Tektronix®

COMMITTED TO EXCELLENCE

4025

COMPUTER DISPLAY TERMINAL

PROGRAMMER'S REFERENCE

4025

COMPUTER DISPLAY TERMINAL

PROGRAMMER'S REFERENCE

061-1551-00

First Printing JAN 1978

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

PRODUCT 4025 Computer Display Terminal

This manual supports the following versions of this product: Level 1.1 and up

MANUAL REVISION STATUS

REV.	DATE	DESCRIPTION
@	1/78	Original Issue

CONTENTS

Section 1 **INTRODUCTION** **Page**

General Description 1-1
  4025 Display 1-2
  4025 Keyboard 1-2
  Communications Interface 1-4
4025 Features 1-4
  Programmable Keyboard 1-4
  The Workspace and the Monitor 1-4
  Buffered Mode and Unbuffered Mode 1-5
  Scrolling, Tabs, and Margins 1-6
Form Fillout Mode 1-7
  Filling Out a Form 1-8
Input Requested by the Host 1-9

Section 2 **COMMANDS AND PROGRAMMING**

Command Format 2-1
  Selecting the Command Character 2-3
  Command Characters and Keyboard Lockout 2-4
Command Descriptions 2-5
Programming the Computer to Control the 4025 2-48
  Invalid Commands 2-50

Section 3 **4025 OPTIONS**

Optional Character Sets 3-1
  Rulings 3-1
Optional Interfaces 3-2
Hard Copies 3-3
Graphics 3-4
  4020-Style Graphics 3-4

Appendix A **RULINGS CHARACTERS**

Appendix B **4010-STYLE GRAPHICS**

Appendix C **SAMPLE COBOL PROGRAM**

ILLUSTRATIONS

Figure	Description	Page
1-1	Sample Form	1-7

TABLES

Table No.	Description	Page
A-1	Alternate Character Fonts for the 4025	A-1
A-2	Ruling Junctions	A-2
B-1	Coordinate Conversion Chart	B-1
B-2	ASCII Code Chart	B-5

Section 1

INTRODUCTION

The purpose of the Programmer's Reference Manual is to describe briefly the 4025 Computer Display Terminal and its basic features, and to describe in detail the programming commands to which the 4025 will respond. This manual also describes some ways in which the 4025 might be used.

Two assumptions are made concerning the person using this manual. First, it is assumed that the person is familiar with computer operations and programming. Secondly, it is assumed the person has access to the 4025 Computer Display Terminal Operator's Manual (hereafter called the 4025 Operator's Manual).

Section 1 provides an overview of the 4025, what it is, and what it can do. Detailed descriptions of commands and options are given in later sections.

GENERAL DESCRIPTION

The 4025 Computer Display Terminal is a microprocessor-based refresh-style terminal designed especially for text editing, form fillout, and graphic display. Internal operations of the 4025 are controlled by a microprocessor and its associated firmware. The 4025 is not intended to be a stand-alone system. It can be connected to a host via a full duplex RS-232 line, or through optional current loop or polling adapter interfaces. It can act as a glass teletype or smart terminal.

The 4025 consists of a display unit, and a keyboard attached by a cable to the display unit. The display unit contains a CRT display, microprocessor and supporting electronics, RS-232 (standard) communications interface, and optional communications/peripheral interfaces.

4025 Display

The 4025 display is a video monitor with several visual enhancements. Brightness and contrast are manually controlled. In addition, the background can be set by commands to one of three brightness levels. Text may be displayed normally, underlined, inverted (dark text on bright background), or blinking. These features can be used to emphasize portions of text and to make form fields easy to recognize. In addition to the standard character font, optional fonts are available and other fonts may be user-defined. An option gives the 4025 graphic ability, including vector generation and addressable screen coordinates. Text on the 4025 screen is also stored in a display memory. The standard 4025 has 4K bytes of display memory, and options are available to expand this to 32K. If the display memory contains more text than the screen can hold, one may scroll through this memory to display various portions of the text.

4025 Keyboard

The keys on the 4025's keyboard fall into three categories: ASCII keys, cursor/numeric pad keys, and function keys.

The ASCII section of the keyboard resembles an ordinary typewriter keyboard. Each key in this section (except the BREAK key) sends a character of the ASCII code to the computer. The ASCII code is described in Appendix B. The BREAK key sends a "break" signal which interrupts the computer's current operation.

The cursor/numeric pad is a group of eleven keys to the right of the ASCII section of the keyboard. This group of keys can function as a cursor pad or a numeric pad. When the NUMERIC LOCK function key is off (unlighted), the group functions as a cursor pad. In this mode the four keys marked with arrows move the cursor, and the two keys marked with triangles scroll the text. When the NUMERIC LOCK function key is on (lighted), the group functions as a numeric pad. In this case the shifted versions of the appropriate pad keys will move the cursor and scroll the text. For example, when the NUMERIC LOCK function key is lighted, SHIFT-8 moves the cursor up.

The function key group consists of the ERASE key, the PT (pad terminator) key, and the sixteen keys along the top of the keyboard. These sixteen keys are divided into four groups of four each. Each key in the rightmost group includes an LED which indicates the status of that key. The ERASE key erases the memory scroll into which the keyboard is currently typing (workspace or monitor). The PT (pad terminator) key is the large key to the right of the cursor/numeric pad. It has no meaning until programmed by the host or the operator. The sixteen keys along the top of the keyboard have the following meanings:

F1 -	Not Assigned
F2 -	Not Assigned
F3 -	Not Assigned
F4 -	Not Assigned
F5 -	HOME
F6 -	Not Assigned
F7 -	Not Assigned
F8 -	SEND**
F9 -	DELETE CHARACTER
F10 -	DELETE LINE
F11 -	ERASE and SKIP
F12 -	INSERT LINE
F13 -	INSERT MODE*
	TTY LOCK*
	NUMERIC LOCK/LEARN*
	COMMAND LOCK OUT /STATUS*

(* - lighted)
(** - This key has no meaning until programmed by the host or operator)

NOTE

If the 4025 contains Option 10 or Option 11, the meaning assigned to some of the function keys will be different. The reader should refer to the 4024/4025 Option 10 Polling Interface Instruction Manual for relevant information.

Most of the keys on the 4025's keyboard can be programmed to have different meanings than those indicated above. A discussion of this is found in the "Programmable Keyboard" section below. The 4025's keyboard is described in detail in the 4025 Operator's Manual.

Communications Interface

The standard 4025 interface is RS-232 ASCII, full duplex. Other interfaces, such as half duplex, GPIB, etc., are available as options.

For the standard interface, the operator uses commands to specify relevant parameters such as local or remote echoing, prompt sequences, parity checking, transmit and receive baud rates, etc.

For the optional interfaces, the operator uses commands to specify parameters that apply to the particular interface, such as character sets, special characters, block header and terminator characters, buffer sizes, etc.

Another option makes the 4025 capable of operating as a polling slave in a multidrop environment.

4025 FEATURES

This section describes the basic features of the 4025 which one must understand in order to program the 4025 effectively.

Programmable Keyboard

Each of the keys on the 4025 (except the TTY LOCK, NUMERIC LOCK/LEARN, COMMAND LOCKOUT, SHIFT, CTRL, and BREAK keys) may be programmed by the operator or the host using the LEARN command. The LEARN command assigns a character string to the programmed key. That character string is generated whenever that key is pressed. This allows commonly used character strings, words or commands to be invoked with a single keystroke. This also allows the user or host to move characters to other keys in order to configure the keyboard for specific applications. Key programming may assign different meanings to the shifted and unshifted versions of the same key. For example, "A" and "a" may be programmed to have different meanings. For a detailed description of the LEARN command, see Section 2, "Command Descriptions."

The Workspace and the Monitor

The 4025's display memory is divided into two sections or scrolls - the workspace and the monitor. The operator or programmer may divide the screen into two areas to display text from each of these scrolls.

The upper area displays text from the workspace. This area is used to create, edit, or review text or forms. Text in the workspace is transmitted to the host only when the SEND command is executed.

The lower area displays text from the monitor. This area is normally used to display terminal commands, and messages to and from the computer. It cannot contain forms or graphics. In general, the monitor allows (1) the operator to communicate with the terminal or the host without this interaction being written over the workspace display, and (2) the host to issue error messages or prompts without writing over the workspace display.

When the terminal is first turned on, or is reset, the entire screen is devoted to the monitor. The numbers of lines on the screen devoted to the monitor and workspace are set by the MONITOR command or the WORKSPACE command. These commands also allow one to specify whether input and/or output is directed into the monitor or into the workspace.

Buffered Mode and Unbuffered Mode

The 4025 can operate either in unbuffered mode or buffered mode. These modes of operation differ in the way that the 4025 processes information.

When the 4025 is in unbuffered mode, each character entered in the monitor is immediately transmitted to the host. Under these circumstances, it is not possible to locally edit the information displayed on the monitor. As soon as a character appears on the 4025 screen, it has already been sent to the computer. If text is entered into the workspace, however, it is not sent to the computer until the SEND command is given and executed. Then the entire contents of the workspace are sent to the computer in a continuous stream.

When the 4025 is in buffered mode, characters entered in either the workspace or monitor are stored in the display memory until RETURN is pressed. Anytime before RETURN is pressed the current line can be edited locally. When RETURN is pressed, the 4025 marks the end of the current line with an end-of-line (EOL) string and stores the line in a transmit buffer. The line remains in the transmit buffer until it can be processed.

The contents of the transmit buffer are processed on a first in-first out basis. When the computer is ready to accept another line, it transmits a prompt string to the 4025. When the 4025 detects this prompt, it examines the oldest line in the transmit buffer. If this line is a command, the 4025 executes the command. If the line is text destined for the computer, the 4025 transmits that line of text to the computer. In either case, the 4025 waits for another prompt from the computer before processing the next line in the transmit buffer.

Prompt strings vary with the program, operating system, and computer. The PROMPT command allows any string to be designated as the prompt string for the 4025. (See Section 2, "Command Descriptions.")

The 4025 powers up in unbuffered mode, and will remain in unbuffered mode until instructed otherwise. The only way to put the 4025 in buffered mode is to use the BUFFERED command. Either the computer or the operator must send !BUF YES or its equivalent to the 4025. (We assume here that "!" is the command character. See Section 2, "Command Descriptions.") In contrast, there are four ways to return the 4025 to unbuffered mode:

1. Send !BUF NO, or its equivalent.
2. The operator may press the RESET button.
3. The operator may press the BREAK key twice in quick succession.
4. The operator may turn the power off, then on again.

Methods 2-4 will change other parameters in the 4025, however, and these should be used only when appropriate.

Scrolling, Tabs, and Margins

Both the monitor and the workspace can each hold more text than the screen can display at one time. If the portion of the screen devoted to the workspace is full, the workspace automatically scrolls up when the next line of text is printed on the screen. Text scrolled off the screen is saved in the 4025's display memory so long as that memory capacity is not exceeded. The scrolled text can be viewed by scrolling back and forth, using the appropriate keys or commands. (See descriptions of the UP, DOWN, RUP and RDOWN commands in Section 2.) The operation for the monitor is similar, except that text scrolled off the screen is discarded automatically when the 4025 needs that memory space for something else. When all available memory has been used, the monitor is reduced to one line. After that, if additional information is entered, the 4025 writes over the last line of the display and over the last line of data stored in memory.

If the user types beyond the right margin, the 4025's bell rings. If he continues typing until the cursor reaches the right edge of the screen, the cursor wraps around to the left margin of the next line. Pressing RETURN also causes the cursor to move to the left margin of the next line. The margins are set by the MARGIN command.

Tab stops for the 4025 are set using the STOPS command, which will define up to 16 tab stops. All stops are set or redefined each time this command is given. Individual stops cannot be changed without resetting all stops. Pressing the TAB key or sending the TAB character to the 4025 causes the cursor to move to the next tab stop. (This action is modified if the 4025 is in form fillout mode. See the following discussion of form fillout mode.)

FORM FILLOUT MODE

A form consists of one or more lines, with each line divided into one or more areas called fields. Some fields will display labels or questions relating to other fields which are blank. The blank fields will be filled in by the operator. A sample form asking for a name, address and ZIP code is shown below.

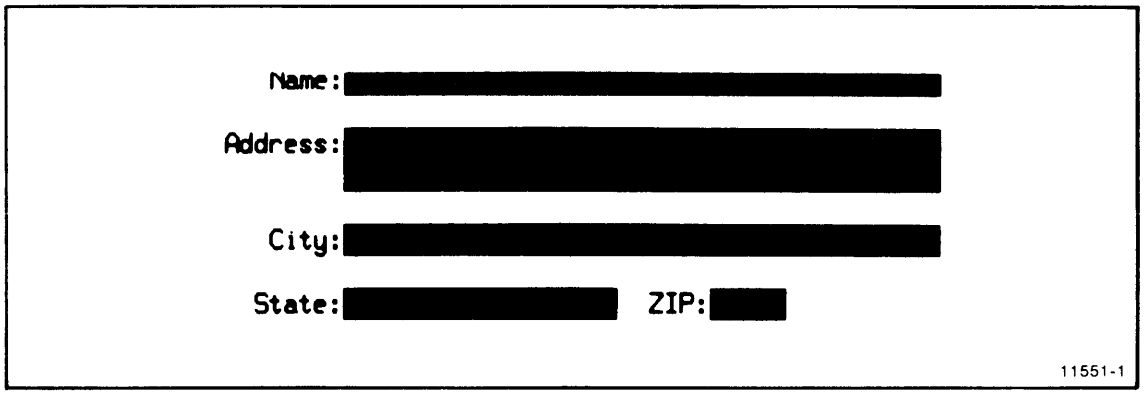


Figure 1-1. Sample Form.

Form fillout allows the operator or the host to display a form in the workspace display, fill out the blanks in the form, and transmit the results to the host or another device for storage or further processing.

When a form is being filled out and the results are transmitted, the 4025 must be in form fillout mode (and in unbuffered mode). At all other times the 4025 should not be in form fillout mode. In particular, when the form is created, or sent to or from the 4025, the 4025 must not be in form fillout mode.

To define a form, each field in a line must be defined separately. The form definition proceeds line by line until the entire form has been defined and sent to the host. Fields are defined using the ATTRIBUTE command (Section 2, "Command Descriptions"). Every field possesses location, length, logical attributes, visual attributes, and font attributes.

Logical attributes define restrictions on the type of data that may appear in a given field. The logical attributes that a field may have are protected (P), modified (M), alphanumeric (A) and numeric (N). Alphanumeric, numeric and protected are mutually exclusive attributes. When the 4025 is in form fillout mode, protected fields cannot be typed into or changed. The default attributes of a field are unprotected, unmodified and alphanumeric.

Visual attributes describe the appearance of the field. The visual attributes are standard (S), underlined (U), inverted (I), and enhanced (E). The standard attribute is defined as the absence of all other attributes. It also is possible to blink the field between any combination of visual attributes (e.g., underscore to enhanced, etc.).

