柔軟に機能を定義できるスクリーン・エディタについて

岡山理科大学・理学部　馬野　元秀
Motohide Umano

1．はじめに

計算機の前に座っている時間のほとんどは，エディタを使っている時間であるとよくいわれるように，エディタはユーザと計算機との最も重要なインタフェースである．したがって，最近のディスプレィ端末の進歩にあいまって，使いやすさの向上のためにスクリーンをフルに利用したスクリーン・エディタが数多く作成されている[1,2]．

しかし，機能のキィーへの割り付け方がエディタごとに異なり，ユーザがそのエディタを使いこなすにはキィーの割り付け方に十分慣れる必要がある．しかし，キィーに機能を自由に割り付けられるスクリーン・エディタがあれば，ユーザの好みや慣れに合ったキィー割り付けが可能となり，憶えやすく，使いやすいものとなる．実際，ユーザがキィーに機能を割り付けることが可能なエディタもある[3,4]が，ごく一部のキィーにしか割り付けることができなかったり，多くのキィーに割り付けができても，その手続きが非常に面倒であったりすることが多い．

そこで，本発表ではキィーに機能を容易かつ自由に定義できるスクリーン・エディタについて述べる．まず，第2節では本エディタの基本的な部分の設計について述べた後，キィー定義の考え方を例により説明する．そして，第3節で基本命令について述べ，第4節で基本命令を組み合わせてできるマクロ命令とそのキィーへの定義について述べる．さらに，第5節で標準エディタのキィー定義について述べる．インプリメントについても，第5節で簡単に触れる．

なお，本スクリーン・エディタは，VAX-11/780 のVAX/VMS 上の Ratfor（ごく一部はアセンブラ）で記述され，DEC の端末VT100 とNEC のパソコン PC-100 に対して稼動している．

2．エディタの概要

まず，本節ではスクリーン・エディタの基本的な部分について述べた後，キィー定義の考え方について例を中心に述べよう．

2.1. 基本設計

スクリーン・エディタを設計するときに，まず考えておくべきことがある．それらは，

(1) 画面には常に何を表示するか．
(2) カーソルはどの範囲に移動可能か．
(3) テキストはどのようにして入力するか．
(4) コマンドはどのようにして入力するか．
(5) 同時に編集できる範囲はどれくらいか．

などであろう．(1)－(4)に対して，本エディタでは以下のように決めた．

(1)については，画面全体でファイルの一部を表示するようにし，画面の下部や上部にコマンド入力領域などは設けない．これは，編集時にできるだけ多くのテキストを見えるようにするためとスクロール・アップやスクロール・ダウンのときにディスプレィ端末の機能を利用しやすくするためである．必要なときのみ，画面の下部に入力領域やメッセージ領域を設け，不必要になれば，テキストの表示に戻すようにする．

(2)については，カーソルはテキストの部分にしか移動できないものと画面全体のどの場所にでも移動できるものが考えられる．本エディタでは，垂直方向のカーソル移動時の視覚的追随性を考え，画面のどの場所にでも移動できるようにする．

(3)については，すべての印字可能キィーによりその文字をカーソル位置の左に挿入するか，その文字でカーソル位置の文字を書き換えるようにする．そのため，挿入モード／書き換えモードの区別を行なう．挿入モードでは，画面の1行の最大文字数を越えては入

力できないようする．行の終わり（行の最後の文字の１つ右）にカーソルがあるときは，どちらのモードでもその文字を挿入するようにする．また，カーソルが行の終わりより右にあるときは，どちらのモードでも行の終わりからその位置の前まで空白を補ってから，その文字を挿入するようにする．これは表などの作成を考慮したからである．

(4)については，CTRL キィー，ESC キィーを使用する．我々が想定している端末では，ファンクション・キィーおよび別モード時の補助キィーボード（テン・キィー）はESC で始まる文字列を発するのでこのESC 系列も考慮する．パラメータを必要とするコマンドもCTRL キィーと ESC キィーで起動してから，画面の下部に入力領域を作り，そこからパラメータのみを読み込むようにする．

(5) については，ファイルの一部をバッファに取り出し，それを順に編集していくものと，ファイルのどの部分でも自由に編集できるものがある．後者は使いやすいが，ファイルの処理が多くなり応答速度が遅くなる危険性がある．しかし，ここではユーザの使いやすさを考えて，後者にする．インプリメントにおいて，直接編成ファイル上の双方向リストを使用して高速化を図る．さらに，画面の行数＋２行分だけの画面用バッファを用いるが，行をファイルに戻すのはその行が画面用バッファから外れ，かつ，修正がある場合にのみ行なう．

2.2. キィー定義

本エディタでは，印字可能キィー以外のすべてのキィーに対してユーザは自由に機能を定義することができる．機能の定義は，スクリーン・エディタに必要な機能をできるだけ簡単な形に分解した「基本命令」を組み合わせることにより行なう．

