医療用端末のマイクロコンピュータ・ソフトウェアシステム†

岡田美保子†† 岡田正彦†††

救急医療情報システムにおいては医師や看護婦などの医療従事者の協力が必要不可欠であるが，通常のコンピュータ端末のような複雑な機械装置では敬遠されがちである．そこで著者らはデザインを工夫し外観を単純にした医療用端末を作製し，その機能を充実させるため独自の端末ソフトウェアシステムを開発した．本端末では日常の業務は電源とキーパッドの操作だけで可能であり，またキーボードを取り付けると端末側でユーザプログラムを開発することができる．プログラミングにはインタプリタ型言語TEXTORを使用する．これは特にディスプレイやデータ通信制御に優れ，初心者にも容易に習得できるように設計された言語である．エディタにはプログラム開発に便利な諸機能があるが，特に一画面分のディスプレイ命令を自動生成する機能は本システム独特のものである．このソフトウェアシステムによりインタホーン同様の感覚で気軽に操作でき，しかも各種の情報サービス業務に広く応用できる強力な端末装置を実現することができた．

1. まえがき

現在，国内各地で救急医療情報システムの構築が進められている．これらのシステムの最も重要な業務の一つは，救急患者が発生した際に適切な診療機関を選択し，案内を行うことである．そのためには時々刻々に変化する各医療機関の応需状況（医師の待機状況，ベットの空き状況など）を正確に把握しておかなければならず，医師や看護婦などの協力が必要不可欠である．ところがこの目的のために通常のコンピュータ端末を医療機関に設置したのでは，操作が複雑なため（あるいは複雑に見えるため），ただでさえ忙しい医療従事者からは敬遠されがちである．そのため従来の救急医療情報システムの大部分は，外見上インタホーンのような簡単な端末（**図1**）を開発し，使用している[1)~3)]．これらの端末は確かに操作は容易であるが，入力装置としての機能しかもたず，たとえば端末を通じて他の診療機関の応需データを入手するなどの用途に用いることはできない．したがって診療機関にとってのメリットはほとんどなく，必ずしも医療従事者の好評を得ているとは言い難い．

著者らはインタホーン同様の感覚で気軽に操作でき，しかも各種の情報サービス業務に広く応用することのできるインテリジェント型の端末装置を開発した．すでに報告した通り，本端末は機械らしさを感じさせないようにデザインを工夫し外観を単純にしたが[4),5)]，簡単な操作で高度な処理もできるように，独自の端末ソフトウェアシステムで機能の充実をはかった．このソフトウェアシステム（ターミナルOSと呼ぶ）の特徴は次のように要約できる．1) ユーザプログラム用に開発した言語TEXTORには1~2行のコーディングでデータ通信が行える命令語がある．2) エディタには画面表示ルーチンを自動的にコーディングする特殊編集モードの機能がある．3) 日常の端末操作は電源ボタンとキーパッドだけでよい（キーボードを接続すれば端末側でユーザプログラムの開発を行うこともできる）．4) 簡単なキー操作によりホストコンピュータとの間でユーザプログラムの送受信ができ

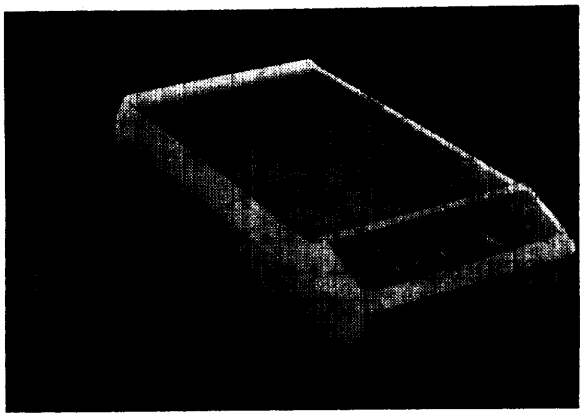

図1 既存の医療機関用端末の例（日本電気カタログより．）
Fig. 1 An existing computer terminal for use in clinical institutions.

† A Microcomputer Software of an Intelligent Terminal for Use in Medical Information Systems by MIHOKO OKADA (National Saigata Sanatorium, (Currently, Department of General Education, Niigata University)) and MASAHIKO OKADA (Department of Neurophysiology, Brain Research Institute, Niigata University).

