LISP 語による知能端末ソフトウェアの試み†

沼田 一道††

知能端末の制御プログラム記述言語として LISP 語を使うことを提案する．その有効性を検証するために，パーソナルコンピュータの標準的 O.S. の一つである CP/M の下に通信機能を備えた LISP 処理系を試作し，その上で動く端末制御プログラムを作成して実験を行った．本論文では，試作システム（処理系と端末制御プログラム）の機能と構成法を述べ，従来用いられてきた BASIC 語と比較し，LISP 語の方が，知能端末記述用の言語として利点が多いことを明らかにする．

1. はじめに

パーソナルコンピュータ（以下，「パソコン」と略記する）を TSS 端末として利用し，これを「知能端末」と呼ぶことが定着しつつある．この利用法は，

（＊） パソコン利用の立場から見て，パソコンの処理能力がホストのそれによって増強される．

（＊＊） TSS 利用の立場から見て，単能の専用端末と比べ，柔軟かつ高度な TSS 処理ができる．

という利点をもち，今後の発展が期待されている．

石田[1]は，この両面を統合し，使い勝手のよいパソコン知能端末を構築するための課題として，次のような指摘をしている．

（1） ホスト側 O.S.（TSS サーバ）の機能整備

（2） 通信施設（方式）の改善

（3） 端末(パソコン)側ソフトウェアの経験的研究

いずれも重要な課題であるが，以下では問題を(3)に限定し，(1),(2)とは独立の（あるいは(1),(2)が変わっても柔軟に対処できる）立場から議論を進める．もちろん，端末ソフトウェアについては，従来から多数の試みがなされ，そのプログラム等も公表されている[2)-6)]．その多くは BASIC 語で記述されており，BASIC 処理系の強力な支援を背景に(＊＊)の面ではかなりの成功をおさめている．しかし，(＊)の面はあまり考慮されていないし，またパソコン知能端末の基本的性格，その可能性を論じたものは，ほとんどない．

本論文では，現実的な要請をふまえて，パソコン知能端末の性格づけを行い，そこでの考察をもとに，

(LISP 処理系)+(LISP 語による端末制御プログラム)

から成る端末ソフトウェアを提案する．また，このシステムをパソコンの標準的 O.S. の一つである CP/M（機械は MZ-80 B）上に試作したので，その機能，構成法，使用例を報告する．

2. パソコン知能端末

2.1 現 状

従来公表されているパソコン知能端末の動作内容は，概略**図1**のとおりである．点線の左側が TSS セッションを管理する通信制御部分，右側があらかじめ組み込まれた仕事を行う知能部分であり，両者は一体となっている．

通信制御部はキーボードから文字列を入力し，それをホストに送るメッセージかパソコン側で処理すべき命令か判別し，対応する処理を行いながら送受信を繰り返す．

知能部分の仕事は，プリンタ出力フラグの on/off，ファイルの転送，パソコン側ファイルのプログラムをホストで実行させ，結果を受け取る等である．セッション開始手続きでは，ユーザ ID，パスワード等の処理を自動的に行うものもある．

このようなパソコン知能端末を記述する言語としては BASIC 語が圧倒的に多く用いられている(BASIC 語で作成されたパソコン知能端末を，以下では「BASIC 端末」と呼ぶ)．その理由としては，

（1） パソコン上で使える言語が BASIC しかない．

（2） 各機種に即した BASIC 処理系がそれぞれの周辺機器制御を強力かつキメ細かに支援している．

ということもあるが，やはり，

（3） (端末制御) プログラムの作成，修正，拡張，

† An Implementation of Intelligent Terminal Software on LISP System by KAZUMITI NUMATA (Department of Information Mathematics, Faculty of Electro-Communications, University of Electro-Communications).

†† 電気通信大学電気通信学部情報数理工学科

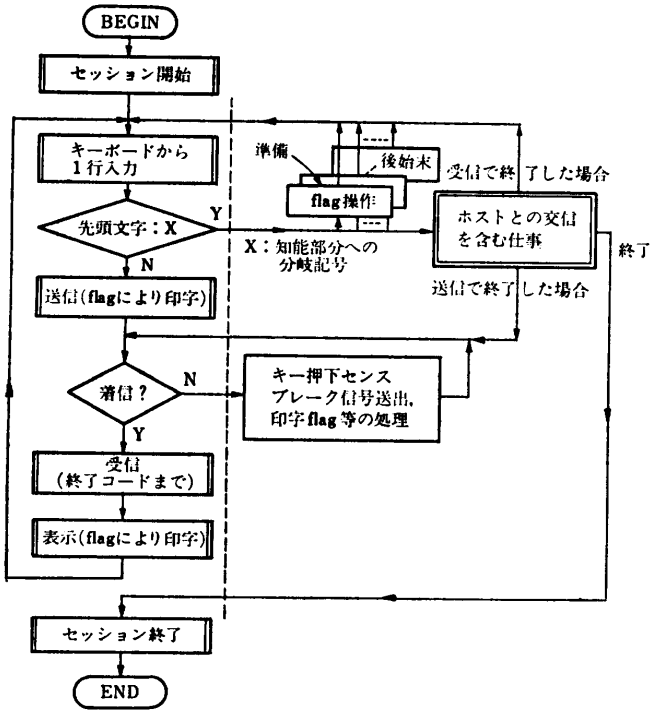

図 1 通常のパソコン知能端末制御プログラムの概略フロー
Fig. 1 General flow of the usual control program for the intelligent terminal using a personal computer.

