MICROPROCESSOR - BASED LED MATRIX MESSAGE DISPLAY SYSTEM

BY

IFEYINWA E. CHIKA

SUPERVISED BY

ENGR. DR. J.I. EJIMANYA

Submitted to the Department of Electrical and Electronics Engineering School of Post-Graduate Studies Federal University of Technology, Owerri.

In partial fulfillment of the requirements for the award of M.Sc (Computer Engineering) Degree.

1998
Microprocessor-Based LED Matrix Message Display System

by

Ifeyinwa E. Chika
92/EE/00443/G

A thesis submitted to the Department of Electrical/Electronics Engineering, School of Postgraduate Studies, Federal University of Technology, Owerri, in partial fulfillment of the requirements for award of M.Sc in Computer Engineering.

1998

Engr. Dr. J.I. Ejimanya
Project Supervisor

Dean, School of Post-Graduate Studies

Signature/Date
ABSTRACT

This report describes the design and implementation of a Microprocessor-Based LED Matrix Message Display system. At the heart of the system is the Intel 8085A microprocessor which is suitably interfaced to the system: the display unit, a 16-key hex keypad, and an 8251A Universal Synchronous Asynchronous Receiver Transmitter (USART). The USART provides serial communication with the PC via an RS-232C port. The system can display a variety of alphanumeric characters and graphical symbols within static or animated form. Characters for display may be input from the hex keypad or from the PC. Current design limits single characters to a maximum of 7 x 5 LED arrangement and a total of 12 such 7 x 5 characters. The report also presents the system software. The software has been developed in two parts: one in assembly language using the 8085A instruction set and resident in the interfaced EPROM and the other written in GW-BASIC and resident in the PC. A major attraction of the system is the ability of displayed messages to "move" (i.e animation). The way in which displayed messages are made to move is only limited by software.
ACKNOWLEDGEMENT

This report will not be complete if I fail to acknowledge the assistance and encouragement of a number of people. Engr Dr J. I. Ejimanya is the kind of supervisor every student should have. Without his assistance, encouragement, advice and suggestions, this report might not have been. He gave all: time (to listen, correct, and educate); patience; advice; understanding; materials; and above all, he was always there when I needed him. Dr Ejimanya, please accept my heart felt gratitude. There is somebody out there without whose care, understanding, encouragement, assistance this M. Sc programme will not have been possible in the first place. This is somebody I can never thank enough, even if I spend a lifetime doing that. Engr Dr S. O. E. Ogbogu. I wish to say thank you very much for all your assistance throughout my period of study. Engr Dr Mrs G. A. C. Chukwudebe, thank you for being my lecturer and a friend. Thank you also for the numerous advice and explanations. Prof M. A. Nwachukwu, thank you for being my lecturer and most importantly for all the explanations on the not quite so clear issues and all the helpful materials and suggestions you gave. Engr E. N. C. Okafor, please accept my thanks for the innumerable help and assistance you gave me. You helped ensure that this M. Sc programme is a reality. I wish also to express my gratitude to Engr Ugochukwu Kamalu, Engr Mike Ndinechi, Engr Nnadozie Beemere and a host of others too many to mention, who in one way or the other were of tremendous help to me. Finally, to my most dear God, I say thank you for your love and steadfast faithfulness
# TABLE OF CONTENTS

Title page.................................................................i
Certification.............................................................ii
Abstract.................................................................iii
Acknowledgement.......................................................iv
Table of contents......................................................v
Chapter 1: Introduction...............................................1
Chapter 2: Literature Review..........................................8
  2.1: Typical Electronic Display Devices.........................8
    2.1.1: Liquid Crystal Display Devices (LCD).................8
    2.1.2: Cathode Ray Tube (CRT).................................10
    2.1.3: Plasma Display Panel (PDP)............................11
    2.1.4: Electroluminescence Display (ELD)....................11
    2.1.5: Light Emitting Diodes (LED)..........................12
    2.1.6: Electrochromic Display (ECD).........................15
    2.1.7: Electrophoretic Image Display (EPID)...............15
    2.1.8: Electroplating Cell....................................16
  2.2: Characterization of Displays................................16
    2.2.1: Luminance or Brightness................................16
    2.2.2: Directional Visibility.................................17
    2.2.3: Power Consumption.....................................17
    2.2.4: Response Time..........................................17
    2.2.5: Color..................................................17
    2.2.6: Resolution and Size...................................17
    2.2.7: Memory function......................................18
    2.2.8: Operating Life Time or Degradation..................18
  2.3: Display Techniques...........................................19
2.3.1: 7 - Segment Display .................................. 19
2.3.2: Numerical and Hexadecimal Display .................. 20
2.3.3: Alphanumeric Displays ................................. 21
2.3.4: Graphics .............................................. 22
2.4: Display Drive Techniques ................................. 23
  2.4.1: 7 - Segment Display Drive Technique ............. 23
  2.4.2: Matrix Addressing ................................... 24
    2.4.2.1: Static Drive .................................... 25
    2.4.2.2: Multiplex Drive ................................ 26
2.5: Display Drive Control .................................... 27
  2.5.1: Hardwired Control .................................. 27
  2.5.2: Microprocessor-Based Control ....................... 28
2.6: Applications of Electronic Display Devices .......... 30
2.7: Market Survey .......................................... 32
2.8: Aims and Objectives of the Project Work ............ 34
Chapter 3: Hardware Design .................................. 36
  3.1: Functional Description ................................ 36
  3.2: LED matrix Multiplexing .............................. 38
  3.3: The Keypad Interface Module ......................... 40
  3.4: The Display Circuit Module ........................... 44
  3.5: Memory Decoding/Mapping (Memory Module) .......... 48
  3.6: The PC Interface Module .............................. 52
Chapter 4: Software Design .................................. 58
  4.1: Start Up .............................................. 58
  4.2: Interrupt Handling ..................................... 58
  4.3: Programming the 8251A ................................ 62
  4.4: Programming the 8353 ................................ 65
  4.5: Subroutines .......................................... 69
  4.6: Brief Sequence of Operation of the Software ....... 72
Chapter 5: Implementation/testing/results

5.1: Implementation

5.2: Testing

5.2.1: Testing the display unit

5.2.1.1: Testing the row and column switches

5.2.1.2: Testing the complete display unit

5.2.2 Testing the communications program for downloading messages from the PC

5.3: Results

5.3.1: Result from testing the row and column switches

5.3.2: Result from testing the complete display circuit

5.3.3: Result from testing the communications program

Chapter 6: Conclusion/Recommendations

References

Appendix 1: List of characters and their equivalent ASCII Hex code supported by the system

Appendix 2: Microprocessor-Based LED Matrix Message Display circuit diagram

Appendix 3: A pictorial representation of the supported characters as they would appear on the display panel

Appendix 4: Software Listing

Appendix 5: List of Components

Appendix 6: Power Supply Unit for the display unit

Appendix 7: Listing of the communications program for downloading messages from the PC
CHAPTER 1
INTRODUCTION

Message displays are in widespread use and are very effective medium for attracting attention. They can be used for a variety of applications including retail advertising, in shops' windows, discos, parties, at wedding for entertainment purposes and for warning security messages. They can also be used on stages during shows, for example, during the Practicing Musicians Association of Nigeria (PMAN) awards. They are also used in Banking halls (e.g. NAL Merchant Bank, Owerri) and hotels (e.g. Concorde Hotels Ltd., Owerri) for informing customers of all available services.

Electronic display devices have been defined as: a[a]n electronic component used to convert electronic signals into visual imagery in real time suitable for direct interpretation by a human operator (1). Display devices produce output of an information system. They are the interface between man and machine. Hand-held personal display systems such as calculators and watches need small displays while advertising panels need large displays since they must be viewed from a distance by many people. Ideally, all displays should be viewable in dark and also in ambient light.

Electronic displays are widely used for presentation of graphs, symbols, alphanumerics and video pictures. Large signs, arrival and departure announcements boards and scoreboard (for example the electronic scoreboard at the National Stadium, Lagos) use electronic means to portray changing messages and data. One of the major applications of electronic display is in home television. Also, the computer terminal using cathode ray tube (CRT) is one of the most important applications of electronic display. The standard computer terminal displays 25 lines of 80 characters. This corresponds to a typed page.
The idea of electronic displays is based on the ability to turn on or off individual picture elements (pix elements or simply pixels) as shown in figure 1.1 below.

![Figure 1.1: Pixel array used for creating electronic display images.](image)

The pixel is the smallest controllable element of the display. An array (popularly called the seven segment display) was evolved solely for displaying numeric characters only. It is also called the seven-bar font (fig 1.2)

![Figure 1.2: Seven - bar numeric font](image)

Each bar is a pixel by definition. A similar 14-bar font is used for alphanumeric characters.

However, for small displays such as that of Fig. 1.2, the display elements have separate electrical contacts. But for large - scale displays such as that of fig 1.1, multiplexing is required in order to address the display elements(2). The simplest addressing system is to form an array
of the display elements as a rectangular matrix as shown below.

Fig. 1.3: Array of display elements

On one surface, all the terminals of the elements in a given row are connected to a common bus which runs out to the edge of the array. Each row has its own bus. On the other surface, the elements terminals in a given column are connected to a common bus. Thus every display element has one terminal connected to a row bus and the other to a column bus. Addressing is done by applying voltages to the row and column bus such that the voltage difference across selected elements is above threshold and gives an optical response while the voltage across the unselected elements is not sufficient to generate a response(1).

Although display systems can be very complex and may involve very sophisticated circuitry, this project work is focused on the use of a bank of Light Emitting Diodes (LEDs) to display messages and graphics. This Microprocessor-Based LED Matrix Message Display is based on the use of a microprocessor to control the multiplexing of messages for display onto the LED Matrix. It includes the ability to design one's own messages and incorporate graphics characters to enhance the visual effect. Also the display is under computer control to enable the downloading of messages from a computer for subsequent display.
The message display is based on the idea of representing characters as a series of dots or "pixels" (picture elements). This is very common in today's digital technology. It is used in printing, facsimile, television and computer display (Cathode Ray Tubes (CRT)). For displays using LEDs in this way, each LED represents one pixel. The pixel is the smallest part of a character or graphic. When the display is viewed from a distance, the pixels emerge into their respective characters.

The major feature of this project is that the displayed characters can be made to move and it is this aspect that makes this project work different and more interesting than static illuminated displays. There are many ways in which the characters and messages can be made to move on the display, and this is limited only by the software. These movements are often referred to as "animations".

In the same way that it has been possible to represent all the numerals on seven-segment displays, it has been found that all English alphabets can be represented on a standard "Matrix" of seven pixels (LEDs) high by five pixels wide, that is a 7x5 matrix of LEDs (3). Any fewer number of pixels will make the character unrecognizable. Fig 1.3 below shows the use of a 7x5 pixel matrix to represent the character A.

```
  O   O   O
  O   O   O
  O   O   O
  O   O   O
  O   O   O
  O   O   O
  O
```

Fig 1.3 A 7x5 pixel matrix font example.
This project work will use a set of characters containing all 26 upper case alphabets, 10 numerals, 12 punctuations, and 2 graphics characters for the display of messages. This adds to the flexibility of the display allowing messages to vary in their context. Fig 1.4 shows graphics characters.

![Car](image1)

![Arrow](image2)

Fig 1.4(a): Car  
Fig 1.4(b): Arrow

Fig. 1.4: Graphic examples.

Fig 1.3 is made up using the general 7x5 LED arrangement. The graphics characters are treated as double characters. Any picture or graphic can be created but the limit of seven rows of LEDs (i.e. a column height of seven LEDs) imposes a restriction on the height of the graphic. The "I" character is only three LEDs wide.

Each character to be displayed will have a bit-map (stored in the EPROM) which is a series of consecutive bytes that represent the rows of each character. These are then read out to the display. For the purpose of the program (i.e. the software aspect of this project), each row of LEDs that make up a character is a byte in memory. For a column in a character the top LED is bit 0 and the bottom is bit 6 (bit 7 is not used since there are only 7 LEDs in columns). Fig 1.5 shows how the bytes represents the various rows of a character. A byte of 1FH represents all LEDs on. This means that a 1 represents "ON" and a 0 represents "OFF".
Summarily, the overall features of this message display system are:
- an array of LEDs for display
- a microprocessor controller
- a 16-key Hex keypad for manual keying of user messages
- personal computer interface for downloading messages from PCs
- Animation
- 2 graphic characters
- one-shot or continuous message display
- ability to handle long messages (using word-breaking and scrolling).

The work is subdivided into six chapters: Chapter 1 is the introductory chapter which attempts to acquaint the reader with what the project work is all about.

Chapter 2 is the literature review which involves a review of existing systems, and market research.

Chapter 3 deals with the hardware design. It gives a functional description of the overall system and the circuit description of the various modules.

Chapter 4 deals with the Software Design. It gives a step by step description of the overall function of the software. The major subroutines
of the software are also briefly described, stating which subroutine calls which.

In chapter 5, the implementation steps, precautions during implementation, testing procedures and results obtained are enumerated.

Chapter 6 concludes the report.
CHAPTER 2
LITERATURE REVIEW

Display is the representation of a data item in visible form as "ON" in lights or indicators on a machine console or other devices (4). The contrast ratio, refresh rate, character size and character clarity combine to produce the quality of the display. A device used to provide this representation of data in a visible form is called a display device (5).

Electronic display devices can be defined in terms of their role as "man-machine" interface (2). They can also be defined in terms of their operation. There are electronic devices that convert various electrical signals into optical signals that can be displayed as an image in a form such as digits, characters or graphics. When the optical signal is displayed by the emission of light it is termed active or emissive display but when the display is effected by the modulation of incident light it is termed passive or non-emissive display (2). The invention of these electronic display devices came as a result of such things as liquid crystals, electroluminescence effects in silicon carbide (SiC) and Zinc sulphide (ZnS) phosphor (3).

2.1: TYPICAL ELECTRONIC DISPLAY DEVICES

2.1.1 Liquid Crystal Display (LCD)

Liquid crystal devices are light-controlling displays that rely on the optical properties of liquid crystal material to control the transmission of light from any external source (6). All LCDs consist of a thin layer of liquid crystal material together with means for changing the optical properties of the layer in accordance with the information to be displayed (7).
Liquid crystals are organic materials that have a melting point at which they change from solid to an opaque or cloudy liquid at one temperature, and then turn into an ordinary transparent liquid at a higher temperature. Within a certain temperature range, a liquid crystal substance behaves externally as a fluid, but at the same time has a crystalline structure which exhibits optical properties (2). This is the liquid crystal phase of this organic material. During this phase, the molecules take up certain spatial organisations based on which liquid crystals are divided into three types: Smectic, Nematic and Cholesteric, hence the variants of the liquid crystals displays. These variants are:

- Nematic LCD sometimes referred to as twisted Nematic LCD
- Field - effect LCD
- Guest - Host (GH) LCD
- Electrically Controlled Birefringence (ECB) LCD
- Phase Change (PC) LCD
- Dynamic Scattering (DS) type
- Thermal effects LCD.

The main principle of operation of LCDs is as follows: the application of a voltage changes the molecular orientation of the liquid crystal which results in changes in the optical characteristics of the crystal. These optical changes are then converted to visible changes. In other words, the LCD is a passive display device that utilizes the modulation of light within a liquid crystal cell. The cell is constructed with a layer of liquid crystal about 10μm thick sandwiched between two glass substrates on which are formed transparent electrodes, and a molecular orientation film is formed on the surface of the electrode to impart a fixed orientation to the liquid crystal molecules(2).
The advantages of LCD are:
- low power consumption
- low operating voltage
- good for colour display

The disadvantages include:
- it is not clear in the dark since it is non-emissive
- it has low contrast
- it has no memory function
- response depends on the ambient temperature.

The development of LCDs has been rapid and the range of applications is still widening. However, to enable them (LCDs) to expand their contributions to the field of electronic display devices, there is need to improve on brightness, contrast and resolution. Also the incorporation of memory function will be an added advantage.

2.1.2: Cathode Ray Tube (CRT)

The CRT works by generating an electron beam that when it strikes the phosphore on the face of the CRT, creates a dot of light. Voltages applied to the two sets of inputs control the X and Y coordinates of the position at which the beam strikes the CRT face. A third input to the CRT controls the intensity of the light emitted. The CRT tube, combined with the electronics that control the position of the electron beam and its intensity, comprise a monitor.

The CRT is the most widely used in computer terminals. Apart from its brightness, colour tones and resolution, it is cost effective.
2.1.3: Plasma Display: Panels (PDP)

A "Plasma display" is a flat display that uses the light emission produced by gas discharge. The basic mechanism of the plasma display is the luminescence produced by the glow discharge of ions that collide with each other.

There are basically 2 types: the ac plasma display and the dc plasma display. They operate by the discharge of light emission from a rare gas. This is made to occur at small dots arranged in the form of matrix and letters and diagrams are displayed by combinations of dots (9). The main difference is that the ac type operates by indirect discharge with the electrodes coated with dielectric while the dc type operates by direct discharge with the electronics exposed to the discharge space (2).

Some of the characteristics of the display include its fast display response and long life.

2.1.4: Electroluminescence Display (ELD)

Electroluminescence is the act of light emission when an electric field is applied to a semiconductor phosphor. In one type of Electroluminescence display, light emission is produced when a voltage is applied between electrodes and in another type, light is emitted when an electric field is applied to a pn junction in a single crystal semiconductor (an example of this type is the light Emitting Diode (LED)) (2).

There are essentially three types of electroluminescent: dc powder; ac powder; and thin film (6). One of the characteristics of this type of display is that ELD can be classified into LED; flat screen display; and solid state display.
2.1.3: Plasma Display Panels (PDP)

A "Plasma display" is a flat display that uses the light emission produced by gas discharge. The basic mechanism of the plasma display is the luminescence produced by the glow discharge of ions that collide with each other.

There are basically 2 types: the ac plasma display and the dc plasma display. They operate by the discharge of light emission from a rare gas. This is made to occur at small dots arranged in the form of matrix and letters and diagrams are displayed by combinations of dots (9). The main difference is that the ac type operates by indirect discharge with the electrodes coated with dielectric while the dc type operates by direct discharge with the electronics exposed to the discharge space (2).

Some of the characteristics of the display include its fast display response and long life.

2.1.4: Electroluminescence Display (ELD)

Electroluminescence is the act of light emission when an electric field is applied to a semiconductor phosphor. In one type of Electroluminescence display, light emission is produced when a voltage is applied between electrodes and in another type, light is emitted when an electric field is applied to a pn junction in a single crystal semiconductor (an example of this type is the light Emitting Diode (LED)) (2).

There are essentially three types of electroluminescent: dc powder; ac powder; and thin film (6). One of the characteristics of this type of display is that ELD can be classified into LED; flat screen display; and solid state display.
2.1.5: Light Emitting Diodes (LED)

LEDs are solid state functional devices that convert electricity to light. The most common and simplest display device used with integrated circuit logic is the LED. They emit light energy when stimulated by a low voltage direct current (dc). The most efficient LED is the visible spectrum and emits red light; it is the most commonly used for LED displays (10). Orange, green and yellow LEDs are also available. Fig 2.1 shows the circuit symbol for an LED.