Filling Out a Form

Once a form has been displayed on the screen, and the 4025 has been placed in form fillout mode, the operator may fill out the form. Pressing the HOME key moves the cursor to the beginning of the first unprotected field on the screen. The operator may type into the field until it is full. When the field is full, the cursor moves to the beginning of the next unprotected field on the screen. If the operator wishes to go to the next field before completely filling the current field, he presses the TAB key and the cursor goes to the beginning of the next unprotected field.

The cursor may be moved freely about the page using the cursor movement keys. If, however, the operator attempts to type into a protected field, the bell rings. If the pressed key was a character key, the character is placed at the beginning of the next unprotected field. The bell also rings when an invalid character is typed (such as a letter in a numeric field).

Whenever the operator enters data in a field, the 4025 marks that field as modified. The next SEND MOD command causes all modified fields to be sent to the host. The fields sent are then marked as unmodified so that they are not transmitted again. Thus, the user may enter the whole page and transmit it with the SEND MOD command, or he may use the SEND MOD command after entering each field and cause the fields to be transmitted separately. If the host detects an error in a field, and notifies the operator, the operator may correct his error, and then give the SEND MOD command again. This will cause only the corrected field to be sent to the host.

NOTE

The SEND key has no predefined meaning. The operator or host may program the SEND key to give the SEND command or the SEND MOD command. If SEND MOD is used, only text modified since the last transmission is sent. See Section 2, "Command Descriptions."

When the form has been filled out correctly, the operator gives the SEND command. Unprotected text in the display list is read into a transmit buffer and sent to its destination at host command. Thus the operator or the host program can erase the unprotected text in the display list (using the ERASE command), and the operator may then fill out the form again.

INPUT REQUESTED BY THE HOST

When the 4025 is in unbuffered mode, all text generated in response to input request commands is transmitted immediately to the host. (The same applies to keyboard text.) In buffered mode, all requests for text are processed immediately, but the data is held in a transmit buffer until a prompt is received from the host; then the contents of that buffer are sent a line at a time.

When the 4025 is in buffered mode, any commands or text typed on the keyboard go into the transmit buffer. When a prompt is received from the host (and no input request commands were received from the host in the last transmission) the transmit buffer is processed, local commands are executed, and the text is sent to the host. The user must terminate his transaction by sending the end-of-message character (a carriage return). It is important to note that commands to the 4025 entered from the keyboard are not executed until a prompt is received from the host.

The commands that may request input are SEND and REPORT. A user-defined or system-defined prompt sequence is also recognized, and indicates that any typed text should be returned.

If the terminal detects a prompt from the host, any data in the transmit buffer is processed, the results are sent to the host, any terminal commands in the buffer are executed, and remaining text transmitted.

SEND - For a description of this command, see Section 2, "Command Descriptions." If the 4025 is in buffered mode, the transmission is done a line at a time, with the terminal waiting for a prompt after sending each line.

REPORT - The status of the indicated device is returned to the host. Besides the physical devices in the terminal system (such as the display unit), several logical entities may have their status interrogated. The format of the report depends on the particular device reporting; however, the format is fixed for each device, so that no grammatical analysis by the host program is required. (See Section 2, "Command Decriptions.")

In buffered mode, information generated by a REPORT command is not sent to the host until a prompt is received. Note further that several REPORT commands can be sent during a single transmission; again, all the returned text is stored in a buffer and sent as a unit when the prompt is received. In unbuffered mode, however, an end-of-line string is appended to each report, and the data is transmitted immediately.

Section 2

COMMANDS AND PROGRAMMING

COMMAND FORMAT

Each 4025 command is represented by an English style ASCII string. In addition, the graphics commands have equivalent counterparts on existing 4010-series terminals, and may be represented using the established 4010-style codes. (See Section 3 for a discussion of the graphics options.)

Command Format - A 4025 command consists of four parts:

a) The command character,

b) the keyword,

c) the command parameters, and

d) the command terminator.

The command character is a unique, user-selectable character that does not normally occur in text. When the terminal recognizes this character, it knows that a keyword immediately follows. In this description, an exclamation point "!" will be used for the command character. The "!" is the standard command character when the 4025 is shipped from the factory. If the command character is changed by the operator or programmer, the 4025 will remember the new command character even when power is turned off and back on again.

The keyword is a single word that describes the operation to be performed. This word can be spelled out entirely, or it can be truncated to at least the first three letters. The keyword must immediately follow the command character. There must be no spaces or other characters between the command character and the keyword.

The command parameters, if any, follow the keyword. The type and number of such parameters depend on the particular command. Parameters may be numbers, character strings, or words. Words may be abbreviated to their first letter. Parameters are separated from the keyword and from each other by separators. A separator can be a comma, or one or more spaces. (There are exceptions to this, notably the ATTRIBUTE command.)

The last character in a command, whether a parameter or the final character of a keyword, is separated from subsequent text by a terminator. A terminator can be either a semicolon or a carriage return, <CR>. Commands are not executed, however, until <CR> is detected.

Examples:

```
!ERASE<CR>     (no parameters)
!MARGIN 10,70;!STOPS 20,30,40,60<CR>
!JUMP 10,3<CR>
!DLINE<CR>
```

Exceptions:

The 4025's firmware will allow certain variations:

1. Command terminators can be omitted if the command to be terminated is followed by a command character. For example:
   !ERASE; !DOWN 5 can be abbreviated to:!ERA!DOW 5.

2. Separators between a keyword or alphabetic parameter and a numeric parameter can be omitted. For example:
   !DOWN 5 can be written as !DOW5 or !HCOPY 3,W can be written as !HCO3W.

3. Separators followed by + or - characters may be omitted. For example:

   !RVECTOR +5,0,-20,-110,+35,-110 may be written as
   !RVE+5,0-20-110+35-110.

4. Some 4025 commands can be continued from one line of program code to the next by inserting a continuation character in the command. There are two cases where this can be done.

   (1) In a VECTOR or RVECTOR command. The ampersand, "&", can be inserted after a numeric parameter and the command continued to the next line.

   Example:

   In BASIC, the statement:

   ```
   100 PRINT "!GRAPHIC 1,20;!VECTOR 0,0,175,175,0,175,0,0;"
   ```

   can be written as:

   ```
   100 PRINT "!GRAPHIC 1,20;!VECTOR 0,0,&"
   101 PRINT "175,175,0,175,0,0;"
   ```

(2) In commands that take delimited ASCII strings as parameters (the EOF, EOL, LEARN, PROMPT, and STRING commands). A delimited string can be divided into two delimited strings on consecutive lines of program code using the hyphen, "-", as a continuation character.

Example:

In BASIC, the statement:

```
200 PRINT "!LEARN F1 /!SEND ALL;!ERA W;/"
```

can be written as:

```
200 PRINT "!LEARN F1 /!SEND ALL;/-"
201 PRINT "/!ERA W;/"
```

The continuation character should **not** be inserted:

(a) Between the command character and the command keyword.

(b) Within the keyword.

(c) Within a numeric parameter or between a plus or minus sign and a number.

(d) Within delimiters of an ASCII string.

NOTE

Often, considerable reduction in line transmission can be achieved using these techniques. However, one should use caution when compacting text in this way.

5. In addition to the above exceptions, individual commands may tolerate minor variations in syntax. For details, see descriptions of the individual commands.

Selecting the Command Character

The choice of command character for a particular applications program should be made carefully. Command characters should not be sent to the 4025 in any role except as command characters. If the command character is sent in another context, the 4025 and the operator will be confused. For example, in a text editing program the exclamation point, "!", would be a risky choice for the command character. In such a program this character might occur in text as a punctuation mark. Another character should be chosen.

CAUTION

Symbols such as "carriage return" or "space," which are normally used during host/terminal communications, should not be used as command characters.

Command Characters and Keyboard Lockout

It may be desirable to prevent an operator from issuing arbitrary commands to the 4025 during an applications program, and still allow him to issue host-determined commands or command sequences to the 4025. For example, during a form fillout program, the operator should not be able to modify the form itself, but should be able to give the SEND MOD command.

Key programming can accomplish this. Suppose "!" is the command character. If the host sends "!LEARN ! 00;" to the 4025, the "!", or SHIFT-1 key will be programmed to generate an ASCII null character. This prevents the operator from using the "!" key to generate the command character. Yet the host can send command characters to the terminal as usual, and function keys can be programmed to issue commands. Only the operator's ability to issue the command character from the keyboard is impaired. At the proper time, control of the 4025 can be returned to the keyboard by sending "!LEARN ! 33;" from the host.

A Note on Syntax Notation

When an expression is enclosed in square brackets, "[...]," that expression is optional. It need not appear for a valid command to be given. (It may have to appear, however, for the command you give to have the effect you desire.) When an expression is not enclosed in square brackets, that expression is necessary to define a valid command. For example, "!BEL[L]" means that either "!BEL" or "!BELL" will cause the 4025's bell to ring; "!BE" is not sufficient. Likewise,
"!ATT[RIBUTE]" means that "ATT", "ATTR", ..., "ATTRIBUTE" are all valid keywords for the ATTRIBUTE command. The characters "AT" are not enough; "ATT" is the minimum.

Whenever the notation "[...|...|...]" is used following a command, it signifies that one choice is to be made from the list of strings separated by vertical bars. Note that square brackets may be nested. Thus, '!FOR[M][Y[ES]|N[O]]" means that:

"!FOR", "!FOR N", "!FORM NO", "!FOR Y", "!FOR YES"

are all valid commands.

Expressions enclosed in pointed brackets, "<...>", are parameter names. (The notation <CR> means carriage return.) When one gives a specific command, the string name inside "<...>" will be replaced with one choice from a specified set of valid replacements. Replacements may be alphabetic or numeric strings. The set of valid replacements for the name enclosed by "<...>" is listed or described for each command. When a listing is used, the assignment operator ":=" appears, with "<...>" to the left, and the set of valid replacements to the right. The set is enclosed with square brackets "[...]", and members of the set are separated by vertical bars, "|."

For example, we have the following syntax:

```
"!BAC[KTAB] [<count>]<CR>

where <count>:=[+][1|2|3...|32767]."
```

The first line indicates that <count> is optional. The second line indicates that when <count> is specified, it should be a positive integer not greater than 32767, preceded by an optional "+" sign.

COMMAND DESCRIPTIONS

ATTRIBUTE

Syntax:

```
!ATT[RIBUTE] [<font>][<logical>][<visual> [- <visual>]] <CR>
```

where:

<font> := <a number from 0 to 31>, and denotes the character font,

<logical> := [M][A|N|P] and denotes the logical attributes where M = modified, A = alphanumeric, N = numeric, P = protected; and

<visual> := [S|E[U][I]|U[E][I]|I[U][E]] and denotes a set of visual attributes where S = standard, E = enhanced, U = underlined, and I = inverted.

Examples of syntax:

```
!ATT AE<CR>         <logical>=<A>, <visual>=<E>
!ATT 3 E-UI<CR>     <font>=<3>, <visual #1>=<E>, <visual #2>=<UI>
!ATT 0 MPS-U<CR>    <font>=<0>, <logical>=<MP>,
                    <visual #1>=<S>, <visual #2>=<U>
```

Restrictions on syntax:

No spaces are allowed between the letters designating the logical and visual attributes.

Action:

This command inserts a field attribute code into the workspace's display list, at the current location of the cursor. The field attribute code marks the beginning of a new field in that line of the workspace, and designates those attributes which apply to that field and are different from the attributes of the field preceding it (if any) in that line.

Field attributes fall into three categories: character font attributes, logical attributes, and visual attributes. Each line in the workspace begins with the default attributes: character font 0, alphanumeric logical attribute, standard visual attribute. In scanning the line from left to right, when an attribute code is encountered, it denotes the start of a new field in the line and may change one or more of the three classes of attributes. If a class of attributes (font, logical, or visual) is not mentioned in an ATTRIBUTE command, then that command does not change that class of attributes; the attributes of the preceding field in the line remain in effect.

Font attribute:	A number from 0 to 31 designating the character font which is to be used for displaying characters in the following field. The default character font is Font 0, the standard font. Font 1 is the "rulings" font. Other fonts may be determined by ROMs inserted in the Character Set Expansion Board (Option 31) or may be defined by the user with SYMBOL commands (requires Option 23, 24, 25, or 26.)

Logical attributes:

M	Modified	When this attribute is attached to a protected field, that field will be sent to the host in the next SEND MOD operation.
A	Alphanumeric	The default logical attribute. Specifies an unprotected field into which any alphanumeric character may be entered.
N	Numeric	Specifies an unprotected field into which numerals and special characters may be entered, but not letters of the alphabet.
P	Protected	Specifies a protected field. In form fillout mode, and only in form fillout mode, protected fields cannot be erased or typed into.

NOTE

All logical attributes have effect only when the 4025 is in form fillout mode.

Visual attributes:

S	Standard	The default visual attribute. Displays characters as light on a dark background. This is the absence of the other visual attributes.
E	Enhanced	Displays the dark background of each character as slightly brighter than usual. (The absolute brightness and contrast are still manually controlled by the operator, however.)
I	Inverted	Displays characters as dark on a light background, rather than the standard light on dark background.
U	Underlined	Displays an underline under all characters and spaces in the field.

NOTE

Combinations of the E, I, and U visual attributes are permitted, e.g.,

```
!ATT EI<CR>
!ATT EU<CR>
!ATT IU<CR>
!ATT EIU<CR>
```

The display may "blink" (alternate) between two visual attributes or sets of visual attributes. To accomplish this, specify both sets of attributes in the ATTRIBUTE command, separating them by a hyphen:

!ATT E-I<CR>	Blinks between enhanced and inverted.
!ATT I-E<CR>	Blinks between inverted and enhanced.
!ATT S-IU<CR>	Blinks between standard and "inverted with underline."

BACKTAB

Syntax:

```
!BAC[KTAB] [<count>]<CR>
```

where:

```
<count>:=[+][1|2|3|...|32767]
```

Action:

Like pressing the BK TAB key <count> times; performs <count> backtab operations.

When not in form fillout mode, each backtab operation moves the cursor to the preceding tab stop, or to the start of the line if there are no tab stops to the left of the cursor. The cursor will not jump to a preceding line of text, regardless of how many backtabs are attempted.

When in form fillout mode, each backtab operation moves the cursor to the beginning of the unprotected field in which it is located. If the cursor is already at the start of an unprotected field, or if it is not inside an unprotected field, a backtab operation moves the cursor to the start of the preceding unprotected field. If the cursor is already at the start of the first unprotected field in the form, a backtab operation leaves the cursor where it is.

Default:

If the <count> parameter is omitted, it is assumed to be 1.

BAUD

Syntax:

```
!BAU[D] <parameter> [parameter>]<CR>
```

where:

```
<parameter>:=[0|50|75|110|150|134|300|600|1200|1800|2400|4800|9600]
```

Action:

This command sets the baud rate. If only one parameter is given, it is used to set both the transmit and receive baud rates. If two parameters are given, the first is for transmit and the second for receive. A parameter of 0 means an external clock is used.

Examples:

!BAU 300<CR>	Sets both transmit and receive rates to 300 baud.
!BAU 300, 600<CR>	Sets transmitting rate to 300 baud, receiving rate to 600 baud.

Default:

If no parameter is specified, the command has no effect.