実行の手軽（容易）さという点で BASIC 語が優れている.

からであろう. BASIC 語の代りにアセンブリ語, コンパイル語を用いることも考えられる（「機械語端末」と呼ぶ）が, 動作速度の点で有利な反面, 作成, 修正, 拡張は面倒になる. 現在の BASIC 端末の最大の利点は「端末機能の可変性」にあるといえよう.

比較のために, 種々の TSS 端末（としても利用できるもの）を「通信を含む処理の自由度」という観点から, 列挙・分類して**表 1** に示しておく[7),8)].

2.2 改善の方向

一般に, パソコン知能端末は次のような性質をもつことが望ましい.

（i） パソコン上で経済的に実現でき, 手軽に使える.

（ii） セッション制御部分を利用者が比較的簡単に作成変更できる.

（iii） ホスト（内プログラム）との通信を含む処理プログラムの作成・組込みが容易.

（iv） オンライン時にも, パソコン上でプログラムを作成し, それを実行させられる.

BASIC 端末の場合, (i), (ii)を満足しているが, (iii), (iv)に問題がある. すなわち, BASIC 端末に新しい機能を追加・実現するには, そのつど当該プログラムを端末制御プログラムへ組み込まねばならず, またその作業を行うには, オフライン状態で, プログラムの作成, 再編集, デバッグを行わなくてはならない（端末制御プログラムの下で別のプログラムの作成, 実行はできない）. 結局 BASIC 端末は,

オンライン時：あらかじめ端末制御プログラムに組み込まれた機能だけを果たす「知能端末」.

オフライン時：ホストとは無関係のたんなる「パソコン」.

であり,「パソコンのコンピュータ・システムとしての能力を TSS システムを利用して増強する」という(*)の面では, 不十分であるといわざるをえない.

ここで, (iii), (iv)を満足し(*)の方向での可能性

表 1 TSS 端末の類別
Table 1 Classification of TSS terminals.

区分		可変性	特徴
専用端末		機能固定	送受信と表示（印字）および付随する制御（エスケープ列の解釈, 実行, パラメータの設定等）だけを行う.
知能端末		機能（半）固定	固有の周辺機器を備えており, それらを利用した通信処理ができる. 利用方法等は定形的.
パソコン知能端末	機械語端末	機能可変（変更は面倒）	専用端末のエスケープ列仕様を実現する場合など, 高い通信・処理速度が要求される用途に適する.
	BASIC 端末	機能可変（変更は容易）	インタプリタ方式の BASIC は遅いが,「TSS を介してプログラムを作成し, 実行・デバッグする」といった用途の端末を構築する場合には大体間に合う.
ワーク・ステーション（複合パソコン）		on-line 状態でプログラムの作成, 編集, 実行等が可能	通信制御プログラムは優先度の高いタスク/ジョブとして動作しており, 別のタスク/ジョブとして走る業務プログラムとは分離されている.「端末」というよりは, 計算機間通信を行う「局」とみなせる.

をもったパソコン知能端末を構成することが問題となる．実現方法としては次の(1),(2)が考えられる．

(1) リアルタイム処理を含むマルチジョブ（タスク）モニタの下で，端末制御プログラムと翻訳・編集ソフトウェア/目的プログラムを併走させる．

(2) 端末制御プログラムの下で翻訳・編集ソフトウェア/目的プログラムを動作させる．

(1)はパソコンをワーク・ステーション化する自明の解決策（表1参照）であるが，パソコンの手軽さ，経済性が失われてしまいあまり魅力的でない．われわれは(2)の方法を採用する．そして，端末制御プログラムの下で動く言語処理系を一つに限定し，端末制御プログラムをもその言語で記述する．こうすることにより通常パソコンの枠内で実現可能となり（複数個の処理系を許せば，結局方法(1)と変わらなくなってしまう），さらにまた(ⅱ),(ⅲ)に関しても好都合である．

2.3 LISP 端末

以上論じてきたパソコン知能端末の方向を実現するものとして，われわれは，

LISP 端末＝(LISP 処理系)＋(LISP 語による端末制御プログラム)

を構想した．LISP 語を採用した最大の理由は，

(EVAL (READ))

と2個* の関数によって（ユーザ）プログラム中で簡単に LISP 処理系そのものを実現できることである．

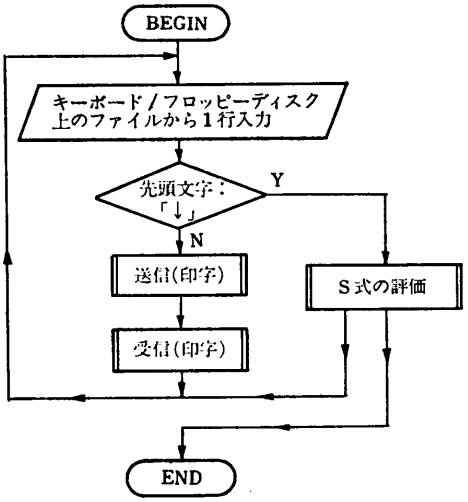

図 2 LISP 端末制御プログラムの概略フロー
Fig. 2 General flow of the LISP terminal control program.