![Diagram of LED circuit symbol](image)

Fig 2.1: Standard Schematic Symbol for an LED

Generally, the output of most logic circuits cannot directly drive an LED at rated current because they cannot sink 20 to 25mA (8). For example standard 74 LSXXX series device can only sink 8mA but some with output buffers, for example, the 74 LS534, an octal D-type flip-flop, can sink 24mA (See fig 2.2 below)

![Diagram of 74LS534 circuit](image)

Fig 2.2: Using a 74LS534 to directly drive LEDs
The LED must always be used with an external series resistor as shown below (fig. 2.3) or driven from a constant current source (11). The external resistor limits the current to the desired value. It is usually called current limiting resistor. LEDs can be used in series but cannot be configured in parallel without a current limiting resistor (11).

![Diagram](image)

Fig 2.3: The use of external current limiting resistor with LED.

For an ac operation, a diode is connected in inverse parallel (fig 2.4) with the LED for equivalent light output as the dc operation. It must be approximately doubled by reducing the value of R (11). The table below is a guide to the resistor value that can be used in series with an LED.

![Diagram](image)

Fig 2.4: Connection of an LED and a diode in inverse parallel for ac operation
<table>
<thead>
<tr>
<th>LED Colour</th>
<th>5V</th>
<th>12V</th>
<th>15V</th>
<th>24V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>270Ω</td>
<td>1kΩ</td>
<td>1.2kΩ</td>
<td>2.2kΩ</td>
</tr>
<tr>
<td></td>
<td>0.125W</td>
<td>0.125W</td>
<td>0.25W</td>
<td>0.25W</td>
</tr>
<tr>
<td>green/yellow/Amber</td>
<td>120Ω</td>
<td>470Ω</td>
<td>560Ω</td>
<td>1kΩ</td>
</tr>
<tr>
<td></td>
<td>0.125W</td>
<td>0.25W</td>
<td>0.50W</td>
<td>0.5W</td>
</tr>
</tbody>
</table>

*Table 2.1*: Series resistor values for LEDs for dc operation

<table>
<thead>
<tr>
<th>LED Colour</th>
<th>5V</th>
<th>12V</th>
<th>15V</th>
<th>24V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>180Ω</td>
<td>470Ω</td>
<td>1kΩ</td>
<td>5.6kΩ</td>
</tr>
<tr>
<td></td>
<td>0.125W</td>
<td>0.125W</td>
<td>0.5W</td>
<td>2.5W</td>
</tr>
<tr>
<td>green/yellow/Amber</td>
<td>100Ω</td>
<td>270Ω</td>
<td>560Ω</td>
<td>3.3kΩ</td>
</tr>
<tr>
<td></td>
<td>0.25W</td>
<td>0.5W</td>
<td>1W</td>
<td>4W</td>
</tr>
</tbody>
</table>

*Table 2.2*: Series resistor values for LEDs for ac operation

*Note*: a diode must be connected in inverse parallel with the LED

LEDs have been found to have some properties that make them attractive for display applications. These include:

- they provide small light sources of high reliability
- they require low drive voltages
- they are capable of high speed response
- they are reliable
- they are relatively cheap
- they have low power consumption
- they have long operating lifetime

The major disadvantage is that their power consumption per element is high (3).

2.1.6: Electrochromic Display (ECD)

The ECD is based on the ability of electrochromic materials to change color when a voltage is applied to it or when a current is passed through it. This colour change is also reversible when the voltage is removed; the polarity of the voltage or current is reversed. The color change may be from clear, transparent to coloured or from one color to another (12).

The characteristics of ECDs include low voltage operation, low power requirements and memory function capabilities, that is storage of the display so that even when power is off the display is preserved. It also has low power consumption and dissipation. The display usually has good contrast and are not affected by external light source like CRT or LED (6).

2.1.7 Electrophoretic Image Display (EPID)

The operation of this display is based on electrophoresis: the motion of charge particles in an electric field (6). It is nonemissive. The characteristics of EPID devices depend on their applications. Generally, some of the characteristics include:

- distinctive appearance
- short life time
- brightness is limited by the optical properties of material used
- low power consumption
practical response time
- high resolution.

In fact EPID has been dubbed electronic paper based on its ability to display a page of text or graphics with excellent resolution (3).

2.1.8 Electroplating Cell

This is based on the deposition of metal atoms by plating onto a transparent electrode to generate a reflecting or an opaque surface depending on the smoothness of the deposit (6). The plated image is stored indefinitely when current is removed, that is, it has memory function. The image can be erased by electroetching when the polarity of the current is reversed. Its characteristics include:

- unevenness of display
- low power consumption
- low cost
- slow response time
- memory function

The EPID's major area of application is in X-ray devices.

2.2 CHARACTERIZATION OF DISPLAYS

2.2.1: Luminance or Brightness

This is said to be the most important property of the LED display. It determines how well the display can compete with the ambient light which is a severe problem in sunlit places.
2.2.2: Directional Visibility

The ability to view a display from all directions is an advantage.

2.2.3: Power Consumption

The applied voltage required to drive a display device is called the operating voltage $V_0$, the current flowing through the device when it is operating in called the operating current $I_0$, and the product of the two the power consumption, $W$. For practical purposes, it is stipulated that a low operating voltage and low current consumption are desirable. The operating voltages of LCD and LED display devices are said to be low(2).

2.2.4: Response Time

This is normally indicated as the rise in time, or the time required from the application of the operating voltage until the display function appears. The decay time is indicated as the time until the display function disappears after removing the voltage function (6).

2.2.5: Color

The color of the light emitted by a display device is termed the display color. The display was found to be determined by the display principles and the material used.

2.2.6: Resolution and Size

The information content of a display increases with the number of picture elements or pixels (1). Resolution has been defined as number of distinguishable lines per unit length (6). As the size of each pixel increases, the driving power increases and hence the resolution.
2.2.7: Memory Function

A display device is said to have a memory function if the display is preserved after the applied voltage is removed or turned off. This will obviously help reduce the power consumption of the display device and will also help to simply the circuit. The LED has no memory function.

2.2.8: Operating Lifetime or Defradation

The life of a device is obviously of great importance. The operating lifetime of a display device is affected both by the display principle and the nature of materials used (2).

Table 2.3 below shows the place of the display devices under the different characterization heading. Only the three main display devices LCD, CRT & LED are considered.

<table>
<thead>
<tr>
<th>CHARACTERIZATION</th>
<th>LCD</th>
<th>CRT</th>
<th>LED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brightness</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Directional Visibility</td>
<td>x</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Response time</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Color</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resolution</td>
<td>low resolution</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Memory function</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Operating lifetime</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2.3: Relation of the display devices to the display characteristics
For matrix displays, all the display characteristics are essential. From these, it is seen that the LED possesses all but one, the memory function, of these characteristics and is therefore excellent for matrix displays.

An ideal display device would modulate the ambient light but would emit bright light in the dark; and be visible from all angles; have high resolution; respond in a very short time but retain the image indefinitely if so desired and consume negligible power at low voltage (6). There are devices possessing many of these qualities but none combines all of them.

2.3: DISPLAY TECHNIQUES

2.3.1: 7-Segment Display

Decimal digits and some letters of the alphabet can be displayed by using seven segments in the font shown below

```
   a
  f |   | b
```
```
  g
 e |   | c
```
```
  d
```

Fig 2.5: Seven-segment display format

These are designed for optimum off/on contrast (11). The common anode variant (Fig 2.6(a)) incorporate a Left Hand Side (LHS) decimal point and the common cathode type (Fig 2.6(b)) incorporates a Right Hand Side (RHS) decimal point.
Fig 2.6: 7-segment version (a) common anode version
(b) common cathode version.

The 7-segment display comes packaged in a variety of ways. Each segment can contain 2 or 3 chips depending on the design and the display is usually recommended for 12V. There are 2-, 4-, 3-, 5- and so on digit 7-segment displays.

2.3.2 Numerical and Hexadecimal Display

The come mostly in the from of 4 x 7 hexadecimal LED display with decimal points either or the left, right or on both sides as shown below

Fig 2.7: 4 x 7 LED hexadecimal Displays

The numeric and hexadecimal devices have on board IC that contains the data memo decoder and display driver functions (11). The numeric devices decode positive BCD logic into characters: 0 to 9, a minus (-) sign
and a decimal point. The hexadecimal device decodes BCD logic into 16 characters: 0 to 9, and A to E.

2.3.3: Alphanumeric Displays

These are 5 x 7 dot matrix LED displays that come in integrated chip (IC) form (fig 2.8). They normally have X - Y select so that ends can be stacked in X and Y directions (i.e. they are end or side stackable) with interlock mechanisms for multidigit applications.

```
   . . . . .
   . . . . .
   . . . . .
   . . . . .
   . . . . .

Fig 2.8: 5 x 7 dot matrix arrangement
```

When cascaded full character displays can be achieved by either of the following:

- Serial in parallel out 7-bit shift register is associated with each digit. This controls constant current LED row drivers. A full character is displayed using a strobe.
- ROM decoders and drivers can be used to drive them
- Character generators can also be used.

Some of them have integral character generator. They have very fast access time and each digit can be independently addressed.
There are also 16 x 1, 5 x 10 and so on LED dot matrix modules with cursor, that are capable of displaying 160 different alphanumeric characters and symbols. The user can also produce character patterns using on board RAM facilities. They have some control commands such as display, clear, on/off and so on (11). These are sometimes called intelligent alphanumeric matrix displays. They can be implemented using LCD, LED and ELD.

2.3.4: Graphics

A range of graphics modules with on board drivers are available for example Hewlett Packard Monolithic 30 x 36 LED array. Graphics displays differ from alphanumeric displays in that alphanumeric displays can only display text, that is alphabets and numerals while graphics displays have the capability to display both text and graphics.

Graphics display modules come in 240 x 64 units or more or less with integral controllers. Some require external controller cards. They have a wide viewing angle and high contrast ratio producing very sharp images. Each module can display both text and graphics and have interface facilities for connection to a microprocessing or a personal computer - based system. Some of the features are

- they can display using 8 x 40 format
- they incorporate 8-bit parallel interface
- they have Rom and RAM facilities.

These graphics display modules would have been ideal for this project work but for their unavailability.
2.4 Display Drive Techniques

2.4.1 7-Segment display drive Technique

This is the most common drive technique the 7-segment display has seven individual segments (see fig 2.5). The display elements have separate electrical contacts as shown in fig 2.9 below.

![Diagram of 7-segment display elements with separate electrical contacts]

Fig. 2.9: 7-Segment display elements with separate electrical contacts

A voltage is applied selectively to each of the segment so that any of the digits 0 to 9 can be displayed.

When interfacing this type of display, two problems must be solved

(a) Decoding of the BCD or hex encoded binary digits to seven-segment code.

(b) Provision of a driver circuit capable of handling the required LED currents.
Problem:

(a) Can be solved by using a BCD - to - seven - segment or hex - to - seven - segment decoder.

(b) is selecting a buffer or driver circuit capable of sinking the LED current

This hardware approach of selective application of voltage to individual segments works fine only when one or two displays are to be interfaced. It becomes cumbersome when several digits of display are needed. For instance, if a 12-digit display is desired, 12 displays, 12 decoders and 12 drivers will be required, that is 36 IC sockets

2.4.2 Matrix Addressing

This is usually used when the display is to be capable of an arbitrary pattern. The matrix, shown in fig 2.10, has N rows and M columns so that by means of selectively applying a voltage to one row line and one column line, any intersection point (pixel) can be activated and character displays and graphics displays can be obtained.

Fig 2.10: Matrix addressing.
The principle drive techniques used for matrix addressing are:

2.4.2.1: Static Drive

In this method, each pixel to be lit is driven separately and simultaneously. Therefore each pixel requires independent circuit elements and a voltage is applied continuously to the selected column while the pixel is to be displayed. Fig 2.11 shows a drive connection for the static drive technique.

![Diagram of drive connection for static drive technique]

Fig 2.11: drive connection for static drive technique

For numeric or alphanumeric display with a large number of digits or characters (which is the case in the project work) the number of drive circuit elements is large. The multiplex drive method described below alleviates this problem.
2.4.2.2: Multiplex Drive

This technique is used for displays with a relatively large number of pixels, such as multidigit numeric displays and also for matrix displays. In this case, the row driver works by operating the N rows in time sequence and the column driver also selectively drives the M columns, the schematic is shown in fig 12.12. This reduces the number of circuit elements and allows one set of decoding and driving circuitry to be shared among several pixels in a display (8).

Fig 12.12: Multiplex Drive Connection

Since all the row elements are connected in common and a whole row is usually activated, voltage will be applied to elements other than those to be displayed. This reduces the display contrast (2). For application such as graphics displays where a large number of rows and columns are
required, switching element can be used for the pixels (13). This is to help improve the display characteristics such as contrast and response.

2.5 DISPLAY DRIVE CONTROL

2.5.1: Hardwired Control

This involves the use of digital IC which are connected together to control the multiplexing of the display elements. On major way of implementing hardwired logic control is the use of character generators.

A character generator is a unit that accepts input in the form of one of the alphanumeric codes and prepares the electrical signals necessary for its display in the proper position on a dot matrix display unit. It is a display controller which draws alphanumeric characters and special symbols for the display unit so that a character is automatic drawn and spaced every time a character code is interpreted. Simply put, it is a hardware device which provides the means of formulating a character font.

The main attractions of the hardwired logic control is its very fast response and modular design features. The disadvantages include:

- **Inflexibility**: Modification can only be made by rewiring the circuit

- **Ghosting can occur**: This occurs when pixels in a display are dimly lit when they should be OFF. These pixels correspond to the pixels that are to be lit in the next digit to be displayed in the multi-plexing sequence. This problem is the result of the digit drivers transistor of one digit being turned OFF while simultaneously the digit driver transistor of the next digit is turned ON. Since transistors are able to turn ON faster than they can turn OFF, the digit turning OFF starts to display the pixel information for the digit turning ON (8).
It is good for a display system with few display elements, otherwise it becomes cumbersome.

Apart from the above mentioned disadvantages of hardwired logic control, character generators could not be used for this project report for the following reasons:

(i) they generate only alphanumeric characters and are therefore not suitable for graphics displays.
(ii) they are unavailable
(iii) they will introduce inflexibility into the work

2.5.2: Microprocessor - based Control

A better control method is to let the microprocessor do the decoding and multiplex the display (14). Fig 2.13 illustrates this technique for a matrix display controlled by the 8085 microprocessor. The display has all its rows connected to a common row driver circuit which is connected to the microprocessor through a row latch or buffer. Also, all the columns are connected to a common column driver circuit which is connected to the microprocessor through a column latch or buffer. It is the software that will make this circuit work. The data in the row latch is used to selectively turn on the rows through the driver circuit. The columns are also selectively energized by software through the column latch so that by turning on a particular row and a particular column, a selected pixel can be turned on.
Fig 2.13: Matrix display under microprocessor control

Obviously, the flexibility of this system is only limited by the software. One approach to the software control is to have an 8-bit display buffer in the microprocessor RAM. The contents of this buffer, which is usually the digital code (BCD, ASCII or Hex) for the character to be displayed, are manipulated by software to create the required bit pattern for the display (8).

The advantages of this multiplexing scheme are:

(i) a fewer number of driver circuits are required instead of one driver circuit for each row or column.

(ii) because the display function is accomplished by software, special characters can be displayed in addition to numerals and letters.
(iii) since decoding and multiplexing are now done by software, the use of decoders have been eliminated.

The only disadvantage of this scheme is that the response is slower compared to the hardwired logic control scheme. One way of improving on these problems is to modify the software to reduce the execution time of the algorithm. For example, replacing subroutine calls with instructions that accomplish the subroutine function saves the time required to call and return from a subroutine and the time needed for subroutine passing (8). This helps to tremendously improve the response time of the scheme.

2.6 APPLICATIONS OF ELECTRONIC DISPLAY DEVICES

Electronic displays devices play a major role in the information world and existing types of electronic display devices are used in a wide variety of applications both industrial and domestic. The major consumer market for display devices is found in personal electronic systems that can be carried on the wrist or in the pocket - mostly watches and calculators (6). The next largest user of display devices is instrumentation. For business use, where large amount of data must be displayed simultaneously, flat panel displays are required.

The first stage in the development of LCDs as practical devices was in 1973 with the implementation of numeric display panel for watches and calculators (2). Later, the application of LCDs was extended to other areas like measuring instruments, domestic electrical and audio equipment, office equipment and game machines. Currently, LCDs are also used in colour displays, large-capacity character displays and graphic display.
Electrochromic displays can also be used in clocks, watches, calculators, signboards vehicle displays and so on. However, the range of applications is limited by the problems of higher power consumption and slower response. On the other hand, because of its memory function capabilities, it is good for displays that require a low frequency of update hence its principal application in watches and clocks.

CRTs are mostly used for display of large quantities of information which explains their dominant use in computer terminals. However, with increase in demand for display systems, and the need to overcome some of the problems of CRTs, such as large size for small display capacity, PDPs are used in applications where the features of CRTs are inadequate. Such applications include office automation where there is need for compactness.

Display can be digital or analogue. Examples of digital displays are numerical and symbol displays such as in measuring instructions. ELDs are good for digital displays and have an added advantage in that they can be made of smallest thickness and larger size which improves the external appearance of the display. ELDs are also used in analogue displays in meters such as car dashboard instrument panel.

The simplest application of LED as a display is in pilot lamps (2). LEDs are widely used in commercial products such as High Fidelity (HIFI) audio equipment and industrial instrument which have LED indicators. For use as an indicator, the LED encapsulation can be shaped in several different ways to control the radiation pattern (§5). The major application of LED is in seven bar segments (for numerals) and alphanumeric displays with a 35 dot matrix arranged in a 5 x 7 array (§5). Large displays are obtained by arranging LEDs in various display patterns and colours.
The applications of display devices can be classified into three major areas: Personal electronic systems such as watches, clocks, temperature thermometers, calculators, HiFi audio equipment etc.; Instrumentation such as cockpits of aircrafts, vehicle instrumentation panels, computer terminals, measuring instrument etc.; Business applications such as electronic billboards and scoreboards, electronic exchange rate display used in banks and bearable change, arrival and departure announcement panels at airports and so on.

2.7 MARKET SURVEY

A market survey was conducted for this project work. The aim of the survey was to get information regarding the commercially available forms or variants of this system. A number of places were visited in the process including NAL Merchant Bank and Concorde Hotels Ltd both in Owerri.

During the survey, it was found that most of the commercially available systems were implemented using proprietary chips and commercially available LED matrix display panels hence the inherent sophistication. A few of the finding are highlighted below.