†† 国立療養所新潟病院情報処理室（現，新潟大学教養部統計学研究室）
††† 新潟大学脳研究所神経生理学部門

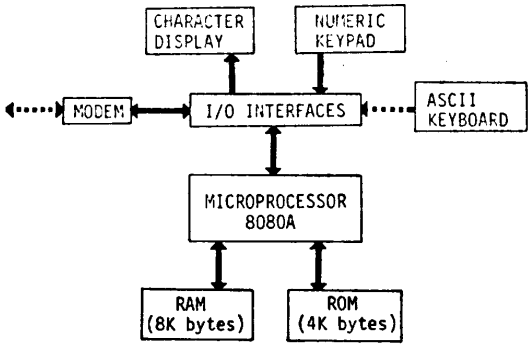

図 2 開発した端末のハードウェア構成
Fig. 2 Hardware configuration of the developed terminal.

る.

以下ターミナル OS について詳説する.

2. ハードウェア構成

図2に試作したインテリジェント端末のハードウェア構成をブロック図で示す. CPU にはマイクロプロセッサ 8080A を用いた. メモリは 4K バイトの EPROM (2708) および 8K バイトの RAM から成る. 入出力装置としてはキャラクタディスプレイ (16×32 文字) とキーパッド (0〜9, . , −の各キーおよび訂正, 頁送, 頁戻, 次の 4 ファンクションキー) がある. ユーザプログラムを自作するユーザには ASCII キーボードをオプションとして提供する. 入出力には 8251 シリアル I/O インタフェース, 8255 パラレル I/O インタフェース, モデムインタフェースなどがある.

当端末を設計する上での一つのねらいは, ユーザの眼からむしろ積極的に ASCII キーボードを隠すことにある. 医療関係者の多くはスイッチ類の多い複雑な機械の操作を極端に嫌うため, この点は特に重要である.

3. メモリ・レイアウト

図3にメモリ・レイアウトを示す. 4 K バイトの ROM 領域 (斜線部) にはモニタ, エディタ, インタプリタ, およびデータ・コミュニケーション・ハンドラ (DCH) から成るターミナル OS が常駐している. ユーザプログラム (TEXTOR プログラム) は 8 K バイトの RAM 領域のうちの 4 K バイト (ユーザプログラム域) に格納される. 残り 4 K バイトはインタプリタの作業域 (1.5K), 受信バッファ (2 K), ディスプレイバッファ (0.5K) の 3 部に分割されている. 受信バッファには TEXTOR プログラムの要求によりホストコンピュータから送信されてきたデータを格納する. ディスプレイバッファは一画面分の文字 (16×32 文字) の記憶域である. 受信バッファ以外の RAM 領域はすべて CMOS RAM を使用し, 水銀電池でバックアップを行っているので, ac 電源を切った後でも記憶内容が消失しない.

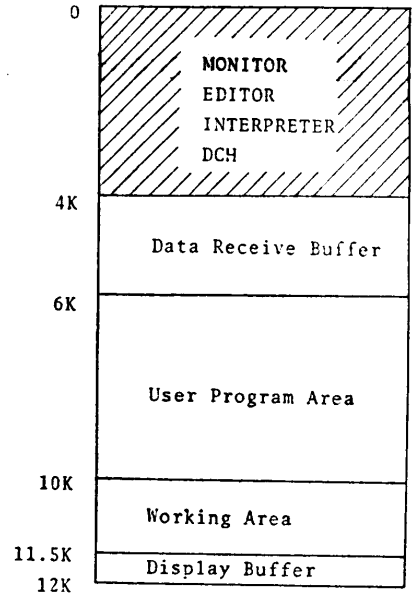

図 3 メモリレイアウト
Fig. 3 Memory layout.

4. ターミナル OS

4.1 システム制御の流れ

ターミナル OS の制御の流れ (ASCII キーボードが端末に接続されている場合) を図4に示す. 電源を投入するとただちにエディタが作動し編集セッションが開始する. プログラムの編集中はエディタの CTRL/G または CTRL/X コマンド (後出) によって制御をインタプリタに移し, 開発中のプログラムを実行させることができる. インタプリタの実行中, シンタックスエラーが見つかると, 制御はエディタに戻る. DCH は, 端末がインタプリタの制御下 (ユーザプログラムの制御下) にある時, TEXTOR 通信制御文 OUT・IN によって呼び出され, ホストコンピュータとの間のデータ転送を行う.

