JAERI-M
8694

CANBERRA8100/QUANTAシステムのプログラミング
— "CLASS" 語によるプログラムの開発 —

1980年3月

吉田　廣・久保　克巳*

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

CANBERRA 8100／QUANTAシステムのプログラミング
－"CLASS"語によるプログラムの開発－

日本原子力研究所東海研究所原子炉工学部
吉田　廣・久保　克巳*

（1980年1月22日受理）

本報告は，CANBERRA 8100型マルチチャンネル波高分析器(以下MCA)と，DEC PDP-11／05ミニコンピュータ（以下計算機)とから成るCANBERRA-QUANTAシステム用ソフトウェアの特徴である，対話型プログラミング言語"CLASS"（＝Canberra Laboratory Automation Software System)の使用法，およびこれを使用して，半導体検出器により得られたガンマ線スペクトルのデータを処理・解析するために開発されたプログラムについて述べたものである。開発されたプログラムには，

(1) MCAで得られたガンマ線光電ピークのチャンネル番号と，そのエネルギーの値との関係式の係数を求めるもの。

(2) MCAで得られた光電ピーク部の全カウント数より，バックグランド成分の差引きを行うもの。

(3) 放射線源の調製時より使用時までの経過時間を日又は年単位で算出するもの。

等がある。

＊ 外来研究員：東京芝浦電気株式会社原子力技術研究所

Programming of Canberra Industries 8100/Quanta System
- Program Development with "CLASS" -

Hiroshi YOSHIDA and Katsumi KUBO*
Division of Reactor Engineering,
Tokai Research Establishment, JAERI

(Received January 22, 1980)

In this report are described usage of an interactive programming language "CLASS" (Canberra Laboratory Automation Software System) which is a feature the software for Canberra Industries 8100/Quanta System consisting of a Canberra Industries 8100 multichannel analyzer(MCA) and a PDP-11/05 mini-computer, and the programs with CLASS developed to process and analyze the data of gamma spectra obtained with semiconductor detectors. The programs are (1) to compute the coefficients in the formulae that relate the channel numbers of gamma-ray photopeaks obtained from MCA and the energy values; (2) to subtract the background component from the total count of a photopeak obtained from MCA, and (3) to calculate the lapse of time in days or years following the preparation of a radiation source.

Keywords: Programming Language, Gamma Spectra, Semiconductor Detectors, Multichannel Analyzers, Photopeaks, Background Subtraction, Data Processing, Program Development,

---

*On leave from Tokyo Shibaura Electric Corporation.

目　　次

Contents

List of Tables

List of Figures

1. 序　　論

本報告では，ガンマ線スペクトルのデータ収集，及び解析用として導入された,CANBERRA−QUANTA System の専用プログラミング言語"CLASS"(＝Canberra Laboratory Automation Software System)の使用法について説明し，[1]次でCLASS 語を使用して開発されたプログラムについて述べる。

"CLASS"のSoftwareは Interpreter であって，命令の実行は入力と同時に開始される。これはDECのPDP−11／05 Mini − computer 用のOperating system "RT−11"[2]にOption として附加されたものである。

CLASSは，

a　数値計算，条件判断の機能

b　Canberra 8100及び8700マルチチャンネル波高分析器の制御

c　種々の入出力機器のサポート

等の諸機能を有する。

1.1　記号類及び注意事項

以後，本報告に於ては，次に述べる記号類を断りなく使用する。

(1) コンソールターミナル（Teletype又はDECwriter，以後単にコンソールと呼ぶ）のKeyboard による入力，及びコンソールの印字部への出力を本文中で例示する際に，次の記号類を適宜使用する。

a．␣　；入力の際はSpace key を1回打って入力を区切ることを示す。印字の場合はSpace 1個が出力されることを示す。

b．＜TAB＞　；TAB(又はForm feed)keyを打つこと，又はTAB信号の出力があることを示す。これにより，印字は各行の左端より8字目毎に設定されているTab point の所までSkipする。

c．＜CR＞　；Carriage return(復帰)keyを打つことを示す。これにより印字部はその行の左端に戻る。プログラムがKeyboardの入力を受入れている時，System は引続き自動的にLine feed(改行)信号を発生するため，プログラムの修正等で必要な場合を除きLine feed keyを打ってはならない。

d．Systemがコンソールの印字部に出力を行う時は，改行の都度Carriage return − Line feed の2信号が引続いて印字部に対して出力される。これらを"<CR><LF>"と表示することがある。

e．Keybordへの入力と，印字部への出力とを同時に表示する時，両者を次のアンダラインで区別することがある。

——（直線のアンダライン）　；印字出力

～～～（波線のアンダライン）　；Keyboardへの入力

（一方のみを使用した時，残部は他の一方の記号が省略されたものと見なす。）

f　この他，字間・行間の空白は，支障ない場合適宜に省略して表示することがある。

(2)　Programの入・出力に於て，"↑"（上向き矢印）記号（"N" keyの上段）を使用することがある。これはTeletype使用の場合であって，本Systemの如くDECwriter使用の時は，これを"∧"（仮に"Circumflex"と呼ぶ。"6" keyの上段）で代用する。コンソールの出力をCopyした実例を参照の時に注意を要する。

(3)　Systemに対しある種の制御をKeyboardより行う時に，CTRL（Control）keyを押しながら同時に他のKey（例えば"C"）を押す場合がある。これらのKeyの組合せを"CTRL／C"の如く表示する。

(4)　Manualその他の資料類には記載されていないか不確実な事項で，著者のテストにより判明した注意すべき事項を，本文中に随時"＊＊〔……〕＊＊"なる記号で図って記載した。

この他，DECwriterは，大文字・小文字の使い分けが可能であるが，本System に於ては，Shift key又はCAPS（＝Capitals）lock keyの操作に関係なく，すべてのアルファベットは大文字として扱われる。

1.2　システムの機器構成

現用SystemのHardwareには，1979年度中に，I／O及びメモリの増設が予定されている。本報告では，Hardware，Software共に同年3月末現在の状態に基いて記述する。

SystemのHardwareの構成はFig.1.1の通りであり，その内訳は以下の通りである。

(1)　マルチチャンネル波高分析器（以下MCA）及び入・出力機器（以下I／O）

a.　CANBRRA 8100 MCA（メモリは4096 channel分実装）

b.　磁気テープデッキ（PERTEC）

c.　磁気テープデッキ制御ユニット（PERTEC）

d.　Teletype ASR 33

e.　高速紙テープパンチ（Tally）

f.　X－Y Recorder（YHP 7004B）

(2)　Mini－computer及びI／O

a.　DEC PDP－11／05 Mini－computer（メモリ16k word実装，以下計算機）

b.　コンソールターミナル装置（DECwriter使用，以下コンソール）

c.　高速紙テープリーダー及びパンチ（DEC）

d.　Disk cartridge drive（System deviceとして使用）

e.　Cassette drive（ユニット2台を実装）

f.　Digital plotter（岩通 DPL－602 Plotter）

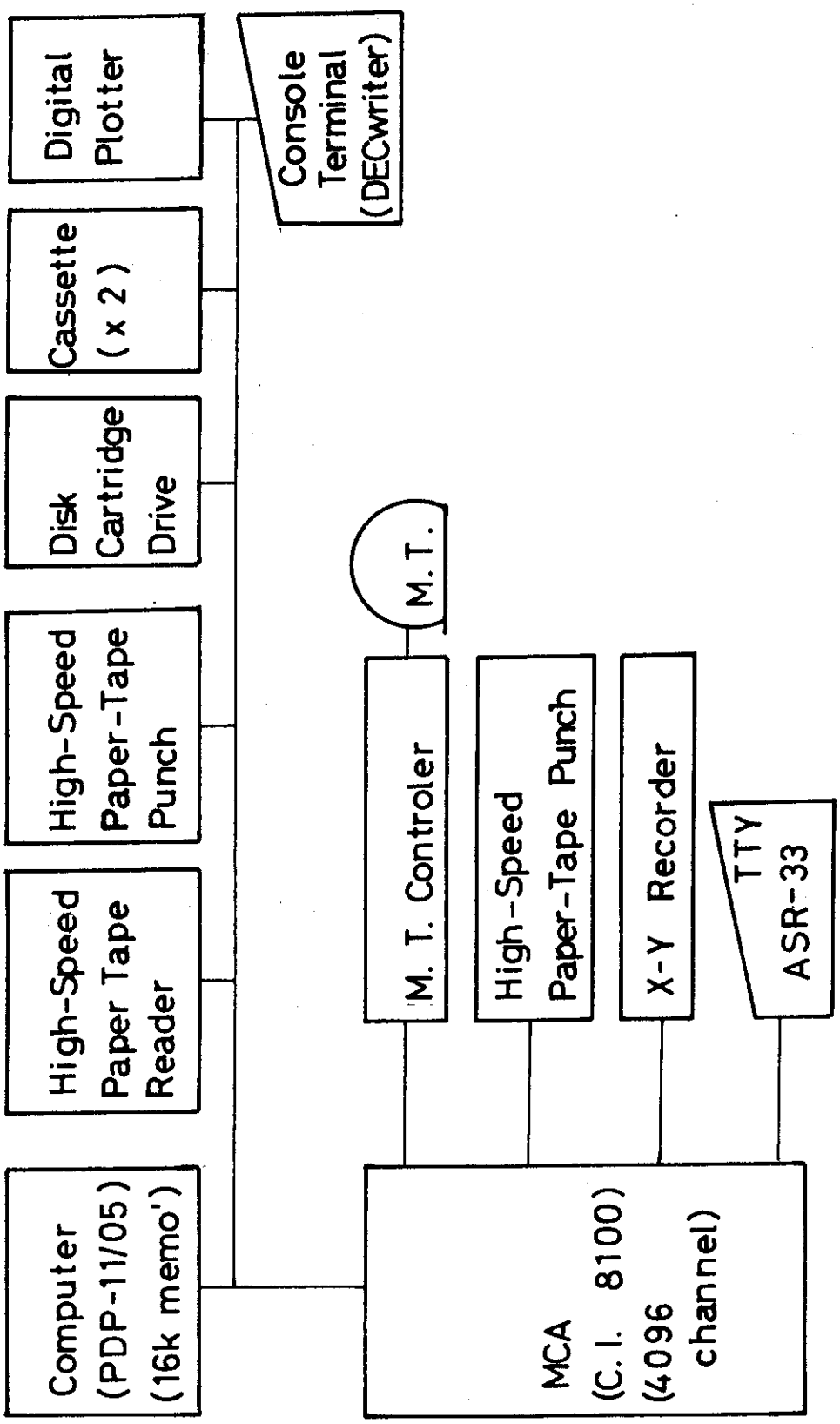


Fig.1.1 Schematic diagram of system hardware construction

1.3 RT－11 Systemの概略[3)]

本節に於ては，RT－11 Sytemの操作について最小限必要な事項を説明し，詳細は別の機会にゆずる。

1.3.1 Systemの起動及び停止

(1) Fig. 1.2にComputer及びDisk drive のパネル面操作部の略図を示す。

(2) 電源を投入する時は，先ずDisk drive のRUN／LOAD switch がLOADになっていることを確認の上，Computer パネル上のKey switch をOFFからONの位置に廻す。Disk drive パネル上の標示灯は先ず"PWR"が点灯し，少し間を置いて"LOAD"が点灯する。"LOAD"点灯状態の時のみDisk の着脱が可能となるから，System software の入ったDisk cartridge を装着する。次にRUN／LOAD switch をRUNにし，Disk が定常廻転の状態になって，"RDY"・"ONCYL"が点灯し，"LOADが消灯して使用可能な状態となった時，WTPROT(＝Write protect) switch を押し，"WTPROT"が点灯したことを確認して次の操作を行う[註1)]。

（コンソール及びDigital plotter は別電源となっているから，それぞれのSwitch を個別に操作する。）

(3) Computer 前面のSwitch registor を $173100_{(8)}$ にSetし，LOAD ADRS(＝Load adress) switch を押し下げる。

（$173100_{(8)}$ の"$_{(8)}$"は8進数表示であることを示す。ADRESS／DATAのSwitch 16個を右端より3個1組とし，各組について右側より1－2－4なるWeight をつけ，On にSetされたものについてのWeight の和を各組について算出すれば，これらSwitch の状態が6桁の8進数で表示される。最上桁はSwitch が1個のみであるから，状態は $0_{(8)}$ 又は $1_{(8)}$ の2通りのみとなる。以下すべて同様である。）

(4) Switch registor を $177406_{(8)}$ にSet する。

(5) ENABLE／HALT switch をENABLEにし，Start switch を押す。

(6) 以上の操作によりSystemが起動し，コンソールは

RT－11SJ V02－02B

・：（S／J Monitorの時[註2)]

又は

RT－11FB V02－02

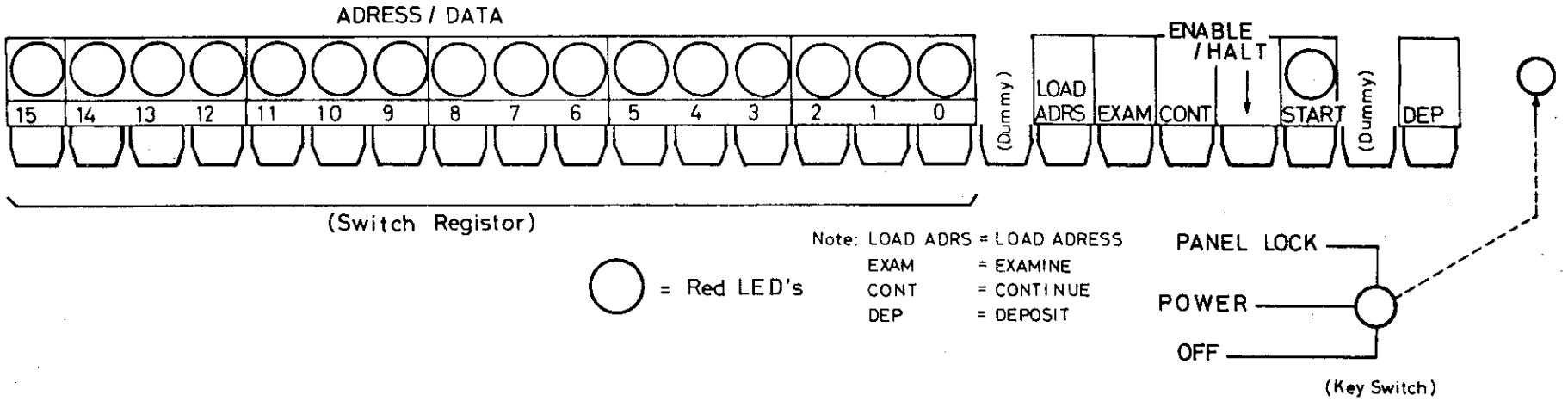
・：(F／B Monitor の時[註3)]

とメッセージ及び"・"を印字する。"・"が印字された時にのみSystem(のMonitor)は命令の受入れが可能である。

(7) Computer パネル上のKey switch をLOCKの位置に廻すと，パネル上の他のSwitch

註1） LOAD又は電源断となる毎にProtect は解除される。

註2，3） 1.3.3参照



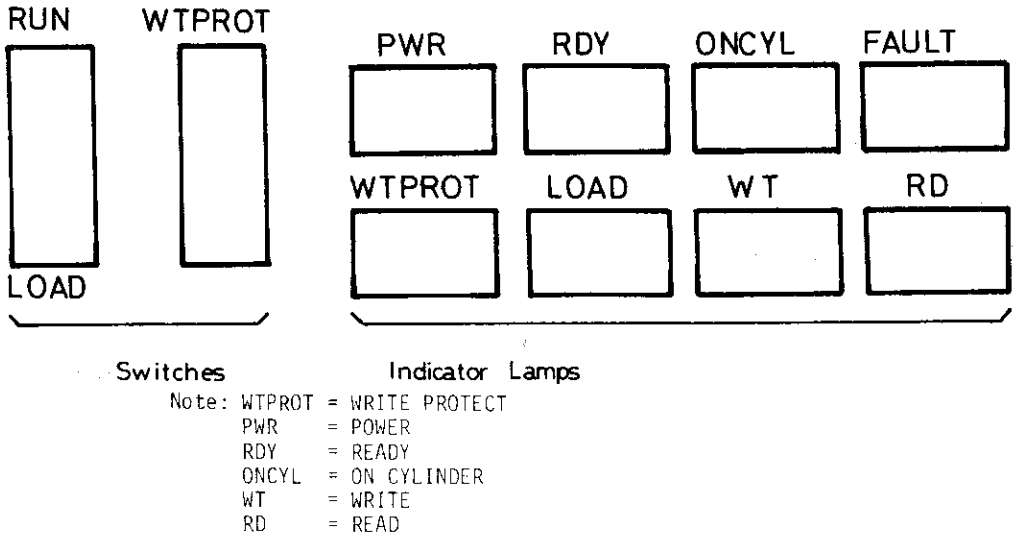


Fig.1.2 Rough sketches of operation consoles on computer (upper) and disk drive unit (right)

類のいかなる操作もSystemの状態を変化させない。長時間の連続運転はこの状態で行うのが安全である。

（Systemの停止は(8)以下の操作による。）

(8) SystemがMonitor以外のプログラムで動作している時は，コンソールのCTRL／C keyを打って[註4)]，SystemをMonitor制御の状態に戻す。

(9) ENABLE／HALT switchをHALTにし，Disk driveのRUN／LOAD switchをLOADにする。

(10) Disk drive前面の表示灯が，"PWR"及び"LOAD"のみ点灯の状態（Diskの廻転が完全に停止し，Cartridgeの着脱が可能となった状態）になったら，Computerパネル上のKey switchをOFFの位置に廻す。

(11) Monitor又は他のプログラム類が動作中に，Disk driveのみを停止させて，Cartridgeを交換することができる。但しDiskの内容の読み書きが行われていない時に限る。

1.3.2 System Fileについて

ソフトウエアはすべて，DEC作製のRT-11オペレーティング・システムを基礎に構成される。これをSystemの動作に併って随時所要部分をメモリに呼び出せる型態のFile（System file）として保持する装置をSystem deviceと呼ぶ。本システムではSystem fileをDisk cartridgeに書込んで使用する[註5)]。System fileを書込んだDiskを以下System diskと呼ぶ。Disk driveは現在1台のみであるから，すべてのDisk上のFileはSystem fileと同じDiskに存在する。

1.3.3 F／B及びS／J Monitor

RT／11 systemのMonitorには，F／J用及びS／J用の2種がある。両者はコンソールのKeyboardからの入力のみで容易に切替えることができる[註6)]。

(1) F／B（＝Foreground／Background）monitorは，リアルタイム処理を行うForegroundと，優先順位の低いBackground（プログラムの開発等）の2つのプログラム（Job）を同時に並行処理する。2つのJobは各々が独立して動作し，またDisk fileやメモリのコミュニケーション・エリアを使用して，相互にDataを交換することができる。但しメモリの容量が不足するため，現在の所使用していない。この場合，CLASSはBackground jobとしてのみ動作する。

(2) S／J（＝Single job）monitorは，(1)のようなF／B動作が不用な場合に使用し，リアル・タイム・ジョブ又はバッチ・ジョブのいずれか一方を実行する。

