

# CHAPTER 1 FUNCTIONAL BLOCK DIAGRAM

The following subsystems of the VT50 are shown in the diagram at right:

- screen - the picture tube itself
- Random Access Memory - contains the screen image as well as the scratch-pad
- receiver - part of UART
- monitor - the linear electronics which deflects the electron beam and modulates the video signal
- uPR - the microprocessor
- ROM - the read-only memory, which contains the microprogram
- keyboard - the keyboard and decoder circuits
- xmtr. - transmitter (part of UART)
- power supply
- the circles at the right of the diagram stand for the interface electronics: opto-isolators, etc.

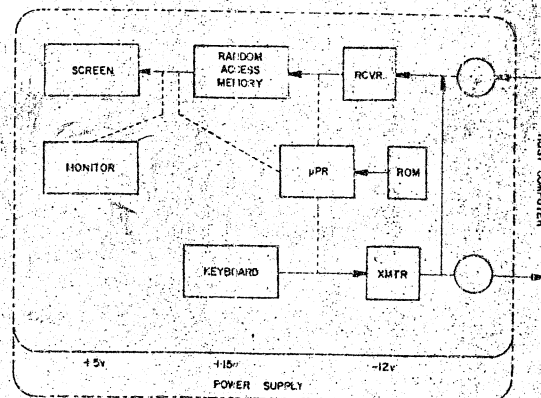


Figure 1  
Block Diagram

An arrow indicates the direction of information flow. A dotted line is drawn from the monitor circuit or the microprocessor where these circuits control the flow of information.

## PHYSICAL LOCATION

The electronics for each of these functional portions of the VT50 is on the indicated board:

- 1 - Power Supply & Monitor module
- 2 - Data Paths, Memory & Decoders module
- 3 - ROM, UART & Timing module
- 4 - Keyboard module
- 5 - The picture tube

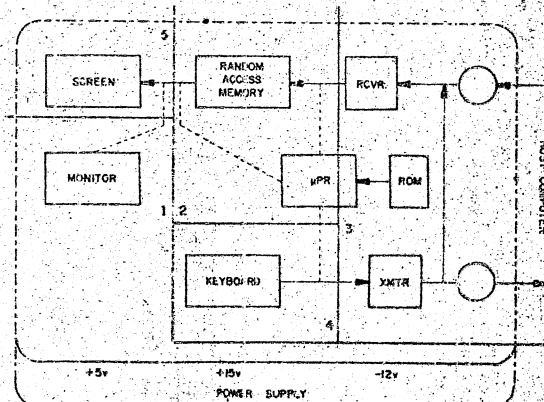


Figure 2  
Location of Subsystems

## IDENTIFYING A PROBLEM WITH THE VT50

The first step in defining the extent of a problem is to evaluate which of the functions of the VT50 work correctly.

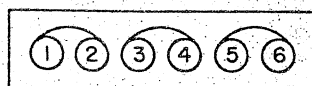
### IN OFF-LINE MODE

Does the terminal work correctly in off-line mode?

Do characters typed on the keyboard appear on the screen?

Do escape sequences typed from the keyboard cause the correct effect?

- and then does the terminal fail when switched on-line and connected to a host? When the terminals of the VT50 are wired together as follows:



TERMINAL STRIP

or when the VT50 is on-line to a computer programmed to echo each character, the VT50 should react exactly as it would in off-line mode. If not, the error is in the interface circuitry.

### AS A TRANSMITTER

Connect the VT50 to a host and switch it on-line. Do characters typed on the keyboard reach the host? If so, the transmitter portion of the VT50 is not malfunctioning.

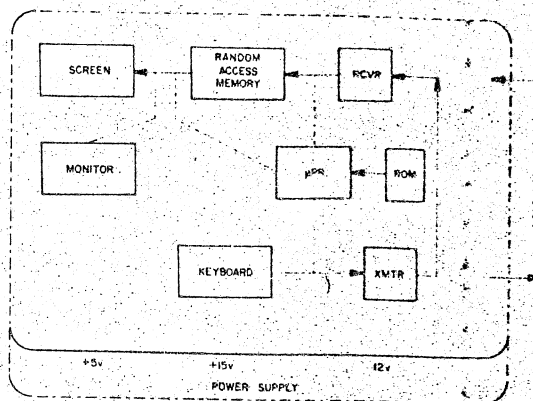


Figure 3  
Subsystems which are good if terminal works correctly in Off-line mode.

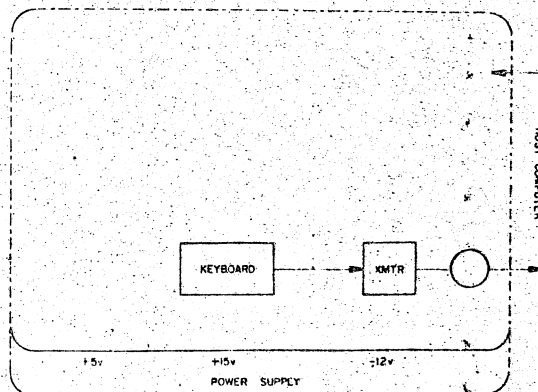
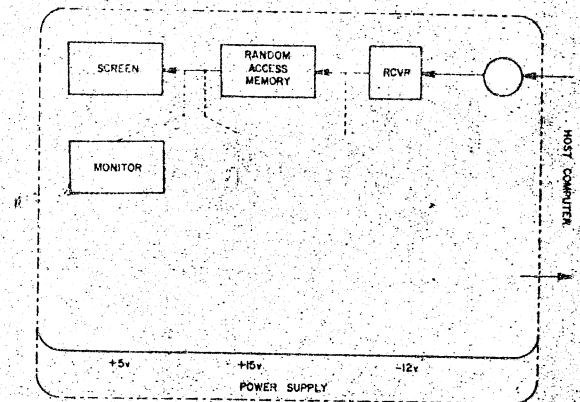


Figure 4  
Transmitter portion of VT50



## AS A RECEIVER

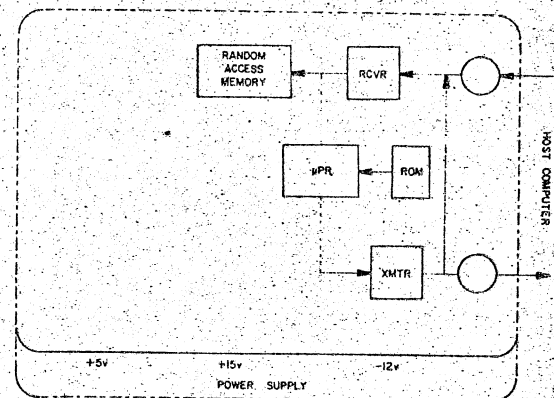
With the VT50 connected to a host and on-line, do characters transmitted by the host appear on the screen? Do escape sequences and control codes produce the correct effect? If so, the receiver portion of the VT50 is not malfunctioning.



**Figure 5**  
Receiver portion of VT50

## TERMINAL SELF-IDENTIFICATION

When the terminal receives ESC Z, it is supposed to transmit ESC / A. Even if the VT50 is malfunctioning such that incoming characters are not displayed on the screen, the portion of the VT50 shown at right can be proven to be working correctly by testing this feature.

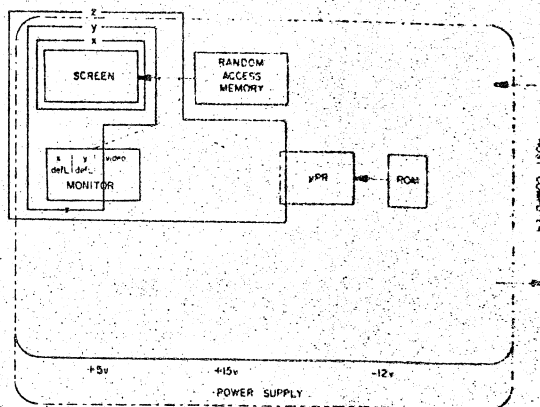


**Figure 6**  
Portion of VT50 which must work in order to produce terminal self-identification.

## LOCALIZING AN ERROR IN THE DISPLAY CIRCUITRY

The figure at right shows which portions of the display circuitry are working if

- x - any light is visible on the screen, with any intensity setting.
- y - the raster is visible if the intensity is turned up all the way.
- z - the cursor is flashing.



**Figure 7**  
**The Display Circuitry**

**IF THE VT50 DOES NOT PERFORM ANY OF ITS FUNCTIONS CORRECTLY:**

1. Check to see if the power supply is producing the correct voltages.
2. If the power supply is functioning correctly, the problem may be that, because of being in the wrong mode or because of failure of a condition flag to register with the microprocessor, it may be stuck in an infinite loop. If so, the outputs of PC can be read with an oscilloscope to determine at which point in the microprogram the processor is stuck. This may provide a clue as to the source of the malfunction.

The linear transformer has a flux shield optimally positioned to prevent interaction between the flux generated by the transformer and the electron beam of the CRT, which would produce, if the line frequency varied from the nominal, a phenomenon known as "swimming". Current goes into a full-wave rectifier with center-tap. The positive and negative outputs of this rectifier are regulated to +15v and -12v, respectively.

Another winding of the transformer is passed through a full-wave, center-tap rectifier and capacitor-input filter to be regulated down to +5v. There is a single voltage reference for the entire power supply: a voltage divider utilizing a Zener diode. Thus there are no adjustments which have to be made to take into account line ripple and fluctuation.

Each regulator has current-limiting and foldback to protect against inadvertent short-circuits during maintenance or for any other reason.

## CHAPTER 2 POWER SUPPLY

Four voltages are produced by the power supply:

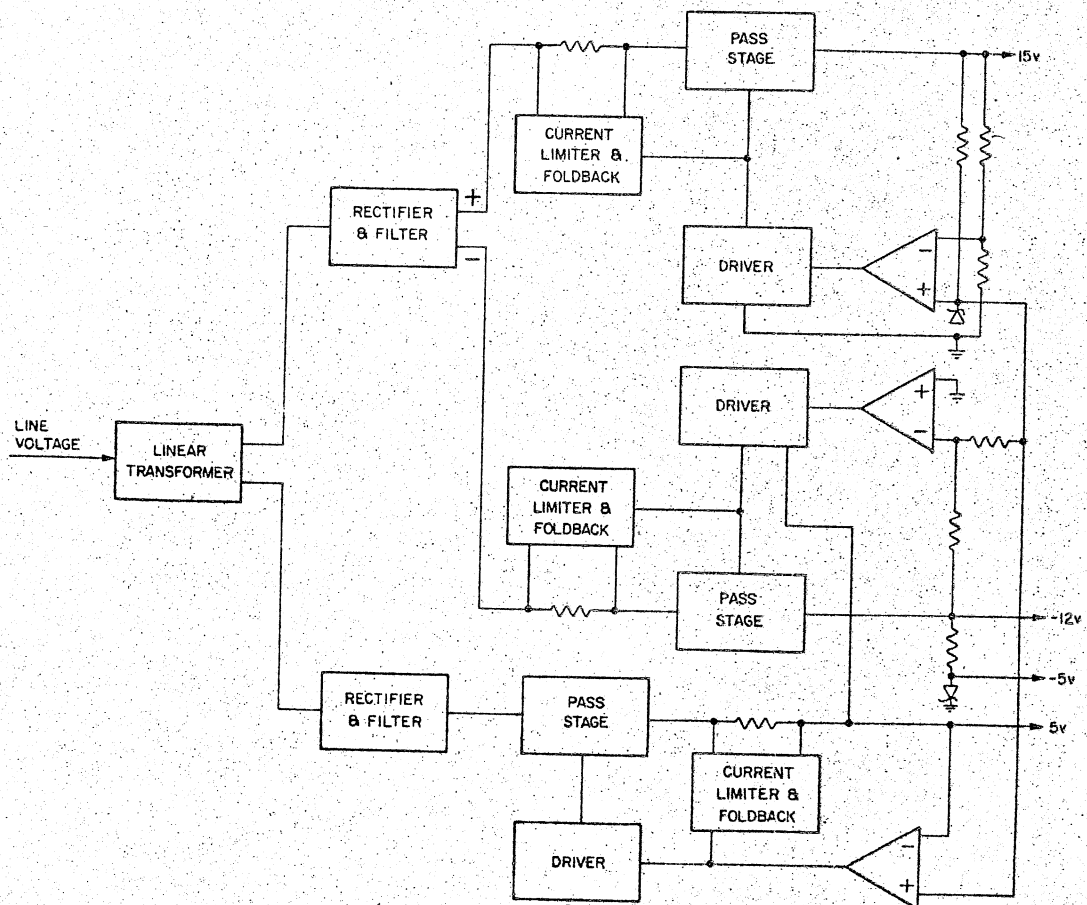
+15v @ 1.5 amp drives the horizontal and vertical deflection circuits, and the DF11 modem option.

-12v @ 0.52 amp also powers the vertical deflection circuit and DF11 option, as well as supplying MOS logic.

-5v @ 15mA drives MOS logic.

+5v @ 4.5A is produced for TTL logic.

