digital

VT61 User's Manual

VT61 User's Manual

EK-VT61-OP-001
Copyright © 1976 by Digital Equipment Corporation. All rights reserved.

This manual is divided into three parts: The first part is titled *For Owners*; the second is *For Operators*; the third is *For Programmers*.

When the terminal is delivered, the Owners' portion of the manual should be referred to first. It tells you how to verify the type of terminal you have received, how to install the terminal, and how to adjust the switch settings for your application. It also contains a description of maintenance considerations.

The people who will be using the VT61 to communicate with a computer can use the second section (*For Operators*) to familiarize themselves with the operation of the terminal, including the functions of all the keys on the keyboard, the meanings of the indicator lights, and the things they see on the video screen. It consists of many "experiments" which guide the operator through the various features of the terminal. The Owners' part of the manual contains important information on how the terminal should be set up to allow the Operators' section to be used to "get to know" the terminal.

The largest portion of the manual is the Programmers' part, which is further divided into five sections. The first section introduces the concepts that are necessary to understand the function of the terminal, such as the idea of a cursor position. In the next four sections, all of the features of the VT61 are defined, grouped by sections into four categories: Terminal Modes, Text Editing and Cursor Control, Input/Output, and Miscellaneous Functions. Four appendices are also provided which describe the keyboards and commands of the terminal. The purpose of this part of the manual is to specify in detail the function of the terminal. The information is provided in various "articles", each of which explains the terminal's response to one of the commands.

Although the titles of the parts of the manual state that they are "for" a certain group of people, the entire manual is suitable to gain a complete understanding of the terminal. The manual does not use any terms which it does not define. The Programmer's part of the manual will be heavy reading for persons who have only a casual acquaintance with the terminal, but it is structured so that they may read one article at a time to answer the questions they have.

8/77-14

TABLE OF CONTENTS

For Owners

This part of the *VT61 Users' Manual* contains information helpful in planning for, installing, and maintaining the VT61.

PREPARING THE ENVIRONMENT FOR THE VT61

The design of the VT61 will normally pose few constraints on the selection of a place to install the terminal. In most cases, any environment which is suitable to the operator of the terminal will be a satisfactory environment in which to operate the terminal. Extremes of temperature and humidity should be avoided. A summary of the guaranteed operating conditions of the VT61 is found at the end of the manual.

If the terminal is ordered with the electrolytic copier, the environmental requirements will be more stringent. The environmental specifications for the copier are also listed at the end of this manual.

Like many video terminals, and all televisions, the VT61's picture-tube electronics emit a high-pitched frequency. Many older persons cannot even hear this whine. Nevertheless, it may be advisable to provide background noise in the operating environment of the terminal if it is being operated by persons who could experience discomfort from hearing this tone for long periods of time.

INSTALLING THE VT61 AND INTERFACING TO A COMPUTER

Take the following steps to set up the VT61:

1. Unpack the unit and place it on the desk or surface where it will be operated.

2. Notice the blue and silver tag at the rear of the unit (see Figure 1). As well as containing the serial number of the unit, this tag indicates the exact model number. This number provides information concerning the interface supplied with the unit and the electrical requirements of the terminal. The model number should be one of the following:

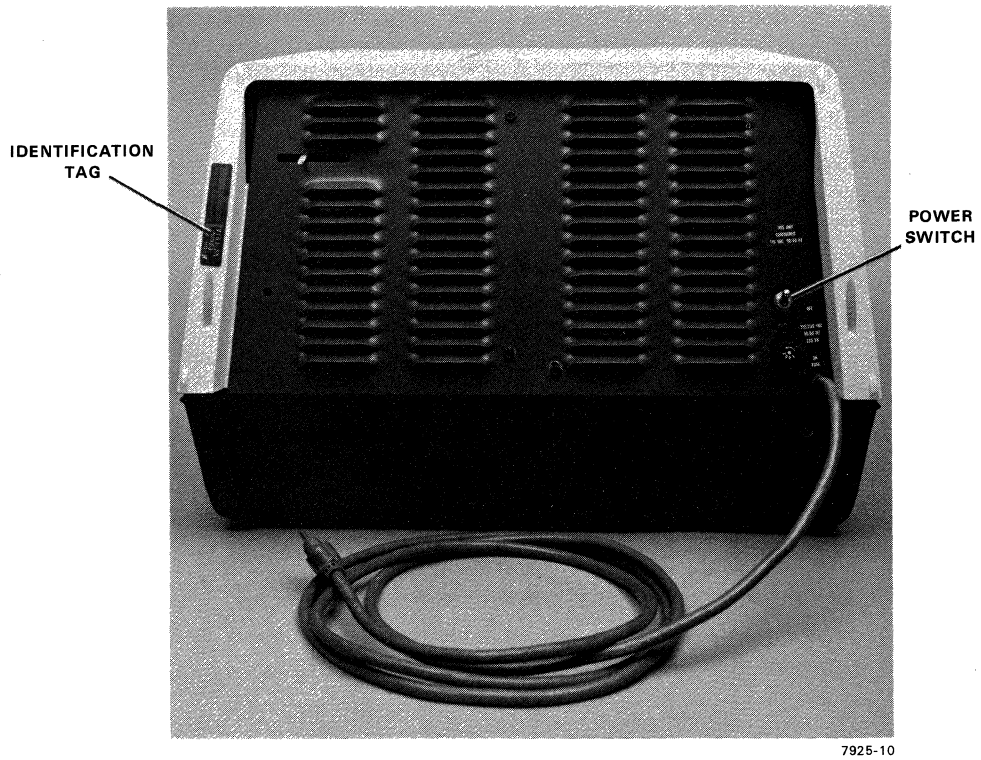


Figure 1

	115V, 60 Hz	230V, 50 Hz
20 mA loop – Mate-'n-Lok ® connector	VT61-AA	VT61-AB
EIA connector with all signals necessary for Half-Duplex operation	VT61-AC	VT61-AD
EIA connector with data lines only – for Full-Duplex operation	VT61-AE	VT61-AF
20 mA loop – 283B plug	VT61-AK	VT61-AL

The letter after the "1" will be a B instead of an A to indicate that the terminal includes an integral electrolytic copier. For example: VT61-BK instead of -AK indicates a VT61 with copier. ("K" means the interface is the 20 mA loop, the connector is the 283B plug, and the terminal requires 115V, 60 Hz.)

The adjustments that a user can perform to change the requirements of his terminal for operating voltage and line frequency are discussed shortly.

All VT61s come equipped with the parallel interface for the printer (LA180 or equivalent). However, the printer is supplied separately and not specified by the model number of the VT61.

3. Plug the three-pronged line cord of the unit into an electrical outlet that meets the requirements of the terminal.

   The power (On/Off) switch is at the rear of the unit (see Figure 1). Make sure it is in the lower, OFF, position.

4. Figure 1 also shows another cable coming from the bottom of the unit. This cable will have a Mate-'n-Lok ® connector, an EIA connector, or a telephone-type plug, depending on which option you ordered. Plug this connector into the input/output port of the computer or modem with which you wish to communicate. (**WARNING:** When performing the familiarization experiments in the **FOR OPERATORS** section, the interface cable should not be connected to a running computer system, since the familiarization routine and the actions of the computer system may interfere with each other.)

5. Making sure there is room on the platform on which the terminal is located, tip the unit back on its rear; there are two tilted positions in which it is stable. It is meant to be tilted back to gain access to the switches on the underside of the unit. As shown in Figure 2, there are two rotary switches (knobs) toward the front of the unit on the underside. One controls the transmission speed, and one controls the reception speed. If the transmission speed and reception speed are to be the same, find the setting for that speed on one of the two switches. Set that switch to that position. Set the other switch to the MATCH position. If the transmission speed and reception speed are going to be different, set the two switches separately for the transmission and the reception speed. Make sure that the transmitting speed of the VT61 matches the receiving speed of the computer, and the receiving speed of the VT61 matches the transmitting speed of the computer.

6. The leftmost slide switch on the underside of the unit is the OFF-LINE/ON-LINE switch (see figure 2). *To set the terminal up for the operator experiments described in this manual:* Slide this switch toward the front edge of the unit. This will cause the terminal to send information back to itself rather than down the input/output cable. It is also important that the slide switch labeled LOCAL LOOP not be in the ON position. *To communicate with a computer:* Slide this switch toward the rear of the unit. This will put the terminal On-Line to the computer.

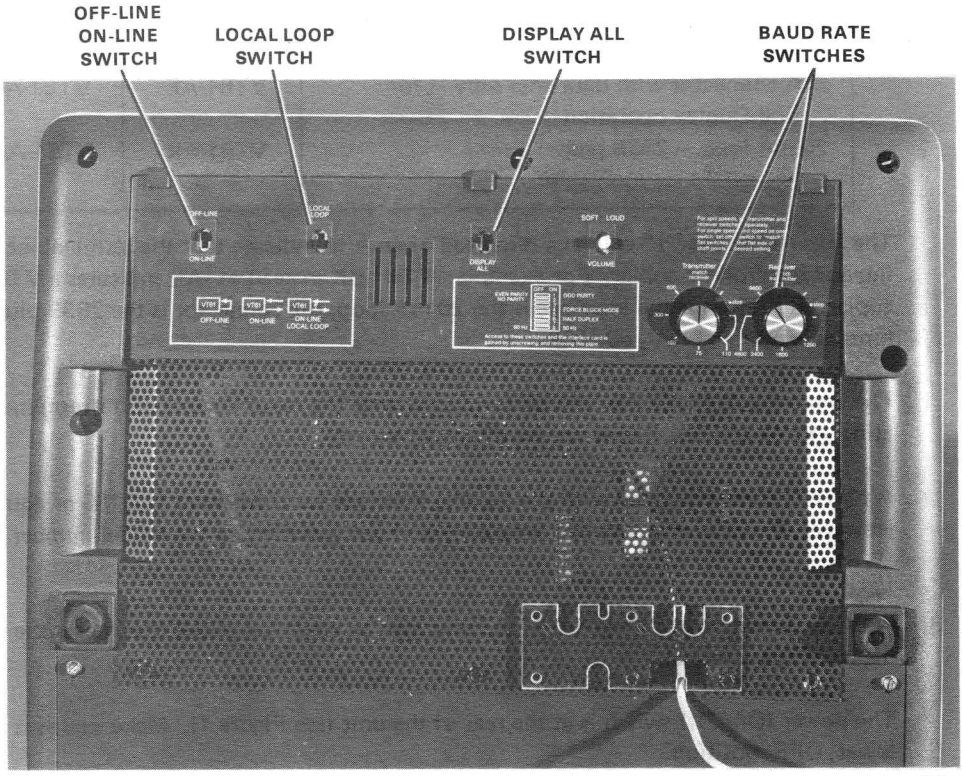


Figure 2

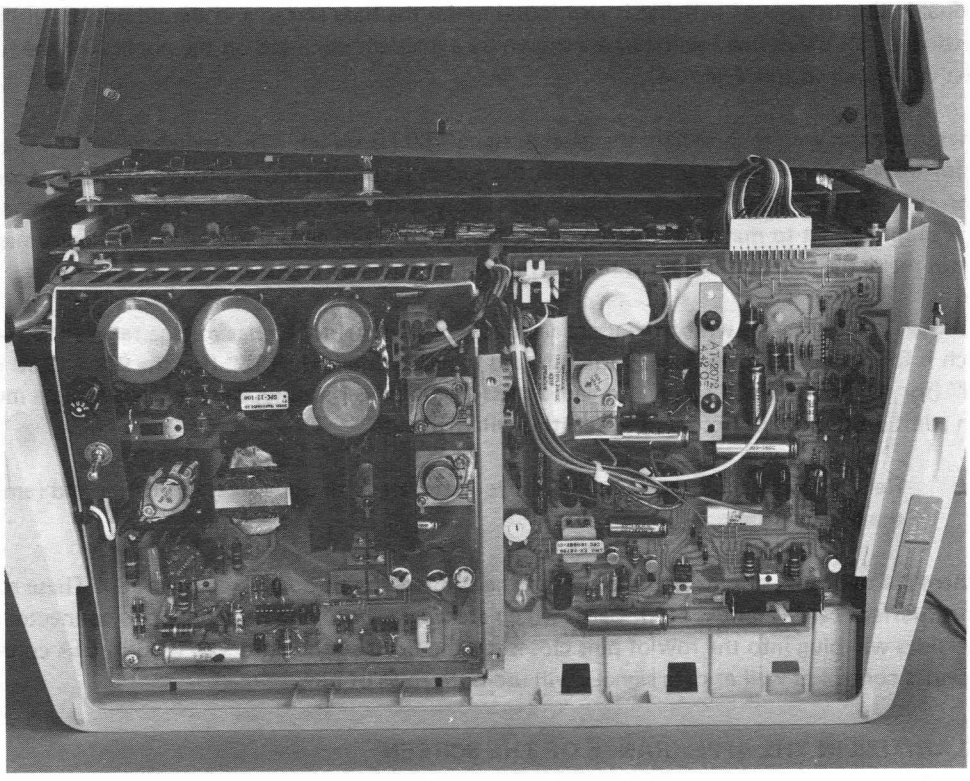
The forward portion of the terminal's base screen can be unscrewed and removed in order to gain access to part of the terminal's electronics. This allows the owner to:

- Remove the interface card and replace it with another communication option.
- Access the FORCE BLOCK MODE, HALF DUPLEX, PARITY ON/OFF, PARITY EVEN/ODD, and 50/60 HZ switches inside the unit.
- Plug in the Integral Modem Card (on terminals equipped with a slot for such cards).
- Plug in the cable to the printer.

The forward portion of the base screen is held down at the rear by three quarter-turn fasteners and at the front by a lip in the shell material. By unscrewing the quarter-turn fasteners, this section of the base can be lifted out. It is not necessary to remove the entire base to perform these installation functions.

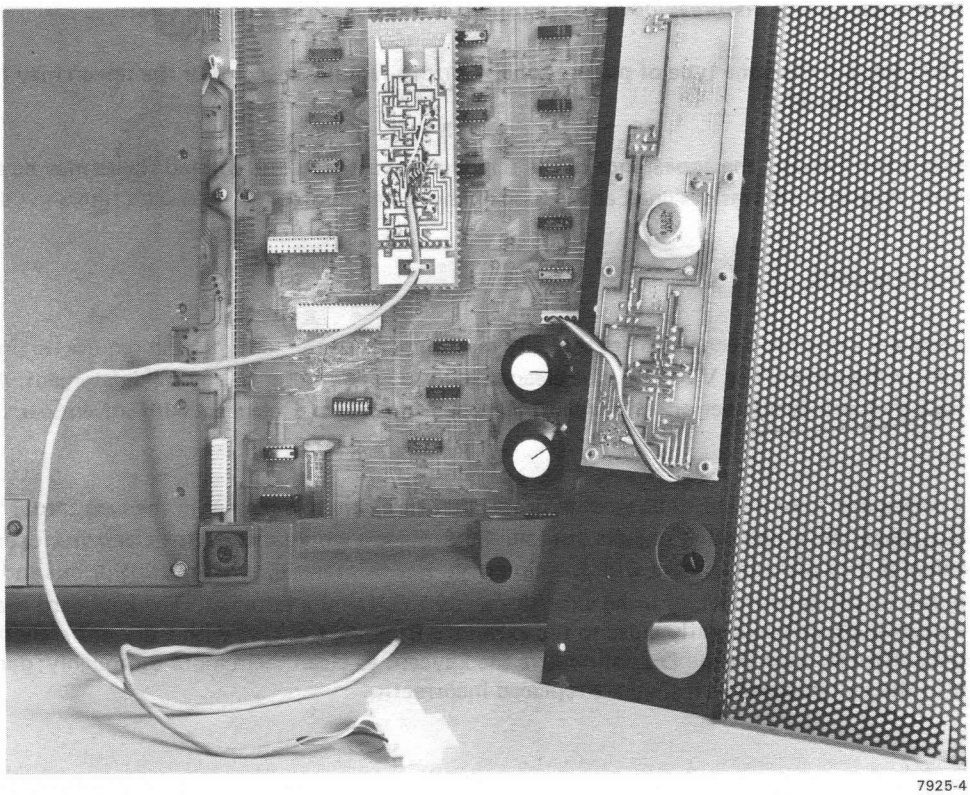
Use care when removing this section of the base, since the switches and speaker which are attached to the screen will still be connected to the internal electronics by a cable (see Figure 3). This cable should be carefully unplugged from the large internal printed-circuit board before proceeding so that the base screen can be removed completely.

A 40-pin male Berg connector will now be visible (see Figure 4). This is the connector for the General Output Register (Printer). The cable from the LA180 or equivalent printer should be attached to this plug so that the lettering on the cable faces upward and can be read.



7925-6

Figure 3



7925-4

Figure 4

A small, dual-in-line battery of switches is also found under the base screen and should now be visible. The proper settings of all the defined switches are shown by a diagram mounted on the portion of the base screen that was removed (see Figure 4).

One of these switches will be shown on the label as the 50/60 Hz switch. This switch controls the timing circuitry for the video to adjust it to the fact that the screen is scanned either 50 or 60 times a second, depending on the line frequency. Although the power supply will work with any frequency from 47 to 63 Hz, this switch should be set to match the line frequency within 1 Hz to ensure that external electromagnetic fields do not make the screen difficult to read. This is the frequency adjustment.

There is also an adjustment by which the user can adapt his terminal for higher voltage ranges (180 – 254V). This switch is found on the power supply module and is clearly marked "115" (for the 90 – 127V range) and "230" (for the 180 – 254V range). To gain access to the power supply, the rear louver plate must be moved. In addition to the steps covered so far, this will involve:

- Removing the four screws in the rear louver plate with a 1/4 inch nut driver and removing the louver plate.

Finally, there are two rows of pins in the center of the printed-circuit board (see Figure 4). These pins connect to the interface card (20 mA, EIA). Cards which are *single-width* and have only one connector on their undersides will plug into the row of pins closest to the front of the unit. The BN62B EIA connector, which provides control as well as data signals, will use both rows of pins.

IRREGULARITIES IN THE APPEARANCE OF THE SCREEN

At some time in its life, the picture tube may appear to darken at each character position. This is a normal phenomenon; it does not affect the life expectancy of the tube and will not interfere with the display of characters when the VT61 is in operation. It is caused by the fact that the terminal lights up the screen only at the character positions, *breaking in* the phosphor at those positions on the screen.

Not all VT61s contain the same type of picture tube. For this reason, the color of the screen may not be the same from unit to unit.

If, for these or other reasons, the appearance of the screen is not satisfactory, colored filters may be attached to the screen. A special nib was designed at the base of the screen in order to hold such filters in place.

USER MAINTENANCE

The keyboard and key-click mechanism are the only moving parts of the terminal and require no preventive maintenance by the owner. The VT61 may be cleaned with soap and water or any mild detergent. Although the terminal's shell provides superior resistance to damage from solvents, cleaners with solvents should not be used.

The VT61 packaging is not meant to be weatherproof; there are several openings in the case through which liquids, coins, paper clips, and other objects can fall. Such objects would disturb the electronic operation of the terminal if they came in contact with the circuitry. For this reason, avoid putting drinks and metal objects on the top of the terminal, or using excessive water to clean the terminal. The keyboard is an area where the electronics are particularly close to the exterior. Rubbing the keys with a dry or barely moist cloth should suffice to clean them. Do not remove the keycaps to clean them more thoroughly; damage may result to the switch contacts if they are replaced incorrectly.

DEC MAINTENANCE ARRANGEMENTS

There are four ways that the owner can arrange for service on a VT61:

1. *Depot Maintenance.* Service will be done at the DEC depot. You buy a mailer, pack the VT61 in it, and return it to the Depot. Upon completion of service, Digital returns the unit.

2. *On-Site Contract.* This is the traditional priority service contract, available with various coverage hours.

3. *On-Site, Per-Call Service.* In this arrangement, Digital charges for time and materials actually used.

4. *On-Site User Maintenance.* Module-Swapping Kits are available. Failed modules are returned to the DEC Depot for repair in the mailers in which the swapping kits are mailed.

In some geographical areas, additional options are available; contact your local Digital Field Service office for information.

For Operators

This part of the *VT61 Users' Manual* is a collection of experiments designed to familiarize you with the VT61. In addition, by performing these experiments, you can make sure the terminal works correctly.

As described in the **FOR OWNERS** part of the manual, the VT61 should be switched Off-Line for this set of experiments. It should not be plugged into a running computer system, since the operator will be experimenting with block transmissions which could interfere with the operation of the computer system. The LOCAL LOOP switch must be turned OFF so that block transmissions will not be looped back to the terminal, which would lock its keyboard.

These experiments assume that the FORCE BLOCK MODE switch inside the unit *IS NOT ON.*

The VT61 is a machine that lets you talk to a computer. It looks like a combination of a typewriter and a television. You talk to the computer by typing on the keyboard, and the computer talks to you by making messages visible on the screen. When you type things to the computer, the computer will usually put what you typed up on the screen so that you can see what you are typing.

In these experiments, your terminal has been switched Off-Line. Instead of the keyboard sending to the computer and the computer sending to the screen, the keyboard will send directly to the screen. You will be using the VT61 in the future to talk to a computer — but you are not talking to it now. Therefore, these experiments will teach you how to use the terminal, but they will not teach you what kinds of things you should say to the computer when the terminal is connected to the computer. That you must discuss with the people who program the computer to act the way it does.

If this is your first experience with a video terminal like the VT61, don't be alarmed at the strange keys on the keyboard. There is no way that you can damage the terminal by simply typing those keys. Some of the keys may produce confusing results when you type them, but this book is here to make them unconfusing.

We've tried to make the terminal's keyboard look as much like a typewriter keyboard as possible. However, many of the keys have special labels on them. You will learn about these labels later on in this discussion. For now, though, just disregard them.

Experiment 1. Locate the On/Off switch. It is a metal toggle switch at the back of the unit. Switch it up, to the ON position. (If the terminal is already on, switch it OFF and back ON to *reset* it.)

Experiment 2. There are eight red lights of the right of the small keyboard. They light up to indicate that the terminal has been set to act in a special way, or that something is happening at the terminal. The one closest to the front of the terminal should be lit up. It is labeled CLEAR TO SEND. This red light does not mean anything is wrong with the terminal!

Experiment 3. Give the TV tube about 1/2 minute to warm up. Then look at the upper left corner of the screen. You should see a flashing rectangle. It is called the *cursor:* If you cannot see the cursor, or if it is brighter than you want it, there is a lever at the back of the terminal which will control the general brightness of everything on the screen. If you'd like to change the intensity of the cursor, get up and move the lever until the cursor is at a comfortable intensity. (If you don't see the cursor no matter how you move the intensity lever, then something is wrong. The terminal may not be plugged in, or it may not really be switched ON, or the fuse may have blown.)

Experiment 4. Type a letter on the keyboard, and look at the screen where you saw the cursor. You should see that letter. The terminal displayed it at the position where the cursor was, and then it moved the cursor to the right, so that the next character you type will go there. Type another letter and you will see that it appears on the screen directly to the right of the first letter, and that the cursor moves over again. This is what the cursor does — it tells you exactly where the next letter you type will go.

Notice that each time you typed a key, the terminal made a clicking sound. This is its way of telling you that it got the message. You can adjust the volume of this clicking sound by turning a knob which is located on the underside of the terminal.

Experiment 5. Continue typing letters and watch them appear on the screen. When you have typed 80 letters, you should reach the end of the line. Look at the cursor — it is under the letter you just typed because it could not move any further to the right. Type another letter. This letter will replace the letter that was at the cursor position before. If you type another letter, that letter will replace what is there now. Type several letters, with the cursor still at the end of the line, and see that each letter replaces what was there before.

This is an important part of how the terminal works. Only one letter can occupy each place on the screen at any time. At the upper right of the large keyboard, there is a blue key labeled BACK SPACE. If you type it, the cursor will move backward, or to the left. It will move left one column for each time you type the key. Using the BACK SPACE key, move the cursor back to the middle of the line. Now, type some more letters, and see that the letters you are typing are replacing what used to be on the screen.

Closest to you on the large keyboard is the blue Space bar. This works exactly like the Space bar on a typewriter. Normally, it will move the cursor one column to the right. But it puts a Space on the screen, just as the letter keys put letters on the screen. Use the Space bar to replace some of the line you typed in with spaces. This is one way to erase information on the screen.

Experiment 6. Near your right hand is a large blue key labeled RETURN. If you type this key, the cursor will move to the left margin of the line it is on. Try out the RETURN key, and verify that the cursor moves to the beginning of the line.

