解　説

ミニコンによるコンピュータ教育*

——*NHK* 教育テレビの場合——

筧　捷　彦**

0. はじめに

昭和 48 年度および 49 年度にかけて，*NHK* 教育テレビで行った「コンピューター講座」について，担当者の1人として，解説する．

コンピューター講座は，昭和 44 年度に開設され，以後毎年継続して放映されている．48, 49 両年度については，島内剛一（立教大学）を主任講師とし，石田晴久（東京大学），筧捷彦(東京大学，後に立教大学)，木村泉（東京工業大学），広瀬健（早稲田大学），米田信夫（学習院大学）の合計6人が，講座の立案企画から，各番組での講師役までを担当した．

47 年度以前の講座との大きな違いの1つに，スタジオ内にミニコンを持ち込み，これの活用を図ったことがあげられる．解説の中心も，このミニコンの活用状況におくことにする．

1. 講座のねらい

講座の対象としたのは，主として計算機に関しての初心者である．この場合「初心者」の意味は，これからプログラミングを習得しようとしている人とか，プログラミングを習得をせざるを得ない人といった狭いものではなく，普通一般の社会人——おそらくはプログラミングする機会も必要もない人——をも含めた広いものである．

講座は，毎週1回1時間分が放映され，26 回（半年）で終了する．テレビ番組としては，この分を残りの半年間再放送する，というものである．また，*NHK* 教育テレビの講座物の慣例として，テキストを作成することになっている．

この与えられた枠の中で講座を構成するにあたって優先させたのは，テレビ放送としての特徴を生かすことであった．

ラジオでの講座とは違い，視聴者は目，耳ともに画面にとられてしまう．したがって，テキストの参照を番組中で多用することは不可能であり，テキストは，特に熱心な受講者の自習用として考えておくことにした．

テレビ画面上での分解能は意外に低く，このことから文字で情報を伝えるにはあまり適してはいない．したがってプログラムも短かく小さく書かれたものを取り上げるに留めねばならない．放送という情報伝達は完全に一方通行のものである．一方，プログラミングの習得には必ず修練が必要であり，教師との会話もそのための重要な役割を担う．視聴者のどれ程が，修練するための場——計算機を利用できる状況——を持っているかは疑わしい．

こうしたことから，「プログラミング教育」をテレビ番組で行なうのは一般的にいって不可能とさえいえる．こうして 48, 49 両年の講座の力点は「コンピュータ教育」に置くこととした．つまり，計算機についての正当なイメージを視聴者に与えることである．このため，それぞれの放映の中では，実際に計算機が応用されている様をフィルムで見せるとか，そうした場面で活躍中の方々に話をしていただくとかを行なった．また，それぞれの放映ごとに読切りの形として，たまたま何の準備もなくチャネルをまわした人にもわかりやすいものとするように企画された．

毎回の放映ごとにテーマを定め，比較的簡単なプログラムを用意し，主としてどういう考え方でプログラムが作られているかを説明した．さらに，すぐその場で計算機にかけてみることとした．この場合，(特定の言語の) 文法の細目をくだくだと述べることはやめた．また，テーマもなるべく実際の応用のされ方が想

---

* Computer Education with a Mini-Computer ——a TV Program of *NHK*——
by Katsuhiko KAKEHI (Rikkyo University, Faculty of Science, Department of Mathematics).

** 立教大学理学部数学科

像しやすいものを選び，スタジオ内でも比較的多量のデータを処理してみせられるよう工夫が行なわれた．実際のところ，文法その他を守りやっとの思いでプログラムが完成した後で，計算機がやってくれることがわずか1枚のカードを読み取り，わずかばかりの計算結果を印刷してくれるのでは，初心者を落胆させるだけの効果しかない．

中には熱心な受講者もいるであろうから，講座全体を通じては，ひととおりの範囲をおおいつくすよう，そして，自習のための引き金となりうるように配慮が行なわれた．

両年度の各放映のテーマと，それぞれ1時間分の番組構成の典型とを図 1,2 に示す．

2. ハードウェアの構成

講座のねらいから，計算機が動作している様子をなるべく生々しく放映することが考えられた．それにはスタジオ内に計算機を持ち込み，いろいろと使ってみることにこしたことはない．