基本命令は，

英字　　＜英字　　＾英字

のいずれかの形をしており，＜と＾は「逆」，「否定」などの意味をできるだけ表わすようにしてある．

基本命令の組み合わせ方はできるだけ簡単に定義できることを考え，Wangのロボット言語[5,6] 風の記述にしてある．

例をあげて具体的に述べよう．

［例１］ １文字消去

基本命令として，

D: カーソル位置の１文字を削除する
＜K: カーソルを１つ左に動かす

があるので，これらを使って，

```
DEL = D
 BS = <KD
```

とすると，DEL キィーには「カーソル位置の１文字を削除する」を定義でき， BS キィーには「カーソル位置の１つ左の１文字を削除する」を定義できる．

もちろん，いままで使用してきたエディタで，DEL キィーを「カーソル位置の１つ左の１文字を削除する」として使い慣れているユーザは，上の定義とは逆に，

```
DEL = <KD
 BS = D
```

とすればよい．

［例２］ ページ・スクロール

１行分のスクロールを行なう基本命令，

U: 画面を１行スクロール・アップさせる
^U: 画面を１行スクロール・ダウンさせる

を使って，&U（CTRL U ）キィーと &D（CTRL D ）キィーにページ単位のスクロール機能として，１画面分に当たる 24 行のスクロールを割り付けるには，

```
&U = 24U
&D = 24^U
```

とすればよい．もし，画面の半分のスクロールにするならば，

```
&U = 12U
&D = 12^U
```

とすればよい．

［例３］ RETURN キィー

RETURN キィー（&M キィー）はスクリーン・エディタでは，いろいろに定義される．

例えば，次の行の先頭への単なるカーソル移動と考えるならば，

```
<E: カーソルを行の先頭へ移動する
 V: カーソルを１つ下に動かす
```

という基本命令を使って，

```
RETURN = <EV
```

とすればよい．しかし，RETURNキィーを「行の終わり」という文字を挿入すると考えて，カーソル位置の前で行を分割する命令と考えるならば，

```
<L: カーソル位置の左で行を２行に分割する
```

という基本命令を使って，

```
RETURN = <L
```

とすればよい．また，プログラミング言語によっては，プログラムを入力するとき字下げをしたい．このときは，

```
RETURN = <EWV
```

とすればよい．ここで，W は

```
W: カーソルを次の単語の先頭の文字に移す
```

という基本命令である．したがって，このとき RETURN キィーを押すと，次の行にカーソルが移動し，現在の行の最初の英数字と同じ位置までカーソルが移動した状態になる．

３．基本命令

2.2 節で述べたように，基本命令はスクリーン・エディタに必要な機能をできるだけ簡単な形に分解したもので，

```
英字    < 英字    ^ 英字
```

のいずれかの形をしている．< と ^ は「逆」，「否定」などの意味をできるだけ表わすようにしてある．

現在，基本命令は表1 に示すように 57 個あり，これらは次の７つに分類できる．

(1) モードの変更に関する命令
(2) 削除と挿入に関する命令
(3) カーソル移動に関する命令
(4) ファイルに関する命令
(5) 探索と置換に関する命令
(6) アキュムレータに関する命令
(7) その他の命令

以下，この分類に従って，順に基本命令を説明していこう．

(1) モードの変更に関する命令

本エディタには，現在のところ「挿入／書き換え」というモードしかなく，印字可能な文字のキィーに対して，挿入モードならばカーソル位置の前にその文字を挿入し，書き換えモードならばカーソル位置の文字をその文字で書き換えるという働きをする．ただし，行の終わり又はそれより右にカーソルがあるときには，モードとは無関係に（必要ならば空白を補って）その文字を挿入する．これに関係する基本命令には，N（挿入モードにする），<N（書き換えモードにする），^N（挿入モードと書き換えモードとを反転させる）の３つがある．

(2) 削除と挿入に関する命令

多くのスクリーン・エディタでは，文字や単語や行を削除すると，それが削除されると同時に対応するバッファにコピーされ，必要に応じてそれをテキストに復帰できる．スクリーン・エディタには，この機能は文字列の移動，削除ミスの回復などの機能として不可欠であるが，基本機能としてはバッファへのコピーと削除は分けておく方が便利である．