The block diagram for the VT50 power supply can be drawn as follows:



## CHAPTER 3 MONITOR

### THE PICTURE TUBE

The picture tube consists of a negatively-biased cathode; a heater which warms the cathode so that electrons will be emitted; a control grid to modulate the emission of electrons; several other grids which control the shape of the beam; and accelerating electrodes and anodes which, by carrying a high positive voltage, draw the electrons toward the screen. When the electrons hit the screen, they are traveling at such a high velocity that the phosphor on the screen gives off light. The position on the screen where the electron beam hits it is controlled by deflecting the beam with magnetic fields generated by a coil on the picture tube called the yoke. There is one winding in the vertical direction and one in the horizontal direction. The angle of deflection is proportional to the strength of the magnetic field.

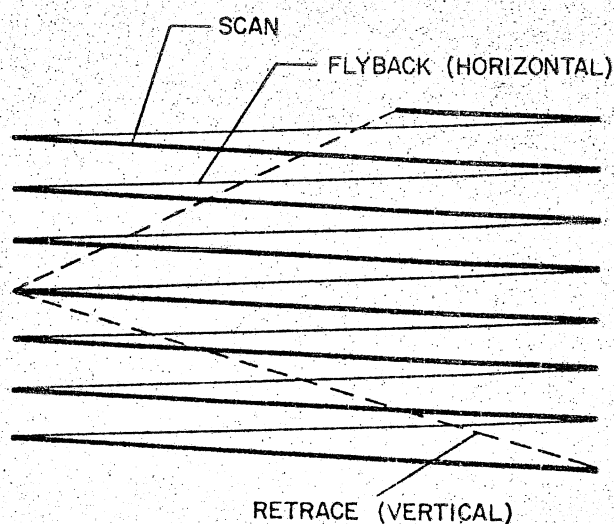
### BEAM DEFLECTION

The current through both the horizontal and vertical windings is in a slowly positive-going and rapidly negative-going sawtooth cycle. The frequency of the cycle for the horizontal yoke is 256 times that of the vertical yoke, if the line frequency is 60Hz.

The effect is that

1. With a constant but slow downward movement, the electron beam scans from the left of the screen to the right, then quickly returns (flies back) to the left. If we ignore this flyback, the path that the electron beam takes is one of nearly horizontal lines, each one under the preceding one. There are 240 of these lines.
2. Sixteen more horizontal lines are painted during which time the current on the vertical yoke is shifting negative, causing the beam to rise to the top of the screen.

Process 1 above is the vertical scan; process 2 is the vertical retrace.

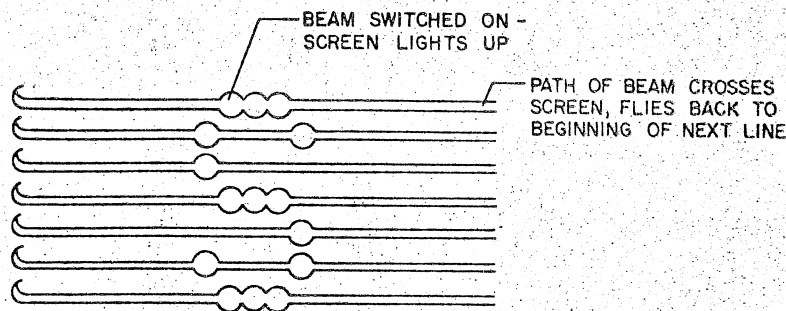




## BEAM MODULATION

The positions covered by the 240 horizontal scans during the vertical scan comprise the "meaningful" locations of the screen; information is never displayed during horizontal flyback or vertical retrace. As a result of signals initiated by the microprocessor, at times during these scans electrons are allowed to strike the screen at full speed, causing it to light up. At other times, they are prevented from reaching the screen. Unlike a television, there are only these two levels of intensity.

A character is painted on the screen by turning on the electron beam for one or more of five consecutive positions on one or more of seven consecutive lines.

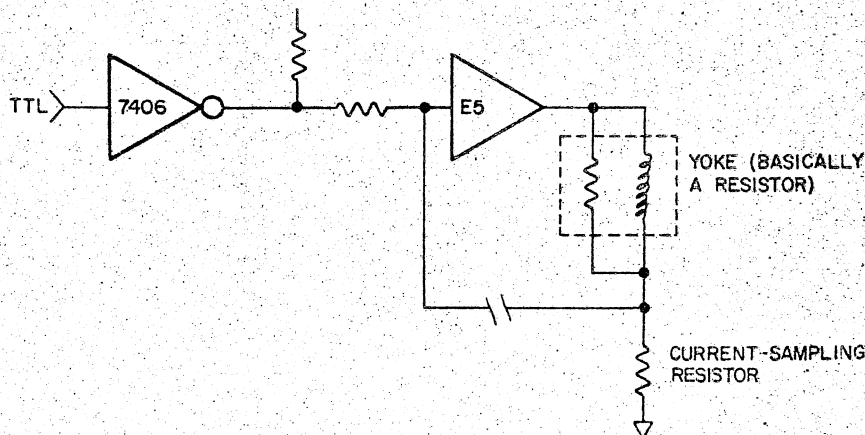


## YOKE DRIVERS

Due to a large change in voltage with respect to time and a greater angle of deflection required, most of the voltage drop across the horizontal yoke is through inductance, whereas, in the vertical yoke, resistance is more significant than inductance in causing the voltage differential. For this reason, the two driver circuits are designed differently.

## VERTICAL YOKE

At the center of the driving circuit for the vertical yoke is an operational amplifier (E5; coordinates B3) in a current-sampling integrator configuration (with capacitor C28). The basic functional circuit is shown below.



A power-driver stage is added by an emitter-follower (transistors Q13 and Q17). Q18 serves as a voltage-level converter. It is a common-emitter amplifier. To prevent DC leakage current from flowing through the vertical yoke, a capacitor is put in series with it. In addition, feedback is brought from the yoke back to the op amp; the Capacitor C40 varies E5's gain. The gain is 1 under DC conditions, when the resistance between the inverting input of E5 and ground is 100K ohms; the AC gain is 10. This ensures saturation of the output of the op amp, which makes the flyback cycle more effective.

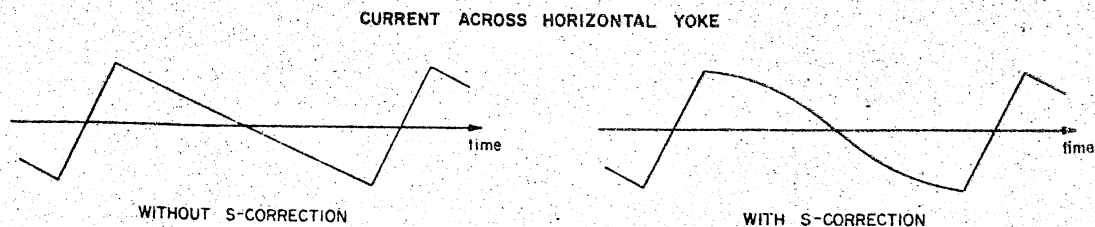
Diode D23 charges capacitor C31 during vertical retrace. This reduces the current requirements of the vertical circuit during the downward scan, since it uses the energy stored in the capacitor.

## HORIZONTAL YOKE

The horizontal yoke is basically an inductor. By supplying a constant voltage, a constantly rising current will result, which becomes the horizontal scan. The momentum of this current across the coil is used to charge a capacitor which supplies the reverse voltage needed in flyback. This capacitor and the yoke form a resonance circuit. A diode is added to keep the capacitor from ringing after flyback has been achieved. As a result of flyback, the high current in the reverse direction is also sent to T1, the flyback transformer, where it performs several functions:

1. It is stepped up to 11,000 volts and sent, through a diode, to the anode of the cathode ray tube, which has a large capacitive effect. This rectifies the 11,000 volts into a DC current.
2. It provides a source of several high voltages.
3. It is used to generate the third harmonic of the waveform of the capacitor as it charges. This signal has about 15% of the amplitude of the original waveform, and is added on the secondary side of the transformer and subtracted on the primary side to produce a higher voltage with the same power consumption.

Again there is a capacitor in series with the yoke. As well as eliminating DC across the yoke, it supplies the S-correction to the yoke. It was mentioned before that the angle through which the yoke deflects the electron beam varies directly with the current through the yoke. But this does not result in a constant rate of speed of the beam's travel across the screen. Since all the points on the screen are not an equal distance from the source of the electron beam, a deflection of 1 degree when the beam is at the center would produce less distance change than the same deflection when the beam is at an edge of the screen. The capacitor performs S-correction by slowing down the movement of the electron beam when it is near the edges.



A variable inductor (T3) in series with the yoke is used to provide adjustment of the width of the scan.

T4, the other coil in series with the yoke, is the linearity coil. Its non-linear inductance corrects for some of the second-order effects: the resistance of the yoke, and some of the voltage drops in the circuit.

A problem encountered was that of storage time of the power transistor Q19, which could delay the signal HORIZ L from the microprocessor. The microprocessor emits another signal, SYNC H, which goes high 5.2 microseconds after the HORIZ L signal began.

The circuit at coordinates D3 uses the SYNC signal as the time base to integrate voltage sampled from pins 8 and 9 of the flyback transformer T1. A fast power transistor, which has started flyback too early with respect to the SYNC signal, will allow more current to be integrated. This circuit will compensate for the power transistor's fast storage by increasing its base current, which will slow it down. A power transistor which is too slow will likewise be compensated for by reducing its base current. Another benefit of this circuit is to guarantee that the power transistor will always be saturated when the horizontal scan begins, which will ensure its linearity.

## VIDEO CIRCUIT

The control grid is used to regulate the general intensity of the screen. Whether the electron beam produces light or not, at any time, is controlled by switching Q21 on and off, which takes the cathode from +40v (dark) to 2v (light). D37 prevents Q21 from saturating, ensuring high speed of operation.

## CHAPTER 4

### MICROPROCESSOR BLOCK DIAGRAM

This section covers the data structures in the VT50. It covers the internal registers of the VT50, what they contain, and how they are manipulated by the microprogram. It covers the organization of the Random Access Memory, and the available internal data paths.

#### The Random Access Memory (RAM)

There are 1024 7-bit words of read/write memory. 960 are used to hold the current contents of the display (12 lines of 80 characters each) and the other 64 are used as a scratch-pad by the microprogram.

#### The need for address-mapping

1024 is 2 to the tenth power, and hence the number of storage locations in a memory device which has 10 address lines. The VT50 uses 7 such devices in parallel, so that there are 7 bits of information at each location (in each word). This is sufficient to store an ASCII code or a counting number from 0 to 127. It is convenient, especially since the RAM represents the information on the screen, to think of a certain number of address lines as selecting a row, and the rest as selecting a column. But if we say that 4 address lines select the row and 6 select a column, the memory will have 16 rows and 64 columns. That is not enough columns to hold our 12x80 screen. But if we take another address line from selecting the row and use it with the other 6 to select the column, our memory has a configuration of 8x128, and 8 is not enough rows.

If we had a memory with 11 address lines, we could use 4 for the row and 7 for the column, yielding a memory configuration of 16x128. But this solution requires twice as much RAM as is really needed.

What happens in the VT50 is that a 16x64 configuration is used, but a hardware scheme—an address mapping—makes it seem to the microprogram that there are another 64 columns. When, by manipulating the address registers, you try to access one of the 64 columns that don't really exist but that you need to retain the extreme right portion of the screen, the address mapping diverts you instead to somewhere in the lower 4 rows which do exist and which aren't necessary to hold the information on the screen.

In short, when the microprogram tries to access certain memory locations, it gets others instead; in this way, the mapping can cut our 16x64 memory into a 12x80 piece and a 16x4 piece. This is shown graphically in the diagram below. The algorithm is explained on the following page.

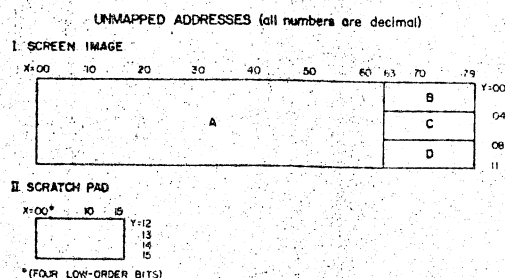


Figure 8

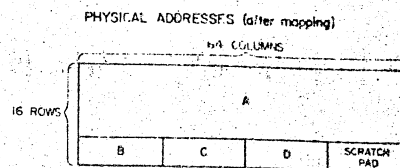


Figure 8a

Figure 8 and 8a

The layout of Random Access Memory (8) as it appears to the program, and (8a) after transformation by the mapping.