ユーザは必要な時点でいつでもプログラムの送信または受信を要求することができるが, この際にも DCH が呼び出される. プログラムを送信したい場合は次の手順を踏めばよい. ①頁送キーを押えながら電源ボタンを押す (これでモニタが作動する). ②'ARE YOU

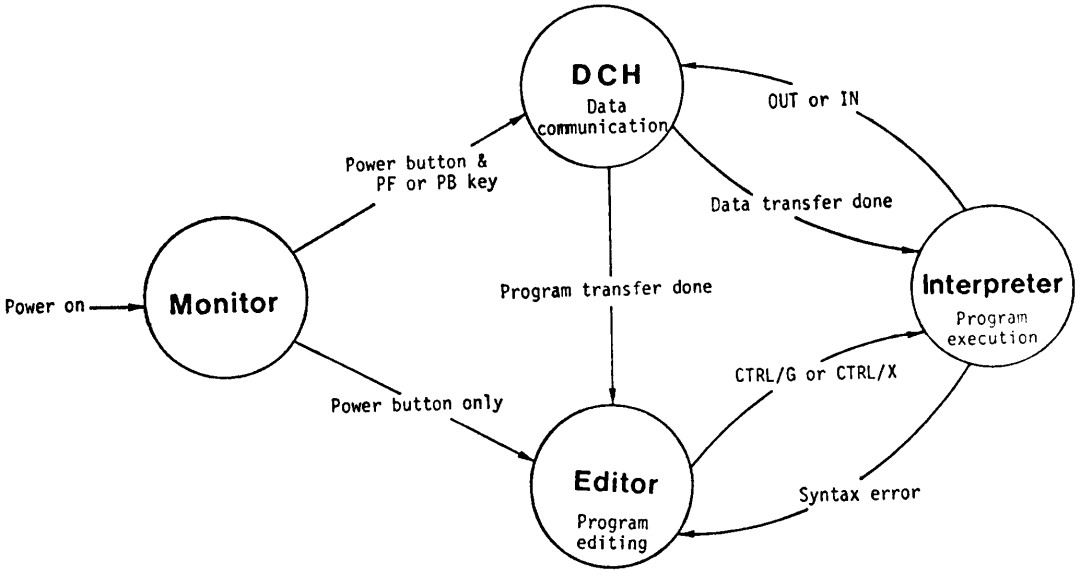

図 4　システム制御の流れ（ASCII キーボードを取り付けてある場合.）

Fig. 4　Flow of the control among the system programs when the ASCII keyboard is attached to the terminal.

SURE? 1. YES 2. NO' の表示に対し 1 を入力する.

③ 'PROGRAM ID=' の表示に対し，送信するプログラムの識別コードを入力する．以上の簡単な手続きで，識別コードと共にユーザプログラムがホストコンピュータへ送信される．プログラムを受信したい場合には①のステップで頁送キーの代わりに頁戻キーを押えればよい．DCH がユーザプログラムの転送を終えると制御はエディタに戻る．ASCII キーボードがはずされている場合は，制御の流れからエディタの介入部分が除かれる．

4.2 プログラミング言語 TEXTOR

TEXTOR は特に次の点に設計目標を置いて開発した端末用のインタプリタ型言語である．

a）ホストコンピュータとのデータ通信が容易である．

b）ディスプレイの制御が容易である．

c）命令語の数が少なく，初心者にも容易に学べる．

d）インタプリタのサイズが小さい．

マイクロコンピュータ用言語はすでにいくつか報告されているが[6)~8)]，これらの条件をすべて満たすものはない．TEXTOR の性能については「5. 言語の評価」の章で論ずることにし，ここでは TEXTOR の特徴の一つであるデータ通信制御命令 OUT・IN についてだけ説明する．

TEXTOR プログラムでは，データリンクを設立した後のデータ転送の手順は次の二通りのうちのいずれかに限られている．

（手順 1）データを端末からホストへ送信し，リンクを打ち切る．この場合は一つの OUT 命令だけで処理を行うことができ，必要なコーディングは一般に

OUT (id, v_1, ⋯, v_n)

と表される．OUT はホストとの間にデータリンクを設立し，変数 $v_1, \cdots, v_n$ の内容をホストへ送信する．id はデータ $v_1, \cdots, v_n$ の識別コードを表す．

