スクリーン・エディタVEDの開発・移植と問題点

横上 昭男
（電子技術総合研究所）

内片 秀樹
（松下電器産業）

1. はじめに

対話形テキスト・エディタ（以下エディタ）は，オンライン・デバッグなどとともに対話的なプログラミング・システムにおけるもっとも基本的な道具の一つであり，その使い勝手の良し悪しが，文書化なども含めたプログラム開発作業の能率とかなり左右することはよく知られている．最近では，どのような小規模のシステムにおいても，そのシステムでプログラム作成を行う限り一定水準以上のエディタを設置することは当然のこととされており，カードやテープをもとにした編集作業は見られなくなってきた．

エディタの中でも，編集されるテキストの様子が常時表示されるスクリーン・エディタは，一旦使いはじめるとよほどのことが無い限り従来のエディタを使う気にさせないほどなじみ易いものであり，最近は文字表示端末の低価格化もあって普及の兆を見せはじめている．しかしながらこのようなエディタを実現するにあたっては，端末制御のために計算機システム側の低レベルI/Oに手を入れることが必要であったり，端末ごとの制御手順の差異のために特定の端末以外での使用が困難であるなどの問題点もあった．

我々は，現在使用中のTSSの一つ（PRIME/PRIMOS）上に適当なスクリーン・エディタが無かったこと，エディタとしては不完全ではあるが土台にできる特定端末用のスクリーン・エディタが有ったこと，[1] などの理由から新しくスクリーン・エディタの開発を行った．このシステムの開発目標は，次の通りである．

(1) 日常よく用いる機能以外の，比較的特殊な機能は排除し，できるだけコンパクトなシステムとする．
(2) コマンド体系はできるだけ使い易いものとし，カーソル移動によるコマンド・パラメタの入力などもできるだけ許すようにする．
(3) 実際上困らない程度の種類の端末を利用可能にする．また簡単な手続きで，エディタの使用開始時点に使用端末を定義できるようにする．
(4) ある一定の条件を満たせば，別のシステムへの移植が可能なようにできるだけ配慮する．

我々の所では，小形計算機上に開発したセミスクリーンエディタをこれまで8年以上にわたって使用してきており，[2] また最近は別のTSS上（DEC-20/TOPS-20）で，エディタの一つの極致ともいえるEMACS[3]も利用できる状況にある．これらの経験をできるだけ生かし，通常のプログラムや英文の編集作業は，どのシステムに切換えても，EMACS等のかわりに，同一のエディタを用いて作業が行える環境を造り出すことが，このエディタを開発する目的の一つである．

2. スクリーン・エディタの種類と考え方

表1は筆者の周辺で実際に使用されているか，少くともおよその外部仕様を入手することのできるスクリーン・エディタを示している．これらは，使用者から見たコマンド・インターフェイスの形によって次の3つに分類できる．

(1) コマンドには特別な区切り文字以外制御文字を用いないシステム：

TV がその代表的なもので，挿入テキスト等は通常のライン・エディタと同様にコマンド・パラメタとして与えられる．基本的には，通常のエディタに画面指向のベリファイ機能を付加したものであると考えられる．制御コードなコマンドやパラメタの区切りにのみ用いられる．この方式のシステムは，

a)ライン・エディタと同一のコマンド体系にすることができる，

b)コマンド実行前の確認が常に可能で，その取止めも容易である，

c)制御コードの使用法に対する制限があまり無い，

などの特長を持っているが，もっとも頻繁に行われるカーソル移動や，テキストの挿入または打直しに必要なキー・ストローク数が多くなる欠点がある．

(2)画面制御用に制御文字と合わコマンドや，座標入力装置等を用いるシステム：

これはこれまでもっとも多く試みられている方式で，表1の中の SCOPE や vi をはじめ，GOLEX, UCSD, PERQ などのシステムもこれに属する．機能的には種々の水準のものがあるが，いずれも前項の方式に比較して，

a)カーソル移動やスクロールなどの画面制御が簡便化され，

b)テキストの挿入や修正途中の様子を入力時に常に観察できる，

などの特長を持っている．けれども，

c)コマンド・モードとテキストの挿入または打直しモードを混乱し易い，

d)制御文字に関する自由度が少くなる，

e)挿入や打直しのために入力中のテキスト自身を自由に編集修正することが困難である，

などの欠点もある．c)の問題に関しては，とくに入力促進テキスト等を表示しないほうが簡潔で良いとする立場もあるが，これには批判も有り[10]，多くのエディタが画面上に何らかの状態表示領域を設けることで解決しようとしている．またd)の問題は，制御文字をテキストとして扱うための特別なコマンドを設けるなどの方法で，実際上支障が生じないようにできる．結局e)項が，