註4） 1.3.4の(1)参照

註5） 外にフロッピーディスク又はDECtape（DEC特有の磁気テープ装置）を使用するSystemも供給される。

註6） Reference [2)] §2－1，p．2－2.1参照。

1.3.4 Keyboard Communications

RT−11 Monitor とUser とのCommunicationは，コンソールのKeyboardを使用して行われる。その方法は次の2種に大別される。

a．Special function keyを使用する。

b．Keyboard commandを使用する。

(1) Special function keyによる操作 − Rubout keyを打ち，又は"CTRL／{‥}なるKeyの組合せを打つことにより行われる。

CTRL／C；"↑C"なるEchoを印字して，現在実行中のJobを停止し，Monitor が入力命令待ちをしている状態に戻る。プログラムがKeyboard からの入力命令待ちの状態では1回，その他の場合は2回打つ。

CTRL／O；"↑O"なるEchoを印字して，実行中のプログラムからコンソールへの出力を停止する。次の3種類のいずれかの操作で元の状態となる。

a 再びCTRL／Oを打つ。

b プログラムを停止し，Monitorに戻る。

c ．RCTRLO命令（プログラムリクエスト）をプログラム中で実行する[註7)](コンソールへの出力の一部を省略する時に利用する。）

CTRL／Q；CTRL／Sにより停止した出力の続きをコンソールに印字する。（Echoは出力されない。

CTRL／S；CTRL／Qが打たれるまでコンソールへの出力を一時停止する。（Echoは出力されない。）

（CTRL／S，CTRL／Qと打つことにより，コンソールへの出力を中途で一時休止させることが可能となる。）

CTRL／U；現在入力中の1行を消去し，コンソールに"↑U＜CR＞＜LF＞"とEcho を印字する。

RUBOUT；現在入力中の行の最後の1字を消去し，Backslash "＼"と消去された文字をEchoする。引続きそのKeyを打てば順次その直前の文字が消去され，消去された文字をEchoする。このKey以外のKey を打てば，消去終了を示す"＼"が印字されて通常の状態に復帰する。消去はその行の先頭まで行うことができる。（現用のDECwriter では"DELETE" keyを使用する。）

(2) Keyboard commandによる操作 − Monitorが印字した"．"の直後へ，以下の"型式"で示したFormat で命令をType し＜CR＞を打つことにより行われる。

（2−1）DATE命令；System内に日附を記入し，読み出しを行う。

（型式）DAT{ E }␣{ dd−mmm−yy }；{ }内は省略可能。

ここでdd−mmm−yyなるArgument は，記入する日−月−年であり，dd は1から31までの10進数，mmmは月名（英語）の先頭3文字，yy は73から99までの10進数である。

註7） Reference [2)] Chapter 9参照。

Argument なしの時は現在の日附を印字する。

（例１）　．DAT　31－MAR－79＜CR＞
　　．DAT＜CR＞
　　31－MAR－79
　　．

記入する日附の型式の誤り，及び日附が未記入の時はError となる。

（２－２）TIME命令　；System内の時刻を読み取り，又は新規時刻の書き込みを行う。

（型式）TIM{ E }␣{ hh：mm：ss }

ここでArgument hh：mm：ss は時．分．秒（24時間制）を表す。いずれかが省略された時，その分は０とおかれる。すべてを省略した時は現在の時刻が印字される。（{ }）内が省略可能な部分）

（例２）　．TIM　17：45：00＜CR＞
　　．TIM＜CR＞
　　17：45：03
　　．

時刻を特に書き込まなかった時，System内の時刻は，System起動時からの経過時間を示す。

F／B monitor 動作時に時刻が24：00に達すれば，時刻は０：00にReset され,同時に日附は１日進む。

＊＊〔但し各日の最後の日附の場合は，その内容が破壊される。〕＊＊

S／J monitor 動作時には，上記のような時刻のReset，日附の更新は行われず，時刻はそのまま積算されて行く。

（２－３）LOAD命令；Device handler（特定I／Oのための入力用プログラム）をメモリにLoadする。

（型式）LOA{ D } dev_1 { , dev_2 }{ , ……dev_n }；{　}部は省略可能。

ここで dev_1，dev_2，……dev_n は，Table １－１のAbbreviation に示した，２又は３文字で表される各Device の名前（Physical device name）である。

本命令は，CLASSに於て高速紙テープリーダ（PR）及び高速紙テープパンチ（PP）を使用する時にあらかじめ行っておかねばならない。

（例３）　．LOAD PR, PP＜CR＞
　　．

＊＊〔コンソール及びSystem device 以外のI／Oについて行う必要がある。但し現在の所Cassette drive unit（CT０，CT１）については特に行う必要はない〕＊＊

（２－４）UNLOAD命令；メモリ内のDevice handler を消去する。

Table 1.1 RT-11 Permanent device names (Setup devices are marked as "o".)

Permanent Name (RT-11 Abbreviation)	I/O
CR:	Card reader(CR11/CM11).
CTn:°	TA11 cassette (n is the unit number, 0 or 1).
DK:	The default logical storage device for all files. DK is initially the same as SY: (see below), but the assignment (as a logical device name) can be changed with the ASSIGN command.
DKn:	The specified unit of the same device as DK.
DPn:	RP02 disk (n is an integer in the range 0-7).
DSn:	RJ03/4 fixed-head disks (n is in the range 0-7).
DTn:	DECtape n, where n is a unit number (an integer in the range 0 to 7, inclusive).
DXn:	RX01 floppy disk (n is 0 or 1).
LP:	Line printer.
MMn:	TUJ16 magtape (n is in the range 0-7).
MTn:	TM11 (industry compatible) magtape (n is an integer between 0 and 7, inclusive).
PP:°	High-speed paper tape punch.
PR:°	High-speed paper tape reader.
RF:	RF11 fixed-head disk drive.
RKn:°	RK disk cartridge drive n (n is in the range 0 to 7 inclusive).
SY:	System device; the device and unit from which the system in bootstrapped. (RT-11 allows bootstrapping from any RK unit.) The assignment as a logical device name can be changed with the ASSIGN command.
SYn:	The specified unit of the same device type as that from which the system was bootstrapped.
TT:°	Terminal keyboard and printer.
CIn:	Canberra 2020 cassette (n = 0 ∿ 2) (optional from Canberra).

（型式）UNL{ OAD }dev_1 { , dev_2 }{, ⋯ dev_n } ；{ } 部は省略可能。dev_1 ⋯ も前記に同じ。

（2－5）R命令；System device からプログラムをメモリに読込み，実行させる。

（型式）　R　filnam{ . ext }

ここで" filnam "は，プログラム等のFileをDevice内で相互に区別するために附されたFilenameであり，" ・ext "はFileの性質を区分するための0～3文字から成るExtentionである。{ } 部を省略すると，" . SAV "と見なされる。

（2－6）REENTER命令；CTRL／C により停止させたプログラムを，Reentry adress（再スタート番地）からスタートさせる。

（型式）　RE { ENTER } ；{ } 内は省略可能。

1.4　CLASS語の動作モード

CLASSは次の2つのいずれかのMode で動作する。

a. Interactive mode ；命令をコンソール，高速紙テープリーダ，磁気テープ等のシリアルファイルから入力すると，直ちにその命令を判断し実行する。

b. Stored program mode ；処理の手順に従った一連の命令にプログラム名を附し，一旦メモリに書込んで後実行する。本システムでは，プログラムは一旦Disk上にFileの型で保存し，使用の都度メモリに書込み実行するのを原則とする。プログラムは，Cassette ,紙テープ，磁気テープ等に記録することも可能である。

1.5　CLASSの起動・停止の操作

(1)　CLASSの起動はR命令により行う。（1.3.4の（2－5）参照）

（例1）　. R␣CLASS { . SAV } <CR>；{ } 部は省略可能

これによって次のメッセージが印字される。

CLASS　V04 . 23RT

15 － FEB－77

次でコンソールは2行おいて次の行の左端に" ＊ "（Asterisk）を印字し，その直後で入力待ちの状態となる。

" ＊ "の直後へ引続きCommandをTypeし，<CR>を打てば，" ＊ "と<CR>の間の命令が直ちに実行され，終了すればCLASSは次の行の左端に" ＊ "を印字して再び待期状態となる。

(2)　プログラムの実行を中止させる時は，CTRL／Cを2回打つ。Systemは" ↑C "をEchoしてMonitorに戻り，次の行の左端に" . "を印字して入力待ちの状態となる。CLASS

が待期状態の時は，CTRL／Cを1回打てば，同様にしてMonitor に戻る。
＊＊〔コンソールの印字部にSystemから出力中の時にCTRL／Cを打つと，直後の出力内容の一部又は全部が無意味なものとなることがある。〕＊＊

(3) 前記のようにして停止したCLASSの再起動は次のいずれかにより行う。

a. REENTER命令による。（1.3.4の（2－6）参照）

（例2） ＊RE＜CR＞

＊

この場合，実行中止時に存在したプログラムやデータの内容は保存される。

b. (1)の起動操作を行う。この場合，以前のプログラムやデータの内容はすべて失われる。

1.6 CLASSに於る特殊Keyの使用

CLASSに於ては，CTRL／Cの外にも，1.3.4の(1)に述べたSpecial function keyを使用することができる。即ち，これらによって，入力命令の修正又は取消，コンソールへの出力の一時停止又は省略等を行うことが可能である。

2. Interactive Mode

CLASSの起動操作が完了した状態をもととして，本章ではCLASSの基本的なCommandについて説明する。2～4まで，すべてI／Oはコンソールの Keyboard とプリンタ（印字部）のみとする。[註8)]

2.1 TYPE命令及び算術記号

(1)数値，指定された文字並び，数式の計算結果を印字する。

（例1）
```
*TYPE␣2<CR>
2.0000000
*TYPE␣2+2<CR>
4.0000000
*TYPE␣2*3<CR>
6.0000000
*TYPE␣2+2↑3<CR>
10.000000
*TYPE␣(2.315+9.26↑3.2)/(8+(3.25)↑0.5)
<CR>
126.65334
*
```

最後の部分は

$$\frac{2.315+9.26^{3.2}}{8+3.25^{1/2}}$$

を計算した例である。本例に現れたものを含めて，CLASSで使用可能な算術記号はTable 2.1の通りである。

註8） DECwriter は，15″ 幅連続用紙を使用して131字／行の印字が可能である。

Table 2.1 Mathematical operators available in CLASS

Symbol	Meaning
+	Add
-	Subtract
*	Multiply
↑	Rise to the power
/	Divide
( )	Operate in the enclosed expression
SIN(X)	Sine of X, where X is in radians
COS(X)	Cosine of X, where X is in radians
ATAN(X)	Arctangent(X), where the result is expressed in radians
SQRT(X)	Square root of X
EXP(X)	Raises e to the Xth power
LOG(X)	Logarithm of X to the base e
ABS(X)	Absolute value of X
IP(X)	Integer part of X
FP(X)	Fractional part of X

(2) 2つ以上の計算を同時に行う時は，各々を〃，〃で区切る。

（例2） *TYPE␣2*2，2*3，2*4<CR>

4.0000000　　6.0000000　　8.0000000

*

(3) 算術記号の優先順位は一般の算術式と同じである。

(4) 数字の打出しFormatはすべて，CLASS起動直後の設定$G（14，8）と同じにして示した。これは整数，小数，指数表示の3タイプを随時選ぶことができる。詳細は5.4参照。なおCLASSで扱う数値には，FORTRANやALGOLの如く，整数型・実数型といった区別は行われない。

(5) 文字を打出す時は，打出すべき文字の並びを〃　〃でかこむ。〃　〃内には<CR>及び特殊キーによるもの，〃符号それ自身を除き，すべての文字・記号・スペースを任意に並べることができる。

（例3） *TYPE〃THE　ANSER　IS〃<CR>

THE　ANSER　IS

*

また，〃！〃を用いてその個数と同じ回数だけ改行の指示を行うことができる。

```
(例4)  *TYPE␣2+2, !<CR>
       4.0000000

       *TYPE"THE ANSER IS", 2+2, !, 3+3, !<CR>
       THE ANSER IS4.0000000
       6.0000000
       *
```

ここで注意すべき点は次の通りである。

a. TYPE命令と！又は〃符号との間のスペースはなくしてもよい。

b. 〃と！及び数字・算術式相互間は必ず〃,〃で区切る。

c. ！符号相異間は〃,〃で区切らず，任意個数だけ並べることができる。

＊＊[〃TYPE〃と〃,〃，又は〃,〃と〃<CR>〃，〃TYPE〃と〃<CR>〃又は〃,〃 同志等の相互間に出力すべき数字・変数（後述）・文字並び等が存在しない時，その部分は0とみなされて出力される。

```
(例5)  *TYPE<CR>
       0.0000000
       *TYPE,<CR>
       0.0000000      0.0000000
       *TYPE,,<CR>
       0.0000000      0.0000000      0.0000000
       *
```

〕＊＊

2.2 SET命令及び変数

(1) 変数をメモリ内に設定し，（Named variable と呼ばれる）数値を変数に入力する。

(2) 変数名の構成は次による

a. 第1字は必ずアルファベット（以下英字）であり，以後は英字及び数字（以下英数字）を任意に使用する。

b. 1文字以上，最大6文字でなくてはならない。

(3) 入力する数値のFormat の指定は不用である。

```
(例1)  *SET␣A=6<CR>
       *SET␣B=5.293<CR>
       *SET␣ALPHA=3.14159<CR>
       *SET␣BRAVO=9<CR>
       *SET␣VAR=4, CONST=6, POWER=3.2<CR>
       *
```

最後の例のように，1行で2箇以上の変数を設定する時は，各々を″，″で区切らなければならない。また式の右辺には設定ずみの変数も使用可能である。

(4) ″SET″の文字は省略することができる。また右辺を計算式とすればその結果が左辺に入力される。

(例2)
```
*A=3，B=2，C=1，X=4<CR>
*Y=A*X↑2+B*X+C<CR>
*TYPE″Y=″，Y，!<CR>
Y=57．000000

*
```

2.3 DO命令

DO命令のFormatは，命令語″DO″，繰返しの回数（DO count），実行すべき文の3部分より成る。

(例1)
```
*DO(3)<TYPE␣2+2，!><CR>
4．0000000
4．0000000
4．0000000
*
```

DO命令の使用法は次の通りである。

(1) 繰返しの回数は必ず″（ ）″に入っていなければならない。

(2) 回数は定数又は前以て設定された変数名で指定する。

(3) 回数は正の値でなくてはならない。

(4) 回数が整数でない時，その小数部は無視される。たとえばN＝3．14とした時，DO(N)＜．．．＞とした場合，繰返しは3回となる。

(5) 実行すべき文はすべて″＜＞″（Angle bracket）に入っていなければならない。
註9）以下このDOから終りの″＞″までをDO－loopと呼ぶ。Interactive modeの場合，このDO－loopは1行内で終らなければならない。

(6) 1組の″＜＞″内に2つ以上の命令が入る時は，各々を″；″で区切らなければならない。

DO命令の代表的な使用法として，変数の値を1回毎に変化させながら計算を行う方法がある。例として，Xの値を1から25まで1ずつ増加させ，その度毎にX^1，X^3，X^5を計算させると次のようになる。

註9）″＜″，″＞″なる記号（それぞれ″，″，″．″Keyの上段）は，BASIC語等では不等号として使用されるものである。

（例２）　＊X＝1<CR>
　　＊DO（25）<TYPE″A＝″，X↑1，！，″C＝″，X↑3，！，
″E＝″，X↑5，！，X＝X＋1><CR>
　　A＝1.0000000
　　C＝1.0000000
　　E＝1.0000000
　　A＝2.0000000
　　C＝8.0000000
　　E＝32.0000000
　　A＝3.0000000
　　C＝27.0000000
　　E＝243.00000
　　{中間省略}
　　A＝25.0000000
　　C＝25625.000
　　E＝97656625.0
　　＊

（DO命令の前に，あらかじめXの初期値を設定している点に注意）

DO－count が１の時はこれを省略できる。即ち

　　DO(1)<・・・>

は

　　DO<・・・>

と書くことができ，その機能は単に″<>″の内部をそのまま書いたものと同じである。

2.4　ERASE命令

一旦変数を設定するとVariable list にStoreされる。これを消去するには本命令を使用する。

（型式１）＊ERASE␣{変数名－２箇以上の時は各々を″，″で区切る}<CR>
；{　}部に指定した変数を消去する。

（型式２）＊ERASE␣ALL<CR>　；すべての変数を消去する。

（設定された変数名のList はINDEX命令を使用して出力させることができる。9.4参照）

2.5　２箇以上の命令の連続

１行に２箇以上の命令が連続する時は（DO－loopの内外を問わず）各々を″；″で区切る。

(例)　＊A＝7；TYPE␣A<CR>

7.0000000

＊

3. Stored Program Mode

このModeを使用することにより，Userは1行のDO命令よりも複雑なプログラムを書くことが可能となる。またこれに附したプログラム名を使用して，メモリ内又は他のファイル内（後述）のプログラムを，他のCLASSの命令と同様に呼出し，実行することができる。

3.1 DEFINE命令

(1) この命令により，CLASSはStored Program mode に切替えられ，プログラムをメモリにStoreする。

（型式1）＊DEFINE␣{プログラム名}<{プログラムの内容}><CR>

(2) プログラム名の構成は，変数名のそれと全く同じである。（2.2参照）

(3) プログラムの本体（最初の〃<〃から最後の〃>〃まで）は任意の行数を取ることができる。行送りは<CR>をTypeする毎に行われる。最後に〃・・・・・・><CR>〃とTypeすることによって，プログラムはメモリにStoreされ，CLASSは次の命令待ちの状態に戻る。

(4) プログラム内には，既述のもの及び後述するCLASS語のすべての命令（プログラム自体を修正するためのEDIT命令 ── 後述 ── を除く）及び他のCLASS語プログラムを任意に使用可能である。また本Modeでは，DO－loopを任意行数に分割することができ，1つのDO－loop内に他のDO－loopを任意に含むことができる。

(5) 以上によりメモリにStoreされたプログラムを実行させるには次の操作による。

（型式2）　＊DO（n）{プログラム名}<CR>

ここでnは実行する回数（≧1）である。n＝1の時は

＊{プログラム名}<CR>

とすることができる。

実例として，2.3の計算を本Modeで行うと次の通りとなる。（出力のFormatは変更）

```
（例1）　＊DEFINE　POWERS<<CR>
        X＝1<CR>
        DO（25）<TYPE〃A＝〃，X↑1，〃C＝〃，X↑3，〃E＝〃，
X↑5，!<CR>
        X＝X＋1>><CR>
        ＊DO(1) POWERS<CR>
        A＝　1.0000000　　C＝　1.0000000　　E＝　1.
0000000
```

```
        A=  2.0000000        C=  8.0000000        E=  32.
000000
        A=  3.0000000        C=  27.000000        E=  24
3.00000
        A=  4.0000000        C=  64.000000        E=  10
24.0000
        {中間省略}
        A=  25.000000        C=  15625.000        E=  97
65625.0
        *
```

本例から，繰返しの回数はプログラム実行時に指定して，プログラム内のDO－loopを外せば次のようになる。

```
(例2)   *DEFINE  POWERS<<CR>
        A=X↑1, C=X↑3, E=X↑5<CR>
        TYPE "A=", A, "C=", C, "E=", E, !<CR>
        X=X+1><CR>
        *X=1<CR>
        *DO (25) POWERS<CR>
        {結果は前記と全く同じとなる（省略）}
        *
```