Above the RETURN key is the LINE FEED key. Type it to move the cursor down to the next line. All the LINE FEED key does is to move the cursor down one line. To get to the start of the next line, you have to type RETURN and LINE FEED. (When you are talking to a computer, it will probably move the cursor to the start of the next line whenever you type the RETURN key, so you won't have to type LINE FEED.)

Type another line of letters and watch it appear under the first line.

Experiment 7. The blue TAB key, which is at the left edge of the large keyboard, moves the cursor to the right, but does not erase information on the screen as the Space bar does. Every eight columns on the screen, there is a *TAB stop.* The TAB key will move the cursor to the right until it reaches the next TAB stop.

Use the RETURN and LINE FEED keys to position the cursor at the beginning of the third line. Type the TAB key, and verify that the cursor moves eight columns to the right. Type an "X" to mark the spot where the cursor stopped. Continue to type the TAB key, and then an "X", to see where all the TAB stops are. Notice that, when the cursor moves past a certain point (column 72), the TAB key will move the cursor only one column to the right. If the cursor is at the end of a line, the TAB key won't move the cursor at all.

With the RETURN and LINE FEED keys, go to the start of the fourth line. Type two or three letters. Then type the TAB key. See that the cursor travels to the TAB stop. Type letters, the BACK SPACE key, and the TAB key to learn that the TAB key does not always move the cursor eight columns to the right, but only that number of columns necessary to position it at the next TAB stop.

On the right side of the keyboard is the blue DELETE key. The terminal ignores the signal that key makes; type it and nothing will happen. However, when the computer gets that signal, it may take some special action.

Experiment 8. You can use either of the SHIFT keys to make the terminal display the capital of any letter you type; otherwise, it will display the lower-case letter. Above the left SHIFT key is a blue key labeled CAPS LOCK. This key is like the SHIFT LOCK key of a typewriter. When this key is down, the terminal will always convert your letters to capitals. Press the CAPS LOCK key and note that it clicks as it locks into place. Press it again to release it and let the terminal display lower-case letters.

Try typing letters using the SHIFT keys and the CAPS LOCK key until you are comfortable with what they do.

Experiment 9. Type all the numeral and symbol keys on the large keyboard, except the ones with special words labeled on them, to see the variety of symbols you can display on the screen. Use the SHIFT keys to display the symbol which is on the top half of the key. Note that the CAPS LOCK key will not do the same thing that the SHIFT keys do on any key which is not a letter. That is, to display a "G" rather than a "g" on the screen, you can press either the SHIFT or the CAPS LOCK keys. But to get a "@" instead of a "2" on the screen, you must type one of the SHIFT keys; the CAPS LOCK key will not do this. Depress the CAPS LOCK key and type "ONE-TWO-THREE." Notice that you don't have to unlock the CAPS LOCK key to type the dashes.

Experiment 10. Type several short lines, each followed by RETURN and LINE FEED. What happens when you have finished the bottom line on the screen? You will type LINE FEED to move the cursor to the next line, but the cursor will already be on the bottom line. Since the terminal must give you a new line, it moves all the information on the screen up one line to make room. We say that the terminal *scrolls* the display. Notice that what was on the top line has gone off the top of the screen. You cannot get that information back. Type several more lines now that the cursor is down at the bottom line. As you enter lines into the screen at the bottom, watch the lines at the top leave the screen.

Experiment 11. The red BREAK key, like the DELETE key, sends a signal that the computer may interpret in a special way. But the terminal will not do anything with the information on the screen when you type the BREAK key now.

The red CTRL (Control) key, on the left side of the keyboard, can be used with the alphabetic keys to produce signals which are neither upper-case letters nor lower-case letters. You have already seen the result of some of these signals: CTRL M is the same as the RETURN key; CTRL J is the same as LINE FEED; CTRL H is the same as BACK SPACE. The RETURN, LINE FEED, and BACK SPACE keys are provided for your convenience. Type CTRL M by holding down the CTRL key while you type the M key. The CTRL key is a little like the SHIFT key – in order to do anything, it has to be held down while you type another key.

CTRL G will make the terminal's bell ring. Hold the CTRL key down and type G.

Experiment 12. The red ESC (SEL) key is also used to give the terminal special commands. Unlike the CTRL key, it does not have to be held down while another key is typed. Simply type it, and then type a letter. First of all, try "ESC H." Type the ESC key, and then type capital H. This is a special command to the terminal to move the cursor back up to where it began – in the upper left corner of the screen. This is called the *Home* position. Has the cursor moved there?

If you now type ESC K, thetop line will be erased. If you type ESC J, the entire screen will be erased.

Experiment 13. So far, the experiments have been concerned with the keys on the large keyboard. Now, let's look at the small (auxiliary) keyboard. It has numeral keys 0 through 9 on it. Type them, and see that they do the same thing as the numeral keys on the main keyboard. Try out the decimal point (.) key. The ENTER key on the small keypad does the same thing as the RETURN key on the main keyboard.

Try out the keys which are marked with arrows. Notice that these keys move the cursor one space in the direction shown by the arrow. Typing the "down-arrow" key will move the cursor down, and typing the "right-arrow" key will move the cursor to the right.

Type one of the arrow keys, and hold it down. After a short delay, the key will repeat, moving the cursor in the chosen direction not once but many times. On the main keyboard, the Space bar, the DELETE key, and the BACK SPACE key are also *auto-repeating.* The delay before the keys start repeating is to make sure you actually meant to hold the key down.

Most of the keys on the keyboard are not auto-repeating. For instance, the E key is not auto-repeating. If you want to place a lot of E's on the screen, however, you don't have to type the E key repeatedly. In the lower right corner of the main keyboard, there is a key marked REPEAT. If you hold down this key, then any key you type will do what it does many times, not just once. The E key will send E's to the screen repeatedly, until you release either the E key or the REPEAT key. Try it out.

Experiment 14. Next to the REPEAT key is a key marked COPY. If your terminal has a copier in the side, this key will copy the information on the screen out to the copier. If your terminal doesn't have a copier, but it does have a printer, the screenful will go there. If your terminal doesn't have an output device of any kind, this key will do nothing. (Please check to see that the copier or printer has paper in it and is ready to do printing before you type this key.)

If you type the COPY key with the SHIFT key down, the top light will go on. If you type it again, the light will go off. The light is labeled AUTO-PRINT. Whenever this light is on, whenever the cursor moves down the screen, the line it moved from will be sent to the copier or printer automatically. If the computer is sending you a lot of information, and you want to get it on paper, type SHIFT COPY to turn the red light on; then every line of text the terminal gets will also go to the copier or printer. When you do this, another red light may come on: The RECEIVER OFF light may be lit as the terminal tells the computer to slow down to wait for the copier or printer to finish. When the copier or printer is ready for more information, the RECEIVER OFF light will go off, and the terminal will receive more information to send to the copier or printer.

BLOCK MODE EXPERIMENTS

The first part of the experiments introduced you to the VT61 keyboard. The second part will show you how to use the advanced features of the VT61 to edit text on the screen. To set the VT61 up for the second group of experiments, give it a special command by typing the following keys, in order: ESC, capital O, capital B. The O and B are not seen on the screen; instead of being put on the screen, the terminal considered them to be a command to it since the ESC key was typed first. However, another of the red lights, labeled BLOCK, should have lit up. (If it hasn't lit up, type those three keys again — you may have typed a lower-case instead of one of the capitals.) Normally, you won't have to type these commands using ESC key; the computer will do it for you.

The red light indicates that the terminal is in Block Mode. In Block Mode, many of the rules change to make it easier to edit text. In the first place, if you were switched On-Line so that you were in contact with a computer, the terminal would no longer tell the computer what keys you had typed. Instead, the terminal would just put the information you typed onto the screen (much the same as it is doing right now because you are not in touch with a computer). If your terminal is in Block Mode and you make a typing mistake, you can use the terminal's commands to back up, fix the mistake, and continue. Normally, every key you type goes to the computer, and the computer must show you what you typed and help correct your errors. In Block Mode, the terminal does not send anything to the computer until you tell it to by giving it a TRANSMIT command.

Make one final adjustment in the settings of the terminal before going on to the Block Mode experiments: On the underside of the terminal, about directly below the A key, there is a slide switch. It can be slid toward you or away from you. When it is slid toward you, the terminal is Off-Line. It will not talk to a computer even if it is wired up to one. It will talk to itself, taking information from the keyboard and putting it on the screen. When the switch is pushed away from you, the terminal is On-Line, and you may be able to communicate with a computer. So far, the terminal has been switched Off-Line, and the text you have typed in has gone directly to the screen. Putting the terminal in Block Mode did the same thing; you will still be able to see what you type on the screen. Therefore, push the slide switch away from you to place the terminal On-Line. You won't be talking to a computer since there is no computer serving the terminal at this time, but in this position there is no chance that the terminal will see its own transmissions, think it is hearing a computer, and lock its keyboard. In short, the experiments will run more smoothly.

Don't worry if pushing the slide switch away from you caused strange characters to enter the screen; this is normal.

Experiment 15. The discussion of the small keyboard did not mention the blue FCN (Function) key. It is the key you will use to make the terminal perform special functions on the information on the screen.

The Home function moves the cursor to the Home position – the top left corner of the screen. This is the position the cursor was in when you switched the terminal on. You made the terminal perform the Home function before using the ESC key. Here is how to make the terminal do it using the FCN key: Look at the Q key on the main keyboard. It has blue lettering on it saying Home. This tells you that the Q key is the key you use for the Home function. If you simply type the Q key, the terminal will place a Q on the screen. But if you type the FCN key first, then Q, the terminal will perform the Home function. Press the FCN key, then the Q key. You have just invoked the Home function. Did the cursor move to the upper left corner?

Many alphabetic keys have blue lettering on them to indicate the function they invoke when they are typed directly following the blue FCN key.

Experiment 16. Type the FCN key, then X. The X key has a blue label, erEOL, which stands for the Erase to End-Of-Line function. When you typed FCN X, the contents on the top line of the screen were erased. The top line was chosen because that was the line the cursor was on. The blue label, erEOS, on the Z key stands for the Erase to End-Of-Screen function. Invoke this function by typing FCN Z. Now the screen is blank.

Experiment 17. On the underside of the terminal, directly below the right SHIFT key, there is a slide switch. switch. This is the DISPLAY ALL switch. Slide it back and forth. Notice that when the switch is on, you see a tiny dot in each character position. This tells you there's *nothing* at that character position. Even a space – which you put in the screen by using the Space bar – is considered to be something. Type the Space bar a few times. See that, with the DISPLAY ALL switch set this way, you can tell the difference between places where there is a space and places where there is *nothing.* You may use this switch at any time to find out exactly what is in the screen – there are several characters you will learn about which you won't be able to see unless the DISPLAY ALL switch is turned on. For now, leave the switch in the position in which the entire screen looks blank.

Experiment 18. Move the cursor back Home by typing FCN Q again. Erase the screen again by typing FCN Z. Now you are ready to use the VT61 to edit text on the screen. First, you'll have to put some text on the screen. Type in a few sentences – but when you reach the end of the first line, JUST KEEP ON TYPING. You will see that the cursor automatically *wraps around* from the end of the first line to the beginning of the second line.

A problem you may have now, though, is that part of a word is at the end of the first line, and the other part is at the beginning of the second line – in other words, the word was split in two pieces. Or the word might have ended right on the last position in the first line, and the space between it and the next word was placed at the beginning of the second line, making the left margin ragged. If you didn't have this problem, type more text on the second line, and, when you get to the end of that line, type a very long word to make sure it gets split in two pieces.

Now, the terminal knows that the character at the end of a line is supposed to be next to the character at the beginning of the next line. The only problem with the word which was split into two pieces is that it doesn't look good. So assume you've finished entering the text into the screen, and now you want it to look good. You want to JUSTIFY the text – keep words from being split in two. The VT61 can do this. It has a function called *Text Justify.* Notice the blue label, TxJus, on the V key, and type FCN V to invoke the Text Justify function. The word which was split in two has been moved to the next line,

and everything on that line has been moved forward to make room.

The Text Justify function places the cursor at the end of the text, so you can continue typing. Type some more text in without paying any attention to where on a line you are. Then re-justify the screen by typing FCN V again.

Experiment 19. When you type the FCN key to invoke a function, the letter key you type to specify which function you want can be either a capital letter or a lower-case letter. All the functions you have used so far work whether you use a capital letter or a lower-case letter. That is, the Text Justify function was invoked whether you typed FCN V or FCN v. This choice is permitted on all FCN functions that have a one-time effect. There are several FCN commands you can give the terminal that will turn on a special feature. Rather than having a one-time effect, these features will remain in effect until you turn them off. For these functions, typing FCN and the *capital letter* turns the feature ON, and typing FCN and the *lower-case* letter turns the function OFF. When a special feature is turned on, we say that the VT61 is *in a special mode.* The first mode we will use is Insert Mode.

Remember that back at the beginning of these experiments you saw how to replace information on the screen by backing up and typing over it. The new information you typed in *replaced* the old information that was at that position.

You can set the VT61 so that, instead of destroying the old information on the screen by replacing it, typing in new information *moves over* the old information to make room for the new information. To set the VT61 to do this, you place it in Insert Mode. The abbreviation "Insrt" on the I key stands for Insert Mode. Typing FCN I (with a capital I) places the terminal in Insert Mode (*enters* Insert Mode). Typing FCN i (with a lower-case i) takes the terminal out of Insert Mode (*exits* Insert Mode). Before you see how Insert Mode works, type FCN I to enter Insert Mode, and type FCN i to exit Insert Mode. Notice that while the terminal was in Insert Mode (between the time you typed FCN I and FCNi), one of the red lights came on — the one labeled INSERT. Typing FCN I turned it on; typing FCN i turned it off.

Using the arrow keys on the small keypad, move the cursor up to the top line. Type in some letters, and verify that they *replace* what was previously in their place, just as they did in the beginning of the experiments. Now type FCN I to enter Insert Mode, and try the same thing. Now, when you type new text in, the old text *moves over* to the right and is not destroyed by what you are typing in. The new text is *inserted* in the middle of the old text.

You can work with the terminal in Insert Mode, or without this feature, whichever you like best, and you can go back and forth (tell the terminal to enter and exit Insert Mode) as often as you like. If you simply want to replace a word on the screen with another word of equal length, leave the terminal out of Insert Mode, move the cursor to the beginning of the word you want to change, and just type over it.

If you want to insert a new word in the screen between two words which are already there, place the terminal in Insert Mode, move the cursor to the beginning of the second of the two words, and type in the word that goes between that word and the one before it. Type a space after typing the new word in to separate it from the word after it.

To change "The bears" to "The goats" . . .

The bears	1. Move the cursor over the "b" in "bears".
The gears	
The goars	2. Take the terminal *out of Insert Mode (FCN i.)*
The goars	
The goats	3. Type in "goat"

To change "The bears" to "The polar bears" . . .

The bears	1. Move the cursor over the "b" in "bears".
The pbears	
The pobears	2. Put the terminal *into Insert Mode (FCN I).*
The polbears	3. Type in "polar."
The polabears	
The polarbears	
The polar bears	4. Type a Space to separate the two words.

Remember that Insert Mode stays in effect (that is, the red light stays on) until you give the terminal a command to return to normal.

Experiment 20. Another function you can invoke is the Delete Character function. It is used to remove a word when you don't want to put anything in its place. The label, DelCh, on the S key stands for Delete Character. Delete Character is not a mode. Therefore, it can be invoked by either FCN S or FCN s. Move the cursor to the first letter in a word, and type FCN S enough times to delete all the characters in that word and to delete the space between that word and the next word.

To change "The polar bears" to "The bears" . . .

The polar bears	1. Place the cursor over the "p" in "polar."
The olar bears	
The lar bears	2. Type FCN S once for each letter in "polar."
The ar bears	
The r bears	
The bears	3. Type FCN S again to get rid of the unnecessary space.
The bears	

Experiment 21. Move the cursor to somewhere on the second line of text. See what happens when you type FCN F (Insert Line) and FCN D (Delete Line). These commands are useful when you want to add or delete a lot of text at once and want to make room for it in the screen. Try FCN W (Cursor to End-Of-Text). This function positions the cursor past everything that has been typed in, exactly as the Text Justify function does, so you can continue typing.

Experiment 22. Use the arrow keys to position the cursor over the first character in a word. Then type FCN E once for each letter in that word. "CEmph" means Change Emphasis. Notice that the word you positioned the cursor over is now displayed in dark-on-light instead of light-on-dark. This different style is called *reverse-video.* If the letter under the cursor had been in reverse-video when you typed FCN E, it would have been changed back to normal.

Experiment 23. Try out the numeral keys and decimal-point (.) key on the small keyboard. They will do the same thing they did when the terminal was not in Block Mode. These keys have labels on the front. These labels are white, not blue. They do not go with the FCN key. That is, even though there is a label saying HOME on the 9 key, typing FCN 9 will NOT move the cursor Home. Here is how the labels on the small keyboard work: The terminal can be set so that typing the numeral keys causes numerals to appear on the screen, or it can be set so that typing them causes the function whose symbol appears on the front of the key to be invoked.

Experiment 24. Place the terminal in Editing-Keypad Mode by typing FCN K (with a capital K). Notice that a red light came on to tell you that the terminal is in Editing-Keypad Mode. Now type the 9 key on the small keyboard — the one with the white label, HOME. Instead of putting a "9" in the screen, this key now moves the cursor Home. Type the decimal-point (.) key, labeled CEmph. Instead of putting the character "." in the screen, this key now invokes the Change Emphasis function.

You now have these two functions, plus the Text Justify function and the Delete Character function, available on the small keyboard. You no longer have to type two keys (the FCN key and another key) to invoke these functions. The 4, 5, 7, and 8 keys on the small keyboard have labels saying F1, F2, F3, and F4. These keys will send messages to the computer. The meaning of these four signals will depend on how your computer has been programmed. You can also use the FCN key with the ten numeral keys on the main keyboard to send the messages F1 through F10 to the computer. The Transmit Data function will transmit to the computer all the information from the cursor to the end of the screen. The Transmit Message command will transmit the information from the Home position up to the cursor. It will also put a marker in the screen. If you have invoked Transmit Message once before, and there is already a marker in the screen above the cursor position, then instead of transmitting the information between the Home position and the cursor position, it will transmit only the information past the first marker – it will transmit everything up to the cursor position that wasn't transmitted the first time you invoked the Transmit Message function.

The cursor will scan the text on the screen which it is transmitting. When the terminal has finished transmitting, the cursor will return to where it was when you invoked the Transmit Message command.

Experiment 25. One of the keys has a label reading I/R. This label stands for Insert/Replace. You can now use this key to switch the terminal into and out of Insert Mode. Type this key – the red light labeled INSERT should come on. If you type the key again, the red light will go off. Typing FCN I and FCN i will also put the terminal in Insert Mode and take it out of Insert Mode, as usual.

There are also two keys on the small keyboard with labels XmMsg and XmDat. These labels stand for Transmit Message and Transmit Data. When you have typed some text into the screen and used the terminal commands to correct any errors, you will use one of these commands to send the information to the computer. Since you are not in contact with a computer now, don't use these keys yet.

Experiment 26. The remaining label on the small keyboard is Cmd. Type this key. A letter C inside a circle has been placed in the screen at the position the cursor was at. When you are talking to a computer, you can send it commands while the terminal is still in Block Mode by:

1. Typing Cmd – either on the small keypad when the terminal is in Editing-Keypad Mode, or by typing FCN R.

2. Typing in a command which is meaningful to the computer you're using.

3. Invoking the Transmit Message command.

The terminal will send to the computer as a *message* everything from the C inside the circle up to the cursor position. Once again, since you are not now in contact with a computer, do not transmit anything yet.

Experiment 27. So far, you have not had to use the RETURN key to enter text into the screen; the terminal has decided where each line would end when you invoked the Text Justify function. In Block Mode, the RETURN key has a special function. It is the way you tell the terminal that all following text *must* go on the next line. You would do this to start a new paragraph.

Find a place in your text where one sentence ends and another begins. Position the cursor over the Space after the period, and use the Delete Character function (FCN S) to delete all the spaces which you typed in between the two sentences. Now the period which ended the first sentence should be next to the first letter of the next sentence, and the cursor should be over that first letter. Type the RETURN key, and watch the screen. Not only has the cursor moved to the beginning of the next line, but so has the second sentence! All the text in the screen ahead of where the cursor was has moved forward to ensure that the sentence you selected can appear at the start of a new line.

Move the DISPLAY ALL switch on the ON position again. Look at the end of the first sentence — there is a paragraph sign (¶) following the sentence. You put it there by typing the RETURN key, and it is that symbol in the screen that forced all the characters out to the next line.

Pushing all those characters forward may have spoiled the effect of the Text Justify function — there may be a word lower on the screen, half of which was pushed onto another line. Invoke the Text Justify function again. Note that the paragraph you created was preserved. The Text Justify function compressed the text on the screen, then reexpanded it so it would look nice. When it came to the paragraph sign that you placed in the screen, it again moved the text following it to a new line.

Experiment 28. Move the cursor up to the character position directly following the paragraph sign, and type the RETURN key again. Re-justify the screen. By typing two paragraph signs together, you can force following text to be placed two lines down — that is, you can double-space between paragraphs.

If you move the cursor to the middle of one of the blank lines, to the right of a paragraph sign, and you invoke the Delete Character function, you will pull the text on the next line up to the same line as the paragraph sign. If you invoke the Text Justify function now, it will push that text back down to the next line.

When you put a paragraph sign in the screen, the terminal acts as if it were in Insert Mode. Placing a paragraph sign in the screen never replaces a character that was always in the screen; it just moves characters over.

Experiment 29. Place the terminal in Insert Mode (FCN I) and move the cursor to the beginning of one of the paragraphs (the sentence that was shifted to a new line by the paragraph sign). When the cursor is over the first letter in the sentence, type the TAB key. Just as typing RETURN shifted all following text to the next line, typing TAB will shift all following characters to the next TAB stop. If the DISPLAY ALL switch is still on, you can see the TAB that was stored in the screen — it is an arrow pointing to the right. (This symbol has nothing to do with the arrow key on the small keyboard.)

If the terminal had not been in Insert Mode, the TAB character would have replaced the character that was formerly at the cursor position. All the remaining characters in the sentence would have been moved over to the TAB stop, but one character would be gone.

When the terminal is in Block Mode, therefore, you must not use the TAB and RETURN keys simply to position the cursor — they position text as well as the cursor. To move just the cursor, use the arrow keys on the small keyboard.

Experiment 30. When the terminal is in Block Mode, there are several things you cannot do from the keyboard. You cannot, for example, use the COPY key with the SHIFT key to turn on the AUTO-PRINT light; nor can you get the terminal out of Block Mode from the keyboard. When you are finished experimenting with text-editing in Block Mode, you will have to turn the terminal off and on again to restore it to its initial state. The screen will be erased, the cursor will be back Home, and the terminal will NOT be in Editing-Keypad Mode or Insert Mode.

The terminal will also NOT be in Block Mode. For this reason, you will have to slide the switch under the left SHIFT key toward you again to place the terminal Off-Line.

FORMS MODE EXPERIMENTS

Experiment 31. In the Block Mode Experiments, you saw how you could make text on the screen show up in either normal video or reverse-video. In Forms Mode, the terminal treats everything that is on the screen in reverse-video as part of a form. The terminal will let you fill out the form, but it won't let you write over the form, or move it or erase it.

To see how Forms Mode works, you will first have to place a form on the screen. Normally, the computer will place on the screen the form it wants you to fill out, but for these experiments you will have to create a form to experiment with.

Type the ESC key, then capital O, then capital J. You have placed the terminal in Reverse-Video Mode. In this mode everything you type in will go in reverse-video – you won't have to use the Change Emphasis function to get it that way. (Normally, you won't be using this mode. In fact, when the terminal is in Block Mode, it won't even let you use it.)

The cursor should be at the Home position. Type in "Name". Using the arrow key, move the cursor over to the middle of the same line and type "Phone". Move the cursor to the start of the next line and type "Address". Move the cursor to the start of the line after that and type "City". In the middle of that line, type "State". This will be the form.

Now place the terminal in Forms Mode by typing the ESC key, then capital O, then capital F. Notice that the red light labeled FORMS has lit up. Notice also that the cursor has been positioned to the right of the word "Name". This position is the start of the form.

Experiment 32. Type your name in the space after the word "Name". Notice that your name went in the screen in normal video. The terminal will now let you use normal video only. The Change Emphasis function will not work.