表１．基本命令一覧表

A: アキュムレータの主ポインタを１つ下げる．
^A: アキュムレータの主ポインタを１つ上げる．

B: アキュムレータの副ポインタを１つ下げる．
^B: アキュムレータの副ポインタを１つ上げる．

C: カーソル位置の文字を文字バッファに入れる．
<C: カーソル位置に文字バッファの内容をもどす．
^C: 文字バッファの内容をアキュムレータに移す．

D: カーソル位置の１文字を削除する．
^D: カーソル位置から行の最後までの文字列を削除する．

E: カーソルを行の終わりへ移動する．
<E: カーソルを行の先頭へ移動する．
^E: カーソルのある行がファイルの最終行か？

F: カーソルをファイルの最後の行へ移動する．
^F: カーソルをファイルの先頭の行へ移動する．

G: カーソル位置から行の最後までの文字列を行バッファに入れる．
<G: カーソル位置に行バッファの内容をもどす．

H: 基本命令一覧表を表示する．
<H: キィーに定義されたマクロ命令とコメントの一覧表を表示する．

I: アキュムレータにカーソルの縦方向の画面座標を入れる．
<I: カーソルの縦方向の画面位置をアキュムレータの示す座標にする．

J: アキュムレータにカーソルの横方向の画面座標を入れる．
<J: カーソルの横方向の画面位置をアキュムレータの示す座標にする．

K: カーソルを１つ右に動かす．
<K: カーソルを１つ左に動かす．

L: カーソルのある行と次の行とを結合する．
<L: カーソル位置の左で行を２行に分割する．

M: キィーにマクロ命令を割り付ける．
<M: マクロ命令のキィーへの割り付けをすべて解除する．

N: 挿入モードにする．
<N: 書き換えモードにする．
^N: 挿入モードと書き換えモードとを反転させる．

O: ファイルを開いて，それを新たな編集対象にする．

P: タブ位置を設定する．
^P: 印字後のカーソルの動く方向を設定する．

Q: 選択されている文字列を探索バッファに入れる．
<Q: 選択されている文字列を置換バッファに入れる．

R: 選択されている文字列を置換バッファの内容で置き換える．
<R: 置換バッファの内容を設定する．

S: ファイルの終わり方向へ探索バッファの内容を探索する．
<S: 探索バッファの内容を設定する．
^S: ファイルの始め方向へ探索バッファの内容を探索する．

T: タブ設定に従って右方向にカーソルを動かす．
<T: タブ設定に従って左方向にカーソルを動かす．
^T: カーソルのある行がファイルの先頭行か？

U: 画面を１行スクロール・アップさせる．
^U: 画面を１行スクロール・ダウンさせる．

V: カーソルを１つ下に動かす．
<V: 何もしない．
^V: カーソルを１つ上に動かす．

W: カーソルを次の単語の先頭の文字に移す．
<W: カーソルを１つ前の単語の最後の文字に移す．

X: カーソル移動キィーにより文字列を選択する．
<X: 画面をリフレッシュする．

Y: アキュムレータの内容を探索バッファの文字列に追加する．
<Y: アキュムレータの内容を置換バッファの文字列に追加する．
^Y: アキュムレータの内容を文字バッファに入れる．

Z: 現在開いているファイルの処理を終了する．

そこで，基本命令 D（カーソル位置の1文字を削除する）および C（カーソル位置の文字を文字バッファに入れる）と^D（カーソル位置から行の最後までの文字列を削除する）および G（カーソル位置から行の最後までの文字列を行バッファに入れる）がある．そして，テキストへの復帰のために<C（カーソル位置に文字バッファの内容をもどす）と<G（カーソル位置に行バッファの内容をもどす）という基本命令がある．なお，現在のところ，単語の削除と挿入に関する基本命令はない．

RETURN を印字可能文字と考えて，RETURN の削除と挿入により行を結合したり，分割したりするエディタも多いが，本エディタでは RETURN は印字可能文字とは考えない．その代わりに，L（カーソルのある行と次の行とを結合する）および<L（カーソル位置の左で行を2行に分割する）という基本命令を使う．これはRETURN キィーを単なるカーソル移動キィーとして使用したいユーザのためである．マクロ命令を使って，RETURNを印字可能な文字と考えたときの動作を実現することは可能である．

(3) カーソル移動に関する命令

カーソル移動の基本は，上下左右への1文字分のカーソル移動である．このために，K（カーソルを1つ右に動かす），<K（カーソルを1つ左に動かす），V（カーソルを1つ下に動かす），^V（カーソルを1つ上に動かす）という基本命令がある．

左，右へのカーソル移動では，カーソルが行の先頭，画面の行の終わりにあるときには，ベルを鳴らし，カーソルを移動しない．また，上，下へのカーソル移動では，画面の最上行，最下行にカーソルがあるときには，それぞれスクロール・ダウン，スクロール・アップを行なう．もちろん，ファイルの先頭行，最終行のときは，ベルを鳴らし，カーソルは移動させない．

スクロール・アップとスクロール・ダウンは，カーソルを画面の最下行または最上行に移動させて，下または上にカーソルを移動させても実現できるが，機能としてはより基本的なので，U（画面を1行スクロール・アップさせる）と^U（画面を1行スクロール・ダウンさせる）を基本命令に入れてある．

また，単語単位のカーソル移動の基本命令として W（カーソルを次の単語の先頭の文字に移す）と<W（カーソルを1つ前の単語の最後の文字に移す）がある．単語の定義にはいく通りか考えられるが，ここでは英数字の列としている．

