FLIGHT TRANSPORTATION LABORATORY
REPORT R84-2

A MICROPROCESSOR DRIVEN LIQUID CRYSTAL GRAPHICS DISPLAY FOR AIRCRAFT USE

Lee Howard Marzke

June 1984

FTL REPORT R84-2

A MICROPROCESSOR DRIVEN LIQUID CRYSTAL GRAPHICS DISPLAY FOR AIRCRAFT USE

Lee Howard Marzke

A complete graphics system for use in modular avionics is built around a liquid crystal flat panel display. Screen refresh is handled by display controller that provides a bit mapped representation of the display in RAM. A 8085 based single board computer is programmed to allow user defined graphics symbols to be moved about easily, and a character generator is included to facilitate the display of ASCII strings. The computer uses a RS-232 interface to receive commands. A demonstration program is also included which demonstrates a simple instrument landing system (ILS) type display without the need for an external device.

ACKNOWLEDGEMENTS

I would like to thank the following people for their special contributions:

- Professor Antonio Elias for being my thesis advisor and for suggesting the general purpose display idea.

- Carter Pfaelzer for day to day help when I was defining my thesis, and sugestions when ever I had problems.

- SIRRUS FLOW TECHNOLOGY INC., especially Carl Johnson for the use of his OSBORN computer to develop the software.

- Spencer Webb for his invaluble encouragement, and for use of his facilities at TERADYNE INC. to draw schematics and program EPROMS's.

- Scott Hathcock, my roomate, for putting up with the mess at our apartment during construction, and for constant prodding to help keep me on schedule.

- Fred Schloetzer for encouraging me to come to MIT in the first place.

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

TABLES

1.0 INTRODUCTION

This thesis consisted of building a high resolution flat panel display for aircraft use. Such displays can be used to emulate just about any of the current mechanical flight instruments in the cockpit. Other applications would include collision avoidance displays, electronic checklists, or compleatly new forms of flight instruments.

Many cathode ray tube (CRT) based displays are currently in use in aircraft such as the Boeing 767. CRT displays are large, bulky, and consume large amounts of power; they are, however, still the best for high resolution applications.

Of the flat panel technologies such as electrolumenescent, plasma, and liquid crystal (LCD), the LCD types have by far the lowest power consumption, and have the best contrast in high ambient light environments. The disadvantages of the LCD display are a limited viewing angle range, and moderate resolution.

The display chosen consists of a dot matrix LCD unit of 128 by 240 pixels, with onboard drive electronics. As the technology improves, higher resolution and lower cost units will make LCD the best choice for the future.

As currently configured the display system is driven from a RS-232 communication port. Fast graphic display response is possable over such a slow link because all symbols to be used are stored in the graphic display unit's (GDU) internal RAM. Real time commands thus need only specify new coordinates for a symbol or new text. Advantages of such displays in small airplanes would include the ability to multiplex different instruments on

the same screen, saving weight and panel space.

The entire project consisted of the following: Choosing a suitable display based on the above considerations, design and construction of a display controller, and finally assembly language programming to store and move text strings and symbols.

I became interested in this project because of my love for flying as a hobby, and my desire to do a project containing hardware design, real time software development, and prototype construction.