(a) Dot Matrix DLR 1414/2416: These are 4 digit 5 x 7 dot matrix displays with integral ASCII ROM decoder, memory, character decoder, multiplex circuitry and drivers. Data entry is asynchronous and each digit is independently addressable. The displays are end stackable so that they can be stacked to get 8-, 12-, 16 - digit displays and so on. They display data until it is replaced. The DLR 2416 has cursor editing facilities.
(b) PDSP2110 SERIES: These are 8-digit 5 x 7 dot matrix with a two hundred and fifty six on-board characters including ASCII. It has programmable features: individual character flashing, full display blinking and self test. They have low current consumption and wide viewing angle. Their applications include: industrial process control, medical equipment, instrumentation and audio and video products.

(c) Tri-colour LED message Displays: These are 2 inch, 14 character, 5 x 7 dot matrix tri-colour LED message displays housed in compact light weight black aluminium casing. These units are capable of displaying both text and graphics and are ideal for application where a clear bright display is required. Each display has an infrared remote control keyboard for programming the display. Alternatively, the display can be programmed from a PC via an RS232 link. The display has a separate power supply. Some of its features include:

- 16 colour variations
- 26 colour combinations
- 96 alphanumeric characters
- 8 character fonts
- 3 speeds of scrolling
- integral clock and calendar
- 90 day memory battery back-up
- user password control and so on.

The display systems at the NAL Merchant Bank and Concorde Hotels Ltd are similar and have most of these features except the colour variation features, they display only the red colour. The only other area of difference
is that the systems at the bank and Hotel have 2-line message display panels.

Other commercially available systems include: the HELX2416 series; the SLX2016 series; and the HDSP2100 series with user accessible RAM hence their read/write capability (16).

2.8 AIMS AND OBJECTIVES OF THE PROJECT WORK

During the market survey and examination of some commercially available message display systems, the following prevalent problems were identified:

- They are not readily available as a result of the fact that they are stocked by very few people.

- They are costly. This also stems from the problem of being stocked by few people which now gives those people monopoly of the system.

- They are not easily maintainable in that most of these systems are built using proprietary chips which are not readily available in the country. When the system breaks down, the chips cannot be found for replacement.

- They are custom made, that is they have fixed purposes, take the electronic exchange rate display panel for example, which makes them inflexible.
The aims or reasons for this project work are therefore to

- design and implement a message display system using locally available chips to facilitate maintainability.

- design and implement a cost effective display system which is readily available and easily affordable.

- design and implement a flexible system which the user can easily adapt to suit his purpose.
CHAPTER 3
HARDWARE DESIGN

3.1 FUNCTIONAL DESCRIPTION:

A block diagram of the message display unit is shown in Fig 3.1. On the right hand side of the block diagram is the LED matrix module which consists of seven rows of 128 LEDs. This implies that the display panel is a $7 \times 128$ (= 896 LEDs) matrix of LEDs. This can display up to 25 (excluding gaps) characters or up to 21 characters, including gaps, simultaneously. Each of the rows has a switch circuit which supplies current from the power supply when activated. These row switches are controlled by a set of seven latches which receive data from the data bus.

The columns also have switch circuits which will sink current to the ground when activated. The column switches are also controlled by latches. The column latches are arranged in sixteen groups of eight. Each group can be written from the data bus under the control of sixteen column latch select lines which make up the select bus. Both the data and select buses are buffered before connection to the column latches.

The system is expandable to up to a $7 \times 256$ LED matrix which gives a total of 1792 LEDs and this can display up to 51 characters simultaneously.

On the left hand side of the block diagram is the CPU/PC INTERFACE module. The clock circuit (the crystal clock connected to x1 and x2 inputs of the microprocessor) provides the CPU timing reference while the reset circuit is used to restart the CPU and ensure that it starts executing program instructions from the beginning when power is first applied. The Erasable Programmable Read Only Memory (EPROM) and the Random Access Memory (RAM) are connected to the Central Processing Unit (CPU)
Fig. 3.1: Block Diagram of the LED Matrix Message Display System

- Data Bus (8-bits)
- Address bus (16-bits)
- 16-key keypad
- Hex keypad encoder
- Memory Address Decoder
- CPU
- EPROM
- RAM
- USART
- PC Interface
- From I/O Address Decoder Circuit
- Reset Circuit
- LED Matrix
- Column Latches
- Row Switches
- Row latch
- Buffer (for the 16-bit latch select bus)
- Column Latches
- Column Switches
- GND
through the address and data buses and are selected under the control of the memory address decoder.

The input/output (I/O) address decoder uses control signals from the CPU to select either the row latches, for writing row data to the row latch; the column latches, for writing data to the column latch; the keypad buffer, for keying in of message; or the personal computer interface chip (the 8251A) for downloading messages from the computer. The keypad buffer holds the output of the 16-key hex keypad encoder. The keypad buffer (IC35a) is connected to the microprocessor via the data bus. The keypad buffer together with the keypad encoder constitute the keypad module.

3.2 LED MATRIX MULTIPLEXING

The LED matrix configuration allows all the 896 LEDs (7 x 128 array) to be controlled with a total of 135 switches (7 switches for the rows and 128 switches for the column) turning ON both a row switch and a column switch completes a circuit and current flows through the LED at their intersection. This arrangement creates a problem by restricting the combination of LEDs that can be switched on simultaneously. For example, assume that just two LEDs are to be illuminated and that they are positioned diagonally adjacent to each other as shown below.

![Diagram of LED matrix multiplexing](image)

Fig 3.2: the LEDs circled by dotted lines are to be illuminated.
To illuminate LEDs D2 and D3, at the same time, both of the rows would be switched so also would both of the columns. It is obvious that this will result in block of the four LEDs, D1, D2, D3 and D4, all being illuminated. This shows that the individual LEDs are not individually controllable.

To effectively control the LED matrix, a multiplexing technique has to be employed. This is done as follows: all the appropriate LEDs along one row (the top row for example) are selected by turning on the corresponding column switches. The row switch is then turned on for a short, but measured period of time. With the row switch turned off, again the column switches are altered to reflect the pattern of LEDs to be illuminated on the next row down. This next row is then switched on for the same short period of time. The remaining rows are activated sequentially in the same manner until all seven rows have been displayed. The entire frame is then re-displayed from the top.

The sequence of row and column switching if carried out at sufficient speed, results in a stable display with all LEDs capable of individual control. If the frame repetition rate is high enough, the human eye can fail to keep pace with the rapid switching and interprets the light output as constant, which is an effect known as "persistence of vision".

The use of a microprocessor in the design which is well suited for high-speed repetitive tasks makes the multiplexing easier and simpler. Multiplexing is preferable compared to the alternative method involving driving each LED from a separate latch which will involve wiring 896 latches. This could become very costly and cumbersome.

The one major problem with this multiplexed display is that each LED is only switched ON for one seventh ($\frac{1}{7}$th) of the frame time). This could result in reduced brightness.
3.3 THE KEYPAD INTERFACE MODULE

The keypad is a 4 x 4 switch matrix, that is, a 16-key Hex keypad. The switches are labelled 0 to 9, A to F so that any character can be keyed in using its ASCII hexadecimal code. For instance, to type in the letter, A one simply types the hex number 41 on the keypad which is the ASCII hex equivalent of A. The block diagram of the keypad interface is shown in fig 3.3.

The keypad is scanned by a counter and each key is checked individually for key closure. The counter counts in the following sequence as shown in Table 3.1.

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_4$</td>
<td>$X_3$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 3.1: The keypad Scanner Count Sequence.*
Fig. 3.3 Block diagram of the Keypad Interface
Bits X1 and X2 are used as inputs to the encoder while bits X3 and X4 are used as inputs to the decoder so that the columns of the keypad are addressed as follows:

<table>
<thead>
<tr>
<th>X2</th>
<th>X1</th>
<th>Column addressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 3.2**

and the rows are addressed thus

<table>
<thead>
<tr>
<th>X4</th>
<th>X3</th>
<th>Row addressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 3.3**

From the count sequence (table 3.1), it can be seen that the column addresses change four times faster than the row addresses. The row address changes after every four counts. That is, the scan sequence is such that while $x_4x_3 = 00$, the values of $x_2x_1$ go from 00 to 11 at the end of which the value of $x_4x_3$ changes to 01 and $x_2x_1$ again goes from 00 to 11. This goes on until the value of $x_4x_3$ returns to 00 and the whole process is
repeated. Once a key closure is detected, the CPU is interrupted to tell it that data is available at the keypad.

When a row is addressed, all the columns are scanned to see if there has been a key closure. If there is no key closure, the next is addressed and all the columns scanned. The keypad interface circuit is shown on the lower left hand side of appendix 2.

The MM74C922N keypad encoder encodes any 16-key keypad that is connected to it. It incorporates all the logic necessary to fully decode an array of 16 single pole/single throw (SPST) momentary contact switches (normally open) into a binary code. The switches arranged in a 4 x 4 matrix are sequentially scanned at a rate determined by an internal clock. The frequency of the clock is determined by the clockout of the CPU.

The keypad encoder can debounce the switches (though this option is not implemented in this project work). The output of the encoder are 3-stated. Internal pull-up resistors on key inputs allow switches with up to 50kΩ "ON" resistance to be connected directly to the device.

The keypad encoder constantly scans for a keyclosure. On detecting a keyclosure, a data available interrupt signal is sent to the 8085's RST6.5 interrupt input and the contents of the internal latches appear on the data output pins (this is because the encoder is permanently enabled for output).

The keypad is selected for input as follows:

The output enable input (OE) pin 9, of the keypad buffer, IC35(a) (8286 octal bus transceiver) has as its input the output of a 2-input NAND gate. The 2-input NAND gate in turn, has as one of its inputs the output of an 8 - input AND gate (I46(q) 74LS30). The inputs of IC46(d) are as shown below
3.4 THE DISPLAY CIRCUIT MODULE

The circuit diagram of the display unit or module is shown on the lower right hand side of appendix 2. The 8085 CPU writes data to the row latch, 74LS377, octal flip flop (IC33) by holding the chip enable input (CE), pin 1, low while a low going pulse is applied to the clock (clk) input, pin 11.

Seven identical row switch circuits are connected between the latch outputs and the LED matrix. The row switch circuit operates as follows: The output of the latch is coupled to the base of TR1, BC548, an npn transistor via a resistor R136 which will turn on the transistor whenever the latch output is high. TR8, TIP 127, an pnp darlington power transistor turns on whenever TR1 is switched on and supplies current to the row. This power transistor must be capable of providing enough current for the 128 row - LEDs, even though not all of them will be ON all the time.

Column data is written into the sixteen groups of column latches 74HCT574, octal D-type flip flop (IC1 to IC16) one group at a time by applying a low going chip select pulse to each of the sixteen clock (clk) inputs in turn. The chip select signals and the data bus signals are buffered using HCT devices, 74HCT245, octal bus transceiver (IC34(a) and (b)) which convert TTL signals from the CPU to CMOS compatible signals.
for the column latch chips (IC1 to IC16) and also provide the required output current to drive the bus lines. The column switches, ULN2803A, octal darlington Driver (IC17 to IC32) are simply npn darlington transistors in packs of eight, and are directly connected to the outputs of the column latches.

Each column of LED has a current limiting resistor (R1 to R128) connected in series. Each of the LEDs in the column uses this resistor in turn.

The value of R1 to R128 is calculated as follows:

It has been stated that for DC operation, the normal operating current of LED is typically 20mA (1). For the LED matrix multiplexing, each LED is switched on only for \( \frac{1}{12} \)th of the frame time. This, as was stated earlier, could lead to reduced brightness. Because of this multiplexing system, for compensation, the LEDs are driven at 100mA, five times their normal maximum current rating. The maximum recommended LED supply voltage is 12V.

**Assumption:**

At least 2V is lost in the power transistor junctions and the LED itself. This leaves about 10V across the series resistor.

To determine the value of the resistor, we use ohms law

\[
R = \frac{V}{I}
\]

\[
R = \frac{10V}{100mA} = \frac{10}{0.1} = 100\Omega
\]
The power dissipation required of the resistor is calculated as follows

\[ P = I^2R \]

\[ = (0.1)^2 \times 100 \]

\[ = 1 \text{ watt} \]

Hence, the series resistors required are 1 watt resistors

or

The IC37(a) selects either the display, the keypad. For a write operation, the display is selected but for a read operation either the keypad or the PC is selected. The truth table for IC37(a) is shown below.

<table>
<thead>
<tr>
<th>C</th>
<th>B</th>
<th>A</th>
<th>( \overline{WR} )</th>
<th>( \overline{RD} )</th>
<th>( \overline{y}_3 )</th>
<th>( y_2 )</th>
<th>( \overline{y}_1 )</th>
<th>( y_0 )</th>
<th>Device Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td>Nothing happens</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>The row latch for display i.e.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>IC37 is selected.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>The keypad buffer is enabled</td>
</tr>
</tbody>
</table>

*Table 3.4: Truth table for IC37(a)*

The IO/m is the chip enable signal for IC37(a)

The Rows are selected as follows:

I/O write (i.e. \( IO/\overline{M} = 1 \)) indicates the CPU wishes to write data to the display unit from memory. It puts the row data on the data bus and sets or resets the control pins as follows

\[ IO/\overline{M} = 1, \ \overline{WR} = 0, \ \overline{RD} = 1 \]
Then the IC33 (the row latch) is clocked and the data on the data bus is driven by the row drive circuit to the rows of the display matrix. Only one row is selected at a time. For example, if data on the data bus is

\[
\begin{array}{c|c}
D7 & Do \\
\hline
00100000 & \\
\end{array}
\]

Then row 5 is selected (i.e. turned on). (The rows go from Row 0 to Row 6).

The Columns are Selected as Follows:

There are 16 column latch select lines from IC36a and (b) used to select only 1 column latch at a time. The truth table for IC36(a) is shown below.

<table>
<thead>
<tr>
<th>G2A</th>
<th>G2B</th>
<th>G1</th>
<th>C</th>
<th>B</th>
<th>A</th>
<th>A0</th>
<th>y7</th>
<th>y6</th>
<th>y5</th>
<th>y4</th>
<th>y3</th>
<th>y2</th>
<th>y1</th>
<th>y0</th>
<th>IC</th>
<th>Written</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/mm</td>
<td>WR</td>
<td>A3</td>
<td>A2</td>
<td>A1</td>
<td>A0</td>
<td>y7</td>
<td>y6</td>
<td>y5</td>
<td>y4</td>
<td>y3</td>
<td>y2</td>
<td>y1</td>
<td>y0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td>Data is written to the row latch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>IC1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>IC2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td>IC3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>IC4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>IC5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>IC6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>IC7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 3.5: Truth table for IC36(a)*

47
A3 = 1 implies the column latch is addressed, A3 = 0 implies the row latch is addressed. Therefore, for the CPU to select columns 9 to 16 for example, it implies that \( \text{IC}_{3\text{A}} \) has to be clocked so that the data on the data bus is latched and the appropriate columns turned on. The CPU will place the data 11111111 (since columns 9 to 16 are to be activated) on the data bus and the address XXXX1001 is sent to the address bus.

The power supply unit is shown in appendix 6.

3.5 MEMORY DECODING/MAPPING (MEMORY MODULE)

The memory module is shown on the top right hand side of appendix 2. The memory decoding is done using IC36(c) whose truth table is shown below.

| A15 | A14 | A13 | \( \bar{y}_7 \) | \( \bar{y}_6 \) | \( \bar{y}_5 \) | \( \bar{y}_4 \) | \( \bar{y}_3 \) | \( \bar{y}_2 \) | \( \bar{y}_1 \) | \( \bar{y}_0 \) | Memory Block Selected |
|-----|-----|-----|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|---------------------|
| 0   | 0   | 0   | 1              | 1              | 1              | 1              | 1              | 1              | 1              | 1              | 0                   | 1st 8KB EPROM       |
| 0   | 0   | 1   | 1              | 1              | 1              | 1              | 1              | 1              | 1              | 0              | 1                   | 2nd 8KB EPROM       |
| 0   | 1   | 0   | 1              | 1              | 1              | 1              | 1              | 0              | 1              | 1              | 1                   | 1st 8KB RAM         |
| 0   | 1   | 1   | 1              | 1              | 1              | 1              | 1              | 0              | 1              | 1              | 1                   | 2nd 8KB RAM         |
| 1   | 0   | 0   | 1              | 1              | 1              | 0              | 1              | 1              | 1              | 1              | 1                   | 3rd 8KB RAM         |
| 1   | 0   | 1   | 1              | 1              | 0              | 1              | 1              | 1              | 1              | 1              | 1                   | 4th 8KB RAM         |
| 1   | 1   | 0   | 1              | 0              | 1              | 1              | 1              | 1              | 1              | 1              | 1                   | not used             |
| 1   | 1   | 1   | 0              | 1              | 1              | 1              | 1              | 1              | 1              | 1              | 1                   | not used             |

Table 3.6: Truth table for IC36(c)

The decoding was achieved by dividing the 64 addressable memory space of the 8085 into 8KB blocks so that each decoder output selects a block. Also exhaustive, instead of partial, memory decoding is used. The address-bit
assignment is as shown below.

<table>
<thead>
<tr>
<th></th>
<th>A₁₅</th>
<th>A₁₄</th>
<th>A₁₃</th>
<th>A₁₂</th>
<th>A₁₁</th>
<th>A₁₀</th>
<th>A₉</th>
<th>A₈</th>
<th>A₇</th>
<th>A₆</th>
<th>A₅</th>
<th>A₄</th>
<th>A₃</th>
<th>A₂</th>
<th>A₁</th>
<th>A₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1st 8K</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ROM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>block</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2nd</td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3FFF</td>
<td>block</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4000</td>
<td>3rd</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>8k</td>
<td>block</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4th</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>8k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>block</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5th</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>8k</td>
<td>block</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>6th</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>8k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFFF</td>
<td>block</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 3.7: memory address-bit assignment**

The select inputs of the decoder are driven by A₁₃, A₁₄ and A₁₅. Consequently, each output of the decoder selects an 8KB block in the 64KB address space of the 8085. The lower 13 address lines A₀ to A₁₂ are not decoded by the decoder. The first two 8KB blocks occupied by the ROM where exhaustively decoded in that the internal decoder of the 2764 (IC54) decodes address bits A₀ to A₁₂ (13 address lines) and the 74LS138 (i.e.
IC36 (c)) decodes address bits $A_{13}$ to $A_{15}$. The 8KB memory locations in the 3rd, 4th, 5th and 6th blocks are partially decoded in that the internal decoders of the 2141 RAM chip (4K x 1 RAM) decodes address bits $A_0$ to $A_{11}$ and the 74LS138 decodes address bits $A_{13}$ to $A_{15}$. Address bit $A_{12}$ is not decoded. The result of not decoding $A_{12}$ is memory foldback (i.e. multiple memory mapping) so that each byte of memory in these blocks responds to two address. The two addresses differ only in the value of $A_{12}$ (i.e. either $A_{12} = 1$ or $A_{12} = 0$). This means that the 4KB RAM in each block occupies only half of the 8K block. The remaining 4KB is foldback. The memory map is shown in the figure below.
Fig 3.5: Memory Map
3.6 THE PC INTERFACE MODULE

A PC is interfaced to the microprocessor through the intel 8251A, Universal Synchronous/Asynchronous Receiver Transmitter (USART). The USART is a programmable communications interface. The component accepts data characters from the PC in serial format and then converts them into continuous parallel data for the CPU.