（手順 2）データを端末からホストへ送信し，ついでホストから返信されるデータを端末で受け取り，リンクを打ち切る．この場合は次のように OUT と IN を併用する．

OUT (id, v_1, ⋯, v_n)

IN (v)

OUT の機能は手順 1 で述べた通りである．IN 命令はホストから返信されるデータを受け取り，受信バッファに格納してその内容を画面表示する．ホストからのデータはディスプレイ画面の 1 ないし数ページ分のテキスト（あらかじめ定められた書式に従い，各行，各ページ，各受信バッファ内のテキスト，および全テキストの終わりをそれぞれ示すコードで編集されたもの）である．IN は 1 ページ分のテキストを表示すると端末操作員の応答を待ち，頁送または頁戻キーの入力に応じて次ページまたは前ページへのスクローリングを行う．数値（あるいは「次」キーのみ）の入力

```
034          DIS(2,3,'2.セイヘ"ツ ヲ タイフ・
035          #シテ クタ"サイ。')
036          DIS(3,5,'----------------
037          #--------')
038          DIS(6,5,'1.  オトコ')
039          DIS(8,5,'2.  オンナ')
040    S1:   GET(SEX)
041          IF(SEX<1) JMP S2
042          IF(SEX<3) JMP S3
043    S2:   DIS(15,3,'・・・ ニュウリョク エラー
044          #')
045          JMP S1
046    S3:   ERS()
047
```

図 5 エディタ標準モードの画面の例

Fig. 5 Example display during editor standard editing session.

があると，その値を変数 v に格納して実行を終了する．この一連の処理は画面が項目選択のメニューから成り，端末操作員から項目番号の入力が行われる場合を想定したものである．

4.3 エディタ

TEXTOR プログラムの開発はエディタの制御下で，ASCII キーボード（プログラム開発のときだけ取り付ける）を用いて行う．制御がエディタに移るとユーザ域内のプログラムが**図 5** のように画面表示される．これは患者データをホストコンピュータへ送信するためのユーザプログラムの一部で，性別を端末から読み込むところを示す．DIS は画面表示の命令で，一般に

DIS (i, j, '$c_1 \cdots c_n$')

の形式で i 行 j 列目から文字列 $c_1 \cdots c_n$ を表示することを指定する．GET はキーパッドから項目番号を読み込み，ERS は画面を消去する命令である．

情報システムにおける日常の端末操作では，普通何種類ものメニューなどの画面表示が必要である．したがって TEXTOR のプログラミングではディスプレイルーチンの作成が大きな比重を占めていることが多い．ある一画面を表示するルーチンを通常の編集機能で作成するには，画面全体の構図を考えながら個々の要素の位置を決め，それを具体的な行番号，列番号で置き換えていかなければならない．そしてプログラムを実行してみて思い通りに表示されなければ修正を加え，また実行してみるという繰り返しが必要である．こうした作業は単純な割に時間を要し，煩わしいものである．そこでエディタには一般的な編集機能（標準編集モード）に加え，一画面分のディスプレイ命令を自動生成することができる特殊編集モードの機能を持たせた．特殊編集モードを開始するにはエディタコマンド（一般に CTRL/a と表し，CTRL キーを押えながらある 1 文字 a をタイプする）のうちの CTRL/F または CTRL/X を用いる．

CTRL/F を入力するとただちに特殊モードに入り，画面は一掃されてカーソルだけが表示される．カーソルは '→', '←', '↑', '↓' のファンクションキーで各矢印の方向に動かすことができる．キーを打つとカーソルの位置にその文字が表示され，同時にディスプレイバッファの対応する番地にその文字コードが格納される．文字の修正は，1) 置き換えモード（カーソルの位置に表示されている文字が入力された文字と置き換わる），2) 挿入モード（カーソルの右側の文字列全体が 1 文字分右へ移動して空いた所に文字が挿入される）のいずれかのモードで行われる．特殊モードにおける CTRL コマンドには 1) カーソルが位置する行より下のすべての表示内容を 1 行分上または下に移動する，2) 文字修正のモードを変更する，3) いま文字が表示されている位置をマークして，その部分に対しては DIS 文が生成されないようにする（マスクオフコマンド CTRL/M），4) 現在表示されている画面をそのまま再現する一連の DIS 文をユーザ域内に生成し，標準モードに戻る (CTRL/E), などの機能がある．