When you have typed in your name, type the TAB key. The cursor will move to the start of the next field – the position to the right of the word "Phone". Type something in this field, then type the TAB key again. You are now at the start of the "Address" field.

The only two keys on either keyboard that you don't know about yet are those at the top of the small keyboard labeled "FFld" and "BFld". FFld stands for Forward Field function which is the same function that the TAB key now invokes. The BFld key is used to invoke the Backward Field function. It moves the cursor back up the form to fields that have already been passed. Use these two keys to move from field to field until you are comfortable with what they do. What happens when you try to move back one field when you are already in the first field?

Experiment 33. Using the arrow keys, move the cursor into part of the form – so that it is over a character that is displayed in reverse-video. Now type some letters. The terminal rings the bell but will not put those letters in the screen because it would destroy part of the form.

Experiment 34. Move the cursor Home (FCN Q). Notice that it does not go Home but instead goes to the start of the form. Try the Erase to End-Of-Line function (FCN X). It will erase only the information in the field the cursor is in. Invoke Erase to End-Of-Screen (FCN Z). It will erase all your responses to the form but none of the form itself.

Experiment 35. Try the other functions that you know about. The Insert Line and Delete Line functions no longer work because you are not allowed to move the form. The Delete Character function still works, and so does Insert Mode, but when you delete or insert characters, you will not be able to change the contents or the location of the form. You can place the terminal in Editing-Keypad Mode (FCN K) if you want to use it to send special messages to the computer; you will probably leave the EDITING KEY-PAD light off if there are a lot of numbers to be entered into the form you are filling out. (If you put the terminal in Editing-Keypad Mode now, it will not perform the functions referred to by the white labels on the front of the keys because the terminal is not in Block Mode.)

When you have finished experimenting with this form, take the terminal out of Forms Mode by typing the ESC key, then capital O, then lower-case f.

OTHER FEATURES

The Ruler. Using the arrow keys on the small keyboard, move the cursor to the middle of the screen. Suppose you now want to find out what column the cursor is in. Type FCN C. A form called *the Ruler* has been written across the screen. This ruler will make it easy to figure out what column the cursor is in. Watch how you use this function, though, because if there were any text on the line the cursor was on when you invoked this function, the ruler will be written over this text, and you won't be able to get the text back; you'll have to retype it.

A good way to get rid of the ruler once you've figured out what column the cursor is in is to invoke the Delete Line function (FCN D).

KEYBOARD LOCKED. One of the red lights is labeled KEYBOARD LOCKED. The VT61 may fail to respond to the keyboard for a variety of reasons. For instance, if it is doing a Text Justify function which is taking a long time, the keyboard will go dead until the terminal is finished justifying the screen. If the terminal is transmitting information to the computer, it won't let you type, so that your typing doesn't get mixed in with its message. The computer has the ability to turn off the keyboard if it doesn't want to hear from you for a while. When the computer turns off the keyboard, this light will go on. Sometimes, after the terminal has finished sending a message to the computer, it will turn the KEYBOARD LOCKED light on and wait for the computer to turn it off.

You will know when the terminal is not responding to your typing because you won't hear the clicking sound. If the keyboard doesn't respond for a long time, the KEYBOARD LOCKED light will tell you that you are waiting for the computer to turn your keyboard on.

The SCROLL Key. Scrolling – the shifting of lines up and off the top of the screen can be a problem if the computer has a lot of information for you and the terminal and computer are set to communicate very rapidly. Information might leave the top of the screen before you have read it. Notice the SCROLL key at the lower left corner of the screen. The computer can set the terminal to wait until you signal before it lets any information leave the screen.

You'll type the SCROLL key to tell the terminal that it's all right to let another line of data onto the screen – even though the information on the top line will go away. You can type the SCROLL key with one of the SHIFT keys down to tell the terminal that you've finished reading all the information on the screen, and the terminal will let 24 new lines onto the screen, replacing all the information that was on the screen.

One of the red lights to the right of the small keyboard is labeled RECEIVER OFF. This light goes on whenever the terminal has told the computer it is busy. It goes off again when the terminal tells the computer it is ready to receive more information from the computer. One of the reasons the terminal might have for telling the computer it is busy is that it is waiting for you to type the SCROLL key. If the light is on for that reason and you do type the SCROLL key, then the light will go off and new information will come onto the screen. When all the new information you asked for is on the screen (either one new line or 24), the light will come back on again, and the screen will hold the information on it so that you can read it.

If you type on the keyboard and nothing happens on the screen, look at the RECEIVER OFF light. If it is on, the computer is waiting for the terminal to tell it to go ahead. The terminal may be waiting for you – type the SCROLL key to let it put more text on the screen.

Sometimes, such as when you are in a dialogue with the computer, the computer may set the terminal NOT to wait for you to type the SCROLL key.

TO LEARN MORE

These experiments have shown you how to use the keyboard of the VT61, how to read the red lights, and how to interpret what you see on the screen. It has shown you how to use the terminal to communicate with a computer, and how the terminal acts in Block Mode.

The experiments have not told you what you are supposed to say to the computer or how you're supposed to say it. In addition, what you see on the screen will depend entirely on the computer which the terminal will be attached to. The particular computer system you are using may have additional rules, may assign special meanings to some of the keys we've covered, and may make the screen appear differently than it has in these experiments.

You must find out what, if anything, the computer will do when it receives *Commands* or the special messages F1 through F10. You must find out when the computer will make the terminal wait for you to type the SCROLL key, and when it will proceed whether or not you type the SCROLL key. You know what commands are meaningful to the terminal; you must find out what commands are meaningful to the computer.

For Programmers

This part of the *VT61 Users' Manual* explains the concepts necessary to understand the workings of the terminal, and describes in detail the effects of all the commands to which the VT61 responds.

SECTION 1
VT61 CONCEPTS

The VT61 is both an interactive and a block-oriented video terminal with advanced editing features and optional hardcopy. Its intelligent features are implemented using Escape Sequence.

HARDWARE DESIGN

The VT61 utilizes a microprocessor rather than discrete logic alone to perform its functions. There are three categories of memory used in the microprocessor, not counting Read Only Memories (ROM) used to generate characters and perform other simple conversions. These three types are:

1. Control ROM (CROM)

   This memory contains the microprogram in a basic language consisting of five instructions (Move, Operate, Compare, Branch, and arithmetic operations). The CROM program has two functions:

   a. Decode the macroprogram found in the Macro ROM.

   b. Perform time-sensitive functions, such as video service.

   The size of the Control ROM is 512 16-bit words.

2. Macro ROM (MROM)

   This memory contains a *macroprogram* in a different language – a language in which each instruction stands for a series of CROM instructions, or a *Macro*. The CROM program decodes MROM instructions by branching to one of various routines depending on the type of MROM instruction encountered. Thus, the MROM program language can have a very sophisticated repertoire without requiring excessive decoding logic.

   4096 8-bit words are used to implement the VT61.

3. Random Access Memory (RAM)

   This memory contains both the data displayed on the screen and *scratch-pad* storage used by the MROM program to store variable data. It is the only memory whose contents may change. There are 2048 8-bit words of RAM used.

TERMINAL CONFIGURATION AND DATA STRUCTURES

The VT61 normally performs a two-part function. It is an input device to a computer (or *host*), such that information entered through the keyboard by the operator is sent out to the computer. It is simultaneously an output device where data coming in from the host is displayed on a video screen.

The terminal has various *modes* in which it reacts differently to incoming data. A command is required to either enter or exit a mode. Such a command may originate either from software or from the keyboard. The modes can be viewed as *settings* of the terminal. When placed in the correct modes, the terminal can be set to react compatibly with most environments.

In particular, one mode, Block Mode, is mentioned here because it changes the overall function of the terminal. In Block Mode, the VT61 buffers characters entered at the keyboard and makes them visible on the screen. The operator can enter and edit a message in a simulated Off-Line environment before choosing to transmit any data to the host.

THE SCREEN AND ITS GEOMETRY

The screen is seen as a matrix of character positions, with 24 rows and 80 columns. However, its structure is that of a single line with 1920 columns. That is, the terminal's advanced editing functions do not operate on a single one of its 24 lines, but on the screen as a whole. For instance, the rightmost character position on a line is considered to be adjacent to the leftmost position on the line under it. Each character position has the capability of retaining any character in the 7-bit displayable ASCII character set, in either light-on-dark or dark-on-light (reverse video). Corresponding to each character position is an 8-bit word of Random Access Memory, of which 7 bits contain the ASCII code for the character at the corresponding position, and the other bit selects the reverse video feature for that character. In Forms Mode these reverse-video characters become *protected* and cannot be changed or moved by the operator.

At any time, one of the 1920 character positions is said to be the cursor position. As well as being relevant to the editing functions of the VT61, the cursor position is the position where incoming displayable characters will appear. To make the location of the cursor obvious to the operator, the character at that position flashes – oscillates between normal and reverse video – approximately three times a second.

Characters with octal codes from 040-176 are called displayable characters. When a displayable character is received, the character is stored at the cursor position, whose previous contents are destroyed. The character position immediately following, or to the right of, this position becomes the new cursor position.

There are several variations to this formula which the host can specify. It does so by placing the terminal in special modes. By placing the terminal in Reverse-Video Mode and/or Graphics Mode, the host can force displayable characters to undergo a conversion before being stored in the screen. By placing the terminal in Insert Mode, the host says that the old data must shift over to make room for the new data, instead of being replaced by it. In Linear-Addressing Mode, if the cursor was at the end of a line, the start of the next line down will be the new cursor position. If the cursor was at the rightmost position on the bottom line, that is, at the last position in the screen buffer, an upward scroll is performed to make more room. In an upward scroll, all the information on the screen appears to move up one line. The information which was on the top line before the scroll was performed is lost; a new blank line appears at the bottom of the screen. The cursor is positioned at the beginning of this line which is filled with NULs (whose octal code is 000) rather than true spaces (040). The terminal can also perform downward scrolls on command.

NULs are not regarded as significant and are inserted and deleted as necessary to align, move, and justify displayable text on the screen, whereas spaces are treated as actual data characters and inserted and deleted only explicitly. Thus, a command to make a gap in the text on the screen would use NULs, never spaces, to fill the gap.

Likewise, when the VT61 is powered up, the contents of the screen buffer are initialized to NULs. At this time, the leftmost position on the top line – the first position in the buffer – is the cursor position.

Characters with codes from 000-037 or 177 are called *control codes.* Control codes are not normally entered into the screen. In Block Mode, however, TAB is stored in the screen upon receipt. Another character, Paragraph Delimiter (PD), may be placed in the screen by a command. TAB and PD are stored since they serve as markers in text justification. NUL, TAB, and PD can be displayed using special graphic characters, or they can be displayed as blanks. A switch on the underside of the VT61 governs this decision. By placing the terminal in Graphics Mode, software can store TAB, PD, and other graphic characters in the screen. The terminal uses the control codes to mark the position of these graphic characters in its memory.

The various special graphic characters are displayed as subscripts, fractions, mathematical symbols, and horizontal bars. Eight of the characters are seen as horizontal bars at eight different heights (video scans) in the character position. These eight graphic characters can be used to form line graphs with good resolution, since the value of a variable can be demonstrated not only by which text line a horizontal bar appears in, but by how high up it is within that line.

OVERFLOW BUFFER

At times, expansion of text in the screen buffer is done by shifting characters toward the end of the screen. A one-character Overflow Buffer exists to retain a character which may have been shifted off the screen by this process. The contents of the Overflow Buffer are transmitted to the host on command.

Any text-expanding operation which shifts non-NUL data out of the screen into the Overflow Buffer is aborted immediately without shifting further data out of the screen. In Block Mode, the terminal can be set to alert the host when a character enters the Overflow Buffer so that the host can take action to prevent the loss of data.

OUTPUT DEVICES

Text can also be output to a general output register (the *printer*) which produces an 8-bit parallel output capable of driving an upper-and-lower-case LA180 printer or equivalent. In addition to the line-by-line output available on the copier, a mode is provided in which all characters received from the host are sent directly to the printer rather than the screen. This mode, in which the VT61 functions essentially as a remote printer-controller, allows programmers to take advantage of all the printer's 132 columns.

If a terminal which has a printer receives one of the COPY commands, it will route the selected data to the printer.

The VT61 maintains an Output Abort Flag by which the host can determine if either the copier or printer failed to signal the terminal properly during an output operation, indicating that the device is out of paper or malfunctioning. The terminal can be set to alert the host when either of its output devices fails in such a way.

The VT61 can also treat the host (computer) to which it is attached as an output device, transmitting any specified portion of the screen, or the Overflow Buffer, on command. In Forms Mode, it is possible to have the VT61 send to the host only responses to the form rather than all the data on the screen, i.e., the responses and the form itself.

THE KEYBOARD

The VT61 employs the DEC-standard keyboard, which differs from a familiar office typewriter keyboard in one important way: The key usually called SHIFT LOCK is known as CAPS LOCK on the VT61. Depressing it will lock only the 26 alphabetic keys in the shifted (upper-case) state. To type a shifted symbol, such as : or $, one of the two SHIFT keys must be depressed. Depressing the CAPS LOCK key will cause the VT61 keyboard to become an upper-case-only keyboard. (Note that, even with the CAPS LOCK key down, the terminal will still be able to transmit `, {, |, and } , symbols which are not included in the 64-character, upper-case-only ASCII subset.)

At the lower left corner of the keyboard, there is a key labeled SCROLL. This key is used when the terminal is in Hold-Screen Mode to allow the operator to request the terminal to let more data from the host onto the screen. (The mechanics of this function are described completely in Section 2 under **HOLD-SCREEN MODE**.)

At the lower right corner of the keyboard, there is a key labeled COPY. This key can be used by the operator to output the contents of the entire screen to the printer. This key is unique in that it invokes the Copy Screen function locally without sending any codes to the host. (In Block Mode, the terminal can be set to send a *request to copy* to the host before performing the output.)

If the COPY key is typed with the SHIFT key held down, the terminal will enter Auto-Copy Mode if it was not in it already, or exit it if it was in Auto-Copy Mode when the key was typed. This signal to the terminal likewise causes no codes to be sent to the host. (This use of the COPY key is not provided in Block Mode.)

To the right of the keyboard is a 19-key keypad. Twelve of these keys transmit numerals and other codes for use in numeric data entry. Software can put the terminal in a mode in which these 12 keys transmit the most common multiple-character commands used in the VT61. Six other keys are dedicated to cursor movement and also transmit more than one character per keystroke. The remaining key, labeled FCN, transmits codes which change the meaning of keys typed subsequently. This key is provided for convenience in forming VT61 commands from the keyboard. Appendix A describes in detail the appearance and function of the auxiliary keypad.

When a key is depressed, the corresponding code or codes are transmitted. If the REPEAT key is held down, these codes will be transmitted repeatedly. The rate of repetition may attain 30 characters per second, or it may be limited to a slower rate if the Baud rate switches are not set to accommodate such rapid transmission.

Several keys are *self-repeating*. They will transmit the proper code or codes once when they are first pressed. If they are held down for longer than half a second, they will begin to transmit the same code or codes repeatedly until the key is lifted, just as if the REPEAT key had been held down. If the REPEAT key was in fact depressed, these keys act as the other keys do – they transmit the appropriate code(s) repeatedly but without a half-second wait. The self-repeating keys are:

1. The four keys marked with arrows (up, down, left, and right) on the auxiliary keypad
2. The Space bar on the main keyboard
3. The BACKSPACE key on the main keyboard
4. The DELETE key on the main keybaord
5. (In Block Mode and Alternate-Keypad Mode) The keys on the auxiliary keypad labeled "DelCh" and "CEmph"

The proper code or codes are sent when the key is first pressed. The operator may press a second key while the first key is still down, and the second key will function normally. If the operator holds three keys down simultaneously, the code(s) for the third key will be sent only if one of the first two keys is lifted before the third key is lifted.

OTHER OPERATOR SIGNALS

Eight light-emitting diodes (LEDs) appear to the right of the keyboard. They indicate a special mode or condition within the terminal. They are labeled to show which mode or condition they indicate. This labeling, along with the relative positions of the LEDs, is given in Appendix C.

The terminal can make two audio signals to the operator: A staccato clicking occurs to acknowledge the pressing of any key. (Those keys which simply modify the function of other keys – SHIFT, CAPS LOCK, CONTROL, and REPEAT – do not produce keyclicks when pressed.) A longer tone, called the bell, is used as an alarm to the operator. Such an alarm sounds when the control code BEL (007) is received or in other situations in which it is desirable to alert the operator. These situations are listed under the description of the corresponding function. A potentiometer is provided to adjust the volume of the audible signals.

SWITCHES

The operator controls can be classified in two groups, based on the ease with which the operator can access them. Unless stated to the contrary, these controls are two-position slide switches. Their functions are described below.

1. Easy Operator Access (located on underside of unit)

VOLUME – This potentiometer varies the volume of both audible signals – the keyclick and the bell.

OFF-LINE/ON-LINE – In the Off-Line setting, information typed on the keyboard will go to the terminal's screen buffer or command interpreter, but only there. In the On-Line setting, communication with a host is enabled subject to the settings of the LOCAL LOOP and FULL/HALF DUPLEX switches. The operator and Field Service use the OFF-LINE/ON-LINE switch to local-test the terminal. In addition, the operator may use the terminal in this setting without a computer to familiarize himself with the terminal's features, or to do work for which a host computer is not necessary, such as creating a memo on the screen and outputting copies to the printer.

LOCAL LOOP ON/OFF – In the ON position, data transmitted to the host is wired back to the receiver of the terminal. Setting the switch to LOCAL LOOP ON does not take the VT61 off-line.

DISPLAY ALL – Determines whether NULs, TABs, and PDs contained in the screen buffer are displayed and copied as blanks or using the special visual characters.

Two rotary switches are also found on the underside of the VT61. They are provided to select the transmitting and receiving Baud rates. One controls the receiver speed and has the following settings:

Receive at 600 Baud
Receive at 1200 Baud
Receive at 1800 Baud
Receive at 2400 Baud
Receive at 4800 Baud
Receive at 9600 Baud
Receiver speed matches transmitter speed

The other switch controls transmission speeds and can be set as follows:

Transmit at 110 Baud
Transmit at 75 Baud
Transmit at 150 Baud
Transmit at 300 Baud
Transmit at 600 Baud
Transmitter speed matches receiver speed

The terminal will not operate if both switches are in the *match* position, nor will it operate if either switch is set to an unlabeled position. When the OFF-LINE/ON-LINE switch is set to OFF-LINE, the clock selected by the receive switch is used for transmission as well as reception.

2. Difficult Operator Access (located on internal electronic board)

FULL/HALF-DUPLEX – The VT61 is set in the HALF DUPLEX setting to communicate through a BELL 202 data set.

50/60 – Is set to match the line frequency.

PARITY ON/OFF – Determines whether the parity bit of characters will be checked on input and provided on output. If this switch is OFF, characters are sent with *space* (0) parity. If the switch is ON, parity depends on the switch described next.

PARITY EVEN/ODD – Determines whether even or odd parity will be supplied on outgoing characters and required of incoming ones. This switch is relevant only if the PARITY ON/OFF switch is ON.

FORCE BLOCK MODE – If this switch is in the ON position, the terminal will enter Block Mode and Transmit-Request Mode when power is first applied to the terminal or when a command reinitializes the terminal. These modes are defined in Section 2. Enabling these modes will cause an Escape Sequence to be transmitted to the host to inform it that the VT61 is powered up.

FULL DUPLEX SYNCHRONIZATION

The terminal can operate at transmission speeds up to 9600 Baud. At these high speeds, the terminal may not be able to keep up with incoming data if it is, for instance, performing output, or if the operator has explicitly commanded the terminal not to display any more incoming data. In this case, the terminal will begin storing incoming characters in a buffer called the *Silo* and processing them on a first-in, first-out basis. When the Silo begins to fill up, the terminal will transmit 023 (XOFF or DC3). On this signal, the host is supposed to suspend its transmission to the terminal. Eventually, if the host stops transmitting, the terminal will process characters out of the Silo exactly as if it had just received them from the host. On processing the last character in the Silo, the terminal will transmit 021 (XON or DC1) to signal the host that it may resume transmission.

If the host fails to respond to an XOFF from the terminal, the Silo will continue to fill up. It is possible that the slower operations, which initially caused incoming characters to be sent to the Silo, can be finished by the terminal, and that the Silo now contains characters that can be processed more quickly than they are received. In this case, the terminal will cut down the backlog of characters, empty the Silo, and send XON to the host. But, if the terminal continues to get commands faster than it can process them (since the host ignored the XOFF), the capacity of the Silo will be exceeded. This condition is called *Silo overflow.*

If the Silo overflows, the VT61 will abort the operation in process and begin to process the characters in the Silo in the proper order. (If the host continues to send, the actions requested by these characters may likewise be aborted.) Consider two examples:

1. The host commands the terminal to dump the screen to the printer. The terminal begins the output and buffers subsequent characters from the host in the Silo. The terminal sends XOFF to the host as the Silo begins to fill up. Now, assume the host ignores XOFF – that is, it continues to send characters to the terminal. Eventually the Silo will overflow. At that time the terminal aborts the printing operation; it ceases to send characters to the printer. The terminal starts processing characters out of the Silo, beginning with the character it received immediately following the print command. If the first thing it encounters from the Silo is another command which will take a long time to execute, it begins the operation, but this operation may also be aborted if the host continues to send. The VT61 will not send a second XOFF to the host.

2. The terminal is in Hold-Screen Mode (which is described in Section 2) in which it will try to avoid loss of data from the screen due to scrolling. The terminal receives a command to scroll the display, so it puts that command – and all characters received after it – in the Silo, eventually sending XOFF to the host. If the Silo overflows, the operation which will be aborted is the waiting that the terminal is doing – it is waiting for the operator to type the SCROLL key which permits the SCROLL command to be executed. If the Silo overflows, the terminal performs the scroll requested by the host, even without permission from the operator. THE TERMINAL DOES NOT LEAVE HOLD-SCREEN MODE.

One of the LEDs on the console, labeled RECEIVER OFF, lights up when the terminal has transmitted XOFF to the host under these conditions; it is turned off when the terminal sends the XON signal. (Sending XOFF and XON manually – by typing CONTROL S and CONTROL Q, respectively – does not affect this LED.)

Software which does not support receipt of the XOFF/XON signals from the terminal can still use the VT61 as a Teletype ® replacement if it never sends the code ESC to the terminal. This constraint eliminates the use of the advanced editing features of the VT61, and the use of the printer or copier, but will ensure that the terminal will not transmit XOFF to the host at any Baud rate. Another constraint which must be followed to avoid XOFFs sent by the terminal is that the COPY key must not be used. The VT61 does not normally inform the host that the COPY key has been pressed, nor give software the ability to prevent the copying from occurring. In this way, the host could send information to the terminal during the slow output operation and might receive XOFF from the terminal.

Two of the terminal's functions, Reset Terminal and Terminal Self-Test, re-initialize the terminal and erase the Silo. This means that if characters received directly following commands to perform these two functions were placed in the Silo, they would be destroyed before they were processed. For this reason, these two functions invoke an *Implied XOFF* rule: Immediately after sending the terminal commands to perform either of these functions, the host must act as if it had just received XOFF from the terminal, sending no more characters until it receives XON. The terminal will transmit XON after it completes the specified operation.

The host can use XOFF and XON in a similar manner to control block-transmissions from the VT61. When the VT61 receives XOFF, it will suspend any block-transmission in progress and wait for XON. When it receives XON, it will resume the transmission from the point at which it left off. The VT61 will transmit no more than two characters without checking to see if the host sent an XOFF. Programmers should take this fact into account – as well as the transmission delays and the ratio of the Transmit Baud rate to the Receive Baud rate – when planning the size of the host's input buffer.

During block (automatic) transmissions from the VT61 to the host, the VT61's receiver remains active. The VT61 is, therefore, capable of true *Full Duplex* communication. However, the VT61 is not capable of processing incoming characters at the same time it is block-transmitting. Characters received during a block-transmission will go to the Silo; as the Silo fills up, the VT61 will embed XOFF in its message to the host to warn it to suspend transmission to the terminal.

> **CAUTION:** Typing the proper keys for one of the Transmit functions when the terminal is set for Off-Line use, or Full Duplex with Local Copy (in which the VT61's transmitter is electrically connected to its receiver), will cause the terminal to transmit to itself, filling up the Silo. When the terminal sends XOFF to itself to suspend the transmission, it will be capable of no further action without being switched off and on again.