表1．スクリーン・エディタの代表例

エディタ名	記述言語	OS	文献
ADAM	FORTRAN	PRIMOS	(1)
ED/GOLEX*	アセンブラ	GOLEX	(2)
EMACS*	アセンブラ	TOPS-20	(3)
TV*	アセンブラ	TOPS-20	(4)
SCOPE*	アセンブラ	CP/M	(5)
VIDED	SIMULA	TOPS-20	(6)
vi	C	UNIX	(7)
UCSD Pascal Screen Oriented Editor*	PASCAL	UCSD Pascal OS	(8)
PERQ Text Editor*	PASCAL	PERQ OS	(9)
TED*	FORTRAN	PRIMOS, TOPS-20,...	本論文

(*印は，筆者およびその周辺で使用中のもの．)

次に述べる方式のエディタと基本的に異る点であるといえよう．

(3)コマンド入力に必ず制御文字やファンクション・キーを用いる方式：

この方式は，ここで述べるVEDをはじめ，EMACS, VIDED などでとられている方式で，非制御文字（プリント可能文字）の入力は原則としてすべて編集テキストと置換えるか，これに付加されるデータとして扱われる．したがってコマンド・モードとテキスト・モードの区別は基本的に存在せず，カーソル位置のみを手掛りに，常に修正可能なタイプライタを打つような感覚で入力を行うことが可能となる．ある程度好みの問題もあるが，挿入あるいは修正のために入力中のテキストもただちに編集対象となるなど，エディタとしてもっとも使い勝手の良いものにできる可能性を有している．欠点としては，

a)制御文字がほとんどコマンドとして用いられるため，制御文字自身をテキストとして扱うのに工夫を要する．また制御文字の自由な使用を許すOSでないと実現できない，

b)コマンドの多くは1～2回の打鍵でただちに実行されてしまうので，重大な結果をもたらす可能性のあるコマンドに対しては，確認やundo機能等による打誤り対策が必要となる．

スクリーン・エディタのその他の操作面での差異としては，

i) PERQエディタのように，キーボード以外の座標入力装置によってカーソルを動かす機能が有るか否か，

ii)画面表示サイズ（文字数）や，画面の更新速度の問題，

iii)メッセージの丁寧さ（terse vs. verbose system），

などが有る．

PERQ やGOLEXのように，計算機の内部記憶の一部を直接表示用に用いる方式のものは，画面上の表示文字数や，画面の更新頻度などの面で自由度が大きく，今後高水準のパーソナル・コンピュータの普及とともに，この方式のものが広く用いられるようになるものと考えられる．

この他に機能的な面での差異として，

iv)何種類ものテキスト・ウィンドウを定義して，同時表示可能かどうか，

v)拡張性や，フォーマット機能，あるいは特定言語向きのチェック機能が組込まれているかどうか，さらにエディタの中から自由にOSの他の機能に直接アクセスすることが許されるかどうか，

などがある．前者に関しては，EMACSのように通常のCRT端末を用いる場合でもある程度その機能を備えたものもあるが，一般的な形で実現するためには高解像度ビット・マップ・ディスプレイを利用可能なことが必要である．これまでいわゆるスーパー・パーソナル・コンピュータ上で，エディタも含めた高水準のプログラミング・システムとして試みられているが[11]，本格的な用途は今後に残された問題である．

最後の機能は，スクリーン・エディタのみに関連する問題ではないが，良いプログラミング環境を実現するという観点からは重要で，種々の側面からの検討が必要な事項である．これを特定言語向きに設定したものは，対話的なプログラム合成器ともいうべきものに近づいてゆくことになる．[12]

3. VEDの構造と機能

3.1 設計方針と内部構造

先に述べたように，我々の目標は適度な使い易さを有する，比較的コンパクトで移植の容易なシステムの実現にあった．ハードウェア環境としては，十分大きなデータ領域を（仮想記憶により）提供できる計算機システムと，これに無手順非同期回線で接続される