特殊モードを開始するもう一方のコマンド CTRL/X ではユーザ域内のプログラムをエディタの現在位置まで実行した後，特殊モードに入る．CTRL/X で特殊モードに入ったときは，画面にプログラムを実行した結果生じた映像がそのまま残っている．この時点で画面に文字を追加し，その追加した部分についてだけ DIS 文を生成したい場合は，先に挙げた 3) のマスクオフコマンド (CTRL/M) を用いる．

ここで具体例を用いて DIS 文自動生成の機能について説明する．いまユーザが図 5 に示すようにエディタ標準モードで TEXTOR プログラムの入力を行っているとする．これから診療科名選択のメニューを表示するルーチンを作るため特殊モードに入る．CTRL/F を入力すると画面が一掃されるので，表示したいメニューを直接画面上にタイプしていく (**図 6** (a))．特殊モードを終了するコマンド CTRL/E を入力すると，いまタイプした画面を再現する一連の DIS 命令がユーザ域の中に生成・追加されて標準モードが再

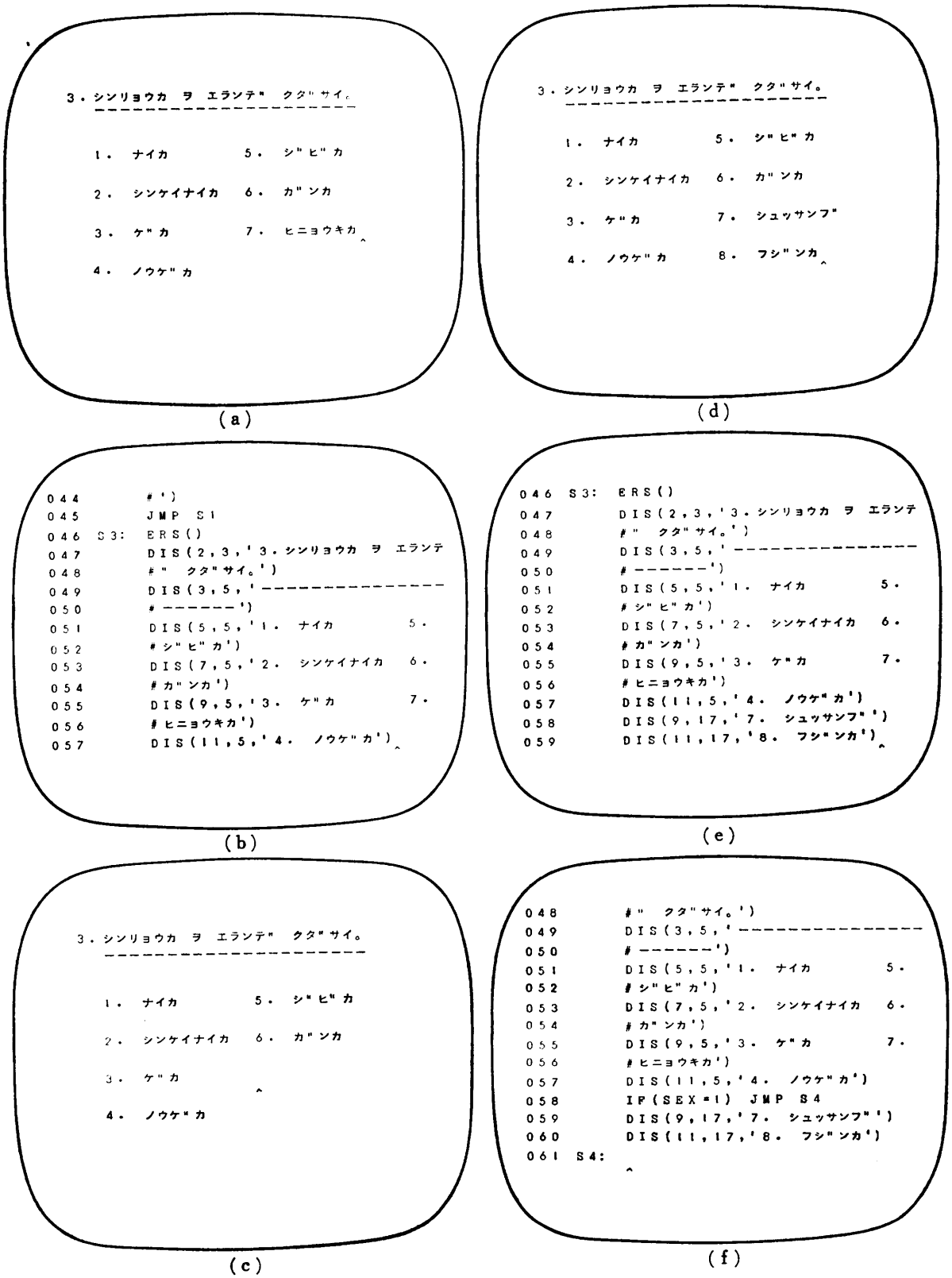

図 6　エディタ特殊モードによるディスプレイ命令自動生成の例

Fig. 6　Display sequence showing automatic generation of DIS statements by editor special editing mode.

開し，いま生成された命令文を見ることができる（図6(b)の行番号047～057）．ここでGET文（図5 040）で読み込んだ変数SEXの値が2のときは図6(a)のメニューの項目7を'7. シュッサンプ'とし，項目'8. フジンカ'を追加したいとする．今度はいま作成したメニューを画面に再現してから特殊モードに入るようにCTRL/Xを入力すればよい．インタプリタが現在のエディタの位置（図6(b)057）まで実行し終えると，画面に図6(a)のメニューを表示したまま特殊モードに入る．この時点でまず項目7の部分を空白にし（図6(c)），CTRL/Mを入力する．