## Addressing the RAM

Two hardware registers, herein referred to as the X- and Y-registers, supply the address to the RAM. When this manual describes some action performed on "the RAM", it is understood to mean "the RAM word addressed by the present contents of the X- and Y-registers". The X-register has a capacity of 7 bits of information; the Y-register has 4 bits. However, 10 bits are alone sufficient to provide a unique address. The following is a description of an address mapping which occurs: Let  $y_3, y_2, y_1$ , and  $y_0$  be the bits of the Y-register, and let  $x_6, x_5, \dots, x_0$  be the bits of the X-register, from most-significant bit to least-significant bit. The VT50 video screen has rows 0-11 and columns 0-79. When the Y-register contains 11 or less and the X-register contains 63 or less (i.e.,  $x_6=0$ ), the address mapping is straightforward and the following are the bits of the address sent to the RAM:

$y_3 \ y_2 \ y_1 \ y_0 \ x_5 \ x_4 \ x_3 \ x_2 \ x_1 \ x_0$

Thus far, the possible addresses in which the two most significant bits are 1 have not been exploited; however, columns 64-79 on the screen have not been given an address. In the situation where the x-register contains a number 64 or greater (i.e.,  $x_6=1$ ), the following mapping is effective:

$1 \ 1 \ y_1 \ y_0 \ y_3 \ y_2 \ x_3 \ x_2 \ x_1 \ x_0$

Notice the  $x_5$  and  $x_4$ , which would both be 0 for x-values from 64-79, are not significant in the address provided to the RAM. We have not covered the case in which the Y-register contains a value 12, 13, 14, or 15. These cases, regardless of the value of the X-register, are covered by the second mapping. Since in these cases  $y_3=y_2=1$ , that mapping degenerates to:

$1 \ 1 \ y_1 \ y_0 \ 1 \ 1 \ x_3 \ x_2 \ x_1 \ x_0$

These "leftover" addresses—addresses whose corresponding locations do not correspond to a character position on the video screen—are used to gain access to the "scratch pad" portion of RAM. Only the four least-significant bits of the X-register are significant at all in selecting an address, when the Y-register is 12 or above (or the x-register is 64 or above), except for specifying that the alternate mapping shall be used). At the end of Section 9 is a chart describing what each cell of the scratch-pad is used for, if anything.



## Microprogram Address Control

In the repertoire of the microprocessor are separate instructions which can:

- \*Clear (set to zero) the Y-register
- \*Increment the Y-register
- \*Decrement the Y-register
- \*Clear the X-register
- \*Increment the X-register
- \*Decrement the X-register
- \*Complement bit x3 of the X-register  
(This instruction is useful when in the scratch pad, where it allows you to access a cell half-way across the scratch pad, taking the place of eight increment or decrement X-register instructions).
- \*Load the X-register from RAM
- \*Load the Y-register from the Y-register-buffer
- \*Load the y-buffer from RAM
- \*Selected combinations of the above

## The Y-buffer

A 4-bit y-buffer is provided, allowing the microprogram to perform "indirect addressing" by executing the following sequence of instructions:

1. Manipulate the X- and Y-registers to point to the RAM word which contains the new y.
2. Load the RAM word into the y-buffer.
3. Manipulate the X- and Y-registers to point to the RAM word which contains the new x.
4. Load the RAM word into the X-register.
5. Load the y-buffer into the Y-register.

Having a y-buffer is crucial to this operation since, once one of the x- and y-registers is changed to the new value, that register is not available to hold a pre-specified address so that the other address can be loaded from RAM.

## The Read-Only Memory (ROM)

VT50's come with 4 chips of 256 word x 4 bit ROM. Since each chip supplies either low- or high- order parts of ROM words, the configuration is 512x8. However, this supply may be expanded to 1024x8. A third and fourth bank (or, set of two banks) of ROM will be enabled if jumper W1 on the ROM, UART & Timing board is cut. The ROM contains the microprogram. See Section 5 for a complete description of the microprocessor language.

## Addressing the ROM

Addresses to the ROM are supplied by a 10-bit register called the microprogram counter (PC). The ROM word addressed by PC is decoded and executed: an action is performed, depending on the contents of the ROM at that location. PC is normally incremented after each instruction, as the microprocessor proceeds to the next instruction. Some instructions require two 8-bit ROM words. In this case, PC would be incremented twice during the execution of one instruction. Under certain conditions, the 8 least significant bits of PC can be loaded from the ROM: specifically, from the second word of one of these two-word instructions. The two most significant bits of PC can be incremented by a microprocessor instruction. Note that, in its standard configuration, the most significant bit of PC is unused.

## Other RAM transfers

Under microprogram control, a constant from ROM can, either conditionally or unconditionally, be loaded into RAM. The constant 40 (octal), the ASCII code for a space, can also be loaded into RAM.

## The Accumulator

The VT50 contains a 7-bit accumulator (AC). Micro-instructions exist to increment, decrement, and clear the AC. In addition, AC can be loaded from or stored in RAM. When AC contains a number from  $001_8$ - $067_8$ , the microprocessor can test the keyboard key of the corresponding number to see if it is depressed.

## UART

The Universal Asynchronous Receiver/Transmitter (UART) is a single integrated circuit which performs shifting and control operations to interface the VT50 with a host. Two microprocessor instructions operate on the UART: One transfers the contents of RAM to the transmitting section of the UART, which uses a shift register to transmit the character to the host, one signal at a time. The other instruction stores the last character received by the UART in RAM. In addition, two flags exist so that the microprocessor can test whether the UART has received a new character since the last one was stored, and whether it is still in the act of transmitting a character.

## Testing

Microprocessor instructions exist to allow conditional branching, on the following conditions (as well as the previous tests mentioned above, and others):

- \*whether AC is less than RAM
- \*whether AC = RAM
- \*whether AC is different from X-register

At the end of this section is a block diagram of the VT50 which summarizes this discussion.

## HOW IT WORKS

On the ROM, UART and Timing specifications, page 3, coordinates A7, there are two flip-flops (E2). They form a 2-bit binary counter. A jump loads new values into the eight least significant bits of PC, that is, E9 and E16. An IROM instruction increments the count in the two most significant bits. But an IROM is not to take effect until the next jump.

The 2-bit counter E2 normally contains the same number as is contained in the low-order (A and B) bits of E12. Both are cleared at initialization. At coordinates A6, an IROM instruction sends a low signal into pin 10 of E5 which increments the counter E2. But this new number does not take effect as the ROM address until the signal LD PC L goes low and the contents of E2 are loaded into E12; that is, until the next branch.

The two most significant bits of PC are also incremented in another case: when an instruction occurs at the last location in an ROM bank -- when there is a carry from the eight least significant bits during a count. Not only must E12 be incremented to take the next ROM word from the next bank, but E2 must be incremented, so that at the next jump, when E2 is loaded into E12, if there has not been an IROM since the last jump, the page remains the same. IROM means "the next jump will be the next page", relative to the one the jump is from. Issuing an IROM should cause the number in E2 to be one greater than the number in E12; if the microprocessor comes to the end of a bank and changes banks automatically because of it, both E2 and E12 must be incremented.

There is one other possibility -- that the microprocessor is executing the last instruction in an ROM bank, and this instruction is an IROM. The count in E12 must be incremented once; the count in E2 must be incremented twice. NAND gate E5 ensures that incrementing E2 because of bank overflow occurs at time TH; but an IROM instruction sends a signal into pin 10 of E5 at time TF. Thus, both can occur in a single instruction cycle. Jumper W1 at coordinates B8 causes E5, pins 1, 2, and 3 to turn whatever output QB of E12 was into a 1. Thus, with the jumper in place, the most significant bit of PC is held to a 1, and banks (pages) 3 and 4 of ROM are disabled. If the microprogram is to be expanded to four banks of ROM, this wire must be removed. If the microprogram will occupy three ROM banks, the wire must be removed, and two IROMs must be used to jump from bank 3 to bank 1. As it is shown, an IROM will be sufficient to cause the next jump to be a jump from page 2 to page 1, or from page 1 to page 2.

Page 1 of the Data Paths specifications shows at coordinates C7, the multiplexer which performs the address mapping described earlier in this section. That is, depending on the Select input (pin 1 or E21), either the B inputs or the A inputs become the Y outputs. This multiplexer chooses between the two forms of the address shown on page . The Select input goes low, and the A inputs become the Y outputs, if (gate E17)  $Y2=Y3=1$  or  $X6=1$ . That is, whenever the Y register is 12 or greater or the X register is 64 or greater. Compare this to the algorithm given above.

## CHAPTER 5 TIMING

Synchronization is needed in many areas of the VT50's operation. Many things have to be done by the microprocessor to execute a single instruction — these things must be done in a certain order, and the time in between these events must be sufficient. The brightening of the picture tube must be carefully timed to take the form of characters. And the VT50 must have an internal timing system so that it can transmit its information at exactly the rate at which the host computer is expecting it.

\*We assume you have Integrated-Circuit experience.

### WHAT IT DOES

The following pulses are generated by the timing circuit:

- \*Microprocessor synchronization pulses  
TE, TF, TW, TG, TH, and TJ, 72 nanoseconds long. The time between two TE pulses (one instruction-time) is 1.3 usec.
- \*Eight UART synchronization pulses, at eight different frequencies, one of which is selected by rotary switch S2 on the underside of the VT50 and used by the UART as a time standard.
- \*Signals H FLY H (horizontal flyback) and VERT H (vertical flyback) which regulate the path of the Raster Scan.
- \*Control logic for the Video Shift Register: one pulse (AUTO INC X H) to automatically increment the X register and thus, make available each RAM location on a particular line; one signal (VSR LD H) to load the shift register; one (B OSC A) to shift signals out to the screen.
- \*Logic to reset the system at power-up and whenever the CONTROL and BREAK keys are simultaneously depressed.

### HOW IT DOES IT

E46 is an oscillator which emits a 13.824 MHz square-wave, buffered to TTL levels. The output from this clocks E14 and E22, which act as a 9-bit shift register. Of outputs QC and QD of E14, outputs QA, QB, QC, and QD of E22, QA and QB of E14, and the output (pin 8), of the 8-input NAND gate E18, exactly one will be low at anytime, on a rotating basis—rotating, in the order listed, every 72 nsec. One complete rotation occurs twice in the time of one instruction.

QB of E14 causes E23 to count up, complementing QA of E23. When QA is low, the signals TE, TF, and TW are enabled. When QA is high, the signals TG, TH, and TJ are enabled. Therefore, these signals are asserted (by going to a high state) every other time the appropriate output from the shift register goes low.

The shift register was constructed as it was so that "neighboring" pulses would be caused by signals from the same chip, to eliminate individual variances which might occur between the reaction times of two different 7495's.

Each time the count in E23 reaches 9, the LOAD goes low. However, the LOAD is asynchronous, and not effected until the next CLOCK. Thus, the E23 is a divide-by-ten counter. Outputs QC and QD from E23 have the same frequency: 1/10 the frequency of the shift register outputs; 1/90 the frequency of the oscillator. This frequency is 16x9600 Hz; it becomes the 9600 Baud time standard for the UART.

The E38 and E33 counters divide this frequency by 2 repeatedly to produce the 75 through 4800 Baud time standards. E29 divides the 1200 Baud signal (whose frequency is 16x1200 Hz) by eleven to produce an acceptable 110 Baud time standard.

The frequency of QD on E33 is 600 Hz. If R36 is present, as it will be in countries with 50 Hz line frequency, a low signal to LOAD will on the next CLOCK, cause the 4 to be loaded into E39; if R36 is absent, a 6 will be loaded in. E39 thus becomes a divide-by-10 or a divide-by-12 counter, with outputs QD and CARRY of the same frequency as the presumed line current. This signal goes up to coordinates C1 to initiate vertical flyback and to coordinates A4 to inform the processor that the electron beam is at the top-of-screen position.

At coordinates A5 is a monostable multivibrator with a period of 125 msec. The discrete circuitry to the left of E43 causes it to fire at power-up time. The logic at coordinates A6 causes it to fire also when CONTROL and BREAK are down simultaneously. All the significant counters and flip-flops are cleared when E43 fires and the micro-program counter (PC) is set to zero to restart the microprogram.

## TROUBLESHOOTING TIPS

When the test point DIS OSC L is tied low, an external pulse may be supplied at EXT H (coordinates A7) which will take the place of the signal from the oscillator. Thus the terminal and microprocessor may be operated at a slowed rate to facilitate signal-tracing. The only two areas of operation which will not work correctly in this mode are the video screen and the UART if the VT50 is attached to a host device which is expecting transmission to occur at the usual rate of speed.



## CHAPTER 6

### MICROPROCESSOR INSTRUCTIONS

Each cell of the ROM contains a number, and each number has a meaning. That is, in its turn, the number inside each cell will, because of what number it happens to be, cause a particular action—such as a transfer of information, a test of conditions within the machine, or an activation of the display. The instruction set is the pairing between numbers and actions. The instruction decoder is the circuit whose inputs come from the ROM and the synchronization signals, and which sends signals to appropriate parts of the VT50 at certain times depending on the number which came from the ROM.

\*We assume you have Integrated-Circuit experience.

\*Section 9 describes the microprogram.

\*Section 4 gives an overview of the data structures within the VT50 which instructions affect.

#### LOCATION

Instruction decoding takes place on the Data Paths module.