BELL

Syntax:

```
!BEL[L]<CR>
```

Action:

Sounds the terminal's bell. (This may also be accomplished by sending the ASCII BEL character, a CTRL-G.)

BUFFERED

Syntax:

```
!BUF[FERED] [ Y[ES] | N[O] ]<CR>
```

Examples of Syntax:

```
!BUFFERED YES<CR>
!BUF Y<CR>
!BUF N<CR>
!BUF<CR>
```

Action:

Controls buffered mode. If Y (for Yes) is specified, the terminal is placed in buffered mode. If N (for No) is specified, the terminal is placed in unbuffered mode.

When a BUFFERED YES command is given by the host, each subsequent line of text typed on the keyboard will be held until the host has sent a prompt requesting that line. When a BUFFERED YES command is typed on the keyboard, the same is true, except that the 4025 behaves as if it has already received the first prompt. That is, the first line typed on the keyboard will be sent as soon as it is terminated, without waiting for a prompt; but subsequent lines will each require a prompt before they are sent.

Default:

If no parameter is specified, Y (for Yes) is assumed.

CLEAR

Syntax:

```
!CLE[AR]<CR>
```

Action:

Removes all the programmed key definitions.

COMMAND

Syntax:

```
!COM[MAND] <parameter><CR>
```

where:

<parameter> may be a single ASCII character, or the ASCII Decimal Equivalent of that character, a two- or three-digit number between 00 and 127.

Examples of syntax:

```
!COMMAND #<CR>
!COM 35<CR>
```

Permissible variations of syntax:

If the parameter is not a letter of the alphabet, the space following the keyword may be omitted.

Action:

Sets the command character to the character specified by the parameter. If the parameter is a single character, that character becomes the new command character. If the parameter is a two or three digit number, that number is the ASCII decimal equivalent of the new command character.

Default:

If no parameter is specified, this command has no effect; the command character remains as it was.

CAUTION

Symbols such as "carriage return" or "space", which are normally used during host/terminal communications, should not be used as command characters.

COPY (Printer)

(Requires Option 3.)

Syntax:

```
!COP[Y] < W[ORKSPACE] | H[OST] > P[RINTER]<CR>
```

Examples of Syntax:

```
!COPY WORKSPACE PRINTER<CR>
!COP W,P<CR>
!COP H P<CR>
```

Details of this command will be supplied in the final version of this manual.

DCHAR (Delete Character)

Syntax:

```
!DCH[AR] [<count>]<CR>
```

Examples of syntax:

```
!DCHAR<CR>
!DCH 3<CR>
```

Action:

This command causes the <count> characters at the cursor position (and immediately following, if <count> is bigger than 1) to be deleted. Characters to the right of the cursor in the same field (if in form fillout mode) or line (if not in form fillout mode) are shifted to the left one place. In form fillout mode, characters will not be deleted from a protected field or from an adjacent unprotected field.

This command is equivalent to pressing the DELETE CHAR key <count> times.

Default:

If the <count> parameter is omitted, it is assumed to be one.

DELAY

Syntax:

```
!DEL[AY] <time><CR>
```

where:

<time> is an integer from 0 to 32767.

Examples of syntax:

```
!DEL 0<CR>
!DEL 15<CR>
```

Action:

Sets the turnaround delay to at least <time> milliseconds. In buffered mode, after a prompt is detected, the 4025 waits this length of time before transmitting anything.

DFONT (Delete Font)

(Requires Option 23, 24, 25, or 26.)

Syntax:

```
!DFO[NT] <font number><CR>
```

where:

<font number> is an integer in the range from 1 to 31.

Action:

This command deletes the symbol definitions in the specified programmable font. The space may be used for graphics when the font is deleted.

DISCONNECT

(Requires Option 1.)

Syntax:

```
!DIS[CONNECT]<CR>
```

Action:

Sends a signal to the modem, causing it to disconnect the 4025 from the communications line. (Turns off the "data terminal ready" signal on the RS-232 interface for about one second. This should cause the the modem to disconnect from the telephone line.)

DLINE (Delete Line)

Syntax:

```
!DLI[NE] [<count>]<CR>
```

Action:

This command causes <count> lines at the cursor position to be deleted. If the 4025 is in form fillout mode, all unprotected character positions in the lines are changed to blanks. However, if the terminal is not in form fillout mode, the lines themselves are deleted and lines below are rolled up to fill the gap.

Default:

If <count> is omitted, it is assumed to be one.

DOWN

Syntax:

```
!DOW[N] [<count>]<CR>
```

Action:

This command causes the cursor to move down <count> lines. If the cursor is on the bottom line of the text window, the text is rolled up. If there is no text available below the current line, a new blank line is created at the bottom of the page to allow rollup to occur.

Default:

If <count> is omitted, it is assumed to be one.

DUPLEX

(Requires Option 1.)

Syntax:

```
!DUP[LEX] [<fulldup>|<halfdup>]<CR>
```

where:

```
<fulldup>:= F[ULL]
<halfdup>:= H[ALF][N[ORM]|S[UPERVISOR]][P[ROMPT]|L[INE]]
```

(Parameters must be separated from the keyword and each other by spaces or commas.)

Examples of syntax:

```
!DUPLEX<CR>
!DUPLEX FULL<CR>
!DUP<CR>
!DUP F<CR>
!DUP H<CR>
!DUP H S<CR>
!DUP H S P<CR>
!DUP H S L<CR>
!DUP H N<CR>
!DUP H N P<CR>
!DUP H N L<CR>
```

Action:

This command sets the 4025 for either full or half duplex data communications. Full duplex mode is used with full duplex communication lines, which permit both terminal and host to transmit at the same time. Half duplex is used with half duplex communications lines, over which only one device (terminal or host) may transmit at a time.

If half duplex is specified, either normal or supervisor mode may be chosen. Also, in half duplex mode one may specify whether the line turnaround condition (rather than the prompt string) is to be used as the prompt in buffered operation.

Default:

If no parameters are specified, full duplex operation is assumed. If half duplex is chosen, but neither "normal" nor "supervisor" mode is specified, "supervisor" mode is assumed. If half duplex is chosen, and neither "prompt" nor "line" is specified, "line" is assumed; the line turnaround is taken as the prompting condition.

ECHO

Syntax:

```
!ECH[O] [ L[OCAL] | R[EMOTE] ]<CR>
```

Action:

This command controls keyboard echoing when text from the keyboard is directed to the monitor and the 4025 is in unbuffered mode. If L (for Local) is specified, characters typed on the keyboard will be displayed (echoed) on the screen. If R (for Remote) is specified, typed characters will not be displayed. (In "remote echo" data communications, the host or modem would echo each character back it receives it from the 4025. It is the host- or modem-supplied "remote echo" that is displayed on the screen.)

Default:

If neither L nor R is specified, L is assumed.

EOF (End-of-File)

(Requires Option 3.)

Syntax:

```
!EOF [<string>]<CR>
```

where <string> may be:

(a) One or more delimited ASCII strings. A delimited ASCII string consists of a string of any printable ASCII characters (except the command character) with a delimiter at each end. The delimiter may be any printing character except the command character, space, comma, semi-colon, numerals, or letters of the alphabet.

(b) A sequence of ADE values (ASCII Decimal Equivalents) separated by spaces or commas.

(c) Any combination of (a) or (b).

Examples of syntax:

```
!EOF "abc"<CR>
!EOF /abc/<CR>
!EOF 17 18 19<CR>
!EOF /abc/17,18,19<CR>
```

Restrictions on syntax:

The <string> may define a maximum of ten ASCII characters.

Action:

This command sets the string which is used to indicate end-of-file when copying text from the host to a 4642 Line Printer. (See the COPY command.)

Default:

If no <string> is specified, "slash, asterisk" (/*) is assumed.

EOL (End-of-Line)

Syntax:

```
!EOL [<string>]<CR>
```

where <string> may be:

(a) One or more delimited ASCII strings. A delimited ASCII string consists of a string of any printable ASCII characters (except the command character) with a delimiter at each end. The delimiter may be any printing character except the command character, space, comma, semi-colon, numerals, or letters of the alphabet.

(b) A sequence of ADE values (ASCII Decimal Equivalents) separated by spaces or commas.

(c) Any combination of (a) or (b).

Examples of syntax:

```
!EOL /abc/<CR>
!EOL 13 10<CR>
!EOL /abc/ 13,10<CR>
```

Restrictions on syntax:

This command may define a string of no more than 10 characters.

Action:

This command sets the end-of-line string, which the 4025 sends to the host at the end of each line of text. When typing on the keyboard, with text from the keyboard directed to the monitor and the terminal in unbuffered mode, the end-of-line string is sent whenever RETURN is pressed. When text from the workspace is sent (with a SEND command), the end-of-line string is appended at the end of each line of text. Similarly, in buffered mode, as the host requests each line of text from the 4025, the 4025 sends that line and appends an end-of-line string at the line's end.

The 4025 remembers its end-of-line string even after power has been turned off.

Default:

If no <string> is specified in the EOL command, the 4025 sets the end-of-line string to "carriage return."

ERASE

Syntax:

```
!ERA[SE] [ W[ORKSPACE] | M[ONITOR] | G[RAPHICS] ]<CR>
```

Examples of syntax:

```
!ERASE WORKSPACE<CR>
!ERA M<CR>
```

Restrictions on syntax:

In order to specify G (for Graphics), Option 23, 24, 25, or 26 must be installed.

Action:

This command erases the specified area. If W or M is specified, the entire workspace or monitor display list is erased. If G is specified, the entire graphics area, including text, is erased. (ERASE G can also be performed by sending ESCAPE-FORM FEED.)

Default:

If no parameter is specified when this command is typed on the keyboard, the command erases whichever scroll (workspace or monitor) into which text from the keyboard is directed. If no parameter is specified when this command is given by the host, the command erases whichever scroll (workspace or monitor) into which text from the host is directed.

FIELD

Syntax:

```
!FIE[LD] [<character>]<CR>
```

where:

> <character> may be a single printing ASCII character or a 2- or 3-digit number from 00 to 127.

Action:

This command sets the character which is used to precede fields of a form when they are transmitted to the host. If no value is supplied, then no character is inserted before a field, and trailing blanks are sent. The <character> parameter can be represented as an ASCII character or a 2- or 3- digit ASCII Decimal Equivalent. Common values of this parameter are TAB, CR, and US.

Default:

If no parameter is specified, it is assumed to be NULL.

FORM

Syntax:

```
!FOR[M] [ Y[ES]|N[O] ]<CR>
```

Action:

This command controls form fillout mode. If Y (for Yes) is specified, the 4025 is placed in form fillout mode. If N (for No) is specified, the 4025 is removed from form fillout mode.

Default:

If no parameter is specified, Y is assumed.

GRAPHIC

(Requires Option 23, 24, 25, or 26.)

Syntax:

```
!GRA[PHIC] <beg row> <end row> [<beg col>[<end col>]]<CR>
```

where:

<beg row>, <end row>, <beg col>, <end col> are positive integers designating rows and columns of the workspace.

Restrictions on syntax:

The parameters must be separated from each other by commas or spaces. The <beg col> and <end col> parameters must be no more than 80. Also, <beg row> must be less than <end row>, and <beg col> less than <end col>. The <end row> parameter may not exceed the <beg row> parameter by more than 52.

Action:

This command sets up a graphic region in the workspace display list and clears it of old characters. If there is no graphic memory available, then an error message, "NO GRAPH MEM," is printed in the monitor. A maximum of 53 lines may be included in the graphics area.

Default:

If the <beg col>, or <beg col> and <end col> parameters are omitted, they are assumed to be 1 and 80, respectively.

Affect on other commands:

The effect of editing keys and editing commands is changed somewhat by the presence of the graphic area. The following describes the major differences.

Delete Character: Inside a graphic area, the character is replaced by a space.

Delete Line: On a line which passes through a graphic area, characters outside the graphic area are deleted. The appearance of the graphic area does not change.

Erase & Skip: In a line that passes through a graphic area, only characters outside the graphic area are affected.

Form Fillout Mode: All locations within the graphic area are protected. If a graphic area is less than 80 columns wide and no form exists in the side area(s), then the side areas are unprotected by default and text may be entered into them. To prevent text from being entered into these areas, they must be protected or the graphic area expanded to include all 80 columns.

Erase Workspace: This erases the entire workspace, including the graphic area definition. A new !GRAPHIC command must be given before further graphics may be displayed.

Cursor movement and typing: Cursor movement and typing are not affected by the presence of the graphics area. A character has priority over graphics inside the graphics area.

Attribute codes: Inside a graphic area, only font codes may be given in the ATTRIBUTE command. All other attributes are ignored. Any visual attributes (enhanced, etc.) which are in effect at the left edge of the graphic area will also affect the corresponding "stripe" running through the graphic area. Any logical attributes and font codes in effect at the left edge of the graphic area do not affect the area itself, but characters to the right of the window are given the same font and logical attributes in effect before the window was created. It is possible to create a new graphic area on the display when there is still one present. If this happens, all further graphics commands will affect only the new area. The old area becomes, for all intents and purposes, a "text" area, since it cannot be further modified graphically. It does, however, maintain its integrity against editing in the same way as the active graphic area. When creating a new graphic area, it is desirable not to overlap it with other existing graphic areas, particularly if the graphic areas are fairly wide. Overlapping graphic areas can cause the limit on information in one character row to be exceeded, causing undesired results on the display.

GTEST (Graph Test)

(Requires Option 23, 24, 25, or 26.)

Syntax:

```
!GTE[ST]<CR>
```

Action:

Executes a WORKSPACE 0 operation, destroying the entire contents of the workspace and monitor and causing the entire screen to be used for displaying the monitor. Then tests the graphic memory. After a delay of about 15 seconds (while it performs the test), the monitor displays the test results, starting with character set 1 and proceeding to character set 31. If no RAM is installed for a particular character set, the 4025 displays a "NO RAM" message. If RAM is installed, each character is tested twice (each bit is tested for both 1 and 0), and the 4025 displays "OK" for each of these two tests if the RAM passes the test. If the RAM for a particular character set fails the test, the 4025 displays "BAD RAM" together with an error code for use by service personnel.

HCOPY (Hard Copy)

(Requires Option 40.)

Syntax:

```
!HCO[PY] [<count>] [ W[ORKSPACE] | M[ONITOR] |S[CREEN] ]<CR>
```

Examples of syntax:

```
!HCOPY WORKSPACE<CR>
!HCO 2 M<CR>
```

Restrictions on syntax:

The <count> parameter is ignored if S (for Screen) is specified.

Action:

This command copies <count> pages from the specified area to a TEKTRONIX 4631 Hard Copy Unit, beginning with the first visible line in that area. If fewer than <count> pages exist in the display list, then only that number of pages in the display list is printed; however, at least one page of hard copy will be produced. If S (for Screen) is specified, the lines copied are exactly what is visible on the screen; <count> has no meaning in this case.

Each page of copy continues until it includes 53 lines of text or until an ASCII form feed character is encountered as the first character of a line. The line of text containing such a form feed is not copied on the current page.

Default:

If <count> is omitted, it is assumed to be one.