GRAPHICS DISPLAY UNIT

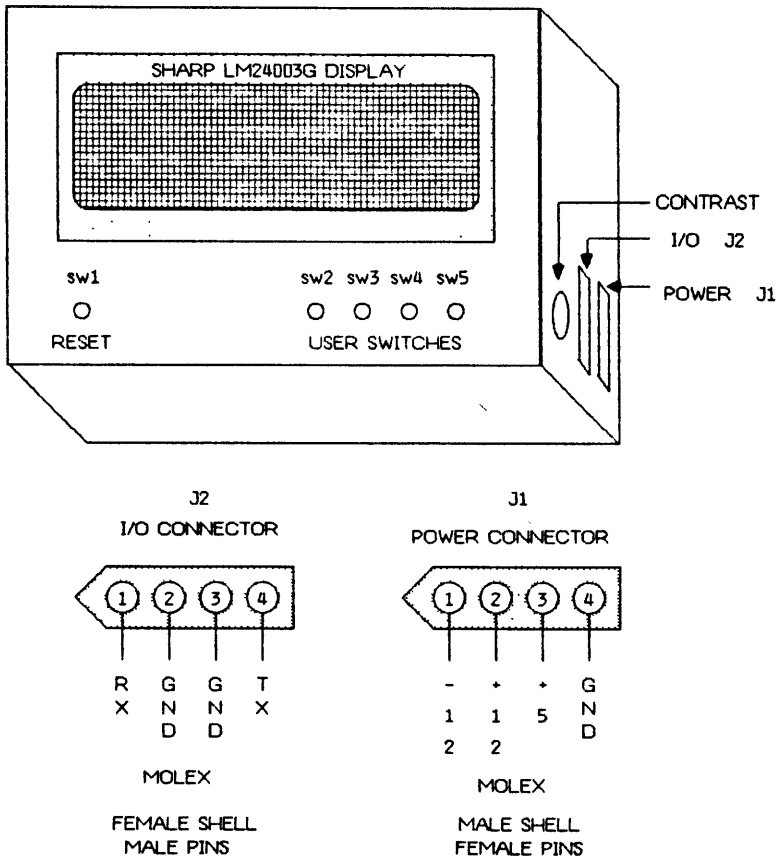


FIG. 1 GRAPHICS DISPLYAY UNIT

2. FUNCTIONAL DESCRIPTION

2.1 SETUP

The graphics display unit (GDU) shown in Fig. 2 is a complete system allowing the user to define graphic symbols of variable size and shape and move them around on the screen. Symbols may overlap one another and can be erased easily. Individual points may also be accessed, and text is easily displayed because of the internal character generator. The software recognizes strings of ASCII characters as commands, thus permitting debugging of command strings with a terminal. A demonstration program is also included on EPROM to demonstrate the functioning of the unit without the need for a terminal.

The GDU requires the following: (see Fig. 2)

POWER	-	5V at 1.5 amp
(J1)	-	+12V at 0.1 amp
	-	-12V at 0.1 amp
COMMUNICATION		RS-232 Port
(J2)		

AIRCRAFT LCD GRAPHICS DISPLAY
BLOCK DIAGRAM

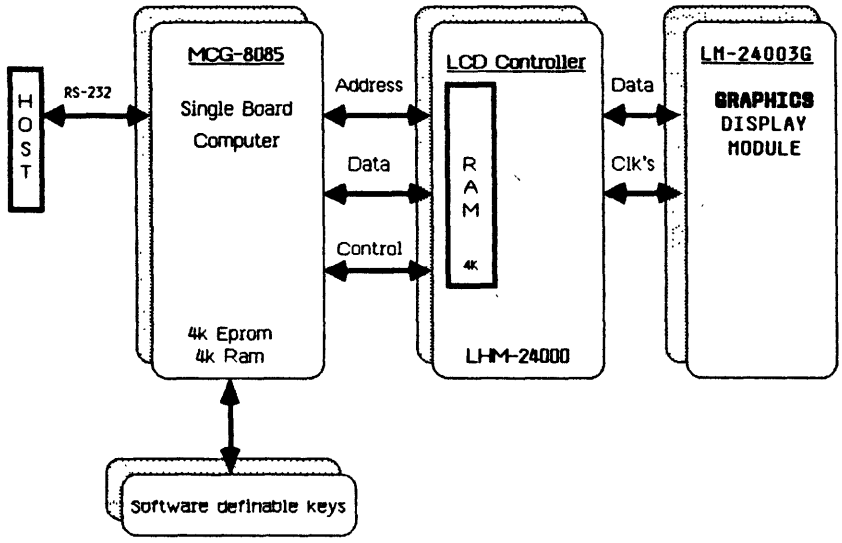


Fig. 2 Graphics display system.

2.2 COMMAND FORMAT

When power is applied and the reset button is pressed a sign-on menu should appear on the LCD display. The setting of the contrast control is critical and should be set fully counter clockwise if the display appears blank or a symbol is missing. The following discussion will assume a terminal is connected to the RS-232 port for demonstration of the individual commands.