（これで後にCTRL/Eを実行しても図6(c)に表示されている要素についてはDIS文が生成されない．）次いで必要な項目を書き加え（図6(d)），CTRL/Eで特殊モードを終了する．標準モードが再開すると，項目7，8を表示するDIS文だけが新たに生成・追加されている（図6(e)058～059）．あとは標準モードでIF文を1行書き加えればよい（図6(f)058）．

5. 言語の評価

5.1 実　験

既存のマイクロコンピュータ用言語の中で現在最も広く用いられているのはBASICである．そこでTEXTORの端末用言語としての性能を客観的に評価するためBASICを比較の対象として取り上げ，端末ユーザプログラムの書きやすさに関する次のような実験を行った．

実験方法　プログラミング経験のない10名の学生（人文，教育，経済，理，工，各学部の教養課程1年生）を5名ずつ，A，B二つのグループに分け，初めにAグループにBASIC，BグループにTEXTORをそれぞれ7時間ずつ講習し（端末を使った演習を含む），両グループに同一の課題を与えてそれぞれの言語で解答させた．次に両グループの言語を交換し，1回目とまったく同じ要領で講習，試験を繰り返した．

課題の内容は次の通りである．

課題1　図7(a)のようにメニューを表示して項目番号の入力を待つ．入力があったら図7(b)のように表示せよ．（TEXTORでは特殊編集モードを使うこと．）

課題2　患者データを読み込み，ホストコンピュータへ送信せよ．ただしデータ入力は次のように行う．まず図7(c)の(1)のように表示して年齢を尋ね，入力があったら続いて(2)のように性別を尋ねる．入力があったら最後に(3)の診療科名を尋ねる．次いで読み込んだ年齢，性別，診療科番号をデータ識別コード（3と仮定する）と共にホストへ送信せよ．

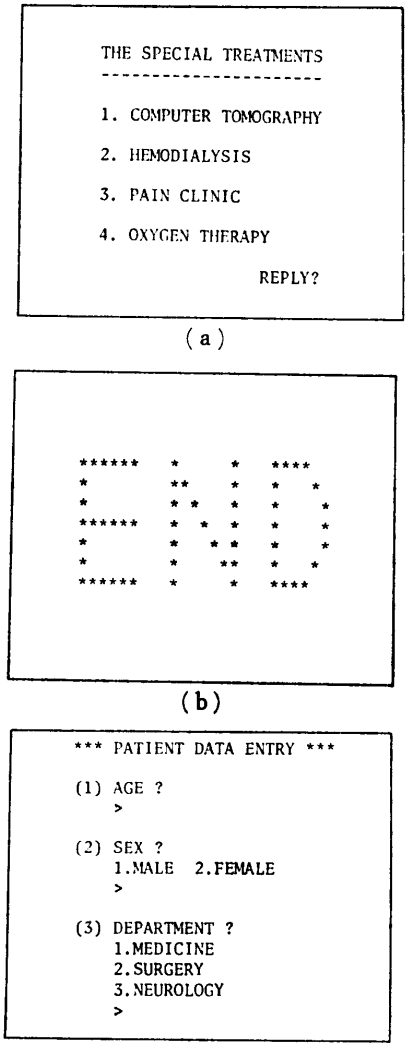

図7　試験課題で表示を指定した画面

Fig. 7　Display frames required to be produced in the tasks.