Before starting data transmission/receiving the 8251A must be initialized with a set of control words generated by the microprocessor. These control words define the function of the 8251A. The control words are:

- Reset
- Mode instruction
- Command instruction

The mode instruction has two different formats, one for asynchronous transmission and the other the synchronous transmission. The command instruction controls the actual operation of the selected format.

The CS input is controlled by a two input AND gate (IC53) which has as one of its inputs the output of an 8-input AND gate (IC46(b)). The C/D input selects the control functions or data functions of the USART as shown in the truth table below

<table>
<thead>
<tr>
<th>C/D</th>
<th>RD</th>
<th>WR</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Read data word</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Write data word</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Read Status word</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Write status word</td>
</tr>
</tbody>
</table>

Table 3.8: Truth table for the C/D input select of the 8251A
The two baudrate clock inputs, TxC and RxC are connected together. This will cause the receiver and transmitter to operate at the same baudrate. Under program control, the baudrate could be chosen as 1x, 16x or 64x the data rate.

The 74 LS 30 (8 - input AND gate, IC46 c) decodes address lines A₁ to A₇ while the C/D input decodes A₀. The value of A₀ determines whether the address on the address bus refers to the status port or the data port. The address bit assignment and port addresses for the USART are shown below.

<table>
<thead>
<tr>
<th>A₇</th>
<th>A₆</th>
<th>A₅</th>
<th>A₄</th>
<th>A₃</th>
<th>A₂</th>
<th>A₁</th>
<th>A₀</th>
<th>Address</th>
<th>Port addressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A0H</td>
<td>Data Port</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>A1H</td>
<td>Status Port</td>
</tr>
</tbody>
</table>

Table 3.9: USART Port address assignment

The Tx Ready output of the 8251A indicates that the transmitter is ready to accept data from the microprocessor. The Rx RDY signal indicates that the 8251A contains data for input to the microprocessor. The Rx RDY signal is automatically reset by the leading edge of the RD signal.

When the PC wishes to send data to the microprocessor for display, it sends an RTS (request to send) signal to the RST5.5 interrupt input of the microprocessor. The microprocessor then puts out the address bits 10100000 = A0H on the address bus. This enables the 8251A for data input. The CS input of the 8251A is also connected to the CTS (Clear To Send) input of the RS 232C to initiate the beginning of the transfer of data from the PC to the 8251A.
When the 8251A buffer is full, it sends an Rx RDY signal to the RST7.5 input of the 8085. On receipt of this interrupt, the microprocessor accepts the data from the 8251A, and continues what it is doing until another buffer-full interrupt occurs. When the PC finishes sending data, it sends an end of transmission (EOT character) signal to the microprocessor. The microprocessor then goes ahead to display the message.

The PC is selected for input based on the output from the circuit shown below.

![Circuit Diagram](image)

**Fig 3.6**: PC Selection logic

It can be seen that the USART is selected to enable input from the PC to the microprocessor or for the µ up to read the status of the USART WHEN \( A_7 - A_0 = 1010000X \).

The 8253, programmable interval timer (PIT) contains three independent 16-bit counters, each with a count rate of up to 2MHz. It generates accurate time intervals controlled by software. It has 6 different modes of operation.

- **mode 0**: interrupt on terminal count
- **mode 1**: a programmable 1-short
mode 2: - a programmable rate generator
mode 3: - a programmable square wave generator
mode 4: - Software trigger strobe
mode 5: - Hardware trigger strobe.

In this report, the 8253 is used in mode 3 as a programmable square wave generator to supply the 8251A with the Transmitter and Receiver Clock, (TxC and RxC) that is, as a baudrate generator. The microprocessor uses address lines A0 and A1, to select the counters (counters 0, 1 and 2). The write operation is shown below.

<table>
<thead>
<tr>
<th>CS</th>
<th>RD</th>
<th>WR</th>
<th>A1</th>
<th>A0</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>write into counter 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>write into counter 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>write into counter 2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>write control word</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>3 - State</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>3 - State</td>
</tr>
</tbody>
</table>

*Table 3.10: The 8253 Write Operation*

The 8-input AND gate, IC46 (b) decodes the address lines A2 to A7 to generate the cs signal for the 8253, while the 8253 itself decodes A0 and A1 to select the counters so that the counter addresses are as shown below.
<table>
<thead>
<tr>
<th>A_7 A_6 A_5 A_4 A_3 A_2</th>
<th>A_1 A_0</th>
<th>Counter addressed</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 0 0 0 0</td>
<td>0 0</td>
<td>Counter 0</td>
<td>E0H</td>
</tr>
<tr>
<td>1 1 1 0 0 0 0</td>
<td>0 1</td>
<td>Counter 1</td>
<td>E1H</td>
</tr>
<tr>
<td>1 1 1 0 0 0 0</td>
<td>1 0</td>
<td>Counter 2</td>
<td>E2H</td>
</tr>
<tr>
<td>1 1 1 0 0 0 0</td>
<td>1 1</td>
<td>Control port</td>
<td>E3H</td>
</tr>
</tbody>
</table>

Table 3.11: Counter address assignment of the 8253

This implies that the chip select signals (cs) occurs when

\[ A_7 - A_0 = 111000XX. \]

For this report, it is desired that the 8251A should transmit and receive at the same baudrate of 400bits/sec. The 8251A is programmed for asynchronous transimission with a baudrate factor of 16x. The input clock frequency, \( f_c \), of the 8085 is 6.14MHz.

\[ f_c = 6.14\text{MHz} \]

The output clock frequency of the 8085 is

\[ f_{o/2} = 3.7\text{MHz} \]

The clock output from the microprocessor has to be decreased to 2MHz or less in order to drive the 8253. Counter 1 is programmed to operate in Mode 3 since the 8253 is only required to generate the baudrate for the 8251A. The choice of counter 1 is arbitrary. The counter value \( N \) is calculated as follows:

\[ N = \frac{f_{\text{Timer}}}{B_{RF} \times B_R} \]

where

\[ N = \text{value of counter} \]

\[ f_{\text{Timer}} = \text{the input clock frequency of 8253 timer} \]
\[ B_{RF} = \text{Baudrate factor of the 8251A} \]
\[ B_R = \text{Baudrate of the 8251A} \]

We have that

\[ f_{\text{Timer}} = 1.85\text{MHz} \]
\[ = \frac{1.85 \times 10^6}{16 \times 400} \]
\[ = 289.0625 \]
\[ = 289D \text{ i.e. } 289_{10} \]
\[ = 0121H \text{ i.e. } 0121_{16} \] (this value is loaded into counter 1 of the 8253).

The memory module is in the centre of the block diagram together with the I/O address decoder. The memory module consists of the EPROM and RAM which are connected to the Central Processing Unit (CPU) through the address and data bus and are selected under the control of the memory address decoder.
4.1 START UP

The main program loop flowchart gives a good insight of how the various tasks of the system are managed. On start-up of the system, the 8251A and 8253 are initialized and interrupts are enabled. Next, the initial start-up message is copied from ROM into RAM for display. The system goes on to display this start-up message and continues to display and animate this start-up message until an interrupt occurs from either the keypad for keying in of a message or from the PC for downloading a message for display. On the receipt of an interrupt, all current tasks are suspended and the microprocessor branches to the appropriate Interrupt Service Routine (ISR). Each ISR first resets the ASCII text buffer in RAM, this is done by the WIPE subroutine, and then writes the new message into it. The next call to the DISPLAY subroutine will begin the display of the new message.

4.2 INTERRUPT HANDLING

The programmed I/O technique used is based on the 8085 microprocessors built-in interrupt capabilities. The 8085 has five interrupt input pins (TRAP, INTR, RST5.5, RST6.5 and RST7.5) so that each of the input devices (keypad and PC) have their own dedicated interrupt inputs. When the microprocessor samples an interrupt pin and finds it high (all the interrupt inputs are active high), control is taken to an ISR. The address of the ISR is determined at interrupt time.
All the interrupt inputs used for the project (RST5.5, RST6.5 and RST7.5) are maskable, that is they can be turned off or disabled by the microprocessor. Interrupts are automatically disabled whenever a RESET occurs or immediately after the receipt of an interrupt request (10).

Two program steps are required to enable these interrupts:
(a) clearing of the interrupt mask
(b) enabling the interrupts

These are implemented using the SIM Instruction format shown below.

![SIM Instruction format diagram]

Fig 4.1: SIM Instruction format
This can be illustrated, using the following bit pattern.

\[
\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{array}
\]

\[00001000 = 08H\]

- RST5.5 is unmasked
- RST6.5 "  "
- RST7.5 "  "
- Mask set is enabled. This is the control that enables the masking or unmasking of RST5.5, RST6.5 and RST7.5
- RST7.5 is not reset
- undefined
- SOD is disabled
- Serial Output Data

To write this bit pattern into the interrupt mask, and enable interrupts, this is achieved with the following assembly language program steps.

\[
\begin{array}{l}
\text{MVI A, 08H} & \quad \text{; move mask pattern into Acc write the}
\end{array}
\]

\[
\begin{array}{l}
\text{SIM} & \quad \text{; mask pattern. This instruction will copy}
\end{array}
\]

\[
\begin{array}{l}
\text{the contents of the Acc into the interrupt mask.}
\end{array}
\]

\[
\begin{array}{l}
\text{E1} & \quad \text{; enable interrupts.}
\end{array}
\]

To disable the RST interrupts, the following bit patterns are used.

\[
\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

- RST5.5 is masked
- RST6.5 "  "
- RST7.5 "  "
- MSE is enabled

\[
\begin{array}{l}
\text{MV 1 A OFH}
\end{array}
\]

\[
\begin{array}{l}
\text{SIM}
\end{array}
\]

\[
\begin{array}{l}
\text{D1}
\end{array}
\]
Once an interrupt occurs, control must be transferred to the ISR. There are two techniques for doing this for the 8085 microprocessors.

(a) vectored interrupts
(b) Direct interrupts

The Direct Interrupt technique is employed. In this case, an interrupt on one of the interrupt pins causes a direct branch to a specific location in memory. The locations branched to for the interrupts are as follows:

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST5.5</td>
<td>002CH</td>
<td>enable 8251</td>
</tr>
<tr>
<td>RST6.5</td>
<td>0034 H</td>
<td>this location contains a jump Instruction to the ISR for the Keypad</td>
</tr>
<tr>
<td>RST7.5</td>
<td>003 CH</td>
<td>a jump Instruction to the ISR for the PC</td>
</tr>
</tbody>
</table>

Interrupt servicing for Keypad:
In the location where the microprocessor (μp) branches to when an interrupt occurs on the RST6.5, the following instructions is put there:

```assembly
KEYP    EQU $           
disable all interrupts
CALL    KEYPROC         
enable all interrupts remove
RETURN
```

The routine for servicing the keypad is stored beginning from location, 2000H.
Interrupt servicing of the PC:

In the location , where the microprocessor branches to when an interrupt occurs on the RST7.5 interrupt input, the following instruction is put there.

```
PC EQU $
  disable all interrupts
  instruction steps to accept data from the PC
  enable all interrupts
  RETURN
```

The routine for serving the PC is stored beginning from this location.

Interrupt servicing for the 8251A:

In the location 002CH, where the microprocessor branches to when an interrupt occurs on the RST5.5 interrupt input, the following instruction is put there.

```
USART EQU $
  enable the 8251A
  RETURN
```

The routine for enabling the 8251A is stored beginning from this location.

4.3 PROGRAMMING THE 8251A

The procedure for the programming of the 8251A (USART) is as follows:

1. When power is first applied, it is necessary to reset the 8251A by sending a zero byte into the command mode Word address.
(2) Using an internal Reset operation, it is possible to select the Mode Word

(3) A reset operation is followed by the Mode Instruction. The type of transmission (asynchronous or synchronous), the length of the character, the number of stop bits, the baudrate factor (only in asynchronous transmission) and the type of synchronization (only in synchronous transmission), etc are selected.

(4) Once the Mode Instruction has been written into the 8251A, the command instruction is inserted, enabling the Receiver and Transmitter. See fig 4.2 below for the Mode instruction format and the Command Instruction format for the 8251A.

For this report, the 8251A is programmed to satisfy the following requirements:

1. Type of transmission: asynchronous
2. Number of stop bits: 1
3. Length of character: 8
4. Baud rate factor: 16
5. Parity enable: even

The initialization sequence that will enable the 8251A to satisfy the stated requirements is as follows:

(1) XRA A ; set the Accumulator to zero
OUT A1H ; Reset the 8251A, A1H is the 8251A address (see table 3.9)
(2) MVI A, 40H ; Internal Reset of the 8215A returns the USART to MODE INSTRUCTION format.

This is from the following

EH IR RTS ER SBRK RXE DTR TXEN

\[ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \] = 40H

Command Instruction Format

(See fig 4.2(b))

OUT A1H

3. MVI A, 7EH ; Mode Instruction from the mode Instruction format

OUT A1H (See fig 4.2(a))

4. MVI A, 05H

; Command Instruction (See fig 4.2(b))

\[ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \]

OUT A1H
4.4 PROGRAMMING THE 8253

The 8253, the Programmable Interval Timer, must first be initialized by software. The CPU sends a set of control words to initialize the selected counter together with the desired mode of operation.

![Diagram showing mode instruction format for 8251A](image)

Fig 4.2 (a): Mode Instruction Format for 8251A
Fig 4.2 (b): Command Instruction Format for 8251A

Each counter is programmed individually by writing a corresponding control Word into the control register. The Control Word format for the 8253 is shown below.
The programming sequence for the 8253 is as follows:

(1) First write a Mode Word to the Mode Word address selecting a counter (either counter 0, 1, or 2), the Read/Load sequence, the type of counter (either binary or decade) and the mode (either mode 0, 1, 2, 3, 4 or 5).
(2) The counter is loaded with the count value. Depending on the Read/Load sequence, the low order byte is written to the selected counter first and then the high order byte is loaded next.

The 8253 has been programmed for the following:

1. Counter 1 is selected (chosen arbitrarily)

2. The Read/Load sequence is load LSB first then MSB (this read/load sequence is chosen because the counter value is up to 2 bytes i.e. $0121_{16}$)

3. Decade counting

4. Mode 3 (chosen because we want 8253 to generate square wave (i.e. baudrate) for the 8251A RxC & TxC clocks.

The initialization sequence that will enable the 8253 to satisfy the stated requirements is as follows:

(1) XRA A ; set the accumulator to zero
OUT E1H ; reset counter 1 E1H is counter 1 port address
(2) MVI A, 00110111 ; mode word which is from the following
SCI SCO RLI RLD M2 M1 M0 BCD
0 1 1 1 0 1 1 1

OUT E3H ; write mode word to control port E3 is the
control port address

(3) MVI A, 21H ; write low order byte of counter value into
the LSB of counter 1 port. The counter value is
0121H

(4) MVI A, 01H ; write high order byte of counter value into
MSB of counter 1 port
OUT E1H

4.5 THE SUBROUTINES

The KEYTEST subroutine is called on entering the ISR for the keypad.
It reads the keypad buffer and checks to see if the character keyed in is
the start-of-message character (STX = 02H). If it is, it returns a 'yes' to
the ISR so that the ISR goes on to service the interrupt by accepting
characters from the keypad and storing them in the ASCII text buffer. Otherwise, the KEYTEST subroutine returns a 'NO' and the ISR is existed.

- Parameters passed to it: none
- Returns: "YES" or "NO" in the Accumulator
- Calls: none
- Called by: the KEYPROC Subroutine

As stated above, based on whether 'YES' or 'NO' is returned, the KEYPROC Subroutine (the ISR for the keypad) is either passed or existed. It encodes the Hex key combination for a character into the 7-bit ASCII code and stores it in a byte of the ASCII text buffer. It continues to accept, encode and write characters until it detects the end-of-message character (ETX = 03H) then the routine is existed.

- Parameters Passed: Address of the first byte in the ASCII text buffer
- Returns: Number of characters read and encoded from the keypad Buffer.
- Calls: the KEYTEST and the WIPE subroutines.
- Called by: ISR for the keypad

The CONVERT subroutine produces the graphical bitmaps of the character codes in the ASCII text buffer by referring to the bitmap table in ROM. It stores the bitmaps in the display buffer in RAM.
Parameters passed : a byte from ASCII text buffer in the Accumulator

returns : Number of bytes written in the DISPLAY RAM

calls : None

Called by : The DISPLAY subroutine

The DISPLAY subroutine carries out the multiplexing of the display by turning off the currently displayed row, compiling the column data for the next row down and then turning that row on. It continues to do this until it turns on the 7th row and the appropriate columns for the row. At this point, a complete frame has been finished. This frame is repeated for a short while (statically).

Parameters passed : (a) address of 1st byte of ASCII text buffer

(b) address of 1st byte of DISPLAY RAM

(c) Number of bytes written in the display RAM

returns : None

calls : CONVERT Subroutine

called by : Main program

After the static display of each complete frame for a while, the ANIMATE subroutine is called. The subroutine controls the scrolling ('movement') of the displayed message on the display panel (the LED matrix). It is programmed to scroll left.
Parameter passed:  (a) Address of the first byte of ASCII text buffer  
                 (b) Address of the first byte of the display RAM  

Returns: None  

Calls:  
        (a) MODIFY Subroutine  
        (b) INITCOLCTR  

Called by: Main Program  

The flowcharts for the different subroutines are shown in Fig 4.4. Appendix 1 shows the character set supported by this system and their corresponding ASCII Hex code. These ASCII Hex codes are the key combinations to be pressed on the keypad for each character to be entered. Appendix 2 shows the format for the display of each character on the LED matrix.

4.6 BRIEF SEQUENCE OF OPERATION OF THE SOFTWARE

On entering a hex character (i.e keying in a character) from the keypad, the code for that character appears on the output pins of the keypad buffer and is read and converted directly into its ASCII code equivalent. ASCII code only uses seven bits, out of eight available bits for a byte in the 8085.

