NAL TM-479

ISSN 0452-2982
UDC 681.323

航空宇宙技術研究所資料

TECHNICAL MEMORANDUM OF NATIONAL AEROSPACE LABORATORY

TM-479

汎用飛行シミュレータ用ディジタル演算部(FSK-II)のプログラム言語の改造(RTSL-II)とそのライブラリ

渡辺　顯・若色　薫

1982年11月

航空宇宙技術研究所
NATIONAL AEROSPACE LABORATORY

目　　次

汎用飛行シミュレータ用ディジタル演算部(FSK-II)のプログラム言語の改造(RTSL-II)とそのライブラリ*

渡辺　顯**　若色　薫**

1. まえがき

本報告では，航技研汎用飛行シミュレータの動特性模擬装置ディジタル演算部(FSK-II)で用いるプログラム言語(RTSL)に関し，改造を行ったもの(RTSL-II)について述べる。ここで，FSK-IIは従来のアナログ計算機をミニコン(三菱製M-70)5台のディジタル計算機システムに更新した装置である。[1] 更新当時(昭和48年～50年)は，航空機の飛行シミュレータの実時間演算部にディジタル計算機を導入するのが我が国で始めてであったため，多くの困難を伴った。一般に，ディジタルシステムはソフトウェアの出来具合で，システム全体の性能が左右されるため，ソフトの設計は難しく，かつ作成後も改良作業を多く伴う。本装置においても，ソフトウェアの評価に多くの時間をかけ，かつ全面的に改訂を行うことによってシステムの性能向上を図ってきている。

RTSL(Real Time System Language)は，FSK-IIを用いて，実時間シミュレーションを行うプログラムを作成するための言語であり，コンパイラをさす。

特徴としては，高級言語仕様であるが，固定小数点演算を比較的容易な手法で記述できるようになっている。このRTSLを改造し，FSK-IIのハード構成に合わせて作り直したものがRTSL-IIである。

以下，この改造点を中心に述べる。なお，RTSL-IIの基本構成は，主としてRTSLのものから成っている。文法規則等全体の詳細については，付録1，2に示してある。(付録の内容は改造後のもの，すなわちRTSL-IIのもので，新たに加えた機能についても全て記述してある)

2. RTSL-IIの特徴と構成

2.1 RTSL-IIの特徴

RTSL-IIはまえがきでも述べた様に，FSK-IIを用いて実時間シミュレーションを行うプログラムを作成するための言語であり，コンパイラである。また，RTSLを評価し，FSK-IIのハード構成に適したものに改造しているので，基本的構成はRTSLと同様である。

RTSL-IIの特徴で，かつRTSLとの大きな違いは，

① 演算式としてFSK-IIで実行可能で，実時間演算に必要な機能のみに限定している。

② コンパイル時の操作性を向上し，コンパイル処理時間の短縮をはかっている。

③ 出力オブジェクトのディスクファイル化を可能にしている。

等である。これらの細かい違いは表1に示してある。また，組み込みライブラリも積分を始め，実時間演算に必要と考えられる種々のものを用意した。

2.2 RTSL-IIの構成

RTSL-IIの基本構成を図1に示す。内部は4段階に分かれている。

(1) 第1段階：アナライザ

この段階はソースプログラムをカードリーダから読み込み，文法チェックを行い，演算順序，各変数の属性(付表5参照)を考慮し中間言語へ変換するこの際にソースリストがラインプリンタに出力される。

* 昭和57年9月7日　受付
** 計測部

表1 RTSL－ⅡとRTSLの相異点

番号	項目	RTSL－Ⅱ	RTSL
1	起動	• IPLから直接	• SCPによる
2	使用計算機	• 任意の1台(32kW)	• SCP用 1台(№φ) • RTSL用 1台
3	モニタ	• BOM－Ⅱ	• BOM改
4	入力媒体	• カード	• カード，紙テープ
5	出力媒体	• ディスク，カード	• 磁気テープ，紙テープ
6	出力ディスクファイル	• 可 (FMPの併用)	• 不可
7	文法エラー時	• アセンブルせず	• 全て実行
8	バッチ用プログラム生成	• 不可	• 可(但し実行はBOM改のみ)
9	演算	• 固定小数点演算 • ビット演算	• 固定小数点演算 • ビット演算 • 浮動小数点演算
10	シンボルテーブル処理	• 他計算機メモリ使用可	• 自身のメモリのみ
11	データ	• 固定小数点データ • ビットデータ	• 固定小数点データ • ビットデータ • 浮動小数点データ
12	入出力変数定義	• INPUTサブプログラム • OUTPUTサブプログラム	• EXTERNALサブプログラムと＝の組合せ
13	非実行サブプログラム有効範囲	次の1つの実行サブプログラムのみ	次に続く実行サブプログラム全て
14	定数属性定義	INIT, DATA文内の定数は省略可	全ての定数はすぐ後で定義する必要あり
15	インラインアセンブル	不可	可
16	ライブラリ(ファンクション サブルーチン)	基本 15種 外付け 29種 (固定小数点用のみ)	基本組込み 28種 (内固定小数点用 14種)
17	乗算小数点位置	$J(M)*J(N)\rightarrow J(M+N-15)$ (他はRTSLと同じ)	$J(M)*J(N)\rightarrow J(M+N-16)$

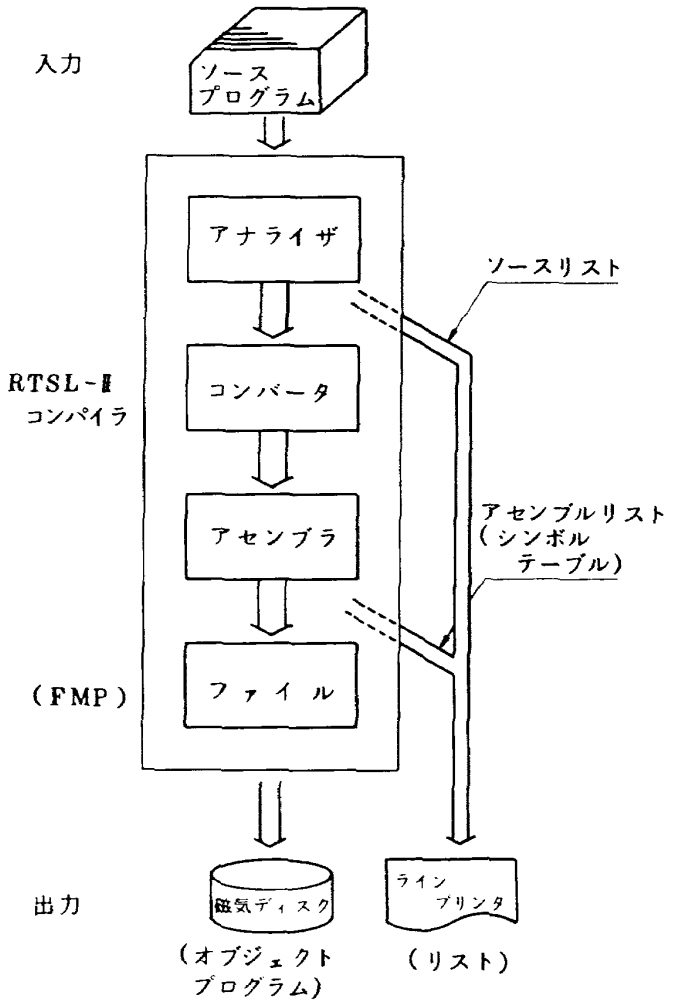


図1 RTSL-Ⅱの基本構成

この段階では，文法チェックのほか，未定義シンボルのチェックも行われ，エラーがある場合には，ソースリストと共にエラー情報が出力される。

また，中間言語はワークエリアとして使用している磁気ドラム記憶装置に格納される。

(2) 第2段階：コンバータ

第1段階で変換された中間言語によるプログラムを，次の段階のアセンブルが行えるように，アセンブラ言語への変換が行われる。

(3) 第3段階：アセンブル

前段階で作成されたアセンブラソースプログラムをM-70計算機のアセンブラでアセンブルし，アセンブルリストをラインプリンタに，オブジェクトプログラムをディスクのワークエリアに出力する。

(4) 第4段階：ファイル

ワークエリアに出力されたオブジェクトプログラムをファイル化するために専用のプログラム(FMP)をドラムから引き出す。オブジェクトプログラムのFMPによるファイル化が終了した場合には，テクトロディスプレイにファイル化メッセージが出力される。

このFMPの終了後は自動的に第1段階に復帰するようになっており，ユーザが再度RTSL-Ⅱを起動する必要はない。

これらの一連の作業は1つのタスク，サブルーチン，あるいはファンクションごとに行われる。

以上の段階をへて作成されたプログラムは，FSK-Ⅱの複合計算機システムに適した形態に編集され，計算機にロードされて，実時間シミュレーションに用いられる。この編集，ロード，及び実行には別のプログラム(LEP-Ⅱ, SCP-Ⅱ)が用いられる。

3. RTSL—Ⅱの文法規則

RTSL-Ⅱの文法規則は基本的には，RTSLのものをそのまま採用している。

相異点の主なものは，

① 入出力変数の表記法を入力用と出力用に分離
② 浮動小数点演算の削除
③ インラインアセンブル表記の禁止
④ 定数の属性定義記述の一部省略

等である。

従って①④以外はRTSLの規則と同様であるため，RTSL文法規則を参照すれば良いが，RTSL-Ⅱでは削除したものが多々あるので，付録1にRTSL-Ⅱの文法規則を記す。ここでは，特に大きく変化した入出力変数定義についてのみ述べる。

(1) プログラム構成

プログラムには目的の演算を行うものとして，TASKサブプログラムとSUBROUTINEサブプログラム，FUNCTIONサブプログラムの3つがある。

これらのサブプログラムは，通常ある仕事ごとに作られる。この時そのサブプログラムに関連する入出力変数をコンパイラに知らせることが必要となる。

従来はこのためにEXTERNALサブプログラムで定義するようになっていた。これは入出力を一括し，入力と出力の違いは，TASK等の実行サブプログラム内の等号(=)の左辺にある変数を出力変数とする方式であった。この方式は入出力変数がわかり

にくいこと，コンパイル時の処理時間が増すこと，さらにデータ領域を確保するために無意味な式（＝を用いる）を加えることが必要となる欠点があった。

このため，RTSL－Ⅱでは，入力変数の定義をINPUTサブプログラムで，また出力変数の定義をOUTPUTサブプログラムで記述することにより，ユーザの記述を容易にし，コンパイル時間の短縮をはかった。