The XOFF/XON synchronization scheme has an advantage over requiring the host to insert delays or filler characters in its data stream. Requiring a minimum of software support, XON/XOFF ensures that every character or command sent to the VT61 will be processed in correct order. It frees interface programs from all timing considerations and results in more reliable communication. Effectively there is room for 23 codes in the Silo. That is, when the terminal determines that it must send XOFF, Silo overflow will not occur unless the number of characters in the Silo increases by 24. (Actually, the Silo will already have a number of characters in it before the terminal determines that it must send XOFF.) For most applications, 23 characters is sufficient to allow for transmission delays. One cause of Silo overflow is excessive processing time and transmission-line delay. To illustrate, if the VT61 were connected to the host such that messages from the terminal reached the host 10 seconds after leaving the terminal, you can see how an XOFF sent by the terminal might not reach the host in time to halt the host's transmission.

Another cause for Silo overflow is the use of a receiving speed which is much more rapid than the transmitting speed. If the VT61 were set to receive at 9600 Baud while transmitting at 300 Baud, 9600/300 = 32 characters could be received in the time it took the terminal to send XOFF. This problem would not exist if the transmission speed were also 9600 Baud. Thus, the use of high Baud rates is not by itself a cause of synchronization failure – although it does magnify the effect of long delay times.

The sufficiency of the VT61's Silo can be verified using the formula

$$(D + d + P)\ Y + \frac{3Y}{X} + 2$$

to yield the number of characters that the VT61 would have to buffer at worst. If this equation produces a number 23 or less, the VT61 will fit the application since the effective length of the Silo is 23 characters. In the formula above,

D is the transmission delay from terminal to host
d is the transmission delay from host to terminal
P is the processing delay

P is the interval from the time the host physically receives XOFF to the time software transmits its last character before stopping in response to the XOFF. All three of these times are in seconds in the above formula.

Y is the receiving speed of the terminal
X is the speed at which the terminal transmits

Both of these times are expressed in characters per second (char/sec). 9600 Baud = 960 char/sec, 1200 Baud = 120 char/sec, etc., but 110 Baud = 10 char/sec. The numbers 2 and 3 in the above formula take into account buffering done not by software at the host or by the microprogram of the terminal, but by the interface hardware. This hardware is typically a pair of UAR/Ts, one inside the VT61 and one inside the computer. If a computer with interface hardware which buffers more than a single character is used, the 2 in the formula should be increased accordingly.

As an example, consider a VT61 communicating to a host at 300 Baud (transmitting and receiving) over a telephone line. The telephone company specifies that the worst-case transmission delay is not greater than 50 msec. In this example, the processing delay was zero since, when the host received XOFF, it trapped to a routine which disabled output. So the last character the host transmitted to the terminal would occur before receiving XOFF. Since transmitting and receiving speeds are both 30 char/sec, the equation is:

$$(.050 + .050 + 0)\ 30 + 3\ \frac{(30)}{(30)} + 2 = 8$$

A terminal with an 8-character input buffer is required. The VT61 with 23-character Silo is adequate for the application.

The formula given above differs slightly from the corresponding formula given in the *DECscope Users' Manual* to describe the timing requirements of the VT50-series terminals, since the VT61 uses a different algorithm to interact with the UAR/T.

Appendix E lists all the Baud rate settings at which the size of the Silo suffices to prevent overflow, and specifies the maximum delay times (due to transmission and processing at the host) by which the host must respond to XOFF to prevent Silo overflow.

HALF DUPLEX

When the FULL/HALF DUPLEX switch is set to HALF DUPLEX, the terminal uses several signals to communicate via a data set. The following signals are used by the data set to send information to the terminal:

1. Carrier Detect (CD)
2. Supervisory Receive Data (SRD)
3. Clear to Send (CTS)

The VT61 uses the following signals to control the data set:

1. Request to Send (RTS)
2. Supervisory Transmit Data (STD)
3. Data Terminal Ready (DTR)

The VT61 asserts the DTR signal whenever it is switched on. When the VT61 is used with a data set which will not hang up until DTR is deasserted, it will be necessary to switch the terminal off in order to terminate a call.

The VT61 uses the RTS signal in order to request control of the line. It attempts to gain control of the line whenever it is requested to perform a transmission to the host. It also attempts to gain control of the line in response to the Escape Sequence ESC O Z from the host. If the CTS signal is asserted, indicating that the VT61 already has the line, the requested transmission begins immediately, or in the case of ESC O Z, the operator may use the keyboard to transmit to the computer. If CTS is not asserted, it will do the following things, in order, to gain control of the line.

1. Wait for CD to become deasserted
2. Deassert STD
3. Assert RTS
4. Wait for CTS to become asserted

One of the LEDs on the console is labeled CLEAR TO SEND and monitors the state of the CTS signal.

When the VT61 has finished transmitting to the host, it drops (deasserts) RTS and asserts STD. In order to ensure that the message to the host has gone entirely through the data set before RTS is deasserted, the VT61 delays approximately 300 milliseconds after the final character has been transmitted before deasserting RTS.

When the HALF DUPLEX switch is off and the terminal is in a Full Duplex environment, the VT61 will assert RTS whenever it is switched on. It will also ignore the CTS signal from the dataset, transmitting to the host regardless of the state of CTS.

If the host made the VT61 gain control of the line in order to receive input from the operator, then, after it has determined that the message from the operator has ended, it must deassert its STD signal. The VT61 will give up control of the line. If the host deasserts its STD signal in the middle of a block-transmission from the terminal, the remainder of the transmission is aborted.

The operator can use the BREAK key on the keyboard to manipulate the STD signal. If the terminal does not have RTS asserted, typing the BREAK key will deassert the VT61's STD for between 250 msec and 400 msec. If the terminal has control of the transmission line, typing the BREAK key will force the data line to the zero (*space*) state for between 250 msec and 400 msec.

The BREAK key can be used to forcibly interrupt the flow of data from the host in Half Duplex situations; it can be used for this purpose even if the host locks the keyboard by putting the terminal in Disable-Keyboard Mode (described in Section 2); the BREAK key is never locked.

> **WARNING:** If the host transmits to the VT61 out of turn, there is the danger of a *race condition* occurring — both host and terminal requesting and getting the line at the same time. When in a Half Duplex situation, the VT61 will work correctly in a controlled environment in which neither host nor terminal transmits out of turn.

TERMINAL COMMANDS

Input (from the host or, in Block Mode, from the keyboard) is displayed on the screen if it is in the range 040-176. Other codes, the control codes, are not displayed and do not implicitly move the cursor right. Some codes initiate special action by the terminal. In particular, the code 033, ESC or Escape, puts the terminal in a state in which one or more displayable characters following the character ESC will nevertheless not be displayed but interpreted as a command to the terminal. The character ESC and all characters following it which are interpreted rather than displayed are known as an *Escape Sequence.*

An Escape Sequence consists of the character ESC (033), optionally followed by any number of intermediate characters (whose octal codes are in the range 040-057), and always followed by a final character (in the range 060-176). For example, ESC A is an Escape Sequence which commands the terminal to move the cursor up one line. The final character in this Escape Sequence is "A"; there are no intermediates. ESC / A is an Escape Sequence with an intermediate character "/". This Escape Sequence is sent to the host by a VT50 to identify the terminal as a VT50. The VT61 has several similar Escape Sequences it uses to identify itself (on command); these Escape Sequences happen to be the only ones used in the VT61 which have intermediate characters.

Several Escape Sequences must be followed by one or more *Parameters.* The octal code for a Parameter may be in the range 040-176 — that is, any displayable character may be used as a Parameter. However, Parameters are not displayed on the screen, but rather interpreted by the terminal (just as the Escape Sequence was). Parameters provide specific information about the type of function to be carried out. The VT61 uses Parameters after Escape Sequences in these situations:

1. ESC ? must be followed by a Parameter. An Escape Sequence of the form ESC ? ⟨parameter⟩ is sent to the host whenever a key on the numeric pad is typed when the terminal is in a special mode. The Parameter specifies which key was typed.

2. ESC O must be followed by a Parameter. Escape Sequences of the form ESC O ⟨parameter⟩ are a group of commands to the terminal which will normally originate from the host. The Parameter specifies which command should be carried out by the terminal. For instance, ESC O B places the terminal in Block Mode, but ESC O T tells the terminal to test itself.

3. ESC P must be followed by a Parameter. Escape Sequences of the form ESC P ⟨parameter⟩ form a group of commands to the terminal which may originate from the host or from the keyboard. As with the ESC O family of commands, the Parameter specifies the exact function to be carried out. ESC P D means delete line, but ESC P J performs output to the printer. To assist the operator in typing these Escape Sequences, the first two characters, ESC P, can be generated by typing a single key: the FCN key on the auxiliary keypad. The operator goes to the main keyboard to select the Parameter for the desired function; abbreviations of all the functions are provided on the fronts of the applicable keys. For example, if the operator wants to delete a line, he should generate the codes for ESC P D. He types FCN D, which generates the codes. To help him remember to type the D, there is a label on the front of the D key which says "DelLn" — an abbreviation of the function he wanted. For convenience,

the final keystroke, which selects the particular function to be performed, may be either shifted or unshifted. In the mode-type commands, a capital letter as the Parameter will cause the terminal to enter the specified mode; if the final keystroke is unshifted, the terminal will leave the mode.

4. ESC Y must be followed by two Parameters. ESC Y ⟨p1⟩ ⟨p2⟩ is a command to the terminal to place the cursor at a specific place in the screen. It is also the Escape Sequence the terminal sends to the host if it asks the terminal where the cursor is at present. The first Parameter specifies a particular line on the screen; the second Parameter specifies a particular column.

The synchronization control codes, XON and XOFF, are acted on promptly after they have been received from the host. Other control codes, which command the terminal to perform functions, may be placed in the Silo for a while. The terminal carries out the command when the control code is *processed* — either directly from the host or out of the Silo. Commands which are Escape Sequences are not carried out until the last character in the Escape Sequence, or the last Parameter, is received. Therefore, if a control code is embedded in an Escape Sequence — if the host sends ESC to the terminal and then sends a control code before it has sent characters to complete the Escape Sequence, the function requested by the control code will be carried out by the terminal before the function requested by the Escape Sequence. However, carrying out the control code function may interfere in unspecified ways with the terminal's ability to carry out the Escape Sequence function correctly.

The functions of the various control codes are described, as are the Escape Sequences, arranged by their functions, in the next four sections.

ERRORS

The operator or host will occasionally give the terminal a command which cannot be carried out under the current circumstances. In some of these situations, the terminal will respond by sounding the alarm to alert the operator that something abnormal has occurred — for instance, that the cursor is in a protected field and the operator is not allowed to enter data there. These situations are called *errors* and are pointed out explicitly at the end of the description of the corresponding mode or function.

In some situations, the circumstances will not allow the requested operation to be carried out, but the terminal will not sound the alarm. An example of such a situation is a command to move the cursor up one line when the cursor is already on the top line. These situations are not termed errors and are not emphasized but, rather, included as part of the definition of the operation.

SERIAL ENCODING: PARITY AND STOP BITS

Codes going through the wires between the terminal and the host are serially encoded. Ten or eleven signals (bits) are sent over the wires, one following another. A bit is sent by holding the transmission line at either the "0" or the "1" state for a certain amount of time, dictated by the setting of the Baud rate switches. (At 300 Baud, this time would be 1/300 second.) The bits are sent down the line in the following order:

Bit 1	A start bit to indicate the beginning of a code
Bits 2 - 8	Seven bits to represent the code
Bit 9	A parity bit
Bit 10	A stop bit

The parity bit transmitted depends on the code transmitted. In the Even Parity scheme, the parity bit is chosen so that an even number of bits 2 through 9 will be "1". If the device which received this code determined that an odd number of those bits was "1", it could tell that it hadn't received that code correctly.

The VT61 can be switched to transmit codes with Even or Odd Parity, or to use Mark Parity in which it always sends a "1" as the parity bit regardless of evenness. The VT61 will always transmit codes in accordance with the settings of the PARITY ON/OFF and PARITY EVEN/ODD switches on the inside of the unit.

If the VT61 receives a character which does not conform to the settings of the parity switches, it will take the following actions:

1. Display a special character in the screen to represent the character received with bad parity.
2. Cause the bell signal to sound.
3. Cancel any pending Escape Sequence. (Remaining characters in the Escape Sequence are treated as if they had been received normally, that is, not within an Escape Sequence.)

If the PARITY ON/OFF switch is OFF, the VT61 will ignore the parity on incoming characters.

At most Baud rate settings, the VT61 transmits a total of 10 bits per character. At 110 Baud, the VT61 will append an extra stop bit at the end of each character transmitted for a total of 11 bits. Note that if the transmitting device were set to transmit 10 bits, but the receiving device were set to expect 11 bits on the incoming characters, then the transmitting device could send two characters, one so soon after the other that the receiver would not be ready for the second character since it would still be waiting for the first character to end. The second character would thus be received improperly. This could be the cause of sporadic transmission errors which occur when transmission is being done as fast as possible under the current Baud rate.

When the VT61 is set to transmit at 110 Baud, it will expect 11 bits per character from the host, regardless of the setting of the receiving speed.

INVOKING FUNCTIONS FROM THE KEYBOARD

In this document, much effort is made to discuss the manner in which the keyboard is used to invoke VT61 functions. The reader is reminded that, except for the SCROLL and COPY keys, the only function the keyboard performs is to *transmit* characters, including Escape Sequences, whereas the functions described herein take effect only when such Escape Sequences are *received* by the VT61. Typing a key does *not* cause a character to appear on the screen *per se*; the correct code must appear at the receiver in order to cause that action to take place. The same is true for the more advanced features, with the following implications:

1. In Full Duplex, Escape Sequences coming from the VT61 must be sent back (echoed) literally in order to produce the requested effect at the terminal. (The host, by failing to echo the Escape Sequence, has the option of *vetoing* the operator's request and/or taking alternative action.

2. When set for Off-Line use, each character which, through action of the keyboard, is sent to the transmitter appears at the receiver due to an electrical connection (and, in fact, is not even sent to the host). The operator, therefore, has all the VT61 functions unconditionally available to him. They will be done automatically as he requests them from the keyboard.

3. When set for Local Loop, the operator likewise has all the VT61 functions at his disposal, but he may be hindered invoking them should the host be sending spurious characters at the same time. For instance, if the operator requests an operation by typing FCN A and if the host simultaneously sent a B to the terminal, the function performed might be FCN B. Therefore, when using the LOCAL COPY setting, both software and the operator should take care that communication is taking place only one way at a time – from terminal to host, or from host to terminal. If the operator types before the host has finished or if software transmits out of turn, not only may the two messages appear to be interlaced on the screen, but execution of the wrong Escape Sequences may occur. (The problem is actually worse than this. Not only may the characters be interposed, but the serial bit streams of the two messages are OR-ed, which can produce characters not in either message.)

4. When the terminal is in Block Mode, typing on the keyboard is sufficient to display characters and invoke functions, although it is due to firmware rather than an electrical connection, and the physical transmitter and receiver are not used. (The operator does not have all the functions at his disposal, since in Block Mode the terminal will not allow some functions to be invoked from the keyboard.)

THE VT61 IN A COMMUNICATIONS ENVIRONMENT

In a communications environment, the VT61 will play the role of the slave. The host is responsible for correcting any errors that may occur, and the VT61 will not take an active role in pointing out that an error has occurred. Consider two examples:

1. The VT61 computes checksums of all transmissions and can detect that an error has occurred, but it will never transmit an unsolicited message to the computer to tell it that an error has occurred.

2. If the host requests a transmission from the VT61 and does not get a response from the terminal, the host must assume that there was an error in the interpretation of the command to transmit, and it is entirely up to the host to rectify the situation.

By placing the terminal in Transmit-Request Mode, however, the host can specify that the terminal notify the host when some situations occur in which intervention by the host may be necessary.

SECTION 2
TERMINAL MODES

NOTE: The table which follows is only a partial listing of the VT61 command forms. Similar tables listing the remaining commands can be found at the beginnings of Sections 3, 4 and 5. A comprehensive table is provided as Appendix D.

Several modes exist in which the ground rules describing the terminal's behavior are changed. Much of what was set forth in the preceding section is dependent on the terminal being in the Default Modes, and can be changed at the will of the operator, programmer, or system designer. This section describes the changes in the ground rules invoked by each mode. Modes are entered and exited upon receipt by the terminal of certain control codes or Escape Sequences which fall into three groups:

	to ENTER	to EXIT	MODE
1. Escape Sequences which the operator will probably want to invoke from the keyboard (FCN stands for ESC P since the FCN key on the auxiliary keypad transmits ESC P).	FCN K	FCN k	editing-keypad
	FCN Y	FCN y	auto-print
	FCN U	FCN u	hold-screen
	FCN I	FCN i	insert
Notice that the final character for each of these commands must be SHIFTED to enter the mode and UNSHIFTED to exit the mode.			
2. Escape Sequences which will usually originate only from the host.	ESC O A	ESC O a	maintenance
	ESC O B	ESC O b	block
	ESC O C	ESC O c	linear-addressing
	ESC O D	ESC O d	new-line
	ESC O E	ESC O e	disable-keyboard
	ESC O F	ESC O f	forms
	ESC O G	ESC O g	alarm
	ESC O H	ESC O h	transmit-request
	ESC O I	ESC O i	auto-tab
	ESC O J	ESC O j	reverse-video
3. Forms added for compatibility with earlier VT models.	ESC =	ESC >	alternate-keypad
	ESC F	ESC G	graphics
	ESC W	ESC X	printer-controller
	ESC [	ESC \	hold-screen
	ESC ^	ESC _	auto-copy

NOTE: The operator may enter or exit Auto-Copy Mode from the keyboard by typing the COPY key with the SHIFT key held down. This causes the terminal to enter the mode if it was not in the mode to begin with. If the terminal was in Auto-Copy Mode, typing the shifted COPY key will cause the terminal to exit the mode. Typing shifted COPY sends no codes to the host. In Block Mode, this use of the COPY key is not allowed.

Note the difference between a *mode* and a *command*: A *command* is required to enter a *mode.* When the terminal is in the mode, its behavior changes. The terminal remains in the mode until a command to exit the mode is received. This differs from usual commands which have an immediate, one-time-only effect.

This document defines the modes of the VT61 so that, when the unit is switched on, all the modes are *exited* (disabled).

INSERT MODE (complement of Replace Mode, default; affects Overflow Buffer; lights second LED from top when enabled)

entered by ESC P I
exited by ESC P i

Normally, each displayable character that the VT61 receives is stored at the cursor position in the screen, destroying (replacing) the old character at that position and moving the cursor right. In Insert Mode, information is inserted between the cursor position and the position directly to the left of it. The cursor, and text there and rightward, are shifted right to make room.

The entire screen buffer to the right of the cursor position is not necessarily shifted rightward to make room for the new (inserted) character; only the text up to the first NUL. If the character at the cursor position was a NUL, no text is shifted rightward. In this case, there is no difference between Insert Mode and Replace Mode. If the character at the cursor position is non-NUL, but there is a NUL on the same line to the right of the cursor, only the text from the cursor position up to that NUL will be shifted rightward; the NUL will be compressed out of the screen. If the nearest NUL to the right of the cursor is on a lower line, all the text up to that NUL will be shifted rightward one character to make room for the new character, and characters on the rightmost column of some lines may wrap around to the first character position on the next line down as part of the rightward movement. If there are no NULs to the right of or below the cursor position, the entire contents of the screen buffer from the cursor position to the end of the screen are shifted rightward, meaning that the character at the last position on the last line is lost from the screen. This situation is an *error* and is described below.

Whether the terminal is in Insert Mode or Replace Mode, after a displayable character is received and stored in the screen buffer, the cursor is moved rightward one position. If the terminal is in Rectangular-Addressing Mode, the cursor will not move if it was already at the rightmost position on a line.

If the terminal is not in Rectangular-Addressing Mode, the cursor moves from the rightmost position on a line to the leftmost position on the next line down; and, if the cursor was on the last position on the bottom line, a scroll is performed, moving the cursor to the beginning of a new line below all the others.

An error occurs when a meaningful (non-NUL) character is shifted out of the screen buffer. It goes to the overflow buffer where it can be read by the host. The normal rightward movement of the cursor still occurs.

In Forms Mode, insertions affect only the single field containing the cursor. *An error occurs* when a meaningful (non-NUL) character is shifted out of the field. It goes to the overflow buffer where it can be read by the host.

The overflow buffer is cleared after each character inserted in Insert Mode if an error did not occur.

LINEAR [-ADDRESSING] MODE (complement of Rectangular [-Addressing] Mode, default)

entered by ESC O C
exited by ESC O c

There are two ways to think of the character positions on the screen. One way is to think of them as you see them: as a rectangle of 24 lines of 80 characters. Another way to view the screen is to consider the last position in one line and the first position in the line under it to be adjacent. The screen becomes one long string of 1920 character positions in this *linear* way of viewing the screen.

The terminal always operates on the screen as if it were one long string, when it comes to text-expanding and text-contracting operations. When text is inserted in the screen, characters which are rippled off the end of one line appear on the next line down. When characters are deleted, other characters may come up from lower lines on the screen to fill up the space.

But when a Cursor Right or Cursor Left command is issued with the cursor already as far as it can go in that direction on the line, the cursor will not move. Likewise, when text is being displayed on the screen, the cursor will stop moving rightward after each character is displayed when the end of the line is reached. In these three situations, the terminal takes a *rectangular* view of the screen.

Software can force the terminal to take a *linear* view in these three situations by placing it in Linear-Addressing Mode. In Linear Mode,

1. If the cursor is at the beginning of a line, a Cursor Left or Backspace command will move it to the end of the line above. (If the cursor was at the beginning of the top line, it does not move.)

2. If the cursor is at the end of a line, a Cursor Right command will move it to the beginning of the line below. (If the cursor was at the end of the bottom line, it does not move.)

3. If the cursor is at the end of a line and a displayable character is received, it is stored at the cursor position, and the cursor is moved to the beginning of the line below. If the cursor was at the end of the bottom line, an upward scroll is performed and the cursor goes to the beginning of the new bottom line.)

REVERSE-VIDEO MODE

entered by ESC O J
exited by ESC O j

Reverse-Video Mode allows characters to be displayed in a reverse-video, reduced-intensity form, providing high contrast with other text.

Any displayable character (codes 040-176, except as part of an Escape Sequence) received with the terminal in Reverse-Video Mode will be displayed in reverse video. The VT61 remembers this fact internally by setting the high-order bit in the byte at the position where this character is stored.

This mode is mutually exclusive with Forms Mode since reverse-video characters are *protected* in Forms Mode. Forms Mode supersedes Reverse-Video Mode. If the terminal is in Reverse-Video Mode, a command to enter Forms Mode will cause the terminal to enter Forms Mode and exit Reverse-Video Mode. If the terminal is in Forms Mode, a command to enter Reverse-Video Mode will not be obeyed.

FORMS MODE (lights fourth LED from top when enabled)

entered by ESC O F
exited by ESC O f

Forms Mode causes the terminal to make a distinction between two kinds of data on the screen: A form (in reverse video) and responses. The form is protected against modification of any kind.

In Forms Mode, any character which is being displayed in reverse video is considered to be part of the form. *Fields* are strings of one or more adjacent characters not in reverse video. (Remember that the rightmost character on a line is *adjacent* to the leftmost character on the next line.) Reverse-video characters on the screen may not be erased, changed, or moved while the terminal is in Forms Mode. The cursor may enter a protected area, but if a displayable character is received, it will not be stored at the cursor position; instead, the bell will ring and the cursor will remain where it was.

Upon entering Forms Mode, the following actions are performed:

1. The terminal enters Linear-Addressing Mode and exits Reverse-Video Mode if it was in it. Furthermore, it is impossible to enter Reverse-Video Mode if the terminal is in Forms Mode.

2. The cursor is moved to the first unprotected (non-reverse-video) position on the screen, i.e., the beginning of the first field.

If all positions on the screen are protected, the cursor will be placed at the *Home* position (the first position in the buffer), but the terminal will not enter any new information into the screen.

AUTO-TAB MODE

entered by ESC O I
exited by ESC O i

Auto-Tab Mode has no effect unless the terminal is in Forms Mode.

Normally, in Forms Mode, if a character is entered into the rightmost position in one field of a form, the usual rightward movement of the cursor which occurs after a character is stored in the screen will cause the cursor to move to a position which contains a protected character. If an attempt is made to enter subsequent characters in the screen without first moving the cursor, the attempt will fail, and the bell will ring.

