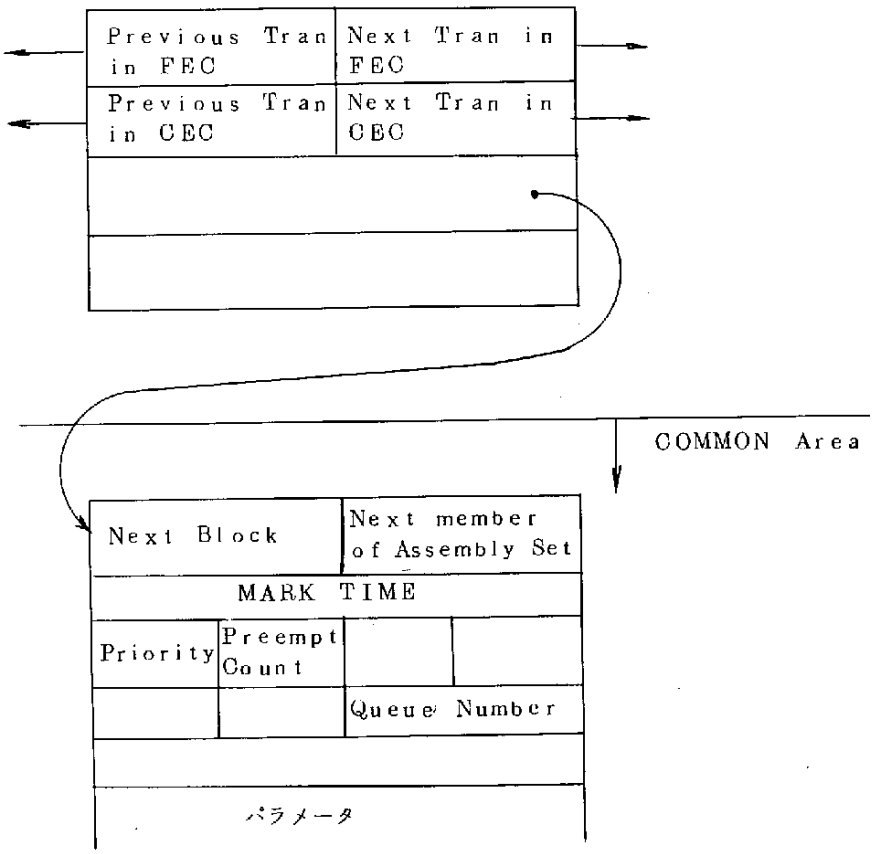


図10　トランザクションのデータフォーマット（GPSS／360の場合）

SIMSCRIPTの場合にはL[6] のブロックに相当するものとして，イベントノーティスとテンポラリーエンティティがある。

イベントノーティスはイベントの為にコントロールブロックとして作成されるもので，その一部は，スケジュールテーブルに登録する為にポインターを入れるエリアを，あらかじめ確保して置かなくてはならない。しかし残りの部分についてはユーザが任意にフィールドの名前と，位置を定義する事が出来る。

テンポラリエンティティは特別のエリアを用意する必要がない点を除いてイベントノーティスとまったく同じである。従ってテンポラリーエンティティのブロックはユーザが全て，

自由に使う事が出来る。

どちらのブロックも，そのサイズについては2^n単位でしか指定する事が出来ない。これはFree Storage の管理を能率的に行う為の制限と思われる。

事実，L^6 に於てもブロックのサイズに，この様な制限を設けており，Free storage の管理を能率的に行っている。

SIMSCRIPT1.5 に於けるテンポラリーエンティティの定義はテンポラリーアトリビュートと関連付けてデフィニションフォームによって行われる。例えば図11に示す様に行われる。

	TEMPORARY SYSTEM VARIABLES																										
	TEMPORARY AND EVENT NOTICE ENTITIES																ATTRIBUTES										
	TEMPORARY OR NOTICE		NAME					RECORD SIZE MASTER	SATELLITE 1	2	3	4	5	6	7	8	TEMPORARY OR NOTICE		NAME					RECORD WORD	PACKII		
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
+	T		E	X	A	M	1	1																			
+	T		E	X	A	M	2	8																			
+	T		E	X	A	M	3	4			8																
+	T		E	X	A	M	4	4	2			1															
+	T		E	X	A	M	5	2																			

EXAM1 は，1語の temporary entityである。

EXAM2 は，8語の temporary entityである。

EXAM3 は，4語のマスターと8語のサテライトからなり，合計12語の temporary entityである。サテライトのレコードの番地は，マスターの3語目の後半語（2／2と表わす）に入る。

EXAM4 は，7語のtemporary entityで，マスターが4語，サテライトの1が2語，サテライトの4が1語からなる。

EXAM5 は，2語のtemporary entityである。

図11　テンポラリーエンティティの定義例

フィールドの定義はＳＩＭＳＣＲＩＰＴではテンポラリーアトリビュートの宣言に対応している。

フィールド名（つまりアトリビュート）は，異るテンポラリーエンティティに共通して定義する事は出来るが，ブロック（エンティティ）に於いて同じ位置を表わすものでなくてはならない。つまり，フィールド名の概念はＬ^{6} の場合と同様で各ブロックごとの区別は行わず，共通して，ブロックに於ける相対位置を表わすものとして扱われる。

図１２はこの様な関係を示したものである。

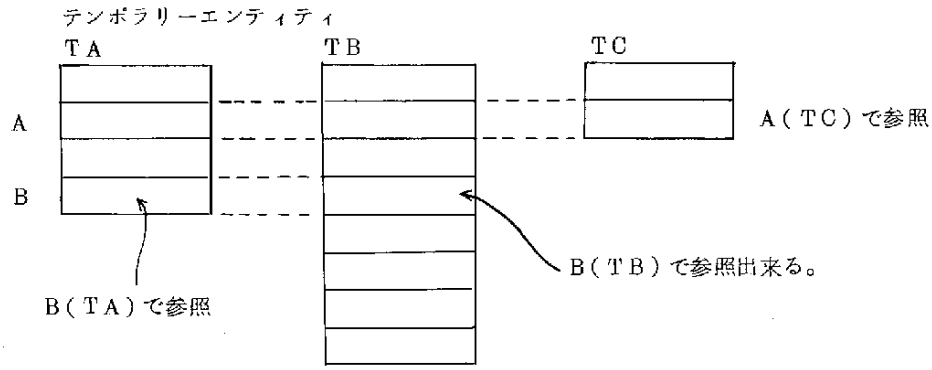


図１２　フィールド名とブロックの対応

テンポラリーエンティティとアトリビュートの定義は，ブロックとフィールドの定義に相当する。これはＳＩＭＵＬＡ に於けるアクティビティとプロセスの関係の様に同じパターンを持つブロックの類を代表して定義することになる。

これをブロック形式と呼ぶことにする。つまりブロック形式とはブロックに於けるフィールドがどの様に定義されているか示すもので，ブロックそのものではない。個々のブロックはブロック形式に従って作成される。

１つのテンポラリーエンティティとアトリビュートの定義が，丁度，１つのブロック形式に対応している。従ってＳＩＭＳＣＲＩＰＴではブロック形式ごとに名前（これはテンポラリエンティティ名）が付いていて他と区別されている。

個々にブロック形式が区別されているにもかかわらずＳＩＭＳＣＲＩＰＴの場合にはフィールド名の扱いがブロック形式ごとに独立でない。

フィールドの扱いはブロックがどのブロック形式から作成されたかにかかわらず，その

「相対的位置」にもとづいて処理される。

ＳＩＭＵＬＡ に於けるブロックは，プロセスに対応して作成される，コントロールブロック（ＳＩＭＵＬＡ でどの様な呼び方をしているか不明なので仮にＰＤＢ（Ｐｒｏｃｅｓｓ Ｄｅｆｉｎｉｔｉｏｎ Ｂｌｏｃｋ）と名付けておく。）とエレメントと呼ばれるブロックである。これらは直接ユーザーからモディファイできるが，他に，プロセスをスケジュールテーブル（ＳＱＳ）に登録する際にシステム側で作成するイベントノーティスがある。

実は，ＰＤＢに関する正確なフォーマットについては不明である。しかしここで扱う問題では，この事はあまり本質的でない。

ＰＤＢは各プロセスに対応するもので，そのアトリビュートとしての性格を持っている。

つまりアクティビティ宣言の中で使われていて，これにローカルな変数はフィールドとしてＰＤＢに割当てられる。従ってＰＤＢのブロック形式はアクティビティ宣言によって定義される。ＰＤＢについてはＧＰＳＳ のトランザクションと同じ様に１つのブロックでなくむしろ２つに分割されている事が考えられる。しかしどちらにしても変数がフィールドとして定義されて，それらの変数名がフィールド名として与えられる事にちがいはない。

従ってＳＩＭＵＬＡ に於けるＰＤＢのフィールドの位置に関してはユーザは一切，関与する事が出来ず，システム側に一任されている。

これはＳＩＭＳＣＲＩＰＴと比べてブロックのサイズとかフィールドの定義を気にする事なく使えて，便利な反面ユーザが自由にフィールドの位置を定義する事が出来ない不便さも持っている。もちろんパッキングの機能を持っておらずＳＩＭＳＣＲＩＰＴほど小細工が出来ない。

ＰＤＢはＳＩＭＳＣＲＩＰＴのテンポラリーエンティティに相当する機能もはたすが，上記のごとく，ブロックに於けるフィールドをユーザが自由に使いこなす事が出来ないので，この様な機能についてはＳＩＭＳＣＲＩＰＴより劣ると言える。

プロセスタイプの言語ではプロセスそのものの概念からして，ブロックの扱いに関してどうしても上記の方法をとらなくてはならず，ブロックの扱いについては鈍重にならざるを得まい。

ＳＩＭＵＬＡ には言語仕様の表面に出されているブロックにエレメントがある。

エレメントは構造体（この場合セットなど）を作る時にコネクターの役割をするもので，ポインターを入れる為に２ｗｏｒｄｓとＰＤＢを参照する為のポインター１ｗｏｒｄ の合計３ｗｏｒｄｓより成っている。

6.3 フィールドのモディファイ

シミュレーションモデルに於てブロックのフィールドをモディファイする問題を考える時，モディファイする側の事を明確にしておかなくてはならない。

シミュレーションモデルはスケジューリングメカニズムの章で見た様に，幾つかのサブプログラムによって構成されている。

しかもプログラムの実行は複雑で，時には同じプログラムが同時にパラレルに実行される事さえある。これらサブプログラムの実行を言語によってイベントと呼んだりプロセスと呼んでいるが，これをまとめてタスクと呼ぶ事にした。

プログラム実行にともなって発生したり消滅させられたりするブロックはこれら個々のタスクによってモディファイされるものである。

L^6 でブロックがベースフィールドから順につながれていれば，これを手掛りにして，そのブロックのフィールドをプログラム中で自由にモディファイする事が出来た。

ところがシミュレーションモデルでは各タスクが任意のブロックのフィールドを簡単にモディファイする事は出来ない。

SIMULA で作成されるブロックは個々のタスク，つまりプロセスに対応付けられている。

たとえ同じアクティビティ宣言により発生されたプロセスであっても，それぞれ別のブロックが作成される。これらの関係を図13に示してある。

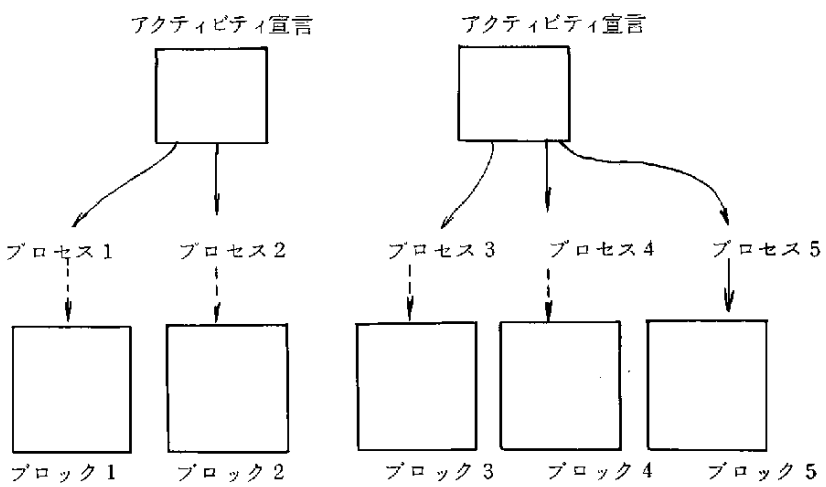


図13　アクティビティ宣言，プロセス，ブロックの関係

ブロックにはアクティビティ宣言の中で使われている変数のエリアがフィールドとして取られている。

このフィールドは単にその変数名を書く事によりアクティビティ宣言の中で使う事が出来る。この場合，変数のエリアは，その時実行しているプロセスのブロックに割付けられるので実行されるごとに別のエリアが割付けられている。

従ってたとえ同じアクティビティに属すプロセスであっても，その中でローカルに定義されているデータを扱う時には，まったく個別に行なわれている。この事は，普通の方法で，他のプロセスにローカルなデータを参照出来ない事を示している。

しかし，プロセスに対応するブロックの性質を考える時，これはプロセスのアトリビュートとしての性格も持っており，シミュレーションモデルの作成については，他のプロセスにローカルなブロックの内容を参照する必要がしばしば起る。これはプロセス間のコミュニケーションの問題である。

SIMULA ではこの為にコネクションステートメントが用意されていて，プロセス同志で，ローカルなデータのモディファイが可能に成っている。

SIMSCRIPTで使うブロックではSIMULA の場合と違ってどのブロックの内容でも，各イベントから自由に参照する事が出来る。

これはブロック形式の定数がSIMULA の様にアクティビティ宣言によってオペレーションルールの記述と同時に行われるのと違い，イベントルーチンとはまったく別に定義される為である。SIMSCRIPTのブロックは言わばグローバルな性格を持っていると言える。

ブロックのフィールドはL^6 流にブロックのアドレスとそのフィールド名によってモディファイする事が出来る。

6.4 OPS－4 の場合

この様なローカルデータの参照手段とかグローバルデータの扱いについてオンラインシミュレータに於ては，特にそれがインタラクティブな実行をする点，かなり異った扱いがなされている。以下OPS－4 の場合について調べてみよう。

アクティビティは別々に書かれたプロシージャであるから，PL/1 では全てのグローバル（外部）変数の宣言が，それを参照する全てのアクティビティにおいてくり返される必要がある。だが，OPS－4 ではその必要がない。OPS－4 は特別のグローバル・シンボル・

テーブルをもっていて，ユーザーがPL/1の宣言ステートメントをコンソールから直接実行すれば，外部アトリビュートが宣言で指定されているかどうかに関係なく，宣言されたシンボルはこのグローバル・シンボル・テーブルに自動的に登録される。即ち，ユーザーは，特に指定しない限り，グローバルなレベルで操作を行なうことになる。これに対し，PL/1の宣言ステートメント（OPS－4プログラム内で現われる外部アトリビュートを有する）は，それが実行されるとき，常に，シンボルの定義をグローバル・シンボル・テーブルに追加しなければならない。

OPS－4でシンボル・テーブルのサーチは，常に，アクティビティのローカル・シンボル・テーブルで定義されていないシンボルがあると，そのシンボルを定義するためにグローバル・シンボル・テーブルをチェックする。従ってOPS－4においては，ユーザーによって直接コンソールから行なわれるにしても，或いは1つのアクティビティによって行なわれるにしても，一度だけ，グローバル・シンボルが定義されなければならない。或るシンボルが二回以上定義される場合には，常に一番新しい定義がその前の定義に取って代ることになる。これによってユーザーは，宣言ステートメントを1つ与えるだけで，変数のアトリビュートを任意の時点で容易に再構成乃至変更できるようになる。

OPS－4プログラムがコンパイルされるとき，OPS－4からPL/1へのトランスレータは，ローカル・シンボル・テーブルでは定義されていないがグローバル・シンボル・テーブルでは定義されているアクティビティにおいて参照された全てのシンボルに対して，外部アトリビュートを有する宣言ステートメントを，自動的につくり出す。

初めにPL/1でプログラムを書きたいなら，ユーザーは全グローバル変数をイクスプリシットに宣言しなければならない。MULTICSで利用できる言語トランスレータによって認められている「ファイル挿入」機能を用いれば，この仕事を簡単にすることができる。この機能によってユーザーは，幾つかのプログラムで用いられるコモン変数の宣言全体を含むファイルをつくることができる。各プログラムでそれらの宣言を書く代りに，ユーザーは唯INSERT FILE XXと書けば良いのである（但し，XXはコモン変数の宣言を含んでいるファイルの最初のネームである）。

ローカル変数は，標準的なPL/1プログラムにおいてと同様，普通の形で取り扱われる。この変数は，それがプログラムによって宣言乃至アロケートされるアクティビティにのみ知られている。互いに区別されているローカル・データ・ベースを識別したり，保持したりする問題はシミュレーション・モデルに固有のものであり，他のインタラクティング・プログ

ラムの集まりからシミュレーション・モデルを区別するものである。しかしこの問題は，幾つかの異なったプロセスによってピュア・プロシージャを同時に実行する場合にMULTICSが遭遇する問題と似ているのである。

6.5 構造体

6.5.1 セット

ブロックを適当にチェイニングする事で作られる構造体の代表的なものとしてセットがあげられる。シミュレーションに於けるセットの概念は「物の集り」を表わす意味で重要である。セットについては2つの問題を扱いたい。1つはユーザサイドから見たセットのオペレーションについて，他の1つはセットのデータ構造についてである。

6.5.2 セットの構造

どの言語でも基本的にセットは，セットヘッドに要素が順方向，逆方向のポインターで対称にリンクされた構造を持っている。

この点SLIP タイプのデータ構造を取っているわけである。ただ違う点は，リンクされる要素が何んであるかということである。図14 に代表的なセットの構造を示す。

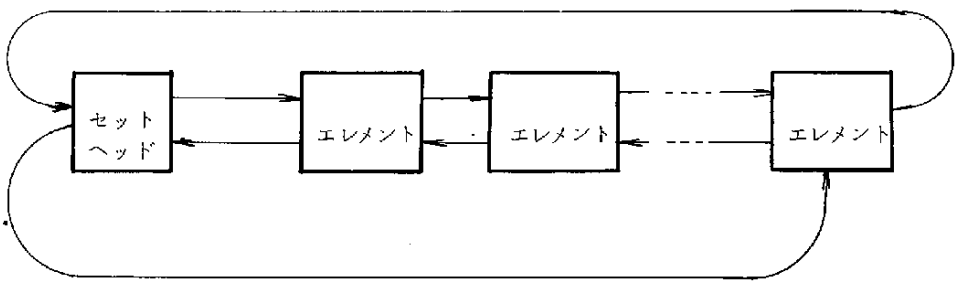


図14　代表的なセットの構造

GPSS の場合にはトランザクションが直接，セット（GPSS ではユーザチェンと言う）にメンバーとして登録されている。又SIMSCRIPTではテンポラリーエンティティやイベントノーティスがセットのメンバーとして，やはり直接リンクされる。

SIMULA に於けるセットはこれらと少々構造が違っている。この言語の場合セットのメンバーとなるのはプロセスであるが，プロセスに対応するブロックが直接セットの要素として登録されずに，エレメントと呼ばれる要素がセットヘッドにリンクされる。エレメントには対応するプロセスを示すポインターが入っており間接的にプロセスがメ

ンバーとして登録されている。図15 にその様子を示す。

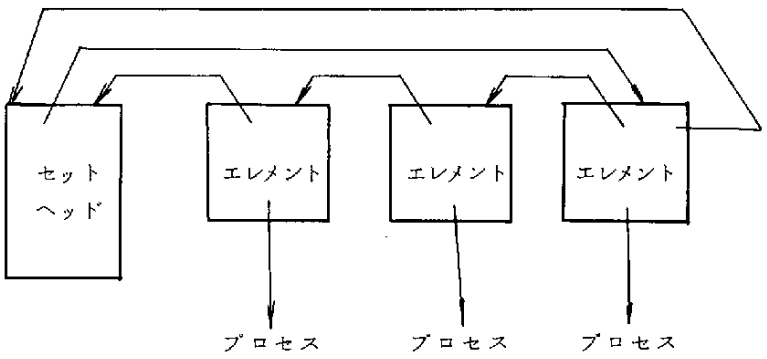


図15　SIMULA に於けるセットの構造

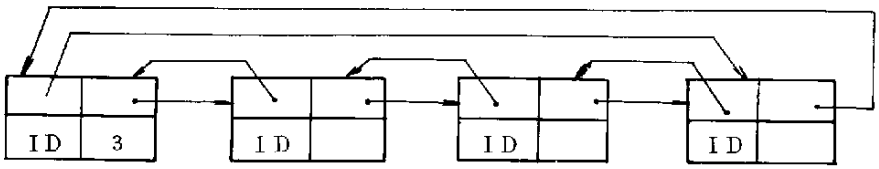
これは1つのプロセスが一時期に1つ以上のセットのメンバーとして登録出来る様に取られた構造であり，もちろん1つのプロセスに対して属しているセットの数だけエレメントが作成される。エレメントを導入した為に，SIMSCRIPTで起る不便な点，つまりもし1つ以上のセットにテンポラリーエンティティを属させ様と思うとテンポラリーエンティティを定義する時あらかじめ，その分だけエリアを確保しておかなければならない事，が解消される。

OPS－4 で扱われるセットはほぼSIMULA と同じ構造をしている。SIMULA に於けるエレメントに相当するものがセットに挿入される時に作られるが，この要素については言語仕様の表面に出ていない点がSIMULA と違っている。又 OPS －4 のセットに於けるメンバーは他の言語と異りスカラー，アレイ，ストリング，ストラクチャ等，幅広い要素をメンバーとすることが出来，たとえセットであってもメンバーとする事が可能である。図16 はOPS－4 に於けるセットの一部を示したものである。

6.5.3　セットオペレーション

普通セットに要素を挿入したり取り出す方法は3つある。

1. First in First out
2. Last in First out
3. Ranked



セット ヘッド

Predecessor	Successor
identifier	エントリー数

メンバーエントリー

Predecessor	Successor
identifier	

エレメント名
或はセットヘッドへのポインター

図16　OPS－4 に於けるセットの構造

これら3つの方法についてはあらためて説明するまでもないであろう。

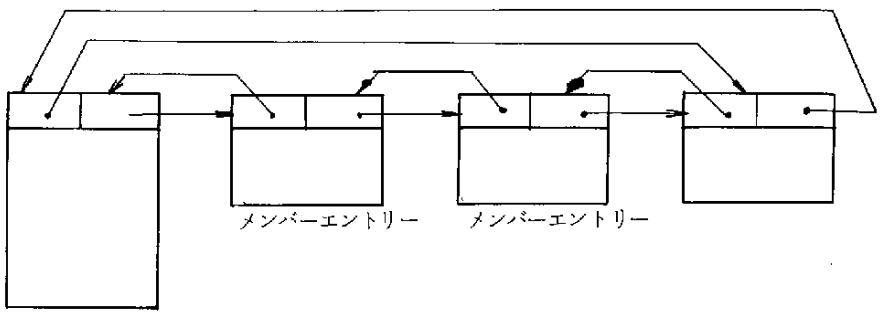
ただ，この様な方法による要素の挿入，削除は，特に削除する場合にあくまで特定の位置（つまり先頭とか最後）にある要素を対象としておりセットの中から特定要素を名ざして取り出したり，セットの任意の位置に要素を挿入したり出来ない。これらを可能にしたのが，SIMULA でのセットオペレーションである。もちろんこの方法でも上記の3方法は簡単に実現する事が出来る。例えばFIFO（First in First Out）にしたければ要素をいつも最後に挿入し，先頭より取り出す様に位置を指定すれば良い。

6.5.4　キュー及びテーブル

シミュレーション言語ではセットの外にも幾つかの構造体が存在する。キュー（Queue）とかテーブル（Table）がそれらであるが，これらのものはどの言語でもそれほど大差はない。

ここに例としてOPS－4の場合を載せておこう。

キューはセットと大変良く似ている。事実，セット処理ステートメントは直接キューにも用いることができる。セットとキューの違いは，OPS－4がキューのサイズをモニタリングしたりキューにおける経過時間を計算したりするためのステートメントを幾つか備えている，という点にある。結局ユーザーは，キューの利用率の統計測定を集めたい時にのみ，キューを用いることになる。図17にこの構造を示しておく。



Queue Head

Queue Head

Predecessor	Successor
Queue Headを識別する為のID	Queueに入ったエレメント総数
現在Queueにあるエレメント数	エレメントの最大数
作成時刻	
Queueが最近に参照された時刻	
maximum duration	
minimum duration	
total duration	

メンバーエントリー

Predecessor	Successor
0	エレメントへのポインター
Queueに入った時刻	

図17　OPS－4に於けるQueueの構造

テーブルはGPSS やSOL やOPS－3 で用いられているテーブルと似ており，これを用いることによってユーザーは容易に分布を知ることができる。テーブルの宣言ではテーブルの範囲の上限と下限，及びきざみ幅を指定する事が出来る。

タブレイト・ステートメントによってユーザーは，指定したカウント数だけ（指定がなければ，1カウントだけ）テーブルをアップデートできる。また，種々のディスプレイ・ステートメントによってユーザーはテーブルを表の形式でプリントできるし，密度分布か累積分布として，また棒グラフか破線グラフとして，それをプロットすることもできる。これらのディスプレイ・ステートメントによって，ユーザーは，ディスプレイされるテーブル・エントリの範囲を指定することができるし，きざみ幅を変えたり，隣接している各きざみを幾つかのグループにまとめたりすることもできる。テーブルの範囲が大きい場合とか，ユーザーがエントリーテーブルのサブセットを広い観点から見たい場合には，この柔軟性が非常に重要な意味をもってくる。ユーザーが特にアウトプット形式を指定しないなら，標準的な形式でアウトプットされる。これらのディスプレイ・ステートメントは，シミュレーションのどの時点でも実行可能であり，優れたデバッギング手段を備えている。

6.6 オンラインシミュレータに於けるデータベースの扱い

データストラクチャーに関する今までの章は，むしろ特にオンライン化を意識しない部分が多かった。以下はオンライン化された場合のデータの取扱に関してOPS－4 の場合を述べる。

6.6.1 グローバル・シンボル・テーブルの内容

グローバル・シンボル・テーブルは，凡ゆるアクティビティにアクセス可能なグローバル変数の他に，他の多くのエンティティの定義を含んでいる。それはパーマネントセクションとトランジェント・セクションとに分けられる。パーマネントセクションは，OPS－4 言語の標準的なステートメント全てに関するエントリー（セグメント・ネームとエントリー・ネームを与える）のリストを含んでいる。シンボルテーブルのこの部分は一般ユーザーがモディファイすることができない。

トランジェント部分は次のようなタイプのエントリーを含んでいる。

a) 全てのグローバル変数の定義

b) まだコンパイルされていない全てのOPS－4 プログラムのネーム

c) 全てのライブラリ・プログラム（user portrayed program）のネーム

ユーザーは，トランジェント・シンボル・テーブルにおけるエントリをどれでも好きな時に削除できる。OPS－4 システムは，OPS－4 プログラムのネームがOPS－4 によってPL/1 トランスレータにコンパイルされるとき，そのネームを自動的に削除する。ライブラリ・プログラムのネームは，ユーザーが指定して削除しなければならない。

6.6.2 グローバル・シンボル・テーブルの取り扱い

グローバル・シンボル・テーブルは，通常どのシミュレーション・モデルでも重要な要素になっている。従って，OPS－4 システム内で直接それを取り扱うために，特殊な機能が用いられることになる。ユーザーは次のような取り扱いができる：

1. ユーザーは現在のシンボル・テーブルにネームを付けて区別することができる。（これは，MULTICS のリネーム・コマンドによっても行なうことができる。）
2. ユーザーは，現在のシンボル・テーブルを削除せずに，特定のセグメントを置き換えることができる。（これは，MULTICS の２つのリネーム・コマンドによって行なうことができる。）
3. ユーザーは，古いシンボル・テーブルと（それを削除することなく）置き換えられた新しいシンボル・テーブルを初期設定し，それにネームを付けることができる。
4. ユーザーは，現在のシンボル・テーブルの初期設定をすることができる，例えば，全ての定義を削除することができる。
5. ユーザーは，現在のシンボル・テーブルにセグメントを附加することができる。（シンボル・テーブルは，ハッシュ・コーディド・テーブルであるから，セグメントを単純につなげるだけではすまないことに注意せよ）。
6. ユーザーは，現在のシンボル・テーブルの内容のリスティングを得ることができる。

以上のオプションは，全てシンボル・テーブルのトランジェント部分にのみ関するものである。OPS－4 の全シンボル・テーブルは，ユーザー自らがそれを行なわない限り，削除されることはない。

6.6.3 OPS－4 データ・ベースの取り扱い

ユーザーは，OPS－4 システムによって作られたデータ・ベースを取り扱う際にも，同様な機能を必要とする。特に，ユーザーが同一モデルで，データを種々に変えて実行してみたいときや，幾つかのモデルを試してみたいときは，現在のデータ・ベースにネ

ームを付けて区別したり，現在のデータ・ベースを別のデータ・ベースと代えたり，更には現在のデータ・ベースをクリアしたりする機能が必要になる。また，ユーザーがインクリメンタルにモデルを作って行く過程で，種々の断片的なデータ・ベースを結合できるかどうかが重要な意味をもってくる。