The left most or high order bit is always zero, that is the 8th bit (bit 7) is always zero. Each character is represented by two hex digits (see appendix 1) e.g. 41H is the ASCII hex code for the character A. (See Table 4.1 below).
<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII hex Code</th>
<th>ASCII binary code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>41 H</td>
<td>01000001</td>
</tr>
<tr>
<td>T</td>
<td>54 H</td>
<td>01010100</td>
</tr>
<tr>
<td>E</td>
<td>45 H</td>
<td>01000101</td>
</tr>
<tr>
<td>S</td>
<td>35 H</td>
<td>00110101</td>
</tr>
</tbody>
</table>

**Table 4.1:** Table showing the ASCII hex code and corresponding binary code for a few characters

To key in the character A, for example, first the hex character 4 is keyed in and the keypad scanner/encoder output is shown in *Table 4.2*. From the figure, when the key for 4 is pressed, the binary value

```
A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 ← input pins of the buffer
0 0 0 0 0 0 1 0 0
```

appears on the input pins of the 8286 buffer. When the buffer is enabled; the binary value appears on the output pins and on to the data bus.

This value is read by the microprocessor into the Accumulator. Rotation operation are performed
<table>
<thead>
<tr>
<th>Row Selected</th>
<th>Row selection</th>
<th>Column Selection</th>
<th>Column Selected</th>
<th>Hex character Keyed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0</td>
<td>0 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0 0 0</td>
<td>0 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0 0 0</td>
<td>1 0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0 0 0</td>
<td>1 1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
<td>0 0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
<td>0 1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
<td>1 0</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
<td>1 1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>1 0</td>
<td>0 0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>1 0</td>
<td>0 1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>1 0</td>
<td>1 0</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>1 0</td>
<td>1 1</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>1 1</td>
<td>0 0</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>1 1</td>
<td>0 1</td>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>3</td>
<td>1 1</td>
<td>1 0</td>
<td>2</td>
<td>E</td>
</tr>
<tr>
<td>3</td>
<td>1 1</td>
<td>1 1</td>
<td>3</td>
<td>F</td>
</tr>
</tbody>
</table>

**Table 4.2:** Keypad Scanner/encoder output and the corresponding keypad button (Hex character) pressed.

On this value in the accumulator. The aim of this rotation operation is to shift the output from the keypad encoder, that is the first 4 bits of the Accumulator (the LSB) into the MSB and set the LSB to zero. This is done as follows:
(Acc)           0 0 0 0 0 1 0 0

Note: (Acc) Contents of the accumulator

(Acc) after the 1st rotation    0 0 0 0 1 0 0 0

(Acc) after the 2nd rotation    0 0 0 1 0 0 0 0

(Acc) after the 3rd rotation    0 0 1 0 0 0 0 0

(Acc) after the 4th rotation    0 1 0 0 0 0 0 0

keypad encoder output

NOTE: (Acc) contents of the Accumulator

The value of the Accumulator at this point is stored in another register say B. Then the keypad buffer is read for the keypad encoded value for the second hex character, 1. The binary value 00000001 is read into the Accumulator. This is not rotated. The contents of the Accumulator is then ORed with the contents of Register B. i.e. (Acc) OR (B)

(Acc) 0100 0000
(reg B) 0000 0001

(Acc) 0100 0001

keypad encoder output for the hex Xter 4.

At this point, the Accumulator contains the ASCII binary code equivalent for the character A. It is this ASCII binary code that is stored in the ASCII text buffer.

When the character A is to be displayed, the CONVERT subroutine reads the value 01000001 from the ASCII text buffer, checks and sees
that it is the ASCII binary code for the character A, and produces the corresponding graphical bitmaps by referring to the bitmap table in ROM. These bitmaps are stored in seven consecutive bytes in the DISPLAY RAM. The bitmaps are actually the seven byte of column data for the seven rows of the display for the character.

For the character to be displayed, the column latch to be addressed has to be determined based on the current value of the column counter (the column counter is used to note the number of columns already activated and to determine the column on which the display of the next character is to start). For example, from Fig 4.4, to display the character K, column latch number 3 will be addressed and to display A column latch numbers 3 and 4 will be addressed.

![Diagram showing column latches and bitmaps]

Fig 4.4: Display of the message 'CHIKA'

The column latch to be addressed is determined as follows:

- IF character-just-displayed = "I" (or ":" or ".") then

  increment column counter by 4 (3 columns for displaying "I" and one column for space before the next character)
ELSE

    Increment column counter by 6

NOTE:

The value of the column counter is the column number from which the next display will begin. Let colctr represent the column counter.

IF colctr ≤ 4 then address latch 1
ELSE
    IF 9 ≤ colctr ≤ 12 then address latch 2
    IF 17 ≤ colctr ≤ 20 then address latch 3
    IF 25 ≤ colctr ≤ 28 then address latch 4
    IF 33 ≤ colctr ≤ 36 then address latch 5
    IF 41 ≤ colctr ≤ 44 then address latch 6
    IF 49 ≤ colctr ≤ 52 then address latch 7
    IF 57 ≤ colctr ≤ 60 then address latch 8
ELSE
    IF 5 ≤ colctr ≤ 8 then address latches 1 and 2
    IF 13 ≤ colctr ≤ 16 then address latches 2 and 3
    IF 21 ≤ colctr ≤ 24 then address latches 3 and 4
    IF 29 ≤ colctr ≤ 32 then address latches 4 and 5
    IF 37 ≤ colctr ≤ 40 then address latches 5 and 6
    IF 45 ≤ colctr ≤ 48 then address latches 6 and 7
    IF 53 ≤ colctr ≤ 56 then address latches 7 and 8
ELSE
    IF colctr ≥ 61 or Colctr = 62 then
        IF character-to-be-displayed = "l" OR ":" or "." Then address latch 8
        ELSE address latch 1 (i.e start the display from the first column)
        Set Colctr to 0
ELSE address latch 1

Set Colctr to 0

For the cases where there is need to address 2 latches in order to write column data, the column data is manipulated as follows:

For example, from fig 4.6, to write the column data for the character A in the message "Chika", latches 3 and 4 have to be addressed. For the character A, the column data to be written are as shown below:

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>00100000</td>
<td>20H</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>01010000</td>
<td>50H</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>10001000</td>
<td>88H</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>10001000</td>
<td>88H</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>11110000</td>
<td>F8H</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>10001000</td>
<td>88H</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>10001000</td>
<td>88H</td>
</tr>
</tbody>
</table>

This is achieved as follows:
(1) We use the table

<table>
<thead>
<tr>
<th>Column Latch</th>
<th>Last Column on the latch starting from the 1st column on latch 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>48</td>
</tr>
<tr>
<td>7</td>
<td>56</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
</tr>
</tbody>
</table>

(2) We calculate a value $R$ based on the current value of the column counter and the latch to be written to as follows:

<table>
<thead>
<tr>
<th>Latch</th>
<th>Value of $R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(8 - \text{current value of column counter} + 1 = R)</td>
</tr>
<tr>
<td>2</td>
<td>(16 - \text{current value of column counter} + 1 = R)</td>
</tr>
<tr>
<td>3</td>
<td>(24 - \text{current value of column counter} + 1 = R)</td>
</tr>
<tr>
<td>4</td>
<td>(32 - \text{current value of column counter} + 1 = R)</td>
</tr>
<tr>
<td>5</td>
<td>(40 - \text{current value of column counter} + 1 = R)</td>
</tr>
<tr>
<td>6</td>
<td>(48 - \text{current value of column counter} + 1 = R)</td>
</tr>
<tr>
<td>7</td>
<td>(56 - \text{current value of column counter} + 1 = R)</td>
</tr>
<tr>
<td>8</td>
<td>(64 - \text{current value of column counter} + 1 = R)</td>
</tr>
</tbody>
</table>

(3) Latch 3 is 1st written to, that is we write the data 11 to latch 3 starting from column 7 of latch 3. This implies that we need to write the data 00000011 to latch 3 but the data we have in the accumulator is
1111000  i.e.  (Acc) = 1111000 [(Acc) implies contents of the accumulator] what we do is to rotate the contents of the accumulator right 8 - R times through carry.

To write to latch 3 starting from its 7th column implies that the current value of the column counter, colctr = 23 therefore:

\[ R = 24 - \text{Colctr} + 1 \]
\[ = 24 - 23 + 1 \]
\[ R = 2 \]

So we rotate right 8 - 2 = 6 times through carry. Each bit that is rotated out in reset to zero. The rotation is done as follows:

(1) reset the carry flag to zero so that

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
\text{C flag} & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
\hline
\end{array}
\]

(2) 1st rotation

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
\text{C flag} & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
\hline
\text{(Acc)} & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
\hline
\end{array}
\]

check if the carry flag is set. If set then reset it again, reset the carry flag to zero otherwise continue so that

(3) 2nd rotation

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
\text{C flag} & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
\hline
\end{array}
\]

check if the carry flag is set if set then reset it.
(4) 3rd rotation

\[
\begin{array}{c|cccccc}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

(5) 4th rotation

\[
\begin{array}{c|cccccc}
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

The carry flag is set so it is reset to zero so that

\[
\begin{array}{c|cccccc}
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

(6) 5th rotation

\[
\begin{array}{c|cccccc}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

The carry flag is set so it is reset so that

\[
\begin{array}{c|cccccc}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

(7) 6th rotation

\[
\begin{array}{c|cccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]

The carry flag is set so it is reset so that

\[
\begin{array}{c|cccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]

i.e after the 6th rotation \(\text{(Acc)} = 00000011\) at this point, latch 3 is addressed and the data 00000011 written to it.

Next the accumulator is set to its original value \(\text{(Acc)} = 1111000\) having written the MSB i.e. \(D_7D_6 = 11\) to latch 3 starting from col 7 of latch 3 we now need to write \(D_5D_4D_3D_2D_1D_0 = 111000\) to latch 4 starting from the 1st column of latch 4. So we rotate the contents of the accumulator, \(\text{(Acc)} = 11111000\), left, \(R = 2\) times through carry. The rotation is as
shown below

We have

<table>
<thead>
<tr>
<th>Carry flag</th>
<th>(Acc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 1 1 1 1 0 0 0 0</td>
</tr>
</tbody>
</table>

(1) 1st rotation

| 1          | 1 1 1 1 0 0 0 0 0 |

The carry flag is set, so it is reset so that

| 0          | 1 1 1 1 0 0 0 0 0 |

(2) 2nd rotation

| 1          | 1 1 1 0 0 0 0 0 0 |

The carry flag is set, so we reset so that

| 0          | 1 1 1 1 0 0 0 0 0 |

After the 2nd rotation

(Acc) = 11100000

at this point, latch 4 is addressed and the data 11100000 written to it.

Appendix 4 shows the software listing.
Fig 4.5 (a) Main program loop flowchart

Keytest sub/v

Fig 4.5(b) Keytest sub/v flowchart
**KEYPROC Sub/r**

Start

- Read 1st Hex xter code into Acc
- Rotate it left 4 times
- Reg B ← (Acc)
- Read 2nd Hex xter code into Acc
- Reg C ← (Acc)
- Acc ← (Reg B)
- (Acc) V (Reg C)

- Store in the ASCII text buffer
- ASCII code = end-of-message xter?

Ret

---

**Display Sub/r**

Start

- ctr = 1024
- rowctr = 0
- colctr = 0
- # of xters = 0

---

Fig 4.5(c): KeyPROC subroutine flowchart
The convert subroutine produces the bitmaps for the character read from the ASCII text buffer. It stores the bitmaps in the display RAM and saves the value of the number of characters in the buffer.

Each byte of display RAM contains the column data for the characters to be displayed.

The byte read from the ASCII text buffer is the xter code.
Increment colctr by 4

xter = :?

xter = :

Increment colctr by 6

Based on the value of the colctr, determine which of the 8 column latches to address for the display of the next xter

61 ≤ colctr ≤ 62?

yes

Select column latch 8

no

colctr ≥ 63?

no

yes

last byte of display RAM was read

yes

no
Fig 4.5(d): Display subr flowchart
Convert subroutine

This subroutine produces the graphical bitmaps of the character codes in the ASCII text buffer by referring to the bitmap table in the ROM.

Parameters to be passed to the routine: address of 1st byte of display RAM

Returns: Column data for the characters to be displayed in the display RAM; and also of the display RAM # of byte in display return written LAST BYTE

Start

Read a byte of the ASCII text buffer

end-of-message character?

no

xter = A?

yes

goto ROM address AA and read 7 bytes and store in display RAM

no

xter = B?

yes

goto ROM address BB and read 7 bytes

no

26 xters

xter = Z?

y

Go to ROM address ZZ and read 7 bytes

xter = O?

y

Go to ROM address A1 and read 7 bytes

26 xters

10 xters

10 xters

A

B

C
Fig 4.5(e): Convert subroutine flowchart

Animate subr

This routine is called after the display subroutine has finished each complete frame. This subroutine controls the scrolling of the messages on the display panel.
Animate Sub/v

Start

BEGINSCROLL
Xterout = 0

Read a byte of ASCII text

NEXT ASCII = address of Next byte of ASCII text

Scroll xter = the ASCII xter just read

Modctr = 0
Xternum = 0
rowctr = 0

end-of-message

Increment ctr

write 1st 7 addresses of display RAM into MODRANGE

Call MODIFY sub/r

Modctr = Modctr + 1

Send row data to energize the 1st row

rowctr = rowctr + 1

XEND

xterout keeps track of the number of characters that have be scrolled out

is used to keep track of the number of characters whose column data for the current row have been displayed

keeps count of the 5 columns in each character 5 x 7 matrix

modctr is used to keep track of the number of times modification has been done on the column data for the xter that is currently being scrolled out
Read 1st byte of display RAM for the row

Write column data to the 1st Column latch

Call Initcolctr

Colctr = Colctr + coliner

last byte of display RAM read?

yes

no

Determine the column latch to write next to based on the value of the colctr

Increment current address of display RAM by 7

Read next byte of display RAM

Write column data to the appropriate column latch

Read a byte of ASCII

is xterm I or : or ..

yes

Increment colctr by 4

no

Increment colctr by 6
Increment by 1

Address 1st byte of display RAM to be read for this Row = Display RAM + ROWCTR

Send row data to the appropriate row

rowctr = rowctr + 1

rowctr ≤ 6 ?

no

Ctrl = Ctrl + 1

is the scroller I or : or ?

no

Ctrl ≥ 5 ?

yes

Xterout = Xterout + 1

Xterout = maxixter ?

no

BEGINSCROLL

first display RAM address + 7 * xterout

Read next ASCII xter to be scrolled out

92

P
Decrement REPEATT by 1

REPEATT \leq 1024 ?

XEND

RET

Fig 4.5(f): ANIMATE Subroutine flowchart
Fig. 4.5 (g): Flowchart for the ISR for PC Input
CHAPTER 5
IMPLEMENTATION/TESTING/RESULTS

5.1: IMPLEMENTATION

The circuit drawing of appendix 2 consists of the following modules: the memory module (RAM, EPROM, and the memory decoder) on the top right hand side; the PC interface module (USART, RS 232C, programmable interval timer and the line driver and receiver) on the top left hand side; the CPU module in the center left; the keypad interface module (Hexkeypad, keypad encoder, and keypad output buffer) on the down left hand side; and the display module (LED matrix, row switch circuits, row latch, column latches, column latch select decoders) on the down right hand side. Complete system implementation was not possible because of difficulties in obtaining components. However, very serious effort was made to implement and test a section of the circuit. The display module or unit has been physically implemented to demonstrate the system's workability. In the design, a 22-character message display system is described which consists of a 7 x 128 LED matrix = 896 LEDs. However, for the purposes of implementation, a 3-character (including spaces) message display which consists of a 7 x 18 LED matrix = 127 LEDs is constructed.

The first stage of the implementation was to insert the 127 LEDs. It was ensured that they were inserted in the correct sequence, (one column at a time starting at the end where the power transistors will be positioned. It was also ensured that the LEDs were inserted the correct way round.

Fig. 5.1: LED Pin Identification
After the insertion of the LEDs, the legs of the LEDs were not cut off at this point.

After soldering in the first column of seven LEDs, a battery and a resistor were connected in series, the negative (−ve) battery terminal was attached to the column track on the board. The positive (+ve) battery terminal was attached via the resistor to each LED anode (leg a in Fig. 5.1) in turn, making sure they all lit up. After testing the LEDs in turn, the anode legs were cut off while each of LED cathode leg (leg k in Fig. 5.1) were not cut off because they were required to support the column wires. Additional columns of LEDs were fitted until all 18 columns were completed. When soldering the LEDs, the printed circuit board (PCB) was held face down on a piece of foam rubber, this is in order to get them to lie as flat a possible against the board.

The 18 column wires were inserted next. But instead of using a copper wire to weave in and out of each of the 18 columns of 7 LED legs to make up each column, the LED cathode legs themselves, in each column, were bent above the board and soldered together to make up the columns. The two legs at both end of each column were bent inwards by weaving them in and out of the other LED legs until they touched the leg in the middle or center of the column. All the legs were then soldered together and the excess ones protruding above the columns were then cut off.

Having fixed the LEDs in place, the remaining components were now put in place starting with the 18 current limiting resistor R1 to R18 which are below the LED matrix. Then the sockets for the various latches and buffer chips were inserted and soldered. When soldering precaution is taken not to apply the hot soldering iron to the terminals or pins of the
components for a long time to avoid component damage. For the R1 to R128 resistors (only R1 to R18 is used in the implementation) which value was calculated to be 100 ohms and its power dissipation to be 1 Watt (see page ), it was discovered 1 Watt resistors are fairly large and expensive. Since 18 of such resistors were to required, it was decided that a smaller one be used (5V) and then the wattage was recalculated as follows:

\[
V = 5V \quad R = 100 \text{ ohms} \\
P = \frac{V}{R} = \frac{(V/R)^2}{R} = \frac{V^2}{R} \\
= \frac{5}{100} = \frac{1}{4} \\
= 0.25 \text{ Watts}
\]

This implies that one quarter Watt, 100 ohm resistors were now used.

The major problem associated with the use of smaller wattage resistor in the circuit is that the resistors might get very hot but this is not the case because since not all the LEDs in each column will be ON at the same time, the resistors are unlikely to get hot. Moreover, each LED in a column will only be on for one seventh (1/7th) of the frame time.

Since the complete system circuit could not be implemented, the implementation of the display unit for subsequent testing by simulating the input signals involved the taking of the following steps:

- It was ensured that the pieces of wire cut off from the tips of the connecting wires or from the LED legs did not fall into the project board to avoid shorting the circuit.
After soldering each LED to the PCB, the two legs of the LED were tested for signal continuity. If there is signal continuity, then the two legs are touching somewhere which should not be the case. The touching is usually through the soldering lead and this has to be corrected before soldering in the next one.

The HCT devices (IC3a to c, 74HCT245 and IC1 to IC3, 74HCT574) could not be obtained locally here so they were replaced by their LS equivalent (that is 74LS245 and 74LS574) instead. This was possible since the HCT devices were intended as signal transceivers to convert the TTL signal from the microprocessor to CMOS compatible signal for the column drive, Circuit.

To enable the testing of the circuit, the following steps were taken (See Fig 5.2).