普通の計算機システムに近いものが望ましいことは明らかであるが，スタジオを計算機室にしてしまうこ

昭和 48 年度　ミニコンによる基礎コース

1	計算機と遊ぶ—同時に4人で—		はじめ
2	御破算で願いましては—計算機の命令—	ハードウェアとアセンブラ語	基礎
3	手直し自由—エディター—		
4	背くらべ—ソーティング—		
5	コンピュータの素顔—2進法—		
6	ベルを鳴らそう—タイミング—		
7	ガソリンは満タン—FORTRAN 語—	FORTRAN によるプログラミング	
8	2次方程式の根—実数と基本関数—		
9	数表をつくる—DO ループ—		
10	王様さがし—配列—		
11	ユークリッドの互除法—副プログラム—		
12	算数の宿題—文字型データ—		
13	バケツの中身—コンパイラ—		なか
14	万年七曜表—法による計算—	やさしい応用	応用
15	サンフランシスコの電車切符—判定とスイッチ—		
16	11213 は素数か—素因数分解—		
17	AからZまで—文字の大小—	ソート・サーチ・マージ	
18	名簿の改訂—マージ—		
19	大圏コース—表を引く—		
20	カメノコ幾何—カンバスとしての配列—	図形	
21	関数を見よう—図形表示—		
22	プラスとマイナスの間—根を探す—		
23	三山崩し—n 進法の変換—	非算術応用	
24	電子サイコロ—乱数とその応用—		
25	ヨッパライの生残時間—モンテカルロ法—		
26	和算家から大型コンピュータへ—高精度計算—		おわり

図-1.1　講座の構成

昭和 49 年度　プログラミングの基礎と応用

1	計算機と仲良く—タイプライタでの会話—		はじめ
2	マジック・ナンバー—エディター—	FOL によるプログラミング	基礎
3	在庫管理—データの集計—		
4	ボーリングのスコアー—アルゴリズム—		
5	邪魔者は消せ—連立1次方程式—		
6	気になる順位—整列—	やさしいアルゴリズム	
7	百人一首—文字処理—		
8	落書きタイプライター—図形処理—		
9	デタラメのつもり—乱数—		
10	当選確実—予想—	統計のアルゴリズム	
11	白か黒か—検定—		
12	待ったなし—実時間処理—		
13	電子聖徳太子—割込み—		なか
14	信号待ち3回—待ち行列—	いろいろなアルゴリズム	応用
15	音を料理する—AD, DA 変換器—		
16	五劫のすりきれ—高精度計算—		
17	山あり谷あり—多項式—	数値的アルゴリズム	
18	宇宙船のゆくえ—常微分方程式—		
19	ギターの音色—偏微分方程式—		
20	ペントミノー人工知能—	プログラミング上の道具	
21	単語帳—表の管理—		
22	石・紙・はさみ—データ構造—		
23	点と線—最短距離—	すすんだ応用	
24	さめたコーヒー—数理計画法—		
25	電卓をまねる—シミュレーション—		
26	素数—プログラム作法—		おわり

図-1.2　講座の構成

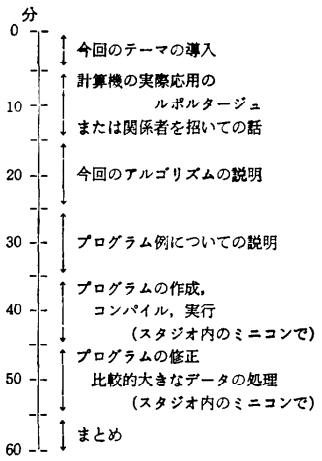


図-2　1回放映分の構成

とは不可能であるし，さらに何にもましてそれだけの予算がなかった．比較的容易に設置でき，価格的にも可能なものとしては，ミニコンのシステムか，大型計算機システムの端末装置が考えられた．

端末装置を通して時分割システムを利用することは前にも試みられていた．ただこの場合，大きなファイルを予約しておくだけの予算がなく，結局はプログラ

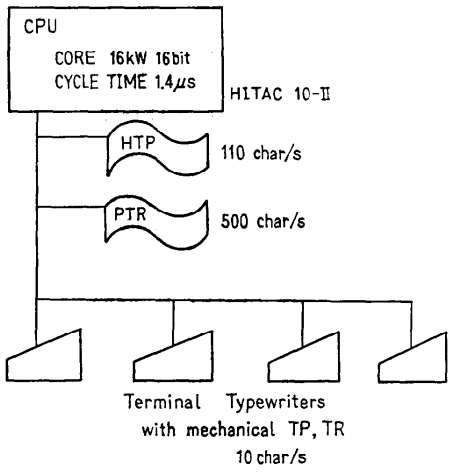


図-3 ハードウェアの構成

ム，データなどは紙テープの形で保存しておかねばならなかった．一方，タイプライタ端末装置だけでは，読込み速度が低く，多量のデータを処理してみせることもままならなかった．また何にもまして，計算機システムそのものを放映できない，という欠点があった．