* 実際にはエラー処理等のため，ERSET の下で EVAL を使う．

LISP はこの他にも，後述するようないくつかの望ましい性質をもっているが，(ⅳ)を満たす上で本質的なのは，関数 EVAL の存在である．LISP 端末制御プログラムは，入力記号列を先頭（Cr，Lf の直後）の記号によって，

(a) ホストへ送出すべきメッセージ

(b) パソコン（LISP 処理系）が処理すべきS式に分別し（記号「↓」で始まるとき(b)，それ以外のとき(a)と約束*），図2のように動作する．

図1と図2の動作はよく似ているが，BASIC 端末の場合，知能部分の動作があらかじめ端末制御プログラムに組み込まれているのに対し，LISP 端末では，知能部分への手懸りとして関数 EVAL だけを組み込んでおき，個々の知能的動作は後から（「↓」に続く）S式で与えればよいという点が異なる．

LISP 語においては，「S式の評価」によって，プログラムの作成，編集，実行，（二次記憶から/へ）の取り出し/格納のいっさいが行われるのであるから，

セッション中の任意の時点でパソコンの計算機としての能力を利用できる

ことになり，結局，

(*)の面での可能性を備えたパソコン知能端末が実現されたことになる．

前節の(ⅰ)～(ⅳ)と照合しながら，LISP 端末の性格を明らかにしておこう．まず(ⅰ)に関しては，LISP 処理系が基本的にインタプリタであることから，BASIC の場合と同程度の手軽さ，扱いやすさが期待できる．(ⅱ)に関しては，＊LINZ，＊SDI，＊RV1，＊RX?，＊BRK 等（表3参照），送受信のための基本関数を処理系に組み込むことで対処する．送受信メッセージ（一般には，リストで表現される）を処理・加工する際の柔軟性という点で LISP 端末は BASIC 端末に勝るとも劣らない．(ⅲ)に関しては，しかるべき機能を関数として定義するだけで当該機能の組込みが完了するという利点がある．このことは，自然な形で，処理単位ごとのモジュール化をもたらし，また各モジュール（関数）は端末制御プログラムから完全に切り離されているので，変更，修正も容易である．(ⅳ)は先に述べたとおり，LISP インタプリタをユーザレベルで記述することにより，完全に解決される．要するに，LISP 端末は，(ⅰ)～(ⅳ)を満たす一つの

* 回線上の符号長を7と仮定しているので，16進80以上のコードをもつ記号を使う（「↓」のコードは 81）．パソコンは，この種のものを「特殊記号」として用意しているので，適当なものを選べばよい．

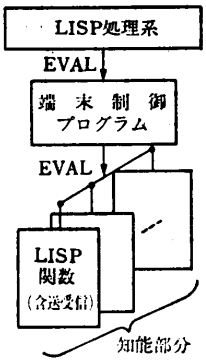

図 3 LISP 端末 (全体の構成)
Fig. 3 LISP terminal (system configuration).

可能解になっているのである.

このような LISP 端末全体 (LISP 処理系をも含めた) の構成は図 3 のようになる.

ユーザは, 任意の関数 (機能モジュール) を

↓ (関数名 実引数1 実引数2…)

という指令で実行させればよいので, LISP 処理系とホスト内の TSS サーバの両者を同等に利用しているという印象を受ける. もちろん, これは錯覚ではなく, 実質的にそのとおりなのである.

さらに, 本論文では立ち入らないが,「ホスト (内プログラム) と人間の介入しない対話処理を行うプログラム」等を考えた場合, 一般の人工知能プログラム同様, 応答の解析, 状況に関する情報の保持・更新の処理に, LISP のリスト構造は大いに力を発揮すると思われる.

3. 試作システム

本章では, 試作した LISP 端末について,

(1) システム構成
(2) LISP 処理系
(3) 端末制御プログラム

の順に説明する.

3.1 システム構成

ハードウェアとして, MZ-80 B* を, ソフトウェア (モニタ) としては, MZ-CP/M 2.2 A** を用いた. LISP 処理系のサイズは約 12 kB, 8080 用のアセンブリ語で記述してあるが, 一部, Z80 の命令も使用している.

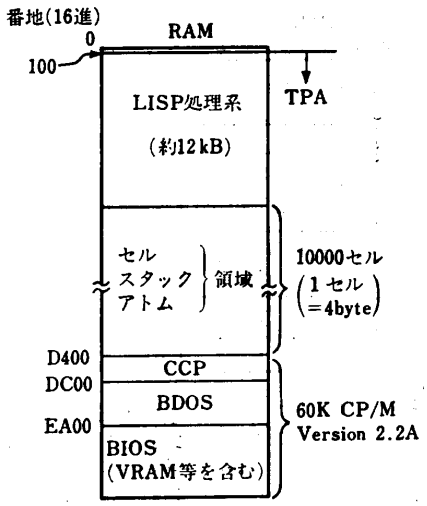

図 4 メモリ・マップ
Fig. 4 Memory map.