1. Pins 5(G2A) and 4 (G2B), the chip select inputs of IC36a, were connected to OV (that is grounded). Pin 6 (the G1) input is tied high. This is to keep the decoder permanently enabled. Three switches, SWA, SWB and SWC are connected to pins 3, 2 and 1 respectively so that these input pins can be manually set with any combination of input signals depending on the column latch to be selected. For example:

<table>
<thead>
<tr>
<th>SWC</th>
<th>SWB</th>
<th>SWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed</td>
<td>Closed</td>
<td>open</td>
</tr>
</tbody>
</table>
FIG 5.2: Pin Connection Modification for Simulation and Testing of the Display Unit
Implies that Pin 1 = OV, Pin 2 = OV and Pin 3 = 5V and IC1 is
clocked to latch the data on the output pins of IC34C.

2. IC33 is permanently enabled by connecting its pin 1 (\(\overline{CE}\)) to OV.

3. IC34C is not permanently enabled. Instead a switch (SW8) is
connected to the chip enable (\(\overline{CE}\)) input, pin 19, to manually enable
the IC to latch the data on the data bus when column data is to be
written to the column latches.
4. Eight switches, SW0, SW1, to SW7, are connected to the data bus D0, D1 to D7 respectively so that the data bus can be manually set with data as desired by closing (when logic 0 is desired) any if the switches or leaving them open (when logic 1 is desired). The switches are used to simulate data and input signals.

5.2 TESTING

5.2.1 Testing the Display Unit

5.2.1.1 Testing the Row and Column Switches

The row and column switches were first tested to ensure that they are working before testing the display unit as a whole by simulating the input data. During construction, a simple test of each of the LEDs and each of the LED columns has already been done. The row and column switches, Fig. 5.3, are tested by connecting a 5V power supply to the power connector, ensuring that the negative and positive connections are the correct way round. A 5V power supply is used to avoid damage to the LEDs from constant illumination during the testing. Two pieces of wire were connected to the positive side of the 5V power supply which is used to turn on combinations of row and column switches. One end of one of the wires is attached to any one of the input pins (pins 1 to 8) of any of the column switches (IC17 to IC19). The free end of the other wire is used to touch any one of the row switches at point A. Any pair of row and column switches can be tested in this way.
FIG. 5.3: Row and Column Switch Circuits for the LED Matrix
5.2.1.2: Testing the Complete Display Unit

The display unit is tested using Fig. 5.2 as follows:

- Row data is put on the data bus manually using switches SW0 to SW7. For example to energize row 0, the data bus is set with data as follows:

<table>
<thead>
<tr>
<th>SW7</th>
<th>SW6</th>
<th>SW5</th>
<th>SW4</th>
<th>SW3</th>
<th>SW2</th>
<th>SW1</th>
<th>SW0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

0 => Switch in closed
1 => Switch in open

So that the data bus contains the data

\[
D7 \quad D0 \\
00000001
\]

- IC33 is clocked to latch the data on the data bus
- Column data is put on the data bus. For example, to energize column 1 the data bus is set with the data

\[
D7 \quad D0 \\
00000001
\]

and to energize column 2, the data

\[
D7 \quad D0 \\
00000001
\]

is put on the data bus.

- IC34c is enable, by closing switch SW8 to latch the column data on the data bus and the data immediately appears on its output pins.
IC36a, being permanently enabled, has its input pins 1, 2 and 3 reset OV, OV, Ov, respectively so that the column latch 1 (IC1) is selected. (clocked) through output pin 11 of IC34a, which is permanently enabled also so that the inputs from the output pins of IC36a immediately appear on it output pins.

IC1, on being closed, latches the column data on the output pins of IC34C and the appropriate column is energized through the column switch circuit of IC17.

5.2.2: Testing the Communications Program for downloading Messages from the PC

A communications program for downloading messages from the PC for display on the display matrix was completely coded, debugged and run in GW Basic. The listing of the program is shown in Appendix 7.

The following steps were taken for the coding of the program:

1. Serial communications parameters where chosen. These are
   Baudrate : 1200
   Number of data bits : 8
   Parity : None
   Number of stop bits : 1

2. The next major step was to initialize a buffer and then access a communication file for data output. The block diagram for this is shown below.
The transmit buffer does not necessarily have to be a disk file, it could be an array.

3. A communications port was selected. Com2 was selected instead of Com1 because the serial mouse is usually connected to Com1: Com1 therefore, might not be available.

4. The OPEN COM statement assigns a buffer for the communication purposes and sets up the communications parameters.

5. Data can then be transmitted through the assigned buffers.

The communications program on being started displays the startup message and offer three choices of either keying in a message; downloading a message from the five already stored messages; or exiting the program. If a message is to be keyed in, the message is accepted from the keyboard and each character in the message is checked to ensure that it is supported by the matrix message display system. The keyed in message is then downloaded through the communications interface.
For the testing of the program, the following items are required:

- A PC with
- Com2 port installed
- an RS-232C female port
- a Hard disk with GWBasic Installed (This is optional since the GWBasic program can be on a floppy diskettes)
- An RS-232C (with a male connector) with wires from pins 1 to 7
- The circuit diagram of Fig 5.5

The test is done as follows:

a. Connect the RS-232C (male connector) to the PC's RS-232C female port with the wires from pins 1 to 7 connected to the circuit of Fig. 5.5.

b. Connect the circuit of Fig. 5.5 to the appropriate power supplies (12V for the line driver, MC1488 and 5V for the line receiver, MC1489).

c. Switch on the PC and start GWBasic by finding the directory in which the GWBasic program is stored. Change to that directory and at the prompt just type d GWBasic.

d. Run the communications program (ensure the keyboard is in Upper case by putting caps lock on).
FIG 5.5: Circuit for testing the Communications program.
e) At the prompt for choice of action, type M to key in a message, alternatively type N to download one of the five stored messages.

f) If a message is keyed in, the message is parsed by the program to remove the characters not supported by the Display System and then the parsed message is echoed on the screen with blanks replacing the characters not supported by the system. If a stored message is selected, the selected message is echoed on the screen. The choice of either re-entering a message from the keyboard, re-selecting a message from the library, continuing or Exiting the program is given. Select continue by Typing Z.

g) The message is the downloaded through the interface to the circuit.

5.3 RESULTS

5.3.1 Result from Testing the Row and Column Switches

When testing the row and column switches, on using the wires connected to the positive side of the power supply to touch a pair of row and column switches, one of the LEDs lit up. On attaching another wire to the positive side of the power supply and using it together with the other two wires, and touching a row switch and any two input pins of the column switch, two LEDs lit up. So also, when more wires were connected and used to touch more than one row switch and more than one input pin of the column switch (ULN2803A), a number of LEDs in more than one row lit up.
5.3.2 Result from testing the complete display circuit

Practically the same results were obtained as in section 5.3.1 above. It was possible to light up any LED on any row and any column by putting the appropriate row and column data on the data bus and enabling the row and column latches to latch the data as appropriate.

5.3.3 Result from Testing the Communications Program

In Fig. 5.5., the RTS signal from the PC (output pin 4 of IC1) is used to clock IC5. Two input pins of IC5 are tied high to represent input data so that when an RTS signal is received, IC5 is clocked and the input data is latched and appears on the output pins. With the output pins high, the LED, connected to the output pins lit up thereby indicating that the PC has actually requested to send data.

The RTS output signal from the interface is fed back to the CTS (pin 5 of IC1) input of the interface to enable the PC to start sending data immediately the RTS signal is received. On receiving the CTS signal, the PC downloads the message keyed in from the keyboard or the message selected from one of the stored messages through the interface. The transmitted data output (pin 2 of IC1) is used to chip-enable IC4 (It was ensured that a string with a number of consecutive 1s is its binary form was transmitted to enable the clocking of IC4). When IC4 is clocked, the data on its input pins are latched and appear on the output pins. Consequently, the LEDs lit up indicating that data has actually been downloaded from the PC through the interface.

Summarily, it can be safely stated that the section of the circuit diagram of the microprocessor-Based LED Message Display System that was physically implemented has been tested and found to be working correctly. Also the PC interface and the coded communications program have equally been tested and found to working.
CHAPTER 6

CONCLUSION/RECOMMENDATIONS

In conclusion, it is evident that electronic displays have come to stay. They are just like octopus with tentacles extending to virtually every aspect of the society. They are in car dashboards, audio systems, electronic billboards and scoreboards, to name but a few. Consequently, a project report involving electronic display of messages is appropriate in view of the fact that they have very high demand and have very many uses.

The choice of display that is most suited for a particular application will depend on many parameters. These parameters are a function of the systems requirements such as size, cost, power consumption and so on. Also, the kind of information and data (numerals, alphabets, alphanumerics and graphics) to be displayed will determine the display technique to be used which in turn affects the choice of the display drive technique. The choice of the display drive control (hardwired control or microprocessor based control) largely depends on the designer.

The use of hardwired control logic will be based on such requirements as very fast response and modular design features without minding the inflexibility that will be introduced into the system. Inflexibility is introduced in the sense that modification can only be made to the system by rewiring the whole system. On the other hand, the use of microprocessor based control will depend on such requirements as flexibility, this is brought about by the software, and fewer chip count which in effect reduces the overall system cost.
Moreover, the modular design of the software goes a long way to simplify the overall systems design and implementation. The way characters and messages are displayed on the display panel (that is display fonts) and the way characters and messages are made to move (that is "animation") is only limited by software. When there is need to add or remove any feature, all that is required is software modification without having to rewire the hardware. This is why it is said that microprocessor-based systems are highly flexible.

Before finally concluding, there is need to make a few recommendations and indicate some areas of further work. Such areas of further work include:

- the design was for a 22-character display panel but the implementation was for a 12-character display. One might want to implement the system in such a way that up to 22 or more characters can be displayed simultaneously. This will involve an increase in the number of LEDs in the LED matrix and a corresponding increase in the column latches and column switches.

- A Hex keypad was used in this design. The use of an alphanumeric keypad can be worked upon

- the incorporation of more character display fonts can be attempted.

- the ability to scroll up, down, left or right instead of the present implementation of being able to scroll left only.
in-built messages can be included from which users can select messages for display.

the ability to key in a new message from the keypad while a previous message remains on the display panel. In this case, the main program loop will represent a simple form of multi-tasking software.
REFERENCES


18. Thomas L. Floyd, Electronic Devices, Charles E. Merrill, Columbus, Ohio, 1984.


<table>
<thead>
<tr>
<th>Character</th>
<th>Hexadecimal Equivalent (ASCII hex Code)</th>
<th>Character</th>
<th>Hexadecimal Equivalent (ASCII hex Code)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>41</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>B</td>
<td>42</td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>C</td>
<td>43</td>
<td>7</td>
<td>37</td>
</tr>
<tr>
<td>D</td>
<td>44</td>
<td>8</td>
<td>38</td>
</tr>
<tr>
<td>E</td>
<td>45</td>
<td>9</td>
<td>39</td>
</tr>
<tr>
<td>F</td>
<td>46</td>
<td>]</td>
<td>5B</td>
</tr>
<tr>
<td>G</td>
<td>47</td>
<td>[</td>
<td>5D</td>
</tr>
<tr>
<td>H</td>
<td>48</td>
<td>?</td>
<td>3F</td>
</tr>
<tr>
<td>I</td>
<td>49</td>
<td>:</td>
<td>3A</td>
</tr>
<tr>
<td>J</td>
<td>4A</td>
<td>.</td>
<td>2E</td>
</tr>
<tr>
<td>K</td>
<td>4B</td>
<td>*</td>
<td>2A</td>
</tr>
<tr>
<td>L</td>
<td>4C</td>
<td>%</td>
<td>25</td>
</tr>
<tr>
<td>M</td>
<td>4D</td>
<td>+</td>
<td>2B</td>
</tr>
<tr>
<td>N</td>
<td>4E</td>
<td>-</td>
<td>2D</td>
</tr>
<tr>
<td>O</td>
<td>4F</td>
<td>=</td>
<td>3D</td>
</tr>
<tr>
<td>P</td>
<td>50</td>
<td>space</td>
<td>60</td>
</tr>
<tr>
<td>Q</td>
<td>51</td>
<td>STX</td>
<td>02</td>
</tr>
<tr>
<td>R</td>
<td>52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>54</td>
<td>(start of message)</td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>55</td>
<td>ETX</td>
<td>03</td>
</tr>
<tr>
<td>V</td>
<td>56</td>
<td>(end of message)</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>5A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
; MAIN PROGRAM
; DEFINE AND INITIALIZE COUNTERS AND 
; VARIABLES
BMEXI EQU 200FH
BXTERNUM EQU 28H ;(40 XTERS)
FIRSTBYTE EQU 4000H ;FIRSTBYTE OF ASCII TEXT BUFFER
DISPRAM EQU 8000H ;FIRSTBYTE OF DISPLAY BUFFER
ROWLATCH EQU FFF7H ;ROWLATCH PORT ADDRESS
ROW0 EQU 01H ;DATA TO BE WRITTEN TO ROWLATCH TO ACTIVATE ROW 0
ROW1 EQU 02H ;DATA TO BE WRITTEN TO ROWLATCH TO ACTIVATE ROW 1
ROW2 EQU 04H ;DATA TO BE WRITTEN TO ROWLATCH TO ACTIVATE ROW 2
ROW3 EQU 08H ;DATA TO BE WRITTEN TO ROWLATCH TO ACTIVATE ROW 3
ROW4 EQU 01H ;DATA TO BE WRITTEN TO ROWLATCH TO ACTIVATE ROW 4
ROW5 EQU 20H ;DATA TO BE WRITTEN TO ROWLATCH TO ACTIVATE ROW 5
ROW6 EQU 40H ;DATA TO BE WRITTEN TO ROWLATCH TO ACTIVATE ROW 6
COLSET1 EQU FFF8H
COLSET2 EQU FFF9H
COLSET3 EQU FFFAH
COLSET4 EQU FFFBH
COLSET5 EQU FFFCH
COLSET6 EQU FFFDH
COLSET7 EQU FFFE7H
COLSET8 EQU FFFFFH
ZEROS EQU 0000H
XTEROUT DS 1
SCROLLXTER DS 1
MODRANGE DS 7
COLINCR DS 1

ORG 0000H

; initialization of the stack pointer
LXI H, A000H
SPHL
LXI H, FIRSTBYTE
SHLD PCASCII

; initialization of the 8251A
XRA A ;set accumulator to zero
OUT A1H ;reset the 8251A
MVI A, 40H ;INTERNAL RESET OF THE 8251A: RETURN TO MODE I
OUT A1H
MVI A, 7EH ;MODE INSTRUCTION WORD
OUT A1H
MVI A, 05H ;COMMAND INSTRUCTION TO ENABLE THE RECEIVER AN
OUT A1H

;INITIALIZATION OF THE 8253
PIT MVI A, 77H ;MODE WORD SELECTING COUNTER1, DECADE COUNTING
OUT E3H
MVI A, 21H ;MOVE LOW ORDER BYTE INTO THE Acc
OUT E1H
MVI A, 01H ;MOVE HIGH ORDER BYTE INTO THE Acc
OUT E1H ;WRITE TO COUNTER 1

;ENABLE INTERRUPTS
MVI A, 08H ;MASK PATTERN
SIM
EI

;MAIN PROGRAM BODY
CALL COPYME
REPEAT CALL DISPLAY
CALL ANIMATE
JMP REPEAT

;DEFINE STORAGES AND BYTES

LASTBYTE DS 2
NUMB DS 1
TOTAL#S DS 1
COLDATA DS 7
COLUMCTR DS 1
MODCTR DS 1
XTERNUM DS 1
ROWCTR DS 1
STATSCOL DS 2
CTR DS 1
COUNT DS 1
PCASCII DS 2

;DISPLAY SUBROUTINE
DISPLAY EQU §
COUNT EQU 1024D
MVI C, COUNT
SUBLOOP PUSH C
START LHLD DISPRAM
XCHG
MVI C, 00H
LHLD FIRSTBYT
READBYTE LDA M
CPI 03H
JZ STARTDISP
CALL CONVERT
INR C
INX H
JMP READBYTE
MOV E, C
STARTDISP MOV A, C
STA TOTAL#S
MVI B, 00H
;SEND ROW DATA TO THE ROW LATCH TO ENERGIZE THE FIRST ROW I,
FRAME MVI A, ROW0
OUT ROWLATCH
INR B ;INCREMENT ROW COUNTER
;READ FIRST BYTE OF THE DISPLAY RAM
XCHG
LDA M
COLSETA EQU COLSET1
COLSETB EQU 0000H
FAT OUT COLSETA ;WRITE COLUMN DATA TO THE FIRST COLUMN LATCH
OUT COLSETB
DCR E
XCHG
;READ FIRST BYTE OF ASCII TEXT BUFFER
LHLD FIRSTBYT
XCHG
LDAX D
CPI 49H ;CHECK IF IT IS THE 'I' CHARACTER
JZ COLON
CPI 3AH ;CHECK IF IT IS THE ':' CHARACTER
JZ COLON
CPI 2EH ;CHECK IF IT IS THE '.' CHARACTER
JNZ CONTINUE
COLON PUSH E
MVI E, 04H
INR C
CONTINUE

; BASED ON THE VALUE OF THE COLUMN COUNTER, ONE OF THE 8
; COLUMN LATCHES IS SELECTED TO BE ADDRESSED FOR THE DISPLAY
; OF THE NEXT COLUMN DATA

MOV A, C
STA COLMCTR

CPI 40H ; COLCTR VALUE = 64 ?
JZ END1
JM END1
CPI 04H ; COLCTR VALUE < or = 4 ?
JC LATCH1
JZ LATCH1
CPI 09H ; COLCTR VALUE > or = 9 ?
JZ LATCH2
JM LATCH2
CPI 11H ; COLCTR VALUE > or = 17 ?
JZ LATCH3A
JM LATCH3A
CPI 19H ; COLCTR VALUE > or = 25 ?
JZ LATCH4A
JM LATCH4A
CPI 21H ; COLCTR VALUE > or = 33 ?
JZ LATCH5A
JM LATCH5A
CPI 29H ; COLCTR VALUE > or = 41 ?
JZ LATCH6A
JM LATCH6A
CPI 37H ; COLCTR VALUE > or = 49 ?
JZ LATCH7A
JM LATCH7A
CPI 39H ; COLCTR VALUE > or = 57 ?
JZ LATCH8A
JM LATCH8A