(2) 記述法

図2に各サブプログラムの記述例を示す。なお，各プログラム内の文の意味については付録1に示してある。

(3) 定数の属性定義

RTSL(RTSL－Ⅱ)は固定小数点演算を基本と

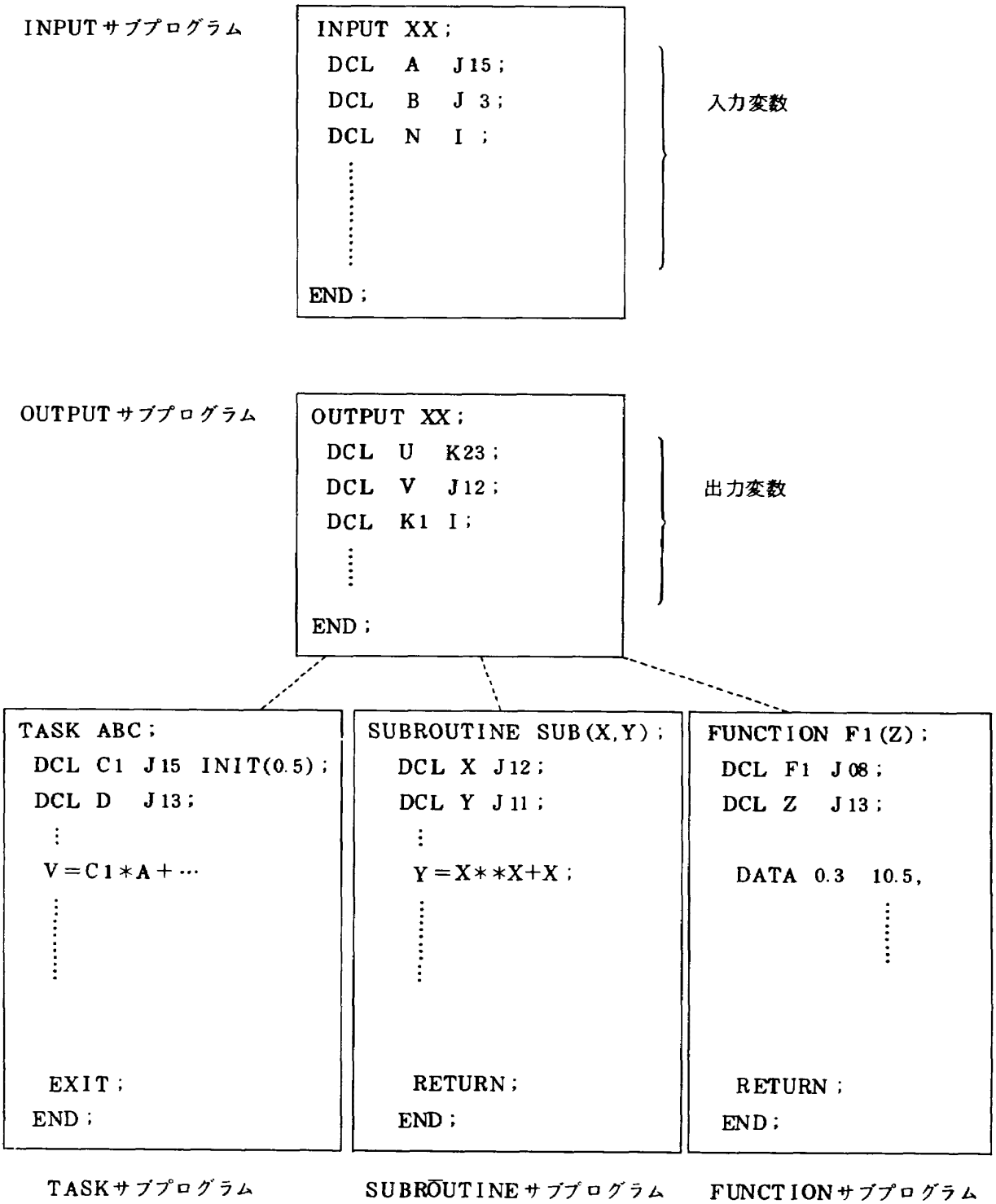


図2 サブプログラム記述例

表2　RTSL-Ⅱライブラリ（ファンクション）

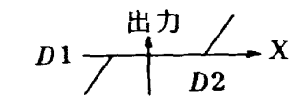
	目的	表記法	引数の属性	結果の属性	備考（＊はユーザプログラムで属性セット）	
1	正弦	SIN(X) SINφ(X) SIN：(X,P)	J14 J14 12≦P≦15	J14 J15 J15	引数の単位 rad	 ＊ ＊
2	余弦	COS(X) COSφ(X) COS：(X,P)	J14 J14 12≦P≦15	J14 J15 J15	引数の単位 rad	 ＊ ＊
3	正接	TAN：(X,PI,PO)	12≦PI≦15	9≦PO≦15	PIは引数Xの小数点位置 POは関数の小数点位置	＊
4	逆正弦	ASIN(X) ASINφ(X) ASIN:(X)	J14 J15 J14	J14 J15 J14	|X|<+1 結果の単位 rad	＊ ＊ ＊
5	逆正接	ATAN(X) ATAN：(X,PI,PO)	J14 9≦PI≦14	J14 9≦PO≦14	結果の単位 rad PIは引数の小数点位置を，POは関数の小数点位置を与える。	＊
6	指数関数	EXP(X)	J14	J13		
7	自然対数	LOG(X)	J11	J13		
8	平方根	SQRT(X)	J(2n)	J(n+7)	nは整数	
9	絶対値	ABS(X)	J(n) J(n)	J(n) K(n)	引数と結果の属性は同じ	
10	符号	SIGN(X)	任意	J15	X≧0 …SIGN(X)=/7FFF X<0 …SIGN(X)=/8000	＊
11	剰余	MOD(X,I)	引数は同一	引数と同じ	X/I＝Nなる整数値を出す。	
12	最大値	MAX(A1,A2,…,AN)	引数は同一	引数と同じ	A1～ANの中の最大値が選ばれる。	
13	最小値	MIN(B1,B2,…,BM)	引数は同一	引数と同じ	B1～BMの中の最小値が選ばれる。	
14	制限	LIMIT(X,L1,L2)	引数は同一	引数と同じ	L1≦出力≦L2	
15	不感帯	DBND(X,D1,D2)	引数は同一	引数と同じ		
16	乱数	RANDOM(X)	J15	J15	擬似一様乱数	
17	ビットサーチ	TRANSφ(X) TRANS1(X)	Xのビット列の中でφまたは1の最上位の位置を符号ビット（最上位）からのずれで示したもの		φまたは1が無いときは，−1@Iがセットされる。	
18	1変数関数発生	FG1L(X,Xφ,Yφ,N)	X,Xφ同一Jタイプ Yφ：Jタイプ N：Iタイプ	Yφと同一Jタイプ	X：入力変数 Xφ：データのX座標先頭アドレス Yφ：データのY座標先頭アドレス N：ブレークポイント数	

表3 RTSL-Ⅱライブラリ（サブルーチン）

	目的	表記法	引数の属性	意味
1	パック	CALL PACK(W1,S)	全てIタイプ	W1を先頭とする16ワードの内容（ϕか$\neq\phi$）によりSのビットをϕか1にセットする。
2	アンパック	CALL UNPACK(S,D1)	全てIタイプ	Sの各ビットの内容に応じてD1を先頭とする16ワードにϕか1をセットする。
3	BCD変換	CALL B:BCD(B,BCD)	全てIタイプ	Bなる1ワードバイナリデータを4BITずつのBCDコードに変換する。
4	BCD逆変換	CALL BCD:B(BCD,B)	全てIタイプ	BCDなるBCDコードデータを1ワードのバイナリデータに変換する。
5	オイラ積分	CALL EULJ(X,XD) CALL EULK(X,XD)	全て同一 Jタイプ Kタイプ	$X=\int XDdt$ $X=X+T*XD$
6	オイラ積分（内部分割タイプ）	CALL EULJN(X,XD,N)	X,XDは同一Jタイプ NはIタイプ	$X=X+T/N*XD$
7	アダムス2次積分	CALL AB2J(X,XD,W) CALL AB2K(X,XD,W)	全て同一Jタイプ 全て同一Kタイプ	$X=X+\frac{T}{2}(3XD-W)$ （Wはワーク，初期値 $\dot{x}_{-1}$）
8	アダムス2次積分（内部分割タイプ）	CALL AB2JN(X,XD,W,N)	X,XD,Wは同一でJタイプ NはIタイプ	$X=X+\frac{T}{2N}(3XD-W)$ （Wはワーク，初期値 $\dot{x}_{-1}$）
9	一次遅れ	CALL LAG:1(U,X,T,P)	U,X…同一Jタイプ T…Jタイプ P…Iタイプ（9～15）	U → $\frac{1}{1+TS}$ → X P…Tのポイント位置
10	擬似正規乱数	CALL GAUSS(V,N)	V…J15 N…Iタイプ	V…正規乱数（3σ） N…一様乱数発生初期値（奇数）

しているため，0.5とか3.6等の定数をセットする場合は，0.5@J15，3.6@K25のようにすぐ後でその属性と小数点位置を定義しなくてはならない。この属性定義はややわずらわしい面もあるのでRTSL－Ⅱでは，初期値定義時と，関数サブプログラム内のDATA文（付録1の1－4(9)参照）の記述において省略できる様に改造した。

初期値定義では，

DCL　A　J15 INIT(0.5)；

と記述し，0.5は0.5@J15と等価である。また，DATA文では，FUNCTION（付録1－4(9)参照）サブプログラムで多数の数値を数値のみの記述が可能となる様にした。

4. RTSL－Ⅱのライブラリ

RTSL-Ⅱのライブラリは，従来のRTSLが持っていたもののほか，新たに追加したものより構成されている。また，従来からあるものでも，演算時間の短縮をはかったもの（実時間演算では重要な要因となる），演算精度を上げたものも含まれている。

ライブラリには次の2種類がある。

① ファンクション

② サブルーチン

各プログラムの種類について表2，3に名称，記述法，引数，属性等を示す。

使用時は，引数の順序，属性等を合わせて用いることが重要である。

各ファンクション，サブルーチンの目的，使用法，プログラム容量，実行時間等の詳細については，付録4に示してある。

5. RTSL－Ⅱの実行

RTSL-Ⅱは従来のRTSLと異なり，IPLプログラム[注)]により，ドラムから計算機に直接引き出される。従って計算機としては1台の計算機(32kW)で実行しえる。（従来は一旦SCPを引き出し，これにより起動された。）

5.1 RTSL－Ⅱの使用ハードウェア

RTSL-Ⅱは2章で示した各段階をへて，プログラムのコンパイル，ファイル化が行われる。このために必要なハードウェアを表4に示す。ユーザは，FSK-Ⅱの周辺装置を使用計算機に接続して実行する。

もし，非常に大きなプログラムで，その中に含まれるシンボル（変数，定数等の名札）の数が100個以上と見込まれる場合は，シンボル処理時間を短縮するために，他の計算機のメモリをワークエリアとして用いることができる。この場合は，他計算機の電源を入れダミーランさせておくことが必要である。

表4に示したものの中には，ユーザの目的によっ

表4　RTSL-Ⅱ使用ハードウェア

	ハードウェア名	使用目的
1	計算機（32kW）	RTSL-Ⅱ 動作用
2	計算機（2台目）	シンボルテーブル用（大容量プログラム時に処理速度向上）
3	ドラム No.1	RTSL-Ⅱ コンパイラ格納
4	ドラム No.2	一時記憶，ワーク用
5	ディスク（Unit 1）	オブジェクトプログラム出力，ファイル化
6	カードリーダ	ソースプログラム，RTSL-Ⅱコマンド入力用
7	ラインプリンタ	リスト出力用
8	テクトロディスプレイNo.1	RTSL-Ⅱ起動用，FMPメッセージ用
9	カードパンチ機	オブジェクトプログラム出力用　（必要時）

注）（Initial Program Loader）計算機を初期スタートするプログラム

て用いないものもある。たとえば，コンパイルのみ行い，オブジェクトプログラムをファイル化しない場合は表4の5のディスクは必要としない。もちろん9のカードパンチ機も必要としない。

この様な選択は付録2に示すRTSL－Ⅱのコマンドを選んでソースカードに加えることによってなされる。通常は，ソースリストとオブジェクトプログラムのディスクファイル化が行われるため，RTSL－Ⅱではこの場合を想定して，コンパイラ内部に初期設定がしてある。従って，ユーザが特に指定しなくともよい。

5.2 RTSL－Ⅱの起動法

初期に作成されたRTSLは，FSK-Ⅱを管理していたSCP(System Control Program)によって起動されていた。このためSCPの動作のために1台，RTSLの動作のために1台の計算機が必要である。その後，SCPが改造されSCP－Ⅱ(Simulation Control Program)と変ったこともあり，1台の計算機のみで起動しえる様にした。

手順は簡単で，使用計算機(任意の32kWメモリを持つ計算機)に表4で示した必要な周辺装置を結合し，IPL操作で，RTSL－Ⅱが起動される。図3にこの手順を示す。

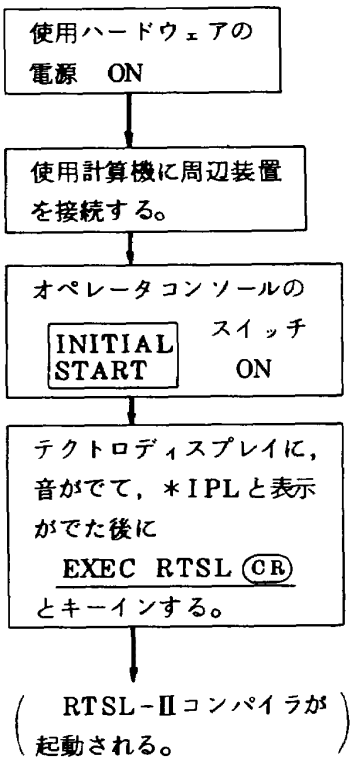


図3 RTSL－Ⅱの起動手順

RTSL－Ⅱの入力はカードであるので，起動されるとカードリーダからの読み込みが開始される。

1つのジョブ(TASK,SUBROUTINEあるいはFUNCTION)ごとにコンパイルが終了するが，終了後は自動的に初期状態に復帰し，カードリーダの読み込み待ち状態になる。

なお，停止は計算機の電源をOFFにすることで終了する。

5.3 出力のファイル化

初期のRTSLは，出力(オブジェクト)のファイル化が磁気テープであったが，操作性が悪いため，ほとんど使用されず，各ジョブごとにカードパンチ機に出力された。RTSL－Ⅱではディスクにファイルできる様にしたが，RTSL－ⅡのモニタがいわゆるDOS(Disk Operating System)でないため，特殊なプログラム(FMP)を作成し，ファイル化を可能とした。ファイルされたプログラムは，次に編集，結合され，さらに計算機にロードされて実時間演算実行可能となる。従って，ファイルに関しては，次の3つのプログラムの整合が必要となる。

① RTSL－Ⅱ (コンパイラ)
② FMP (ファイル管理)
③ LEP－Ⅱ (編集，結合，ロード)

ファイル化は次の方式で実施される。

RTSL－Ⅱはコンパイルエラーがないと，ディスクの固定部へオブジェクトを出力する。この後，ファイル管理専用のプログラム(FMP)をコアにセットする。この時RTSL－Ⅱは一旦消滅する。ただしオブジェクトが，タスクかライブラリかの区別の情報はコアの一部に残される。FMPはこの情報を基に，ディスクの固定部に出力されたオブジェクトをカートリッジディスクへ，LEP－Ⅱが解読できる形式でファイル化する。

FMPはファイル化が終了すると，再びRTSL－Ⅱをコアへセットして，RTSL－Ⅱを起動する。

以上によりユーザは，一旦RTSL－Ⅱを起動すればコンパイルが可能となり，エラーの無いかぎり，オブジェクトがファイル化され，次いでRTSL－Ⅱが自動的に再起動され，エンドレスにコンパイルが可能となる。なお停止は計算機の電源を切れば良い。

6. RTSL－Ⅱの評価

RTSL－ⅡはRTSLを改造後現在迄に約3年を経過している。初期のRTSLは機能を多種もたせようとしたため，実用上多くの問題点を残した。このため，RTSLが存在するにもかかわらず，アセンブラを用いざるを得ない時期があった。

RTSL－Ⅱは，機能を限定し，FSK－Ⅱハードウェアに見合ったシステム構成で製作したため，製作後ただちに使用可能となり，以後のシミュレーションプログラムは全てRTSL－Ⅱを用いて作成されている。

RTSL－ⅡとRTSLの違いは既に述べたが，RTSL－Ⅱを，いわゆる機械語に近いアセンブラと，通常のFORTRAN（DOSを用いると仮定）と比べると，表5に示す程度となり，シミュレーション用として，能力のあるものと考えられる。

RTSL－Ⅱによるプログラム記述例を付録5に示すが，FORTRANの知識がある者は容易に記述できるものと考える。

現在迄に，RTSL－Ⅱを用いて実施したシミュレーションとしては，

(1) 統合計器評価シミュレーション試験
(2) 非干渉制御系評価シミュレーション試験（第1次，第2次）
(3) 4D誘導シミュレーション試験（第1次～第3次）
(4) STOL 実験機飛行性評価シミュレーション試験（第1.1次～第3次）

等がある。

プログラムバグは一部あったが，基本的に変更を要する点はなく，一部ライブラリの充実を計って現在に至っている。

FSK－Ⅱのハード構成に見合ったものとしては，次の一点を除き，最良に近いものと考えている。

今後の改修すべき一点は，本言語の最大特徴の1つである固定小数点変数の属性を，オブジェクト出力とともにファイル化することである。

これにより，シミュレーション実行時の操作が著しく向上するものと期待できる。このためには，RTSL－Ⅱのほか，FMP，LEP－Ⅱ，SCP－Ⅱ 等の改造を必要とする。