In Auto-Tab Mode, when a character is entered into the last position in a field of a form, the bell rings, and the cursor moves to the first position in the next field. The cursor movement performed by the terminal when this happens is the same as in the TAB function, which is described in Section 3.

Auto-Tab Mode thus relieves the operator of having to type the TAB key after filling each field in order to move the cursor to the next field. This is useful when filling out forms in which every field in the form is to be filled in completely. On forms where the response to an item may not occupy the entire field of the form, the operator will typically type TAB anyway to explicitly move to the next field.

GRAPHICS MODE

entered by ESC F
exited by ESC G

Using Graphics Mode, it is possible to display 32 special graphic characters in the screen buffer; these can be displayed along with normal text. The following table describes the appearance of these characters.

In Graphics Mode, transmitting the following code ⟨character⟩ . . .	causes a symbol to be stored in the screen buffer, which is called . . .	and which looks like . . .
137 ⟨ _ ⟩	gr(137)	a dot in the middle of the character position
140 ⟨ ‘ ⟩	gr(140)	a symbol for the command delimiter
141 ⟨a⟩	gr(141)	▮
*142 ⟨b⟩	gr(142)	1/
*143 ⟨c⟩	gr(143)	3/
*144 ⟨d⟩	gr(144)	5/
*145 ⟨e⟩	gr(145)	7/
146 ⟨f⟩	gr(146)	° (degrees)
147 ⟨g⟩	gr(147)	±
150 ⟨h⟩	gr(150)	→
151 ⟨i⟩	gr(151)	. . (ellipsis)
152 ⟨j⟩	gr(152)	÷
153 ⟨k⟩	gr(153)	↓
**154 ⟨l⟩	gr(154)	a horizontal bar at the top video scan of the character position
155 ⟨m⟩	gr(155)	a horizontal bar, one scan lower than gr(154)
156 ⟨n⟩	gr(156)	a horizontal bar, one scan lower than gr(155)
157 ⟨o⟩	gr(157)	a horizontal bar, one scan lower than gr(156)
160 ⟨p⟩	gr(160)	a horizontal bar, one scan lower than gr(157)
161 ⟨q⟩	gr(161)	a horizontal bar, one scan lower than gr(160)
162 ⟨r⟩	gr(162)	a horizontal bar, one scan lower than gr(161)
163 ⟨s⟩	gr(163)	a horizontal bar through the lowest scan in the character position
164 ⟨t⟩	gr(164)	subscript 0
165 ⟨u⟩	gr(165)	subscript 1
166 ⟨v⟩	gr(166)	subscript 2
167 ⟨w⟩	gr(167)	subscript 3
170 ⟨x⟩	gr(170)	subscript 4
171 ⟨y⟩	gr(171)	subscript 5
172 ⟨z⟩	gr(172)	subscript 6
173 ⟨{ ⟩	gr(173)	subscript 7
174 ⟨ ¦ ⟩	gr(174)	subscript 8
175 ⟨ } ⟩	gr(175)	subscript 9
176 ⟨~⟩	gr(176)	¶ (paragraph marker)

* These four characters are used preceding the subscripts to form the fractions 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, and 7/8.

**gr(154) through gr(163) can be used to paint a bar graph on the screen which has eight times the vertical accuracy as one produced with only one variety of horizontal bar, since the value of a quantity can be represented not only by the text line on which a horizontal bar occurs, but by how far up in the character position the bar appears. However, there are two video scans between each two text lines where horizontal bars cannot be displayed (four scans on 50 Hz models). This introduces a vertical distortion into graphs formed by the use of the horizontal bars.

Codes not in the range 137-176 are not affected by this conversion to graphic characters. If these other codes are codes are received, they are stored in the screen buffer as usual. This means that, if the terminal is in Graphics Mode, it is not necessary to issue a command to exit Graphics Mode in order to output numerals, upper-case letters, and most symbols to the terminal.

Internally, the VT61 uses the control codes to mark the position of graphic characters. In the screen buffer, the code used to represent gr(x) is x–137. The use of Graphics Mode to place control codes in the screen buffer can cause side effects in some cases, since the VT61 places some of the same control codes in its screen to serve as markers.

1. gr(137) is NUL, a character which the text-editing operations of the VT61 will insert and delete indiscriminately for positioning purposes.

2. gr(150) is TAB and goes into the screen in Block Mode to serve as a marker to the Text Justify operation that following text should be pushed rightward to the next TAB stop.

3. gr(176) is PD and goes into the screen to serve as a marker to the Text Justify operation that following text should be pushed ahead to the next line.

4. gr(140) is CD, the Command Delimiter for Transmit Message operations.

Graphics Mode and Reverse-Video Modes are not mutually exclusive; any graphic character can be displayed in reverse video. This results in the following additional side-effect:

5. A reverse-video gr(137) is the same character that the VT61 uses as its message delimiter for Transmit Message operations.

If Graphics Mode is used to place these characters in the screen, the VT61 will treat them specially in certain situations. For instance, some of these codes, occurring in a portion of the screen that is to be transmitted to the host, will not be represented using the usual format for graphics characters in certain forms of TRANSMIT commands. (For full details, see **OUTPUT TO THE HOST** in Section 4.) Control codes placed in the screen may cause block-transmissions to the host to be ambiguous to the host. Text-editing operations will assign special significance to certain of the graphic characters listed above.

To avoid these side-effects, software in a Full-Duplex situation should prevent the operator from using Graphics Mode to place in the screen codes which could interfere with the other features of the VT61 which are to be used. Please note that in Block Mode, where the operator enters and edits data on the screen without interacting with the host, the terminal prevents the operator from enabling Graphics Mode, ensuring that these side-effects will not occur.

HOLD-SCREEN MODE

entered by ESC [or ESC P U
exited by ESC \ or ESC P u

In Hold-Screen Mode, the terminal tries to avoid scrolling the display until manually directed to by the operator, so that the data on the screen will not be replaced with new data before it has a chance to be read. Since the operator can choose to see a new line of data, or a new page-full, the operator can use Hold-Screen Mode to produce controlled, page-by-page output from any software, written as it is. The host need not know the line-capacity of the terminal nor record how many lines have been output since the last operator request. The terminal makes these calculations and automatically signals the host to suspend transmission when the requested data has been received.

A scroll counter is useful in understanding the mechanics of Hold-Screen Mode. Upon entering Hold-Screen Mode, it is set to zero. When the operator presses the SCROLL key, the scroll counter increases by 1. When the operator presses the SCROLL key with the SHIFT key down, the scroll counter is incremented by 24.

Each time a scroll is performed, the scroll counter will be decremented. If a character is received which would cause a scroll to occur and the scroll counter is 0, then, rather than performing a scroll, that character and all characters following it will be sent to the Silo. As the Silo fills, the terminal will transmit XOFF to request the host to stop sending.

The characters will be processed out of the Silo whenever the operator increases the value of the scroll counter by typing the SCROLL key. If the terminal exhausts the backlog of data in the Silo without obtaining the number of lines requested by the operator, it will transmit XON to the host and process incoming data until it again must hold the information on the screen.

If the Silo overflows because the host did not respond promptly to XOFF, the terminal will perform the scroll it was trying to avoid performing, even though the operator has not typed the SCROLL key. The terminal remains in Hold-Screen Mode, and the scroll counter remains at zero, but the terminal will process characters until another command to scroll is encountered.

Pressing the SCROLL key sends no signal to the host in and of itself.

AUTO-OUTPUT MODES (affects top LED)

The host or operator may command the terminal to output information from its screen to a printer or copier attached to the terminal. These commands are listed in Section 4. In addition to performing output in response to explicit commands, the terminal can be set to regard octal code 012 as an automatic output command. When the terminal is in Auto-Print Mode, receipt of 012 will cause the cursor line to be output to the copier. Simply entering or exiting an Auto-Output Mode does not cause any data to be output.

The control code 012 will also perform its usual function, but *after* the output is made (012 is Line Feed or New Line). Setting the terminal in an Auto-Output Mode, therefore, provides that when the cursor moves downward off a line (by a LINE FEED or NEW LINE command), that line is output. Under some circumstances, which are listed in Section 3, the Line Feed or New Line operation cannot be carried out properly since scrolling is inhibited. However, the command will still cause the cursor line to be output.

If the terminal is in Linear-Addressing Mode, performing a Cursor Right operation with the cursor at the right end of a line causes the cursor to wrap around to the first position in the next line down. An implicit Cursor Right operation, which can also cause wrap-around, is performed in many other cases: after receipt of a displayable character, by the Change Emphasis function and the TAB function, and others, which are described in Section 3. When the terminal is in an Auto-Output Mode, this downward movement of the cursor due to wrap-around will also cause the line the cursor was on to be output.

Unlike the implementation in some VT50-series terminals, other commands which cause downward movement of the cursor (e.g., the Cursor Down command or Direct Cursor Addressing) will not cause output of any line to the copier.

Auto-Output Modes would typically be used to output a variable-length file to the copier or printer. The host would do this by placing the terminal in the correct Auto-Output Mode, transmitting the body of the file to the terminal, and exiting the Auto-Output Mode. A VT61 used conversationally can be left in an Auto-Output Mode to obtain a hard record of all transactions between the host and the operator.

Auto-Print Mode and Auto-Copy Mode are not mutually exclusive; both modes may be in effect at the same time. The uppermost LED on the console is lit when either mode is enabled; it is off when neither mode is in effect.

See the descriptions of the explicit COPY and PRINT commands in Section 4 for information on the format in which the information is output.

AUTO-PRINT MODE

entered by ESC P Y
exited by ESC P y

In Auto-Print Mode, the code 012 becomes an implicit command to output the contents of the cursor line to the printer. After this output is done, 012 will perform its usual function. If a printer is not present, these commands have no effect.

AUTO-COPY MODE

entered by ESC ^
exited by ESC _

Auto-Copy Mode is exactly analogous to Auto-Print Mode, except that the electrolytic copier is the destination of the information.

If a copier is not present on the terminal but a printer is, then commands to enter and exit Auto-Copy Mode will enter and exit Auto-Print Mode instead. If neither copier nor printer is present, these commands have no effect.

The COPY key on the main keyboard, if typed with the SHIFT key down, will affect the Auto-Output Modes. If the terminal was not in Auto-Copy Mode when the shifted COPY key was typed, it will enter Auto-Copy Mode. If the terminal was in Auto-Copy Mode, it will exit the mode. Typing the shifted COPY key never causes anything to be transmitted to the host. If a copier is not present but a printer is, the shifted COPY key will change the state of Auto-Print Mode instead of Auto-Copy Mode. The keyboard is locked during copying; to exit Auto-Copy Mode while the copier is running, it may be necessary to hold down the SHIFT and COPY keys one or two seconds until a click is heard. In Block Mode, the above use of the shifted COPY key has no effect, since the operator of a Block Mode VT61 cannot invoke these mode commands from the terminal.

BLOCK MODE (lights fourth LED from bottom when enabled)

entered by ESC O B
exited by ESC O b

The terminal is normally in a conversational mode of operation. The keyboard functions only to transmit codes to the host. The screen and the command processor respond only to codes from the host.

In Block Mode, data, control codes, and Escape Sequences entered from the keyboard go directly to the screen buffer and command processor as if they had been received from the host. They are not sent to the host. The operator can thus fill the screen with text, edit it in a simulated Off-Line environment, and then transmit it to the host, using the TRANSMIT commands described in Section 4.

Some of the VT61 functions cannot be invoked from the keyboard when the terminal is in Block Mode. They can be invoked only if the proper Escape Sequence is received from the host. The following classes of commands cannot be invoked from the keyboard in Block Mode:

1. Any command which is invoked by an Escape Sequence beginning with "ESC O". In general, these are the commands which deal with aspects of the terminal with which the operator will not normally be concerned.

2. A command to enter or exit any mode (i.e., all the commands discussed in this section), except Alternate-Keypad Mode and Insert Mode.

3. The IDENTIFY TERMINAL TYPE command. This command, discussed in Section 4, is used by the host to determine the type of terminal present.

4. Any command which would scroll text off the top or bottom of the screen. The terminal can be set to alert the host when an upward scroll would have occurred and in certain other situations where the host may want to intervene. This is called Transmit-Request Mode and is the next mode discussed.

Some important implications of these rules are that the operator of a terminal in Block Mode cannot cause the terminal to exit Block Mode; he cannot enter or exit Forms Mode; and he cannot lock his own keyboard.

Note, also, that the operator cannot enter or exit Reverse-Video Mode from the keyboard in Block Mode. However, he can use the Change Emphasis function to convert data in the screen to reverse video if the terminal is not in Forms Mode.

Entering Block Mode makes two other changes to set up the environment for the operator: The terminal enters Linear-Addressing Mode and performs the Insert Paragraph Delimiter function whenever the RETURN key is typed. The operator creating text on the screen in Block Mode will typically type without regard to the capacity of any one line on the screen. Linear-Addressing Mode provides that the cursor will implicitly move from the end of one line to the start of the next line as information is typed in. By this process, words may be split between two lines. The operator can invoke the Text Justify operation after entering all the text. The Text Justify operation will align to a single line the words which crossed line boundaries. (It is described completely in Section 3.)

The only time the operator will type the RETURN key is to tell the terminal explicitly that the text which follows must unconditionally be forced to a new line – in other words, to insert a Paragraph Delimiter. (This function is also defined in Section 3.) The Text Justify operation will force text after a Paragraph Delimiter to a new line, preserving the paragraphs specified by the operator. (Tabular data formatted with the TAB character is also preserved.)

When filling out a form instead of entering text in Block Mode – that is, when the terminal is in Block Mode and Forms Mode together – the RETURN key does not perform the Insert Paragraph Delimiter function. In this case, the RETURN key normally performs the Carriage Return function. The host may force the RETURN key to perform the New Line function by placing the terminal in NL Mode, which is described shortly.

For further operator convenience, typing the DELETE key when the terminal is in Block mode will cause the terminal to execute the CURSOR LEFT command followed by the DELETE CHARACTER command, effectively *rubbing out* the character to the left of the cursor. (In Forms Mode, if that character is protected, it will not be rubbed out, but the leftward movement of the cursor will still occur.) See Section 3 for the specification of these two commands.

When the terminal first enters Block Mode, it *pays attention* to the keyboard, displaying text and processing commands typed in by the operator. All transmissions from the host, however, are ignored, except for the synchronization characters XOFF and XON, and the character STX (octal code 002). When the terminal receives STX, it turns its attention to its receiver to accept a message from the host, and the keyboard becomes inactive. The keyboard will become active again when the host tells the terminal it has completed its message by sending EOT (octal code 004). Thus, the host *must* precede every message to the Block Mode terminal with STX and follow every message by EOT. The host may do this even if the terminal is not in Block Mode, since in that case STX and EOT have no effect at the terminal.

Whenever the host sends STX to the terminal, the terminal exits Reverse-Video Mode and Graphics Mode, so that these modes will be in a known state to process the body of the message. In addition, any pending Escape Sequence is aborted: If STX is embedded in an Escape Sequence, then the characters following STX are interpreted as if ESC had not been sent. Sending a second STX in the middle of a message will have the same effect. The operator enters Escape Sequences directly to the terminal by using the FCN key; if the host sent STX (to begin a message) after the operator has typed the FCN key, but before the operator can type the final character to identify the particular function to be performed, the sequence is aborted, and the FCN key will have to be retyped by the operator after the message from the host ends. In applications where the host may send messages *out of turn* to the terminal, the operator can avoid the possibility of having a keystroke sequence interrupted by the host by using only the functions which are implemented on the auxiliary keypad as single-keystroke functions.

Any commands the host is to give the terminal in conjunction with placing it in Block Mode should *precede* the command to place the terminal in Block Mode. Any commands which follow the command to place the terminal in Block Mode must be sent as formal messages – i.e., preceded by STX and followed by EOT.

The host may place the terminal in Reverse-Video Mode or Graphics Mode by sending the appropriate Escape Sequences to the terminal in its message. If the host places the terminal in either of these modes, it must explicitly take the terminal out of each mode before ending the message, or else when control returns to the typist after receipt of EOT, the terminal will process the keyboard input in Reverse-Video or Graphics Mode. If the host sends STX after a keystroke is detected (at which time the clicking sound is made) but before that character can be processed, the terminal will buffer that character for processing after the message from the host has been processed.

Block Mode is in effect when power is first applied to the terminal, if the FORCE BLOCK MODE switch on the underside of the unit is set. Block Mode can be exited by the appropriate command from the host.

TRANSMIT-REQUEST MODE

entered by ESC O H
exited by ESC O h

Transmit-Request Mode, like all the modes discussed in this section, is normally not in effect when the terminal is first switched on. However, if the FORCE BLOCK MODE switch on the underside of the unit is set so that the terminal will be in Block Mode when it is first turned on, the terminal will also be in Transmit-Request Mode.

In Transmit-Request Mode, every *message* from the terminal to the host is preceded by the character STX (octal code 002) and followed by EOT (004). Any information which is requestable by the host and automatically transmitted to the host by the terminal, is included in the definition of a *message*. Messages can contain information about the contents of the screen, the presence of a printer or copier, or the position of the cursor. Codes generated by the keyboard are not included as messages; information typed by the operator when the terminal is in Conversational Mode is not preceded by STX nor followed by EOT.

The keyboard never responds to typing while a transmission to the host is in progress, so that data from the keyboard will not be embedded in a message. However, in Transmit-Request Mode, the keyboard remains locked after the transmission ends. The host must explicitly unlock the keyboard using the command described in the next mode discussed, Disable-Keyboard Mode.

Also, in Transmit-Request Mode, the terminal sends several Escape Sequences to the host, on an unsolicited basis, in a variety of situations in which the host may want to intervene. They are as follows:

ESC O { When the terminal is first switched on, it will transmit this sequence to the host. (The only condition under which the terminal will be in Transmit-Request Mode when first switched on is if the FORCE BLOCK MODE switch on the underside of the unit is set.) This Escape Sequence is also transmitted to the host whenever a function is invoked which performs the same functions performed on power-up. These functions are Reset the Terminal and Test Terminal, both of which are described in Section 5.

ESC O y A hardware error occurred at the copier during an attempt to output, and the Copy operation was aborted.

ESC O z A hardware error occurred at the printer during an attempt to output, and the Print operation was aborted.

Several additional situations prompt the terminal to send unsolicited messages to the host if the terminal is in Block Mode:

ESC O ¦ When the operator invokes from the keyboard any function that would have performed an upward scroll, the terminal transmits this sequence to the host in lieu of scrolling.

ESC O } When the operator invokes from the keyboard any text-expanding operation and an error occurs because a non-NUL character was shifted off the end of the screen, the terminal transmits this sequence. The character that was lost from the screen is retained in the overflow buffer.

When the operator tries to invoke a COPY or PRINT command from the keyboard, the command is not carried out; instead, the Escape Sequence for that command is sent, as a message, to the host. If the host returns that Escape Sequence to the terminal (*echoes* it), the requested output will occur. When the operator tries to invoke a TRANSMIT command from the keyboard, the specified information is not transmitted to the host; rather, the terminal transmits the Escape Sequence for that TRANSMIT command to the host as a message. The transmission requested by the operator will begin if and when the host reissues that Escape Sequence to the terminal.

Each of these Escape Sequences constitutes a *message*; each is preceded by STX and followed by EOT; and the terminal enters Disable-Keyboard Mode on the completion of each transmission. Whether or not the host takes any action after the terminal has sent it a sequence, it must transmit ESC O e back to the terminal to unlock the keyboard so that the operator may proceed.

DISABLE-KEYBOARD MODE (lights third LED from bottom when enabled)

ESC O E locks the keyboard
ESC O e unlocks the keyboard

Host software uses these two Escape Sequences to explicitly lock and unlock the keyboard. When the keyboard is locked, all activity at the keyboard is ignored except for the BREAK key.

The keyboard is implicitly locked, independently of Disable-Keyboard Mode, in the following situations:

1. In Block Mode, when a message is being received from the host. The keyboard remains locked until the host sends EOT to the terminal.

2. During an automatic transmission from the terminal to the host.

The terminal does not scan the keyboard during the execution of a command. If the command is a lengthy one, such as Text Justify, then the keyboard will appear to lock during the execution of the function. In this case, even the BREAK key will not function until the terminal finishes the command.

The operator is made aware that the keyboard is not being sensed by the absence of the keyclick sound.

ALTERNATE-KEYPAD MODE (complement of Numeric-Keypad Mode, default; lights third LED from top when enabled)

entered by ESC = or ESC P K
exited by ESC > or ESC P k

The auxiliary keypad has 10 keys which transmit codes for the numerals, a key which transmits the code for period or decimal point, and a key, marked ENTER, which transmits CR (015). When the terminal is in Alternate-Keypad Mode and not in Block Mode, each of these keys transmits instead a unique Escape Sequence. The host can use Alternate-Keypad Mode to distinguish between the typing of keys on the auxiliary keypad and the typing of corresponding keys on the main keyboard.

In Block Mode, when the terminal is placed in Alternate-Keypad Mode, the same keys perform common editing functions locally. Therefore, in Editing-Keypad Mode, some of the functions which required typing FCN and a final character can be invoked with a single keystroke. Most of the keys which can transmit these alternate codes have the corresponding function names engraved on them, as well as symbols for the codes they normally transmit.

Appendix A describes in detail the appearance and function of the auxiliary keypad in either mode.

NL MODE (complement of CR Mode, default)

entered by ESC O D
exited by ESC O d

NL Mode exists because different systems use different characters for line delimiters. Most software expects the terminal to send CR (015) as a line delimiter but echoes CR,LF for cosmetic effect. To support this software, the VT61 does the following in CR Mode:

1. Transmits CR when the RETURN key is typed.
2. Transmits LF when the LINE FEED key is typed.
3. Transmits CR to mark the end of a screen line during a block transmission.
4. On receiving LF, moves the cursor down one line (scrolling if necessary), keeping the cursor in the same column it was in.

The newest (1974) ASCII proposal on the subject renames 012 "NL" (New Line) and suggests that it alone be used in both input and output as a line delimiter. The VT61 can become compatible with systems adhering to this convention, if it is placed in NL Mode – so that it:

1. Transmits NL when the RETURN key is typed.
2. Transmits CR when the LINE FEED key is typed.
3. Transmits NL to mark the end of a screen line during a block transmission.
4. Functions so that NL alone is sufficient to move the cursor to the beginning of a new line on the screen (scrolling if necessary).

ALARM MODE

entered by ESC O G
exited by ESC O g

In Alarm Mode, the entire screen oscillates between normal and reverse video in the same manner as the cursor does normally. The frequency of this oscillation is approximately 8 Hz. When Alarm Mode is exited, the screen ceases to oscillate except at the cursor position.

PRINTER-CONTROLLER MODE

entered by ESC W
exited by ESC X

Printer-Controller Mode gives host software complete control over the printer attached to the VT61 through its General Output Register.

In Printer-Controller Mode, all displayable characters received from the host are sent directly to the printer and do not appear on the screen. Control codes from the host, except XOFF and XON, are sent to the printer as they were received and are not processed by the terminal in the normal way. Escape Sequences are also passed through to the printer. ESC W, which caused the terminal to enter Printer-Controller Mode, does not get sent to the printer. ESC X is the only Escape Sequence to which the VT61 will react in a special way if it is in Printer-Controller Mode. The VT61 will exit Printer-Controller Mode. The ESC in the sequence ESC X will have been output to the printer already; the VT61 outputs CAN (octal code 030) to the printer instead of the final X, since ESC CAN will not cause the printer to do anything.

Using the regular PRINT commands specified in Section 4, the terminal outputs data from the screen to the printer, taking certain steps to ensure that the text will look the same as printer output as it did when it was on the screen. In Printer-Controller Mode, the terminal simply passes characters through from host to printer, performing no code-conversion or insertion of line delimiters. Software can use Printer-Controller Mode to cause characters to appear on all 132 columns of the printer despite the size limitations of the screen buffer. The VT61's function is essentially that of a remote printer-controller.

Printer-Controller Mode does not disable the keyboard. Communication can take place from operator to host at the same time the host is communicating to the printer.

MAINTENANCE MODE

entered by ESC O A
exited by ESC O a

In Maintenance Mode, any time any key is typed (except the SHIFT, CONTROL, and CAPS LOCK keys), the terminal transmits the keyboard address of that key to the host instead of the ASCII code for that character, or whatever control code or Escape Sequence that key would normally have transmitted.

In addition, all messages to the hosts are preceded by the control code STX (002) and terminated by EOT (004), and the terminal locks the keyboard after transmission of a message, as in Transmit-Request Mode. Remember, a message is a transmission of data or status to the host on command; the keyboard address which is sent to the host when a key is typed does not constitute a message.