The switch under the "DISPLAY" prompt (sw3) is depressed to use the RS-232 port for input. A space character must first be sent to allow computation of the BAUD rate, the GDU will respond with a "#" prompt.

Two types of objects may be defined and stored in the GDU, "strings" of ASCII characters, and user defined graphic "symbols" of selectable height and width. Each string or symbol is associated with a two digit hex number, hereafter shown as ##. This number may range from 00_H to $3F_H$, the upper two bits being used internally to distinguish between strings and symbols. Once a string or symbol is defined it may be moved on the screen by an instruction to move string ## or symbol ## to the specified coordinates.

Other commands allow individual points on the screen to be turned on one at a time, the screen to be cleared, and functions to be called that define or move many symbols.

The following symbology is used below:

- Two digit hex number that identifies a particular string or symbol. (00_H - $3F_H$)

XX - Two digit hex number. (00_H - EF_H)

YY - Two digit hex number (00_H - $7F_H$)

<CR> - Carriage Return

ASCII STRINGS

A string is first defined by typing 'A##/STRING<CR>' 'STRING' may contain ASCII characters from 20_H to 60_H (numbers, uppercase letters, and symbols). To display the string, type 'A##, XX, YY<CR>', where XX and YY define the x and y coordinate of the left edge of the string. Thereafter, if the same string is referenced with new coordinates, the string will be erased and redisplayed at the new location. The 'STRING' may be changed by reentering 'A##/NEWSTRING<CR>', however 'NEWSTRING' must contain the exact number of characters as the original string by appending spaces where necessary. The first definition of a string, therfore, must be long enough to reserve space for later changes.

EXAMPLE:	A02/BEARING<CR>	--	Defines string 02 to be "BEARING"
	A02, 10, 20 <CR>	--	Puts string 02 on screen at location 10, 20
	A02/COURSE[]<CR>	--	Redefines string 02 to be "COURSE". Note added space "[] " to make string 7 characters long.
	A02, 20, 20<CR>	--	Moves string 02 to location 20, 20

The spacing of characters may be changed from the

normal two pixels between characters (wide) to one pixel spacing (narrow) by the 'N' and 'W' commands respectively.

GRAPHIC SYMBOLS

A graphic symbol is defined by entering 'D##,HHWWBBBBB...<CR>' The height, HH (in pixels) may be any hex number from 1_H to 127_H. The width, WW is measured in words (multiples of 8 pixels wide) and may range from 1 to 30_H. 'BBBB...' represents groups of 8 bits (2 hex dig.) defining the graphic symbol sequentially (as if reading a book) 8 pixels at a time.

EXAMPLE: D02,0501,010204F8FF<CR>
Defines the graphic symbol shown below having a height of 5, and a width of 1 (8 pixels).

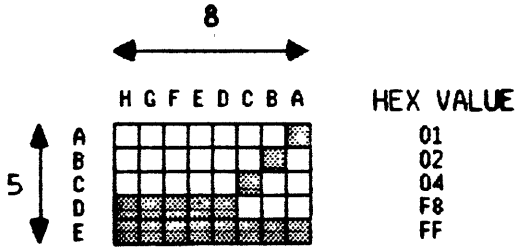


Fig. 3 Symbol #02

Once the symbol is defined it may be written to the screen by entering 'P##,XX,YY<CR>', where the upper left corner of the symbol is placed at the indicated

coordinates. Another 'P' command will place another copy of the symbol on the screen at a different location. To move a symbol use 'M##,XX,YY<CR>', which first erases the last location of the symbol, then places a copy at the new coordinates.

To have a transparent symbol in which underlying symbols can still be seen, place the symbol on the screen with the 'X' command. This command exclusive or's the symbol with the current contents of screen memory. To erase the symbol, execute this command a second time - this removes the symbol leaving the origional background intact.

POINTS

To turn on a specific point the 'L' command is used. The format is 'LXXYY,XXYY,XXYY<CR>, where XXYY are the x and y coordinates of the points. For example to turn on the points (10,10) and (10,11) the command would be 'L1010,1011<CR>'. Any number of points may be entered this way. Because these points are not stored they can not be moved or erased without clearing the screen or displaying a symbol on top of them.