表5　RTSL－Ⅱの特徴

		RTSL－Ⅱ	アセンブラ	FORTRAN
1	目的	FSK－Ⅱ専用 実時間シミュレーション	ミニコン機械語	一般科学技術 演算用
2	プログラム作成	比較的容易	訓練を要し， 誤まり易い	容易
3	表記法	わかり易い	わかり難い	わかり易い
4	演算方式	固定小数点，整数	固定小数点，整数 浮動小数点	整数 浮動小数点
5	オブジェクトプログラムの生成時間	時間がかかる	短かい	時間がかかる
6	オブジェクトプログラムの長さ	アセンブラによるものの約1.3倍以上	最小	一般に長くなる
7	プログラムの変更	比較的容易	時間を要する	容易
8	デバッグ	比較的容易	わかりにくい	一般に容易
9	特殊演算（ビット演算／操作）	可	可	不可
10	インラインアセンブル	不可	（当然）	可
11	各種ライブラリ	種類が少い	自作となる	一般に豊富

7. あとがき

航技研飛行シミュレータのディジタル演算部で用いるプログラム言語（RTSL）を改造し，新しい機能を加えたRTSL－Ⅱの基本構成，および実施法等について述べた。RTSLをベースに構成してあるが，実時間シミュレーション言語およびコンパイラとして，実用に供し得るものとして完成したと考える。

飛行シミュレータの使用者は，一般にはかぎられた範囲の研究者あるいは技術者であるが，多くのユーザに使用してもらうことが，RTSL－Ⅱをより良くする一つの方法と考えるので，以下に使用に当っての一般的注意事項を付記する。

① 表記法を十分理解すること。

② 1つのプログラム（タスク等）をあまり長くしないこと。

③ 倍精度は必要時のみ用いること。（倍精度演算はソフトウェアで実行するため，演算時間が長く，かつプログラム容量も増す）

④ 小数点位置はできるだけ上位ビットに置くよう，変数のダイナミックレンジの予測を正しく行うこと。

⑤ ビット演算も時間がかかるため，使用は必要最小限にすること。

⑥ ユーザ間でプログラム作成例を参照しあうこと。

等がある。特に始めて使用する場合は，⑥項が大切で間違いを少なくし，短時間でプログラムを完成する早道であろう。

最後に，本プログラムの評価作業には，名越孝行技官（元計測部自動制御第一研究室員）と鶴田知宏君（元日本大学研修生）の協力があったことを付記する。また，プログラムの製作に当っては，三菱プレシジョン(株)に依頼した。特に同社の藤野勝氏に多大の努力を払っていただいた。ここに感謝する。

参考文献

1) 原田；航空宇宙技術研究所汎用飛行シミュレータ用複合計算機（FSK－Ⅱ），航技研報告 TR－553，1978.2

2) ――；RTSL取扱い説明書，1975

3) 渡辺；航技研ディジタルフライトシミュレータの評価（その1），第19回自動制御連合講演会，1976.11

4) 森，他；クインエア機の縦の安定操縦微係数の推定，航技研報告 TR－406，1975.2

付録1　RTSL－Ⅱの文法規則

RTSL－Ⅱの文法の多くはRTSLのものを踏襲している。従って以下の多くは再掲する形で示すことになる。

1.1　プログラム構成

RTSL－Ⅱのプログラムは一般に次のものにより構成されている。

サブプログラム
文 式 データ

まずプログラムは付表1に示すように，シミュレーション目的の演算を記述するプログラム（実行サブプログラム …… TASK, SUBROUTINE, FUNCTION）と実行サブプログラムの補助をし，入出力変数について定義するプログラム（非実行サブプログラム …… INPUT, OUTPUT）とに分けられる。

付表1　サブプログラムの種類

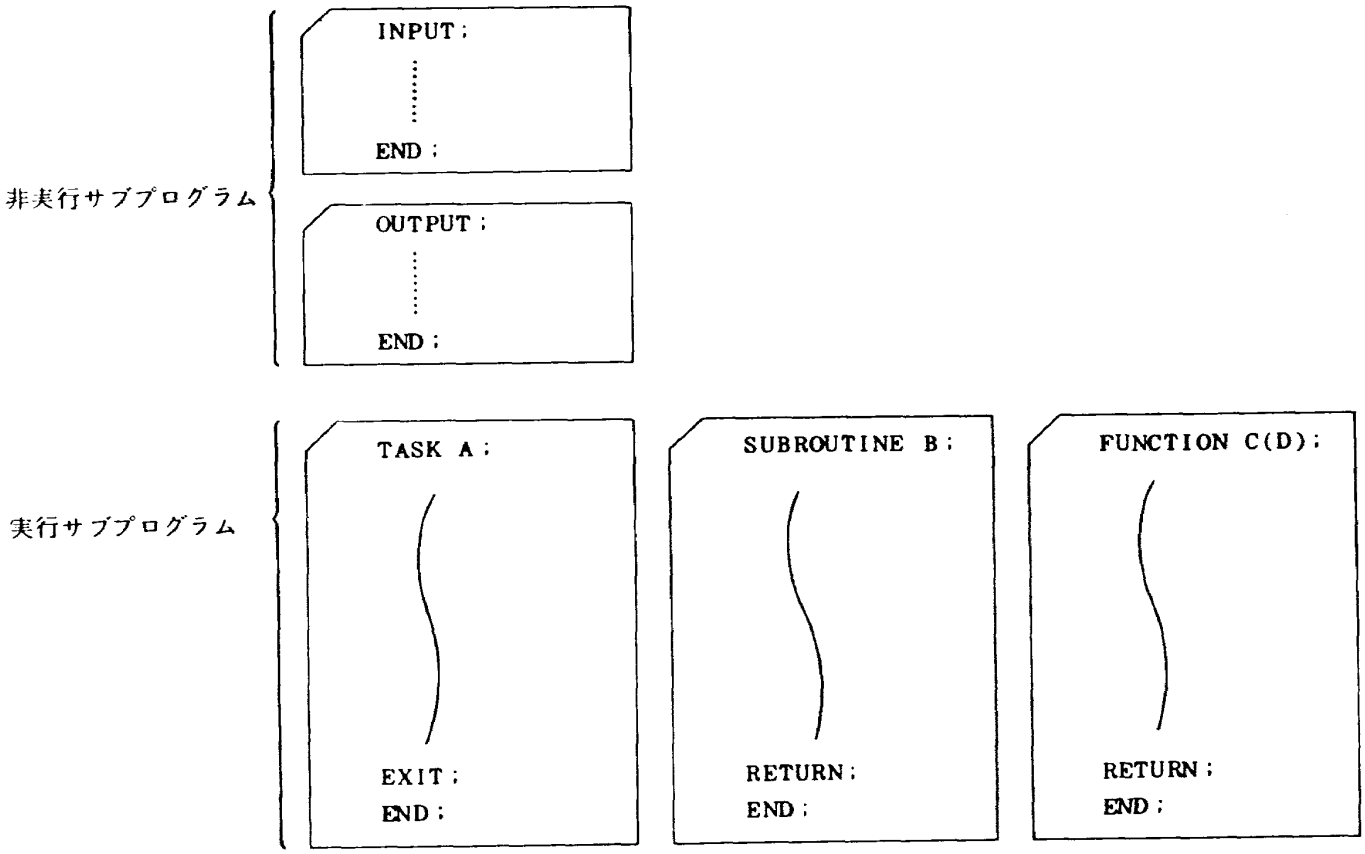
	種　　類
非実行サブプログラム	INPUT　サブプログラム OUTPUT　サブプログラム
実行サブプログラム	TASK　サブプログラム SUBROUTINE サブプログラム FUNCTION サブプログラム

(1)　非実行サブプログラム

このサブプログラムにはINPUTサブプログラムとOUTPUTサブプログラムがある。

(イ)　INPUTサブプログラム

これは付図1に示すように実行サブプログラムの前に置かれ，その実行サブプログラムへの入力変数をDCL文（1.4(7)参照）で定義する。すなわちその実行サブプログラムで使用し，かつ外部で発生される変数をさす。この時その変数の属性（その変数の性質…1.4(7)参照）を合わせて定義する。



付図1　サブプログラム基本構成

書式を次に示す。

```
INPUT    T1;
  DCL    A    J15;
  DCL    B    J03;
  DCL    D    K-05;
  DCL    M    I;
  ⋮
END;
```

INPUT文で始まり，入力変数をDCL文で定義し，END文で終る。各変数に値をもたせることはできない。

(ロ) OUTPUT サブプログラム

これはINPUTサブプログラムと逆に，次に定義する実行サブプログラムの出力変数についてDCL文で定義する。合わせて変数の属性を定義する。

書式を次に示す。

```
OUTPUT      T2;
  DCL   X   J13;
  DCL   Y   K31;
  DCL   N   I;
  ⋮
END;
```

OUTPUT 文で始まり，出力変数をDCL文で定義し，END 文で終了する。各変数に値をもたせることはできない。

(2) 実行サブプログラム

実行サブプログラムとしては，TASK, SUBRŌUTINE, FUNCTIONの3つがある。この3種類のプログラムの中で，TASKは目的の演算を記述するためにかならず作られる。作り方としては，作成の便宜上，いくつかに分割することが可能である。この場合は，各TASKごとに入出力変数を規定するINPUT, OUTPUT のサブプログラムを前段に付加してコンパイルすることになる。

(イ) TASKサブプログラム

書式例を次に示す。

```
TASK    ABC;
    DCL  D1   J03;
    DCL  D2   J15;
    DCL  C3   J12 INIT(15.3);
    ⋮
    D1=SIN(A);
    D2=C3*D1+B;
    ⋮
    EXIT;
  END;
```

TASKサブプログラムはTASK文で始まり，そのTASKのみで用いるローカル変数(例ではD1, D2, C3)をDCL文で属性を定義し，次に目的の演算式を書き演算の終了を実行時のモニタに知らせるEXIT文を1つ以上加え，最後にTASKサブプログラムの終了をコンパイラに知らせるEND文を加えて終る。

(ロ) SUBRŌUTINE サブプログラム

このプログラムは他の実行サブプログラム(メインプログラム)内のCALL 文により呼ばれるプログラムで1つの独立した単位である。

書式例を次に示す。

```
SUBRŌUTINE    S1   (X, Y);
    DCL    X    J14;
    DCL    Y    J13;
    ⋮
    A=FUNC(Y)*0.3@J15;
    X=A+B;
    ⋮
    RETURN;
  END;
```

この例に示す様に SUBRŌUTINE サブプログラムはSUBRŌUTINE 文で始まり，演算の終了を知らせるRETURN 文を1つ以上加え，最後に本プログラムの終了を知らせるEND文を書いて終る。

メインプログラムとサブルーチンとの変数の引き渡しは引数で行うか，INPUT, ŌUTPUTのサブプログラムを用いて行われる。引数の場合はメイン側と引数の順序および属性を一致させておくことが

必要である。

なお，サブルーチンを再入可能*)にするためにはSUBROUTINE 文の最後にREENTRANTと加える。ただし，RTSL－Ⅱは擬似再入可能方式で内部処理としては再入不可能（すなわちこのサブルーチンの全計算が終了しないと，このサブルーチンをコールできないようにする）にして処理している。

SUBROUTINE 文の仮引数として許されるものは数値変数（スカラ），配列名，関数名，および他のサブルーチン名である。

(ｲ) FUNCTIONサブプログラム

これもSUBROUTINE と同様他の実行サブプログラム（メイン）の中で関数形式で使用された関数に対応したプログラムである。従って関数値と引数の属性，引数の順序はメイン側と一致していることが必要である。

本言語のFUNCTIONは通常の関数値の定義のほか，フライトシミュレーションで多用されるテーブル・ルック・アップ方式のデータを容易な表記法（DATA文を用いる）で作成しえるようにもなっている。

書式の例を次に示す。例1がデータテーブルを作成する例である。

一般構成は FUNCTION 文で始まり，使用変数，定数の属性をＤＣＬ文で定め，演算終了を知らせるRETURN 文を1つ以上加え，最後にEND 文を加えて終る。

例1

```
FUNCTION    F2(Y,X);
   DCL      Y     J15;
   DCL      X     J14;
   DCL      F2    J12;
   DATA     0.4    0.0    2.5,
                   0.1    3.5,
                   0.5    4.7,
                   0.8    5.4,
            0.2    "      1.3,
                          2.4,
                          4.4,
                          5.0;
   RETURN;
END;
```

例2

```
FUNCTION    FUNC(A,B) REENTRANT;
   DCL      FUNC    J17;
   DCL      A       J15;
   DCL      B       J17;
   DCL      C1      J17 INIT(0.03);
   FUNC = C1*A + B ;
   RETURN;
END;
```

このプログラムもサブルーチンと同様に擬似再入可能とすることができ，例2で示すようにREENTRANTを加える。

DATA 文を書くことで関数サーチのプログラムができることになるが，変数の数は1，2，3が可能で，データは線形補間がほどこされる。

DATA 文の定数については属性をつけないが，第1変数，第2数，第3変数は FUNCTION 文の引数の順序と同じであり，DCL文の変数と一致する変数名の属性が適用される。（1.4(9)DATA文参照）

1.2 使用文字

本言語では，コマンドを含め，式，文を記述するのに次に示す数字，アルファベット，記号文字を用いる。特殊記号の多くはビット演算式で使われるが，その使用に当っては注意が必要である。

数　字：　0～9

アルファベット：A～Z

記　号：．＜（＋＆＄＊）；－／，

＞↑：＃＠｜＝！＂￢＼＇

変数はアルファベットか，：で始まる8文字以内で記述する。

1.3 文の書式

コンパイラの入力媒体はカードであるため，カードのフォーマットに合わせた部分がある。

書式上の規定は次の通りである。

(1) 文は1行の中の1～72桁のどの桁に書いても良い。（カードにおいて,73～80桁はコンパイル時に無視される。ただしリストには印字される）

(2) 文の終りは；で示される。

---

*) 再入可能とはそのサブルーチンの全演算が終了しないうちに，違うプログラムによって呼ばれても演算，制御が正しく行われる構造になっていることをさす。

(3) 文（但し実行文）に識別子（その文の名札）を付けることができる。識別子は文の先頭に置かれ，本体とは：で区別される。

(4) コメント文は，／＊で始まり＊／で終る。

1.4 文の種類

文には非実行文と実行文があり，付表2に示す種類がある。以下，各文の意味を簡単に述べる。

(1) INPUT文

INPUTサブプログラムの先頭に置かれ書式は

```
INPUT ;
  または
INPUT      A ;
```

である。INPUTのプログラム名（この場合A）を付けても良い。コンパイル時は無視される。リストには出る。

付表2 文の種類

非実行文	INPUT 文	
	OUTPUT 文	
	TASK 文	
	SUBROUTINE 文	
	FUNCTION 文	
	END 文	
	DCL 文	
	OVERFLOW 文	
	DATA 文	
実行文	代入文	数値代入文
		ビット列代入文
	制御文	GO TO 文
		IF THEN 文
		ELSE 文
		DO 文
		CALL 文
		EXIT 文
		RETURN 文
	複合文	BEGIN ～ END

