MICROCOMPUTER BASED MULTILINGUAL TEXT EDITOR

A DISSERTATION

submitted in partial fulfilment of
the requirements for the award of the degree
of
MASTER OF ENGINEERING
in
ELECTRICAL ENGINEERING
(Systems Engineering and Operations Research)

By

CHANDRASHEKHAR B. RAJE

DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247667 (INDIA)

February, 1988

CANDIDATE'S DECLARATION

I hereby, certify that the work which is being presented in the dissertation entitled, 'MICROCOMPUTER BASED MULTILINGUAL TEXT EDITOR' in partial fulfilment of the requirements for the award of the degree of MASTER OF ENGINEERING in ELECTRICAL ENGINEERING with specialization in SYSTEMS ENGINEERING AND OPERATION RESEARCH, submitted in the Electrical Engineering Department, University of Roorkee, Roorkee (India), is an authentic record of my own work carried out for a period of about six months from August, 1987 to February, 1988, under the supervision of Sh. M.K. Vasantha, Reader, Electrical Engineering Department, University of Roorkee, Roorkee, India.

The matter embodied in this dissertation has not been submitted by me for the award of any other degree.

Dated 15th Feb'88

CHANDRASHEKHAR B.RAJE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

M Vasantha 15.2.1988

(M.K.VASANTHA)
READER
ELECTRICAL ENGINEERING DEPTT.
UNIVERSITY OF ROORKEE,
ROORKEE.

ACKNOWLEDGEMENT

I consider it my proud privilege to express my deep sense of gratitude, appreciation and regards to my guide Shri M.K.VASANTHA, Reader, Department of Electrical Engine ring, University of Roorkee, Roorkee, for his very able guidance, encouragement and inspiration. His kind co-operation in permitting me to handle many equipments freely have helped me tremendously in overcoming several obstacles.

I would fail in my duty if I do not thank to Prof. J.D. Sharma, Prof. H.K.Verma, Deptt. of Electrical Engineering, University of Roorkee, Roorkee, for their co-operation in carrying out this work.

I would like to gratefully acknowledge the direct or indirect part played by Prof. A.K.Pant, Prof. P.Mukhopaddya (HOD), Electrical Engineering Department, University of Roorkee, Roorkee.

Thanks are due to Sh. N.P.Aterkar, M/S. Soilex Consultants Pvt.Ltd., Roorkee for having helped me out of number of difficulties.

The completion of thesis with all the help available would not have been possible without the co-operation and advise of my close friends, Mr. V.Prabhu, Mr. M.Jain, Mr. A. Punde, Mr. B.Shailesh, Mr. P.Kadwe, Mr. P.Bhave, Mr. N.-Nandekar and Mr. R.Dixit, who have been associated with the thesis work right from the budding to the completion stage.

The µP and computer lab. staff, Mr. R.Singh and Mr. K.Singh, who have played a role of prime importance in getting my thesis completed, also deserve a special word of thanks from me.

Last but not least I express my thank to Mr. D.C. Bhardwaj, for his efficient and time bound typing of this thesis.

CHANDRASHEKHAR B.RAJE

ABSTRACT

The basic idea of the present work is to develop different Indian languages script on the CRT terminal using facilities of 8275 CRT controller, since the general purpose CRT terminal is used to implement these languages, a monitor program is developed to suit all the requirements. Extra hardware is added to the existing CRT interface for developing the scripts. This gives facility of many languages to be used in the system. There is provision for 8 languages in the present system. The languages attempted for the present work are, English, Hindi, Marathi and Kannada. Keyboardsof these languages are designed accordingly.

Next phase of this work is the Editor software. This helps to correct the typed information. Editor can work irrespective of the language selected for use. For the development of Editor, 8086 Microcomputer kit is used. (VMC-86/3 manufactured by VINYTICS PERIPHARALS Pvt.Ltd) Standard functions are developed in the editor to correct the data.

The printer infaced is done in the present work to get the hard copy of the typed information. The languages that printer can print are, English, Hindi, Marathi, Software program is developed to get printing in Hindi and Marathi. Printer is used in the dot addressable graphic mode for the purpose.

CONTENTS

contd...

CHAPTER - 1

INTRODUCTION

1.1 GENERAL

Of all the achievements of the human mind, the birth of the alphabet is the most momentous. Every script has contained the spirit of its age, of its people, of the inventions and tools of that time. Almost every language has its own way of writting the scripts. This way of writting the scripts changes as the ages pass.

India is the only country, which has an integration of different languages and scripts. Different languages are used in the different parts of the country for the communication. This communication is restricted to that particular part of the country. Nowdays computers are widely used for the communication. But English being the only language to use these computers, other languages can not be communicated through computers. Nowadays the use of Indian languages on computers is highly encouraged.

The basic philosophy of using Indian languages on computers, is to develop scripts for these languages. Putting these languages onto the computer has, for some time now, caught the fancy of researcher. It is very difficult to produce Indian scripts in a satisfactory manner on computers. This difficulty even increases if the conventional ways of implementation are used. The conventional way of implementing the English on the computer is simple as it has fixed characters format and each character is required sequentially. But in case of Indian languages the script may be of different size and shape. There are 'matras' to put on the scripts. And this requires some special approach to develop these languages on computers.

1.2 NECESSITY OF INDIAN LANGUAGES ON COMPUTERS.

In India there are many languages used for communication. The tradition of using the local language as an official language is still present in the Government sector. Nowadays the computers have entered into the government sector as well. The combination of these two, that is to use local language and also to use computers, makes it necessary to develop Indian languages on the computers.

Nowadays personal computers (PC) are available with sophisticated softwares. This prompted the researchers to use Indian languages on it. According to MAHABALA [8] the use of Indian languages for the data processing is most desirable. Computer society of India (CSI) has chosen Information Technology in Indian languages as a theme of CSI-88. CSI is encouraging the Hardware and software development for Indian languages applied for Information Technology. CSI is also trying to use the Indian languages as programming languages. CSI is equally interested in the use of High level language in English with facility to perform Input/Output in Indian languages.

The most essential thing for using the Indian languages for data processing is to adopt the standards. The standards needed are keyboard standards, internal representation, and data communication. Some of these standards are given by DOE Journal [9].

Some other works done in this area are briefly enumerated below:

i) Prof Donald Becker of the University of Wisconsin, has done extensive work in creating software that prints out Devnagari, Telugu from IBM PC on to Toshiba printers.

ii) Zhang Liansheng, a visiting lecturer at Berkeley from Peking, has developed a Tibetan script for the Macintosh microcomputer.

iii) Hinditron Equipments manufacturing ltd. has developed a bilingual software named AALEKH.

iv) ORG systems has developed a multilingual word processor software.

1.3 SCRIPT DEVELOPMENT TECHNIQUES

A script development on computer becomes the Ist stage of producing different languages. Different techniques are tried by different researchers. Geosge L. Hart [3] when tried to get the script of the South Indian languages on APPLE-11 he faced many difficulties. Later after few years Macintosh micro computer appeared. Using Macintosh George Hart developed Tamil and Devnagari script. The Macintosh differs from other computers and specially for non-Roman alphabets. For example, on most microcomputer, when 'a' is pressed on the keyboard, the machine looks in ROM, retrieves the pattern corresponding to 'a' and puts it at the current screen position. On the Macintosh, when 'a' is pressed, the operating system looks to see what the character font (character set) is, what the current character size is, and what the character attribute is. If it does not have that font already in memory (RAM), it loads it in from a file called 'system', and then proceeds to print the character in the desired pattern, using routines built into the operating system.

The process of implementing Devnagari and other Indian languages on Macintosh is relatively simple. It consists of defining a matrix mapping for each character, deciding width and placement of the character and then putting that information, all coded in the right format, call 'resource' into the operating system using the option keys. The macintosh can generate twice as many characters as a normal typewriter. The difficulty of using this machine is that, it is unable to write in Arabic script. The Arabic scripts doesnot follow the alphabetic rule and hence did not get the place on the computers easily.

This method of developing scripts is used by many nanufacturers. All of them use the script in a graphics node. Software packages can be developed to produce scripts on the computers. This method is relatively simple as the script is predefined in the software.

The other method of developing a script on the computer is to use ROM as a character generator. The required character pattern is accessed by the system after pressing the key. This method is the most convinient but it requires Hardware modification in normal computers.

This ROM based disbplaying of script is done by GARWAL [1] et.al. They have used the CRT controller chips. his chip handles Roman script easily, but for other scripts ome extra hardware is to be developed with this chip. GARWAL [1] et.al. uses the method of composite character eneration. Each composite character is made up of a onsonant and a matra The internal form of the text is hat consonant is followed by the matras. Six bits are used or coding consonant and matras and the seventh bit is used o signify the composite group. Each group starts with 1 in lag bit position and ends with a 0 in the flag bit position.

For each composite character all primitives in that roup are fetched from 8275 by simulating a character pulse nd are routed to different pattern PROMS. Figure 1.1 hows the way composite **front** shapes are generated by sing PROM outputs, and by shifting of matras.

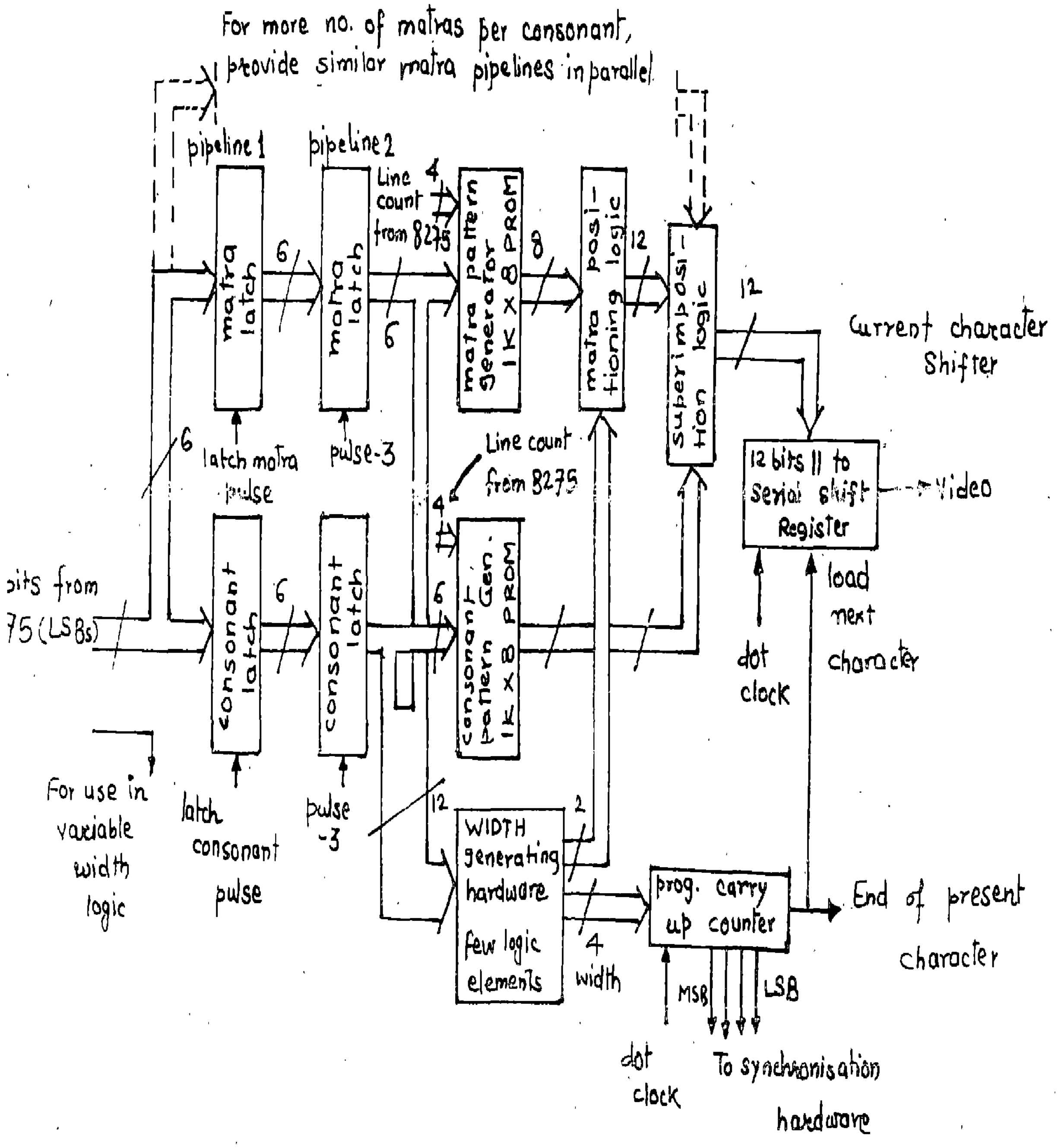

For more no. of matras per consonant,
provide similar matra pipelines in parallel
pipeline 1
pipeline 2
matra latch
matra latch
6
6
Line count from 8275
4
matra pattern generator 1K x 8 PROM
8
matra posi-tioning logic
12
superimposi-tion logic
12
Current character Shifter
12 bits ll to Serial shift Register
video
load next character
dot clock
latch matra pulse
pulse-3
6
Line count from 8275
4
6
bits from
75 (LSBs)
consonant latch
consonant latch
consonant pattern Gen. 1K x 8 PROM
latch consonant pulse
pulse -3
For use in variable width logic
12
2
WIDTH generating hardware
Few logic elements
4
width
prog. carry up counter
MSB
LSB
dot clock
To synchronisation hardware
End of present character

FIG 1·1

1.4 CONTRIBUTION OF AUTHOR

The technique adopted in the present work is unique. It differs from all other techniques used earlier. The other techniques were developed earlier on the special machines. But for the present work the terminal made for English languages is converted for use in Indian languages. Without changing the available hardware, attempts are made to develop Indian languages on it. The extra hardware is interfaced with the available hardware, which makes the terminal self controlled, displaying script. This technique eliminates the need of software package to display script through graphics.

In the present work the facility is provided such that the user can use any one language out of the eight specified. The editor software is prepared to correct the typed information. The Editor developed in the present work is very simple to use. Attempts are also made to get the hard copy on the dot matrix printer. This system consists of CRT terminal, microcomputer 8086 kit and a dot matrix printer.

CHAPTER - 2

EXISTING CRT INTERFACE-OF-VDT-85 TERMINAL

2.1 BASIC CRT

On the CRT the image is displayed which is generated by the series of lines called raster. This display is called the raster scan display. Generally, the beam starts from the upper left hand corner of the display and simultaneously moves left to right and top to bottom. This horizontal and vertical movement of beam is controlled by two independent but simultaneously operating circuits. While the electron beam is moving accross the CRT the third circuit controls the flowing current in the beam. The brightness of the image is proportional to this current.

When the beam reaches the end of a line, it is brought back to the begining of the next line at the rate which is much faster than that was used to generate the line. This is called 'Retrace. Retrace does not appear on the screen. As the beam is moving across the screen horizontally, it is also moving downward. And because of this, each successive line starts slightly below the previous line. When the beam finally reaches the bottom right hand corner of the screen, it retraces back vertically to the top left corner. The total time taken by the beam to travel from top to bottom and again to top is refered to as frame. The horizontal sweep frequency is decided by circuitary which is in the range of KHz, and the vertical sweep frequency is 50 Hz.

As the horizontal frequency increases the number of horizontal lines per frame increases. Thus the resolution on the vertical axis increases. For graphic terminals and special text editing terminals high resolution is needed.

The characters that are displayed on the screen are formed by a series of dots that are shifted out of the controller while the electron beam moves across the CRT face. The circuits that create this timing are referred to as the dot clock and the character clock. The character clock frequency is equal to the dot clock frequency divided by the number of dots used to farm a character along the horizontal axis.

2.2 FUNCTIONING OF 8275

The 8275 programmable CRT controller has a function to refresh the display by buffering the information from the main memory. It constantly keeps track of the display position on the screen.

2.2.1 DISPLAY REFFRESHING

The 8275 used for specific screen format generates a series of DMA requestsignals which results in the transfer of a row of characters from display memory to the 8275's row buffers. The 8275 has two row buffers. While one row buffer is being used for display, the other is being filled with the next row of characters to be displayed. The number of display characters per row and number of character rows per frame are software programmable. The 8275 requests DMA or CPU to fill the row buffer that is not being used for display. It displays character rows one at a time. The 8275 provides special control codes which can be used to minimize load on the software. It also generates the cursor, its position is software controlled.

The 8275 presents the character codes to the external character generator using its output signals CCO-CC6. External dot timing logic is then used to transfer the parallel output data from the character generator ROM serially to the video input of the CRT. This chip has 4 line count outputs, LCO-LC3 which are applied to the character generator to perform the line selection function. Lines are displayed one by one. For each display same process is repeated. At the beginning of the last displayed row, the 8275 issues an interrupt by setting the IRQ output line. The 8275 interrupt output will normally be connected to the interrupt input of the CPU, which causes the CPU to execute the Interrupt Service Subroutine (ISS). This typical ISS has to re-initialize different parameters for the next display refresh cycle, and to execute appropriate functions. The 8275 has two types of programming registers, the command register (PREG) and Status Registers (SREG). The command register can only be written into. The other one that SREG can only be read. Generally the 8275 expects command followed by a sequence of parameters (maximum 4). The 8275 instruction set consist of eight commands as given in the appendix A-2.

To generate display format the 8275 provides a number of display parameters. Display formate like 1 to 80 characters per row 1 to 64 rows per screen, and 1 to 16 horizontal lines per row can be used. In addition to the refreshing of characters from memory to the CRT screen, the 8275 also controls the cursor. The cursor position is decided by X and Y cursor position registers. Any cursor format can be used.

2.2.2 CRT TIMING

There are two timing outputs provided by the 8275 called HRTC and VRTC. These two timing output syncronise the vertical and horizontal oscillator in refresh cycle. There is one more timing output called Video supress (VSP). This VSP is active when HRTC or VRTC is active. This output sends the blinking signal to the dot timing logic.

The light enable (LTEN) signal is used to enable the video signal to the CRT. This output is active at the programmed Underline cursor position and at the position specified by attribute codes. Another timing output Highlight (HLGT) is used to increase the CRT beam intensity greater than normally used. The 8275 has another output called Reverse Video (RVV) which causes the system video to

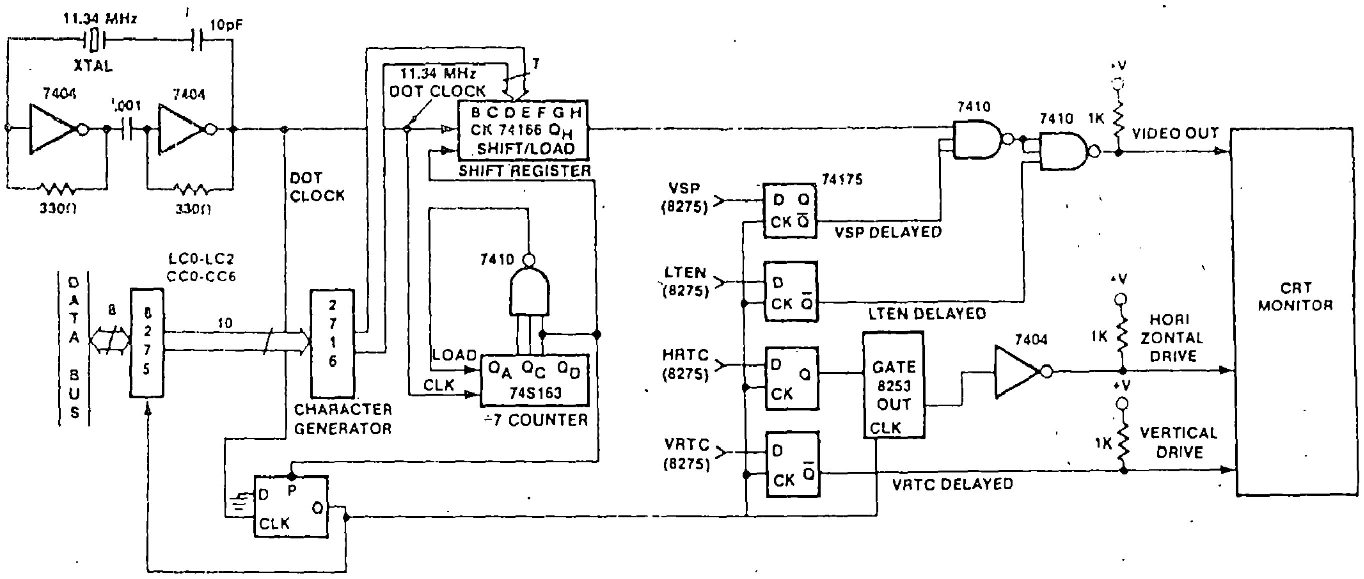
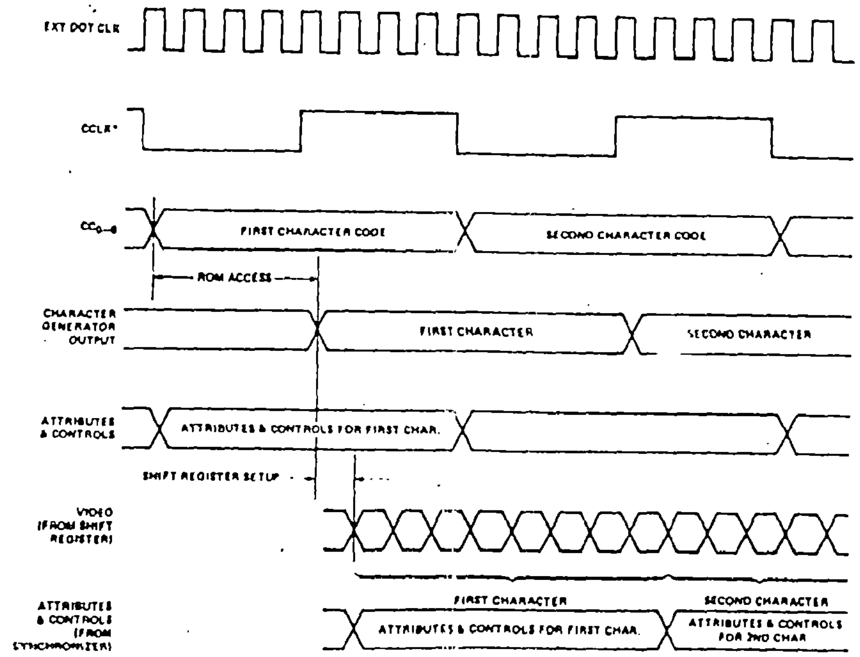


FIG 2·1


EXT DOT CLK
CCLK*
CC0-6
FIRST CHARACTER CODE
SECOND CHARACTER CODE
ROM ACCESS
CHARACTER GENERATOR OUTPUT
FIRST CHARACTER
SECOND CHARACTER
ATTRIBUTES & CONTROLS
ATTRIBUTES & CONTROLS FOR FIRST CHAR.
SHIFT REGISTER SETUP
VIDEO (FROM SHIFT REGISTER)
FIRST CHARACTER
SECOND CHARACTER
ATTRIBUTES & CONTROLS (FROM SYNCHRONIZER)
ATTRIBUTES & CONTROLS FOR FIRST CHAR.
ATTRIBUTES & CONTROLS FOR 2ND CHAR

*CCLK IS A MULTIPLE OF THE DOT CLOCK AND AN INPUT TO THE 8275.

FIG 2·2

be inverted. The dot timing logic is shown in the figure 2.1 and figure 2.2 . Basic requirement of the Dot Timing Logic is to have a 11.34 MHz clock frequency. A separate crystal is used to generate this frequency. A shift register 74LS166 and a counter 74LS163 uses this frequency. This generated 11.34MHz frequency is called Dot Clock. Using Dot Clock, counter 74LS163 generates character clock (CCLK). Character clock is applied to 8275 to generate character counts (CCO-CC). Using character count, character generator finds character and sends it to shift register. Shift register outputs the character to video.

2.3 HARDWARE DISCRIPTION

The heart of the CRT terminal is the 8085 microprocessor. The 8085 initializes all devices in the system. It programs the CRT controller, assembles characters to be transmitted. It decodes the incoming characters and determines where the character is to be placed on the screen. Thus all the way CPU is quite busy. The layout of hardware is shown in the APPENDIX- A4

The 8275 is used as the CRT controller in the system. 2716 is used as a character generator. It uses four 6116 chips. There provides 8K RAM area. It has 4K ROM having address from 0000 onwards for the monitor program. To support the CPU these are some standard LSI peripheral devices arround 8085. There are three 8251S, one is used for receiving characters from the keyboard, another for RS-232 Main for serial communication, and third one is used for auxilary RS-283 serial communication.

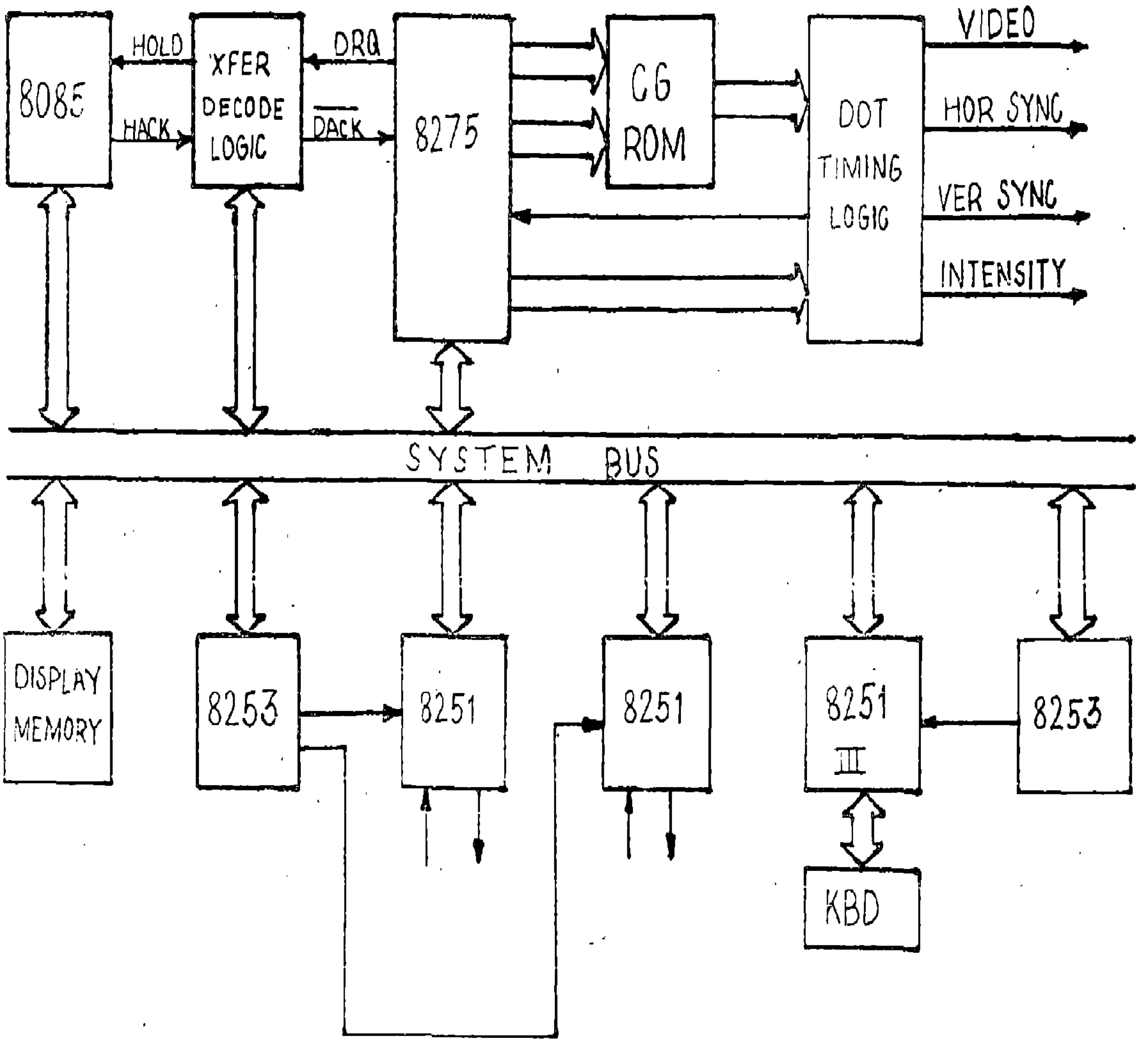
Two 8253 supports the control card to provide clock frequencies to different chips and circuits.

Other than above chips same chips like multiplexer, Demultiplexers, shift registers and buffers are used. All peripheral devices are I/O mapped.

2.4 SYSTEM OPERATION

The starting operation of the control card is to initialize all the chips in proper mode of operation. The basic CRT system diagram is shown in the figure 2.3 .As the initialization is over the CPU has to poll the 8251 connected to the keyboard continusly in local mode, and other 8251 in line mode. When the CPU receives the character it has to decode it and to take appropriate action.While the 8085 is executing the above program, it is being interrupted after a particular interval by the 8275. There are two interrupts from the 8275 at different fixed time. These interrupt service subroutines are used to refresh and to load the row buffers on the 8275. This ISS gives the status of the 8275.

Generally the particular set of instructions is used to rapidly move the contents of the display RAM into the row buffers of the 8275. These are nothing but codes of the display



CRT SYSTEM BLOCK DIAGRAM

FIG 2.3

characters. These codes are transferred to the character code outputs CC0- CC6. These output lines are connected to the address lines of the character generator as shown in the figure 2.4 Line count outputs LC0-LC2 of the 8275 are connected to the another address lines of the character generator. Line count outputs are used to select the line which has a character selected by the address lines of the character generator. Following the transfer of the first line to the other timing logic, the line count is incremented and then the second line of character row is selected. This process of transfer is continued until the last line of the row is transferred to the dot timing logic.

After the character row has transferred to the dot timing logic, the dot timing logic latches the output of the character generator ROM into a parallel in serial out synchrous shift register 74LS166. Through this shift register character is transferred to the video one by one bit i.e. serially. Normally for one character, shift register has to transfer seven bytes serially. The clock applied to the shift register is fixed by the dot timing logic. Thus continuosly it supplies the character input to the CRT.

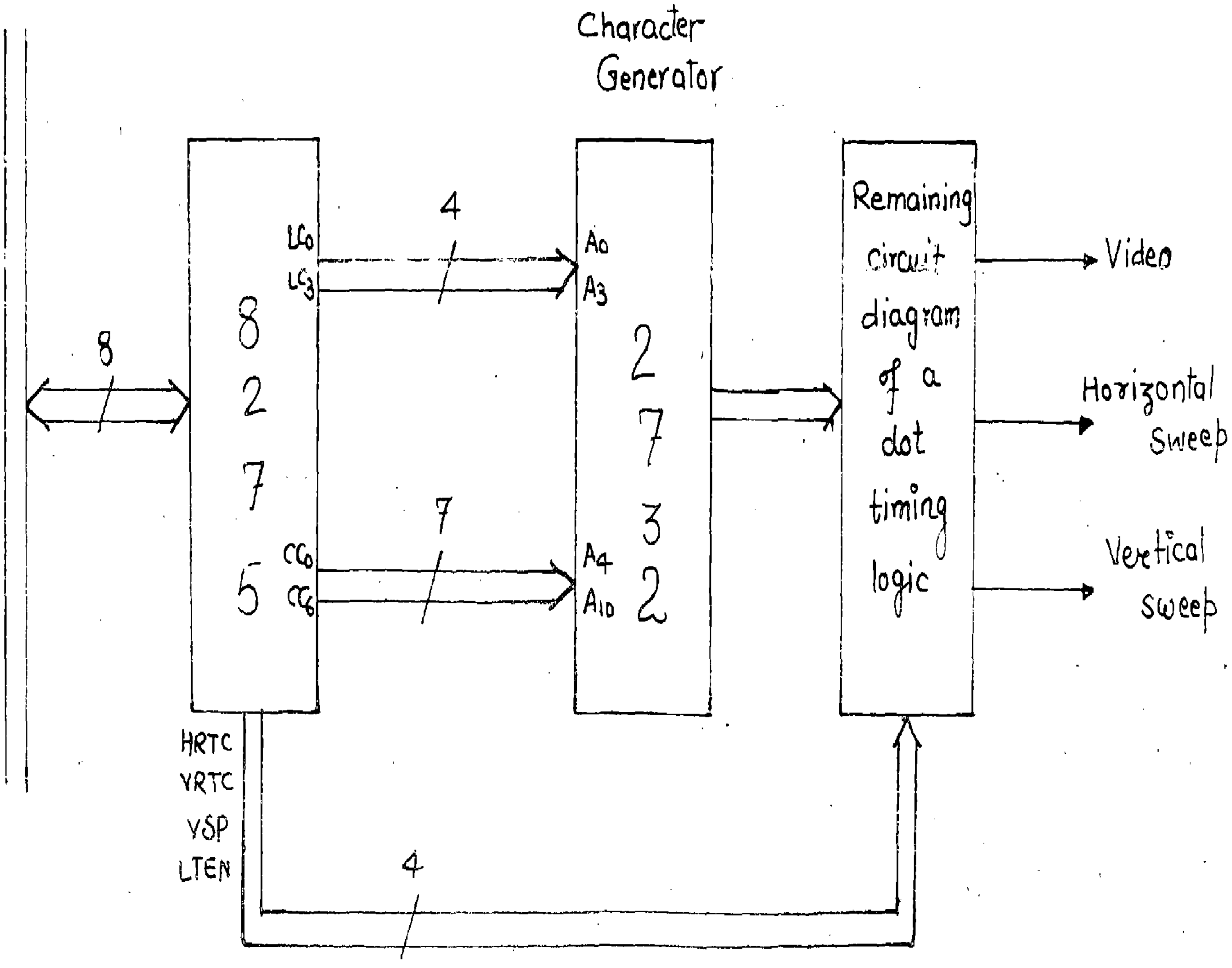

Character Generator
8
8275
LC0
LC3
4
A0
A3
2732
7
CC0
CC6
A4
A10
Remaining circuit diagram of a dot timing logic
Video
Horizontal Sweep
Vertical sweep
HRTC
VRTC
VSP
LTEN
4

FIG 2·4

2.5 FILLING ROW BUFFERS OF 8275

Normally the 8275 is used with the DMA controller. In case of filling the row buffers of the 8275, the DMA transfers data rapidly. DMA controller increases a need of additional circuitary. Need of DMA controller can be eliminated by using interrupt driven routines which transfers data to 8275. For the present work RST 7.5 is used instead of DMA controller.

The present system makes a use of SOD line of 8085 This is used as a special transfer bit. Filling the row buffer of 8275 with the help of SOD line makes use of decoder 74LS157 Figure 2.5 . This decoder generates the required signal whenever necessary.

To fill the row buffer of 8275, a special technique is used. This technique transfers data to 8275 using a set of 40 POP instruction. This transfers data rapidly.

Decoder 74LS157 along with the SOD forces 8275 to write the data whenever necessary. As shown in the Fig. 2.6 the input S determines the data transfer alongwith $\overline{E}$ (low). The S input of the decoder is SOD. The SOD converts the processor's read into $\overline{DACK}$ and $\overline{WR}$. It masks processor fetch cycles from the 8275, so that a fetch cycle does not write into the 8275. SOD functions along with SØ of 8085 which is low during read operation.

As discussed earlier the data is transferred by POP instruction. To read this data stack pointer is loaded with

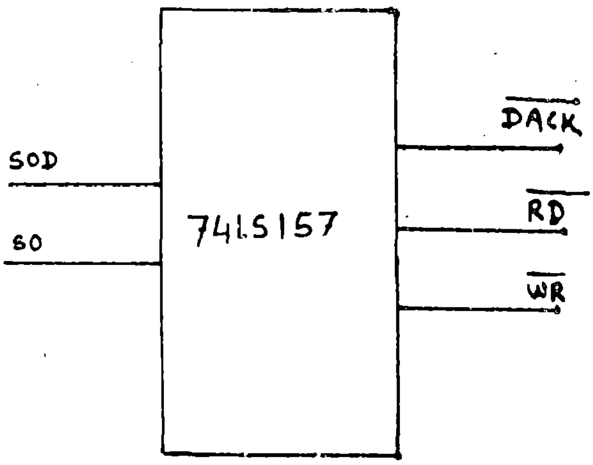

SOD
S0
74LS157
$\overline{DACK}$
$\overline{RD}$
$\overline{WR}$

FIG 2.5

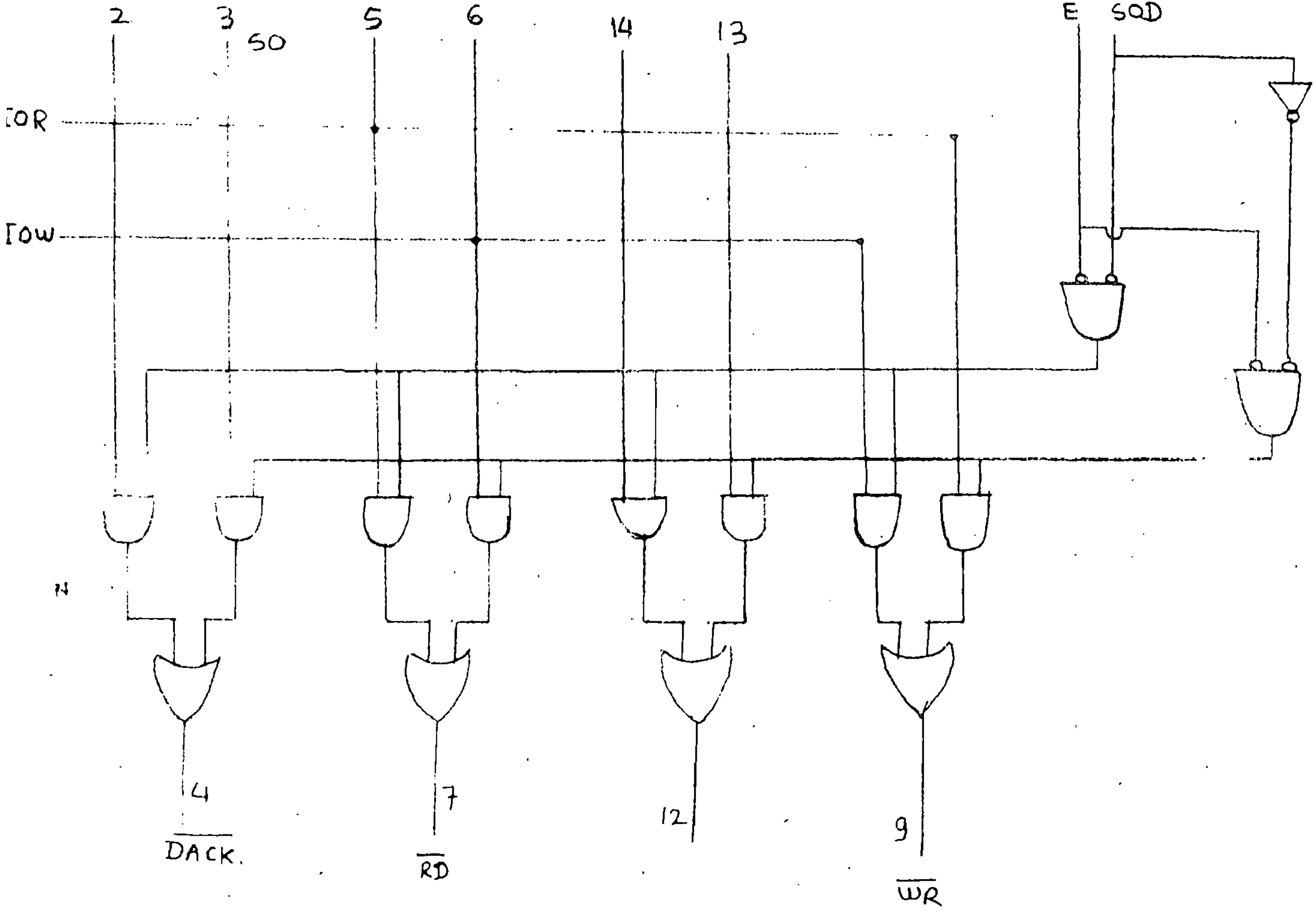

2
3
S0
5
6
14
13
$\overline{E}$
SOD
IOR
IOW
N
4
$\overline{DACK}$.
7
$\overline{RD}$
12
9
$\overline{WR}$

FIG 2.6

that specified memory location while the stack pointer is saved in DE register earlier. When the first POP instruction op code comes on the data bus, the mask provided doesnot write this data on 8275. The CPU executes this POP instruction by reading data from the specified memory location. When this data is on data bus, it tries to load the L register, since the 8275 is ready to write data, same data is written into 8275 simultaneously. In the next read of this instruction, again the same process occurs except the register filled is H. Thus for all the 40 POP instructions data (read by CPU) of 80 bytes can be transfered to 8275 regardless of loading the HL 40 times. This process of transfer is very fast as the same data is transfered to HL register and 8275 simultaneously from data bus.

Thus whenever SOD line is high every memory read machine cycle makes $\overline{\text{WR}}$ and $\overline{\text{DACK}}$ low for writting onto the 8275 buffer from data bus and inhibits this writting for any operation fetch cycle.

CHAPTER - 3

HARDWARE DEVELOPMENT

3.1 NECESSITY

The original hardware made for the CRT terminal is for English language usage. This ,hardware has very limitted facility as being used for general purpose. In the present dissertation work multilanguages are used to display the script on the CRT terminal, of course ,many character generators have been used to solve the problem. For each language a seperate character generator is used. This use of many character generators necessiates the extra hardware.

The original hardware is used only for English language. The characters developed for this language are of the fixed format and of lesser size. 2716 EPROM works as a character generator. In case of Indian languages the uniformity in the script is not these. So the character developed for these languages have got a different format. Other than this, Indian languages uses matras. Because of these matras the size of the cursor and corresponding character to be displayed is of greater dimension. Though the format of the characters has changed to bigger one, the number of characters to be generated are same. Hence the character generator having more memory capacity (2732) has to be used. This changed character generator requires some extra hardware.

The selection of the language which is to be displayed is done through software only. For this perpose the additional hardware helps.

3.2 FUNCTIONING

The extra hardware developed supports the original hardware. Because of this some more facilities has increased. The circuit diagram of the supporting hardware is shown in the Appendix-A4 This contains commonly used peripheral chips, ROM, buffers and logic circuits.

All the data lines, power supply lines, address lines are taken from the main CRT control card. All the data lines address lines, reset signal, chip select signal, data lines of the 8275 etc. are buffered to avoid the loading. Similarlly all the output data of the character generator which is input to the CRT is also being buffered.

3.2.1 CHARACTER GENERATOR

The main idea behind this hardware development is to suit all the requirements. The first basic requirement is to increase the size of the character. Such facility is provided by the 8275. Accordingly character generator has to be changed. The character generator has an important role to plot for displaying the characters. The character code data is supplied to the character generator from the 8275 as 7 bit address and is shown in the figure 2.4

Normally for English the dot matrix dimensions of the character is 5X7. And this 5X7 dot matrix is accomodated in the 7X10 field on the CRT terminal. These dimensions are programmable. For the standard English keyboard these are 128 characters to be displayed. Thus considering the 5X7 dot matrix character size, 2716 can accomodate all the characters.

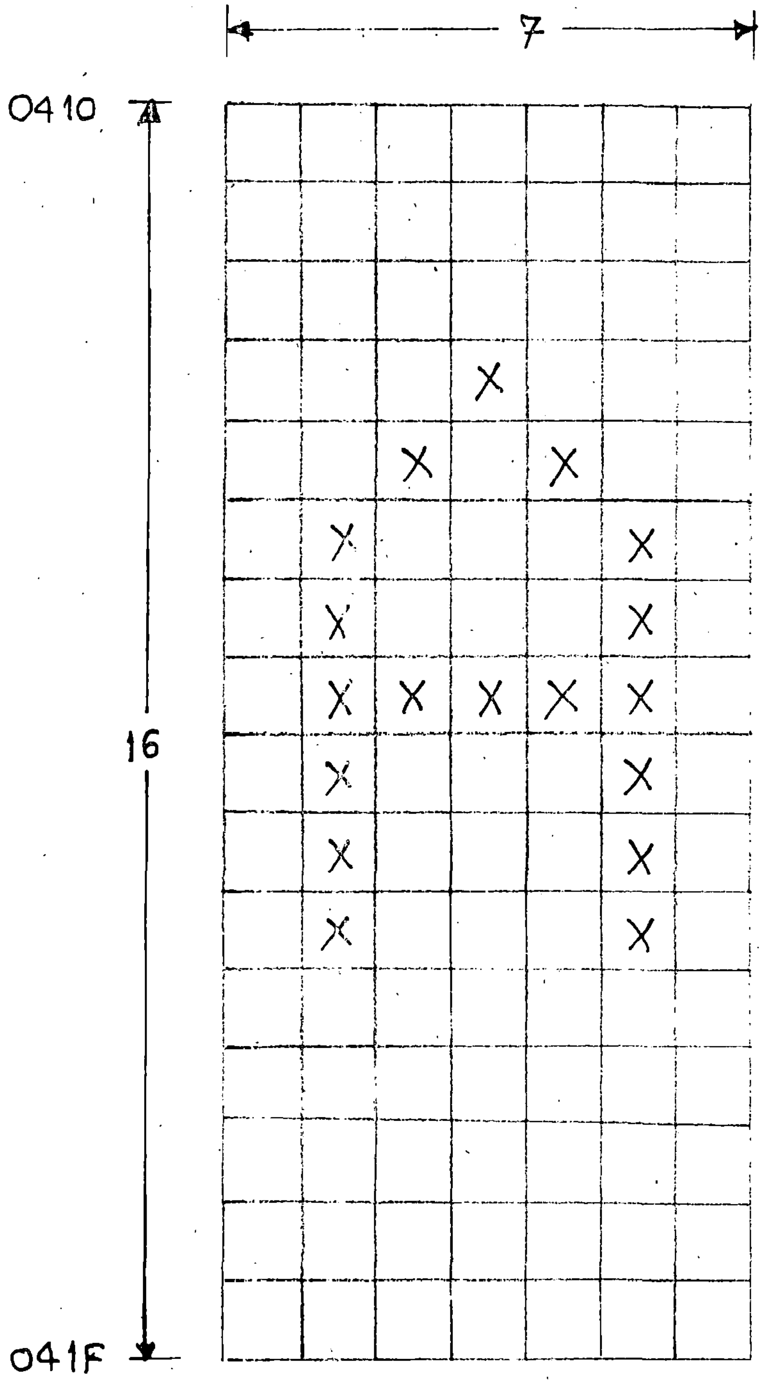
In the present dissertation work the character field has been changed to 7X16, and the character size is 7X16 dot matrix maximum. The character dimensions are changed depending on the language. For the English same standard dimensions i.e. 5X7 is used but in the 7X16 field.