3.2 DELETE命令

DEFINE命令でメモリにStoreされたプログラムを消去する。型式は次の通りである。

（型式1）指定したプログラムのみ消去する時。

```
*DELETE␣{プログラム名（2箇所以上の時は各々を ","で区切る }
<CR>
```

（型式2）すべてのプログラムを消去する時。

```
*DELETE␣ALL<CR>
```

（メモリ内のプログラム名のListは，変数名のそれと共にINDEX命令により出力させることができる。9.4参照）

3.3 Argument（ARG）及びReturn Value

CLASSでは，プログラム実行の度毎に，Argument（引数）を使用して，変数を使用せずに必要な数値をプログラム内部に送り込むことができる。

(1) Argument の指定方法

＊ プログラム名 (a_1 , a_2 , ···· , a_n) <CR>

なる方法による。ここで( )内のa_1 , a_2 , ···· a_n をArgument (引数) と呼ぶ。プログラム内ではそれぞれの値をARG (1) , ARG (2) , ···· ARG (n) なる名前で使用する。

実例として，任意の3数値の平均を計算するプログラムは次のようになる。

(例1)
```
*DEFINE AVERAG<<CR>
TYPE (ARG (1)+ARG (2)+ARG (3)) /3><CR>
*AVERAG (2, 3, 4) <CR>
3.0000000
*
```

(2) 引数には，定数及び変数を使用することができる。

(3) すべての場合，プログラム最後の行に変数又は計算式をただ1箇のみ書けば，その変数の値又はその計算式の結果を，そのプログラム名の持つ数値として扱うことができる。この数値をReturn value と呼ぶ。これを応用して，前記のプログラムを次のように変更することができる。

(例2)
```
*DEFINE AVERAG<<CR>
(ARG (1) +ARG (2)+ARG (3) /3><CR>
*TYPE AVERAG (2, 3, 4) <CR>
3.0000000
*
```

またCLASSの基本関数COS，SINを利用して，tan θ = sin θ / cos θ なる公式により，TAN関数を作れば次のようになる。

(例3)
```
*DEFINE TAN<<CR>
SIN (ARG (1) ) /COS (ARG (1) ) ><CR>
*TYPE TAN (0.785)
1.0000000
*
```

(4) プログラム内で，ARG (i) のiの最大ものの値が l ならば，引数は必ず l 個指定しなければならない。また引数の個数nが l<nならば，(l + 1)からn番目までの引数は計算に関係しない。

(5) プログラム内では，引数の個数の値をARG (0) なる名前で使うことができる。これによって先述のAVERAGプログラムを，任意個数の数値の平均を計算できるようにした例を示す。

(例4)
```
*DEFINE AVERAG<S=0, I=1, N=ARG (0) <CR>
DO (N) <S=S+ARG (I) , I=I+1><CR>
S/N>
```

```
*TYPE  AVERAG(2,4,6)<CR>
4.0000000
*TYPE  AVERAG(2,4,6,12)<CR>
6.0000000
*
```

3.4 IF命令

条件判定を行い，プログラムの流れを制御する。

(1) IF命令は次の3部分より成る。

a．命令語 "IF"

b．判定を行う条件式（Relationship）

c．条件式の判定結果が "TRUE"（成立）であった時に実行される命令

(2) 使用できる条件式は次の通りである。

Symbol	Relationship	beeing tested
EQ（A，B）	IS	A＝B ?
NE（A，B）	IS	A≠B ?
LT（A，B）	IS	A＜B ?
LE（A，B）	IS	A≦B ?
GT（A，B）	IS	A＞B ?
GE（A，B）	IS	A≧B ?

A，Bは定数（どちらか一方のみ ── 両方共定数ではIFを使う意味がなくなる），変数又は算術式である。

(3) IFの使用法は次のようになる。

（型式）　IF（{条件式}）{条件が "TRUE" と判定された時の実行命令}<CR>

実行すべき命令が2箇以上の時は，各々を "；" で区切る。これらはすべてIFと同一行内でなくてはならない。もしこれら実行命令の並びを2行以上に分割する時はDO－loop を利用する。

（例1）　IF（{条件式}）DO<・・・・・・><CR>

定数0と比較を行う時は，これを省略して書くことができる。

（例2）　IF（EQ（A，0））→ IF（EQ（A））etc。

(4) 絶対値が1.0未満の定数と "EQ" の判定を行うと，内部の桁数の関係で "EQ" が成立しない場合がある。定数を変数に入力の上判定を行うのが確実である。

（例3）　IF（EQ（A，0.1））→ B＝0.1；IF（EQ（A，B））

3.5 GOTO命令とターゲット

(1) GOTO命令は，通常IF命令と組合せて，プログラムの流れを制御するために使用する。単独で使用することも可能である。

(型式) GOTO␣{ターゲット}<CR>

ここでターゲットは行先行の先頭を示す名前であって，その構成は変数名およびプログラム名と同じである。(2.2参照)

(2) ターゲットは，GOTO命令に使用されたものと同一の名前のものがそのプログラム内のいずれかの行の先頭に必ず一箇所だけあって，その直後に"："(Colon)がついていなくてはならない。

(3) GOTO命令はそれ自身の存在する行の先頭を指すことはできない。またDO－loopの内部より外部を，外部より内部を指すことはできない。

3.6 STOP命令

IFと組合せて，ある条件の時にプログラムの途中で実行を終らせるために使用する。3.4～3.6までの使用例を示す。

(例)
```
*DEFINE ROOT<<CR>
    IF(GE(X,0))TYPE"SQRT X=",SQRT(X),!;
STOP<CR>
    TYPE"ERROR: X IS NEGATIVE",!><CR>
*X=4<CR>
*ROOT<CR>
SQRT X= 2.0000000
*X=-2<CR>
*ROOT<CR>
ERROR: X IS NEGATIVE
*
```

3.7 ブール演算

(1) IF命令で使用される条件式は，結果が成立("TRUE")の時は1，不成立の時は0なる値をReturn value(3.3参照)として持つ。一方IF命令は，( )内の値が>0ならばその直後の命令を実行する。故に条件式相互間で和又は積を計算することにより，論理和又は論理積の判定を行うことができる。

(2) 論理和(OR)；"+"記号で結合された各条件式のいずれかが成立の時"TRUE"と判定される。

（例1）

ＩＦ（ＧＴ（Ａ，２）＋ＬＴ（Ａ，－３））ＴＹＰＥ！，″ＡＢＣ″＜ＣＲ＞　；Ａ＞２又はＡ＜－３の時に″ＡＢＣ″なるメッセージを印字する。

(3) 論理積（ＡＮＤ）；″＊″記号で結合された各条件式のすべてが成立の時″ＴＲＵＥ″と判定される。

（例2）

ＩＦ（ＧＴ（Ａ，２）＊ＮＥ（Ｂ，０））ＴＹＰＥ！，″ＡＢＣ″＜ＣＲ＞；Ａ＞２でかつＢ≠０の時にメッセージを印字する。

（例3）

ＩＦ（ＧＴ（Ａ，２）＋ＧＴ（Ａ，１）＊ＧＴ（Ｂ，１））ＴＹＰＥ！，″ＡＢＣ″＜ＣＲ＞；Ａ＞２であるか，又はＡ＞１でかつＢ＞１の時にメッセージを印字する。

＊＊〔ＡＮＤ，ＯＲの演算を行う時，条件式中にReturn value を使用する算術式が存在すると，System 全体がHalt（停止）状態となり，再起動を必要とすることがある。この時は，算術式の結果を一旦変数に入力し，改めて条件式中で使用するように，プログラムを変更する。〕＊＊

4. Data Storage

CLASSでは，これまでに使用した定数，文字並び（Literal test string），変数の3種類のデータの外に最高3次元までの配列（Named array）を使用可能である。

4.1 DIMENS命令

(1) 配列の指定にはこのDIMENS｛＝Dimension｝命令を使用する。その型式は次の通りである。

1次元の配列： DIMENS␣ARY（n_1）

2次元の配列： DIMENS␣ARY（n_1，n_2）

3次元の配列： DIMENS␣ARY（n_1，n_2，n_3）

ここでARYは配列名であって，その構成は変数名，プログラム名等と同じである。（2.2参照）但し他の変数名と同じであってはならない。

(2) 要素数1個の配列は指定できない。即ちDIMENS A(1) はエラーとなる。

(3) n_1，n_2，n_3 は定数又は既に設定された変数である。

(4) 指定された配列の各要素は，たとえばARY（l，m，n）に於ては添字l，m，n，の値を定数，変数又は算術式で指定することによって，1箇の変数と同等に使用可能である。

(5) 2又は3次元の配列は次のように1個の添字で指定可能である。

（例1） DIMENS ARY（M，N）なる配列で，ARY（m，n）なる要素はA（i）と指定する，但し i ＝（m－1）×N＋n

（例2） DIMENS ARY（L，M，N）なる配列で，ARY（l，m，n）なる要素はA（i）と指定する，但し i＝（l－1）×M×N＋（m－1）×n＋n

(6) 配列が指定された時，その要素のすべての内容は0になっている。

＊＊〔一旦指定された配列と同じ名前の配列を再び指定した時も同じである。〕＊＊

(7) 一旦指定された配列は，変数と同様にERASE命令で消去することができる。

4.2 DATA命令

配列の初期値設定に使用する。2個以上の要素に同時にデータを設定することが可能である。以下の3通りの型式で使用する。

（型式1） DATA␣ARY<VAR1，VAR2，・・・VARM><CR>

これにより，ARY（1）＝VAR1，ARY（2）＝VAR2，・・・ARY（M）＝VARMなる値が設定される。

（型式2）　DATA　ARY（N）＜VAR1，VAR2，・・・VARM＞＜CR＞
これにより，ARY（N）からARY（N+M−1）までの要素がそれぞれVAR1，・・・VARMなる値に設定される

（型式3）　DATA　ARY＜VAR1，VAR2，N：VAR3＞＜CR＞
これにより，ARY（1）＝VAR1，ARY（2）＝VAR2，ARY（3）～ARY（2＋N）＝VAR3なる値が設定される。

4.3　関数SCAN（A，X，Y）

（4.3～　4.11で述べる関数は，主としてSpectrum analysisのために開発されたものであって，配列内のデータ処理を高速で行うことが可能である。）
SCANの機能は，配列A内の要素を最初から順にScanし，要素の値がX±Yの範囲に入る最初のものを検出し，その要素が最初から何番目かを示す数値を関数のReturn valueとする。もし条件を満す要素がなければ，この値は0となる。引数Yを省略すればY＝0とみなされる。

4.4　関数MAX（A）

配列Aの要素中，最大値のものが最初から何番目かを示す数値をReturn valueとする。

4.5　関数MIN（A）

配列Aの要素中，最小値のものを検出し，MAXと同じ処理を行う。

4.6　関数DUPL（A，B）

配列A内のデータをそのまま配列Bに複製する。もしAの要素数がBより大ならば，余りの分は複製されない。逆にBのそれがAより大ならば，Bの余分の要素内部のデータは変化しない。

4.7　関数SMOOTH（A，N）

配列A内のデータを5点法によりN回平滑化を行う。Nを省略すればN＝1と見なされる。平滑化は，配列要素の値をa_1，a_2，・・・・a_mとして，次式により行われる。

$$a_1 = a_1$$
$$a_2 = (a_1 + 2a_2 + a_3)/4$$
$$a_i = (a_{i-2} + 4a_{i-1} + 6a_i + 4a_{i-1} + a_{i+2})/16$$

$(3 \leqq i \leqq m-2)$

$a_{m-1} = (a_{m-2} + 2a_{m-1} + a_m) / 4$

$a_m = a_m$

4.8 関数DIFFER(A,N)

配列AのN次の差分（微係数，Nth derivation）を計算してAに入れる。従ってAのもとのデータは失われる。計算式は次の通り（a_1，a_2，‥‥ a_mは4.7と同じ）

N＝0　平滑化しない1次微係数を計算する。

$a_1 = a_1$

$a_2 = (a_3 - a_1) / 2$

$a_i = (a_{i+1} - a_{i-1}) / 2$

$a_{m-1} = (a_m - a_{m-2}) / 2$

$a_m = a_m$

N＝1　平滑化された1次微係数を計算する。

$a_1 = a_1$

$a_2 = a_2$

$a_i = (2(a_{i+2} - a_{i-2}) + (a_{i-1} - a_{i+1})) / 10$

$a_{m-1} = a_{m-1}$

$a_m = a_m$

N＝n（n≧2）　N＝1の時と同じ計算を1回した後，N＝0の時と同じ計算を（n－1）回繰返してn次の微係数を計算する。

4.9 関数FIND(A,B)

配列Aのデータを順次調べて，データの符号が反転する部分を検出し，次の計算を行って結果を順に配列Bへ入れる。

符号が変化する前のIndex番号をi，変化後のそれをi+1として，

$$B(N) = i + \left| \frac{A(i)}{A(i) - A(i+1)} \right|$$

この時，符号が－→＋となる時は正値，＋→－となる時は負値が取られる。

上記演算と同時に，検出された点の数がReturn value となる。

FINDは通常DIFFERと共に使用される。

4.10 関数AREA(A,B,C)

配列A(M)について次の計算を行う。

$$AREA = \sum_{n=B}^{B+C-1} A(n) - \frac{(A(B)+A(B+C-1))*C}{2}$$

B，Cが省略されれば

$$AREA = \sum_{n=1}^{M} A(n) - \frac{(A(1)+A(M))*M}{2}$$

を計算する。

4.11 関数TOTAL(A,B,C)

配列A(M)について次の計算を行う。

$$TOTAL = \sum_{n=B}^{B+C-1} A(n)$$

B，Cが省略されれば

$$TOTAL = \sum_{n=1}^{M} A(n)$$

を計算する。

4.12 MCAのデータ解析の概略

MCAで得られたガンマ線スペクトルのデータ解析を行うための，CLASS語で書かれた一連の基礎的なプログラムが，"SPECTRAN III"なる名称でCANBERRAより供給されている。[4] この詳細は，著者らの開発したプログラムと共に別の機会に述べる予定である。ここでは，4.3～4.11で述べた関数が，"SPECTRAN III"中でどのように利用されているかについて，概略の説明を行う。

(1) MCAのデータは，先ず適当な配列に読み込まれる。(11.1.6参照)以下この配列をAとし，その添字IはChannel No.に対応するものとする。Aに関数DIFFERを作用させて求めたAの1・2・3次微係数をそれぞれA'，A''，A'''とする。これらをFig. 4.1の〔A〕(シングルピークの場合)及び同〔B〕(複合ピークの場合)に概念的に示した。註10)

註10) 実際には，Aの内容を保存するため，これを更に別の配列に関数DUPLを使用してCopyし，これにDIFFER等の操作を行う。

(2) Aの内容がシングルピークの場合，A′ がIの増加と共に＋→－と変化する点のIの値を関数FINDで検出すれば，これがピークの頂点附近のChannel No.と推定することが可能であることが判る。

(3) Aの内容が複合ピークの場合，Fig. 4.1の〔B〕で判る通り，A″ が＋→－となる点のみでは小さいピークの方は検出不能であり，A‴ の値が負の領域で最小となる点のIの値が，各ピークの頂点附近になるものと推定可能であることが判る。

(4) 検出されたピークを単一ピークとして扱う時は

$$C = a * I^2 + b * I + c$$

なる式により10分の1値巾の推定値をチャンネル単位で求め，（係数a，b，cはあらかじめ標準線源のデータを所定のプログラムにより解析して求めたものを，DiskにStoreしておく）ピーク左端のChannel No. Bを算出する。ここで関数TOTAL（A，B，C）

なる計算を行えば，Fig. 4.2に示した領域（P），（Q）の合計面積が得られる。

次に，ピーク両端の値をA（B），A（B＋C－1）とし，この両端を結んだ直線より下の領域（Q）の面積は

$$\{A(B) + A(B+C-1)\} * C/2$$

となる。これをBackgroundの値として，前出の（Gross area）の値より引けば領域（P）の面積（Net area）が得られ，その値は

$$(\text{Net area}) = \text{AREA}(A, B, C)$$

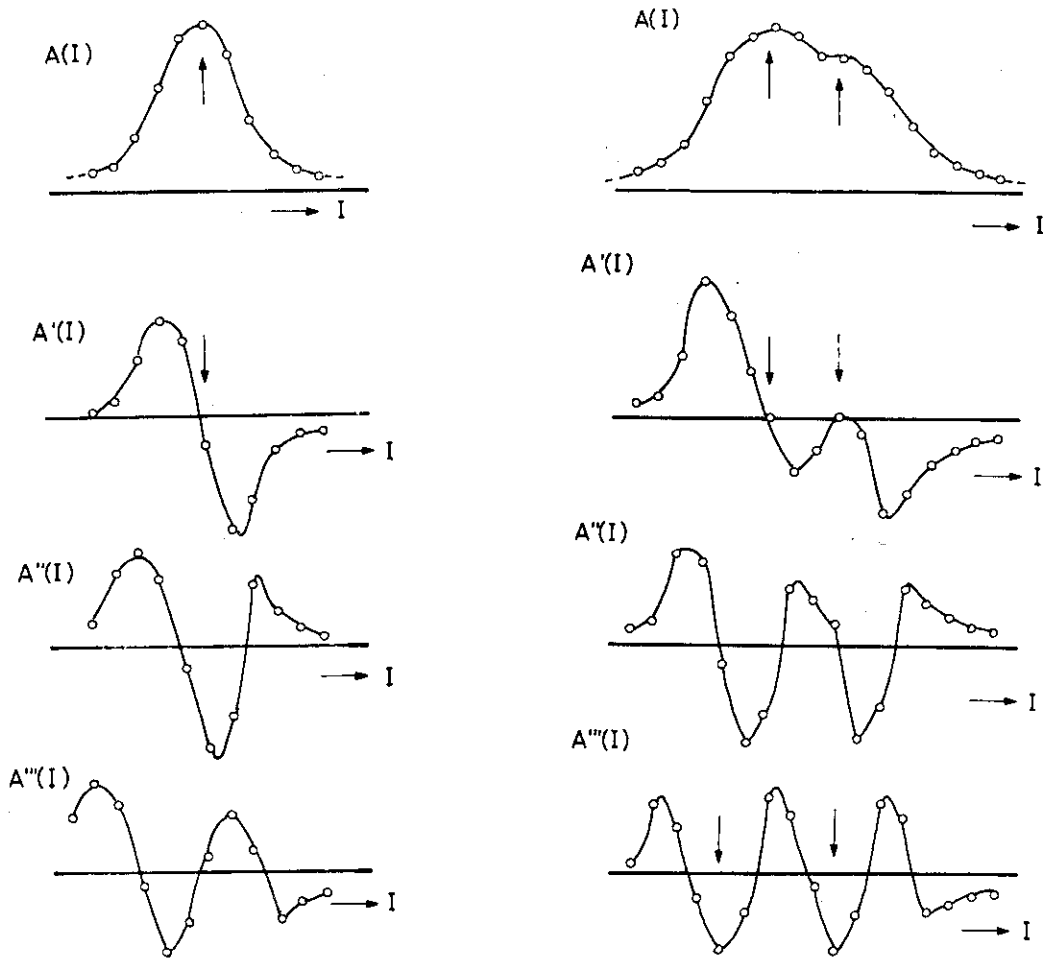
と関数AREAを使って得られるものと同じであることはFig. 4.2にも示した通りである。