(2) OUTPUT文

OUTPUTサブプログラムの先頭に置かれ，書式は

```
OUTPUT ;
  または
OUTPUT  VARIABLE ;
```

である。プログラム名（VARIABLE）はコンパイルリストにのみ関係する。

(3) TASK文

TASKサブプログラムの先頭に置かれる。書式を次に示す。

```
TASK     TEST ;
```

タスク名（この場合TEST）がかならず必要である。

(4) SUBROUTINE文

SUBROUTINEサブプログラムの先頭に置かれる。書式を次に示す。

```
SUBROUTINE SUB1 ;
SUBROUTINE SUB2(X, Y) ;
SUBROUTINE SUB3(Z) REENTRANT;
```

第2の例は引数（X, Y）がある場合で，第3の例は擬似再入可能とするため，REENTRANTが付加されている。

(5) FUNCTION文

FUNCTIONサブプログラムの先頭に置かれる。書式を次に示す。

```
FUNCTION  F : CL (AL) ;
FUNCTION  FA(Y, Z)  REENTRANT;
```

第2の例は擬似再入可能とした例である。

(6) END文

各サブプログラムの終了および内部関数や複合文（(19)参照）の終了を示す文である。

書式を次に示す。

```
END ;
```

(7) DCL文

変数あるいは関数式の属性を宣言するものである。

また，内部定数については属性とともに，その値も記述できる。

記述例を以下に示す。

```
1   DCL   A              J14;
2   DCL   B              J12 INIT(3.504);
3   DCL   C              K04;
4   DCL   D              K18 INIT(800.5);
5   DCL   E              I;
6   DCL   F              I INIT(13);
7   DCL   G              B:(F,10);
8   DCL (H1,H2,H3)       J13;
9   DCL (K1,K2,K3)       I INIT(2,3,4);
10  DCL P(5,4)           J+02;
```

上の例に示したように，DCLの後に変数あるいは定数の名札を書き，次にその属性（B，I，J，K等…付表5参照）を表記し，最後に必要な場合はその変数または定数の初期値をINITの後に書く。

例の7は，ビット変数を示すもので，変数Fの10番目のビットの名前をGとすることを示す。

同一の属性の変数は例8，例9に示すように( )でまとめることができる。

(8) OVERFLOW文

書式を次に示す。

```
ON    OVERFLOW   J1;
    ⋮
（代入文）
    ⋮
OFF   OVERFLOW;
```

ON OVERFLOW文により，計算機のオーバフロー状態がリセットされ，OFF OVERFLOW文がくるまで，この間にある代入式ごとにオーバフロー状態がチェックされ，オーバフローが起きるとJ1の点へプログラムの流れが移される。

(9) DATA文

通常のFORTRANのDATA文と異なり，1～3変数関数（テーブルルックアップ方式）を発生させるためのデータを記述する文である。

書式は，1変数，2変数，3変数の場合について若干異なる。例を付図2(a)(b)(c)に示す。2～3変数関数の場合に，同一横軸データは*で表わし，再度同一データを書く必要がない。

指定点数およびきざみ幅は任意にとれる。また，各点間の補間は直線補間である。

以上が非実行文である。実行文としては代入文と，制御文に分かれる。以下これらについて説明する。

(10) 数値代入文

＝記号を用いた文で，右辺での値が左辺に代入される。

書式例を次に示す。

```
A=SIN(X)+2.0@J10/Y;
```

小数点位置は各変数について，DCL文で宣言されたものに対応するが，右辺の結果と左辺のAの小数点位置が異なる場合は，代入時に左辺のAに対応したものに調整される。

(11) ビット列代入文

＃記号を用いた文で，右辺の結果が左辺に代入される。

書式例を次に示す。

```
B#J&K;
```

この例はJとKの論理積(&)の結果が，Bに代入される。

(12) GO TO文

書式を次に示す。

```
GO TO L1;
```

名札L1をもつ文にプログラムの流れを移す。

(13) IF～THEN文

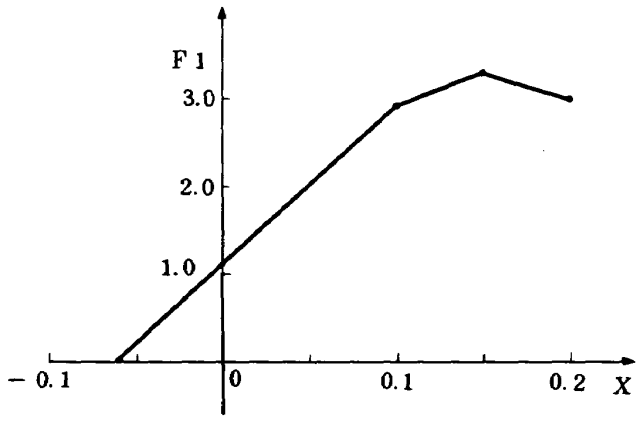
書式を次に示す。

```
1   IF   X>=XMIN THEN GO TO J1;
2   IF   V==VM   THEN BEGIN
                  A=B+C;
                  D=A*X;
                  END;
         ELSE     D=Y;
```

IFとTHENとの間での条件式が満たされた時，次の実行文が実行される。2番目の例のTHENの

```
/* ONE VARIABLE FUNCTION */
   FUNCTION  F1(X);
      DCL     X     J15;
      DCL     F1    J12;
      DATA          -0.07     0.0,
                     0.10     2.9,
                     0.15     3.3,
                     0.20     3.0;
      RETURN;
   END;
```

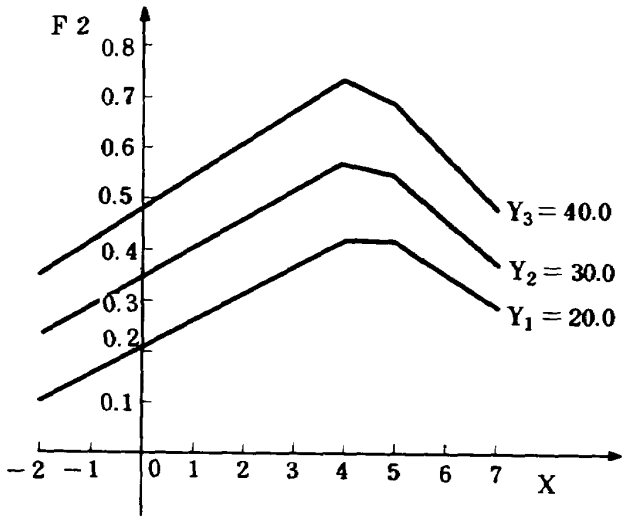
X F1



(a) 1 変数関数（プログラム例とグラフ）

```
/* TWO VARIABLES FUNCTION */
   FUNCTION  F2(Y,X);
      DCL     Y     J07;
      DCL     X     J10;
      DCL     F2    J15;
      DATA          20.0     -2.0     0.10,
                                4.0     0.42,
                                5.0     0.42,
                                7.0     0.29,
                    30.0        "      0.23,
                                       0.57,
                                       0.54,
                                       0.38,
                    40.0        "      0.35,
                                       0.73,
                                       0.69,
                                       0.48;
      RETURN;
   END;
```

Y X F2



(b) 2 変数関数（プログラム例とグラフ）

```
/* THREE VARIABLES FUNCTION */
   FUNCTION  F3(Z,Y,X);
      DCL     Z     J08;
      DCL     Y     J09;
      DCL     X     J10;
      DCL     F3    J11;
      DATA          50.0     10.0     1.0     3.0,
                                      2.0     5.0,
                                      3.0     7.0,
                                      4.0     4.0,
                               12.0     "       3.5,
                                                5.5,
                                                7.5,
                                                4.5,
                               15.0     "       4.0,
                                                6.0,
                                                8.0,
                                                5.0,
                    100.0      "       "       2.0,
                                                3.0,
                                                5.0,
                                                4.0,
                                       "       2.5,
                                                3.5,
                                                5.5,
                                                4.5,
                                       "       3.0,
                                                4.0,
                                                6.0,
                                                5.0;
      RETURN;
   END;
```

Z Y X F3

(c) 3 変数関数（プログラム例，グラフは省略）

付図 2 多変数関数記述例

次の実行文は複合文（BEGINで始まりENDで終了する）で，条件が満たされるとこれら全ての実行文が実行され，満たされない場合はELSE文が実行される。なおIF～THENの次にELSE文がある場合は，条件が満たされた後の流れは，THENの次の実行文が実行され，その後はELSE文の後へ移る。

⑭ ELSE文

書式を次に示す。

```
ELSE      X＝A＋B；
```

これはIF文の後で使用され，IF文の条件が満たされない場合に，ELSEの次の実行文が実行される。実行文は複合文でも良い。

⑮ DO文

書式例を次に示す。

```
DO      I＝N1,N2,N3,
X(I)＝ A(I)＋B(I)；
```

IにまずN_1をセットし，DOの次の実行文を実行し，次にN_3を加えてIの値がN_2より大きくなる迄，X(I)の演算が実行される。N_1はϕが許される。実行文は複合文でも良い。

⑯ CALL文

書式を次に示す。

```
CALL    SUB (A,B)；
```

サブルーチン（SUB）をコールする。この場合，A，Bは引数で，受け側のサブルーチンと同一の意味になるよう一致させることが必要である。

⑰ EXIT文

書式を次に示す。

```
EXIT；
```

タスクサブプログラムの演算終了を知らせる文で，この文の実行により実時間モニタ（RTSM）へ計算機の実行の流れが移される。

⑱ RETURN文

書式を次に示す。

```
RETURN；
```

サブルーチンあるいはファンクションサブプログラムの演算終了を知らせる文で，この文の実行によりメインルーチンへもどる。

⑲ 複合文

複数個の実行文をまとめた文である。

書式を次に示す。

```
BEGIN
  A＝ X(1)＋Y(1)；
  B＝ X(3)＋A；
  ⋮
END；
```

BEGINで始まりENDで終了する。

1.5 式

式とは文によって記述される操作の対象となるものである。式の種類としては付表3に示すものがある。この式についての意味は，

①式はデータそのもの，あるいはデータ，演算子，（　）等を適当に組み合わせたものをさす。（　）は引数の組，あるいは演算優先順位を示すために用いる。

②式はデータと同様なモード（付表5参照）をもち，そのモードによってビット式，整数式等と呼ばれる。

③式に，（　）の無い場合の演算順序は演算子の優先順位によって定まる。優先順位が同じである時は，原則として左から順に行われる。

④式の中で以下示すものは擬似変数という。

付表3　式の種類

	式　名
1	算術式
2	ビット列操作式
3	数・論理混合式
4	スケール式
5	属性付加式
6	配列要素式
7	関数式
8	サブルーチン式

配列要素式，関数式，組込関数式，$B:(X,Y)$，変数，擬似変数あるいは，これらに属性を付加したものは，値またはビット列を代入することができる。

(1) 算術式

算術演算子と数式によって構成される。演算子の種類を付表4に，またモードを付表5に示す。算術式の実行に当っては以下の規則に従う。

(イ) 複数の数式から構成され，そのモードが異なる場合は，原則として高いモードに合わせた後，演算または比較を行う。モード変換は以下の規則に従う。

$$I \rightarrow J$$
$$J \rightarrow K$$

(ロ) 固定小数点同志の加算，減算は小さい方のスケールに合わせて行う。

(ハ) 固定小数点同志の乗算，除算は以下の規則による。M，Nは固定小数点位置を示す。

$$J(M) * J(N) \rightarrow J(M+N-15)$$
$$J(M) / J(N) \rightarrow J(M-N+16)$$
$$K(M) * K(N) \rightarrow K(M+N-32)$$
$$K(M) / K(N) \rightarrow K(M-N+32)$$

ここで注意を要することは除算（／記号）である。これは用いている$M-70$ディジタル計算機のハードウェアの特徴で，除算演算結果の小数点位置が上がるためである。

今，次式を考える。

$A(M) / B(N) = C(M-N+16)$

（カッコ内は小数点位置）

$M = N + i$

としたとき，答Cの値が$< \frac{1}{2^{i+1}}$ならば，除算はこのまま実行して良い。もし，この条件が満たされない場合は，Cの値（場合により推定値）により，付表8を用いてこのCの値の小数点位置（Lとする）を知って，

$M-N+16 = L-1$

となる様にMの値を求め，この値にAの小数点位置をシフトしてから除算を実行する。

例えば，$0.24 \div 0.49 \fallingdotseq 0.4898$を実行する時，0.24の小数点位置は$J17$（従って$M=17$），また0.49は$J16$（$N=16$）である。これより，

$i = M-N = 17-16 = 1$，従って

$$\frac{1}{2^{i+1}} = \frac{1}{2^2} = \frac{1}{4} = 0.25$$

付表4　算術式の演算子記号

種類	演算子	定義	表記例	結果	優先順位
符号	＋	何もしない	$+A$, $+0.5$ @$J15$	そのまま	2
	－	符号反転	$-X$, -3.0 @$K25$	符号反転	2
演算式	＊	乗算	$C*D$	上位レジスタ分	3
	／	除算	E/F	上位レジスタ分	3
	＋	加算	$G+H$		4
	－	減算	$P-Q$		4
	＝	代入	$X=A*B-C$		10
条件式	＝＝	等しければ真	$A==B$		5
	＞	より大きければ真	$C>D$		5
	＜	より小さければ真	$E<F$		5
	＞＝	より大きいか等しければ真	$G>=H$		5
	＜＝	より小さいか等しければ真	$P<=Q$		5