2732 EPROM is used as character generator for each language. The working of the character generator with the 8275 interface is very straight forward. The 8275 has four line count outputs LC0-LC3. These are connected to the four lower order address lines of the character generator (A_0-A_3). Similarly the character count outputs CC0-CC6 of the 8275 are connected to the A4-A11 address lines of the character generator. The character data from the memory location, identified by the address is inputted to the video through shift register.

Any character which is to be displayed is read through the character generator. The character generator basically has a fixed bit pattern at different addresses. For the present work since 7X16 dot format is used, 16 bytes are reserved for each character. According to the ASCII codes each character has its own address in the character generator e.g. 'A' in English has a ASCII code 41H, and for 'A' the address in the character generator. is 0410 to 041F. The bit pattern for 'A' is shown in Fig. 3.2 From next location the bit pattern of the next ASCII code viz. for B, is stored. Thus only one character will be accessed at a time. The bit pattern for the 'अ' letter in Devnagari is as shown in the figure 3.3a. Similarly for all the characters of the different languages, bit pattern is arranged. For matras of the Indian languages also a bit pattern is developed. This is shown in the Figure 3.3b. for 'ृ' of devanagari language. Once the bit map of the identified character is obtained as explained, this fixed dimensioned bit map (16 byte) is transferred to the shift register, one after another byte. This single byte becomes a parallel input of the shift register, This parallel to serial converting shift register is used for seven bit input maximum. The seven bit data is sent to the VDU serially. This displaying of one by one bit on CRT is at a very fast rate (around 11.34 MHz).

3.2.2 THE DECODING LOGIC CIRCUIT FOR SELECTING CHARACTER GENERATOR

This logic circuit is necessary for selecting the character generator of different languages. This circuit consist of a decoder 74LS138 which has a 1 of 8 decoding



			X			
		X		X		
	X				X	
	X				X	
	X	X	X	X	X	
	X				X	
	X				X	
	X				X	

FIG 3·2

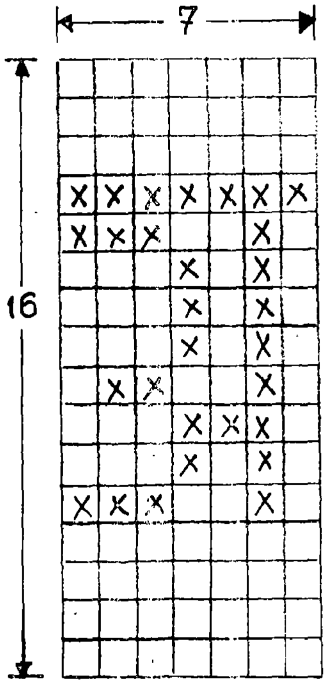


FIG 3·3 (a)

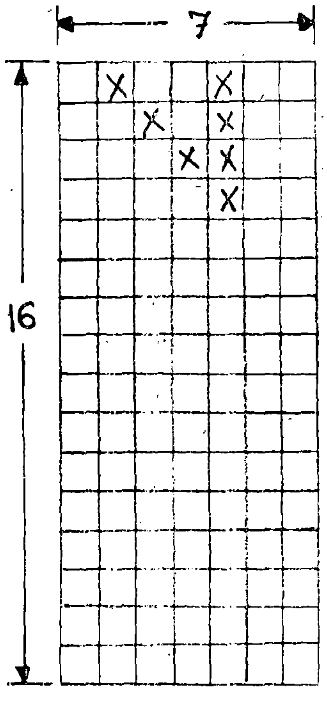


FIG 3·3 (b)

facility. This TTL chip is driven by the PPI 8255, the port C upper is used as the input to the decoder. The necessary address lines, data lines, control signal lines are taken out from the main PCB of the CRT terminal. All these lines are buffered before using the same in the character generator unit fabricated.

The 8255 is programmed in the monitor program of the CRT terminal. This is programmed in such a way that for the specified keys, the bit pattern of port C upper changes as desired. These Port-C lines are directly used as the input of the decoder Depending upon the status of these three input lines corresponding output of the decoder becomes low. This is the $\overline{CE}$ signal of the corresponding character generator.

In the additional hardware the facility for eight character generator is provided. For these eight character generators, eight chip enable ($\overline{CE}$) signals are generated. At a time only one $\overline{CE}$ signal will be active i.e. only one character generator will be selected, all other character generators has high $\overline{CE}$ signals and hence disabled.

3.2.3 OVERALL SYSTEM OPERATION

The basic system operation without hardware development is already explained in the chapter-II (section 2.4). Because of the additional hardware the overall system operation has changed to multilingual C.R.T. terminal. The circuit diagram of the overall system (including additional hardware developed) is shown in the Appendix-A4 That portion of the existing hardware which are not relevant to this discertation are not shown in this figure

The facilities given in the present system are

(a) Interface to microcomputer through the RS-232C Main port (existing).

(b) Language selection through software (developed).

(c) Increased dimension of the cursor and the character to be displayed (developed).

(d) Superimposition of the two characters which is required in the scripts of Indian languages (developed).

(e) Two screen memories of equal size are used.

The character displayed on the CRT terminal uses the principle of scanning. These characters are formed on the CRT using dots. Two display RAMS (screen memories) are used for storing the displaying characters. Firstly the character received from the keyboard is stored in the RAM. The character code decides in which RAM to store the character. From this display RAM, CPU has to send these characters to the 8275 whenever it demands. CPU takes the action according to character code. If the code is for language selection then it is not transmitted to 8275. The 8275 fills its row buffer with the character code sent by CPU, and then sends code to the character generator. For the character generators this code is a address of the character stored. The bit map corresponding to this address is accessed. This bit map may be a character to be displayed. Depending upon the character generator selected for use, the character may be matra. The character generator outputs the one byte of character to the shift register which transmits this serially to the video Display unit (VDU). One after another all bytes of a character are sent to display, this forms a single character.

Present system works in such a way that it takes the decision only after receiving two characters. This principle is quite useful for displaying Indian languages. After receiving the first character it is placed in one display RAM and the system is displaying alternating both display RAMs. Since only one RAM is having character and other one has blank codes, the display somewhat flickers. When it receives the next character, it decodes it whether to be superimposed on the previous character. If this character is to be superimposed, then it is placed in the second display RAM with the same position as the first character has. Both the characters are displayed alternately with a fast rate and seems to be one character. If the second received character is not to be superimposed on first character, then first character is placed in both the display RAMs in the identical location. The displaying of this character will be brighter as at both the time of alternate displaying same character is displayed. The second character is placed in the next location of first display RAM. Again the next character is checked for superimposition, placed in the corresponding display RAM. This process is repeated. For example displaying of 'आप' word, firstly 'अ' is pressed, then it is placed in Ist screen memory. When `ा' is pressed the system knows that next character is not/to be superimposed on previous one. In both the screen memory 'अ' is placed in same identified location. `ा' is placedin next location of the first screen memory and checks the next character. Similarly 'प' is placed and displayed.

In case of composite characteres e.g. 'तुम' the system receives first character, places it in first memory. As the second character is matra it is placed in the same identified position of the second screen memory. Both 'त ' and ' ु ' are displayed alternately and seems to be 'तु'. After receiving 'म ' the word 'तुम' is displayed.

CHAPTER - IV

C.R.T.MONITOR SOFTWARE DEVELOPMENT

4.1 GENERAL OVERVIEW

The software developed for the CRT monitor uses the interrupt routines. All the available interrupts of the 8085 are used for this purpose. The 8275 CRT controller works efficiently with the help of DMA controller, however since D.M.A. is not used in the existing hardware, the CPU is loaded heavily. Since the existing hardware is modified in this dissertation, this dissertation work does not involve DMA. The important functions of the CRT controller such as refreshing the display, reading the data from the display RAM, are done through interrupt driven routines. These interrupts occurs after every fixed specified time and keeps the display refreshed. Another interrupt is used to read the character received from the microcomputer.

Other software techniques used in this work are, alternate displaying of the characters, selecting any specified character generator. The alternate displaying of the character becomes very important while, using the Indian languages. In these languages matras are present along with the character. This alternate displaying is done using two display RAM as explained in Chapter -III. Different memory pointers are very efficiently handlled in this software.

4.2 SOFTWARE ORGANIZATION

The software is organized in the following manner:

a. Initialization of all the pointers, cursor position, display RAM location stack pointer.

b. The interrupt routines used in development work.

c. Initialization of two display RAM loading blanks in all memory location. Initialization of the 8275 and all other used peripheral chips (8251, 8253,8255).

d. Receiving the character from the keyboard. Decides whether received character is for language selection and if not then in which RAM it has to be stored. Decide the position of the character on the display. Calculation of the cursor position. Determine the position in the row and if it is 80th character, nto carriage return and Linefeed. These process is repeated untill the 17th row. At the end of 17th row scrolling is provided. 17 rows are used because of enlarged curser dimension to accommodate Indian languages with'matras'.

4.3 SOFTWARE DESCRIPTION

4.3.1 Initialization

In this part of the software all the memory pointers used are initialized. Stack pointer is initialized as 6F00H. Both the display memory area are defined. One display memory is initialized as 2100H and another one is 3100H. These are available memory space in the existing hardware. Different pointers used in these sections are;

i)	TPDIS-1	Starting address of Ist display memory.
ii)	TPDIS-2	Starting address of 2nd display memory.
iii)	CURAD	Current address of the cursor.
iv)	CURSX	X-position of the curssor.
v)	CURSY	Y-position of the cursor.
vi)	LAST1	Last address of the Ist display memory.
vii)	LAST2	Last address of the 2nd display memory.

After initializing all the pointers, the display memory space is blanked out by loading 20H in all memory locations. It this is not done the CRT will display some unknown charecters.

4. Next step is to initialize the 8275 chip. This CRT controller chip has wide variety of programming facilities. The list of programming commands available in the 8275 is given in the Appendix-A2 The important commands are

i) Reset command : As this command is written, DMA requests stops, all the 8275 interrupts are disabled. The VSP output is used to blank the screen. HRTC and VRTC are not affected. This command has four parameter bytes. Using these fourbytes 8275 is initialized.

ii) Start display command : All the 8275 interrupts are enabled, DMA requests starts and the video enable status flag is set.

iii) Load cursor command This is a two byte command which places the row number and column number in the cursor positi n registers.

iv) Preset counters command This presets all the internal timing counters corresponding to a screen display position at the top left corner.

The details of these commands are given in Appendix. A-2

Initialization of all the other peripheral chips is then carried out. All the three converters of the 8253 times are used for generating clocks necessary for different circuits. counter0 and counter2 are used as the TXC and RXC of the 8251-1 and 8251-2 respectively. 8251-1 is used to input the character code from keyboard to CRT while 8251-2 is used for creating main RS-232C interface for the communication with microcomputer. Required clocks are obtained by loading the counters property. The mode and command words of 8251 s are loaded as per requirement. The interval reset of the chip is done through programming. For the communication of the 8251-2, which is used as a RS-232 Main port, a dummy character 01 is used, for transmission from CRT to microcomputer, this dummy character eliminates the need of modem signals.

4.3.2 RST 5.5 INTERRUPT

Use of this interrupt driven routine is very straightforward. For the present work the CRT is used only in the on line mode, so what ever the character pressed is displayed only after receiving back from the microcomputer.

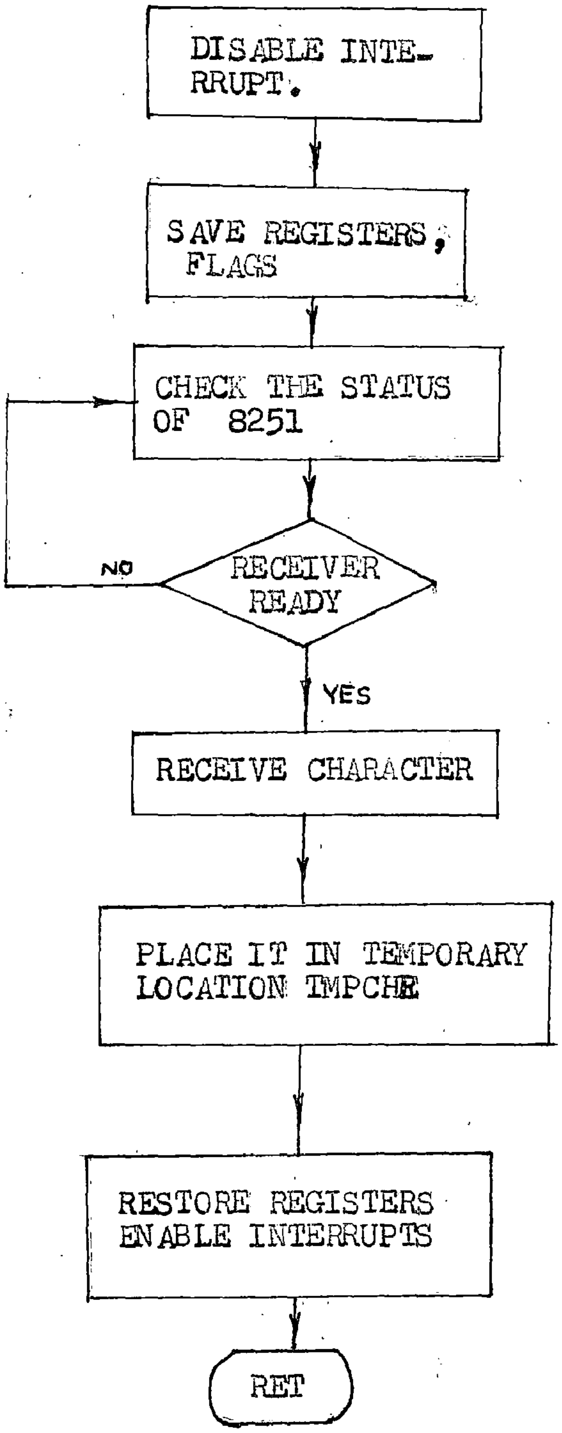

DISABLE INTE-
RRUPT.
SAVE REGISTERS,
FLAGS
CHECK THE STATUS
OF 8251
RECEIVER
READY
NO
YES
RECEIVE CHARACTER
PLACE IT IN TEMPORARY
LOCATION TMPCHR
RESTORE REGISTERS
ENABLE INTERRUPTS
RET

FIG 4.1

The job of receiving this character from microcomputer and storing it in temporary location is done by this interrupt routine. Receiver ready (RXRDY) of the 8251 is connected the RST 5.5 pin of the CPU. This pin senses that a character has to be received and causes the corresponding ISS to be executed.

4.3.3 RST 6.5 INTERRUPT

This routine is used in conjunction with the 8275. This ISS is a frame routine which occurss once every 20 millisecond, since the frame is of 50 Hz. The main perpose of this internupt is to read the status of the 8275. IRQ pin of the 8275 is connected to RST 6.5 Pin of the CPU.

This interrupt updates the CURAD pointer. This pointer has a current starting address of the screen memory. Thus reading the status of the 8275 and updating the current address is the main feature of this interrupt.

For the present work the RST 6.5 interrupt is used for alternate display of the screen memory in addition to the above two purposes. Fig. 4.2 shows the working principle of this routine. TAG is a pointer which is initialized as 01. In this routine after reading the 8275 status the TAG is checked. If it is not 01 then the first screen memory is refreshed. This memory is used to store normal characters. All the pointers are updated according to the first screen memory. The value of TAG is decremented so that in the next interrupt the next screen memory will come into picture.

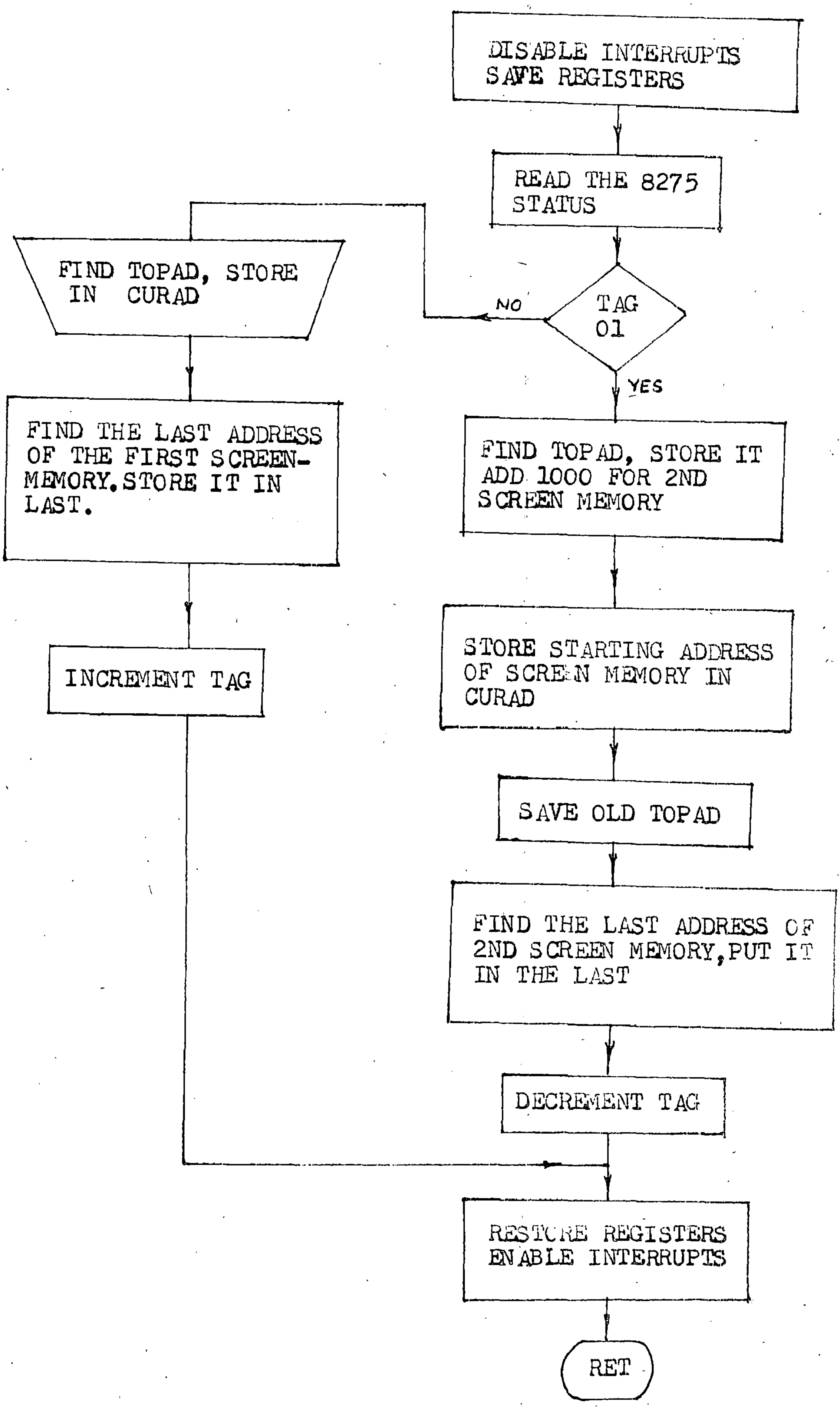

DISABLE INTERRUPTS
SAVE REGISTERS
READ THE 8275
STATUS
TAG
01
NO
YES
FIND TOPAD, STORE
IN CURAD
FIND THE LAST ADDRESS
OF THE FIRST SCREEN-
MEMORY. STORE IT IN
LAST.
INCREMENT TAG
FIND TOPAD, STORE IT
ADD 1000 FOR 2ND
SCREEN MEMORY
STORE STARTING ADDRESS
OF SCREEN MEMORY IN
CURAD
SAVE OLD TOPAD
FIND THE LAST ADDRESS OF
2ND SCREEN MEMORY, PUT IT
IN THE LAST
DECREMENT TAG
RESTORE REGISTERS
ENABLE INTERRUPTS
RET

FIG 4.2

When the TAG has 00 value then it selects the second to be refreshed screen memory. This memory is selected by adding 1000 to first memory, while storing the first memory pointers as it is. All the commonly used pointer are now updated according to the second screen memory. After this again the TAG is incremented for the selection of the first memory.

4.3.4 RST 7.5 INTERRUPT

The RST 7.5 does the functions of the DMA controller. This routine transfers the data from the memory to the row buffers of the 8275 rapidly.

The rate of the execution of this routine depends upon the horizontal frequency applied to the CRT. At a time this routine transfers a full row from memory to the row buffer. In this process the speed of transfer is important. And hence a special technique with POP instructions is used. In this routine the CURAD which is a address of the current lihe to be displayed is transfered to the stack pointer after saving the monitor stack pointer in (D,E). The stack pointer is pointing to the memory location having the data to be transfered. The POP instructions are used to move this data to the 8275. Using fourty POP instructions a full row is filled. This method of data transfer is fully explained in section 2.5. At the end of this routine the monitor stack saved in (D,E) is transferred back to stack.

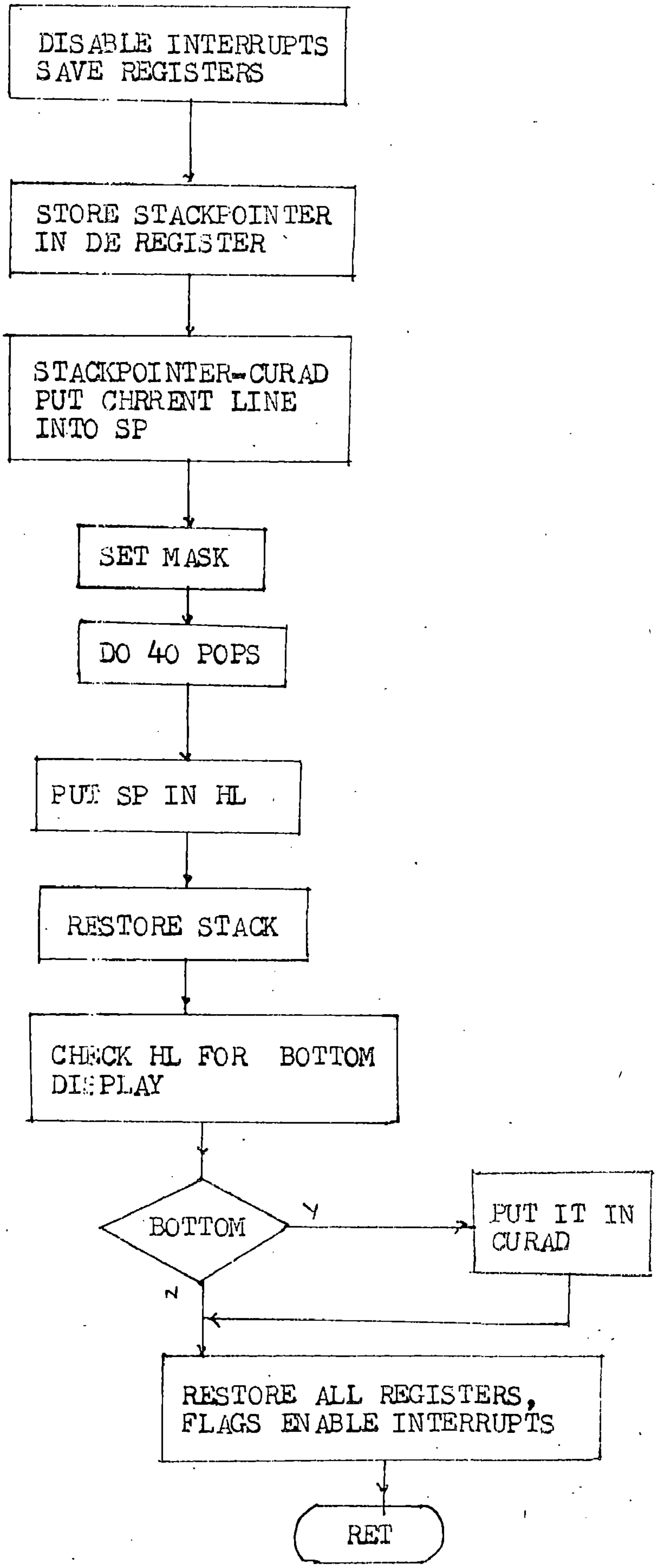

DISABLE INTERRUPTS
SAVE REGISTERS
STORE STACKPOINTER
IN DE REGISTER
STACKPOINTER=CURAD
PUT CHRRENT LINE
INTO SP
SET MASK
DO 40 POPS
PUT SP IN HL
RESTORE STACK
CHECK HL FOR BOTTOM
DISPLAY
BOTTOM
Y
N
PUT IT IN
CURAD
RESTORE ALL REGISTERS,
FLAGS ENABLE INTERRUPTS
RET

FIG 4·3

4.3.5 CURSOR ADDRESS CALCULATION

To find the position of the cursor on screen, the CALCU subroutine is used. This routine takes the 7-position of the cursor and calculates the X-position corresponding. The final cursor position is obtained in the HL pair.

The next part of this is to put a character on the screen and to increment the X-position of the cursor. On the 81st Decimal character this part of the program provides a auto carriage Return and Line feed. This process is done by CHRPUT routine with the help of CALCU. One by one the character is placed on the screen. The current position of the cursor is always noted. At the end of the last row the first row is cleared. Instead a new line can be added. This is a scrolling principle of the CRT terminal when the last character of the last row is reached the starting address of the first line is noted by LOC80 pointer. Using this pointer the first line is cleared. To clear the line a set of PUSH instruction is used, CLLINE routine does the job. This routine disables all the interrupts in the beginning. It takes the starting address of the line which is to be cleared through LOC80 pointer. Stack pointer register is again made use of to transfer blank code 20H to the required memory location using PUSH H instruction forty lines current SP content is saved during this process and restored after this process is over. The second line becomes the first one after the scrolling. If one more line is added then again first line gets cleared. This process can be repeated to any extent.

While executing the above subroutine the care is taken to know the X-position of the cursor. The auto carriage return and Line feed is provided after the end of each line. This process is done by subroutine. CGRT and LNFD. The CGRT routine find the cursor X-position and brings it to initial position. So in the next turn cursor comes to initial position along the X-direction. After this a linefeed works which increments the cursor Y-position by one. Thus the cursor come in the next line of the screen display.

Other subroutines which works in this part of the software are LEFT and ADX. The subroutine LEFT is used to decrement the cursor X-position by one. This subroutine doesnot affect all other pointer. The subroutine ADX helps to calculate the line position as well as cursor X-position on the screen.

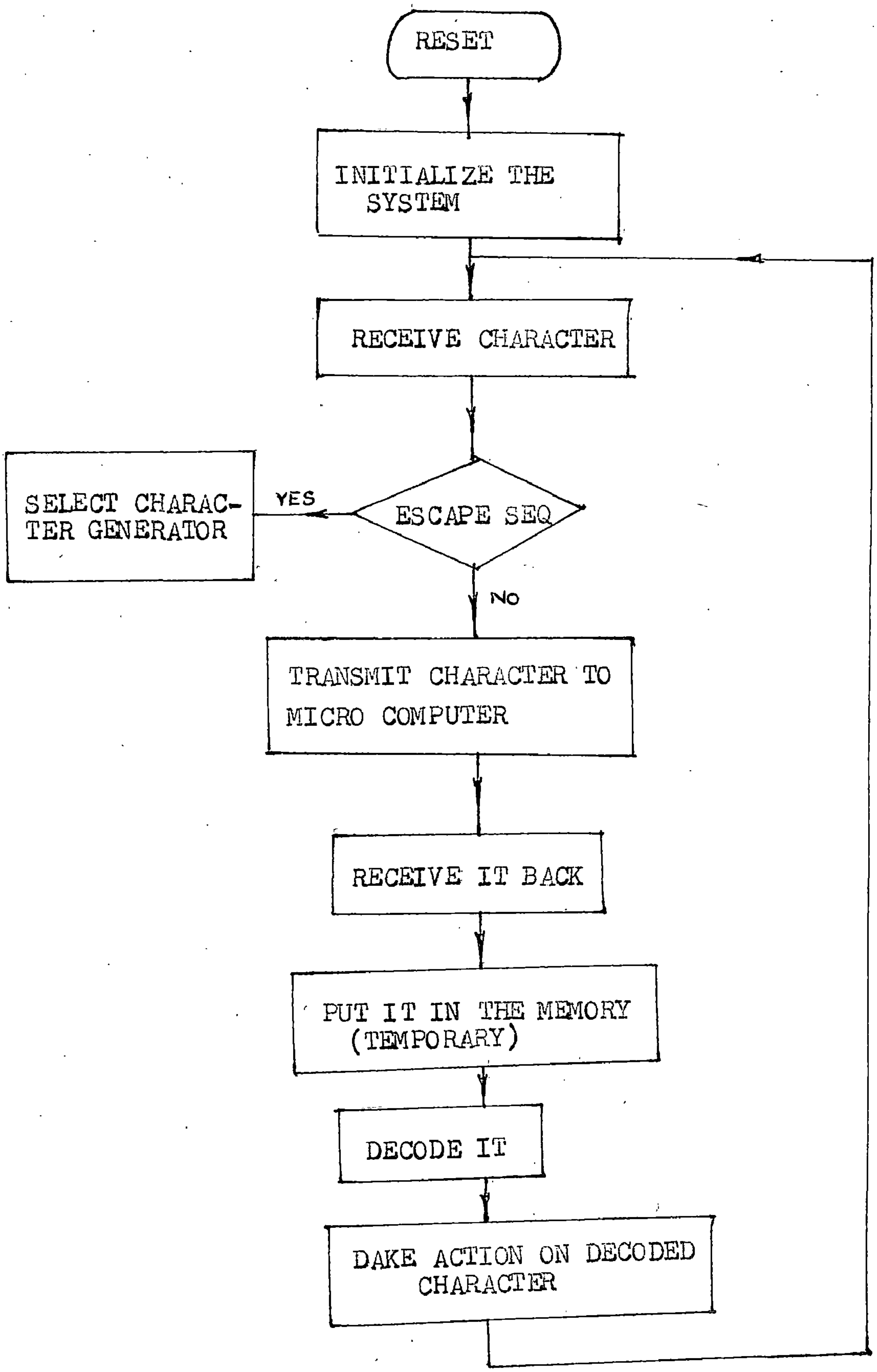
For getting the starting address address of each line a look-up table LINTAB is used. This look-up table has starting address of each line having 80 Decimal characters. The starting address of each line in the second screen memory is obtained by adding the offset of 1000H to each of the starting addresses.

4.3.6 TERMINAL INTERFACE TO MICROCOMPUTER

The character which is to be displayed on the screen is received from the microcomputer. This means the CRT terminal is used only in 'on line'mode. The character is displayed only after receiving the transmitted character.

The terminal monitor takes care that which character is to be sent. ESCAPE sequence character are not transmitted to the microcomputer as these are used for the selection of language to be used. Also at the end of each line a auto carriage Return and line feed provided by CRT for its internal use are not transmitted to the microcomputer. In the same way while receiving the characters from microcomputer the terminal doesnot need a CR and LF after 80th decimal character.

The character received from the microcomputer is first checked whether it is to be placed in the first screen memory or in the second one. All the special characters like 'matras' are to be placed in the second screen memory. The monitor is developed in such a way to have a special character only after a normal one. After receiving the character from the microcomputer it is placed in first screen memory and displayed. It is not displayed on another memory untill the second character is received and checked. Assuming the first character to be normal if the second one is also normal then the first character is placed in both the screen memory in the identical location. So in the alternate displaying, the same character is displayed twice. On the other hand if the second character is a special one (Viz. a Matra) then the first character is placed in the first screen memory while the second character


RESET
INITIALIZE THE SYSTEM
RECEIVE CHARACTER
ESCAPE SEQ
YES
SELECT CHARACTER GENERATOR
NO
TRANSMIT CHARACTER TO MICRO COMPUTER
RECEIVE IT BACK
PUT IT IN THE MEMORY (TEMPORARY)
DECODE IT
DAKE ACTION ON DECODED CHARACTER

BASIC CRT TERMINAL SOFTWARE

FIG 4.4 (a)

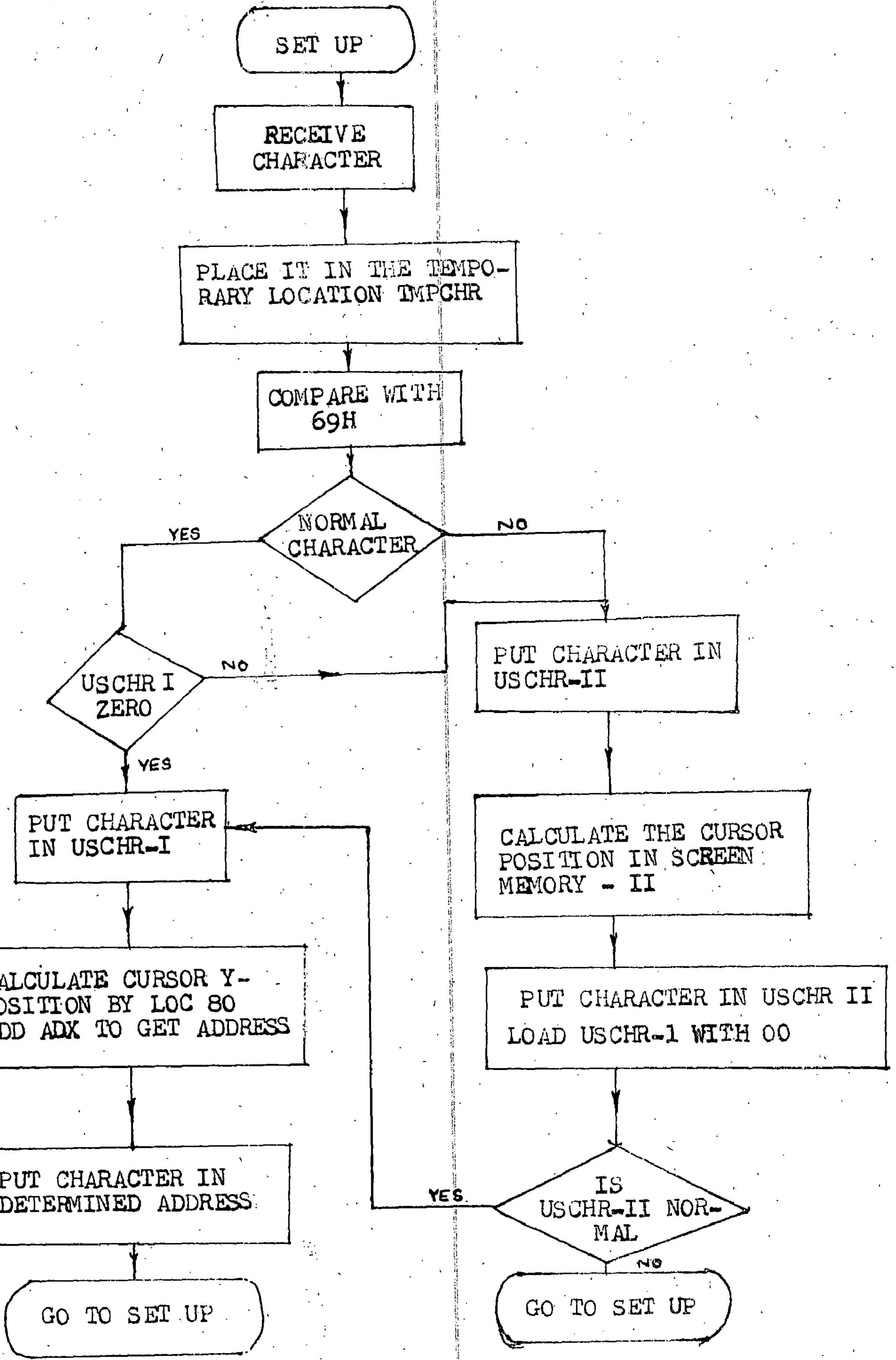

SET UP
RECEIVE CHARACTER
PLACE IT IN THE TEMPORARY LOCATION TMPCHR
COMPARE WITH 69H
NORMAL CHARACTER
YES
NO
USCHR I ZERO
NO
YES
PUT CHARACTER IN USCHR-II
PUT CHARACTER IN USCHR-I
CALCULATE THE CURSOR POSITION IN SCREEN MEMORY - II
CALCULATE CURSOR Y-POSITION BY LOC 80 ADD ADX TO GET ADDRESS
PUT CHARACTER IN USCHR II
LOAD USCHR-1 WITH 00
PUT CHARACTER IN DETERMINED ADDRESS
YES
IS USCHR-II NORMAL
NO
GO TO SET UP
GO TO SET UP

FIG 4.4 (b)

is placed in the other screen memory. In this case the alternated display will have two character to display. This rapid alternate displaying of two characters on the same location will appear to be superimposed. This process is shown in the Figure. 4.4. The subroutines used for above principle are NALTDIS, SET UP along with the help of subroutines used in address calculation.

For placing the character in the screen memory its line address should be known, which is obtained by LOC80 for the first screen memory. For the alternate displaying the character position is same for both the screen memory. The starting address for the second screen memory is obtained by LOC80 pointer. This LOC81 is nothing but the LOC80 plus the offset 1000H. This process is done by the subroutine NEW

Part of the software labled as NCALCU takes the decision of putting the character in the screen memory. The pointer USCHR-1 is used for this perpose. It is initialized as zero, when the character is received it is kept in temporary location TMPCHR. The USCHR-1 is then checked for zero, if it zero then the character received is normal and it is to be placed in first screen memory. Second character is received and checked for special, if it is special then stored it in the USCHR-II, if not then check USCHR-1 for zero, if it is not zero that means the second character is normal, otherwise special. Thus stored in the second screen memory obtained by offset 10000H and first screen memory.

The characters like CR, LF and Backspace are considered to be normal.

CHAPTER V

MICROCOMPUTER INTERFACE WITH CRT TERMINAL AND TEXT EDITING FACILITIES

The CRT terminal used for the present work is used only in the line mode. Hence the microcomputer interface becomes important. The microcomputer used for the present work is VMC 86 Kit. This Kit has INTEL8086 CPU in maximum mode, it provides a sufficient amount of user's memory area. All the peripheral chips required for this work are available with this kit.

5.1 INTERFACE WITH CRT TERMINAL

5.1.1 GENERAL

The microcomputer has a serial communication with the CRT terminal. The universal synchronous asynchronous receiver transmitter (USART) 8251 is used for this communication. The terminal is interfaced with the microcomputer through the RS 232 interface.

Since the CRT terminal works only after receiving the character from the microcomputer, this microcomputer should respond for every character. The baud rate of both, the microcomputer and the terminal is kept as 2400.

The serial communication is done through the USART 8251. It is programmed accordingly. This chip has the pins named Receiver ready and Receive data. These pins are important while receiving the data from the CRT. Receiver ready pin shows the status of the 8251. This status is continuously checked, whenever this pin is having high level the data is received through Receive data. Similarly while transmitting the data two pins namely Transmitter ready and Transmit data are active. The data is transmitted only after the pin receiver ready is high. The general working principle of USART is shown in the Figure 5.1. The 8251 USART is used in the Asynnonous mode for the present work. The received character is transmitted back to terminal. To acknowledge this, terminal sends one dummy character 01H. The microcomputer continuously checks for the real character.

5.1.2 DATA STORING

All the characters sent by the terminal are received by microcomputer and it takes a decision accordingly. Only required characters are stored. All the data is stored in a particular format. The memory location of the microcomputer used for storing is 02000H onwards.

The data storing used in the present work is line oriented. The capacity of each line is specified as 160 decimal characters. This number is fixed as in case of Indian languages, one row of the terminal may have 160 decimal characters, because of superimposition. One row of the terminal display is specified as a line of the data. Whenever the row of the terminal is changed by CR and LF the line number of the data is also changed. The pointer PTRB keeps

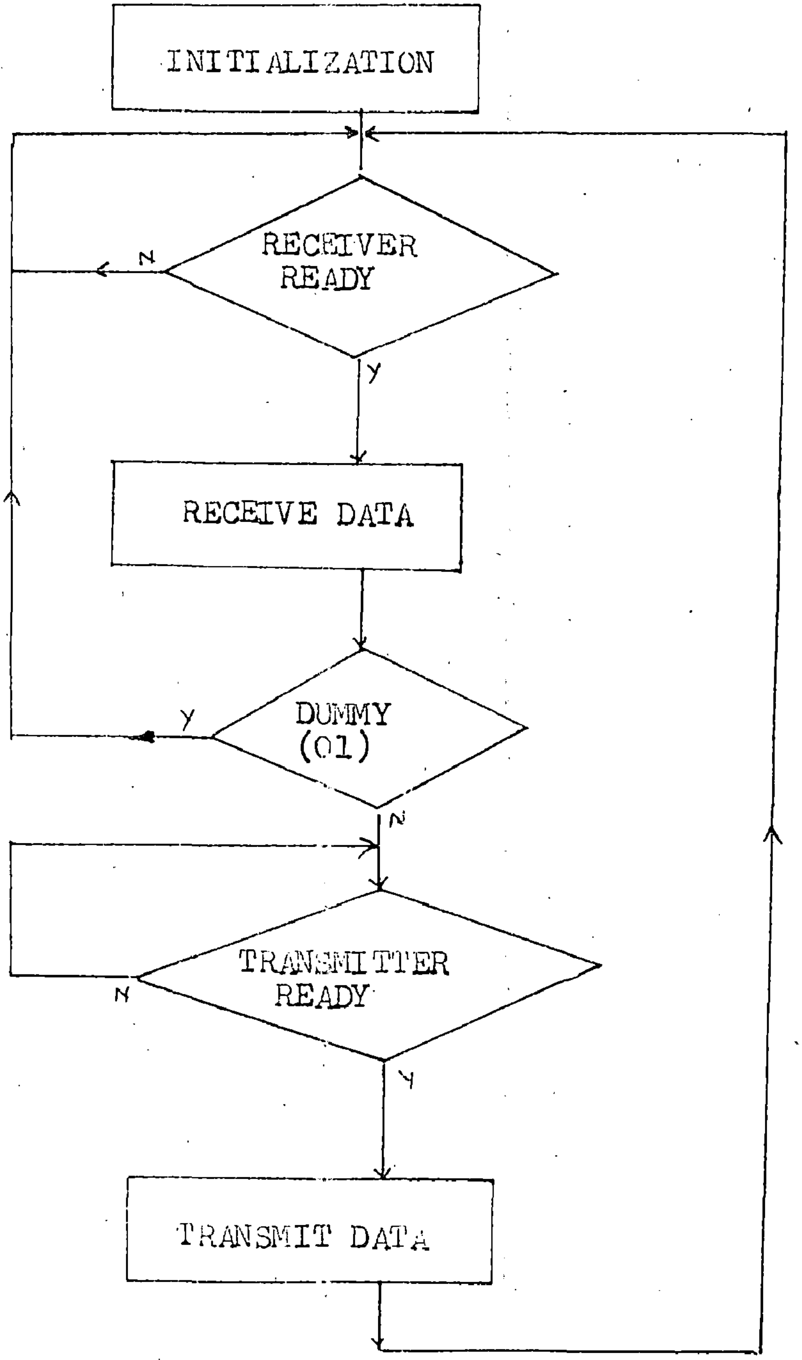

INITIALIZATION
RECEIVER READY
N
Y
RECEIVE DATA
DUMMY (01)
Y
N
TRANSMITTER READY
N
Y
TRANSMIT DATA

FIG. 5.1

a constant record of line number. Address of this pointer is 6110H onwards in the difference of two **bytes.** The line number is accounted in two bytes. As per the lines are stored, corresponding line number recorded. Correcponding to this line numbers attempts are made to find out the starting address of each line. Another pointer called NOLN is used to know the number of lines available in the memory. Initial value of this pointer is one and it is incremented after every next received line. The line change in the data storage is recognized by two characters CR and LF. After the sequence of these two characters all pointers are updated.

The software is developed on the microcomputer having 8086 CPU. This has string manipulation facilities. Use of these facilities is done effectively. The 8086 has two index registers memory SI and DI. These two registers are convinienty handled to solve the perpose of storing editor data. While storing the data, operations to be done are updating line numbers, recording starting address of each line SI and DI registers help a lot for these operations.

The general idea of the software developed for the data storage is shown in the figure 5.2 . The received character is first checked for the End of File (EOF) i.e. CTRL/Z. Other than this the received character may be special (matras, or character to be superimposed on other character) or normal . Two seperate counts are made to have a record of normal and special characters present in the line.