#### THE INSTRUCTION SET

An instruction-word from the ROM has 8 bits; we will designate them bits A through H. An instruction-word is shown thus:

ABCDEFGH

—where A through H can have the values 0 or 1. The action taken by the VT50 will depend on the values of A through H.

#### IF BIT A=0

If bit A is zero, bits B, C, and D select one of eight instruction subsets. In each of these subsets, if bit E is 1, a certain action will take place at time TE; if bit F is 1, a certain (different) action will occur at time TF; if bit G is 1, another specified action will occur; if bit H is 1, a certain flag (condition within the VT50) will be tested, and a jump may occur. One word of ROM may cause one or more of these actions to take place by having one or more of bits E through H a 1. If a word is to cause more than one action to take place, these actions must be in the same subset—they must require the same number in bits B, C, and D.

If bits E through H are all 0, then another action takes place instead of any of the four mentioned above.

## TWO-WORD INSTRUCTIONS

An instruction in which A=0 and H=1 is a test instruction. Regardless of whether the test condition is satisfied or not, the next sequential ROM location is considered to contain not another instruction, but the jump address for the test. If the test condition is satisfied, the number in this location is loaded into the 8 least significant bits of PC. The next instruction will be taken from that address. The two most significant bits of PC are not affected; the jump is to a spot on the same page as the next instruction would normally have been taken from, unless an IROM has been executed since the last branch.

If the test condition is unsatisfied, the ROM location containing the jump address is skipped and the next instruction is taken from the ROM location following that location.

The following is a summary of the instructions possible when bit A=0:

ABCD	EFGH	Action taken and (abbreviation)
0000	0000	Set Cursor Flip-flop (SCFF)
	1xxx	Clear both the X and Y registers (ZXZY)
	x1xx	Decrement both the X and Y registers (DXDY)
	xx1x	Load AC with the contents of the RAM (M2A)
	xxx1	(in mode 0) Reserved for printer option (in mode 1) Jump if UART has received a character (URJ)
0001	0000	Set Video Flip-flop (SVID)
	1xxx	Complement eight-bit of X register (XB)
	x1xx	Increment AC (IA)
	xx1x	Store AC into RAM (A2M)
	xxx1	(mode 0) Jump if at a tab stop (TABJ) (mode 1) Jump if AC=RAM (AEMJ)
0010	0000	Load Y Buffer into Y register (B2Y)
	1xxx	Increment X; decrement Y (IXDY)
	x1xx	same as IA (IAI)
	xx1x	Load RAM into UART for transmission (M2U)
	xxx1	(mode 0) Jump if key-click (KCLJ) (mode 1) Jump if AC<RAM (ALMJ)
0011	0000	Complement Bell Flip-flop (CBFF)
	1xxx	Increment X register (IX)
	x1xx	Increment Y register (IY)
	xx1x	Load octal 40 into RAM (L40M)
	xxx1	(mode 0) Jump if 60 Hz line freq. (FRQJ) (mode 1) Jump if AC not equal to X register (ADXJ)
0100	0000	Clear Cursor and Video Flip-flops (ZCAV)
	1xxx	Clear AC (ZA)
	x1xx	Decrement Y register (DY)
	xx1x	Load X register from RAM (M2X)
	xxx1	(mode 0) Reserved for printer option (mode 1) same as AEMJ (AEM2J)

ABCD	EFGH	Action taken and (abbreviation)
0101	0000	Reserved for printer option
	1xxx	Set mode Flip-flop = 1 (M1)
	x1xx	Increment ROM bank (IROM)
	xx1x	Store character just received in RAM (U2M)
	xxx1	Jump (TRUJ)
0110	0000	Reserved for printer option
	1xxx	Clear X register (ZX)
	x1xx	Decrement X register (DX)
	xx1x	Load Y Buffer from RAM (M2B)
	xxx1	(mode 0) Jump if UART is transmitting (UTJ) (mode 1) Jump if Video Scan Flag (VSCJ)
0111	0000	Reserved for printer option
	1xxx	Set Mode flip-flop = 0 (M0)
	x1xx	Decrement AC (DA)
	xx1x	no-op
	xxx1	(mode 0) Jump if not top of screen (TOSJ) (mode 1) Jump if key is not depressed (KEYJ)

"x" means that the specified action will occur regardless of whether that bit is 0 or 1. If all bits marked "x" are 0, the corresponding action is the only one taken; instructions may be combined, as follows:

IA (0001 x1xx) and A2M (0001 xx1x) may be combined. The binary code for the combined instruction IA!A2M is 0001 0110. In this example, bits A through D are 0001 to select the instruction subset containing IA and A2M. Bit F is a 1 to cause an IA to be performed at time pulse TF. Bit G is a 1 to cause an A2M to be performed at time pulse TG. Since TG occurs after TF, this composite instruction means, "Add one to AC and store the incremented value in the RAM."

Bits E and H need not be 0; however, if bit E were a 1, an X8 instruction would be performed at time pulse TE. If bit H were a 1, one of the two tests (TABJ and AEMJ) would occur, following all other activity for this instruction.

Some instructions require that bits E through H be 0. For this reason, they cannot be combined with any other instruction as described above.

Instructions from different instruction subsets—that is, instructions in which bits A through D would have to take different values at the same time—likewise cannot be combined.

## IF BIT A = 1

If bit A of the instruction word is a 1, the Mode Flip-flop determines the action taken. If the Mode Flip-flop is 0, the seven least significant bits of the instruction (the constant BCDEF GH) is loaded into the RAM.

If the Mode Flip-flop equals 1, the following algorithm is carried out:

The AC is incremented and then compared to the contents of RAM. If  $AC < RAM$ , no further action is taken. If  $AC > RAM$  and the Done Flip-flop is 0, the BCDEFGH portion of the instruction is stored in RAM and the Done Flip-flop is set to 1. The Done Flip-flop gets reset to 0 by each instruction in which bit A=0.

This is to say that, for a string of instructions in a row in which A=1 (and assuming the Mode Flip-flop=1), the first instruction which causes the AC to equal (or exceed) the RAM will cause a constant to be loaded into the same RAM location.

(NOTE: In this form of the load RAM instruction—with the Mode Flip-flop=1—if the CONTROL key is depressed at time the instruction is executed and if conditions are correct for a constant to be loaded into the RAM, that constant will be 00DEFGH, rather than BCDEFGH.)

Example: Assume  $AC=22$ ,  $RAM=25$ , and Mode=1.

Executing the ROM word "1xxxxxxx" (where the values of the bits marked "x" are irrelevant) causes AC to assume the value 23.

After executing another "1xxxxxxx", AC will equal 24.

If the next instruction to be executed is 1BCDEFGH, its execution will cause

- the AC to reach the value 25.
- a constant to be loaded into the RAM, replacing the number 25. If the CONTROL key is now depressed, the constant will be 00DEFGH; otherwise it will be BCDEFGH.
- the Done Flip-flop to be set to 1.

If the next instruction is 1xxxxxxx, the AC will contain the value 26, but no constant will be loaded into the RAM because the Done Flip-flop is 1.

In this way, a conversion table can occur in the middle of a program segment. Such a table is used to convert keyboard-identification codes to ASCII codes.

## INSTRUCTION DECODING

The circuits marked "E", "F", and "Q" are decoders. When they are enabled (by a low logic level to the D input, pin 12), exactly one of the outputs will go low, depending on the state of inputs A, B, and C. When disabled, all outputs are high.

Chip "E" (E33) is enabled by three things happening simultaneously (gate E12, pins 1, 2, and 13):

1. Bit E from the ROM is a 1.
2. Bit A is a 0.
3. Time pulse TE is active (high).

The inputs marked A, B, and C on E33 come from bits B, C, and D of the ROM outputs. This agrees with what was said previously: in instructions in which bit A=1, bit E being 1 will cause a certain action to be performed at time state TE. The action is selected by bits B, C, and D of the ROM instruction word.

Chip "F" (E39) does the same thing if bit A=0 and bit F=1, at time state TF. Chip "Q" (E47) also does similar decoding for instructions in which bits A, E, F, G, and H are all 0.

Chip "G" (E43) is a dual decoder. The instructions M2A, M2X, M2U, and M2B are identified by the top half of this chip. The appropriate outputs will go low at time TH. The instructions A2M, U2M, and L40M, which cause the RAM to be loaded with something, are decoded by the lower half of E43. When the appropriate output goes low, one or both of the signals MUX A and MUX B go high. This half of E43 remains enabled for the entire time of the instruction, unlike the others, which are enabled only at a certain time-state. These signals are the control inputs to a multiplexer which selects the correct source for input to the RAM. A low signal from the bottom half of E43 also causes E2 (coordinates B3) to go to the 1 state when it is clocked at time TW. This sends the signal to the "write" inputs of the RAM. This signal is removed when the flip-flop is reset at time TG. Whatever was coming in through the multiplexer at that time—signals from either AC, the UART, or the constant 40—is written into the RAM. Chip E34 is the flag selector. Depending on the status of the select inputs (pins 11, 13, 14, and 15) one of the input signals shown on the left side of the chip will determine the output signal at pin 10. This chip will select one of the sixteen flags, depending on bits B, C, and D of the instruction, and the status of the Mode flip-flop. To change modes is to cause one of the different set of eight flags to be chosen.

Flip-flop E16 (located to the right of the flag decoder on the diagram) is normally held high by a signal to pin 4. This preset signal is removed when a flag-testing instruction is executed.

If the flag that E34 selects is low, this low signal is clocked into the E16 flip-flop, which causes PC to be loaded with new value.

At coordinates C2 on the diagram is the Mode Flip-flop. Notice that when this flip-flop, as drawn, is in the "1" state (pin 6 high; pin 3 low), the processor is in Mode 0.

Also at coordinates A3 is the bell circuit. When a CBFF instruction is decoded by chip "Q", the output from that chip labeled "KEY CLICK" goes low. This signal clocks a logical 1 into flip-flop E16. The relay closes, making a noise. But its closing clears the flip-flop, which releases it. The Done flip-flop, used in conditional Load RAM instructions, is located at coordinates C8.

## TROUBLESHOOTING TIPS

Malfunctions of some of the more self-explicit microprocessor instructions can be easily traced to this circuit: For instance, unilateral failure of the "bell" relay to click at any time would clearly be caused by a failure in the logic that decodes the CBFF instruction. Lack of any visible video might be caused by faulty decoding of the SVID instruction, and so on.

But for most of the firmware-oriented instructions, the problem might not be as evident. For instance, if an IA (increment AC) instruction fails to produce the correct signal at the outputs of the decoders, complete discoordination of the terminal may result, since the microprogram does



not use IA once for a simply-pinpointable effect, but many times to do many things. Faulty decoding of IA instructions might not be distinguishable from faulty decoding of many other instructions: both situations would result in a totally useless terminal.

However, since the microprocessor governs virtually all synchronized and "intelligent" features of the VT50, the firmware may be an indirect source of any general failure of any feature to perform correctly. As well as the possibility of instructions not being decoded properly, other sources of firmware failure are: failure of the ROM's to contain what they're supposed to contain, and failure of the microprocessor synchronization pulses (TE, TF, etc.) to fire when they should.

## CHAPTER 7 KEYBOARD

The keyboard is the input device of the VT50. It is a panel of switches, arranged to represent a typewriter keyboard. The operator pushes certain keys in a certain sequence to transmit information. Which keys, as well as the order in which they are depressed, must be recorded by the VT50. The operator may push more than one key at the same time. This is a situation which the VT50 must be prepared for.

\*We assume you have Integrated-Circuit experience.

\*Section 9 contains more details about how the microprocessor asks for information on the status of the keyboard.

### LOCATION

The location of the keyboard is evident. Access to the module containing the keyboard is gained by unscrewing the ROM, UART & Timing module from the shell, unplugging the keyboard cable from the terminal-side of the ROM module, and unplugging the ROM module from the Data Paths board. The keyboard module may then be unscrewed and removed.

### WHAT IT DOES

When the Mode flip-flop is set to 1, the microprocessor instruction 0111 xxx1 (for any xxx-abbreviated KEYJ) is a keyboard test instruction. If the key corresponding to the number in AC is not depressed, control is passed to the ROM address contained in the next sequential ROM location: that number is loaded into the 8 least significant bits of PC. If the keyboard key whose number is in AC is not depressed, the next instruction is executed. (The ROM location containing the jump address is skipped.)

Each key on the keyboard has an identification number, except for CONTROL and BREAK, which make their differences independently of, and cannot be tested by, the microprocessor. The highest identification number is octal 67; the lowest is 01. If the 6 least significant bits of AC contain a number from 70 to 77, KEYJ will never cause a jump; if they contain 00, KEYJ will always jump. The most significant bit of AC is irrelevant to KEYJ.

Thus, the microprocessor can test the status of the keyboard by loading 67 into AC and entering a loop whose only outlet is KEYJ, decrementing the AC each time through that loop.

After KEYJ sends the microprocessor out of this loop, AC will contain the code for the depressed key—or 00 if no key was depressed.