If the HCOPY command is typed on the keyboard, and neither W nor M nor S is specified, then a copy is made of whichever scroll (workspace or monitor) receives text from the keyboard.

If the HCOPY command comes from the host, and neither W nor M nor S is specified, then a copy is made of whichever scroll (workspace or monitor) receives text from the host.

HRULE (Horizontal Rule)

(Requires Option 32.)

Syntax:

```
!HRU[LE] <row> <column> [<length> [<width>] ]<CR>
```

Restrictions on syntax:

All parameters are positive integers. The <column> parameter must not exceed 80. The sum of <column> and <length> must not exceed 81. The <width> parameter, if specified, must be either 1 or 2.

Action:

Draws a horizontal ruling in the workspace. The first character of the ruling is inserted at the row and column specified by the <row> and <column> parameters, and the ruling continues to the right for a total of <length> character cells. The ruling is a single line if <width> is one, and a double line if <width> is two.

Default:

If <length> or <width> are omitted, they are assumed to be one.

ICHAR (Insert Character)

Syntax:

```
!ICH[AR]<CR>
```

Action:

The terminal is put in insert mode, as if the INSERT MODE key had been pressed. Henceforth, all characters to the right of the cursor are moved to the right when a character is received from the host or typed on the keyboard. This mode of operation is terminated on receipt of a cursor movement control character or a cursor movement command, or when the operator types a cursor movement character or moves the cursor.

ILINE (Insert Line)

Syntax:

```
!ILI[NE] [<count>]<CR>
```

Action:

This command inserts <count> blank lines into the text, immediately below the line holding the current cursor position. The cursor is left at the beginning of the newest line. Existing lines of text below the cursor position are scrolled down to make room for the inserted blank lines. This command makes it easy to insert new text between lines of old text. Use the ILINE command to create several blank lines at the desired location. The operator may then type into the blank lines, overwriting the blanks with new text. The DLINE command can then be used, if necessary, to delete unwanted blank lines left over.

NOTE

For text editing applications, the first line entered into the workspace should be blank. Then if new text must be inserted above the old text, the cursor is moved to the beginning of the workspace and the ILINE command creates space for the new text. If the first line of the workspace already contains text, this procedure inserts blank lines below the first line of old text, rather than above it.

Default:

If <count> is omitted, it is assumed to be one.

JUMP

Syntax:

```
!JUM[P][<row> [<column>]]<CR>
```

Restrictions on syntax:

The two parameters must be positive integers; <column> may not exceed 80.

Action:

Moves the workspace cursor to the location specified by the <row> and <column> parameters. The workspace will be scrolled up or down if necessary to display the cursor in the specified location.

The <row> and <column> parameters are absolute workspace coordinates. If the workspace cursor is at the beginning of the fifth line, the command !JUM 3<CR> moves the cursor to the beginning of the third line. The command !JUM 1<CR> always moves the cursor to the beginning of the first line of the workspace. If the 4025 is in form fillout mode, the command sequence !JUM 1;!TAB<CR> will send the cursor to the beginning of the first unprotected field in the form. This command sequence is equivalent to pressing the HOME key in form fillout mode.

Default:

If <column> is omitted, the cursor is put at the beginning of the desired row. If <row> is also omitted, row 1 is assumed.

NOTE

JUMP cannot be used to move the cursor in the monitor.

LEARN

Syntax:

```
!LEA[RN] <key> [<string>]<CR>
```

The <key> parameter may be any of the following:

(a) A single ASCII character.

(b) A 2- or 3-digit ADE value (ASCII Decimal Equivalent) from 00 to 127.

(c) A mnemonic representing a non-ASCII key (function key or cursor/numeric pad key:

F1 - F13	Function keys 1 through 13. (Function key 13 is the Insert Mode key, which can be programmed.)
S1 - S13	Function keys 1 through 13 with SHIFT depressed.
P0 - P9,P.,PT	Numeric pad and terminator.

(d) A "pseudo-ADE value" representing a non-ASCII key:

128 Function Key 1	144 SHIFT- Function Key 1
129 Function Key 2	145 SHIFT- Function Key 2
130 Function Key 3	146 SHIFT- Function Key 3
131 Function Key 4	147 SHIFT- Function Key 4
132 Function Key 5	148 SHIFT- Function Key 5
133 Function Key 6	149 SHIFT- Function Key 6
134 Function Key 7	150 SHIFT- Function Key 7
135 Function Key 8	151 SHIFT- Function Key 8
136 Function Key 9	152 SHIFT- Function Key 9
137 Function Key 10	153 SHIFT- Function Key 10
138 Function Key 11	154 SHIFT- Function Key 11
139 Function Key 12	155 SHIFT- Function Key 12
140 Function Key 13	156 SHIFT- Function Key 13
160 Pad Key 0	172 ERASE
161 Pad Key 1	173 SHIFT-ERASE
162 Pad Key 2	174 BK TAB
163 Pad Key 3	
164 Pad Key 4	
165 Pad Key 5	
166 Pad Key 6	
167 Pad Key 7	
168 Pad Key 8	
169 Pad Key 9	
170 Pad Key .	
171 Pad Terminator Key	

The <string> parameter may be any of the following:

(a) One or more delimited ASCII strings. A delimited ASCII string consists of a string of any printable ASCII characters (including spaces, semicolon, commas, etc., including the command character) with a delimiter on each end. The delimiter can be any printing character except the command character, carriage return, space, comma, semi-colon, numerals, or letters of the alphabet.

(b) A sequence of ADE values (ASCII Decimal Equivalents) separated by commas or spaces.

(c) A sequence of pseudo-ADE values representing non-ASCII keys, separated by commas or spaces. These pseudo-ADE values are the same as those described above for the <key> parameter.

(d) Any combination of (a), (b), or (c).

The <string> parameter may not include function key mnemonics like those described under (c) for the <key> parameter.

When the LEARN command is given from the host computer, it may be continued from one line to another by inserting a hyphen (-) outside a delimited ASCII string as the last character on the line before the carriage return which terminates the line. (This may not be done in LEARN commands typed on the keyboard.) This causes the <CR>, up to one LF, and all NUL, rubouts, and syncs to be ignored until another character is seen.

Examples of syntax:

!LEA[RN] <key> [<string>]<CR>

!LEARN F1 33 /WOR 20 H K/ 13<CR>	<F1> designates Function Key 1. <String> designates the !WOR 20 H K <CR> command. 33 is the ADE for "!".
!LEA 172 13<CR>	<172> designates the ERASE key. <13> designates "carriage return."

Action:

This command redefines the key designated by its <key> parameter, assigning to it the string of ASCII characters and/or keystrokes designated by the <string> parameter. Since <string> may include the command character, the key may be programmed to give any command or series of commands.

There are only a few keys which may not be programmed, or included as elements in the programming of other keys. These are:

- The rightmost three of the lighted function keys - TTY LOCK, NUMERIC LOCK, COMMAND LOCKOUT and their shifted versions, LEARN and STATUS.
- CTRL, SHIFT, and BREAK.

Default:

If the <string> parameter is omitted, the key named in the <key> parameter is assigned its default meaning.

Examples:

!LEARN Q /abc/<CR>	Assigns the meaning "abc" to the uppercase Q key. (Does not affect lowercase q.)
!LEA F1 33 /WOR 20 H K/ 13<CR>	Assigns Function Key 1 the meaning: "! WOR 20 H K <CR>" Henceforth, pressing that key gives the WORKSPACE 20 H K command.
!LEA 172 13<CR>	Redefines the ERASE key to mean "carriage return."
!LEA 172<CR>	Causes the ERASE key to revert to its original meaning.
!LEA 156 "You shouldn't" - "press SHIFT when you" - "press INSERT MODE."<CR>	Programs SHIFT-INSERT MODE (SHIFT-Function Key 13) to give an error message. This example illustrates how to continue the key definition from one line to another. (This technique may not be used when typing the command on the keyboard.)

LEFT

Syntax:

```
!LEF[T] [<count>]<CR>
```

Action:

Moves the cursor to the left <count> character positions. (This command is equivalent to pressing the "left cursor" key <count> times.) If <count> is great enough to require the cursor to move to the left of the left margin, it will wrap around to column 80 of the previous line. If necessary, rolldown will occur, but not past the top of the workspace.

Default:

If <count> is omitted, it is assumed to be one.

LINE

(Requires Option 23, 24, 25, or 26.)

Syntax:

```
!LIN[E] [<line type>]<CR>
```

where <line type> may be one of the following:

- A digit from 1 to 8.
- The letter E.
- The letter P.

Action:

Sets the type of line that subsequent VECTOR commands draw. Line type 1 is a solid line, and types 2 through 8 are different styles of dashed lines. Line type E causes subsequent VECTOR commands to draw "erase vectors" which erase lines previously drawn. Line type P causes subsequent VECTOR commands to plot isolated points.

Default:

If no <line type> is specified, it is assumed to be one (solid line).

MARGINS

Syntax:

```
!MAR[GINS] [<left marg> [<right marg>] ]<CR>
```

where:

<left marg> and <right marg> are positive integers.

Restrictions on syntax:

The <left marg> must be less than the <right marg>, and both must be between 1 and 80, inclusive.

Action:

Sets the left and right margins in the workspace. (Margins are not defined in the monitor.)

Default:

If <right marg> is omitted, it is assumed to be 80. If <left marg> is omitted as well, it is assumed to be 1.

MONITOR

Syntax:

```
!MON[ITOR] [<number>] [ H[OST] ] [ K[EYBOARD] ]<CR>
```

Restrictions on syntax:

The <number> parameter, if included, must be an integer in the range from 1 to 34, inclusive.

Action:

If the <number> parameter is included, this command erases both the workspace and the monitor. If <number> is less than 34, it then defines a workspace and reserves the top (34 - <number>) lines the screen for displaying that workspace. The bottom <number> lines of the screen are used to display the monitor.

If H (for Host) is specified, text coming from the host computer will be directed to the monitor.

If K (for Keyboard) is specified, text coming from the keyboard will be directed to the monitor.

Default:

If no parameters are specified, and the MONITOR command comes over the communications line from the host computer, it is executed as a MONITOR H command. If no parameters are specified and the command is typed on the keyboard, it is executed as a MONITOR K command.

PARITY

Syntax:

```
!PAR[ITY] [ E[VEN] | O[DD] | N[ONE] | D[ATA] ]<CR>
```

Action:

This command sets the terminal parity. If EVEN is specified, the 4025 transmits even parity and checks for even parity of incoming characters. ODD behaves similarly for odd parity. NONE indicates that parity is not checked on input; the parity bit is set to zero on output. DATA means that the parity bit is treated as data, and set to 0 for output.

Default:

If the parameter is omitted, it is assumed to be N (for NONE).

PROMPT

Syntax:

```
!PRO[MPT] [<string>]<CR>
```

where <string> may be:

(a) One or more delimited ASCII strings. A delimited ASCII string consists of a string of any printing ASCII characters (except the command character) with a delimiter at each end. The delimiter may be any printing character except the command character, space, comma, semi-colon, numerals, or letters of the alphabet.

(b) A sequence of ADE values (ASCII Decimal Equivalents) separated by spaces or commas.

(c) Any combination of (a) or (b).

Restrictions on syntax:

The <string> parameter may not define a string of more than ten ASCII characters.

Action:

This command sets the prompt character or string. In buffered mode, the host must send a prompt before the 4025 will respond.

Default:

If <string> is omitted, the prompt string is set to the single character, "line feed."

REPORT

Syntax:

```
!REP[ORT] <device><CR>
```

where <device> is a two-digit number.

Restrictions:

For the <device> parameter, device numbers are as follows:

Device Number	**Device**
00	System Status Block
01	Alphanumeric Cursor
02,03	Not Defined
04-07	Tape Drives 1-4
08-11	Disc Drives 1-4
12,13	Plotters 1,2
14	Printer

Action:

The status of the specified device is returned to the host. Besides the physical devices in the terminal system (such as the display or the tape unit) several logical entities (such as the cursor location in the workspace) may have their status interrogated. The returned status is in the following form:

<command character> ANS <device> <parameter string> <semicolon>

The format of the parameter string depends on the particular device reporting, as seen below. However, for a given device, the number, size, and type of each field is constant, therefore requiring no grammatical analysis by the host program.

The report identifier, ANS (for ANSwer), is followed by one space, then by the two-digit device code. This field and subsequent fields are separated by commas; the last field is followed by a semicolon.

Examples:

!REPORT 00<CR>

!ANS 00,<p1>,<p2>;

where: <p1>
is a 4-digit decimal number giving the number of unused blocks of memory. (A block consists of 16 8-bit bytes of memory.)

<p2>
is a 3-digit decimal number specifying the system status byte. Bit values in hexadecimal are:

01 = Buffered
02 = Form Fillout Mode
04 = Monitor present

!REP 01<CR>

!ANS 01, <p1>,<p2>,<p3>;

where: <p1>
is a 3-digit decimal number giving the row of the workspace in which the cursor is located.

<p2>

is a 3-digit decimal number giving the column of the workspace in which the cursor is located.

<p3>

is a single character, denoting the character at the cursor position.

If no workspace is defined, <p1> AND <p2> are all zeroes, and <p3> is an ASCII "space" character.

RDOWN (Roll Down)

Syntax:

```
!RDO[WN] [<count>]<CR>
```

Action:

This command rolls the current scroll (monitor or workspace) down <count> lines. The action stops when the top line of the scroll is visible on the screen.

Default:

If <count> is omitted, it is assumed to be one.

RIGHT

Syntax:

```
!RIG[HT] [<count>]<CR>
```

Action:

This command causes the cursor to move <count> character positions to the right. If <count> would move the cursor beyond column 80, the cursor goes to the following line and continues moving until a total of <count> locations have been skipped over. Rollup may occur if the cursor was previously at the bottom of the part of the display list in view on the screen.

Default:

If <count> is omitted, it is assumed to be one.

RUP (Roll Up)

Syntax:

```
!RUP [<count>]<CR>
```

Action:

This command causes the current scroll (workspace or monitor) to roll up <count> lines. Action stops when the bottom of the current display list is in view on the screen.

Default:

If <count> is omitted, it is assumed to be one.

RVECTOR (Relative Vector)

(Requires Option 23, 24, 25, or 26.)

Syntax:

```
!RVE[CTOR] <rel x0>, <rel y0>, <rel x1>, <rel y1>, [<rel x2>, <rel
y2>,...,<rel xn>, <rel yn>]<CR>
```

Examples of syntax:

```
!RVECTOR 0,-10,10,0,0,10,-10,0<CR>
!RVE 0,-10,10,0,0,10,-10,0<CR>
!RVE 0,0,10,10<CR>
```

Action:

Draws relative vectors. The parameters for this command are x- and y-coordinates relative to the last vector beam position (the ending point of the last VECTOR or RVECTOR command).

Example:

```
!RVE 0, -10, 10, 10, -10, 10, -10, -10 <CR>
```

Suppose that the last VECTOR or RVECTOR command left the beam at the graphic coordinates (100,100). This RVECTOR command moves the beam to the point (100+0, 100-10) = (100,90). From there, it draws a line segment to (100+10, 90+10) = (110, 100). From there, it draws segments to (100,110) and (90,100), leaving the beam at (90,100). [A subsequent RVECTOR command would specify its coordinates relative to (90,100).] A string of coordinates may be continued to the next line by sending an ampersand "&", as the "last" character of the current line. (Valid for host commands only.)