MISCELLANEOUS COMMANDS

The screen may be cleared with the Clear (C) command. The Init (I) command clears the screen and erases all stored symbols and strings.

Incoming characters are normally echoed by the command recognizer for use with a terminal. Echo may be

turned off with the Kill echo (K) command and back on with the Echo on (E) command.

FUNCTIONS

The Function (F) command starts execution of a user defined program that may define symbols and strings, or move existing ones.

The hexidecimal byte following the F command is placed in the accumulator and a call is made to address 3050_H.

TABLE 1

COMMAND SUMMARY

A##, XX, YY<CR>	Move ASCII string ## to location XX, YY
A##/MESSAGE<CR>	Change ASCII string ## to MESSAGE
C<CR>	Clear Screen
D##, HHWW, BBBB. . . <CR>	Define symbol ## of height HH and width WW according to the bits that follow.
E<CR>	Echo on
F##<CR>	Calls user defined function ##.
I<CR>	Init (Clear all strings, symbols and screen).
K<CR>	Kill echo of command characters to terminal.
LXXYY, XXYY, XXYY. . <CR>	List of points to turn on.
M##, XX, YY<CR>	Move graphic symbol ## from old location to XX, YY
N<CR>	Narrow character spacing.
P##, XX, YY<CR>	Put graphic symbol ## at XX, YY
W<CR>	Wide character spacing.
X##, XX, YY<CR>	XOR graphic symbol ## with contents of screen at XX, YY
X##/<CR>	Erase graphic symbol ## from it's most recently used coordinate.
Z<CR>	Jump to monitor.