標準的なCRT端末（表示画面サイズ80コラム×24行）を想定している．

図1は本システムの編集ウィンドウの考え方を示したもので，使用者は上下，および左右のスクロール・コマンドにより，このウィンドウを編集テキスト上の任意の位置へ移動し，その中でカーソル位置を適宜設定してテキストの修正，挿入等を行う．ウィンドウの移動量は，行数やコラム数などの数値で与えることも，カーソル・コマンドを用いてカーソル移動量により指定することも可能である．

VEDではコマンド入力とテキスト入力を制御文字で始まるか否かで区別しており，数値や検索文字列をパラメタとして入力する場合以外は，制御文字以外の入力テキストはそのままカーソル位置の文字を書きかえる．いいかえれば，鉛筆と消しゴム，および挿入場所を作るためのハサミと糊を用いて編集を行うイメージである．

システムの記述は主としてFORTRANと，一部低レベルI/Oおよび文字処理を標準FORTRAN外のシステム関数，またはアセンブラー・ルーチンを用いて行っている．編集されるテキストは，図2のように左スタックと右スタックに分割して蓄えられており，それぞれカーソル以前，およびカーソル以降のテキストを表わしている．

3.2 コマンド形式と使い易さの実現

図3は本システムのコマンド形式を示したものである．上述のようにVEDのコマンドは基本的には制御コードで構成されており，できるだけ操作性を良くするために，各コマンドの割付けには次のような配慮を行った．

(1)カーソル移動やスクロール制御など使用頻度の高いコマンドは，1ストロークで実行可能にする．

(2)［ESC］+［プリント文字］の形のコマンドは，原則として入出力や雑命令に，それ以外を編集命令に割当てる．

(3)コマンドの意味と，そのコマンドに割付けたキー・トップの文字とこじつけて憶え易くする．

例）

［↑S］--- Search

［↑R］--- Replace

［ESC］+ W --- Save World/Write

（↑はコントロール・キーを表す）など．ただし標準的な割付けの明確なコードは，これを優先させる．

例）［↑I］=［TAB］

［↑H］=［Back_Space］など．

(4)逆の意味を持つコマンド対は，一方に［ESC］を付加することで実現する．

例）［↑S］（順方向Search）と

［ESC］+［↑S］（逆方向Search）．

［↑E］（行末－End of Line）と

［ESC］+［↑E］（行の先頭）など．

またコマンドの割付け以外に関しても，

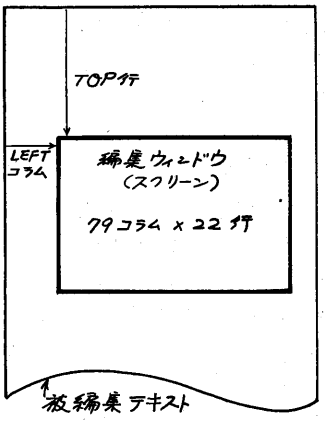

図1 編集ウィンドウ

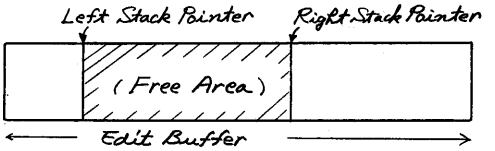

図2 編集バッファの構造

```
                Command ::= No_Param_Command | Command_with_Parameter

        No_Param_Command ::= [Control_Code] | [ESC] + [Any_Code]

  Command_with_Parameter ::= [[ESC]] + "string" + No_Param_Command |
                             [[ESC]] + "string" + No_Param_Command
                                                + "string" + [Term_Code] |
                             [[ESC]] + <Cursor_Movement> + No_Param_Command |
                             [[[ESC]]] + [Special_Character]

             [Term_Code] ::= [ESC] | [^Z] | [^W]

                 [[ESC]] ::= [ESC] + [ESC]

               [[[ESC]]] ::= [ESC] + [ESC] + [ESC]

    [Special_Character] ::= [Blank] | ! | " | # | $ | % | & | ' | ( | .....

                      + ::= Reads followed by.
```

図3 VEDのコマンド形式

(5)できるだけシステムが何をやっているか表示する(verbose system). 図4はVEDの画面構成を示したものであるが, 表示可能な24行のうち常に2行を状態表示に割当て, ファイルの入出力など時間のかかる処理では, システムのやっていることを最下段に一時的に示すようにしている.