そして，E（カーソルを行の終わりへ移動する）と<E（カーソルを行の先頭へ移動する）および F（カーソルをファイルの最後の行へ移動する）と^F（カーソルをファイルの先頭の行へ移動する）という基本命令がある．

さらに，タブ設定に基づくカーソル移動として T（タブ設定に従って右方向にカーソルを動かす）と<T（タブ設定に従って左方向にカーソルを動かす）という基本命令がある．タブの設定はPという基本命令を使って行なう．これは，画面の下部に窓をつくり，現在のタブを表示するので，→キィー，←キィーでタブの変更場所に移動させ，↑キィーで設定，↓キィーで解除を行なう．なお，タブの初期設定は72文字目までが8文字ごとで，以降は1文字ずつになっている．

最後に，表を作成するのに便利な基本命令がある．普通，印字可能なキィーを押すとその文字を挿入するか，その文字で書き換えて，カーソルはその文字の1つ右に移る．しかし，表などを作成するときは，その文字の下に移動する方が便利である．このときに使うのが^Pという基本命令である．これにより，印字後のカーソルの位置を上下と左右と斜め4個所と印字した文字の場所の計9個所のうちのいずれかに移すことができる．

(4) ファイルに関する命令

最初に編集するファイルはエディタを実行させたときに指定するが，あるファイルを編集中にさらに別のファイルを開いて，編集することができる．そして，そのファイルの編集を終えると，そのファイルを開く直前と同じ状態になるので，前のファイルの編集の続きを行なうことができる．これは，編集対象のファイルがスタック構造をしていると考えればよい．この機能は分割コンパイルが可能な言語のプログラムの編集などに非常に有用である．

まず，ファイルを開き，それを新たな編集対象にする基本命令が０である．この基本命令は，画面下部に窓を作り，ファイル名の入力を促す．もちろん，システムに存在しないファイル名を指定すると新たにそのファイルを作成する．

一方，現在，編集しているファイルの編集が終わったので，そのファイルを閉じて，１つ前のファイルに編集対象を戻すには，Ｚという基本命令を用いる．これは，画面下部に窓を開いて，終わり方をきいてくるが，このとき，３種類の終わり方がある．すなわち，

(a) 存在しているファイルに上書きする
　　… RETURN のみを入力
(b) 新たにファイルを作成する
　　… ファイル名を入力
(c) 編集結果を無効にしファイルを作らない
　　… / を入力

の３種類である．この場合，(a) を標準と考えて，簡単な入力で済むようにしてある．

(5) 探索と置換に関する命令

大きなファイルを編集するときには，探索の機能は必須である．探索時に探索用のパターンを入力させ，そのパターンを探索し，それ以降は同じパターンでも探索できるという機能をもったエディタが多い．しかし，本エディタでは，探索バッファ（探索用のパターンをいれておくところ）へパターンを入力する基本命令Ｓとそのパターンに従ってファイルの終わり方向へ探索を行なう基本命令Ｓとファイルの始め方向へ探索を行なう基本命令 ^S に分けている．

探索のパターンは正規表現を可能とする．これは，デバッグ時にはあまり必要としないかもしれないが，プログラムのデバッグ後に書法の統一を図りたいときなどには必須である．探索パターンの表記法は，文献[7] に従っている．まとめると以下のようになる．

?: 改行符号を除く任意の文字
%: 行の先頭
$: 行の終わり
[…]: …の文字のうちの任意の１文字
[~…]: …の文字以外の１文字
*: 直前パターンの０回以上の繰り返し
@: 脱出記号

ここで，[…] または [~…] の中の…には，単なる文字列以外に「文字 - 文字」の形の文字列を書くことができ，前の文字から後ろの文字までのすべての文字を表わす（例えば，a-z はすべての英小文字を表わす）．

テキストを探索して，パターンにマッチする文字列を見つけると，その文字列全体の白黒を反転し，その文字列の先頭にカーソルを移動する（以下，画面で白黒が反転した文字列は選択されているということにする）．マッチした文字列が画面内にあるときは，そのまま画面内の文字列の白黒を反転し，カーソルを移動させるだけだが，画面内にないときは，マッチした文字列を含む行が画面の中央にくるように画面をすべて書き直す．

選択されている文字列は置換の対象になる．置換は基本命令 R（選択されている文字列を置換バッファの内容で置き換える）により行なわれる．そして，置換バッファに文字列を入力するには <R という基本命令を使えばよい．置換文字列は普通の文字列であるが，& により選択されている文字列を表わすことができる．