## HOW IT DOES IT

Assume the 6 least significant bits of AC contain the octal number pq. Chip E1 is a 7442 Decoder. Depending on the number input to this chip, exactly one of its outputs will go to a low logic level; the rest will remain high. The number q from AC is the input to this chip; depending on it, a signal will be sent to one row of keys.

Chip E5 is a 74151 Selector. Depending on the number at the select inputs (pins 9, 10 and 11), the output will assume the state of the corresponding input, F0 through F7.

Connected to these selector inputs are the outputs of NAND gates, which will go to a high logic level if any of its inputs are at a low level. Two things have to happen for an input to go low:

1. A key must be depressed.
2. The decoder must be sending a low signal through the row on which the depressed key is located.

What the selector selects, then, is one of the "columns" of keys. And the channel-selecting input to the selector is the p portion of the number in AC.

Example: Suppose a KEYJ is executed with the number 23 in AC. The input to the decoder is therefore 3, which means only output F3 will go low. The channel-selecting input to the selector is 2, which means that the output of the selector will vary with input F2. There is only one key which at any time has a low-logic-level signal sent to one side of it by the decoder, and for which that signal crossing to the other side of the switch is detectable by the selector--there is only one key at the intersection of any logical "row" and "column". In this case, that key is the E key. The output from pin 6 of the selector will be low if the E key is depressed; high otherwise. The E key was selected by the number 23 in AC. The 3 caused a signal to be sent through row 3; the 2 caused that signal to be sought coming down column 2.

## CHAPTER 8

### CHARACTER AND CURSOR GENERATION

#### WHAT IT DOES

The character generator converts the contents of RAM, the ASCII codes for the characters to appear on the screen, to the signals which, painted as dots on the screen, take the form of the appropriate characters.

At any moment, as it is wired into the VT50, the character generator's outputs consist of one row of the dots which comprise one character. The character is selected by the RAM outputs. If, at the address selected by the X- and Y-registers, RAM contains the ASCII code for S, the outputs of the character generator will be one of the rows of dots which, when all eight rows are painted above each other, will form an S.

Which row is output is selected by the three least significant bits of AC. A 000 in the three least significant bits causes the top row of the character to be output; a 111 causes the bottom line to be output.

The video shift register, on signal, is parallel-loaded with the character-generator outputs. Then, on each oscillator pulse, one bit is shifted out, and becomes (through an inverter) the signal VID Z H, which goes to the monitor board to be amplified and output to the screen. This allows each output from the character generator to be present at the output VID Z H, one at a time, for 72 nanoseconds each. Since the electron beam is traveling from left to right across the screen, this causes the five character-generator outputs to appear on the screen next to each other in a horizontal line. Since the Video Shift Register is loaded on every ninth oscillator pulse, at four oscillator-times of the nine, the output at VID Z H will not have come from the character generator. This accounts for the space between characters.

When the VSR is not enabled to be loaded from the character generator, logical 1's are shifted into and through it. This causes VID Z H to remain low, and no dots to be painted on the screen.

However, when a cursor (horizontal line) is to be painted, logical 0's are shifted through the VSR.

#### THE VIDEO PROCESS

The video process is enabled by a SVID microprocessor instruction, and disabled by a ZCAV instruction. During the video process, except during the time that the electron beam is traveling back to the left side of the screen ("horizontal flyback"), the X-register is incremented and the VSR is parallel-loaded from the character generator every ninth oscillator pulse (twice each instruction-time). This process automatically references RAM cells which represent the contents of a line on the screen. The contents of each RAM word is passed to the character generator. Signals for the dots comprising each character are, in turn, loaded into and shifted out of the VSR.

During horizontal flyback, the VSR is not allowed to be loaded. Therefore, in shift-mode, 1's are shifted through and no light appears on the screen. Loading of the VSR is also disabled when the Cursor Flip-flop is set.

During one scan, thus, one row of dots for each of 80 characters is painted on a horizontal line across the screen. Between scans, (during horizontal flyback), the microprocessor:

- a. Sets the automatically-incremented X-register back to zero.
- b. Increments the AC, to cause the next row to be painted on the next scan.
- c. Tests AC to see if all eight rows of dots have been painted for that line of text.

During scans, the microprocessor goes into a waiting loop which tests the Video Scan Flag. This flag becomes false when horizontal flyback begins, and the microprocessor leaves the loop. Ten instruction-times later, the next scan will begin.

## THE CURSOR PROCESS

The cursor process is enabled by a SCFF microprocessor instruction, and disabled by a ZCAV instruction. The Video Flip-flop must be set for the cursor process to work properly, since it causes the X-register to be incremented automatically. SCFF disables the parallel-load feature of the Video-Shift Register, so that 1's are shifted through the VSR. The X-register is automatically incremented. When the contents of the X-register and AC are equal, 0's are shifted through the VSR. These 0 signals, when they reach the screen, appear as the horizontal line.



## CHAPTER 9 THE MICROPROGRAM

### WHAT IT DOES

The microprogram coordinates all the data transfer and processing which takes place in the VT50. This can be summarized as follows:

1. Refreshing the video screen with the contents of RAM 60 times each second.
2. Checking the UART for reception of a character from the host at least 960 times each second.
3. Moving incoming data to the RAM.
4. Taking special action upon receipt of Escape Sequences and control codes.
5. Checking for the existence of depressed keyboard keys.
6. Moving the corresponding ASCII codes to the UART for transmission to the host in the correct order.

In addition, the microprogram directs the advanced features of the VT50, such as saving information in a silo when in Hold Screen Mode, and automatically transmitting XOFF and XON requests to the host. Section 1, which explains the data structures available within the VT50, and Section 7, which details the repertoire and thus is a description of the machine language and the symbolism used herein, are prerequisites of this section.

This section contains a listing of the most recent version of the microcode. Please note that, since it was assembled through the PDP-11 assembler, all the ROM addresses are even. The addresses specified in this section correspond to the addresses in that listing; but both addresses are twice what the PC will contain at execution time. Addresses 0000-0776 correspond to locations in the first ROM bank; addresses 1000-1776 are from the second bank. For example, the word whose address is listed as 0452 is contained in the word, in the first ROM bank, addressed when PC=0225.

### MICROPROGRAM FLOWCHART

Figure is a chart showing the tasks that the microprocessor supervises. Note that some functions are performed after every frame is painted on the screen, and some are performed after every line is painted.

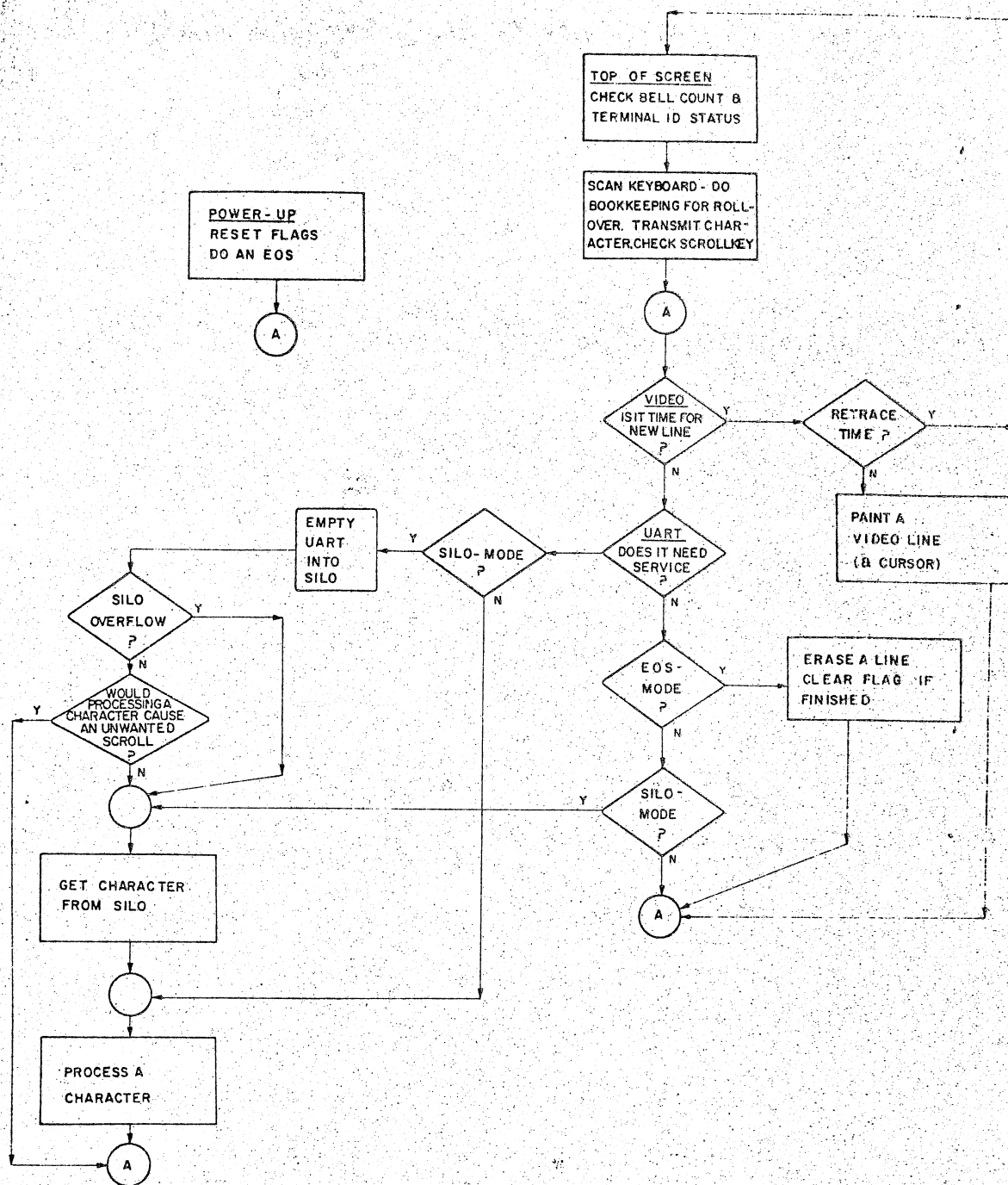


Figure 9  
Microprogram Flowchart

The "scratch pad" portion of RAM is accessed whenever the Y-register contains a number greater than 13. The description of the RAM address mapping which is contained in Section 4 shows why only<sup>8</sup> the 4 least significant bits are used in the address computation in this case. We can draw a picture of scratch-pad memory, with four rows, one of which is selected by whether the Y-register contains 14, 15, 16, or 17; and with 16 columns, one of which is selected by the contents of the 4 least significant bits of the X-register; in octal, 0-17. Each of the cells, which represents a 7-bit RAM word, contains a description of its use. These descriptions are elaborated on, while and after several underlying concepts are explained. The instruction ZXZY!DXY is sufficient to access the word labeled "scan count/KB counter" in the figure, since it sets both the Y-register and the 4 least significant bits of the X-register to 17. Then the IX, IY, DX, DY, and X8 instructions can be used to move around in the scratch pad. If these instructions are used such that you go off the right or left side of the chart, you will be addressing the word clear back on the other side of the row. The diagram might better be printed on a circular ring than a flat page. However, if the Y-register is modified to send you off the top or bottom of the diagram, you enter the portion of RAM used as the screen image.

Y=14	S I L O													
15	SCROLL COUNT	EOS LINE	CUR X	FRAME COUNT							SILO OUTPUT	SILO INPUT		
16	B  STORE KEYBOARD ADDRESSES	HOLD SCREEN FF	CUR Y	TOP LINE							SCREEN LINE	EOS FF	SILO COUNTER	
17	A	BELL FF	SCAN COUNT / KB COUNTER	ID	VIDEO LINE	LINE COUNT					ESC FF	INCOMING CHAR / CUR-FF	LF- BUFFER	
	X=14	15	16	17	0	1	2	3	4	5	6	7	10	11

When the VT50 receives a LF character while the cursor is already at the bottom of the screen, a Scroll is performed: all the lines on the screen move up one position, with the information on the top line being lost, and a "new" blank line appears at the bottom of the screen.

screen position	before scroll	after scroll
0	A	B
1	B	C
2	C	D
3	D	E
4	E	F
5	F	G
6	G	H
7	H	I
10	I	J
11	J	K
12	K	L
13	L	new line of blanks

However, if the contents of each RAM line (all RAM words accessible with a certain number in the Y-register) were physically moved to a new RAM location, the process would be too time-consuming to be feasible. In reality, the information "B" through "L" in the above diagram stays where it was in RAM, and the RAM line which contained the information "A" is cleared to form the same line of blanks which is then displayed at the bottom of the screen. Viewed in terms of RAM line rather than screen line, the process of scrolling appears thus:

#### RAM line

(contents of Y-register)	information contained	
	before scroll	after scroll
0	*A	blanks
1	B	*B
2	C	C
3	D	D
4	E	E
5	F	F
6	G	G
7	H	H
10	I	I
11	J	J
12	K	K
13	L	L

\*"this line is to be displayed at the top of the screen"

The information transfer is much more efficient, but a new counter is needed - one which does what the asterisk in the above example does. The scratch-pad cell which serves this purpose is the word TOS LINE. It contains a number from 0-11 which is the RAM line which contains the information to be displayed at the top position on the screen. It is incremented (modulo 12) after each scroll. At system initialization, it contains 0, indicating that the line of RAM accessed when a 0 is in the Y-register contains the information to be displayed at the top of the screen.