VT61s may be offered on which the keyboards function differently from the VT61 described in this document. A diagnostic program will be able to place any such terminal in Maintenance Mode so that the keyboards can be tested compatibly.

SECTION 3

TEXT EDITING AND CURSOR CONTROL

These commands are used to change the position of the cursor and modify the text appearing on the screen. The commands take four forms:

1. Escape Sequences which the operator will probably want to invoke from the keyboard (FCN stands for ESC P since the FCN key on the auxiliary keypad transmits ESC P).	FCN C or FCN c	write the ruler
	FCN D or FCN d	delete line and ripple up
	FCN E or FCN e	change emphasis
	FCN F or FCN f	insert line and ripple down
	FCN Q or FCN q	cursor home
	FCN R or FCN r	clear and justify
	FCN V or FCN v	text justify
	FCN W or FCN w	cursor to end of text
	FCN X or FCN x	erase to end of line
	FCN Z or FCN z	erase to end of screen
2. Escape Sequences which will usually originate only from the host.	ESC O N	delete line and ripple down
	ESC O O	insert line and ripple up
3. 2-character Escape Sequences compatible with 50-series terminals. To invoke one of these functions not found in Group 1 above, the operator types the appropriately labeled key on the keypad.	ESC A	cursor up
	ESC B	cursor down
	ESC C	cursor right
	ESC D	cursor left
	ESC H	cursor home
	ESC I	reverse line feed (not found on keypad)
	ESC J	erase to end of screen
	ESC K	erase to end of line
	ESC R	forward field
	ESC Q	backward field
4. Control codes.	(010)	backspace (cursor left)
	(011)	tab
	(012)	line feed or new line
	(015)	carriage return

THE ELEMENTARY CURSOR MOVEMENT FUNCTIONS

CURSOR UP	invoked by ESC A
CURSOR DOWN	invoked by ESC B
CURSOR RIGHT	invoked by ESC C
CURSOR LEFT	invoked by ESC D or BS (010)

The four basic cursor movement functions allow the operator or host to maneuver the cursor to any position on the screen. The operator characteristically uses these commands to move the cursor to a point on the screen he has located by sight. The commands cause the cursor to move one character position in the specified direction.

If the cursor is on the top line, a CURSOR UP command will not move the cursor. If the cursor is on the bottom line, a CURSOR DOWN command will not move the cursor. None of the cursor commands ever cause a scroll to be performed.

If the cursor is at the rightmost position in a line, a CURSOR RIGHT command will not move the cursor if the terminal is in Rectangular Mode. If the terminal is in Linear Mode, the cursor will move to the leftmost position on the next line down (unless it was on the bottom line to begin with).

If the cursor is at the leftmost position in a line, a CURSOR LEFT command will not move the cursor if the terminal is in Rectangular Mode. If the terminal is in Linear Mode, the cursor will move to the rightmost position on the next line up (unless it was on the bottom line to begin with).

An implicit CURSOR RIGHT command is performed after a displayable character is stored in the screen buffer. However, if the character was stored in the last column on the bottom line, a scroll will be performed by this implicit cursor movement, and the cursor will go to the rightmost position on this new line (unless the terminal is in Rectangular-Addressing Mode).

THE BASIC LINE-ORIENTED COMMANDS

Two keys on the main keyboard are used to invoke the functions described below, which are usually thought of as *line-delimiting*. There is a double-sized key labeled RETURN and a regular-sized key labeled LINE FEED. The terminal is designed so that the operator will normally use the RETURN key. This is achieved by a set of rules which describe the function of these keys in different settings. The rules are as follows:

1. In Conversational Mode (the opposite of Block Mode), the RETURN key is used for the Carriage Return function. The LINE FEED key is used for the Line Feed function. (Remember that in Full Duplex the keys only transmit codes to the host. The host must echo the codes in order for the function to take effect.)

2. The terminal may be placed in NL Mode in which the RETURN key transmits the correct code for the New Line function. (In NL Mode, the terminal will interpret this code as a NEW LINE command rather than a LINE FEED command.) The LINE FEED key is used for the Carriage Return function. It is still true that the function is not performed unless the code is transmitted to the terminal by the host.

3. If Block Mode is on and Forms Mode is off so that the operator is entering text into the screen in an Off-Line environment, the RETURN key invokes the Insert Paragraph Delimiter function. (The description of Block Mode in Section 2 explains the reasons for this rule; the Insert Paragraph Delimiter function is defined later in this section.) The LINE FEED key does whatever it would have done normally, depending on whether the terminal is in NL Mode.

CARRIAGE RETURN

invoked by CR (015)

The control code CR causes the cursor to be moved to the leftmost position on the line it is on. If the cursor is already at the leftmost position on a line, no action nor error occurs.

In Forms Mode, the cursor is moved to the first unprotected character in the line it is on. If that line contains nothing but protected characters, the cursor goes to the leftmost position on that line.

LINE FEED

invoked by 012 when the terminal is *not* in NL Mode
(in CR Mode)

The cursor is moved down one line. If the cursor was on the bottom line to begin with, an upward scroll of all data on the screen is performed instead. The cursor remains in the same column that it was in.

In Forms Mode, a scroll is never performed. If the cursor was on the bottom line, it stays where it was.

In Block Mode, a keyboard command cannot cause scrolling to occur. If the operator invokes this function such that a scroll would occur, the bell rings and the cursor stays where it was. If the terminal is in Transmit-Request Mode, an Escape Sequence will be sent to the host and the keyboard will lock. The host may send 012 to the terminal to scroll the display, provided the terminal is not in Forms Mode.

NEW LINE

invoked by 012 when the terminal is in NL Mode

The cursor is moved to the leftmost position on the next line down. If the cursor is on the bottom line and moving it thus would send it off the logical bottom of the screen, a scroll up is performed first.

This function combines the Line Feed and Carriage Return functions, performing them in that order. The exceptions for Block and Forms Modes which are found in the descriptions of Line Feed and Carriage Return are preserved. In Forms Mode, this function will position the cursor at the first unprotected character on the next line down.

REVERSE LINE FEED

invoked by ESC I

The cursor is moved up one line. If the cursor was on the top line to begin with, a downward scroll of all the data on the screen is performed instead. The cursor remains in the same column that it was in.

In Forms Mode, this function has no effect.

In Block Mode, this function cannot be invoked from the keyboard.

This function is comparable to Line Feed, except that its function is not modified by placing the terminal in NL Mode. Reverse Line Feed is added for compatibility with the VT52.

TAB (affects overflow buffer)

invoked by TAB (011)

Columns 8, 16, 24, 32, 40, 48, 56, 64, 72, and 0 of a line (where the columns are numbered 0 to 79) are called *TAB stops.*

When the VT61 receives the control code TAB, it moves the cursor right at least one position until it is at a TAB stop. If the cursor is in or to the right of column 72, the cursor will move to the right one column only. If the cursor is in the rightmost column on a line, it will stay there if the terminal is in Rectangular-Addressing Mode; if the terminal is in Linear-Addressing Mode, the cursor will wrap around to the leftmost column of the next line down. An upward scroll of the display will occur if the cursor was in the lower right corner of the screen with the terminal in Linear-Addressing Mode.

In Block Mode, the control code TAB is placed at the cursor position. TAB is the graphic character gr(150); it is displayed on the screen as an arrow pointing rightward if the DISPLAY ALL switch is set; otherwise, it is not displayed on the screen. The cursor is moved right at least one position until it is in a TAB stop. If there are any positions between the TAB character and the new cursor position (if the cursor was moved rightward at least two positions), then NULs are inserted there.

The cursor is never wrapped around to the next line down if the terminal is in Rectangular-Addressing Mode. If the cursor was at or to the right of column 72 on a line, it moves to column 79.

Normally, TAB is a cursor movement command only. In Block Mode, TAB is thought of as a character as well as a command. If the terminal is not in Insert Mode, TAB will cause all characters which were to the right of where the TAB code was stored in the screen to be pushed ahead to the next TAB stop. If the terminal is in Insert Mode, the character which occupied the position that the TAB code now occupies will also be pushed forward. After this command is processed, the nearest non-NUL character to the right of the TAB marker will be at a TAB stop. TAB in Block Mode can be used by the operator to arrange in tabular form data which is already on the screen. Additional editing of the screen may disturb the tabular form, but the terminal's JUSTIFY commands can be used to re-align the text. These commands use the TAB markers in the screen for reference.

The Block Mode TAB function can cause a scroll to occur if the cursor was in the lower right corner of the screen. Remember that in Block Mode scrolling from the keyboard is inhibited. If the terminal is in Transmit-Request Mode, a *request* will be sent to the host by the terminal in lieu of scrolling.

An error occurs if the insertion of either TAB or NULs causes a non-NUL character to be shifted out of the screen. This character moves to the overflow buffer where it can be read by the host. In Transmit-Request Mode, the host is notified of this error condition.

The overflow buffer is cleared by the Block Mode TAB function if an error did not occur. The overflow buffer is not affected by the TAB function if the terminal is not in Block Mode.

The TAB character occuring in the screen is used by the Text Justify and Clear and Justify operations as a justification marker. Receipt of TAB itself places the TAB character in the screen when the terminal is in Block Mode. When the terminal is not in Block Mode, Graphics Mode can be used to place the TAB code in the screen. Since the graphic character gr(150) is TAB, software may place the terminal in Graphics Mode and transmit the character "h" (octal code 150) to the terminal in order to store a TAB code in the screen. This will not cause an immediate forward shifting of text as the Block Mode TAB command does, however.

In Forms Mode, TAB is the same as the Forward Field function.

INSERT PARAGRAPH DELIMITER (affects overflow buffer)

invoked by ESC P A or ESC P a

The Block Mode form of the TAB command described above causes movement of following text to the next TAB stop on two occasions: at the time the command (TAB) is received and in a Text Justify operation.

There is a corresponding marker which forces all text to the right of it to the beginning of the next line. This marker is the Paragraph Delimiter (PD). It is normally not seen on the screen but will be displayed as a paragraph sign if the DISPLAY ALL switch is set. A special graphic character, gr(176), is used in the screen memory to mark the position of the Paragraph Delimiter. In Block Mode, the TAB marker was placed in the screen by sending the terminal the code for TAB itself. To place PD in the screen, the host sends the codes for the Insert Paragraph Delimiter function, an Escape Sequence.

When this Escape Sequence is received, PD is inserted in the screen at the cursor position – as if the terminal were in Insert Mode, even if it is not. Then NULs are inserted to move down any text that was to the right of PD and on the same line so that the first non-NUL character in that text appears at the start of the next line. The cursor will move to the right end of the line it was on if the terminal is in Rectangular-Addressing Mode; it will wrap around to the start of the next line down if the terminal is in Linear-Addressing Mode, performing an upward scroll if it was on the bottom line.

In Block Mode, the INSERT PARAGRAPH DELIMITER command works the same way, but, unless the terminal is also in Forms Mode, the operator can invoke this function by typing the RETURN key.

If the cursor was at the rightmost position on a line at the start of this operation, PD will be inserted, and the operation will end without inserting NULs. If the terminal is in Linear Mode, the cursor will wrap around to the start of the next line; if the terminal is in Rectangular Mode, it will not move. If the cursor was at the lower right corner of the screen, inserting the PD in Linear-Addressing Mode will cause an upward scroll of the display.

An error occurs if the insertion of either PD or NULs causes a non-NUL character to be shifted out of the screen. This character moves to the overflow buffer where it can be read by the host.

The overflow buffer is cleared by this operation if an error did not occur.

In Forms Mode, this function has no effect.

FORWARD FIELD

invoked by ESC R
invoked by TAB if the terminal is in Forms Mode

When the terminal is not in Forms Mode, this function has no meaning. ESC R has no effect at the terminal; TAB invokes the TAB function.

In Forms Mode, the cursor is moved to the beginning of the next field (to the first unprotected character to the right of the first protected character at or to the right of the cursor position). If the cursor is in or past the last (rightmost) field on the screen, it moves to the end of the bottom line of the screen.

BACKWARD FIELD

invoked by ESC Q

This function has no effect unless the terminal is in Forms Mode.

In Forms Mode, the cursor is moved to the beginning of the previous field (to the leftmost consecutive unprotected character to the left of the first protected character at or to the left of the cursor position). If the cursor is in or to the left of the first field on the screen, it moves to the Home position – the beginning of the top line.

ERASE TO END-OF-LINE

invoked by ESC K or ESC P x or ESC P X

All the information at the cursor position and rightward to the end of the line is erased (NULs are deposited at those character positions).

If the cursor is at the rightmost column on a line, the character at the cursor position will be the only character to be erased. If the cursor is at the leftmost column on a line, the entire line will be erased.

In Forms Mode, this function becomes Erase to End-of-Field. NULs are stored at the cursor position and rightward up to and including the last position in the current field.

ERASE TO END-OF-SCREEN

invoked by ESC J or ESC P z or ESC P Z

All the information from the cursor position to the end of the screen is erased (NULs are deposited at those character positions). This function does what Erase to End-of-Line does and, in addition, erases the information in every line below the line the cursor is on.

If the cursor is at the lower right corner of the screen, one character will be erased. If the cursor is at the Home position of the screen, all the information on the screen will be erased.

In Forms Mode, NULs are stored only in unprotected positions at and to the right of the cursor. Protected characters are not disturbed.

CURSOR TO HOME

invoked by ESC H or ESC P Q or ESC P q

The cursor is moved to the Home position — the character position at the upper left corner of the screen. If it was there to begin with, it stays there.

In Forms Mode, the cursor is moved to the first position containing an unprotected character, i.e., to the start of the form. If the entire screen is protected, the cursor is moved to the Home position.

CURSOR TO END-OF-TEXT

invoked by ESC P W or ESC P w

The cursor is moved to the character position directly to the right of the last non-NUL character in the screen. If the character at the lower right corner of the screen is non-NUL, so that there is no position to the right of it, the cursor is placed over that character. If the entire screen contains NULs, the cursor will go Home.

The operator might use this function while entering text into the screen. If he notices an error made a few lines earlier, he stops typing in text and maneuvers the cursor over the old error and corrects it. Now, by invoking the Cursor to End-of-Text function, the cursor can be moved to where it was when he stopped typing in text.

The host might use this function followed by the function Request Cursor Position (described in Section 4) to determine the relative fullness of the screen with text.

The functions Text Justify and Clear and Justify invoke this function to reposition the cursor after the justification phase is complete.

DELETE CHARACTER

invoked by ESC P S or ESC P s

If the character at the cursor position is not NUL, move characters left one position beginning with the character to the right of the cursor position and continuing rightward until a NUL is moved.

If the character at the cursor position is NUL, the terminal searches rightward for non-NUL characters. The operation described above is performed starting with the last consecutive NUL to the right of the cursor. If there are no non-NUL characters to the right of the cursor position, no action is taken.

If the character in the last position in the screen buffer (or in Forms Mode the last position in the current field) is moved, NUL is deposited in the last position and the operation ends.

THE LINE MOVEMENT FUNCTIONS

These four functions allow line-by-line movement of text on the screen. Functions are provided to insert a line in the middle of the text, moving neighboring lines outward to make room, and to delete a line of text from the screen, moving neighboring lines to take up the free space. The VT61 user has a further choice concerning in which direction – upward or downward – this compression or expansion will cause lines to move. A total of four functions are provided. Two of them feature special keyboard-labeling since the operator will want to invoke them from the keyboard using the FCN key. These two cause movement of the lines below the cursor and are suited for creation of text on the screen with the terminal in Block Mode, since all the unused space on the screen will typically be below the cursor. Two alternate functions are provided which move the lines above the cursor. The host can invoke these functions to scroll the text on the screen downward, or to maintain two separate activities on the screen using and scrolling the upper and lower half independently.

As a typical application, assume that the host is going to maintain two separate displays on the screen with the break between the top and bottom displays occurring between lines 14 and 15. For instance, one area might be used for instructions to the operator or for a dynamic reference table describing how to use the program. The other area might be a conversational area in which the operator can see what he has typed in and the computer's immediate response to it. To perform an upward scroll of the information on the top half of the screen, the host moves the cursor to line 14 and performs an INSERT LINE AND RIPPLE UP command. To perform an upward scroll of the information on the bottom half of the screen, the host moves the cursor to line 15 and issues a DELETE LINE AND RIPPLE UP command. Thus, both halves appear to scroll independently of each other.

DELETE LINE AND RIPPLE UP

invoked by ESC P D or ESC P d

The line containing the cursor is deleted. All lines in the buffer below the cursor line move up one line, and NULs are deposited at every position in the bottom line. The cursor is not moved.

In Forms Mode, this function has no effect.

DELETE LINE AND RIPPLE DOWN

invoked by ESC O N

The line containing the cursor is deleted. All lines in the buffer above the cursor line move down one line, and NULs are deposited at every position in the top line. The cursor is not moved.

In Forms Mode, this function has no effect.

INSERT LINE AND RIPPLE DOWN

invoked by ESC P F or ESC P f

The line containing the cursor and all lines below it are shifted down one line. The line now containing the cursor is filled with NULs, and the cursor is moved to the leftmost position in that line.

In Forms Mode, this function has no effect.

INSERT LINE AND RIPPLE UP

invoked by ESC O O

The line containing the cursor and all lines above it are shifted up one line. The line now containing the cursor is then filled with NULs, and the cursor is moved to the leftmost position in that line.

In Forms Mode, this function has no effect.

If the cursor is on the bottom line, this function scrolls up the entire screen – just as the Line Feed function does.

TEXT JUSTIFY (affects overflow buffer)

invoked by ESC P V or ESC P v

Editing text on the screen, and the character-rippling caused in Insert Mode and by the Delete Character function, can have effects such as: blank space in the middle of text, tabular data (using the character TAB) having crooked columns, and words moving across physical line boundaries. The Text Justify function arranges text on the screen into an aesthetic, left-justified form.

Using Text Justify, the operator can type in text without having to consider its position on the line or type line delimiters when the text approaches the right margin. On command, the Text Justify function moves to the next line any word which is split between two lines, and shifts other text rightward to accommodate this change.

Text Justify can be divided into two phases: compressing the text and re-expanding it. To compress the text, NULs embedded within the text are removed and reinserted following the text, in order to arrange the buffer so that all non-NUL data appears first (toward the top of the screen) with no embedded NULs, and NULs appear thereafter (below the non-NUL text).

In the expansion stage, NULs are inserted (as if the terminal were in Insert Mode), from the beginning of the screen buffer forward, in the following three situations to position the text ahead:

1. If the character TAB is encountered, NULs are inserted to the right of the TAB until the character following the NULs is at a TAB stop. (If it was at a TAB stop to begin with, no NULs are inserted.)
2. If a Paragraph Delimiter (PD) is encountered, NULs are inserted to the right of the PD until the character following the NULs is at the leftmost position on a line.
3. If a word part of which is in the rightmost column on a line is encountered, NULs are inserted at the beginning of the word until the first character in the word is in the first column of a new line. (No attempt is ever made to align a word of more than 79 characters.)
4. If a word ends in the rightmost column of a line so that the space between it and the next word is at the beginning of the next line, that word is likewise moved to the lower line so the left margin will not be ragged.

With control characters and spaces (and the beginning and end of the screen buffer) qualifying as word separators, a word is defined as any string of non-separators (codes 041-176) between two separators.

If the insertion of NULs should at any time cause a non-NUL character to go off the end of the screen, the Text Justify operation ends immediately, and that character is placed into the overflow buffer. Conversely, if the available space on the screen was sufficient to accommodate the text fully re-expanded, the overflow buffer will be empty at the end of the Text Justify operation.

The Text Justify function invokes the Cursor to End-of-Text function, described earlier in this section, which moves the cursor past the last non-NUL character in the screen. Therefore, if the operator was entering text into the screen and invoked the Text Justify function, the cursor will be returned to the end of the text – even though the end of the text may now be at a different position in the screen – so that typing can continue.

In Forms Mode, this function has no effect.

CLEAR AND JUSTIFY (affects overflow buffer)

invoked by ESC P R or ESC P r

NULs are deposited in every character position from the beginning of the screen up to and including the cursor position. Then the Text Justify function is performed.

As a result of this operation, the first non-NUL character ahead of the cursor will be shifted up to the Home position (the start of the top line). Neighboring characters will be likewise moved in accordance with the Text Justify rules described above.

The operator would use this function in a situation in which the screen is filling up with data which he is entering. He moves the cursor to a breakpoint in the text, and types the Transmit Message function (described in Section 4) in order to transmit all data from Home to the cursor position. Now the Clear and Justify function erases from the screen all the data that has just been transmitted to the host and re-justifies the text that remains on the screen. Thus more room on the screen is made available for further text entry.

As in the Text Justify function, the cursor is moved past the last non-NUL character in the screen by the Cursor to End-of-Text function whenever the Clear and Justify function is invoked.

An error can occur as in Text Justify. Even though this operation has the effect of deleting information from the screen, there may not be enough room in the screen in which to fit the *formatted* text that the Text Justify operation will produce. The character which was shifted out of the screen can be found in the overflow buffer; as usual, a NUL will be found there if the justification succeeded.

In Forms Mode, this function has no effect.

WRITE THE RULER

invoked by ESC P c or ESC P C

The operator can use the ruler to easily determine the horizontal position of the cursor or text. The ruler is an 80-character string consisting of "1234567890" repeated eight times, with every other repetition in reverse video. (The first repetition is in normal video.) Write the Ruler causes this ruler to be written on the line containing the cursor. The old contents of that line are replaced by the ruler.

This function is disabled in Forms Mode.

CHANGE EMPHASIS

invoked by ESC P E or ESC P e

The character at the cursor position is modified so that it will be displayed in reverse video if it was in normal video before the command was issued. If it used to be in reverse video, it is set to be displayed in normal video. (If the character at the cursor position is a control code (000-037), this part of the function is not performed.)

The cursor is then moved to the next character position, using the procedure described under Cursor Right.

In Forms Mode, this function has no effect since it would allow the form portion of the screen to be altered.

SECTION 4
OUTPUT

This section describes the interaction of the VT61 with its peripherals, the printer and copier, and with the host. Supplementary information is found in the general discussion in Section 1 (which should be read first) and the description of Block Mode and Transmit-Request Mode in Section 2.

The commands which moderate the interaction between the VT61, the hardcopy units, and the host take the following forms:

SIGNALS RECEIVED BY THE VT61 FROM THE HOST

1. Escape Sequences which the operator will probably want to invoke from the keyboard (FCN stands for ESC P since the FCN key on the auxiliary keypad transmits ESC P).

FCN B or FCN b	transmit cursor line
FCN G or FCN g	copy cursor line
FCN H or FCN h	print screen
FCN J or FCN j	print cursor line
FCN M or FCN m	transmit message
FCN N or FCN n	transmit unprotected

NOTE: In Block Mode and Transmit-Request Mode, typing these sequences will cause the terminal to send to the host a request to perform the specified output. These requests are listed in Group 4b below.

FCN T or FCN t	insert command delimiter

2. Other Escape Sequences, to be sent to the VT61 by the host.

ESC O P	set start of selected area
ESC O Q	set end of selected area
ESC O S	transmit selected area
ESC O V	transmit all
ESC O W	transmit cursor character
ESC O X	transmit overflow buffer
ESC O Y	transmit cursor position
ESC O Z	get the line (see Section 1, HALF DUPLEX)
ESC O [	clear receiver checksum
ESC O \	clear transmitter checksum
ESC O]	transmit receiver checksum
ESC O ^	transmit transmitter checksum
ESC O _	transmit abort flag
ESC O ‘	initialize abort flag
ESC]	copy screen

NOTE: The operator invokes the Copy Screen function by typing the COPY key on the main keyboard. This invokes the function *locally* – without sending any codes to the host – except when the terminal is in Block Mode and Transmit-Request Mode.

ESC Y (A) (B)	move the cursor to line (A)-040, column (B)-040
ESC Z	identify terminal type

3. Control codes.

(023)	Stop sending data!
(021)	Resume sending data.

SIGNALS TRANSMITTED BY THE VT61 TO THE HOST

The following is a summary of all the signals which host software can receive from the VT61 automatically (i.e., in situations where the terminal operator did not actually type the codes). Most of these signals are responses to commands by the host for the VT61 to transmit to it the contents of a portion of the screen.