実際のプログラムでピーク面積の計算を行う時は，着目しているピークの両端の境界が確実にBackgroundとなるように，10分の1値巾の両端に各々2チャンネルを加える。また最終的に次の式を満足する場合をピークと判定する。

$$(\text{SIG}) = \frac{(\text{Net area})/C}{\{(\text{Gross area})/C\}^{1/2}}$$

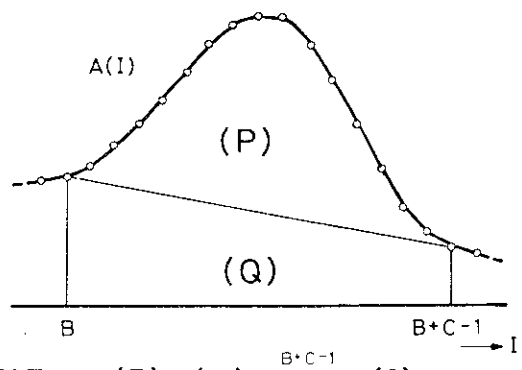
ここで（SIG）はUserが指定する定数（＞0）であって，小なるほど微小なピークまでが検出される。現在迄の使用実績では，約1.5が最適である。

(5) 複合ピークの場合は，Background（Channel No.の一次式で表わされる）を差引いたデータについて，Gauss関数にFittingが可能であったものについて各ピーク毎の面積が計算される。但しプログラムの内この部分はAssemblerで組まれていて，その内容は不明である。



(A) Single peak (B) Complex peak

Fig.4.1 Relations between the tops of photopeaks and their 1st-3rd differential coefficients



$$\mathrm{TOTAL} = (P) + (Q) = \sum_{B}^{B+C-1} A(I)$$

$$\mathrm{AREA} = (P) = \mathrm{TOTAL} - (Q)$$

$$= \mathrm{TOTAL} - (\,A(B) + A(B+C-1)\,) \times C/2$$

Fig.4.2 Peak area calculation by the functions "TOTAL", "AREA"

5. I/O の 3 分類

5.1 一　般

CLASS語によるプログラムに於ては，今までに述べたコンソールのKeyboard及びPrinterの外に，RT－11がSupportしているすべてのI/O（Table 1.1）が使用可能である。（但し，Teletypeの低速紙テープリーダ/パンチはSupportしない）Table 1.1左側の欄は各I/Oのコード名であって，プログラム中でI/Oの指定をする時に使用される。

System device（RK0又はSY0），Cassette及びコンソール以外のI/OをCLASSで使用する時は，CLASS起動前に，1.3.4の（2－3）に主べた操作を行っておく必要がある。

5.2 I/Oの3分類

I/Oは次の3グループに分類され，入出力に使用する命令，CLASSで扱えるデータの種類も異る。

（グループ1.）　Serial access, no file directory —

機器の種類；コンソールターミナル，高速紙テープリーダ/パンチ，ラインプリンタ，カードリーダ。

扱えるデータ；ASCII[註11)]で表わされる数字，アルファベット，記号の入出力。但しこれらはファイル名で区別することはできない。

使用できる命令；5.5に述べる命令。

（グループ2.）　Serial access with file directory —

機器の種類；DECcassette（CTn），Magtape（MTn），CANBERRA 2020 Cassette（CIn）。

扱えるデータ；

a. ASCIIで表わされる数字，アルファベット，記号の入出力。入出力は指定された名前のFileに対して行われる。

b. SAVE，RUN命令等によるCLASS語プログラムの入出力。

使用できる命令；

前項aの場合 — 5.5で述べる命令。

前項bの場合 — 6.1で述べる命令。

（グループ3.）　Random access I/O —

機器の種類；Disk，Floppy disk，DEC tape

註11) ASCII＝American Standard Code on Information Interchange

扱えるデータ；

a. グループ2のaと同じ。

b. グループ2のbと同じ。

c. CLASS Data File と呼ばれるRandom access fileに対するデータの入出力。

使用できる命令；

前項aの場合 — 5.5で述べる命令。

前項bの場合 — 6.1で述べる命令。

前項cの場合 — 6.2で述べる命令。

5.3 CLASSの扱うFileの標準Extension

RT－11では前記グループ2及び3のI／O（Mass storage deviceと呼ぶ）のファイルに，ファイル名の外に3文字のExtension をつけることができる。[註12] 5.5及び6に述べる命令で扱うファイルに使用されるExtension は次の通りである。

5.5の命令で扱うファイル： .CSF（=CLASS serial file）

6.1の命令で扱うファイル： .CPF（=CLASS program file）

6.2の命令で扱うファイル： .CDF（=CLASS data file）

（File名とExtenison の間には必ず"．"が入る。）

但し命令文により別のものを指定できる。

5.4 数値出力のFormat

Serial access I／Oに於ける数値出力のFormatは次の4種類である。

5.4.1 整数型$I

（型式）$I（w）

(1) 数値を整数型で出力する。wは総文字数である。

(2) このFormatで出力できる数値Nは，－32769＜N＜32768 の範囲である。これを超えると"ERROR 146"のメッセージが出力される。[註13]

5.4.2 固定小数点型$F

（型式）$F（w，d，p）

ここで

w；総文字数

d；小数点以下の桁数

註12) Reference [2] §2.2.4，p．2－5～7参照。

註13) エラー・メッセージについては13参照。

p；数値を10^p倍して出力する。（出力される数値自体は変化しない。）

w≧d＋2であること。

(1) 数値を固定小数点型で出力する。

(2) d，pは省略できる。その場合にはd＝2，p＝0とみなされる。

5.4.3 指数型$E

（型式） $E（w，d，p）

ここで

w：総文字数。

d：表示桁数（先頭の0を含む），但しp≠0ならば小数点以下の桁数。

p：小数点の左側に表示される桁数。

w≧d＋p＋6であること。

(1) 数値を指数型で出力する。

(2) d，pは省略できる。その場合にはd＝2，p＝0とみなされる。

5.4.4 浮動小数点型$G

（型式） $G（w，d，p）

ここでw，d，pは$Eと同じに指定する。

(1) 数値Nが$0.1 \leq N < 10^d$の時は有効数字d桁の浮動小数点表示を行う。この場合pは無視され，またdが省略されれば2とみなす。

(2) 数値が(1)の範囲外であれば$E（w，d，p）のFormatで表示する。d，pは指定されなければそれぞれd＝2，p＝0とみなす。

5.4.5 Format 指定の使用法

(1) CLASS起動時のFormatは$G（14，8）に設定される。

(2) Format 指定の使用法は次のいずれかによる。

（型式1） ｛Format指定｝ ｛；又は<CR>｝

（型式2） ｛Format指定｝＋｛変数名又は配列の要素｝｛；又は<CR>｝

＊＊［型式2の方法はManual 等に記載がなく，既製のプログラムの使用例をもとに著者が試用したものである。］＊＊

(3) $I，$Fの出力は右詰め，$E，$Gの出力は左詰めとなり，余分の字数分はSpaceとして出力される。数値先頭の＋は常にSpaceで出力される。（$Eの指数部を除く。）

(4) 字数が指定のwを超える時は“＊”をw個だけ出力する。

5.4.6 数値の精度及び最大限度

(1) RT／CLASSの数値精度は次の通りである。

4語長　　17桁
3語長（標準）　12桁（本システムの場合）[註14)]
2語長　　7桁

(2) 数値Nは $10^{-39}<N<10^{38}$ の範囲である。計算中にこれを超える数値が発生すると，Error 60のメッセージが出力される。

5.5 I/O入出力の命令

本節の命令は，CLASSのSupportするすべてのI/Oに対し使用可能である。

5.5.1 I/OのOpen及びClose の操作

(1) CLASS起動時には，コンソールのKeyboard入力，同印字部の出力のみがOpenの状態となっている。これ以外のI/Oの入出力を開始するには，それぞれのI/Oを次によりOpenしなければならない。

（型式1）　OPEN{ x }␣{ dev }:{ filnam }{ . ext }

ここで

{ x }；R（I/OからCPUへ入力）又はW（CPUからI/Oへ出力）
{ dev }；I/Oの記号（Table 1.1のAbbreviation）
{ filnam }；Filename（変数名等と同じく構成する — 2.2参照）
{ . ext }；FilenameのExtensoion，英数字で最大3字文。

但し，I/OがTT，LP，CR，PP，PRの場合，FilenameとExtensionは指定しなくともよい。

(2) OpenされたI/Oの入出力が完了したら，次のI/OをOpenする前に必ずClose操作を行う。特にグループ2及び3のI/Oは，プログラム終了までに必ずClose操作を行う。命式の型式は次の通りである。

（型式2）　CLOSE

5.5.2 特定I/OのOpen-Close の操作

Table 5.1に示すI/Oは，“OPENx”,“CLOSE”より短い専用の命令でOpen又はClose が可能である。但し標準CLASSでは，この内の$OT，$IKのみしか含まないので，必要ならば1.3.4の（2－3）に述べた操作をCLASS起動前に行っておく。

以上の命令については，次のような特徴がある。

(1) OPENxと異り，2種以上のI/Oを同時にOpenできる。但しClose は必ず対応の命令を使う。OPENx及びCLOSE命令と併用してはならない。この条件を満す限り1種の

註14) 2又は4語長のCLASSは註文により作製される。

Table 5.1 Standard device functions

Function (CLASS verb form)	Meaning
$OT	Open the console for output
$IK	Open the keyboard for input (default)
(Followings are optional)	
$IH	Open the high-speed reader for input
$CHI	Close the high-speed reader
$OH	Open the high-speed punch for output
$CHO	Close the high-speed punch
$OP	Open the line printer for output
$CP	Close the line printer for output

I／OをOPENx，$xx のどちらででもOpenできる。

(2) ここに挙げたI／Oを$CxでCloseすると自動的にコンソールのKeyboardがOpenとなる。KeyboardをOpenした時はCLOSE又は$Cxを実行する必要がなく，また$Cxにはそのための命令は存在しない。

(3) 2種以上のSerial I／OがOpenされた時，CLASSは自動的に，最後にOpen されたものを“Active”と見なし，その他のOpenされたものをOpen状態でかつ“Inactive”と見なす。(5.5.3～4, 5.5.6と関連する。)

5.5.3 TYPE及びDirected TYPE命令

(1) TYPE 命令；数値をあらかじめ指定されたFormatで現在“Active”なI／Oに出力する。

（例1） ＊TYPE␣12＜CR＞

数値を出力する。

（例2） ＊A＝12；TYPE␣A＜CR＞

変数，配列の各要素の値等を出力する。

（例3） ＊TYPE“TEXT”，／＜CR＞

“　”内の文字並びを出力し，/記号1箇各に1組の“<CR><LF>”を出力する。

（例4）　*DIMENS　AI（10）<CR>
　　*N=1；DO（10）<AI（N）=N*10；N=N+1><CR>
　　*TYPE　AI<CR>

本例では，配列の各要素内の値を順次出力し，1要素分の出力毎に1組の“<CR><LF>”が出力される。

（例5）　Paper tape punchへ出力
　　*OPENW␣PP：<CR>
　　*TYPE␣25<CR>
　　*CLOSE<CR>

（例6）　Cassetteへ配列を出力
　　*DIMENS␣X（10）<CR>
　　{Xにデータを入れる}
　　*OPENW␣CT0：DATAX.CDF<CR>
　　*TYPE　X<CR>
　　*CLOSE<CR>

（数値がオーバーフローすると“*”が出力されるため，上記2例の如く紙テープ又はCassette等に出力した場合には，読取りの時にErrorを生じて，プログラムの実行が打切られる。出力時のFormatを$E，$G等に指定して出力を行えば安全である。）

(2)　Directed　TYPE；TYPE命令が現在“Active”なI/Oのみに出力するのに対し，（Openされていれば）“Active”，“Inactive”いずれの状態にあるI/Oに対しても出力を行う。

（型式）　TYPE［n］␣{出力すべき数値，変数等}

ここで

n=0：コンソールへ出力
　2：高速紙テープパンチへ出力
　3：ラインプリンタへ出力
　10：Serial　fileへ出力（グループ2又は3のI/O）

5.5.4　READ及びDirected　READ命令

(1)　READ命令；現在“Active”なSerial　I/Oから，変数又は配列要素の数値を読取る。

（例1）　*READ␣VAR1，VAR2...<CR>

変数の値を読む。

（例２）　＊DIMENS　ARY（10）＜CR＞
　　＊READ␣ARY（1），ARY（2）．．．＜CR＞
配列の値を読む。

(2)　Directed　READ命令
（型式）　READ［n］␣｛変数，配列等｝
nの値等はDirected　TYPEのそれに準ずる。

(3)　READ命令で入力される数値は，整数，小数，指数表示のいずれでもよい。1つのREAD命令に対し入力データが2箇以上ある時は，各々をSpace，Comma，Semicolon，＜CR＞のいずれかで区切る。また一連のデータの終りには必ず＜CR＞を入れる。

(4)　もし最後の＜CR＞までにREADが指定したものより多数のデータが入力されると，先頭から指定された個数のデータのみが読込まれ，残りは無視される。
（例３）　Cassetteのデータファイルより配列へ数値を読込む。
　　＊　DIMENS␣X（10）
　　＊　OPENR␣CT0：DATAX｛ここで＜CR＞を入れると，読取時にErrorが発生する｝；N＝1；DO（10）＜READ␣X（N）；N＝N＋1＞；CLOSE＜CR＞｛ここまでを1行にまとめて入力するか，一連のプログラムに組立てる。｝

5.5.5　$MODE及び$TERM命令
(1)　$MODE（1）　；READ命令で読込むDataの区切り及び終りを表す記号は，5.5.4の（3）の通りであるが，（Normal mode）この命令によって，Table 5.2の第1欄でデータの終りを指示可能になる。（Special mode）

Table 5.2　Terminational codes on the spacial mode

Charactor	Octal code	Decimal value
Space	40	32
Line feed	12	10
Comma	54	44
Semicolon	73	59
Left branket "["	133	91
Right branket "]"	135	93

(2) $TERM (0) ：前記のSpecial modeをNormal modeに戻す。またSpecial modeの間，数値1個を読取る毎に前記Tableの第3欄に示した区切記号に対応するDecimal valueが$TERMにReturnされる。

5.5.6 READS及びTYPES命令

(1) READS，TYPES命令は，変数及び配列に対してASCII stringの入出力を行う。

(2) Directed READS/TYPESも同じ機能を持ち，I/Oの指定番号もDirected READ/TYPEと同じである。

(3) 変数又は配列が読込み可能な文字数Nは次の通りである。

変数：N＝2*F－1

配列：N＝2*F*M－1

ここでMは配列の要素数，Fはメモリ内の語長であって，本システムではF＝3（標準値）である。[註15] この限度を超える分は無視される。入力字数がこの限度未満であれば，メモリ内の残部は“Null”（空白－Spaceとは別であることに注意）となる。

**［変数又は配列内のASCII stringは，他のそれと，IF命令によって等しい（“EQ”）か否か（“NE”）の判定を行うことができる。これにより対話型式でプログラムの流れを制御することが可能である。

（例） START：

```
……
TYPE!,“OK？”；READS␣ANS<CR>
IF (NE (ANS,“Y”) +NE (ANS,“YES”)) GOTO␣ START
<CR>
……
```

なるプログラムをKeyboard activeの状態で実行すると，コンソールは

OK？

と印字して，Keyboardから入力があるまで実行を中止する。ここでY又はYES次で<CR>と打てばプログラムは次へ進み，また，たとえばN<CR>と打てばプログラムは“START”なるターゲットの存在する行の先頭から実行を開始する。］**

註15） 5.4.6参照

6. Mass Storage Operations

6.1 Mass Storage Device又はVirtual Memory（仮想メモリ）について

(1) ここでは5.2で述べたグループ2及び3のI/Oにプログラム及びデータを入出力する命令について述べる。

(2) これらのI/OはVirtual memory（仮想メモリ）又はMass Storage Deviceと呼ばれる。(以下MSと記す)

(3) 6.1ではプログラムの入出力，6.2ではデータの入出力について述べる。

6.1.1 SAVE命令

(1) DEFINE命令等によりメモリに書込まれたプログラムをMSに書込む。(メモリの内容はもとのまま残る。)

（型式） SAVE␣{ dev : }{ プログラム名 }{ . ect }

ここで

{ dev : } ；プログラムが書込まれているI/O名。(Table 1.1のAbbriviation)省略すればSystem device SY（本SystemではRK0に同じ）と見なされる。

{ . ext } ；英数字で最大3文字のExtension。(但し“. SYS”と“. BAD”を除く。)省略すれば自動的に“. CPF”と附加される。

通常は{ dev : }と{ . ext } を省略して

SAVE␣{ プログラム名（2個以上の時は“，”で区切る）}

という型式で使用する。

(2) SAVE命令を実行する際には，同じMS内に同じプログラム名とExtensionの組合せがないことを確認する。[註16] もし存在する時は（或は初めから）REPLACE命令を使用する。(6.1.3参照)

(3) 一旦メモリ内にDEFINE命令で作製したプログラムはSAVE命令でMSに保存して後，実行又は修正を行うのを原則とする。(CLASSの停止によりメモリの内容が失われる場合に備えるため)

6.1.2 LOAD命令

(1) SAVE命令等でMSに書込まれたプログラムをメモリに入力する。(MSの内容は不変である。)

（型式） LOAD␣{ dev : }{ プログラム名 }{ . ext } ；{ dev : }{ . ext }

註16) Reference[2] §4.2.6，p. 4-15～18参照。

等はSAVEの場合と全く同じである。

(2) LOADを実行する時に，同名のプログラムがメモリ内にあればErrorとなる。この時はあらかじめDELETEを行う。

6.1.3 REPLACE命令

SAVEと同様に，メモリ内のプログラムをMSへ書込む。但しSAVEと異り，同名のプログラムがあればこれを消去して書込む。

(型式) REPLACE {dev：}{プログラム名}{.ext} ;{dev：},{.ext}

等はSAVEの場合と全く同じである。

6.1.4 REMOVE命令

(1) SAVE，REPLACE命令でMSに書込まれたプログラムを消去する。

(型式) REMOVE␣{dev：}{プログラム名}{.ext} ;{dev：},{.ext}

等はSAVEの場合と全く同じである。

(2) 本命令は6.2で作製したデータファイルの消去にも使用される。この場合はプログラム名の代りに対象となるデータファイル名を使用する。{.ext}を省略すれば".CDF"と見なされる。

(すべてのFile類の内容の消去は，System file内の"PIP"を使用して行うことができる。—Reference[2] §4.2.4，p.4-13～15参照)

6.1.5 RUN命令

(1) MS内のプログラムをメモリに移して実行する。

(型式) RUN␣{dev：}{プログラム名}{.ext} ;{dev：},{.ext}等はSAVEの場合と全く同じである。

この命令は次のような動作をする。

(1-1) 同名のプログラムが既にメモリ内にあればそれを実行する。

(1-2) もしなければ指定のI/Oよりプログラムをメモリに入れて実行し，実行が終ればそれをDELETE可能な状態に設定する。

(1-3) メモリがオーバーフローする状態になった時，(1-2)の終了状態になったプログラムは消去される。(LOAD命令で書込まれたプログラムにはこの処置はなされない。)