Although this counter is present, the microprocessor does not have the capability of direct addition, so for many purposes, two counters must be maintained, an absolute and a relocatable counter - that is, one to identify a row of RAM words that have a certain property, and one to identify a line on the screen that has that property.

CUR X and CUR Y contain the address coordinates for the current cursor position. CUR Y points to a RAM address; however, SCREEN LINE points to the line relative to the top line on the screen. If the cursor is at the bottom line on the screen, SCREEN LINE will always contain the value 13 (octal); CUR Y will contain the Y coordinates of the RAM words which contain the bottom line's text.

During a video scan, VIDEO LINE will contain the Y-register contents for the line currently being painted, while LINE COUNT will contain the screen-line number. LINE COUNT will always start at 0 and go to 13.

During processing of an ESC J command ("Erase text to end of screen"), EOS LINE will contain the number of the RAM line being erased, while EOS FF will contain the number of the

line on the screen being erased. This word, although it is a counter, is called a "flip-flop" since the only time it is used is to determine whether the entire screen has been erased. Thus, EOS FF has only two relevant states.

$\text{EOS FF} < 14$  means another line must be erased:

$\text{EOS FF} \geq 14$  means the bottom line has been erased.

## SCAN COUNT/KB COUNTER

The word addressed by having 17 in both the Y-register and the low part of the X-register, since it is one of the easiest scratch-pad locations to gain access to, serves a dual purpose. During the painting of the screen, it is used as a scan counter. This counter is initialized to 17, and incremented each time the SYNC flag is received. The counter goes from 20-27 in the time that a line is being painted. A copy of this count is kept in AC, where the three least significant bits, which are 0-7, go to the character generator. When the counter reaches 30, the electron beam is on the scan on which a cursor may have to be painted. If the position of the cursor is on the line that was just painted, the registers are set up, and an SCFF is executed. When the count goes to 31, the screen is disabled for 11 scans with a ZCAV instruction. When the count reaches 42, it is reset to 17 if another line is to be painted.

During vertical retrace, the same location is used to interrogate the keyboard.

## FLIP-FLOPS

ESC FF is a word which indicates that the terminal is in Escape Mode. Any incoming character is then treated as part of an Escape sequence, which means that special actions are taken. When the terminal is not in Escape Mode, ESC FF is set to 0. HOLD SCREEN FF is set to 1 when the terminal is in Hold Screen Mode. Incoming text will be stored in the silo rather than sent to the screen image portion of RAM to be displayed, unless the silo would overflow. When HOLD SCREEN FF contains anything else the terminal is not in Hold Screen Mode.

BELL FF begins at zero. Every time a CTRL G is received or the cursor is moved into column 73, BELL FF is set to octal 25. After each frame is painted, BELL FF is tested. If positive, the relay is clicked once and BELL FF is decremented by 1. If zero, the terminal is silent. The net effect is that the relay clicks 21 times at 60 Hz. This is the terminal's audible warning signal. The same noise would be achieved if a typist could type at 60 keys per second. The counter is called BELL FF because, once again, it assumes only two states which are relevant to the microprocessor: positive and zero.

## SERVICING THE KEYBOARD

The keyboard is interrogated once each frame (i.e., with the same frequency as the line current) to discover if keys are down. Section 7 outlines the mechanics of keyboard interrogation; we summarize here by saying that each key except CONTROL and BREAK has an identification



number, and there is a "pseudo-key" with the number 0 which appears to be always depressed. When an operator presses a key, it will generally remain pressed for more than one scan. But this action must not therefore generate more than one character. A character is transmitted only on the scan on which that key has just been pressed.

The microprocessor uses the scratch-pad words KEYBOARD A and KEYBOARD B to ensure this. These two locations contain zero in their idle state. When a key is pressed, the keyboard code (not the ASCII code) is stored in one of these two locations. If a key whose code is stored in KEYBOARD A or KEYBOARD B is found to be down on subsequent scans, it causes no further transmission to take place.

If the number N is in AC, the following microcode will cause AC to contain the highest number  $< N$  for which the corresponding key is down.

```
TEST:DA!KEYJ
      .AD TEST
```

AC will contain 0 if no key with ID number  $< N$  is down.

The microprogram uses the following logic. The numbers in parentheses are ROM addresses which contain that part of the logic.

1. Are the keys whose codes are contained in KEYBOARD A and KEYBOARD B still down? If not, clear the corresponding word. (1646-1672)
2. If the UART isn't ready to transmit another character, don't even bother checking the keyboard. (1676)
3. Otherwise, starting at 67, go through the keyboard to find a key whose ID number is not zero, and is not contained in KEYBOARD A or KEYBOARD B (a new key). If at least one of those two locations contains zero, store the key's ID therein, convert it to ASCII and transmit it. If both KEYBOARD A and KEYBOARD B are non-zero, then at least two other keys are already down. The key just identified as being depressed cannot be processed and transmitted, since no RAM space is allocated to remember that it was transmitted, and, if the key is down 1/60 second later when the keyboard is again interrogated, the character would be transmitted again (1100-1366).

## THE SILO/HOLD SCREEN MODE

The silo portion of RAM is accessed whenever the Y-register contains 14. When the terminal is in Hold Screen Mode, it attempts to avoid scrolling the display. LF (Line Feed) characters are not processed. Instead, the first LF received and all characters following it are routed to the scratch-pad portion of RAM. The LF is stored in the RAM cell labeled LF BUFFER; the subsequent characters go into the silo. XOFF is sent to the host, to request that it stop sending. If the silo is about to overflow or if since the host has not stopped sending, the SCROLL key is pressed, one line (the LF and all characters following it, up to but not including the next LF) is processed. If, in processing this line, the silo is depleted, XON is transmitted, and the remainder of the line will be taken directly from the host.

The operator presses SCROLL with the shift key down to get a new screenful of data. In this situation, the terminal will process all the text in the silo and then from the host up to the 13th LF before sending XOFF and diverting incoming characters to the silo.

The count of how many more scrolls are allowed before display must be suppressed is contained in the scratch-pad location SCROLL COUNT.

Space in the silo is allocated dynamically. The silo utilizes the "ring" characteristics of the buffer, in that, it takes only one DX instruction to get from the "leftmost" location to the "rightmost". SILO OUTPUT contains the X-coordinate of the silo location containing the character that will be processed next. SILO INPUT contains the X-coordinate of the first free silo location. The 3 high-order bits of SILO INPUT and SILO OUTPUT are irrelevant and need not be 0. SILO COUNTER contains the difference between these two X-coordinates. It represents the number of characters currently being held in the silo. When the silo is empty, SILO INPUT = SILO OUTPUT and SILO COUNTER = 0. LF BUFFER is used as a flip-flop. It has only two significant states: 012, the code for LF, when one is pending; 0 otherwise. As an example, assume SILO INPUT = SILO OUTPUT = 5 and SILO COUNTER = 0. This indicates that the silo is empty. Assume also that the terminal has just been put in Hold Screen Mode. Incoming characters are processed normally until a LF is received. That LF, which would cause a scroll if processed, must go into LF BUFFER. If the host then transmitted the character W, it would go into the silo rather than the screen-image portion of RAM. It is stored at RAM coordinates Y=14, X=5. As the first character to enter the silo, it causes the VT50 to transmit the XOFF code. Assume the host transmits three more characters — a question mark, a carriage return, and another line feed before processing the XOFF and ceasing transmission. Those characters are stored in the silo in the following places:

low-order 4 bits of X register		4	5	6	7	10	11
Y register 14	Contents of		W	?	CR	LF	

The 4 least significant bits of SILO INPUT contain 11 now; SILO COUNTER contains 4. Now assume the operator depresses the SCROLL key. This increments SCROLL COUNT, which tells the processor that another scrolling is allowed. SCROLL COUNT will contain 1. The LF in LF BUFFER will now be processed, causing a scroll to occur. SCROLL COUNT will be decremented; LF buffer will be set to zero. The W and ? will be displayed on the screen. The carriage return will then move the cursor to the left side of its present line. The line feed, although removed from the silo, cannot be processed, since the operator has not authorized a second scrolling to occur—since SCROLL COUNT contains zero. So it will be placed back in LF BUFFER. As the last character to leave the silo, it causes the terminal to send XOFF to the host. The characters in the silo can then be processed from left to right. SILO OUTPUT will be incremented to contain 11; SILO COUNTER will be decremented to contain 0.

SCROLL COUNT contains the number of LF's which may be processed. When the SCROLL key is pressed, SCROLL COUNT is incremented. When SCROLL is pressed with the shift key down, SCROLL COUNT is set to twelve.

The microcode to control the silo begins at 0330.

## TERMINAL IDENTIFICATION

When the terminal receives ESC Z, it transmits ESC / A to identify itself as a VT50 without copier. The problems inherent in this kind of an operation stem from the fact that those three characters cannot be transmitted simultaneously, and they must not be interspersed with characters typed in from the keyboard.

The VT50 identifies itself by transmitting one of the three characters after each of three consecutive frames are painted. (If, due to a low transmission speed — under 600 baud — the UART is not ready to transmit the next character at the end of a frame, the microprogram goes on, checking the UART again after the next frame). The keyboard is ignored until all three characters are sent, a period nominally 1/15 of a second, and at most (at 110 baud) 2/5 of a second.

The scratch-pad location entitled ID is used to identify the stage in the identification sequence presently being carried out. ID takes five characteristic values:

- 0 - An ESC Z command is not pending.
- 135 - An ESC Z has been received, but no action has been taken.
- 33 - ESC has been sent.
- 57 - / has been sent.
- 101 - A has been sent.

ID is set to 135 as soon as an ESC Z is received and identified, typically in the middle of a frame, between the painting of two lines. It assumes its other three nonzero values when, at the end of a frame, the UART is discovered to be ready to transmit, and loaded with one of the three characters in the identification sequence. The value of ID tells the microprogram which character of the three is currently due to go out. In fact, the ASCII code transmitted remains in ID as the marker of completion of that stage. When ID contains 101 and, at the end of a frame, the UART's transmitting section is ready, ID is set back to zero to indicate that the identification sequence has been completed and the keyboard can now be interrogated. See locations 1710-1734 for the portion of the microprogram which performs the identification.

## FLASHING THE CURSOR

FRAME COUNT is a scratch-pad location which is incremented each time the line the cursor is on is painted. When overflow occurs, the cursor is painted and FRAME COUNT is reset to 160. Thus the cursor is painted with every 21st frame.

## POWER-UP

At system initialization, rows 17 and 16 of the scratch-pad are set to 0, and row 15 is set to 40 (the ASCII code for a space) except for the location CURSOR X, which is set to 0. This has the following implications:

A 40 in FRAME COUNT causes a delay of nearly two seconds to occur before the first cursor line is painted, which will happen when the count overflows.

A 0 in EOS F/F causes the screen to be erased.

The terminal is not in Escape or Hold Screen Mode.

The following are important considerations when planning to modify the microcode:

1. Deadlines must be met. If the UART is not checked every 1/960th of a second, incoming characters may be lost. If the position of the scanning beam is not kept track of, either by prior calculation or by waiting for the SYNC Flag, or if a routine takes up so much time that the SYNC Flag is missed, the spacing between lines may not be adequate or constant. If a routine likewise causes a TOS Flag to be missed, one frame will be skipped (not painted) while the microprocessor waits for the next TOS flag. Painting the screen at 30 Hz instead of 50 or 60 will result in a visible flickering.
2. The Mode flip-flop must be correct. Being in the wrong mode changes the meaning of all the conditional branches. This can have disastrous effects, as a simple loop as this one, which waits for the scan to return to the top-of-screen position:

```
LOOP: TOSJ  
      .AD LOOP
```

becomes, in the other mode, a loop which will effectively shut down the terminal until a certain key is pressed:

```
LOOP: KEYJ  
      .AD LOOP
```

Load RAM instructions (instructions in which bit A=1) also have distinctly different meanings when the microprocessor is in the wrong mode.

[illegible]





## VT50/VT50H TECH-TIP

A new revision uart module (54-10902), is now being shipped in the VT50 and VT50H and in SPARES KITS. The keyboard cable which plugs into this new REV. module must now be connected in such a way that the color code on the cable, (which follows the standard resistor color code; black, brown, red, orange, etc). Starts its sequence from the REAR of the VT50; exactly opposite that of older REV. uart module.

The new REV. board can be easily identified by a row of straight pins running parallel to the terminal strip and a group of straight pins located on the right side of the module above and to the left of the large uart chip.



ADDENDUM

TO

VT50 INSTALLATION AND MAINTENANCE GUIDE

INTENDED FOR USE IN DEPOT REPAIR

Voltage Check Points

Note: Connectors and pins are referenced with VT50 unit upside down and viewed from the rear. Numbering sequence from left to right.

Power Supply/Monitor board connection with Datapath board.