課題3　0～9の範囲内の数字が5個入力される．これらを読み込んで，小さいものから順に並べ変え，画面に表示せよ．

このうち課題1は，あらかじめ決めた順番に従い1人ずつ端末上でプログラムを作成し，1人が課題1を行っている間，残りの者は課題2，3を行うよう指示した．課題2，3については机上でコーディングを行うだけでよいことにし，監督者が見て誤りがないと判定した時点で終了とした．3課題を通して制限時間は3時間半とした．

現在BASICをサポートするマイクロコンピュータ

表 1 BASIC と TEXTOR の規模の比較

Table 1 Comparison of the size between BASIC and TEXTOR.

項　　目	BASIC	TEXTOR
全命令語の数	67	19
講習を行った命令語の数	32	19
インタプリタのサイズ	8K	2K*
エディタのサイズ**	2K	2K

* DCH を含む.

** BASIC ではエディタだけの大きさ. TEXTOR ではモニタも含む.

システムは数多く市販されており，機種により拡張命令に多少の差があるが，基本的な命令語に関してはほとんど同じである．本実験では，データ通信の機能を備え，しかも最も販売台数の多い機種の一つである PET (米国 Commodore 社)[8),9)]を使用した．

5.2 結果および考察

まず両言語の規模を比較するため，全命令語の数，講習を行った命令語の数，インタプリタのサイズを**表 1** に示す．参考のため表 1 にはエディタのサイズも示してある．**表 2** は講習を行った命令語の一覧である．BASIC の講習では時間の制約上，端末ユーザプログラムの作成に最も影響が少ないと思われる機能（組込関数，クロック制御，絶対番地による実行制御など）の説明を省いた．BASIC では高度な数値計算の機能や類似の機能を持った命令語の重複があるため，命令語の数が多く，インタプリタのサイズも大きい．これに対し TEXTOR の機能は端末ユーザプログラムの作成に必要なものに限られているので命令語数が少なく，インタプリタもはるかに小さい．TEXTOR の場合は 7 時間の講習で全命令語を網羅することができた．

次に各課題を終了できた人数，平均所要時間，およびプログラムの平均ステップ数を**表 3** に示す．課題 1 についてみると，BASIC では 1 回目，2 回目でそれぞれ 1 人，2 人しか終了していないのに対し，TEXTOR（特殊編集モードを使用）では 4 人，5 人がそれぞれ終了しており，TEXTOR の所要時間はいずれの回も BASIC のわずか 1/3 となっている．課題 2 では最後にホストコンピュータへデータを送信することが要求されているが，BASIC でこれを行うには伝送制御のルーチンを作成しなければならない．しかしそれにはデータ通信に関する専門知識が必要であり，予備知識のまったくない被験者が簡単な講習の直後にこれを作成することはきわめて困難である．そこで，BASIC の場合は後で伝送制御のルーチンを挿入することを仮定して，一般の I/O 機器と同じ要領でデータを出力するだけでよいことにした．課題 2 の成績はこのような仮定に基づくもので，したがって所要時間の多くはディスプレイ命令のコーディングに費やされたものであるが，ここでも課題 1 と同様の傾向が見られ，TEXTOR では BASIC の半分以下の時間で済んでいる．またステップ数についてみても，課題 1，2 共に TEXTOR の方がかなり短くなっている．課題 3 はロジカルな問題におけるコーディングのしやすさをみるために出題したものであるが，1 回目は BASIC で 3 人，TEXTOR で 2 人，2 回目は BASIC 4 人，TEXTOR 5 人と，両言語でほぼ同数の者が終了している．所要時間はいずれの回も BASIC の方が短いが，ステップ数では 1 回目 BASIC，2 回目は TEXTOR の方が短かった．いずれにしても，課題

表 2 講習を行った命令語の一覧

Table 2 List of instructions covered in the training course.