これらの機能は，大部分，MULTICS コマンドによって行なうことができるが，その場合シミュレーション・データ・ベース以外に幾つかのセグメントを付け合わせて行なわなければならないので，不必要にわずらわしくなってしまう。一定のモデルに関するデータ・ベース全体を同時に取り扱える特殊なOPS－4 ステートメントがあれば，それは大変便利なことなのである。

6.6.4　プライベートなデータ・ベース

複数個のアクティビティがお互いにデータをアクセスしたくても，そう簡単にはいかない。従って，これらのアクティビティは，ローカル・データ・ベースや一般的なグローバル・データ・ベースの他に，プライベートなセミ・グローバル・データを必要とする。コンパイルされたルーチンにおけるこの種の問題を記述するには，PL／1の外部宣言アトリビュートで足りる。外部宣言を含んでいるこれらのルーチンだけが，これらプライベートなグローバル変数の存在を知っていることになる。従って，種々のルーチンは外部宣言の色々な組合せをもつことができるのである。２つのルーチンに共通なデータがそれぞれの宣言で定義されていなければならない。これは，FORTRAN IV のネームド・コモンの特性と似ている。従って，全ての外部変数は，同じデータ・セグメントにストアされるけれども，勝手に利用することはできないのである。

しかしながらOPS－4 プログラムには，コンパイルされたプログラムに対するようなプロテクションが存在しない。グローバル・シンボル・テーブルで宣言された変数は全て，他の凡ゆるOPS－4 プログラムに用いることができる。特に宣言されたプログラムだけにアクセスを制限するためには，宣言ステートメントにおいて許される標準的なPL／1のアトリビュートの他に，アクセスと呼ばれるアトリビュートが附け加えられねばならない。このアクセス・アトリビュートの後に，宣言された変数をアクセスできるプログラムのリストが続くこともある。逆にアクセス・アトリビュートがない場合には，これらネームド・プログラムのアクセスはできず，それ以外のアクセスは，全て可能になる。これらのリストはグローバル・シンボル・テーブルにストアされ，OPS－4プログラムが変数を参照できるようになる前に，OPS－4 によってチェックされる。

OPS－4 プログラムがコンパイルされるときアクセス上の制限を蒙らず，特に，コンパイルされているプログラムへのアクセスが可能であるような全てのグローバル変数に対して，OPS－4 からPL/1 へのトランスレータが外部宣言を発生させる。

コンソールから直接ステートメントを実行している場合，ユーザーは常に全変数をアクセスできるのである。もしそうでない場合には，ユーザーは，プログラムを，そのプログラム内でしか，デバッグできない。このような制限は大変不本意なものである。

6.6.5　階層的に構成されたデータ・ベース

MULTICS デバッギング・パッケージによって，シンボル・テーブルを階層的に構成することができる。従って，対応するデータ・ベースが互に階層的に関係づけられることになる。これによって，ブロック・ストラクチャの概念を，別々に書かれているか或いはコンパイルされているルーチンにまで実行中に動的に拡張することができるようになる。シンボル・テーブルのリンケージは，上層ルーチンから下層ルーチンへのコントロールの流れとして動的に作り出されることになろうし，OPS－4 における全てのシンボル・テーブル探索は常にローカル・シンボル・テーブルでスタートして，それからグローバル・シンボル・テーブルへ進むから，階層的に上層のOPS－4 プログラムで宣言された変数は全て，下層のOPS－4 プログラムにも知られることになろう。

第7章　統計データの収集と解析

シミュレーションに於ける統計収集の問題は，モデルを記述するプログラムが完成してしまってから考えられる場合が多い。そのため，取るべき統計を考える段階で，モデルの構造或はデータベースを基本的に変更しなければならないはめに陥る事もしばしばあるであろう。

この様なやっかいな問題を回避するために，従来のバッチ処理シミュレーション言語に於ても様々なアプローチが取られている。

統計収集もシステム側でどこ迄自動的に行なうかと云う臨界点の設定に関しては，各種言語に於てかなり明確になされている。

各種統計量をシステムが自動的に収集する事により，ユーザの負担を軽減させる事も可能である。この場合ユーザーの負担の軽減の度合に応じシステム負荷が増大する事は云うまでもない。更に自動収集を行なうとなると別の面で複雑な問題を抱え込む事になる。

収集すべき対象の選定と云う問題は，自動収集と云う概念をふえんして見た所で解決し得ない問題を含んでいる。

どの様な統計量を収集すべきかと云う事をシステム側で決定してしまう事は，そのシステムの柔軟性を枯渇させる事になるかも知れない。

GPSS 言語は自動統計収集を行なう言語である。GPSS に於ては，自動的に取るべき統計量として待行列に関するものとエンティティに関するものとがある。エンティティがどこ迄ユーザーの望む統計収集対象を表現し得るかによって，GPSS の使用範囲が限定される訳だが，エンティティで代表される対象の広汎な事と，利用頻度の高い待行列統計とによって，柔軟性を保持しているとも云える。

しかし乍らシステム側で自動収集しない統計量を得たい時などは，GPSS のように自己完結しすぎる言語に於ては，致命的である。

シミュレーション・システムが提供する範囲を超える要請をも受け容れるために，GPSS では，HELP ブロックを用意し，拡張使用を許している。これはまさにオートマチックの不備をマニュアルで補償しようとする行き方であり，GPSS 的な形態を取る言語が当然持たねばならないものであるが，GPSS に於けるHELP ブロックは次の様な点で好ましい解決を与えるものとはいい難いであろう。

何故ならこのブロックを使用するためには，シミュレーションに取って本質的でないいくつかの問題を解決しなければならないからである。

例えば，その1つとしてGPSS の構造を細部に渡って詳細に知る事や，GPSS と連結すべき言語の構造を熟知する事等々である。又GPSS の提供する出力形式が一定である事やコメントを任意に挿入し得ない，等々と云った不満が残されたままになっている。これに反してSIMSCRIPT の様に，自動的な統計収集は一切行なわずに，ユーザーの手に総て任かせてしまう行き方もある。

ユーザーの負担と云う点では，問題が残るかも知れないが，システム側で収集統計量を限定するという困難な問題を回避し得るし，柔軟性の点でも優れたものとなるだろう。ここでは，平均とか標準偏差と云ったものを計算するステートメントのみをシステムが提供するのである。その使用がユーザーに任されているだけに自由度の高いものである事は云うまでもない。

しかし乍ら使用頻度の高い待行列統計などをそのつど書かなければならないと云う事は，非常に煩わしい事である。

出力形式に於ては，レポート・ジェネレーターの利用により所望のものが得られる利点を持っている。

上述の対照的な2つのアプローチの中間を行くものとしてOPS－4 を挙げる事が出来よう。OPS－4 に於ては使用頻度の高い待行列統計だけは自動的に収集し，他の総ての統計量の収集はユーザーに任かせている。

以下OPS－4 で行なっている統計収集を見てみよう。

7.1 時間に関する統計

イベント中心の離散系シミュレーションでは時間に関する統計が必要になる場合がある。たとえば，「平均処理時間」「平均待ち時間」あるいは「平均利用率」といった様なものである。

離散系のシミュレーション・システムはとびとびに時間が動かされるので，無条件にスケジュールされたアクティビティの実行時間を追跡しなければならなくなる。

条件付きでスケジュールされたアクティビティに関しては： イベントの生起時刻が既知でないので時間に関する統計としてイベントが実行されるまで待たされた時間が取られる。ディレイの間隔は現在時刻から実行時間を引くことによって計算される。OPS－4 に於て実行時間ETは，シミュレーション時間の更新にも用いられるのでエイジェンダスキャン

（Agenda Scan）の行なわれるメインブランチの各々の条件付きエントリーに対して，このためのストーレッジを一つ割り当てている。

イベント間のシミュレート時間は，あるアクティビティによって消費された時間を表わし，この間隔は設備利用率を計算するのに用いられる。そのために，エイジェンダー上のエントリーにつき２つめのストーレッジが割り当てられている。このストーレッジは，アクティビティで実行された最後のイベントの実行時間（ＬＴ）を含んでいる。したがって，イベント間におけるシミュレーション間隔はＥＴからＬＴを引くことによって簡単に計算される。最後に，アクティビティ中心のシステム（アクティビティは一連のイベントからなっている）においては，アクティビティの総経過時間を知る必要がたびたび生ずる。この為に，エイジェンダー上の各エントリーについて３つ目のストーレッジを割り当てている。それは，アクティビティが最初に計画された時間（ＢＴ）を記録するためのものである。（ＬＴ），（ＢＴ）は時間の更新に直接関係する部分ではないのでエイジェンダ上におかれたエントリーにＡＤＢへのポインターを持たせＡＤＢ上にそれぞれ１ワードづつの領域を持たせてある。これら３つのデータは各々ＥＴ，ＬＴ，ＢＴというシステムが定義した関数を通して利用できる。さらに条件付きで実行された，アクティビティの待ち時間は関数ＷＴによって得られる。アクティビティにおける現在のイベントと先行するイベントとの間のディレイ間隔は関数ＤＴによって得られ，アクティビティの総経過時間は関数ＡＴによって得られる。（ＥＴ），（ＬＴ），（ＢＴ）の所在を図１８に示す。

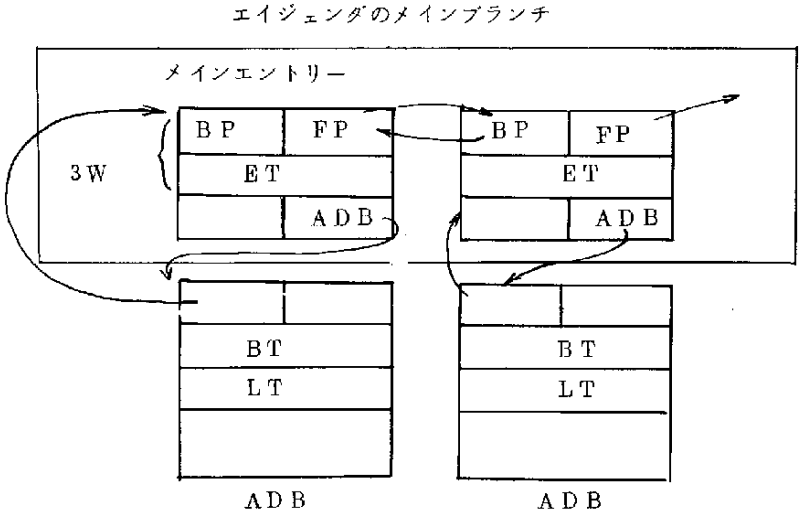


図１８　メインエントリーとＡＤＢに於けるＥＴ，ＢＴ，ＬＴ

7.2 基本的な統計処理

平均，標準偏差，分散といった基本的な統計量を計算するためにＳＩＭＳＣＲＩＰＴと同様なアキュムレイト・ステートメントが用意されている。例えば

ACCUMULATE number sum and sumq of X in XNUM XS and XSQ

は変数XNUM を１づつ自動的に増加させ，変数ＸＳにはXの総和，ＸＳＱにはXの二乗和が計算されて入れられる。

更に，平均，分散，標準偏差を計算するために，コンピュート・ステートメントが利用できる。

COMPUTE mean MX var VX st-dv SDX from XNUM SX and XSQ

更に，

COMPUTE mean MZ and var VZ of Z

の様に書く事によりZ(1)〜X(n)までの配列内の平均や分散を計算する事も可能である。

7.3 分布の収集と表示

多くのモデルでは，データの平均，分散，標準偏差のほかにデータ値の完全な分布が必要になる。一般には分布を得るためにかなりの量のプログラムを書かなければならない。これを簡単にするためにＯＰＳ－４ では３つのオペレーターが用意されている。

(1) テーブルの名前，下限，間隔，上限を与えて，テーブルを定義するオペレーター。

例えば

DECLARE T table 10 5 100

それは１０〜１５……９５〜１００ までの離散的な間隔をもつテーブルＴを定義している。この１８の間隔のほかに，アンダーフローとして値１０以下のもの及びオーバーフローとして値１００以上のものが自動的にテーブルに付け加えられる。

(2) テーブルにデータを書きこむためのタブレイト・オペレーター

TABULATE X in T weight W

ここでXというのは，テーブルＴのインデックスでありＷは加算する重みである。もしＷが記述されていなければ，１とみなされる。もしＸが２５であれば重みＷは２５〜３０の間隔の間に加えられる。しかしながら，Ｘが１００である場合はオーバーフローとしてではなく９５〜１００の間に加えられるようになっている。

(3) テーブルを表示するためのディスプレイ・オペレーター

DISPLAY T

これによりテーブルの内容が表示され各エントリーの個数，全エントリーの総和，平均，標準偏差及び個々の区間の総和，百分率，累積百分率も表示される。また，テーブルの一部分を表示することもできる。例えば

DISPLAY T from 30 to 75 cell 15

により３０以下，３０～４５，４５～６０，６０～７５，及び７５以上の５つの区間を表示する事が出来る。もし，ここでのテーブル間隔が頭初のものよりも小さければフラッグがセットされ，ここで定義されたテーブル間隔が用いられる。

更に，分布を棒グラフ及び破線グラフで描くこともできる。また分布及びその累積分布の表示も可能である。例えば

PLOT T bar from 25 to 80 cell 5/20

は，テーブルTの２５～８０までのものが棒グラフとして表示され，刻みとして５が用いられ横軸方向には２０単位まで書くことができる。もし，横軸方向で利用可能なスペースをこえる様なことがあればフラッグがセットされ最大の幅が用いられる。

グラフの高さは利用できるスペースの量により制限される。このような区分の調整といったようなやっかいな問題はプロット・ルーチンによりなされ，もし範囲をこえるようなことがあればフラッグがセットされ，有効最大長が自動的にセットされるようになっている。

PLOT CUM of T line

は，テーブルTの累積分布を破線グラフとして表示する。刻みの調整は自動的に行なわれるので縦軸方向のスケーリングに関しては，なんら問題はない。有効スペースは０～１までとして一様にスケーリングされる。

例えば

TABULATE WTIME in WT

これにより，WTと呼ばれるテーブルに，処理に対する待ち時間がタブレイトされる。変数Qが，アクティビティが実行されるまでの待ちの長さを表わすとき，Qの長さを示す確率分布QDは，Qの長さが変えられる直前に次のステートメントを実行することによって得られる。

TABULATE Q in QD weight Sys-Time-OTime

ここで変数OtimeはQがモディファイされた最後の時間を示すように更新されている。もし完全な分布でなく平均待ち長さが必要ならば以下のステートメントを実行すれば良い。

ACCUMULATE number and sum of Q*(Sys-Time-O Time) in QNUM and Q SUM

7.4 Queueに関する統計

OPS－4 に於けるQueueの構造は6.5.4の図17に示してある。自動統計収集を行なう関係でQueue head に対してしかるべきエリアが割り当てられている。

キューヘッドに対しては記憶管理のために2つのポインターを持つ1ワード長のヘッドリストがあたえられる。ヘッドリストの1つのポインターは，Qヘッドを指し示し，もう1つはヘッドリスト上の次のエレメントをさす。Qheadは8ワードから成っている。

Queueのメンバーは3ワードから成り，前方向及び後方向を指し示す2つのポインターを持つ対称リスト構造をなしている。

通常のセット処理ステートメントにより新しいエレメントがQに挿入されたり，あるいはQから除去されるたびにすべてのQに関する統計は自動的に更新される。それらの統計量と云うのは先に図に示したようにマキシマムコンテンツ（Maximum contents），アベレッジコンテンツ（Average contents），カレントコンテンツ （Current contents），トータルエントリー（Total entry)を含んでいる。

カレントコンテント，アベレージコンテント，マキシマムコンテント，トータルエントリー等は，キューの名前を付けて関数Size q, AVG q, Tmax q, Total q,をコールすることによりいかなる時点でも直接値を得ることができる。Qに関する時間的な統計は，関数avg-time, max-time, min-time, zero-time により得られる。それらは，ディスプレイ・ステートメントを用いることにより同時に得ることもできる。Qにおける特定なエレメントの待時間は，Qネーム及びエレメント名をアーギュメントとして関数Qタイムにより計算される。

7.5 時系列表示

OPS－4 ではダイナモにおいて提供される機能と同様な時系列表示も可能である。その為にはプロットと呼ばれる属性を変数に付随させれば良い。この属性はスケールの最大値及び最小値ならびに表示文字（同時に表示する他の変数と識別するためのもの）を記述することによりあたえられる。もし最大値あるいは最小値が記述されていなければ，0～100までの標準的な範囲が取られる。また表示文字が示されていなければ変数名の最初の一文字が

表示文字として取られる。エイジェンダー・スキャンを通じてもし無条件アクティビティがアクティブになり，その結果システム・タイムが更新されるとプロット属性を持つすべての変数が表示される。同一の実行時間（ＥＴ）を持つアクティビティ（条件付き無条件にかかわらず）がアクティブになってもプロット属性を持つ変数は表示されない。この表示ルールはユーザーによっていかなる時点でも修正することができる。それは，チェンジ・プロットと云うルーチンを呼ぶことにより行なわれる。変更ルールは４つあり，

(1) すべてのアクティビティがアクティブになった時（all）

(2) すべての無条件アクティビティがアクティブになった時（ unconditional）

(3) すべての条件付きアクティビティがアクティブになった時（conditional）

(4) ノーマルモード（normal）

Change plot ルーチンにより

プロット属性が修正され個々の変数が動的に再宣言され，スケールの最大値及び最小値が修正され，表示文字が変えられる。時刻は離散的に変えられるので表示された結果と云うのは多くの起伏をともなう。時間軸は決まった間隔で自動的にラベルが付けられる。もし，標準的なモードが用いられず時間軸が増加しない場合はそれ以後の表示は同じ場所になされる。上述したチェンジ・プロット・ルーチンは，

(1) 単位分割あたりの時間量の修正

(2) ラベルを付ける間隔の修正

(3) 同一時刻に起った幾つかの変数を異ったフレームに表示させる。

為に用いられる。

この様にＯＰＳ－４では，時系列データの表示とか，累積分布の表示とか云ったものが簡単に行なえるようになっている。

使用頻度の高いものは自動的に収集し，それ以外のものはユーザーの責任に於て収集させる（但しそのために必要な命令は，使い易い形で提供する）と云う行き方は，システムの柔軟性と云う点からも，ユーザーの負担の軽減と云う点からも好ましい解決策であろう。

第８章　モデルのランニングとディバッキング

トレーシング機能の利点は，その種類が豊富だと言う点にあるのではなく，柔軟な使い方ができると言う点にある。トレーシングは初期のコンピューターでも利用されていたが，その形式には限界があった。事実ＩBM７０４は特殊なハードウェア機能をもっていて，プログラム内でのコントロールの転送をトレーシングできたのであるが，それはほとんど用いられなかった。この機能の難点は情報量が多すぎるという点にあり，同様な難点がGPSS ⅡのTRACE 及びUN-TRACE 機能にもある。

これら２つのトレース機能は，ユーザーを実行中のプログラムから切り離してしまうと云う欠陥を持っている。バッチ処理形式で操作するユーザーは，トレーシングをスタートさせる時とストップさせる時とを推測しなければならない。推測がはずれた場合，ランを浪費したり，マシンの次のショットを待たねばならなくなってしまうから，ユーザーは，トレーシング範囲を過度に大きく指定するようになりがちであるし，その結果アウトプットも膨大な量になってしまう。

オンライン・シミュレーションに於ては，この様なジレンマに陥る事はなく特に，推測がはずれた場合には直ちにもう一度実行することが出来るようになる。しかしながら，もっと重要なのは推測を排除できると言う点である。ハンターが獲物を追いかけるようにシミュレーションの経過を見守り，テストをくり返すことが可能であり，ユーザーを問題に集中させる事ができる。ユーザーは，プログラムのバッグのあるところが危急を要する場合には，どんなタイプのトレーシングでも指定することができる。間違いが警告されると直ちにトレーシングをやめ，バッグが見つかればすぐさまランをアボートさせてエラー訂正を行ない，エラーのある結果を正しい結果とおき代えてシミュレーションを続行させ，さらにバッグを見つけたり，モデルに残っているものが正しく構成されているかどうかを見ることができるのである。

ＯＰＳ－４に於ける豊富な各種トレース機能は上記の事柄をほとんどつくしている。そこで以下にこれらの機能について順に説明して行く事にする。

8.1 トレース指定

GPSS 等が提供するトレース機能は無駄な情報を多く含むので，必要な程度に応じてレベル付けでトレース出来る事が望ましい。

OPS－4 では2つの違ったレベルでトレースの指定ができる様になっている。ユーザーはトレースをグローバルに実行するように，指定することもでき，あるいは個々のステートメントにタグを付けておいてトレースすることもできる。OPS－4 システムで指定可能なトレース・オプションのスイッチは，システム側で管理しており，これらスイッチはTrace（No Trace）ステートメントを実行することによりセット（リセット）することができる。

8.2 コントロール・フローのモニタリング

シミュレーション・モデルにおいて，アクティビティ間のコントロールの流れは多分に確率的要素を含んでいるので，その流れをモニターできることはデバッギングに大変役立つ。OPS－4 では，さらにアクティビティ間でやり取りするパラメータの値もわかるようになっている。すべてのアクティビティやプロシージャの呼び出しに関する，モニタリングは，呼び出しが終わる前にコール・トレース・スイッチの状態を調べることによって行われる。もしコール・スイッチがオンならば，コールされたアクティビティあるいはプロシージャの名前が表示される。また指定があれば，すべてのパラメータの名前やリテラルの値も表示される。

このようにアクティビティ相互の関連が，ユーザーの意図した通りに実際の対象を表現して居るか否かをチェックし得る事は，モデルの信頼性と誤りの検出と云う点で有力な武器となり得るのである。

8.3 シミュレーション時刻のモニタリング

離散系モデルのシミュレーションにおいては，シミュレーション・クロックの変化を知ることは大事な要素であり，このシステムでは，その為に2つのモニタリング機能が用意されている。

(1) システム・タイムのトレースを指定する機能。システム・タイムが変わるごとに，旧時刻と新時刻，及び時間変化を起したアクティビティが表示される。

(2) すべてのアクティビティに関する実行時間（ＥＴ）のトレースを指定する機能。コントロールが1つのアクティビティから他のアクティビティに流れる度に，新しいアクティビティのＥＴを表示する。

8.4 ステートメント・ラベル参照のモニタリング

これは，ステートメントが実行されるごとに，そのステートメント・ラベルを表示する機能で，プロシージャ内のコントロールの流れがはっきり分るので，便利なトレース機能である。

8.5 アクティビティ・コールのモニタリング

アクティビティの名前とETの他に，トレーシィング機能が必要になってくる場合がしばしばある。その場合，アクティビティのパラメータやアクティビティ内のエントリー・ポイントの名前や，さらに，すべてのコンディショナル・エントリーと関連する条件などを参照することが重要になってくる。この機能は，アクティビティ・トレースと呼ばれ，プロシージャ・コールのトレースと似ているが，これら2つのトレース機能の重要な違いは次の点にある。

(1) アクティビティ・トレースはアクティビティ・コールにだけ適用され，プロシージャ・コールには適用されない。

(2) アクティビティ・トレースは，常にパラメータや，エントリー・ポイントの名前や，アクティビティ・エントリーと関連している条件を表示する。

プロシージャ・コール，ステートメント・ラベル及びアクティビティ・トレースは，プログラムにおけるエントリー・ポイントの名前を表示する手段を3つ備えている。3つのトレース・スイッチが同時にオンになるなら，エントリ・ネームを3回も表示するのは無駄なことになるが，この問題は特定の順番でトレース・スイッチをテストすることによって解決される。最も包括的なトレースが最初にテストされ，最後に特定のトレースがテストされる。従ってアクティビティ・トレースの指定があった場合には，プロシージャ・コールとステートメント・ラベルのトレース・スイッチはテストされない。

8.6 エイジェンダーの変化に対するモニタリング

エイジェンダーにおけるアクティビティのスケジューリング，再スケジューリング，インターラプティング，再開始及びキャンセリングはモデルを作る人にとって最も興味のあることである。エイジェンダーに何らかの変更が生じた場合は，システム・タイムの値やアクティビティの名前（これはエイジェンダーやそれ自身をモディファイする）を知ることが重要になってくる。エントリーは，アクティビティ・シークェンシング・ステートメント（ac-

tivity sequencing statement) によってエイジェンダーにおかれる。すべてのシークェンシング・ステートメント(外部ステートメントであれ内部ステートメントであれ)による変更も，特定のシークェンシング・ステートメントによる変更もすべてモニターされる。たとえばTrace Entriesはシークェンシング・ステートメントによる変更を表示し，Trace Delay EntriesはDelay ステートメントによる変更を表示する。どちらの場合にも，システム・タイムの値とモディファイを行なうアクティビティとが表示される。

8.7 エイジェンダーのモニタリング

もう1つ重要なトレース機能として，次の対象となるべきエントリーが選択される時，エイジェンダーのスキャンを追いかける機能があげられる。Trace Agenda の指定は，スキャン中に対象になるすべてのエントリー(次に選ばれるエントリーも含む)に対して，アクティビティの名前やその条件やアクティビティのパラメータを表示する。これは，イベントのシークェンシングが計画された通りにいっていない時には特に有力なデバッギング手法になる。Display Agendaをもちいれば，エイジェンダーの全体乃至，その一部を，アクティビティ内からでもコンソールからでも，表示させることができる。全エイジェンダーの表示によって，モデルを作る人はシミュレーションの将来の方向を概観できるようになる。エイジェンダーのスキャン・トレイスは現在のアクティビティに焦点を絞っている。OPS－3に関する経験は，これら2つが大変有効なデバッギング手段であることを示している。

エイジェンダーは種々のデバッギングに用いられる。通常，エイジェンダーはアクティビティのコールしか含んでいないが，OPS－4 のステートメントをエイジェンダーにおくことも可能である。DO LOOP ステートメント，GO TO ステートメント，I/O ステートメント等もすべてエイジェンダーにおかれる。事実，Trace及びNO-Trace ステートメントをエイジェンダーにおくことは，シミュレーションにおける．トレーシングの期間をコントロールする有効な手段になる。これらステートメントは，Schedule 乃至Cancel ステートメントを用いて挿入乃至削除される。

8.8 ステートメント実行のモニタリング

実行されるすべてのステートメントをアクティビティ内でモニターすることが可能である。

すなわち或るステートメントが実行される前に，ステートメント全体が表示されるのである。これは，プログラムにおけるコントロールの流れをモニタリングする最も完全な手法と言える。それは大量のアウトプットを発生させるけれども，モデル開発の初期の段階では特に有効であり，バッグが他の方法で発見できない時にも有力な手段となる。ステートメント実行の結果を表示する機能がこれと関連している。このトレースは，結果を定義できるようなステートメント（例えばSet，Draw，If 及び Repeat ステートメント）に対してのみ意味がある。

8.9 特定変数のモニタリング

OPS－4 ステートメント，における特定の変数の参照乃至はその変数の変更をモニタリングすることは，特に有効なデバッキング手段である。これをできるだけ効率的に履行するために特殊な機能が設計されている。変数がOPS－4 ステートメントのパラメータとして参照される場合には，常にその変数の名前と現在の値とが表示される。結果のトレースは新しい値を表示するのに用いられるから，新しい値をとっている変数は，それがこわされる前に，すぐ前の値を取ることになる。

アレイ変数のトレーシングはそれぞれの場合に応じて適切に用いられなければならない。ユーザーは，全アレイが，代入ステートメントによって参照される時に表示されるようにすることもできるし，アレイ内の特定の要素だけをモニターすることもできる。全アレイが参照される時，トレースされているアレイ内の要素を1つだけ指定することはできない。しかしながら，アレイの一部だけが参照されたりモディファイされたりしている場合には，参照された部分だけが表示されることになる。値そのものの他に，常に添字の値も表示される。

8.10 変数の自動定義とエラーのモニタリング

さらに3つのトレース機能が利用されている。その1つはTrace Error であり，プログラムの実行中にエラーがみつかると，完全エラー診断コメントをプリントする。エラートレースの指定がなければ，簡単なエラー・コメントだけが与えられる。どちらの場合にも，エラーが見つかると，シミュレーションは停止する。もう一つの機能はTrace Flag である。これは，時計と逆方向に動いたとか，エイジェンダーに登録されていないイベントをキャンセルしたりするような異常な事態をシステムが発見した場合，それについての完全なコメントをプリントする。フラッグ・トレースが指定されていない場合，コメントはプリントさ

れないが，プログラムによって調べることのできる内部スイッチはセットされる。どちらの場合にもシミュレーションは続行される。3番目の機能はTrace Defineである。これは，自動的に定義される新しい結果変数の名前とタイプとを指定するコメントをプリントする。もしこのトレースが指定されていないなら，コメントはプリントされない。

第9章　ディスプレイの利用

インタラクティブにシミュレーションが行える点にオンラインシミュレーション言語の最大のメリットがある以上，その窓口となるべき端末の選択は重要である。

最近のグラフィックディスプレイのハードウェア，或はソフトウェア技術の進歩により，シミュレーション用の端末としてグラフィックディスプレイがおおいに注目されている。

グラフィックディスプレイのシミュレーションへの利用は，特に特殊目的のシミュレーションに多くの例を見る事が出来る。例えば，ハイウェーの交通量に関するシミュレーションとか，飛行機のランディングシミュレーションなどを挙げる事が出来る。