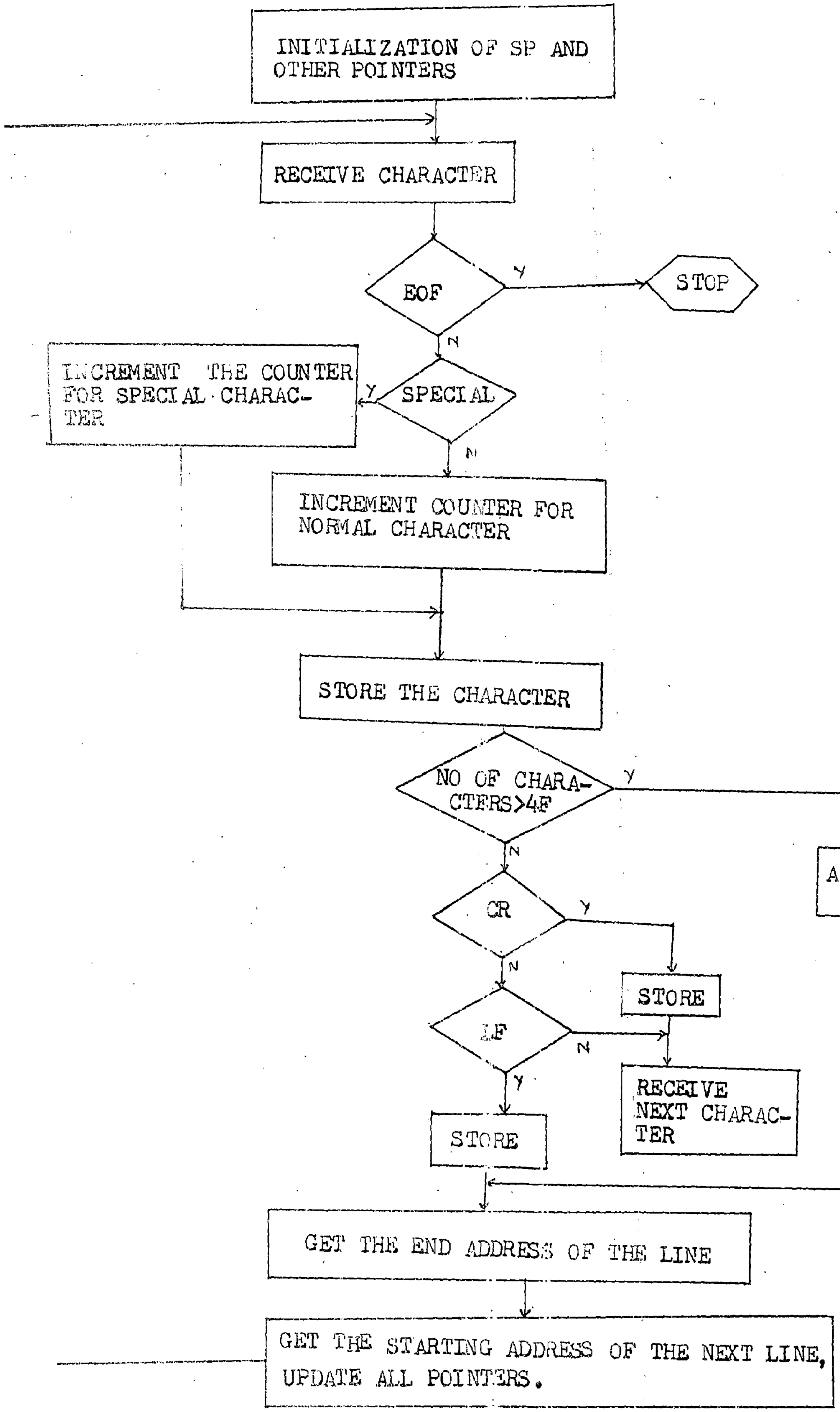

INITIALIZATION OF SP AND OTHER POINTERS
RECEIVE CHARACTER
EOF
Y
STOP
N
INCREMENT THE COUNTER FOR SPECIAL CHARACTER
Y
SPECIAL
N
INCREMENT COUNTER FOR NORMAL CHARACTER
STORE THE CHARACTER
NO OF CHARACTERS>4F
Y
N
AUTO CR, AND LF
CR
Y
N
STORE
LF
N
Y
RECEIVE NEXT CHARACTER
STORE
GET THE END ADDRESS OF THE LINE
GET THE STARTING ADDRESS OF THE NEXT LINE, UPDATE ALL POINTERS.

FIG. 5-2.

Total number of characters present in one line are sum of normal and special characters. Thus knowing the number of characters present in one line, starting address of the next line can be easily obtained. Thus for all the lines same process is repeated. At last the EOF is recognized by the microcomputer. This EOF closes the file and the CPU comes out of the data storing routine.

5.2 TEXT EDITOR

Text editing is nothing more than the creation and modification of textual material (such as program, letters and scripts). Text editor is a cornerstone of the computer applications. Development of the text editor is an important phase of this work. Using the text editor attempts are made to modify the data (which is a script) present in the file. In user point of view the editor developed for the present work is very straight forward. All conventional facilities are present in this editor. The editor program is developed on the microcomputer having 8086 CPU. 8086 CPU provides additional advantages for writting an editor.

The editing functions developed in the present work are

i) PRINT This function reads a line or a set of lines and display it on the terminal.

ii) INSERT This function provides a facility of inserting a line in the file of data in any place.

iii) DELETE This function deletes any line available in the file.

iv) COPY This function provides a facility of copying any line to any other line position.

v) TRANSFER This function transfers a line from the specified position to other specified position.

All the above functions developed are line oriented. The one more function is developed which operates as character oriented editor.

vi) APPEND This is a character oriented editor function. This has following facilities.

(a) INSERT This function inserts the character or a set of characters in the specified line.

(b) DELETE This function deletes the character or a set of characters from the specified line.

(c) REPLACE This function replaces the character or a set of characters of the specified line.

The general structure of the text editor is shown in the fig.5.3After entering in the edit mode the CPU constantly checks for the edit command. It takes action according to the specified command.

5.2.1 PRINT command.

This editor function is the most important one. This command is useful to constantly see the original data, and after modifying it, to see the modified data. The program developed for this command is able to print a single line or a set of lines.

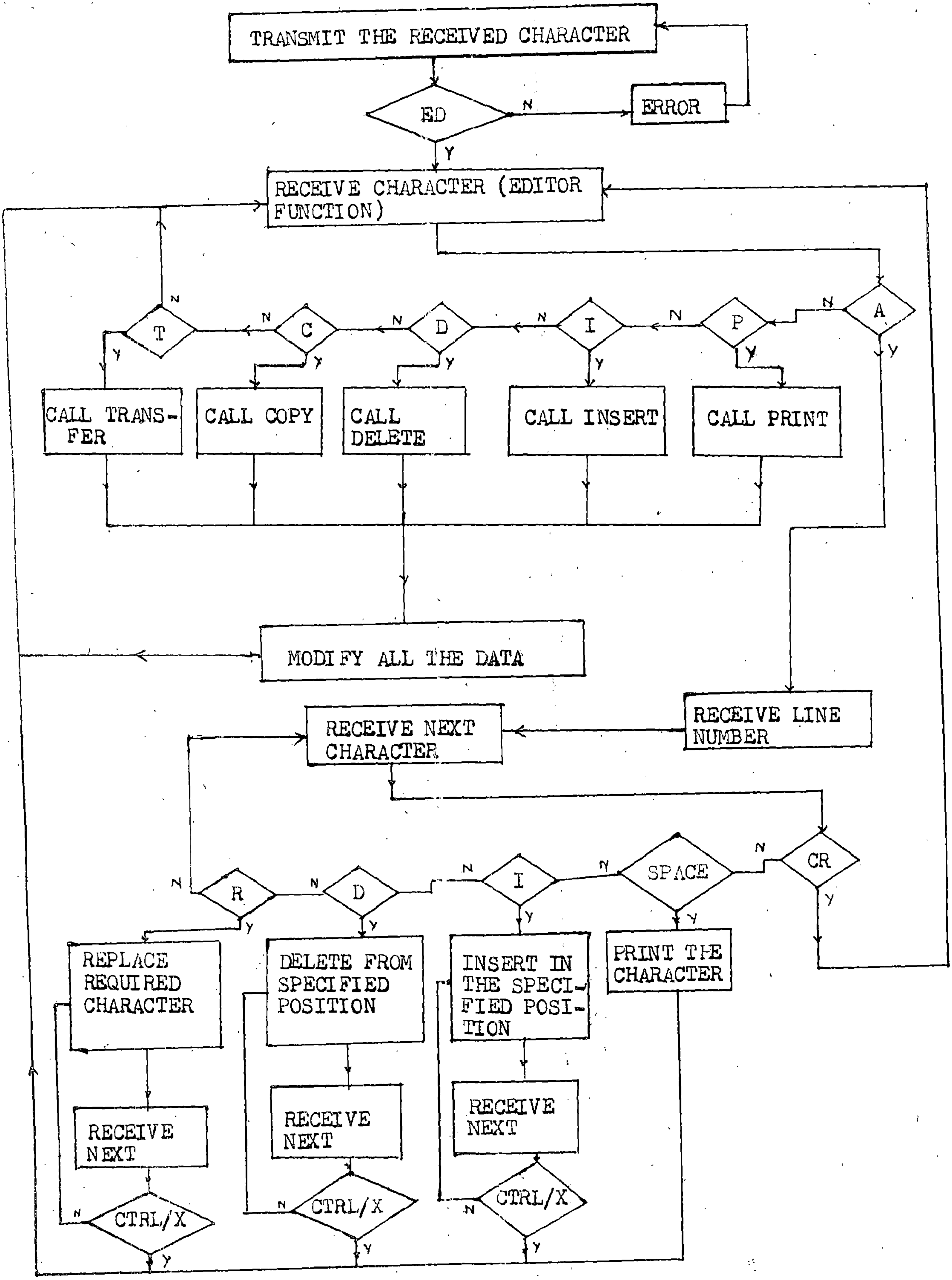

TRANSMIT THE RECEIVED CHARACTER
ED
N
ERROR
Y
RECEIVE CHARACTER (EDITOR FUNCTION)
A
N
Y
P
N
Y
I
N
Y
D
N
Y
C
N
Y
T
N
Y
CALL TRANSFER
CALL COPY
CALL DELETE
CALL INSERT
CALL PRINT
MODIFY ALL THE DATA
RECEIVE LINE NUMBER
RECEIVE NEXT CHARACTER
CR
N
Y
SPACE
N
Y
I
N
Y
D
N
Y
R
N
Y
REPLACE REQUIRED CHARACTER
DELETE FROM SPECIFIED POSITION
INSERT IN THE SPECIFIED POSITION
PRINT THE CHARACTER
RECEIVE NEXT
RECEIVE NEXT
RECEIVE NEXT
CTRL/X
N
Y
CTRL/X
N
Y
CTRL/X
N
Y

FIG. 5·3

Once the system enters the editor mode, to print a line P and a line number is to be specified. If this line numbers is greater than the number of lines present in the file CPU comes out of this routine and checks for the other command. This command is completed only after pressing CR. To print a set of lines, P is followed by first line number, comma, and second line number with CR. Again if any one of these line number is greater than the number of lines present in the file CPU doesn't execute it. One more facility is provided in this function that is to compare the two specified line numbers. The second line number specified must be greater than the first one. Otherwise again this routine is not executed. The PRINT command is given in the following fashion.

i) P l1 ↓

ii) P l1 , l2 ↓

where l1 and l2 are line numbers.

The functioning of this command is as shown in the figure. 5.4. It transmitts the command code then the line number. The line number code is converted in decimal for convenience. After this line number if the next character is CR then the specified line number is displayed on the screen. The specified line is first checked for within the file line numbers. If it is more than whatever available in the file this routine ends. Instead of CR if second character is a line number then again its availability in the file is checked. Also the

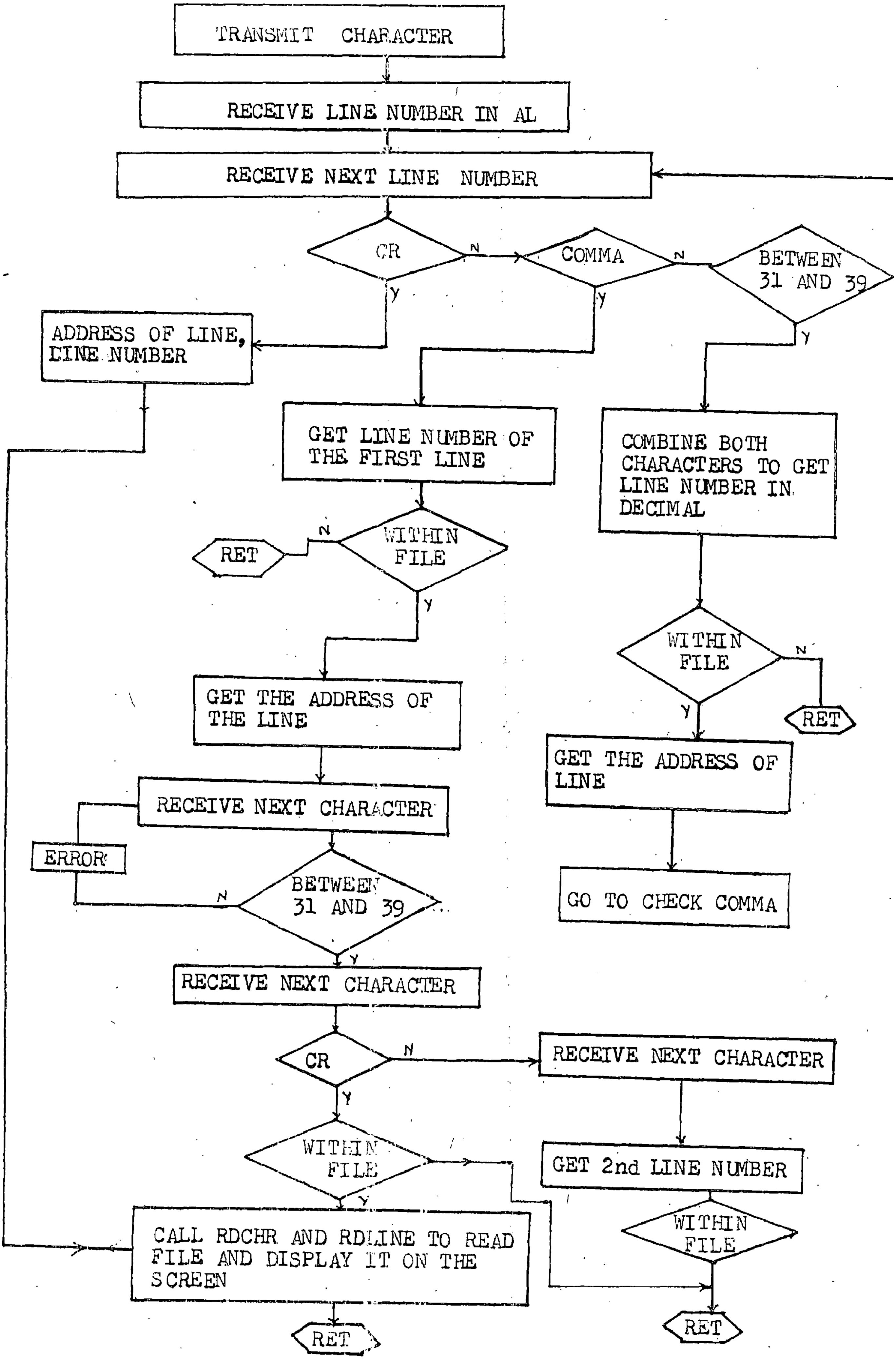

TRANSMIT CHARACTER
RECEIVE LINE NUMBER IN AL
RECEIVE NEXT LINE NUMBER
CR
N
COMMA
N
BETWEEN 31 AND 39
Y
Y
Y
ADDRESS OF LINE, LINE NUMBER
GET LINE NUMBER OF THE FIRST LINE
COMBINE BOTH CHARACTERS TO GET LINE NUMBER IN DECIMAL
RET
N
WITHIN FILE
Y
WITHIN FILE
N
RET
Y
GET THE ADDRESS OF THE LINE
GET THE ADDRESS OF LINE
RECEIVE NEXT CHARACTER
ERROR
N
BETWEEN 31 AND 39
GO TO CHECK COMMA
Y
RECEIVE NEXT CHARACTER
CR
N
RECEIVE NEXT CHARACTER
Y
WITHIN FILE
GET 2nd LINE NUMBER
Y
CALL RDCHR AND RDLINE TO READ FILE AND DISPLAY IT ON THE SCREEN
WITHIN FILE
RET
RET

FIG. 5.4

comparison between the two lines is done. If second line has less number than first one then this routine ends. This routine can print the line maximum upto 99 decimal if it is available in the file. The received ASCII codes are converted into the decimal for convenie ce, the facility of maximum two codes for each line is given, i.e. the largest acceptable number is 99 decimal.

For converting the received line numbers to decimal the routine ADJUST is used. This routine also gives the starting address of the specified line. If the line number is of two digits then this combines these two ASCII codes and converts it to decimal number. Use of this routine is also taken to decide the specified line within the file. The working of this routine ADJUST is shown in the Figure.5.5. To get the line number and the starting address of that line correspondingly, the relationship between two pointers is developed. The pointers having a line number has address 100H more than the pointers having the starting address. Thus if any one of the pointer is known the other can be easily found out.

The next important routine used in the PRINT command is RDLN. This routine helps to read the data of the specified line. This routine takes the help of another routine RDCHR which reads one by one charactér in the file. Fig. 5.6. shows the working of this routine. This routine gets the starting address of the first line and the second line.

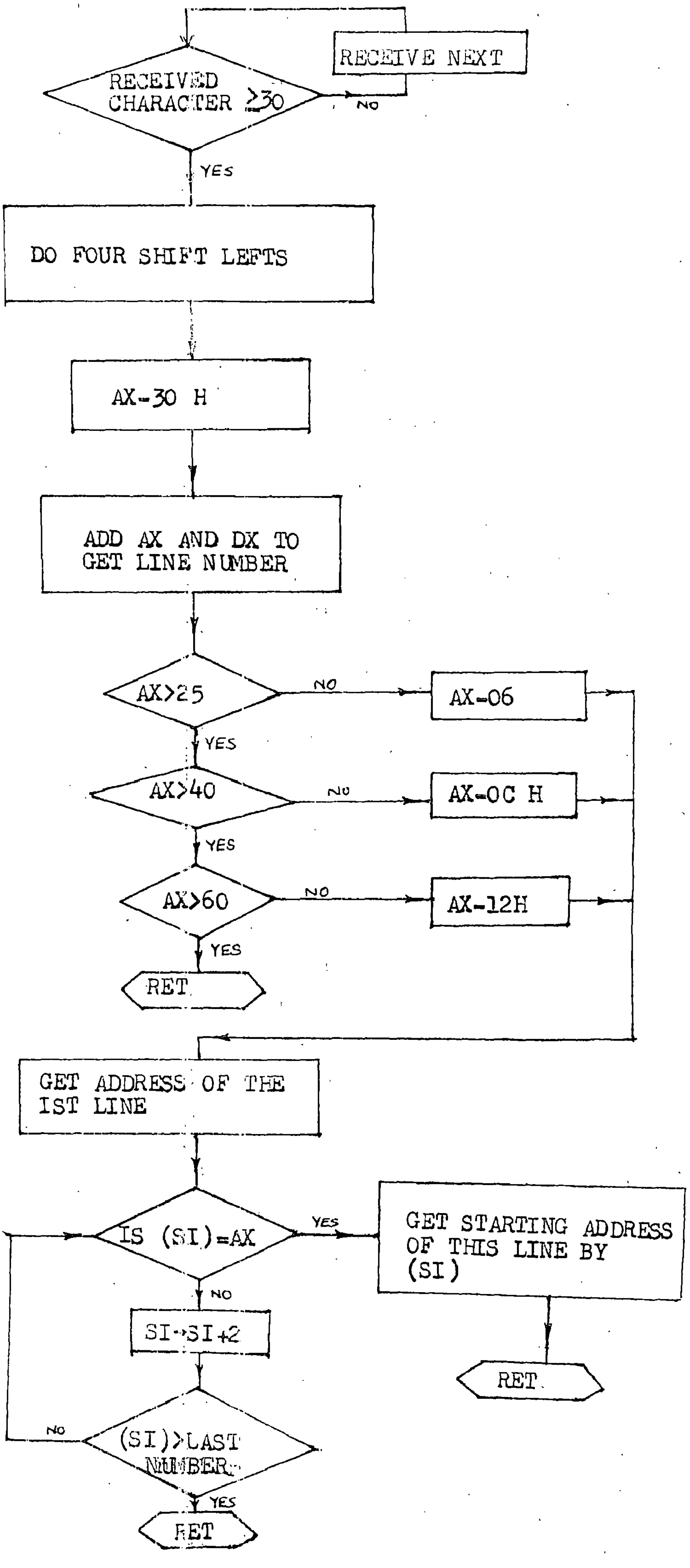

RECEIVE NEXT
RECEIVED CHARACTER ≥30
NO
YES
DO FOUR SHIFT LEFTS
AX-30 H
ADD AX AND DX TO GET LINE NUMBER
AX>25
NO
AX-06
YES
AX>40
NO
AX-0C H
YES
AX>60
NO
AX-12H
YES
RET
GET ADDRESS OF THE IST LINE
IS (SI)=AX
YES
GET STARTING ADDRESS OF THIS LINE BY (SI)
RET
NO
SI→SI+2
(SI)>LAST NUMBER
NO
YES
RET

FIG 5.5

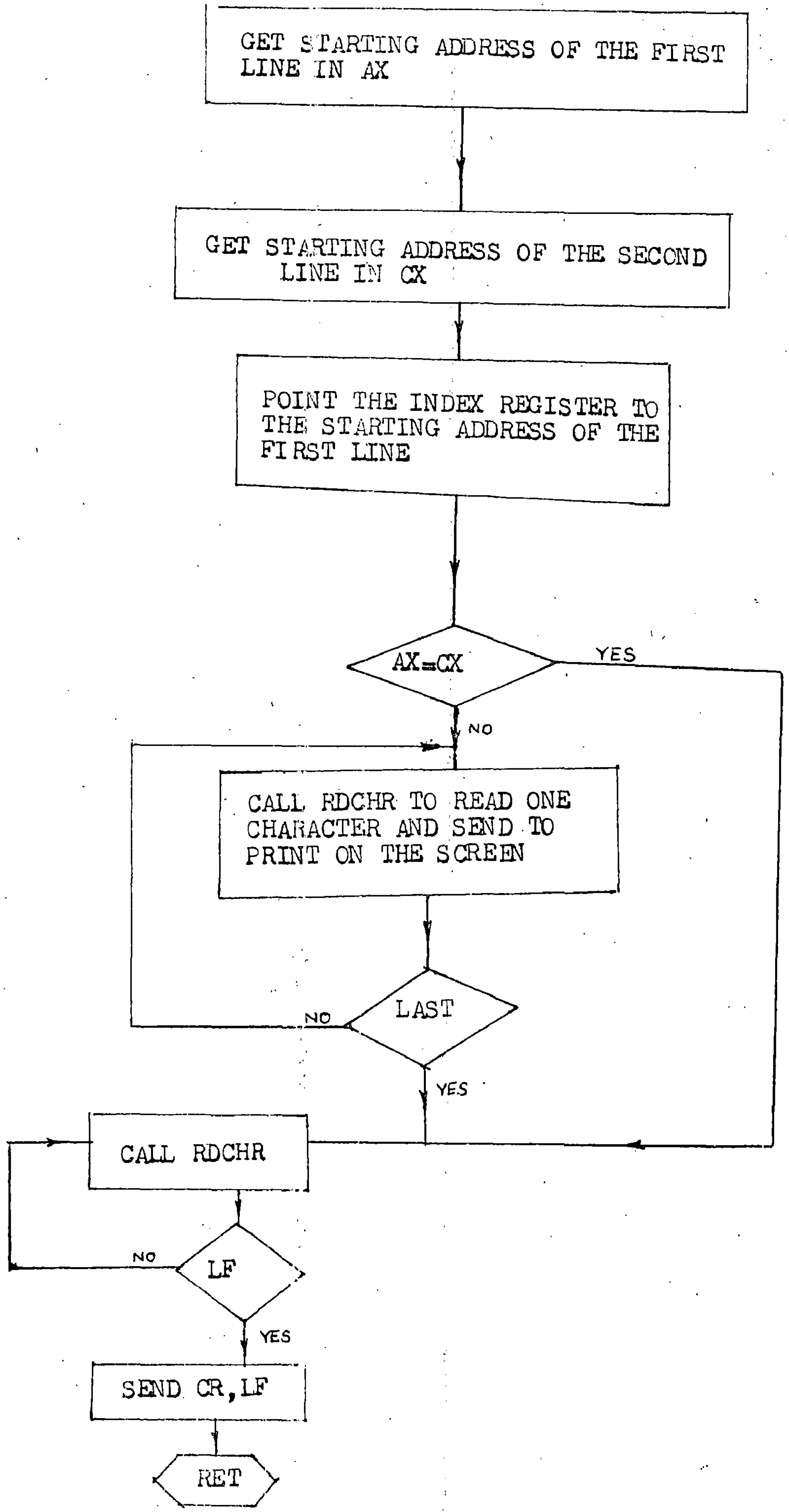

GET STARTING ADDRESS OF THE FIRST LINE IN AX
GET STARTING ADDRESS OF THE SECOND LINE IN CX
POINT THE INDEX REGISTER TO THE STARTING ADDRESS OF THE FIRST LINE
AX=CX
YES
NO
CALL RDCHR TO READ ONE CHARACTER AND SEND TO PRINT ON THE SCREEN
LAST
NO
YES
CALL RDCHR
LF
NO
YES
SEND CR, LF
RET

FIG. 5·6

In the absence of second line, both the pointers have got same same starting address of first line. Then comparing these two starting addresses the character is read and sent one by one. If both the pointers have got different starting addresses then one by one character is printed until the last line is reached. Once the last line is reached the character to be printed on screen should be a LF.

The RDCHR routine reads one character from the specicied address given by the index register S1. After reading this character it calls TRANSMITTER routine which causes to transmit it to terminal.

5.2.2 INSERT command

This command helps to insert a line to the specified position. The line number where the line is to be inserted is specified by the user. The line which is present at that position shifts downward and gives the space to newly added line. The command is given as Iℓ) where ℓ is the line number. Figure 5.7 shows the operation of this command. For this routine specific memory location are used as EDITBUFFER. Whatever the data to be inserted, is kept in this buffer. The number of characters kept in this buffer is recorded.

This routine first sends the command code i.e.I. Then it expects the line number where the line is to be inserted. To calculate this line number ADJUST routine is used. Once the linenumber is obtained, its starting address

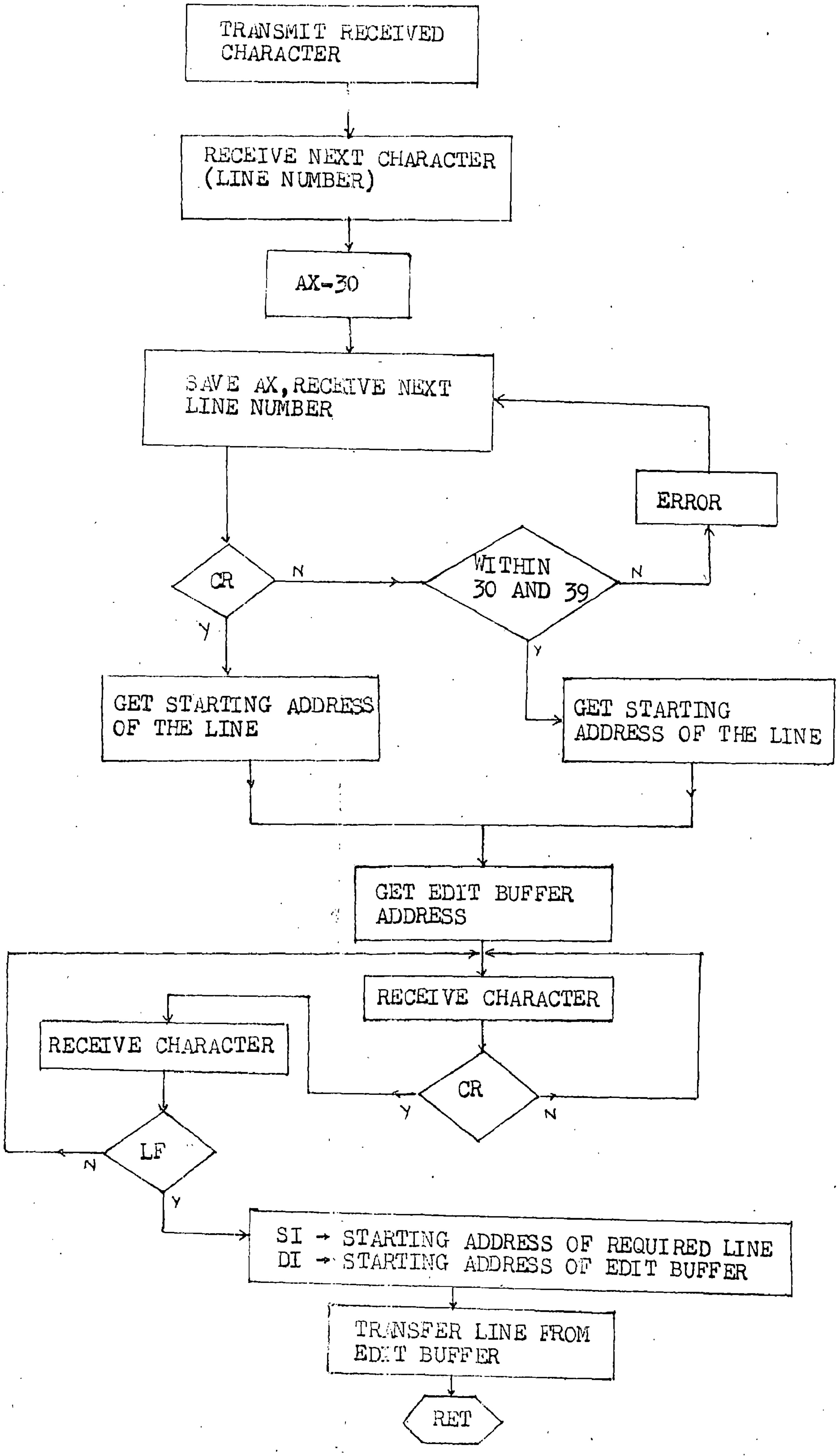

TRANSMIT RECEIVED CHARACTER
RECEIVE NEXT CHARACTER (LINE NUMBER)
AX-30
SAVE AX, RECEIVE NEXT LINE NUMBER
ERROR
CR
N
Y
WITHIN 30 AND 39
N
Y
GET STARTING ADDRESS OF THE LINE
GET STARTING ADDRESS OF THE LINE
GET EDIT BUFFER ADDRESS
RECEIVE CHARACTER
RECEIVE CHARACTER
CR
Y
N
LF
N
Y
SI → STARTING ADDRESS OF REQUIRED LINE
DI → STARTING ADDRESS OF EDIT BUFFER
TRANSFER LINE FROM EDIT BUFFER
RET

FIG. 5.7

s easy to get. After this the routine expects the data hich is to be insertéd. The data to be inserted in is ept in the edit buffer. One pointer is used to record he amount of received characters. Thus one by the characters re received by the edit buffer. This command is terminated y CR. Once this code has been received, this routine takes urther actions. Out of this actions the first one is to eadjust the starting addresses of each line. The pointers hich are having these starting addresses are updated ccordingly.

The next step is to make the space for the line to e inserted. This thing is done by using two index registers. ne index register is having the address from which location he data is to be shifted, the another register has a address here to shift the data. This address is nothing but the address f source register plus the no. of characters received. Once his has been achieved, the complete file data is shifted ownward. The space is made in the specified position for a pecified number of characters.

The last step of this routine is to put the inserted ata present in the edit buffer to the main file. Again for his two index registers are convinientły used. The pointer aving a record of total numbers of lines in the file is ncremented by one.

5.2.3 DELETE Command

This command works whenever the required line is to be deleted. The command code and the line number is to be specified. This command is written as Dλ↲ . where D is command code and λ is the line number. The working principle of this command routine is shown in the figure.5.8.

This routine first sends the command code and expects the line number. Once it is received the ADJUST is active and gives the starting address of the line. Then this shifts this line to the edit buffer. During this process the number of characters present in the specified line are recorded. Once this full line is shifted to edit buffer the next step is to update the pointers which are having starting addresses of the lines and corresponding line numbers. The total number of lines present in the file is decremented by one.

The next step of this routine is to change the starting address of each line. Since the number of characters deleted (number of characters in the specified line) are known the starting addresses can easily be changed.

The last step is to move the full data of file upward. The address of the deleted line becomes the destination of the data and the address of the line next to deleted line, becomes a source address. The end of this process is a EOF character.

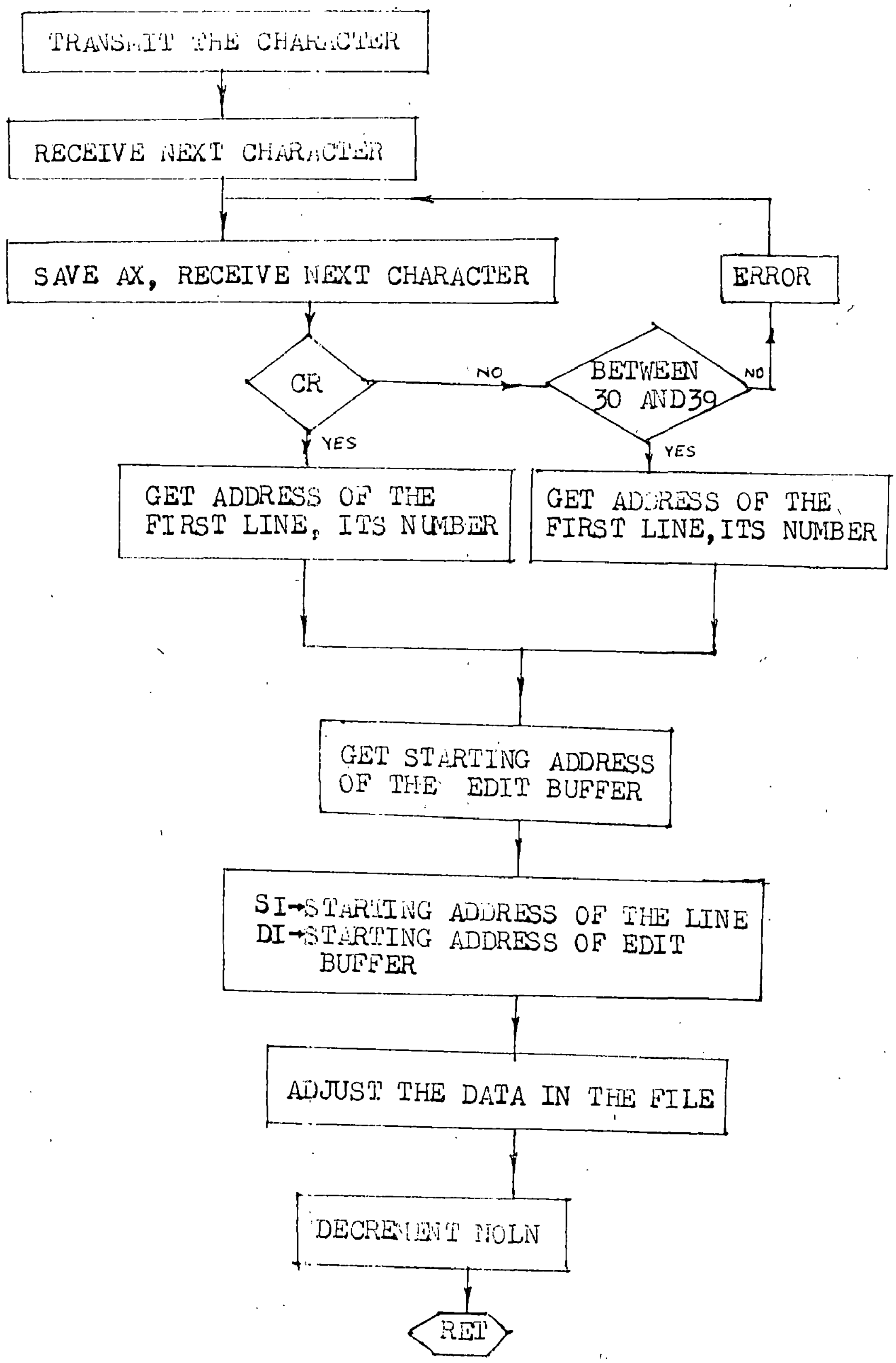

TRANSMIT THE CHARACTER
RECEIVE NEXT CHARACTER
SAVE AX, RECEIVE NEXT CHARACTER
ERROR
CR
NO
BETWEEN 30 AND39
NO
YES
YES
GET ADDRESS OF THE FIRST LINE, ITS NUMBER
GET ADDRESS OF THE FIRST LINE,ITS NUMBER
GET STARTING ADDRESS OF THE EDIT BUFFER
SI→STARTING ADDRESS OF THE LINE
DI→STARTING ADDRESS OF EDIT BUFFER
ADJUST THE DATA IN THE FILE
DECREMENT NOLN
RET

FIG.5.8

5.2.4 COPY command

This command is useful whenever the user want to have the specified line to some other specified position. This command makes the use of the INSERT command principle. Only the variation is that the line to be inserted is available in the file itself. The code used for this command is (l_1, l_2 where l_1 and l_2 are the line numbers present in the file.

Operation of this command is explained in the figure 5.9. After transmitting the command code this routine expects a source linenumbers. Then it expects a comma and finally the destination linenumber. The starting addresses of both the lines is obtained. The full source is first copied to edit-buffer. During the copying process its number of characters present are noted. Then this routine updates all the pointers having starting addresses. The starting addresses of each of the line is changed accordingly. Once this has been achieved this routine shifts the full data downward to provide the space for new line. While shifting the data downward the source index register is having the address of EOF character, on the other hand the destination index register is having a address which is equal to the addition of source address and the number of characters in the line. Both the index registers are decremented after each shift. The check of this process is the starting address of the destination line.

Once the data is shifted downward the last but important stage is achieved which is to move the line from

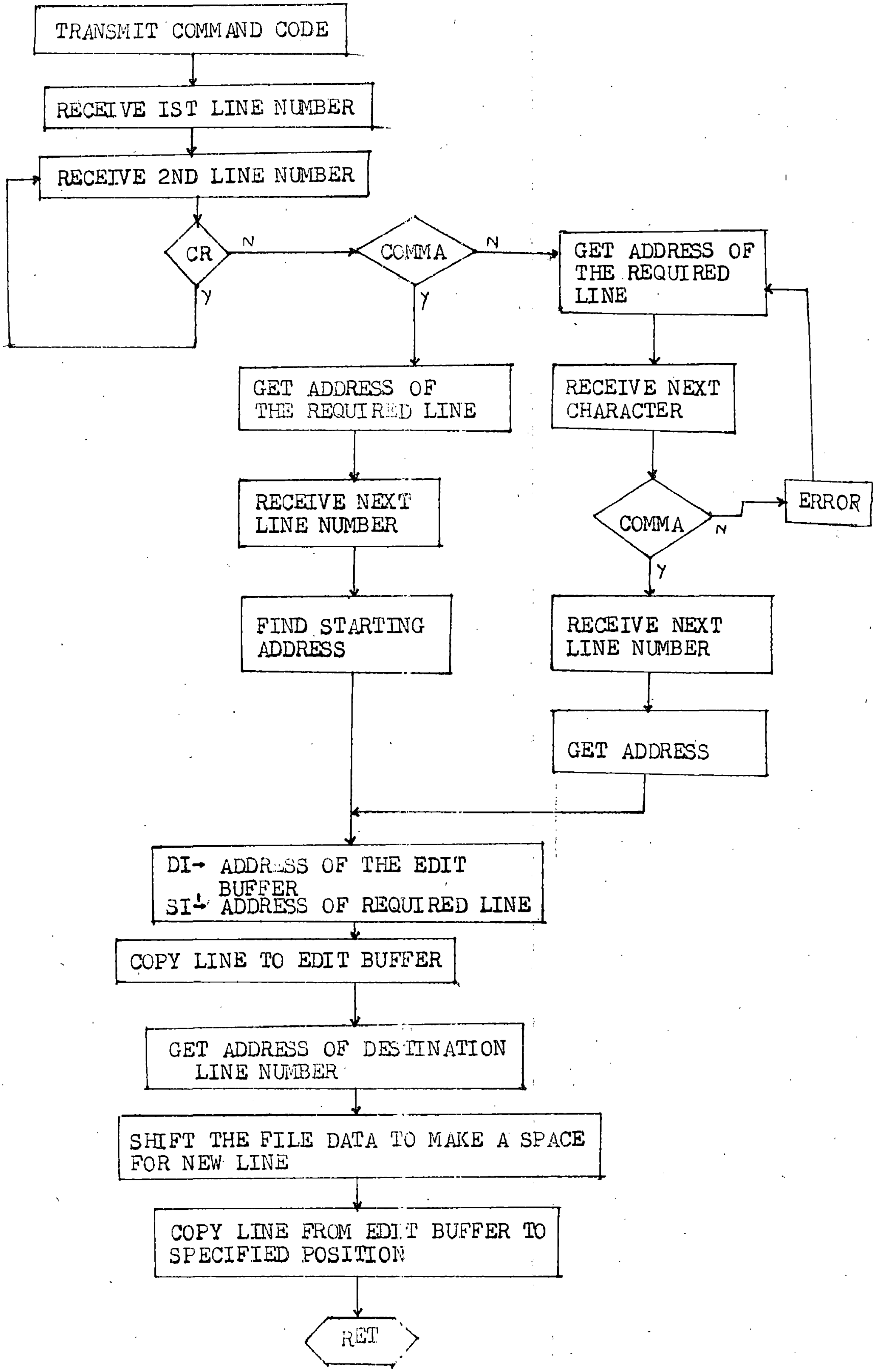

TRANSMIT COMMAND CODE
RECEIVE IST LINE NUMBER
RECEIVE 2ND LINE NUMBER
CR
N
Y
COMMA
N
Y
GET ADDRESS OF THE REQUIRED LINE
GET ADDRESS OF THE REQUIRED LINE
RECEIVE NEXT CHARACTER
RECEIVE NEXT LINE NUMBER
COMMA
N
Y
ERROR
FIND STARTING ADDRESS
RECEIVE NEXT LINE NUMBER
GET ADDRESS
DI← ADDRESS OF THE EDIT BUFFER
SI← ADDRESS OF REQUIRED LINE
COPY LINE TO EDIT BUFFER
GET ADDRESS OF DESTINATION LINE NUMBER
SHIFT THE FILE DATA TO MAKE A SPACE FOR NEW LINE
COPY LINE FROM EDIT BUFFER TO SPECIFIED POSITION
RET

FIG. 5-9

edit buffer to the specified location. The pointer having a record of number of lines present in the file is incremented by one.

5.2.5 TRANSFER command

This editing function is same as the COPY command, only the difference is that the source line is deleted. This command is a combination of both the DELEVE and the INSERT command. The code used for this is T ℓ1,ℓ 2 ↲ where ℓ 1 and ℓ_2 are linenumbers.

Working principle of this command is as shown in the figure 5.10. This command first transmitts the command code and waits for the character which is a line number of the source line. The line number may be of one or two digits if the linenumber is of two digits then second character must be a ASCII code between 31H and 39H. For the single digit line number second character must be a comma. Once this comma accepted it waits for the second character which is a destination line. This may also be a single digit or two digit line number. The last accepting character is CR.

Once the CR has reached the main part this routine starts functioning. The starting address of the line which is to be transfered is noted. From the main file this line is copied to the Edit buffer. After this since the addresses of each line has been changed, all the pointers having starting addresses are updated. The number of characters present in the line(which is to be transfered) are known. The data is shifted upward since the line has been deleted.