SEND

Syntax:

```
!SEN[D] [ A[LL] |M[OD] ]<CR>
```

Action:

SEND ALL. If not in form fillout mode, this command sends all data in the workspace to the host computer. Both the text in the workspace and any attribute codes are sent; the attribute codes are encoded as the corresponding ATTRIBUTE commands.

If in form fillout mode, SEND ALL sends all unprotected data. If a field separator has not been defined (see the FIELD command), each unprotected field is sent in its entirety, including trailing spaces. If a field separator has been defined, trailing spaces are suppressed from each field and each field is preceded by a field separator. The last field is followed by a carriage return.

SEND MOD. If not in form fillout mode, this command acts the same as SEND ALL: it sends the entire contents of the workspace to the host, encoding any field attribute codes as ATTRIBUTE commands.

If in form fillout mode, SEND MOD sends all unprotected data which has been modified since the last SEND command and all protected fields with an attribute of M. Each field is preceded by two three-digit numbers, separated by a comma. These numbers specify the row of the workspace, and the character position within that row, where the field begins. For instance, if a modified field starts in Row 5, Column 3, when that field is sent to the computer it will be preceded by "005,003." The first character of the field immediately follows the last digit of the second three-digit number. Trailing blanks and separators are handled as described above for the SEND ALL command.

Default:

If neither ALL nor MOD is specified, ALL is assumed.

SHRINK

(Requires Option 23, 24, 25, or 26.)

Syntax:

```
!SHR[INK] [ Y[ES] | H[ARDCOPY] | B[OTH] | N[O] ]<CR>
```

Examples of syntax:

```
!SHRINK YES<CR>
!SHR Y<CR>
!SHR<CR>
!SHR B<CR>
```

Action:

SHRINK YES. This command causes the 4025 to "shrink" x- and y-coordinates in subsequent VECTOR or RVECTOR commands, multiplying them by a factor of approximately 5/8. This accommodates the 4025 to the range of possible coordinates in 4010-style graphics commands.

In graphics commands for the TEKTRONIX 4010-series terminals, the x-coordinates can be as great as 1023, while the greatest possible x-coordinate in a 4025 VECTOR command is 639. (This would occur in a graphics area which occupied all 80 columns.) Since 5/8 of 1024 is 640, the SHRINK command can accommodate the 4025 to display a 4010-style graphics command file.

To use the 4025 to display a command file written for 4010-series terminals, first dimension the graphics area to hold 35 rows of 80 columns. (!GRA 1,35,1,80 or !GRA 10,44 are possible GRAPHIC commands to accomplish this.) Then give a SHRINK YES command to put the 4025 in "graphics shrink mode."

SHRINK HARDCOPY. This command puts the 4025 in "hardcopy shrink mode." In this mode, the y-coordinates (but not the x-coordinates) in VECTOR and RVECTOR commands are reduced by a factor of 7/8. This pre-distorts graphs drawn on the 4025's screen, so they appear in the proper proportion when copied on a 4631 Hard Copy Unit. (However, character strings inserted in the graphics area with the STRING command will not be shrunk, as SHRINK HARDCOPY affects only the vectors and not the characters.

SHRINK BOTH. This command puts the 4025 in both "graphics shrink" and "hardcopy shrink" mode. The x-coordinates in subsequent VECTOR and RVECTOR commands are multiplied by approximately 5/8, while the y-coordinates are multiplied by approximately 35/64.

SHRINK NO. This command removes the 4025 from both shrink modes.

Default:

If no parameter is specified, YES is assumed.

SHRINK NO is default at power up.

SNOOPY

Syntax:

```
!SNO[OOPY] [ Y[ES] | N[O] ]<CR>
```

Examples of syntax:

```
!SNOOPY<CR>
!SNO<CR>
!SNO Y<CR>
!SNO NO<CR>
```

Action:

SNOOPY YES places the 4025 in "snoopy mode," and SNOOPY NO removes it from that mode. In snoopy mode, the non-printing ASCII control characters are represented on the display by two-letter menmonics. (The "rubout" or "delete" character is represented by a "blotch" of fine diagonal lines.)

Snoopy mode is useful for troubleshooting, as it allows the operator to examine all ASCII characters received on the communications line. It is also useful for inserting control characters into text stored in the workspace. (When the 4025 is not in snoopy mode, control characters typed into the workspace are not stored there, although cursor movement characters or the "bell" character may be immediately executed.) Commands are still executed in snoopy mode.

Restriction:

To see the ASCII "null" character printed when examining incoming data, it is necessary to have "parity" set to "data." (See the PARITY command.)

Default:

If neither parameter is specified, YES is assumed.

At power up, default is SNOOPY NO.

STOPS

Syntax:

```
!STO[PS] [<stop 1> [...<stop n>...]<CR>
```

Restrictions on syntax:

No more than 16 parameters are allowed.

Action:

This command sets the tab stops. Stops are indicated by their column number from 2 to 80 inclusive. All stops must be set at once and must be in ascending order.

Default:

If no parameters are specified, all stops are cleared.

STRING

(Requires Option 23, 24, 25 or 26.)

Syntax:

```
!STR[ING] <string> <CR>
```

where <string> may be:

(a) One or more delimited ASCII strings. A delimited ASCII string consists of a string of any printing ASCII characters (except the command character) with a delimiter at each end. The delimiter may be any printing character except the command character, space, comma, semicolon, numerals, or letters of the alphabet.

(b) A sequence of ADE values (ASCII Decimal Equivalents) separated by spaces or commas.

(c) Any combination of (a) or (b).

Action:

This command is used to put text in a graphic area. The text begins at the cell containing the current beam position. Any vectors or characters that were previously present in the character cells where the string is put are no longer visible, since each character of the string replaces the entire character cell in which it is placed.

Example:

```
!STR |ABC|<CR>
```

Inserts the characters "ABC" in the graphic area.

SYMBOL

(Requires Option 23, 24, 25, or 26.)

Syntax:

```
!SYM[BOL] <symbol number> <font number> <value 1> <value
2>...<value 14><CR>
```

where:

<symbol number> is an integer from 0 to 127,

<font number> is an integer from 1 to 31,

<value n> is an integer from 0 to 255.

Restrictions on syntax:

The <font number> parameter must specify a character font for which graphics RAM (Random Access Memory) is installed. You can ascertain which character fonts have graphics RAM installed by running a GTEST command on the terminal.

Action:

Defines the dot matrix pattern by which a character (denoted> by <symbol number>) of the specified character font (<font number>) is to be displayed. Each symbol is displayed as a dot matrix with 14 rows of 8 dots each. <value n> is the decimal representation of the binary numeral whose individual bits specify the individual dots of row n in the dot matrix.

Example:

Suppose we are defining symbol 97 of character set 24 to represent a Greek letter alpha. We might have the following dot matrix:

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 X 0
0 0 X X 0 X 0 0
0 X 0 0 X 0 0 0
0 X 0 0 X 0 0 0
0 0 X X 0 X 0 0
0 0 0 0 0 0 X 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

(Here the letter X represents a dot which is turned on, and the numeral "0" (zero) represents a dot which is turned off: The fourteen rows of the matrix may be represented by these binary numerals:
00000000, 00000000, 00000000, 00000000, 00000010, 00110100, 01001000, 01001000, 00110100, 00000010, 00000000, 00000000, 00000000, 00000000.
The decimal representations of these numerals are 0,0,0,0,2,52,72,72,52,2,0, 0,0,0.

The SYMBOL command to set symbol number 97 of character font 24 to this matrix, then, is:

```
!SYMBOL 97,24,0,0,0,0,2,52,72,72,52,2,0,0,0,0<CR>
```

Default:

If fewer than 14 rows are specified, the last rows are assumed to be zeroes (i.e., have all their dots turned off).

SYSTAT

Syntax:

```
!SYS[TAT]<CR>
```

Action:

This command causes the 4025 to display in its monitor a list of system parameters. Parameter names are abbreviated in the display as follows:

TB -	transmit baud rate
RB -	receive baud rate
DL -	delay time
LM -	left margin
RM -	right margin
WL -	number of workspace lines displayed on the screen
V# -	firmware version number
TS -	tab stops
CC -	command character
FS -	field separator
PR -	prompt string
EL -	end-of-line string
DU -	duplex ("DU=F" means full duplex, "DU=H" means half duplex.)
EC -	echo ("EC=R" means remote echo, "EC=L" means local echo.)
BU -	buffered mode ("Y" for buffered, "N" for unbuffered.)
FF -	form fillout mode ("Y" for yes, "N" for no.)
SN -	snoopy mode ("Y" for yes, "N" for no.)
KB -	keyboard ("KB=M" means text typed on the keyboard is directed to the monitor, "KB=W" means text from the keyboard is sent to the workspace.)
CM -	communications line ("CM=M" means text from the communications line is directed to the monitor, "CM=W" means such text is sent to the workspace.)
PA -	parity ("N" for "none," "D" for "data," "E" for "even," "O" for "odd.")

Additional SYSTAT parameters:

If Option 1 (Half Duplex) is installed and the 4025 is set for half duplex, the DU field may contain one or two additional letters. See the DUPLEX command description for details.

If Option 10 (Polling Interface) is installed, then an additional field, "PL=", will appear, followed by a two-digit decimal number indicating the receiving address of this display station.

When the 4025 is turned off or reset, it remembers some of the parameter settings listed in the SYSTAT message, and resets others to default settings. Those settings which are remembered are: TB, RB, DL, LM, RM, TS, CC, FS, PR, EL, DU, EC, and PA. (And the PL setting, if present.)

When the 4025 is powered up or reset, it will always be the case that:

(a) WL = 0 (there is no workspace defined)
(b) BU = N (the 4025 is in unbuffered mode)
(c) FF = N (the 4025 is not in form fillout mode)
(d) SN = N (the 4025 is not in snoopy mode)
(e) KB = M and CM = M (text from both the keyboard and the computer is directed into the monitor)

The V# setting will not change unless a different firmware version is installed in the 4025.

TAB

Syntax:

```
!TAB [<count>]<CR>
```

Action:

This command performs the same action as <count> tab characters. For each tab, the cursor is moved to the next tab stop (if not in form fillout mode) or to the beginning of the next unprotected field (in form fillout mode). A tab may also be performed by sending the ASCII tab character.

Default:

If <count> is omitted, it is assumed to be one.

TEST

Syntax:

```
!TES[T]<CR>
```

Action:

1. System ROM (Read Only Memory), system RAM (Random Access Memory), and display RAM are checked. The four system ROM checksums are displayed. An error in display RAM will cause the bad block of memory not to be used. The number of free blocks will therefore be reduced, but the terminal will run correctly.

2. After the memory test, the lights on the four lighted function keys are turned on, all 128 ASCII characters are printed in snoopy mode, and font 1 (the ruling set) is displayed. (If this character set is not installed, each of its characters will be displayed as a dot matrix with every dot turned on.)

   After the two character sets, the seven visual attributes are presented.

3. At the end of the test, the lights on the function keys are turned off and the bell is rung.

   Should the test reveal a failure in the system RAM, the message "RAM ERROR" will appear. In that case, service personnel should be called.

   NOTE

   Running this test destroys any text or key definitions which may have been stored in the 4025's memory.

UP

Syntax:

```
!UP [<count>]<CR>
```

Action:

This command moves the cursor up <count> lines. If <count> is great enough to move the cursor off the visible part of the workspace or monitor scroll, that scroll will be rolled down to keep the cursor in view. Action stops when the cursor reaches the top line in the display list.

Default:

If <count> is omitted, it is assumed to be one.

VECTOR

(Requires Option 23, 24, 25 or 26.)

Syntax:

```
!VEC[TOR] <xo> <yo> <x1> <y1> [<x2> <y2> ... <xn> <yn>]<CR>
```

Action:

This command draws vectors (line segments) in the graphics area of the workspace. The parameters <xo>,<yo> specify the x- and y-coordinates of the starting point for the first line segment; <x1>,<y1> specify that segment's end point. If a third pair of coordinates <x2>,<y2> are specified, then a second segment will be drawn starting at the end of the first segment and continuing to the point specified by <x2>,<y2>. Additional pairs of coordinates cause additional line segments to be drawn in a like manner.

The x- and y-coordinates refer to the position of each point in the dot matrix which comprises the graphic area. (In order for the VECTOR command to operate, such a graphic area must previously have been defined with a GRAPHIC command.) Each cell of the graphic area is an 8 x 14 dot matrix. Consequently, if the graphic area includes ten rows of cells, each with 14 rows of dots, it will have 140 rows of dots, numbered in y-coordinate from 0 to 139. (Y-coordinate zero specifies the bottom row, and y-coordinate 139 the top row.) If the graphics area is 50 cells wide, it will likewise have 400 columns of dots, numbered in x-coordinate from 0 to 399.

Each pair of coordinates in a VECTOR command must specify a point within the graphic area; otherwise the line segments to that point will not be drawn.

VRULE (Vertical Rule)

(Requires Option 32.)

Syntax:

```
!VRU[LE] <row> <column> [ <length> [<width>] ]<CR>
```

Restrictions on syntax:

All parameters are positive integers. The <column> parameter must not exceed 80, and the <width> parameter, if specified, must be either 1 or 2.

Action:

Draws a vertical ruling in the workspace. The first ruling character is inserted at the row and column specified by the <row> and <column> parameters, and the ruling continues downward for a total of <length> character cells. The ruling is a single line if <width> is one, and a double line if <width> is two.

Default:

If <length> or <width> are omitted, they are assumed to be one.

WORKSPACE

Syntax:

```
!WOR[KSPACE] [<number>] [ H[OST] ] [ K[EYBOARD] ]<CR>
```

Restriction on syntax:

The <number> parameter, if specified, must be an integer in the range from 0 to 33. This always leaves at least one line for the monitor.

Action:

If the <number> parameter is included, this command erases both the workspace (if one is defined) and the monitor. It then defines a workspace, and allots the top <number> lines of the screen for displaying that workspace. The remaining lines below are used to display the monitor.

If H (for Host) is specified, text from the host computer will be directed to the workspace.

If K (for Keyboard) is specified, text from the keyboard will be directed to the workspace. (However, commands to the 4025 which are typed on the keyboard will still be displayed in the monitor.)

Default:

If no parameters are specified, and the command comes over the communications line from the host computer, it will be executed as a WORKSPACE H command. If no parameters are specified and the command is typed on the keyboard, it is executed as a WORKSPACE K command.

If <number> is the only parameter specified, then a workspace is defined, but the direction of text from the keyboard and the host is not changed. However, a WORKSPACE 0 command directs text from both the keyboard and the host to the monitor (since no workspace is defined).

PROGRAMMING THE COMPUTER TO CONTROL THE 4025

The person who programs a computer to control the 4025 may use almost any programming language. The only requirement is that the language have an instruction to display literal information. Common examples are the WRITE statement in FORTRAN, the PRINT statement in BASIC, or the DISPLAY statement in COBOL.