また，探索を使わずに文字列を選択することも可能である．このための基本命令がＸで，カーソル移動キィーにより文字列を選択することができる．ただし，選択できる文字列は１行のなかの文字列だけで，複数行に渡る文字列を選択することはできない（これは探索でも同じ）．そして，このようにして選択した文字列を探索バッファまたは置換バッファに移すことができる．これは Q（選択されている文字列を探索バッファに入れる）または <Q（選択されている文字列を置換バッファに入れる）という基本命令により行なわれる．これにより，画面上のテキストから，探索パターンや置換文字列を取り出すことができる．

(6) アキュムレータに関する命令

アキュムレータに関する基本命令は，全部で 14 個あるが，アキュムレータの概念はマクロ命令に関連しているので，次節のマクロ

命令のところで述べる．

(7) その他の命令

その他の命令には，実用上重要なものが多い．これらは，マクロ命令，ヘルプ機能，画面のリフレッシュ，ノー・オペレーションに関するものである．

まず，マクロ命令に関する基本命令として，M（キィーにマクロ命令を割り付ける）と<M（マクロ命令のキィーへの割り付けをすべて解除する）がある．ただし，キィーにマクロ命令を割り付ける方法については，3.2 節で詳しく述べる．

次に，ヘルプ機能の基本命令には，H（基本命令一覧表を表示する）と<H（キィーに定義されたマクロ命令とコメントの一覧表を表示する）の２つがある．基本命令の一覧表は決まったファイルの内容を書かせるだけであるが，マクロ命令の一覧表は現在定義されているものをコメント付きで表示する．

画面のリフレッシュは <X という基本命令で行なう．これはエディタを使用しているときに他のユーザからメッセージが送られて，画面が乱されたときなどに使う．

最後に，ノー・オペレーションに当たる基本命令として<V がある．

４．キィー定義とマクロ命令

前節では基本命令について述べた．本節ではキィー定義とマクロ命令について述べよう．

4.1. キィー定義

キィー定義は基本命令M を実行することにより行なわれる．この基本命令を実行すると，画面下部に窓をつくるので，

キィー記号 = マクロ命令

と入力すると，キィー記号で表わされるキィーへマクロ命令が割り付けられる．したがって，これ以降，このキィーを押すと対応するマクロ命令が実行されることになる．ここで，キィー記号は，&，@，$ の後ろに英字を並べたもので，&，@，$ は

&: CTRL

@: ESC

$: ESC [又は ESC O 又は ESC 1

を表わす． ESC [と ESC O は，DEC の端末 VT100 のファンクション・キィーと別モードの補助キィーボード（テン・キィー）が発するESC 系列であり，ESC 1 はユーザが入力しやすいように考えたESC 系列である．これらはすべて同じキィー記号に対応するので，いずれのESC 系列でも同じマクロ命令を実行することになる．キィー記号の制約からすると，2.2 節の例１ のBS，DEL という表記と例３ の RETURN という表記は，実際には上記のキィー記号を使って書き直す必要がある．

基本命令M はマクロ命令のキィー記号への割り付けが済むと，ユーザにコメントの入力を促す．そして，マクロ命令とコメントを組にして憶えておいて，基本命令H が実行されると，マクロ命令とともにコメントも表示するようになっている．

4.2. マクロ命令

マクロ命令はWang のロボット言語[5,6]風に記述する．マクロ命令を作るとき，

(1) 順次実行
(2) 固定回数の繰り返し
(3) アキュムレータの使用
(4) ?[ ][ ] による選択

を使って記述できる．以下，順に述べよう．

(1) 順次実行

命令を単に並べると，左から順に実行する．基本命令を順に実行する例は2.2 節で例をいくつか示した．しかし，並べることのできるのは，基本命令だけではなく，' 文字列 '，" 文字列 "，キィー記号，そして，(2)- (4) も並べることができる．ここで，' 文字列 ' と " 文字列 " はモードにより文字列をカーソル位置に挿入するか又はカーソル位置からその文字列で書き換えるかする．また，キィー記号を書くとそのキィー記号に定義されているマクロ命令を実行する．これは，手続き呼び出しに対応すると考えられる．

(2) 固定回数の繰り返し

符号のない整数を命令の前に付けると，その回数だけ命令を繰り返して実行する．基本命令を繰り返して実行する例は2.2節の例2で示した．ここで注意すべきことは，繰り返し指定は次の命令にしか有効でないことである．したがって，複数の命令を繰り返したいときには，命令列を( )でくくる必要がある．

(3) アキュムレータの使用

ロボット言語では，アキュムレータと呼ばれるメモリを1個だけもっていて，簡単な計算や不定回数の繰り返しを実現している．ここでは，少し拡張してアキュムレータを配列とする．しかし，参照と操作についてはかなりの制限を設ける．すなわち，主ポインタと副ポインタという2つのポインタを考え，参照はこれらの指している所だけに限定し，操作は主ポインタの指している所だけに限定する．