(2) RUN命令でメモリに書込まれたプログラムに対してEDIT，DUMP(10.2，10.3参照)，DELETE命令は使用できない。

6.1.6 DECLARE命令

(1) MS内のプログラムをメモリ内にあるものと同様にDO命令で実行可能なものにする。

（型式）　DECLARE␣{ dev : }{ プログラム名 }{ . ext } ;{ dev : } ,{ . ext }
等はSAVE の場合と全く同じである。

(2) メモリの余裕に応じて任意数だけDECLARE 可能である。

6.2 Random Access Data Fileの取扱い

CLASSでは，データの入出力にSerial access file（グループ1及び2のI／O）とRandom access file（グループ3のI／O）の2種類が使用可能である。前者については5.5及び6.1に述べた命令のみでデータの入出力を行う。後者については，これらの外に以下で述べる命令でデータの入出力が可能である。

6.2.1 CREATE命令

(1) Random access data file では，必ず前以てデータ入力用のSpacc を準備しなければならない。このために本命令を使用する。

（型式）　CREATE␣{ dev :}{ ファイル名 }{ . ext }{ ／switch }

ここで

{ dev : }　；Fileを作製するI／O名（Table 1-1のAbbreviation）

{ . ext }　；Extension，省略すればCDF と見なされる。

／{ switch }；次に示す3種のSwitchを指定する。

／F：n　又は　／F！n　；データブロックの数

／R：n　又は　／R！n　；1ブロック当りのデータ数

／W：n　又は　／W！n　；データ当りの語数，データの必要精度に応じ2，3，4の内から任意に選ぶ。

ここでnの値は，“：”の後では8進数，“！”の後では10進数で与える。

（例）　＊CREATE␣SY0 ：DATA1／F！10／R！256／W！3 <CR>

これによって，DATA1・CDFなるFileの領域がシステム・ディスク内に作られる。

(2) Switch中のnを変数で与える時は6.2.2の方法による。

(3) CLASSに於てData file上の各ブロックを指定するには，データファイル名に配列の如く添字を附して表す。前記例のFileで1，2，…… n番目のブロックは，DATA 1(1) DATA(2)，……DATA(n) と表される。

6.2.2 変数＄FV，＄RV，＄WVによるSwitch指定

CREATE命令で，変数によりSwitch内のnを指定する時は，F，R，Wの各Switchに対応して，＄FV，＄RV，＄WVなる特別な変数に，所要の数値を代入することにより行う。

（例）　＊＄RV＝Z＊5 <CR>

＊CREATE␣SY0 ：DATA1／F！2／R：0 <CR>

これによってR！n又はR：nに於けるnはZを5倍した数値となる。$RV等に対応するSwitchは本例の如くR：0等と記す。(Wは省略されたので標準語長に設定される)

6.2.3 OPENF命令

(1) あらかじめCREATEされたFile に対しデータの読み書きを行う前に必ずこの命令でFileをOpen する。

(型式) OPENF␣{dev：}{ファイル名}{.ext }{／switch }

ここでファイル名, .ext, ／switchはCREATEと同じに指定する。特に／switch の内容がCREATE時と異ればデータの読み書きが不正になる。

(2) データファイルはClose を必要としない。

6.2.4 COPY 命令

(1) COPY 命令によりデータをCREATEしたFile と配列の間で相互に転送する。Random access data file の場合, データは配列とFile の間でのみ転送可能である。

(2) COPYを行う前に, 必ず所要のFileをOPENする。但しCREATEした直後にそのFileへCOPY する場合を除く。

(3) File からメモリへ読出し

(型式) COPY␣{ファイル名}({Record＃})：{配列名}

ここでRecord ＃ はデータファイル中のブロックNO. である。配列は必ずそのFile の1ブロック当りのデータ数以上の大きさでなくてはならない。

(4) メモリからファイルへ書込み

(型式) COPY␣{配列名}：{ファイル名}({Record ＃})

詳細は(3)に同じ。

7. CLASS String Function

(1) RT／CLASSでは数値データの外に，文字並びをもデータとして，入出力，記録が可能である。文字データをStringと呼ぶ。

(2) 文字データの入出力については5.5.6のREADS，TYPES等を参照のこと。文字データが配列に入っていれば，6.2に述べたRandom access data fileとの入出力も数値データと同じに行うことができる。

7.1 Stringの記憶型式

(1) CLASSに於てStringは終りに"Null"（空白－Spaceとは別）を附して記憶される。従ってn文字の記憶には（n＋1）文字分，（n＋1）byteの領域を必要とする。数値用の領域に記憶できるStringの字数は5.5.6の(3)に述べた通りである。

(2) Stringは，次に述べるSTRING命令により，独自の領域を設けて記憶可能である。

7.2 STRING命令

(1) String data用のスペースを指定する。

（型式） STRING␣｛変数名｝（領域の大きさ）

（例）＊STRING␣XYZ(80)，PDQ(25)＜CR＞

本例では，XYZなる名前に80文字，PDQには25文字が記憶可能となる。

(2) 実際に入力可能な文字数は数値の記憶長をw(＝2，3，4)として，2nw-1(n＝1，2，……)なる式で表される文字数の内，〔指定した字数＋1〕より大きく，最も近い字数である。

(3) この領域に文字を入力するには次の2法による。

（3－1） READS命令を使う。

（例） ＊READS␣XYZ＜CR＞

（3－2） 7.3に述べる諸命令を使う。

7.3 String Function

(1) RT／CLASSではString dataに種々の処理を行う機能がある。本項ではそれらについて述べる。

(2) Stringは"……"で囲んで直接表すこともできる。また本節ではString名の例として

S1，S2，S3等を使用する。

7.3.1 関数$CONVERT

数値を指定されたFormatのASCII Stringに変換する。

（型式） $CONVERT（V，S1，F）

ここで

V： 数値又は変数名

S1： 変換したString dataを記憶するString名

F： 変換のFormat（5.4参照），省略した時は，現在シリアル出力のためのFormatが使用される。本命令で指定したFormatはシリアル出力のそのを変更しない。

7.3.2 関数$COMPARE

(1) 2つのStringの"Stringとしての値"を比較する。この値は各文字のASCII Codeの8進表示（Table 7.1）を数値とみなしたものである。

（型式） X＝$COMPARE（S1，S2，N）；S1とS2の先頭からN文字を比較し，Xに次の値を入力する。

S1＜S2 ならば -1

S1＝S2 ならば 0

S1＞S2 ならば +1

Nを省略すればS1とS2の全体を比較する。

（例） ABC＞ABB

68Q＞58Q

ABC＝ABC

ACB＞ABCDE

ABCD＞ABC

＊＊〔この比較の方法は次のように考えることが可能である。(a)Table 7.1の数値（先頭の0を含めて，各文字についてはすべて3桁に取る）を，文字の順に連続して並べる。(b)こうして得られた数を，すべて先頭に"0"を附してでき上った小数と考えて，その大小を比較する。

A＝0.101

B＝0.102

C＝0.103

ABC＝0.101102103

ABB＝0.101102102

∴ ABC＞ABB

他の組合せについてもすべて同様に扱うことができる。〕＊＊

Table 7.1 ASCII 7-bit octal codes

Note : [1]Null(空白),[2]Line feed(改行), [3]Cariage return(復帰), [4]Space, [5]特殊記号(各称不明, Comma/Apostropheとも異る)

Octal code	Char.	Octal code	Char.	Octal code	Char.	Octal code	Char.
000	NUL[1]	040	SP[4]	100	@	140	‘[5]
001	SOH	041	!	101	A	141	a
002	STX	042	"	102	B	142	b
003	ETX	043	#	103	C	143	c
004	EOT	044	$	104	D	144	d
005	ENQ	045	%	105	E	145	e
006	ACK	046	&	106	F	146	f
007	BEL	047	'	107	G	147	g
010	BS	050	(	110	H	150	h
011	HT	051	)	111	I	151	i
012	LF[2]	052	*	112	J	152	j
013	VT	053	+	113	K	153	k
014	FF	054	,	114	L	154	l
015	CR[3]	055	-	115	M	155	m
016	SO	056	.	116	N	156	n
017	SI	057	/	117	O	157	o
020	DLE	060	0	120	P	160	p
021	DC1	061	1	121	Q	161	q
022	DC2	062	2	122	R	162	r
023	DC3	063	3	123	S	163	s
024	DC4	064	4	124	T	164	t
025	NAK	065	5	125	U	165	u
026	SYN	066	6	126	V	166	v
027	ETB	067	7	127	W	167	w
030	CAN	070	8	130	X	170	x
031	EM	071	9	131	Y	171	y
032	SUB	072	:	132	Z	172	z
033	ESC	073	;	133	[	173	{
034	FS	074	<	134	\	174	|
035	GS	075	=	135	]	175	}
036	RS	076	>	136	↑	176	~
037	US	077	?	137	←	177	DEL[6]

7.3.3　関数$CONCATENATE

(1)　2つのStringを接続する。

（型式）　X＝$CONCATENATE（S1，S2，S3，N）

これにより，S1の後にS2を接続したものの頭からN文字までがS3に記憶される。（S1とS3は同じものでよい。）

(2)　Nは省略可能である。この時接続されたStringのS3に入り切らない部分は無視される。

(3)　Xには接続されたStringの全部がS3に入った時は0，オーバーした時は－1が入力される。

7.3.4　関数$COPY

（型式）　$COPY（S1，S2，N）

(1)　S1中のN文字をS2にCopyする。Nを省略した時は全文字をCoyyする。

(2)　X＝$COPY（S1，S2）とした時，S2にS1が入り切ればXに0，オーバーすれば－1が入される。オーバーした分は無視される。

7.3.5　関数$EVALUATE

(1)　Stringを数値に変換する。（7.3.1，$CONVERTの逆を行う。）

（例）　＊$COPY（"123.07"，S1）<CR>

＊X＝$EVALUATE（S1）<CR>と実行すれば，X＝123.07となる。

(2)　このString中で，文字"E"及びTable 2.1中の記号はすべて使用可能である。変数名は使えない。

7.3.6　関数$INDEX

（型式）　$INDEX（S1，S2，N）

(1)　S1中に，S2のStringと同じ文字配列の所を検出し，N回検出したら，その部分の最初の文字がStringの先頭から何字目かの数値をReturnする。

(2)　Nを省略すればN＝1と見なされる。また指定の文字並びが検出されなければ数値0をRetunする。

（例1）＊X＝$INDEX（"ABDEFCDDEFEF"，"DEF"，2）<CR>

を実行すればX＝8となる。

（例2）＊X＝$INDEX（"ABDEFCDDEFEF"，"DEF"）<CR>

を実行すればX＝3となる。

7.3.7　関数$INSERT

（型式）　$INSERT（S1，S2，I，J）

(1)　S2のI番目からJ個の文字（Jを省略すればS1に含まれる字数だけ）をS1の文字に変更する。

(2) X＝$INSERT (S1, S2, I)とした時，S2の文字数がS1の全文字と置換可能な大きさであればX＝0，不足ならばX＝-1となる。

7.3.8 関数$LENGTH

(型式) X＝$LENGTH (S1)

Xには現在記憶されている文字数が入力される。

7.3.9 関数$LMAX

(型式) X＝$LMAX (S1)

S1に記憶可能な最大の文字数がXに代入される。

7.3.10 関数$SUBSTRING

(型式) $SUBSTRING (S1, S2, I, J)

(1) S1のI番目からJ個の文字(Jを省略すれば最後まで)をS2にCopyする。

(2) X＝$SUBSTRING (S1, S2, I, J)とすれば，S2に記入可能な文字数が実行するのに充分な時X＝0，不足すればX＜0となる。

7.3.11 関数$TRANSLATE

(型式) $TRANSLATE (S1, S2, S3)

S1中にS2に含まれる文字があれば，それに対応するS3中の文字と置換する。

(例) ＊$COPY ("FRED IS IN THE ROOM", S1)＜CR＞

＊$TRANSLATE (S1, "REI, " PD)＜CR＞

本例ではRをPに，EをDに変更し，Iを消去(対応する文字なし)する。結果としてS1の内容は" FPDD S N THD POOM "となる。

7.3.12 関数$SUPPRESS

(型式) $SUPPRESS (S1)

Stringの頭にあるSpaceをすべて消去する。

7.3.13 関数$TRIM

(型式) $TRIM (S1)

String末尾のSpaceをすべて消去する。

7.3.14 関数$VERIFY

(型式) $VERIFY (S1, S2)

S1内に，S2に含まれない文字が最初に現れた所が何文字目かを数値でReturnする。

(例) ＊X＝$VERIFY ("FRED IS IN THE ROOM","ABCDEF"＜CR＞

では最初のRが2文字目に現れるからX＝2となる。

8. 間 接 指 定

CLASSでは次のものの名前を間接指定可能である。

Variable（変数）

Array（配列）

プログラム

Target

File

指定法は以下の通りである。

8.1 プログラムの間接指定

(1) 指定すべきプログラム名をStringとして変数名に書込む。

(2) その変数名の頭に"@"記号を付して，内容のStringと同じ名前のプログラムを実行する命令とする。

たとえばREADS ANSとして，変数ANSに"M10"なる文字並びを入力し，@ANS<CR>と操作すれば，別にメモリに書込んだ"M10"なるプログラムが1回実行される。

8.2 変数，配列，ターゲットの間接指定

プログラムに準じて行う。即ち，

(1) 指定すべき名前をStringとして変数名に書き込む。

(2) その変数名の頭に"@"記号を附けることにより，内容のStringと同名の変数，配列，ターゲット名と同等に使用可能となる。（ターゲット名としてはGOTO命令でのみ使用する。）

8.3 Fileの間接指定

8.3.1 Serial FileのOPENW，OPENRの場合

(1) 指定すべき名前，ExtensionをStringとして変数名に書込む。

(2) その変数名の頭に"&"記号（8.1，8.2と異る）を附けることにより，内容のStringと同名，同じEtensionの組合せとして使用可能となる。

8.3.2 Random Access FileのCREATE，OPENFの場合

8.3.1と同様に"&"記号で間接指定を行う。String並びではF，R，W等のSwitch(6.2)も含める。

8.3.3　Random Access File と配列間の COPY の場合

8.3.1～2 と異り "@" 記号を用いる外は前記と同じである。

9. Utility Command

9.1 システム変数$S

(1) $SはCLASS固有の変数（System variable －以下システム変数と呼ぶ)であり，その値は常にメモリのUser's area 中の空いている部分のByte数に等しい。

（例） ＊TYPE␣ $S＜CR＞
12758.000
＊

本例はCLASS起動直後の状態であって，User は 12758 byte 分までメモリに入力可能であることを示す。

(2) 入力の際に必要なメモリ領域は次の通りである。

a. Text の1字毎 － 1 byte
b. 変数1個毎 － 14 byte （Symbol に8，数値に6 byte
c. 配列 － 20 byteに数値用として各要素毎に6 byte 増加
d. Defined program － 10 byte に加えて，1文字毎に1 byte 増加（＜，＞，＜CR＞，＜LF＞等もすべて1文字として数える。

(3) $S＜CR＞と使用することにより，その時メモリに存在するすべてのプログラム，変数を消去することができる。（PROTECT操作 － 9.3参照 － を行ったものを除く。）

9.2 FIX命令

(1) 本命令は，定義されたプログラム，変数をPermanent な状態とする。

（型式） FIX␣｛プログラム名／変数名｝；｛ ｝部に2個以上の名前がある時は各々を"，"で区切る。

(2) この命令で処理されたプログラム，変数はCLASS固有のものと同等となり，DELETE，ERAS及びEDIT（10.2に述べる)で処理不可能となり，CLASSを一旦停止するまで保持される。

9.3 PROTECT命令

（型式） PROTECT␣｛プログラム名／変数名｝；｛ ｝部に2箇以上の名前がある時は，各々を"，"で区切る。

(1) この命令で処理されたプログラム，変数，配列は，DELETE ALL， ERASE

ALL等では消えず，DELETE｛プログラム名｝，ERASE｛変数名／配列名｝なる操作で各個に消去可能である。

9.4 INDEX命令

（型式）　＊INDEX＜CR＞

(1) メモリ内に存在するプログラム名，変数名等を次の区分記号と共にListとして出力する。

F － DEFINEされたプログラム

V － 変数

A － 配列

D － DECLAREされたFunction

O － OpenされたFile

（PROTECTで処理されたものは記号の直後に"＊"が出力される。）

(2) Listの出力順序は，メモリ入力（EDIT － 10.2 － の終了を含む）の逆順となる。

（型式2）　＊INDEX␣ALL＜CR＞

(1)の出力に続けてCLASS内部のFunction名に符号"S"を附けて出力する。

9.5 RENAME命令

(1) すべての名前（命令，変数，配列，プログラム等）を変更する。

（型式）　RENAME␣｛現在の名前$_1$｝：｛変更後の名前$_1$｝，｛現在の名前$_2$｝：｛変更後の名前$_2$｝……＜CR＞

(2) CLASS固有，又はUser defined のいずれにも使用可能である。但しUser defined の名前はメモリ内のものに限る。

9.6 REMARK命令

(1) プログラム中にComment を挿入する。プログラムの動作には全く関係しない。

（型式）　REMARK␣｛文字並び｝｛；又は＜CR＞｝

(2) "；"又は＜CR＞が現れる所まで有効である。

＊＊〔文字並びの中には上記の外に"："を使用できない。〕＊＊

9.7 KBWAIT命令

コンソールのKeybard のいずれかのKey が打たれるまで，プログラムの進行を中止する。

9.8 QUIT命令

すべてのプログラムの実行を停止して，次の命令待ちの状態とする。（STOP はそれを含むプログラムのみを停止させる。）

9.9 関数DEF

（型式）　DEF（｛プログラム名／変数名／配列名｝いずれか1個のみ｝

（　）内の名前を持ったプログラム，変数，配列が存在すれば1，なければ0なる値が Return される。

9.10 LOCAL命令

(1) DEFINE命令で作製したプログラムの内部でのみ使用される変数を指定する。

（型式）　LOCAL␣｛変数名1｝，｛変数名2｝，etc

(2) そのプログラム内でのみ必要な配列を単独又は変数と同時に指定することができる。

（例）　LOCAL␣A（20）

(3) これらの変数，配列はこれらを含むプログラムの実行停止と共に自動的に消去される。また同時にメモリ内に存在する他のプログラム内に同名の変数，配列が存在しても，それらは互に全く影響を与えない。

9.11 時刻変数$TIME

(1) この変数にはRT－11 System内のClockの時刻（秒単位）に電源周波数を乗じた数値が入っている。[註17] 故に適当なFormatに変換し，時刻のデータとして利用可能である。

(2) 時刻を｛時：分：秒｝なるFormatで出力するプログラムの一例をFig. 9.1に示した。本例では時・分・秒の各数値を2文字のStringに変換し，（分・秒の値がそれぞれ10未満の時は，先頭に"0"を挿入する）Colon"："を含めた8文字を"$CLOCK"なる名前の[註18] 配列に一旦記入している。