使用した CP/M は Z80-SIO* を完全にはサポートしていないので,

- SIO 初期化 (割込設定, バッファ・クリア)
- ブレーク信号発生
- 受信割込処理
- 環状バッファ (127 byte) による受信処理

のルーチンを LISP 処理系に埋め込んである. また通信速度は 300 ボーに設定してある.

3.2 LISP 処理系

以下のような基本部分は, 文献 9) の MINI-LISP に準じている.

(1) 数アトム, 文字値アトムの内部表現 (図 5)
(2) セル, スタック, アトム領域の割付け (図 5).
(3) アトム (印字名, 値) の構造 (図 6).
(4) ゴミ集めの方式 (逆転ポインタ法).
(5) 基本組込み関数とその実現方法 (表 2).

MINI-LISP と異なる点は次のとおりである.

(6) トップレベルは EVAL.
(7) 関数 PROG は FSUBR として実現.
(8) 関数 COND の実行部には複数個の S 式を許す.
(9) 文字アトムは 256 個 (ASCII コード) 使用可能.
(10) 「'」(≡QUOTE),「'"文字列"」の導入.

---

* シャープ(株)製. CPU は Z 80 A (4 MHz), RAM (64 kB), タイマー, サウンド, CRT (10 インチ), キーボード, フロッピーディスク (280 kB×2), プリンタ (80 char/sec), RS 232 C インタフェース (2 channel) を装備.

** マイクロソフトウェア・アソシエイツ (株) 製 (シリアル番号 2-348-00268). CP/M はディジタル・リサーチ社の商標である.

* 通信制御用の Z 80 周辺 LSI (SIO は Serial Input Output の略). シャープからは, インタフェース・カード MZ-8 BI 03 として供給される. 以下で「RS 232 C インタフェース」と呼ぶのはこれのことである.

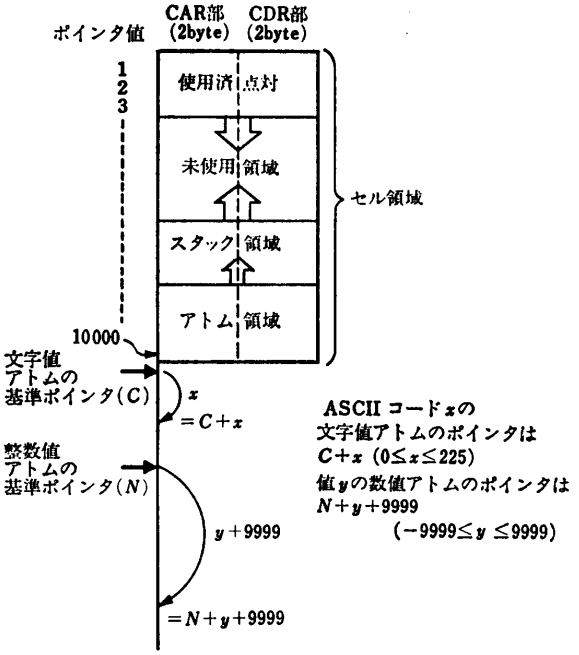

図 5 ポインタ値とセル領域
Fig. 5 Pointer values and cell area.

(11) 「[」,「{」,「〈」を「(」の代りに，また「]」,「}」,「〉」を「)」の代りに使用することもできる．

(12) ESC キーによる割込み，割り込んだ状態でのS式の評価，セルの消費状況表示，ゴミ集め，トップレベルへの復帰，割り込む直前の状態への復帰ができる．

この他に，パソコン周辺機器，CP/M の機能を利用するため，**表 3** に示すような関数を備えている．

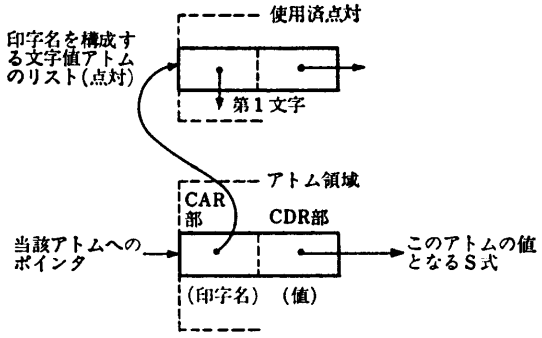

図 6 アトムの内部表現
Fig. 6 Internal representation of ATOMs.