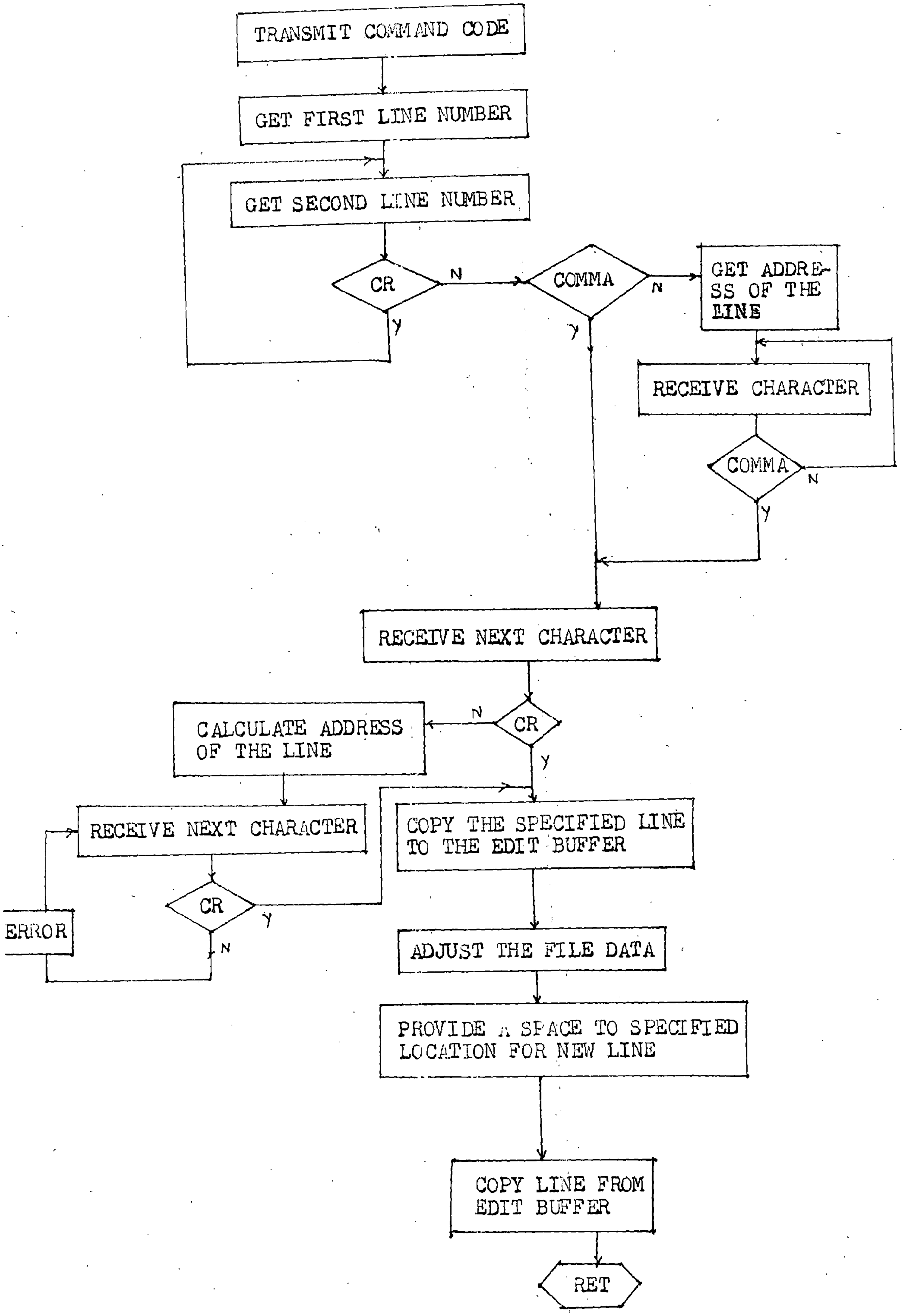

TRANSMIT COMMAND CODE
GET FIRST LINE NUMBER
GET SECOND LINE NUMBER
CR
N
Y
COMMA
N
Y
GET ADDRESS OF THE LINE
RECEIVE CHARACTER
COMMA
N
Y
RECEIVE NEXT CHARACTER
CR
N
Y
CALCULATE ADDRESS OF THE LINE
RECEIVE NEXT CHARACTER
CR
Y
N
ERROR
COPY THE SPECIFIED LINE TO THE EDIT BUFFER
ADJUST THE FILE DATA
PROVIDE A SPACE TO SPECIFIED LOCATION FOR NEW LINE
COPY LINE FROM EDIT BUFFER
RET

FIG. 5-10

The next part of this routine is to insert the line to specified position from the edit buffer. For this process the space is made to the specified position. Since the number of characters to be inserted is known, the starting addresses of each of the lines are changed properly. The full data of the file is shifted downward using two index registers. Updates all the pointers. The last stage of this routine is to transfer the line from edit buffer to the specified location.

5.2.6 APPEND command

This is the last but very important facility given in the present Text Editor. This is the only character oriented command. The structure of this command is as shown in the figure 5.11. This command works with the help of three other subcommands namely INSERT, DELETE, REPLACE.

A) The code for this command is A $\ell \downarrow$ where ℓ is a one digitcr two digit line number. Once the linenumber is obtained it waits for the next subcommand. If without any subcommand CR is pressed then this routine end. Other than the subcommands if the space bar is pressed then from the specified line this routine reads the data using ROLN and displays it on CRT terminal. One space code displays one character from the line. Thus one by one character the whole line can easily be displayed on the screen. The difference of this operation with the PRINT command is that, the PRINT command displays a full specified line without any

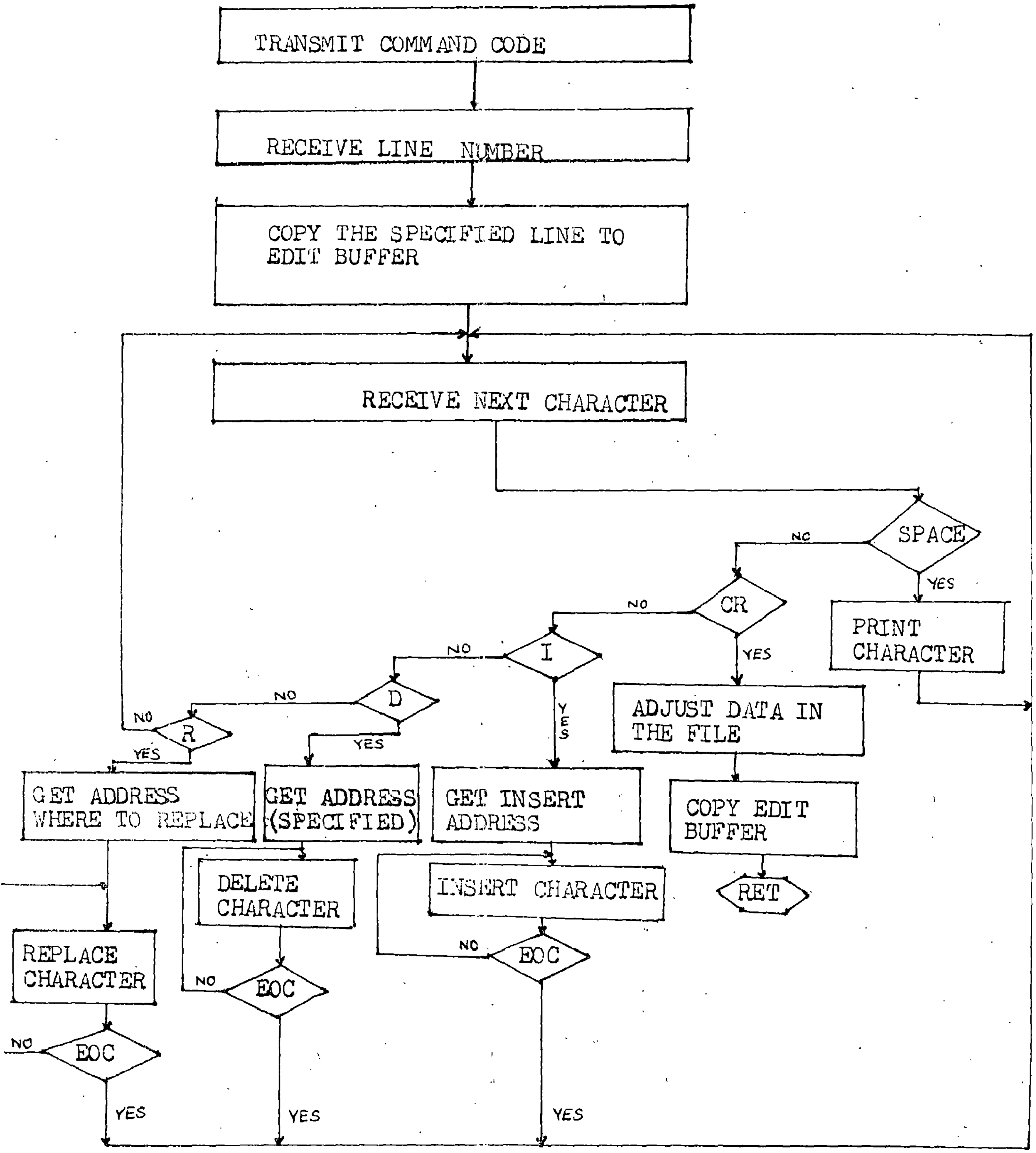

TRANSMIT COMMAND CODE
RECEIVE LINE NUMBER
COPY THE SPECIFIED LINE TO EDIT BUFFER
RECEIVE NEXT CHARACTER
SPACE
NO
YES
PRINT CHARACTER
CR
NO
YES
ADJUST DATA IN THE FILE
COPY EDIT BUFFER
RET
I
NO
YES
GET INSERT ADDRESS
INSERT CHARACTER
EOC
NO
YES
D
NO
YES
GET ADDRESS (SPECIFIED)
DELETE CHARACTER
EOC
NO
YES
R
NO
YES
GET ADDRESS WHERE TO REPLACE
REPLACE CHARACTER
EOC
NO
YES

FIG. 5-11

control on the characters on the other hand APPEND command with pressing space bar has control over display the character by character. Whenever the command is to be terminated the CR is pressed as any position of the error cursor.

For this routine also one edit buffer is specified. The specified line is kept in the edit buffer. While printing the characters one by one, it reads the data from this edit buffer only.

If the I . follows by the A) code or the space code then the INCERT is called to execute. This routine has to insert the character in the edit buffer from the specified location. If the I is followed just offer the APPEND code the characters are inserted from the start of the specified line. On the other hand if I is follows some space codes then from that position the required characters will be inserted. Figure 5.12 shows the working of the INSERT routine. The principle used here is almost same as used in the INSERT line command. Once the required characters are inserted the INSERT command is terminated by CTRL/X code. After this, INSERT routine changes the line accordingly and counts the number of character increased. This ends the work of INSERT routine, and starts the work of APPEND routine again. APPEND routine increments the starting addresses of each of the line. It updates all the required pointers. It shifts the data downward for required amount . And last step it does is to transfer the line from edit buffer to the specified (original) position.

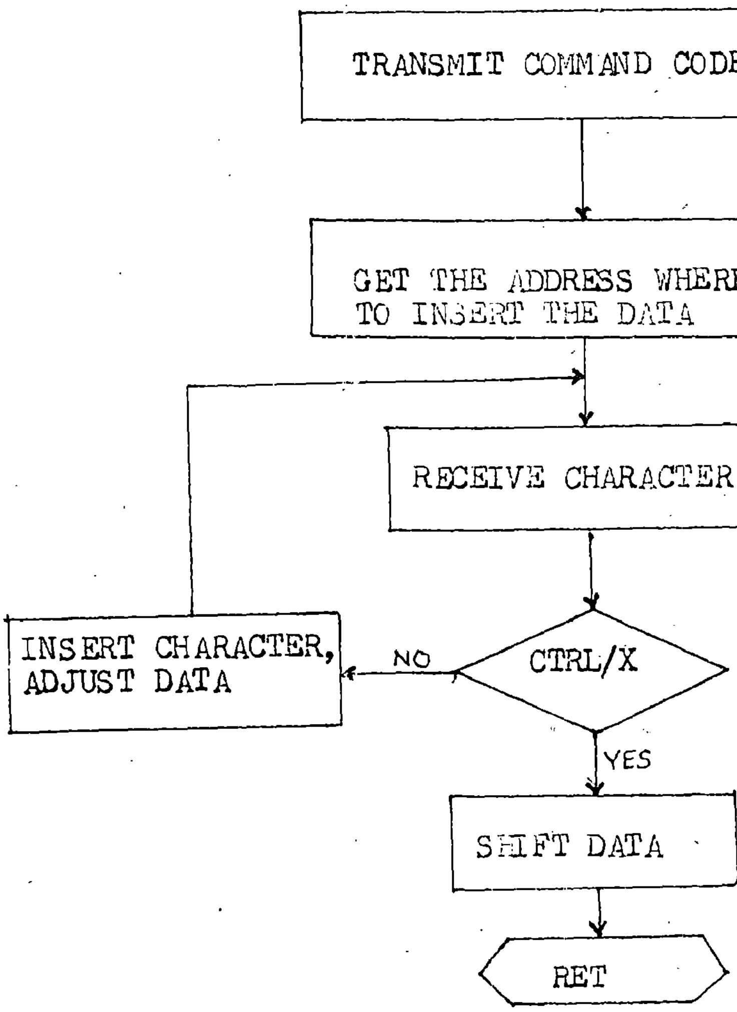

TRANSMIT COMMAND CODE
GET THE ADDRESS WHERE TO INSERT THE DATA
RECEIVE CHARACTER
CTRL/X
NO
INSERT CHARACTER, ADJUST DATA
YES
SHIFT DATA
RET

FIG. 5·12

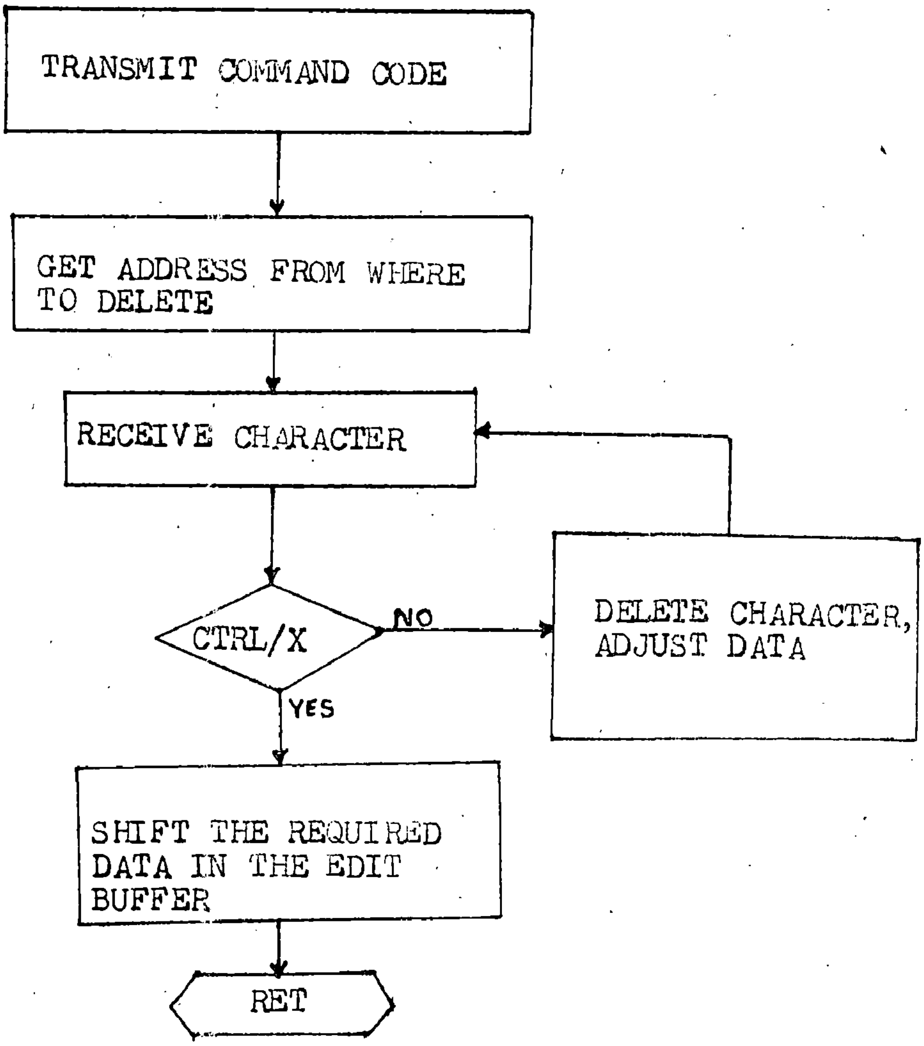

TRANSMIT COMMAND CODE
GET ADDRESS FROM WHERE TO DELETE
RECEIVE CHARACTER
CTRL/X
NO
DELETE CHARACTER, ADJUST DATA
YES
SHIFT THE REQUIRED DATA IN THE EDIT BUFFER
RET

FIG. 5·13

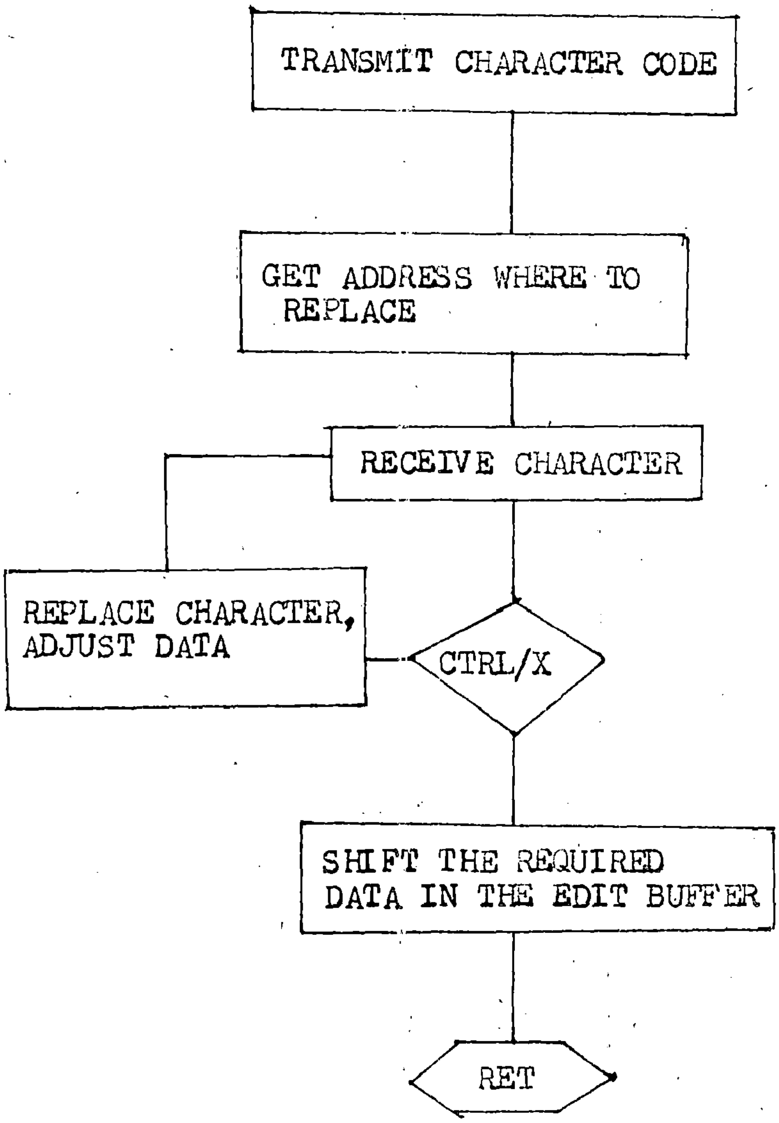

TRANSMIT CHARACTER CODE
GET ADDRESS WHERE TO REPLACE
RECEIVE CHARACTER
REPLACE CHARACTER, ADJUST DATA
CTRL/X
SHIFT THE REQUIRED DATA IN THE EDIT BUFFER
RET

FIG. 5-14

B) Second part of the APPEND routine is to delete the characters from the specified line. This DELETE routine works when D follows by APPEND code. The description of this routine is given in the figure 5.13. The starting address from where the characters is to be acted is obtained by the pointer. The characters can be deleted from any place within the line. If just after the APPEND code D is accepted then the starting address is same as the starting address of the line. After pressing D, next characters are deleted using the space code. Each of the space code indicates the next character to be deleted. This command is terminated by CTRL/X code.

After this the working of remaining APPEND routine starts. It changes all the starting addresses of each of the line updating all the pointers are equally important. Once this has achieved the next step is to move the data upward by required amount. The modified line in the edit buffer is then copied to the original position.

c) The Third subfunction of the APPEND routine replaces the required characters. The code used for this sub function is R. This code may follow the APPEND code or space code. And thus accordingly it gets the starting address where to modify the characters. This command is explained in the figure 5.14 Unless the command terminating code is pressed this routine replaces the character one by one. At the end of this sub-function the main APPEND routine has just to COPY the line from edit buffer to the original line position.

CHAPTER - VI

PRINTER INTERFACING

6.1 INTRODUCTION

Printers are used for obtaining hard copies of the outputs of the equipme.its to which the printers are interfaced. Printers have possibilities of giving the hard copies using different letters and numeral types. Generally printers are classified into two main types, viz. i) Serial Printer and ii) Line Printers. Based on the method of character generation printers may also be classified as impact printers and non-impact printers. Impact printer strike the media with the printing element to form a character. Non impact printers generally use thermal or electrostatic techniques that do not require impact character formation techniques provide another way of classifying printers as character printers and matrix printers. Character printers use fully formed characters, whereas matrix printers use combination of either dots or lines to form complete character.

Serial printers are also called Dot matrix printers. Print speed in the case of dot matrix printers is normally specified in terms of character per second (CPS). The common speeds are 50 to 200 CPS. Print quality of these printer is comparatively poor.

Line printer is the term normally used to denote a band, chain or drum printer. Here, printing is done line by line instead of character by character. These printers are faster than serial printers.

6.2 DOT MATRIX SERIAL IMPACT PRINTER MODEL 7500

6.2.1 The MODEL 7500 printer is designed to operate through software control supplied from any general purpose computer. It can be used for RS-232C and standard centronics parallel interface. RS-232C can be used because of interface board which is housed in the printer above the mother card. This card converts the serial bits into byte and then through the parallel interface card the data is printed.

MODEL 7500 dot matrix printer has character set of 96 in normal front, 64 in character generated graphics. 14 in European, 64 in Greek front and control characters. All these can be accessed by control codes. Apart from software codes, some hardware facilities are provided from DIP switches. This printer provides facilities of printing bold characters, line feed pitch, underline printing, dot addressable graphics. These facilities can be used by giving proper command. These commands are enumerated in the APPENDIX C-2.

6.2.2 The MODEL 7500 printer can be used either for serial (RS-232C) interface or parallel interface. For the use of serial interface it has a fixed factory set boud rate 300. On the main card of this printer, there is one more card which converts this serial data into parallel. The searial interface of this printer performs the operation at slower rate.

Another interface available with this printer is centronics parallel interface. This interface is used to input 8 bit parallel data. Commonly used control signals in

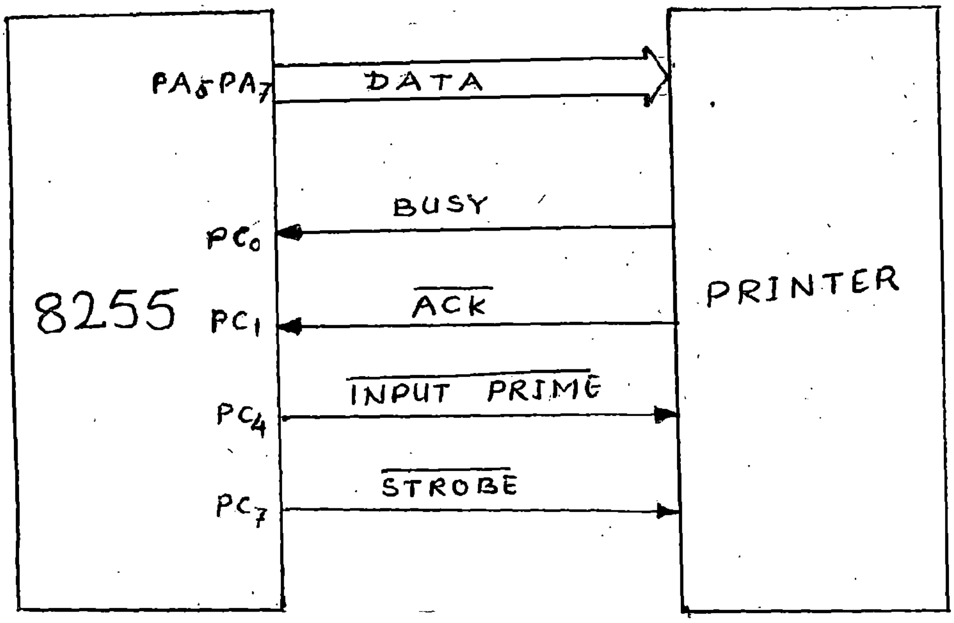

PA0-PA7
DATA
BUSY
PC0
8255
PC1
ACK
INPUT PRIME
PC4
STROBE
PC7
PRINTER

FIG 6.1

this interface are, $\overline{\text{INPUT-PRIME}}$ $\overline{\text{DATA STB}}$ INPUT-BUSY. A standard 36 pin connector is used, pin assignment is shown in the APPENDIX. C-1.

For the present work centronics parallel interface is used. The working principle of this interface is shown in the figure 6.1. 8255 programmable peripheral interface is used as the I/O chip in the 8086 microcomputer kit. This chip is made use for interfacing the printer. This chip is used in mode-0. Port-C lower is used as input port to read necessary signals from the printer. Port C upper is used as output to output control signals to the printer. Port A is used as a media of sending data to the printer from computer.

Working of this interface can be easily understood with the help of figure 6.2. Timing diagram is shown in the figure. 6.3.

6.3 SCRIPT DEVELOPMENT

If the characters supplied to the printer are ASCII code, then it prints these character in a normal way programmed in monitor. The size and type of these characters depends upon the control command received by the printers. This way of printing is possible only for English script. Also few special script of Greek and Graphic symbols are possible on this printer.

To develop a script of Indian language, special technique is used. This technique uses a principle of dot addressable

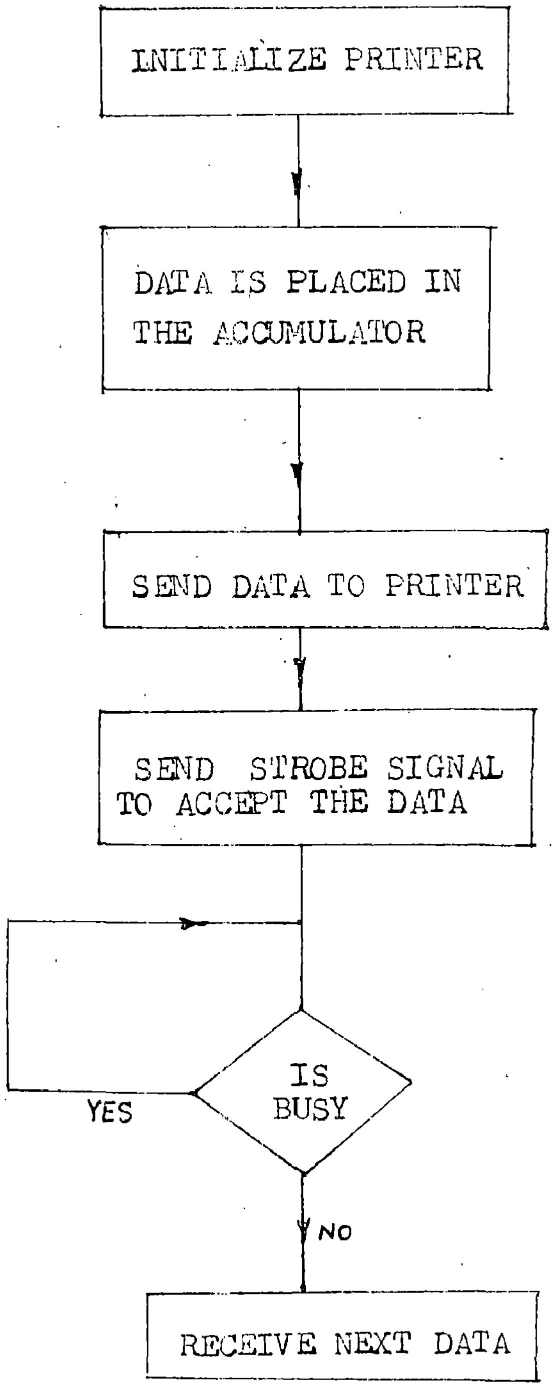

INITIALIZE PRINTER
DATA IS PLACED IN THE ACCUMULATOR
SEND DATA TO PRINTER
SEND STROBE SIGNAL TO ACCEPT THE DATA
IS BUSY
YES
NO
RECEIVE NEXT DATA

FIG 6·2

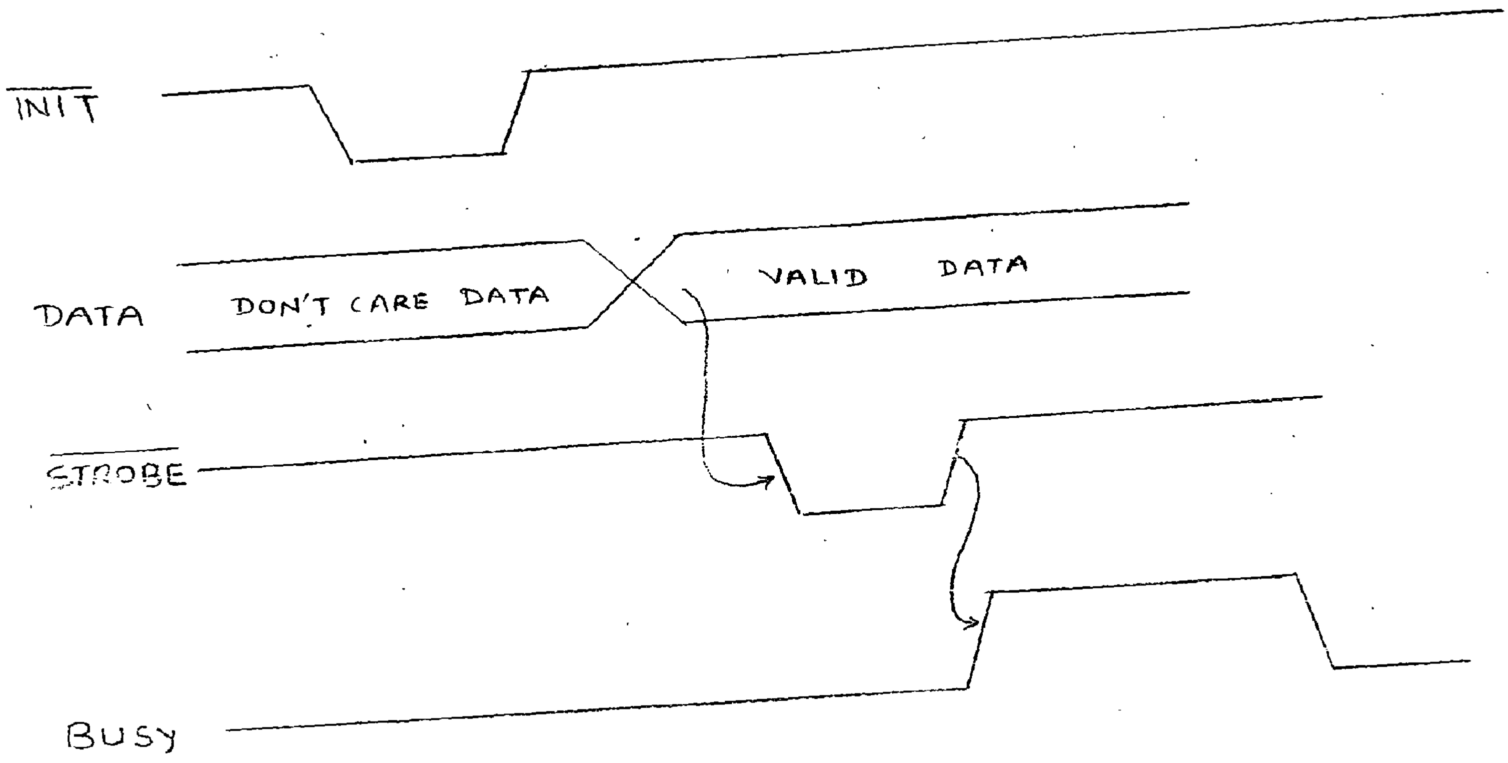


FIG 6·3

graphics. This special facility available in this printer provides a wide variety of applications.

Once the basic printing principle is understood script development becomes clear. As shown in the figure 6.4(a) print head has 8 dot tips. Bit image data and dot tips are related. If a bit is 1, the print head fires and if a bit is 0, it does not fire. The required print head is fired by supplying a proper bit map. For example if FFH data is supplied to the printer, it prints a vertical line as shown in the fig. 6.4(b).

The above principle is made use of for developing the script of Devnagari. This script is developed using dot addressable graphics. The command to enter this mode is ESC S(n1)(n2),(n3)(n4). ESC S is a control command character, and (n1)...(n4) are hexadecimal ASCII numbers (30 to 39), which together gives the length of data that follows for printing. For the present work each character is having fixed dimensions. Character is accomodated in the 8x9 dot matrix format. All the basic characters (without matra) are formed in a 5X9 dot matrix. Since the printer head has 8 dot tips, height of each character would be maximum of 8 data. In case of Devnagari script, the character may have matra, which are accomodated in the same format. This reduces the size of character.

A bit map for each character in Devnagari is developed. This bit map for character 'ज्ञ' is shown in the figure.. 6.5

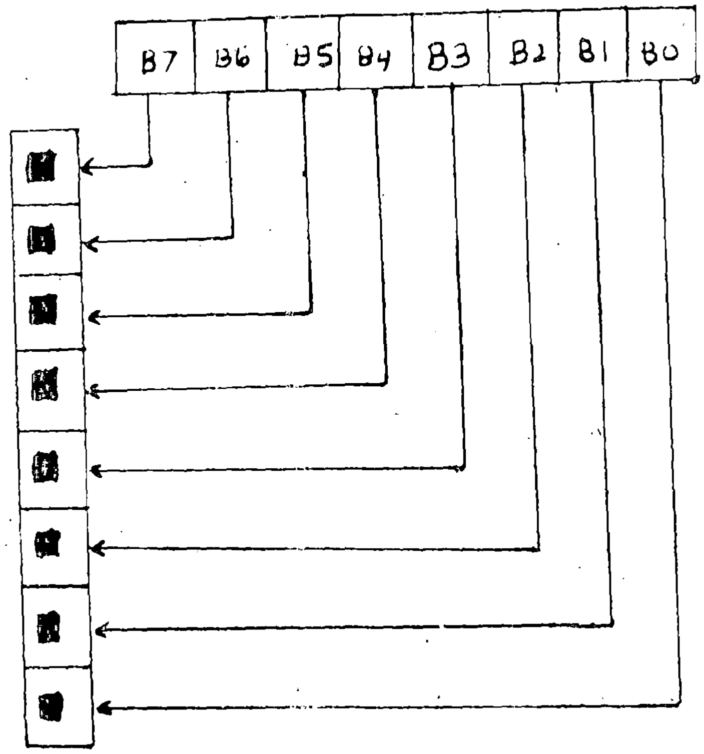


FIG 6.4 (a.)

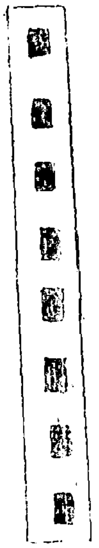
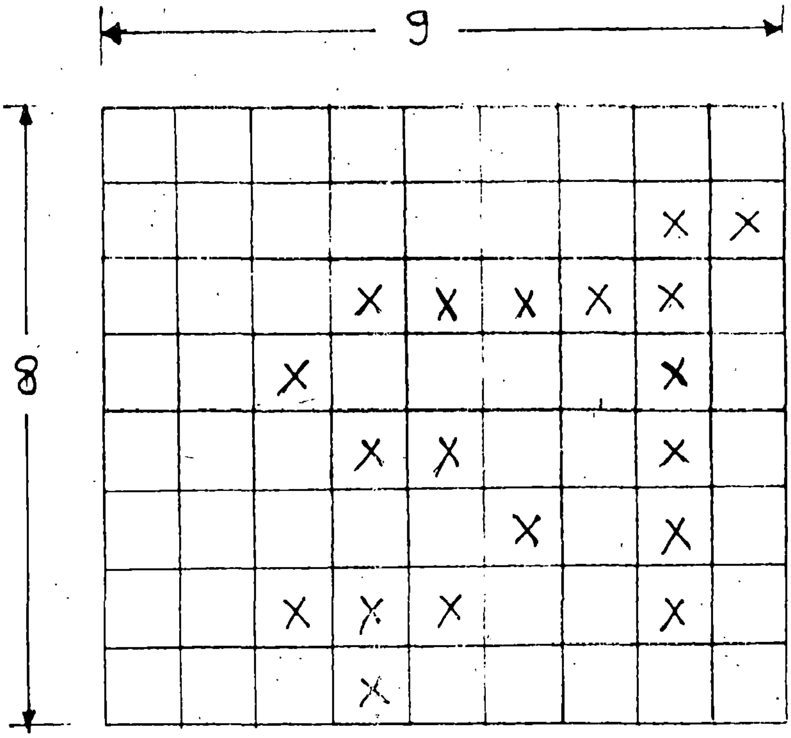
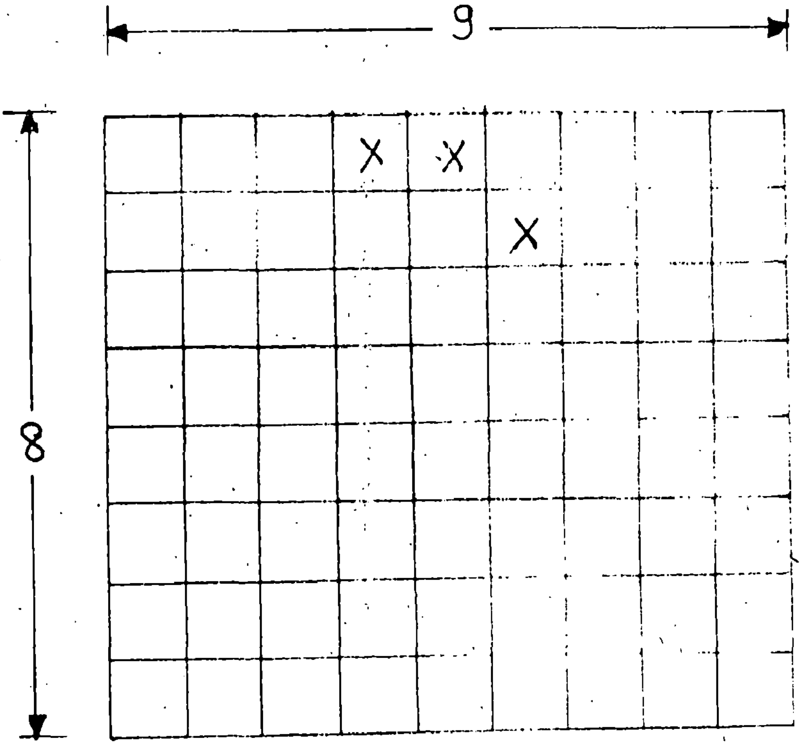


FIG 6.4 (b)



							X	X
			X	X	X	X	X	
		X					X	
			X	X			X	
					X		X	
		X	X	X			X	
			X					

FIG 6·5 (a)



			X	X				
					X			

FIG 6·5 (b)

his is a 9 byte bit map which follows the 6 byte control ommand. The format of the data applied for printing this Devnagari letter would be,

1B 53 30 30 30 39 00 00 48 D4 54 24 04 7E 02

To print these characters a look-up table is made. After taking a character from a required memory location processor decodes it through a look-up table stored in its memory. Data is sent through this table. In the same way matras are developed. Matras are printed in the second phase of printing which is a overprinting on a same line.

5.4 SOFTWARE DESCRIPTION

Software developed for the printer interface first checks whether the printer has to print characters in English or Devnagari and then proceeds accordingly. The general structure of this software development is given in the form of a flow chart in Fig. 6.6.

Initialization of the printer is done in the starting of the operation. LDCD pointer determines the script to be printed. Processor loads each character to its AL register. If the language to be printed is English then processor sends this character to the printer directly and this process repeats. On the other hand if Devnagari is to be printed, then processor finds a corresponding bit map for that character through look-up table and sends the same to the printer as discussed earlier.

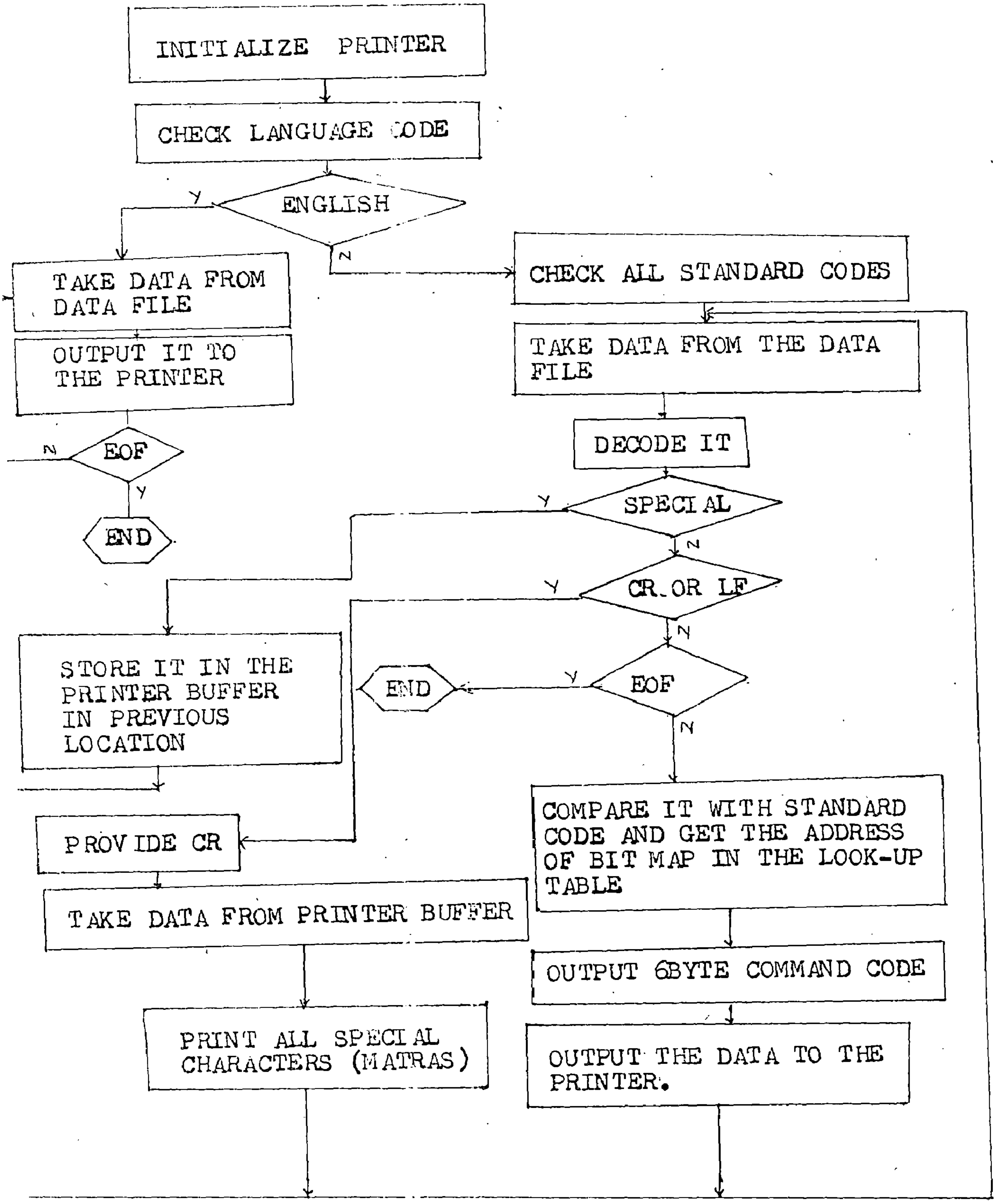

INITIALIZE PRINTER
CHECK LANGUAGE CODE
ENGLISH
Y
N
TAKE DATA FROM DATA FILE
OUTPUT IT TO THE PRINTER
EOF
N
Y
END
CHECK ALL STANDARD CODES
TAKE DATA FROM THE DATA FILE
DECODE IT
SPECIAL
Y
N
CR.OR LF
Y
N
STORE IT IN THE PRINTER BUFFER IN PREVIOUS LOCATION
END
EOF
Y
N
PROVIDE CR
TAKE DATA FROM PRINTER BUFFER
COMPARE IT WITH STANDARD CODE AND GET THE ADDRESS OF BIT MAP IN THE LOOK-UP TABLE
OUTPUT 6BYTE COMMAND CODE
PRINT ALL SPECIAL CHARACTERS (MATRAS)
OUTPUT THE DATA TO THE PRINTER.

FIG 6·6

Along with the main program subroutines used in his software are, GRAPH, PRTCHR, FTPR.GRAPH routine ends the graphics command codes to the printer whenever ecessary. PRJCHR subroutine takes care of sending data hrough look-up table. FTPR routine checks the control ignal and makes the printer ready to accept data.

The character codes for Devanagari alphabets and atras are identical to ISCII codes with some modification or convenience of usuage. These are enumerated already in napter-3 and listed in Appendix.. A-5.

CHAPTER - VII

CONCLUSION

The CRT monitor developed in the present work can display any language provided the script is present in the character generator. Character generators (2782) can be programmed for any required language using bit maps. Once the character generator is selected using a ESCAPE sequence, the language displaying is simple. The languages used for the present work are English, Hindi, Marathi and Kannada. Facility for 8 languages can be developed using this system. The dimension of the cursor is increased by programming 8275 CRT controller Properly. Hence the script of bigger size is displayed.

Editor software is developed on the 8086 microcomputer kit. The multi-lingual CRT terminal developed is interfaced to this microcomputer. Whatever the script is typed on the CRT terminal, its codes are stored in the memory of the microcomputer. These available codes of the scripts are data for the editor. For corrections and modifications in the original data, edit buffer is used. Simple editor functions are developed to edit the original data. The language selected for the use does not affect the editor function.

The parallel printer interface is developed in the present work. Printer gives the hard copy of the data (script) present in the microcomputer memory. The languages script that printer can print are English, Hindi, Marathi. Microcomputer gives a command to printer to print particular language script. Since the graphics mode is used to develop other script, large amount of data is stored in the memory as a look-up table.