付表5 属性(モード)と表記例

属性			内容			変数表記例	定数表記例
モード		記法	値域	占有語数(ワード)	配列		
ビット		B	1, 0	1/16	不可	B	φB 1B
整数		I	−32,768～+32,767	1	可	N K	127 −294
固定小数点数	単精度	J−128～J127	$\pm 2^{15-M}$	1	可	X1 Y(2,3)	5.3@J11
	倍精度	K−128～K127	$\pm 2^{31-M}$	2	可	A:3 B(3,6,2)	−238.0@K23
名札		L	——	—	—	P3	—
サブプログラム(FUNCTIONの属性)			——	—	—	F3(X,Y)	—
			−128≦M≦127			☆16進数表記法 $nZxx\cdots x$@A (n=(4,8,16), x;16進数字, A;属性) (例) 4Z12FE@J14 8ZFFE432AB@K20 ☆文字列表記法 $nHyy\cdots y$@A (n=(2,4,8), y;文字, A;属性) (例) 2HAB@I 4H1Z3B@K20	

で，答(0.4898)はこれより大きい。

答(0.4898で)の小数点位置は$J16$($L-16$)であり，Mは，$M=L+N-17=16+16-16=15$より，15とする必要がある。

(ニ) 右辺の演算結果の代入は左辺の属性に合わせた後に行う。

(2) スケール式

固定小数点数式のスケールを操作するための式であり，書式を次に示す。

($A+B$)，5

これは$A+B$の演算結果のポイントの位置を5とする。

(3) ビット列操作式

ビット列操作演算子と数式またはビット式から構成される式とする。この式の種類，式のモード等を付表6に示す。複数の式から構成され，その語長が異なる場合には，最上位ビットを合わせた状態で，演算または比較を行う。

(4) 配列要素式

配列の要素を指定する式である。書式は次のようになる。三次元までとする。

$X(5)$, $Y(3,2)$, $Z(3,2,9)$

(5) 数，論理混合式

条件によって異なる値をもつ数式であり，書式の例を次に示す。

$B:(1,2)*(A+B)\cdots=A+B$($B:(1,2)=1$の時)
$=\phi$ ($B:(1,2)=\phi$の時)

付表6 ビット列操作式の記号

演算子	定義	表記例	意味	優先順位
< >	左向きローテイト	A< >3	Aの中身を3ビット左向きにローテイトする	1
≪	左シフト	B ≪ 5	Bの中身を5ビット左にシフトする	1
¬または＼	否定	¬(A+B)		6
&	論理積	(C+D)&E		7
|または↑	論理和	F|G		8
!	排他的論理和	K!H		8
#	代入	B#C&D	CとDの論理積をBに代入する	10
# #	等しければ真	N##K	NがKに等しければ真(N, Kビット)	9
' '	最上位ビット	'J'	Jの最上位ビットの中身	1

(6) 属性付加式

数式，データにそれが本来もっているものとは異なる属性を付加することを可能とするものである。

書式の例を以下に示す。

(A*B) @J 13

(7) 関数式

関数サブプログラムを利用するための式である。書式の例を次に示す。

FUNC(X, Y), F(X*B, Y*2.0 @J13)

SIN(TH)

ここで関数式は値をもち，式の一変数として用いることができる。( )内の式を実引数といい，実引数の個数，順序，属性等は，仮引数のそれらと一致していなければならない。

組み込み関数については付録4で述べる。また組み込みでない関数式の属性は宣言もしくは属性付加をしなければならない。

(8) サブルーチン式

他の実行サブプログラム内でのCALL文の対象としてサブルーチン式がある。

書式例を次に示す。

CALL EULJ (X, XD)；

サブルーチン式の( )内の変数あるいは式を実引数というが，この個数，順序，属性はサブルーチンサブプログラムの仮引数と一致していなければならない。

1.6 データ(変数，定数)

式によって記述される操作の対象となるものであり，次の定義に従う。

① 変数とは，プログラムの実行によって変り得るものを言い，変り得ないものを定数という。

② 変数には，スカラ及び配列がある。配列は，1，2，3次元があり，その要素は同一の属性を持つ。要素に対する番号付けはφから始まる。

A(φ)， B(5, φ)

③ 変数，名札，サブプログラムは識別子によって表現される。識別子は英字で始まり，英数字，：から構成される長さ8文字以下の文字列とする。

④ 識別子は属性のいかんによらず一意である。識別子として，RTSL-Ⅱが本来持っている記号列（DO，IF等）と一致してもさしつかえない。

⑤ 属性のビット，整数，単精度固定小数点数等をモードといい，モードの高さは，次の順による。（付表5参照）

（下位）$B \rightarrow I \rightarrow J \rightarrow K$（上位）

Jタイプ，Kタイプの変数および定数の属性の決め方については，付録3のデータの扱いについてを参照すること。

付録2　RTSL－Ⅱのコマンド

RTSL-Ⅱを動作させる場合，ユーザの目的に応じて，リストの出力のみであったり，オブジェクトプログラムをカードパンチ機に出させたりすることができる。これらのために，RTSL-Ⅱにいくつかのコマンドが用意されており，必要に応じてソースカードと一緒に読み込ませる。

コマンドを付表7に示す。

全てのコマンドは内部に初期設定がしてあり，通常のケースに於いては，特にコマンドを指定しなくとも良い。

付録3　データの取扱いについて

固定小数点データとしては，単精度データと倍精度データがある。FSK-Ⅱの使用計算機（M-70ミニコン）の1ワードは16ビットで，この1ワードであらわすデータを単精度といい，2ワードデータを倍精度という。

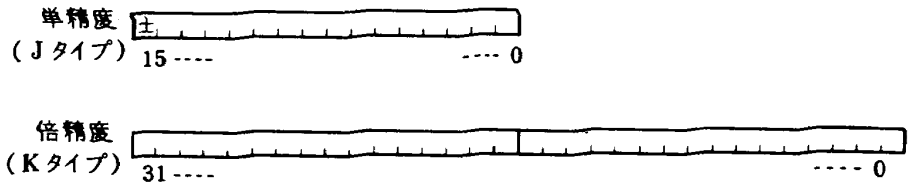
各データとも付図3に示すように先頭ビットを符号ビットとする。

単精度にするか倍精度が必要かは，そのデータの必要とする精度に依存するが，その決め方は，付図4に従って決めることができる。

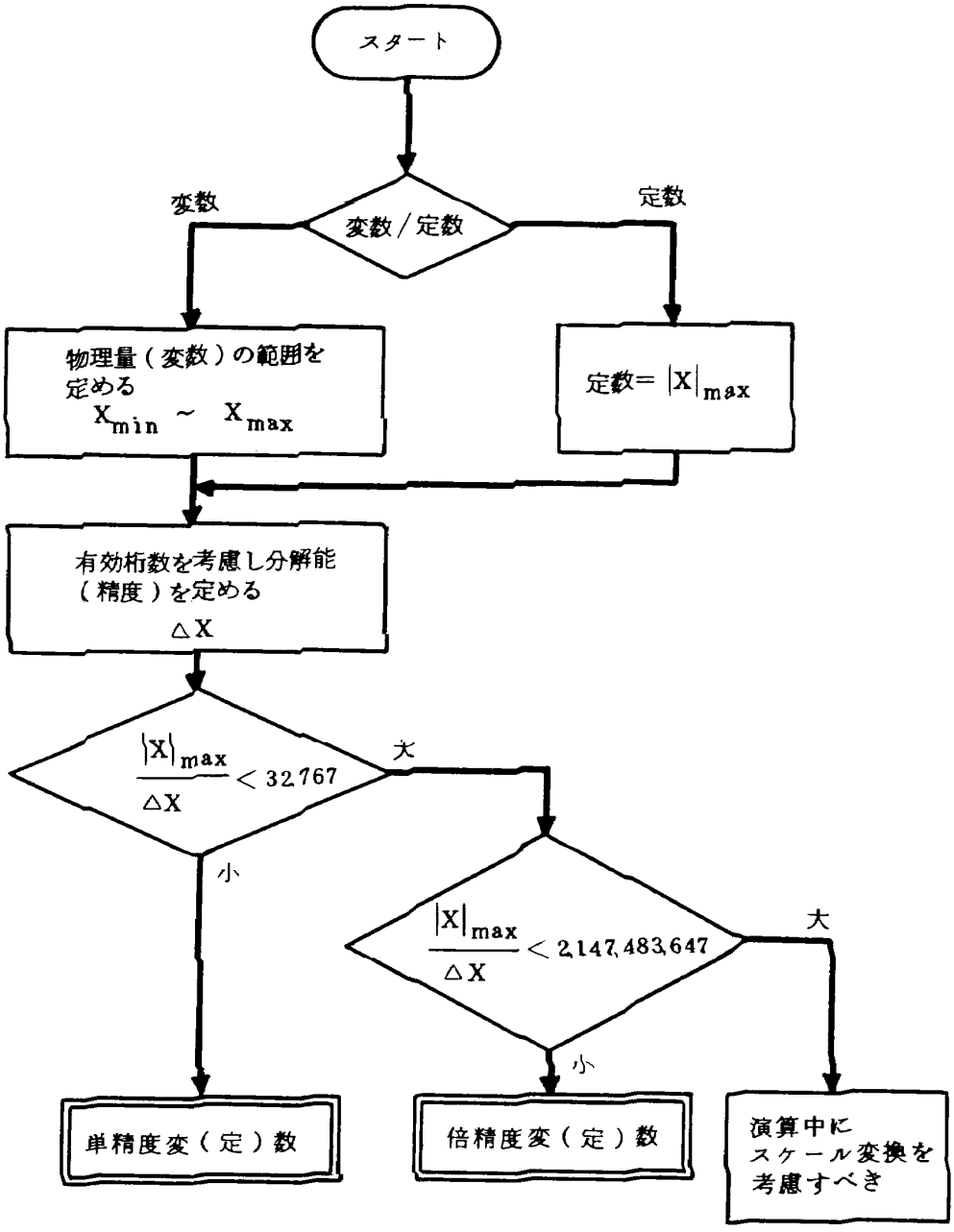
次に，固定小数点位置を定める。すなわち，付図3において一番右側に固定小数点があると見なした場合は整数扱いとなる。また，符号ビットのすぐ右

付表7　RTSL-Ⅱのコマンド

	記号	定　　義	表　記　例（* は初期設定条件）
1	CL	コンパイルリスト出力機器指定	CL *LP；（ラインプリンタ） CL φ；（出力せず）
2	AL	アセンブルリスト出力機器指定	AL LP；（ラインプリンタ） AL *φ；（出力せず）
3	ST	シンボルリスト出力機器指定	ST LP；（ラインプリンタ） ST *φ　（出力せず）
4	$\overline{O}$M	オブジェクトプログラム出力機器指定	（デフォルト）*（ディスク） $\overline{O}$M CP；（カードパンチ） $\overline{O}$M φ；（出力せず）
5	WA	ワークデバイス（ドラム）のワークエリアの指定	WA N1,N2； … ワークエリアをN1セクタからN2セクタまでとする。 この指定がないとドラムの全領域をワークとする。
6	EP	コンパイルリストの改ページ	EP； ソースカードの必要な場所にはさんでおく。



付図3　データ長



付図4　データの属性の決め方

側に小数点があると見た時は最大値が1.0（＋の場合は1.0未満となる）のデータとなる。

RTSL-Ⅱでは単精度固定小数点数をJタイプ，倍精度固定小数点数をKタイプのデータという。

データをJタイプとするか，Kタイプとするかは，上記の付図4に従って定めるが，どこに小数点位置を定めるかは付表8による。この時，表中の例に示すように，その値をはさむ場所をさがし，その左または右端に示してある小数点位置を定めれば良い。負数については符号を無視して定める。ただし，表と一致する数値（例えば1.0，2.0，0.5，0.25等）については，表の左側（すなわち1.0以上）では下側の小数点位置をとり，右側（1.0未満）では上側の位置を用いる。例えば2.0はJ13かK29，0.25はJ16かK32とする。

以下，定数および変数の決め方の例を示す。

(i) 定数の場合

数　値	単精度	倍精度
−20.0	J10	K26
−0.037	J19	K35
0.0054	J22	K38
205.4	J06	K22

(ii) 変数の場合

(イ) 飛行高度　H：0 m～10,000 m

分解能 ΔH：0.1 m

$$\frac{|H|_{max}}{\Delta H} = 100,000$$

従って付図より倍精度とすべきで，付表8より，Hの属性はK17。

(ロ) 迎角　ALPHA：−10 ～40

（−0.1745 rad～0.698 rad）

分解能　$\Delta\alpha$：0.1　(0.001745 rad)