(6)入力テキストは, 既存テキストを打直すことを基本状態とする. これはEMACSのデフォルトが挿入モードであるのと対比されるものである. 打直しと挿入のいずれの機会が多いかは一概にはいえないけれども, 回線接続された端末を使用する場合には, 画面の書替えの機会をできるだけ少くして, 応答を早くすることも重要である.

VEDではテキストの挿入は, 引数付挿入コマンドなどで行うこともできるが, 普通は空白を設けるコマンドで作った空白部分にテキストを打込むことを基本操作としている.

(7)コマンドの引数は, 数値の入力のほか, カーソル移動による入力も可能にする(ここから, あそこまで等).

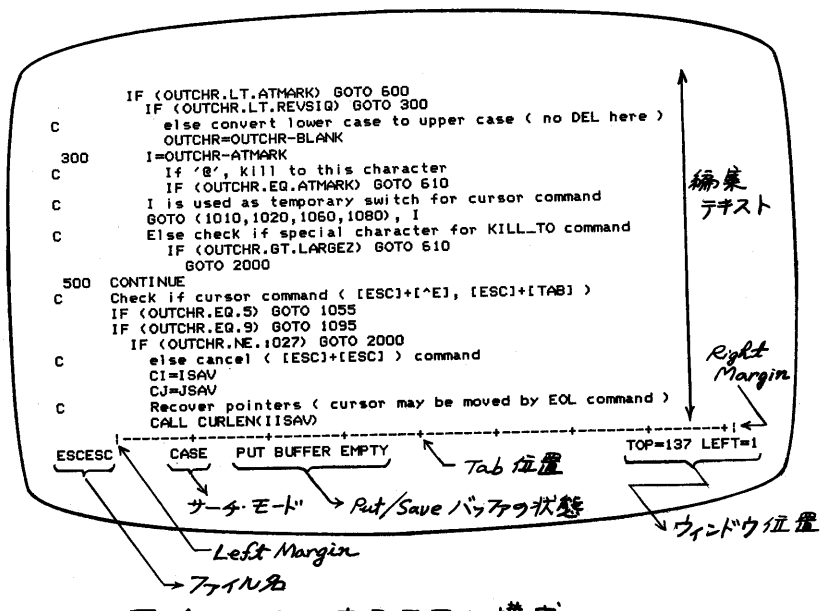

図4 VEDの表示画面の構成

3.3 端末間の差異の吸収

最近は端末も非常に多くの種類が出まわり，一つのシステムに多種類の端末が接続されることも多くなってきた．ここで問題なのは，端末の制御コマンド（制御コード・シークエンス）がまちまちで統一されていないことである．これを逃れるために我々のとった対策は，

(1)端末のハードウェア機能は，必要最小限のものしか用いない，

(2)使用端末の型をエディット開始時に使用者に必ず指定させる．プログラムは以後必要に応じてこの型により定まる端末制御コマンドを出力する．

具体的に利用する端末のハードウェア機能としては，

a) カーソルの絶対アドレス指定，
b) カーソル以降行末までの消去，
c) カーソル以降の画面消去，または画面全体の消去，

の3種類で，このうち最初の2つは必須であるが，最後の項目は端末に備わっている場合のみ利用する．

したがって挿入機能を持つ端末を用いる場合でも，挿入操作ではカーソル以降の消去と書直しが行われることになり，その意味では端末のインテリジェント化に逆行する方式となっている．しかしながら本システムでは，1字毎の挿入動作はあまり頻繁には行われないので，端末間の差異を小さくすることを優先させた．

3.4 ホスト・システムに要求される機能

このシステムを動作させるうえでホスト計算機側に要求される機能としては，

(1)原則として制御コードを含むすべてのコードをユーザ・プログラム内で取扱うことが可能であること，

(2)エコーバック，フィル・コードなどの制御をユーザ・プログラムから行うことが可能なこと（回線属性の制御）．

(3)ユーザ・プログラムから1文字単位の入出力を行うことが可能なこと（OSが余分なコードを付加したり，要求したりしない）．

(4)全2重（非同期）回線に端末を接続可能であること．

(5)十分大きな記憶領域を使用できること（編集可能なテキストの大きさ）．

これらは必ずしも現在世の中で使用されている計算機システムに標準的に備わっている機能であるとはいえないが，本システムに限らず使い易い対話形システムを実現するための基本媒体が最低限満たすべき条件のうちのいくつかであり，そのような目的の計算機システムの導入にあたっては，一般に留意すべき事項であると考えられる．