LEFT CONNECTOR

Check Pin for Volts

2	-12
3	- 5
4	+ 5 SENSE
5	+ 5
6	+ 5
7	GROUND SENSE

RIGHT CONNECTOR

Check Pin for Volts

1	+ 5
2	GROUND
10	+15

The two bottom boards, the Datapath and ROM/UART board, interconnect using a total of 77 pins arranged in two sections - 55 pins on left half and 22 pins on right half of boards.

Pins are numbered consecutively from left to right (again viewed from the rear with unit upside down) from No. 1 through No. 77.

Check Pin for Volts

31	-12
36	(POWER OK L)
39	+15
42	+ 5
43	+ 5
44	GROUND
45	GROUND



# APPENDIX B MICROPROGRAM LISTING

VT50 SYMBOLS AND MICRO-PROGRAM MACRO V06-03 27-JAN-75 13:32 PAGE 2

1	000000	START:	AD	UART	
2	000002	000132	M1:U2M		;NO-OP AT POWER-UP TIME
3	000004	000010	ZXZY		;CLEAR UART AND GET IN MODE 1
4	000006	000160	ZY		
5	000010	000114	ZA:DY		;GET LAST ROW IN RAM
6	000012	000022	AZM		;CLEAR THE LOCATION
7	000014	000114	ZA:DY		;GET NEXT ROW
8	000016	000022	AZM		
9	000020	000114	ZA:DY		
10	00022	000073	IX:IL40M:ADXJ		;GET NEXT COLUMN
11	00024		AD	PULF	;DO ANOTHER COLUMN IF NOT FINISHED
12	00026	000014	ZXZY:DXDY		;CARRIAGE RETURN ROUTIN
13	00030	000114	DY:ZA		
14	00032	000114	DY:ZA		;GET CUR-X LOCATION
15	00034	000023	AZM:AEMJ		;CLEAR IT
16	00036		AD	JUMP	
17	00040	000014	ZXZY:DXDY		;DO A TAB, MODULO 8...
18	00042	000050	IXDY		
19	00044	000006	DXDY:M2A		;GET CUR-X
20	00046		LD	110	
21	00050	000041	ALMJ		;BEFORE LAST TAB-STOP ?
22	00052		AD	TAB2	;YES-DO AN HONEST TO GOD TAB
23	00054	000131	M1:TRUJ		;NO-DO A CURSOR RIGHT INSTEAD
24	00056		AD	CURT2	;OTHERWISE DO CURSOR RIGHT
25	00060	000030	X8		
26	00062	000007	DXDY:M2A:URJ		;IS UART FLAG SET ?
27	00064		AD	CHPROC	;YES-DO PROCESS THE NEW CHARACTER
28	00066		LD	15	;GET ERASE-COUNTER
29	00070	000045	IAZ:ALMJ		;FINISHED ?

B-2

66	00202	000014	BUFF:	ZXZY!DXDY	;SET LF-BUFFER
67	00204	000071	IDEN:	IX!ADXJ	;SET IDENTIFIER TO TRANSMIT "ESC,\,A"
68	00206			AD	HOLD4
69	00210	000023	SCRL1:	A2M!AEMJ	
70	00212			AD	SCRL2
71	00214	000006	EOS:	DXDY!M2A	;GET SCREEN LINE
72	00216	000070		IX	;GET EOS-COUNTER
73	00220	000026		IA!A2M	;SET IT WITH INCREMENTED SCREEN LINE
74	00222	000156		ZX!DX!M2B	
75	00224	000002		M2A	
76	00226	000004		DXDY	;GET EOS-LINE
77	00230	000022		A2M	;SET IT WITH CUR-Y
78	00232	000014		ZXZY!DXDY	
79	00234	000104	EOL:	DY	
80	00236	000156		ZX!DX!M2B	;GET LINE TO BE ERASED
81	00240	000104		DY	
82	00242	000002		M2A	;GET CUR-X
83	00244	000071	5#:	IX!ADXJ	;DELAY LOOP
84	00246			AD	5#
85	00250	000131	ERASL:	M1!TRUJ	
86	00252			AD	ERASL2
87	00254	000070	CHPROC:	IX	
88	00256	000111		ZA!AEM2J	;IS SILO SET ?
89	00260			AD	CHPR2
90	00262	000002	LOADER:	M2A	;GET SILO COUNT
91	00264	000026	LOAD1:	IA!A2M	;INCREMENT IT
92	00266	000144		DX	
93	00270	000006		DXDY!M2A	;GET INPUT POINTER
94	00272	000026		IA!A2M	;INCREMENT IT
95	00274	000102		M2X	
96	00276	000004		DXDY	;GET SILO LOCATION
97	00300	000137		M1!IROM!U2M!TRUJ	;LOAD CHARACTER IN SILO
98	00302			AD	VID
99	00304	000064	CHPR2:	IX	;GET LF-BUFFER
100	0306	000111		ZA!AEM2J	;IS IT SET ?
101	0310			AD	CHPR3
					;NO-PROCESS NEW CHARACTER

102 0312	LD	23	
103 0314	M2U		; TRANSMIT XOFF
104 0316	DY:ZA:AEM2J		; UNCONDITIONAL JUMP
105 0320	AD	LOAD1	
106 0322	M1:U2M		
107 0324	IX:IY:ADXJ		
108 0326	AD	CHPR4	
109 0330	ZA:DY:AEM2J		; IS SILO SET ?
110 0332	AD	SIL3	; NO-QUIT
111 0334	IA2:ALMJ		; IS THIS LAST ITEM ?
112 0336	AD	XONSK	; NO-SKIP XON ROUTINE
113 0340	LD	21	; GET XON CODE
114 0342	M2U:ALMJ		; TRANSMIT
115 0344	AD	SIL2	
116 0346	M2A		; DECREMENT SILO COUNT
117 0350	DA		
118 0352	A2M		; GET OUTPUT POINTER
119 0354	DXDY:M2A		; INCREMENT IT
120 0356	IA:A2M		
121 0360	M2X		
122 0362	DXDY:M2A		; LOAD CHARACTER
123 0364	LD	0	; CLEAR LOCATION
124 0366	ZXZY:DXDY		
125 0370	IA:XG	41	; GET SCRATCH LOCATION
126 0372	LD		; UPPER LIMIT FOR CONTROL CODES
127 0374	ALMJ		; IS IT A CONTROL CODE ?
128 0376	AD	CONTR	
129 0400	A2M		; STORE CODE
130 0402	DX		; GET ESCAPE FF
131 0404	ZA:AEM2J		; SET ?
132 0406	AD	DISP	; NO-DISPLAY THIS CHARACTER
133 0410	A2M		; CLEAR ESCAPE FF
134 0412	IX		
135 0414	M2A		; GET ASCII CODE
136 0416	LD	102	
137 0420	AEMJ		; IS IT A ?

138 0422	AD	CURUP	
139 0424	LD	111	
140 0426 000021	AEMJ		, IS IT H ?
141 0430	AD	HOME	
142 0432	LD	104	
143 0434 000021	AEMJ		, IS IT C ?
144 0436	AD	CURRT	
145 0440	LD	135	
146 0442 000021	AEMJ		, IS IT \ ?
147 0444	AD	HOLDIS	
148 0446 000025	IA:AEMJ		, IS IT L ?
149 0450	AD	HOLDEN	
150 0452 000025	IA:AEMJ		, IS IT Z ?
151 0454	AD	IDENT	
152 0456	LD	116	
153 0460 000021	AEMJ		, IS IT K ?
154 0462	AD	EOL	
155 0464 000025	IA:AEMJ		, IS IT J ?
156 0466	AD	EOS	
157 0470 000135	M1:FROM:TRUJ		
158 0472	AD	VID	
159 0474	LD	13	
160 0476 000021	AEMJ		, IS IT LF ?
161 0500	AD	LINEF	
162 0502 000025	IA:AEMJ		, IS IT TAB ?
163 0504	AD	TAB	
164 0506 000025	IA:AEMJ		, IS IT ESPACE ?
165 0510	AD	ESPACE	
166 0512 000025	IA:AEMJ		, IS IT BELL ?
167 0514	AD	BELL	
168 0516	LD	37	
169 0520 000021	AEMJ		, IS IT ESC ?
170 0522	AD	ESCAPE	
171 0524	LD	21	
172 0526 000021	AEMJ		, IS IT CR ?
173 0530	AD	CRTN	



174	0532	000135	M1:FROM:TRUJ			
175	0534		.AD	VID		
176	0536	000144	DX	ESCAPE		;CAN BE REPLACED BY A "DX"
177	0540	000111	ZAI:AE2J	ESCA2		;GET ESCAPE FF
178	0542		.AD	CURT3		;IS IT SET ?
179	0544	000023	A2M:AE2J			;NO-SET IT
180	0546		.AD	JUMP		;CLEAR IT
181	0550	000070	IX	DISP		;GET CHARACTER TO BE DISPLAYED
182	0552	000002	M2A			;COMPENSATE FOR PREVIOUS INCREMENT
183	0554	000164	DA			
184	0556	000104	DY			
185	0560	000156	ZX:DX:M2B			;GET CUR-Y
186	0562	000106	DY:M2X			;GET CUR-X
187	0564	000040	B2Y			;GET CURSOR LOCATION
188	0566	000022	A2M			;LOAD CHARACTER
189	0570	000014	CURRT			;CURSOR RIGHT ROUTINE
190	0572	000050	IXDY			;GET CUR-X
191	0574	000006	DXDY:M2A			
192	0576		.LD	117		
193	0600	000021	AEMJ			;IN RIGHT MARGIN?
194	0602		.AD	JUMP		;EXIT-VT50 DOES NOT WRAP THE CURSOR
195	0604	000027	IA:A2M:AE2J			;INCREMENT IT
196	0606		.AD	JUMP		
197	0610	000004	DXDY	HOME		;GET SCREEN LINE
198	0612		.LD	0		;CLEAR IT
199	0614	000150	ZX			;GET TOP LINE
200	0616	000074	IX:IY			
201	0620	000006	DXDY:M2A			
202	0622	000144	DX			
203	0624	000023	A2M:AE2J			;LOAD IT WITH TOP LINE
204	0626		.AD	CRTN2		
205	0630	000004	DXDY	CURUP		;GET SCREEN LINE
206	0632	000111	ZAI:AE2J			;IS IT 0 ?
207	0634		.AD	JUMP		
208	0636	000002	M2A			
209	0640	000164	DA			

210	0642	000023	A2M!AEMJ	DECREMENT IT
211	0644		AD	DECR
212	0646	000114	DY:ZA	
213	0650	000114	DY:ZA	
214	0652	000031	X8:AEMJ	;GET CUR-X
215	0654		AD	;IN LEFT MARGIN ?
216	0656	000154	JUMP	;YES-QUIT
217	0660	000002	ZX:DX	
218	0662	000164	MZA	;LOAD CUR-X
219	0664	000041	DA	;DECREMENT IT
220	0666		ALMJ	
221	0670		AD	CRTN3
222	0672	000135	LD	13
223	0674		M1:FROM:TRUJ	
224	0676	000004	AD	VID
225	0700	000110	DXDY	
226	0702	000036	ZA	
227	0704	000030	X8:IA:A2M	
228	0706	000114	X8	;SET HOLD SCREEN FF
229	0710	000144	ZA:DY	;AND CLEAR SCROLL COUNT
230	0712	000033	DX	
231	0714		X8:A2M:AEMJ	
232	0716	000174	AD	JUMP
233	0720	000025	MO:DA	
234	0722		IA:TABJ	
235	0724	000027	AD	5\$
236	0726		IA:A2M:TABJ	
237	0730	000026	AD	JUMP
238	0732	000154	IA:A2M	
239	0734	000002	ZX:DX	
240	0736		MZA	
241	0740	000045	LD	14
242	0742		IAZ:ALMJ	
243	0744	000131	AD	CRTN3
244	0746		M1:TRUJ	
245	0750	000022	AD	ESCAZ
			A2M	;INCREMENT EDS-COUNTER

246	0752	000154	ZX!DX	;GET EOS LINE
247	0754	000006	DXDY!MZA	
248	0756		LD 14	
249	0760	000045	IAZ!ALMJ	; INCREMENT IT
250	0762		AD 5\$	
251	0764	000110	ZA	
252	0766	000022	A2M	
253	0770	000142	M2B	/LOAD LINE TO BE ERASED IN MASY
254	0772	000040	ERASL2: E2Y	;GET LINE TO BE ERASED
255	0774		LD 117	
256	0776	000002	M2A	
257	1000	000144	DX	;GET RIGHT MARGIN
258	1002	000073	IX!L40M!ADXJ	; LOOP WHILE ERASING LINE
259	1004		AD 5\$	
260	1006	000016	ZXZY!DXDY!MZA	; GET SCAN COUNT
261	1010	000026	IA!A2M	
262	1012	000027	VIDW: IA!A2M!AEMJ	; INCREMENT IT TWICE
263	1014		AD VID	
264	1016	000170	SHIFTL: NO	;GET IN MODE 0
265	1020	000252	WORD	252, 300, 242, 337, 335, 243, 245, 276, 272, 241, 250, 253, 251
	1022	000300		
	1024	000242		
	1026	000337		
	1030	000335		
	1032	000243		
	1034	000245		
	1036	000276		
	1040	000272		
	1042	000241		
	1044	000250		
	1046	000253		
	1050	000251		
266	1052	000244	WORD	244, 336, 274, 246, 277
	1054	000336		
	1056	000274		