$$\frac{|\mathrm{ALPHA}|_{max}}{\Delta\alpha} = 400$$

付図4より単精度で良く，付表8より，ALPHAの属性は，J9(°)，またはJ15(rad)。

付録4　RTSL－Ⅱライブラリ

4.1　ファンクション

ファンクションの種類は表2に示してあるので，ここでは，各ファンクションの目的，使用条件等について示す。

4.1.1　正弦（SIN）

1) 目　的

固定されたバイナリポイント（以下BPと記す）を持つ単精度固定小数点2進数の正弦を計算する。

計算できるアーギュメントθは（−2 rad＜θ＜2 rad）の範囲内のものである。

2) 呼出し形式

例1　A＝SIN（X）；

例2　B＝SIN（$X*Y+Z$）；

3) 使用時の条件と結果

データ：BP14の変数名または算術式を記述する。

結　果：BP14の計算結果がSINに代入される。

4) プログラム容量

235ワード

5) 演算方式

テーブル・ルック・アップ

6) 実行時間　（最大）

143 μsec

7) 最大誤差

1.2×10^{-4}

4.1.2　正弦　（SINϕ）

1) 目　的

固定されたBPを持つ単精度固定小数点2進数の正弦を計算する。

計算できるアーギュメントθは（−2rad＜θ＜2rad)の範囲内のものである。

2) 呼出し形式

例1　A＝SINϕ（X）；

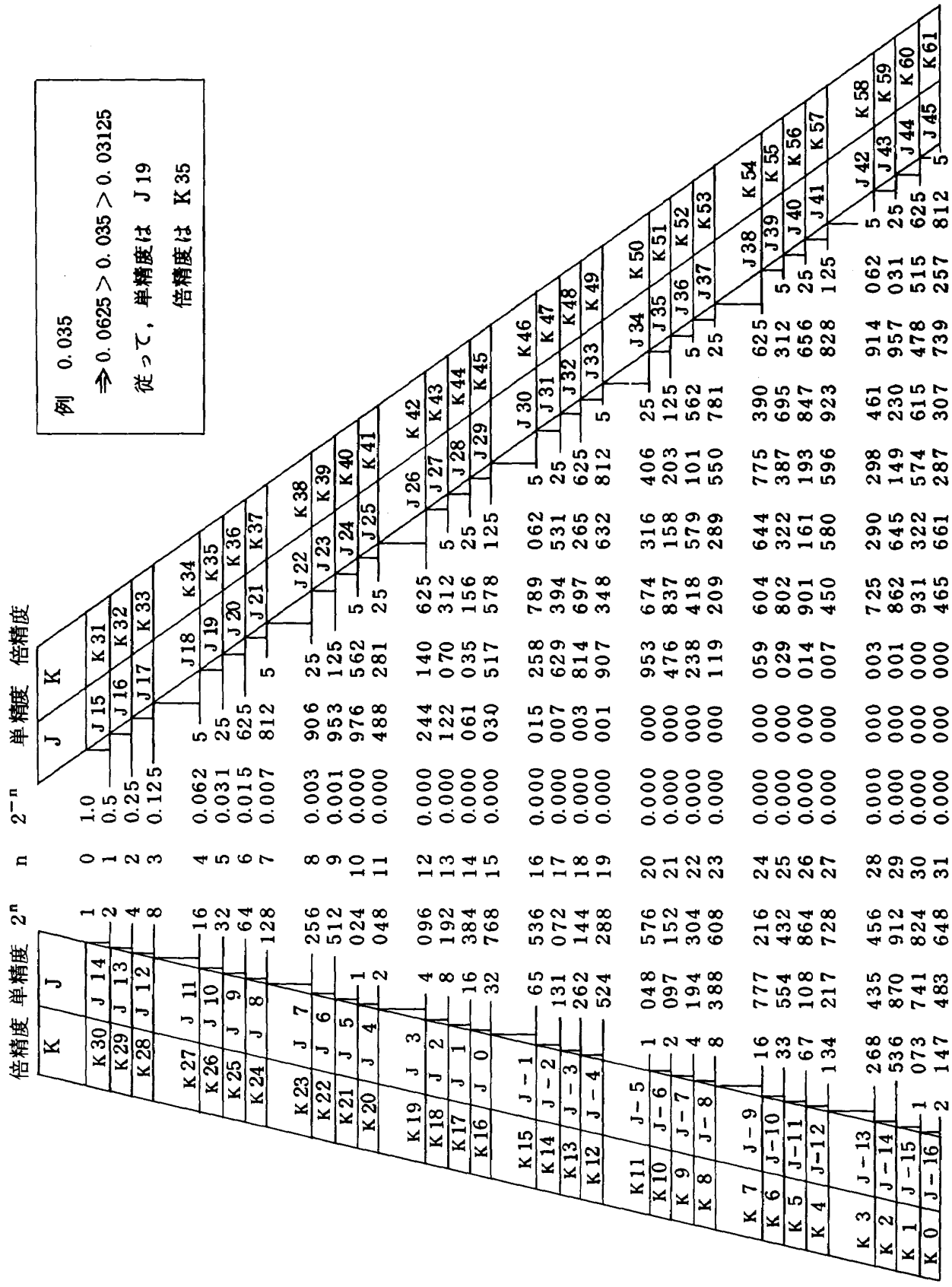
例2　B＝SINϕ（$X*Y+Z$）；

3) 使用時の条件と結果

データ：BP14の変数名または算術式を記述する。

結　果：BP15の計算結果がSIN ϕに代入される。

付表8 固定小数点位置



4) プログラム容量
236ワード
5) 演算方式
テーブル・ルック・アップ
6) 実行時間　（最大）
139 μsec
7) 最大誤差
1.2×10^{-4}

4.1.3　正弦　（SIN：）

1) 目　　的

指定された BP を持つ単精度固定小数点2進数の正弦を計算する。

計算できるアーギュメント θ は（$-2\text{rad}<\theta<2\text{rad}$）の範囲内のものである。

2) 呼出し形式

例1　A＝SIN：（X, BP）；

例2　B＝SIN：（X, 15）；

3) 使用時の条件と結果

データ：変数名又は数値を記述する。

BP：データの BP を整数型で指定する。
（$12\leqq BP\leqq15$）

結　果：BP15 の計算結果が SIN：に代入される。

4) プログラム容量
209ワード
5) 演算方式
テーブル・ルック・アップ
6) 実行時間　（最大）
129 μsec
7) 最大誤差
1.2×10^{-4}

4.1.4　余弦　（$\overline{\text{COS}}$）

1) 目　　的

固定された BP を持つ単精度固定小数点2進数の余弦を計算する。

計算できるアーギュメント θ は（$-2\text{rad}<\theta<2\text{rad}$）の範囲内のものである。

2) 呼出し形式

例1　A＝$\overline{\text{COS}}$（X）；

例2　B＝$\overline{\text{COS}}$（$X*Y+Z$）；

3) 使用時の条件と結果

データ：BP14 の変数名または，算術式を記述する。

結　果：BP14 の計算結果が $\overline{\text{COS}}$ に代入される。

4) プログラム容量
225ワード
5) 演算方式
テーブル・ルック・アップ
6) 実行時間　（最大）
138 μsec
7) 最大誤差
1.2×10^{-4}

4.1.5　余弦　（$\overline{\text{COS}}\phi$）

1) 目　　的

固定された BP を持つ単精度固定小数点2進数の余弦を計算する。

計算できるアーギュメント θ は（$-2\text{rad}<\theta<2\text{rad}$）の範囲内のものである。

2) 呼出し方式

例1　A＝ $\overline{\text{COS}}\phi$（X）；

例2　B＝ $\overline{\text{COS}}\phi$（$X*Y+Z$）；

3) 使用時の条件と結果

データ：BP14 の変数名または，算術式を記述する。

結　果：BP15 の計算結果が $\overline{\text{COS}}\phi$ に代入される。

4) プログラム容量
224ワード
5) 演算方式
テーブル・ルック・アップ
6) 実行時間　（最大）
135 μsec
7) 最大誤差
1.2×10^{-4}

4.1.6　余弦　（$\overline{\text{COS}}$：）

1) 目　　的

指定された BP を持つ単精度固定小数点2進数の余弦を計算する。

計算できるアーギュメント θ は（$-2\text{rad}<\theta<2\text{rad}$）の範囲内のものである。

2) 呼出し形式

例1　A＝CŌS：(X, BP)；

例2　B＝CŌS：(X, 15)；

3) 使用時の条件と結果

データ：変数名又は，数値を記述する。

BP：データの BP を整数型で指定する。

($12 \leqq BP \leqq 15$)

結　果：BP15の計算結果がCŌS：に代入される。

4) プログラム容量

199ワード

5) 演算方式

テーブル・ルック・アップ

6) 実行時間　(最大)

111 μsec

7) 最大誤差

1.2×10^{-4}

4.1.7　正接　(TAN：)

1) 目　　的

指定された BP を持つ単精度固定小数点2進数の正接を計算する。

計算できるアーギュメント θ は(-1.4 rad$< \theta <$ 1.4 rad)の範囲内のものである。

2) 呼出し形式

例1　A＝TAN：(X, $BP1$, $BP2$)；

例2　B＝TAN：(X, 14, 9)；

3) 使用時の条件と結果

データ：変数名又は数値を記述する。

$BP1$：データの BP を整数型で指定する。

($9 \leqq BP1 \leqq 15$)

$BP2$：結果の BP を整数型で指定する。

($9 \leqq BP2 \leqq 15$)

結　果：指定された BP の結果がTAN：に代入される。

4) プログラム容量

294ワード

5) 演算方式

テーブル・ルック・アップ

6) 実行時間　(最大)

247 μsec

7) 最大誤差

9.2×10^{-4}

4.1.8　逆正弦　(ASIN)

1) 目　　的

固定された BP を持つ単精度固定小数点2進数の逆正弦を計算する。

計算できるアーギュメント X は($-1 < X < 1$)の範囲内のものである。

2) 呼出し方式

例1　A＝ASIN(X)；

例2　B＝ASIN($X*Y+Z$)；

3) 使用時の条件と結果

データ：BP14の変数名または算術式を記述する。

結　果：BP14の計算結果がASIN に代入される。

4) プログラム容量

236ワード

5) 演算方式

テーブル・ルック・アップ

6) 演算時間　(最大)

161 μsec

7) 最大誤差

2.5×10^{-3}

4.1.9　逆正弦　(ASINϕ)

1) 目　　的

固定された BP を持つ単精度固定小数点2進数の逆正弦を計算する。

計算できるアーギュメント X は($-1 < X < 1$)の範囲内のものである。

2) 呼出し方式

例1　A＝ASINϕ (X)；

例2　B＝ASINϕ ($X*Y+Z$)；

3) 使用時の条件と結果

データ：BP15の変数名または算術式を記述する。

結　果：BP14の計算結果がASINϕに代入される。

4) プログラム容量

237ワード

5) 演算方式

テーブル・ルック・アップ

6) 演算時間　(最大)

164 μsec

7） 最大誤差

2.5×10^{-3}

4.1.10 逆正弦 （ASIN：）

1） 目　的

固定された *BP* を持つ単精度固定小数点2進数の逆正弦を計算する。

計算できるアーギュメント *X* は（$-1 < X < 1$）の範囲内のものである。

2） 呼出し方法

例1　*A*＝ASIN：(X)；

例2　*B*＝ASIN：(X*Y＋Z)；

3） 使用時の条件と結果

データ：*BP*14の変数名または算術式を記述する。

結　果：*BP*14の計算結果がASIN：に代入される。

4） プログラム容量

213ワード

5） 演算方式

テーブル・ルック・アップ

6） 演算時間 （最大）

133 μsec

7） 最大誤差

2.5×10^{-3}

4.1.11 逆正接 （ATAN）

1） 目　的

固定された *BP* を持つ単精度固定小数点2進数の逆正接を計算する。

計算できるアーギュメント X は（$-2 < X < 2$）の範囲内のものである。

2） 呼出し方式

例1　*A*＝ ATAN(X)；

例2　*B*＝ ATAN(X*Y＋Z)；

3） 使用時の条件と結果

データ：*BP*14の変数名または算術式を記述する。

結　果：*BP*14の計算結果がATANに代入される。

4） プログラム容量

392ワード

5） 演算方式

テーブル・ルック・アップ

6） 実行時間 （最大）

232 μsec

7） 最大誤差

2.4×10^{-4}

4.1.12 逆正接 （ATAN：）

1） 目　的

指定された *BP* を持つ単精度固定小数点2進数の逆正接を計算する。

計算できるアーギュメント X は（$-64 < X < 64$）の範囲のものである。

2） 呼出し形式

例1　*A*＝ATAN：(X,*BP*1,*BP*2)；

例2　*B*＝ ATAN：(X, 14, 9)；

3） 使用時の条件と結果

データ：変数名又は数値を記述する。

*BP*1：データの *BP* を整数型で指定する。

（$9 \leqq BP1 \leqq 14$）

*BP*2：結果の *BP* を整数型で指定する。

（$9 \leqq BP2 \leqq 14$）

結　果：指定された *BP* の結果がATAN：に代入される。

4） プログラム容量

365ワード

5） 演算方式

テーブル・ルック・アップ

6） 実行時間 （最大）

208 μsec

7） 最大誤差

2.4×10^{-4}

4.1.13 指数関数 （EXP）

1） 目　的

固定された *BP* を持つ単精度固定小数点2進数の指数関数を計算する。

計算できるアーギュメント X は（$-1 \leqq X \leqq 1$）の範囲内のものである。

2） 呼出し方法

例1　*A*＝EXP(X)；

例2　*B*＝EXP(X*Y＋Z)；

3） 使用時の条件と結果

データ：$BP14$の変数名または算術式を記述する。

結　果：$BP13$の計算結果がEXPに代入される。

4) プログラム容量

147ワード

5) 演算方式

近似式（三菱電機製，詳細不明）

6) 演算時間　（最大）

188 μsec

7) 最大誤差

1×10^{-8}

4.1.14 自然対数　（LOG）

1) 目　　的

固定されたBPを持つ単精度固定小数点2進数の自然対数を計算する。

計算できるアーギュメントXは（0.03＜X＜16）の範囲内のものである。

2) 使用時の準備と結果

例1　A = LOG(X)；

例2　A = LOG($X*Y+Z$)；

3) 使用時の条件と結果

データ：$BP11$の変数名または算術式を記述する。

結　果：$BP13$の計算結果がLOGに代入される。

4) プログラム容量

63ワード

5) 演算方式

近似式（三菱電機製，詳細不明）

6) 演算時間　（最大）

195 μsec

7) 最大誤差

2×10^{-8}

4.1.15 平方根　（SQRT）

1) 目　　的

正の単精度固定小数点2進数の平方根を計算する。アーギュメントXは負であってはならない。

2) 呼出し方式

例1　A = SQRT(X)；

例2　B = SQRT($X*Y+Z$)；

3) 使用時の条件と結果

データ：$BP(2n)$の変数名または算術式を記述する。

結　果：$BP(n+7)$の計算結果がSQRTに代入される。

4) プログラム容量

64 ワード

5) 演算方式

1次最良近似多項式を初期値として，ニュートン・ラプソン法による反復で求める。

1次最良近似多項式

$$f^*(x)=\begin{cases} Ax+B & \left(\frac{1}{2}\leqq x\leqq 1 \text{の場合}\right) \\ 2Bx+\frac{1}{2}A & \left(\frac{1}{4}\leqq x\leqq \frac{1}{2} \text{の場合}\right) \end{cases}$$