4. Escape Sequences sent to the host on command.

ESC P A	A Paragraph Delimiter occurred in the screen at this position.
ESC O J	The following characters occurred in the screen in reverse video.
ESC O j	The following characters occurred in the screen in normal video.
ESC O x	No output abort has occurred since the Abort Flag was last initialized.
ESC O y	The most recent output error was at the copier.
ESC O z	The most recent output error was at the printer.
ESC F	The following string represents graphic characters which occurred in the screen.
ESC G	End of graphic string.
ESC ? s	The following message is a command. (The Transmit Message function uses this sequence when it encounters a Command Delimiter as the start of the message.)
ESC P ⟨n⟩, where ⟨n⟩ is a numeral (0-9)	In Block Mode, transmitted to indicate that the operator typed FCN ⟨n⟩.
ESC Y (A) (B)	The cursor is at line (A)-040, column (B)-040.
ESC / (C)	I am a type (C) terminal.

4b. Escape Sequences spontaneously transmitted by the terminal in Transmit-Request Mode to tell the host of a situation which may require intervention (see Section 2, TRANSMIT-REQUEST MODE, for details).

ESC O y	A hardware error has occurred at the copier.
ESC O z	A hardware error has occurred at the printer.
ESC O {	The terminal has been switched on.
ESC O ¦	An upward scroll has been requested by the operator.
ESC O }	A text-expanding operation has shifted a character off the screen into the overflow buffer.
ESC P B ESC P G ESC P H ESC P J ESC P M ESC P N ESC _	The operator has tried to invoke this output function from the keyboard. (If the host sends this Escape Sequence back to the terminal as a message, the requested output will occur.)

5. Control codes.

(002)	Start of message.
(004)	End of message.
(011)	A TAB occurred in the screen at this position. Field Format transmissions: The preceding characters occurred in the same field on the screen.
(012) [when terminal is in NL Mode] (015) [when terminal is not in NL Mode]	Line Format transmissions: The preceding characters occurred on the same physical line on the screen. Field Format transmissions: The preceding characters occurred on the same physical line of a multi-line field. Character Format transmissions: (012) and (015) are never transmitted.
(023)	Stop sending data!
(021)	Resume sending data.
(000)	A NUL occurred in the screen at this position. (Character Format only)
(177)	A character which had been received with bad parity occurred in the screen at this position. (Preceded by ESC O J)

COPY SCREEN

invoked by ESC]

The contents of the screen are printed on the printer.

COPY LINE

invoked by ESC V or ESC P G or ESC P g

The contents of the line containing the cursor are printed on the printer.

The operator can invoke Copy Screen by typing the COPY key, with one exception: If the terminal is in Block Mode and Transmit-Request Mode, a *request* to copy the screen will be sent to the host. This request will be the Escape Sequence ESC] itself; if the host sends it back to the terminal as a message, the function will be performed. In all other situations, when a request need not be sent to the host, the COPY key invokes the Copy Screen function without sending any codes to the host.

If the SHIFT key is held down when the COPY key is typed, the terminal interprets the action as a different command — a command to change the state of Auto-Copy (or Auto-Print) Mode. This operator command is described with the discussion of Auto-Output Modes in Section 2.

PRINTER OUTPUT

When a line is output to the General Output Register (the printer), codes are output exactly as they appeared in the screen, except that codes 000 (NUL) and 011 (TAB) are output as spaces. A CR and LF follow the last non-NUL character in the line; trailing NULs which may occur in the line are not output to the printer.

If a printer is not present, these operations do not function. Data is never sent to the copier instead of the printer.

Recall that another way to use the printer is by placing the VT61 in Printer-Controller Mode. In this mode, the conversions described above are not performed. Text received by the VT61 is sent directly to the printer. Host software can use Printer-Controller Mode rather than the explicit PRINT commands described below to output to all 132 columns of the printer.

PRINT LINE

invoked by ESC P J or ESC P j

The line containing the cursor is output to the printer.

PRINT SCREEN

invoked by ESC P H or ESC P h

The contents of the entire screen buffer are output to the printer as 24 lines.

OUTPUT TO THE HOST

The host can request the terminal to transmit information on the contents of all or part of the screen, the position of the cursor, the overflow buffer, and the type of terminal it is in contact with.

When information from the screen is being transmitted to the host, any string of one or more characters which were displayed in reverse video is preceded by ESC O J and followed by ESC O j. To illustrate: If software wishes to record which video setting each character was displayed in, it should keep a logical flag. Receipt of ESC O J should cause this flag to be set to a state standing for "Reverse Video"; ESC O j should set it to "Normal". Then each character received appeared in normal or reverse video, depending on the state the logical flag was in when that character was received. Note that, at the beginning of any transmission, the logical flag *must* be set to "Normal" since, if there are no reverse-video characters in the transmitted text, niether Escape Sequence will be sent. Note also that ESC O J and ESC O j are the same Escape Sequence which set the terminal's analogous logical flag, i.e., enter and exit Reverse-Video Mode.

In a similar manner, any contiguous string of graphic characters (except gr(137), gr(15), and gr(176), which are NUL, TAB and PD) will be preceded by ESC F and followed by ESC G. The graphic characters in this string will be denoted in the transmission using the same codes the software would transmit to store those graphic characters in the screen. For example, the graphic character gr(165) occurring in the screen between two non-graphic characters (call them x and y) would be encoded as:

x ESC F ⟨165⟩ ESC G y

The four Escape Sequences ESC F, ESC G, ESC O J and ESC O j are sent only to mark transitions between normal and reverse video, and graphic and normal encoding. If the first character in the text of a message is in reverse video, ESC O J is sent before that character; if it is in normal video, no Escape Sequence precedes it. If the first character is a graphic character, ESC F precedes it; if it is in the range 040-176, no Escape Sequence precedes it. No Escape Sequence will follow the last data character in a message because there is no transition to mark.

PDs occurring in the screen are not transmitted to the host literally; instead, an Escape Sequence (ESC P A) is transmitted to mark the position of the PD relative to the other data.

TAB characters contained in the screen buffer are transmitted exactly as they appear.

Recall that the VT61 stores a marker in the screen if it receives a character with bad parity. This marker will be stored at the position the cursor was in at the time the bad character was received. If the terminal is ordered to transmit a portion of the screen which contains this marker, it will transmit DEL (177) to represent the marker. The terminal will treat this marker as a reverse-video character; ESC O J will precede it if necessary.

NULs, TABs and PDs are not preceded in a transmission by either ESC F or ESC G. Since Graphics Mode can be used to enter these characters in the screen in reverse-video, ESC O J and ESC O j may precede the representations of these characters if necessary to indicate in which video type they appeared .

While the transmission is in progress, the cursor traces the portion of the screen being transmitted. At the end of the transmission, the cursor returns to its former position, and the bell sounds. The keyboard is not scanned during a transmission to the host.

CROSS-REFERENCES CONCERNING OUTPUT TO THE HOST

NOTE 1: The host may use the control codes XOFF and XON to moderate the transmission from terminal to host. Review FULL DUPLEX SYNCHRONIZATION in Section 1 for details of this feature.

NOTE 2: When the terminal is set for Half Duplex operation, the electrical signals CD, SRD, STD, RTS and CTS control the communication between terminal and host. This is discussed in HALF DUPLEX in Section 1.

NOTE 3: By placing the terminal in Transmit-Request Mode, the host may specify that all messages to the host are preceded by the control code STX (002) and followed by EOT (004), and that the keyboard remains locked after the transmission is over. See TRANSMIT-REQUEST MODE in Section 2 for details on this aspect of transmissions to the host.

FORMATS FOR CONVEYING POSITIONAL INFORMATION

In addition to this protocol for identifying characters that appeared in reverse video and encoding graphic characters, the VT61 also provides several different ways in which the host can determine where on the screen the data appeared.

Some of the VT61's TRANSMIT commands (Transmit Overflow Buffer, Transmit Cursor Character, Transmit All, Transmit Selected Area, and Transmit Cursor Line when the terminal is in Forms Mode) cause transmissions to be made in Character Format. In Character Format, something is transmitted to stand for every character position in the area transmitted, even if that character position contained nothing (a NUL). In that case, NUL will be transmitted to mark that character position. An *erased* line (containing nothing but NULs) would be portrayed as follows if one of these Transmit functions were used:

NUL NUL NUL NUL . . . NUL NUL NUL NUL

for a total of 80 characters. Remember that a *space* is different from a NUL. A space has octal code 040; a NUL has code 000. A space is deposited in the screen only when the terminal receives the code 040 or the operator types the Space bar. NULs are added, deleted, and moved indiscriminately by the terminal to position other data; they can clearly be thought of as "what's in the screen when nothing (else) is".

Character Format allows the computer to obtain a complete specification of the contents of the screen. This would be useful in the case of an operator using the screen to create a form which will be filled in many times in the future with the terminal in Forms Mode. The host needs to know the exact position in the screen of every character so that it can retransmit the form properly. For an example of a transmission using Character Format, see the description of TRANSMIT OVERFLOW BUFFER in this Section.

Other TRANSMIT commands (Transmit Data, Transmit Message, and Transmit Cursor Line when the terminal is not in Forms Mode) do not include NULs in their transmission to the host, but simply skip over them when they are encountered in the screen. However, these functions follow each line with a character called EPL (End to Physical Line) to indicate that all data that has been transmitted to the host so far (since the last EPL) has been representing text that was seen on the same line of the screen. Therefore, a line which contained 80 NULs would be portrayed in one of these transmissions as simply:

EPL

The terminal would thus tell the host, "No significant (non-NUL) information occurred on this line of the screen." These transmissions are said to use Line Format. EPL will be the control code CR (015) if the terminal is in CR Mode; if software has placed the terminal in NL Mode, EPL will be NL (012). Transmit Cursor Line will have exactly one EPL; Transmit Data can have from one to 24 EPLs.

Line Format is used to transmit text. It tells the host the line that each character occurred on, but it does not tell the host the position on that line of the text unless the operator used TABs or spaces to explicitly position the text. It omits the NULs on the screen since NULs are not considered significant in the terminal's text operations. This transmission format would typically be used if the host sent data to the screen to be edited by the operator. If the operator justified the text (using the Text Justify operation) and then transmitted the revised text to the host, the EPL characters which are provided would be necessary for the host to determine the results of the justification. For an example of a transmission in Line Format, see the description of TRANSMIT CURSOR LINE in this Section.

In Forms Mode, the Transmit Message function is used for Transmit Field. The Transmit Data function becomes Transmit Unprotected Data. These two functions use a third format, Field Format. In Field Format, the field – a string of consecutive unprotected characters – is a fundamental part of the information transmitted to the host. As in Line Format, NULs which occur in the screen are not represented in the message to the host. But, just as Line Format used EPL characters to separate data that occurred on separate lines

of the screen, Field Format uses TAB (011) characters to separate data occurring in different fields of the form. Every time one of these functions transmits to the host a representation of data that occurred in a single field, it will follow that representation with a TAB.

Field Format uses EPL characters in situations where a field occupies the rightmost column of a physical line. The EPL will follow the text which occurred on that line of the screen. If the last character in a field is located in the last position on a line, TAB will directly follow EPL.

For an example of a transmission in Field Format, see the description of TRANSMIT DATA (Transmit Unprotected Data) in this Section.

TRANSMIT OVERFLOW BUFFER

invoked by ESC O X

This function lets the host interrogate the overflow buffer which contains any non-NUL character shifted out of the screen by the last text-expanding command. The INSERT PARAGRAPH DELIMITER, TAB, and TEXT JUSTIFY commands are text-expanding commands. In addition, when the terminal is in Insert Mode, any displayable character may be a text-expanding command. By reading the overflow buffer, the host can recover from an error in which a character was shifted off the screen, preventing that character from being lost.

If the overflow buffer is empty, indicating that the most recent text-expanding command succeeded (did not shift a non-NUL character out of the screen), this function will cause the terminal to transmit a NUL as the message. If the overflow buffer is not empty, the contents (one character) are transmitted. If a TAB character was in the overflow buffer, it alone constitutes the message.

If a Paragraph Delimiter was in the overflow buffer, the Escape Sequence for the INSERT PARAGRAPH DELIMITER command is transmitted:

ESC P A

If the character in the overflow buffer was displayed in reverse video when it was on the screen, it is transmitted thus:

ESC O J ⟨ch⟩

If the character in the overflow buffer was a graphic character (except gr(137), gr(150) and gr(176) which are NUL, TAB and PD), a message of the following form is transmitted:

ESC F ⟨ch⟩

This says that the graphic character gr(⟨ch⟩) was in the overflow buffer. A reverse-video graphic character could be in the overflow buffer. In this case, both of the Escape Sequences illustrated above would occur in the message.

If the VT61 received a character with bad parity and placed its special marker in the screen, and if that marker has moved to the overflow buffer, then the transmission will be as follows:

ESC O J DEL

TRANSMIT CURSOR CHARACTER

invoked by ESC O W

The character at the cursor position is transmitted, using Character Format, exactly as Transmit Overflow Buffer does.

TRANSMIT CURSOR LINE

invoked by ESC P B or ESC P b

The line containing the cursor is transmitted to the host. TABs, Paragraph Delimiters, other control characters, and reverse-video characters are handled as previously described. The transmission uses Line Format; an EPL character follows text to indicate that it was seen on the same physical line. To illustrate, assume the specified line contains:

1. "aBc" in normal video
2. "d" in reverse video
3. six NULs
4. "e" in reverse video
5. "f" in normal video
6. TAB
7. the graphic character gr(154)
8. a Paragraph Delimiter
9. NULs in the remainder of the line

Then the following characters will be transmitted to the host (they are numbered to correspond with the item listed above which each stands for):

	STX	Transmitted only if the terminal is in Transmit-Request Mode.
1.	a B c	Characters as they occurred in the screen.
	ESC O J	Start of string of characters that appeared in reverse video on the screen.
2.	d	
3.	ESC O j ESC O J	The NULs between the "d" and "e" are passed over, but the terminal sees that the NULs were not in reverse video and signals the transitions.
4.	e	This character was in reverse video.
	ESC O j	End of reverse-video string
5.	f	Which means that the "f" was in normal video.
6.	TAB	
	ESC F	Start of a graphic string.
7.	154	Stands for the graphic character gr(154).
8.	ESC P A	Stands for Paragraph Delimiter.
	EPL	End of Physical Line: 015 (CR) if the terminal is in CR Mode; 012 (NL) if the terminal is in NL Mode.
	EOT	Transmitted only if the terminal is in Transmit-Request Mode.

In Forms Mode, Transmit Cursor Line uses Character Format. The NULs in items 3 and 9 above would have been transmitted as they occurred in the screen. EPL is not included in the transmission.

TRANSMIT DATA

invoked by ESC P N or ESC P n

A message is sent to the host which consists of all the non-NUL data in the screen from (and including) the cursor position to the end of the screen. If there is no data in this area of the screen, only EPL characters are sent, one for each line or part of a line included in the transmitted area.

The area of the screen transmitted is the same area as is affected by an ERASE TO END-OF-SCREEN command. If the cursor is at the rightmost position on the bottom line, a one-character message (or a null message, if there is a NUL at that position) will be transmitted. If the cursor is Home, the entire contents of the screen will be sent.

In Forms Mode, this function is known as Transmit Unprotected Data and uses Field Format. It causes a message to be sent consisting of non-NUL data from the start of the field containing the cursor to the end of the form. Each field on the screen is transmitted, followed by a TAB. If a single field occupies more than one line on the screen, EPL characters* will be sent to separate the portions of the text which occurred on different physical lines. Following is a sample message:

John Doe TAB 146 Main St. EPL* Maynard, Mass. TAB TAB TAB 897-5111 TAB

Conclusions: "John Doe" was typed in the current field on the form; the following field crossed a physical line boundary. The street address shown was typed on the first line of that field; the city and state were seen on the second line of that field. The next two fields were left blank (contained nothing but NULs), and the given telephone number appeared somewhere in the next field. There were six fields on the form to be filled in which were at or beyond the current field since six TABs were transmitted.

COMMAND (INSERT COMMAND DELIMITER) (affects overflow buffer)

invoked by ESC P T or ESC P t

This function inserts a Command Delimiter in the screen as if the terminal were in Insert Mode.

The Command Delimiter is used in conjunction with the Transmit Message function described below. To issue a command, the operator typically types the Cmd key, types the command, and then invokes the Transmit Message function at the end of the command.

An error occurs if inserting the Command Delimiter caused a non-NUL character to be shifted off the end of the screen. The overflow buffer contains this character; it is empty if no error occurred.

An octal code of 001 occurring in the screen serves as the Command Delimiter.

TRANSMIT MESSAGE

invoked by ESC P M or ESC P m

The Transmit Message function allows the operator to send a variable-length message to the host. This function is typically used in Block Mode. The position of the cursor marks the end of the message area; the end of the previous message area marks the beginning of the current message.

This function causes a Message Delimiter to be placed in the screen at the cursor position, destroying the character that was formerly at that position (regardless of whether the terminal is in Insert Mode).

*EPL (End of Physical Line) is a one-character delimiter. It is normally CR (015), but software can select NL (012) as the EPL character by placing the terminal in NL Mode (see Section 2).

Next, a message is sent to the host consisting of all non-NUL data between the last Message Delimiter and the one just entered (including neither delimiter). If no Message Delimiter occurs toward the beginning of the screen buffer from the one just entered, all non-NUL data from the beginning of the screen to the Message Delimiter is transmitted.

If, as the terminal is searching leftward for a Message Delimiter, it finds a Command Delimiter, the Command Delimiter marks the start of the message. The Command Delimiter Escape Sequence, ESC ? s, is transmitted preceding the body of the message. (If the terminal is in Transmit-Request Mode so that messages are preceded by STX, then ESC ? s will be transmitted directly following STX.) This is the only situation in which the Command Delimiter is represented with the Escape Sequence. In all other situations, the terminal represents CD with the usual conversion for graphic characters.

Finally, the cursor is placed directly to the right of the Message Delimiter just entered. (The cursor will be placed over the Message Delimiter if it is at the lower left corner of the screen.)

If the cursor is at the Home position, no Message Delimiter is placed in the screen, and the message consists of all non-NUL data on the screen. The function, which still uses Line Format, is equivalent to the Transmit Data function.

In Forms Mode, this function becomes Transmit Field. The field containing the cursor, followed by TAB, is automatically transmitted to the host. If the cursor is not within a field, the field to the right of the cursor is transmitted. If the cursor is past all the fields on the screen, a TAB is sent to the host, constituting the entire message.

An octal code of 200 in the screen serves as the Message Delimiter.

TRANSMIT ALL

invoked by ESC O V

The complete contents of the screen buffer from (and including) the character at the cursor position to the end of the screen are transmitted to the host as a message in Character Format. This means that NULs are transmitted as they occur in the screen, but EPL characters are not transmitted.

Transmit All is comparable to Transmit Data (when the terminal is not in Forms Mode) except that Transmit Data uses Line Format.

TRANSMIT SELECTED AREA

invoked by ESC O S

This function transmits a complete image of an area of the screen to the host, in Character Format, as in Transmit All. The portion transmitted, the Selected Area, is dynamically selectable. To define the Selected Area, the host uses the following commands:

SET START OF SELECTED AREA invoked by ESC O P

The present cursor position becomes the first character position in the Selected Area.

SET END OF SELECTED AREA invoked by ESC O Q

The present cursor position becomes the last character position in the Selected Area.

Note that if both of the commands above were given with the cursor at the same position, the Selected Area would contain one character.

If the Start of Selected Area is placed beyond the End of Selected Area in the screen, then Transmit Selected Area will transmit data from the Start of Selected Area to the end of the screen.

The Start of Selected Area and End of Selected Area markers are moved only by the commands described above. Their location is always defined relative to the screen, not to text on the screen. For instance, if a scroll shifts all text up one line, the markers do not also move up one line. The Start and End of Selected Area markers are not *characters*; they do not occupy a character position in the screen. When the terminal is switched on or reinitialized, they are located at the start and end of the screen, respectively.

CHECKSUMS

For a string of characters, a checksum is a number which depends on all the characters. If this string of characters had an error included in it – if one of the characters was changed, or missing, or there was a new character added – then the checksum of the string would be different. The probability that multiple errors introduced into the string would cancel out each other's effects on the checksum is small. Therefore, if the string is duplicated and the copy's checksum matches the original checksum, one can assume that there was no error in the copying.

The VT61 computes checksums of transmitted and received data. The host can clear and read the VT61's checksum results. If the host computes checksums in the same way the VT61 does, the host can use this feature of the VT61 to determine whether a transmission error has occurred.

Receiver Checksum

At power-up, the receiver checksum is 0. By issuing a command to the terminal, the host can reset the receiver checksum to 0. The host can command the terminal to transmit to it the current checksum.

Every time a character is received by the VT61 – except for the synchronization characters XON and XOFF, and NUL, which the terminal never reacts to on input – the receiver checksum is updated. Every character in an Escape Sequence causes the checksum to be updated. In particular, if the host sends the Escape Sequence for Transmit Receiver Checksum to the terminal, the checksum that the VT61 transmits will have been updated to reflect the reception of the same Escape Sequence.

Transmitter Checksum

At power-up, the Transmitter Checksum is 0. The host can command the VT61 to reset the Transmitter Checksum to 0. The host can also command the VT61 to transmit the current Transmitter Checksum to the host.

Every time the VT61 transmits a character to the host, the transmitter checksum is updated, except when the VT61 automatically transmits XOFF or XON as synchronization characters. (If the operator types XOFF or XON from the keyboard, however, their transmission will cause the transmitter checksum to be updated.) If the host gave the terminal two consecutive commands to Transmit Transmitter Checksum, the checksums transmitted would be different – the second checksum would reflect the characters in the first transmission.

Algorithm

The following procedure is used to update the appropriate checksum upon transmission or reception of a character. First, the old checksum is rotated left one position. The old high-order bit becomes the new low-order bit. The character transmitted or received is combined with the result of the rotation using the Exclusive OR (XOR) operation. Only the seven data bits of the character are used in the XOR operation, and they are XORed with the low-order seven bits of the rotated checksum. The high-order bit of the rotated checksum is not changed in this stage of the operation.

In the case of a received character with bad parity, instead of XORing the character with the rotated checksum, the VT61 complements all eight bits of the rotated checksum (which is the same as XORing 377 into the rotated checksum).

This procedure yields the new checksum. Consider the following example: The present contents of the Receiver Checksum are octal 147 (binary 01 100 111). What happens when the VT61 receives the character "G" (octal 107; binary 01 000 111)?

1.	01 100 111	First, each bit in the Receiver Checksum is shifted one position to the left.
2.	0 11 001 11	
3.	11 001 110	The old high-order bit (0) comes around to the low-order bit. This is the rotated checksum.
	11 001 110	The rotated checksum is combined with the character just received, using the XOR function.
	XOR 1 000 111	
	10 001 001	The high-order bit of the checksum is not affected – it remains a 1.
4.	10 001 001	This is the new checksum: octal 211.

IF THE VT61 HAD RECEIVED THE "G" WITH INCORRECT PARITY . . .

	11 001 110	. . . then, after the rotated checksum had been computed (in step 3), instead of going on to perform the XOR function,
4a.	00 110 001	all the bits of the rotated checksum would have been complemented. The new Receiver Checksum would have been octal 061.

Transmission Format

The Transmitter and Receiver Checksums are transmitted in Character Format. The transmissions closely resemble the formats of the Transmit Character and Transmit Overflow Buffer operations. If the designated checksum is a number from octal 040-176, it will be sent directly.

STX ⟨checksum⟩ EOT

If the checksum is 177, it will still be sent directly; the character DEL (177) will be the message to the host. If the checksum is in the range 000-037, the Graphics Mode conversion will be used, and the message will appear thus:

SOM ESC F ⟨checksum+137⟩ EOM

Exceptions are 000 and 011 which are sent directly, and 037 which is encoded as ESC P A.

If the checksum is in the range 200-377, the reverse-video conversion is used. ESC O J will appear in the message. The rest of the message will be the same as it would be for a checksum 200 less than the one requested.

CLEAR RECEIVER CHECKSUM

invoked by ESC O [

CLEAR TRANSMITTER CHECKSUM

invoked by ESC O \

The appropriate checksum is reset to 0.

TRANSMIT RECEIVER CHECKSUM

invoked by ESC O]

The current contents of the Receiver Checksum are transmitted to the host using the format described above. The checksum that the VT61 will transmit will have been updated to reflect the fact that the terminal has received ESC O]. If the next command the host issues to the terminal is TRANSMIT RECEIVER CHECKSUM, the host will receive a different checksum since three additional characters will have been received by the terminal since the last checksum was transmitted

TRANSMIT TRANSMITTER CHECKSUM

invoked by ESC O ^

The current contents of the Transmitter Checksum are transmitted to the host using the format described above.