3.3 端末制御プログラム

端末制御プログラムは次のような関数で構成され，全体として図2の流れ図を実現している．

(1) TERM : CRT 画面クリア，F1 キー（ファンクションキー1番）の設定*，開始音およびメッセージ出力．COMMUNICATION を呼び出す．

(2) COMMUNICATION : RS 232 C インタフェース初期化・オペレータのキー入力を待ち，入力を検知したらブレーク信号を送出（オペレータは回線接続後 F1 キーを押下する）．以後，RECEIVE, SEND の呼出しを繰り返す．

(3) RECEIVE : ベルコード (16 進 07) を検知するまで，F1 キーの押下をセンスしながら1文字ずつ受信文字をバッファから取り出し，表示 (印字) する．F1 キー押下の場合はブレーク信号送出．ベルコードを受信したら，終了音を発生して戻る．

(4) SEND : キーボードより一行入力し，先頭文字を調べる．先頭が「↓」ならば I-EVAL を呼んで行の残りを評価する．この後，以上の動作を再び行う．先頭文字が 16 進の1A (F1 キーに対応づけられた2番目のコード) であれば**，ブレーク信号を送

表 2 組込み関数
Table 2 Ready made functions.

引数の評価 / 引数の数	関数評価の前に引数を評価する	引数を評価しない
0	READ READ1 RSERV TERPRI	
1	PRINT PRINT 1 INTP ATOM CAR CDR NULL EVAL	QUOTE
2	CONS RPLACA RPLACD SET EQ A+ A− A∗ A/ APPLY GTP LTP	PROG2 SETQ
3	—	DEFUN (FEXPR)
不定個	LIST APPEND (EXPR)	AND OR COND PROG RETURN GO

* 16 進で，「18, 1A, oD」のコード列を対応させる．コード 18 (コントロールキー＋X) は行入力キャンセル (再入力待ち) の効果をもつ (CP/M の機能)．次の脚注参照．

** 送信 (入力) 中に F1 キーを押下すると，コード 18 によって，そこまでの入力がキャンセルされ，2番目のコード 1A が先頭文字として取り込まれ，次の oD で入力終了となる．

表 3 試作処理系に特有の関数
Table 3 Characteristic functions of the trial LISP Processor.

	関数名と引数（の例）	引数の数	動作	返す値
通信関係	(*LINZ)	0	割込みルーチン，SIO装置，受信バッファを初期設定する	NIL
	(*SD 1 文字値アトム)	1	与えられた文字値アトムに対応する ASCII コードを送出する	NIL
	(*PV 1)	0	受信バッファの先頭に在る ASCII コードを対応する文字値アトムに変換して返す	文字値アトム
	(*RX ?)	0	受信バッファに受信文字が存在する (T) か否か (NIL) を調べて T/NIL を返す	T/NIL
	(*BRK)	0	ブレーク信号を送出する	NIL
タイマー・サウンド	(*BELL 数アトム1 数アトム2)	2	引数で指示された高さ，長さの音を出す	NIL
	(S*TIME 時 分 秒)	3	CP/M の管理する時計の時刻を設定する	NIL
	(R*TIME)	0	上記時計の時刻を読み出す	時分秒のリスト
ファイル出力	(NEW "[ドライブ名：] ファイル名")	1	指定された名前のファイルを新規に作成し，書き込みのできる状態とする	NIL
	(OLD "[ドライブ名：] ファイル名")	1	指定された名前の既存のファイルの最終データに続けて書き込みを行う状態とする	NIL
	(CLOSE)	0	NEW/OLD で開いたファイルを閉じる	NIL
	(*PUTC 文字値アトム)	1	NEW/OLD で開いたファイルに1文字書き出す	NIL
ファイル入力	(DIN "[ドライブ名：] ファイル名")	1	指定された名前のファイルの先頭データから順に読込みの出きる状態とする	NIL
	(*GETC)	0	DIN で開いたファイルから1文字読込み，対応する文字値アトムを返す	文字値アトム
	(*EOF ?)	0	DIN で開いたファイルのデータが尽きた (T) か否か (NIL) を調べて T/NIL を返す	T/NIL
ディレクトリ	(DIR ["ドライブ名："])	0～1	指示されたドライブ中のフロッピーディスクの全ファイル名をリストにして返す	ファイル名のリスト
	(RENAME "新名前" "旧名前")	2	名前の付け変えを行う	NIL
	(ERASE "[ドライブ名：] ファイル名")	1	指定されたファイルを消去する	NIL
文字値関係	(CHARP any)	1	数が文字値アトムか (T) 否か (NIL) を調べ T/NIL を返す	T/NIL
	(ASC 文字値アトム)	1	引数の文字値アトムに対応する ASCII コードを数アトムに変換して返す	数アトム
	(CHR$ 数アトム)	1	引数の数アトムを ASCII コードとみなし，それに対応する文字値アトムを返す	文字値アトム
システムの入力割当て	(INPUT "[ドライブ名：] ファイル名")	1	LISP 処理系の入力先を指定された名前のファイルに切り替える（ファイルが尽きるとキーボード入力に戻る）	NIL
	(IPT-KB)	0	LISP 処理系の入力先をキーボードに切り替える	NIL
	(FKEEP-IPT-KB)	0	現在入力中であるファイル名，位置を保持したまま，入力先をキーボードに切り替える	NIL
	(FRESUME)	0	上記 (FKEEP-IPT-KB) で中断したファイルからの入力に戻る	NIL
	(IPT-EOF ?)	0	ファイルから入力中の場合，データが尽きた (T) か否 (NIL) か調べ T/NIL を返す	T/NIL
	(IPT ?)	0	現在の入力先がキーボード (0) かファイル (1) か，ファイル留保中 (2) か調べて各値を返す	0/1/2
実行制御	(CATCH S式 tag)	2	S式を評価する（次の THROW によって評価が強制終了させられることもある[10]）	
	(THROW S式 tag)	2	等しい tag をもつ CATCH の評価を終了させ，第1引数のS式を結果として返す[10]	
	(ERSET S式)	1	S式を評価する．正常に評価し終わった場合には結果をリストにしたものを，エラーが起こった場合には NIL を返す[10]	