しかし残念な事に離散系の汎用シミュレーション言語で，グラフィックディスプレイを用いた例はそう多くない。もともと，ディスプレイを使う事自体，オンライン化につながる事を思えば，オンラインシミュレーション言語の実施例の少ない現在，むしろ当然の事と言える。

少ない例の中で，前に述べたGPSS／360—NORDEN を挙げる事が出来る。このシステムでは端末として，IBM2250 ディスプレイユニットを使用し，プログラムのインプットや結果の表示などをこの装置から行っている。

又，MIT に於けるOPS－3 がタイプライターを端末として使っていたのに対しOPS－4 ではグラフィックディスプレイユニットを端末として計画している。

ところで一般的に，オンラインシミュレーション言語で使う端末にはどの様な機能が必要であろうか？　いくつかの点をあげてみよう。

(1) インタラクティブなモデルビルディングを可能にする事，モデルビルディングの段階では，モデル（ソースプログラム）や，データーの修正は付きものである。この意味で，自由にテキストエディティング（text editing）が出来る事が必要である。

(2) シミュレーションランの結果を即座に把握出来る事，その為には結果を数値表示のみでなく，図形表示出来る事。

(3) シミュレーションに於ては種々の構造が扱われる，例えばネットワーク構造とかモデル自身の階層的構造等と云ったもの。これらの関係を表示して，全体的な見通しを良くする事。

(4) モデルのランニング中に時々刻々と変化する状態を表示出来る事。過度現象の把握，安定状態の検出等には不可欠なものである。

この様な要求に答えるには，グラフィックディスプレイが確かに最適な端末と言えよう。しかし現在のそのハードウェア，ソフトウェアの価格を考慮に入れると，かならずしも実用的とは言いがたい。

そこで，同じCRTディスプレイであってもキャラクター・ディスプレイがその代用にならないかということになる。

(1)の条件，つまりインタラクティブにテキスト・エディティングをおこなう機能はキャラクター・ディスプレイでも充分である。ただライトペンやタブレットを使用した便利さの点でやゝ劣るのは止むを得ない。

(2)の図形表示はいかんともしがたい。しかし，どうしても図形でなければという部分は，スピードは劣るがX－Yプロッタや，場合によってはプリンターでカバーすることも考えられる。

(3)および(4)は，使用の仕方を工夫すればあまり支障はない。棒グラフなどはキャラクター・ディスプレイでも表示可能である。

一般にオンラインあるいはタイムシェアリング・システムの端末としては，従来，タイプライタが最も普及していた。勿論，今後もこの傾向は強いとは思うが，表示のスピードや視覚的な条件はCRTディスプレイの方がはるかにすぐれている。そういった意味で，グラフィック・ディスプレイとまではゆかなくとも，オンライン・シミュレーション用の端末としてキャラクター・ディスプレイを使用するのは有効なことであろう。

第10章　付録

10.1 OPS－4のステートメント

1. 一般的なステートメント（これは，最初の2つを除けば，PL／1の標準的なシンタックスである。）

これらのステートメントは，デバッギング機能やストーレッジ・アロケーション機能と同様，一般の算術式，コントロール及びデータ定義の機能を有する。

SET（代入ステートメント）

SET－EVENT　イベントの名前

IF

GO TO

DO

BEGIN

END

PROCEDURE

DECLARE

ENTRY

RETURN

ON

SIGNAL

REVERT

ALLOCATE

FREE

2. セット処理ステートメント

これらのステートメントは，名前を付けた変数やローカル・データ・ベースをとりあつかう為の制限付きリスト処理機能を有する。

ENTER アイデンティファイア { IN / 〔AT〕 TOP / BOTTOM / BEFORE / AFTER } 〔OF〕 アイデンティファイア 〔IN〕 セットの名前

REMOVE { アイデントファイア 〔FROM〕 / TOP / BOTTOM / 〔ELEMENT〕 } 〔ELEMENT〕〔OF〕 BEFORE / AFTER アイデントファイア〔IN〕 セットの名前

CLEAR セットの名前

3. グローバル・シンボル・テーブル処理ステートメント

これらのステートメントによって，ユーザーは，OPS－4 モデルのグローバル・シンボル・テーブルを直接取り扱えるようになる。

NAME　アイデンティファイア

USE　アイデンティファイア

CREATE アイデンティファイア

CLEAR　アイデンティファイア

APPEND アイデンティファイア

VIEW

4. グローバル・データ・ベース処理ステートメント

これらのステートメントによって，ユーザーはOPS－4 モデルのグローバル・データ・ベースを直接取りあつかったり，データ・ベースをすばやくクリアしたり，スイッチしたり，モディファイしたりすることができる。

NAME－DB　アイデンティファイア

USE－DB　アイデンティファイア

CLEAR－DB　アイデンティファイア

APPEND－DB アイデンティファイア

5. ローカル・データ・ベース処理ステートメント
これらのステートメントによって，あるアクティビティが他のアクティビティのローカル・データ・ベースにアクセスできるようになる。

CONNECT　　アクティビティ

DISCONNECT　アクティビティ

6. インプット／アウトプット・ステートメント（PL／1の標準的なシンタックス）

これらのステートメントは，OPS－4 におけるI／O オペレーションに関する種々のコントロール機能を備えている。

OPEN

CLOSE

DELETE

FORMAT

GET

PUT

READ

WRITE

DISPLAY

7. データ生成ステートメント

指定された分布からサンプリングを行なうのにDRAW ステートメントが用いられる。

DRAWアイデンティファイア〔FROM〕
- NORMAL〔平均ST－DV〕
- EXPONENTIAL〔平均〕
- UNIFORM〔下限上限〕
- アレイ〔C／D〕

NEW ステートメントは新しいデータ・オブジェクトを動的に作り出し，それに名前をつける。

NEW アイデンティファイア〔NAMED アイデンティファイア〕

8. スケジューリング・ステートメント

A　外部

これらのステートメントによって，或るアクティビティをアクティブにすることができるようになる。

SCHEDULE
〔アイデンティファイアNAMED〕 アクティビティ 〔FIRST〕
{ BEFORE / AFTER / IN PLACE OF } アクティビティ
TOP 〔WITH パラメータ〕
BOTTOM 〔WITH パラメータ〕
AT 式
WHEN { イベントの名前 〔(カウント)〕 }

最後の2つのオプション「AT」と「WHEN」は，ATフェーズの後でしかもWITHフェーズの前にWHENフェーズを付け加えることによって結合される。たとえば

AT式〔AND〕WHEN{イベントの名前〔(カウント)〕}〔WITHパラメータ〕

CANCEL アクティビティ

INTERRUPT アクティビティ

RESUME アクティビティ〔FIRST〕

B 内部

これらのステートメントは，現在のアクティビティのステートに働きかける。

DELAY 式

WAIT { 〔UNTIL〕 / 〔FOR〕〔EVENT〕 } イベントの名前〔(カウント)〕}

DELAY式 〔AND〕WAIT 〔UNTIL〕 / 〔FOR〕〔EVENT〕 イベントの名前〔(カウント)〕}

INTERRUPT

CONTINUE 〔NEXT〕

RETURN 〔NEXT〕

9. トレース・ステートメント

TRACE ステートメントによって，ユーザーはシミュレーションの動きをモニタリングすることができる。

TRACE
- CALLS〔AND〕〔PARAMS〕
- TIME
- ET
- LABELS
- ACTIVITIES
- SCHEDULE ┐
- RESCHEDULE │
- CANCEL │
- INTERRUPT ├ ENTRIES
- RESUME │
- DELAY │
- WAIT │
- CONTINUE ┘
- AGENDA
- STATEMENTS
- RESULTS
- VARIABLE(S)　アイデンティファイア
- ERROR
- FLAG
- DEFINE

NO—TRACEステートメントのシンタックスはTRACE ステートメントのシンタックスと同じである。

10. 実行コントロール・ステートメント

これらのステートメントによって，ユーザーはモデルの実行を直接コントロールすることができる。

EXECUTE プロシージャ 〔FROM {ラベル / LINEナンバー}〕

〔TO{ラベル / LINEナンバー}〕〔NEXTナンバーLINES〕〔ナンバーTIMES〕

EXIT

RESUME

START プロシージャ 〔FROM {ラベル / LINEナンバー}

〔STOP{ ATタイム / AFTER カウント / WHEN {イベントの名前〔(カウント)〕} }〕

STOP { ATタイム / AFTER カウント / WHEN {イベントの名前〔(カウント)〕} }

11. 統計収集ステートメント

これらのステートメントによって，ユーザーはモデルの統計収集及び処理を行なうことができる。

ACCUMULATE〔NUMBER〕〔SUM〕〔AND〕〔SUMSQ〕〔OF〕

式 〔IN〕 変数

〔変数〕 〔AND〕〔変数〕

COMPUTE 〔MEAN 変数〕〔VAR 変数〕〔AND〕

〔ST−DV 変数〕 {FROM 変数〔変数〕〔AND〕〔変数〕 / OF ベクトル}

TABULATE 式 〔IN〕テーブル名〔重さ〕〔式〕

DISPLAY テーブル名〔FROM 式TO式〕〔CELL 式〕

PLOT〔CUM〕〔OF〕テーブル名{〔BAR〕/〔LINE〕}〔FROM式 TO式〕〔CELL 式/式〕〔HEIGHT〕〔MIN 式〕〔INTERVAL式〕〔MAX 式〕

CHANGE−PLOT { ALL / UNCONDITIONAL / CONDITIONAL / NORMAL } { UNITS 式 / LABEL 式 / DISTINCT }

12. メモリ管理ステートメント

これらのステートメントによって，ユーザーはAGENDA とセットのメモリ・アロケーションを直接管理することができる。

REORDER AGENDA

PURGE { アクティビティ / リスト } 〔連続的に〕

13. 新しい宣言アトリビュート

A プロシージャのための

1. SEQUENTIAL
2. SIMULTANEOUS
3. USER

B 変数のための

1. ACCESS プロシージャ1〔プロシージャ2〕
2. SET
3. QUEUE
4. TABLE
5. PLOT〔MIN式〕〔MAX式〕〔プロット・キャラクタ〕

14. 特殊なOPS－4 インクリメンタル・エディターEDOPS

15. 特殊なOPS－4 関数

A アクティビティと関連するシミュレーション時間を得るため

BTIME

ETIME 〔(アクティビティ)〕

LTIME

B パラメーターの指定

LATER—VALUE （変数）

CURRENT—VALUE （変数）

C セット処理（SIMULA から導入）

HEAD（セット名）

SUCCESSOR（アイデンティファイア）

PREDECESSOR（アイデンティファイア）

SAME（アイデンティファイア・アイデンティファイア）

SIMILAR（アイデンティファイア・アイデンティファイア）

FIRST（セット名）

LAST（セット名）

MEMBER（アイデンティファイア・セット名）

EXIST（アイデンティファイア）

EMPTY（セット名）

D アクティビティの参照

CURRENT

STATE（アクティビティ名）

第4編
インタラクティブ学習システム

目　次

第Ⅰ部　CLASS

第2部　ＣＡＩ言語

第1部　CLASS

第1章　システムの概要

インタラクティブ学習システム（Interactive Learning System 略してILS）は，コンピューターと人間が相互に密接な連絡をとりながら効率よく学習をおこなうシステムである。

当センターで開発しつつあるシステムCLASS（Conversational Language ASsisted System)は，FACOM 230-60 のデマンド処理で行なえる会話型言語を教育するシステムである。

このシステムの特長は次の通りである。

1) ガイダンス機能を持つ。

一般に会話型言語はバッチ処理用の言語と比べてガイダンス機能やディバック機能がすぐれていることが要求されているが，現実には，言語プロセッサーの負担がかなり大きくなり効率が下がるのを恐れて，必ずしも充分な機能を持っているとは言い難い。そこで一般にはガイダンス用の独立したシステム（例えばＨＥＬＰ）を作成するという傾向もあり，とくに初心者用の学習機能という面は，むしろ会話型言語プロセッサと切り離すべきであろう。

2) 初心者，経験者を問わず，それぞれの目的に合った学習が可能である。

一般に初心者はＣＰＬ言語の全般についての学習を希望するであろうし，経験者は言語の一部，あるいは必要に応じて適当なガイダンスを希望するであろう。そこで簡単なコマンドを用いて学習機能の適当な選択ができるようにしてある。

3) 練習問題として学習者が作ったプログラムは言語プロセッサによって直ちに実行されるのでその学習効果は著しい。

一つの学習単位の中には必ずいくつかの練習問題が含まれている。それらは学習者が実際にＴＳＳ端末でプログラムを作成し，実行してみることができるようになっている。

4) 学習端末の比較を考慮している。

学習端末として何を選ぶかによって，学習プログラムを作成する上で，その学習方式は大きく変ってしまう。当センターにある他の機種でも，同じようなシステムを作成し，学習端末による学習効果の違いについても比較検討する予定である。

第2章　システムの構成

2.1 学習端末の構成

学習端末はFACOM230-60 のＴＳＳ端末であるタイプライタとＣＲＴディスプレイ装置それからランダム・アクセス・スライドを使用する。スライドは説明や問題の提示装置である。ＣＲＴディスプレイ装置は補助提示装置と反応入力装置を兼ねている。タイプライタは提示される練習問題を実際に作成実行させる装置である。

2.2 学習対象

このシステムはＦＡＣＯＭ230-60のデマンド処理で使用できるアプリケーション・システムである。

デマンド処理で使用できる会話形言語にはＢＡＣＣＵＳとＣＰＬがあるがこのシステムは当センターで開発したＣＰＬ（Conversational PL／Ｉ）を対象に設計した。

また学習対象者は初心者（学習形式１）と経験者（学習形式２）の両方を考えている。

2.3 学習プログラムの構成

学習プログラムは図１のように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，対話が行なわれる。

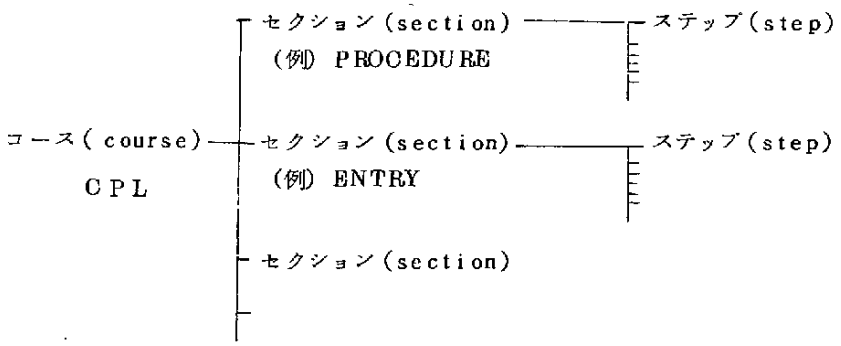


図　１　－　１

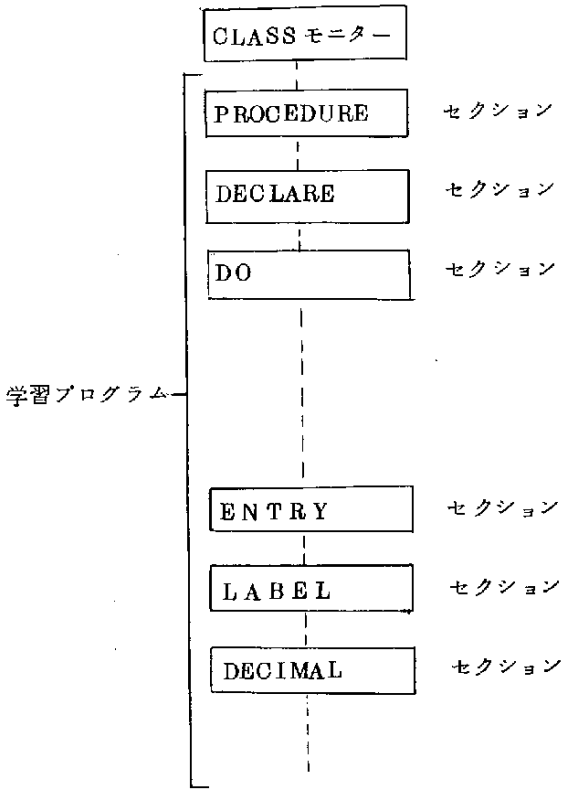

CLASS モニター
PROCEDURE
セクション
DECLARE
セクション
DO
セクション
学習プログラム
ENTRY
セクション
LABEL
セクション
DECIMAL
セクション

図　1　—　2

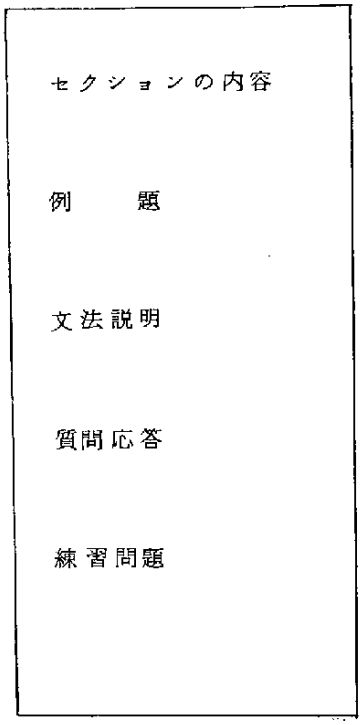

セクションの内容
例　　題
文法説明
質問応答
練習問題

図　1　—　3

第3章　学習プログラムの設計

3.1 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）等によって決まる。

3）学習者からは，ヒント要求，正答要求等を出すことができる。

4）学習経過記録は学習形式1（初心者の学習，すなわち言語全般について学習したいという人）の学習者に対してだけとられる。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。

3.2 学習方式

学習プログラムは前述のとおり，初心者でも経験者でも学習できるようにするため，学習者との対話により初心者であるか経験者であるか，または経験者ならば言語のどの部分を学習したいかを質問要求できるようになっている。

各セクションの大部分は個別指導型（tutorial instruction）を主として演習問題等はドリル演習型（drill and practice）により学習を進める。

3.3 学習プロセス

学習形式1（初心者）の学習者にとってはこのシステムは一般的ＣＡＩの学習方法と特に変るところはない。しかし学習形式2（経験者）との兼ね合いでＣＰＬ言語の文法のみでなくより効率のよいプログラムの作り方とか，ディバッグに対するガイダンス，エラーメッセージに対する処置の方法など，プログラミング上のテクニックも学習することができる。また各セクションはステートメント単位や属性単位で構成され，学習者はセクション単位で学習を打ち切ることもできる。学習形式2による学習者の場合は最初にシステムとの会話で必要なセクションが呼び出され，言語の一般形，構文規則等それぞれの要求に応じて学習することが

できる。

3.4 学習プログラム記述言語

記述言語として標準ＣＡＩ言語を利用しようと検討したが，学習端末の操作，ファイル処理等の機能的な面で不充分であることがわかった。そこで内外の既存のＣＡＩ言語を一通り調査した結果，今回は特に新しい言語系を作ることを止めて，PL／1に必要なＣＡＩ機能を付加し，プリコンパイラ及び，ＣＡＬＬタイプのサブプロシーデュアとして処理することとした。

第4章　ソフトウエアの構成

このシステムのソフトウエアはCLASSガイドと学習プログラムからなっている。

(1) CLASSガイド

① CLASSの使い方

② JP1500(TSS端末のタイプライタ)の使い方

これらは操作の名称がインデックス化されており，学習者がわからないとき必要な操作は写真や図によって，すぐ理解できるようになっている。

③ セクション名リスト

(2) 学習プログラム

実際に実行できる学習プログラムはコース作成者が各セクション単位にPL／I言語を使って書き，それをバッチのPL／Iコンパイラで翻訳して，学習プログラムファイルとして集団ディスクに登録する。

第５章　学習プログラムの実例

ＩＦセクションでの学習ステップの一部を次に示す。

[1] このセクションではＩＦステートメントの機能とか使い方について学習しましょう。

[2] あなたは日常の生活で沢山の比較・判断・選択といったことをしていますね。例えば，明日、天気がよかったら何々しようといったことですが，そうしたことをコンピュータでやらせるためにはどうしたらよいか，それをこれから考えていきます。

[3] いま，あなたがＡとＢのどちらが大きいかということを記号を使って書くとしたら，下のように書くでしょう。

Ａ＞Ｂ

[4] ＣＰＬでは，何かと何かを比較するための記号として次のような，８つの種類があります。

① ＞　より大きいことを示す。
② ¬＞　より大でないことを示す。
③ ＞＝　より大か等しいことを示す。
④ ＝　等しいことを示す。
⑤ ¬＝　等しくないことを示す。
⑥ ＜＝　より小か等しいことを示す。
⑦ ＜　より小であることを示す。
⑧ ¬＜　より小でないことを示す。

[5] 問題１

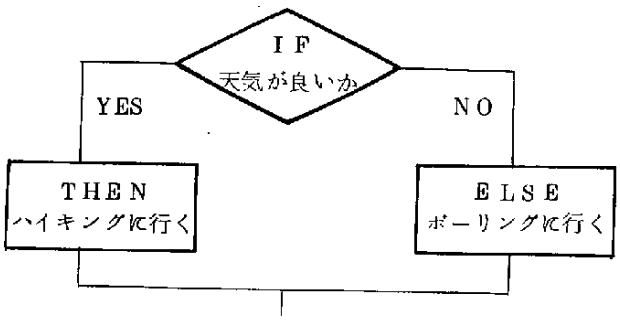
ＡがＢより大きいかを比較するために，Ａ＞Ｂと書きましたが，ＡがＢより大きいか等しいというのはＡ[　　]Ｂと書きます。　＞＝

[6] もし，天気がよければハイキングに行く。天気が悪ければボーリングに行くということをＩＦステートメントを使って書くと下のようになります。

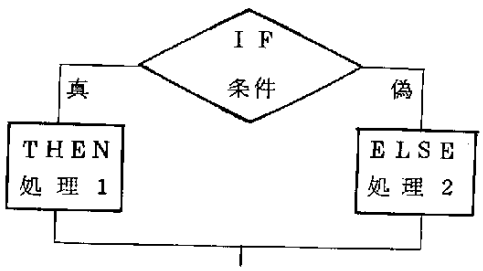
ＩＦ　天気が良ければ　ＴＨＥＮ　ハイキングに行く

ＥＬＳＥ　ボーリングに行く

7　ＩＦ　ＴＨＥＮ　ＥＬＳＥ　の関係を図にあらわしてみましょう。



8　ＩＦステートメントを使うことによりプログラムの流れを変えることができます。もう一度ＩＦ　ＴＨＥＮ　ＥＬＳＥ の関係を流れ図にあらわすと下のようになります。



9　ＩＦ　条件　ＴＨＥＮ　処理１　ＥＬＳＥ　処理２　で条件のところに書けるのはスカラ式です。次のような式とか

Ａ＋Ｂ

Ａ＊Ｂ＞0

Ａ＞0

Ｂ　＆　Ｃ

Ａ　｜　Ｂ

¬ Ｂ

また，これらを組み合せたものを書きます。

10 ＩＦステートメントが実行されると，まず，このスカラ式の演算が行なわれ，次にその結果の値がビットストリングに変換されるのです。ビットストリングとは０と１の値をもつビット列でしたね。いくつか例をあげてみましょう。

０

１

０００

０１０

１０１１

11 問題２

ＩＦステートメントの条件のところに書けるのは　　式であり，最初にその演算が行なわれ，その結果がビットストリングに変換されます。ビットストリングとはある長さをもつ　　か　　の値です。

スカラ

０

１

12 ＩＦステートメントの条件とは，さきほどのビットストリングの値が全て０か，１の値が１つでもあるかによって，この条件は偽であるか真であるかが決まるのです。

０００……偽　　０１００……真

13 問題３

条件はビットストリングの値のどれか［　］なら真（ＹＥＳ）全て［　］なら偽（ＮＯ）となります。

１

０

14 さて，次に条件によるプログラムの流れの分岐について考えてみましょう。

ＩＦステートメントが次の形式なら

ＩＦ　条件　ＴＨＥＮ　ステートメントｍ

ＥＬＳＥ　ステートメントｎ

ステートメントｐ

プログラムの流れの分岐は条件が真ならステートメントｍが実行され，次にステートメントｐが実行されます。また条件が偽ならステートメントｎが実行され，次にステートメントｐが実行されるのです。

15 ＩＦステートメントの各部分を次のように呼ぶことにします。

ＩＦ　条　件　　ＴＨＥＮ　ステートメントｍ
（条件式と呼ぶ）　　（ＴＨＥＮ節と呼ぶ）

ＥＬＳＥ　ステートメントｎ
（ＥＬＳＥ節と呼ぶ）

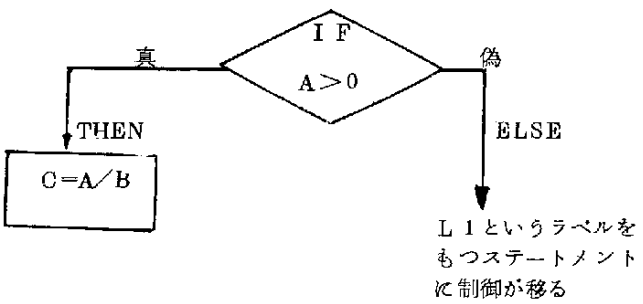
16 ＣＰＬではステートメントの最後に必ず"；"を書くことになっていますが，ＩＦステートメントの場合は次のように決まっているのです。

条件式には　；　はいらない。

ＴＨＥＮ節とＥＬＳＥ節には　；　がいります。

17 ＩＦステートメントやＧＯＴＯステートメントのようにプログラムの実行順序を変えることのできるステートメントを，コントロール・ステートメントといいますが，ＴＨＥＮ節やＥＬＳＥ節にも，そういったコントロール・ステートメントを書くこともできるのです。

18 ＩＦ　Ａ＞０　ＴＨＥＮ　Ｃ＝Ａ／Ｂ；
ＥＬＳＥ　ＧＯＴＯ　Ｌ１；
このステートメントの実行は次のようになります。



19 問題4

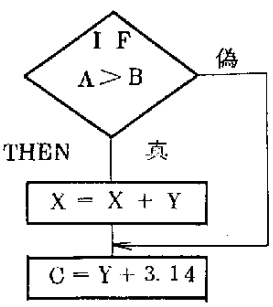
```
A＝2
ＩＦ　Ｐ＝０　ＴＨＥＮ　ＷＯＲＫ＝Ａ＊＊２；
ＥＬＳＥ　ＷＯＲＫ＝Ａ＊＊４；
```

Ｐに5が代入されている状態で，このプログラムを実行するとＷＯＲＫに代入される値は　　になります。

20 いままで，必ずＩＦ　ＴＨＥＮ　ＥＬＳＥ　という書き方をしてきましたが，ＩＦステートメントの使い方として必ずしもＥＬＳＥ節を必要としないのです。

```
ＩＦ　Ａ＞Ｂ　ＴＨＥＮ　Ｘ＝Ｘ＋Ｙ；
Ｃ＝Ｙ＋3.14；
```

このプログラムの場合は条件が真ならＴＨＥＮ節（Ｘ＝Ｘ＋Ｙ）が実行され次にＩＦステートメントの次のステートメント（Ｃ＝Ｙ＋3.14　）を実行します。条件が偽のときにはＴＨＥＮ節を実行せず直接次のステートメントを実行します。これを図にあらわすと次のようになります。



21 問題5

ＩＦステートメントの使い方でＥＬＳＥ節を書かないときの制御の流れは，条件が［　］ならＴＨＥＮ節を実行してから次のステートメントへ，［　］のときには直接次のステートメントに制御が移る。

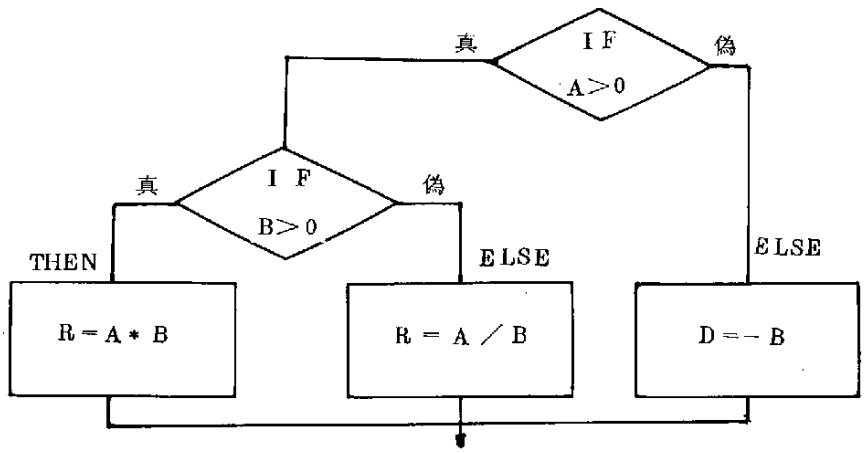
真
偽

22 ＩＦステートメントのＴＨＥＮ節やＥＬＳＥ節にコントロール・ステートメントが使用できることを習いましたが，それを利用してもう少し複雑な使い方ができます。

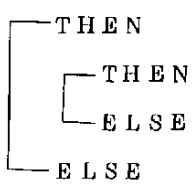
23
```
IF  A>0  THEN  IF  B>0  THEN
R=A*B;  ELSE  R=A/B;
ELSE  D=-B;
```