Commands to the 4025 are sent as literal information in a programming language instruction, as though the command were to be displayed on the 4025 screen. When the 4025 sees the command character, however, it assumes a command follows, and looks for one. When it recognizes a command, it executes that command instead of displaying it. The command character is the vital ingredient. For example, the BASIC statement:

```
100 PRINT "!WOR 20 K"
```

will create a workspace of 20 lines, and direct text from the keyboard into the workspace. However, the BASIC statement:

```
200 PRINT "WOR 20 K"
```

will cause the text:

```
WOR 20 K
```

to be displayed on the 4025's screen. The difference is the presence of the command character "!" in line 100. In that line, "!" identifies the text:

```
WOR 20 K
```

as a command to be executed. In line 200, there is no command character, and:

```
WOR 20 K
```

is treated simply as text to be displayed on the screen.

Examples:

The BASIC instruction:

```
100 PRINT "!WOR 20 K;!STOPS 5,10,20,60;!BUF YES;"
```

creates a workspace of 20 lines, directs text from the keyboard into the workspace; creates tab stops at columns 5, 10, 20 and 60; and places the 4025 in buffered mode.

In our sample COBOL program (Appendix C), when TFFO-7 is displayed (lines 018600-018700) the following string of characters is received by the 4025:

```
!JUM7;!ATT PS;   OUR ACCT!ATT AE;    !ATT PS;
```

This string of characters causes the following events occur:

The cursor moves to the beginning of line 7 and defines a protected field with standard visual attributes.

The text "   OUR ACCT" is placed in that field.

A new field is begun at the character position after the "T" in "ACCT," with attributes of alphanumeric, enhanced.

Several spaces are entered in that field.

Finally, the rest of that line is defined as a field with the attributes of protected, standard.

Invalid Commands

If the 4025 receives an invalid command, the results depend on the origin of the invalid command. In the following examples suppose the command keyword "STOPS" is misspelled "STEPS".

If the invalid command:

```
!STEPS 20,40,60; <CR>
```

is given from the keyboard, an error message is printed and the invalid command repeated:

```
WHAT?
!STEPS 20,40,60;
```

Suppose this same invalid command is sent from the host in the following BASIC statement:

```
100 PRINT "!STEPS 20,40,60;"
```

The invalid command:

```
!STEPS 20,40,60
```

will be treated as text and printed in whichever scroll prints text from the host.

Suppose this same invalid command is part of a string of commands. If the command string comes from the keyboard, commands which precede the invalid command will be executed. The rest of the line will be printed, with an error message.

For example, if the command string:

```
!ERA W;!STEPS 20,40,60; !BEL; <CR>
```

is entered from the 4025 keyboard, the following events occur:

(a) The workspace is erased; and

(b) WHAT?
!STEPS 20,40,60;!BEL;

is printed in the monitor.

However, suppose this command string is sent from the host, as in the BASIC statement:

```
100 PRINT "!ERA W;!STEPS 20,40,60;!BEL;"
```

then the following events occur:

(a) The workspace is erased;

(b) The text

!STEPS 20,40,60;

is printed in whichever scroll prints text from the host; and

(c) The terminal's bell rings.

Note also that if the 4025 receives a command that requires a workspace, when no workspace is defined, that command string will simply cease to be. Nothing will be executed, and no error message will appear.

How, then, does one display commands on the 4025 screen in order to read, debug, or modify an existing program? The 4025 *operator* must place the 4025 in COMMAND LOCKOUT mode by pressing the COMMAND LOCKOUT key. This prevents the 4025 from executing any commands. Instead, all information that the 4025 receives from the computer is treated as text, and all printed characters are displayed on the screen. This may be used with SNOOPY mode to examine all information received, both printed and non-printed characters, since in SNOOPY mode each non-printed ASCII character is assigned a printed two-letter mnemonic. Note that the SNOOPY YES command may be given from the computer, but the COMMAND LOCKOUT mode cannot be entered or left by computer command. Thus it will be the 4025 operator who sets up the 4025 to read, modify and debug programs containing 4025 commands.

See Appendix C for a sample program in COBOL using the form fillout features of the 4025.

Section 3

4025 OPTIONS

OPTIONAL CHARACTER SETS

With the proper options, the 4025 can accommodate up to 32 different fonts, each containing up to 128 characters. Thirty of these fonts may be user-defined. The standard font is Font 0, and cannot be modified by the 4025 operator or by the host. In addition to the standard font, two other pre-defined fonts are available: Math Characters (Option 34) and Rulings Characters (Option 32). (See the following discussion of rulings for details on this font.) Fonts other than the standard font are selected using the ATTRIBUTE command.

The SYMBOL command is used to define alternate fonts, one symbol at a time. The DFONT (Delete Font) command deletes a user-defined font that is no longer needed. User-definable fonts that are not currently programmed with alternate character sets can be used to store and recall graphic information.

See Section 2, "Command Descriptions" for proper use of these commands.

Rulings

Option 32 for the 4025 provides a set of rulings characters. Rulings provide a way of visually outlining the structure of a form. A ruling line is either a single or double line, either horizontal or vertical, and occupies a row or column of character cells.

There are two ways to create rulings:

1. Use the HRULE and VRULE commands; or

2. Use the ATTRIBUTE command to select Font 1 (which is always the rulings font on the 4025), and select the appropriate ruling characters from Font 1. (See Appendix A, Table 1.)

When drawing rulings on the 4025 display using the HRULE and VRULE commands, vertical and horizontal rulings generally will not meet properly. There will be gaps or unwanted crossings. The variety of ruling characters provided in Option 32 allows the operator or programmer to make neat, well-fitted junctions by selecting the appropriate characters. Appendix A, Table 1, shows the correspondence between ASCII Decimal Equivalents (ADE), Font 0 characters, and Font 1 characters. Appendix A, Table 2, provides a graphic reference sheet for making the proper junctions. A detailed example is found in the 4025 Operator's Manual.

NOTE

Referring to Table 2, Appendix A, note, for example, that both the ASCII "NULL" (ADE 000) and the ASCII "@" (ADE 064) give the same ruling character in Font 1. The same holds throughout the table; two different ASCII characters will generate the same ruling character. It is strongly recommended that the operator or programmer use the ASCII characters with ADE's 064-126 to generate ruling characters in Font 1. Problems may arise when ADE's 000-063, and 127 are used, depending on the internal configuration of the particular terminal.

OPTIONAL INTERFACES

Several optional interfaces are available for the 4025. These make possible many useful system configurations using the 4025.

Current Loop Interface (Option 2) - This allows the 4025 to operate on a 20 mA current loop.

Polling Interface (Option 10) - This option allows the 4025 to operate as a polling display station in an IBM 3270 multidrop environment.

Polling Controller (Option 11) - This option converts the 4025 into a polling controller for up to eight 4025 display stations in an IBM 3270 environment. The Option 11 plugs into one of the 4025 display stations and controls all the 4025's in the group, including the 4025 in which it resides.

Half Duplex (Option 1) - When the half duplex option is installed, the DUPLEX command is added to the terminal command set.

RS-232 Peripheral Interface (Option 3) - This interface allows the 4025 to communicate with a 4662 Plotter and a 4642 Printer, and direct data from the host to one of these devices. (See Hard Copy Section below.)

GPIB Peripheral Interface (Option 4) - In addition to routing data between the host and the various peripherals, the terminal will be able to perform some operations involving interaction between the terminal itself and various peripherals. These include:

Creating command files on tape and then using them to drive the terminal.

Transferring files from one mass storage device to another.

Spooling pictures onto mass storage and then routing them to the display or plotter.

HARD COPIES

The 4025 Computer Display Terminal is compatible with 4631 Hard Copy Unit and 4642 Printer operations.

To make copies on the 4631 Hard Copy Unit, the 4025 must have Option 40 (Hard Copy and Video Out) installed. The 4631 Hard Copy Unit can copy whatever is in the workspace, in the monitor, or on the 4025 screen, including alternate font characters, rulings, and graphic display. Copies on the 4631 are made using the HCOPY command. Note that the 4631 copies 53 lines of text, which is more than the 4025 screen is able to display at one time. Text in the 4025 display memory need not be displayed on the screen to be copied. The 4631 copies are approximately 8-1/2" x 11".

To make copies on the 4642 Printer, the 4025 must have Option 3 (RS-232 Peripheral Interface) installed. The 4642 Printer will make printed copies of data from either the 4025 workspace or from the host. The COPY command is used to do this. The 4642 cannot print graphics, however, and all rulings characters are replaced with asterisks. A further note of caution is needed. The 4025 will continue to copy data from the host computer onto the 4642 until one of three things happens: (1) The computer sends an "end-of-file" string, telling the 4025 to stop copying; (2) The 4025, in buffered mode, receives a prompt string from the host; or (3) The operator types on the 4025 keyboard with the 4025 in unbuffered mode.

GRAPHICS

If the 4025 has a Graphics Memory Option (Option 23, 24, 25 or 26), the operator or programmer can create a graphics area in which a variety of vector types can be drawn.

To set up a graphics area in the workspace, use the GRAPHIC command. To draw vectors in the graphics area, use the VECTOR and RVECTOR commands. To specify the type of vector to be drawn, use the LINE command, and to erase the vectors in the graphics area, use the ERASE G command. See Section 2, "Command Descriptions" for a complete discussion of these commands.

4010-Style Graphics

The 4025 with a Graphics Memory Option, will accept 4010-style graphic commands when these commands are sent from the host. (The 4025 will not accept 4010-style graphics commands entered from its keyboard.) The 4010-style graphics are characterized by addressable screen coordinates and the use of ASCII characters which encode these addresses.

To enable the 4025 to respond properly to 4010-style graphic commands, type:

```
!GRAPHIC 1,35 <CR>
!SHRINK <CR>
```

This sets up a graphics area which is the right shape for holding a 4010-style graphics display. In particular, the graphics area is approximately 1024X by 780Y, in terms of screen coordinates.

The commands that the 4025 will accept from the host are as follows:

1. The GS command places the 4025 in 4010-style graph mode.
2. The US command exits the 4025 from graph mode, and positions the cursor at the character cell nearest the final beam position in graph mode.
3. The ESC command causes the 4025 to exit 4010-emulation in order to handle escape sequences.
4. The ESC-Form Feed command exits the 4025 from graph mode, erases the display, and moves the cursor to the home position.

Addressing the Display Beam

The beam is addressed to a point in the graphics area by sending to the terminal the binary equivalents of the Y address and the X address of the point. Each binary equivalent must be separated into two bytes: the 5 most significant bits and the 5 least significant bits. For example, 205Y,148X translates to 0011001101 Y,0010010100X (binary). The 0011001101Y becomes 00110 HiY and 01101 LoY; the 0010010100 becomes 00100 HiX and 10100 LoX. If these bytes are sent to the terminal in the GS mode, the beam will move to the 205Y,148X position in the graphics area. These bytes are encoded as ASCII equivalents. See the Coordinate Conversion Chart in Appendix A. In this example, 205Y,148X would be addressed in GS mode as "&M$T."

Graph Mode Memory

When an address is sent to the terminal, the HiY, LoY and HiX bytes are stored in a register. Therefore, if the next address sent to the terminal repeats some of these bytes, they need not be reentered from the terminal. (The LoX must always be sent, since it activates the move or draw.) Even if the 4025 leaves graph mode and reenters it later, these three bytes are retained. The following table shows which bytes must be sent in response to specific byte changes.

Bytes Which Change	Bytes Which Must be Sent			
	Hi Y	Lo Y	Hi X	Lo X
Hi Y	#			#
Lo Y		#		#
Hi X		#	#	#
Lo X				#

Graphics Creation

After receiving the GS command, the 4025 waits for the 4010-style codes. If the GS command is followed by BEL (ADE 7) the first LoX causes a draw. If no BEL is received, the first LoX causes a move (dark vector). HiX/Y's are taken to be HiY until a LoY is found, after which they are taken to be HiX's. Subsequent LoY's replace the old LoY (as on the 4010) so the extra bits are ignored. LoX's cause the next vector to be drawn. The last coordinates in 4010-mode are remembered until the next time GS mode is entered.

Thus a US command can appear, followed by a GS command with fewer than 4 coordinate characters. Control characters between a GS command and the next US command are swallowed, except for ESCAPE and ESCAPE-Form Feed, as noted previously.

When 4010-mode is exited, the communications port is returned to the portion of display memory it was in before entering graph mode (workspace or monitor).

Appendix A

RULINGS CHARACTERS

Table A-1

ALTERNATE CHARACTER FONTS FOR THE 4025

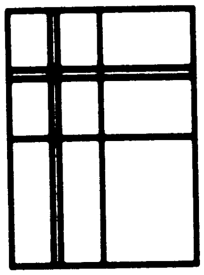
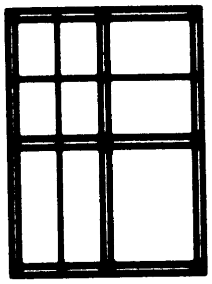
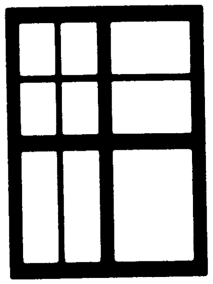
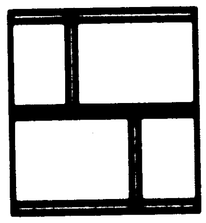
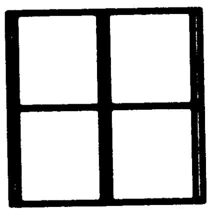
FONT 0: Standard Character Set FONT 1: Rulings Characters

(NOTE: Characters numbered 000 to 031, and character 127, are "control" characters and are not ordinarily displayed. When making rulings, for these characters, use characters 064 through 095, and character 063.)