	関数と引数（の例）	引数の数	動作	返す値
補助的関数	(*TYP 文字値アトム)	1	文字値アトムに対応する文字を画面に表示する	NIL
	(*GKB)	0	キーボードから1文字入力する（入力があるまで待つ）	文字値アトム
	(*GKBC)	0	キーボードから1文字入力する（入力がない場合は NULL コードを返す）	〃
	(PGRE {0,1,2} [{0,1,2} [{0,1,2} [{0,1,2}]]])	1～4	P：画面へ表示する際印字も行う．G：ゴミ集めの際，セルの使用状況を表示する．R：結果の表示．E：システム入力がファイルの場合，入力列を画面に表示する．いずれも 0：表示(印字)せず 1：表示する．2：現在のまま	NIL
	(PGRE ?)	0		({0,1} {0,1} {0,1} {0,1})
	(PROMPT T/NIL)	0	トップレベル入力の際，プロンプト記号を出力する (T) か否 (NIL) か指示する	NIL
	(CALL-GBC)	0	ゴミ集めルーチンを陽に起動する	NIL
	(REG-ATOM '(*ab. c*))	1	(例) abc をアトムとして登録する	アトム
デバッグ・編集	(TRACE '(*f*1 *f*2……))	1	関数 *f*1, *f*2, … の追跡を指示する[9]	NIL
	(UNTRACE '(*f*1 *f*2……))	1	追跡を解除する[9]	NIL
	(LED '*f*)	1	関数 *f* の編集，ファイルへの格納を行う[11]	*f*
	(EXIT)	0	CP/M システム（除 BIOS）を再ロードし，コマンドモードに戻る	—

出して戻る．それ以外の場合は，行中の各文字を順に送出して戻る．

（5） I-EVAL：ERSET の下で，EVAL を用い，「↓」より後の S式を評価する．システム入力がフロッピーディスクを割り当てられた場合は，アトム「*EOF*」を検知するまで評価をくり返す．

この他に若干の補助関数を必要とするが，端末制御プログラム自体の規模は，きわめて小さく済んでいる．

4. 使用例

つぎに，LISP 端末を用いた処理の例を示す．

4.1 関数の登録と引用

作成済みのプログラム（関数群）を蓄積しておき，必要に応じてそれを引用する．個々のプログラムの機能は BASIC 端末と同等であるが，新しいプログラムを順次登録していくことにより，「端末制御プログラム」が自然に充実していく．試作システムのライブラリ中には，端末機能に関係するものとして，

- PON, POFF：プリンタの on/off 等．
- R/DATA, X/DATA, X/COM：ホストのエディタを経由したファイルの転送．

```
)
)+(TYPE "B:GCDLCM.PAS")
  program  GCDLCM2(input,output);
  var      A,B,C,D,M,X,Y:integer;
  begin  read(X,Y);
     while (X)0) and (Y)0) do
        begin  A:=X;  B:=Y;
           repeat  C:=A mod B;  A:=B;  B:=C  until B=0;
           D:=A;
           M:=X*Y div D;
           writeln('G.C.D. of (',X,',',Y,')= ',D);
           writeln('L.C.M. of (',X,',',Y,')= ',M);
           read(X,Y)
        end
  end.
==NIL
)+(X/DATA "M.PASCAL(GCDLCM)" "B:GCDLCM.PAS")

B:GCDLCM.PAS ---) M.PASCAL(GCDLCM)

  1)  2)  3)  4)  5)  6)  7)  8)  9)  10)  11)  12)  13)  14) 0
-------------------END OF FILE---
==NIL
)
)LIST M.PASCAL(GCDLCM)
00010   program  GCDLCM2(input,output);
00020   var      A,B,C,D,M,X,Y:integer;
00030   begin  read(X,Y);
00040      while (X)0) and (Y)0) do
00050         begin  A:=X;  B:=Y;
00060            repeat  C:=A mod B;  A:=B;  B:=C  until B=0;
00070            D:=A;
00080            M:=X*Y div D;
00090            writeln('G.C.D. of (',X,',',Y,')= ',D);
00100            writeln('L.C.M. of (',X,',',Y,')= ',M);
00110            read(X,Y)
00120         end
00130   end.
)TIME
TIME=14:48:21 C-TIME=00:00:01 SERVICE=5003 E-TIME=00:19:58 DATE=84-03-08
)
```