(3) 上記のプログラムを，電源周波数60 Hzで使用する時は，3行目の

X＝X／50／60／60，H＝IP（X）

にある数字"50"を"60"に変更する。

註17) 1.3.4の（2－2）参照

註18) "$"を始め一部の記号はこのように名前の中で使用可能であるが，CLASS固有の変数名・関数名等と同一にならぬよう注意を要する。

```
*LOAD TIME;DUMP TIME

DEFINE TIME   <LOCAL X,S,M,H
X=$TIME;IF(EQ(DEF($CLOCK)))DIMENS $CLOCK(2)
X=X/50/60/60,H=IP(X)
X=FP(X)*60,M=IP(X)
X=FP(X)*60,S=IP(X)
$CONVERT(H,$CLOCK(1),$I(2))
$CONCATENATE($CLOCK(1),":",$CLOCK(1),3)
IF(LT(M,10))GOTO M0
$CONVERT(M,X,$I(2));GOTO S10
M0:$CONVERT(0,X,$I(1))
$CONCATENATE($CLOCK(1),X,$CLOCK(1),4)
$CONVERT(M,X,$I(1))
S10:$CONCATENATE($CLOCK(1),X,$CLOCK(1),5)
$CLOCK(2)=":";IF(LT(S,10))GOTO S0
$CONVERT(S,X,$I(2));GOTO FIN
S0:$CONVERT(0,X,$I(1))
$CONCATENATE($CLOCK(2),X,$CLOCK(2),2)
$CONVERT(S,X,$I(1))
FIN:$CONCATENATE($CLOCK(2),X,$CLOCK(2),3)
TYPES $CLOCK(1),$CLOCK(2),!
PROTECT $CLOCK>

*


RT-11SJ    V02C-02B

.TIM
00:00:07

.TIM 17:04:28

.TIM
17:04:30

.R CLASS

CLASS V04.23-RT
15-FEB-77


*RUN TIME
17:04:41

*
```

Fig.9.1 "TIME" program and its applications

9.12 日附変数$DATE

(1) RT－11 Systemに{ y }年{ m }月{ d }日なる日附が設定されていれば，この変数には$m*2^{10}+d*2^5$なる数値が入っている。故に適当な変換によって日附のデータとして利用可能である。

(2) Fig. 9.2に$DATEのデータを{日－月－年}なるFormatで出力するプログラムの一例を示す。（月は英語名の先頭3文字で表現）

(3) $DATEには年号の数値を含まないので，そのまま本プログラムを使用すれば，年号として日と同じ数字を出力する。この補正には{日}－{年}に相当する数値をArgumentで入力する。この補正値は"$DIFYR"なる変数（PROTECT処理される）に保存されるから，CLASSを停止するまでは再補正を必要としない。

註19) 1.3.4の（2－1）参照

```
*LOAD DATE;DUMP DATE

DEFINE DATE   <LOCAL YE,DA,MO,X,SW1
IF(EQ(DEF($CALEN)))DIMENS $CALEN(2);$DIFYR=0
IF(GT(ARG(0)))$DIFYR=ARG(1)
IF(EQ(DEF(CALEN)))SW1=0
X=$DATE
X=X/32/32
MO=IP(X)
X=FP(X)*32
DA=IP(X)
YE=X+$DIFYR
IF(NE(SW1))GOTO LOOP1
LOCAL CALEN(12)
DATA CALEN<'JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC'>
SW1=1
LOOP1:MO=CALEN(MO)
$CONVERT(DA,$CALEN(1),$I(2))
$CONCATENATE($CALEN(1),'-',$CALEN(1),3)
$CONCATENATE($CALEN(1),MO,$CALEN(1),5)
$SUBSTRING(MO,$CALEN(2),3,1)
$CONCATENATE($CALEN(2),'-',$CALEN(2),2)
$CONVERT(YE,DA,$I(2))
$CONCATENATE($CALEN(2),DA,$CALEN(2),4)
TYPES $CALEN(1),$CALEN(2),!
PROTECT $DIFYR,$CALEN>

*


RT-11SJ    V02C-02B

.DAT 14-MAR-79

.DAT
14-MAR-79

.R CLASS

CLASS V04.23-RT
15-FEB-77


*RUN DATE
14-MAR-14

*RUN DATE(65)
14-MAR-79

*RUN DATE
14-MAR-79

*
```

Fig.9.2 "DATE" program and its applications

10. プログラムの修正，保存のための入出力操作

10.1 一般

(1) 本章に述べる命令は，すべてメモリ内のプログラムに対してのみ有数である。従ってグループ2及び3のI／O（5.2参照）にSAVE命令で記憶されたプログラムは，一旦LOAD命令でメモリに書込む。

(2) 修正を行ったプログラムは，直ちにREPLACE命令によりDisk等の内容を更新し，CLASSの停止等によるメモリ内容の消失に備える。

(3) プログラムの作製，編集，修正等は，System file内のText editor（EDIT）を使用して行うことも可能であり，プログラムが長い時は特に有数である。[註20)]

(4) 紙テープ，Cassette，MT，Disk等相互間でのプログラム（及びデータ）ファイルの Copy は，System file内の"PIP"で行うことも可能である。[註21)]

10.2 EDIT命令によるプログラムの修正

(1) KeyboardにEDIT␣{プログラム名}<CR>と打てば，CLASSはターミナルに"〔"と印字して一旦停止する。ここでCLASSに添て使用可能な文字等を一つだけ打てば，CLASSはその次へ"〕"を印字し，引続いて先に打った文字等（Search charactor と呼ぶ）が最初に現れるまでプログラム・リストを出力し，そこで一旦停止する。（或はそのまま(2)の操作を行ってもよい。）

(2) ここで以下の操作を行う。

（2－1） 文字類，<CR>等を打てばそのまま挿入される。

（2－2） Rubout keyを打てばその数と同じ字数だけ始めの方へ消去が行われる。（"＼"がEchoされる。）

（2－3） CTRL／Dと打てばその数と同じ字数だけ終りの方へ消去が行われる。（"%"をEchoする。）[註22)]

（2－4） CTRL／R と打てばPointerは1字ずつ終りの方へ進む。（1字ずつListを出力）

（2－5） CTRL／Eと打てばPointerの所から終りの方全部が消去される。

（2－6） Alt mode key（ALT又はESC＝Escape）を打てば，Pointerの所から始めの方全部が消去される。

（2－7） CTRL／Vを打てばその度毎に次行の先頭にPointerが進む。（その行の先頭の文

註20) Reference[2)] Chapter 3，p・3－1～35.

註21) Reference[2)] §4.2.2及び§4.2.3，p・4－9～13.

註22) 改訂前はCTRL／S，これを打つとSystem全体が停止するから注意。

字をEchoする。)

(2−8) CTRL/Tを打てばEditは終了する。(1及び2−4, 2−5, 2−7, 2−9, 2−10の操作でPointerがプログラムの最後まで行った時もEditは終了する。)

(2−9) CTRL/Lと打てば次のSearch charactorの所までPointerが進む。

(2−10) CTRL/Gと打てば新しいSearch charactorを指定できる。("〔"がEchoされるから, 後は1と同じ操作をする。)

(3) Editは(2)の命令で終了した時に完成する。これ以外の原因で打切られた時, プログラムの内容は修正前のままである。(メモリのオーバーフローによるError発生, CLASSの再起動を行った時等)

10.3 DUMP命令

本命令は, メモリ内のプログラム(DEFINE, EDIT命令等で作製又は修正したもの)をSerial fileに出力する。

(型式1) DUMP␣{プログラム名……}<CR>

(以下10.3に於て"プログラム名……"とあれば, プログラム名が2個以上ある時は各々を", "で区切ることを示す。) プログラムを, 名前を並べた順に出力する。

(型式2) DUMP␣ALL<CR>

メモリ内のプログラムを, 後から記憶された(Editが完成した時も, 新しく記憶されたものと同等になる)順にすべて出力する。

10.3.1 プログラム・リストの出力

(1) ターミナル出力の時は次の順に操作する。

(例1) ＊$OT<CR> ;他のI/OがすべてCloseされている時(通常の場合)は不用。

＊DUMP␣{プログラム名……}(又はALL)<CR>

(2) ライン・プリンタ出力の時は次の順に操作する。

(例2) ＊OPENW␣LP:<CR>

＊DUMP␣{プログラム名……}(又はALL)<CR>

＊CLOSE<CR>

(3) 高速紙テープパンチ出力の時は次の順に操作を行う。

(例3) ＊$OH<CR>

＊DUMP␣{プログラム名……}(又はALL)<CR>

＊TYPE"$IK", !<CR>

＊$CHO<CR>

(3) Cassetteへ出力の時は次の順に操作を行う。

(例4) ＊OPENW␣CT{n}:{ファイル名}.{Extension}<CR>
＊DUMP␣{プログラム名……}(又はALL)<CR>
＊TYPE"CLOSE", !<CR>
＊CLOSE<CR>
＊

{n}はCassette drive unitにより0(左側)又は(右側)とする。{ファイル名}はプログラム名のいずれかと同一であってもよい。

(4) MT存びDiskへの出力方法は,(例3)に於ける"CT{n}"がI/Oに応じて"MT{n}", "RK{n}"となる外は同一である。(Diskの場合には"SAVE", "REPLACE"等の命令も使用できることは言うまでもない。

(5) Cassette, MT, Disk等に2個以上のFileを書込む時,ファイル名とExtensionの組合せに,同一のものが2個以上あってはならない。[註23] また不用のFileの消去は,System file内の"PIP"を使用して行うことが可能である。[註24] またこれらのInitializationも同様である。[註25]

(6) コンソール,ラインプリンタ以外への出力時には,リストの直後へ必ず"$IK"又は"CLOSE"の文字と,<CR><LF>を出力させる点に注意。これらは,I/OがFileを読取ってメモリに書込みを終った時,そのI/Oに対してCLOSEを自動的に行わせ,コンソールのKeyboardをOpenさせるためである。(SAVE又はREPLACE命令で作製したFileを除く。)

10.3.2 プログラムの入力

10.3.1で出力したプログラムを実行のためメモリに入力する時は次の方法による。

(1) 紙テープの入力は,紙テープを高速紙テープリーダに装着して後,次の命令を実行する。

(例1) ＊$IH<CR>

(2) Cassetteからの入力は,次の命令を実行する。

(例2) ＊OPENR␣CTn:{ファイル名}.{Extension}<CR>

CTnは10.3.1の(3)と同じ,nは(入力時と異ってもよい。)またファイル名とExtensionは出力時と同じものを使う。

(3) MT, Diskからの入力は,(2)に於けるCTnがMTn又はRKnとなるだけで他は同じである。[註26]

註23) Reference[2] §4.2.4, p.4−13〜15.

註24) Diskの場合"REMOVE"命令が使用できる(6.1.4参照)

註25) Reference[2] §4.2.7, p.4−18〜19.

註26) Diskの場合はLOAD, RUN等の命令が使用可能である。(6.1.2, 6.1.5参照)

11. Kernel Function及びQUANTAプログラム

(1) RT－11／CLASSは，10までに述べた命令類の外に，Model 8100 MCAを，計算機により制御し，かつMCAにDataの入出力を行うための命令を有する。

(2) MCAを制御し，又はこれにDataを入出力する命令を使用する時は，必ず前以てMCAパネル面のI／O－switchを"REMOTE"に切替える。これを忘れるとCLASSの再起動が必要となる。

11.1 Kernel Function

Kernel functionは，MCAを制御する基本命令である。

11.1.1 $MC命令

（型式） $MC（n）

MCAが2台以上計算機に接続されている時，MCAを番号nで指定する。CLASSを起動した時には常にn＝1とした時の状態となる。

11.1.2 CONTROL命令

（型式） CONTOL｛5文字のString｝

ここで｛｝内のStringは次のA～Cのいずれかの要領で指定する。

A. 5文字のStringを直接与える。

（例1） CONTROL "ABCDE"

B. 5文字が書込まれたString variableで与える。

（例2） VAR＝"ABCDE"

CONTROL VAR

C. 書き込まれた文字の合計が5文字となるようなString variable並びで与える。

（例3） V1＝"AB"，V2＝"CD"，V3＝E

CONTROL V1，V2，V3

＊＊〔String直接とVariableの組合せで与えることも可能である。(例3)の場合は次のようにしても同等の結果となる。

CONTROL "AB"，V2，V3

CONTRO V1，"CD"，V3 etc.〕＊＊

(1) ここでCONTROL "c_1 c_2 c_3 c_4 c_5"なる型式で命令を書くと，c_1～c_5はそれぞれ次の機能を持つ。

c_1 ；MCAのChannel領域の指定

c_2 ；Time－LSDの指定

c_3 ；Time－MSDの指定

（c_2，c_3にそれぞれn，mなる数字を与えれば，Timerの設定時間が$n*10^m$（PHA－modeの時）又は$n*10^{-m}$（MCS－modeの時）secとなる。

c_4 ；Modeの設定

c_5 ；ROI（＝Region of interest）の制御

実際に与えるべき文次等とその機能をTable 11.1に示す。

(2) 本命令でTimerの時間を設定する時は，MCA本体のTimerのM，Nを共に0に設定する。また本体側のTimerを使用する時には，c_2，c_3共に必ず"0"を与える。

11.1.3 WAITD命令

(1) WAITD（＝Wait for done）命令は，MCAが現在行っている動作が完了するまで計算機を待機させる。

(2) CONTROL命令でMCAがCollect，Readout（MCA側のMT，Plotter等）のModeに切替える時は，次のCONTROL命令が現れる前に必ずWAITDを使う。

＊＊〔MCAのメモリをClearする命令の後にも必要な場合がある。〕＊＊

(3) WAITDは1行に1回だけ使用できる。

11.1.4 関数LOW

(1) 本命令はROIに関する情報を入出力する2種類のKernel functionの内の1つであって，次の2つの一般型を持つ。

（型式） A： X＝LOW ；MCAから計算機へ情報を送る。

B： X＝LOW（n） ；計算機からMCAへ情報を送る。

(2) 型式のAの場合は次のように使用される。（Fig. 11.1参照）

A. MCAがNot busyであることを確認する。

B. CONTROL"c_1 0 0 JC"なる命令を実行する。ここでc_1は目的のChannel領域に適合する内容とする。

C. VAR＝LOWなる命令を実行する。

Fig. 11.1のデータの場合，VARには＋5なる数値が入力される。これは現在位置（最初にLOWを使用する時は，必ず＃0 channelから最初のIntensfied(或はHigh) channelまでの，Non－intensified（又はLow）channelの個数である。実行後の現在位置のChannel）は6となる。

D. もしこの状態のままで再びVAR＝LOWを実行すると，VARには数値0が入力され，現在位置のChannel＃は7となる。

E. もし最後のChannelが検出されれば，VARには負の値が代入される。

(3) 型式のBの場合は次のように使用される。

Table 11.1 Control codes for "CONTROL" command

C_1		C_2		C_3	
Charactor:	Function: Memory Select:	Charactor:	Function: Time-LSD:	Charactor:	Function: Time-MSD:
A	1/1	1	1	1	1
B	1/2	2	2	2	2
C	2/2	3	3	3	3
D	1/4	4	4	4	4
E	2/4	5	5	5	5
F	3/4	6	6	0	0
G	4/4	7	7		
H	A/2	8	8		
I	A/4	9	9		
		0	0		

C_4		C_5	
Charactor:	Mode Selection:	Charactor:	ROI Control:
A	PHA-ADD	A	INTENS On
B	PHA-SUB	B	INTENS and BAND ENTER Off
C	MCSS	C	INTENS and BAND ENTER On
D	MCSR	D	BAND ENTER Off
E	MMCS	@	Do not alter current state
F	Display		
G	Plotter Out		
H	Tape Out		
I	Remote Out (from 8100 to PDP-11)		[X-ray Option]
J	Remote In (from PDP-11 to 8100)	E	Enter Cursors
K	Clear All	F	Display Cursors and INTENS
L	Clear Channel #0	G	Display Cursors
M	Transfer		

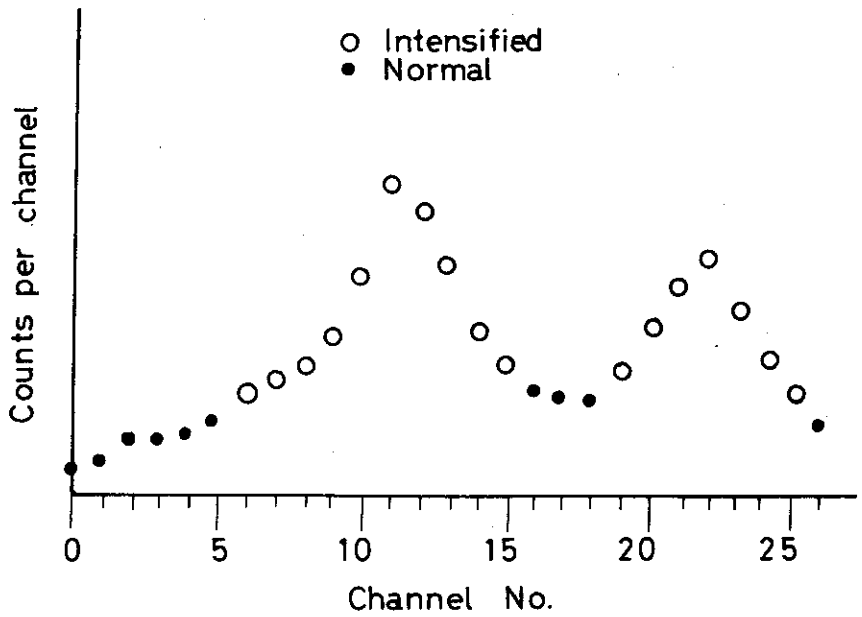


Fig.11.1 Relations between the functions "LOW", "HIGH" and ROI (=region of interest)

A. MCAがNot busyであることを確認する。

B. CONTROL " c_1 0 0 IC " なる命令を実行する。ここで c_1 は目的のメモリ領域に適合する内容とする。

C. X＝LOW（n）なる命令を実行する。これにより現在位置からn個目までのChannelのIntensifyがLow（或は0）の状態にSetされる。

D. もし最後のChannelが検出されれば負の値がReturnされる。

11.1.5 関数HIGH

(1) X＝HIGHなる型式で使用すると，Non－intensified channelが見つかるまでIntensified channelをSkipし，SkipしたChannelの個数をReturnする。

(2) X＝HIGH（n）なる型式で使用すると，現在位置よりn個のChannelをIntensifyする。

(3) HIGHとLOWは通常組合せて使用し，ROIの位置を検出し，又ROIを設定する。

11.1.6 TRANSFER命令

(1) TRANSFER命令は，MCAと計算機との間で実際のDataの入出力を行う。

（型式） TRANSFER（a, b, c, d）

ここで

a ；Dataを入出力する配列の名前

b ；Transferを行う配列の最初の添字，必ずしも1から行わなくともよい。

c ；Transferを行うChannelの個数

d ；Transferの方向

－1 MCAから計算機へ

＋1 計算機からMCAへ

(2) 本命令も，LOW・HIGHと同じく直前に適当な内容のCONTROL命令を実行する。例えばCONTROL " c_1 0 0 IB " 等。これに引続く最初のTRANSFER命令は，常にMCAの＃0 channelから順にDataの転送を行う。

11.2 QUANTAプログラム

(1) QUANTAは，計算機によるMCAの制御を容易に行うため，前節のKernel functionを利用して組立てられた，CLASS語プログラムのセットである。現在，次に述べる10種のプログラムが使用可能である。