ニュートン・ラプソン法

$$a_n+1=\frac{1}{2}\ (a_n+x/a_n)$$

6) 演算時間　（最大）

155 μsec

7) 最大誤差

$\sim 1 \times 10^{-4}$

4.1.16 絶対値　（ABS）

1) 目　　的

任意のBPを持つ固定小数点2進数の絶対値を計算する。

なお，単精度・倍精度の判断はRTSL-Ⅱがコンパイル時に自動的に行い，演算を実行する。

2) 呼出し方法

例1　A = ABS(X)；

例2　B = ABS($X-Y$)；

3) 使用時の条件と結果

データ：任意のBPを持つ変数名または算術式を記述する。

結　果：データのBPと同一のBPを持つ計算結果がABSに代入される。

4) プログラム容量

単精度：20 ワード

倍精度：27 ワード

5) 演算時間　（最大）

単精度：21 μsec

倍精度：24 μsec

4.1.17 符号 （SIGN）

1) 目 的

任意の*BP*を持つ単精度固定小数点2進数の符号*bit*により，結果を出力する。

2) 呼出し方法

例1 A＝SIGN(X)；

例2 B＝SIGN($C-D$)；

3) 使用時の条件と結果

データ：任意の*BP*を持つ変数名または算術式を記述する。

結 果：*BP* 15 の計算結果がSIGN に代入される。

正符号時：/7FFF (16進表示) (0.9999 @J15)

負符号時：/8000 (16進表示) (－1.0 @J15)

4) プログラム容量

19 ワード

5) 演算時間 （最大）

25 μsec

4.1.18 剰余 （MOD）

1) 目 的

任意の*BP*を持つ固定小数点2進数の除算を行い，余りを結果として整数値で出力する。

但し，計算$\{X-(X/Y)*Y\}$を行うための変数は，同一の*BP*であること。

2) 呼出し方法

例1 A＝MOD(DATA1, DATA2)；

例2 N＝MOD(X, I)；

3) 使用時の条件と結果

データ1：被除数となる変数名を記述する。

データ2：除数となる変数名を記述する。

但し，*BP*は被除数と同一である。

結 果：被除数，除数と同一の*BP*を持つ整数値の計算結果がMOD に代入される。

4) プログラム容量

単精度：112 ワード

倍精度：364 ワード

5) 演算時間 （最大）

単精度：104 μsec

倍精度：405 μsec

4.1.19 最大値 （MAX）

1) 目 的

任意の個数の数値から最大値を選び出す。

2) 呼出し方法

例1 MAX J＝MAX(A, B, C, D)；

例2 M＝MAX($D_1, D_2, \cdots D_n$)；

3) 使用時の条件と結果

データ：*BP*は任意であるが，同一であること。
データの個数は任意である。

結 果：データの中で最大値のものがMAXに代入される。
*BP*は，データと同一である。

4) プログラム容量

単精度：103 ワード

倍精度： 99 ワード

演算時間 （最大）

単精度：110＋($n-2$)×20 μsec

倍精度：107＋($n-2$)×70 μsec

但し，$n \geqq 2$（nは数値の個数）

4.1.20 最小値 （MIN）

1) 目 的

任意の個数の数値から最小値を選び出す。

2) 呼出し方法

例1 MINK＝MIN(A, B, C, D)；

例2 M＝MIN($D_1, D_2, \cdots D_n$)；

3) 使用時の条件と結果

データ：*BP*は任意であるが同一であること。
データの個数は任意である。

結 果：データの中で最小値のものがMIN に代入される。
*BP*は，データと同一である。

4) プログラム容量

単精度：103 ワード

倍精度： 98 ワード

5) 演算時間 （最大）

単精度：107＋($n-2$)×20 μsec

倍精度：110＋($n-2$)×20 μsec

但し，$n \geqq 2$（nは数値の個数）

4.1.21 制限 （LIMIT）

1) 目 的

入力データに対して，指定された範囲内のデータだけを出力する。($L1 \leqq$出力$\leqq L2$)

2) 呼出し方法

例1 A＝LIMIT(DATA, $L1$, $L2$)；

例2 B＝LIMIT(A, −2, 5)；

3) 使用時の条件と結果

データ：制限を加えようとする変数名を記述する。BP は任意である。

下限値：制限の範囲の下限を設定する。($L1$)
BP はデータと同一であること。

上限値：制限の範囲の上限を設定する。($L2$)
BP はデータと同一であること。

結　果：データと同一の BP を持つ結果がLIMIT に代入される。

4) プログラム容量

単精度： 81 ワード

倍精度：129 ワード

5) 演算時間 （最大）

単精度：217 μsec

倍精度：272 μsec

4.1.22 不感帯 (DBND)

1) 目　的

入力データに対して，指定された範囲外のデータはそのまま出力し，範囲内は ϕ を出力する。

2) 呼出し方法

例1 A＝DBND(DATA, $B1$, $B2$)；

例2 B＝DBND(A, −2, 5)；

3) 使用時の条件と結果

データ：不感帯を設定する変数名を記述する。BP は任意である。

下限値：不感帯の下限を設定する。($B1$)
BP はデータと同一であること。

上限値：不感帯の上限を設定する。($B2$)
BP はデータと同一であること。

結　果：データと同一の BP を持つ結果がDBND に代入される。

4) プログラム容量

単精度： 71 ワード

倍精度：129 ワード

5) 演算時間 （最大）

単精度：217 μsec

倍精度：266 μsec

4.1.23 乱数 (RANDOM)

1) 目　的

入力データから擬似一様乱数を発生させる。乱数 N の範囲は ($0 \leqq N \leqq 2^{16}-1$) である。

2) 呼出し方法

例1 A＝RANDOM(X)；

例2 N＝RANDOM(0.5 @J15)；

3) 使用時の条件と結果

データ：初期値となる乱数値を設定する。BP は15である。

結　果：BP 15 の擬似一様乱数が RANDOM に代入される。

4) プログラム容量

33 ワード

5) 演算時間 （最大）

50 μsec

4.1.24 ビットサーチ (TRANSϕ, TRANS 1)

1) 目　的

1ワードのビット列の中から最上位にある ϕ または1の位置を符号ビットからのずれで求める。

ただし，目的とする ϕ または1が無い場合には −1@Iが答えとなる。

2) 呼出し形式

$X\phi$＝TRANSϕ (X)；

X1＝TRANS 1 (X)；

3) 使用時の条件と結果

入出力変数ともI型を使用する。

ϕ ビットを検出するには TRANSϕ を，1ビットでは TRANS 1 を使用する。目的のビットが無い場合，結果は −1@I である。

4) プログラム容量

TRANSϕ ： 32 ワード

TRANS 1 ： 35 ワード

5) 演算時間 （最大）

TRANSϕ ： 160.2 μsec

TRANS 1 ： 148.9 μsec

4.1.25 1変数関数発生 (FG 1L)

1) 目　的

1変数のテーブル・ルック・アップ方式で1変数関数値を求める。

プログラム内で作成した1変数関数のデータ・テーブルを与え，このファンクションで，変数に

対応する関数値を求める。

2) 呼出し形式

F＝FG1L(X, $X\phi$, $Y\phi$, N)；

3) 使用時の条件と結果

Xは入力変数，$X\phi$, $Y\phi$ は一変数関のブレークポイントの座標を示し，特に$X\phi$, $Y\phi$はその先頭アドレスを与える。Nはブレークポイント数を与える。

Xと$X\phi$のBPは同一のこと。

結果は，Xに対応する関数値が線形補間で計算され，FG1Lに代入される。

4) プログラム容量

532ワード

5) 実行時間 （平均）

391.1 μsec

4.2 サブルーチン

RTSL-Ⅱのライブラリには，表3に示してあるシミュレーションに必要とされているいくつかのサブルーチンが用意されている。以下，各サブルーチンの目的と使用条件等を述べる。

4.2.1 パック （PACK）

1) 目 的

16ワードのデータの値（1かϕ）に対応した16ビットをもつ1ワードを出力する。

2) 呼出し方法

CALL PACK (DATA, OUT)；

3) 使用時の条件と結果

データ：入力データ（16データ）の先頭変数名を記述する。

結 果：先頭データの内容（1かϕ）に対応したビット（1かϕ）を持つ1ワードをセットする。この時先頭ビット（MSB）から対応させる。

4) プログラム容量

39 ワード

5) 演算時間 （最大）

170 μsec

4.2.2 アンパック （UNPACK）

1) 目 的

16ビット1ワードのデータの各ビット内容を16ワードの連続したデータに分解し格納する。

2) 呼出し方法

CALL UNPACK (DATA, BUF)；

3) 使用時の条件と結果

データ：16ワードに分解した内容に従って16ワードのデータがセットされる。BUFは16ワードの先頭データ名である。

この時ビット1には整数1がセットされ，ϕにはϕがセットされる。

4) プログラム容量

38 ワード

5) 演算時間 （最大）

170 μsec

4.2.3 BCD変換 （B：BCD）

1) 目 的

整数のバイナリデータを4桁のBCDコードに変換する。

2) 呼出し方法

CALL B：BCD (BINARY, BCD)；

3) 使用時の条件と結果

データ：整数型データの変数名を記述する。

結 果：指定された変数名に変換されたデータが代入される。（BCDに代入される）

4) プログラム容量

69 ワード

5) 演算時間 （最大）

118 μsec

4.2.4 BCD逆変換 （BCD：B）

1) 目 的

4桁のBCDコードを整数のバイナリデータに変換する。

2) 呼出し方法

CALL BCD：B(BCD, BINARY)；

3) 使用時の条件と結果

データ：BCDデータの変数名を記述する。

結 果：指定された変数名に変換されたバイナリデータが代入される。
（BINARYに代入される）

4) プログラム容量

55 ワード

5) 演算時間 （最大）

106 μsec

4.2.5 オイラ積分 (EULJ, EULK)

1) 目 的

オイラ積分を行う。単精度と倍精度のデータに対応して，2種類用意されている。

2) 呼出し方法

i) 単精度 CALL EULJ (X, XD)；

ii) 倍精度 CALL EULK (X, XD)；

3) 使用時の条件と結果

$X = X + T * XD$ なる演算が行われる。ここで，Tは積分きざみ幅で，このプログラムが動作している周期が自動的に用いられる。

結果はXにセットされる。Xが単精度の場合は，EULJを用い，XDも単精度である。倍精度の場合はEULKを用いる。この時，XとXDのBPは同一とする。

4) プログラム容量

単精度： 37 ワード

倍精度： 77 ワード

5) 演算時間 (平均)

単精度： 71.5 μsec

倍精度： 546.5 μsec

4.2.6 オイラ積分内部分割 (EULJN)

1) 目 的

オイラ法による積分を行うが，プログラム演算周期をn分割して積分を実行する。

2) 呼出し方法

CALL EULJN (X, XD, N)；

3) 使用時の条件と結果

$X = X + \frac{T}{N} * XD$ なる演算が行われる。

ここで，Tはプログラムの演算周期で，実行時のTが自動的に用いられる。

X, XDは単精度とし，Nは整数とする。

4) プログラム容量

47 ワード

5) 演算時間 (平均)

100.2 μsec

4.2.7 アダムス2次積分 (AB2J, AB2K)

1) 目 的

アダムス・バッシュフォースの2次の積分を行う。単精度と倍精度のデータに対応して，2種類用意されている。

2) 呼出し方法

i) 単精度 CALL AB2J (X, XD, W)；

ii) 倍精度 CALL AB2K (X, XD, W)；

3) 使用時の条件と結果

$X = X + \frac{T}{2}(3XD - W)$なる演算が行われ，演算後XDの値が$W$にセットされる。$T$は積分きざみ幅で，このプログラムが動作している周期が自動的に用いられる。

X, XD, Wの精度とBPを合わせて用いること。

4) プログラム容量

41 ワード

5) 演算時間 (平均)

単精度： 98.9 μsec

倍精度： 598.3 μsec

4.2.8 一次遅れ (LAG：1)

1) 目 的

時定数を指定した一次遅れを行う。

2) 呼出し方法

CALL LAG：1 (U, X, TAU, P)；

3) 使用時の条件と結果

次の演算を行う。

$$X = \frac{1}{1 + TAU \cdot S} \cdot U$$

(Sはラプラス演算子)

積分はオイラ積分EULJ を用いる。

TAUは時定数でPはTAUのBPであり，

$9 \leqq P \leqq 15$の範囲で指定する。

UとXは単精度としBPは同一であること。

4) プログラム容量

66 ワード

5) 演算時間 (平均)

351.5 μsec

6) 外部サブルーチン

EULJ

4.2.9 擬似正規乱数 (GAUSS)

1) 目 的

正規乱数(白色ノイズ)を発生させる。

2) 呼出し方法

CALL GAUSS (V, N)；

3) 使用時の条件と結果

発生の方法は一様乱数を発生させ，中心極限定

理の適用でこの一様乱数を12回加えて，正規乱数とする。固定小数点演算のため，3σのみが±1.0に入る様に修正している。

Vは正規乱数の結果で，Nは一様乱数発生の初期値である。VのBPは15である。

4) プログラム容量

94 ワード

5) 演算時間 （平均）

626.8 μsec

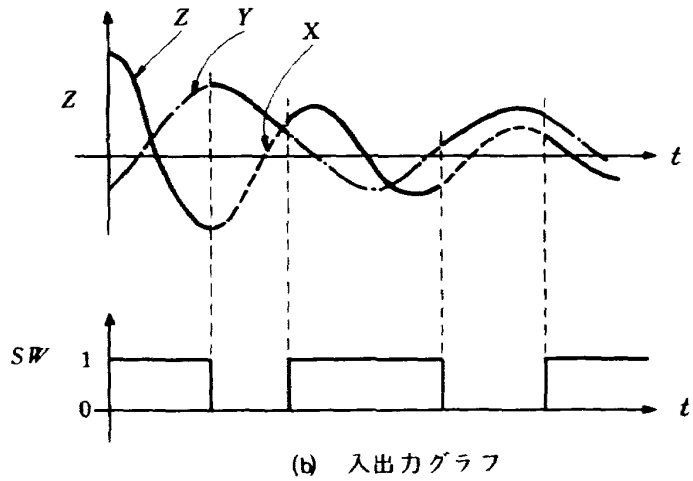
付録5 RTSL-Ⅱによるプログラム記述例

(例1) 混合演算の例

計算機内の他のTASKによって発生されたXとYの信号を外部のスイッチ信号（SW）により切換えて，外部に出力（Z）するプログラムを付図5(a)に示す。この時のZ信号の動きを同図(b)に示す。

```
/* EXAMPLE PROGRAM      */
   INPUT;
     DCL   SW     I;
     DCL   X      J15;
     DCL   Y      J15;
   END;
/*                       */
   OUTPUT;
     DCL   Z      J15;
   END;
/*                       */
   TASK EXAMPLE1;
     Z = (SW==1)*X + (SW==0)*Y ;
   EXIT;
   END;
```