いま，主ポインタを＊で表わし，副ポインタを％で表わすとすると，より具体的には，繰り返し指定のところに＊と％を書くことができ，主ポインタの指している所に「1を加える」という演算＋と「1を引く」という演算－を可能にする．これら＋と－は基本命令と同じレベルで書くことができる．

これらを使って，例えば

```
*-: アキュムレータをクリアする
%+*v: (＊の内容＋％の内容)行だけカーソルを下へ移動する
```

と書ける．なお，単にアキュムレータというときは，主ポインタの指している所を指すものとする．

ここで，3節で説明できなかったアキュムレータに関する基本命令について述べよう．

まず，主ポインタを動かすために，A(主ポインタを1つ下げる)と^A(主ポインタを1つ上げる)という基本命令がある．また，副ポインタについても同様で，B(副ポインタを1つ下げる)と^B(副ポインタを1つ上げる)がある．

また，アキュムレータでは＋，－の演算を使って，簡単な計算ができるので，各種バッファとアキュムレータ間でデータを移動させるための基本命令がある．これらは，^C(文字バッファの内容をアキュムレータに移す)および^Y(アキュムレータの内容を文字バッファに入れる)とY(アキュムレータの内容を探索バッファの文字列に追加する)および<Y(アキュムレータの内容を置換バッファの文字列に追加する)である．

また，画面のカーソル位置をアキュムレータに取り込む基本命令として，I(アキュムレータにカーソルの縦方向の画面座標を入れる)とJ(アキュムレータにカーソルの横方向の画面座標を入れる)がある．そして，これの逆を行なう基本命令として，<I(カーソルの縦方向の画面位置をアキュムレータの示す座標にする)と<J(カーソルの横方向の画面位置をアキュムレータの示す座標にする)がある．

(4) 選択

条件による簡単な分岐を記述できる．

```
?[ 命令列1 ][ 命令列2 ]
```

により，アキュムレータが正ならば命令列1を，正でなければ命令列2を実行する．もちろん，命令列の中にも選択を書けるので，入れ子も可能である．

これと関連した基本命令がある．それは，^T(カーソルのある行がファイルの先頭の行か？)と^E(カーソルのある行がファイルの最後の行か？)である．これは条件が成立すればアキュムレータに1を，成立しなければ0を入れるので，?[ ][ ]を使って実行を制御できる．

キィー定義も含めたマクロ命令の構文は図1の構文図のようになる．

5．標準エディタ

3節では基本命令について，4節ではキィー定義とマクロ命令について述べた．

キィーにマクロ命令を定義をしておいて，ユーザがそのキィーを押すと，対応するマクロ命令をインタプリタが実行する．しかし，最初，キィーに何も定義されていないならば，

エディタにはいるたびに，キィーに使いたい機能を定義しなければならない．これでは，まったく使いものにならない．

そこで，基本命令0を実行するときに，編集対象のファイルを開いて，読み込むだけではなく，ファイルからキィー定義とそのコメントを読み込むようにする．これは，編集対象のファイル名の最後に- を付けるとエディタはキィー定義ファイルの名前をきいてくるので，キィー定義ファイルの名前を指定すればよい．しかし，もし- がなければ，決まった名前の標準キィー定義ファイルを読み込む．この標準キィー定義ファイルは図2に示すようなものである． ここで，$0 というキィー定義はこのファイルを読み込んだらすぐに自動的に実行してしまうキィー定義である．

この標準キィー定義ファイルによるエディタ（標準エディタ）はVAX/VMS のEDT エディタと同じように補助キィーボードを利用する．補助キィーボードの部分のキィー割り付けは図3のようになる．また，ESC M にキィー定義用の基本命令が割り付けられているので，これを使ってエディタの中で，キィー定義を変更したり，追加したりできる．

そして，キィー定義は，エディタを終了するとき（基本命令Z を実行するとき）に，標準ファイル又はユーザの指定したファイルに出力することもできる．そして，次にエディタを使用するときにそのファイルを使うわけである．このようにして，標準キィー定義ファイルを自分が使いやすいように改良していける．また，編集対象に合わせて，いろいろなキィー定義ファイルを利用できるようにもなる．