CPI 05H ; COLCTR VALUE > or = 5 ?
JZ LATCH12A
JM LATCH12A
CPI 0DH ; COLCTR VALUE > or = 13 ?
JZ LATCH23A
JM LATCH23A
CPI 15H ; COLCTR VALUE > or = 21 ?
JZ LATCH34A
JM LATCH34A
CPI 1DH ; COLCTR VALUE > or = 29 ?
JZ LATCH45A
JM LATCH45A
CPI 25H ; COLCTR VALUE > or = 37 ?
JZ LATCH56A
JM LATCH56A
CPI 2DH ; COLCTR VALUE > or = 45 ?
JZ LATCH67A
JM LATCH67A
CPI 35H ; COLCTR VALUE > or = 53 ?
JZ LATCH78A
JM LATCH78A
LATCH2A CPI 0CH ; COLCTR VALUE < or = 12 ?
JP LATCH2B
LATCH3A CPI 14H ; COLCTR VALUE < or = 20 ?
JP LATCH3B
LATCH4A CPI 1CH ;COLCTR VALUE < or = 28 ?
    JP LATCH4B
LATCH5A CPI 24H ;COLCTR VALUE < or = 36 ?
    JP LATCH5B
LATCH6A CPI 32H ;COLCTR VALUE < or = 44 ?
    JP LATCH6B
LATCH7A CPI 34H ;COLCTR VALUE < or = 52 ?
    JP LATCH7B
LATCH8A CPI 3CH ;COLCTR VALUE < or = 60 ?
    JP LATCH8B
LATCH12A CPI 08H ;COLCTR VALUE < or = ?
    JP LATCH12B
LATCH23A CPI 10H ;COLCTR VALUE < or = ?
    JP LATCH23B
LATCH34A CPI 18H ;COLCTR VALUE < or = 24 ?
    JP LATCH34B
LATCH45A CPI 20H ;COLCTR VALUE < or = 32 ?
    JP LATCH45B
LATCH56A CPI 28H ;COLCTR VALUE < or = 40 ?
    JP LATCH56B
LATCH67 CPI 30H ;COLCTR VALUE < or = 48 ?
    JP LATCH67B
LATCH78A CPI 38H ;COLCTR VALUE < or = 56 ?
    JP LATCH78B

LATCH1 EQU $ ;LAST COLUMN ON THE FIRST LATCH IS 8
LASTCOL EQU 08H
COLSETA EQU COLSET1
COLSETB EQU ZEROS
    JMP BOOT

LATCH2B EQU & ;LAST COLUMN ON THE SECOND LATCH IS 16
LASTCOL EQU 10H
COLSETA EQU COLSET2
COLSETB EQU ZEROS
    JMP BOOT

LATCH3B EQU $ ;LAST COLUMN ON THE FIRST LATCH IS
LASTCOL EQU 18H
COLSETA EQU COLSET3
COLSETB EQU ZEROS
    JMP BOOT

LATCH4B EQU & ;LAST COLUMN ON THE SECOND LATCH IS
LASTCOL EQU 20H
COLSETA EQU COLSET4
COLSETB EQU ZEROS
    JMP BOOT

LATCH5B EQU $ ;LAST COLUMN ON THE FIRST LATCH IS
LASTCOL EQU 28H
COLSETA EQU COLSET5
COLSETB EQU ZEROS
    JMP BOOT

LATCH6B EQU & ;LAST COLUMN ON THE SECOND LATCH IS
LASTCOL EQU 30H
COLSETA EQU COLSET6
COLSETB EQU ZEROS
    JMP BOOT

LATCH7B EQU $ ;LAST COLUMN ON THE FIRST LATCH IS
LASTCOL EQU 38H
COLSETA EQU COLSET7

129
LATCH8B EQU &
LASTCOL EQU 40H ;LAST COLUMN ON THE SECOND LATCH IS
COLSETA EQU COLSET8
COLSETB EQU ZEROS
JMP

LATCH12B EQU $
LASTCOLA EQU 08H
COLSETA EQU COLSET1
COLSETB EQU COLSET2
JMP BOOT

LATCH23B EQU $
LASTCOLA EQU 10H
COLSETA EQU COLSET2
COLSETB EQU COLSET3
JMP BOOT

LATCH34B EQU $
LASTCOLA EQU 18H
COLSETA EQU COLSET3
COLSETB EQU COLSET4
JMP BOOT

LATCH45B EQU $
LASTCOLA EQU 20H
COLSETA EQU COLSET4
COLSETB EQU COLSET5
JMP BOOT

LATCH56B EQU $
LASTCOLA EQU 28H
COLSETA EQU COLSET5
COLSETB EQU COLSET6
JMP BOOT

LATCH67B EQU $
LASTCOLA EQU 30H
COLSETA EQU COLSET6
COLSETB EQU COLSET7
JMP BOOT

LATCH78B EQU $
LASTCOLA EQU 38H
COLSETA EQU COLSET7
COLSETB EQU COLSET8
JMP BOOT

CPI 3DH ;COLCTR VALUE > or = 64?
COT

CPI 3FH ;COLCTR VALUE > or = 63?
JNC END3
JMP BOOT

COT
CPI 3EH ;COLCTR VALUE < or = 62?
JNC AI
JMP COL63

AI
LDAX D ;READ A BYTE OF ASCII i.e. THE ASCII CODE FOR
POLE
INX D ;THE NEXT CHARACTER TO BE DISPLAYED
CPI 49H ;CHECK IF IT IS THE 'I' CHARACTER?
JZ LATCH8
CPI 3AH ;CHECK IF IT IS THE ' ' CHARACTER ?
JZ LATCH8
CPI 2EH ;CHECK IF IT IS THE '.' CHARACTER ?
JNZ END2

LATCH8
EQU $ COLSETA
EQU COLSETB

END2
CPI 3FH ;COLCTR VALUE > OR = 63 ?
JNZ BOOT
JC BOOT
MVI C, 00H
MOV A, C
STA COLUMCTR

;CHECK IF THE LAST BYTE WRITTEN IN THE DISPLAY RAM HAS BEEN ;READ AND DISPLAYED

BOOT
PUSH D
PUSH H
XCHG
LHLD FIRSTBYT
DAD D
JC FADE
POP H
POP D

;INCREMENT CURRENT ADDRESS OF THE DISPLAY RAM BY 7 TO GET ;THE COLUMN DATA FOR THE CURRENT ROW FOR THE NEXT CHARACTER ;TO BE DISPLAYED
PUSH B ;REGISTER B IS THE ROW COUNTER
MVI B, 07H
INX H
dcr B
JNZ TOOL
POP B
dcr E ;DECREMENT THE TOTAL # OF CHARACTERS
JC IFFY
LDA M

;READ A BYTE OF THE DISPLAY RAM TO GET THE COLUMN DATA FOR ;THE NEXT CHARACTER WHICH COLUMN DATA IS TO BE DISPLAYED ON ;ON THE CURRENT ROW
PUSH H
PUSH B
PUSH C
PUSH PSW
;FIRST CHECK IF ONLY ONE OR TWO COLUMN LATCHES ARE TO BE ;SELECTED IN ORDER TO DISPLAY THE CHARACTER
LXI B; COLSETB
;if only one is to be selected, COLSETB = ZEROS OTHERWISE ;IT CONTAINS A VALUE
LXI H, COLSETB
LXI D, COLUMCTR
LDAX D
MOV D, A ;(D) <-- CURRENT VALUE OF THE COLUMN CTR
DAD B ;ADD (B-C) REGISTER PAIR TO (H-L)
JC KIND
MOV C, A
MVI A, LASTCOL
SUB D ;LASTCOLUMN - COLCTR (LASTCOLUMN REFERS ;TO THE LAST COLUMN ON THE SELECTED LATCH)
ADI 01H ;(LASTCOLUMN - COLCTR) + 1 = R
MOV E, A ;STORE THE VALUE R IN REGISTER E
;ROTATE THE COLUMN DATA RIGHT 8 - R TIMES THROUGH CARRY. IF
;ROTATE THE COLUMN DATA RIGHT 8 - R TIMES THROUGH CARRY. IF
;THE CARRY FLAG IS SET AFTER THE SHIFT, IT IS RESET BEFORE
;THE NEXT SHIFT
MVI A, 08H
SUB E                ;SUBTRACT R FROM 8
MOV B, A            ;STORE THE RESULT IN REGISTER B
;RESET THE CARRY FLAG
STC
CMC
XLOOP
RAR
JC   COMPA
DCR B
JNC XLOOP
JZ   POW
COMPA
CMC
JMP POW
POW
OUT COLSETA
JMP ADORE
KIND
MOV C, A
MVI A, LASTCOLA
SUB D
ADI 01H
MOV E, A

;FIRST ROTATE THE COLUMN DATA 8 - R TIMES THROUGH CARRY
MVI A, 08H
SUB E
MOV B, A
POP PSW
PUSH E
MOV E, A
STC
CMC
YLOOP
RAR
JC   COMPB
DCR B
JNZ YLOOP
JZ   LOVE
COMPB
CMC
JMP FAITH
LOVE
OUT COLSETA

;NEXT ROTATE THE COLUMN DATA LEFT R TIMES THROUGH CARRY
;CHECKING AND COMPLEMENTING THE CARRY FLAG
MOV A, E
POP E
STC
CMC
ULOOP
RAL
JC   COMPC
DCR E
JNZ ULOOP
JZ   UNITY
COMPC
CMC
JMP IDEALS
UNITY
OUT COLSETB
POP C
POP B
POP H

;READ THE NEXT BYTE OF THE ASCII TEXT BUFFER
ADORE
LDAX D
JMP 'POLE
IFFY
PUSH H

132
LHLDR TOTALS
LDA M
MOV E, A
POP H
;POINT TO THE ADDRESS OF THE FIRST BYTE OF THE DISPLAY RAM
LXI H, DISPRAM
INR B ;INCREMENT ROW COUNTER
;INCREMENT THE ADDRESS OF THE DISPLAY RAM BY THE VALUE OF
;THE ROW COUNTER
PUSH B
INX H
DCR B
JNZ JADE
POP B
;CHECK IF THE VALUE OF THE ROW COUNTER I > or = 8
MOV A, B
CPI 08H
JZ FADE
JM FADE
;SEND ROW DATA TO THE ROW LATCH TO ENERGIZE THE NEXT ROW
MOV A, B
CPI 02H
JNZ TRY3
MVI A, ROW1
OUT ROWLATCH
JMP WHITE
TRY3
CPI 03H
JNZ TRY4
MVI A, ROW2
OUT ROWLATCH
JMP WHITE
TRY4
CPI 04H
JNZ TRY5
MVI A, ROW3
OUT ROWLATCH
JMP WHITE
TRY5
CPI 05H
JNZ TRY6
MVI A, ROW4
OUT ROWLATCH
JMP WHITE
TRY6
CPI 06H
JNZ TRY7
MVI A, ROW5
OUT ROWLATCH
JMP WHITE
TRY7
CPI 07H
JNC FADE
MVI A, ROW6
OUT ROWLATCH
;READ A BYTE OF THE DISPLAY RAM
LDA M ;H-L ALREADY CONTAINS THE ADDRESS OF THE
;NEXT BYTE TO BE READ
JMP PAT
;CONTINUE TO DISPLAY (i.e STATIC DISPLAY) FOR A WHILE BASED
;ON THE VALUE IN THE COUNTER
FADE
POP C
DCR C
JNZ FRAME
RET
; ANIMATE SUBROUTINE
EQU $  
REPEATT
EQU 1024D

; INITIALIZE VARIABLES
STAA
MVI A, 00H
STA COUNTER
; COUNTER KEEPS COUNT OF THE NUMBER OF THE 5 COLUMNS IN EACH
; OF THE 5X7 CHARACTER MATRIX THAT HAVE BEEN SCROLLED OUT
STA XTERNUM
; XTERNUM IS USED TO COUNT THE NUMBER OF CHARACTERS WHOSE
; COLUMN DATA FOR THE ROW HAVE BEEN DISPLAYED
STA MODCTR
; MODCTR IS USED TO NOTE THE NUMBER OF TIMES MODIFICATION HAS
; BEEN DONE ON THE COLUMN DATA OF THE CHARACTER CURRENTLY
; BEING SCROLLED OUT
STA XTEROUT
STA ROWCTR
; ROWCTR IS USED TO KEEP TRACK OF THE NUMBER OF ROWS ALREADY
; ENERGIZED
STA BEGSCROLL
; AFTER A CHARACTER HAS BEEN SCROLLED OUT, BEGSCROLL CONTAINS
; ADDRESS WHERE THE DISPLAY AND SCROLLING WILL BEGIN IN THE
; DISPLAY RAM
STA COLUNCTR
LXI H, FIRSTBYT
INX H
LXI D, FIRSTBYT
XCHG
; READ FIRST BYTE OF ASCII TEXT TEXT BUFFER
LDA M
STA SCROLCTR ; ASCII CODE JUST READ. THIS IS THE CODE
; FOR THE CHARACTER THAT IS BEING SCROLLED OUT
; CHECK IF IT IS THE END OF MESSAGE CHARACTER
CPI 03H
JZ XEND
; SET COLUMN COUNTER TO ZERO
MVI A, 00H
STA COLUNCTR
LXI H, DISPRAM
; WRITE THE FIRST 7 BYTES OF THE DISPLAY RAM INTO MODRANGE
; IN ORDER TO MODIFY THE COLUMN BIT ACCORDINGLY
ESCORT
PUSH D
PUSH H
MVI B, 07H
ZION
LDA M
LXI D, MODRANGE
STAX D
INX H
INX D
DCR B
JNZ ZION
; INCREMENT MODCTR
LXI H, MODCTR
LDA M
ADI 01H
STA MODCTR
CALL MODIFY
; ENERGIZE THE FIRST ROW
MVI A, ROW0
OUT ROWLATCH
POP H
; WRITE COLUMN DATA TO THE FIRST COLUMN LATCH
BIC
LDA M
134
`EQU COLSETA
EQU COLSETB
EQU ZEROS
OUT COLSETA CALL INICOLCTR ;INCREMENT COLUMN COUNTER
LXI D, COLUMCTR
MOV B, M
LXI D, COLINCR
LDAx D
CPI 00H
JZ TAP
MOV D, A
MOV A, B
MAT ADI 01H
DCR D
JNZ MAT
TAP STA COLUMCTR ;DETERMINE THE COLUMN LATCH TO WRITE NEXT BASED ON
;THE VALUE OF THE COLUMN COUNTER
A1 CPI 40H
JNC BEND1
CPI 04H
JNC LATCH1
CPI 09H
JNC LATCH2A
CPI 11H
JNC LATCH3A
CPI 19H
JNC LATCH4A
CPI 21H
JNC LATCH5A
CPI 29H
JNC LATCH6A
CPI 37H
JNC LATCH7A
CPI 39H
JNC LATCH8A
CPI 05H
JNC LATCH12A
CPI 0DH
JNC LATCH23A
CPI 015H
JNC LATCH34A
CPI 1DH
JNC LATCH45A
CPI 25H
JNC LATCH56A
CPI 2DH
JNC LATCH67A
CPI 35H
JNC LATCH78A
LATCH2A CPI 0CH
JP LATCH2B
LATCH3A CPI 14H
JP LATCH3B
LATCH4A CPI 1CH
JP LATCH4B
LATCH5A CPI 24H
JP LATCH5B
LATCH6A CPI 32H
JP LATCH6B
LATCH7A CPI 34H
JP LATCH7B
LATCH8A CPI 3CH`
LATCH12A  JP  LATCH8B
LATCH23A  JP  LATCH12B
LATCH34A  JP  LATCH23B
LATCH45A  JP  LATCH34B
LATCH56A  JP  LATCH45B
LATCH67A  JP  LATCH56B
LATCH078A JP  LATCH67B
          JP  LATCH78B

LATCH1   EQU  $
FASTCOL  EQU  08H
COLSETA  EQU  COLSET1
COLSETB  EQU  ZEROS
          JMP  ANT
LATCH2B  EQU  $
FASTCOL  EQU  10H
COLSETA  EQU  COLSET2
COLSETB  EQU  ZEROS
          JMP  ANT
LATCH3B  EQU  $
FASTCOL  EQU  18H
COLSETA  EQU  COLSET3
COLSETB  EQU  ZEROS
          JMP  ANT
LATCH4B  EQU  $
FASTCOL  EQU  20H
COLSETA  EQU  COLSET4
COLSETB  EQU  ZEROS
          JMP  ANT
LATCH5B  EQU  $
FASTCOL  EQU  28H
COLSETA  EQU  COLSET5
COLSETB  EQU  ZEROS
          JMP  ANT
LATCH6B  EQU  $
FASTCOL  EQU  30H
COLSETA  EQU  COLSET6
COLSETB  EQU  ZEROS
          JMP  ANT
LATCH7B  EQU  $
FASTCOL  EQU  38H
COLSETA  EQU  COLSET7
COLSETB  EQU  ZEROS
          JMP  ANT
LATCH8B  EQU  $
FASTCOL  EQU  40H
COLSETA  EQU  COLSET8
COLSETB  EQU  ZEROS
          JMP  ANT
LATCH12B EQU  $
FASTCOL  EQU  08H
COLSETA  EQU  COLSET1
COLSETB  EQU  COLSET2
          JMP  ANT
LATCH23B EQU  $
FASTCOL  EQU  10H
COLSETA  EQU  COLSET2
COLSETB EQU COLSET3
JMP ANT
LATCH34B EQU $
FASTCOL EQU 18H
COLSETA EQU COLSET3
COLSETB EQU COLSET4
JMP ANT
LATCH45B EQU $
FASTCOL EQU 20H
COLSETA EQU COLSET4
COLSETB EQU COLSET5
JMP ANT
LATCH56B EQU $
FASTCOL EQU 28H
COLSETA EQU COLSET5
COLSETB EQU COLSET6
JMP ANT
LATCH67B EQU $
FASTCOL EQU 30H
COLSETA EQU COLSET6
COLSETB EQU COLSET7
JMP ANT
LATCH78B EQU $
FASTCOL EQU 38H
COLSETA EQU COLSET7
COLSETB EQU COLSET8
JMP ANT
CPI 3DH
JNC MONT
MAY CPI 3FH
JNC JUNE
JMP ANT
MONT CPI 3EH
JP SEPT
JMP MAY
;READ THE NEXT BYTE OF ASCII
POP D
;D CONTAINS ADDRESS OF NEXT BYTE OF ASCII
LDAX D

BOOL
INX D
CPI 49H
JZ LATCH8
CPI 3AH
JZ LATCH8
CPI 2EH
JNZ JULY
LATCH8 EQU $
COLSETA EQU COLSET8
COLSETB EQU ZEROS
JMP ANT
JULY CPI 3FH
JNZ ANT
JC ANT
JUNE MVI C, 00H
MOV A, C
STA COLUMCTR
;INCREMENT CURRENT ADDRESS OF DISPLAY RAM BY 7. AT TH13
;POINT, H-L STILL CONTAINS THE ADDRESS OF THE CURRENT
;BYTE OF DISPLAY RAM
ANT MVI E, 07H
LAPP INX H
DCR E
JNZ LAPP
;READ NEXT BYTE OF DISPLAY RAM