（他にTIMER，PUT，GETの3種については，制御される機器を設置する予定がないため省略した。）

(2) 使用に先立ち次のようにコンソールへ入力する。

（例） ＊RUN QUAUTA＜CR＞

これにより次節以下に述べる各プログラム，及び所要の変数がメモリ内に記憶される。

(3) 上記の操作が行われると，それまでにメモリ内にあった変数，プログラム等はすべて消却される。（PROTECTを行ったものを除く。）また，DELETE ALL，ERASE ALLの命令が実行された後では，上記の操作を再び行わねばならない。

11.2.1 CLEAR

CLEARはMCAメモリのClear又はResetを行う。

（型式1） CLEAR

Argument のない時は，全Channel のメモリをClear する。

（型式2） CLEAR（Arg）

Arg（＝Argument）により，Table. 11.2 に示した部分のメモリをClear する。

11.2.2 DISPLAY

MCAを強制的にDisplay mode とする。

次の3型式が許される。

（型式1） DISPLAY ；全メモリが強制的に選出され，INTESIFYとBAND ENTERは強制的にOff となる。

（型式2） DISPLAY（Arg 1） ；Table 11.2 のメモリ領域が選出される外は前記に同じ。

（型式3） DISPLAY（Arg 1，Arg 2） ；Arg 1でTable 11.2 のメモリ領域が選出される。Arg 2でTable 11.3 によりROI の制御が行われる。

11.2.3 COLLECT

MCAをCollect mode とする。次の型式のみが許される。

（型式1） COLLECT（Arg 1，Arg 2，Arg 3）

ここで

Arg 1 ；メモリ領域の指定（Table 11.2参照）

Arg 2 ；Data collection mode（Table 11.4参照）

Arg 3 ；NMなる2数字で表されたPreset time．MCAでは（$N*10^M$）second として扱われる。

（型式2） COLLECT（Arg 1，Arg 2，Arg 3，Arg 4）

ここで

Arg 1～Arg 2 ；型式1に同じ。

Arg 3 ；型式1のArg 3のNを指定する。

Arg 4　；型式1のArg 3のMを指定する。

11.2.4　XFER

本命令はTRANSFERの動作を行う。使用可能な型式はXFER（Arg）のみである。ArgumentはTransferされたDataを書き込むメモリの領域をTable 11.2により指定する。

11.2.5　PLOT

本命令は，MCAに接続されたX－Y　Plotterにメモリ内のDataをPlotさせる。型式はPLOT（Arg）であって，ArgumentはTable 11.2によりPlotすべき領域を指定する。

11.2.6　WRITE

MCA側の磁気テープ（Industry－compatible）にDataを記録する。次の2型式が許される。

（型式1）　WRITE（Arg）　；Argumentで指定されたメモリ領域（Table 11.2による）の内容をMTに書き込む。

（型式2）　WRITE（Arg 1，Arg 2）　；Arg 1で内容をMTに書き込むメモリ領域を，Arg 2で書き込むべきDataの種別を選定する。（Table 11.2 及び3による。）

＊＊〔MCAの磁気テープ出力は，1024 channel分を1　recordとして行われ，各Recordの先頭には，磁気テープ制御装置のパネル面に設定された識別番号（Tag word）が附加される。これらが出力すべきChannelの個数に応じて1，2又は4 recordが1回のデータ出力毎に記録される。

これらのRecordは記算センタ大型機のFORTRANで書式つきREAD文を使用して読み取ることができる。変換のFormatは，識別番号・各Channelの内容共に"I6"である。識別番号を変数"ID"に，各Channelの内容（1024 channel分）を"MC"なる配列に読み込ませるための命令類は次のようになる。

```
        DIMENSION MC (1024)
        READ (L, 500)  ID, MC
500     FORMAT (I6, 8(128 I6))
```

ここでLは大型機システムにより指定された，I／OのLogical №である。変換のFormatを"I6，1024 I6"，"1025 I6"等の如く指定すると，システムのErrorが発生する。

実際に大型機で磁気テープの処理をする際には，"ラベルなし磁気テープ"のControl cardを使用し，Logical №もこれにより指定する。〕＊＊

11.2.7　PRINT

本命令はSpectrumの生データを計算機のターミナル或はラインプリンタのいずれかOpen

されている方へ出力する。(Argument 不用) IntensifyされたROIを，最大50領域までPrintする。

11.2.8 XSTOP

本命令はMCAを強制的にStop modeにする。Data collection又は動作中のI／Oはすべて停止する。

11.2.9 INPUT

(1) 本命令はMCAのDataを計算機で読み取るために使用する。

INPUT (Arg 1, Arg 2)

ここで

Arg 1 ；メモリ領域の指定 (Table 11.2による)

Arg 2 ；ROI control (Table 11.3による)

(2) 実際にはこの後にTRANSFER命令 (11.1.6参照) を実行することによりDataがTransferされる。

11.2.10 OUTPUT

本命令は計算機からMCAへDataをTransferするために使用する。型式，Argumentはすべて INPUTに同じであり，この後にTRANSFER命令を実行することも同じである。

11.2.11 Argumentの数値 (Value)

Table 11.2～11.4に〔Value〕と示した欄は，この"〔 〕"内の数値を定数又は変数の型で，対応するArgumentの代りに使用できることを示す。(Table 11.3の"@"を除く)

Table 11.2 Memory region specifiers

Argument	[Value]	Definition
TIME.	[0]	Channel #0 only
FULL	[1]	Full memory
H1	[2]	First half only
H2	[3]	Second half only
Q1	[4]	First quarter only
Q2	[5]	Second quarter only
Q3	[6]	Third quarter only
Q4	[7]	Fourth quarter only
A2	[8]	Remotely selected half
A4	[9]	Remotely selected quarter

Table 11.3 ROI control arguments

Argument	〔Value〕	Definition
REGION	〔1〕	Intensify がOn となり，MCAに入出力の時はROI部分のみが対象となる。
DATA	〔2〕	Intensify と Band enter がOff となり，Data のみが Display 又は入出力される。
BAND	〔3〕	Intensify と Band enter がOn になる。ROIの位置情報のみが入出力される。
"@"	〔−〕	Intensify と Band enter を現状のままにする。

Table 11.4 Data collection modes

Arguments	[Value]	Definition
ADD	[1]	PHA, Add to memory
SUB	[2]	PHA, Subtract from memory
MCSS	[3]	MCSS, Add to memory
MMCS	[5]	MMCS, Add to memory

12. プログラムの実例

本章では，著者の開発したプログラムについて述べる。

12.1 LIN

(1) 本プログラムは，MCAで得られた2個の光電ピークのチャンネル番号をX1，X2とし，それぞれに対応するエネルギーの値をY1，Y2とする時，これを式Y＝A＊X＋Bで表わされる直線上の2点の座標とみなして係数A，Bの値を計算し，

Y＝｛Aの値｝＊X＋（｛Bの値｝）

なるFormatで結果を出力する。

(2) 使用型式には次の2種類がある。

（型式1） RUN LIN（X1，Y1，X2，Y2） ；座標の値をArgumentで与える。

（型式2） RUN LIN ；Argumentが3個以下か又は省略された時は，対話型式で座標の値を入力する。

(3) プログラムのリストをFig. 12.1に示す。ターゲットの直後，又はターゲットのない行の先頭の空白部は，Spaceではなく，TAB keyを打って生じさせたものである。

(4) Fig. 12.2に使用例を示す。5及び6番目の用例は，81 keV及び1274 keVの光電ピーク中心がMCAの171及び2654 channelであった時，関係式の係数を2通りの型式で計算したものである。

(5) 特に，X1＝X2の場合は，最後の使用例の如くX1又はX2の一方（本プログラムではX2の方）の値のみをこのように出力する。即ち直線は点（X1，0）又は（X2，0）を通ってY軸に平行であることを示す。

12.2 AR

(1) 本プログラムは，MCAで得られたスペクトルのデータについて次の処理を行う。即ち，

Q ；光電ピーク領域（以下本節中では単に領域）内の各Channelのカウント数の合計

L，BL ；領域下端のChannel Noとそのカウント数

H，BH ；領域上端のChannel Noとそのカウント数

T ；データの積算時間（sec単位）

とした時，[註27)]

註27）これらのデータはすべてMCAのブラウン管Display上で読み取り可能のものである。

```
*LOAD LIN;DUMP LIN

DEFINE LIN     <
        LOCAL X1,X2,Y1,Y2,A,B
        IF(LT(ARG(0),4))GOTO NEXT
        X1=ARG(1),Y1=ARG(2),X2=ARG(3),Y2=ARG(4)
        GOTO START
NEXT:   TYPE!,"X1,Y1 =  ";READ X1,Y1
        TYPE"X2,Y2 =  ";READ X2,Y2
START:  $G(14,8);IF(EQ(X2-X1))TYPE!,"X =",X2," ONLY",!!;STOP
        A=(Y2-Y1)/(X2-X1),B=-A*X1+Y1
        TYPE!,"Y = ",A," *X + (",B," )",!!
>

*
```

Fig.12.1 Program list of "LIN"

```
*RUN LIN

X1,Y1 =  0,0
X2,Y2 =  100,100

Y =  1.0000000     *X + ( 0.0000000     )

*RUN LIN

X1,Y1 =  -64,-64
X2,Y2 =  128,128

Y =  1.0000000     *X + ( 0.0000000     )

*RUN LIN

X1,Y1 =  -64,64
X2,Y2 =  128,-128

Y = -1.0000000     *X + ( 0.0000000     )

*RUN LIN(-32,0,0,34\4\2)

Y =  1.0000000     *X + ( 32.000000     )

*RUN LIN

X1,Y1 =  171,81
X2,Y2 =  2654,1274

Y =  0.4804671     *X + (-1.1598872     )

*RUN LIN(171,81,2654,1274)

Y =  0.4804671     *X + (-1.1598872     )

*RUN LIN

X1,Y1 =  8,16
X2,Y2 =  8,-16

X = 8.0000000     ONLY

*
```

Fig.12.2 Applications of "LIN" program

バックグランドの計数値

BG＝（H－L＋1）＊（BL＋BH）／2

光電ピークの面積　P＝Q－BG

Pの統計誤差　D＝$Q^{1/2}$ 及びその相対値 100＊D／Q　（%）

を計算してリストアップする。特にQ，BG，P，Dについては，毎秒当りの値もリストアップする。

(2) プログラム・リストをFig. 12.3に，使用例をFig. 12.4に示す。本プログラムは，既に手計算で処理ずみのデータを改めて検算する目的で作製されたものであって，データはすべて対話型式で入力される。一旦入力がすめば，入力ずみのデータをそのままリストアップして

OK　？

とメッセージを出力する。"Y"又は"YES"以外のメッセージを入力すれば再び最初の"DATA　ID：　"のメッセージ出力の所へ戻る。このメッセージの後へは，データ識別のためのメッセージを最大20文字まで入力することができる。但し，これは用紙の上でデータを識別する目印となるだけで，プログラム内では全く使用されない。

(3) すべての計算結果が出力されれば

NEXT　？

とメッセージを出力する。"Y"又は"YES"と入力すれば，再び次の"DATA　ID：　"のメッセージ出力から繰返される。

12.3 DAYS

(1) 本プログラムは与えられた2つの日附（それぞれDate－a，Date－bと呼ぶ，順序は前後してもよい）の間の日数を計算し，指定によってはそれを年単位に換算するものである。本プログラムは別にその内部でYRAD，LY，CHKDATとの3種のプログラムを使用する。これらはあらかじめディスク上のファイルとして書き込まれていなければならない。これらのプログラム全部のリストをFig. 12.5に示す。

(2) 入力データの内，年号の数値は次のようにして西暦に換算される。（閏年の判定のために必要な処理である － (3)参照）今，この数値をyとして，

$y \leqq 60$ ならば昭和の年数とみなす。（1925を加える。）

$60 < y < 100$ ならば西暦の下2桁とみなす。（1900を加える。）

以上の処理を行うのがYRADプログラムである。

YRAD（y）

の如くArgumentで年数を入力すると，所定の計算結果がReturnとして得られる。

(3) LYプログラムは閏年（Leap year）の判定を行う。Argumentで西暦年数を入力すると，平年ならば0，閏年ならば1なる値をReturnする。閏年の条件はいうまでもなく，

a．西暦年数が4で整除される年

b．但し100で整除される年は400で整除される年に限り閏年。

である。従ってこの判定を厳密に行うには，与えられた年数を4，100及び400で割った余り

を求める演算が必要である。余りを求める演算を別の独立したプログラムとすればもう少しこのプログラムのリストは短くなる筈であるが，著者の場合そのようにすると，システム全体が停止するトラブルが発生したため，このような型に落着いた。

なお，年代を西暦1901（明治34）年から同2099年までに限るなら，単に4で整除される年を閏年とするのみで，また年数も1901～1999までは下2桁を使用すれば充分であることはいうまでもない。

(4) CHKDATプログラムは，暦法上あり得ない日附が誤って入力されたか否かをチェックする。即ち，CHKDAT（西暦年，月，日）なる型で3箇の数値を入力すれば，次の条件のいずれかに該当する（Illegal date）時は1，そうでない（Legal date）時は0なる値をReturnする。

a．（年）≦0

b．（月）＜1又は（月）＞12

c．（日）＜1

d．（月）＝4，6，9，又は11でかつ（日）＞30

e．（月）＝2でかつ（日）＞28（閏年の時は＞29）

f．（日）＞31

(5) Argumentなしで（又は5箇以下で）本プログラムを実行すれば，対話型式で日附の入力が要求されるから，その都度　年　月　日　の順で入力する。もしIllegal dateを入力すれば，その分について再入力が要求される。結果は日数及び年数（本プログラムでは1年＝365.2422日とする）でリストアップされる。

(6) Argumentを7箇与えれば，計算結果がReturn valueとして得られる。型式は次の通りである。

（型式）　RUN DAYS（年$_a$，月$_a$，日$_a$，年$_b$，月$_b$，日$_b$，Arg 7）

即ちDate－a，Date－bをそれぞれ年－月－日の順で入力する。（a，bは前後してもよい。）Arg 7を省略するか，或は0又は負の値を与えた時は日数単位で，正の値を与えた時はそれを年単位に換算した値が得られる。Illegal dateを入力した時には，常に0なる値が得られる。

(7) 使用例をFig．12.6に示す。本プログラムでは，初日が0日と計算される。これは，放射線源の調製から測定時までの日数（又は年数）を算出して，半減期との比を求める事を目的としたためである。最後の例は，2つの期間の比率を直接Retern value同志の除法で求め，結果を%単位で出力させたものである。

```
*LOAD AR;DUMP AR

DEFINE AR      <
        LOCAL A,B,C,D,BG,BH,BL,H,L,P,Q,T
HEAD:   TYPE!!,"DATA ID:  ";READS A,B,C,D
        TYPE!,"TOTAL(COUNTS) =  ";READ Q
        TYPE"LOW (CH' #, COUNTS):  ";READ L,BL
        TYPE"HIGH(CH' #, COUNTS):  ";READ H,BH
        TYPE"COLLECT TIME(SEC) =  ";READ T
        IF(LE(T))T=1
        TYPE!,"TOTAL =",$F(9,0)+Q," COUNTS"
        TYPE!,"LOW (CH' #, COUNTS):",L,BL
        TYPE!,"HIGH(CH' #, COUNTS):",H,BH
        TYPE!,"COLLECT TIME =",T
        TYPE!,"OK ?  ";READS A
        IF(NE(A,"YES")*NE(A,"Y"))GOTO HEAD

        $G(14,B)
        BG=(H-L+1)*(BL+BH)/2,D=SQRT(Q),P=Q-BG
        TYPE!!,"TOTAL       =",Q," /",Q/T," /SEC"
        TYPE!,"BACKGROUND =",BG," /",BG/T," /SEC"
        TYPE!,"AREA        =",P," /",P/T," /SEC"
        TYPE!,"(ERROR) = +-",D," /",D/T," /SEC"
        TYPE!,"            (",100*D/Q," % )"

        TYPE!!,"NEXT ?  ";READS A
        IF(EQ(A,"YES")+EQ(A,"Y"))GOTO HEAD
        STOP
>

*
```

Fig.12.3 Program list of "AR"

```
*RUN AR

DATA ID:  BA-133/81KEV

TOTAL(COUNTS) =  143426
LOW (CH' #, COUNTS):  165, 2481
HIGH(CH' #, COUNTS):  175, 1274
COLLECT TIME(SEC) =  1000

TOTAL =  143426. COUNTS
LOW (CH' #, COUNTS):      165.     2481.
HIGH(CH' #, COUNTS):      175.     1274.
COLLECT TIME =     1000.
OK ?  YES


TOTAL       = 143426.00      / 143.42600      /SEC
BACKGROUND  = 20652.500      / 20.652500      /SEC
AREA        = 122773.50      / 122.77350      /SEC
(ERROR) = +- 378.71625       / 0.3787162      /SEC
            ( 0.2640499      % )

NEXT ?  YES


DATA ID:  CO-57/122KEV

TOTAL(COUNTS) =  4496
LOW (CH' #, COUNTS):  251,294
HIGH(CH' #, COUNTS):  260,300
COLLECT TIME(SEC) =  2000

TOTAL =     4496. COUNTS
LOW (CH' #, COUNTS):      251.      294.
HIGH(CH' #, COUNTS):      260.      300.
COLLECT TIME =     2000.
OK ?  Y
```

Fig.12.4(1/2) Applications of "AR" program (1/2)

```
TOTAL      = 4496.0000     / 2.2480000     /SEC
BACKGROUND = 2970.0000     / 1.4850000     /SEC
AREA       = 1526.0000     / 0.7630000     /SEC
(ERROR) = +- 67.052218     / 0.3352610E-01 /SEC
          ( 1.4913750     % )

NEXT ?  Y


DATA ID:  MN-54/834KEV

TOTAL(COUNTS) =  5422
LOW (CH' #, COUNTS):  1730,13
HIGH(CH' #, COUNTS):  1744,5
COLLECT TIME(SEC) =  1000

TOTAL =    5422. COUNTS
LOW (CH' #, COUNTS):    1730.      13.
HIGH(CH' #, COUNTS):    1744.       5.
COLLECT TIME =    1000.
OK ?  Y


TOTAL      = 5422.0000     / 5.4220000     /SEC
BACKGROUND = 135.00000     / 0.1350000     /SEC
AREA       = 5287.0000     / 5.2870000     /SEC
(ERROR) = +- 73.634231     / 0.7363423E-01 /SEC
          ( 1.3580640     % )

NEXT ?  Y


DATA ID:  CO-60/1173KEV

TOTAL(COUNTS) =  37060
LOW (CH' #, COUNTS):  2433,104
HIGH(CH' #, COUNTS):  2450,60
COLLECT TIME(SEC) =  1000

TOTAL =   37060. COUNTS
LOW (CH' #, COUNTS):    2433.     104.
HIGH(CH' #, COUNTS):    2450.      60.
COLLECT TIME =    1000.
OK ?  Y


TOTAL      = 37060.000     / 37.060000     /SEC
BACKGROUND = 1476.0000     / 1.4760000     /SEC
AREA       = 35584.000     / 35.584000     /SEC
(ERROR) = +- 192.50974     / 0.1925097     /SEC
          ( 0.5194542     % )

NEXT ?  Y


DATA ID:  NA-22/1274KEV

TOTAL(COUNTS) =  24340
LOW (CH' #, COUNTS):  2644,19
HIGH(CH' #, COUNTS):  2662,8
COLLECT TIME(SEC) =  1000

TOTAL =   24340. COUNTS
LOW (CH' #, COUNTS):    2644.      19.
HIGH(CH' #, COUNTS):    2662.       8.
COLLECT TIME =    1000.
OK ?  Y


TOTAL      = 24340.000     / 24.340000     /SEC
BACKGROUND = 256.50000     / 0.2565000     /SEC
AREA       = 24083.500     / 24.083500     /SEC
(ERROR) = +- 156.01282     / 0.1560128     /SEC
          ( 0.6409729     % )

NEXT ?  N