図 7 関数 X/DATA の使用例➡
Fig. 7 Example of using the function X/DATA.

```
>
>LISTC
IN CATALOG:SYS1.CENTCAT.IPC
R2A4016.A.PLI
R2A4016.D.DATA
R2A4016.EDITSAVE.PASCAL
R2A4016.EIAMPLE.FORT
R2A4016.KN.PASCAL
R2A4016.KN.P3
R2A4016.N.PASCAL
R2A4016.P2.DATA
R2A4016.P2.PASCAL
R2A4016.P3KN.OBJ
R2A4016.P3KN.PASCAL
R2A4016.PXLIB.PASCAL
R2A4016.SYSPROF
R2A4016.TEST.COBOL
R2A4016.XYZ.PASCAL
END OF JSCVSUT, MAXCC IS 0
>((DIR "B:")
==(B:)
>((GT/ALL "R2A4016" '(COBOL FORT PLI) )
++++++++++++++ START ++++++++++++++++++++++++++++++++++++++++++++++++++

A.PLI(P01) ---> B:AMPLI.001

  1< 2< 3< 4< 5< 6< 7< 8< 9< 10< 11< 12< 13< 14< 15< 16<
----------------------END OF FILE---

A.PLI(P02) ---> B:AMPLI.002

  1< 2< 3< 4< 5< 6< 7< 8< 9< 10< 11< 12< 13< 14< 15< 16<
 35< 36< 37< 38< 39< 40< 41< 42< 43< 44< 45< 46< 47< 48< 49< 50<
----------------------END OF FILE---

EIAMPLE.FORT(MAIN) ---> B:EIAMPLE.FORT.003

  1< 2< 3< 4< 5< 6< 7< 8< 9< 10< 11< 12< 13< 14< 15< 16< 17< 18< 19< 20<
----------------------END OF FILE---

EIAMPLE.FORT(SUB1) ---> B:EIAMPLE.FORT.004

  1< 2< 3< 4< 5< 6< 7< 8< 9< 10< 11< 12< 13< 14< 15< 16< 17< 18< 19< 20< 2.
 35< 36< 37< 38< 39< 40< 41< 42< 43<@
----------------------END OF FILE---

EIAMPLE.FORT(SUB2) ---> B:EIAMPLE.FORT.005

  1< 2< 3< 4< 5< 6< 7< 8< 9< 10< 11< 12< 13< 14< 15< 16< 17< 18< 19< 20< 21<
 35< 36< 37< 38< 39< 40< 41< 42< 43< 44< 45< 46< 47< 48< 49< 50< 51< 52< 53< 54< 55<
----------------------END OF FILE---

TEST.COBOL ---> B:TESTACOBOL.006

  1< 2< 3< 4< 5< 6< 7< 8< 9< 10< 11< 12< 13< 14< 15< 16< 17< 18< 19< 20< 21
 35< 36< 37< 38< 39< 40< 41< 42< 43< 44< 45< 46< 47< 48< 49< 50< 51< 52< 53< 54< 55<
 69< 70< 71< 72< 73< 74< 75< 76< 77< 78< 79< 80< 81< 82< 83< 84< 85< 86< 87< 88<@
----------------------END OF FILE---

++++ END ++++++++++++++++ 6 DATA SETS XFERED. ++++++++++++++++++
==NIL
>((DIR "B:")
==(B: AMPLI.001 AMPLI.002 EIAMPLE.FORT.003 EIAMPLE.FORT.004 EIAMPLE.FORT.005 TESTACOB.006)
>
```

図 8 関数 GT/ALL の使用例

Fig. 8 Illustrative use of the function GT/ALL.

- GT/ALL：当該ユーザのホスト上の全登録データセット（指定された属性をもつもの）の内容を転送し CP/M のファイルとして登録する．ファイルの名前はホスト上での名前をもとにして自動生成する．