1060	000246				
1062	000277				
267	1064	000121	TRUJ		
268	1066		AD		
269	1070	000104	DY	XMT	GET TOP LINE
270	1072	000002	MZA		
271	1074	000074	IXIY		GET VIDEO LINE
272	1076	000022	AZM		SET THEM EQUAL
273	1100	000016	ZXZY!DXDY!MZA		GET KEYBOARD COUNT
274	1102	000165	DA!KEYJ		IS THE KEY UP ?
275	1104		AD	5\$	
276	1106	000022	AZM		SAVE ADDRESS
277	1110	000111	ZA!AEM2J		IS IT ZERO ?
278	1112		AD	FINISH	
279	1114	000002	MZA		
280	1116	000144	DX		
281	1120	000004	DXDY		GET BUFFER
282	1122	000032	X8!AZM		STORE IN SCRATCH LOCATION
283	1124	000064	IY		
284	1126	000022	AZM		STORE IN BOTH SCRATCH LOCATIONS
285	1130	000031	X8!AEMJ		ALREADY TRANSMITTED ?
286	1132		AD	KEYBO	
287	1134	000105	DY!AEM2J		
288	1136		AD	KEYBO	
289	1140	000111	ZA!AEM2J		IS THIS BUFFER AVAILABLE ?
290	1142		AD	KEYST	
291	1144	000064	IY		
292	1146	000041	ALMJ		BOTH BUFFERS FULL ?
293	1150		AD	FINISH	
294	1152	000030	X8		
295	1154	000002	MZA		GET KEY
296	1156	000161	KEYJ		CHECK KEY AGAIN
297	1160		AD	FINISH	
298	1162	000032	X8!AZM		STORE IT
299	1164	000034	X8!IA		
300	1166		LD	67	LOAD TOP ADDRESS

301	1170	000025	IA!AEMJ		
302	1172		AD	PAGE	
303	1174	000002	M2A		
304	1176	000161	KEYJ		
305	1200		AD	UNSHL	, IS SHIFT KEY DOWN ?
306	1202		LD	42	, NO-DO UNSHIFTED LIST
307	1204	000002	M2A		, LOAD LIMIT FOR SINGLE MESSAGE KEY
308	1206	000014	ZXZY!DXDY		
309	1210	000041	ALMJ		
310	1212		AD	SHIFTL	, IS KEY IN SHIFTED RANGE ?
311	1214	000170	MO		, YES-USE SHIFTED LIST
312	1216	000110	ZA		, GET IN MODE Q
313	1220	000331	WORD		
	1222	000311			331, 311, 240, 303, 301, 215, 233, 307, 312, 302, 326, 327, 332
	1224	000240			
	1226	000303			
	1230	000301			
	1232	000215			
	1234	000233			
	1236	000307			
	1240	000312			
	1242	000302			
	1244	000326			
	1246	000327			
	1250	000332			
314	1252	000377	WORD		
	1254	000211			377, 211, 316, 315, 306, 305, 323, 330, 210, 331, 324, 325, 334
	1256	000316			
	1260	000315			
	1262	000306			
	1264	000305			
	1266	000323			
	1270	000330			
	1272	000210			
	1274	000321			
	1276	000324			



1300 000325  
1302 000314  
315 1304 000317 WORD 317, 304, 320, 212, 334, 310, 313, 322, 270, 262, 247, 255, 333  
1306 000304  
1310 000320  
1312 000212  
1314 000334  
1316 000310  
1320 000313  
1322 000322  
1324 000270  
1326 000262  
1330 000247  
1332 000255  
1334 000333

316 1336 000263 WORD 263, 265, 256, 273, 261, 271, 275, 260, 264, 266, 254, 267, 257  
1340 000265  
1342 000256  
1344 000273  
1346 000261  
1350 000271  
1352 000275  
1354 000260  
1356 000264  
1360 000266  
1362 000254  
1364 000267  
1366 000257

317 1370 000043 XMT: M2U:KCLJ TRANSMIT CODE AND CHECK CLICK--JUMPER  
318 1372 .AD CLIKSK  
319 1374 000060 CBFF  
320 1376 000130 CLIKSK M1  
321 1400 .LD 110  
322 1402 000002 M2A  
323 1404 000104 DY  
324 1406 000105 DY: AEM2J

325	1410	AD	BELLA	, RING BELL IF IN RIGHT MARGIN
326	1412	000014	FINISH:	
327	1414	000144	DX	
328	1416	000111	ZA:AEM2J	, IS BELL FF SET ?
329	1420		AD	FIN
330	1422	000002	MZA	
331	1424	000164	DA	
332	1426	000023	AZM:AEMJ	, DECREMENT BELL COUNT
333	1430		AD	RING
334	1432	000014	PAGE:	
335	1434	000060	CEFF	
336	1436	000004	DXDY	, SPARE INSTRUCTION
337	1440	000004	DXDY	, GET SCROLL COUNT
338	1442	000161	KEYJ	, IS SHIFT KEY DOWN ?
339	1444		AD	CONTI
340	1446		LD	13
341	1450	000002	CONTI:	
342	1452	000027	MZA	
343	1454		IA:A2M:AEMJ	, INCREMENT SCROLL COUNT
344	1456	000022	PAINT:	
345	1460	000146	DX:M2B	
346	1462	000002	MZA	, GET VIDEO LINE
347	1464	000154	ZX:DX	
348	1466	000105	DY:AEM2J	
349	1470		AD	CURSOR
350	1472	000110	FAKE:	, IS THIS THE CURSOR LINE ?
351	1474	000165	DA:KEYJ	, YES-CHECK BLINKING RATE
352	1476		AD	PAINT2
353	1500	000050	CURSOR:	
354	1502	000002	MZA	, GET FRAME COUNT
355	1504	000045	IA2:ALM	
356	1506		AD	FLASH
357	1510	000023	A2M:AEMJ	, TIME TO FLASH CURSOR ?
358	1512		AD	FAKE
359	1514		LD	160
360	1516	000154	ZX:DX	, NO-INCREMENT COUNT

361 1520 000002 M2A ; GET CUR-X  
362 1522 000014 PAINT2 ZXYZ:DXDY  
363 1524 LD 17  
364 1526 000032 X8:A2M ; RESET SCAN COUNT  
365 1530 000010 ZXYZ ; LOAD CURSOR FLIP-FLOP  
366 1532 000050 IADY ; GET VIDEO LINE  
367 1534 000002 M2A  
368 1536 LD 14  
369 1540 000045 IAZ:ALMJ ; INCREMENT IT  
370 1542 AD PAINT3  
371 1544 000110 ZA  
372 1546 000022 PAINT3 A2M  
373 1550 000150 ZX ; START AT BEGINNING OF LINE  
374 1552 000040 E2Y  
375 1554 000141 5# VSCJ  
376 1556 AD 5# ; WAIT FOR END OF SCAN FLAG  
377 1560 000016 ZXYZ:DXDY:M2A  
378 1562 000020 SVID ; TURN ON VIDEO  
379 1564 LD 30  
380 1566 000045 IAZ:ALMJ ; INCREMENT SCAN COUNT AND COMPARE  
381 1570 AD PAINT3  
382 1572 000026 IAZ:A2M ; GET PROPER COUNT  
383 1574 000030 X8  
384 1576 000002 M2A  
385 1600 000000 SCFF  
386 1602 000150 ZX ; TURN ON CURSOR  
387 1604 000141 VID: VSCJ  
388 1606 AD VID  
389 1610 000100 ZCAV ; TURN OFF CURSOR AND VIDEO  
390 1612 000016 ZXYZ:DXDY:M2A ; GET SCAN COUNT  
391 1614 LD 40  
392 1616 000041 ALMJ ; TIME FOR ANOTHER UART CHECK ?  
393 1620 AD UARTJ  
394 1622 LD 42  
395 1624 000041 ALMJ ; TIME FOR NEXT LINE ?  
396 1626 AD ; NO-WAIT A SCAN

397 1630 000102 M2X ;GET LINE COUNT  
398 1632 000002 M2A ;CHECK FOR 50/60 HZ COMPENSATION  
399 1634 LD 15 ;MORE LINES TO PAINT ?  
400 1636 000025 IA: AEMJ EXTRA  
401 1640 AD ;  
402 1642 000041 ALMJ PAINT  
403 1644 AD ;  
404 1646 LD 0 TOPROU:  
405 1650 000014 ZXZY:DXDY  
406 1652 LD 66  
407 1654 000144 DX ;GET READY TO SCAN KEYBOARD  
408 1656 000006 DXDY:M2A ;GET KEYBOARD BUFFER  
409 1660 000161 KEYJ ;KEY STILL DOWN ?  
410 1662 AD TOPROU ;CLEAR IT IF KEY IS UP  
411 1664 000064 IV ;GET NEXT BUFFER  
412 1666 000002 M2A ;  
413 1670 000161 KEYJ ;  
414 1672 AD TOPROU ;  
415 1674 000170 MO ;GET IN MODE 0  
416 1676 000151 ZX:UTJ ;IS U-TRANSMITTER READY ?  
417 1700 AD FIN ;NO-SKIP KEYBOARD CHECK  
418 1702 000130 M1 ;  
419 1704 000111 ZA: AEM2J ;ALL DONE ?  
420 1706 AD TOP2 ;YES-CONTINUE  
421 1710 000002 M2A ;  
422 1712 LD 57 ;CHECK IDENTIFIER  
423 1714 000041 ALMJ ;  
424 1716 AD IDE  
425 1720 LD 101  
426 1722 000021 AEMJ ;  
427 1724 AD TOPROU  
428 1726 000041 ALMJ ;  
429 1730 AD IDE  
430 1732 LD 33  
431 1734 000042 IDE M2U  
432 1736 000131 MI:TRUJ

B-11



DIGITAL EQUIPMENT CORPORATION  
MAYNARD, MASSACHUSETTS

ENGINEERING SPECIFICATION

DATE 12/19/74

TITLE VT50 ACCEPTANCE TEST

H. Church

REVISIONS

REV	DESCRIPTION	CHG NO	ORIG	DATE	APPD BY	DATE

Upon removal from shipping container, inspect for physical damage, then make the following checks before connecting your VT50 terminal to a system:

1) FLASHING CURSOR CHECK

Set terminal, using a screwdriver or small coin, to "Local Mode" and "9600" baud as shown on label mounted to underside of terminal. Plug terminal into line, move the ON/OFF slide switch located on the right side of terminal to the ON position.

After a one minute warm-up period, a flashing cursor should appear on the screen. If nothing is seen or display is too bright, reach over and adjust the intensity control on the rear of the terminal at the top right hand corner. Control moves to the right for increased brightness.

2) REMOTE MODE CHECK - FULL DUPLEX

Set terminal for full-duplex operation, locate terminal strip, with screws numbered 1 through 6 on underside, jumper 1 and 2 together then 3 and 4 and finally 5 and 6. Use any wire for jumpers.

The terminal should now display characters as keys are depressed.

ENG	APPD	SIZE	CODE	NUMBER	REV
		A			

This drawing and specifications, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.



## TITLE VT50 ACCEPTANCE TEST

3) HALF DUPLEX, ESCAPE AND CONTROL COMMAND CHECKS

Remove all jumpers added to the terminal strip in check (2). Set terminal for half-duplex operation, then depress keys and check for corresponding characters on screen.

Control Code Checks:

	<u>DEPRESS KEYS</u>	<u>ACTION</u>
a.	CTRL G	Ring buzzer
b.	CTRL H	Move cursor left one position
c.	CTRL I	Move cursor to next tab stop
d.	CTRL J	Move cursor down one line
e.	CTRL M	Move cursor to leftmost position on line
f.	ESC A	Move cursor up one line
g.	ESC C	Move cursor right one position
h.	ESC H	Move cursor to the HOME position (top left of screen)
i.	ESC J	Erase from cursor to end of screen
j.	ESC K	Erase from cursor to end of line
k.	ESC Z	Identify terminal type (terminal will transmit ESC/A) check for a character "A" to appear on the screen.

Check Hold Screen Mode Operation As Follows:

- Place cursor on bottom line
- Type "ESC [", Hold Screen Mode
- Type "LF"
- Type "VT50" - check characters do not appear on screen
- Type "Page Con't" key - the message "VT50" should now appear on the screen
- Type "ESC \", exit Hold Screen Mode
- Depress "LF" key - check for message to scroll up

4) ON-LINE ACCEPTANCE TEST

Connect VT50 to a PDP-11 computer via a DL11-A/B interface. Load and run the VT50 acceptance test program for at least one pass.

MAINDEC-11-DZVTC-A