このようにＴＨＥＮ節やＥＬＳＥ節にまたＩＦステートメントがくるような場合をＩＦステートメントの入れ子といいます。

24 前のステップの制御の流れは次のようになります。



このような場合のＴＨＥＮとＥＬＳＥの対応は最初にあらわれたＥＬＳＥがすぐ直前のＴＨＥＮに対応します。そして後のＥＬＳＥが前のＴＨＥＮに対応します。そのことを図にあらわすと次のようになります。



25 問題 6

ＩＦステートメントのＴＨＥＮ節やＥＬＳＥ節にまたＩＦステートメントを書くことをＩＦステートメントの［　　］といいます。

入れ子

26 これでＩＦセクションは終りです。ここで少し休憩してからまた頑張って次のセクションに進みましょう。

第6章　今後の課題

今年度の目標として，ＣＡＩに関する内外の文献調査，学習端末装置として何を選ぶべきか，また学習プログラムの事例として何を選ぶべきか等を議論検討してきた。

1）学習端末装置

学習端末は教材や問題を提示する機能と，学習者が解答やその意志を計算機に知らせる機能をもっている。いずれにしても学習効果をあげるためにはまず日本人向きであることが必要である。提示装置としてはスライド，ＣＲＴディスプレイ，タイプライタが現在の代表的なものである。インタラクティブな面を考えればＣＲＴディスプレイが選ばれるであろうが，漢字の表示が困難であるので，ＣＲＴディスプレイとスライドが重畳したような提示装置が望まれる。また反応入力装置についていえば，キーボード，キーセット，ライトペン等が代表的なものである。これは提示装置との関係もあり，それぞれ一長一短があるが，日本人はキーを打つことになれていないので，キーボードなら鍵盤配列を特に考慮する必要がある。複数教科の学習ができるようなシステムではキーセットはキーマットにより各鍵盤にいろいろの意味をもたせうるので大変便利である。このＩＬＳシステムではいくつかの代表的提示装置を使って，それぞれ学習プログラムを作成し，学習端末による学習効果の比較を46年度に予定している。

2）学習プログラム

ＣＡＩシステムのポイントは学習プログラムの作成技術であるといっても過言ではあるまい。学習プログラムの作成はかなりの労力と教育的能力を必要とする。特にコンピュータ言語を教えるというような人間の思考的行動の学習は，後述したカメラの操作の学習プログラムのように目に見える行動に比較して，その行動を分析するむずかしさがある。

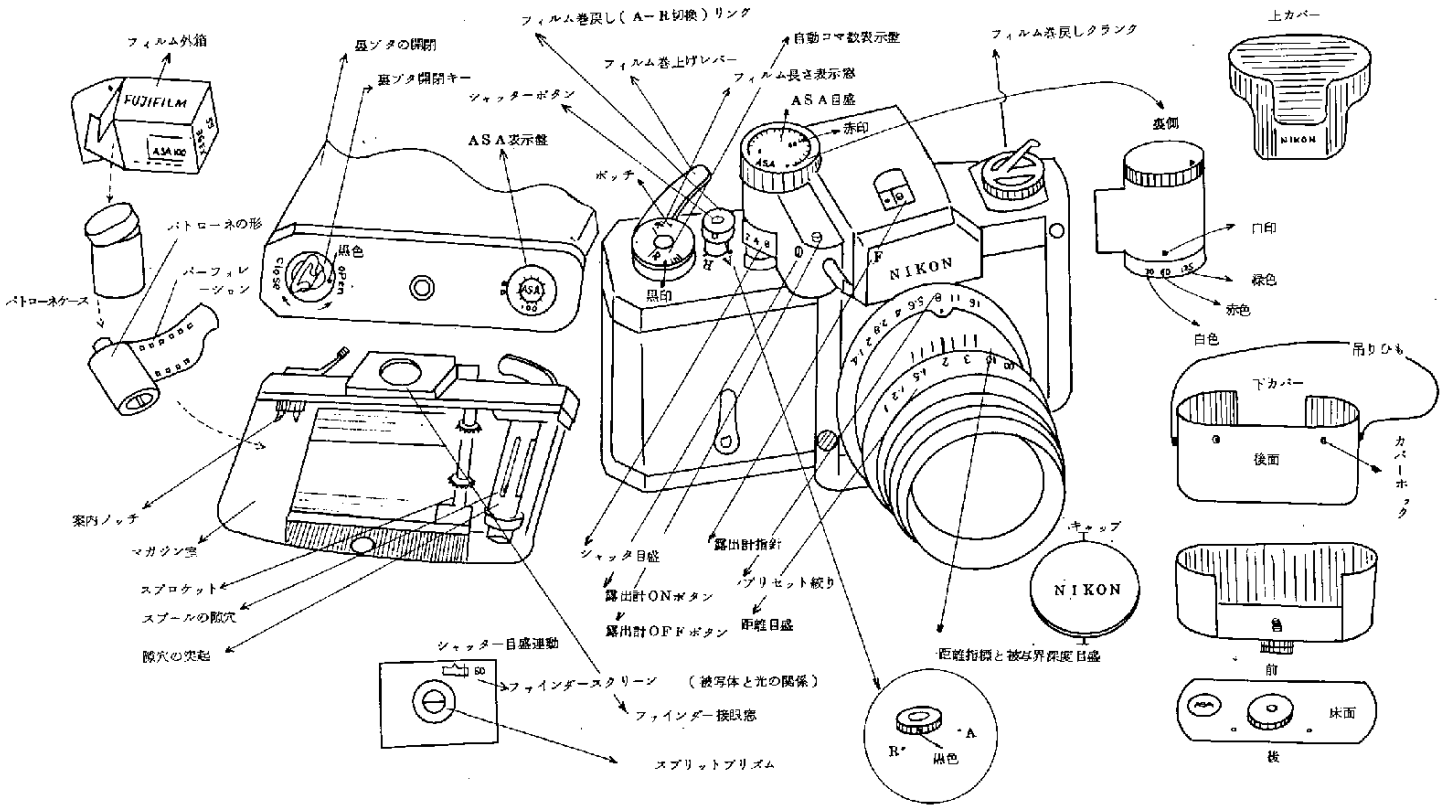

フィルム外箱
FUJIFILM
パトローネの形
パトローネケース
パーフォレーション
裏ブタの開閉
裏ブタ開閉キー
ASA表示盤
黒色
案内ノッチ
マガジン室
スプロケット
スプールの軸穴
軸穴の突起
フィルム巻戻し（A－R切換）リング
フィルム巻上げレバー
シャッターボタン
ポッチ
自動コマ数表示窓
フィルム長さ表示窓
ASA目盛
赤印
黒印
フィルム巻戻しクランク
NIKON
シャッター目盛
露出計ON・OFFボタン
露出計指針
プリセット絞り
距離目盛
シャッター目盛連動
ファインダースクリーン
（被写体と光の関係）
ファインダー接眼窓
スプリットプリズム
距離指標と被写界深度目盛
キャップ
R
A
黒色
上カバー
裏側
口印
緑色
赤色
白色
吊りひも
下カバー
後面
カバーホック
前
底面
後

図 7－2

第7章　プログラム学習テキストの実例

学習プログラムの作成には，特別の技術が必要であり，そのための教育あるいは人材の養成が要求されている。

(財)能力開発工学センターではそうした社会の要請に対して，独自の学習システム設計者養成講座を開催している。当センターでもこれを受講し，その中の実習において1つのプログラム学習テキストを作ったので，その設計からトライアウトまでの過程を順を追って説明することにする。

7.1 概　　要

学習プログラムの設計は図7－1の通りに目標行動の設定からトライアウトまで順次作業を進めるようになっている。

学習対象はニコンFカメラの操作である。

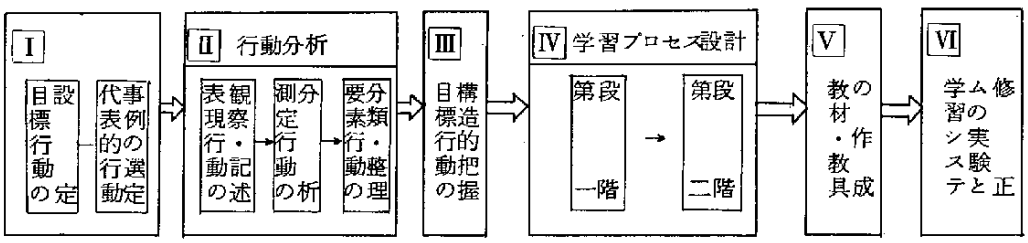


図7－1　（プログラミングのプロセスチャート）

7.2 目標行動の設定と事例

1）設　　定

① カメラ操作に関して初心者

② カメラは机の上に準備されている。

③ 中卒程度の知識で理解できるもの。

2）事　　例

① 静止物の撮影

② 距離……　5m前後の人物と40m前後の風景

③ 明るさ……　明るい場合と暗い場合

3）ニコンFカメラの部品名称と構造

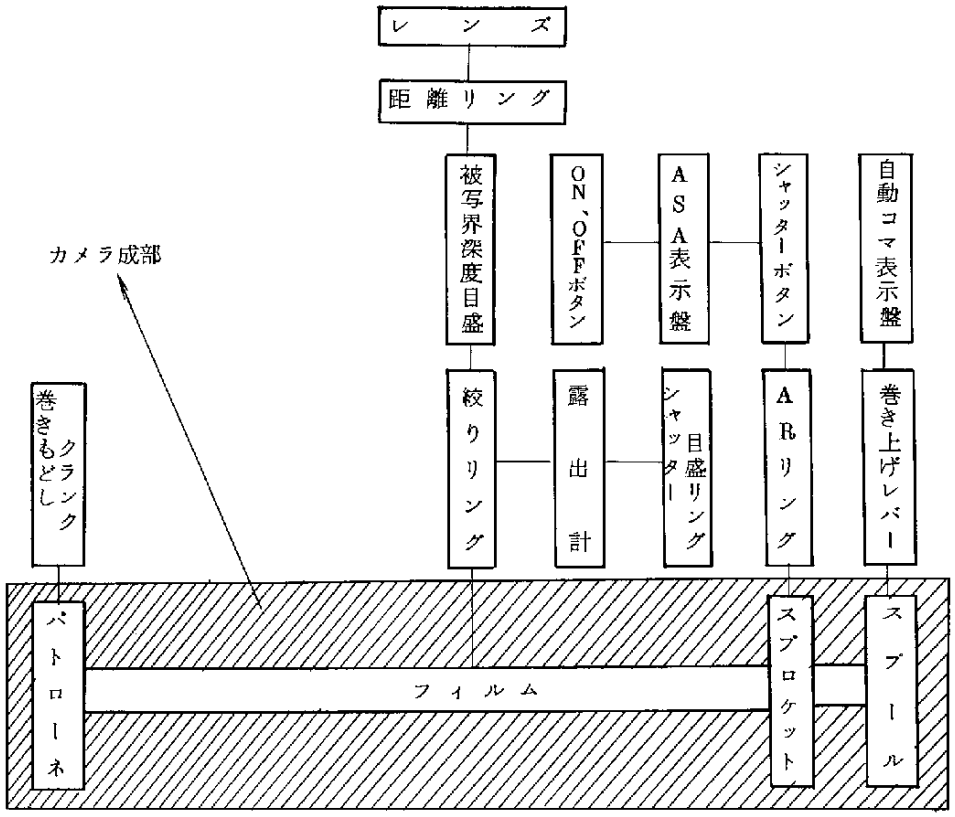


図7－3　　ニコンFの構造

7.3 行動分析

人間が対象に向って行動するとき頭の中で何を判断（測定行動）し，その結果どういう行動となって表われた（表現行動）かを観察して分析することを行動分析という。

以下は実際にこのカメラが操作できる人をモデルとしてその行動を観察分析したものである。

表現行動	測定行動
1　カメラとフィルムを自分の前におく	1　カメラとフィルムの有無

2	上カバーのホックをはずして右側におく。 ・後に2つある ・下に2つある	1 2 3	上カバーの位置，形状，機能 ・ホック式であること ・ホックの位置 ・ホックの数 ホックのはずれた音 上カバーを置く位置
3	カメラを横にしてレンズを下に向ける。そして下カバーのネジを回してはずす。 ・半時計方向 ・瞬間的に力を入れる ・右側におく	1 2 3 4 5	カメラの位置，向き ・たおれないように 下カバーの位置，形状，機能 ・ネジ止めであること ネジを回す方向 ・力の入れ方 手ごたえ 下カバーを置く位置
4	カメラをもとの位置にしてARリングの●印をRに合わせる。	1 2 3 4	カメラの位置 ARリングの位置，形状，機能 ・小さい丸いつまみ ・黒い点 ・AとRの指示 つまみを回す方向 手ごたえ
5	カメラをレンズを下に向けて裏ぶたのキーを立ててOPENに合せる。 ・半時計方向 ・キーの▼を矢印の方向に	1 2 3 4 5	カメラの位置と向き置き方 裏ぶたの位置，形状 裏ぶた開閉キーの位置，形状，機能 ・▼印 ・OPEN，CLOSEの指示 キーのたて方，回す方向 手ごたえ
6	裏ぶたを引きぬく ・右方向に ・横におく	1 2 3	裏ぶたの位置，形状，機能 ・水平，右に引く 手ごたえ 裏ぶたの置く位置
7	フィルム外箱をあけパトローネケースを出しパトローネを取り出す。	1	フィルム箱の位置，形状 ・箱の指示（ASA，枚数，色）

		2 パトローネケースの形状
		3 パトローネの位置，形状
		4 フィルム箱とパトローネケースの置く位置
	パトローネの凹の方を案内ノッチに合わせてマガジン室に入れる。	1 マガジン室の位置，形状，機能 ・案内ノッチ ・くぼみ 2 パトローネの入れ方 ・凹部を案内ノッチ ・フィルムの引出し口 3 おさまりぐあい ・手ごたえ ・音 ・見た感じ
8		
9	フィルムを引き出す ・10cm位	1 フィルムの長さ
10	スプールのすきまにフィルムの先端をさしこむ。 ・すき穴の突起にパーフォレーションを合せる。	1 スプールの位置，形状，機能 ・すきま 2 フィルムのさしこみ方 ・突起 ・長さ
11	フィルムのパーフォレーションをスプロケットに合わせスプールを回して巻き込む。 ・フィルムのたるみがなくなるまで ・左手でフィルムを押える ・右手でスプールを半時計方向に回す	1 スプロケットの位置，形状，機能 ・スプロケットの爪 2 フィルムのパーフォレーションとスプロケットの爪との一致 3 スプールの回す方向 ・半時計方向 4 フィルムの両側のパーフォレーションとスプロケットの爪との一致
12	裏ぶたをはめる ・本体のみぞに合わせて ・1cm位すきまを残して上から押さえる ・水平におしこむ	1 裏ぶたの位置，形状，機能 ・フィルム圧着板 2 カメラ本体のみぞの位置，形状 ・圧着板の位置 3 裏ぶたをみぞに置く位置 ・1cm

		4	裏ぶたとみぞが一致したか
		5	力を入れる方向
			・静かに
		6	手ごたえ
	裏ぶた開閉キーをCLOSEに合わせる	1	開閉キーの位置，形状，機能
	・キーの　を矢印の方向に		・OPEN, CLOSEの指示
	・時計方向に		・▼部
13	・キーのつまみをＯＰＥＮの方にたおす	2	開閉キーのまわす方向
			・時計方向
		3	手ごたえ
			・音
	下カバーをはめる	1	下カバーの位置，形状，機能
	・ネジをしめる		・方向
14	・時計方向に	2	ネジを回す方向
		3	手ごたえ
15	吊ひもを首にかけ左手にカメラをもつ	1	吊ひもの位置，形状，機能
	・下からささえる	2	カメラをもつ位置
	レンズキャップのボッチを親指と人さし指	1	レンズキャップの位置，形状，機能
	で押してはずす	2	手ごたえ
16	・押えてひっぱる	3	キャップを置く位置
	・レンズをきずつけないように静かに		
	ＡＳＡ目盛りを100にセットする	1	フィルム箱の位置，形状
	・親指と人さし指，中指でつかみ力いっ		・ＡＳＡ表示
	ぱい上にもち上げる	2	ＡＳＡ目盛りの位置，形状，機能
	・赤　を100に	3	ＡＳＡ目盛りのまわす方向
17			・つかむ位置
			・ひっぱる方向
			・力
		4	目盛りと赤　印との一致
	ＡＲリングをＡに合わせる	1	ＡＲリングの位置，形状
18		2	ＡＲリングのまわす方向
		3	黒●印とＡとの一致
	フィルム長さ表示窓の突起を爪でひっかけ	1	フィルム箱の位置，形状
19	て36の文字が出る方向に回す		・表示枚数

		2 フィルム長さ表示窓の位置，形状，機能
		3 まわす方向
		・突起
		4 表示枚数と表示の一致
	フィルム巻き上げレバーを右にいっぱい回して放す。	1 フィルム巻き上げレバーの位置，形状，機能
20		2 まわす方向
		・右に
		・力
		3 巻き上げの手ごたえ
	シャッターボタンの赤●が手前になっているかたしかめて，もしそうなっていなければ巻き上げレバーを少し回し赤●が手前になるようにして放す	1 シャッターボタンの位置，形状，機能
		・赤●の位置
21		2 巻き上げレバーの位置
		3 巻き上げレバーのまわす方向とシャッターボタンの赤●印の位置
	シャッターを押す	1 シャッターボタンの位置，形状，機能
		2 シャッターを押す方向
22		・力
		3 シャッターの切れた音
	自動コマ表示盤の1が　印のところにきているかたしかめる。	1 自動コマ表示盤の位置，形状，機能
		2 巻き上げレバーの位置，形状，機能
23	・巻き上げレバーを右に回す	3 レバーのまわす方向
		4 印と表示文字の一致
	露出計ONボタンを押す	1 ONボタンの位置，形状，機能
		2 手ごたえ
24		・OFFボタンが出る
	カメラをレンズの下にしてもって，シャッター目盛盤を60にセットする	1 カメラの位置，向き
		2 シャッター目盛盤の位置，形状，機能
25	・60が白●印になるようASA目盛をまわす	3 まわす方向
		4 目盛盤の60と白●印との一致
	・もち上げないで	
	カメラを被写体に向ける。上から露出計を見ながら針が真中にくるまでプリセット絞りを回して合せる。	1 被写体の方向
		・カメラの向き
26		2 露出計の位置，形状，機能

	・針を右から左にするときは絞りを目盛の小さい方に回す ・針を左から右にするときは絞りを目盛の大きい方に回す。	3 4 5 6	絞りリングの位置，形状，機能 絞りリングのまわす方向 絞りリングのまわす方向と矢印の動き 針と◎印との一致
27	左手でカメラを下から支さえる。 両腕を体につける 両足を開く ファインダーに目をあてる	1 2 3 4 5 6	カメラをもつ位置 カメラの固定をたしかめる 両足の位置 両腕の位置 ファインダーの位置，形状，機能 カメラを固定する位置
28	ファインダースクリーン内の露出計の針の位置が真中かどうかたしかめ調整する。	1 2 3 4 5	ファインダースクリーン内のシャッター速度 ファインダースクリーン内の露出計の位置 絞りリングの位置 リングを回す方向とファインダースクリーン内の露出計の針の一致 露出計の針が中央にあるか
29	距離目盛りを見る ・左手でつかむ ・カメラを下げる	1 2	カメラの位置 距離目盛リングの位置，形状，機能
30	ファインダースクリーン内のスプリットプリズムの上下の像を距離リングを回して調整する。 ・左手で ・正しい姿勢	1 2 3	ファインダーの位置・形状・機能 スプリットプリズム内の像の上下位置 距離リングのまわす方向と像の上下の一致
31	右手の人さし指でシャッターを静かに押す	1 2 3	シャッターの位置，形状，機能 シャッターボタンの赤●印が正確に手前にあるか シャッターの切れた音 ・手ごたえ
32	フィルム巻き上げレバーを右にいっぱい回して放す	1 2 3	フィルム巻き上げレバーの位置，形状，機能 巻取りの確認 自動コマ数表示盤のコマ数が1つ送られたか確認

33	全部とり終ったらARリングをRに合せる	1 ARリングの位置，形状，機能 2 Rの文字の位置 3 ARリングの黒●印の位置 4 ARリングの黒●印とRの文字の位置と一致したか
34	レンズのふたを閉める	1 レンズのふたのある位置 2 カメラのレンズの位置 3 正しくセットされたか
35	露出計OFFボタンを押す	1 露出計OFFボタンの位置，形状，機能 2 ボタンを押したときの音
36	巻き戻しクランクのつまみをおこして矢印の方向に指で回す ・手ごたえがある ・クランクをもとの位置に	1 フィルム巻戻しクランクの位置・形状・機能 2 クランクの起し方 ・矢印の方向 3 全部巻き終った手ごたえの感じ
37	首から吊バンドをはずし机の上におく	1 机のカメラを置く位置 2 吊ひもを手にもつ位置 3 カメラの重さ 4 カメラを机の上に置くときの向き
38	カメラを横にし，レンズを下に向ける。そして下カバーのねじを回してはずす ・半時計方向 ・瞬間的に力を入れて ・右側におく	1 カメラの位置，向き 2 レンズを向ける方向 3 カメラを向ける方向 4 下カバーのネジの位置，形状，機能 5 ネジを回す方向 6 瞬間的な力 7 ネジが本体からはずれたかどうか 8 カバーを置く机の位置
39	裏ぶたをはずす ・キーをOPENに合せる ・半時計方向 ・キーの　印を矢印の方向に	1 裏ぶた開閉キーの位置，形状，機能 2 OPENという文字の位置 3 キーの　印の位置 4 手ごたえ 5 裏ぶたを引きぬく方向 6 裏ぶたを置く机の位置

40	パトローネをマガジン室からはずす	1	マガジン室の位置
		2	パトローネを引きぬく方向
41	パトローネをパトローネケースに入れる ・凸部を下に入れる ・ふたを閉める	1	パトローネの凸部の位置
		2	パトローネケースの位置，形状
		3	パトローネケースのふたの位置，形状
42	裏ぶたをはめる	1	裏ぶたのある位置
		2	裏ぶたの向き
		3	本体の裏ぶたをはめるみぞの位置
		4	裏ぶたを押しこむ方向
43	裏ぶた開閉キーをCLOSEにする	1	開閉キーの位置，形状，機能
		2	キーのまわし方，倒し方
44	下カバーをつける	1	下カバーのある位置
		2	下カバーの向き
		3	下カバーのネジの位置，形状，機能
		4	ネジを回す方向
		5	正しくしめたか
45	上カバーをする	1	上カバーのある位置，形状
		2	上カバーの向き
		3	カメラ本体の位置
		4	下カバーのホックのある位置
		5	ホックをはめたときの音

7.4 目標行動の構造的把握

行動分析の結果，人間が対象に向かって一連の行動をしていることがわかる。では目標となる行動がいかなる構造をもっているのか分析することを行動の構造把握という。

図7－4は7.3の行動分析の結果を構造として表わしたものである。

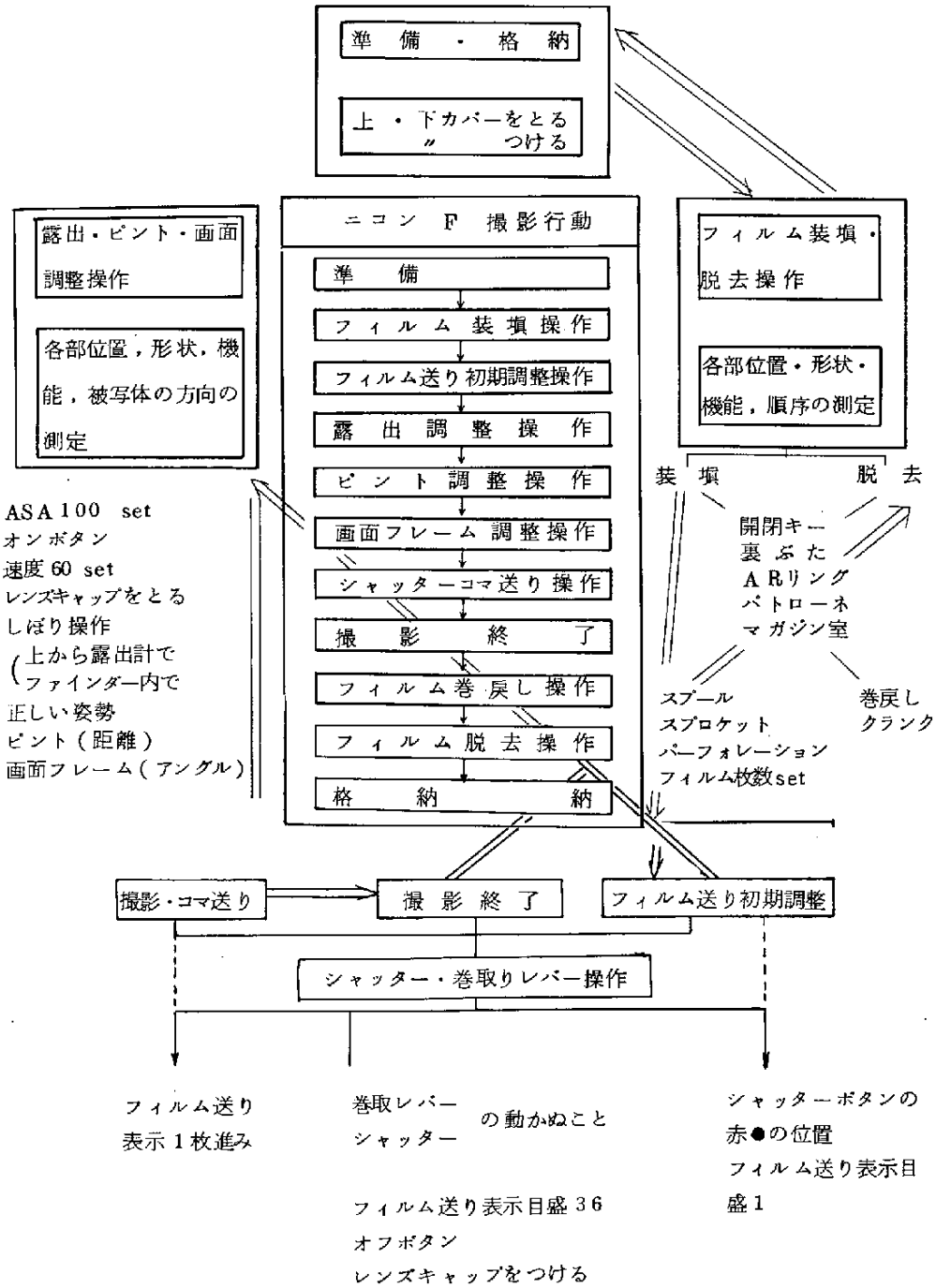


図7－4. ニコン F撮影操作行動構造図

7.5 学習プロセスの設計

ここでは行動の構造に順序付けを行ない，目標行動はどのようなまとまり（単位行動）で形成されているか，学習ステップはどのような要素で構成するかを決定する。

図7－5は単位行動を，図7－6は単位行動の順序付けを示している。

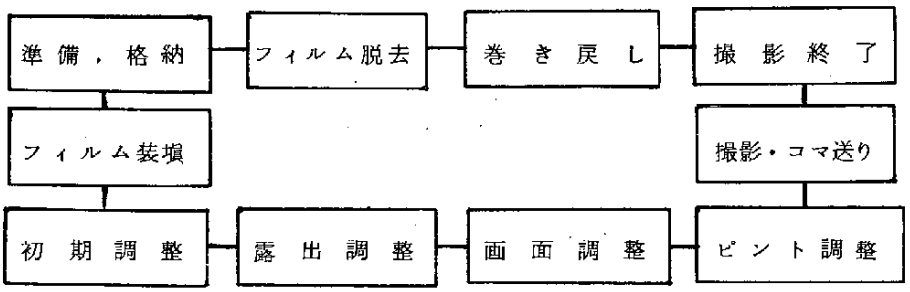


図7－5. 単位行動図

1. 準備
2. 初期調整その1
3. 露出調整
4. 画面調整
5. ピント調整
6. 撮影，コマ送り
7. 下カバーをとる
8. 裏ぶたをとり内部機構
9. フィルム装塡
10. 初期調整その2
11. 撮影
12. 36枚撮影完了操作
13. フィルム脱去