Decimal Values	Fonts 0	1	Decimal Values	Fonts 0	1	Decimal Values	Fonts 0	1	Decimal Values	Fonts 0	1
000 064	@	┌	016 080	P	└	032 096	\`		048 112	0 p	╘
001 065	A	┬	017 081	Q	┴	033 097	! a	╓	049 113	1 q	╨
002 066	B	┐	018 082	R	┘	034 098	" b	╖	050 114	2 r	╛
003 067	C	╤	019 083	S	╧	035 099	# c	╤	051 115	3 s	╧
004 068	D	╦	020 084	T	╩	036 100	$ d	█	052 116	4 t	█
005 069	E	╦	021 085	U	╩	037 101	% e	█	053 117	5 u	█
006 070	F	╥	022 086	V	╩	038 102	& f	█	054 118	6 v	█
007 071	G	╥	023 087	W	╨	039 103	' g	█	055 119	7 w	█
008 072	H	├	024 088	X	╫	040 104	(h	╒	056 120	8 x	█
009 073	I	┼	025 089	Y	─	041 105	) i	╙	057 121	9 y	█
010 074	J	┤	026 090	Z	╫	042 106	* j	╕	058 122	: z	█
011 075	K	╪	027 091	[	│	043 107	+ k	╪	059 123	; {	█
012 076	L	╫	028 092	\	╞	044 108	, l	█	060 124	< |	╞
013 077	M	╫	029 093	]	═	045 109	- m	█	061 125	= }	■
014 078	N	╫	030 094	^	╡	046 110	. n	█	062 126	> ~	╡
015 079	O	╫	031 095	_	║	047 111	/ o	█	063 127	?	█

Table A-2

RULING JUNCTIONS

Rulings (Font 1)	Standard (Font 0)
	@YYGYYAYYYYYB [_ [[\]]M]]K]]]]]^ [_ [[HYYOYYIYYYYYJ [_ [[[_ [[[_ [[PYYWYYQYYYYYR
	D]]C]]E]]]]]F _ [_ _ XYYIYYOYYYYYZ _ [_ _ L]]K]]M]]]]]N _ [_ _ _ [_ _ _ [_ _ T]]S]]U]]]]]V
	d}}c}}e}}}}}f ? [? ? xYYIYYoYYYYYz ? [? ? l}}k}}m}}}}}n ? [? ? ? [? ? ? [? ? t}}s}}u}}}}}v
	h]]]E]]]]]]]j [_ [[_ [:}}}y}}}(}}}~ [_ [[_ [p]]]]]]]U]]]r
	a]]]]]g]]]]]b _ ? _ _ ? _ XYYYYYoYYYYYX _ ? _ _ ? _ qYYYYYwYYYYYi

Appendix B

4010-STYLE GRAPHICS

Table B-1

COORDINATE CONVERSION CHART

Low Order X		X or Y Coordinate								Low Order Y	
ASCII	DEC.									DEC.	ASCII
@	64	0	32	64	96	128	160	192	224	96	`
A	65	1	33	65	97	129	161	193	225	97	a
B	66	2	34	66	98	130	162	194	226	98	b
C	67	3	35	67	99	131	163	195	227	99	c
D	68	4	36	68	100	132	164	196	228	100	d
E	69	5	37	69	101	133	165	197	229	101	e
F	70	6	38	70	102	134	166	198	230	102	f
G	71	7	39	71	103	135	167	199	231	103	g
H	72	8	40	72	104	136	168	200	232	104	h
I	73	9	41	73	105	137	169	201	233	105	i
J	74	10	42	74	106	138	170	202	234	106	j
K	75	11	43	75	107	139	171	203	235	107	k
L	76	12	44	76	108	140	172	204	236	108	l
M	77	13	45	77	109	141	173	205	237	109	m
N	78	14	46	78	110	142	174	206	238	110	n
O	79	15	47	79	111	143	175	207	239	111	o
P	80	16	48	80	112	144	176	208	240	112	p
Q	81	17	49	81	113	145	177	209	241	113	q
R	82	18	50	82	114	146	178	210	242	114	r
S	83	19	51	83	115	147	179	211	243	115	s
T	84	20	52	84	116	148	180	212	244	116	t
U	85	21	53	85	117	149	181	213	245	117	u
V	86	22	54	86	118	150	182	214	246	118	v
W	87	23	55	87	119	151	183	215	247	119	w
X	88	24	56	88	120	152	184	216	248	120	x
Y	89	25	57	89	121	153	185	217	249	121	y
Z	90	26	58	90	122	154	186	218	250	122	z
[	91	27	59	91	123	155	187	219	251	123	{
\	92	28	60	92	124	156	188	220	252	124	¦
]	93	29	61	93	125	157	189	221	253	125	}
^	94	30	62	94	126	158	190	220	254	126	~
_	95	31	63	95	127	159	191	223	255	127	RUBOUT (DEL)
DEC. →		32	33	34	35	36	37	38	39		
ASCII →		SP	!	"	#	$	%	&	'		
		High Order X & Y									

Coordinate conversion chart, part 1 of 4. INSTRUCTIONS: Find coordinate value in body of chart; follow that column to bottom of chart to find decimal value or ASCII character which represents the High Y or High X byte; go to the right in the row containing the coordinate value to find the Low Y byte, or go to the left to find the Low X byte. EXAMPLE: 200Y, 48X equals 38 104 33 80 in decimal code, and equals & h ! P in ASCII code.

Table B-1 (cont.)

COORDINATE CONVERSION CHART

Low Order X		X or Y Coordinate								Low Order Y	
ASCII	DEC.									DEC.	ASCII
@	64	256	288	320	352	384	416	448	480	96	`
A	65	257	289	321	353	385	417	449	481	97	a
B	66	258	290	322	354	386	418	450	482	98	b
C	67	259	291	323	355	387	419	451	483	99	c
D	68	260	292	324	356	388	420	452	484	100	d
E	69	261	293	325	357	389	421	453	485	101	e
F	70	262	294	326	358	390	422	454	486	102	f
G	71	263	295	327	359	391	423	455	487	103	g
H	72	264	296	328	360	392	424	456	488	104	h
I	73	265	297	329	361	393	425	457	489	105	i
J	74	266	298	330	362	394	426	458	490	106	j
K	75	267	299	331	363	395	427	459	491	107	k
L	76	268	300	332	364	396	428	460	492	108	l
M	77	269	301	333	365	397	429	461	493	109	m
N	78	270	302	334	366	398	430	462	494	110	n
O	79	271	303	335	367	399	431	463	495	111	o
P	80	272	304	336	368	400	432	464	496	112	p
Q	81	273	305	337	369	401	433	465	497	113	q
R	82	274	306	338	370	402	434	466	498	114	r
S	83	275	307	339	371	403	435	467	499	115	s
T	84	276	308	340	372	404	436	468	500	116	t
U	85	277	309	341	373	405	437	469	501	117	u
V	86	278	310	342	374	406	438	470	502	118	v
W	87	279	311	343	375	407	439	471	503	119	w
X	88	280	312	344	376	408	440	472	504	120	x
Y	89	281	313	345	377	409	441	473	505	121	y
Z	90	282	314	346	378	410	442	474	506	122	z
[	91	283	315	347	379	411	443	475	507	123	{
\	92	284	316	348	380	412	444	476	508	124	¦
]	93	285	317	349	381	413	445	477	509	125	}
^	94	286	318	350	382	414	446	478	510	126	~
_	95	287	319	351	383	415	447	479	511	127	RUBOUT (DEL)
DEC. →		40	41	42	43	44	45	46	47		
ASCII →		(	)	*	+	,	–	.	/		
		High Order X & Y									

Coordinate conversion chart, part 2 of 4. (Refer to part 1 for interpretation instructions.)

Table B-1 (cont.)

COORDINATE CONVERSION CHART

Low Order X		X or Y Coordinate								Low Order Y	
ASCII	DEC.									ASCII	DEC.
@	64	512	544	576	608	640	672	704	736	`	96
A	65	513	•545	577	609	641	673	705	737	a	97
B	66	514	546	578	610	642	674	706	738	b	98
C	67	515	547	579	611	643	675	707	739	c	99
D	68	516	548	580	612	644	676	708	740	d	100
E	69	517	649	581	613	645	677	709	741	e	101
F	70	518	550	582	614	646	678	710	742	f	102
G	71	519	551	583	615	647	679	711	743	g	103
H	72	520	552	584	616	648	680	712	744	h	104
I	73	521	553	585	617	649	681	713	745	i	105
J	74	522	554	586	618	650	682	714	746	j	106
K	75	523	555	587	619	651	683	715	747	k	107
L	76	524	556	588	620	652	684	716	748	l	108
M	77	525	557	589	621	653	685	717	749	m	109
N	78	526	558	590	622	654	686	718	750	n	110
O	79	527	559	591	623	655	687	719	751	o	111
P	80	528	560	592	624	656	688	720	752	p	112
Q	81	529	561	593	625	657	689	721	753	q	113
R	82	530	562	594	626	658	690	722	754	r	114
S	83	531	563	595	627	659	691	723	755	s	115
T	84	532	564	596	628	660	692	724	756	t	116
U	85	533	565	597	629	661	693	725	757	u	117
V	86	534	566	598	630	662	694	726	758	v	118
W	87	535	567	599	631	663	695	727	759	w	119
X	88	536	568	600	632	664	696	728	760	x	120
Y	89	537	569	601	633	665	697	729	761	y	121
Z	90	538	570	602	634	666	698	730	762	z	122
[	91	539	571	603	635	667	699	731	763	{	123
\	92	540	572	604	636	668	700	732	764	¦	124
]	93	541	573	605	637	669	701	733	765	}	125
^	94	542	574	606	638	670	702	734	766	~	126
_	95	543	575	607	639	671	703	735	767	RUBOUT (DEL)	127
DEC ⟶		48	49	50	51	52	53	54	55		
ASCII ⟶		0	1	2	3	4	5	6	7		
		High Order X & Y									

Coordinate conversion chart, part 3 of 4. (Refer to part 1 for interpretation instructions.)

Table B-1 (cont.)

COORDINATE CONVERSION CHART

Low Order X		X or Y Coordinate								Low Order Y	
ASCII	DEC.									DEC.	ASCII
@	64	768	800	832	864	896	928	960	992	96	`
A	65	769	801	833	865	897	929	961	993	97	a
B	66	770	802	834	866	898	930	962	994	98	b
C	67	771	803	835	867	899	931	963	995	99	c
D	68	772	804	836	868	900	932	964	996	100	d
E	69	773	805	837	869	901	933	965	997	101	e
F	70	774	806	838	870	902	934	966	998	102	f
G	71	775	807	839	871	903	935	967	999	103	g
H	72	776	808	840	872	904	936	968	1000	104	h
I	73	777	809	841	873	905	937	969	1001	105	i
J	74	778	810	842	874	906	938	970	1002	106	j
K	75	779	811	843	875	907	939	971	1003	107	k
L	76	780	812	844	876	908	940	972	1004	108	l
M	77	781	813	845	877	909	941	973	1005	109	m
N	78	782	814	846	878	910	942	974	1006	110	n
O	79	783	815	847	879	911	943	975	1007	111	o
P	80	784	816	848	880	912	944	976	1008	112	p
Q	81	785	817	849	881	913	945	977	1009	113	q
R	82	786	818	850	882	914	946	978	1010	114	r
S	83	787	819	851	883	915	947	979	1011	115	s
T	84	788	820	852	884	916	948	980	1012	116	t
U	85	789	821	853	885	917	949	981	1013	117	u
V	86	790	822	854	886	918	950	982	1014	118	v
W	87	791	823	855	887	919	951	983	1015	119	w
X	88	792	824	856	888	920	952	984	1016	120	x
Y	89	793	825	857	889	921	953	985	1017	121	y
Z	90	794	826	858	890	922	954	986	1018	122	z
[	91	795	827	859	891	923	955	987	1019	123	{
\	92	796	828	860	892	924	956	988	1020	124	¦
]	93	797	829	861	893	825	957	989	1021	125	}
^	94	798	830	862	894	926	958	990	1022	126	~
_	95	799	831	863	895	927	959	991	1023	127	RUBOUT (DEL)
DEC →		56	57	58	59	60	61	62	63		
ASCII →		8	9	:	;	<	=	>	?		
		High Order X & Y									

Coordinate conversion chart, part 4 of 4. (Refer to part 1 for interpretation instructions.)

Table B-2
ASCII CODE CHART

BITS B7 B6 B5				Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4	B3	B2	B1	CONTROL		HIGH X & Y GRAPHIC INPUT		LOW X		LOW Y	
Ø	Ø	Ø	Ø	NUL Ø	DLE 16	SP 32	Ø 48	@ 64	P 8Ø	` 96	p 112
Ø	Ø	Ø	1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
Ø	Ø	1	Ø	STX 2	DC2 18	″ 34	2 50	B 66	R 82	b 98	r 114
Ø	Ø	1	1	ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
Ø	1	Ø	Ø	EOT 4	DC4 2Ø	$ 36	4 52	D 68	T 84	d 1ØØ	t 116
Ø	1	Ø	1	ENQ 5	NAK 21	% 37	5 53	E 69	U 85	e 1Ø1	u 117
Ø	1	1	Ø	ACK 6	SYN 22	& 38	6 54	F 7Ø	V 86	f 1Ø2	v 118
Ø	1	1	1	BEL 7 BELL	ETB 23	′ 39	7 55	G 71	W 87	g 1Ø3	w 119
1	Ø	Ø	Ø	BS 8 BACK SPACE	CAN 24	(40	8 56	H 72	X 88	h 1Ø4	x 12Ø
1	Ø	Ø	1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 1Ø5	y 121
1	Ø	1	Ø	LF 1Ø	SUB 26	* 42	: 58	J 74	Z 9Ø	j 1Ø6	z 122
1	Ø	1	1	VT 11	ESC 27	+ 43	; 59	K 75	[91	k 1Ø7	{ 123
1	1	Ø	Ø	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 1Ø8	¦ 124
1	1	Ø	1	CR 13 RETURN	GS 29	- 45	= 61	M 77	] 93	m 1Ø9	} 125
1	1	1	Ø	SO 14	RS 3Ø	. 46	> 62	N 78	∧ 94	n 11Ø	~ 126
1	1	1	1	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	RUBOUT (DEL) 127