等が入っている．X/DATA と GT/ALL の利用例を，図 7，8 に示す．

4.2 オンラインデバッグ

TSS のセッション中に，パソコン側でプログラムを作成，編集，実行できるということは，通信処理を含むモジュール（関数）のデバッグを行う上で大いに役立つ．たとえば，図 9 では，受信コードを 16 進数でも表示するよう，関数 RECEIVE を変更している．

```
>
>TIME
TIME=12:54:51 C-TIME=00:00:04 SERVICE=12631 E-TIME=01:07:25 DATE=84-03-09
>↓(LED 'RECEIVE)    関数 RECEIVE の編集
| P    上から2レベルまで印字
(LAMBDA NIL (PROG @ @ LOOP @ @ @))
| 3 6 P    対象を第3要素の第6要素に移す.
(@TYP C)
| 0 P      現在の対象を第1レベルの要素として含むS式に対象を移す.
(PROG (C) (@RV1) LOOP (COND @ @ @ @) (@TYP C) (GO LOOP))
| (-7 (SETQ C (HEXLH C)) (PRIN1 47C) (PRIN1 (CDR C)) (PRIN1 (CSAR C)) (PRIN1 47C))    第7要素の前に以下のS式を挿入する.
| P
(PROG (C) (@RV1) LOOP (COND @ @ @ @) (@TYP C) (SETQ C @) (PRIN1 /) (PRIN1 @) (PRIN1 @) (PRIN1 /) (GO LOOP))
| OK    編集を有効にしてLEDから脱出
==RECEIVE
>TIME
/0F/T/54/1/49/N/4D/E/45/=/3D/1/31/2/32/:/3A/5/35/7/37/:/3A/1/31/4/34/ /20/C/43/-/2D/T/54/1/49/N/4D/E/45/=/3D/0/30/0/30/:/3A/0/30/0/30/:/
3A/0/30/4/34/ /20/S/53/E/45/R/52/V/56/1/49/C/43/E/45/=/3D/1/31/2/32/6/36/7/37/0/30/ /20/E/45/-/2D/T/54/1/49/N/4D/E/45/=/3D/0/30/1/31/:/3
A/0/30/9/39/:/3A/4/34/9/39/ /20/D/44/A/41/T/54/E/45/=/3D/0/30/4/34/-/2D/0/30/3/33/-/2D/0/30/9/39/
/08//0F/>/3E//0F/↓(LED 'RECEIVE)
| 3 P
(PROG (C) (@RV1) LOOP (COND @ @ @ @) (@TYP C) (SETQ C @) (PRIN1 /) (PRIN1 @) (PRIN1 @) (PRIN1 /) (GO LOOP))
| (7)(7)(7)(7)(7)      第7要素以降の5個のS式を削除する.
| P
(PROG (C) (@RV1) LOOP (COND @ @ @ @) (@TYP C) (GO LOOP))
| OK
==RECEIVE
>TIME
TIME=12:58:56 C-TIME=00:00:04 SERVICE=12730 E-TIME=01:11:30 DATE=84-03-09
>
```

図 9 セッション中の編集例（関数 RECEIVE の一時的変更）
Fig. 9 Example of on-line editing (temporary modification of function RECEIVE).

5. むすび

以上，LISP 処理系をベースとしたパソコン知能端末—LISP 端末—を提案し，その特長，構成法，使用例を報告した．そこでは，LISP 端末が柔軟性，拡張性，デバッグのしやすさの点で，確かに優れていることを明らかにしたが，反面，速度の点では「もう一つ」であることも報告しておかなくてはならない（これは，本研究用に試作した LISP 処理系の未熟さも大きな要因であるが，一般に，柔軟性と処理速度の両立はむずかしい）．また，パソコン諸機能のサポート（画面エディット，グラフィックス）に関しても，試作システムは，通常の BASIC 処理系に比べ見劣りがする．しかしながら，この速度の低さと諸機能の不備は致命的ではなく，処理系の改良（再設計，コンパイラ方式の導入，キメ細かな諸機能のサポート）等によって十分克服できるものと考え，今後の課題としたい．

ともかく，現在の速度（伝送速度 300 ボーには十分追随しうるが，1200 ボーには追いつかない）でも，「プログラムの作成・試行・変更を中心とする作業，およびそれに伴う情報管理」といった利用環境では十分実用になるし，またその柔軟性，拡張性から受けるメリットは，非常に大きいと判断する．

謝辞 日頃ご討論いただく花田孝郎氏，ご指導・ご鞭撻下さる牛島照夫先生，小林光夫先生に感謝いたします．また，貴重なコメントを下さった査読者に，お礼申し上げます．

なお，本研究は文部省科学研究費（奨励 A，No. 59740097）の援助を受けた．

参考文献

1) 石田晴久：インテリジェントな TSS 端末としてのマイクロコンピュータ，情報処理学会マイクロコンピュータ研究会資料，79-9 (1979).
2) 西本，石田：マイコン PC 8001 と IF 800 を TSS 端末にするための BASIC プログラム，東大・大型計算機センターニュース，Vol. 13, No. 4, pp. 25-32 (1981).
3) 西本史雄：パーソナルコンピュータをインテリジェント TSS 端末にする端末制御プログラムの作り方，東大・大型計算機センターニュース，Vol. 13, No. 12, pp. 33-38 (1981).
4) 市川伸一：オフライン・エディタ付 TSS 端末としてのパソコンの利用，東大・大型計算機センターニュース，Vol. 14, No. 7, pp. 38-42 (1982).
5) 石田晴久：マイコンをインテリ端末にするための FORTRAN プログラム，東大・大型計算機セ

ンターニュース, Vol. 12, No. 10, pp. 46-55 (1980).
6) 山本, 川端：割り込み機能を使用した CP/M ターミナルプログラム, 東大・大型計算機センターニュース, Vol. 14, No. 11, pp. 27-41 (1982).
7) 石田晴久：TSS 端末とマイコン端末の選び方, 東大・大型計算機センターニュース, Vol. 14, No. 12, pp. 17-21 (1982).
8) 平野正信：出そろった多機能, 複合パソコン, 日経コンピュータ, No. 44, pp. 49-65 (1983).
9) 後藤, 戸島, 石畑：記号処理の基礎と応用（情報処理叢書 8), p. 126, 情報処理学会（オーム社), 東京 (1982).
10) 黒川利明：LISP 入門, p. 309, 培風館, 東京 (1982).
11) 安西, 佐伯, 難波：LISP で学ぶ認知心理学 2, p. 244, 東京大学出版会, 東京 (1982).

(昭和 58 年 11 月 1 日受付)
(昭和 59 年 6 月 19 日採録)