これらのことから，ミニコンのシステムを利用してみることとなった．ハードウェアの構成は図-3 に示す．CPU の記憶容量は，49 年度には 24 kw に拡張された．

2次記憶装置が付けられていない点では，ミニコンのシステムとはいえ，現状にそぐわないかもしれない．ソフトウェアについては，すべて自前で作製することにした．2次記憶装置をつけなかったのは，1つには予算の問題があったからであり，もう1つには，信頼性のある2次記憶利用のソフトウェア・システムが比較的短期間の内に開発できるかどうかわからなかったからである．

実際のところ，ソフトウェア・システムの開発にあてられた期間は，ほぼ1月～3月の3ヵ月位であり，しかも，ハードウェアが搬入されたのは3月もなかばを過ぎた頃であった．

こうして，すべてが紙テープを基本とするシステム構成となってしまったが，逆に，多量のデータが読み取られていく様は「生々しい」ものとなった．

3. エディタ，モニタ

プログラムやデータの作成，修正，実行等は，視聴者から希望者を募り，行ってもらうことにした．このため，4人位で同時に使える形のシステム構成とすることにした．

紙テープを基本とするシステムでは，エディタは必須のものである．こうしたオンラインのエディタについてはこれまでに多くの報告がある．それぞれの利点欠点を考慮の上で，初心者にもすぐ使える，ということに重点を置いた新しいエディタを作ることにした．

新しいエディタの基本方針は次のとおりである．

（1） 1文字単位の編集ができるものにする．

（2） 同じキーの打鍵は，いつも同じ動作をひきおこす．

（1）については説明はいらないと思う．絶対的な行番号がふられ，その番号で行単位の編集をする形のものが，オンライン用としては落第であることは使ったことがある人には自明であろう．（2）は，いわゆるモードを設けない，と言いかえてもよい．エディタのモードによって，打鍵の意味が変わるのは，少なくとも初心者にとっては，慣れにくいものである．

こうして，新しいエディタでは，キーごとに固定した意味が与えられた．図形文字――その文字を挿入する，Lf――行がえ，RUBOUT――1字抹消，その他の制御用文字――コマンド，というふうである．

タイプライタの印刷部では，いつも現在着目している文字を含む行が，その文字まで印字されて，プリント・ヘッドがそこで停止しているように設計されている．各種のコマンドで，着目している文字位置が変われば，ただちに，上の条件を満たすように印刷が行なわれる．位置が左へ移動したときは，タイプライタの特性上，印刷された文字を消すことができないので，もう1度行の左端から印刷しなおされる．

印刷は，コマンドに対するエディタの反応結果を表示するだけである．印刷中に次のコマンドを打鍵することも可能で，この場合にもエディタは直ちに対応する内部操作を行ない，進行中の印刷は中止して，新たな結果を印刷しなおす．こうして，たて続けに行がえのコマンドを与えたとすると，印刷は最終行のみにすることも可能である．