The act of transmitting this information causes the Transmitter Checksum to be updated, but this fact is not reflected in the transmission.

OUTPUT ABORT FLAG

The host can determine whether an output operation to the printer or copier failed by commanding the terminal to transmit the contents of the Output Abort Flag.

TRANSMIT ABORT FLAG

invoked by ESC O _

A message specifying the status of the Output Abort Flag is sent to the host. The possible responses from the terminal are as follows:

ESC O x	No output has been aborted since power-up or since the last command to initialize the Abort Flag.
ESC O y	The most recent output abort occurred at the copier.
ESC O z	The most recent output abort occurred at the printer.

INITIALIZE ABORT FLAG

invoked by ESC O `

The abort flag is reset to the ESC O x state.

In Transmit-Request Mode, the terminal will take the initiative to transmit ESC O y or EXC O z to the host if a hardware output error should occur. (These messages are preceded by STX and followed by EOT, and the keyboard is locked – see TRANSMIT-REQUEST MODE in Section 2.)

DIRECT CURSOR ADDRESSING

The host can interrogate the VT61 as to the position of the cursor, or move it to any position on the screen with one command. In this section, line# and column# are excess-40 parameters which refer to a position on the screen. line# is 040 for the top line and 067, which is 040 plus 23 (decimal), for the bottom line. column# goes from 040 (leftmost column) to 157, which is 79 (decimal) greater than 040. If either line# or column# is too large a number to specify a line or column, it is said to be out of bounds. If line# is out of bounds, the line the cursor is on is not changed, although it may move to a different column; if column# is out of bounds, the cursor stays on the same column but may move vertically.

SET CURSOR POSITION

invoked by ESC Y line# column#

The cursor is moved to the line and column specified in the respective parameters, as described above.

REQUEST CURSOR POSITION

invoked by ESC O Y

The cursor is not moved. The VT61 will report to the host the position of the cursor, using an Escape Sequence of the same form as the SET CURSOR POSITION command (ESC Y).

REQUEST TERMINAL CONFIGURATION

invoked by ESC Z

The host sends ESC Z to interrogate the VT61 as to the variety of devices which are present. The VT61 responds with an Escape Sequence of the form:

ESC / ‘	if there is no copier and no printer
ESC / a	if there is a copier but no printer
ESC / b	if there is a printer but no copier
ESC / c	if both printer and copier are present

The VT61 and all special versions will transmit a final character to this Escape Sequence in the following format:

binary code of final character: 1 1 x x x p c

The final character will have an octal code of 140 or greater. In that character, the bits labeled x x x will denote the type of VT61; x x x will be 0 0 0 for the terminal described in this document. The bit labeled p will be 1 if a printer is present. The bit labeled c will be 1 if a copier is present.

"A printer is present" means that a device is plugged into the General Output Register. "A copier is present" means that the integral electrolytic copier is attached. These Escape Sequences make no statement about the readiness of any device to do output.

The host can use this feature to make sure that a VT61 is attached to it and to determine what peripherals of the terminal are available. This allows terminals to be moved without requiring reconfiguration of software, as each application program checks to make sure that a terminal with proper features is using the program.

SECTION 5
SPECIAL ESCAPE SEQUENCES

Several Escape Sequences have not yet been discussed. They are as follows (where FCN stands for ESC P since the FCN key on the numeric pad transmits ESC P):

ESC O R	reset terminal (to initial state)
ESC O T	terminal self-test
FCN ⟨n⟩, where ⟨n⟩ is a numeral (0-9)	reserved for definition by user
FCN \ or FCN ¦	cancel

FUNCTIONS WHICH RE-INITIALIZE THE TERMINAL

The two functions described next, Reset Terminal and Terminal Self-Test, reset the terminal to its initial state. One effect of this is that the Silo is erased. The Silo will contain any characters that the host sent to the terminal following the command to Reset the Terminal or perform the Self-Test. Although these characters were put in the Silo to be processed later, they will be destroyed by the re-initializing operation. To avoid this, when software issues one of these re-initializing commands, it must act as if it had just received XOFF from the terminal and cease to transmit characters to the terminal. This will keep the Silo empty until the re-initializing operation is finished. The terminal will transmit XON when the re-initialization is finished to signal that the host may resume transmission.

The terminal does not send an XON after it is initialized by being switched on. This is the only difference between invoking a function to re-initialize the terminal and switching the terminal off and on again.

If the FORCE BLOCK MODE switch inside the terminal is on, the terminal will send a "Power On" Escape Sequence to the host any time it is initialized – either by being switched on or by use of the two re-initializing commands. If XON is sent as described above, the "Power On" Escape Sequence will be sent after the XON.

RESET THE TERMINAL

invoked by ESC O R

The terminal is re-initialized. The screen is filled with NULs, the cursor is moved to the Home position, and all modes are set to their initial states.

The terminal will be unable to communicate with the host for a short time. The synchronization scheme for this situation is described above.

TERMINAL SELF-TEST

invoked by ESC O T

The terminal performs a routine which exercises the microprocessor and tests the integrity of the microprogram.

The first test the terminal performs is to test every location of RAM by writing, reading back, and comparing. All LEDs on the console light up during this test. If the memory does not contain what was just written into it, the test will be repeated indefinitely until it is successful.

The second test the terminal performs is to verify the Macro ROMs. Each Macro ROM in the terminal contains a checksum; by reading the contents of the ROM and recomputing the checksum, the terminal can assure that the microprogram is intact. If any ROM should be found whose contents do not agree with its checksum, the terminal will stop responding to the keyboard and receiver. The left column of LEDs on the console will contain a binary encoding of the ROM which failed the test.

In the event of a failure of any kind in the Terminal Self-Test, the video screen may be full of characters, and the terminal may make a continuous beeping sound. The keyboard will not respond to typing, and the terminal must be physically switched off and back on before use of it can continue.

By verifying the integrity of the microprogram, the self-test function relieves diagnostic programs of the necessity of testing all of the VT61 functions in order to detect a ROM error which only interferes with the execution of certain functions.

If all tests are successful, the terminal will be re-initialized.

The terminal will be unable to communicate with the host for a short time. The synchronization scheme for this situation is described above.

The host may interpret the XON or the Escape Sequence which the terminal transmits after re-initialization (as described above) as proof that the test was successful.

SPECIAL ESCAPE SEQUENCES FOR USER DEFINITION

ESC P 1	ESC P 2	ESC P 3	ESC P 4	ESC P 5
ESC P 6	ESC P 7	ESC P 8	ESC P 9	ESC P 0

These Escape Sequences are reserved for definition by user programs. The terminal will not respond to them.

In Block Mode, when the operator types FCN ⟨n⟩, where ⟨n⟩ is a numeral, the terminal will transmit the following message to the host:

ESC P ⟨n⟩

where ⟨n⟩ is the code for the same numeral that the operator typed.

In Block Mode and Alternate-Keypad Mode, the operator can make the terminal send the messages:

ESC P 1
ESC P 2
ESC P 3
ESC P 4

to the host by typing the keys on the auxiliary keypad labeled F1, F2, F3 and F4, respectively.

Using this feature, the operator can communicate with the host without having to leave Block Mode or transmit text from the screen.

CANCEL

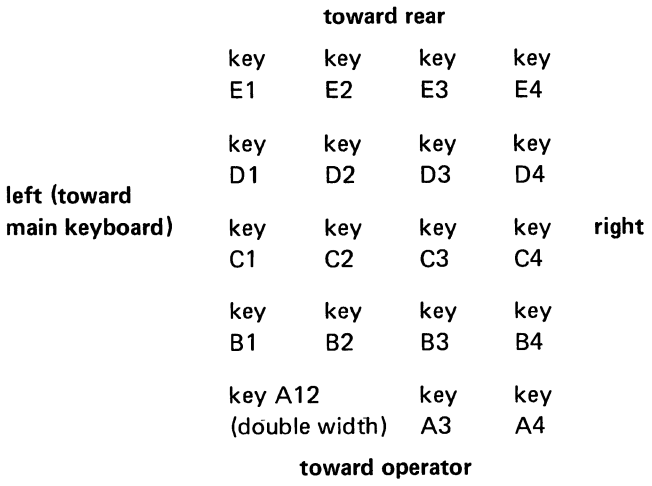
The following Escape Sequences will never produce any effect at the terminal:

ESC P \
ESC P ¦

The operator may use the \ key if he typed the FCN key and then decides he does not want to invoke any functions but, for instance, continue typing.

APPENDIX A
THE AUXILIARY KEYPAD

Spatial Relation of the 19 keys on the keypad:



In the diagram above, the key named . . .	has these markings in white on top of key . . .	and has these markings in white on front of key . . .	and transmits the following code(s) . . . in Numeric-Keypad Mode	and transmits the following code(s) . . . in Alternate-Keypad Mode	and, in Block Mode and Alternate-Keypad Mode, performs the function called . . .
Numeric keys					
A12	0	Delete Character	060	ESC ? p	Delete Character
B1	1	TxJus	061	ESC ? q	Text Justify
B2	2	XmMsg	062	ESC ? r	Transmit Message
B3	3	Cmd	063	ESC ? s	Insert Command Delimiter
C1	4	F3	064	ESC ? t	ESC P 3 [1]
C2	5	F4	065	ESC ? u	ESC P 4 [1]
C3	6	I/R	066	ESC ? v	[2]
D1	7	F1	067	ESC ? w	ESC P 1 [1]
D2	8	F2	070	ESC ? x	ESC P 2 [1]
D3	9	Home	071	ESC ? y	Cursor Home
A3	.	CEmph	056	ESC ? n	Change Emphasis
A4	ENTER	XmDat	015	ESC ? M	Transmit Data [3]
Editing keys					
E2	BFld		ESC Q	ESC Q	Backward Field
E3	FFld		ESC R	ESC R	Forward Field
E4	↑		ESC A	ESC A	Cursor Up
D4	↓		ESC B	ESC B	Cursor Down
C4	→		ESC C	ESC C	Cursor Right
B4	←		ESC D	ESC D	Cursor Left
Multiplexing key					
E1	FCN		ESC P		depends on key typed subsequently

[1] The Escape Sequence listed is sent to the host as a message.
[2] If the terminal was in Insert Mode, it switches to Replace Mode (exits Insert Mode). If the terminal was in Replace Mode, it switches to Insert Mode.
[3] In Block Mode, the ENTER key will invoke the Transmit Data function whether the keypad is in Numeric Mode or Alternate Mode.

APPENDIX B

FUNCTION NAMES ON THE MAIN KEYBOARD

Some of the keys on the main keyboard are used in conjunction with the multiplexing (FCN) key on the auxiliary keypad to allow the user to transmit the Escape Sequences easily. Special labels appear on such keys to identify their functions when used with the FCN key. EXAMPLE: Deleting a line – The FCN key transmits ESC P. On the front of the D key on the main keyboard, the symbol "DelLn" appears. Typing the FCN key, then D, transmits ESC P d to the host, which, if echoed, will indeed cause the terminal to perform its Delete Line function.

Some functions which the operator is not expected to perform or which can be invoked from the numeric keypad's editing keys do not appear as names on keys.

TERMINAL MODES – described in Section 2

The key on the main keyboard named . . .	**has a blue label on the front saying . . .**	**because typing the blue FCN key and that key [2] will transmit the codes for the mode named . . . [3]**
Y	APrt	Auto-Print Mode [4]
U	HldSc	Hold-Screen Mode [4]
I	Insrt	Insert Mode
K	EditK	Editing-Keypad Mode

TEXT EDITING AND CURSOR CONTROL FUNCTIONS – described in Section 3

The key on the main keyboard named . . .	**has a blue label on the front saying . . .**	**because typing the blue FCN key and that key [1] will transmit the codes for the function named . . . [3]**
Q	Home	Move Cursor Home
W	EndTx	Cursor to End of Text
E	CEmph	Change Emphasis
R	ClJus	Clear and Justify
A	Parag	Insert Paragraph Delimiter
S	DelCh	Delete Character
D	DelLn	Delete Line
F	InsLn	Insert Line
Z	erEOS	Erase to End-of-Screen
X	erEOL	Erase to End-of-Line
C	Ruler	Write the Ruler
V	TxJus	Text Justify

INPUT/OUTPUT AND DEVICE CONTROL FUNCTIONS – described in Section 4

The key on the main keyboard named . . .	**has a blue label on the front saying . . .**	**because typing the blue FCN key and that key [1] will transmit the codes for the function named . . . [3]**
T	Cmd	Command (Insert Command Delimiter)
G	CpyLn	Copy Cursor Line
H	PrtSc	Print Screen
J	PrtLn	Print Cursor Line
B	XmLn	Transmit Cursor Line
N	XmDat	Transmit (Unprotected) Data
M	SmMsg	Transmit Message

SPECIAL ESCAPE SEQUENCES – described in Section 5

The key on the main keyboard named . . .	**has a blue label on the front saying . . .**	**because typing the blue FCN key and that key will transmit the proper codes for . . . [3]**
\ [1]	Cancl	a null (not to be implemented) function

[1] Shifted or unshifted.
[2] Capital letter must be typed to enter mode; lower-case letter must be typed to exit mode.
[3] In Full Duplex, the function is actually performed only if these codes are echoed back to the terminal. In Block Mode, the function is performed directly instead of sending codes. However, for some functions, performing the function will require that the terminal send codes to the host.
[4] In Block Mode, this function cannot be invoked from the keyboard.

THE SPECIAL ESCAPE SEQUENCES RESERVED FOR USER DEFINITION – described in Section 5

The numeral keys on the main keyboard have blue labels on the front saying "F1" through "F10", since typing the blue FCN key and a numeral produces an Escape Sequence which is reserved for user programs and since, in Block Mode, typing the FCN key and a numeral causes a special user-defined message to be transmitted to the host.

APPENDIX C

THE LIGHT-EMITTING DIODES

There are eight red light-emitting diodes (LEDs) to the right of the auxiliary keypad. Each LED indicates a mode or condition of the terminal. It is lit when the condition is in effect or when the terminal is in the indicated mode, enabling special features. All the LEDs light up for a short time during the Terminal Self-Test.

Each LED bears a label to tell which mode or condition it indicates. These labels and the relative positions of the LEDs are described below.

Position and Label	Initial State	Refer to . . .
● AUTO-PRINT	Off	Sec. 2, AUTO-OUTPUT MODES
● INSERT	Off	Sec. 2, INSERT MODE
● EDITING KEYPAD	Off	Sec. 2, ALTERNATE-KEYPAD MODE
● FORMS	Off	Sec. 2, FORMS MODE
● BLOCK	Off [1]	Sec. 2, BLOCK MODE
● KEYBOARD LOCKED	Off [2]	Sec. 2, DISABLE-KEYBOARD MODE
● RECEIVER OFF	Off	Sec. 1, FULL DUPLEX SYNCHRONIZATION
● CLEAR TO SEND	On [3]	Sec. 1, HALF DUPLEX

● Indicates the position of a LED.

[1] The initial state of this LED is On if the FORCE BLOCK MODE switch is set.

[2] This LED will also be on initially if the FORCE BLOCK MODE switch is set until the host responds to the "Power On" message from the terminal and unlocks the keyboard.

[3] This LED monitors the condition of the Clear to Send signal from the modem. If it is initially off, it does not necessarily indicate a malfunction.

APPENDIX D
VT61 COMMANDS

1. CONTROL CODES

Octal Code	Name	Use
000	NUL	provides positioning information in some transmissions to the host; always ignored upon reception by the terminal
002	STX	begins messages
004	EOT	ends messages
007	BEL	sounds audible alarm
010	BS	backspace (a command; same as Cursor Left)
011	TAB	TAB command
012	LF	line feed
012	NL	new line (if terminal is in New Line Mode)
015	CR	carriage return
033	ESC	Escape; introduces sequences listed below
177	DEL	ignored upon reception by the terminal

2. TWO-CHARACTER ESCAPE SEQUENCES

Final Char.	Meaning	Final Char.	Meaning	Final Char.	Meaning
=	enter altern keypad	H	home	W	enter printer ctrl
>	exit altern keypad	I	reverse line feed	X	exit printer ctrl
?	[see group 6]	J	erase to end-of-scr	Y	[see group 5]
A	cursor up	K	erase to end-of-line	Z	identify term type
B	cursor down	O	[see group 3]	[	enter hold screen
C	cursor right	P	[see group 4]	\	exit hold screen
D	cursor left	Q	backward field	]	copy screen
F	enter graphics	R	forward field	^	enter auto copy
G	exit graphics	V	copy line	_	exit auto copy

3. THE ESC O GROUP

Param.	Meaning	Param.	Meaning	Param.	Meaning
A / a	enter/exit maintenance	O	insert line (up)	[	clear rec checksum
B / b	enter/exit block	P	set start sel area	\	clear xmit checksum
C / c	enter/exit linear addr	Q	set end sel area	]	transmit rec checksum
D / d	enter/exit new line	R	reset terminal	^	transmit xmit checksum
E / e	lock/unlock keyboard	S	transmit sel area	_	init abort flag
F / f	enter/exit forms	T	test terminal	`	xmit abort flag
G / g	enter/exit alarm	V	transmit all	x	no output aborted
H / h	enter/exit xm request	W	transmit cursor char	y	last abort copier
I / i	enter/exit auto tab	X	transmit ovfl buffer	z	last abort printer
J / j	enter/exit reverse vid	Y	transmit cursor pos	{	terminal is on
N	delete line (down)	Z	get half-duplex line	|	request to scroll
				}	screen overflow

4. THE ESC P GROUP

Param.	Meaning	Param.	Meaning	Param.	Meaning
A / a	paragraph delim	I / i	enter/exit insert	S / s	delete char
B / b	xmit cursor line	J / j	print cursor line	T / t	command
C / c	write the ruler	K / k	enter/exit alt keyp	U / u	enter/exit hold scr
D / d	delete line (up)	M / m	transmit message	V / v	text justify
E / e	change emphasis	N / n	transmit data	W / w	cursor to end of text
F / f	insert line (down)	Q / q	cursor home	X / x	erase to end of line
G / g	copy cursor line	R / r	clear and justify	Y / y	enter/exit auto print
H / h	print screen			Z / z	erase to end of scr

ESC P \ and ESC P ¦ are not to be assigned.

5. ESC Y

First Param.	Meaning
(040)	top line
(041)	line #2
(042)	line #3
.	.
.	.
.	.
(065)	line #22
(066)	line #23
(067)	bottom line

Codes 070 and above as the first parameter are "out of bounds".

Second Param.	Meaning
(040)	leftmost column
(041)	column #2
(042)	column #3
.	.
.	.
.	.
(065)	column #22
(066)	column #23
(067)	column #24
(070)	column #25
(071)	column #26
.	.
.	.
.	.
(155)	column #78
(156)	column #79
(157)	rightmost column

Codes 160 and above as the second parameter are "out of bounds".

NOTE: Numbers in parentheses are octal.

6. THE ESC ? GROUP

When the terminal is in Alternate-Keypad Mode but not in Block Mode, the key labeled . . .	transmits the Escape Sequence . . .
0/Delete Char	ESC ? p
1/TxJus	ESC ? q
2/XmMsg	ESC ? r
3/Cmd	ESC ? s
4/F3	ESC ? t
5/F4	ESC ? u
6/I/R	ESC ? v
7/F1	ESC ? w
8/F2	ESC ? x
9/Home	ESC ? y
./CEmph	ESC ? n
ENTER/XmDat	ESC ? M

In addition, when a Transmit Message command is executed and the leftward search for the beginning of a message finds a Command Delimiter, the Escape Sequence ESC ? s is transmitted to the host.

7. ESCAPE SEQUENCES WITH INTERMEDIATE CHARACTERS

ESC / ‘	I am a VT61 without copier or printer.
ESC / a	I am a VT61 with copier but no printer.
ESC / b	I am a VT61 with printer but no copier.
ESC / c	I am a VT61 with printer and copier.

APPENDIX E

SILO SUFFICIENCY CHART

When outputting to a hardcopy device or performing a time-consuming operation, the VT61 will send XOFF to the host to request that it suspend transmission. An XON will be sent to indicate that transmission may resume. The host must receive and respond to an XOFF signal from the terminal within the listed interval from the time the VT61 transmitted it to ensure that the Silo does not overflow. The indicated times include processing time as well as transmission delays both ways. The derivation of these time specifications, and other details as to their meaning, are found in Section 1 under FULL DUPLEX SYNCHRONIZATION.

Baud rate settings which are not listed in this table should not be used if the VT61 is used in a situation where it might have to send XOFF, since Silo overflow may occur regardless of the response time.

	VT61 Transmission Speed (in Baud)	VT61 Reception Speed (in Baud)	Maximum Delay Time
	75	75	2.4 sec
	110	110	1.8 sec
	150	150	1.2 sec
	300	300	.6 sec
	600	600	.3 sec
	1200	1200	150 msec
	1800	1800	100 msec
	2400	2400	75 msec
	4800	4800	38 msec
	9600	9600	19 msec
SPLIT SPEEDS	110	600	50 msec
	150	600	150 msec
	300	600	250 msec
	300	1200	75 msec
	600	1200	125 msec
	300	1800	17 msec
	600	1800	67 msec
	600	2400	38 msec

SUMMARY OF SPECIFICATIONS
VT61 VIDEO TERMINAL

DISPLAY	Cathode Ray Tube – 12" diagonal measure
Display Format	24 lines of 80 characters (1920 characters)
Character Set	96-character displayable ASCII set: includes upper-, lower-case, numerals, symbols 32 special graphic characters including 8 scaled horizontal bars
Character Format	7x8 dot matrix in 10x10 area; one-scan descenders
Alternate Video Setting	Reduced-intensity reverse-video characters intermixable with regular characters
KEYBOARD	DEC Standard Keyboard
Transmittable Characters	Any pattern of 7 bits – upper-, lower-case, numerals, symbols, control codes
Auxiliary Keypad	19 keys; two modes make it suitable for Numeric Data Entry or Text Editing
INTERFACE SETTINGS	Standard; switch-selectable
Half-Duplex	103 Dataset, 202 Dataset and Current Loop
Parity	Even/odd/none
Variable Speeds	Transmission and reception at 75 to 9600 Baud
Split Speeds	Transmission at 75/110/150/300/600 Baud with reception at 1200/1800/2400/4800/9600 Baud
INTERFACE HARDWARE	Choice of: 20 mA Current Loop – Mate 'n Lok® connector 20 mA Current Loop – 4-pin telephone plug EIA Connector for Full-Duplex Operation EIA Connector with Half-Duplex Control Signals
PRINTER (OPTIONAL)	LA180 or equivalent (adapter to drive a serial printer will be available)
Character Set and Format	Device-dependent; lower-case supported by VT61
Width	At least 80 columns required

MODES	Software-controlled options
Block Mode	Simulated off-line editing environment
Insert Mode	New data moves old data rather than destroying it
Reverse-Video Mode	Characters received appear in reverse-video
Graphics Mode	Used to enter special graphic symbols in screen
Forms Mode	Reverse-video characters become "protected"
Hold-Screen Mode	Rate of incoming data regulated by operator
Auto-Output Modes	Automatic output to printer or copier as data leaves screen
NL Mode	Software can select compatibility with old or new ASCII line-delimiter standards
FUNCTIONS	Software-controlled
Text Justify	Compacts text; aligns tabular data; repositions lines; wraps words which cross line boundaries
Text Erasure	To end of field, line, or screen
Line Movement	Line-at-a-time insertion or deletion with line-rippling in either direction
AUTOMATIC INTERACTION	
Information Requestable by Host	Cursor location Character at cursor position Data on cursor line All data/all unprotected data on screen Terminal type (VT61 is distinguishable from VT50, VT52)
Error-Checking	Parity errors detected; checksums calculated on data transmitted and received from host
ELECTRICAL SPECIFICATIONS	
Line Voltage	(US Models) 90-127V (European Models) 180-254V
Line Frequency	47-63 Hz for proper operation of terminal; 59-61 Hz (US Models) or 49-51 Hz (European Models) to ensure correct appearance of video
PHYSICAL SPECIFICATIONS	
Dimensions	Height: 360 mm (14.1 in.) Width: 530 mm (20.9 in.) Depth: 690 mm (27.2 in.) Minimum Table Depth: 450 mm (17.7 in.)
Weight	20 kg (44 lbs)
Case Material	Injection-molded Noryl thermoplastic
Screen Phosphor	P4
ENVIRONMENTAL SPECIFICATIONS	Contact a Digital Sales Office for the current definitions of the following environments
VT61	DEC STD 102 – Class B Environment

digital

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754

PRINTED IN U.S.A.