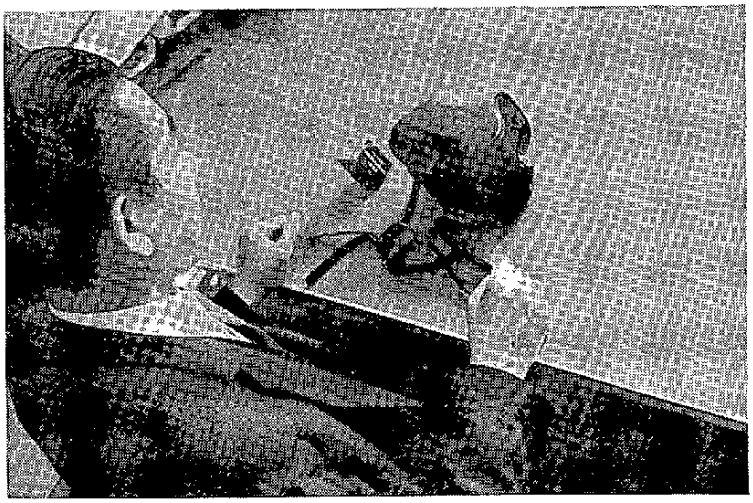
図7－6. 単位行動の順序付け

7.6 学習ステップの作成

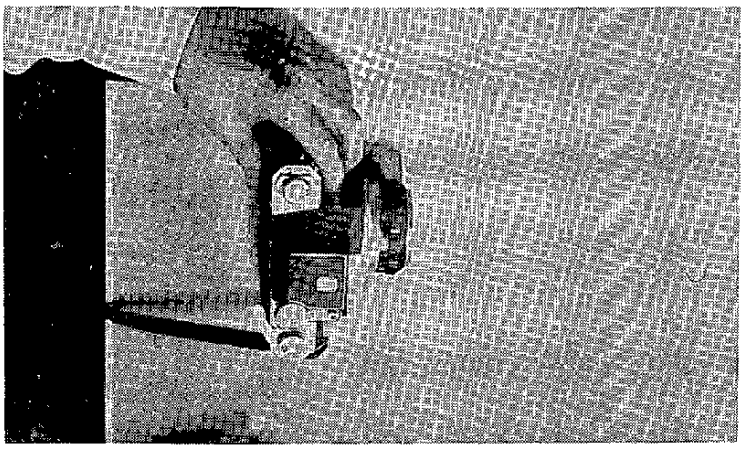
以下に示すのは出来上がった単位行動の順序に従い，教材，教具に合せて作成した学習ステップである。

1　これから，カメラ（ニコンF）の操作のしかたを学びましょう。机の上にカメラとフィルムがあるでしょう。

2　まず，カメラの上カバーをはずしなさい。上カバーはうしろに2ケ，下に2ケ，合計4ケのホックでとめてあります。



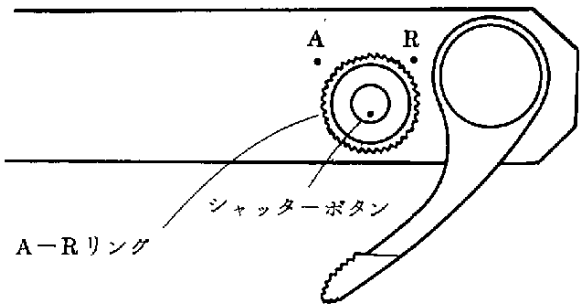
3　上カバーをはずしたら，カメラを机の上におきなさい。



4　カメラ上部の右から2番目のボタンを見なさい。

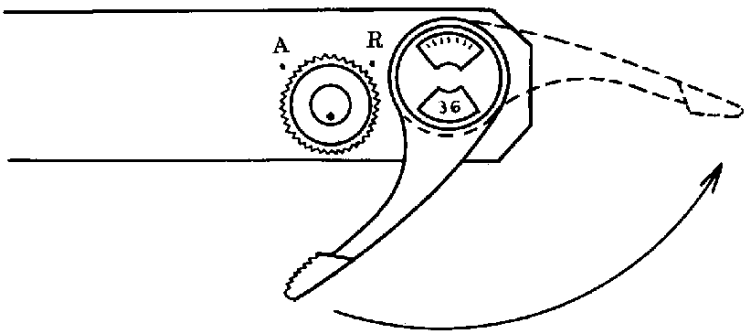
内側にある，赤い・印のついたボタンがシャッターボタンです。その外側に黒い・印のついたリング（環）があり，リングの外側にAとRの表示があります。これをA－Rリングと呼びます。

A－Rリングの黒・をAに合わせなさい。



5　次に，巻上レバーを動かしてみましょう。

A－Rリングの右側にあるレバーを右手の親指で右の方に動かしなさい。図の点線の位置になつたところで，レバーがそれ以上まわらなくなったら，親指を離しなさい。レバーは自動的に元へ戻ります。



6　右手人さし指で，シャッターボタンを静かに押してみましょう。

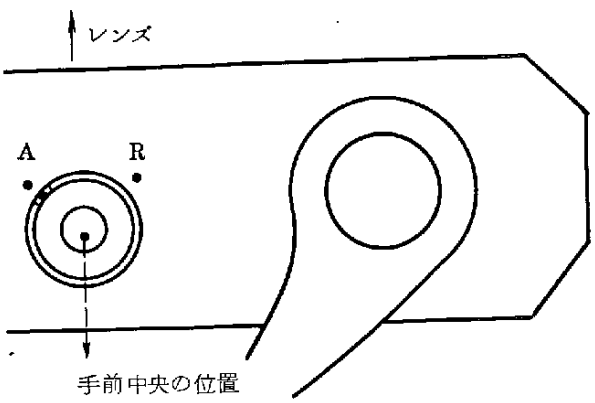
7　シャッターはうまく動きましたか？

シャッターがうまく動いた場合はステップ８に，動かなかった場合はステップ１３に進みなさい。

8　この場合，シャッターが動いたのは，いわば偶然であると言えます。必ずしも，いつもこのようにうまくいくとは限りませんから，少し練習をしておきましょう。

9　シャッターボタンの赤・の位置に注意しなさい。

下図のように，手前中央の位置になければ，シャッターはおりません。



10　巻上レバーを少しずつ動かしながら，シャッターボタンの赤・が手前中央の位置に来るように調整しなさい。ほんのわずか巻きすぎても，シャッターは動かず，またやり直さなくてはなりませんから，レバーはゆっくりと慎重に操作して下さい。

11　これで，シャッターがセットされた状態になっているのです。
シャッターを押しなさい。シャッターの閉じる音と軽い手ごたえがあります。

12　もう1度，巻上レバーをまわし，シャッターを押しなさい。

13　A－Rリングを Rにセットして，巻上レバーをまわしなさい。

14　さきほどとちがって，レバーが空まわりしたような感じがしましたね。
念のためシャッターを押してみなさい。
これも動きませんね。

15　それでは，もう1度A－RリングをAにセットして，巻上レバーを右にいっぱいにまわしシャッターを押してみなさい。

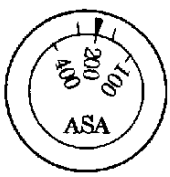
16　シャッターは，やはり動きませんね。
シャッターボタンの赤・が手前中央の位置に来るように巻上レバーを調整して，シャッターを押してみなさい。
ほんのわずかな狂いでも，シャッターは動きませんから，ゆっくりと慎重に操作しなさい。

17　うまくいきましたか？
これで，どうやらA－Rリング，巻上レバー，シャッターの間には，何か関係があるらしいということが分りましたね。

18　次にＡＳＡ目盛の合せ方について練習しましょう。
写真のように，親指，人さし指，中指の3指を使ってＡＳＡ目盛のツマミをつまんで，力一杯もち上げ，赤・印を図のように２００のところに合わせなさい。この時ＡＳＡ目盛の軸

も一緒に回ってしまうことがありますが別にさしつかえありません。

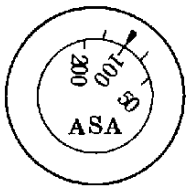
ＡＳＡ目盛



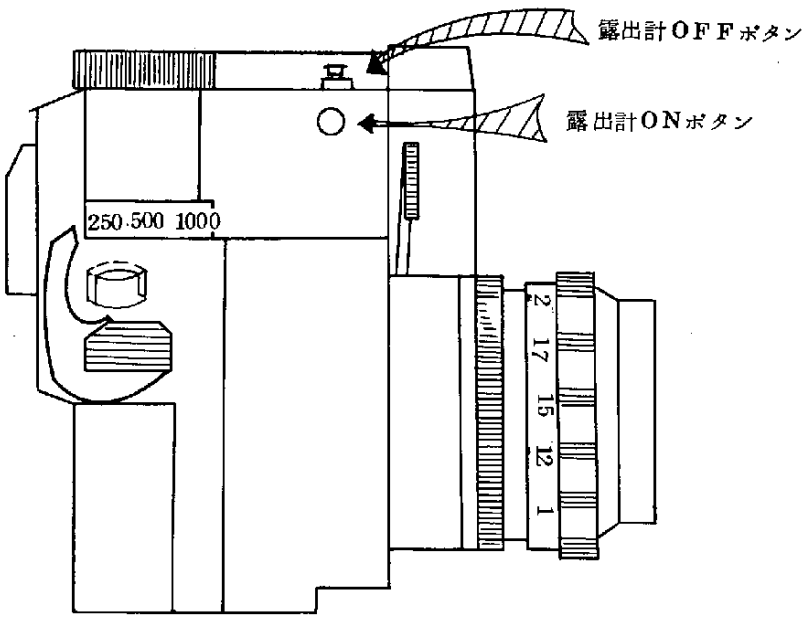
19　ＡＳＡ目盛はフィルム感度を表わしているのです。

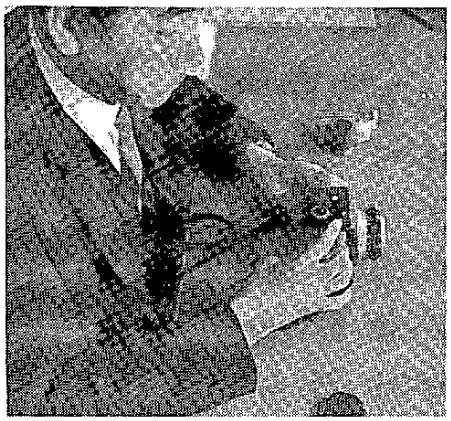
いま練習した要領で，こんどは赤＇印を１００に合せなさい。

ＡＳＡ目盛

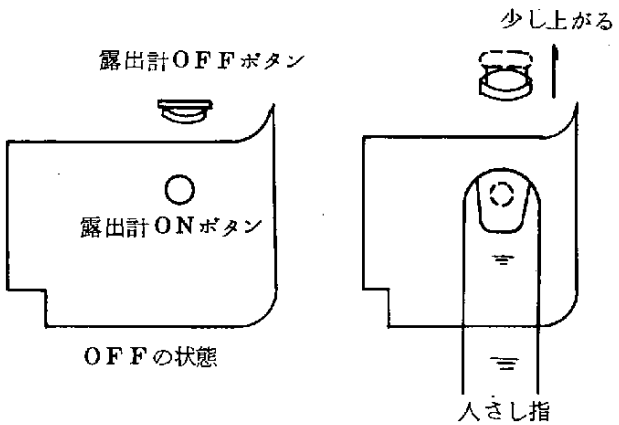


20　露出計にはＯＮボタンとＯＦＦボタンがあります。図を見ながらそれを確めなさい。

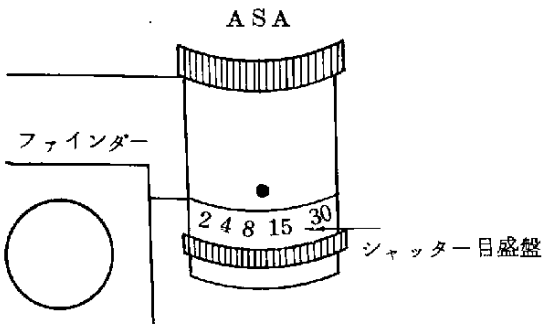


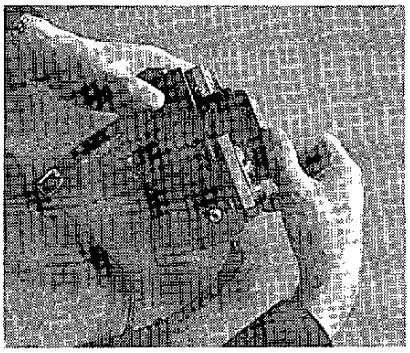


21　露出計のＯＮボタンをＯＮにすることにより，このカメラの露出機構に電源が入ります。ではその露出計のＯＮボタンを図のように人さし指で押しなさい。その時音がして露出計のＯＦＦボタンが図のように少し上がります。



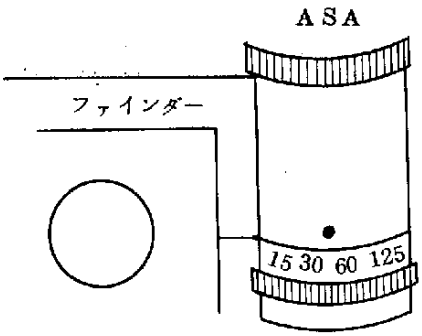
22　写真のようにレンズを下に向けてカメラをもちなさい。先ほどのＡＳＡ目盛ツマミを親指と人さし指でつまみ，今度はもち上げないでそのまま左右に回して図のシャッター目盛盤の数字が変ることをたしかめなさい。





23　シャッター目盛盤の数字を図の・印に合せることにより，その時のシャッター速度が決ります。

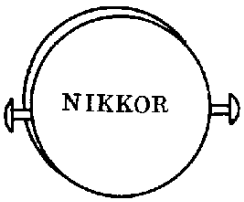
では，前の要領でシャッター目盛盤を６０に合せなさい。この６０が標準シャッター速度で特に赤数字になっています。

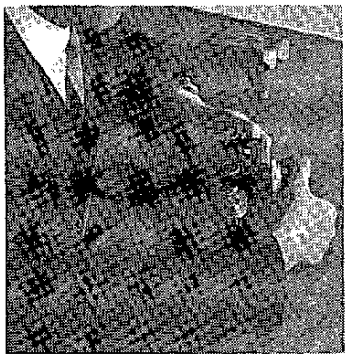


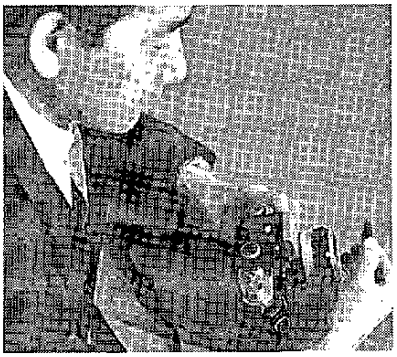
24　レンズキャップを次の要領に従ってはずしなさい。図のようにキャップの両側にボタンがあります。このボタンを右手親指と人さし指で押して静かにはずすのです。

①キャップボタンを指で押したところ　　②キャップをはずしたところ

レンズキャップ

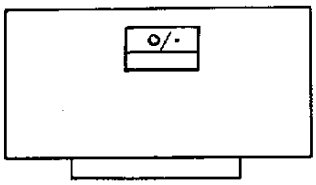


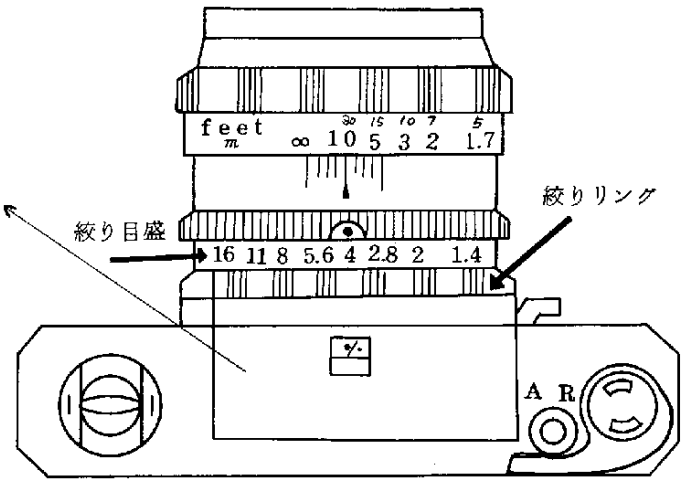




25　図を見ながら，露出計と絞りリングの位置をたしかめなさい。次にその絞りリングを右手でもって左右に回してごらんなさい。露出計の針がふれるはずです。

①露出計の位置　　②絞りリングの位置

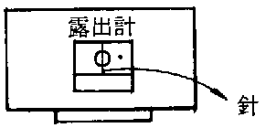




26　カメラを被写体に向けなさい。そして上から露出計を見ながら右手で絞りリングを回して露出計の針をほぼ中央に合せなさい。

①針を右から左に動かすときには絞りリングの数字を小さい方に合せる。

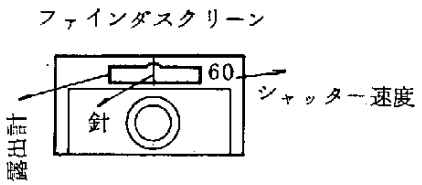
②針を左から右に動かすときには絞りリングの数字を大きい方に合せる。

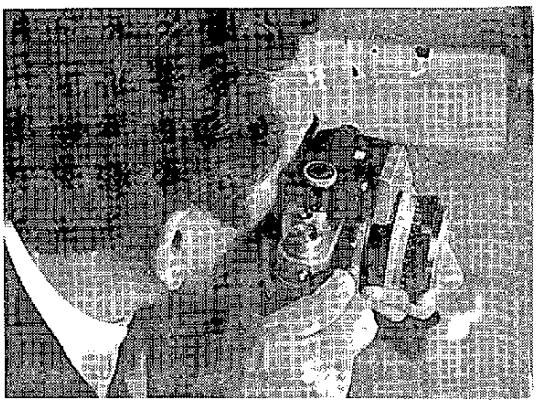


ファインダー

27　カメラを写真のように両手でもち，ファインダーに目をあて，ファインダースクリーンをのぞいてスクリーン上部の露出計とその右側のシャッター速度の数字が６０になっていることをたしかめなさい。次にその露出計の針が中央になかったら，ファインダースクリーンをのぞいた姿勢で絞りリングを回し，針がほぼ中央にくるように合せなさい。

ファインダースクリーン





28　もし，あなたが撮影した写真が，カメラぶれのためにぶれた写真になってしまったらがっかりですね。そうしたことをなくすためにも撮影の際に正しい姿勢をとることが大切なわけです。

29　では次の要領に従って正しい姿勢をとってみなさい。

①カメラの下カバーについているつり革を写真①のように首にかけて右手でカメラをもち，左手で下からささえるようにする。

②　両脇を体につけて，両足を写真のように少し開く。

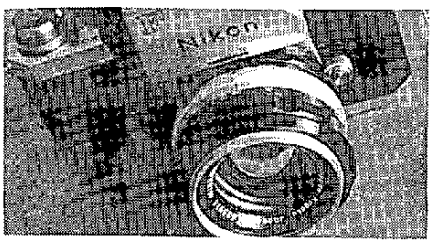
③写真②のようにファインダーを目の位置にあて顔にしっかり固定する。

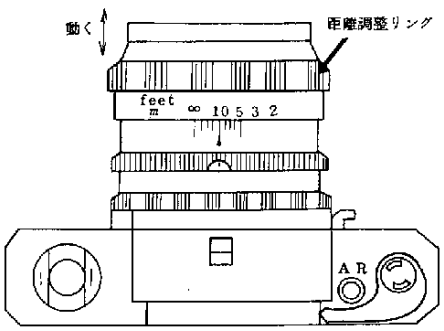
①

②

30　今，あなたがのぞいているファインダースクリーンに見えている部分が写真として写るわけですから，あなたの撮影したい人物（又は風景）が，ファインダースクリーン内の中央にくるようにカメラの方向を決めなさい。これを画面アングル調整といいます。

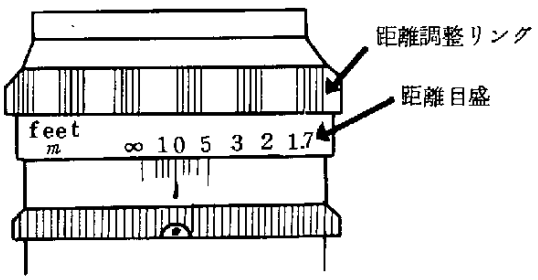
31　距離調整リングの合せ方について練習しましょう。まずカメラを目の位置から下げて写真の矢印のリングを左手でつかんで左右に一杯回してみなさい。その時レンズの部分が前後に動くはずです。





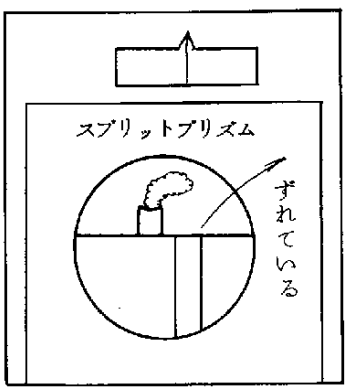
32　カメラではこの距離調整リングでレンズを前後させることによりフィルムに写る像のピントを変えているのです。

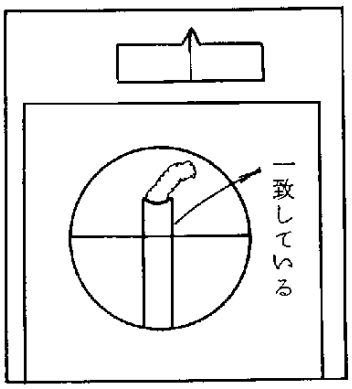
図のように距離調整リングのところに目盛がついていますが，これを距離目盛といいます。



33　ファインダースクリーンを見ながら図のようにスプリングプリズムの上下の像が一致するように距離調整リングを回して合せなさい。

①　　②





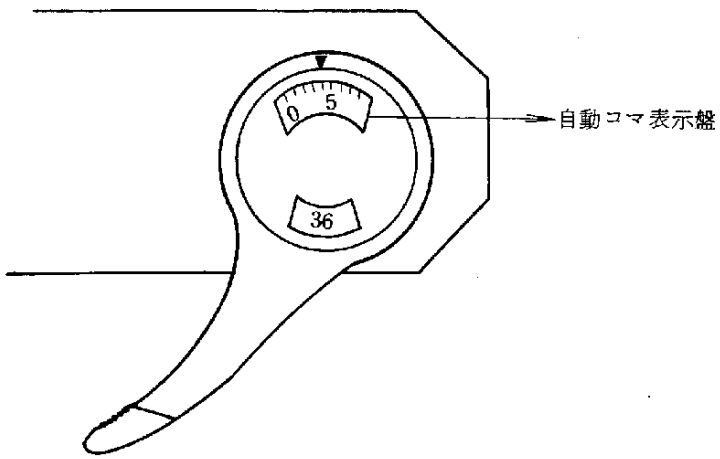
34　そのままの姿勢でシャッターボタンを右手人さし指で静かに押しなさい。もし，シャッターが切れなければ巻上げレバーを右に一杯回してから，シャッターボタンを押しなさい。これでフィルムが装てんされていれば撮影されるわけです。

35　ちょっとくたびれたでしょう。ここでひと休みしてから次のステップに進みましょう。

念には念を入れ，ということがある。こうしたカメラの操作もくり返しやって，瞬間的に反応できる神経をつくるのが大切です。

ひと休みしたら，またがんばって次のステップに進みましょう。

36　巻上レバーを右に一杯回しなさい。この時 1コマ，フィルムが送られて自動コマ表示盤の目盛が1つ進みます。



37　今度は被写体を変えて先程の露出，画面アングル，距離調整をもう一度練習しなさい。

38　つり革を首からはずし，カメラを机の上に静かに置きなさい。

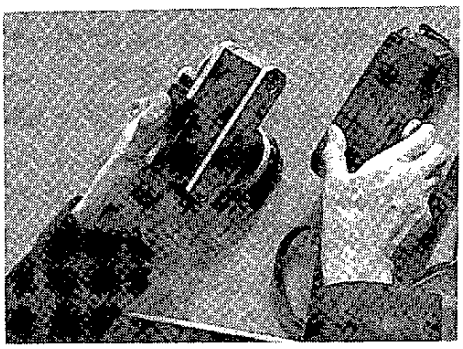
39　次に，カメラの内部の機構をみてみましょう。

まず，カメラが倒れないようにして，カバーをはずしなさい。

①カメラの位置を，レンズの下に横向きに置いて。

②下カバーのネジを，瞬間的に力をいれて，反時計方向にまわす。

③下カバーを水平に引出すようにとる。

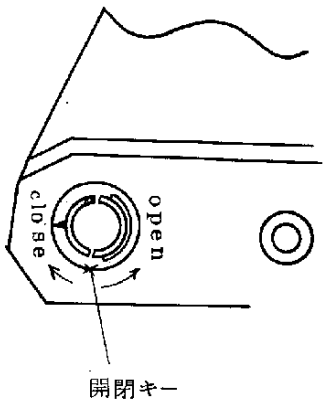


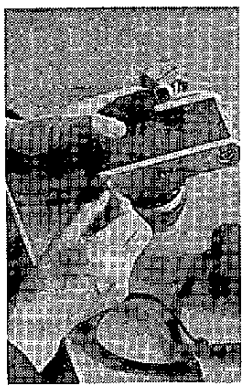
40　カメラの裏ブタをはずしなさい。

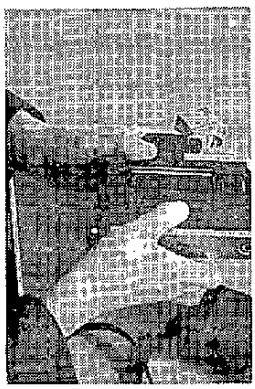
①裏底の開閉キーを右手親指ではねあげるような感じで起こす。

②開閉キーを　印が Close にあうまで回わす。

③右手で，裏ブタの中央を軽くつまんで，静かに水平に，引出す。

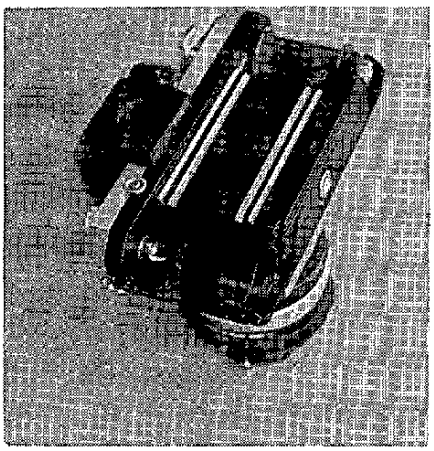






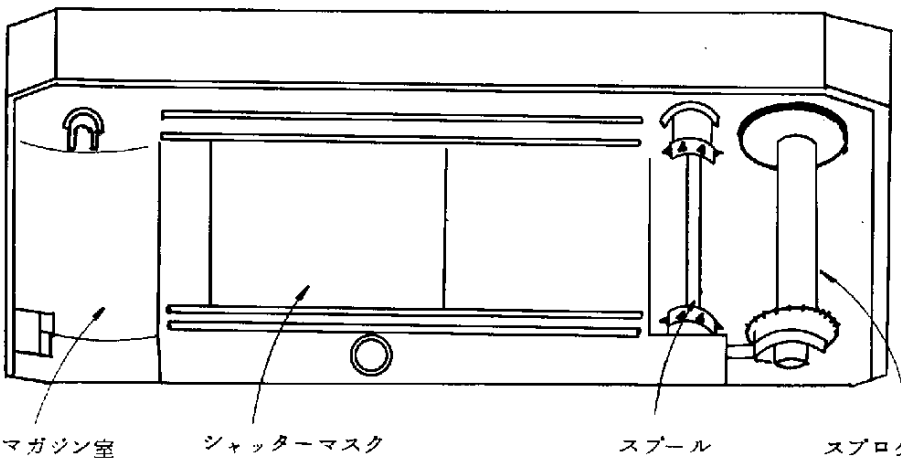
41　カメラの内臓部が出てきました。フィルムを入れてしまうと，もう見られませんから，今のうちによく見ておきなさい。

カメラをレンズを下にしたまま，正面におきかえなさい。

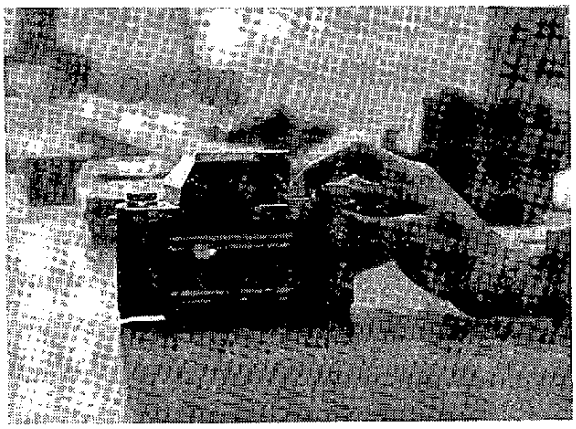


42　左の方にある丸いくぼみはマガジン室といって，フィルムが入るところです。右にあるスプールとスプロケットはフイルムのコマ送りと巻取りのためのものです。

その真中にある黒布は，シャッターマスクといい，この真うしろに来るフィルムに入る光をさえぎる役目をしています。



43　しぼりを1.4，シャッター速度を1にあわせなさい。カメラを両手で持ち，シャッター・マスクに注目しながらシャッターを押しなさい。もし，シャッターが切れなかったら，どういう操作をするのでしたか。



44　次にしぼりをそのままにして，シャッター速度を2にしてシャッターを切りなさい。シャッター・マスクに注目して。

45　シャッター速度の数字，1，2，……　，60，……　はシャッターマスクが開いている時間の逆数を表わしています。つまり，60は1/60秒のことです。
シャッター速度を次々変えてシャッターを切り，その感じをつかみなさい。

46　シャッターを切ると，シャッター・マスクが右に開いて，レンズを通して光が入ってくるでしょう。ここにフィルムがきていることを頭の中で想像しながら，もう一度，シャッター速度を1に合わせて，シャッターを切りなさい。