コマンド等については，その一覧を図-4（次頁参照）に示す．図中 ^T 等とあるのは，CNTL のキーとTのキーとを押すことで作られる制御用文字を意味する．

^B のコマンドは，1度に大量のファイル削除を行なうコマンドを誤って使ったとき，もとへ戻すことを可能にするために必要なものである．不当なコマンド

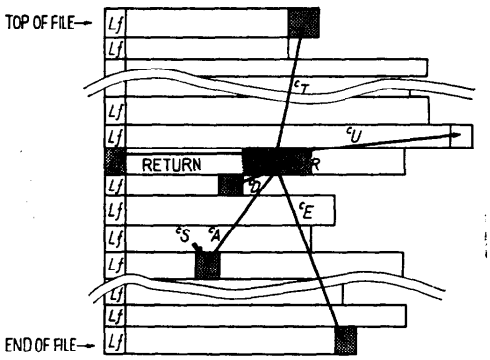


図-4.1 エディタのコマンド

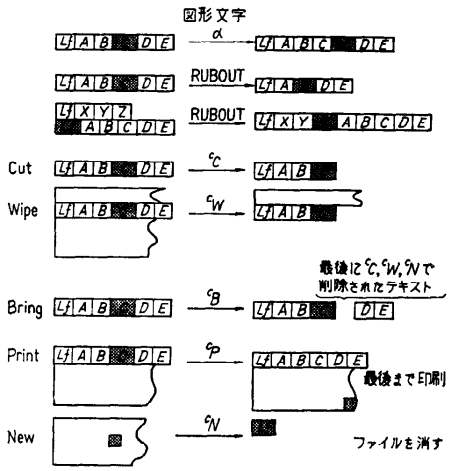


図-4.2 エディタのコマンド

については，ベルをならすだけで，ファイルについては何らの変更が加わらないようにしてある．

このシステムでは，タイプライタの打鍵と，付属のMTR（機械式テープ読取り装置）からの入力とは区別されない．こうして1度編集された紙テープはMTRにかければ，ファイルにそのまま読み込まれる．

49 年度には，次のコマンドが追加され，大きなファイルの編集の便が図られた．^F——最上行の内容と左端が一致する行を見つける，^I——現在の文字位置の行数を最上行に入れる，^O——最上行が示す数だけ行をさげる，^V——^B と同様，ただし複写して挿入する．

鍵打	メジッセー
A	&& JOB ← ASSEMBLER
C	&& JOB ← COMPLER
L	&& JOB ← LOADER
I	&& JOB ← FILE-IN
O	&& JOB ← FILE-OUT
1	&& INPUT ← PTR
3	&& INPUT ← PRINTER
5	&& INPUT ← XXXX
7	&& INPUT ← FILE
2	&& OUTPUT ← HTP
4	&& OUTPUT ← KB
6	&& OUTPUT ← XXXX
8	&& OUTPUT ← FILE
Lf	&& START JOB
Space	&& RESTART JOB
ESC	&& INTERRUPT JOB
(終了)	&& FINISH JOB
(入力不足)	&& PAUSE JOB
RETURN	&& EDITOR

図-5 モニタのコマンド

各種の言語プロセッサ等を使う場合の，いろいろな指定を行なうものとして，モニタを作った．モニタへ制御が移るのは，ESC のキーを打鍵したとき，プロセッサが動作終了したとき，入力データが不足したときである．

モニタに制御があるときは，行の左端に && が印字され，コマンド待ちとなっている．モニタのコマンドもまた，1字だけの打鍵で与える．

モニタのコマンドの一覧は図-5 に示す．数字は入出力機器の指定である．システム内では，各機器は1ケタの番号で引用される．5と6は標準入出力装置ともいえるもので，ふつうのプロセッサはこれを用いる．モニタのコマンドで，この5，6に特定の機器を割り当てる．

不当な文字の打鍵に対しては REJECTED とメッセージを出すようにしてある．

プロセッサは他にもいくつか用意されている．FILE-IN, -OUT は，ファイルへの読込み，ファイルの書出しを司る．

4. プログラミング言語 FOL[1]

プログラミング——とくに，特定の言語でプログラムが書けるようにすること——は，講座の直接の目的でないことは前にも述べた．それよりも，文法の，しかも多くは処理系の都合に焦点を合わせたような，つまらない制約にとらわれないで，計算機を動かすプログラムというものを見せることの方が大切である．

テレビ番組であることから，プログラムが大きくなりすぎては画面上に収めることが難しくなる．この点からすると COBOL はテレビ用としては不適当である．多量のデータの処理，およびその結果のきれいな印刷が楽にできることも必要で，BASIC 等はいまひとつ欠けたものといえる．

48 年度には，処理系作成の時間的予裕が無かったので，以前に作られていた ELF を用いた．ELF は，ほぼ JIS 3000 レベルの FORTRAN 処理系で，文字型データの取扱いもできるものである．