機　　能	BASIC	TEXTOR
入 力	READ, DATA INPUT GET	GET CHR
出 力	PRINT TAB SPC	DIS ERS
判 定	IF...THEN IF...GOTO	IF, ELS
繰り返し・サブルーチン	FOR, STEP, NEXT GOSUB ON...GOSUB RETURN	EXC, END RET
飛び越し	GOTO ON...GOTO	JMP
代 入	(LET) =	=
配列宣言	DIM	SIZ
注 釈	REM	;
ビット型変数操作	—	BIT ON OFF INV
データ通信	—	OUT IN
文字列操作	LEFT$ RIGHT$ MID$ LEN	—
実行停止・終了	STOP END	—
ファイル操作	OPEN	—
操 作	CLR CONT LIST	—
	RUN NEW	エディタコマンドに対応

表 3 評価実験の結果

Table 3 Performance on the tasks given in the experiment.

試験	グループ	言語	課題1	課題2	課題3
1回目	A	BASIC	1 人*	2 人	3 人
			130 分**	103 分	74 分
			60 行***	34 行	25 行
	B	TEXTOR	4 人	4 人	2 人
			40 分	30 分	100 分
			30 行	18 行	31 行
2回目	B	BASIC	2 人	5 人	4 人
			90 分	51 分	48 分
			41 行	27 行	27 行
	A	TEXTOR	5 人	5 人	5 人
			29 分	24 分	77 分
			30 行	17 行	21 行

* 終了者数，** 平均所要時間，*** 平均ステップ数．

3に関しては両者に大きな差は見られない．

これだけの結果から，どちらの言語がより優れているか単純に結論づけることはできない．しかし端末用の言語に要求される特殊な機能に関する限りは次のことが言える．まず第1に，TEXTOR では被験者のほぼ全員が課題2に対して完全なプログラムを作成することができたことからもわかるように，データ通信の制御が BASIC に比べはるかに容易である．第2に，課題 1，2 の結果から，ディスプレイの多い処理では TEXTOR の方がはるかにプログラミングが容易である．

6. むすび

救急医療情報システムにおける医療機関用端末のマイクロコンピュータ・ソフトウェアシステムについて報告した．端末ユーザプログラムはインタプリタ型言語 TEXTOR を用いて書く．エディタのディスプレイ命令自動生成の機能は本システム独特のものであり，これを用いることによりディスプレイルーチンの作成に要する時間を大きく短縮することができる．TEXTOR の端末用言語としての性能を評価するため，プログラミング経験のない10名の学生を被験者として実験により BASIC との比較を行ったところ，TEXTOR ではデータ通信の制御やディスプレイルーチンの作成が BASIC に比べはるかに容易であることが明らかになった．

本端末では独自のマイクロコンピュータ・ソフトウェアシステムにより，電源とキーパッドの簡単な操作だけで情報検索などの処理ができる一方，コンピュータの専門家でない医療従事者が必要に応じてプログラムを作製・変更することも可能となった．

本研究は，新潟市医師会救急医療情報システム開発協議会の研究活動の一環として行われた．委員各位のご協力に深謝する．

参考文献

1) 青山松次：救急医療情報システムの研究開発，経過と現状と将来，神奈川県医師会報，No. 299, pp. 1～23 (1977).
2) 日本電信電話公社：救急医療情報システム，日本電信電話公社.
3) 秦 資宜：茨城県における救急医療システム設定の経過とその概要について，情報処理学会医療情報学研究会資料 4-3-1 (1980).
4) 岡田正彦，岡田美保子：医療用インテリジェント端末の開発，医用電子と生体工学，Vol. 17, No. 6, pp. 448-449 (1979).
5) Okada, M. and Okada, M.: A new intelligent terminal for clinical computer network, Medical & Biological Engineering & Computing (in press).
6) Chien, Y. P. and Dreizen, H. M.: A high-level language for microprocessor applications, Microcomputer Conference (USA), pp. 30-39 (1977).
7) 水野忠則，井手口哲夫，首藤 勝，中村敏行：マイクロコンピュータ用言語 PL/Iμ の設計と作成，情報処理，Vol. 18, No. 5, pp. 424-429 (1977).
8) 対馬勝英，松田 稔：PET BASIC 入門，ワールド美術印刷株式会社，大阪 (1979).
9) 対馬勝英，松田 稔，加賀英徳：PET 2001 を計算機教育に使って—PET ソフトウェアと大学教育，bit, Vol. 12, No. 11, pp. 1601～1607 (1980).

(昭和55年8月12日受付)
(昭和56年2月19日採録)