SUGGESTIONS FOR FURTHER DEVELOPMENT

In the present work two screen memories are used to display two set of data present in the memories. This solves the problem of superimposing two characters. Only two letters can be superimposed on each other, and hence number of keys used to specify complete set of characters is more. To reduce number of keys used, some other technique has to be used to represent a composite letter.

Eight languages can be selected for use, using this system. By changing the structure of extra hardware (developed), CRT can be made to display the character of many other languages.

The next phase of development is multi-lingual Telex system. In the present work, once the CRT is converted into a CRT terminal of one language (selected by ESC command), it is not possible to go to another language directly. The system has to be reset before selecting the next language. This is a limitation of this system and may be solved using graphics character generation through software.

In the present work the system can create one file which stores data in it and does other required operations. While completing all operations with this file, another file can not be created until the system reset. This limitation can be solved using some efficient file creation techniques.

T record of these file also can be kept using directory. Some standard file structures can be used.

No efforts have been made in text editor system to select terminclogy in Devanagari for different editor commands, this has left as a research activity for a language enthusiast.

Printer interface is used to get the hard copy of typed information. Printing in other language other than English becomes difficult as it has fixed number of dot tips on its print head (8 tips in the available printer). Using this dot tips the size of the letter becomes smaller (vertically), to increase this size seperate lines can be printed for composite letters. First line being basic characters size and next line with a shorter LF pitch. All these can be avoided if a special printer with more dot tips in its print head is made available for Devanagari. The printer having back space defined in its monitor, can avoid overprinting of the line for printing composite letters.

REFERENCES

1. Agarwal, S.K. and H.N.Mahabala, 'Character ROM based Display for Indian Languages'.

2. 'Dot Matrix Serial Impact Printers, Model 7500-User's Manual', C-.Itoh and company limited, Japan.

3. George Hart, G., 'Integrating the Indian Languages', Computer's Today, pp 17-20, October 1985.

4. Ghosh, P.K., 'Crossing Language Barriers', Computer's Today, pp. 30-49, October 1987.

5. 'Intel iAPX 88 Book', Reston publishing company, Inc. USA, 1983.

6. Katausky, I., 'A Low cost CRT Terminal using The 8275', Intel Application note, AP-62, Intel Corporation, USA, Nov. 1979.

7. Liu, Y.C. and G.A.Gibson, 'Micro-computer systems: The 8086/8088 family', Prentice Hall of India Private ltd., New Delhi, 1986.

8. Mahabala, H.N., 'Information Technology in Indian Languages', The Hindu, pp.22, January 6, 1988.

9. Mathur, M.N., et.al., 'Report of the committee for standardization of keyboards layout for Indian script Based Computers', DOE, Electronics Information and planning, Vol. 14, No.1, pp. 3-24, October 1986.

10. VMC-86, Microprocessor Development Kit User's manual', Vinytics Peripherals limited, Delhi.

APPENDIX - A

A1 PIN DIAGRAM OF 8275

A2 INSTRUCTION COMMAND OF THE 8275

A3 SOFTWARE FOR CRT MONITOR DEVELOPMENT

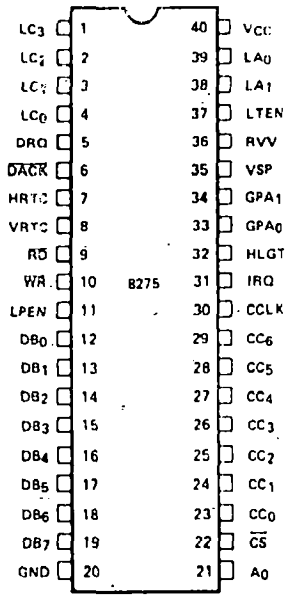
A4 BASIC CRT DIAGRAM

EXTRA HARDWARE DIAGRAM, OVERALL DIAGRAM

A5 CHARACTER SET.

APPENDIX - A1

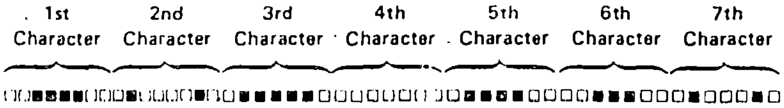
PIN CONFIGURATION



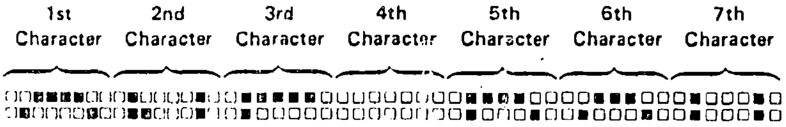
APPENDIX - A2

COMMAND	NO. OF PARAMETER BYTES	NOTES
RESET	4	Display format parameters required
START DISPLAY	0	DMA operation parameters included in command
STOP DISPLAY	0	
READ LIGHT PEN	2	---
LOAD CURSOR	2	Cursor X,Y position parameters required
ENABLE INTERRUPT	0	---
DISABLE INTERRUPT	0	---
PRESET COUNTERS	0	Clears all internal counters

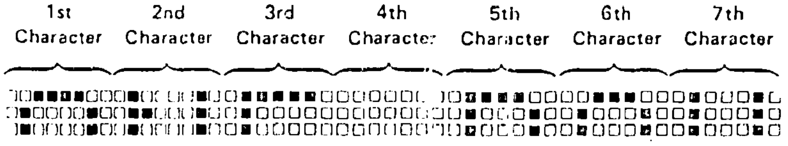
827:'s Instruction Set



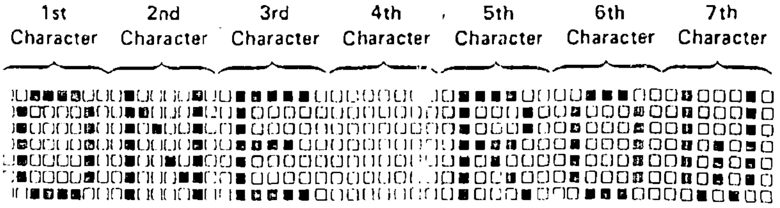
First Line of a Character Row



Second Line of a Character Row



Third Line of a Character Row



Seventh Line of a Character Row

1. Reset Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Reset Command	0	0	0	0	0	0	0	0
Parameters	Write	0	Screen Comp Byte 1	S	H	H	H	H	H	H	H
	Write	0	Screen Comp Byte 2	V	V	R	R	R	R	R	R
	Write	0	Screen Comp Byte 3	U	U	U	U	L	L	L	L
	Write	0	Screen Comp Byte 4	M	F	C	C	Z	Z	Z	Z

Action — After the reset command is written, DMA requests stop, 8275 interrupts are disabled, and the VSP output is used to blank the screen. HRTC and VRTC continue to run. HRTC and VRTC timing are random on power-up.

As parameters are written, the screen composition is defined.

Parameter — S Spaced Rows

S	FUNCTIONS
0	Normal Rows
1	Spaced Rows

Parameter — HHHHHHH Horizontal Characters/Row

H H H H H H H	NO. OF CHARACTERS PER ROW
0 0 0 0 0 0 0	1
0 0 0 0 0 0 1	2
0 0 0 0 0 1 0	3
.	.
.	.
.	.
1 0 0 1 1 1 1	80
1 0 1 0 0 0 0	Undefined
.	.
.	.
.	.
1 1 1 1 1 1 1	Undefined

Parameter — VV Vertical Retrace Row Count

V V	NO. OF ROW COUNTS PER VRTC
0 0	1
0 1	2
1 0	3
1 1	4

Parameter — RRRRRR Vertical Rows/Frame

R R R R R R	NO. OF ROWS/FRAME
0 0 0 0 0 0	1
0 0 0 0 0 1	2
0 0 0 0 1 0	3
.	.
.	.
.	.
1 1 1 1 1 1	64

Parameter — UUUU Underline Placement*

U U U U	LINE NUMBER OF UNDERLINE
0 0 0 0	1
0 0 0 1	2
0 0 1 0	3
.	.
.	.
.	.
1 1 1 1	16

Parameter — LLLL Number of Lines per Character Row

L L L L	NO. OF LINES/ROW
0 0 0 0	1
0 0 0 1	2
0 0 1 0	3
.	.
.	.
.	.
1 1 1 1	16

Parameter — M Line Counter Mode

M	LINE COUNTER MODE
0	Mode 0 (Non-Offset)
1	Mode 1 (Offset by 1 Count)

Parameter — F Field Attribute Mode

F	FIELD ATTRIBUTE MODE
0	Transparent
1	Non-Transparent

Parameter — CC Cursor Format

C C	CURSOR FORMAT
0 0	Blinking reverse video block
0 1	Blinking underline
1 0	Nonblinking reverse video block
1 1	Nonblinking underling

Parameter — ZZZZ Horizontal Retrace Count

Z Z Z Z	NO. OF CHARACTER COUNTS PER HRTC
0 0 0 0	2
0 0 0 1	4
0 0 1 0	6
.	.
.	.
.	.
1 1 1 1	32

Note: uuuu MSB determines blanking of top and bottom lines (1 = blanked, 0 = not blanked).

2. Start Display Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB LSB
Command	Write	1	Start Display	0 0 1 S S S B B
No parameters				

S S S BURST SPACE CODE

S S S	NO. OF CHARACTER CLOCKS BETWEEN DMA REQUESTS
0 0 0	0
0 0 1	7
0 1 0	15
0 1 1	23
1 0 0	31
1 0 1	39
1 1 0	47
1 1 1	55

B B BURST COUNT CODE

B B	NO. OF DMA CYCLES PER BURST
0 0	1
0 1	2
1 0	4
1 1	8

Action — 8275 interrupts are enabled, DMA requests begin, video is enabled, Interrupt Enable and Video Enable status flags are set.

3. Stop Display Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB LSB
Command	Write	1	Stop Display	0 1 0 0 0 0 0 0
No parameters				

Action — Disables video, interrupts remain enabled, HRTC and VRTC continue to run, Video Enable status flag is reset, and the "Start Display" command must be given to re-enable the display.

4. Read Light Pen Command

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB LSB
Command	Write	1	Read Light Pen	0 1 1 0 0 0 0 0
Parameters	Read	0	Char. Number	(Char. Position in Row)
	Read	0	Row Number	(Row Number)

Action — The 8275 is conditioned to supply the contents of the light pen position registers in the next two read cycles of the parameter register. Status flags are not affected.

Note: Software correction of light pen position is required.

5. Load Cursor Position:

	OPERATION	A_0	DESCRIPTION	
Command	Write	1	Load Cursor	1 0 0 0 0 0 0 0
Parameters	Write	0	Char. Number	(Char. Position in Row)
	Write	0	Row Number	(Row Number)

Action — The 8275 is conditioned to place the next two parameter bytes into the cursor position registers. Status flags not affected.

6. Enable Interrupt Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB LSB
Command	Write	1	Enable Interrupt	1 0 1 0 0 0 0 0
No parameters				

Action — The interrupt enable status flag is set and interrupts are enabled.

7. Disable Interrupt Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB LSB
Command	Write	1	Disable Interrupt	1 1 0 0 0 0 0 0
No parameters				

Action — Interrupts are disabled and the interrupt enable status flag is reset.

8. Preset Counters Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB LSB
Command	Write	1	Preset Counters	1 1 1 0 0 0 0 0
No parameters				

Action — The internal timing counters are preset, corresponding to a screen display position at the top left corner. Two character clocks are required for this operation. The counters will remain in this state until any other command is given.

This command is useful for system debug and synchronization of clustered CRT displays on a single CPU.

APPENDIX – A3

```
ASM80 :F1:CRT1.SRC


ISIS-II 8080/8085 MACRO ASSEMBLER, V4.1          MODULE   PAGE    1

  LOC  OBJ           LINE        SOURCE STATEMENT

                        1 ;*************************************************************************
                        2 ;
                        3 ;                     VDT-85 C R T MONITOR  PROGRAM
                        4 ;                              OCT.1987
                        5 ;                         AUTHOR: CHANDRASHEKHAR RAJE
                        6 ;
                        7 ;
                        8 ;*************************************************************************
                        9                 DSEG
  0000                 10         FILE1:  DS      2000
  2000           D     11         TOPAD   EQU     FILE1
  2002           D     12         CURAD   EQU     FILE1+2
  2004           D     13         CURSY   EQU     FILE1+4
  2006           D     14         CURSX   EQU     FILE1+6
  2008           D     15         USCHR2  EQU     FILE1+8
  200A           D     16         LOC80   EQU     FILE1+10
  20C4           D     17         TAG     EQU     FILE1+196
  20C6           D     18         LAST1   EQU     FILE1+198
  20C8           D     19         LAST2   EQU     FILE1+200
  20CA           D     20         LAST    EQU     FILE1+202
  20CC           D     21         LOC81   EQU     FILE1+204
  20D0           D     22         TMPCHR  EQU     FILE1+208
  20D2           D     23         MSRY    EQU     FILE1+210
  20D4           D     24         USCHR1  EQU     FILE1+212
  20F2           D     25         USCHR   EQU     FILE1+242
  20F6           D     26         ESCPU   EQU     FILE1+246
  2100           D     27         TPDIS1  EQU     FILE1+8448
  2550           D     28         ALAST   EQU     FILE1+1360
  3B00           D     29         STKPTR  EQU     FILE1+64
  3100           D     30         TPDIS2  EQU     FILE1+12544
  3650           D     31         BLAST   EQU     FILE1+5712
                       32                 CSEG
                       33         PUBLIC  MAIN
                       34         PUBLIC  REFRSH
                       35 ;--------------------------------------------------------------------------
                       36 ;       START PROGRAM
                       37 ;       ALL VARIABLES ARE INITIALISED
                       38 ;--------------------------------------------------------------------------
  0000 F3              39         START:  DI
  0001 31003B    D     40                 LXI     SP,STKPTR
  0004 210021    D     41                 LXI     H,TPDIS1
  0007 220A20    D     42                 SHLD    LOC80           ;STORE IT IN LOC80
  000A 220020    D     43                 SHLD    TOPAD           ; TOP ADDRESS OF THE SCREEN
  000D 220220    D     44                 SHLD    CURAD           ;CURRENT ADDRESS OF THE CURSOR
  0010 3E01            45                 MVI     A,01H
  0012 320620    D     46                 STA     CURSX           ;X-POSITION OF CURSOR
  0015 32C420    D     47                 STA     TAG             ;SORE IN TAG
  0018 3E00            48                 MVI     A,00H               ;LOAD A
  001A 320420    D     49                 STA     CURSY           ;Y-POSITION OF CURSOR
  001D 215025    D     50                 LXI     H,ALAST         ;LOAD FIRST MEMORY
  0020 C38C00    C     51                 JMP     MAIN
  0023 C3FE02    C     52         INT55:  JMP     CHRCVR
  0026 00              53                 NOP
  0027 C3D003    C     54         INT65:  JMP      REFRSH
  LOC  OBJ         LINE        SOURCE STATEMENT

  002A 00           55                 NOP
  002B 00           56                 NOP
  002C 00           57                 NOP
  002D 00           58                 NOP
                    59 ;---------------------------------------------------------------------------
                    60 ;        THIS ROUTINE IS LOCATED IN THE LOCATION RST 7.5 OF THE 8085 THIS
                    61 ;        ROUTINE TRANSFERS THE DATA FROM MEMORY LOCATION TO THE 8275
                    62 ;---------------------------------------------------------------------------
  002E 00           63                 NOP
  002F F5           64         POPDAT: PUSH     PSW                  ;SAVE A AND FLAGS
  0030 E5           65                 PUSH     H                    ;SAVE H & L
  0031 D5           66                 PUSH     D                    ;SAVE D & E
  0032 210000       67                 LXI      H,0000H              ;ZERO H & L
  0035 39           68                 DAD      SP                   ;PUT STACK POINTER IN H & L
  0036 EB           69                 XCHG                          ;PUT STACK IN D & E
  0037 2A0220   D   70                 LHLD     CURAD                ; GET CURRENT ADDRESS
  003A F9           71                 SPHL                          ; PUT IT IN SP
  003B 3EC0         72                 MVI      A,0C0H               ; SET MASK FOR SIM  'SOD' HIGH
  003D 00           73                 NOP                           ;REPLACE BY SIM INST.
  003E E1           74                 POP      H
  003F E1           75                 POP      H
  0040 E1           76                 POP      H
  0041 E1           77                 POP      H
  0042 E1           78                 POP      H
  0043 E1           79                 POP      H
  0044 E1           80                 POP      H
  0045 E1           81                 POP      H
  0046 E1           82                 POP      H
  0047 E1           83                 POP      H
  0048 E1           84                 POP      H
  0049 E1           85                 POP      H
  004A E1           86                 POP      H
  004B E1           87                 POP      H
  004C E1           88                 POP      H
  004D E1           89                 POP      H
  004E E1           90                 POP      H
  004F E1           91                 POP      H
  0050 E1           92                 POP      H
  0051 E1           93                 POP      H
  0052 E1           94                 POP      H
  0053 E1           95                 POP      H
  0054 E1           96                 POP      H
  0055 E1           97                 POP      H
  0056 E1           98                 POP      H
  0057 E1           99                 POP      H
  0058 E1          100                 POP      H
  0059 E1          101                 POP      H
  005A E1          102                 POP      H
  005B E1          103                 POP      H
  005C E1          104                 POP      H
  005D E1          105                 POP      H
  005E E1          106                 POP      H
  005F E1          107                 POP      H
  0060 E1          108                 POP      H
  0061 E1          109                 POP      H
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0062 0F            110                 RRC                      ;SET UP A FOR LOW SOD
 0063 00            111                 NOP                      ;REPLACE BY SIM INST.
 0064 210000        112                 LXI     H,0000H          ;ZERO H & L
 0067 39            113                 DAD     SP               ; ADD STACK
 0068 EB            114                 XCHG                     ;PUT STACK IN H & L
 0069 F9            115                 SPHL                     ;RESTORE STACK
 006A 2ACA00   D    116                 LHLD    LAST             ;PUT BOTTOM DISPLAY IN DE
 006D EB            117                 XCHG                     ; PUT IT IN D & E
 006E 7A            118                 MOV     A,D              ; PUT HIGHER ORDER IN A
 006F BC            119                 CMP     H                ; COMPARE WITH H
 0070 C27B00   C    120                 JNZ     KPTK             ; IF NOT LEAVE
 0073 7B            121                 MOV     A,E              ; PUT LOWER ORDER IN A
 0074 BD            122                 CMP     L                ; COMPARE WITH L
 0075 C27B00   C    123                 JNZ     KPTK             ; IF NOT LEAVE
 0078 C37302   C    124                 JMP     PDTEXT           ;JUMP TO EXTEND
 007B 220220   D    125         KPTK:   SHLD    CURAD            ;STORE CURRENT ADDRESS
 007E 3E18          126                 MVI     A,18H            ; SET MASK
 0080 00            127                 NOP                      ;REPLACE BY SIM INST.
 0081 D1            128                 POP     D                ; GET D & E
 0082 E1            129                 POP     H                ; GET H & L
 0083 F1            130                 POP     PSW              ; GET A,FLAGS
 0084 FB            131                 EI                       ; ENABLE INTERRUPTS
 0085 C9            132                 RET                      ; GO BACK
 0086 00            133                 NOP
 0087 00            134                 NOP
 0088 00            135                 NOP
 0089 00            136                 NOP
 008A 00            137                 NOP
 008B 00            138                 NOP
                    139 ;------------------------------------------------------------------------
                    140 ;      THIS ROUTINE CLEARS THE SCREEN MEMORY-1 BY PUTTING BLANKS ON IT
                    141 ;------------------------------------------------------------------------
 008C 215036   D    142         MAIN:   LXI     H,BLAST          ;LOAD SECOND MEMORY
 008F 22C820   D    143                 SHLD    LAST2            ;STORE IT IN LAST 2
 0092 215025   D    144                 LXI     H,ALAST          ;LOAD FIRST MEMORY
 0095 22CA20   D    145                 SHLD    LAST             ;STORE IT
 0098 3E00          146                 MVI     A,00H            ;LOAD A
 009A 32D020   D    147                 STA     TMPCHR           ;STORE IN TEMPORARY LOCATION
 009D 32D420   D    148                 STA     USCHR1           ;STORE IT IN FIRST CHARACTER LOCATION
 00A0 320820   D    149                 STA     USCHR2           ;STORE IT IN SECOND CHARACTER LOCATION
 00A3 210021   D    150                 LXI     H,TPDIS1         ;LOAD TOP OF THE SCREEN MEMORY-1
 00A6 015025   D    151                 LXI     B,ALAST          ;LOAD BOTTOM SCREEN MEMORY-1
 00A9 3620          152         LOOP1:  MVI     M,20H            ;     PUT  BLANK IN MEMORY
 00AB 23            153                 INX     H                ;INCREMENT POINTER
 00AC 7C            154                 MOV     A,H              ;GET H
 00AD B8            155                 CMP     B                ;COMPARE WITH B
 00AE C2A900   C    156                 JNZ     LOOP1            ;IF LESS LOOP AGAIN
 00B1 7D            157                 MOV     A,L              ;GET L
 00B2 B9            158                 CMP     C                ;COMPARE WITH C
 00B3 C2A900   C    159                 JNZ     LOOP1            ;IF LESS LOOP AGAIN
                    160 ;------------------------------------------------------------------------
                    161 ;       THIS  ROUTINE  CLEARS  THE SCREEN MEMORY-2 BY PUTTING BLANKS ON IT
                    162 ;------------------------------------------------------------------------
 00B6 210031   D    163                 LXI     H,TPDIS2         ;LOADS THE TOP OF THE SCREEN MEMORY-2
 00B9 015036   D    164                 LXI     B,BLAST          ;LOADS BOTTOM SCREEN MEMORY-2
 LOC  OBJ         LINE        SOURCE STATEMENT

 00BC 3620         165        LOOP2:  MVI     M,20H          ;PUT BLANKS IN MEMORY
 00BE 23           166                INX     H              ;INCREMENT POINTER
 00BF 7C           167                MOV     A,H            ;GET H
 00C0 B8           168                CMP     B              ;COMPARE WITH B
 00C1 C2BC00   C   169                JNZ     LOOP2          ;IF LESS LOOP AGAIN
 00C4 7D           170                MOV     A,L            ;GET L
 00C5 B9           171                CMP     C              ;COMPARE WITH C
 00C6 C2BC00   C   172                JNZ     LOOP2          ;IF LESS LOOP AGAIN
                   173 ;--------------------------------------------------------------------------
                   174 ;                     INITIALIZATION OF 8253
                   175 ;--------------------------------------------------------------------------
 00C9 3E72         176                MVI     A,72H          ;CONTROL WORD FOR 8253(COUNTER-1)
 00CB D363         177                OUT     63H
 00CD 3E32         178                MVI     A,32H          ;LOAD COUNTER 1 WITH LOWER BYTE
 00CF D361         179                OUT     61H
 00D1 3E00         180                MVI     A,00H          ;LOAD COUNTER 1 WITH HIGHER BYTE
 00D3 D361         181                OUT     61H
 00D5 3EB6         182                MVI     A,0B6H       ;CONTROL WORD FOR 8253(COUNTER-2)
 00D7 D363         183                OUT     63H
 00D9 3E14         184                MVI     A,14H          ;LOAD COUNTER 2 WITH LOWER BYTE
 00DB D362         185                OUT     62H
 00DD 3E00         186                MVI     A,00H          ;LOAD COUNTER 2 WITH HIGHER BYTE
 00DF D362         187                OUT     62H            ;OUT
 00E1 C3C302   C   188                JMP     EXTEND         ;JUMP TO EXTEND
                   189 ;--------------------------------------------------------------------------
                   190 ;                    INITIALIZATION OF 8275
                   191 ;--------------------------------------------------------------------------
 00E4 3E00         192        FRSX:   MVI     A,00H          ;RESET 8275
 00E6 D3A1         193                OUT     0A1H
 00E8 3E4F         194                MVI     A,4FH          ;SCREEN PARAMETER BYTE 1
 00EA D3A0         195                OUT     0A0H
 00EC 3ED0         196                MVI     A,0D0H         ;SCREEN PARAMETER BYTE 2
 00EE D3A0         197                OUT     0A0H
 00F0 3E8F         198                MVI     A,8FH          ;SCREEN PARAMETER BYTE 3
 00F2 D3A0         199                OUT     0A0H
 00F4 3EEC         200                MVI     A,0ECH         ;SCREEN PARAMETER BYTE 4
 00F6 D3A0         201                OUT     0A0H
 00F8 CDFD01   C   202                CALL    LDCUR          ;LOAD THE CURSOR
 00FB 3EE0         203                MVI     A,0E0H         ;RESET COUNTERS
 00FD D3A1         204                OUT     0A1H
 00FF 3E23         205                MVI     A,23H          ;START DISPLAY
 0101 D3A1         206                OUT     0A1H
                   207 ;----------------------------------------------------------------------
                   208 ;                     RECEIVE CHARACTER FROM KEYBOARD
                   209 ;--------------------------------------------------------------------
 0103 3E18         210        SETUP:  MVI     A,18H          ;SET MASK
 0105 00           211                NOP                    ;LOAD MASK
 0106 FB           212                EI                     ;ENABLE INTERRUPTS
 0107 DB01         213        AGAIN:  IN      01H            ;INPUT STATUS OF 8251-1
 0109 E602         214                ANI     02H            ;CHECK FOR RECEIVER READY
 010B C21303   C   215                JNZ     URCVR          ;IF NOT CHECK AGAIN
 010E 3AD020   D   216                LDA     TMPCHR         ;LOAD TEMPORARY LOCATION
 0111 FE00         217                CPI     00H            ;CHECK FOR 00
 0113 CA0701   C   218                JZ      AGAIN          ;IF 00 THEN READ THE STATUS AGAIN
 0116 C3A803   C   219                JMP     L2             ;OTHERWISE JUMP TO L2
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

                     220 ;----------------------------------------------------------------------
                     221 ;   THIS ROUTINE PUTS THE CHARACTER IN THE REQUIRED MEMORY LOCATION
                     222 ;   AND INCREMENTS CURSOR X POSITION,AFTER 80D CHARACTER LINE FEED
                     223 ;   IS INSERTED
                     224 ;----------------------------------------------------------------------
  0119 7E            225        CHRPUT1:MOV     A,M             ;GET THE CHARACTER
  011A FEF0          226                CPI     0F0H            ;CHECK FOR CLEAR LINE
  011C 220A20   D    227                SHLD    LOC80           ;SAVE LINE TO CLEAR
  011F CCA001   C    228                CZ      CCLINE          ;CLEAR LINE
  0122 2A0A20   D    229                LHLD    LOC80           ;GET LINE
  0125 CD7F01   C    230                CALL    ADX             ;ADD CURSOR X
  0128 3AF220   D    231                LDA     USCHR           ;GET CHARACTER
  012B 77            232                MOV     M,A             ;PUT IT ON SCREEN
  012C 3A0620   D    233                LDA     CURSX           ;GET CURSOR X
  012F 3C            234                INR     A               ;INCREMENT CURSORX
  0130 FE50          235                CPI     0050H           ;HAS IT GONE TOO FAR
  0132 C23B01   C    236                JNZ     OKI             ;IF NOT JUMP TO OKI
  0135 CD8701   C    237                CALL    LNFD1           ;DO A LINE FEED
  0138 C30C02   C    238                JMP     CGRT            ;DO A CR
  013B 320620   D    239        OKI:    STA     CURSX           ;SAVE CURSOR
  013E CDFD01   C    240                CALL    LDCUR           ;LOAD THE CURSOR
  0141 C3BD03   C    241                JMP     RCRV            ;LEAVE
                     242 ;-------------------------------------------------------------------------
                     243 ;          THIS ROUTINE PROVIDES A ALERNATE DISPLAY OF SCREEN MEMORY.
                     244 ;          ALL CHARACTER GREATER THAN 69H ARE CONSIDER TO BE 'MATRAS'.
                     245 ;-------------------------------------------------------------------------
  0144 CD4402   C    246        ALTDIS: CALL    NALTDS          ;DO A ALTERNATE DISPLAYING
  0147 3E00          247                MVI     A,00H           ;SET ACCUMULATOR
  0149 32D420   D    248                STA     USCHR1          ;INITIALIZE USCHR1
  014C 3A0820   D    249                LDA     USCHR2          ;GET SECOND CHARACTER
  014F FE69          250                CPI     69H             ;CHECK FOR MATRA
  0151 DA3802   C    251                JC      EQUPUT          ;IF YES JUMP
  0154 C3BD03   C    252                JMP     RCRV            ;IF NOT LEAVE
                     253 ;--------------------------------------------------------------------------
                     254 ;       THIS ROUTINE TAKES THE TOP ADDRESS AND THE Y CURSOR LOCATION AND
                     255 ;       CALCULATES THE ADDRESS OF THE LINE THAT THE CURSOR IS ON.THE
                     256 ;       RESULT IS STORED IN HL.
                     257 ;--------------------------------------------------------------------------
  0157 211402   C    258        CALCU:  LXI     H,LINTAB        ;GET LINE TABLE INTO HL
  015A 3A0420   D    259                LDA     CURSY           ;GET CURSOR Y
  015D 07            260                RLC                     ;SET UP FORLOOK TABLE
  015E 0600          261                MVI     B,00H           ;ZERO B
  0160 4F            262                MOV     C,A             ;PUT CURSOR INTO C
  0161 09            263                DAD     B               ;ADD LINE TABLE TO CURSOR Y
  0162 7E            264                MOV     A,M             ;PUT LOW LINE TABLE INTO A
  0163 4F            265                MOV     C,A             ;PUT LOW LINE TABLE INTO C
  0164 23            266                INX     H               ;CHANGE MEMORY POINTER
  0165 7E            267                MOV     A,M             ;PUT HIGHER TABLE INTO A
  0166 47            268                MOV     B,A             ;PUT IT INTO B
  0167 21000F        269                LXI     H,0D0F00H       ;TWOS COMPIMENT OF CREEEN LOCATION
  016A 09            270                DAD     B               ;SUBTRACT OFFSET
  016B EB            271                XCHG                    ;SAVE HL IN DE
  016C 2A0020   D    272                LHLD    TOPAD           ;GET TOP ADDRESS IN HL
  016F 19            273                DAD     D               ;GET DISPLACED ADDRESS
  0170 EB            274                XCHG                    ;SAVE IT IN DE
```

```
 LOC  OBJ         LINE      SOURCE STATEMENT

 0171 21B090            	LXI	H,0D90B0H	;TWOS COMPLIMENT SCREEN LOCATION
 0174 19                	DAD	D		;SEE WHETHER OFF THE SCREEN
 0175 DA7A01    C       	JC	FIX		;IF YES FIX IT
 0178 EB                	XCHG			;GET DISPLACED ADDRESS BACK
 0179 C9                	RET			;GO BACK
 017A 21B0A0            FIX:	LXI	H,0F0A0B0H	;SCREEN BOUNDRY
 017D 19                	DAD	D		;ADJUST SCREEN
 017E C9                	RET			;GO BACK
                        ;-----------------------------------------------------------------------
                        ;	THIS ROUTINE ADDS THE X POSITION TO THE ADDRESS THAT IS IN
                        ;	THE HL REG AND STORES THE RESULT HL REG
                        ;-----------------------------------------------------------------------
 017F 3A0620    D       ADX:	LDA	CURSX		;GET CURSOR
 0182 0600              	MVI	B,00H		;ZERO B
 0184 4F                	MOV	C,A		;PUT CURSOR X IN C
 0185 09                	DAD	B		;ADD CURSOR X IN HL
 0186 C9                	RET			;GO BACK
                        ;-----------------------------------------------------------------------
                        ;		THIS ROUTINE PROVIDES A LINE FEED
                        ;-----------------------------------------------------------------------
 0187 3A0420    D       LNFD1:	LDA	CURSY		;GET CURSOR Y POSITION
 018A FE10              	CPI	10H		;SEE IF BOTTOM OF SCREEN
 018C CADE01    C       	JZ	ONBOT		;IF YES JUMP
 018F 3C                	INR	A		;INCREMENT
 0190 320420    D       	STA	CURSY		;SAVE NEW CURSOR
 0193 CD5701    C       	CALL	CALCU		;CALCULATE ADDRESS
 0196 220A20    D       	SHLD	LOC80		;SAVE TO CLEAR LINE
 0199 CDA001    C       	CALL	CCLINE		;CLEAR THE LINE
 019C CDFD01    C       	CALL	LDCUR		;LOAD THE CURSOR
 019F C9                	RET			;GO BACK
                        ;-----------------------------------------------------------------------
                        ;	THIS LINE CLEARS THE LINE WHOSE FIRST ADDRESS IS IN LOC80.
                        ;	40 PUSH INSTRUCTIONS ARE USED TO CLEAR THE LINE RAPIDLY
                        ;-----------------------------------------------------------------------
 01A0 F3                CCLINE:	DI			;DISABLE INTERRUPTS
 01A1 2A0A20    D       	LHLD	LOC80		;GET LOC80
 01A4 115000            	LXI	D,0050H		;GET OFFSET
 01A7 19                	DAD	D		;ADD OFFSET
 01A8 EB                	XCHG			;PUT START IN DE
 01A9 210000            	LXI	H,0000H		;ZERO HL
 01AC 39                	DAD	SP		;GET STACK
 01AD EB                	XCHG			;PUT STACK IN DE
 01AE F9                	SPHL			;PUT START IN SP
 01AF 212020            	LXI	H,2020H		;PUT SPACES IN HL
 01B2 E5                	PUSH	H
 01B3 E5                	PUSH	H
 01B4 E5                	PUSH	H
 01B5 E5                	PUSH	H
 01B6 E5                	PUSH	H
 01B7 E5                	PUSH	H
 01B8 E5                	PUSH	H
 01B9 E5                	PUSH	H
 01BA E5                	PUSH	H
 01BB E5                	PUSH	H
 01BC E5                	PUSH	H
 LOC  OBJ         LINE        SOURCE STATEMENT

 01BD E5           330                PUSH    H
 01BE E5           331                PUSH    H
 01BF E5           332                PUSH    H
 01C0 E5           333                PUSH    H
 01C1 E5           334                PUSH    H
 01C2 E5           335                PUSH    H
 01C3 E5           336                PUSH    H
 01C4 E5           337                PUSH    H
 01C5 E5           338                PUSH    H
 01C6 E5           339                PUSH    H
 01C7 E5           340                PUSH    H
 01C8 E5           341                PUSH    H
 01C9 E5           342                PUSH    H
 01CA E5           343                PUSH    H
 01CB E5           344                PUSH    H
 01CC E5           345                PUSH    H
 01CD E5           346                PUSH    H
 01CE E5           347                PUSH    H
 01CF E5           348                PUSH    H
 01D0 E5           349                PUSH    H
 01D1 E5           350                PUSH    H
 01D2 E5           351                PUSH    H
 01D3 E5           352                PUSH    H
 01D4 E5           353                PUSH    H
 01D5 E5           354                PUSH    H
 01D6 E5           355                PUSH    H
 01D7 E5           356                PUSH    H
 01D8 E5           357                PUSH    H
 01D9 E5           358                PUSH    H
 01DA EB           359                XCHG                    ;PUT STACK IN HL
 01DB F9           360                SPHL                    ;PUT BACK IN SP
 01DC FB           361                EI                      ;ENABLE INTERRUPTS
 01DD C9           362                RET                     ;GO BACK
                   363 ;---------------------------------------------------------------------------
                   364 ;          THIS ROUTINE PROVIDES A LINE FEED IF THE CURSOR IS
                   365 ;          ON THE BOTTOM OF THE SCREEN MEMORY .
                   366 ;---------------------------------------------------------------------------
 01DE 2A0020    D  367        ONBOT:  LHLD    TOPAD           ;GET TOP ADDRESS
 01E1 220A20    D  368                SHLD    LOC80           ;PUT IT IN LOC80
 01E4 115000       369                LXI     D,0050H         ;LINE LENGTH
 01E7 19           370                DAD     D               ;ADD HL TO DE
 01E8 21C620    D  371                LXI     H,LAST1         ;GET BOTTOM LINE
 01EB 7C           372                MOV     A,H             ;GET
 01EC B8           373                CMP     B               ;SAME AS B
 01ED C2F301    C  374                JNZ     ARND            ;IF NOT SAME GO BACK
 01F0 C35B02    C  375                JMP     NTBT            ;IF SAME JUMP
 01F3 220020    D  376        ARND:   SHLD    TOPAD           ;STORE TOPAD
 01F6 CDA001    C  377                CALL    CCLINE          ;CLEAR LINE
 01F9 CDFD01    C  378                CALL    LDCUR           ;LOAD CURSOR
 01FC C9           379                RET                     ;GO BACK
                   380 ;---------------------------------------------------------------------------
                   381 ;         THIS ROUTINE INITIALIZES THE CURSOR POSITION BY SPESIFYING
                   382 ;         X AND Y POSITIONS .
                   383 ;---------------------------------------------------------------------------
 01FD 3E80         384        LDCUR:  MVI     A,80H           ;LOAD A
```

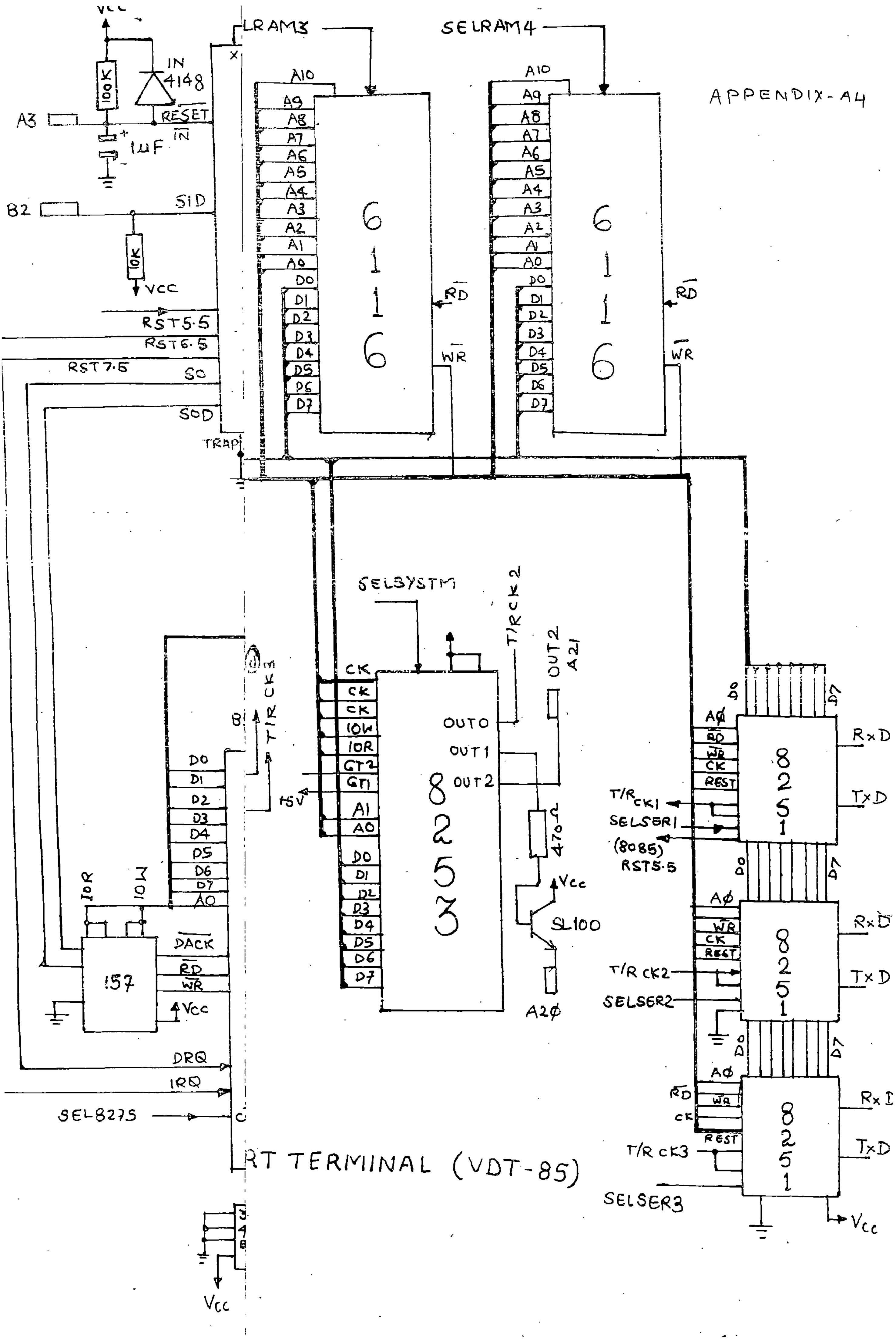
```
LOC  OBJ          LINE        SOURCE STATEMENT

01FF D3A1          385                  OUT     0A1H              ;OUT
0201 3A0620    D   386                  LDA     CURSX             ;LOAD CURSOR X
0204 D3A0          387                  OUT     0A0H              ;OUT
0206 3A0420    D   388                  LDA     CURSY             ;LOAD CURSOR Y
0209 D3A0          389                  OUT     0A0H
020B C9            390                  RET
                   391 ;-----------------------------------------------------------------------
                   392 ;        CARRIAGE RETURN IS PROVIDED BY THIS ROUTINE BY SETTING CURSX 00
                   393 ;-----------------------------------------------------------------------
020C 3E00          394          CGRT:   MVI     A,00H             ;LOAD A WITH 00
020E 320620    D   395                  STA     CURSX             ;STORE CURSOR X
0211 C3BD03    C   396                  JMP     RCRV              ;LEAVE
                   397 ;-----------------------------------------------------------------------
                   398 ;           THIS PROVIDES A SET OF LINE NUMBER USED ,IT GIVES
                   399 ;            STARTING ADDRESS OF EACH LINE.
                   400 ;-----------------------------------------------------------------------
0000               401 LINTAB:          LNMBR   SET     0
0214 0021      D   402                  DW      TPDIS1+(0050H*LNMBR)
                   403 ;-----------------------------------------------------------------------
                   404 ;       THIS ROUTINE FINDS THE ADDRESS OF SECOND SCREEN MEMORY BY
                   405 ;       ADDING OFFSET TO FIRST SCREEN MEMORY.
                   406 ;-----------------------------------------------------------------------
0216 2A0A20    D   407          NEW:    LHLD    LOC80             ;GET LOC80
0219 22CC20    D   408                  SHLD    LOC81             ;STORE IN LOC81
021C 110010        409                  LXI     D,1000H           ;LOAD D
021F 19            410                  DAD     D                 ;ADD DE
0220 22CC00    D   411                  SHLD    LOC81             ;GET SECOND SCREEN MEMORY
0223 CDA001    C   412                  CALL    CCLINE            ;CLEAR LINE
0226 2ACC20    D   413                  LHLD    LOC81             ;GET LOC81
0229 220A20    D   414                  SHLD    LOC80             ;STORE  IN LOC80
022C CDFD01    C   415                  CALL    LDCUR             ;LOAD CURSOR
022F C9            416                  RET
                   417 ;-----------------------------------------------------------------------
                   418 ;         THIS ROUTINE PUTS CHARACTER IN THE SCREEN MEMORY AND CHECKS
                   419 ;         THE CHARACTER FOR 'MATRAS'.
                   420 ;-----------------------------------------------------------------------
0230 3AD420    D   421          NCALCU: LDA     USCHR1            ;GET FIRST CHARACTER
0233 FE00          422                  CPI     00H               ;CHECK IF 00H
0235 C24401    C   423                  JNZ     ALTDIS            ;IF NOT JUMP
0238 3AD020    D   424          EQUPUT: LDA     TMPCHR            ;GET TEMPORARY CHARACTER
023B 32D420    D   425                  STA     USCHR1            ;STORE FIRST CHARACTER
023E CD5701    C   426                  CALL    CALCU             ;CALCULATE ADDRESS
0241 C31901    C   427                  JMP     CHRPUT1           ;JUMP
                   428 ;-----------------------------------------------------------------------
                   429 ;       THIS ROTINE SAVES THE SECOND CHARACTER AND TAKES ACTION .
                   430 ;-----------------------------------------------------------------------
0244 320820    D   431          NALTDS: STA     USCHR2            ;STORE SECOND CHARACTER
0247 2A0A20    D   432                  LHLD    LOC80             ;GET LOC80
024A 010010        433                  LXI     B,1000H           ;LOAD B
024D 09            434                  DAD     B                 ;ADD B TO HL
024E 3A0620    D   435                  LDA     CURSX             ;GET CURSOR X
0251 3D            436                  DCR     A                 ;DECREMENT
0252 0600          437                  MVI     B,00H             ;LOAD B
0254 4F            438                  MOV     C,A               ;PUT A IN C
0255 09            439                  DAD     B                 ;ADD B TO HL
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 0256 3A0820   D   440                   LDA     USCHR2          ;GET SECOND CHARACTER
 0259 77           441                   MOV     M,A             ;PUT IT IN MEMORY
 025A C9           442                   RET
                   443 ;-------------------------------------------------------------------------
                   444 ;       THIS ROUTINE UPDATES THE POINTERS ,INITIALIZES TAG
                   445 ;-------------------------------------------------------------------------
 025B 3E01         446          NTBT:    MVI     A,01H           ;LOAD A
 025D 32C400   D   447                   STA     TAG             ;LOAD TAG
 0260 215025   D   448                   LXI     H,ALAST         ;LOAD HL
 0263 22CA00   D   449                   SHLD    LAST            ;STORE IN LAST
 0266 210021   D   450                   LXI     H,TPDIS1        ;GET TOP OF DISPLAY
 0269 220020   D   451                   SHLD    TOPAD           ;STORE IN TOP ADDRESS
 026C CDA001   C   452                   CALL    CCLINE          ;CLEAR THE LINE
 026F CD1602   C   453                   CALL    NEW             ;UPDATE POINTERS
 0272 C9           454                   RET                     ;GO BACK
                   455 ;---------------------------------------------------------------------------
                   456 ;             THIS ROUTINE IS THE EXTENTION OF POPDAT ROUTINE ,WHICH
                   457 ;             SELECTS EITHER OF THE SCREEN MEMORY DEPENDIG UPON TAG .
                   458 ;---------------------------------------------------------------------------
 0273 3AC420   D   459          PDTEXT: LDA     TAG             ;GET TAG
 0276 FE01         460                   CPI     01H             ;CHECK FOR 01
 0278 CA8A02   C   461                   JZ      PNXT            ;IF ZERO ,JUMP
 027B 210021   D   462                   LXI     H,TPDIS1        ;GET TOP DISPLAY
 027E 220220   D   463                   SHLD    CURAD           ;STORE IN CURRENT ADDRESS
 0281 215025   D   464                   LXI     H,ALAST         ;LOAD LAST1
 0284 22CA20   D   465                   SHLD    LAST            ;STORE IN LAST
 0287 C39602   C   466                   JMP     BYPASS          ;JUMP
 028A 210031   D   467          PNXT:    LXI     H,TPDIS2        ;LOAD SECOND TOP DISPLAY
 028D 220220   D   468                   SHLD    CURAD           ;STORE IT IN CURRENT ADDRESS
 0290 215036   D   469                   LXI     H,BLAST         ;LOAD LAST2
 0293 22CA20   D   470                   SHLD    LAST            ;STORE IT IN LAST
 0296 3E18         471          BYPASS: MVI     A,18H           ;LOAD A
 0298 00           472                   NOP                     ;SET MASK
 0299 D1           473                   POP     D               ;RESTORE D
 029A E1           474                   POP     H               ;RESTORE H
 029B F1           475                   POP     PSW             ;RESTORE PSW
 029C FB           476                   EI                      ;ENABLE ALL INTERRUPTS
 029D C9           477                   RET
 029E CD8701   C   478          LNFD:    CALL    LNFD1           ;DO LINE FEED
 02A1 C3BD03   C   479                   JMP     RCRV            ;LEAVE
                   480 ;----------------------------------------------------------------------------
                   481 ;        THIS ROUTINE  DECREMENTS THE CURSOR POSITION ALONG X DIRECTION.
                   482 ;----------------------------------------------------------------------------
 02A4 3A0620   D   483          LEFT:    LDA     CURSX           ;GET CURSOR X
 02A7 FE00         484                   CPI     00H             ;CHECK FOR 00H
 02A9 C2B402   C   485                   JNZ     NOVER           ;IF NOT JUMP
 02AC 3A0420   D   486                   LDA     CURSY           ;GET CURSOR Y
 02AF FE00         487                   CPI     00H             ;CHECK FOR 00
 02B1 CABD03   C   488                   JZ      RCRV            ;IF ZERO JUMP
 02B4 3D           489          NOVER:   DCR     A               ;DECREMENT A
 02B5 320420   D   490                   STA     CURSY           ;STORE CURSOR Y
 02B8 3E4F         491                   MVI     A,4FH           ;LOAD A WITH 4FH
 02BA 320420   D   492                   STA     CURSY           ;STORE IN CURSOR Y
 02BD CDFD01   C   493                   CALL    LDCUR           ;LOAD CURSOR
 02C0 C3BD03   C   494                   JMP     RCRV            ;JUMP
```