48 年度講座の反省から，新たな言語およびその処理系を作成することとなった．ELF 程度の言語仕様を基本として，次のような点に注意が払われた．

（1） ALGOL の長所を取り入れる——名前の長さを自由にする，ブロック構造を取り入れる，**if then else** や複合文といった構造を取り入れる．

（2） 文字データの取扱いを便利にする——FORTRAN や ALGOL のようなまま子扱いはやめ，1人前に扱う．

こうして，FORTRAN と ALGOL の長所を組合せた形で言語 FOL が作られた．3000 レベル FORTRAN あるいは 4030 レベルの ALGOL で書かれたプログラムなら，手直しせずに FOL のプログラムとして受けつける．ただし，両言語の合併集合を作るのではなく，同様の機能は融合させてある．

FOL の特徴を列挙すれば次のようになる．

（1） データ型は，実数型，整数型，名札型とする．整数型のデータとしては，整数，論理値，文字列を考える．ふつうは，あたかもこの3者は別の型であるかのようにプログラムを作ることができる．名札型の導入で，計算型 GO TO 文やスイッチの機能を容易に表現できるようにした．

（2） 配列は，コンパイル時にその大きさを定める．配列要素名の添字式は，任意の式でよい．添字として0も許す．

（3） 名前の長さは任意の英数字の列である．暗黙の型宣言を認め，I～N で始まるものは整数型その他の英字で始まるものは実数型，数字で始まるものは名札型となる．

（4） ブロック構造は柔軟に考えられ，FORTRAN でのプログラム単位の列としてのプログラムも受けつける．

（5） 入出力は，ALGOL 風の簡易な手続き呼出しによるほか，FORTRAN 風の書式制御による入出力文がある．

（6） 実数型と整数型とが混在した式も受けつける．一般に多くの場所で任意の形の式を受けつける．

（7） ALGOL 風に ; で文を区切ることもできるし，行の切れ目で文を区切ってもよい．また，THEN, ELSE のような記号の前後で行がえをしてもよい．特に明記して行がえをするには，前の行の最後に — をおく．

（8） @ 以後その行の最後までは注釈である．

（9） 文字データは，1変数に4つはいる．これの取扱いのため，ENCODE(*s*, *i*, *n*)——文字列 *s* の *i* 番目の文字をコードとして取り出し *n* に入れる——等の手続きが豊富に用意してある．

図-6 として FOL によるプログラム例を示す．FOL の文法の詳細はここでは触れないが，図のプログラムはたやすく理解できると思う．FOL の特徴を読みとっていただきたい．

5. ソフトウェアの構成[2)]

初心者がプログラミングの実習をする場合，その大部分の時間はプログラムやデータの編集作業にあてられる．このことから，少なくともエディタは，複数の

```
* *** HYAKUNIN ISSYU  ***
* * COMPUTER KOZA STYLE *
* COMMON AREA
    STRING KASYU(100,0:11)
    INTEGER MIDAS:(100)
* SUBPROGRAM
*
    SUBROUTINE SAGASU(L,M,K)
      N=0
      K1=K/4; K2=K-K1*4
      DO 1 FOR I UNTIL M
        ENCODE(KASYU(MIDASI(I),K1),K2,J)
      1 IF L=J THEN
          BEGIN N=N+1; MIDASI(N)=MIDASI(I) END
      M=N
    END
*
* MAIN PROGRAM
    READ(5,F1) (KASYU(I,J) FOR J=0,17) -
               FOR I=1,100
     F1:FORMAT(18A4)
    FOR I UNTIL 100 DO MIDASI(I)=I
    K=0; M=100
    WRITE(4,F2); F2:FORMAT(//'?',A1)
 KURIKAESI: INCODE(3,L)
    OUTCODE(4,L)
    CALL SAGASU(L,M,K); K=K+1
    IF M>1 GO TO KURIKAESI
    IF M=0 WRITE(4,F3)
    IF M=1 THEN
      WRITE(4,F4) KASYU(MIDASI(1),J) FOR J=0,17
  F3:FORMAT(/'AKIMASEN!!')
  F4:FORMAT(/18A4)
 END
*
```

図-6 FOL のプログラム

端末から同時に利用できるシステムにすることが必要な条件となる．

できうれば，複数の端末から，エディタに限らず，プログラムのコンパイルや実行等も同時に行なえるような時分割システムを作りあげることが望まれる．ところが，かなりの大きさのプログラム——データ領域を含め——をコンパイルし実行するには，それだけの記憶容量が必要であり，ミニコンの場合では1人分のコンパイル，実行が精一杯のところである．2次記憶装置も付加されていないので，スワップ等の技法も用いることができない．