47　次に，しぼりを4にあわせて，シャッターを切りなさい。しぼりが，1.4の時とくらべて光の入り方はどうなるかを観察しなさい。

48　しぼりを16にあわせて，シャッターを切りなさい。しぼりの数字が大きい時は，レンズに入る光量が少なくなることが分るでしょう。

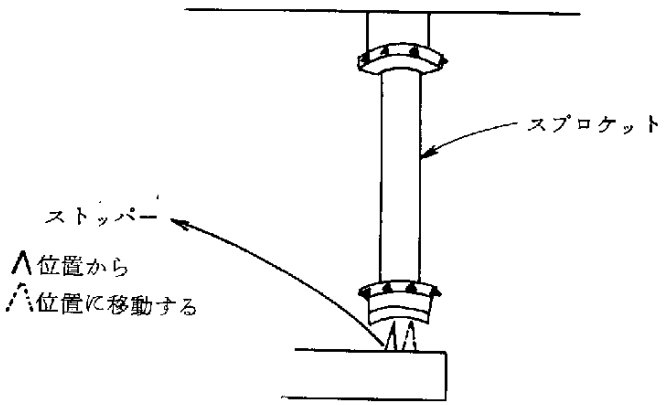
49　シャッター・マスクのちょうど真うしろに，フイルムが1コマずつ送られて来ます。シャッターを切った時に，シャッター・マスクが開いて光がフィルムに当り，フィルムを感光させます。シャッター速度としぼりの2つで光量を加減しているのです。

50　シャッター速度の標準は，いくつでしたか。
シャッター・マスクが開いている時間は，1／60秒という非常に短かい時間です。これで普通の明るさの所では充分に写せるのですからフィルムの感光感度は，相当よいことが分るでしょう。

51　次にフィルム巻取りの機構を見ておきましょう。巻上げレバーをまわしなさい。スプロケット，スプールが，回転するでしょう。その分だけ，フイルムが巻取られるのです。その分とは，丁度フイルムの1コマにあたります。そうして，次のフイルム1コマ分がシャッター・マスクの真うしろに送られて，撮影の準備がされるのです。

52　シャッター，巻上げレバー，A－Rリングの関係を調べましょう。前に，この3つの間には何か関係があるらしいといいましたね。

53　レンズを下向けにもって，図のようにスプロケットの下をよく見ながら，シャッターを押しなさい。
スプロケットの下から小さな爪のようなものが，シャッターを押した瞬間，下に引込み，次に右の方に移動するのが分りましたか。これをストッパーといいます。



54 ストッパーに注目しながら，巻上げレバーを回わしなさい。ストッパーが元の位置に戻ることが分るでしょう。ストッパーが右に動く事を確めながら，もう一度シャッターを押しなさい。

55 その状態でシャッターを押しなさい。ストッパーが下におりないため，シャッターが切れないでしょう。同じコマのところで2度撮りしないようにしてあるのです。
巻上げレバー，シャッターを交互にしないといけないのは，そのためです。この小さなストッパーが，その役目を持っています。

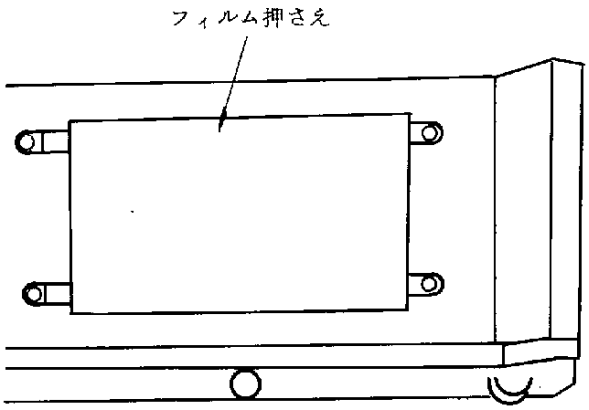
56 ストッパーをよく見て，ARリングをRにしなさい。ストッパーが下におりてしまって見えなくなるでしょう。巻上げレバーを回しなさい。ARリングがAにあった時にくらべて違う点があるはずです。

57 スプロケットが回らなかったでしょう。そしてレバーを回す手応えが違います。前に空回りすると言ったのはこのことです。

58 今度はスプロケットを手で回しなさい。簡単に回るでしょう。フィルムをカメラに入れる時，出す時にARリングをRにしておくのですが，それは，このようにして，着脱を容易にしています。

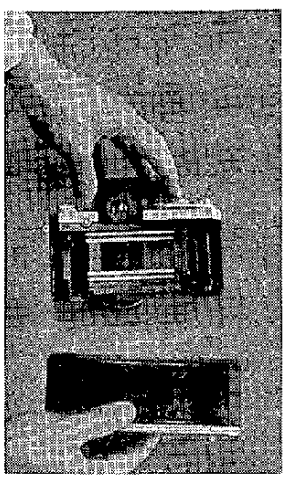
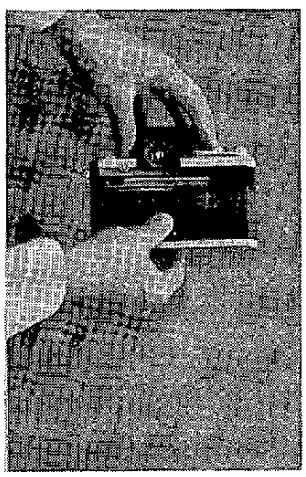
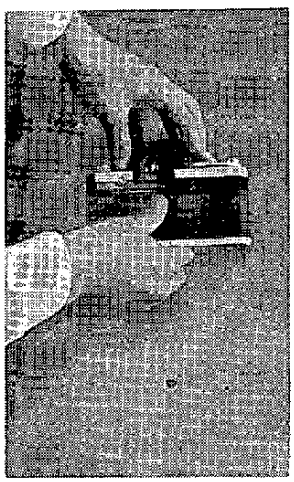
59 ARリングをAに戻して，スプロケットを手で回しなさい。回りませんね。ストッパーが上ってきて，巻上げレバーを操作しないとまわらなくなるのです。

60 カメラ内部の観察はこの位にして，裏ブタをしめましょう。図のようにバネのついたフィルム押えが裏ブタにあります。これはフィルムがたるまないように固定するものです。



61　カメラをレンズを下にして机の上におきなさい。フィルム押えがありますから，裏ブタを閉めるには一寸要領がいります。

右手で裏ブタをもち，カメラの両側の溝にあわせて，またカメラ内部の白い金属のフィルムガイドが一本見える位に残した位置で，裏ブタを静かに，水平に保ってカメラを上に置くようにはめなさい。水平に静かに押しこみなさい。

62　裏ブタ開閉キーを操作して，裏ブタをしっかり固定します。

①時計方向に，▲ 印が Close の指示に一致するまで。

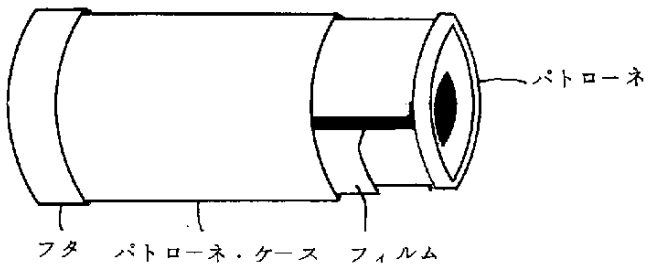
②右手親指で軽く，キーを Open 方向に倒す。

カメラを持って，裏ブタがしっかり固定されていることを確めなさい

63　それでは，いよいよフィルムを実際に装てんしてみましょう。

フィルムの入った箱をあけて，銀紙の袋に入ったフィルムを取り出し袋を破りなさい。

フィルムは，プラスチックのケース（パトローネ・ケース）と金属製のケース（パトローネ）に入っています。



64　A－RリングをRにセットしなさい。

65　前の要領で，裏ブタをはずしなさい。

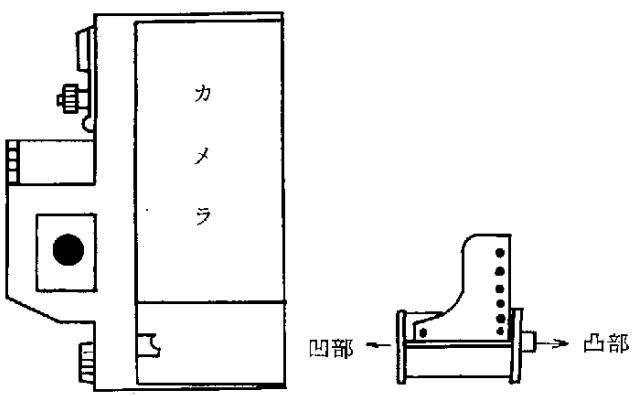
①カメラは，レンズを下にし，裏ブタが右側になるように置く。

②開閉キーを Open に合せる。

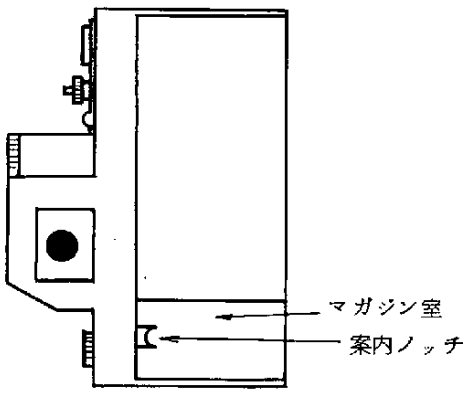
③裏ブタを水平方向に静かに引きぬく。

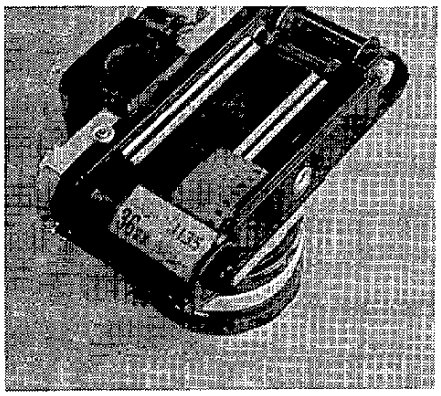
66　右手でパトローネをもちなさい。

パトローネは，凸部が右，凹部が左になるようにもちなさい。



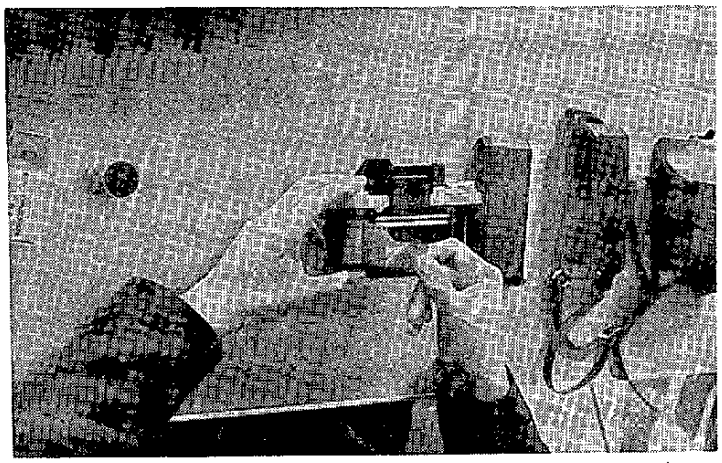
67　下図に示したマガジン室の案内ノッチに，パトローネの凹部がうまく入るようにして，パトローネをマガジン室に入れなさい。そのときには，フィルム引出し口が上側になるように注意しなさい。





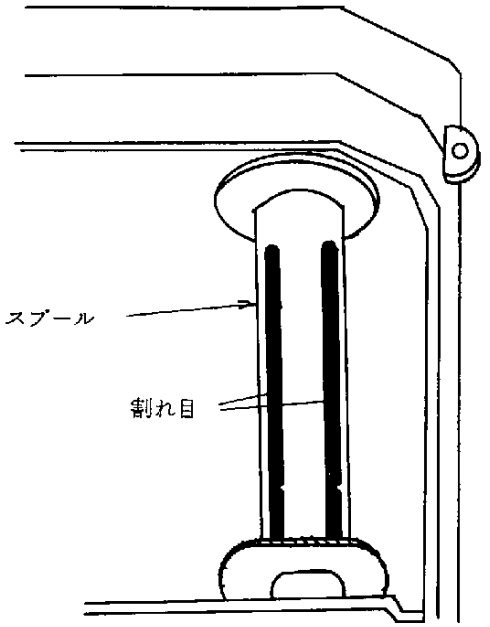
68　フィルムを１０cm程度引き出しなさい。

写真のように左手でカメラを支え，左手親指でパトローネをおさえながら，右手で引出しなさい。

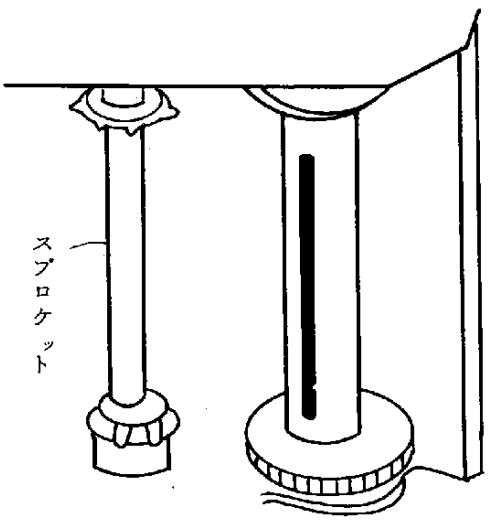


69　こゝで，スプールをよく見なさい。

スプールには，合計4カ所の割れ目がありますね。フィルムの先端をこの割れ目に1cm程度さしこみなさい。

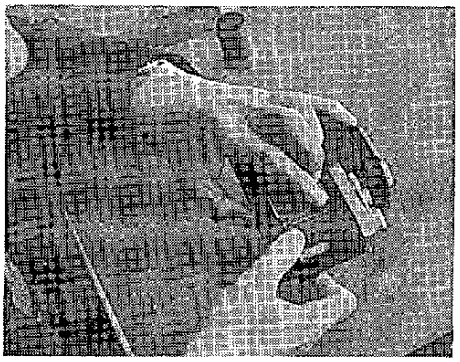


70　左手人さし指（または親指）でフィルムをおさえ，フィルム右端の送り穴（パーフォレーション）をスプロケットの突起にかみこませなさい。



71　スプールのギザギザがついたリングを右手の親指で反時計方向に回しなさい。スプールのつめにパーフオレーションがかみこみ，フィルムにスプールが巻きとられます。

フィルムの左端のパーフオレーションがスプロケットにかみこみ，フィルムにたるみがなくなるまでゆっくりとやりなさい。



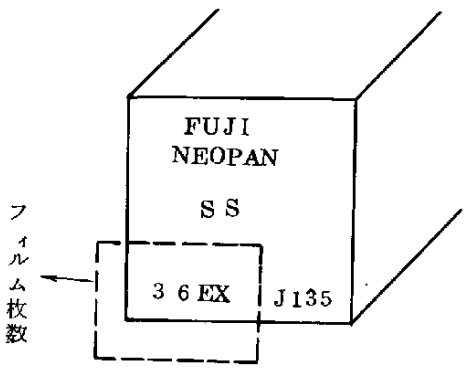
72　裏ブタをはめ，次に下カバーをはめなさい。

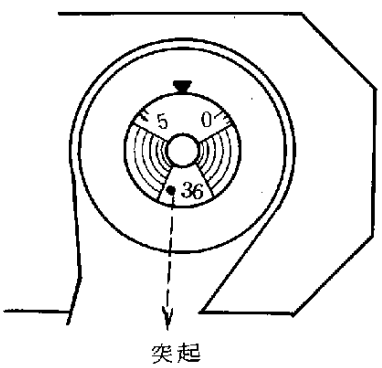
下カバーは，ネジを時計方向に回して，しっかりとめておきましょう。

73　下カバーをはめ終ったら，A－RリングをAにセットしなさい。

74　巻上レバーの軸の所に，図のような表示窓があります。これは，フィルムの枚数を忘れないようにメモしておくものです。

表示窓の突起を指で引っかけて右又は左に動かし，フィルム箱の表示枚数に合せなさい。



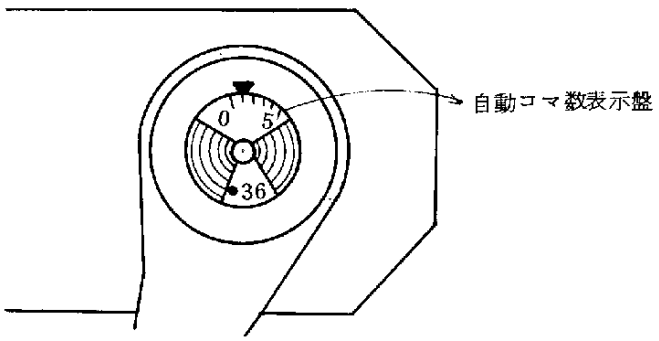


75　カメラのつり革を首にかけ，左手でカメラを下から支えるようにもちなさい。

76　巻上レバーを操作して，シャッターボタンの赤・が手前中央の位置にくるように調整し，シャッターを押しなさい。

フィルムを送りすぎないように，巻上レバーはゆっくりと慎重に操作しなさい。

77　先程の枚数表示窓の反対側にある自動コマ数表示盤を見ながら，表示枚数が1（0の1つ右側）を指すまで，フィルムを送りなさい。



78　さて，これから実際に撮影することになるのですが念のためもう一度各部分を確かめてみましょう。

最初にＡＳＡ目盛とフィルム外箱のＡＳＡ表示と合っていますか。合っていれば次のステップに進みなさい。

もし，合っていなければ，フィルム外側のＡＳＡ表示をよく見てＡＳＡ目盛を指に力をいれてもち上げ合せなさい。

79　露出計ＯＮボタンはＯＮになっていますか。なっていなければＯＮにしなさい。

80　次はシャッター目盛盤は６０になっていますか。なっていなければ６０に合せなさい。

81　上から露出計を見て，針が中央にくるように絞りリングを回して調整しなさい。

82　正しい姿勢でカメラをもちファインダをのぞきなさい。ファインダースクリーンにあなたが撮影しようとする被写体がくるようにカメラの方向を決めなさい。この時カメラを向ける方向が逆光にならないように注意しなさい。

83　ファインダーをのぞいた状態でファインダースクリーン内上部にある露出計の針がほぼ中央にくるように絞りリングを回して合せなさい。もしこの時絞りリングを左右どちらに回しても針が中央にこない時は先程のシャッター目盛盤のツマミをつまんで，その場所が暗いときには左に，明るいときには右に回して露出計の針がほぼ中央にくるようにシャッター速度も調整しなさい。

84　カメラを被写体に向けてファインダー内のスプリットプリズムに写る像の上下が一致するように前にやった要領で距離調整リングを回して合せなさい。

85　これで撮影のための準備はすべて終りました。さあ，シャッターを押して下さい。これで写真が1枚写せたことになります。つぎは巻上げレバーを右に一杯回しなさい。この時自動コマ数表示盤の目盛が1つ進んだことを確めなさい。

86　同様にして，露出，画面アングル，距離調整を毎回忘れず調整して，次々撮影しては巻上げレバーでフィルムコマ送りをしていきます。

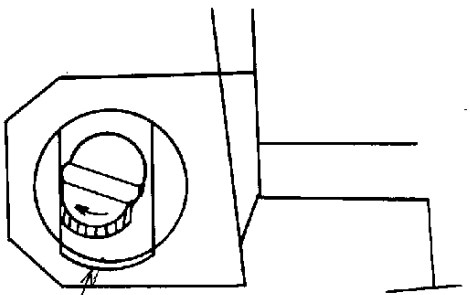
87　巻上げレバーが途中までしか動かないところが出てきます。自動コマ数表示盤を見て，フィルム枚数表示数がそれ以上になっていれば，フイルムを全部撮り終ったことになります。念のためシャッターを押してごらんなさい。動かないでしょう。

88　もしコマ数表示が規定数まで行っていないのに，巻上げレバーが動かないで，シャッターが切れない場合は，フィルム装置が悪くてフィルムがひっかかったか，カメラの故障かどちらかです。無理に巻上げレバーを回そうとしないで，カメラをそのままにして，写真屋さんに行くことです。

89　あなたの場合はうまくいきましたか。よかったですね。
それでは，最後の段階にいきましょう。

90　レンズキャップを，2つのボタンを軽く押しながら，レンズにはめなさい。はめこみが非常に浅いので，ちょっと不安な感じがしますよ。
露出ＯＦＦボタンを押して，露出計の電源をきりなさい。

91　フィルムの巻戻しをしましょう。ARリングをRにします。巻戻しクランクで操作しますが，カメラの上部左の方にあることを確めなさい。



92　右手親指で，巻戻しクランクのツマミを起こします。ツマミを矢印の方向にまわしていきなさい。引張られる感じがあるでしょう。

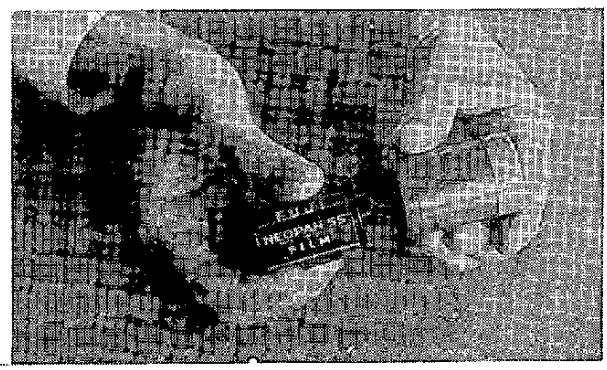
93　どんどん回していくと，急に固く引掛った感じがして，次に急に軽くなります。この時，スプールに入っていたフィルムがはずれて，フィルムがパトローネに収容されたのです。念のためもう少し回してみて，ARリングの赤・が動いていないなら，間違いなく巻取れていますから回すのをやめて，ツマミを戻しなさい。

94　巻戻した撮影済フィルムをカメラから取り出しましょう。
レンズを下にして机の上に横向きに置きなさい。
下カバーをはずしなさい。

95　裏ブタをはずしなさい。要領は前と同じですが，念のために書いておきましょう。
①開閉キーを起し，Open　のところまで回す。
②裏ブタを水平方向に静かに引出す。

96　フィルムがパトローネに巻取られているでしょう。巻戻しクランクを回す操作が不充分で完全に巻取られていなかったら，折角撮影したフィルムに光が入って２～３枚だめになります。パトローネをマガジン室からはずしなさい。

97　パトローネをパトローネケースに入れなさい。パトローネの凸部が中に入るようにしてやりなさい。パトローネケースにフタをしなさい。



98　これで無事３６枚撮影が完了しました。

カメラに裏ブタをはめなさい。カメラの溝，フィルムガイドの白線に注意して，静かに上から裏ブタをのせるように置いてから水平に押しこむのでしたね。

開閉キーを Close のところまで回すことも忘れないで下さい。

99　カメラに下カバー，上カバーをかけてしまいなさい。下カバーのネジ，上カバーのホックを忘れないように。別のフィルムを持ってきて，フィルムを装てんから後の操作をやってみて，もう１本写してごらんなさい。テキストを見ないで間違いなくできるまでくり返せば，このカメラの基本操作には，自信が持てるようになるでしょう。

100　ごくろうさまでした。これで NIKON F の操作はおわりです。

7.7 学習プログラムのトライアウトと評価修正

学習プログラムによるトライアウトを実施し，その結果について，学習者の反応行動の分析や学習者が目標行動を獲得しているかどうかの診断を行ない，学習プログラムを評価し修正する。

ニコンFカメラ操作の学習プログラムをトライアウトした結果，学習時間は60分であった。学習者からはカメラのもち方の指示，原理の説明が足りない等の意見が出された。

これらの結果から学習プログラムを評価し，カメラのもち方の指示，写真の使い方，図の使い方等を考慮して修正した。

第2部　CAI言語

第1章　CAIシステムのタイプ

ＣＡＩ言語の機能は，システムの規模や学習方式，特に対話方式が採用されるか，否かによってかなり異なる。

まずＣＡＩシステムのタイプをいくつかの角度から眺めてみよう。

(1) もっとも基本的なレベルのシステムでは，生徒と計算機の相互交信は最小限である。計算機は，タイプライタ又はスライドのような表示装置に教材を表示し，生徒は表示をみてフレームを理解し，理解ができたら，スイッチまたはボタン等をおして次のフレームへ行くことを計算機に合図する。計算機は次のフレームを表示するか，それに関する質問を出す。その質問に対する答は，ノートや教科書に記されており，学習者自身が答をチェックする。これは本質的には，プログラム学習書で学習しているのとはとんどかわらない。各フレーム（ステップ）ごとに，学習者は，次のフレームへ進んでもらいたい場合は，計算機に合図し，そして終りまでこの動作をくり返す。ここでは答の判断を学習者自身に行なわせるので比較的原始的なシーケンスに限定される。

(2) (1)に加えて，答のチェックを計算機（学習プログラム）で行なう。答のチェックには，原始的なパターン一致と，非常に精巧でかつ融通性にとんだ評価，解釈及び変形（式の）がある。後者は，本格的にやると自然言語の解釈（認識）と同じ問題につきあたる。

(3) 計算機が学習者の応答を記録保存することによって教師側に指導上の参照資料を提供する能力が拡張される。計算機は各生徒に対する教科の進行経路，質問に対する応答，試行回数及びその得点等を完全に記録することができる。

(4) ＣＡＩの環境（エンバイロメント）下では，色々な周辺装置（学習端末）が，計算機によってコントロールされ，教材を表示する。特にＣＡＩ用の周辺装置としては，スライド，マイクロ・フィシュ，テレタイプライタ，カソート・レイ・チューブ（CRT），手がき入力用のグラフィック・タブレット，タッチ・センシブ装置とテープ・レコーダ等があり，色々な組合せで利用されている。

今日もっともよく使われている装置は，タイプライタ（又はＣＲＴ）とスライドを組合わせ

たものです。

(5) ＣＡＩは本来コンピュータの力を借りて，各種の教育の効果をあげるというのが目的であるが，コンピュータも大型のものになるとその使用料は馬鹿にならない。特に１人の生徒の学習中に１つのコンピュータを専有してしまうような方法では非常に効率が悪い。少なくとも数個の学習端末あるいは他のジョブ（例えばバッチ処理）とのマルチ処理が出来るシステムであることが望ましい。

更に遠隔地から学習端末により教育を受けるということが普及すると，いわゆるタイムシェアリングシステムの下で運用することが必要となる。

第2章　基本CAI言語

学習プログラムの流れを記述するのに必要な基本的な命令は，学習端末に質問をだす命令，学習者の応答（答）の正誤を判定する正答命令及び誤答命令と，予想解答と一致しない場合の処理を行なう命令，学習の正規のシーケンスを変える命令，学習端末に説明やメッセージをタイプする命令などである。

ここでは，広く使われているＩＢＭ COUSEWRITER のサブセットを用いて詳説し，実例を示す。

命令の説明

命令には2つのクラス，メジャー（major ）とマイナ（miner ）がある。メジャー命令は教材を提示したり，質問に対する学習者の答をテストするのに直接関係する命令である。マイナ命令は本質的にはメジャー命令の補助である。そして，それらは，学習プログラム作成者が予想したいくつかの答と一致したかどうかの結果によって実行されたり，されなかったりする。メジャー命令は，それらに結びついている暗黙の分枝ロジックをもっており，マイナ命令は，普通は，シーケンス中の次の命令に続く。（マイナ命令の一種で，正規のコース・シーケンスをかえる分枝命令（br）をのぞく）。

qu（本文）

qu 命令は，メジャー命令である。本文には何を書いてもよい。この命令は，質問用に使われ，計算機に対して，学習端末に本文，すなわち質問をそのまま表示することを指令する。

これはＣＡＩ言語の基本命令で，教材および質問の提示用であり，のこりの命令の大部分は，質問に対する生徒の応答のインタプリートに関係する。

ca（本文）

このメジャー命令の本文の部分は，予想される生徒の答である。学習プログラム作成者は，予想した解答の全部をこの命令と次の cb 命令を使って表わすことができる。計算機は生徒の応答とこの命令の本文とくらべ，それが一致していれば1つのアクションを起し，異なっておれば別のアクションを起す。ca は correct answerの略語で，一般には正解を示すのに使われる。

cb（本文）

これは，予想される正解が沢山ある場合，前で説明した ca 命令で1番目の予想解答を指定し

残りの予想解答を1個ずつ本文欄に指定するマイナ命令である。この命令は生徒の応答と本文が一致した場合，先行するca と同じアクションを起す。

wa（本文）

このメジャー命令は，普通は，予想される誤答を表わすのに使われる。それは，次に実行されるアクションがその質問に対して生徒に別の応答を要求すること以外はca と同様である。

wb（本文）

これとwa の関係は，cb とca の関係と同じである。

un（本文）

すべてのca ，cb ，wa ，wb を試みた後，生徒の応答を認識できない場合に起るアクションを表わすときこの命令が使われる。それは，メジャー命令である。普通，学習プログラム作成者は，考えられるすべての応答を予想したり認識したいと思うが，もし認識できない応答があった場合は，制御はそのフレーム（ステップ）の先頭にもどる。この場合，命令の本文の部分はマッチングには使用されないで，それは生徒の端末にプリント・アウトされるメッセージである（たぶん▼あなたの答がわからないので，もう一度答えて下さい。）。un が実行された後は，学習プログラムが実行すべき他のアクションを特に指定していなければ，生徒はもう一度実行することを要求される。

br（名札）

これは，命令の正規のシーケンスを変化した場合に使われるマイナ命令である。計算機に次の命令を実行させるのではなく，本文で示された名札の付いたフレーム（命令）へ制御を渡すことを示す。もし，wa 又はun の次にbr があれば，生徒の別の答を要求するwa 又はun 命令の機能を無効にする。

ty（本文）

このマイナ命令は，本文欄のメッセージを生徒のコンソール・タイプライターにタイプ・アウトする。qu とちがって，メッセージ・プリント命令ty は，タイピング以外の機能をもたない。