インプリメントについては，スクリーン・エディタとしての基本機能と基本命令を実現している核部分と，キィー定義されたマクロ命令を実行するインタプリタに分れる．核部

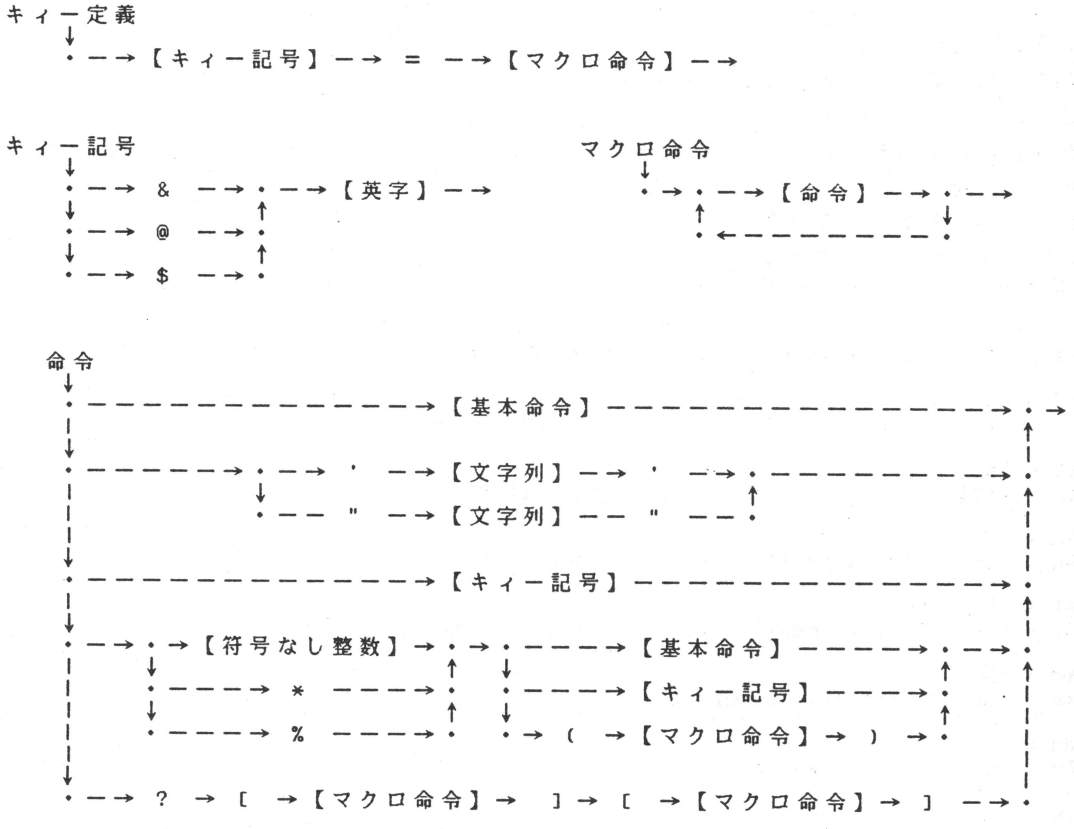

図１．キィー定義とマクロ命令の構文図

```
$0 = N
$0 = esc [ 0: automatic initialization command

&A = L
&A = ctrl A: append the next line to cursor line

$A = ^V
$A = esc [ A (^): move cursor up

$B = V
$B = esc [ B (v): move cursor down

@C = C
@C = esc C: copy a character to the character buffer

$C = K
$C = esc [ C (->): move cursor right

&D = ^D
&D = ctrl D: delete to end of line

$D = <K
$D = esc [ D (<-): move cursor left

&F = O
&F = ctrl F: open file

@F = <MO
@F = esc F: clear macro area and open file

&G = G
&G = ctrl G: copy characters to the line buffer

@G = <G
@G = esc G: restore the line buffer

&H = <KCD
&H = ctrl H (BS): delete the previous character

@H = H
@H = esc H: help macros

$H = <H
$H = esc [ H: help primitives

&I = T
&I = ctrl I (TAB): move cursor to the next tab stop

&J = <N
&J = ctrl J (LF): switch insert and overwrite modes

&L = E<L
&L = ctrl L: create a line after cursor line

@L = <E<L^V
@L = esc L: create a line before cursor line

$l = <E<L^V<G<E
$l = esc O l (,): create a line and restore the line buffer

&M = <L
&M = ctrl M (RETURN): separate line

@M = M
@M = esc M: define macro

$M = <EG^D^E?[L][^VL]
$M = esc O M (ENTER): delete line and copy to the line buffer
```

図２．標準キィー定義ファイル

```
$m = ^AJ<EG<JAV
$m = esc O m (-): copy a line to the line buffer

$n = <C
$n = esc O n (.): restore the character buffer

$P = ^F
$P = esc O P (PF1): move cursor to top of file

$p = CD
$p = esc O p (0): delete a character

$Q = <S
$Q = esc O Q (PF2): get search pattern

$q = <W
$q = esc O q (1): move cursor to the previous word

@R = <X
@R = esc R: refresh screen

$R = S
$R = esc O R (PF3): search

$r = W
$r = esc O r (2): move cursor to the next word

@S = X
@S = esc S: select characters

$S = R
$S = esc O S (PF4): replace

$s = ^U
$s = esc O s (3): scroll down

&T = <T
&T = ctrl T: move cursor to the previous tab stop

$t = <E
$t = esc O t (4): move cursor to beginning of line

$u = E
$u = esc O u (5): move cursor to end of line

$v = U
$v = esc O v (6): scroll up

&W = Q
&W = ctrl W: get search pattern from select

@W = <Q
@W = esc W: get replace characters from select

$w = F
$w = esc O w (7): move cursor to end of file

$x = <R
$x = esc O x (8): get replace characters

$y = ^S
$y = esc O y (9): back search

&Z = Z
&Z = ctrl Z: close file
```