こうして，エディタ（およびモニタ）だけが時分割で動くシステムとなった．それ以外のプロセッサやPTR 等の入出力機器を使う権利は一時には1つの端末にだけ与えられる．

3. で述べたモニタのコマンドは，こうした権利を保有している端末で受付けられるものである．各端末は，必要に応じてこの権利を獲得したり放棄したりする．権利保有の有無を示すため，モニタがコマンド待ちの時印字するのは，&& または & としてある．この権利に関してのコマンドは，次のとおりである．

^CG	&	HOLD PLOCESSORS
	&&	
RUBOUT	&&	RELEASE PROCESSORS
	&	

1人分のエディタとモニタの大きさはほぼ1kw である．これを4つコアに入れるのはもったいなさすぎるし，かといって，4つの端末の制御の切替えまでをエディタ・プログラムに含ませるのは，今日はやりのプログラムの構造化の精神にもとるというものである．

そこで，エディタとモニタのプログラムは，あたかも1台の端末しかないかのように作っておき，しかも複数台の制御をするようシステムが設計された．もちろん，言語プロセッサが動作中でも，他の端末でエディタは優先的に使えるようにしなければならない．これらの制御の切替えを司るプログラムは制御プログラムとして独立に作られた．

エディタとモニタとは，こうしてリエントラントなプログラムとして作られたことになる．この場合，2つのエディタ間での制御の切替えは，入出力動作時に限ることとして，制御プログラムの繁雑化を避けた．実際，1つのコマンドに対応するエディタやモニタの内部動作は充分に速く，その間他のプロセスが待たされても困ることはないからである．

エディタとモニタとは，純粋な手続き部分と，状態変数および大局的変数に厳格に区分される．大局的変数は，4つの端末を含め共通に使われるものであり，状態変数は特定の端末にだけ対応する変数である．状態変数は，制御プログラムにより，各端末に正しく対応づけられるように保たれる．

プログラムはアセンブラ語で書かれているが，これらの区分の様子は，高級言語風にかけば**図-7** のようになる．図中の構文要素で

case *i* **of** α: *A*; β: *B*; ••• **esac**

とあるのは，変数 *i* の値が条件 α を満たせば *A* を，β を満たせば *B* を，••• 実行することを示す．他の構文要素については特に説明を要さないであろう．

エディタのファイルは，コア上にとられる．文字単位の編集を早くかつ効率よく行なうために，4語ごとのブロックに分け，これを一方向きのリストとしてつないだものを1つのファイルとする．未使用のものは，あきリストとしてつながれ，*free* にその根元が入れてある．各ブロックには最大6文字が入れられるが

```
activation block    cp, lp, tp, mp, ch, s, p, w;
global free, processor, input, output;
Editor:
  begin
    loop: read (ch);
          if ch=RETURN then
          begin write (RETURN); read (ch);
                if ch≠Lf then
                  begin write (Lf); lp←cp end
          end;
          case ch of
          graphic: (insertion; cp←cp↑; write (ch));
          Lf: (insertion; cp, lp←cp↑; write (RETURN, Lf) );
              • • •
          esac; go to loop
  end;
Monitor:
  begin
    if s=EM then clear ( ); s←EM;
    write (RETURN, Lf, &); if hold( ) then write (&);
    print reasons;
    loop: write (RETURN, Lf, &); if hold ( ) then write(&);
          read(ch);
          if ch=RETURN then
          begin write ('EDITOR'); go to Editor end;
          if hold ( ) then
          case ch of
          even digit: (output←ch; •••);
          odd digit: (input←ch; •••);
          letter: (processor←ch; •••);
            •••
          esac else
            if ch=^G then
              if get ( ) then write (↑HOLD•••↑)
                         else write (↑REJECTED↑)
              else write (↑REJECTED↑);
          go to loop
  end
```

図-7 エディタ，モニタの構造

そのブロック内に行の区切りが含まれるか否かの旗はポインタの語に含ませてある．逆むきのポインタ移動はファイルの先頭またはその行の左端位置からたどりなおしてみつける．行単位の移動のときは，この旗だけをみながらリストをたぐるだけで済み，高速化に寄与している．