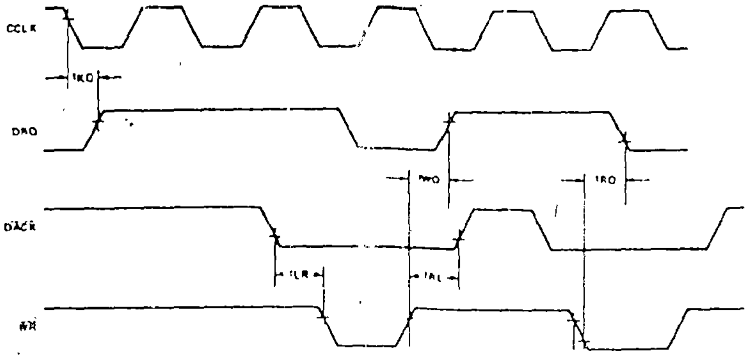
```
  LOC  OBJ         LINE        SOURCE STATEMENT

                    495 ;-----------------------------------------------------------------------
                    496 ;          THIS ROUTINE IS THE EXTENTION OF INITIALIZATION ,IT INITIALIZES
                    497 ;          8251-1,8251-2,8253 AND 8255 IT ALSO SELECTS THE REQUIRED
                    498 ;          CHARACTER GENERATOR FOR THE USE OF LANGUAGE.
                    499 ;-----------------------------------------------------------------------
 02C3 3E36          500        EXTEND: MVI    A,36H           ;LOAD A
 02C5 D363          501                OUT    63H             ;OUT CONTROL WORD FOR 8253 COUNTER 0
 02C7 3E14          502                MVI    A,14H           ;LOAD A WITH MSB
 02C9 D360          503                OUT    60H             ;OUT MSB
 02CB 3E00          504                MVI    A,00H           ;LOAD A
 02CD D360          505                OUT    60H             ;OUT LSB
 02CF 3E4E          506                MVI    A,4EH           ;MODE WORD FOR 8251
 02D1 D301          507                OUT    01H             ;OUT IT
 02D3 3E26          508                MVI    A,26H           ;COMMAND WORD
 02D5 D301          509                OUT    01H             ;OUT IT
 02D7 3E00          510                MVI    A,00H           ;LOAD A
 02D9 D341          511                OUT    41H             ;OUT
 02DB 3E00          512                MVI    A,00H           ;LOAD   A
 02DD D341          513                OUT    41H
 02DF 3E00          514                MVI    A,00H
 02E1 D341          515                OUT    41H
 02E3 3E40          516                MVI    A,40H           ;INTERNAL RESET OF 8251-1
 02E5 D341          517                OUT    41H
 02E7 00            518                NOP
 02E8 3E4F          519                MVI    A,4FH           ;MODE WORD FOR 8251-2
 02EA D341          520                OUT    41H             ;OUT IT
 02EC 3E27          521                MVI    A,27H           ;COMMAND WORD OF 8251-2
 02EE 3E80          522                MVI    A,80H           ;CONTROL WORD FOR 8255
 02F0 D383          523                OUT    83H             ;OUT IT
 02F2 3E00          524                MVI    A,00H           ;SELECT CHARACTER GENERATOR1
 02F4 D382          525                OUT    82H             ;OUT
 02F6 3E00          526                MVI    A,00H           ;LOAD A
 02F8 32F600    D   527                STA    ESCPU           ;STORE ESCAPE SEQUENCE
 02FB C3E400    C   528                JMP    FRSX            ;JUMP
                    529 ;-----------------------------------------------------------------------
                    530 ;          THIS IS THE INTERRUPT RST5.5 ,IT RECEIVES CHARACTER FROM
                    531 ;          THE MICROCOMPUTER AFTER EACH EXECUTION.
                    532 ;-----------------------------------------------------------------------
 02FE F3            533        CHRCVR: DI                     ;DISABLE ALL INTERRUPTS
 02FF F5            534                PUSH   PSW             ;SAVE FLAGS
 0300 E5            535                PUSH   H               ;SAVE H
 0301 D5            536                PUSH   D               ;SAVE D
 0302 DB41          537        AGN:    IN     41H             ;READ STATUS
 0304 E602          538                ANI    02H             ;CHECK FOR RECEIVER READY
 0306 CA0203    C   539                JZ     AGN             ;IF NOT CHECK AGAIN
 0309 DB40          540                IN     40H             ;INPUT THE DATA
 030B 32D020    D   541                STA    TMPCHR          ;STORE IT IN TEMPORARY LOCATION
 030E D1            542                POP    D
 030F E1            543                POP    H
 0310 F1            544                POP    PSW
 0311 FB            545                EI
 0312 C9            546                RET
                    547 ;-------      ----------------------------------------------------------
                    548 ;          THIS ROUTINE TAKES CHARACTER FROM TEMPORARY LOCATION AND
                    549 ;          DECODES IT,IF IT IS A LANGUAGE SELECTION CODE SELECTS THE
 LOC  OBJ         LINE        SOURCE STATEMENT

                   550 ;            CHARACTER GENERATOR.OTHER CHARACTERS ARE TANSMITTED TO
                   551 ;            MICROCOMPUTER .
                   552 ;-----------------------------------------------------------------------------
 0313 DB01         553        URCVR: IN     01H              ;READ STATUS OF 8251
 0315 E602         554               ANI    02H              ;CHECK FOR RECEIVER READY
 0317 CA1303   C   555               JZ     URCVR            ;IF ZERO JUMP
 031A DB00         556               IN     00H              ;INPUT THE DATA
 031C 32F220   D   557               STA    USCHR            ;STORE IT IN SECOND MEMORY
 031F FE1B         558               CPI    1BH              ;CHECK ESCAPE SEQUENCE
 0321 C22A03   C   559               JNZ    PRCK             ;IF NOT JUMP
 0324 32F620   D   560               STA    ESCPU            ;STORE IT IN ESCAPE
 0327 C3BD03   C   561               JMP    RCRV             ;JUMP
 032A 3AF620   D   562        PRCK:  LDA    ESCPU            ;LOAD A WITH ESCAPE
 032D FE1B         563               CPI    1BH              ;CHECK FOR ESCAPE
 032F C29A03   C   564               JNZ    TMTR             ;IF NOT JUMP
 0332 3E00         565               MVI    A,00H
 0334 32F620   D   566               STA    ESCPU            ;STORE 00 IN ESCPU
 0337 3AF220   D   567               LDA    USCHR            ;GET SECOND CHARACTER
 033A FE45         568               CPI    45H              ;CHECK FOR 'ENGLISH`
 033C C24603   C   569               JNZ    ML               ;IF NOT JUMP
 033F 3E00         570               MVI    A,00H            ;LOAD A
 0341 D382         571               OUT    82H              ;OUT
 0343 C3BD03   C   572               JMP    RCRV             ;JUMP
 0346 FE4D         573        ML:    CPI    4DH              ;CHECK  FOR `MARATHI'
 0348 C25203   C   574               JNZ    K                ;IF NOT JUMP
 034B 3E10         575               MVI    A,10H            ;LOAD A
 034D D382         576               OUT    82H              ;OUT
 034F C3BD03   C   577               JMP    RCRV             ;LEAVE
 0352 FE4B         578        K:     CPI    4BH              ;CHECK FOR 'KANNADA`
 0354 C25E03   C   579               JNZ    G                ;IF NOT JUMP
 0357 3E20         580               MVI    A,20H            ;LOAD A
 0359 D382         581               OUT    82H              ;OUT
 035B C3BD03   C   582               JMP    RCRV             ;LEAVE
 035E FE47         583        G:     CPI    47H              ;CHECK  FOR'GUJARATHI`
 0360 C26A03   C   584               JNZ    BL               ;IF NOT JUMP
 0363 3E30         585               MVI    A,30H            ;LOAD A
 0365 D382         586               OUT    82H              ;OUT
 0367 C3BD03   C   587               JMP    RCRV             ;LEAVE
 036A FE42         588        BL:    CPI    42H              ;CHECK    'BENGALI'
 036C C27603   C   589               JNZ    T                ;IF NOT JUMP
 036F 3E40         590               MVI    A,40H            ;LOAD A
 0371 D382         591               OUT    82H              ;OUT
 0373 C3BD03   C   592               JMP    RCRV             ;JUMP
 0376 FE54         593        T:     CPI    54H              ;CHECK FOR 'TAMIL`
 0378 C28203   C   594               JNZ    LL               ;IF NOT JUMP
 037B 3E50         595               MVI    A,50H            ;LOAD A
 037D D382         596               OUT    82H              ;OUT
 037F C3BD03   C   597               JMP    RCRV             ;JUMP
 0382 FE4C         598        LL:    CPI    4CH              ;CHECK FOR
 0384 C28E03   C   599               JNZ    Y                ;IF NOT JUMP
 0387 3E60         600               MVI    A,60H            ;LOAD A
 0389 D382         601               OUT    82H              ;OUT
 038B C3BD03   C   602               JMP    RCRV             ;LEAVE
 038E FE59         603        Y:     CPI    59H              ;CHECK FOR
 0390 C2BD03   C   604               JNZ    RCRV             ;IF NOT LEAVE
 LOC  OBJ         LINE        SOURCE STATEMENT

 0393 3E70         605               MVI     A,70H           ;LOAD A
 0395 D382         606               OUT     82H             ;OUT
 0397 C3BD03  C    607               JMP     RCRV            ;JUMP
 039A 3AF220  D    608        TMTR:  LDA     USCHR           ;GET SECOND CHARACTER
 039D F5           609               PUSH    PSW             ;SAVE PSW
 039E DB41         610      URTMTR:  IN      41H             ;READ STATUS
 03A0 E601         611               ANI     01H             ;CHECK FOR TRANSMITTER
 03A2 CA9E03  C    612               JZ      URTMTR          ;IF ZERO JUMP
 03A5 F1           613               POP     PSW             ;RESTORE PSW
 03A6 D340         614               OUT     40H             ;OUT
 03A8 3E00         615        L2:    MVI     A,00H           ;LOAD A
 03AA 32F220  D    616               STA     USCHR           ;STORE A
 03AD 3AD020  D    617        REFT:  LDA     TMPCHR          ;GET TEMPORARY CHARACTER
 03B0 FE00         618               CPI     00H             ;CHECK FOR ZERO
 03B2 CAAD03  C    619               JZ      REFT            ;IF ZERO JUMP
 03B5 3AD020  D    620               LDA     TMPCHR          ;GET TEMPORARY CHARACTER
 03B8 FE69         621               CPI     69H             ;CHECK FOR `MATRAS'
 03BA D24401  C    622               JNC     ALTDIS          ;IF NOT ,JUMP
 03BD DB41         623        RCRV:  IN      41H             ;READ STATUS
 03BF E601         624               ANI     01H             ;CHECK FOR TRANSMITTER READY
 03C1 CABD03  C    625               JZ      RCRV            ;IF ZERO ,JUMP
 03C4 3E01         626               MVI     A,01H           ;LOAD A
 03C6 D340         627               OUT     40H             ;OUT
 03C8 3E00         628               MVI     A,00H           ;LOAD A
 03CA 32D020  D    629               STA     TMPCHR          ;STORE IN TEMPORARY LOCATION
 03CD C30301  C    630               JMP     SETUP           ;JUMP TO SETUP
                   631 ;-----------------------------------------------------------------------
                   632 ;     THIS IS A RST6.5 INTERRUPT ROUTINE .THIS READS THE STATUS OF 8275.
                   633 ;     IT UPDATES ALL REQUIRED POINTERS TO KEEP THE DISPLAY REFRESHED.
                   634 ;-----------------------------------------------------------------------
 03D0 F3           635      REFRSH:  DI                      ;DISABLE INTERRUPTS
 03D1 F5           636               PUSH    PSW             ;SAVE FLAGS
 03D2 E5           637               PUSH    H               ;SAVE   H
 03D3 D5           638               PUSH    D               ;SAVE   D
 03D4 DBA1         639               IN      0A1H            ;READ STATUS
 03D6 3AC400  D    640               LDA     TAG             ;GET TAG
 03D9 FE01         641               CPI     01H             ;CHECK TAG
 03DB CAF103  C    642               JZ      ZCO             ;IF ZERO JUMP
 03DE 2A0020  D    643               LHLD    TOPAD           ;LOAD TOP ADDRESS
 03E1 220220  D    644               SHLD    CURAD           ;STORE IT IN CURRENT ADDRESS
 03E4 2AC620  D    645               LHLD    LAST1           ;LOAD FIRST MEMORY
 03E7 22CA20  D    646               SHLD    LAST            ;SAVE IT
 03EA 3C           647               INR     A               ;INCREMENT
 03EB 32C420  D    648               STA     TAG             ;STORE TAG
 03EE C30E04  C    649               JMP     ZDE             ;JUMP
 03F1 2A0020  D    650        ZCO:   LHLD    TOPAD           ;LOAD TOP ADDRESS
 03F4 22D220  D    651               SHLD    MSRY            ;STORE IT
 03F7 110010       652               LXI     D,1000H         ;LOAD D
 03FA 19           653               DAD     D               ;ADD TO GET SECOND MEMORY
 03FB 220220  D    654               SHLD    CURAD           ;STORE IN CURRENT ADDRESS
 03FE 2AD220  D    655               LHLD    MSRY            ;LOAD MSRY
 0401 220020  D    656               SHLD    TOPAD           ;STORE IN TOP ADDRESS
 0404 2AC820  D    657               LHLD    LAST2           ;LOAD HL
 0407 22CA20  D    658               SHLD    LAST            ;STORE IT IN LAST
 040A 3D           659               DCR     A               ;DECREMENT
```

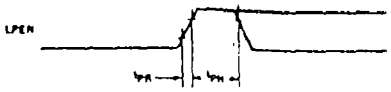
```
  LOC  OBJ         LINE        SOURCE STATEMENT

  040B 32C400   D   660                 STA     TAG              ;STOTRE IT IN TAG
  040E 3E18         661          ZDE:   MVI     A,18H            ;LOAD A
  0410 00           662                 NOP                      ;SET MASK
  0411 D1           663                 POP     D                ;RESTORE D
  0412 E1           664                 POP     H                ;RESTORE H
  0413 F1           665                 POP     PSW              ;GET FLAGS
  0414 FB           666                 EI                       ;ENABLE ALL INTERRUPTS
  0415 C9           667                 RET
  008C          C   668                 END     MAIN

PUBLIC SYMBOLS
MAIN   C 008C    REFRSH C 03D0

EXTERNAL SYMBOLS


USER SYMBOLS
ADX    C 017F    AGAIN  C 0107    AGN    C 0302    ALAST  D 2550    ALTDIS C 0144    ARND   C 01F3    BL     C 036A
BLAST  D 3650    BYPASS C 0296    CALCU  C 0157    CCLINE C 01A0    CGRT   C 020C    CHRCVR C 02FE    CHRPUT C 0119
CURAD  D 2002    CURSX  D 2006    CURSY  D 2004    EQUPUT C 0238    ESCPU  D 20F6    EXTEND C 02C3    FILE1  D 0000
FIX    C 017A    FRSX   C 00E4    G      C 035E    INT55  C 0023    INT65  C 0027    K      C 0352    KPTK   C 007B
L2     C 03A8    LAST   D 20CA    LAST1  D 20C6    LAST2  D 20C8    LDCUR  C 01FD    LEFT   C 02A4    LINTAB C 0214
LL     C 0382    LNFD   C 029E    LNFD1  C 0187    LNMBR  A 0000    LOC80  D 200A    LOC81  D 20CC    LOOP1  C 00A9
LOOP2  C 00BC    MAIN   C 008C    ML     C 0346    MSRY   D 20D2    NALTDS C 0244    NCALCU C 0230    NEW    C 0216
NOVER  C 02B4    NTBT   C 025B    OKI    C 013B    ONBOT  C 01DE    PDTEXT C 0273    PNXT   C 028A    POPDAT C 002F
PRCK   C 032A    RCRV   C 03BD    REFRSH C 03D0    REFT   C 03AD    SETUP  C 0103    START  C 0000    STKPTR D 3B00
T      C 0376    TAG    D 20C4    TMPCHR D 20D0    TMTR   C 039A    TOPAD  D 2000    TPDIS1 D 2100    TPDIS2 D 3100
URCVR  C 0313    URTMTR C 039E    USCHR  D 20F2    USCHR1 D 20D4    USCHR2 D 2008    Y      C 038E    ZCO    C 03F1
ZDE    C 040E

ASSEMBLY COMPLETE,   NO ERRORS
```

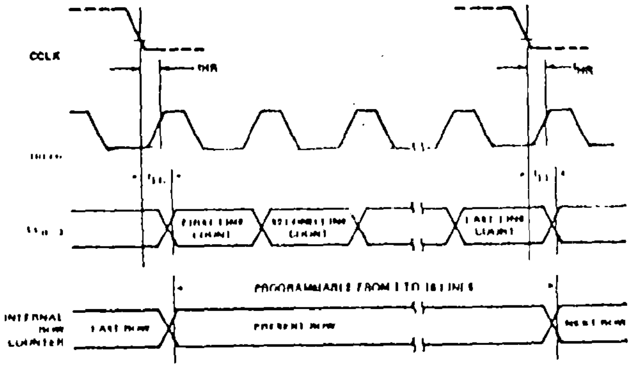
APPENDIX-A4

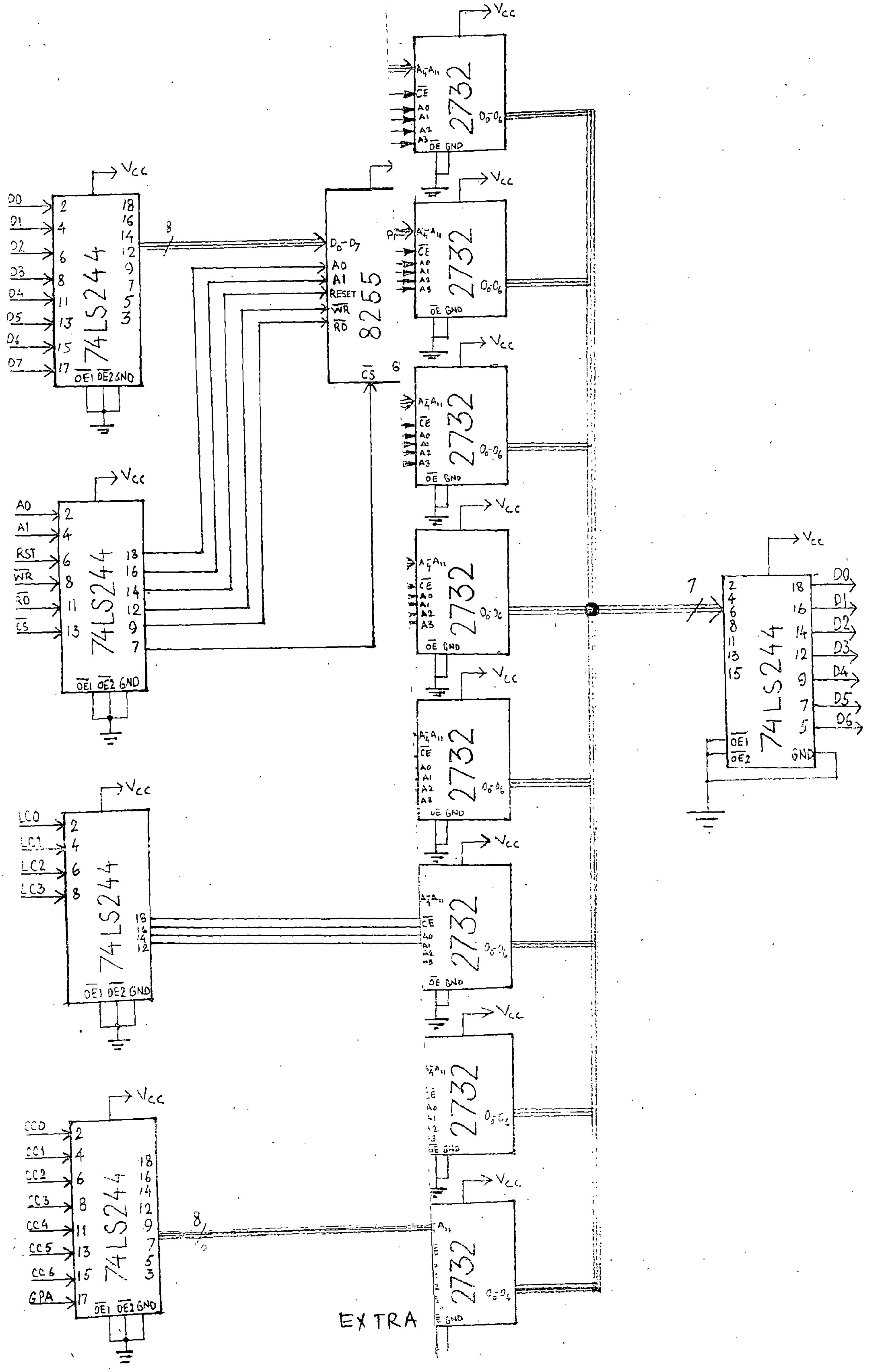


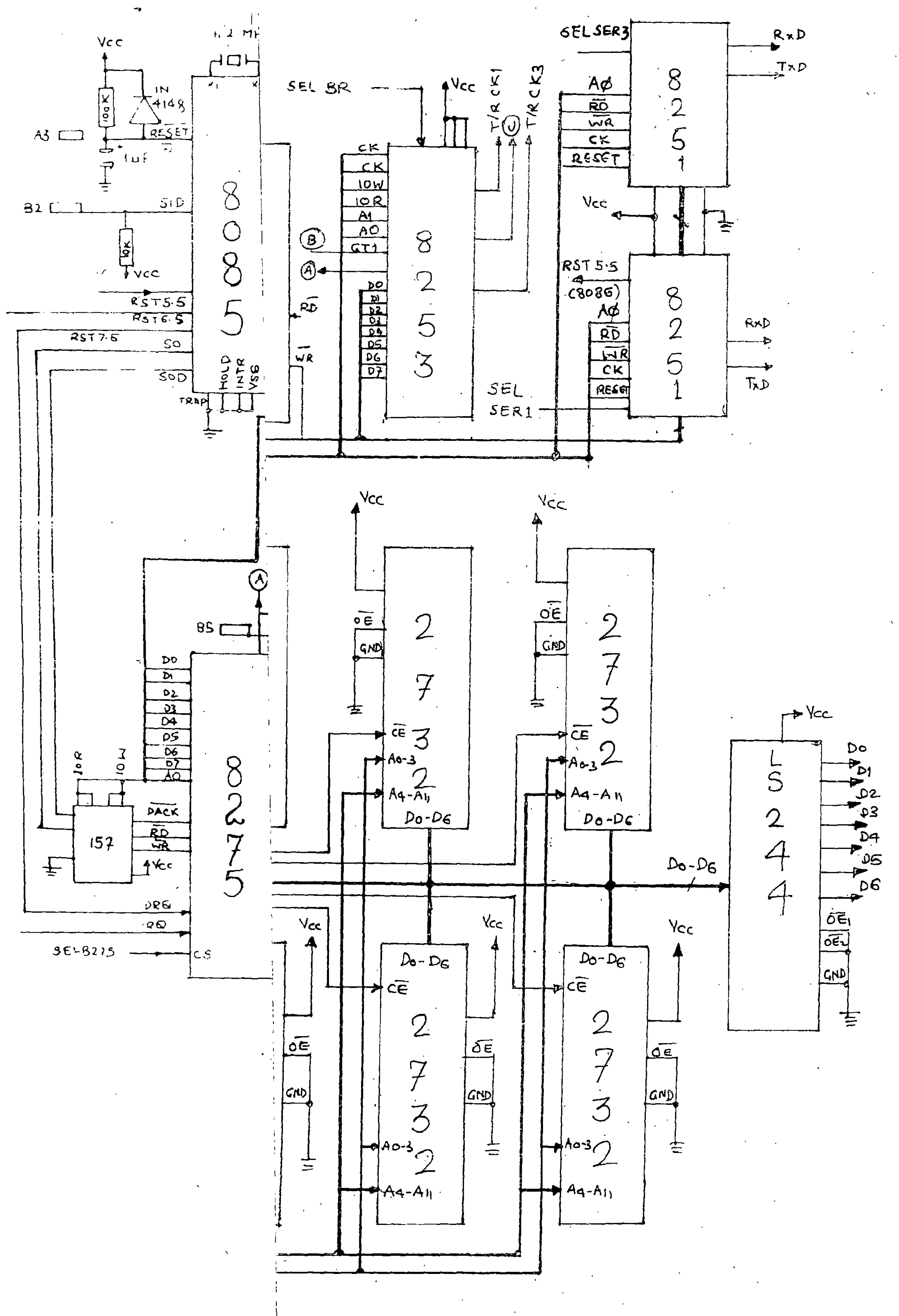
RT TERMINAL (VDT-85)


CCLK
tRQ
DRQ
tWQ
tRQ
DACK
tLR
tRL
WR


LPEN
tPR
tPH


CCLK
tHR
tHR
tLL
tLL
FIRST LINE COUNT
SECOND LINE COUNT
LAST LINE COUNT
PROGRAMMABLE FROM 1 TO 16 LINES
INTERNAL ROW COUNTER
LAST ROW
PRESENT ROW
NEXT ROW


Vcc
D0
D1
D2
D3
D4
D5
D6
D7
2
4
6
8
11
13
15
17
18
16
14
12
9
7
5
3
74LS244
OE1 OE2 GND
8
D0-D7
A0
A1
RESET
WR
RD
8255
CS
6
A0
A1
RST
WR
RD
CS
LC0
LC1
LC2
LC3
CC0
CC1
CC2
CC3
CC4
CC5
CC6
GPA
A4-A11
CE
A0
A1
A2
A3
2732
O0-O6
OE GND
7
EXTRA


Vcc
1N 4148
10K
RESET
A3
1uF
6.2 MH
X1
X
8085
B2
SID
10K
Vcc
RST 5.5
RST 6.5
RST 7.5
SO
SOD
HOLD
INTR
VSS
TRAP
RD
WR
SEL BR
Vcc
T/R CK1
T/R CK3
CK
CK
IOW
IOR
A1
A0
GT1
B
A
D0
D1
D2
D3
D4
D5
D6
D7
8253
SEL SER3
A0
RD
WR
CK
RESET
8251
RxD
TxD
Vcc
RST 5.5
(8086)
A0
RD
WR
CK
RESET
8251
RxD
TxD
SEL
SER1
Vcc
Vcc
OE
GND
2732
CE
A0-3
A4-A11
D0-D6
OE
GND
2732
CE
A0-3
A4-A11
D0-D6
A
BS
D0
D1
D2
D3
D4
D5
D6
D7
A0
8275
IOR
IOW
157
DACK
RD
WR
Vcc
DRQ
RQ
SEL-8275
CS
D0-D6
Vcc
LS244
D0
D1
D2
D3
D4
D5
D6
OE1
OE2
GND
Vcc
Vcc
Vcc
D0-D6
CE
OE
GND
2732
A0-3
A4-A11
OE
GND
D0-D6
CE
OE
GND
2732
A0-3
A4-A11

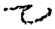
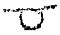
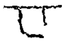
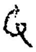
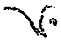
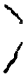
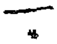
D HARDWARE

APPENDIX - A5

HEXCODE	ENGLISH	DEVANAGARI
00	NUL	ड
01	SOH	झ
02	STX	य
03	ETX	ड़
04	EOT	ख
05	ENQ	रू
06	ACK	फ
07	BEL	अ
09	HT	उ
0B	VT	श्र
0C	FF	ह्
0E	SO	ह
0F	SI	~
10	DLE	ढ
11	DC1	द्ध
12	DC2	द
13	DC3	फ़

HEXCODE	ENGLISH	DEVANAGARI
21	!	!
22	"	"
23	#	हृ
24	$	ह्य
25	%	ड्ड
26	&	ड्ढ
27	'	'
28	(	(
29	)	)
2A	*	ध
2B	+	‿
2F	/	ा
30	0	0
31	1	1
32	2	2
33	3	3
34	4	4
35	5	5
36	6	6
37	7	7
38	8	8
39	9	9
3A	:	:
3B	;	ड़
3C	<	ढ
3E	>	ट
3F	?	?

HEXCODE	ENGLISH	DEVANAGARI
40	@	ि
41	A	अ
42	B	घ
43	C	ब
44	D	क
45	E	त
46	F	थ
47	G	ह
48	H	झ
49	I	ण
4A	J	ङ
4B	K	ळ
4C	L	व
4D	M	ग
4E	N	ड
4F	O	ट
50	P	प
51	Q	न
52	R	ज्
53	S	ी
54	T	ज
55	U	ल
56	V	फ
57	W	स
58	X	च
59	Y	ल्

HEXCODE	ENGLISH	DEVANAGARI
5A	Z	
5B	[	
5C	\	स
5D	]	ण
5E	↑	
5F	-	.
60	—	
61	a	•
62	b	
63	c	
64	d	
65	e	
66	f	
67	g	
68	h	
69	i	
6A	j	
6B	k	
6C	l	
6D	m	
6E	n	
6F	o	
70	p	
71	q	
72	r	
73	s	

APPENDIX - B1

EDITOR COMMANDS

COMMAND	ENGLISH	DEVANAGARI
PRINT	P	प
INSERT	I	न
DELETE	D	क
COPY	C	ब
TRANSFER	T	ज
APPEND	A	अ
(i) INSERT	I	न
(ii) DELETE	D	क
(iii) REPLACE	R	उ

FUNCTION NAME   FILE STORAGE

INPUT   AL

OUTPUT   AL, edit buffer memory space

CALLS   INITIALIZATION,RECEIVER,TRANSMITTER, TRANSMITTER-B

DESTROYS   All registers

DESCRIPTION   This routine receives character from the CRT terminal, stores it in edit buffer and sends it back to the terminal

```
LEBEL   ADDRESS   CONTENTS   MNEMONICS AND          COMMENTS
                             OPERANDS

        01000     BC 00 6F   MOV SP,6F00            initialize SP
        01003     BB C0 10   MOV BX,INITIALIZA-     Initialize all
                             TION                   required pointers
        01006        FF D3   CALL INITIALIZATION
        01008        B0 40   MOV AL,40
        0100A     BB 20 11   MOV BX,TRANSMITTER     Transmit
        0100D        FF D3   CALL TRANSMITTER       the character
        0100F     B8 00 20   MOV AX,20000H          get line address
        01012     A3 06 60   MOV LNADR,AX
START:  01015     BB 00 11   MOV BX,RECEIVER        Receive next
        01018        FF D3   CALL RECEIVER          character
        0101A        AA      STOSB                  store it
        0101B        50      PUSH AX
        0101C        3C 1A   CMP AL,1A              check for Eof
        0101E        74 7E   JZ HALT                if yes, stop
        01020        3C 69   CMP AL,69H             check for special
        01022        7C 0C   JL NLCTR               if not,jump
        01024     A1 02 60   MOV AX,SNCHR           increment a
        01027        40      INCAX                  counter of special
        01028     A3 02 60   MOV SNCHR,AX           characters
        0102B     A1 00 60   MOV AX,NNCHR
        0102E        EB 07   JMP CHECK
NLCTR   01030     A1 00 60   MOV AX,NNCHR           increment a counter
                                                    of normal character
        01033        40      INC AX
```

```
LEBEL   ADDRESS   CONTENTS     MNEMONICS AND        COMMENTS
                               OPERANDS

        01034     A3 00 60     MOV NNCHR,AX
CHECK   01037        3C 4F     CMP AX,4F            check for auto LF
        01039        7D 19     JGE AUTO             if yes,jump
        0103B        58        POP AX
        0103C        3C OD     CMPAL, OD            check for  CR
        0103E        75 02     JNZ LF               if not, jump
        01040        EB OB     JMP RCVNXT
LF      01042        3C OA     CMP AL,OA            check for LF
        01044        75 07     JNZ RCVNXT           if not, jump
        01046     BB 20 10     MOV BX,TRANSMITTER   transmit
        01049        FF D3     CALL TRANSMITTER     character
        0104B        EB OD     JMP  UPDATE          jump
RCVNXT  0104D     BB 20 11     MOV BX,TRANSMITTER       transmit
        01050        FF D3     CALL TRANSMITTER         character
        01052        EB C1     JMP START
AUTO    01054        58        POP AX
        01055     BB 20 11     MOV BX,TRANSMITTER       transmit
        01058        FF D3     CALL TRANSMITTER         character
UDDATE  0105A     A1 02 60     MOVAX, SNMR
        0105D        89 C2     MOVDX,AX
        0105F     A1 00 60     MOVAX, NNCHR
        01062        03 C2     ADD AX, DX
        01064     A3 04 60     MOV LNCTN, AX
        01067        90        NOP
        01068        89 C2     MOV DX, AX
        0106A     A1 06 60     MOVAX, LNADDR
        0106D        03 C2     ADD AX, DX           change the
        0106F    83  C6 02     ADD SI, 02           pointer,having
                                                    line addresse
        01072        89 04     MOV(SI),AX
        01074    A3  06 60     MOV LNADDR,AX
        01077        56        PUSH  SI
        01078        89 FO     MOV AX, SI
        0107A     A3 CC 60     MOV 60CC, AX
        0107D     2D OE 60     SUB AX, 600E
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	01080	BB 02 00	MOVBX, 02	get new pointers which
	01083	F6 F3	DIV AX, BXH	
	01085	81 C6 00 01	ADD S1, 0100H	has addresses of new lines
	01089	89 04	MOV(SI), AX	
	0108B	A3 00 61	MOV NOIN, AX	
	0108E	5E	POP S1	
	0108F	B8 00 00	MOV AX, 00	
	01092	A3 00 60	MOV NNCHR, AX	
	01095	A3 02 60	MOV SNCHR, AX	Store all required pointers.
	01098	A3 04 60	MOV LNCTNAX	
	0109B	E9 77 FF	JMP START	
	0109E	58	POP AX	
	0109F	89 F8	MOVAX, D1	
	010A1	48	DCR AX	
	010A2	A3 0E 60	MOV 600E, AX	
	010A5	BB 40 12	MOV BX,TRANSMITTER_B	provide CR.LF
	010A8	FF D3	CALL TRANSMITTER-B	
	010AA	E9 53 F6	JMP MAIN	jump to EDIT mode

FUNCTION NAME	:	INITIALIZATION
INPUT	:	AL
OUTPUT	:	AL
CALLS		NONE
DESTROYS	:	AX
DESCRIPTION	:	This routinie initializes all the necessary pointers, 8251 USART of the microcomputer kit.

LEBER	ADDRESS	CONTENTS	NNEMONICS AND OPERANDS	COMMENTS
	C1CC0	BE 10 60	MOV SI, 6010	Initialize
	C1CC3	BF 00 20	MOVDI, 2000	SI, DI
	C1CC6	B8 00 00	MOV AX, 00	initialize
	C1CC9	A3 00 60	MOV NNCHR,AX	with 00
	C1CCC	A3 02 60	MOV SNCHR, AX	
	C1CCF	A3 04 60	MOV LNCNT, AX	
	C1CD2	40	INC AX	
	C1CD3	A3 10 61	MOV ERLCN,AX	initialize
	C1CD6	A3 00 61	MOV NOIN,AX	with 01
	C1CD9	B8 00 20	MOVAX, 2000H	
	C1CDC	A3 10 60	MOV PTRA,AX	
	C1CDF	BA F2 FF	MOVDX, FFF2	
	C1CE2	B0 4F	MOVAL, 4F	initialize
	C1CE4	EE	OUT DX	8251
	C1CE5	B0 27	MOVAL, 27	
	C1CE7	EE	OUT DX	
	C1CE8	C3	RET	

FUNCTION NAME	:	RECIEVER
INPUT	:	AL
OUTPUT	:	AL
CALLS	:	None
DESTROYS	:	DX
DESCRIPTION	:	This routine receives a character ASCll code from the CRT terminal in AL.

LEBER	ADDRESS	CONTENTS	NNEMONICS AND OPERANDS	COMMENTS
BACK	C11C0	BA F2 FF	MOVDX, FFF2	
URIVR	C11C3	ED	IN AL, DX	read status
	C11C4	A9 02 00	TEST 02	check for receiver
	C11C7	74 FA	JZ URIVR	ready,if not ready
	C11C9	BA FO FF	MOV DX,FFFC	jump if yes, input
	C11CC	ED	IN AL, DX	data
	C11CD	25 7F 00	AND AL,7F	
	C111C	3D 01 00	CMP AL, 01	check for
	C1113	74 EB	JZ BACK	dummy, if
	C1115	C3	RET	yes jump back

FUNCTION NAME	:	TRANSMITTER
INPUT	:	AL
OUTPUT	:	AL
CALLS	:	NONE
DESTROYS	:	DX
DESCRIPTION	:	This routine transmits back the received character present in AL.

```
LEBER     ADDRESS    CONTENTS     NNEMONICS AND       COMMENTS
                                  OPERANDS

          C1120        50         PUSH  AX
          C1121      BA F2 FF     MOV DX,FFF2         initialize 8251
          C1124      B8 27 00     MOV AL, 27
          C1127        EE         OUT DX
TRMTR     C1128        ED         IN AL, DX           read status
          C1129      A9 01 00     TEST 01             check for TX RDY
          C112C        74 FA      JZ TRMTR            is not, jump
          C112E        58         POP AX
          C112F      BA F0 FF     MOV DX, FFF0        if yes,Transmit
          C1132        EE         OUT DX              data
          C1133        CB         RET
```

APPENDIX - B2

FUNCTION NAME	MAIN
INPUT	AL, data stored in the file.
OUTPUT	AL
CALLS	RECEIVER, TRANSMITTER-B, TRANSREC, PRINT, INSERT, DELETE, COPY TRANSFER, APPEND
DESTROYS	All registers.
DESCRIPTION	This controls all editors functions. This also specifies the language code for the printer.

```
LEBEL  ADDRESS  CONTENTS  MNEMONICS AND           COMMENTS
                          OPERANDS

   0:  0700     BB 00 11  MOV BX,RECEIVER       } Receive
       0703        FF D3  CALL RECEIVER           character
CMPE   0705        3C 45  CMP AL,E                check for E
       0707        74 07  JZ STRI                 if yes, jump
       0709     BB 10 12  MOV BX,TRANSREC
       070C        FF D3  CALL TRANSREC
       070E        EB F5  JMP CMPE
STR1   0710     BB 20 11  MOV BX,TRANSMITTER}     Transmit
       0713        FF D3  CALL TRANSMITTER        character
       0715     BB 00 11  MOV BX,RECEIVER       } Receive
       0718        FF D3  CALL RECEIVER           next
CMPD   071A        3C 44  CMP AL,D                Check for D
       071C        74 07  JZ STR2                 if yes,jump
       071E     BB 10 12  MOV BX,TRANSREC
       0721        FF D3  CALL TRANSREC
       0723        EB F5  JMP CMPD
STR2   0725     BB 20 11  MOV BX,TRANSMITTER      Transmit
       0728        FF D3  CALL TRANSMITTER      } character
       072A     BB 00 11  MOV BX,RECEIVER       } Receive
       072D        FF D3  CALL RECEIVER           next
CMPCR  072F        3C OD  CMP A1,OD               check for 'CR'
       0731        74 07  JZ STR-3                if yes, jump
       0733     BB 20 12  MOV BX,TRANSREC.
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	0736	FF D3	CALL TRANSREC	
	0738	EB F5	JMP CMPCR	
STR3	073A	BB 40 12	MOV BX,TRANSMITTER-B	Provide CR,LP
	073D	FF D3	CALL TRANSMITTER-B	
S-UP	073F	BB 00 11	MOV BX,RECEIVER	Receive next
	0742	FF D3	CALL RECEIVER	
AS-UP	0744	3C 50	CMP AL,P	Check for 'P'
	0746	75 07	JNZ I	if not, jump
	0748	BB A0 12	MOV BX,PRINT	do a printing operation
	074B	FF D3	CALL PRINT	
	074D	EB F0	JMP S-UP	go back
I	074F	3C 49	CMP AL,I	check for 1
	0751	75 07	JNZ D	if not, jump
	0753	BB B0 1C	MOV BX,INSERT	Insert a line
	0756	FF D3	CALL INSERT	
	0758	EB E5	JMP S-UP	
D	075A	3C 44	CMP AI,D	Check for D
	075C	75 07	JNZ C	Ifnot, jump
	075E	BB 30 15	MOV BX,DELETE	delete required line
	0761	FF D3	CALL DELETE	
	0763	EB DA	JMP S-UP	
C	0765	3C 43	CMP AL,C	check for C
	0767	75 07	JNZ T	if not, jump
	0769	BB 50 16	MOV BX,COPY	Copy a required line
	076C	FF D3	CALL COPY	
	076E	EB CF	JMP-S-UP	
T	0770	3C 54	CMPAL,T	check for T
	0772	75 07	JNZ A	if not, jump
	0774	BB 00 18	MOV BX,TRANSFER	Transfer a required line
	0777	FF D3	CALL TRANSFER	
	0779	EB C4	JMP S-UP	
A	077B	3C 41	CMP AL,A	check for A
	077D	75 07	JNZ QJ	if not, jump
	077F	BB 10 1A	MOV BX,APPEND	} Append a required line
	0782	FF D3	CALL APPEND	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	0784	EB B9	JMP S-UP	
QJ	0786	3C 45	CMP AL,E	check for E
	0788	75	JNZ CPM	if not jump
	078A	50	PUSH AX	
TERM	078B	BB 00 11	MOV BX,RECEIVER	Receive next character
	078E	FF D3	CALL RECEIVER	
	0790	BO 20 11	MOV BX,TRANSMITTER	Transmit it
	0793	FF D3	CALL TRANSMITTER	
JGN	0795	3C OD	CMP AL,OD	Check for CR
	0797	75	JNZ TERM	if not, jump
CPM	0799	58	POP AX	
	079A	EB	JMP STOP 1	
	079C	3C 4D	CMP AL,M	check for M
	079E	75	JNZ CTRLZ	if not, jump
	07A0	50	PUSH AX	
TRME	07A1	BB 00 11	MOV BX,RECEIVER	Receive next character
	07A4	FF D3	CALL RECEIVER	
TMRE	07A6	BB 20 11	MOV BX,TRANSMITTER	transmit it
	07A9	FF D3	CALL TRANSMITTER	
	07AB	3C OD	CMP AL,OD	check for CR
	07AD	75 F7	JNZ TMRE	if not, jump
	07AF	58	POP AX	
	07B0	EB 06	JMP STOP-2	
STOP1	07B2	A3 FO OF	MOV LDCD,AL	givelanguage code
	07B5	EB 51 01	JMP MPRINT	Printer routine
STOP2	07B8	A3 FO OF	MOV LDCD,AL	
	07BB	EB 55 01	JMP MPRINT	
CTRLZ	07BE	3C 1A	CMP AL,1A	check for EOF
	07C0	75 83	JNZ S-UP	
	07C2	BO 40	MOV AL,40	
	07C4	BB 20 11	MOV BX,TRANSMITTER	Transmit character
	07C7	FF D3	CALL TRANSMITTER	
	07C9	CC	HALT	

FUNCTION NAME	ADJUST
INPUT	AL
OUTPUT	DX,
CALLS	NONE
DESTROYS	AX, CX, DX
DESCRIPTION	This routine converts a Hexadecimal ASCII code to decimal. It gives line number to each stored lines. It keepts a record of number character present in the line.

```
LEBEL  ADDRESS   CONTENTS   MNEMONICS AND      COMMENTS
                            OPERANDS

       1140      3D 30 00   CMP AX,30          check for zero
       1143         7D 04   JGE CNTNU          if greater,jump
       1145         89 DO   MOV  AX,DX
       1147         EB 2F   JMP SEARCH         if not,jump
CNTNU  1149         50      PUSH AX
       115A         89 DO   MOV  AX,DX
       114C         D1 EO   SAL  AX            shift left the
       114E         D1 EO   SAL  AX            nibble.
       1150         D1 EO   SAL  AX
       1152         D1 EO   SAL  AX
       1154         89 C2   MOV  DX,AX
       1156         58      POP  AX
       1157      2D 30 00   SUB  AX,30         get difference
       115A         01 DO   ADD AX,DX          add to lower nibble
       115C      3D 25 00   CMP  AX,25H        check for 25
       115F         7F 05   JG SCMN            if greater,jump
       1161      2D 06 00   SUB  AX,06         if not adjust the
       1164         EB 12   JMP SEARCH         decimal number
SCMN   1166      3D 40 00   CMP  AX,40H        check for 40H
       1169         7F 05   JG TCMN            if greater jump
       116B      2D OC 00   SUB  AX,OCH        if not adjust next
       116E         EB 08   JMP SEARCH         decimal number
TCMN   1170      3D 60 00   CMP  AX,60H        check for 60H
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	1173	7F 34	JG RET	if greater, jump.
	1175	2D 12 00	SUB AX,12H	if not, adjust
SEARCH	1178	90	NOP	decimal number
	1179	BE 10 61	MOV SI,6110	store decimal
	117C	3B 44 FD	CMP AX, (SI -10)	line numbers in the pointers
	117F	7F 1B	JG RETN	
	1181	3B 04	CMP (SI),AX	
	1183	74 05	JZ OF ST	
OFST	1185	83 C6 02	ADD SI,02H	
	1188	EB F7	JMP REPET	
	118A	81 EE 0001	SUB SI, 0100H	
	118E	8B 14	MOV DX, (SI)	get starting
	1190	89 C3	MOV BX,AX	address of the
	1192	A3 06 61	MOV 6106,AX	line in DX
	1195	89 F0	MOV AX,SI	
	1197	A3 08 61	MOV 6108,AX	
	119A	90	NOP	
RETN	119B	C3	RET	
	119C	B8 00 00	MOV AX,CO	load printer
	119F	A3 06 61	MOV 6106,AX	with zero
	11A2	BB 12 40	MOV BX,TRANSMITTER-B	Provide CR,LF
	11A5	FF D3	CALL TRANSMITTER-B	
	11A7	C3	RET	

FUNCTION NAME	RDFLN
INPUT	Data stored in the file
OUTPUT	AL
CALLS	RDCHR
DESTROYS	All registers
DESCRIPTION	This routine reads a required line from the stored data. For reading each character RDCHR is used.