第３章　基本CAI言語の拡張機能

生徒の応答パターンを診断したり，応答に応じて適切に選択された治療ルーチン（remedial routine）をコールしたり，またはそれに対する種々の学習経過に応じて異なったコース・セグメント（ステップ）へ分枝する学習プログラムの構造について述べる。

前に述べたような基本ＣＡＩ言語では，最後の生徒の応答しか分析に使うことができないが，生徒の学習履歴をずっと収集し続けることはかなり面倒ではあるけれど不可能ではない。学習プログラムでこれを実行するには２つの方法がある。

第１の方法は，生徒は階層，すなわち質問の木構造を通って分枝する。生徒が木の任意の特別な枝の終端に到着したとき，学習プログラム作成者は，生徒が正しく応答したか又はしなかったかというあるシーケンスによってそこに到着したことがわかる。生徒の学習履歴の全体のパターン，あるいは全体の効率を判定する判定基準を明白に書き下す必要がない。この技術は，後で例示するように，すでにのべられた基本ＣＡＩ言語でも実施することができる。

もう１つの方法は，各質問の応答を記録し，ある質問の集合が完了した後で応答の集合を分析する。この方法では，判定過程（decision－making）は判定する点までの応答の完全な集合を直接分析することによってなされる。このモードでは，学習プログラム作成者がはっきりと判定（分類）基準を表わすことが必要である。前述の基本ＣＡＩ言語ではこの技術に対する機能をもっていない。そこで，やゝ拡張されたＣＡＩ言語としては次のような機能をもつことが望ましい。

1. 学習プログラム作成者が指定した応答あるいは他の情報を各々の生徒ごとの学習経過記録とは別に主記憶装置のエリアに格納する。
2. 条件によって異なったフレームに分枝する。すなわち，主記憶の内容を使って，br 命令を実行するか無視するかを決定する。
3. 主記憶の内容で演算を実行する。たとえば，学習プログラム作成者は，コースが進行しているときの正誤答の回数，あるいは，特定の問題に対する生徒の誤答回数，あるいは，生徒が正しく答を答えるまでに別の質問を何問やったかの回数等をカウントすることを可能にする。
4. 生徒の応答又は応答のサブセットを記録する。記録されるサブセットは，正答，誤答，認識できない答，すべての答，その他である。これらの学習経過は，磁気テープ又はディスクに記

録され，ＣＡＩ言語で書いたプログラムでは直接読んで利用することはできないが，後で学習プログラム作成者及び教師の利用のため，プリント・アウトされる。

5. 任意の機能を実行したり，生徒の応答を自由に処理したりするオウンコーディング（ own coding）のプログラムを組込める。簡単な例は，長い応答を記憶されている標準解答とのチエックを簡単にするため，シフト・コードや句読点を取りのぞく場合である。（例えば The Quick Broun "Fox" は the quick broun fox 又は thequickbroーunfox と等しいと判定できる。）

6. 生徒のコーンソールに付属した音声と画像表示装置の使用を制御する。これらの命令は，論理的には ty と似ており，計算機にプロジエクタから特定のスライド，あるいはテープ・レコーダから録音した１コマのメッセージを提示することを命令する。

第4章　基本CAI言語による学習プログラムの例

次の学習プログラムは，初等代数の2次方程式を教えるために，この基本ＣＡＩ言語をどのように使って記述するかを述べた簡単な例である。

名札	命令	本　　文	注　　釈
A1	qu	2次方程式の一般形は $ax^2+bx+c=0$ です。ここでa，b，cは定数で， xは変数です。 2次方程式は何個の根があるか知っていますか。	この命令は教材を表示し，質問をたずねる。
	ca	2	2つの正しい形が予想される。しかしまだ他の解が可能です。（すなわちTwo）。
	cb	two	
	ty	Good	もし正しければ生徒は，ほめられる。実行される次の命令はA2の名札がついたフレームである。
	un	あなたの答は正しくない。これはあなたにとっては最初のＣＡＩコースであるので，タイプやオペレーションの誤りをおかしているかもしれない。 印刷の誤まりをチェックして，もう一度答えて下さい。	生徒はまちがった答を入力したので，その質問に対しもう1回答えなければなりません。生徒の最初の認識できない答のあとにこのメッセージが打たれる。生徒が再び認識できない応答を入れると，次のｕｎが実行される。
	un	2次方程式には2つの根　すなわち2つの解がある。	この時点では，生徒は答を知らないと仮定する。答をおしえて，次のフレームへ分枝する。
	br	A2	
A2	qu	方程式$3x^2+2x+1=0$に対す	2次方程式の一般形についての知識を簡単に

名札	命令	本　　文	注　　釈
		る a の値はいくらですか。	テストする。
	ca	3	もし，答があっていれば，生徒はほめられ，
	ty	Good	次のフレームへ行く（br 命令は必要なしに）
	wa	2	これらの答のうちどちらかは，タイプ・エラ
	wb	1	ーよりも教材を理解していないことを示して
	ty	方程式の一般形で，a は x^2 の係数	いる。生徒にヒントを与え，もう一度答えさ
		であることを思いだして下さい。	せる。
		一般形を再度チェックしてもう一	
		度答えて下さい。	
	un	あなたの答は a，b，c のどれにも	生徒の答は予想された誤答の一つでない（又
		相当しない。もし助言が必要なら	は正しくない）ので，生徒は非常にこまって
		ば先生を呼びなさい。あるいはも	いるかもしれない。先生から助言をしてもら
		う一度答えて下さい。	った方が良いとおもわれる。
	un	たぶんあなたは助言をきいた方が	2回ともあやまった答をした場合は，この2
		よいですよ。	次方程式の一般形を全然理解してないことを
	br	END	はっきり示しており，生徒の学習を打切る。
A 3	qu	b？	生徒は，ここに達するためには，a を正しく
	ca	2	答えなければならない。今度は b についてた
	ty	Good	ずねる。
			生徒が1回で正解に達したかどうか判定し，
			もし1回で正しく答えられなかった場合，こ
			こで，別のフレームへ飛んで，システムのタ
			イプやオペレーションの仕方を復習させるこ
			ともよいことである。これを実行するために
			は，前の結果を記録する主記憶装置を必要と
			する。
			特に 'GOOD，それは，two ですね。' あ
			るいは 'いまあなたは正しい答に達した。'（そ

名札	命令	本　　文	注　　釈
			のフレームに対して何回かまちがい，ここで始めて正解を答えた場合）などと表示したり，あるいは，もし両フレーム（A1，A2）に対して2回以上誤答した場合，簡単な説明を表示することは望ましいことかもしれない。
	wa	3	ここでは生徒の学習経過に関して暗黙のうちに，いくつかの情報を得ている。すなわち生徒はフレームA2を正しく答えたにちがいない。それでなければ，ここに到達しなかったからである。それ故に，ここでは最後の答に関する非常に特別なコメントを与える。
	ty	しかし，あなたは直前に正しく $a=3$ と答えた。それでは b はいくつですか。	
	wa	1	先生はもはや我慢できなくなった。簡潔なヒントが与えられ生徒はもう一度答える。
	ty	b は x の係数です。もう1度答えて下さい。	
	un	あなたの答はまちがっています。もう一度答えるか，助言を聞いて下さい。	もし，答がまだわからなければ，生徒はシステムの操作方法がわからなくて困っているにちがいない。
	un	先生に相談して下さい。	先生がかれの学習経過記録を調べ，トラブルの原因をみつけるまでは，生徒との学習を続行することができない。
	br	END	
A4	qu	そのとき，c は＿＿＿でなければならない。	ここでも生徒が今までのすべての問題に正しく答えた（たとえ1回で答えられなくても）ということがわかっている。そこで今度は，文脈中に答を埋める形の質問が出される。
	ca	1	この答は次のフレームへ制御を渡す。
	ty	Right	
	wa	2	
	wb	3	

名札	命令	本　　文	注　　釈
	ty	あなたは，前の問題に対してこの答をあたえた。cは方程式中の定数項です。 もう一度答えて下さい。	
	un	あなたの答は認識できない。助けをよぶか，もう一度答えて下さい	もし２つのunがあれば，最初の認識できない答の後で最初のunが実行され，第２回目の認識できない答の後で２番目のunが実行される。
	un	あなたの答は，まだ認識できない 先生から助言をうけて下さい	これは生徒がまちがいを続けている時，先生に段々厳しくしたり，さらにヒントを与える機会をあたえる。
	br	END	
A5	qu	２次方程式の根は $x=\frac{-b\pm(b^2-4ac)^{1/2}}{2a}$ です。もしそれぞれの根をr_1とr_2であらわせば $r_1=\frac{-b+(b^2-4ac)^{1/2}}{2a}$ です。	生徒にａ，ｂとｃの意味を思い出させる。今，この記法は，新しい教材中によく使われる。
		ではr2に対する？印の所をうめなさい。 $r_2=\frac{-b?(b^2-4ac)^{1/2}}{2a}$	多分，ここで学習プログラム作成者がほんとうにたずねたいと思っているのは－'r^2の式は何んですか？'－という質問であるが，この方程式をタイプライタ上に整頓して印字することはたいへんである。そこで，ここでは生徒の理解力をテストするのが主目的であるので，より扱い易い単純化された質問の形を使っている。
	ca	－	
	cb	minus	
	ty	Very Good	

名札	命令	本　　文	注　　釈
	wa	＋	
	wb	plus	
	ty	もしこれが正しければ r_1 と r_2 のちがいはなんですか。根の一般形をよくみてもう一度答えて下さい。	
	un	あなたの答はまちがっています。もう一度答えて下さい。	
	un	あなたの答はまだまちがっています。再び続ける前に先生の助言をうけて下さい。	

次にこの学習プログラムで学習した例を示す。生徒の応答は下線が引かれている。

学　習　経　過	注　　釈
2 次方程式の一般形は $ax^2+bx+c=0$ です。ここで a, b, c は定数で, x は変数です。	
2 次方程式は何個の根があるか知っていますか。	
<u>yes</u>	生徒の応答は，学習プログラム作成者が期待したものではない。
あなたの答は正しくない。これはあなたにとっては最初のＣＡＩコースであるので，タイプやオペレーションの誤りをおかしているかもしれない。 印刷の誤りをチェックしてもう一度答えて下さい。	

学　習　経　過	注　　釈
2	
Good 方程式 $3x^2+2x+1=0$ に対する a の値はいくらですか。	
2	
方程式の一般形で，a は x^2 の係数であることを思いだして下さい。 一般形を再度チェックしてもう一度答えて下さい。	
3	
Good b？	
2	
Good そのとき，c は＿＿＿でなければならない。	
2	
あなたは，前の問題に対してこの答をあたえた。c は方程式中の定数項です。 もう一度答えて下さい。	
1	
Right 2 次方程式の根は $x=\frac{-b\pm(b^2-4ac)^{1/2}}{2a}$ です。もしそれぞれの根を r_1 と r_2 で表わせば $r_1=\frac{-b+(b^2-4ac)^{1/2}}{2a}$ です。では r_2 に対する？印の所をうめなさい。 $r_2=\frac{-b?(b^2-4ac)^{1/2}}{2a}$	
minus	
Very Good	

第5章　CAI言語の種類

他の分野と同様，ＣＡＩの分野においてもコンピュータの利用方法がバッチ的なものから，人間と機械のよりインタラクティブな形えと拡張されつつある。

学習者とコンピュータ間のコミニュケーションは，ＣＡＩの本質的要求の一つであり，今更言うまでもないことであるが，最近は特に学習プログラム作成者がインタラクティブにコンピュータと会話をしながらプログラムを作成したいという要求も出て来た。

現在までにいろいろなＣＡＩ言語が提案され，更にある種のコンピュータにインプレメントされ，実用に供されているものも少なくはない。そしてまたこれらの言語をある程度分類して整理する試みもすでにおこなわれているが，ここでは，大きく4つに分け，プログラムの作成が対話方式で行なわれるかどうか，言語の適用範囲が汎用かあるいはＣＡＩ専用かどうかの組合せで分類してみる。

5.1　非対話方式汎用言語

この範疇には，各種のアセンブラや通常のコンパイラ，たとえばFORTRAN，ALGOL，COBOL等が含まれる。これらの言語を使用して学習プログラムを作成する場合，ＣＡＩ特有の機能がないためプログラムをマクロ的に表現できず，概して複雑になり，プログラムの作成に非常な労力を要し，またその修正も容易でない。しかし，汎用言語であるので制約の少ない自由なプログラムが可能であり，熟達したプログラマならば大規模な効率のよい学習プログラムを作成することができる。しかし，一般には学習プログラムの作成者は計算機になじみがうすいので負担が大きくなる。

最近開発されたＰＬ／Ｉ言語は非常に強力なプログラミング言語であり，一般のＣＡＩ言語で省略した複雑な文字処理，編集およびファイル処理機能を有している。生徒と端末を通して対話（学習）している場合，質問に対して同じことを表わす色々な答が想定される場合が多いが，ＰＬ／Ｉの編集機能を使用するとかなり簡単にコーデング（表現）することができる。

5.2　対話方式汎用言語

この範疇は，ダートマス大学で開発されたBASIC(Beginners All-purpose Instruction Symbol Code)と，RAND社が開発した人間と機械との対話に重点をおく，JOSS(Johniac Open-Shop System)等がある。

これらの言語は科学計算用に設計されており，対話方式の特徴である簡潔な表現でステートメントをタイプライタから入力できる反面，多種類の学習端末(周辺装置)を制御し，複雑な判定をともなう学習プログラムの記述には適していない。

BASIC言語を使った例としては，東芝と日本システムが開発したGE-635TSSの元で動くBASIC言語を教えるTUTORシステムがある。このシステムは，学習プログラムをBASIC言語で記述したのではなく，TSSのユーティリティ・プログラムを利用して，問題そのものをプログラミングし，ソース・プログラムをファイルに登録し，学習者は，テキストを見て，問題のファイルを調べ，ファイルからソース・プログラム(問題)を呼び出し，ソース・プログラムを修正し，コンパイルおよび実行をおこない，結果(答)をプリントし，結果(プログラム)が合っているかどうかを学習者自身が調べるようになっている。表2－1に例を示す。

これはテキストのセクション5(四則演算とべき)の2問目のプログラムを訂正する問題である。学習者はTSS用のLISTコマンドをタイプインして問題を計算機からプリントさせ，そして次のように答をタイプインしてRUMコマンドを指定する。

```
470 LET X=B+B↑2*C↑3-2/A
RUN
```

すると計算機から578と出力する。正答はプログラムのライン番号320に記載されている。学習者自身が答をチェックする。

5.3 非対話方式CAI言語

この種に属する言語は，CAI向きに考えられた言語であるが，対話方式になっておらず作成者が学習端末(一般にタイプライタ)から学習プログラムを作成する場合，非常な労力を要し，修正もたいへんである。

この範疇にはIBMのCOURSEWRITER Ⅰ／Ⅱ／Ⅲ，CAL(Course Author Language)，RCAのISL-1(Instructional System Language-1)，UNIVACのCOPI(Computer Oriented programmed Instruction)，Illinoi大学のTUTORなど各種ある。この他にFORTRANにいくつかのサブルーチンを追加してこの種

```
10   REM * THIS IS «TUT5-2»
20   REM * CORRESPONDING SECTION FOR THE TEXT IS
30   REM   SECTION 5 :    FUNDAMENTAL ARITHMETIC.
40   REM
50   REM
60   REM
70   REM      ****************************************
80   REM
90   REM      *****            PROBLEM            *****
100  REM
110  REM      ****************************************
120  REM
130  REM
140  REM * FIND X
150  REM   WHEN AX+2=A(B+(B*B)*(C*C*C))
160  REM   AND  A=2   B=3   C=4
170  REM
180  REM
190  REM
200  REM      *******       SUGGESTIONS         *******
210  REM
220  REM
230  REM * CORRECT THE PROGRAM WHICH BEGINS AT LINE NUMBER 460.
240  REM
250  REM * USE ARITHMETIC OPERATION «↑» WHEN NEED.
260  REM
270  REM
280  REM
290  REM      *******     PRINT  OUT       *******
300  REM
310  REM
320  REM   5 78
330  REM
340  REM
350  REM
360  REM      ***********  HINTS      ***********
370  REM
380  REM * BE CAREFUL TO CHECK THE CORRECT ORDER OF EACH
390  REM                                        STATEMENT.
400  REM
410  REM
420  REM
430  REM      **********  PROGRAM      *********
440  REM
450  REM
460       READ   A, B, C
470       LET    X=
480       PRINT  X
490       DATA   2, 3, 4
500       STOP
510       END
```

表2－1. TUTORの実例

の言語としたり，ALGOLにCAI用ステートメントを追加したドイツのアーヘン（Aachen）大学のPADALOGICA(Padagogisch Logica) がある。

ここでは，PADALOGICAの文法をバッカス記法をもちいて紹介する。

<プログラム>::=<ブロック> (A-47)

<ブロック>::=<ラベルのつかないコック>|
<ラベル>:<ブロック> (B-47)

<ラベルのつかないブロック>::=<ブロックの頭部>;
<複合文の尾部> (C-46)

<ブロックの頭部>::=BEGIN<宣言>|
<ブロックの頭部>;<宣言> (D-45)

<複合文の尾部>::=<実行命令>END|
<実行命令>;<複合文の尾部> (D-42)

<ラベル>::=<命令記号>|<符号のついていない整数> (C-48)

<命名記号>::=<文字>|<命令記号><文字>|
<命令記号><数字> (O-4)

<符号のついていない整数>::=<数字>|
<符号のついていない整数><数字> (D-49)

<文字>::=a|b|……|y|z|A|B|……|Y|Z (P-6)

<数字>::=1|2|……|8|9 (P-3)

<宣言>::=<型の宣言>|<配列の宣言>|<スイッチの宣言>|
<手続きの宣言>|<時間の宣言> (E-47)

<型の宣言>::=<型><型のリスト> (F-54)

<型>::=REAL|INTEGER|BOOLEAN|TEXT (G-55)

<型のリスト>::=<単純変数>|<単純変数>,<型のリスト> (G-53)

<配列の宣言>::=<型>ARRAY<配列のリスト> (F-49)

<配列のリスト>::=<配列要素>|<配列のリスト>,<配列要素> (G-50)

<配列要素>::=<配列の命名記号>〔<区間のリスト>〕|
<配列の命名記号>,<配列要素> (H-50)

<区間のリスト>::=<添字式>|<添字式>,<添字式> (J-48)

<添字式>::=<添字>|<添字><加減算記号><添字>　(K-48)

<添字>::=<単純変数>|<符号のついていない整数>　(L-48)

<時間の宣言>::=TIMER<命令記号>;<実行命令>

<実行命令>::=<条件のつかない実行命令>|<条件つき実行命令>|
<くり返し実行命令>　(E-31)

<条件のつかない実行命令>::=<基本実行命令>|<複合実行命令>|
<ブロック>　(F-19)

<基本実行命令>::=<計算記録命令>|<ロード命令>|<出力命令>|
<音声命令>|<画像命令>|<位置命令>|<消去命令>|
<ウエート命令>|<時間制限命令>|<飛び越し命令>|
<スプリット命令>|<空の実行命令>|<手続き実行命令>|
<ラベル>:<基本実行命令>　(F-19)

<計算記録命令>::=<左部分のリスト><式>　(H-10)

<左部分のリスト>::=<左部分>|<左部分のリスト><左部分>　(J-10)

<左部分>::=<変数>{:=, ←}|<配列の命名記号(形はTEXT)>:=　(K-10)

<変数>::=<単純変数>|<添字つき変数>　(L-3)

<単純変数>::=<変数の命名記号>　(M-2)

<変数の命名記号>::=<命名記号>　(N-2)

<添字つき変数>::=<配列の命名記号>〔<添字のリスト>〕　(M-4)

<配列の命名記号>::=<命令記号>　(N-4)

<添字のリスト>::=<添字式>|<添字式>,<添字式>　(O-6)

<式>::=<計算式>|<論理式>|<テキストの式>　(J-12)

<計算式>::=<項>|<加減算記号><項>|<計算式><加減算記号><項>　(L-11)

<項>::=<要素>|<項><乗除算記号><要素>　(M-11)

<要素>::=<符号のついていない数>|<変数>|<関数記号>|
(<計算式>)　(N-11)

<乗除算記号>::=×|／　(N-10)

<加減算記号>::=＋|－　(M-10)

<符号のついていない数>::=<10進数>　(O-11)

<10進数>::=<符号のついていない整数>|
<符号のついていない整数>

＜小数＞ (P-11)

＜小数＞::＝．＜符号のついていない整数＞ (R-10)

＜テキストの式＞::＝＜テキスト＞|＜変数（形はTEXT）＞ (K-12)

＜テキスト＞::＝＊＜基礎記号の連り＞＊ (L-12)

＜基礎記号の連り＞::＝＜基礎記号＞|＜基礎記号の連り＞＜基礎記号＞ (M-13)

＜基礎記号＞::＝＜文字＞|＜数字＞|＜限定記号＞ (N-14)

＜限定記号＞::＝＋|－|×|／|．|，|；|？|！|␣| |… (O-18)

＜ロード命令＞::＝LOAD(＜ロードのリスト＞) (H-12)

＜ロードのリスト＞::＝＜ロードのセグメント＞|

＜ロードのリスト＞，＜ロードのセグメント＞ (J-14)

＜ロードのセグメント＞::＝＜変数＞|＜配列の命令記号（形はTEXT）＞ (K-16)

＜出力命令＞::＝＜表示命令＞|＜印刷命令＞ (H-17)

＜表示命令＞::＝WRITE (K-17)

＜印刷命令＞::＝PRINT（＜印刷のリスト＞） (K-18)

＜印刷のリスト＞::＝＜印刷のセグメント＞|

＜印刷のリスト＞，＜印刷のセグメント＞ (K-19)

＜印刷のセグメント＞::＝＜変数＞|＜テキスト＞|＜配列の命令記号＞ (L-17)

＜音声命令＞::＝PHONE（＜添字＞，＜添字＞） (H-18)

＜画像命令＞::＝SHOW（＜添字＞） (H-24)

＜位置命令＞::＝＜画像位置命令＞|＜音声位置命令＞ (H-25)

＜画像位置命令＞::＝SELECT（＜添字＞） (K-25)

＜音声位置命令＞::＝RUN（＜添字＞） (K-27)

＜消去命令＞::＝ERASE (H-28)

＜ウエート命令＞::＝WAIT（＜添字式＞） (H-29)

＜時間制限命令＞::＝＜命名記号＞（＜時間のパラメータ＞） (H-32)

＜時間のパラメータ＞::＝＜添字式＞ (K-30)

＜飛び越し命令＞::＝GO TO＜ラベル＞ (H-33)

＜スプリット命令＞::＝SPLIT（＜狭義の部分＞，＜広義の部分＞） (H-35)

＜狭義の部分＞::＝＜変数（形はTEXT）＞|

＜配列の命名記号（形はTEXT）＞ (K-35)

<広義の部分>::=<配列の命名記号(形はTEXT)>　(K-37)

<空の実行命令>::=<空>　(H-36)

<空>::=　(J-36)

<複合実行命令>::=<ラベルのつかない複合文>|
<ラベル>:<複合実行文>　(G-6)

<ラベルのつかない複合文>::=BEGIN<複合文の尾部>　(H-6)

<条件つき実行命令>::=<ＩＦ付き実行命令>|<ＩＦ付き実行命令>
ELSE<実行命令>|<条件節><くり返し実行命令>　(F-42)

<ＩＦつき実行命令>::=<条件節>|<ラベル>:<ＩＦ付き実行命令>　(G-42)

<条件節>::=ＩＦ<論理式>THEN　(H-42)

<論理式>::=<単純論理式>　(J-43)

<単純論理式>::=<包含関係式>|<単純論理式>ＥＱＶ<包含関係式>　(K-43)

<包含関係式>::=<論理項>|<包含関係式>ＩＭＰ<論理項>　(L-42)

<論理項>::=<論理要素>|<論理項>ＯＲ<論理要素>　(M-42)

<論理要素>::=<論理素要素>|<論理要素>ＡＮＤ<論理素要素>　(N-42)

<論理素要素>::=<論理素子>|ＮＯＴ<論理素子>　(O-42)

<論理素子>::=<論理値>|<変数>|<関数記号>|<関係>|
(<論理式>)　(P-42)

<論理値>::=ＴＲＵＥ|ＦＡＬＳＥ　(R-40)

<関係>::=<計算式><関係演算記号><計算式>|
<テキストの本体><対等記号><テキストの本体>　(R-43)

<関係演算記号>::=<対等記号>|<比較記号>　(S-45)

<対等記号>::=EQUAL | NOT EQUAL　(T-44)

<比較記号>::=LESS | NOT LESS | GREATER |
NOT GREATER　(T-47)

<テキストの本体>::=<変数(形はTEXT)>|＊<基礎記号>＊　(S-41)

<くり返し実行命令>::=<くり返しの節><実行命令>|
<ラベル>:<くり返し実行命令>　(G-2)

<くり返し節>::=ＦＯＲ<変数>$\left\{ \begin{matrix} : \\ \leftarrow \end{matrix} = \right\}$<添字式>ＳＴＥＰ<添字式>
ＵＮＴＩＬ<添字式>ＤＯ　(H-1)

5.4 対話方式ＣＡＩ言語

この種の言語は，計算機に関して何ら特別な知識のない学習プログラム作成者でもテレタイプを介して計算機と直接やりとりしながら学習プログラムを作成し，または修正，改良を加えることができる。

この種の言語には，ＳＤＣ（System Development Corporation ）が本格的な強力なＣＡＩ言語として開発したPLANIT（Programming Language for Inteructive Teaching）やＩＢＭのＣＧ－Ⅰ／Ⅱ（Cource Generation）等がある。

ここでは，PLANI Tを例をもちいて簡単に紹介する。学習プログラムの作成はシステムからの問いかけに応答して行なわれ，問いかけがわからない場合は逆にシステムに質問することができるようになっている。

例えば，＊WHO INVENTED THE ELECTRIC LIGHT?＊（誰が電燈を発明したか？）という問題に対する学習プログラムは，次に示す対話によってプログラミングされる。

DIALOGUE	説　　明
User:　＊CO	レッスン組立モードにおける開始オペレーション。＊印は，ユーザからの入力を必要とする場合，システムからプリントされる。
System: P／Q／M／D／C	システムが省略形でユーザが組み立てようとしているレスン・フレームのタイプの選択を要求する。
User:　＊？	ユーザが上のプリント・アウトの説明を要求する。
System: (P)ROBLEM／(Q)UESTION／ (M)ULTIPLE CHOICE／ (D)ECISION／(C)OPY	
User:　＊Q	ユーザは，質問フレームを定義する。
System:　1. FRAME 25.000 LABEL＝＊HISTORY	システムはレスン・フレーム番号25を割当てる。ユーザはレーベルをHISTORY（歴史）とする。
System:　2. SQ。	質問を定義する。

DIALOGUE		説　　明
User:	* WHO INVENTED THE ELECTRIC LIGHT ?	
System:	3. SA.	生徒からの予想される答を定義する。
User:	*A + EDISON	Aの後の＋印は，これが正答であるこ
	*B MARCONI	とを示している。
	*C BELL	
System:	4. SAT.	生徒からの応答により次の行なわれる行動を定義する。
User:	* A + F : GOOD WORK . B : 5	ユーザは，もし生徒が正しくAを答えたならば，実行されるべきフィードバック・コマンドを定義する；そのときはプログラムはレスン・フレーム5へ分枝する。生徒がB又はCを答えた場合，リピート（反復）・コマンドが実行される。もし，生徒が他の応答をしたならば，正解をプリントし，治療用インストラクションのレスン・フレーム1に戻るようにプログラムされている。
	*B R: THAT WAS THE WIRELESS TELEGRAPH. TRY AGAIN.	
	*C R: BELL WAS THE TELEPHONE MAN. TRY AGAIN.	
	*— C: B: 1	

第6章　CAI言語の設計

6.1　設計方針

ＣＡＩ言語は，前述のとおりシステムの規模や，使用形態，学習形式等によってかなり異なってくるが，ここではなるべく汎用性のある言語を考える。

まず，設計方針を明確にしておく。

1. ＣＡＩの機能を十分に満足すること。
2. 主要学習端末としてキャラクタ・ディスプレイ装置を想定し，各種の周辺装置を自由にコントロールできること。
3. 生徒の学習は，タイム・シェアリング・システムの元で実行できること。
4. 強力なデバック・ファシリティが完備していること。
5. 学習プログラムは，カードから入力でき，オンラインでプログラム修正ができること。
6. 応答解析に十分柔軟性をもたせること。
7. 学習プログラムを分割して作成できること。

以上の要求を満たすFORTRANをベースとした言語を設計した。コンパイラは FORTRAN文をオブジェクトするプリ・コンパイラ形式を採用しているので，FORTRANコンパイラが具備しているデバック・ファシリティを完全に包含し，さらにFORTRAN 言語及びアセンブラ言語で作成したサブルーチン（ユーティリティ）を利用することができる。

システムの流れを図２－１に示す。

6.2　学習プログラムの構成

学習プログラムは，いくつかのセクションから構成され，セクションは，１つのMain Sequenceと補助的Hint Sequenceからなり，さらに学習したい生徒のため，Advance Sequenceを追加することができる。