制御プログラムは，4つの端末の制御の切替えを行なうが，このため，各端末のプロセスが現在どの条件待ちになっているかの表を持っている．表は，割込み処理が終了するたびに調べられ，条件の満たされたプロセスに制御が移される．もちろん，エディタやモニタの方がプロセッサに優先する．

割込み処理は，割込み要因ごとに用意された処理ルーチンが行なう．そして，条件の旗が変えられる．割り込まれたプロセスがエディタやモニタの場合には，割込み処理後ただちに制御が戻されねばならない．そこでこれらは，「常に成立している条件」を持っているものとして表に登録される．

割り込まれたプロセスが，エディタ側であるか否かをハードウェア的に調べる機構として利用できるのは，割込みの生じた番地情報だけである．こうして，コアは大きく分割され，エディタ側とプロセッサ側に分けられている．

49 年度に用いたシステムの構成とメモリ地図とを**図 8,9** に示す．

ハードウェア的には何らの保護機構を持っていない

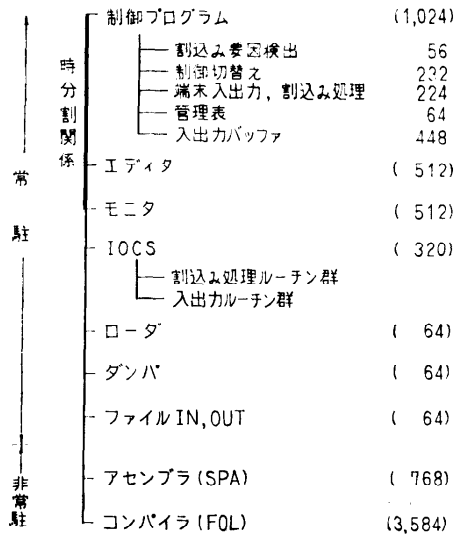


図-8　システム構成とプログラムの寸法

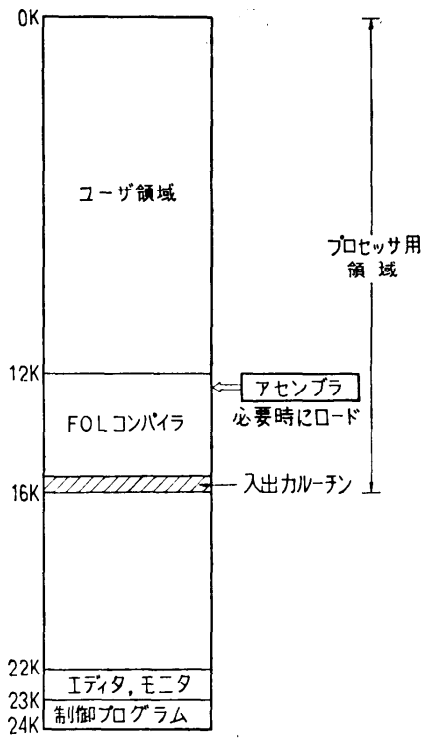


図-9　メモリ地図

ので，コンパイラの目的プログラムの内，配列の引用等危険な部分は実行時ルーチンを呼ぶ形にしてある．またアセンブラ使用時は，目的プログラムをインタプリタによって実行するようにした．

システムの大要は島内と筧が立案作成した．エディタとモニタは筧が，制御プログラムは島内が担当し，独立に作業した後1週間程度のテバックで仕上げた．FOL プロセッサは島内が作成した．入出力ルーチン等はすでに作ってあった SPS[3] システムを流用した．

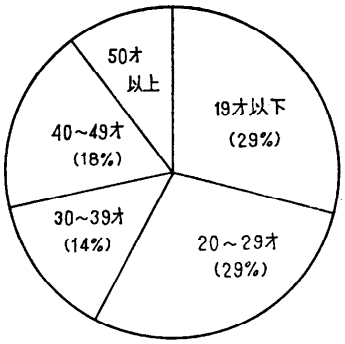
6. 特殊入出力の応用

ミニコンの通常の用途の1つに，非標準の入出力装置をユーザ側で作成結合し，オンライン的作業を行なうというのがある．番組の中でも，これらの特質を生かすべく非標準入出力装置を用いたので紹介しておく．