4. むすび

以上スクリーンエディタを設計するうえで，特に操作性の面での考え方，および比較的コンパクトなエディタの開発例について報告した．

エディタ自身の機能については，拡張性や，その他の高レベルの機能について考慮したい面もいくつか有るが，本システムでは日常の編集作業での使い勝手を損わない範囲でできるだけコンパクトなシステムにすることを設計指針としており，これらの機能の多くは省略されている．

エディタに関しては個人の好みの問題も有って，一概にどれか一つの方式が良いとすることは困難であるが，操作性の問題をできるだけ定量的にとらえて使い易い対話形システム実現のための基礎データにしようとする努力も行われている[13]．この報告も，このような方向での研究や，これから類似のシステムの開発を行う場合の参考になれば幸いである．

最後に，御検討いただいた大石泰作，河越正弘氏ならびに情報システム研究

の方々と熱心なユーザの方々に厚く感謝いたします.

参考文献

(1) Kittlaus, E.: ADAM Users Manual, Management Inf. Sys., Orange County Transit District, California (1980).

(2) 佐藤孝紀：オンライン・テキスト・エディタ および コア・エディタ, 昭49 電子通信学会全国大会.

(3) Stallman, R.M.: EMACS Manual for TWENEX Users, AI Memo 555, MIT (Sept. 1980).

(4) TOPS-20 TV Editor Manual, Digital Equipment Co. (Jan. 1980).

(5) SCOPE — Screen Oriented Program Editor for CP/M, Vector Graphic Inc. (Feb. 1980).

(6) Palme, J.: VIDED — A Display Oriented Text Editor, National Defence Res. Inst., Sweden (1977).

(7) 石田晴久：どんなCRT端末からでも使える画面エディタ (VE), bit, Vol. 13, No. 14, pp. 1778-1783 (1981).

(8) Pascal MICROENGINE Reference Manual, The Microengine Co., Newport Beach (Nov. 1979).

(9) Strait, J.P.: Editor V1.9 Quick Guide, Three Rivers Computer Co., Pittsburgh (1981).

(10) Norman, D.A.: The Trouble with UNIX, Datamation, Vol. 27, No. 12, pp. 139-150 (1981).

(11) Teitelman, W.: A Display Oriented Programmer's Assistant, 5th IJCAI, pp. 905-915 (1977).

(12) Teitelbaum, T., etal.: The Cornell Program Synthesizer: A Syntax Directed Programming Environment, CACM, Vol. 24, No. 9, pp. 563-573 (1981).

(13) Embley, D.W., etal.: Behavioral Aspects of Text Editors, Computing Surveys, Vol. 13, No. 1, pp. 33-70 (1981).

付録　VED コマンド表の一部 (Help File の一部)

```
3. Window/Page Handling ( Scroll Control )
   VED displays the text through 79_column_by_22_line window on the
file. The window can be moved vertically as well as horizontally using
one of the following commands. The position of the window is shown at
the status line ( TOP=nn LEFT=mm ).
                        [^L] :  Scroll forward ( scroll up ) 10 lines.
          [[ESC]]+<n>+[^L] :  Scroll forward n lines.
       [[ESC]]+<curs>+[^L] :  Scroll current line to cursor position.
                 [ESC]+[^L] :  Scroll backward ( scroll down ) 10 lines.
    [[ESC]]+<n>+[ESC]+[^L] :  Scroll backward n lines.
                        [CR] :  Scroll 1 line if at the bottom of screen.
         [ESC]+[ or [ESC]+< :  Skip to TOF.
         [ESC]+] or [ESC]+> :  Skip to EOF.
                        [^V] :  Skip to the next page.
          [[ESC]]+<n>+[^V] :  Skip to the next nth page.
                              ( Equivalent to scroll 22*n lines ).
                  [ESC]+[^V] :  Skip back to the previous page.
    [[ESC]]+<n>+[ESC]+[^V] :  Skip back to the previous nth page.
          [[ESC]]+<n>+[^W] :  Slide the window n columns horizontally.
                              ( If n<0, the window moves leftward ).
       [[ESC]]+<curs>+[^W] :  Slide current column to cursor position.
```