*
```

Fig.12.4(2/2) Applications of "AR" program (2/2)

```
*LOAD DAYS,YRAD,LY,CHKDAT

*DUMP DAYS,YRAD,LY,CHKDAT

DEFINE DAYS   <
LOCAL MK(12),ML(12),D,SW,SWYD,I,J,K,IYA,YA,MA,DA,IYB,YB,MB,DB
DATA MK<31,28,31,30,31,30,31,31,30,31,30,31>
DATA ML<31,29,31,30,31,30,31,31,30,31,30,31>
IF(GT(DEF(CHKDAT)))DELETE CHKDAT
IF(GT(DEF(LY)))DELETE LY
IF(GT(DEF(YRAD)))DELETE YRAD
LOAD CHKDAT,LY,YRAD

D=0,SW=1;IF(LT(ARG(0),6))SW=0;GOTO START1
IYA=ARG(1),MA=ARG(2),DA=ARG(3),IYB=ARG(4),MB=ARG(5),DB=ARG(6),SWYD=0
IF(GT(ARG(0),6))SWYD=ARG(7)
YA=YRAD(IYA),YB=YRAD(IYB),I=CHKDAT(YA,MA,DA),J=CHKDAT(YB,MB,DB)
IF(NE(I)+NE(J))GOTO FIN
GOTO START3

START1:TYPE !,"DATE(A) - Y,M,D ?  ";READ IYA,MA,DA
YA=YRAD(IYA);IF(EQ(CHKDAT(YA,MA,DA)))GOTO START2
TYPE "  * ILLEGAL INPUT DATE(A) *";GOTO START1
START2:TYPE !,"DATE(B) - Y,M,D ?  ";READ IYB,MB,DB
YB=YRAD(IYB);IF(EQ(CHKDAT(YB,MB,DB)))GOTO START3
TYPE "  *ILLEGAL INPUT DATE(B) *";GOTO START2
START3:IF(LT(YA,YB))GOTO START4
IF(EQ(YA,YB)*LT(MA,MB))GOTO START4
IF(EQ(YA,YB)*EQ(MA,MB)*LE(DA,DB))GOTO START4
J=YA,YA=YB,YB=J
J=MA,MA=MB,MB=J
J=DA,DA=DB,DB=J

START4:D=D+DB,K=YB-1;IF(LT(K,YA))K=MB-1;GOTO FIN1
IF(LT(MB,2))GOTO MID1
I=1,J=MB-1;IF(GT(LY(YB)))GOTO START5
DO(J)<D=D+MK(I),I=I+1>;GOTO MID1
START5:DO(J)<D=D+ML(I),I=I+1>

MID1:IF(EQ(K,YA))GOTO FIN0
I=YA+1,J=K-I+1;DO(J)<D=D+365+LY(I),I=I+1>

FIN0:K=12
FIN1:IF(LT(K,MA))GOTO FIN3
J=K-MA+1,I=MA;IF(GT(LY(YA)))GOTO FIN2
DO(J)<D=D+MK(I),I=I+1>;GOTO FIN3
FIN2:DO(J)<D=D+ML(I),I=I+1>
FIN3:D=D-DA
FIN:DELETE CHKDAT,LY,YRAD
IF(GT(SW))GOTO END
$I(5);TYPE !," ** START",YA,MA,DA," - FINISH",YB,MB,DB
TYPE " : ",D," DAYS, "
TYPE $G(12,5)+D/365.2422," YEARS **",!,!;STOP
END:IF(GT(SWYD))D=D/365.2422
D>
```

Fig.12.5(1/2) Program list of "DAYS" — main program

```
DEFINE YRAD   <LOCAL Y
Y=ARG(1);IF(LT(Y))GOTO F
IF(LE(Y,60))Y=Y+1925;GOTO F
IF(GT(Y,60)*LT(Y,100))Y=Y+1900
F:Y>

DEFINE LY     <LOCAL X,Y,A,B
X=0,Y=IP(ARG(1)),A=Y-4*IP(Y/4)
IF(NE(A))GOTO F
A=Y-100*IP(Y/100),B=Y-400*IP(Y/400)
IF(EQ(A)*NE(B))GOTO F
X=1
F:X>

DEFINE CHKDAT <LOCAL X,Y,M,D
X=0,Y=ARG(1),M=ARG(2),D=ARG(3)
IF(LE(Y))GOTO E
IF(LT(M,1)+GT(M,12))GOTO E
IF(LT(D,1))GOTO E
IF(EQ(M,11)*GT(D,30))GOTO E
IF(EQ(M, 9)*GT(D,30))GOTO E
IF(EQ(M, 6)*GT(D,30))GOTO E
IF(EQ(M, 4)*GT(D,30))GOTO E
IF(EQ(M, 2)*GT(D,28+LY(Y)))GOTO E
IF(GT(D,31))GOTO E
GOTO F
E:X=1
F:X>

*
```

Fig.12.5(2/2) Program list of "DAYS" — sub-programs

```
*RUN DAYS

DATE(A) - Y,M,D ?  67,2,29
  * ILLEGAL INPUT DATE(A) *
DATE(A) - Y,M,D ?  67,2,28

DATE(B) - Y,M,D ?  67,2,28

 ** START 1967    2   28 - FINISH 1967    2   28 :     0 DAYS,   0.0000     YEARS **

*RUN DAYS

DATE(A) - Y,M,D ?  68,2,1

DATE(B) - Y,M,D ?  68,2,29

 ** START 1968    2    1 - FINISH 1968    2   29 :    28 DAYS,   0.7666E-01 YEARS **

*RUN DAYS

DATE(A) - Y,M,D ?  68,1,1

DATE(B) - Y,M,D ?  68,12,31

 ** START 1968    1    1 - FINISH 1968   12   31 :   365 DAYS,   0.9993     YEARS **

*RUN DAYS

DATE(A) - Y,M,D ?  69,1,1

DATE(B) - Y,M,D ?  69,12,31

 ** START 1969    1    1 - FINISH 1969   12   31 :   364 DAYS,   0.9966     YEARS **

*RUN DAYS

DATE(A) - Y,M,D ?  31,6,15

DATE(B) - Y,M,D ?  53,6,19

 ** START 1956    6   15 - FINISH 1978    6   19 :  8039 DAYS,   22.010     YEARS **

*LOAD DAYS

**G(12,5)

*TYPE DAYS(31,6,15,53,6,19,0)
  8039.0
*TYPE DAYS(31,6,15,53,6,19,1)
  22.010
*TYPE DAYS(1968,6,15,1978,6,19)
  3656.0
*TYPE 100*DAYS(1968,6,15,1978,6,19)/DAYS(1956,6,15,1978,6,19),"%"
  45.478    %
*
```

Fig.12.6 Applications of "DAYS" program

13. エラーメッセージ

13.1 Program Nesting Level

既に述べた通り，CLASS語によるプログラムは，全体を"＜　＞"でかこまれており，その内部では更にDO－loop部分で"＜　＞"にかこまれた部分が独立に，或は組合わされて存在する。最も外側の"＜　＞"の内部のNesting levelを1とし，その内側の"＜　＞"内部のNesting levelを2……と順次定義する。即ち次の例に示す通りとなる。("　"内の数字がNesting level)

(例)　……"0"……DEFINE SAMPLE＜……"1"……DO (i)＜……"2"……
……DO (j)＜……"3"……　……＞……"2"……＞……"1"……＞……"0"……

13.2 エラーメッセージの型式

CLASSに於ては，

a. Keyboard又はその他のI／Oより入力した命令又はDataの構成がCLASSの規則に適合しなかった時。

b. 実行中のプログラムに，CLASSの規則に適合しない構成の箇所があって，プログラムの実行がその箇所に及んだ時。(CLASSの Software は Interpreter であるから，プログラムは逐次機械語に翻訳されながら実行される)

c. プログラムの実行中にエラーが発生した時。(数値のオーバーフロー等)

直ちにその箇所で命令又はプログラムの実行を打切り，次の順序でメッセージを出力して，次の命令の入力待ちの状態に戻る。

(1) 上記いずれかの状態が発生した直後に
ERROR　{Error No}:{Nesting level No}:{Errorが発生した箇所のSymbol}
なる1行を印字する。

(2) 次の行にErrorを発生した命令入力の内容，又は実行中のプログラムのErrorを発生した1行を印字する。

(3) 次の行に，(2)で印字した行中の，Errorを発生したSymbolの最後の文字の真下の所へ"↑"と印字する。

(4) 次の行に
{Error No}{Error Noに対応するMessage}
なる内容を印字する。

(5) 次の行の先頭に"＊"を印字して，次の命令待ちの状態となる。

(6) Error No及びそれに対応するMessageはTable. 13.1の通りである。

Table 13.1 (1/2) CLASS V04.13-RT/X-A Error message list

1 ILLEGAL SYNTAX
2 ILLEGAL CHARACTER
3 ILLEGAL NAME
4 ILLEGAL EXPRESSION OR UNDEFINED NAME
5 ILLEGAL FORM
6 SYMBOL AREA OVERFLOW ON CREATE
7 NAME ALREADY IN USE
8 ILLEGAL TYPE
9 WRONG ARGUMENT COUNT
10 OVERFLOW OF SYMBOL AREA
11 ARGUMENT LIST ERROR
12 UNDEFINED NAME OR ILLEGAL TYPE
13 "<>" ERROR
14 ILLEGAL REPEAT EXPRESSION - DO
15 NOT A USER FUNCTION NAME
16 ILLEGAL FIELD SIZE SPECIFICATION
17 ARRAY DIMENSION NOT SUFFICIENT FOR OPERATION
18 GC BLOCK ERROR - SEE USER MANUAL
19 ILLEGAL MODE
20 ILLEGAL FILE SPECIFICATION LINE
21 ILLEGAL FILE SWITCH
22 ILLEGAL DEVICE NUMBER
23 NOT ENOUGH ROOM FOR DRIVER TO LOAD
24 ILLEGAL INPUT EXPRESSION
25 ILLEGAL STRING ARGUMENT
26 FIELD WIDTH INSUFFICIENT
27 UNDEFINED NAME
28 ILLEGAL ARGUMENT
29 OVERFLOW OF SYMBOL AREA DURING EDIT
30 ILLEGAL LIST TERMINATOR
31 ILLEGAL FORMAT SPECIFICATION FIELD
32 ILLEGAL CHARACTER IN ARGUMENT LIST
33 ILLEGAL 'ARG' FUNCTION CALL
34 ILLEGAL DIMENSION IN ARRAY CALL
35 DEVICE ERROR
36 ILLEGAL DIMENSIONED VARIABLE NAME
37 ERROR IN DIMENSION STATEMENT
38 ILLEGAL NAME OR RECURSIVE CALL - DATA
39 UNDEFINED OR ILLEGAL DATA TYPE - DATA
40 ILLEGAL CHARACTER OR ILLEGAL EXPRESSION - DATA
41 NESTED STRING FUCTION CALL
42 LABEL NOT FOUND - GOTO
43 ILLEGAL STRING DECLARATION
44 ILLEGAL SYNTAX - DEFINE
45 ILLEGAL 6911 INTERRUPT
46 ILLEGAL COMMAND - 6911
47 CAN'T EVALUATE ARGUMENT OR $EVAL
48 ILLEGAL TERMINATION FOR $EVAL CALL
49 ERROR IN 'WAIT' OR 'SENSE' ARGUMENT
50 DEVICE CHANNEL NOT AVAILABLE
51 ILLEGAL DATA - LOCATE
52 TOO MANY 6726 CHARACTERS ON READOUT
53 TOO MANY 6911 CHARACTERS ON READOUT
54 COMMAND ISSUED TO 6911 ON MANUAL
55 DEVICE CANNOT DO RANDOM ACCESS I/O
56 ANALYZER NOT IN 'STOP'

(Continued)

Table 13.1 (2/2)

(Continue)

57 ILLEGAL MCA ADDRESS
58 START CHANNEL IS GREATER THAN STOP CHANNEL
59 'RESULT' ARRAY IS OF INSUFFICIENT SIZE
60 INPUT CONVERSION ERROR
61 IMPROPERLY DIMENSIONED WORKING ARRAY
62 INDEX TOO LARGE
63 ILLEGAL FORM IN INDIRECT FILE SPECIFICATION
64 UNDEFINED NAME IN INDIRECT FILE SPECIFICATION
65 ILLEGAL NAME IN INDIRECT FILE SPECIFICATION
66 ILLEGAL DATA FORM IN INDIRECT FILE SPECIFICATION
67 INSUFFICIENT CORE AVAILABLE FOR INVERSION
68 TOO MANY PEAKS TO RESOLVE
69 ILLEGAL 1788 DEVICE NUMBER
70 ILLEGAL CONTROL CHARACTER - 1788
71 ILLEGAL CHARACTER RESEIVED - 8100
74 NAME IN USE - LOAD
75 FILE NOT FOUND - LOAD
76 OUT OF CORE - LOAD
77 NO DRIVER ON SYSTEM FILE
78 HANDLER NOT RESIDENT
80 ILLEGAL USE OF STRING IN ASSIGNMENT
81 ILLEGAL USE OF RESERVED WORD
82 UNDEFINED INDIRECT NAME
83 ILLEGAL INDIRECT NAME - NOT A VARIBLE
84 ILLEGAL CHARACTER IN OCTAL SUB-FIELD
85 'SAVE' ERROR
86 FILE ALREADY 'SAVED'. USE 'REPLACE' OR 'RENAME
87 ILLEGAL FILE SPECIFICATION LINE
88 CAN'T DO INPUT ON CHANNEL SPECIFIED
89 DEVICE CHANNEL NOT OPEN
90 FILE NOT FOUND
91 I/O LOOKUP OR ENTER FAILURE
92 FILE INPUT ERROR
93 FILE OUTPUT ERROR
94 CAN'T DO OUTPUT ON CHANNEL SPECIFIED
95 FILE ALREADY OPEN - OPENR
96 FILE ALREADY OPEN - OPENW
97 FILE NOT FOUND - OPENR
98 RECORD NUMBER TOO LARGE
99 SYSTEM ERROR - SEE USER MANUAL
100 ILLEGAL FORM OF USER CALL - MISSING ARGUMENT LIST
101 ILLEGAL ARGUMENT IN USER CALL LIST
125 EXPONENT OVERFLOW IN ADDITION
126 EXPONENT OVERFLOW IN ADDITION
127 DIVIDE BY ZERO
128 EXPONENT OVERFLOW IN DIVISION
130 EXPONENT OVERFLOW IN DIVISION
132 DIVIDE BY ZERO
134 EXPONENT OVERFLOW IN MULTIPLICATION
135 EXPONENT OVERFLOW DURING NEGATION
136 EXPONENT OVERFLOW IN MULTIPLICATION
140 "EXP" ARGUMENT GREATER THAN 88
141 "LOG" ARGUMENT LESS THAN OR EQUAL TO ZERO
142 "SQRT" ARGUMENT LESS THAN ZERO
143 "EXP" ARGUMENT GRATER THAN 88
146 INTEGER OUTSIDE RANGE ON REAL TO INTEGER CONVERSION
149 "SQRT" ARGUMENT LESS THAN ZERO

14. 結　　言

実際のガンマ線スペクトル解析用に，CLASS語で書かれた一連の基礎的なプログラムが"SPECTRAN III"なる名称で供給されていることは，4.12で既に述べた。

また，CLASSにDigital plotterを制御する命令を附加した"CLASSP"や，これによりDataのPlotを行うCLASS語のプログラムを東陽通商（株）に発註してあったものが，1979年度より正式に使用可能となった。この中には，ガンマ線スペクトルのデータを自動的に収集・解析するプログラムも含まれている。但しSPECTRAN IIIやこれらのプログラムのみでは，必ずしも実際の測定時の要求をすべては満たさないので，所要のプログラムの新規作製又は改造を，著者らは目下行っている。

SPECTRAN III，CLASSP等を始めとして，これらのプログラムについては，別の機会に改めて説明する予定である。

References

1) Canberra Industries, Inc. : RT－11 CLASS Manual, CI－CE－402 (Sept. 1976).
2) Digital Equipment Corporation : RT－11 System Reference Manual (Reviced Jan. 1976).
3) Reference [2], Chapter 1 & 2.
4) 東陽通商(株) : SPECTRAN III User's Guide (June 1976), p. 14－17 (和文).
5) Canberra Industries Inc. : QUANTA COMMANDS.

14. 結　　言

実際のガンマ線スペクトル解析用に，CLASS語で書かれた一連の基礎的なプログラムが"SPECTRAN III"なる名称で供給されていることは，4.12で既に述べた。

また，CLASSにDigital plotterを制御する命令を附加した"CLASSP"や，これによりDataのPlotを行うCLASS語のプログラムを東陽通商（株）に発註してあったものが，1979年度より正式に使用可能となった。この中には，ガンマ線スペクトルのデータを自動的に収集・解析するプログラムも含まれている。但しSPECTRAN IIIやこれらのプログラムのみでは，必ずしも実際の測定時の要求をすべては満たさないので，所要のプログラムの新規作製又は改造を，著者らは目下行っている。

SPECTRAN III，CLASSP等を始めとして，これらのプログラムについては，別の機会に改めて説明する予定である。

References

1) Canberra Industries, Inc. : RT－11 CLASS Manual, CI－CE－402 (Sept. 1976).
2) Digital Equipment Corporation : RT－11 System Reference Manual (Reviced Jan. 1976).
3) Reference [2], Chapter 1 & 2.
4) 東陽通商(株) : SPECTRAN III User's Guide (June 1976), p. 14－17 (和文).
5) Canberra Industries Inc. : QUANTA COMMANDS.

[JAERI-M 8694 正誤表]

(頁)	(行)	(誤)	(正)
目次2.	上から 6.	解析	解析
目次3.	下から 6.	OUIT命令	QUIT命令
目次6.	下から 11.	virtural	virtual
14.	上から 8.	なくしても	なくても
〃	上から 10.	相呉間	相互間
15.	上から 13.	*DO(3)<---	*DO(3)<---
30.	上から 1.	I/Oの3分類	I/O Control
33.	下から 15.	3字文	3文字
37.	下から 9.	IF(NE(ANS,"Y")+---	IF(NE(ANS,"Y")*---
40.	下から 10.	DATA1・CDF	DATA1.CDF
44.	(Table 7.1 のNoteに追加)		,bDelete(抹消)
46.	下から 14.	(S1,"REI,"PD)	(S1,"REI","PD")
59.	上から 8.	文次等	文字等
〃	下から 7.	Intensfied	Intensified

(以上)

山田整印刷所納