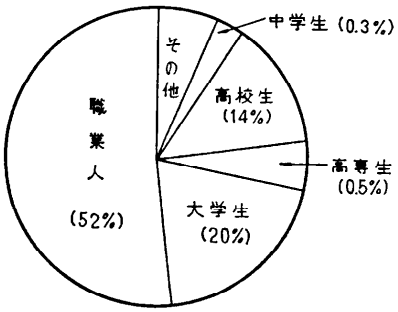
（1） 音楽の演奏

CPU のパネルに表示されるビットの内1つを選択しそれを直流増幅してスピーカを動作させる．プログラムで，ビットのオン・オフを適当な周期で繰返して音楽を演奏する．

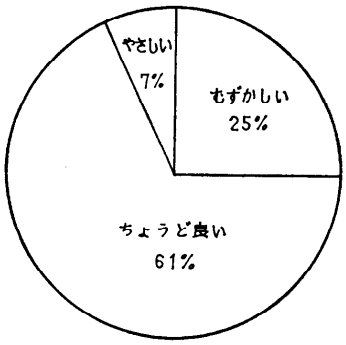
年令構成



学生/社会人



難易



アンケートの結果 609
(昭和49年11月15日現在)

意見欄からの抜粋
○番組全体について
・つめこみすぎ
・説明，進度が早すぎる
・ルポルタージュは有用
○内容について
・もっと初心者的に
・プログラミングのこつを主体に
・概説として見ると良くできている
○言語について
・初心者にわかりやすい (FOL)
・実際に使える FORTRAN を

図-10 アンケートの結果

(2) モール・コードの解読

電鍵のオン・オフの変化が生じるごとに割込みを生じるような簡単なインタフェースを作成した．CPUの時計でこのオン・オフ各期間の時間測定をするサブルーチンをアセンブラ言語で作り，あとは FOL のプログラムとして，手打ちのモールス・コードの解読を即時に行なうものを作った．単純な移動平均法の応用であるが，確度高く解読ができ，実時間処理の様子を見せるには格好の題材であった．

(3) 波形の表示

AD, DA コンバータを用いて，集録した波形から雑音成分を除いた波形を作り表示する例題をとりあげた．さらに，偏微分方程式の応用として，弦の振動をとりあげ，異なる初期条件の下での解をオッシロ・スコープに表示すると同時に，音としても聞かせた．難解になりがちなこうした例題も，音と図形と同時に伝えることができるというテレビ放送の特徴を生かせば，十分に理解させ易くすることができるのである．

7. おわりに

48, 49 両年度にわたった講座については，視聴者に対する効果において十分に成果があがったと思う．また初心者のみならず計算機について知識がある人でも見て面白いものであったと思う．ミニコンをスタジオにおいて番組を構成したことの成果であろう．

講師をひきうけた6人についても，それぞれに勉強になる点が多かった．ただ，内容的に欲ばりすぎて，全部を理解しようとかかると困難なものになったことは否めない事実であるし，比較的興味の似た講師の集まりであったことによるかたよりがあったかもしれない．

図-10 に，視聴者からのアンケートの結果を示す．このアンケートはテキストに用紙が綴じ込まれているものである．したがって，実際の視聴者全体の意見を代表していないかもしれないことを断っておく．

アンケートは，年齢区分，職業，講座内容の難易に限られており，他に意見を自由記入できるようにしたものである．意見中，FORTRAN 等を望むものがあるのは，プログラム実習の機会にめぐまれた視聴者の声と思われる．

作成したシステムは，教育用のものとして，この限られたハードウェア構成に対して，1つの解答となりうるものだと思う．同様なことを考えておられる方の参考になれば幸いである．

最後に，番組構成には，*NHK* の須之部淑男氏，伊藤和明，加賀美鉄雄，久保田久文，古賀義二郎，永辻功，野村洋子の各ディレクタ，安藤梢アナウンサの尽力およびアシスタントの南美香（西田真知子）さん，戸部加代子（渋谷加代子）さんの協力があったことを記して，感謝の意を表します．

参考文献

1) 島内剛一，筧捷彦：プログラミング言語 FOL，第16回プログラミング・シンポジウム報告集，pp. 119～126，(1975)．
2) 島内剛一，筧捷彦，加賀美鉄雄：ミニコンによる時分割システム，第 15 回プログラミング・シンポジウム報告集，pp. 260～270，(1974)．
3) 島内剛一："システムプログラムの実際"，サイエンス社，(1972)．

(昭和50年2月10日受付)
(昭和50年3月11日再受付)