APPENDIX C

SAMPLE COBOL PROGRAM

```
000100    IDENTIFICATION DIVISION.
000020    PROGRAM-ID. FORM.
000300    REMARKS.
000400*
000500**          THIS IS A SAMPLE PROGRAM WHICH SENDS A FORM
000600**          TO THE 4025 AND PROCESSES THE DATA RETURNED.
000700**
000800**  << INPUT/OUTPUT ASSUMPTIONS >>
000900
001000**  THIS PROGRAM USES THE 'ACCEPT' AND 'DISPLAY' VERBS TO
001100**  COMMUNICATE WITH THE 4025 VIDEO TERMINAL. FILE OUTPUT
001200**  IS AS PER USUAL.
001300**
001400**  THE INPUT FROM THE 4025 IS VARIABLE IN LENGTH WITH
001500**  THE FIRST 7 CHARACTERS ALWAYS PRESENT AND IN THE
001600**  SAME FORMAT.
001700**
001800**  FROM 4025:  ROW,COL.........DATA.......
001900**              !   !           !        !
002000**              !   !           !        !_END OF DATA
002100**              !   !           !________START OF DATA
002200**              !   !_______________COLUMN DATA STARTS IN
002300**              !___________________ROW DATA STARTS IN
002400**
002500**  THIS PROGRAM WAS ORIGINALLY DEVELOPED ON A
002600**  DECSYSTEM-10 AND WITH MODIFICATIONS TO THE 'SELECT'
002700**  AND 'FD' STATEMENTS AND CHANGING THE 'DISPLAY-7' TO
002800**  WHATEVER DISPLAY ALLOWS INPUT OF UPPER AND LOWER CASE
002900**  CHARACTERS SHOULD CONVERT IT TO ANOTHER SYSTEM.
005200    ENVIRONMENT DIVISION.
005300    INPUT-OUTPUT SECTION.
005400    FILE-CONTROL.
005500
005600            SELECT   TEK-FORM
005700                     ASSIGN TO DSK
005800                     ACCESS MODE IS SEQUENTIAL
005900                     PROCESSING MODE IS SEQUENTIAL.
```

```


 DATA DIVISION.



 FILE SECTION.

 FD TEK-FORM
          VALUE OF ID IS 'DATA SEQ'
          DATA RECORD IS TEK-FORM-DATA.
 01 TEK-FORM-DATA              USAGE IS DISPLAY-7.
     02 TF-REF                     PIC X(10).
     02 TF-CUST.
          03 TF-CUST-1             PIC X(4).
          03 TF-CUST-2             PIC X(10).
     02 TF-INSTRUMENT              PIC X(8).
     02 TF-VALUE                   PIC X(8).
     02 TF-TYPE                    PIC X(5).
     02 TF-FC-AMOUNT               PIC X(13).
     02 TF-RATE                    PIC X(12).
     02 TF-DOLLARS                 PIC X(15).
     02 TF-OUR-ACCT                PIC X(4).
     02 TF-CONTRACT-NR             PIC X(8).
     02 TF-CREDIT-REFERENCE        PIC X(12).
     02 TF-CHECK-OK                PIC X(1).

 WORKING-STORAGE SECTION.


 01 TEK-FORM-FILL-OUT       USAGE IS DISPLAY-7
*==============

     02 TFFO-0              PIC X(33) VALUE IS
     '!FOR N;!BUF;!WOR 24 K H;!JUM;!ERA'.
     02 TFFO-1              PIC X(12) VALUE IS
     '!JUM;!ATT IP'.

     02 TFFO-1A             PIC X(19) VALUE IS
     '!JUM 1,11;T   E   K   '.

     02 TFFO-1B             PIC X(33) VALUE IS
     '!JUM 1,27;SAMPLE OF FORM FILL OUT'.
```

```
013600
013700         02 TFFO-1C              PIC X(27) VALUE IS
013800         '!JUM 1,79; !ATT PS;!JUM 3,1'.
013900*
014000         02 TFFO-3               PIC X(19) VALUE IS
014100         '!ATT P;REF !JUM 3,6'.
014200
014300         02 TFFO-3A              PIC X(27) VALUE IS
014400         '!ATT EA;       !JUM 3,16'.
014500
014600         02 TFFO-3B              PIC X(36) VALUE IS
014700         '!ATT PS;     CUST !ATT EA;      !ATT PS'.
014800
014900         02 TFFO-3C              PIC X(27) VALUE IS
015000         ' !ATT EA;!JUM 3,39; !ATT PS'.
015100
015200         02 TFFO-3D              PIC X(46) VALUE IS
015300         '   INSTRUMENT !ATT EA;       !ATT PS;!JUM 3,61'.
015400
015500         02 TFFO-3E              PIC X(33) VALUE IS
015600         ' VALUE DT !ATT EA;       !ATT PS'.
015700
015800*
015900*    LINE 5
016100*
016200         02 TFFO-5               PIC X(55) VALUE IS
016300         '!JUM 5;!ATT PS;   TYPE   !ATT EA;   !JUM 5,17;!ATT PS'.
016400
016500         02 TFFO-5A              PIC X(45) VALUE IS
016600         ' F/C  AMT !JUM 5,27;!ATT AE;!JUM 5,40; !ATT PS'.
016700
016800         02 TFFO-5B              PIC X(33) VALUE IS
016900         ' RATE !ATT EA;!JUM 5,60; !ATT PS'.
017000
017100         02 TFFO-5C              PIC X(30) VALUE IS
017200         '    $ !ATT EN!JUM 5,79; !ATT PS'.
017300
018000*
018100*    LINE 7
018200*
018400*
018500
018600         02 TFFO-7               PIC X(45) VALUE IS
```

```
018700          '!JUM 7;!ATT PS;   OUR ACCT !ATT AE;     !ATT PS'.
018800
018900          02 TFFO-7A              PIC X(30) VALUE IS
019000          ' CONT# !ATT AE;       !ATT PS'.
019100
019200          02 TFFO-7B              PIC X(46) VALUE IS
019300          ' CREDIT REFERENCE# !ATT AE;!JUM 7,62; !ATT PS'.
019400
019500          02 TFFO-7C              PIC X(29) VALUE IS
019600          '    CHECK OK !ATT AE; !ATT PS'.
019700
030100*
030200*     LINE 23: SPECIAL END OF SCREEN DATA FLAG
030300*
030400          02 TFFO-EOD-FLAG        PIC X(45) VALUE IS
030500          '!JUM 23;!ATT PM;01'.
030600*
030700*     FORM END
030800*
030900          02 TFFO-END             PIC X(13) VALUE IS
031000          '!JUM 3,6;!FOR'.
031100
031200      01 TERMINAL-DATA            USAGE IS DISPLAY-7.
031300*=============
031400
031500              02 TD-LOCATION.
031600              03 TD-LOC-ROW       PIC X(3).
031700              03 FILLER           PIC X.
031800              03 TD-LOC-COL       PIC X(3).
031900              02 TD-DATA                     PIC X(40).
032000
032100          01 INDICATORS.
032200*=============
032300
032400              02 END-OF-JOB-IND              PIC X(3).
032500              88 END-OF-JOB                  VALUE IS 'EOJ'.
032600
032700              02 LINE-IS-EMPTY-IND           PIC X(3).
032800              88 LINE-IS-EMPTY               VALUE IS 'EOL'.
```

```
032900
033000        01 CONSTANTS.
033100*===============
033200
033300           02 PROMPT-4025             PIC X VALUE IS '?'.
033400
033500        01 VARIABLES.
033600*===============
033700
033800           02 TIMES-SCREEN-RESET    PIC 9(2).
034100        PROCEDURE DIVISION.
034200*
034400
034600     INITIALIZATION.
034800
034900     OPEN OUTPUT TEK-FORM.
035000        MOVE SPACES TO TEK-FORM-DATA.
035100        PERFORM TERMINAL-SET-UP.
035200        PERFORM DISPLAY-THE-FORM.
035400
035500
035700     MAIN-PART.
035900
036000        MOVE SPACES TO END-OF-JOB-IND.
036100        PERFORM  PROCESS-THE-TERMINAL-DATA  UNTIL  END-OF-JOB.
036200*    FLUSH THE LINE 23 FIELD STILL BUFFERED UP
036300        DISPLAY PROMPT-4025.
036310        ACCEPT TERMINAL-DATA.
036320      DISPLAY PROMPT-4025.
036330        ACCEPT TERMINAL-DATA.
036400        CLOSE TEK-FORM.
036500        PERFORM TERMINAL-SET-DOWN.
036600        STOP RUN.
036700*    << LOGICAL END OF PROGRAM >>
037000     PROCESS-THE-TERMINAL-DATA.
037200*===============
037200
037300          MOVE SPACES TO TERMINAL-DATA.
037400          PERFORM GET-BUFFER-LINE.
037500
037600          MOVE SPACES TO LINE-IS-EMPTY-IND
037700          PERFORM DISASSEMBLE-4025-INPUT UNTIL
```

```
037800                LINE-IS-EMPTY
037900              IF NOT END-OF-JOB
038000                 WRITE TEK-FORM-DATA
038100                 DISPLAY PROMPT-4025
038200                 MOVE SPACES TO TEK-FORM-DATA
038300                 DISPLAY PROMPT-4025
038400              ELSE
038500                 NEXT SENTENCE.
038600              DISPLAY '!ERA;!BEL'.
038700*
038800       TERMINAL-SET-UP.
039010*==============
039000
039120*      IT IS BEST TO SET THE 4025 TO A KNOWN STATE
039130*      RATHER THAN TO ASSUME WHAT STATE IT IS IN.
039100*      SET PROMPT := <?><CARRIAGE-RETURN><LINE-FEED>
039200*      SET F1 := <CARRIAGE-RETURN><LINE-FEED>
039300*      SET PT := <SEND MODIFIED>
039400*      SET FIELD-SEPERATOR := <CARRIAGE-RETURN>
039500
039600              DISPLAY '!PRO 63,13,10'.
039700              DISPLAY '!LEA F1 13,10'.
039800              DISPLAY '!LEA PT /!REP 1;!SEN MOD¼3'.
039900              DISPLAY '!FIE 13'.
040000
040200       TERMINAL-SET-DOWN.
040210*==============
040400
040500*      PURPOSE:
040600*
040700*      RETURN THE 4025 TO COMMUNICATION WITH THE HOST,
040800*      THE KEYBOARD TO THE MONITOR SPACE, NON-FORM FILL
040900*      OUT, UNBUFFERED, ALL FUNCTION AND OTHER KEYS
041000*      UN-LEARNED, AND DISPLAY IN THE MONITOR SPACE OF
041100*      THE 4025 A MESSAGE SAYING WHAT HAS BEEN DONE.
041210*      IN THIS WAY, IT IS POSSIBLE FOR THE USER TO
041220*      COMMUNICATE WITH THE COMPUTER WITHOUT LOSING THE LAST  SCREEN.
041200
041300              DISPLAY '!FOR N;!BUF N;!MON K H'.
041400              DISPLAY '!CLEAR'.
041500              DISPLAY '<< END OF TEK FORM FILLOUT EXAMPLE >>'.
041600
```

```
        GET-BUFFER-LINE.
*==============
*       SINCE INFORMATION IS 'BUFFERED' , IT IS NECESSARY TO
*       'PROMPT' THE 4025 TO SEND THE DATA AND THEN WAIT
*       FOR IT TO BE SENT.

            DISPLAY PROMPT-4025.
            ACCEPT TERMINAL-DATA.

        DISASSEMBLE-4025-INPUT.
*==============

*       THIS SECTION IS A 'CASE' STATEMENT THT EXAMINES
*       THE DATA FROM THE 4025 AND PERFORMS A ROUTINE
*       WHICH FURTHER EXAMINES THE LINE.
*
*       CASE TD-LOC-ROW OF
*           '003' : PERFORM ROW 3 PROCESSING
*           '005' : PERFORM ROW 5 PROCESSING
*           '023' : PERFORM ROW 23 PROCESSING
*           '   ' : SET LINE INDICATOR TO 'EMPTY'
*
*       WHEN THE PROTECTED MODIFIED FIELD ON LINE 23 IS
*       PROCESSED, EOL IS SET TRUE AND THE BUFFER CONSIDERED
*       PROCESSED. THE WHOLE PROCESS STARTS OVER AGAIN AND
*       A NEW BUFFER IS PROCESSED.
            IF TD-LOC-ROW = '003'
               PERFORM ROW-3

            ELSE IF TD-LOC-ROW = '005'
               PERFORM ROW-5

            ELSE IF TD-LOC-ROW = '023'
               PERFORM ROW-23.

*       NOTE:  IF ADDITIONAL LINES EXIST ON THE FORM
*                  ADDITIONAL 'ELSE IF' LINES MAY BE CODED.
*       HOWEVER, THERE IS A LIMIT TO THE DEPTH TO WHICH
*       MOST COMPILERS WILL ALLOW IF'S TO BE NESTED.
*       TO OVERCOME THE NESTING LIMIT, START UP ANOTHER
*       'IF...THEN...ELSE'.

            IF LINE-IS-EMPTY
```

```
050000              NEXT SENTENCE
050200              ELSE    PERFORM GET-BUFFER-LINE.
050300
050500*
050600*      WORK ROUTINES
050700*
050800*      THESE ROUTINES FURTHER BREAK DOWN THE DATA JUST
050900*      RECEIVED FROM THE 4025 TERMINAL. EACH OF THE
051000*      ROUTINES THAT FOLLOW IS DEDICATED TO ONE ROW FROM
051100*      THE 4025. THESE ROUTINES WILL EXAMINE THE COLUMN
051200*      NUMBER AND TAKE APPROPRIATE ACTION. IN THIS PROGRAM
051700*      THE DATA IS JUST SAVED IN A DISC FILE BUT ADDITIONAL
051800*      PROCESSING COULD HAVE BEEN DONE.
051900
052000*******
052100 ROW-3.
052200*******
052300
052400              IF TD-LOC-COL = '006'
052500                 MOVE TD-DATA TO TF-REF
052600
052700              ELSE IF TD-LOC-COL = '025'
052800                 MOVE TD-DATA TO TF-CUST-1
052900
053000              ELSE IF TD-LOC-COL = '030'
053100                 MOVE TD-DATA TO TF-CUST-2
053200
053300              ELSE IF TD-LOC-COL = '053'
053400                 MOVE TD-DATA TO TF-INSTRUMENT
053500
053600              ELSE IF TD-LOC-COL = '071'
053700                 MOVE TD-DATA TO TF-VALUE
053800
053900              ELSE
054000                 NEXT SENTENCE.
054100
054200              IF TF-REF = 'QUIT'
054300                 MOVE 'EOL' TO LINE-IS-EMPTY-IND
054400                 MOVE 'EOJ' TO END-OF-JOB-IND
054500              ELSE
054600                 NEXT SENTENCE.
054700
054800*******
```

```
054900 ROW-5.
055000*******
055100
055200              IF TD-LOC-COL = '011'
055300                 MOVE TD-DATA TO TF-TYPE
055400
055500                 ELSE IF TD-LOC-COL = '027'
055600                 MOVE TD-DATA TO TF-FC-AMOUNT
055700
055800              ELSE IF TD-LOC-COL = '048'
055900                 MOVE TD-DATA TO TF-RATE
056000
056100              ELSE IF TD-LOC-COL = '066'
056200                 MOVE TD-DATA TO TF-DOLLARS
056300
056400              ELSE
056500                 NEXT SENTENCE.
056600
056700*******
056800       ROW-7.
056900********
057100              IF TD-LOC-COL = '012'
057200                 MOVE TD-DATA TO TF-OUR-ACCT
057300
057400              ELSE IF TD-LOC-COL = '023'
057500                 MOVE TD-DATA TO TF-CONTRACT-NR
057600
057700              ELSE IF TD-LOC-COL = '051'
057800                 MOVE TD-DATA TO TF-CREDIT-REFERENCE
057900
058000              ELSE IF TD-LOC-COL = '076'
058100                 MOVE TD-DATA TO TF-CHECK-OK
058200
058300              ELSE
058400                 NEXT SENTENCE.
058500
068100
068200********
068300 ROW-23.
068400********
068500
068600              MOVE 'EOL' TO LINE-IS-EMPTY-IND.
068700
```

```
068800        DISPLAY-THE-FORM.
071300*===============
071400
071500                 DISPLAY TFFO-0.
071600                 DISPLAY TFFO-1.
071700                 DISPLAY TFFO-1A.
071800                 DISPLAY TFFO-1B.
071900                 DISPLAY TFFO-1C.
072000                 DISPLAY TFFO-3.
072100                 DISPLAY TFFO-3A.
072200                 DISPLAY TFFO-3B.
072300                 DISPLAY TFFO-3C.
072400                 DISPLAY TFFO-3D.
072500                 DISPLAY TFFO-3E.
072600                 DISPLAY TFFO-5.
072700                 DISPLAY TFFO-5A.
072800                 DISPLAY TFFO-5B.
072900                 DISPLAY TFFO-5C.
073100                 DISPLAY TFFO-7.
073200                 DISPLAY TFFO-7A.
073300                 DISPLAY TFFO-7B.
073400                 DISPLAY TFFO-7C.
075600                 DISPLAY TFFO-EOD-FLAG.
075700*
075800*
075900                 DISPLAY TFFO-END.
076000*
076100*        << PHYSICAL END OF PROGRAM >>
```