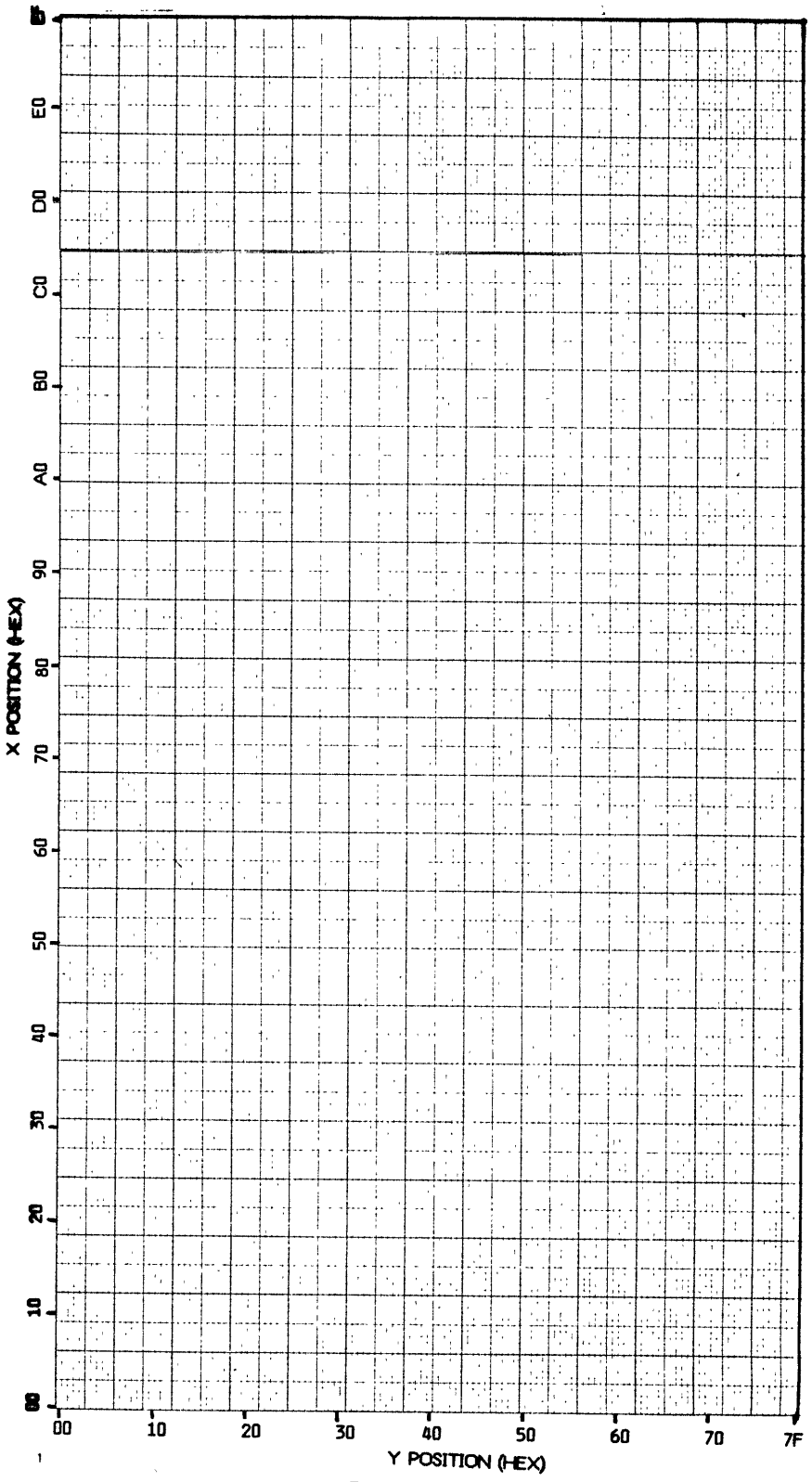


Fig. 4 SCREEN COORDINATES

2.3 DEMONSTRATION ILS DISPLAY

To enter the demonstration program the button under the 'DEMO' prompt (sw4) is depressed from the menu screen. A simple Instrument Landing System (ILS) type display appears on the left side of the display, with prompts over each of the 'soft' buttons. The pointers of the ILS instrument move when the appropriate buttons pushed, both moving at the same time if desired.

If the left"‹" and right "› buttons are depressed simultaneousely, the two pointers will move in repetitive pattern. To exit this mode hold down one of the four soft buttons until the pattern stops. Pressing all four buttons will cause a jump to the command mode without clearing any symbols, so that experimentation with the instrument is possible.

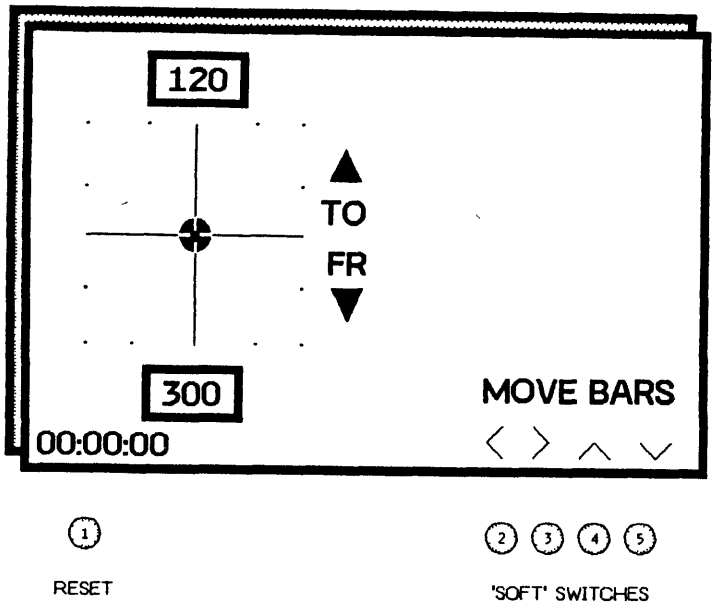


Fig. 5. ILS Demonstration Screen

TABLE 2

LIST OF ASSIGNMENTS IN ILS DEMONSTRATION PROGRAM

SYMBOLS:	COORDINATES:
01 - Horizontal bar	(10, 37)
02 - Vertical bar	(37, 10)
03 - Hollow boxes	(2B, 00), (2B, E0)
04 - Up triangle, "TO"	(61, 26)
05 - Center mark	(33, 34)
06 - Down triangle, "FR"	(61, 39)
07 - up arrow	(C7, 7A)
08 - down arrow	(E8, 7A)
09 - right arrow	(A6, 78)
0A - Left arrow	(85, 78)
STRINGS:	
01 - Time "00: 00: 00"	(00, 79)
02 - "120"	(2E, 03)
03 - "300"	(2E, 67)
04 - "MOVE BARS"	(A6, 6E)