137
LDA  M
PUSH H
PUSH B
PUSH C
PUSH PSW
LXI B, COLSETB
LXI H, COLSETB
LXI D, COLUMCTR
LDA X D
MOV D, A
DAD B
JC SMILE
MOV C, A
MVI A, FASTCOL
SUB D
ADI 01H
MOV E, A
MVI A, 08H
SUB E
MOV B, A
STC
CMC
MALU
RAR
JC NAMA
DCR B
JNZ MALU
JZ LAWN
NAMA
CMC
JMP MSC
LAWN
OUT COLSETA
JMP CHILD
SMILE
MOV C, A
MVI A, FASTCOLA
SUB D
ADI 01H
MOV E, A
MVI A, 08H
SUB E
MOV B, A
POP PSW
PUSH E
MOV E, A
STC
CMC
CANDY
RAR
JC MANA
ADO
DCR B
JNZ CANDY
JZ MIND
MANA
CMC
JMP ADO
MIND
OUT COLSETA
MOV A, E
POP E
STC
CMC
EMOTION
RAL
JC RAY
DEAR
DCR E
JNZ EMOTION
JZ FART
RAY
CMC
JMP DEAR
FART
OUT COLSETB
; READ NEXT BYTE OF ASCII CODE FOR A CHARACTER
XCHG
LDA M
INX H
XCHG
PUSH D
; CHECK WHETHER THE CHARACTER JUST READ IS 'I' or ':' or '.'
CPI 49H
JZ WAP
CPI 3AH
JZ WAP
CPI 2EH
JZ WAP
; INCREMENT COLUMN COUNTER BY 6
LXI D, COLUMCTR
LDAX D
MVI C, 06H
ZONAL
ADI 01H
DCR C
JNZ ZONAL
JMP RAP
; INCREMENT COLUMN COUNTER BY 4
WAP
LXI D, COLUMCTR
LDAX D
MVI C, 04H
CENTRE
ADI 01H
DCR C
JNZ CENTRE
RAP
LXI D, XTERNUM
LDAX D
ADI 01H
STAX D
; CHECK IF XTERNUM IS EQUAL TO THE TOTAL NUMBER OF CHARACTERS
PUSH H
LDAX D
MOV B, A
LXI H, TOTAL#S
LDA M
SUB B
JNZ A1
; INCREMENT ROW COUNTER
LXI H, ROWCTR
LDA M
ADI 01H
STA ROWCTR
CPI 06H
JM WRAP
LDA M
CPI 01H
JZ LINE1
CPI 02H
JZ LINE2
CPI 03H
JZ LINE3
CPI 04H
JZ LINE4
CPI 05H
JZ LINE5
JMP WRAP
LINE1
MVI A, ROW1
OUT RowlATCH
MVI A, ROW2
OUT ROWLATCH
JMP AXID

MVI A, ROW3
OUT ROWLATCH
JMP AXID

MVI A, ROW4
OUT ROWLATCH
JMP AXID

MVI A, ROW5
OUT ROWLATCH
JMP AXID

AXID
LXI H, DISPRAM
LXI D, ROWCTR
LDAX D

;GET FIRST BYTE OF THE DISPLAY RAM FOR THE NEW ROW
MOV C, A
REDUCER
INX H
DCR C
JNZ REDUCER
JMP BIC

;INCREMENT COUNTER
WRAP
LXI D, CTR
LDAX D
ADI 01H
STAX D
MOV E, A

;CHECK WHETHER THE SCROLLXTER IS 'I' or ':' or '.'
LXI D, SCROLLXTER
LDAX D
CPI 49H
JZ LIFE3
CPI 3AH
JZ LIFE3
CPI 2EH
JNZ LIFE5
CPI 49H

LIFE3
MOV A, E
CPI 03H
JNC BERT

;CHECK IF VALUE OF CTR = 5
MOV A, E
CPI 05H
JC COLSCROL ;GO AND BEGIN SCROLLING OUT A CHARACTER

;INCREMENT THE TOTAL NUMBER OF CHARACTERS THAT HAVE ALREADY BEEN SCROLLED OUT
LXI D, XTEROUT
LDAX D
ADI 01H
STAX D
MOV B, A

;CHECK IF THE NUMBER OF CHARACTERS THAT HAVE ALREADY BEEN SCROLLED OUT IS EQUAL TO THE TOTAL NUMBER OF CHARACTERS
LXI H, TOTAL#S
LDA M
CMP B
JZ AGAIN

;STATSCOL = CURRENT-ADDRESS(BYTE) - OF-DISPRAM + 7 * XTEROUT
;(REG B) <-- XTEROUT
MOV A, 00H

MULTIPLI
ADI 07H
DCR B
JNZ MULTIPLI
MOV C, A
LXI H, DISPRAM
INX H
DCR C
JNZ BOXER
SHLD STATSCOL
;READ THE ASCII CODE FOR THE NEXT CHARACTER TO BE SCROLLED
;OUT i.e THE CHARACTER WHICH ASCII CODE IN THE ASCII BUFFER
;HAS THE ADDRESS: FIRSTBYTE + XTEROUT
LXI D, FIRSTBYTE
LXI H, XTEROUT
MOV B, M
INX D
DCR B
JNZ MICAH
;READ THE NEXT CHARACTER TO BE SCROLLED OUT
LDA X D
STA SCROLLXTER
CPI 03H
;CHECK IF IT IS THE END OF MESSAGE (ETX)
JZ AGAIN
MVI A, 00H
STA COLUMNTR
LHLD STATSCOL
JMP ESCORT
AGAIN
MVI A, REPEATG
DCR A
JNZ STAAT
RET
;COPYME SUBROUTINE
EQU $  
MVI C, EXTERNUM  
LHLD FIRSTBYTE  
XCHG  
LHLD BMEX1  
LOOP  
LDA M  
STAX  
INX D  
INX H  
DCR C  
JNZ LOOP  
XCHG  
SHLD LASTBYTE  
RET

;MODIFY SUBROUTINE
EQU $  
PUSH B  
PUSH D  
PUSH PSW  
;READ VALUE OF MODCTR  
LXI H, MODCTR  
LDA M  
CPI 01H  
JZ BIT1  
LDA M  
CPI 02H  
JZ BIT2  
LDA M  
CPI 03H  
JZ BIT3  
LDA M  
CPI 04H  
JZ BIT4  
LDA M  
CPI 05H  
JZ BIT5  
JMP ENEDIT  
BIT1  
MVI C, 07H  
LXI H, MODRANGE  
NOSE  
LDA M  
STC  
CMC  
RAL  
DCR C  
JZ SOUT  
INX H  
JMP NOSE  
BIT2  
MVI C, 02H  
WEAR  
MVI C, 07H  
LXI H, MODRANGE  
WARE  
LDA M  
STC  
CMC  
RAL  
DCR C  
JZ PEACE  
INX H  
JMP WARE  
PEACE  
DCR B  
JNZ WEAR  
JMP SOUT
BIT3
MVI C, 03H
MVI C, 07H
LXI H, MODRANGE
PINE
LDA M
STC
CMC
RAL
DCR C
JZ BIRO
INX H
JMP PINE
BIRO
DCR B
JNZ WEAR
JMP SOUT
BIT4
MVI C, 04H
PUNK
MVI C, 07H
LXI H, MODRANGE
AMIT
LDA M
STC
CMC
RAL
DCR C
JZ PITY
INX H
JMP AMIT
PITY
DCR B
JNZ PUNK
JMP SOUT
BIT5
MVI C, 05H
WOLF
MVI C, 07H
LXI H, MODRANGE
WONDER
LDA M
STC
CMC
RAL
DCR C
JZ EXITS
INX H
JMP WONDER
EXITS
DCR B
JNZ WOLF
SOUT
POP PSW
POP D
POP B
RET

;INICOLCTR SUBROUTINE
INICOLCTR EQU $00
PUSH PSW
PUSH H
PUSH B
PUSH D
;READ MODCTR
LXI H, MODCTR
LDA M ;(Acc) <--MODCTR
LXI H, SCROLLXTER
MOV B, M
CPI 01H ;CHECK IF MODCTR = 1
JZ BIT1
CPI 02H
JZ BIT2
CPI 03H
JZ BIT3

143
BIT1
MOV A, B
CPI 49H ; CHECK IF IT IS THE 'I' CHARACTER
JZ TAB1
CPI 3AH ; CHECK IF IT IS THE ':' CHARACTER
JZ TAB1
CPI 2EH ; CHECK IF IT IS THE '.' CHARACTER
JZ TAB1
; SET COLCTR TO 5 (INSTEAD OF 6) BECAUSE ONE COLUMN IS ALREADY SCROLLED OUT
MVI A, 05H
STA COLINCR
JMP STOPPING
; SET COLINCR TO (INSTEAD OF 4)
TAB1
MVI A, 03H
STA COLINCR
JMP STOPPING

BIT2
MOV A, B
CPI 49H
JZ TAB2
CPI 3AH
JZ TAB2
CPI 2EH
JZ TAB2
; SET COLCTR TO 4 (INSTEAD OF 6) BECAUSE 2 COLUMNS ARE ALREADY SCROLLED OUT
MVI A, 04H
STA COLINCR
JMP STOPPING

TAB2
MVI A, 02H
STA COLINCR
JMP STOPPING

BIT3
MOV A, B
CPI 49H
JZ TAB3
CPI 3AH
JZ TAB3
CPI 2EH
JZ TAB3
; SET COLCTR TO 3 (INSTEAD OF 6) BECAUSE 3 COLUMNS ARE ALREADY SCROLLED OUT
MVI A, 03H
STA COLINCR
JMP STOPPING

TAB3
MVI A, 01H
STA COLINCR
JMP STOPPING

BIT4
MOV A, B
CPI 49H ; CHECK IF IT IS THE 'I' CHARACTER
JZ TAB4
CPI 3AH ; CHECK IF IT IS THE ':' CHARACTER
JZ TAB4
CPI 2EH ; CHECK IF IT IS THE '.' CHARACTER
JZ TAB4
; SET COLCTR TO 2 (INSTEAD OF 6) BECAUSE 4 COLUMNS ARE ALREADY SCROLLED OUT
MVI A, 02H
STA COLINCR
JMP STOPPING

144
TAB4
MVI A, 00H
STA COLINCR
JMP STOPPING

BIT5
MOV A, B
CPI 49H ;CHECK IF IT IS THE 'I' CHARACTER
JZ TAB5
CPI 3AH ;CHECK IF IT IS THE ':' CHARACTER
JZ TAB5
CPI 2EH ;CHECK IF IT IS THE '.' CHARACTER
JZ TAB5
;SET CYLCTR TO 1 (INSTEAD OF 6) BECAUSE 5 COLUMNS ARE ALREADY SCROLLED OUT
MVI A, 01H
STA COLINCR
JMP STOPPING

TAB5
MVI A, 00H
STA COLINCR
JMP STOPPING

STOPPING
POP D
POP B
POP H
POP PSW
RET

;PCPROC SUBROUTINE
EQU $
ORG 003CH
;SAVE CPU ENVIRONMENT
PUSH H
PUSH PSW
;DISABLE INTERRUPTS
MVI A, 0FH
SIM
DI
CALL WIPE
LHLD PCASCII
;READ DATA FROM THE 8251A DATA PORT
IN A0H
CPI 03H
JZ ANOGU
;STORE CHARACTER IN ASCII TEXT BUFFER
IN A0H
STA M
INX H
SHLD PCASCII
JMP DIOKWU
ANOGU
LXI H, 0017H
PCHL
;ENABLE INTERRUPTS
DIOKWU
MVI A, 08H
SIM
EI
POP PSW
POP H
RET

;KEYP SUBROUTINE
EQU $
ORG 0034H
;SAVE CPU ENVIRONMENT
PUSH PSW
;ENABLE INTERRUPTS
MVI A, 0FH
SIM
DI
;PROCESS KEYPAD INPUT
CALL KEYPROC
;ENABLE INTERRUPTS
LXI H, 0017H
PCHL
MVI A, 08H
SIM
EI
;RESTORE CPU ENVIRONMENT
POP PSW
RET

;USARTA SUBROUTINE
EQU $
ORG 002CH
;DISABLE INTERRUPTS
PUSH PSW
MVI A, 0FH
SIM
DI
;ENABLE THE 8251A
MVI A, 00H
OUT A0H  ;THIS, APART FROM ENABLING THE 8251A,
;ALSO Initializes ITS DATA PORT TO ZERO
;ENABLE INTERRUPTS
MVI A, 08H
SIM
EI
POP PSW
RET

;WIPE SUBROUTINE
EQU $
PUSH PSW
LHL D DISPRAM
MIX
MVI A, 00H
STA M
INX H
XCHG
LHL D LASTBYTE
DAD D
JC BULLY
XCHG
JMP MIX
BULLY
XCHG
MVI A, 00H
STA M
RET

;KEYPROC SUBROUTINE
EQU $
KEYADDR EQU AFH  ;KEYPAD BUFFER ADDRESS
PUSH H
PUSH PSW
PUSH B
PUSH C
; SERVICE THE KEYPAD
CALL KEYTEST
CPI 'YES'
JNZ FINISHA
CALL WIPE
MVI C, 00H ; REGISTER IS A COUNTER USED TO COUNT THE
; NUMBER OF CHARACTERS WRITTEN i.e ACCEPTED FROM THE KEYPAD
BEGIN
MVI E, 00H
XTERIN
IN KEYADDR ; READ KEYPAD BUFFER
INR E
MOV B, A
MOV A, E
SUI 01H
JNZ SKLEY
FIRSTKEY
MVI D, 04H
MOV A, B
ROTATE
RLC
DCR D
JNZ ROTATE
MOV B, A ; STORE THE FIRST PART OF THE ASCII HEX
; CODE FOR THE CHARACTER KEYED IN IN REGISTER B
JMP XTERIN
SKLEY
MOV D, A ; MOVE THE SECOND PART INTO REGISTER D
MOV A, B ; MOVE BACK THE FIRST PART INTO THE ACC
ORA D ; MERGE THE TWO PARTS TO GIVE THE ASCII
; HEX CODE FOR THE CHARACTER KEYED IN
CPI 03H ; CHECK IF IT IS THE END OF TEXT CHARACTER
JZ FINISHA
STA M ; WRITE THE ASCII HEX CODE FOR THE CHARACT
; INTO THE ASCII TEXT BUFFER
INX H ; POINT TO NEXT BYTE OF ASCII TEXT BUFFER
INR C ; INCREMENT COUNTER
JMP BEGIN
FINISHA
SHLD LASTBYTE
MOV A, C
STA NUMB ; WRITE THE COUNTER VALUE INTO LOCATION NU
POP C
POP B
POP PSW
POP H
RET

; KEYTEST SUBROUTINE
KEYTEST EQU $ 
MESAG1 EQU 'YES'
MESAG2 EQU 'NO'
CHECKKEY
MVI E, 00H
REED
IN AFH
MOV D, A
INR E
MOV A, E
CPI 01H
JZ BKEY
AKY
MOV A, D
MOV C, 04H
NOTE
RLC
DCR C
JNZ RATE
MOV B, A
JMP REED
BKEY
MOV C, A
MOV A, B
ORA C
CPI 02H
JNZ CHECK
MVI A, MESAG1
JMP ENDI
CHECK
MVI A, MESAG2
END1
RET
APPENDIX 6: POWER SUPPLY
<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>DESCRIPTION</th>
<th>PART NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED</td>
<td>LIGHT EMITTING DIODE</td>
<td></td>
</tr>
<tr>
<td>TR1</td>
<td>BC548, NPN SILICON TRANSISTOR</td>
<td>7</td>
</tr>
<tr>
<td>TR8</td>
<td>TIP127, PNP POWER DARLINGTON TRANSISTOR</td>
<td>7</td>
</tr>
<tr>
<td>IC33</td>
<td>74LS377, OCTAL FLIP FLOP</td>
<td>1</td>
</tr>
<tr>
<td>IC34 a, b, c</td>
<td>74HCT245, OCTAL BUS TRANCEIVER</td>
<td>2</td>
</tr>
<tr>
<td>IC1 TO IC16</td>
<td>74HCT574, OCTAL D-TYPE FLIP FLOP</td>
<td>16</td>
</tr>
<tr>
<td>IC17 TO IC32</td>
<td>ULN2803A, OCTAL DARLINGTON DRIVER</td>
<td>16</td>
</tr>
<tr>
<td>R1 TO R128</td>
<td>1 WATT RESISTORS</td>
<td>128</td>
</tr>
<tr>
<td>IC36 a, b, c</td>
<td>74LS138, 3-TO-8 DECODER</td>
<td>2</td>
</tr>
<tr>
<td>IC34 a, b</td>
<td>8286, OCTAL BUS TRANSCEIVER</td>
<td>2</td>
</tr>
<tr>
<td>IC43</td>
<td>2-INPUT AND GATE</td>
<td>1</td>
</tr>
<tr>
<td>IC46 a, b, c</td>
<td>74LS30, 8-INPUT AND GATE</td>
<td>3</td>
</tr>
<tr>
<td>IC38 a, b</td>
<td>2-INPUT OR GATE</td>
<td>2</td>
</tr>
<tr>
<td>IC53</td>
<td>2-INPUT NAND GATE</td>
<td>2</td>
</tr>
<tr>
<td>IC41</td>
<td>74LS06, INVERTER</td>
<td>15</td>
</tr>
<tr>
<td>IC42</td>
<td>MM74C922N, 16-KEY KEYPAD ENCODER</td>
<td>1</td>
</tr>
<tr>
<td>IC39</td>
<td>8212, 8-BIT INPUT/OUTPUT PORT</td>
<td>1</td>
</tr>
<tr>
<td>IC40</td>
<td>8282, 8-BIT OCTAL LATCH</td>
<td>1</td>
</tr>
<tr>
<td>IC35</td>
<td>8205, 3-TO-8 BINARY DECODER</td>
<td>1</td>
</tr>
<tr>
<td>IC37</td>
<td>74LS139, 2-TO-4 DECODER</td>
<td>2</td>
</tr>
<tr>
<td>IC48</td>
<td>8253, PROGRAMMABLE INTERVAL TIMER</td>
<td>1</td>
</tr>
<tr>
<td>IC49</td>
<td>8251A, PROGRAMMABLE COMMUNICATIONS INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>IC51</td>
<td>MC1488, LINE DRIVER</td>
<td>1</td>
</tr>
<tr>
<td>IC50</td>
<td>MC1489, LINE RECEIVER</td>
<td>1</td>
</tr>
<tr>
<td>IC55</td>
<td>2141, 4K x 1 RAM</td>
<td>32</td>
</tr>
<tr>
<td>IC54</td>
<td>2764, 8K x 8 EPROM</td>
<td>2</td>
</tr>
<tr>
<td>IC45</td>
<td>8085A, MICROPROCESSOR</td>
<td>1</td>
</tr>
<tr>
<td>IC60</td>
<td>L78505CV, VOLTAGE REGULATOR</td>
<td>1</td>
</tr>
<tr>
<td>C1</td>
<td>100n CAPACITOR</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>100µ CAPACITOR</td>
<td>1</td>
</tr>
<tr>
<td>C3</td>
<td>220µ CAPACITOR</td>
<td>1</td>
</tr>
</tbody>
</table>