```
LEBEL   ADDRESS   CONTENTS   MNEMONICS AND          COMMENTS
                             OPERANDS

   0:   11C0      A1 02 61   MOV  AX,6102           get starting
        11C3         89 C6   MOV  SI,AX             address of Ist line
        11C5      A1 04 61   MOV  AX,61C4           get starting
        11C8         89 C1   MOV  CX,AX             address of 2nd line
        11CA      A1 02 61   MOV  AX,6102
        11CD         3B C1   CMP  AX, CX            compare both if
        11CF         74 1C   JZ AGAIN               same jump if not
HERE    11D1      BB F5 11   MOV BX,RD CHR          read character
        11D4         FF D3   CALL RDCHR
        11D6         89 EC   MOV  AX,51
        11D8         3B C1   CMPAX,CX               check for required
        11DA         75 F5   JNZ HERE               line
        11DC      A1 02 61   MOV  AX,6102           if not, continued
        11DF         89 C6   MOV SI,AX
AGAIN   11E1      BB F5 11   MOV BX,RDCHR           read characters
        11E4         FF D3   CALL RDCHR             from required line
        11E6         8B 04   MOV  AX (SI)
        11E8         3C 0A   CMP  AX,LF             check for LF
        11EA         74 04   JZ TRB                 if yes, stop
        11EC         3C 1A   CMP AL,IA              if not, check EOF
        11EE         75 F1   JNZ AGAIN              if not,continue
TRB     11F0      BB 40 12   MOV BX,TRANSMITTER-B   provide CR,LF
        11F3         FF D3   CALL TRANSMITTER-B
        11F5         C3      RET                    Return
```

FUNCTION NAME	RDCHR
INPUT	AL, data stored in the buffer.
OUTPUT	AL
CALLS	None
DESTROYS	DX, AL
DESCRIPTION	This routine reads each character from stored data and sends it to the CRT terminal

```
LEBEL   ADDRESS   COMMENTS   MNEMONICS AND        COMMENTS
                             OPERANDS

        0:11F5    BA F2 FF   MOV DX,FFF2
TRMTR   11F8         EC      IN  AL               Read status
                                                  if 8251
        11F9         A8 01   TEST 01
        11FB         74 FB   JZ TRMTR             if transmitter Read
        11FD         AC      LODSB                get AL with data
        11FE      BA FC FF   MOV DX,FFFC          send it to CRT
        1201         EE      Out DX               terminal
        1202      BA F2 FF   MOV DX,FFF2
URCVR   1205         EC      IN AL                check for
        1206         A8 02   TEST 02              Receiver Ready
        1208         74 FB   JZ URCVR             if not, check
        120A         C3      RET                  Return
```

FUNCTION NAME	TRANSREC
INPUT	AL
OUTPUT	AL
CALLS	None
DESTROYS	AL,DX
DESCRIPTION	This routine sends '?' after receiving incorrect character. Then this routine receiver next characters.

```
LEBEL   ADDRESS   CONTENTS    MNEMONICS AND      COMMENTS
                              OPERANDS

        0:1210    BA F2 FF    MOV  DX,FFF2       Read status
TRMR-3   1213        EC       IN  AL,DX          of 8251
         1214        A8 01    TEST 01
         1216        74 FB    JZ  TRMR-3         check for
         1218        B0 3F    MOV  AL,3F         transmitter  ready
         121A     BA FC FF    MOV  DX,FFFC       if yes, output data
         121D        EE       Out DX
BACK     121E     BA F2 FF    MOV DX,FFF2        check for
RCVR-S   1221        EC       IN  AL,DX          Receiver,Ready
         1222        A8 02    TEST 02
         1224        74 FB    JZ RCVR-S          if not, check
         1226     BA FC FF    MOV  DX,FFFC       if  yes,receive
         1229        EC       IN  AL,DX          check for
         122A        3C 01    CMPAL,01           dummy character
         122C        74 FC    JZ BACK
         122E        C3       RET
```

FUNCTION NAME	TRANSMITTER-B
INPUT	AL
OUTPUT	AL
CALLS	TRANSMITTER
DESTROYS	AL,DX,BX
DESCRIPTION	This routine sends ASCII codes of CR,LF and * whenever required.

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	0:1240	B0 0D	MOV AL,0D	load AL with CR
	1242	BB 20 11	MOV BX,TRANSMITTER	Output the Code
	1245	FF D3	CALL TRANSMITTER	
	1247	B0 0A	MOV AL,0A	load AL with LF
	1249	BB 20 11	MOV BX,TRANSMITTER	output the code
	124C	FF D3	CALL TRASMITTER	
	124E	B0	MOV AL *	load AL with '*'
	1250	BB 20 11	MOV BX,TRANSMITTER	output the code
	1253	FF D3	CALL TRANSMITTER	
	1255	C3	RET	Return

```
ME      PRINT
        AL, data present in the edito buffer
        AL
        RECEIVER,TRANSMITTER,ADJUST',RDFLN,TRANSMITTER-B
        All registers.
I       This function reads required data from the
        edit buffer and sends it to the CRT terminal
--------------------------------------------------------------------------------
RESS   CONTENTS    MNEMONICS AND               COMMENTS
                   OPERANDS
--------------------------------------------------------------------------------
)       BB 20 11   MOV  BX,TRANSMITTER        } Transmit the
3          FF D3   CALL TRANSMITTER             character
3       BB C0 11   MOV  BX,RECEIVER             Receive
3          FF D3   CALL RECEIVER              } next character
)       BB 20 11   MOV  BX,TRANSMITTER          transmit it to
)          FF D3   CALL TRANSMITTER           } display
2       2D 30 00   SUB  AX,30H                  Subtract 30 from AL
5          89 C2   MOV  DX,AX                   put it in DX
7          52      PUSH DX                      save DX
3       BB C0 11   MOV  BX,RECEIVER             Receive next
3       FF D3      CALL RECEIVER              } character
D          5A      POP  DX                      Restore DX
E          3C CD   CMP  AX, CR                  Check for'CR'
D          90      NOP
1          75 31   JNZ  COMMA                   if not then jump
3          50      PUSH AX                      Save AX
4       A1 06 61   MOV  AX,6106                 Get line number
7          89 C1   MOV  CX,AX                   put it in CX.
9          58      POP  AX                      restore AX
A       83 F9 09   CMP  CL,09                   check for 09H
D          7F 14   JG STAR                      if greater jump
F       BB 40 11   MOV  BX,ADJUST               get address of
2          FF D3   CALL ADJUST                } next line, line
                                                number
4       A1 06 61   MOV  AX,6106
7          3C C0   CMP  AX,00                   Compare for 0P,
9          74 18   JZ STP                       if 00 jump
B          89 D0   MOV  AX,DX
D       A3 03 61   MOV  6102,AX                 Store line address
O       A3 04 61   MOV  6104,AX                 in two pointers.
3       BB 40 12   MOV  BX,TRANSMITTER-B        Cursor on next
6          FF D3   CALL TRANSMITTER-B          }line
--------------------------------------------------------------------------------
```

```
LEBEL   ADDRESS  CONTENTS   MNEMONICS AND          COMMENTS
                            CPERANDS

        12E8     BB 10 11   MOV BX,RDFLN           Read characters
        12EB        FF D3   CALL RDFLN             from file.
        12ED     B8 00 00   MOV AX,00
        12F0     A3 06 61   MOV 6106,AX
STP     12F3        C3       RET
        12F4        3C 2C   CMP AL,COMMA           Compare for comma,
        12F6        74 03   JZ MHN                 if yes print it.
        12F8     E9 8B 00   JMP NUMBER             Otherwise take next
MHN     12FB        52      PUSH DX                line
        12FC     BB 20 11   MOV BX,TRANSMITTER    }transmit
        12FF     FF D3      CALL TRANSMITTER       character
        1301        5A      POP DX
CORRECT1302         50      PUSH AX
        1303     A1 06 61   MOV AX,6106             get line number
        1306        83 C1   MOV CX,AX               save it in CX
        1308        58      POP AX
        1309     83 F9 09   CMP CX,09H              Compare with 09
        130C        7F 14   JG NXT                  if greater jump
        130E     BB 40 11   MOV BX,ADJUST          }calculate new
        1311     FF D3      CALL,ADJUST             address of line
        1313     A1 06 61   MOV AX,6106
        1316     3C 00      CMP AX,00
        1318     74 6B      JZ RNN
        131A     89 D0      MOV AX,DX               get new address
        131C     A3 02 61   MOV 6102,AX             store it
        131F     A3 04 61   MOV 6104,AX
NXT     1322     BB 00 11   MOV BX,RECEIVER         Receive next
        1325     FF D3      CALL RECEIVER           character
TEST    1327     3C 30      CMP AX,30H
        1329     7C 04      JL ERROR                 compare with
        132B     3C 39      CMP AL,39H               30 and 39H
        132D     7E 09      JLE FURTHER
ERROR   132F     50         PUSH AX
        1330     BB 10 12   MOV BX,TRANSREC          if not a
                                                     required
        1333        FF D3   CALL TRANSREC            character send
                                                     error message
        1335        58      POP AX
        1336     EB EF      JMP TEST                 take correct
                                                     character
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
FURTHER	1338	52	PUSH DX	
	1339	BB 20 11	MOV BX TRANSMITTER	transmit the character
	133C	FF D3	CALL TRANSMITTER	
	133E	5A	POP DX	
CTT	133F	2D 30 00	SUB AX,30	convert it to decimal
	1342	89 C2	MOV DX,AX	
	1344	52	PUSH DX	
	1345	BB 00 11	MOV BX,RECEIVER	Receive next character
	1348	FF D3	CALL RECEIVER	
	134A	5A	POP DX	
	134B	3C 0D	CMP AX,0D	check for CR
	134D	74 03	JZ HHN	if yes jump
	134F	E9 81 00	JMP EITHER	if not take newline
HHN	1352	BB 40 11	MOV BX,ADJUST	calculate address
	1355	FF D3	CALL ADJUST	
	1357	A1 06 61	MOV AX,6100	find no. of lines present
	135A	3C 00	CMP AX,00	
	135C	74 29	JZ RNN	
	135E	89 D0	MOV AX,DX	have address in memory
	1360	A3 04 61	MOV 6104 AX	
	1363	50	PUSH AX	
	1364	A1 02 61	MOV AX,6102	get address of the line
	1367	89 C1	MOV CX,AX	
	1369	58	POP AX	
	136A	3B C1	CMP AX,CX	
	136C	70 07	JGE RNE	
	136E	BB 40 11	MOV BX,TRANSMITTER-B	Provide CR,LF
	1371	FF D3	CALL TRANSMITTER-B	
	1393	EB 10	JMP RNN	
RNF	1375	BB 40 12	MOV BX TRANSMITTER-B	Provide CR,LF
	1378	FF D3	CALL TRANSMITTER-B	
	137A	BB C0 11	MOV BX,RDFLN	Read character }from the file
	137D	FF D3	CALL RDFLN	
	137F	B8 00 00	MOV AX,00	
	1382	A3 06 61	MOV 6106 AX	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
RNN	1385	C3	RET	
	1386	3C 30	CMP AL,30L	
	1388	7C 04	JL ER	Compare character between 30 and 39H
	138A	3C 39	CMP AL,30L	
	138C	7E 08	JLE TMR	
ER	138E	BB 10 12	MOV BX,TRANSREC	if non-required character send error
	1391	FF D3	CALL TRANSREC	
	1393	E9 28 FF	JMP LNG	
TMR	1396	52	PUSH DX	
	1397	BB 20 11	MOVBX,TRANSMITTER	transmit character
	139A	FF D3	CALL TRANSMITTER	
	139C	5A	POP DX	
NUMBER	139D	BB 40 11	MOV BX ADJUST	Calculate address of next line
	13A0	FF D3	CALL ADJUST	
	13A2	A1 06 61	MOV AX,6106	
	13A5	3C 00	CMP AX,00	Compare with OC if yes, Jump.
	13A7	74 DC	JZ RNN	
	13A9	89 D0	MOV AX,DX	
	13AB	A3 02 61	MOV 6102,AX	store address in the memory
	13AE	A3 04 61	MOV 61C4 AX	
	13B1	BB 00 11	MOV BX,RECEIVER	Receive next character
	13B4	FF D3	CALL RECEIVER	
CHECK	13B6	3C 0D	CMP AX, 0D	compare for CR
	13B8	74 77	JZ DECNDD	if yes, jump
	13BA	3C 2C	CMP AX,2C	compare for ,
	13BC	75 08	JNZ ADDN	if not equal jump
	13BE	BB20 11	MOV BX,TRANSMITTER	transmit }character
	13C1	FF D3	CALL TRANSMITTER	
	13C3	E9 3C FF	JMP CORRECT	jump
ADDN	13C6	B8 00 00	MOV AX,00	
	13C9	A3 06 61	MOV 6106,AX	
	13CC	BB 10 12	MOV BX,TRANSREC	if not correct give error and receive
	13CF	FF D3	CALL TRANSREC	
	13D1	EB E3	JMP CHECK	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
EITHER	13D3	50	PUSHAX	save AX
	13D4	3D 30	CMP AX,30	
	13D6	7C 04	JL RORE	compare with 3C and 39H
	13D8	3C 39	CMP AX,39	
	13DA	7E 09	JLE NPAGE	
RORE	13DC	BB 10 12	MOV BX,TRANSREC	
	13DF	FF D3	CALL TRANS REC.	
	13E1	90	POP AX	
	13E2	E9 66 FF	JMP EITHER	
NPAGE	13E5	52	PUSH DX	
	13E6	BB 20 11	MOV BX,TRANSMITTER	transmit
	13E9	FF D3	CALL TRANSMITTER	character
	13EB	5A	POP DX	
	13EC	BB 40 11	MOV BX,ADJUST	calculate new address
	13EF	FF D3	CALL ADJUST	
	13F1	A1 06 61	MOV AX,6106	
	13F4	3C 00	CMP AX 00	compare with if yes, jump
	13F6	74 38	JZ RNG	
	13F8	89 D0	MOV AX,DX	
	13FA	A3 04 61	MOV 6104 AX	
	13FD	50	PUSH AX	
	13FE	A1 02 61	MOV AX,6102	get address of the line store it in CX
	1401	89 C1	MOV CX,AX	
	1403	58	POP AX	
	1404	3B C1	CMP AX,CX	
	1406	70 07	JG GGG	
	1408	BB 40 12	MOV BX TRANSMITTER-B	Provide CR.LF
	140B	FF D3	CALL TRANSMITTER-B	
	140D	EB 21	JMP RNG	
GGG	140F	BB 00 11	MOV BX,RECEIVER	Receive character
	1412	FF D3	CALL RECEIVER	
CR	1414	3D 0D 00	CMPAX,0D	
	1417	74 07	JZ REMNDR	
	1419	BB 10 12	MOV BX,TRANSREC	give error if incorrect character
	141C	FF D3	CALL TRANS REC }	
	141E	EB F4	JMP CR	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
REMNDR	1420	BB CO 11	MOV BX,RDFLN	Read characters from edit buffer
	1423	FF D3	CALL RDFLN	
	1425	B8 00 CO	MOV AX,00	
	1428	A3 06 61	MOV 6106 AX	
RGG	142B	BB 40 12	MOV DX,TRANSMITTER-B	Provide CR,LE
	142E	FF D3	CALL TRANSMITTER-B	
RNG	1430	C3	RET	
DECNDD	1431	E9 8F FE	JMP DECN	Jump back

FUNCTION NAME   DELETE

INPUT   AL, data stored in the memory location 4000 H onwards

OUTPUT   AL

CALLS   RECEIVER,TRANSMITTER,ADJUST,TRANSMITTER-B

DESTROYS   All registers

DESCRIPTION   This editor command deletes a required line from the file of data stored by keeping it in edit buffer.

```
LEBEL    ADDRESS  CONTENTS   MNEMONICS AND          COMMENTS
                             OPERANDS

     0:  1530     BB 20 11   MOV BX,TRANSMITTER     Transmit
         1533        FF D3   CALL TRANSMITTER       character
         1535     B8 00 00   MOV AX,00
         1538     A3 06 61   MOV 6106,AX
         153B     BB 00 11   MOV BX,RECEIVER        Receive next
         153E        FF D3   CALL RECEIVER          character
         1540     BB 20 11   MOV BX,TRANSMITTER     transmit back
         1543        FF D3   CALL TRANSMITTER       the character
         1545     2D 30 00   SUB AX,30              Subtract 30 to
         1548        89 C2   MOV DX,AX              convert it into
                                                    decimal
         154A        52      PUSH DX
         154B     BB 00 11   MOV BX,RECEIVER        Receive character
         154E        FF D3   CALL RECEIVER
         1550        5A      POP DX
         1551        3C 0D   CMP AX,0D              check for CR
         1553        75 0C   JNZ NXTCHR             if not jump
         1555     BB 40 11   MOV BX,ADJUST          calculate address
         1558        FF D3   CALL ADJUST            of the line
         155A        89 D0   MOV AX,DX
         155C     A3 02 61   MOV 6102,AX            store the address
         155F        EB 21   JMP OKED
         1561        52      PUSH DX
NXTCHR:  1562     BB 20 11   MOV BX,TRANSMITTER     Transmit
         1565        FF D3   CALL TRANSMITTER       the character
         1567        5A      POP DX
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENT
	1568	BB 40 11	MOV BX,ADJUST	Calculate address of new line
	156B	FF D3	CALL ADJUST	
	156D	89 DO	MOV AX,DX	Store it
	156F	A3 02 61	MOV 6102,AX	
	1572	BB 00 11	MOV BX,RECEIVER	Receive next character
	1575	FF D3	CALL RECEIVER	
CMPRN	1577	3C OD	CMP OD	Check for CR
	1579	74 07	JZ OKED	if yes, jump
	157B	BB 10 12	MOV BX,TRANSREC	
	157E	FF D3	CALL TRANS REC	
	1580	EB F5	JMP CMARN	
OKED	1582	BB 40 12	MOV BX,TRANSMITTER-B	provide CR,LF
	1585	FF D3	CALL TRANSMITTER-B	
	1587	A1 08 61	MOV AX (6108)	
	158A	89 C7	MOV DI,AX	get SI,DI
	158C	89 C6	MOV SI,AX	
	158E	8B 04	MOV AX,(SI)	
	1590	83 C6 02	ADD SI,02	
	1593	8B 1C	MOV BX,(SI)	
	1595	2B D8	SUB BX,AX	
	1597	56	PUSH SI	
	1598	81 C6 FE 00	ADD SI,FE H	get line number in decimal
	159C	8B OC	MOV CX,(SI)	
	159E	89 C8	MOV AX,CX	
	15A0	A3 0A 61	MOV 610A,AX	
	15A3	5E	POP SI	
LOOP	15A4	29 1C	SUB (SI),BX	change the starting addresses of retu required lines
	15A6	83 C6 02	ADD SI,02	
	15A9	41	INC CX	
	15AA	A1 00 61	MOV AX,(6100)	
	15AD	3B C1	CMP AX,CX	check for last line
	15AF	75 F3	JNZ LOOP	if not continue
	15B1	29 1C	SUB (SI),BX	
	1583	A1 02 61	MOV AX,6102	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	15B6	89 C7	MOV DI,AX	set SI and DI
	15B8	03 C3	ADD AX,BX	
	15BA	89 C6	MOV SI,AX	
	15BC	FC	CLD	
BOL	15BD	AC	LODSB	shift the data
	15BE	AA	STOSB	upward
	15BF	3C 1A	CMP A1,1A	
	15C1	75 FA	JNZ BDL	
	15C3	A1 08 61	MOV AX,6108	
	15C6	89 C7	MOV D1,AX	set SI and DI
	15C8	89 C6	MOV SI,AX	
	15CA	83 C6 02	ADD SI,02	
	15CD	A1 0A 61	MOV AX,610A	
	15D0	89 C1	MOV CX,AX	update
	15D21	FC	CLD	pointers
EXT	15D3	AD	LODSW	which has starting
	15D4	AB	STOSW	address of lines.
	15D5	41	INC CX	
	15D6	A1 00 61	MOV AX,6100	
	15D9	3B CI	CMP AX,CX	check for last line
	15D8	75 F6	JNZ EXT	if not jump
	15DD	A1 00 61	MOV AX,6100	
	15E0	48	DCRAX	decrement a count
	15E1	A3 00 61	MOV 6100,AX	of number lines.
	15E4	BB 40 12	MOV BX,TRANSMITTER-B	Provide
	15E7	FF D3	CALL TRANSMITTER-B	CR,LF
	15E9	C3	RET	

FUNCTION NAME COPY
INOUT AL, data stored in the buffer
OUTPUT AL
CALLS RECEIVER,TRANSMITTER,ADJUST,TRANSMITTER-B
DESTROYS All Registers
DESCRIPTION This routine can copy a specified line to required position through edit buffer.

```
LEBEL   ADDRESS   CONTENTS   MNEMONICS AND          COMMENTS
                             OPERANDS

   0:   1650      BB 20 11   MOV BX,TRANSMITTER      Transmit
                                                   } character
        1653         FF D3   CALL TRANSMITTER
        1655      B8 00 00   MOV AX,00H
        1658      A3 06 61   MOV 6106,AX
        165B      BB 00 11   MOV BX,RECEIVER         Receive Ist
                                                   } linenumber
        165E         FF D3   CALL RECEIVER
        1660      BB 20 11   MOV BX,TRANSMITTER      transmit it
        1663         FF D3   CALL TRANSMITTER        to display
        1665      2D 30 00   SUB AX,30               save the line
                                                   } number
        1668         89 C2   MOV DX,AX
        166A         52      PUSH DX
        166B      BB 00 11   MOV BX,RECEIVER         Receive next
                                                    }character
        166E         FF D3   CALL RECEIVER
        1670         5A      POP DX
 CGRT   1671         3C 0D   CMP AX,0D               check for :CR
        1673         75 07   JNZ NCOM                if not jump
        1675      BB 10 12   MOV BX,TRANSREC         receive required
        1678         FF D3   CALLTRANSREC            character
        167A         FB F5   JMP CGRT
        167C         52      PUSH DX
 NCOM   167D      BB 20 11   MOV BX,TRANSMITTER      Transmit character
        1680         FF D3   CALL TRANSMITTER
        1682         5A      POP DX
        1683         3C 2C   CMP AX,COMMA            chefck for comma
        1685         74 23   JZ  ADJ                 if zero jump
NXTBIT  1687      BB 40 11   MOV BX,ADJUST           calculate line
                                                     number in decimal
        168A         FF D3   CALL ADJUST             and its address
```

```
LEBEL    ADDRESS   CONTENTS    MNEMONICS AND          COMMENTS
                               OPERANDS

         168C      A1,08 61    MOV  AX,6108
         168F      A3,08 60    MOV  6008,AX
         1692         89 D0    MOV  AX,DX             get address and
         1694      A3 02 61    MOV  6102,AX           store it
         1697      A3 04 61    MOV  6104,AX
         169A      BB 00 11    MOV  BX,RECEIVER       Receive next
         169D         FF D3    CALL RECEIVER          character
CMA      169F         3C 2C    CMP, COMMA             check for comma
         16A1         74 16    JZ TRANS               if zero jump
         16A3      BB 10 12    MOV  BX,TRANSREC       Receive correct
         16A6         FF D3    CALL TRANSREC          character
         16A8         EB F5    JMP  CMA
         16AA      BB 40 11    MOV  BX,ADJUST         calculate  address
         16AD         FF D3    CALL ADJUST            of new line
         16AF         89 D0    MOV  AX,DX
         16B1      A3 02 61    MOV  6102,AX           store address
         16B4      A3 04 61    MOV  6104,AX           in pointers
         16B7      A1 08 61    MOV  AX,6108
         16BA      A3 08 60    MOV  6008,AX
         16BD         EB 05    JMP  TRANS-1
TRANS    16BF      BB 20 11    MOV  BX,TRANSMITTER
         16C2         FF,D3    CALL TRANSMITTER       transmit
                                                      character
TRANS-1  16C4      BB 00 11    MOV  BX,RECEIVER       Receive next
         16C7         FF D3    CALL RECEIVER          character
LEQ      16C9         3C 30    CMP  AX,30
         16CB         7C 04    JLERR                  check for correct
         16CD         3C 39    CMP  AX,39H            character
         16CF         7E 07    JLE  CHK
         16D1         BB 10 12 MOV  BX,TRANS REC
           D4         FF D3    CALL TRANSREC
           D6         EB F1    JMP  LEQ
         16D8      BB 20 11    MOV  BX,TRANSMITTER    transmit
         16DB         FF D3    CALL TRANSMITTER       character
         16DD      2D 30 00    SUB  AX,30
         16E0         89 C2    MOV  DX,AX
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	16E2	52	PUSH DX	
	16E3	BB 00 11	MOV BX,RECEIVER	Receive next character
	16E6	FF D3	CALL RECEIVER	
	16E8	5A	POP DX	
	16E9	3C 0D	CMP AX,0D	Check for CR
	16EB	75 12	JNZ MYB	
	16ED	BB 40 11	MOV BX,ADJUST	Calculate address of new line
	16F0	FF D3	CALL ADJUST	
	16F2	89 DC	MOV AX,DX	
	16F4	A3 04 61	MOV 6104,AX	store it
	16F7	A1 08 61	MOV AX,6108	
	16FA	A3 0A 60	MOV 600A,AX	
	16FD	EB 27	JMP PRB1	
	16FF	52	PUSH DX	
MYB	1700	BB 20 11	MOV BX,TRANSMITTER	transmit character
	1703	FF D3	CALL TRANSMITTER	
	1705	5A	POP DX	
	1706	BB 40 11	MOVBX ADJUST	
	1709	FF D3	CALL ADJUST	find line address
	170B	A1 08 61	MOV AX,6108	
	170E	A3 0A 60	MOV 600A,AX	save it.
	1701	89 D0	MOV AX,DX	
	1713	A3 04 61	MOV 6104,AX	
	1716	BB 00 11	MOV BX,RECEIVER	Receive next character
	1719	FF D3	CALL RECEIVER	
PREPR	171B	3C 0D	CMP 0D .	check for CR
	171D	74 07	JZ PRB1	if yes, jump
	171F	BB 10 12	MOV BX,TRANS REC	
	1722	FF D3	CALL TRANSREC	
	1724	EB F5	JMP PREPR	
PRBL	1726	BB 40 12	MOV BX,TRANSMITTER-B	Provide CR,LF
	1729	FF D3	CALL TRANSMITTER-B	
	172B	BB 00 40	MOV DI,4000H	get D1 and SI
	172E	A1 02 61	MOV AX,6102	
	1731	89 C6	MOV SI,AX	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	1733	B9 00 00	MOV CX,00	
LOSB	1736	AC	LODSB	Copy the first line to edit. buffer.
	1737	AA	STUSB	
	1738	41	INC.CX	
	1739	3C 0A	CMP AC,0A	check for LF
	173B	74 04	JZ FTR	if zero jump
	173D	3C 1A	CMP AL,1A	check for EOF
	173F	75 F5	JNZ LO SB	
FTR	1741	89 CB	MOV BX,CX	total number of characters in copied line
	1743	A1 0A 60	MOV AX,600A	
	1746	89 66	MOV SI,AX	
	1748	56	PUSH SI	
	1749	81 C6 0001	ADDSI,0100H	get the number
	174D	8B 0C	MOV CX,(SI)	put it in CX
	174F	89 C8	MOV AX,CX	
	1751	A3 0A 61	MOV 610A,AX	
	1754	5E	POPSI	
IMCT	1755	01 1C	ADD(SI),BX	update the pointers accordingly
	1757	41	INC CX	
	1758	A1 00 61	MOV AX,NDLN	
	175B	83 C6 02	ADD SI,02	
	175E	3B C1	CMP AX,CX	check for last line,
	1760	75 F3	JNZ IMCT	if not jump
	1762	01 1C	ADD(SI),BX	
	1764	A1 0C 60	MOV AX,600C	
	1767	89 C6	MOV SI,AX	get SI,DI
	1769	89 C7	MOV DI,AX	
	176B	83 C7 02	ADD DI,02	
	176E	A1 08 61	MOV AX,6108	
	1771	FD	STD	
TFR	1772	A5	MOV SW	Define new starting addresses required lines.
	1773	3B C6	CMP SI,AX	
	1775	75 FB	JNZ TFR	
	1777	A5	MOV SW	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	1778	A1 00 61	MOV AX,6100	increment a count of total number of lines.
	177B	40	INC AX	
	177C	A3 00 61	MOV 6100,AX	
	177F	8B 05	MOV AX, (DI)	
	1781	2B C3	SUB AX,BX	
	1783	89 05	MOV (DI) AX	
	1785	A1 0E 60	MOV AX,600E	get last address of file
	1788	89 C6	MOV SI,AX	set SI
	178A	03 C3	ADD AX,BX	
	178C	89 C7	MOV DI,AX	Set DI with next line
	178E	A1 04 61	MOV AX,6104	
	1791	FD	STD	
VSMO	1792	A4	MOV SB	shift data to down word to accomodate new line
	1793	3B F0	CMP SI,AX	
	1795	75 FB	JNZ VSMO	
	1797	A4	MOVSB	
	1798	FC	CLD	
	1799	BE 00 40	MOV SI,4000	get SI and DI
	179C	A1 04 61	MOV AX,6104	
	179F	29 C3	SUB AX,BX	
	17A1	89 C7	MOV DI,AX	
WHT	17A3	AC	LODSB	
	17A4	AA	STOSB	copy the line present in edit buffer to required position.
	17A5	3C 0A	CMP AL,0A	
	17A7	74 04	JZRT	
	17A9	3C 1A	CMP AL,1A	check for EOF
	17AB	75F6	JNZ WHT	Jump if not have
RT	17AD	C3	RET	Return.

FUNCTION NAME	TRANSFER
INPUT	AL, data stored in the edit buffer
OUTPUT	AL
CALLS	RECEIVER,TRANSMITTER,ADJUST,TRANSREC, TRANSMITTER-B
DESTROYS	All registers
DESCRIPTION	This routine transfers a required line to the specified line position through edit buffer

```
LEVEL   ADDRESS    CONTENTS    MNEMONICS AND      COMMENTS
                               OPERANDS

        0:1800     BB 20 11    MOV BX,TRANSMITTER }Transmit
          1803     FF D3       CALL TRANSMITTER    character
          1805     B8 00 00    MOV  AX,00
          1808     A3 06 61    MOV 106, AX
          180B     BB 00 11    MOV BX RECEIVER
                                                  }Receive next
          180E     FF D3       CALL  RECEIVER      character
          1810     BB 20 10    MOV BX,TRANSMITTER
          1813     FF D3       CALL TRANSMITTER   }Transmit it
          1815     2D 30 00    SUB  AX,30         }subtract 30 to
          1818     89 C2       MOV  DC, AX         convert into
          181A        52       PUSH DX             decimal
          181B     BB 00 11    MOV BX,RECEIVER     Receive next
          181E     FF D3       CALL  RECEIVER     } digit of line
                                                   number
          1820        5A       POP DX
RTCG      1821        3C OD    CMP  AX,OD          check for CR
          1823        75 09    JNZ MCONS           if not jump
          1825        52       PUSH
          1826     BB 10 12    MOV  BX,TRANSREC    give error
          1829        FF D3    CALL  TRANS REC    } message
          182B        5A       POP DX
          182C     EB F3       JMP  RTCG
MCONS     1830     BB 20 11    MOV BX,TRANSMITTER} transmit
          1833     FF D3       CALL TRANSMITTER    character
          1835        5A       POP  DX
          1836     BC 2C       CMP  AX,COMMA       check for ,
          1838        74 23    JZ  AJD             if yes jump
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	18 3A	BB 40 11	MOV BX,ADJUST	Calculate address of line.
	18 3D	FF D3	CALL ADJUST	
	18 3F	A1 08 61	MOV AX,6108	
	18 42	A3 08 60	MOV 600Σ,AX	
	18 45	89 D0	MOV AX,DX	Get the line address and store it
	18 47	A3 02 61	MOV 6102,AX	
	18 4A	A3 04 61	MOV 6104,AX	
	18 4D	BB 00 11	MOV BX,RECEIVER	Receive next character
	18 4E	FF D3	CALL RECEIVER	
	18 51	3C 2C	CMP AX,COMMA	check for ,
	18 53	74 1C	JZ SKIT	if yes jump
	18 55	BB 10 12	MOV BX,TRANSREC	error message
	1858	FF D3	CALL TRANSREZ	
	18 5A	EB F5	JMP CMPA	
	18 5C	BB 40 11	MOV BX,ADJUST	Calculate address of new line
	18 5F	FF D3	CALL ADJUST	
	18 61	89D0	MOV AX,DX	
	18 63	A3 03 61	MOV 6102,AX	Store the address
	18 65	A3 04 61	MOV 6104,AX	
	18 68	A1 08 61	MOV AX,6108	
	18 6B	A3 08 60	MOV 6008,AX	
	18 6E	EB 05	JMP SKIT-1	
SKIT	18 71	BB 20 11	MOV BX,TRANSMITTER	transmit character
	18 74	FF D3	CALL TRANSMITTER	
	18 76	BB 00 11	MOV BX,RECEIVER	Receive next character
	18 79	FF D3	CALL RECEIVER	
	187B	3C 30	CMP AX,30	check for within 30 and 39 H
	187D	7C 04	JL EORR	
	187F	3C 39	CMP AX,39	
	1881	7E 07	JLE KHC	if equal or less jump
RR	1883	BB 10 12	MOV BX,TRANSREC	
	1886	FF D3	CALL TRANSREC	
	1888	EB F1	JMP SEQ	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
KHC	188A	BB 20 11	MOV BX,TRANSMITTER	} Transmit character
	188D	FF D3	CALL TRANSMITTER	
	188F	2D 30 00	SUB AX,30	} subtract 30, to convert to decimal.
	1892	89 C2	MOV DX,AX	
	1894	52	PUSH DX	
	1895	BB 00 11	MOV BX,RECEIVER	} Receive next
	1898	FF D3	CALL RECEIVER	
	189A	5A	POP DX	
CMD	189B	3C 0D	CMP AX,0DH	} check for CR if not equal jump
	189D	75 12	JNZ MZT	
	189F	BB 40 11	MOV BX,ADJUST	} Calculate address of line
	18A2	FF D3	CALL ADJUST	
	18A4	89 D0	MOV AX,DX	
	18A6	A3 04 61	MOV 6104,AX	Store it
	18A9	A1 08 61	MOV AX,6108	} get the line number store it
	18AC	A3 0A 60	MOV 600A,AX	
	18AF	EB 36	JMP NPRBM	jump
NZT	18B1	3C 30	CMPAL,30H	check for within 30 to 39, if not give error message.
	18B3	7C 04	JZ MZT-1	
	18B5	3C 39	CMP AL,39H	
	18B7	7E 07	JLE MZT-2	
MZT-1	18B9	BB 10 12	MOV BX,TRANSFER	
	18BC	FF D3	CALL TRANSREC	
	18BE	EB DB	JMP CMD	
MZT-2	18C0	52	PUSH RX	
	18C1	BB 20 11	MOV BX,TRANSMITTER	} transmit character
	18C4	FF D3	CALL TRANSMITTER	
	18C6	5A	POP DX	
	1867	BB 40 11	MOV BX,ADJUST	} Calculate address of line
	18CA	FF D3	CALL ADJUST	
	18CC	89 D0	MOV AX,DX	} store the address
	18CE	A3 04 61	MOV 6104,AX	
	18D1	A1 08 61	MOV AX,6188	
	18D4	A3 0A 60	MOV 600A,AX	
	18D7	BB 00 11	MOV BX,RECEIVER	} Receive character
	18DA	FF D3	CALL RECEIVER	

```
LEBEL   ADDRESS  CONTENTS   MNEMONICS AND                 COMMENTS
                            OPERANDS

RECTRA  18DC     3C 0D      CMP AX,0D                     check for CR
        18DE     74 07      JZ NPR BM                     if same jump
        18E0     BB 10 12   MOV BX,TRANSREC
        18E3     FF D3      CALL TRANSREC
        18E5     EB F5      JMP RECTRA                    jump
NPRBM   18E7     BF 00 40   MOV DI,4000H                  SI points to4000H
        18EA     B9 00 00   MOV CX,00
        18ED     A1 08 60   MOV AX,6008                   get address of
        18F0        89 C6   MOV SI,AX                     next line,store
        18F2        8B 04   MOV AX,CSI                    it to SI
        18F4        89 C6   MOV SI,AX
LBSB    18F6        AC      LODSB                         transfer the
        18F7        AA      STOSB                      }  data
        18F8        41      INC CX                        increment CX
        18F9        3C 0A   CMP AX,DA
        18FB        74 04   JZ GT
        18FD        3C 1A   CMP AL,1A                     check for EOF
        18FF        75 F5   JNZ LBSB                    } if not jump
GT      1901        89 CB   MOV BX,CX
        1903     A1 D8 60   MOV Ax,6008                 } SI points to
        1906        89 C6   MOV SI,AX                     address of end line
        1908        56      PUSH SI
        1909   81 C6 00 01ADD SI,0100                   } get the line
        190D        8B 0C   MOV CY,(SI)                   number from
        190F        89 C8   MOV AX,CX                     look-up table
        1911     A3 0A 61   MOV 61 0A,AX
        1914        5E      POP SI
        1915     83 C6 02   ADD SI,02
        1918        90      NOP
        1919        90      NOP
        191A        90      NOP
        191B        90      NOP
        191C        90      NOP
        191D        90      NOP
        191E        90      NOP
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
LPN	191F	29 IC	SUB (SI),BX	} Fine new address of line
	1921	83 C6 02	ADD SI,02	
	1924	41	INCCX	
	1925	A1 00 61	MOV AX,6100	
	1928	3B,C1	CMP AX,CX	
	192A	75 F3	JNZ LPN	
	192C	29 1C	SUB(SI),BX	
	192E	A1 0 61	MOV AX,6102	get address of the first
	1931	89 C7	MOV DI,AX	
	1933	03 C3	ADD AX,BX	line store it inD1
	1935	89 C6	MOV SI,AX	SI → address of next line
	1937	FC	CLD	
LBD	1938	AC	LOD SB	Shift data upward
	1939	AA	STOS B	
	193A	3C 1A	CMPLAL,1A	
	193C	75 FA	JNZ LBD	
	193E	A1 08 60	MOV AX,6008	
	1941	89 C7	MOV DI,AX	
	1943	89 C6	MOV SI,AX	
	1945	83 C8 02	ADD SI,02	update all the reouired pointers
	1948	A1 0A 61	MOV AA,610A	
	194B	89 C1	MOV CX,AX	
	194D	FE	CLD	
XTE	194E	AD	LODSW	
	194F	AB	STOSW	
	1950	41	INC CX	
	1951	A1 00 61	MOV AA,61 00	
	1954	3B C1	CMP AX,CX	
	1956	75 F6	JNZ XTE	
OBJ	1958	A1 00 61	MOV AX,61 00	decrement count of total number of lines
	195B	48	DCR AX	
	195C	A3 00 61	MOV 6100,AX	
	195F	A1 0C 60	MOV AX,6000C	
	1962	48	DCR AX	
	1963	48	DCR AX	
	1964	A3 06 60	MOV 6006, AX	
	1967	A1 0A 60	MOV AX,6000.	
	196A	89 C6	MOV SI,AX	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	196C	56	PUSH SI	
	196D	81 C6 00 01	ADD SI,0100	} get the line number
	1971	8B 0C	MOV CX,(SI)	
	1973	89 C8	MOV AX,CX	
	1975	A3 0A 61	MOV 610A,AX	
	1978	5E	POP SI	
INTC	1979	01 1C	ADD (SI),BX	
	197B	41	INC CX	Change the addresses of required lines
	197C	A1 00 61	MOV AX,6100	
	197F	83 C6 02	ADD SI,02	
	1982	3B C1	CMP AX,CX	
	1984	75 F3	JNZ IMTC	
	1986	01 IC	ADD (SI) BX	
	1988	A1 0C 60	MOV AX,600C	
	198B	89 C6	MOV SI,AX	
	198D	89 C7	MOV D1,AX	Up date all the required pointers.
	198F	8B C7 02	ADD D1,02	
	1992	A1 0A 60	MOV AX,600A	
	1995	FD	STD	
RFT	1996	A5	MOVSW	
	1997	3B C6	CMP SI,AX	
	1999	75 FB	JNZ RFT	
	199B	A5	MOVSW	
	199C	A1 00 61	MOV AX,6100	}increment the count of total number of lines
	199F	40	INCAX	
	19A0	A3 00 61	MOV 6100,AX	
	19A3	8B 05	MOV AX (DI)	change addresses of all required
	19A5	2B C3	SUB AX,BX	
	19A7	89 05	MOV(DI),AX	
	19A9	2B C3	SUB AX,BX	
	19AB	89 C6	MOV SI;AX	
	19AD	56	PUSH SI	
	19AE	A1 0A 60	MOV AX,600A	
	19B1	2D 02 00	SUB AX 02	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	19B4	89 C6	MOV SI,AX	transfer the
	19B6	8B C4	MOV AX,SI	data to the required location
	19B8	5E	PUP SI	
	19B9	FD	STD	
	19BA	A4	MOV SB	
	19EB	3B F0	CMP SI,AX	
	19BD	75 FB	JNZ MOS	
MUS	19BF	A4	MOV SB	
	19C0	3B FC	CMP SI,AX	
	19C2	75 FB	JNZ MUS	
	19C4	A4	MOV SB	
	19C5	FC	CLD	
	19C6	BE 00 40	MOV SI,4000	SI→ edit buffer
	19C9	A1 0A 60	MOV AX,60A	DI→ required line position address
	19CC	2D 02 00	SUB AX,02	
	19CF	89 C7	MOV DI,AX	
	19D1	8B 3D	MOV DI,(DI)	
	19D3	AC	LODSB	Shift the edit
	19D4	AA	STOS B	buffer data to requires position
	19D5	3C 0A	CMPAL,0A	Check for LF
	19D7	74 04	JZ TRE	
	19D9	3C 1A	CMP AL,IA	Check for EOF
	19D8	75 F6	JNZ OD TD	if not same just
TRE	19DD	BB 40 12	MOV BX,TRANSMITER-B	Provide CR,LF
	19E0	FF D3	CALL TRANSMITTER -B	
	19E2	C3	RET	Return