3. TECHNICAL DESCRIPTION

3.1 LIQUID CRYSTAL CELL

The liquid crystal display used in this project depends upon the twisted nematic effect. The individual molecules of the liquid crystal are asymmetric and tend to align themselves with one another. This alignment may extend over distances up to 1 mm. The LCD cell contains a thin film of a liquid crystal with alignment layers forcing the liquid crystal to twist 90 degrees from face to face. Because of the large refractive index anisotropy in the liquid crystal, the plane of polarized light is guided through the cell twisting with the crystal.

The cell is sandwiched between two crossed polarizers. In its 'off' state the incoming polarized light is rotated, allowing the light to pass through the second polarizer.

The cell is turned 'on' by applying an AC signal across the LCD cell, forcing all the liquid crystal molecules to align with the electric field. Now the incoming light is not rotated and is blocked from passing through the cell, so the pixel element appears dark.

Insulating layers are added on top of each electrode to insure that no DC current flows through the cell, since the resulting electrolisis of the liquid crystal would ruin it.

LIQUID CRYSTAL CELL

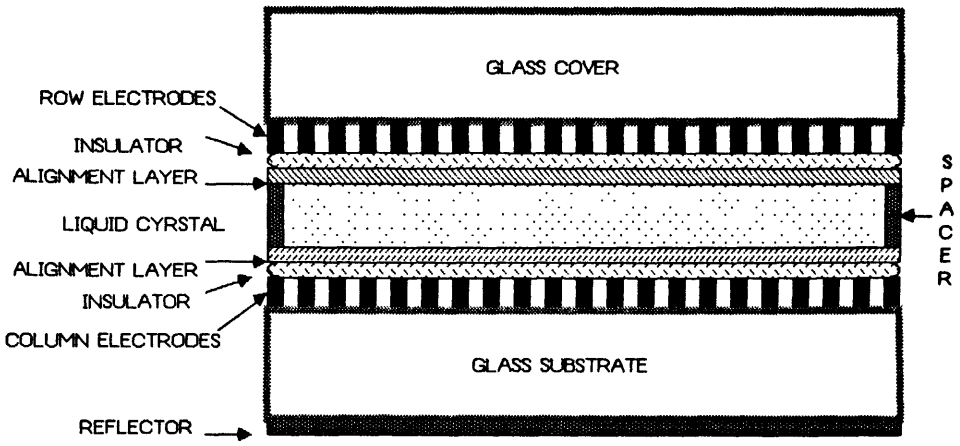


FIG. 6 LCD - CROSS SECTION

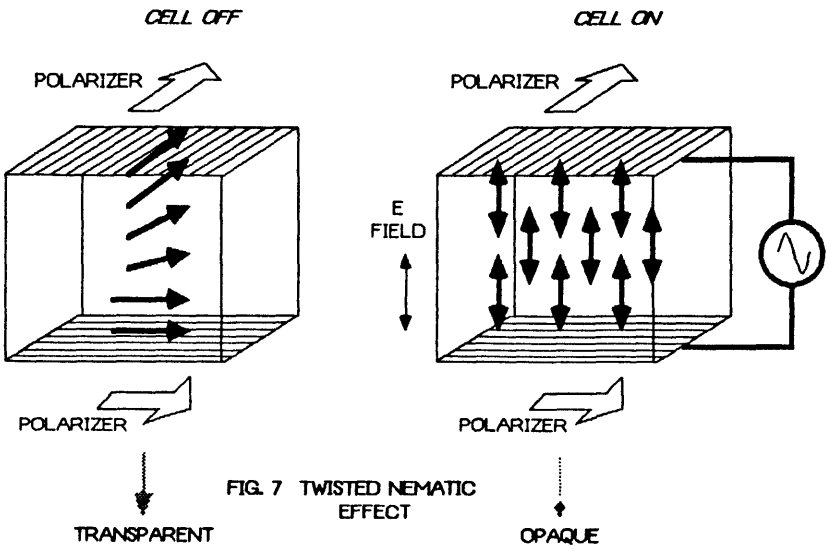


FIG. 7 TWISTED NEMATIC EFFECT

3.2 CIRCUIT DESCRIPTION

The LCD screen is bit mapped into display RAM which is shared with the 8085 single board computer (SBC). The display controller hardware constantly refreshes the LCD screen with the data in display RAM.