(a) プログラム



(b) 入出力グラフ

付図5 混合演算の例

(例2) 線形飛行縦運動の応答

昇降舵面あるいはスラスト変化による縦運動の応答を求めるプログラムの例を示す。

(1) 演算式

信号入力

$$\begin{cases} \delta_e = K\delta_e \cdot \delta_{ei} \\ \delta_T = K\delta_T \cdot \delta_{Ti} \end{cases} \quad \cdots\cdots (1)$$

運動方程式

$$\begin{cases} \dot{u} = X_u \cdot u + X_\alpha \cdot \alpha - \dfrac{g\cos\gamma_o}{180}\theta + X_{\delta T} \cdot \delta_T \\ \dot{\alpha} = Z_u \cdot u + Z_\alpha \cdot \alpha + \dot{\theta} - \dfrac{g}{U_o}\sin\gamma_o \cdot \theta + Z_{\delta e} \cdot \delta_e + Z_{\delta T} \cdot \delta_T \\ \ddot{\theta} = M_u \cdot u + M_{\dot{\alpha}} \cdot \dot{\alpha} + M_\alpha \cdot \alpha + M_q \cdot \dot{\theta} + M_{\delta e} \cdot \delta_e + M_{\delta T} \cdot \delta_T \end{cases} \quad \cdots\cdots (2)$$

出力

$$\begin{cases} u_R = C_u \cdot u \\ \alpha_R = C_\alpha \cdot \alpha \\ \theta_R = C_\theta \cdot \theta \\ \dot{\theta}_R = C_{\dot{\theta}} \cdot \dot{\theta} \\ \delta_{eR} = C_{\delta e} \cdot \delta_e \\ \delta_{TR} = C_{\delta T} \cdot \delta_T \end{cases} \quad \cdots\cdots (3)$$

入力は外部のポテンショメータの出力電圧を用いる。（δ_{ei}, δ_{Ti} は最大1.0で入力される）

出力はu, α, θ, $\dot{\theta}$, δ_e, δ_T をペン書きレコーダ（アナログ信号）に記録する。この時，最大値が1.0に対応する様にスケールを変更することが必要である。（式(3)のC_u, C_α, C_θ, $C_{\dot{\theta}}$, $C_{\delta e}$, $C_{\delta T}$ はこのための変更係数である）

(2) 飛行条件および空力微係数

Queen Air機の着陸形態を想定し，文献4)の微係数を用いる。飛行条件は，

$g = 9.8$ m/sec²

$\gamma_o = -3$ deg

$U_o = 100$ kts $= 51.4$ m/sec

微係数の値を付表9に示す。演算は単精度で行うものとし，付表8より各値に対する属性を定める。またプログラムの際の記号を合わせて示す。

(3) 各変数の変化範囲

付表9 微 係 数 値

微 係 数	値	ディメンジョン	属 性	プログラム内記号
X_u	−0.06	1/sec	J 19	XU
X_α	0.035	m/sec²・deg	J 19	XA
X_δ	0.055	m/sec²・%	J 19	XDET
Z_u	0.40	deg/m	J 16	ZU
Z_α	−0.90	1/sec	J 15	ZA
Z_{δ_e}	−0.40	1/sec	J 12	ZDE
$Z\delta_T$	−0.07	deg/sec・%	J 18	ZDET
M_u	0.0	1/m・sec	J 15	MU
$M_{\dot{\alpha}}$	−0.30	1/sec	J 16	MAD
M_α	−0.90	1/sec²	J 15	MA
M_q	−1.3	1/sec	J 14	MQ
$M\delta_e$	−5.0	1/sec²	J 12	MDE
$M\delta_T$	0.13	deg/sec²・%	J 17	MDET
$C_1=\frac{g\cos\gamma_0}{180}$	−0.054	m/sec²・deg	J 19	C 1
$C_2=\frac{g\sin\gamma_0}{U_0}$	−0.001	1/sec	J 24	C 2

RTSL−Ⅱでは固定小数点数演算であるので，オーバースケールとならないように，各変数の変化範囲をあらかじめ定めておくことが必要である。本プログラムの各変数については付表10に示すように定める。プログラム内での記号も合わせて同表に示す。

ペン書きレコーダへの出力は，最大値(1.0)をいくらに対応させるかが問題で，これも付表10の値を対応させるとし，これらの最大値を1.0に対応させるため，付表11に示すスケール変換係数を定める。

(4) プログラム記述例

本例題を以上の条件のもとにRTSL−Ⅱのフォーマットで記述した例を付図6に示す。

積分としては，RTSL−ⅡライブラリのAB2Jを用いる。

(5) ステップ応答例

本プログラムをFSK−Ⅱを用いて解いた例を付図7に示す。同図にはFORTRANでRunge−Kutta 4次の積分を用いて解いた値も示してある。

付表 10　変数の変化範囲

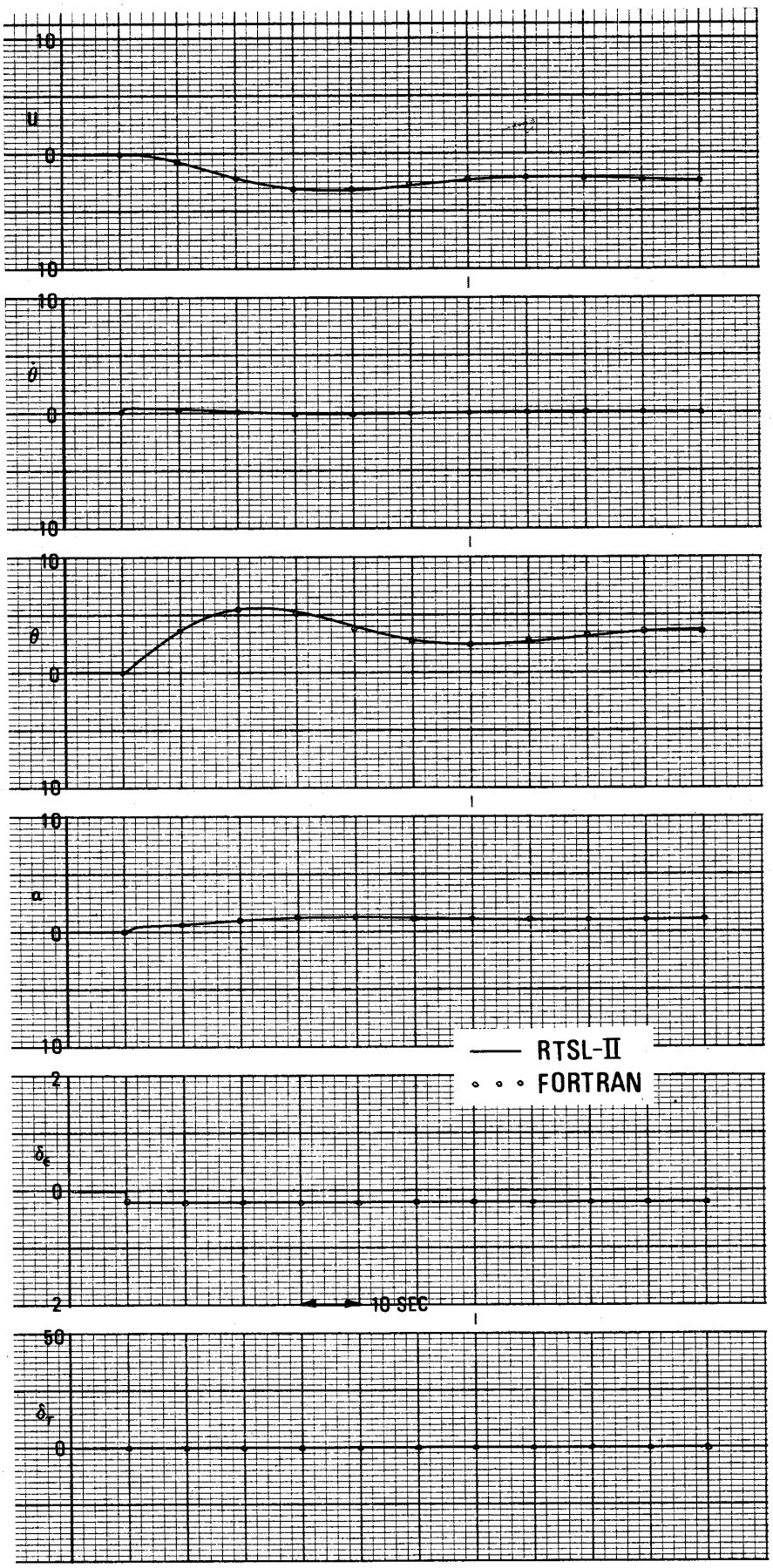
変　数	変化範囲	ディメンジョン	属　性	プログラム内記号
u	± 10	m/sec	J 11	U
α	± 10	deg	J 11	ALPHA
$\dot{\alpha}$	± 10	deg/sec	J 11	ALD
θ	± 10	deg	J 11	THETA
$\dot{\theta}$	± 10	deg/sec	J 11	THD
$\ddot{\theta}$	± 10	deg/sec^2	J 11	THDD
δ_e	± 2	deg	J 14	DE
δ_T	0 ～ 50	%	J 09	DET

付表 11　スケール変換係数

係　数	値	ディメンジョン	属　性	プログラム内記号
K_{δ_e}	2.0	deg	J 13	KDE
K_{δ_T}	50.0	%	J 09	KDET
C_u	0.1	1/m/sec	J 18	CU
C_α	0.1	1/deg	J 18	CA
$C_{\dot{\theta}}$	0.1	1/deg/sec	J 18	CT
C_{δ_e}	0.5	1/deg	J 14	CDE
C_{δ_T}	0.02	1/%	J 20	CDET

```
/*----------------------------------------*/
/* EXAMPLE PROGRAM 2                      */
/* AIRCRAFT LONGITUDINAL EQUATION         */
/*----------------------------------------*/
   INPUT;
      DCL  SW1       I  ;
      DCL  SW2       I  ;
      DCL  DEI       J15;
      DCL  DETI      J15;
   END;
/*----------------------------------------*/
   OUTPUT;
      DCL  U:R       J15;
      DCL  THD:R     J15;
      DCL  THETA:R   J15;
      DCL  ALPHA:R   J15;
      DCL  DE:R      J15;
      DCL  DET:R     J15;
   END;
/*----------------------------------------*/
/*----------------------------------------*/
   TASK QA;
/* VARIABLES */
      DCL  U         J11 INIT(0,0);
      DCL  UD        J11;
      DCL  ALPHA     J11 INIT(0,0);
      DCL  ALD       J11;
      DCL  THETA     J11 INIT(0,0);
      DCL  THD       J11 INIT(0,0);
      DCL  THDD      J11;
      DCL  DE        J14;
      DCL  DET       J09;
      DCL  U1        J11 INIT(0,0);  /* WORK AREA */
      DCL  A1        J11 INIT(0,0);  /*  "        */
      DCL  T1        J11 INIT(0,0);  /*  "        */
      DCL  T2        J11 INIT(0,0);  /*  "        */
/* COEFFICIENTS */
      DCL  XU        J19 INIT(-0,06);
      DCL  XA        J19 INIT(0,035);
      DCL  XDET      J19 INIT(0,055);
      DCL  ZU        J16 INIT(-0,40);
      DCL  ZA        J15 INIT(-0,90);
      DCL  ZDE       J16 INIT(-0,40);
      DCL  ZDET      J18 INIT(-0,07);
      DCL  MU        J15 INIT(0,0);
      DCL  MAD       J16 INIT(-0,30);
      DCL  MA        J15 INIT(-0,9);
      DCL  MQ        J14 INIT(-1,3);
      DCL  MDE       J12 INIT(-5,0);
      DCL  MDET      J17 INIT(0,13);
      DCL  C1        J19 INIT(-0,054);
      DCL  C2        J24 INIT(-0,001);
      DCL  KDE       J13 INIT(2,0);
      DCL  KDET      J09 INIT(50,0);
      DCL  CU        J18 INIT(0,1);
      DCL  CA        J18 INIT(0,1);
      DCL  CT        J18 INIT(0,1);
      DCL  CTD       J18 INIT(0,1);
      DCL  CDE       J15 INIT(0,5);
      DCL  CDET      J20 INIT(0,02);
/*----------------------------------------*/
/* CALCULATION START */
/* INPUT */
      IF  SW1 == 0  THEN  DE  = 0,0@J14 ;
                    ELSE  DE  = KDE*DEI ;
      IF  SW2 == 0  THEN  DET = 0,0@J09 ;
                    ELSE  DET = KDET*DETI ;
/* EQUATION */
      UD   = XU*U + XA*ALPHA + C1*THETA + XDET*DET ;
      THDD = MU*U + MAD*ALD + MA*ALPHA + MQ*THD + MDE*DE + MDET*DET ;
      ALD  = ZU*U + ZA*ALPHA + THD + C2*THETA +ZDE*DE + ZDET*DET ;
/* INTEGRAL */
      CALL  AB2J (U,UD,U1) ;
      CALL  AB2J (ALPHA,ALD,A1) ;
      CALL  AB2J (THETA,THD,T2) ;
      CALL  AB2J (THD,THDD,T1) ;
/* OUTPUT */
      U:R     =  CU*U ;
      THD:R   =  CTD*THD ;
      THETA:R =  CT*THETA ;
      ALPHA:R =  CA*ALPHA ;
      DE:R    =  CDE*DE ;
      DET:R   =  CDET*DET ;
/*----------------------------------------*/
/* TASK END */
      EXIT ;
      END ;
```

付図6　縦運動演算プログラム



付図7　縦運動ステップ応答

航空宇宙技術研究所資料479号

昭和57年11月発行

発行所 航空宇宙技術研究所
東京都調布市深大寺町1880
電話武蔵野三鷹(0422)47-5911(大代表) 〒182

印刷所 株式会社 共進
東京都杉並区久我山5－6－17

Printed in Japar