学習プログラムの最小単位はステップであり，各ステップ毎に生徒の学習経過記録が自動的に取られる。記録内容はシーケンス名（後述），ステップ名（後述），生徒の応答内容，応答所用時間，試行回数等である。

学習プログラムを計算機で実行する場合，Sequence 単位に主記憶装置へローデングする。この様にした理由は，ＣＡＩに於いては，多数の生徒が同時に同じプログラムを利用す

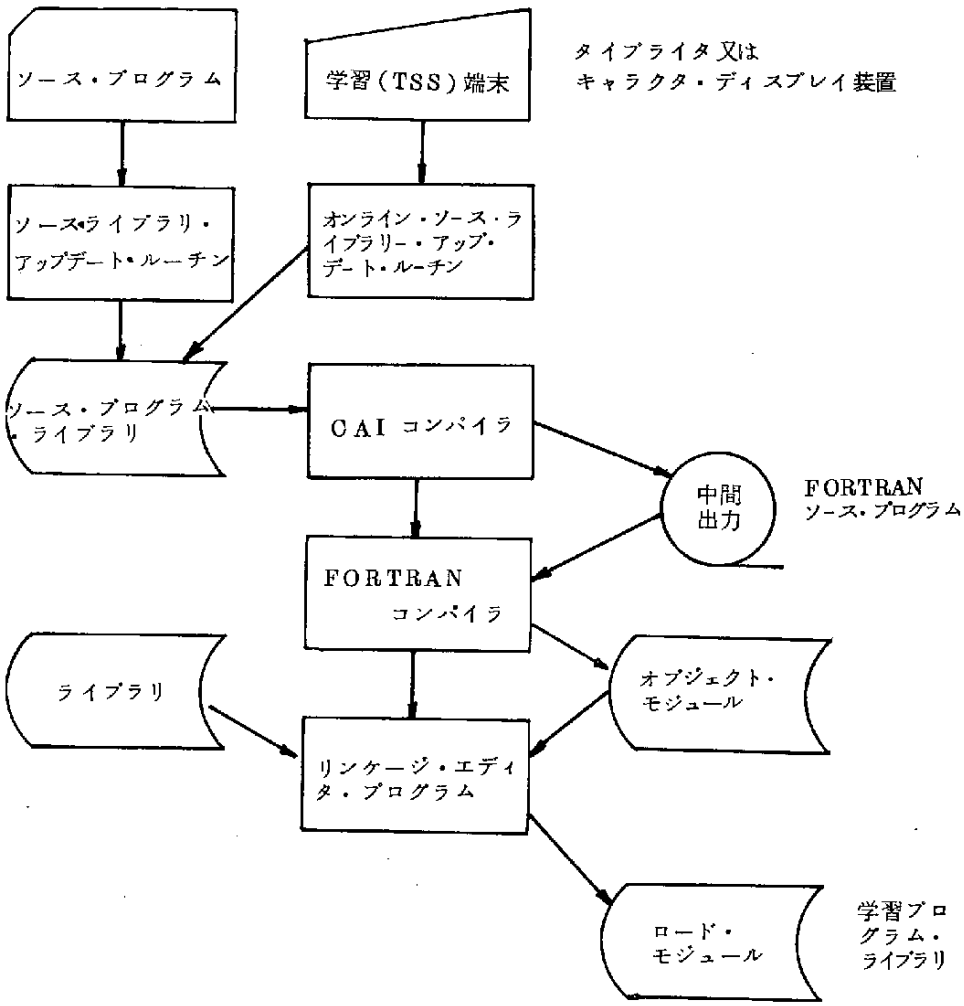


図2－1. システムの流れ

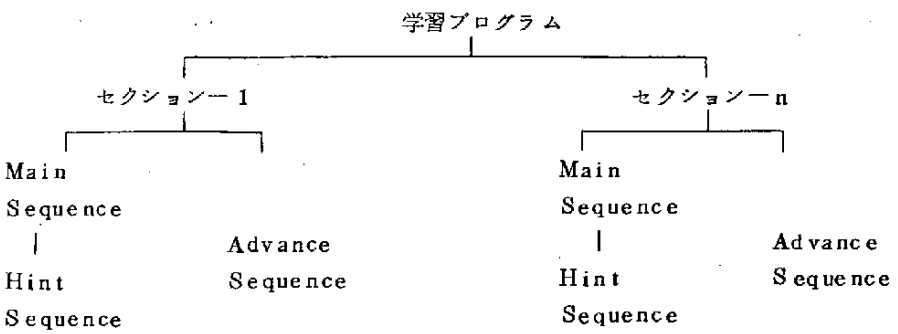


図2－2. 学習プログラムの組織図

るので，リエントラント形式のプログラムがのぞましいが，既製のノンリエントラントなFORTRANコンパイラを利用しているため，同時に複数の生徒が学習を行なうと各生徒(ジョブ)ごとにそれぞれ全プログラムが主記憶にローデングされてしまい主記憶容量をオーバするのでなるべく小さな単位に分割し，すなわちSequence単位にローデングするようにプログラム構造を作成する。

現在のオペレーテング・システムでは，Sequence単位にローデングするためには，リンケージ・エデタで図2－3に示すようなプログラム構造に組立てなければならない。

6.3 Sequenceとステップについて

コンパイルの単位は，Sequenceである。Sequenceは，FORTRAN言語のサブルーチンと同等であり，SEQUENCE文で始まり，END文で終る。これをSequence副プログラムと呼ぶ。

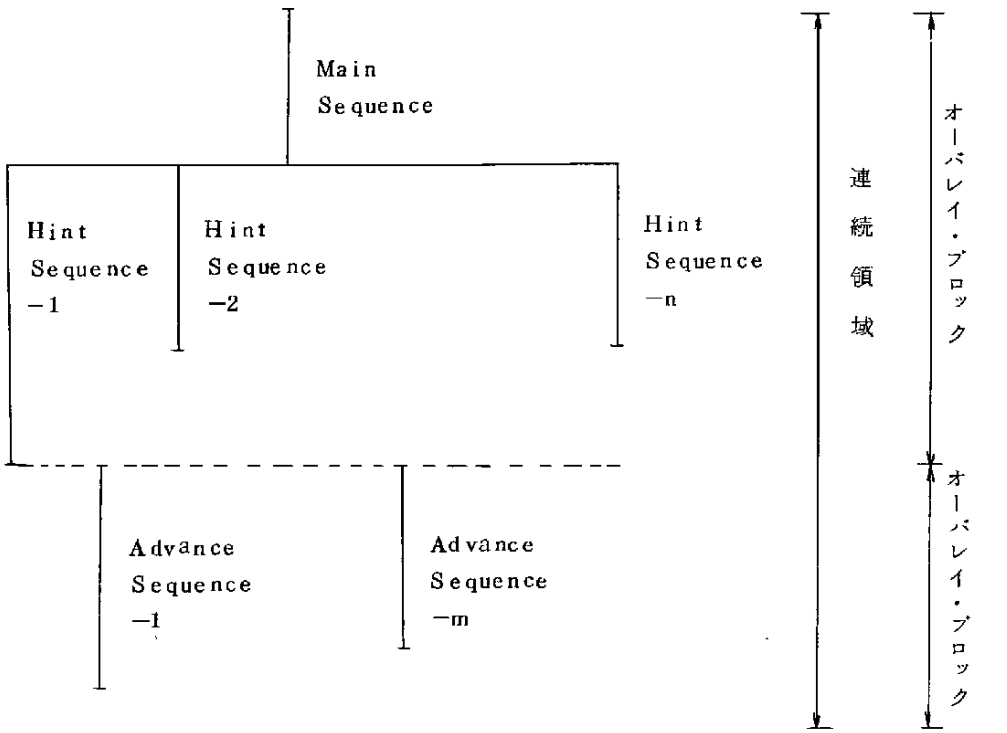


図2－3. プログラム構造

Sequence副プログラムは，いくつかのステップから形成されているとする。

Sequence副プログラムの単位

Sequence副プログラムは，SEQUENCE文で始まり，本文，END文とつづく。これはコンパイルの単位でもあり，またローデングの単位でもある。

SEQUENCE文

SEQUENCE文は次の形とする。

{ MAIN | HINT | ADVANCE } SEQUENCE＜シーケンス名＞

シーケンス名は定義されるシーケンスの英字名とする。

例1. MAIN SEQUENCE ALGBRA

例1は，Main SequenceにALGBRA(Algebra) というシーケンス名をつける。

ステップの単位

ステップは一般に生徒に対する1回の応答関係を記述した文の集まりである。学習プログラム作成者は，前述した如く，ステップ毎に学習経過記録が自動的にとられるので，別の見方からすれば学習経過記録の単位と考えてもよい。

STEP文

STEP文は，ステップの始まりを示し，次のSTEP文か，またはEND文までを1つのステップとすることを指示する。

STEP文は次の形とする。

STEP＜ステップ名＞

ステップ名は，ステップの名前を定義し，Sequence内ではユニークでなければならない。

```
MAIN SEQUENCE ARITH
    •
    •                    } 1つのステップ
STEP ADDITI
    •
    •                    }      〃
STEP SUBTRA
    •
    •                    }      〃
STEP MULTIP
    •
    •                    }      〃
STEP DIVISI
    •
    •                    }      〃
STEP GENERA
    •
    •                    }      〃
END
```

図2－4. SEQUENCE文とSTEP文の例

6.4 言語の詳細

A. 変数と操作

プログラムで利用できる変数は，8種類ある。それらは，バッファ，カウンタ，スイッチ，タイム・カウンタ，アンサ・スイッチ，リクエスト・スイッチ，正誤カウンタの7種類のシステム変数と，プログラム内でユーザが任意に宣言（陰に宣言された変数も含む）できるユーザ変数である。

プログラム作成者は，あたかも1人の生徒と会話しているがごとく変数の全集合を使うことができる。というのは，各々の生徒が学習を開始すると各生徒ごとにプログラムと変数のユニークな集合が割当てられるからである。

システムはステップごとにシステム変数をファイルに記録し，途中から再開するときはシステム変数を中断時点の状態にもどす。

システム変数

1. バッファー： 52字までの文字例を記憶するために使う10個のバッファーがある。文字列は引用符でかこまれている。この場合，引用符もバッファー内に入っているので，正味の文字列は，50字までである。バッファーは，B0，B1，……，B9 という名前で識別される文字形変数である。バッファーB0は，RESPONS文においてバッファー名が省略されたとき，その文によって，暗黙に指定されたかのように使用される。
2. カウンター： 整数値を含むことのできる20個のカウンターがある。その絶対値の範囲は $2^{35}-1$ である。カウンターは，C0，C1，……，C19の名前で識別される整数形変数である。
3. スイッチ： スイッチは，20個あり，trueかfalseのどちらかの状態のみをもつ。スイッチは，S0，S1，……，S19の名前で識別される論理形変数である。
4. タイム・カウンター： カウンターには，3種類ある。一番目は，学習の開始からの全経過時間，2番目は現在のSequenceの開始からの経過時間，そして3番目は，キーボードのロック（RESPONSE文による）から，応答の終り（SENDボタンがおされたとき）までの経過時間をそれぞれ記憶しておく。タイム・カウンタはそれぞれT0，T1，T2の名前で識別される整数形変数である。
5. アンサー・スイッチ： アンサー・スイッチは，2種類あり，RESPONS文による学習者の応答が，ANSリストの項目のどれか1つと一致するかどうかのスイッチと，各項目ごとに一致するかどうかを表わすスイッチである。前者はA0，後者はA1，A2，……，A19

の名前で識別される論理形変数である。

6. リクエスト・スイッチ： リクエスト・スイッチは6種類あり，RESPONSE文による学習者の応答がリクエストを要求しているとき，各々の対応するスイッチにセットされる。スイッチは，R0（又は，HINT），R1(CA)，R2(NEXT)，R3(REV)，R4(CALL)，及びR5(OK)という名前で識別される論理形変数である。

R0は，学習者がヒントを要求した場合

R1は，学習者が正解を要求した場合

R2は，学習者が次を要求した場合

R3は，学習者があと戻りを要求した場合

R4は，学習者がコールを要求した場合

R5は，学習者が完了を要求した場合

を表わす。

7. 正誤カウンタ： 正誤カウンタは，正解答カウンタと誤解答カウンターの2種類あり，各々は，学習の開始からの正解答及び誤解答の回数をシステムによって計数する。それぞれW0（又はCC……correct counter）とW1（WC……wrong counter）の名前で識別される整数形変数である。

ユーザ変数

ユーザ変数は，システム変数以外の変数で学習プログラム作成者が，FORTRAN言語の規則内で自由に宣言（暗黙の宣言も含む）して使うことができる。変数名は，システム変数と同じ名前を使ってはならない。

算術式

算術式は，算術演算子と変数又は定数を含む式である。算術式を使う目的は2つある。1つは，数値変数を組合わせて計算し，たぶん他の変数に結果を格納したり，あとで使用するためそれらを記録しておく。もう1つは，変数の値の計算結果を利用して，バッファー，カウンター，あるいはスイッチを選ぶ場合である。

算術演算子

算術演算子は右のものとする。

演算子	意味
+	加算，正符号
−	減算，負符号
*	乗算
/	除算
**	ベキ乗

算術式は，FORTRANと同様に，演算子の優先順位に従って評価される。式の実行順序はベキ乗，乗算又は除算，そして最後が加算又は減算である。式は左から右へ評価され，しかしカッコが使われている場合は，カッコ内を先に評価する。

関　係　式

関係式は，二つの算術式を関係演算子によってつないだものとし，その関係式が成り立つか成り立たないかに応じて，それぞれtrueまたはfalseの値をもつ。

関係演算子

関係演算子は，つぎのものとする。

演　算　子	意　　味
.LT.	小さい（less than，<）
.LE.	小さいか等しい（less than or equal to,≦）
.EQ.	等しい（equal to，＝）
.NE.	等しくない（not equal to，≠）
.GT.	大きい（greater than，>）
.GE.	大きいか等しい（greater than or equal to，≧）

論　理　式

論理式は，関係式，論理定数及びスイッチのいずれであるか，これらを論理演算子でつないだものとし，true（correct）または，false(wrong) の値を持つとする。

演　算　子	意　　味
.OR.	論　理　和
.AND.	論　理　積
.NOT.	否　　定

論 理 定 数

論理定数は論理型定数で，.TRUE.又は.CORRECT.と.FALSE.又は.WRONG.の二つがあり，それぞれ真および偽の値をもつ。

論理定数	意　味
.TRUE.	真
.CORRECT.	真
.FALSE.	偽
.WRONG.	偽

文　字　式

文字式は，文字演算子と文字列変数を含む式である。

文字列は，引用符でくくられている。文字列変数に記憶されている文字列の有効な部分は，最初の引用符から次の引用符でくくられた文字列である。

文字演算子

文字演算子は次のものとする。

演算子	意　味
||	連　結

例えば，

B1 || ' AM␣' || ' A␣' || B2

でB1に ' I␣' ，B2に ' BOY ' が入っていれば，結果は，

' I　AM　A　BOY '

となる。

変数のアドレス計算

システム変数の名前は，2つの部分，文字（B，C，S，T，A，W）と数字（0，1，……，19）から構成されている。文字は変数のタイプを示し，数字は変数のアドレスを示す。変数を参照する普通のもっとも直接な方法は，前で述べた識別名B0，B9，C0，C19，S0，S19，T0，T1，A0，A19，R0，R5，W0，W1，によってそれを示す。

システム変数を参照する別の方法は，変数の識別名のアドレスの位置を算術式として計算し，それをカッコでくくり，変数のタイプを表わす文字の後に置いて表わす。例えばB（C1－1）は，カウンタ1の内容によってバッファ0～9の内の1つを選ぶ。もし，C1－1の式の値が0～9の範囲を越えた場合は，エラーとなる。

代　入　文

代入文は，算術代入文，論理代入文及び文字代入文の3種類ある。

算術代入文

算術代入文は，つぎの形とする。

v ＝ e

ここで，vは論理形及び文字形以外の変形名または配列要素名とし，eは算術式とする。この文の実行によって式eが評価され，その値がvに割当てられるものとする。

論理代入文

論理代入文は，つぎの形とする。

v = e

ここでvは，論理形の変数名または配列要素名とし，eは論理式とする。

この文の実行によって式eが評価され，その値がvに割当てられるものとする。

文字代入文

文字代入文は，つぎの形とする。

v = e

ここで，vは文字形の変数名または配列要素名とし，eは文字式とする。この文の実行によって式eが評価され，その値がvに割当てるものとする。

B. 各種の文

この言語は，一般的な文の書き方は，FORTRAN文法規則に準拠している。ここでは，FORTRAN言語に追加された文についてのみ記す。

1. DEFINE文

この文は，指定文の一種で，使用する周辺装置のデータ・セット定義番号，付加情報を記述する。

一般形は，DIFINE<周辺装置名>d(r)　である。

次に各周辺装置ごとに説明する。

(1) スライド装置

DEFINE SLIDE d(r)

d：スライド装置のデータセット定義番号で，標準は1とする。

r：符号なしの整数で，dで指定されたデータセットに於けるスライドの枚数を指定する。省略した場合は，スライド装置のマガジンの最大収容枚数とする。

注) 現在のシステム構成においては，スライド装置はキャラクタ・ディスプレイ装置でシュミレートする。

(2) 音声装置

DEFINE AUDIO d(r)

d：音声装置のデータセット定義番号で，標準は2とする。

r：符号なしの整数で，dで指定されたデータセットにおける音声のコマ数を指定する。省略した場合は，無限大とみなす。

(3) CRTディスプレイ装置

DEFINE DISPLAY d(y,x)

d：CRTディスプレイ装置のデータセット定義番号で，標準は3とする。

y：符号なしの整数で，dで指定されたデータセットに於ける最大行数を指定する。省略した場合は，20とみなす。

x：符号なしの整数で，dで指定されたデータセットに於ける最大列数を指定する。省略した場合は，50とみなす。

(4) タイプライタ装置

DEFINE TYPE d(x)

d：タイプライタ装置のデータセット定義番号で，標準は4とする。

x：符号なしの整数で，dで指定されたデータセットに於ける1行分の文字数を指定する。

注）現在のシステム構成では，タイプライタ装置は，TSS端末装置と同一であるとする。

(5) カード読取装置

DEFINE READER d(x)

d：カード読取装置のデータセット定義番号で，標準は5とする。

x：符号なし整数で，dで指定されたデータセットに於けるカードの最大カラム数を指定する。省略した場合は80とみなす。

(6) ラインプリンタ装置

DEFINE PRINTER d(x)

d：ラインプリンタ装置のデータセット定義番号で，標準は6とする。

x：符号なし整数で，dで指定されたデータセットにおけるラインプリンタの行の最大文字数を指定する。省略した場合は，133とみなす。

(7) カードさん孔装置

DEFINE PUNCH d(x)

d：カードさん孔装置のデータセット参照番号で標準は7とする。

x：符号なし整数で，dで指定されたデータセットにおけるカードの最大カラム数を指定する。省略した場合は，80とみなす。

(8) 補助記憶装置

DEFNIE FILE d(x)

d：補助記憶装置のデータセット参照番号で一般には8又は9が使われる。

r：符号なしの整数でdで指定されたデータセットにおける最大レコード数を指定する。省略した場合は，無限大とみなす。

以上をまとめてバッカス記法で示す。

<DEFINE文>::＝DEFINE<周辺装置名><データセット定義番号>
(<周辺装置の属性>)

<周辺装置名>::＝SLIDE | DISPLAY | AUDIO | TYPE | READER |
PRINTER | PUNCH | FILE

<データセット定義番号>::＝<符号なし整数>

<周辺装置の属性>::＝<符号なし整数> | <符号なし整数>，<符号なし整数>

2. スライド指示文

<スライド指示文>::＝SLIDE(<データセット参照番号>，<スライド参照番号>)

<データセット参照番号>::＝<添字>

<スライド参照番号>::＝<添字式>

<添字式>::＝<添字> | <添字><加減算記号><添字>

<添字>::＝<単純変数> | <符号なし整数>

<加減算記号>::＝＋ | －

この文は，指示されたスライド参照番号のスライドをスライド装置でスクリーンに映写する。現在のシステム構成では，指定されたスライド番号に相当するレコードをファイルから読みこみ，キャラクタ・ディスプレイに表示する。

例1. SLIDE(1，10)

例1は，第10番目のスライドをスライド装置（＃1）でスクリーンに映写する。

3. 表示文

<表示文>::＝DISPLAY(<データセット参照番号>，<位置指定>)
〔<表示リスト>〕

<位置指定>::＝<行指定>〔，<列指定>〕

<行指定>::＝<添字式>

<列指定>::＝<添字式>

<表示リスト>::＝<表示セグメント> | <表示リスト>，<表示セグメント>

<表示セグメント>::＝<システム変数> | <添字付きシステム変数> | <文字列>

この文は，キャラクタ・ディスプレイ装置の指定された位置に表示リストの内容を表示する。列指定が省略された場合は，列指定が1とみなされる。

例1.　DISPLAY(3,I,J)'REPONSE␣TIME␣=',T2,'␣SEC'

例1は，キャラクタ・ディスプレイ装置(#3)の第I行目の第J列目から表示リストを表示する。例えば，I=5，J=3，T2=100 ならば図2－5の様に表示される。

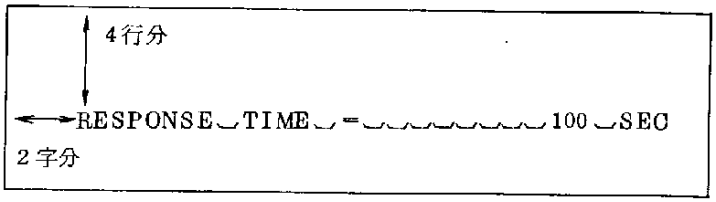


図2－5.　キャラクタ・ディスプレイ画面

4.　消去命令

＜消去命令＞::＝ERASE(＜データセット参照＞〔,＜位置指定＞〕)

この文は，キャラクタ・ディスプレイ画面の消去命令で，位置指定が省略された場合は全画面の消去を意味し，位置が指定された場合は，指定された位置から，その行の終りまでを消去する。

例1.　ERASE(3)

例2.　ERASE(3,10)

例3.　ERASE(3,10,5)

例1は全面消去を表わし，例2は10行目を消去することを表わし，例3は10行目の5列目からその行の終りまでを消去することを表わす。

5.　ヘッデング・マーク表示命令

＜ヘッデング・マーク表示命令＞::＝MARK(＜データセット参照番号＞,
＜位置指定＞)

この文は，指定された位置にヘッデング・マークを表示する。ヘッデング・マークは，一般には，キャラクタ・ディスプレイ装置からデータを読みこむときの位置を示すのに使われる。

例1.　MARK(3,20)

例 2.　MARK(ICRT, M, N)

6.　分割命令

<分割命令>::＝SPLIT(<データセット参照番号>, <位置指定>)

この文は，キャラクタ・ディスプレイ装置の画面を分割し，指定した位置の上半分をCPU領域とし，下半分をEDIT領域とする。

例 1.　SPLIT(3, 10)

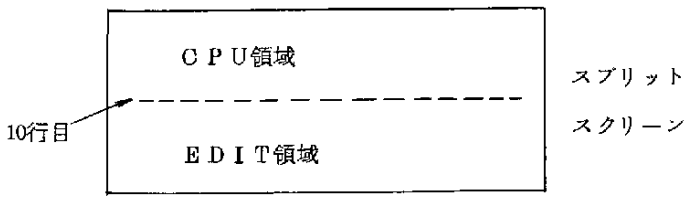


図2－6.　スプリット・スクリーンによる分割

例 1.は，10行目の上半分をCPU領域とし，下半分をEDIT領域とする。カーソルがEDIT領域にある場合，オペレータ（生徒）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領域はオペレータによって書き替えられる事はない。

7.　応答文

<応答文>::＝RESPONSE(<データセット参照番号>,〔<パラメータ参照番号>〕〔,<位置指定>〕)〔<バッファ名>〕

<パラメータ参照番号>::＝<パラメータ文につけたステートメント番号>

<バッファ名>::＝B0 | B1 | …… | B9

この文は指定された位置（位置指定が省略されている場合は，カーソルよりも前にあって一番近いヘッデング・マーク）から，生徒の応答を指定されたバッファ（省略された場合はB0）によみこみ，パラメータに従ってアンサー・スイッチ，リクエスト・スイッチ等をセットし，かつ応答時間をタイム・カウンタT2にセットする。

パラメータ参照番号が省略された場合，指定されたバッファーに読みこみだけ行なう。

例 1.　RESPONSE(3, 10, 5, 3)　B2

例1は，キャラクタディスプレイ装置の5行目の3列目から生徒の応答を読みこみ，ステートメント番号10の書式に従って各種スイッチと応答時間をT2にセットする。

8.　位置づけ文

<位置づけ文> :: = FIND (<データセット参照番号>, <レコード番号>)

<レコード番号> :: =<添字式>

この文は，指定されたデータセット参照番号の装置に指定されたレコード番号のレコードをすぐに読んだり，映写したりできる状態に位置づけする。

例1. FIND(1,10)

例1はデータセット参照番号1の装置，すなわち，標準データセット番号ならばスライド装置の第10番目のスライドをすぐ映写できる状態に位置づけする。

この文は，他の文とちがって，命令を実行すると命令の完了をまたずに次の文を実行する。

9. 待ち時間文

<待ち時間文> :: =WAIT (<待ち時間>)

<待ち時間> :: =<添字式>

この文は，待ち時間を指定する文で，単位は秒である。

例1. WAIT(100)

例1は，100秒間プログラムの実行を待つ。

10. 音 声 文

<音声文> :: = PHONE (<データセット参照番号>, <ナレーション番号>)

<ナレーション番号> :: =<添字式>

この文は，指定されたデータセット参照番号の音声装置から指定されたナレーション番号のナレーションを出す。

11. タイプ文

<タイプ文> :: =TYPE (<データセット参照番号>) 〔<タイプ・リスト>〕

<タイプ・リスト> :: =<タイプのセグメント> | <タイプ・リスト>, <タイプのセグメント>

<タイプのセグメント> :: =<システム変数> | <添字つきシステム変数> | '<文字例>' | <改行記号>

<改行記号> :: =／

この文は，指定されたデータセット参照番号のタイプライタ装置に，タイプ・リストを印字する。

例1. TYPE(4) ／,'GOOD␣BYE␣', B19

例1は，バッファ19に'ＴＯＭ'が格納されていれば，タイプライタ装置から改行して

GOOD␣BYE␣ＴＯＭ

と印字される。

12．パラメータ文

＜パラメータ文＞::＝PARAM(＜パラメータ・リスト＞)

＜パラメータ・リスト＞::＝＜パラメータのセグメント＞|

＜パラメータ・リスト＞，＜パラメータのセグメント＞

＜パラメータのセグメント＞::＝ＡＮＳ(＜解答リスト＞)｜ＲＥＱ(＜要求リスト＞)

＜解答リスト＞::＝＜解答のセグメント＞|＜解答リスト＞，＜解答のセグメント＞

＜解答セグメント＞::＝＜解答の種類＞'＜解答列＞'

＜解答の種類＞::＝〔Ｃ｜Ｗ〕

＜解答列＞::＝Ａ：＜文字列＞｜Ｉ：＜整数＞｜Ｒ：＜実数＞

＜要求リスト＞::＝＜要求のセグメント＞|＜要求リスト＞，＜要求のセグメント＞

＜要求のセグメント＞::＝¥ＨＩＮＴ｜¥ＣＡ｜¥ＮＥＸＴ｜¥ＲＥＶ｜¥ＣＡＬＬ｜¥ＯＫ

この文は，応答文のパラメータを指定する文である。解答リストは予想される解答を列記し，要求リストは，許される要求を列記する。解答の種類のＣは正答（ｃｏｒｒｅｃｔ ａｎｓｗｅｒ）を表わし，Ｗは誤答（ｗｒｏｎｇ ａｎｓｗｅｒ）を表わし，省略された場合は，正，誤答のどちらでもないとみなす。解答列のＡは生徒の応答を文字列とみなし文字列で比較することを表わし，Ｉは生徒の応答を整数とみなし，整数として比較し，Ｒは実数とみなして比較することを表わす。

要求のセグメントの

¥ＨＩＮＴ　は，生徒がヒント　要求する。

¥ＣＡ　　は，〃　正　解　〃

¥ＮＥＸＴ　は，〃　次　を　〃

¥ＲＥＶ　　は，〃　あと戻り　〃

¥ＣＡＬＬ　は，〃　コール　〃

¥ＯＫ　　は，〃　確　認　〃

ことを許すことを表わす。

例１　PARAM(ANS(C'A:NEWTON', W'A:EULER'), REQ(¥CA))

例２.　PARAM(ANS(C'I:10', C'R:10.0)

例1は，生徒の応答が，文字列としてＮＥＷＴＯＮならば正答で，ＥＵＬＥＲか又はそれ以外の場合は誤答であることを示し，要求としては，正答を要求することのみ許す。

例2は，生徒の応答として，整数でも実数の10でも正答とする場合の例である。

昭和46年5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21号地1番5
機械振興会館内
TEL(434)8211(代表)
印刷所　有限会社　秀嶺社
東京都新宿区戸塚町四丁目六二五番地
電話(364)2448
45-S002