FUNCTION MAE	APPEND
INPUT	AL, data buffer
OUTPUT	AL
CALLS	RECEIVER, TRANSMITTER, ADJUST, TRANSREC, TRANSMITTER-B, INSER-A, DELETE-A, REPLACE.
DESTROYS	All registers, required memory locations.
DESCRIPTION	This routine can modify a required line. Along with this routine 4 functions are used to modify the data. Required line is specified to modify. Modifications are done through edit buffer.

```
LEBEL  ADDRESS  CONTENTS   MNEMONICS AND         COMMENTS
                           OPERANDS

 0'    1A10     BB 20 11   MOV BX,TRANSMITTER    Transmit
       1A13        FF D3   CALL TRAN$MITTER      character
       1A15     B8 00 00   MOV  AX,00
       1A18     BB 00 11   MOV BX,RECEIVER       Receive next
       1A1B        FF D3   CALL RECEIVER         character
THN    1A1D        3C 30   CMP  AL,30H
       1A1F        7C 04   JL  RER               check for the
       1A21        3C 39   CMP  AL,39H           character within 30
       1A23        7E 07   JLE RGT               and 39 H if not
RER    1A25     BB 10 12   MOV BX,TRANSREC       Receive next.
       1A28        FF D3   CALL TRANSREC
       1A2A        EB F1   JMP THN

RGT    1A2C     BB 20 11   MOV BX,TRANSMITTER    Transmit
       1A2F        FF D3   CALL TRANSMITTER      character
       1A31     2D 30 00   SUB  AX,30            subtract 30H
       1A 34       89 C2   MOV DX, AX
       1A 36       52      PUSH DX
       1A37     BB 00 11   MOV BX,RECEIVER       Receive
       1A3A        FF D3   CALL RECEIVER         next character
       1A3C        5A      PUSH DX
LFCD   1A3D        3C 0D   CMP  AL,0D            check for CR
       1A3F        74 23   JZ   RRG              if yes, jump
       1A41        3C 30   CMPAL,30
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	1A43	7C 04	JLRR	
	1A45	3C 39	CMP A1,39	check the character between 30 and 39H, if not receive next.
	1A47	7E C9	JLE XTPG	
	1A49	52	PUSH DX	
	1A4A	BB 10 12	MOV BX,TRANSREC	
	1A4D	FF D3	CALL TRANSREC	
	1A4F	5A	POP DX	
	1A5C	EB EB	JMP LFCD	
XTPG	1A52	52	PUSH DX	
	1A53	BB 20 11	MOV BX,TRANSMITTER	transmit character
	1A56	FF D3	CALL TRANSMITTER	
	1A58	5A	POP DX	
WRG	1A59	52	PUSH DX	
	1A5A	BB C0 11	MOV BX,RECEIVER	Receive next
	1A5D	FF D3	CALL RECEIVER	
	1A5F	5A	POP DX	
	1A60	3C 0D	CMPAL,0D	check for 'CR'
	1A62	75 F5	JNZ WRG	if not, jump
	1A64	52	PUSH DX	
	1A65	BB 40 12	MOV BX,TRANSMITTER-B	Provide CR,LF
	1A68	FF D3	CALL TRANSMITTER-B	
	1A7A	5A	POP DX	
	1A6B	BB 40 11	CALL BX,ADJUST	
	1A6E	FF D3	CALL ADJUST	Calculate address of line storage it in SI
	1A70	8D D0	MOV AX,DX	
	1A72	A3 02 61	MOV 6102,AX	
	1A75	89 C6	MOV SI,AX	
	1A77	BB 00 40	MOV D1,4000H	DI →Edit buffer
	1A7A	B9 00 C0	MOV CX,00H	
	1A7D	FC	CLD	Clear DF
LSB	1A7E	AC	LODSB	
	1A7F	AA	STOSB	Store the full line in the edit buffer
	1A80	41	JNCCX	
	1A81	3C 0A	CMP AL,0A	
	1A83	74 04	JZ NXTG	
	1A85	3C 1A	CMPAL, 1A	

```
LEBEL  ADDRESS  CONTENTS     MNEMONICS AND          COMMENTS
                             OPERANDS

       1A87        75 F5     JNZ BLSB
NXTG   1A89     81 C1 FF 3F  ADD CX,4000
       1A8D        89 C8     MOV AX,CX
       1A8F     A3 C0 41     MOV 4100,AX
       1A92     BB C0 40     MOV AX,4000
       1A95     A3 C2 41     MOV 4102,AX
RERV   1A98     BB C0 11     MOV BX,RECEIVER        Receive next
       1A9B        FF D3     CALL RECEIVER          character
       1A9D        3C 0D     CMP AL,0D H            check for 'CR'
       1A9F        75 03     JNZ INST               if not jump
       1AA1     E9 EF 00     JMP HALT               if test, Return

INST   1AA4        3C 49     CMP A1,L               check the I command
       1AA6        75 07     JNZ DEL                if not check next
       1AA8     BB BC 1B     MOV BX,INSERT.A        Insert required
       1AAB        FF D3     CALL INSERT-A          characters.
       1AAd        EB 4F     JMP APT                Jump
DEL    1AAF        3C 44     CMP AL,D               Check for D command
       1AB1        76 07     JNZ REPLS              if not check next
       1A63     BB 10 1C     MOV BX,DELETE-A        Delete reouired
       1AB6        FF D3     CALL DELETE-A          characters
       1AB8        EB 44     JMP APT                jump
REPLS  1ABA        3C 52     CMPAL,R                Check for R command
       1ABC        75 07     JNZ SPAXE              if not check next
       1ABE     BB 55 1C     MOV BX,REPLACE-A       Replace required
       1AC1        FF D3     CALL REPLACE-A         characters
       1AC3        EB 39     JMP APT
SPACE  1AC5        3C 20     CMP A1,20H             check for space
       1AC7        75ZA      JNZ CRGTN
       1AC9     A1 C2 41     MOV AX,4102            SI →address of
       1ACC        89 C6     MOV SI,AX                  line
       1ACE        FC        CLD                    Clear FD
       1ACF        90        NOP
       1AD0     BB FS 11     MOV BX,TRANSMITTER     Transmit
       1AD3        FF D3     CALL TRANSMITTER       character
       1AD5     83 3C 69     CMP CSD 69H            check for special
       1AD8        7C 09     JGE SPLC               if greater jump
       1ADA     A1 C2 41     MOV AR,4102
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	1ADD	40	INC AX	increment the address
	1ADE	A3 02 41	MOV 4102,AX	
	1AE1	EB B5	JMP RCRV	
SPLC	1AE3	90	NOP	
	1AE4	BB F5 11	MOV BX,TRANSMITTER	transmit character
	1AE7	FF D3	CALL TRANSMITTER	
	1AE9	A1 02 41	MOVAX,4102	
	1AEC	40	INC AX	increment the address by two
	1AED	40	INC,AX	
	1AEE	A3 02 41	MOV 4102,AX	
	1AF1	EB A5	JMP RCRV	
CRGTN	1AF3	3C 0D	CMP AL,0D	check for CR
	1AF5	74 07	JZ APT	if yes, jump
	1AF7	BB 00 11	MOV BX,RECEIVER	
	1AFA	FF D3	CALL RECEIVER	Receive next
	1AFC	EB A6	JMP INST	
APT	1AFE	A1 00 41	MOV AX,4100	
	1B01	89 C1	MOV CX,AX	
	1803	A1 04 41	MOV AX,4104	Check the pointer with subroutine pointer,jump if less
	1B06	3B C1	CMP AX,CX	
	1B08	7E 3B	JLE DELRPL	
	1B0A	2B C1	SUB AX,CX	find difference
	1B0C	89 C3	MOV BX,AX	
	1B0E	A1 08 61	MOV AX,6108	find line number
	1B11	89 C6	MOV SI,AX	
	1B13	56	PUSH SI	
	1B14	81 C6,0001	ADD SI,0100	
	1B18	8B 0C	MOV CX,(SI)	get address in CX
	1B1A	89 C8	MOV AX,CX	
	1B1C	A3 0A 61	MOV 610A,AX	
	1B1F	5E	POP SI	
	1B20	83 C6 02	ADD SI,02	increment SI and its content
DAO	1B23	01 1C	ADD (SI),BX	

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	1B25	41	INC CX	
	1B26	A1 00 61	MOV AX,6100	Check for the last line
	1B29	83 C6 02	ADD SI,02	
	1B2C	3B C1	CMP AX,CX	
	1B2E	75 F3	JNZ DAD	
	1B30	A1 0E 60	MOV AX,(600E)	get last address
	1B33	89 C6	MOV Si,AX	store it in SI
	1B35	03 C3	ADD AX,BX	Update it
	1B37	89 C7	MOV DI,AX	get address for DI
	1B39	A1 02 61	MOV AX,(6102)	
	1B3C	FD	STD	
SHIFT	1B3D	A4	MOVESB	transfer the required data.
	1B3E	3B F0	CMP SI,AX	
	1B 40	75 FB	JNZ SHIFT	
	1B42	A4	MOVSB	
	1B43	EB 38	JMP PAST	
DELRPL	1B45	3B C1	CMP AX,CX	compare AX,CX
	1B47	74 34	JZ PAST	if same jump
	1B49	26 C1	SUB CX,AX	if not, get difference
	104B	89 C6	MOV BX,CX	put it in BX
	1B4D	A1 08 61	MOVAX6108	
	1B50	89 C6	MOV SI,AX	
	1B52	56	PUSH SI	get line number in CX
	1B53	81 C6 0001	ADDSI,0Y00	
	1B57	8B 0C	MOV CX,(SI)	
	1B59	5E	POPSI	
	1B5A	83 C6 02	ADD SI,02	
	1B5D	89 F2	MOV DY,SI	
LPL	1B5F	29 1C	SUB (SI),BX	get starting of line in SI
	1B61	83 C6 02	ADD SI,02	
	1B64	41	INC CX	increment it.
	1B65	A1 00 61	MOV AX,(6100)	
	1B68	3B C A	CMP AX,CX	Check for last line
	1B6A	75 F3	JNZ LPL	if not, jump

```
LEBEL   ADDRESS   CONTENTS   MNEMONICS AND          COMMENTS
                             OPERANDS

        1B6C      89 D6      MOVSI,DX
        1B6E      8B 04      MOV  AX,(SI)           get SI  and DI
        1B70      89 C7      MOV  DI,AX
        1B72      03 C3      ADD  AX,BX
        1B74      89 C6      MOV  SI,AX
        1B76      FC         CLD
LPP     1B77      AC         LODSB                  shift the data
        1B78      AA         STOSB                  to required location.
        1B79      3C 1A      CMP  AL   A
        1B7B      75 FA      JNZ LPP
PAST    1B7D      FC         CLD                    clear DF
        1B7E   BE 00 40      MOV SI,4000H
        1B81   A1 02 61      MOV  AX,(6102)
        1B84      89 C7      MOV  DI,AX             shift data
LSBB    1B86      AC         LODSB                  from edit
        1B87      AA         STOSB
        1B88      3C 0A      CMP  AL,0A
        1B8A      74 04      JZ PREHLT
        1B8C      3C 1A      CMPAL,1A               check for EOF
        1B8E      75 F6      JNZ LSBB
PREHLT  1B90   E9 05 FF      JMP.RCRV
HALT    1B93   BB 40 12      MOV  BX,TRANSMITTER-B   Provide
                                                     CR,LF
        1B96      FF D3      CALL TRANSMITTER-B
        1B98      C3            RET                  Return
```

FUNCTION NAME	INSERT-A
INPUT	Al, data stored in edit buffer
OUTPUT	AL
CALLS	RECEIVER,TRANSMITTER
DESTROYS	All registers
DESCRIPTION	This routine can insert a set of characters in the line, which is available in the edit buffer.

```
LEBEL  ADDRESS  CONTENTS   MNEMONICS AND         COMMENTS
                           OPERANDS

   C:  1BB0     BB 20 11   MOV  BX,TRANSMITTER   transmit
       1BB3     FF D3      CALL TRANSMITTER      character
       1BB5     B9 00 00   MOV  CX,00
       1BB8     A1 00 41   MOV  AX,4100
       1BBB     A3 04 41   MOV  41 04,AX         initialize
       1BBE     A1 02 41   MOV  AX,4102          pointers
       1BC1     A3 06 41   MOV  4106,AX

CTLQ   1BC4     BB 00 11   MOV  BX,RECEIVER      Receive
       1BC7        FF D3   CALL RECEIVER         next character
       1BC9        3C 18   CMP  AL,CLQ           Check for EOC
       1BCB        74 32   JZ END                if yes, jump
       1BCD     BB 20 11   MOV  BX,TRANSMITTER   transmit
       1BD0        FF D3   CALL TRANSMITTER      character
       1BD2        89 C2   MOV  DX,AX
       1BD4     A1 04 41   MOV  AX,4104          get SI
       1BD7        89 C6   MOV  SI,AX
       1BD9        40      INC  AX
       1BDA        89 C7   MOV  DI,AX            get DI
       1BDC     A1 06 41   MOV  AX,4106
       1BDF        FD      STD                   shift the
USP    1BE0        A4      MOVSB                 data after
                                                 inserting.
       1BE1        3B F0   CMP SI,AX
       1BE3        75 FB   JNZ USD
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	1BE5	A4	MOV SB	
	1BE6	A1 06 41	MOV AX,4106	
	1BE9	89 C7	MOV DI,AX	
	1BEB	89 D0	MOV AX,DX	store the character, increment count.
	1BED	AA	STOSB	
	1BEE	41	INC CX	
	1BEF	A1 06 41	MOV AX,41 06	
	1BF2	40	INC AX	
	1BF3	A3 06 41	MOV 4106,AX	update required pointers.
	1BF6	A1 04 41	MOV AX,4104	
	1BF9	40	INC AX	
	1BFA	A3 04 41	MOV 4104,AX	
	1BFD	EB C5	JMP CTLQ	Receive next character
END	1BFF	C3	RET	Return.

FUNCTION NAME   DELETE-A

INPUT   Al, data stored in the edit buffer

OUTPUT   AL

CALLS   RECEIVER, TRANSMITTER

DESTROYS   All registers, required line in the data buffer

DESCRIPTION   This routine can delete a set of required characters from the specified line through edit buffer.

```
LEBEL  ADDRESS  CONTENTS   MNEMONICS AND          COMMENTS
                           OPERANDS

   0:  1C10     BB 20 11   MOV BX,TRANSMITTER     transmit
       1C13        FF D3   CALL TRANSMITTER       character
       1C15     B9 00 00   MOV  AX,00
       1C18     A1 00 41   MOV  AX,4100           initialize
       1C1B     A3 04 41   MOV  4104,AX           pointers
       1C1E     A1 02 41   MOV  AX,4102
       1C21     A3 06 41   MOV  4106,AX

RECR   1C24     BB 00 11   MOV  BX,RECEIVER       Receive
       1C27        FF D3   CALL RECEIVER          character
       1C29        3C 18   CMP  AL,C/X            check for EOC
       1C2B        74 27   JZ STOP                if yes, stop
       1C2D        3C 20   CMPAL,20               check for space
       1C2F        75 F3   JNZ RECR               if not, jump
       1C31        B0 44   MOV  AL,44
       1C33     BB 20 11   MOV  BX,TRANSMITTER    transmit
       1C36        FF D3   CALL TRANSMITTER       character
       1C38     A1 06 41   MOV  AX,4106
       1C3B        89 C7   MOV  DI,AX             set SI
       1C3D        40      INC  AX                and DI
       1CBE        89 C6   MOVSI,AX
       1C40        FC      CLD                    shift
UDS    1C41        AC      LODSB                  data
       1C42        AA      STOSB
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	1C43	3C 0A	CMPAL,0A	check for LF
	1C45	7404	JZ PTR	if yes, jump
	1C47	3C 1A	CMPAL,1A	check for EOF
	1C49	75 F6	JNZ UDS	if not jump
PTR	1C4B	A1 04 41	MOV AX,4104	update pointer
	1C4E	48	DEC AX	
	1C4F	A3 04 41	MOV 4104,AX	
	1C52	EB D0	JMP RECR	Receive next character
STOP	1C54	CB	RET	Return

FUNCTION NAME   REPLACE-A

INPUT   AL, data stored in the edit buffer

OUTPUT   AL

CALLS   RECEIVER, TRANSMITTER

DESTROYS   ALL registers, data in the edit buffer.

DESCRIPTION   This routine can replace a set of characters from the specified line through edit buffer.

```
LEBEL  ADDRESS  CONTENTS   MNEMONICS AND         COMMENTS
                           OPERANDS

   0:  1C55     BB 20 11   MOV BX,TRANSMITTER    transmit
       1C58        FF DB   CALL TRANSMITTER      character
       1C5A     BB 00 00   MOV  CX,00            initialize
       1C5D     A1 00 41   MOV  AX,4100          pointers
       1C60     A3 04 41   MOV  4104,AX
       1C63     A1 02 41   MOV  AX,4102
       1C66     A3 06 41   MOV  4106,AX
RPLC   1C69     BB 00 11   MOV  BX,RECEIVER
       1C6C        FF D3   CALL RECEIVER
       1C6E     BB 20 11   MOV  BX,TRANSMITTER
       1C71        FF D3   CALL TRANSMITTER
       1C73        3C 18   CMP  AL,C/X            check for EOC
       1C75        74 19   JZ,END                 if yes, stop.
       1C77        50      PUSH AX
       1C78     A1 06 41   MOV  AX,4106
       1C7B        89 C7   MOV  DI,AX             set DI
       1C7D        58      POP  AX
       1C7E        FC      CLD                    clear DF
       1C7F        AA      STOSB                  store character
       1C80     A1 06 41   MOV  AX,4106
       1C83        40      INC  AX                update pointers
       1C84     A3 06 41   MOV  4106,AX
       1C87     A1 02 41   MOV  AX,4102
       1C8A        40      IN,AX
       1C8B     A3 02 41   MOV  102,AX
       1C8E        ED D9   JMP  RPLC              Receive next
                                                  charater
END    1C90        C3      RET                    Return
```

FUNCTION NAME INSERT

INPUT AL

OUTPUT AL

CALLS RECEIVER,TRANSMITTER,ADJUST,TRANSMITTER-B

DESTROYS ALL REGISTERS

DESCRIPTION This functional routine stores the required (tuped) line to the specified location in the file through edit buffer.

```
LEBEL   ADDRESS   CONTENTS   MNEMONICS AND          COMMENTS
                             OPERANDS

   0•   1CB0      BB 20 11   MOV BX,TRANSMITTER     Transmit the
                                                    code for command
        1CB3      FF D3      CALL TRANSMITTER
        1CB5      BB 00 11   MOV BX,RECEIVER        Receive first
        1CB8      FF D3      CALL RECEIVER          line number
        1CBA      BB 20 11   MOV BX,TRANSMITTER     Transmit it
        10BD      FF D3      CALL TRANSMITTER
        1CBF      2D 30 00   SUB AX,30              Subtract 30 to
        1CC2         89 C2   MOV DX,AX              convert into
        1CC4         52      PUSH DX                decimal put in DX
        1CC5      BB 00 11   MOV BX,RECEIVER        Receive next
        1CC8      FF D3      CALL RECEIVER          character
        1CCA         5A      POP DX
        1CCB      3C OD      CMP AX,OD              Check for CR
        1CCD         75 OC   JNZ NXTDGT             if not,jump
        1CCF      BB 40 11   MOV BX,ADJUST          if yes calculate
        1CD2      FF D3      CALL ADJUST            address of a line
        10D4         89 DO   MOV AX,DX
        16D6      A3 02 61   MOV 6102,AX            Store it in 6102H
        1CD9      EB 21      JMP OKYI               Jump for further
 MXTDGT1CDB          52      PUSH DX                operation
        1CDC      BB 20 11   MOV BX,TRANSMITTER     save DX,
        1CDF      FF D3      CALL TRANSMITTER       Transmit character
        1CE1         5A      POP DX
        1CE2      BB 40 11   Mov BX,ADJUST          find address of the
        1CE5      FF D3      CALL ADJUST            line having two digits
                                                    numbers
        1CE7      89 DO      MOV AX,DX              save it
        1CE9      A3 02 61   MOV 6102,AX
        1CEC      BB 00 11   MOV BX,RECEIVER        Receive next
        1CEF      FF D3      CALL RECEIVER          character
```

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
CRGT	1CF1	3C OD	CMP AX,OD	check for CR
	1CF3	74 07	JZ OKY1	if yes jump
	1CF5	BB 10 12	MOV BX,TRANSREC	if not give
	1CF8	FF D3	CALL TRANSREC	
	1CFA	EB F5	JMP CRGT	Jump to CRGT
OKNI	ICFC	BB 40 12	MOV BX,TRANSMITTER-B	Provide CR,LF
	1CFF	FF D3	CALL TRANSMITTER-B	
	1D01	B9 00 00	MOV CX,00	
	1D04	BF 00 40	MOV DI,4000	DI pointing to edit buffer
EDBPR	1D07	BB 00 11	MOV BX,RECEIVER	
	1D0A	FF D3	CALL RECEIVER	Store the received character in it.
	1D0C	AA	STOSB	
	1D0D	41	INC CX	
	1D0E	BB 20 11	MOV BX,TRANSMITTER	transmit it back
	1D11	FF D3	CALL TRANSMITTER	
	1D13	3C 0A	CMP AX,OD	check for CR
	1D15	75 F0	JNZ EDBFR	if not jump
	1D17	89 CB	MOV BX,CX	
	1D19	A1 08 61	MOV AX,(6106)	get the address of Ist line in SI
	1D16	89 C6	MOV SI,AX	
	1D1E	56	PUSH SI	
	1D1F	81 C6 00 01	ADD SI,0100H	add 0100 to
	1D23	8B 0C	MOV CX,(SI)	get its number
	1D25	89 C8	MOV AX,CX	put it in 610A
	1D27	A3 0A 61	MOV 61,0A,AX	
	1D2A	5E	POP S1	
ICMT	1D2B	01 1C	ADD(SI),BX	increment SI to point to next line
	1D2D	41	INC CX	
	1D2E	A1 00 61	MOV AX,NOLN	check for the last line in the file
	1D31	83 C6 02	ADD SI,02	
	1D34	3B C1	CMP AX,CX	
	1D36	75 F3	JNZ ICMT	if not repeat
	1D38	01 1C	ADD(SI),BX	if yes
	1D3A	A1 0C 60	MOV AX,60 0C	using SI and DI transfer
	1D3D	89 C6	MOV SI AX	
	1D3F	89 C7	MOV D1,AX	the data to required location.

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	1D41	83 C7 02	ADD DI,02	
	1D44	A1 C8 61	MOV AX,61 C8	
	1D47	FD	STD	
MOV	1D48	A5	MOV SW	
	1D49	3B C6	CMP SI,AX	Check for last line
	1D4B	75 FB	JNZ MOV	if not repeat
	1D4D	A5	MOV SW	
	1D4E	A1 C0 61	MOV AX,61,C0	increment the
	1D51	40	INC AX	total number of
	1D52	A3 C0 61	MOV 61 C0 AX	line by one
	1D55	8B 05	MOV AX,(DI)	
	1D57	2B C3	SUB AX,BX	get the address
	1D59	89 05	MOV (DI),AX	of required line in DI
	1D5B	90	NOP	
	1D5C	A1 0E 60	MOV AX,60,0E	
	1D5F	89 C6	MOV SI,AX	transfer the data to
	1D61	03 C3	ADD AX,BX	required location
	1D63	89 C7	MOV DI,AX	
	1D65	A1 C2 61	MOV AX,6102	
	1D68	FD	STD	
MOV	1D69	A4	MOV SB	
	1D6A	3B F0	CMP SI,AX	Check for last line ,
	1D6C	75 FB	JNZ MOV	if not repeat if yes
	1D6E	A4	MOV SB	
	1D6F	FC	CLD	
	1D70	BE 00 40	MOV SI,4000H	
	1D73	A1 C2 61	MOV AX,6102	
	1D76	89 C7	MOV DI,AX	
LDST	1D78	AC	LODSB	
	1D79	AA	STOSB	
	1D7A	3C 0A	CMP AX,CR	
	1D7C	75 FA	JNZ LDST	
	1D7E	BB 40 12	MOV BX,	Transmitter-B
	1D81	FD D3	CALL	Transmitter -B
	1D83	C3	RET	

LEBEL	ADDRESS	CONTENTS	MNEMCNICS AND CPERANDS	COMMENTS
	1D41	83 C7 02	ADD DI,02	
	1D44	A1 C8 61	MOV AX,61 C8	
	1D47	FD	STD	
MOV	1D48	A5	MOV SW	
	1D49	3B C6	CMP SI,AX	Check for last line
	1D4B	75 FB	JNZ MOV	if not repeat
	1D4D	A5	MOV SW	
	1D4E	A1 CC 61	MOV AX,61,CC	increment the total number of line by one
	1D51	40	INC AX	
	1D52	A3 CC 61	MOV 61 CC AX	
	1D55	8B C5	MOV AX,(DI)	
	1D57	2B C3	SUB AX,BX	get the address of required line in DI
	1D59	89 C5	MOV (DI),AX	
	1D5B	9C	NOP	
	1D5C	A1 OE 60	MOV AX,60,OE	
	1D5F	89 C6	MOV SI,AX	transfer the data to required location
	1D61	03 C3	ADD AX,BX	
	1D63	89 C7	MOV DI,AX	
	1D65	A1 C2 61	MOV AX,6102	
	1D68	FD	STD	
MOV	1D69	A4	MOV SB	
	1D6A	3B FC	CMP SI,AX	Check for last line
	1D6C	75 FB	JNZ MOV	if not repeat if yes
	1D6E	A4	MOV SB	
	1D6F	FC	CLD	
	1D70	BE 00 40	MOV SI,400CH	
	1D73	A1 C2 61	MOV AX,6102	
	1D76	89 C7	MOV DI,AX	
LDST	1D78	AC	LODSB	
	1D79	AA	STOSB	
	1D7A	3C OA	CMP AX,CR	
	1D7C	75 FA	JNZ LDST	
	1D7E	BB 40 12	MOV BX,	Transmitter-B
	1D81	FD D3	CALL	Transmitter -B
	1D83	C3	RET	

APPENDIX- C

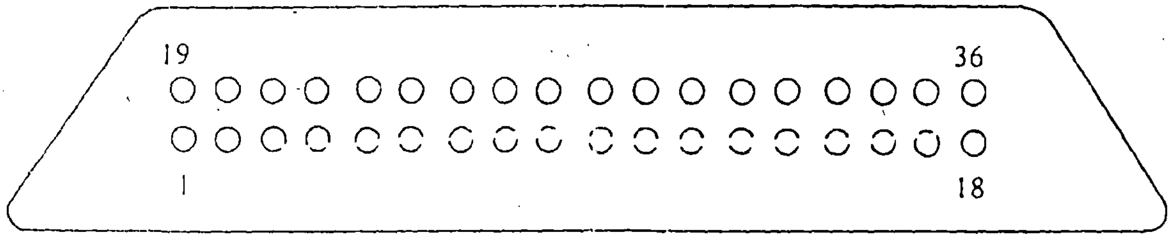
C-1 PIN CONNECTION DIAGRAM OF PRINTER INTERFACE.

C-2 PRINTER CONTROL COMMANE CODES

C-3 SOFTWARE FOR PRINTER INTERFACE

APPENDIX - C1

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	$\overline{\text{DATA STB}}$	19	TWISTED PAIR GND PIN 1
2	DATA 1	20	2
3	2	21	3
4	3	22	4
5	4	23	5
6	5	24	6
7	6	25	7
8	7	26	8
9	DATA 8	27	9
10	$\overline{\text{ACK}}$	28	10
11	INPUT-BUSY	29	11
12	PE	30	TWISTED PAIR GND 31
13	SELECT	31	$\overline{\text{INPUT-PRIME}}$
14	0V	32	$\overline{\text{FAULT}}$
15	NC	33	0V
16	0V	34	NC
17	CHASSIS GND	35	NC
18	+5V DC	36	INPUT-BUSY



APPENDIX - C2

Control Codes

(1) CR (0D) H

(a) This is a print command code.

(b) Line feed can be selected with DIP SW 1-8.

(c) This code will be ignored when no data is stored in the Buffer. However LF will be performed when SW 1-8 is ON.

(2) LF (0A) H

(a) This is a Line feed code.

(b) It can double as a print command code when SW 1-7 is ON. When SW 1-7 is OFF, this code will be ignored until CR.

(c) When BOTTOM is set in the VFU, a one line feed from the last line causes a skip to the next TOF position.

(3) VT (0B) H

(a) This is a multiple line feed code. Automatic line feed to the vertical tab position set in channel 2 of the VFU.

(b) It can double as a print command code when SW 1-7 is ON. When SW 1-7 is OFF, this code will be ignored until CR.

(c) If no vertical tab has been set in channel 2 of the VFU, automatic line feed to the next TOF position is performed.

(4) FF (0C) H

(a) This is a multiple line feed code. Automatic line feed to the TOF position set in channel 1 of the VFU. (The form will NOT stop at the VFU BOTTOM.)

(b) It can double as a print command code with SW 1-7 ON. With SW 1-7 OFF it will be ignored until CR.

(5) CAN (18) H

(a) This code cancells 1 line of data received prior to this code.

(b) All control codes in effect, prior to this code, remain valid. The last mode received prior to this code is maintained.

(6) SO (0E) H

(a) This is the Double Width Character code.

(b) This code will be valid until reception of the SI code.

(c) It also selects the KANA characters when the JA7 bit mode is used. (SW 1-1 ~ 3 OFF, SW 2-[illegible] ON)

(7) SI (0F) H

(a) This code clears the above SO code.

(b) When set to the JA7 bit mode, it clears the KATA KANA character area, and selects the Alphanumeric & Symbol characters.

(8) DC1 (11) H

(a) This code sets the Printer to the SELECT state.

(b) It can be made ineffective by setting SW 1–5 ON.

(9) DC3 (13) H

(a) This code sets the Printer to the DESELECT state.

(b) It can be made ineffective by setting SW 1–5 ON.

(10) HT (09) H

(a) This code moves the Carriage to the nearest preset Horizontal Tab position. It is ignored when no tabs are set.

(11) DC2 (12) H

(a) For enlarged characters in the JA 7 bit mode.

(b) It is valid until reception of the DC4 code.

(c) It is ignored in modes other than the JA 7 bit mode.

(12) DC4 (14) H

(a) This clears the above DC2 code.

(b) It is ignored in modes other than the JA 7 bit mode.

(13) US (1F) H

(a) This is a start commanu ior Vertical Tab as preset in the VHU format. Combined with the following 1 byte, channel specification can be performed.
This Printer uses channels 1 and 2 ONLY.

(b) When this code is received, VT setting is performed according to the channel format specified by the following byte. The form will be fed to the next TOF position if channel setting is 1 line below the current position.
Also, depending on the content of the 1 byte following this code, it is possible to just feed the form N/lines. (0 ≤ N ≤ 15)

(c) No feed when N = 0.

Example:	7	6	5	4	3	2	1	0	←—— bit No.
	0	0	0	1	1	1	1	1	←—— US code
	x	x	x	0	0	C	C	C	←—— at Channel VT
Commands after the US code	x	x	x	1	N	N	N	N	←—— at N/line feed, x may be either 0 or 1

Note: When CCC = 001 the form is fed to the next BOTTOM or TOF position.
When CCC = 000 this code is ignored. This code is ineffective other than when 1 ≤ C ≤ 6. When this code is ignored, the US code is cancelled, too.

(d) This code doubles as a print command when DIP SW 1–7 is ON. This code is ignored after print data is received, and until the next CR code, when DIP SW 1–7 is OFF.

(14) DEL (7F) H

(a) This code is not effective.

APPENDIX -C3

FUNCTION NAME	MPRINT
INPUT	AL, data stored in the data file
OUTPUT	AL
CALLS	SEND, PRTSPL,GRAPH, PRTCHR,FTPR,DELAY
DESTROYS	All registers
DESCRIPTION	This main routine is useful for printing the whole data file present in the micro-computer memory. The routine can print both English, Devnagari script.

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
0:	0900	BA FE FF	MOV DX,FFFE	Control Register address.
	0903	BO 81	MOV AL,81H	control word
	0905	EE	OUT DX,AL	
	0906	BA FC FF	MOV DX,FFC	
	0909	BO 90	MOV AL,90H	INIT, STORE Hingh
	090B	EE	OUT DX,AL	
	090C	BO 80	MOV AL,80H	INIT LOW,STROBE HIGH
	090E	EE	OUT DX,AL	
	090F	B9 09 00	MOV CX,0009	Load counter
LP	0912	90	XCHG AL,AL	Do NOP operation
	0913	E2 FD	LOOP LP	
	0915	B0 90	MOV AL,90H	INIT low, STROBE High
	0917	EE	OUT DX,AL	
	0918	AO FO OF	MOV AL,OFFO	Get the pointer
	091B	3C 45	CMP AL,45	Check the language Code
	091D	75 03	JNZ PTR	
	091F	E9 CC 00	JMP L1	jump
PTR1	0922	BE 00 20	MOV SI,2000	Print to datafile
LC1	0925	BF 00 75	MOV DI,7500	buffer of special character
	0928	FC	CLD	clear DF

LEBERL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
L∅	0929	AC	LODS B	load AL
	092A	3C 0A	CMP AL,0A	check for LF
	092C	75 03	JNZ PIR2	if not, jump
	092E	E9 AF 00	JMP L2	
PTR 2	0931	3C 0D	CMPAL,0D	check for CR
	0933	75 03	JNZ PIR3	if not, jump
	0935	E9 A8 00	JMP L2	
PTR 3	0938	3C 1A	CMP AL, 1A	check for EOF
	093A	75 03	JNZ PTR4	if not, jump
	093C	E9 A1 00	JMPL2	
PTR 4	093F	3C 69	CMPAL, 69	check for 69 H
	0941	7C 03	JNGE PTR5	if less, jump
	0943	E9 95 00	JMP L4	
PTR 5	0946	3C 21	CMP AL, 21	check for 21H
	0948	75 03	JNZ PTR 6	if not, jump
	094A	E9 83 00	JMP L5	
PTR 6	094D	3C 22	CMP AL, 22	check for 22 H
	094F	75 03	JNZ PTR7	if not, jump
	0951	E9 7C 00	JMP L5	
PTR 7	0954	3C 27	CMP AL,27	check for 27H
	0956	7C 07	JL L51	if less ,jump
	0958	3C 29	CMP AL, 29	check for 29H
	095A	7F 03	JNLE L51	if grater jump
	095C	E9 71 00	JMP L5	
L51	095F	3C 30	CMP AL,30	Check for 30
	0961	7C 04	JL L52	if less, jump
	0963	3C 3A	CMPAL,3A	check for 3A
	0965	7E 69	JLE L5	if less, jump
L52	0967	3C 3F	CMP AL, 3F	check for
	0969	74 65	JE 15	if yes jump
	096B	3C 41	CMPAL, 41	check for A
	096D	7C 13	JL L6	if less, jump
	096F	2C 41	SUB Al, 41	act difference
	0971	BB 09	MOV BI, 09	load B
	0973	F6 E3	MUL AL, BL	multiply

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	C975	15 48 70	ADD AX,7048	Add result to address
L8A	C978	BB A0 0A	MOV BX, PRTCHR	Print the character
	C97B	FF D3	CALL PRTCHR	
	C97D	B0 20	MOV AL, 20	
	C97F	AA	STOSB	store space code
	C980	EB A7	JMP L0	
L6	C982	3C C8	CMP AL, C8	check for BS
	C984	7D 09	JGE L7	if greates jump
	C986	B3 09	MOVBL, 09	load B
	C988	F6 E3	MULAL, BL	multiply
	098A	15 10 6F	ADDAX,6FIO	add toget address
	098D	EB E9	JMP L8A	
L7	C98F	3C C9	CMPAL, 09	check for C9
	C991	75 05	JNE L8	if not equal jump
	C993	A1 58 6F	MOVAX, 6F58	get address
	C996	EB E0	JMPL8A	
L8	C998	3C CC	CMPAL, CC	check for CC
	C99A	7F 0B	JG L9	if greates jump
	C99C	2C 0B	SUBAL, 0B	get difference
	C99E	B3 C9	MOV BL, 09	load B
	C9A0	F6 E3	MUL AL, BL	multiply
	C9A2	15 61 6F	ADD AX, 6F61	get required address
	C9A5	EB D1	JMPL 8A	
L9	C9A7	3C 14	CMPAL, 14	check for 14
	C9A9	7F 0B	JG L 10	if greater jump
	C9AB	2C 0E	SUB AL, 0E	get difference
	C9AD	BB 09	MOV BL, 09	load B
	C9AF	F6 E3	MUL AL, BL	multiply
	C9B1	15 73 6F	ADDAX, 6F 73	get address
	C9B4	EB C2	JMP L8A	
L10	C9B6	3C 27	CMPAL, 27	check for 27 H

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	09B8	7F 0B	JG L11	if greates jump
	09BA	2C 23	SUB AL, 23	get difference
	09BC	B3 09	MCV BL, 09	load B
	09BE	F6 E3	MUL AL, BL	multiply
	09C0	15 B2 6F	ADDAX, 6FB2	get address
	09C3	EB B3	JMP L8A	
L11	09C5	2C ED	SUBAL, 3D	get difference
	09C7	B3 09	MOVBL, 09	load
	09C9	F6 E3	MVL AL, BL	multiply
	09CB	15 DF 6F	ADD AX, 6FDF	get address
	09CE	EB A8	JMPL 8A	
L 5	09D0	BB 20 0A	MOVBX, SEND	print required
	09D3	FF D3	CALL SEND	character
	09D5	B0 20	MOVAL, 20	
	09D7	AA	STO5B	store space code.
	09D8	E9 4E FF	JMPLO	
L 4	09DB	4F	DEC DI	Special character
	09DC	AA	STOSB	store in privious
	09DD	E9 49 FF	JMPLO	location
	09E0	50	PUSH	
L 2	09E1	BB 30 0A	MOVBX, PRTSPL	print special
	09E4	FF D3	CALL PRTSPL	character
	09E6	58	POPA	
	09E7	3C 1A	CMPAL,1A	check for EOF
	09E9	74 13	JE L3	if yes jump
	09EB	E9 37 FF	JMP L01	
L1	09EE	BE 00 20	MOV SI, 2000	get data file
	09F1	FC	CLO	
L35	09F2	AC	LODSB	get character
	09F3	3C 1A	CMP AL, 1A	check for EOF
	09F5	74 07	JE L3	if yes jump
	09F7	BB 20 0A	MOV BX, SEND	print the
	09FA	FF D3	CALL SEND	character
	09FC	EB F4	JMP L 35	
L3	09FE	C3	RET	

FUNCTION NAME	:	SEND
INPUT	:	AL
OUTPUT	:	AL
CALLS	:	FTPR, DELAY
DESTROYS	:	AX, BX, DX
DESCRIPTION	:	This routine transmits the character to the printer and also produces delay between sending of two characters.

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
0:	OA20	BB DO OA	MOV BX, FTPR	output the
	OA23	FF D3	CALL FTPR	character to printer
	OA25	BB O OA	MOV BX, DELAY	provide delay
	OA28	FF D3	CALL DELAY	betn the character
	OA2A	C3	RET	

FUNCTION NAME : PRTSPL

INPUT : AL, data stored in the script look-up table

OUTPUT : AL

CALLS : SEND, PRTCHR

DESTROYS : AX, BX, DX, SI, DI

DESCRIPTION : This routine helps the main routine to print a set of special characters. These Special Characters in Devanagar Script are matras.

```
LEBEL    ADDRESS    CONTENTS    MNEMONICS AND    COMMENTS
                                  OPERANDS

     0:  0A30       56          PUSH SI          save SI
         0A31       BE 00 75    MOV SI, 7500     get SI address of
                                                 special character
         0A34       BO OD       MOV AL, OD
         0A36       BB 20 0A    MOV BX,SEND      print character
         0A39       FF D3       CALL SEND
L 28     0A3B       3B FE       CMP SI,DI        compare DI SI
         0A3D       74 1C       JE L12           if same jump
         0A3F       AC          LODSB            load accmmulabs
         0A40       3C 20       CMP AL,20        check for 20H
         0A42       74 10       JE L13           if yes, jump
         0A44       2C 69       SUB AL,69
         0A46       B3 09       MOV BL, 09       load B.
         0A48       F6 E3       MUL AL, BL       multiply
         0A4A       15 BO 71    ADD AX, 71BO     get correct address
         0A4D       BB A0 0A    MOV BX,PRTCHR    output character
         0A50       FF D3       CALL PRTCHR
         0A52       EB E7       JMP L28
L 13     0A54       BB 20 0A    MOV BX, SEND     print character
         0A57       FF D3       CALL SEND
         0A59       EB E0       JMP L28
L12      0A5B       BO OD       MOV AL,OD        provide CR
         0A5D       BB2C 0A     MOV BX,SEND      Output it.
         0A60       FF D3       CALL SEND
         0A62       BO 0A       MOV AL, 0A       provide 'LF'
         0A64       BB 20 0A    MOVBX, SEND      output it
         0A67       FF D3       CALL SEND
         0A69       5E          POP SI
         0A6A       46          INC SI
         0A6B       C3          RET
```

FUNCTION NAME : GRAPH

INPUT : AL, data for control command present in the memory locatioh.

OUTPUT : AL

CALLS : FTPR, DELAY

DESTROYS : AL, BX, DX, SI, DI

DESCRIPTION : This routine reads 6 bytes of control command stored in memory location.for each letter of deuanagari this routine outputs 6 bytes.

```
LEBEL     ADDRESS    CONTENTS     MNEMONICS AND      COMMENTS
                                     OPERANDS

      0:  CA80          51        PUSH C             Save Registers

          0A81          56        PUSH SI
          CA82       BE 92 72     MOVSI,7292         SI points to look
                                                     up table
          CA85          B1 06     MOVCX, 0006
L 1       CA87          AC        LODSB              Load accumulates
          0A88       BE D0 CA     MOVDX,FTPR         print it
          CA8B          FF D3     CALL  FTPR
          0A8D       BB F0 CA     MOVDX, DELAY       wait before
                                                     next character
          0A90          FF D3     CALL DELAY
          CA92          E2 F3     LOOPL1             take next character
          CA94          5E        POPSI
          0A95          59        POPC
          0A96          C3        RET
```

FUNCTION NAME : PRTCHR

INPUT : AL, data present in the memory location for control commahd.

OUTPUT : AL

CALLS : GRAPH, FTPR, DELAY

DESTROYS : AX, BX, CX

DESCRIPTION : For printing deuanagari script, this routine is used. This routine sends required character to the printer.

```
LEBEL   ADDRESS     CONTENTS     MNEMONICS AND      COMMENTS
                                  OPERANDS

   0,   CAA0           56        PUSH SI
        CAA1           51        PUSH C
        CAA2           89 C6     MOW SI,AX          Get SI
        CAA4           B1 09     MOV CS, 0C09       load Cx
        CAA6        BB 8C OA     MOV BX, GRAPH      output 6 byte
        CAA9           FF D3     CALL GRAPH         Command
L 1     CAAB           AC        LODSB              load accumulates
        OAAC        BB DC OA     MOVBX, FTPR        print character
        CAAF           FF D3     CALL FTPR          provide delay
        CAB1        BB F0 OA     MOV BX, DELAY
        CAB4           FF D3     CALL DELAY
        CAB6           E2 F3     LOOP L1            print next
        CAB8           59        POP C
        CAB9           5E        POP SI
        CABA           C3        RET
```

FUNCTION NAME	FTPR
INPUT	AL
OUTPUT	AL
CALLS	NONE
DESTROYS	AL,DX
DESCRIPTION	This routine outputs the received to the printer and after sending each character this routine checkes the Busy signal of the printer.

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
0;	0AD0	BA F8 FF	MOV DX,FFF8	output the data through PA0-PA7
	0AD3	EE	OUT DX,AL	
	0AD4	BA FC FF	MOV DX,FFFC	
	0AD7	B0 10	MOV Al,10	make strobe row
	0AD9	EE	OUT DX,AL	
	0ADA	B0 90	MOV AL,90	Make strobe high.
	0ADC	EE	OUT DX,AL	
AGN	0ADD	EC	IN AL,DX	Read status
	0ADE	D0 E8	SHR	Check busy
	0AE0	72 FB	JNZ AGN	if yes,check again
	0AE2	C3	RET	

FUNCTION NAME	DELAY
INPUT	CX
OUTPUT	-
CALLS	None
DESTROYS	CX
DESCRIPTION	This routine provides a delay setting the count this delay can be varied.

LEBEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
O:	0AF0	BO FF	MOV CX 00FF	load counter, do NOP, loop back
DCR	0AF2	91	XCHG AL, AL	
	0AF3	E2 FD	LOOP DCR	
	0AF5	C3	RET	