図２．標準キィー定義ファイル（続き）

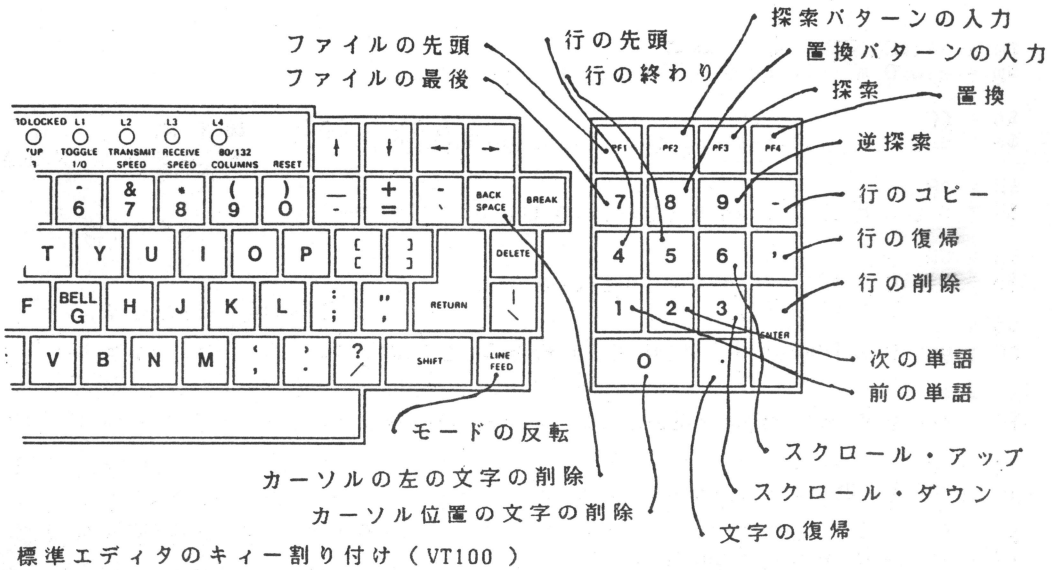

図３．標準エディタのキィー割り付け（VT100）

分は，以前にpdp-11/45 用に作成したスクリーン・エディタ[8]をVAX-11/780 に移植し，改良したものである．インタプリタは新たに作成した．実現にはVAX-11/780 の VAX/VMS 上で Ratfor を用いた（ごく一部はアセンブラである）．概念的に核部分とインタプリタというように大きく２つの部分に分れているので，実現は比較的容易であった．なお，使用端末はDEC のVT100 である．したがって，標準キィー定義ファイルのキィー割り付けはこの端末向けである．また，最近は，NEC のパソコンPC-100 でも使用可能になっている．さらに，他の端末装置への移植も容易であると思われる．

６．おわりに

以上，キィーに機能を容易かつ自由に定義できるスクリーン・エディタについて述べた．

本エディタにより，ユーザは自分の好みにあったスクリーン・エディタを定義して利用することができるだけでなく，編集対象に合わせたエディタも使用できるようになる．

基本機能の中には，必要でないものもあるだろうし，もっと重要なものが欠落しているかもしれない．今後は，使用経験に基づいて基本命令などの強化を図っていきたい．

最後に，本エディタの実際のインプリメントは，当時本学の学生であった，笹川敬章君，広田直之君，下拂美奈子さん，高尾朋子さんによる所が大きい．深く感謝する．

［参考文献］

1. 「特集 エディタ」，情報処理，Vol.25, No.8 (1984).
2. N. Meyrowitz and A. van Dam, "Interactive Editing Systems : Part I, II", ACM Computing Surveys, Vol.14, No.3, pp.319-352; pp.353-415 (1982).
3. 斉藤：「拡張可能な画面エディタEMACS」，情報処理，Vol.25, No.8, pp.777-789 (1984).
4. 安川：「YALE-TOOLS の画面エディタZ」，情報処理，Vol.25, No.8, pp.790-799 (1984).
5. L. Wang : "An Interactive Programming Language for Cotrol of Robots", Dr. Dobb's Journal, Vol.2, Issue 10, pp. 60-63 (1977).
6. 石田：「ロボット言語で怪物曲線を描こう」，bit, vol.15, No.10, pp.1102-1109 (1983).
7. B. W. Kernighan and P. J. Plauger : Software Tools, Addison-Wesley, Reading, USA (1976).
8. 馬野：「文字ディスプレィ用スクリーン・エディタの設計と作成」，情報処理学会第 23 回（昭和56 年後期）全国大会，pp.259-260, No.7I-5 (1981)．