(次頁に続く)

4. Inserting Text

VED always stays in type-in/replace mode, which means anything you type in goes into the file you are editing except control characters. A character at cursor position will be replaced by a new character if it is typed over in basic mode.

```
                    [^G] :  Get/insert one null ( CRLF only ) line.
          [[ESC]]+<n>+[^G] :  Get/insert n null ( CRLF only ) lines.
       [[ESC]]+<curs>+[^G] :  Insert null lines upto new position.
                    [^X] :  Insert a blank.
          [[ESC]]+<n>+[^X] :  Insert n blanks before cursor.
       [[ESC]]+<curs>+[^X] :  Insert blanks upto new cursor position.
 [[ESC]]+"string"+[ESC]+I  :  Insert "string" at cursor position.
                             Also clear/save the string in PUT buffer.
                  [ESC]+I  :  Insert the string in the PUT buffer.
   [[ESC]]+"file"+[ESC]+L  :  Load/insert file before cursor position.
                  [ESC]+L  :  Load/insert the same file loaded last time.
          [[ESC]]+<n>+[^]] :  Insert character whose code is n decimal.
                    [^]]   :  Insert character defined by above command.
```

5. Deleting Characters

Some of the delete commands save the deleted text in the PUT buffer. You can recover or insert the text using insert command described above.

```
                    [^D] :  Delete one character at cursor.
          [[ESC]]+<n>+[^D] :  Delete n characters from cursor.
       [[ESC]]+<curs>+[^D] :  Delete upto the new position in a line.
                 [ESC]+[^D] :  Delete to EOL from cursor.
                    [DEL] :  Backup and replace by space in basic mode.
                             Cancel prev. character in parameter mode.
               [[[ESC]]]+x :  Delete to 'x' where 'x' is a special char.
                             including space, !, ", #, $, %, &, ', (,...
               [[[ESC]]]+E :  Delete to EOF ( Confirmed by "SURE?" ).
               [[[ESC]]]+K :  Delete to TOF ( Confirmed by "SURE?" ).
                    [^U] :  Delete one line.
          [[ESC]]+<n>+[^U] :  Delete n lines.
       [[ESC]]+<curs>+[^U] :  Delete upto the new cursor position.
                    [^K] :  [^K] commands are the same as [^U] commands
          [[ESC]]+<n>+[^K] :     except that they save the deleted text
       [[ESC]]+<curs>+[^K] :        in the PUT buffer for the later use.
                 [ESC]+[^K] :  [ESC]+[^K] commands are the same as [^K]
    [[ESC]]+<n>+[ESC]+[^K] :     commands except that they clear the PUT
 [[ESC]]+<curs>+[ESC]+[^K] :        buffer before saving the text.
```

6. Searching and Replacing String

```
       [[ESC]]+"sstr"+[^S] :  Search forward the given string "sstr".
                    [^S] :  Search forward the string searched before.
 [[ESC]]+"sstr"+[ESC]+[^S] :  Search backward the given string "sstr".
                 [ESC]+[^S] :  Search backward the string searched before.
                    [^_] :  Search control code forward.
                 [ESC]+[^_] :  Search control code backward.
[[ESC]]+"sstr"+[^R]+"rstr"+[term]
                          :  Replace forward "sstr" to "rstr".
                    [^R] :  Replace forward previous "sstr" to "rstr".
[[ESC]]+"sstr"+[ESC]+[^R]+"rstr"+[term]
                          :  Replace backward "sstr" to "rstr".
                 [ESC]+[^R] :  Replace backward previous "sstr" to "rstr".
```

[term] is a string terminator which defines the replace mode as follows.

```
[term]=[^Z]   ==>  Replace all strings found,
      =[ ESC]  ==>  Inquire whether searched string to be replaced,
      =[^W]   ==>  Quit replace operation.
```

7. Manipulating PUT/SAVE Buffer

```
                    [^Y] :  Clear the PUT/SAVE buffer.
                            This command works in parameter mode also.
                    [^Q] :  Display the content of the PUT/SAVE buffer.
                 [ESC]+P :  Pick/save 1 line into the PUT/SAVE buffer.
    [[ESC]]+<n>+[ESC]+P  :  Pick/save n lines into the PUT/SAVE buffer.
 [[ESC]]+<curs>+[ESC]+P  :  Pick/save into the PUT/SAVE buffer start-
```