The SHARP display requires data for the top half and bottom half of the screen concurrently as well as various clock signals. (see Fig. 8) The data is sent in serial form at 600 Kbits/sec. over two lines, with another four lines required for clock signals. The data for the two sections of the display are stored in seperate RAM's. The addresses for display RAM are shown in Fig. X. The hardware thus provides the following:

- constant refresh of the screen
- display RAM shared with the 8085
- all the clocks necessary to run the display

TIMING GENERATOR:

All timing and clocks are derived from the 3MHz 8085 processor clock input 3MH. (see Fig. 9) This signal is divided by five by the decade counter U4 to generate the 600 KHz CP2 clock, which is the serial data rate used by the controller. CP2 clocks the shift registers in the display RAM section, is divided by eight by counter U5 to provide the address counter clock ADCLK, and is used to derive the other clocks for the display. Gate U13a resets U5 when 8RES goes low at the end of every page.

Counters U6 and U7 divide CP2 by 240 to provide a clock CP1 that is high for the last pixel of every line. U6 and U7 are wired as a divide-by-30 counter that is enabled once

every eight cycles of CP2 by ADCLK.

A divide-by-64 counter is implemented by U14 and U15 to provide a signal LLINE that is high for the entire last line of the display. J-K flip-flop U10a sets at the falling edge of CP1 if LLINE is high, then resets at the next CP1, generating a signal FLINE that is high for the entire first line of the display. This pulse is delayed approximately 5us by R1, C1 and U16 to provide the page clock signal S to the display. FLINE toggles U9a to provide M, which causes the LCD drive polarity to alternate on succesive pages.

ADDRESS COUNTER

An 11 bit address counter is implemented by U11 and U12. ADCLK increments the counters sequentially through all of the valid addresses in display RAM. The 8 bits of each word are shifted out serially to the display between succesive counts. The counters are reset to zero at the beginning of each page by ADRES.

DISPLAY RAM

The 240 by 128 pixel display is mapped into 4K words of RAM consisting of U21 and U22. The upper half of the display resides in all but the last 128_H locations of U21, and the lower half similiarly in U22. This gap of 128 locations between the upper and lower storage locations of the screen must be handled by software.

The display RAM is shared between the display controller and the computer. The computer addresses the RAM as a 4K x 8 block of memory, however the controller addresses it as a 2K x 16 block because it must send data simultanously to both halves of the display. The address

lines of display RAM are switched between the controller and computer by the MUX's U18, U19, and U20. The data paths from the two RAM's are connected to seperate shift registers U25 and U26, but are also wire or'ed together through data buffers U23, U24, U27 and U28 to the computer data bus.

The shift registers are loaded with the a word from memory immediatly after each count of the display address counter by a low level on SRLOAD.

CONTROL LOGIC

This section (Fig. 10) controls the sharing of display RAM between the processor and display controller.

The amount of time for a memory access by the computer is so small (< 1us) that the display controller has almost unrestricted use of display RAM. Thus, the MUX's are normally selected to the counter, the data buffers are off (high impedance), and the shift registers are sending data to the display (Fig. 11).

When the computer accesses display RAM several things must occur. When chip U1 detects an address in the range of 4000_H to $4FFF_H$ (display RAM) it deselects the correct RAM preventing access by the display controller. The MUX's select the the computer address bus for input. Finally the data buffers are enabled in the correct direction by U2, U3, and U8 depending on the status of the read and write lines (Fig. 12)

SHARP LM-24003G GRAPHICS DISPLAY

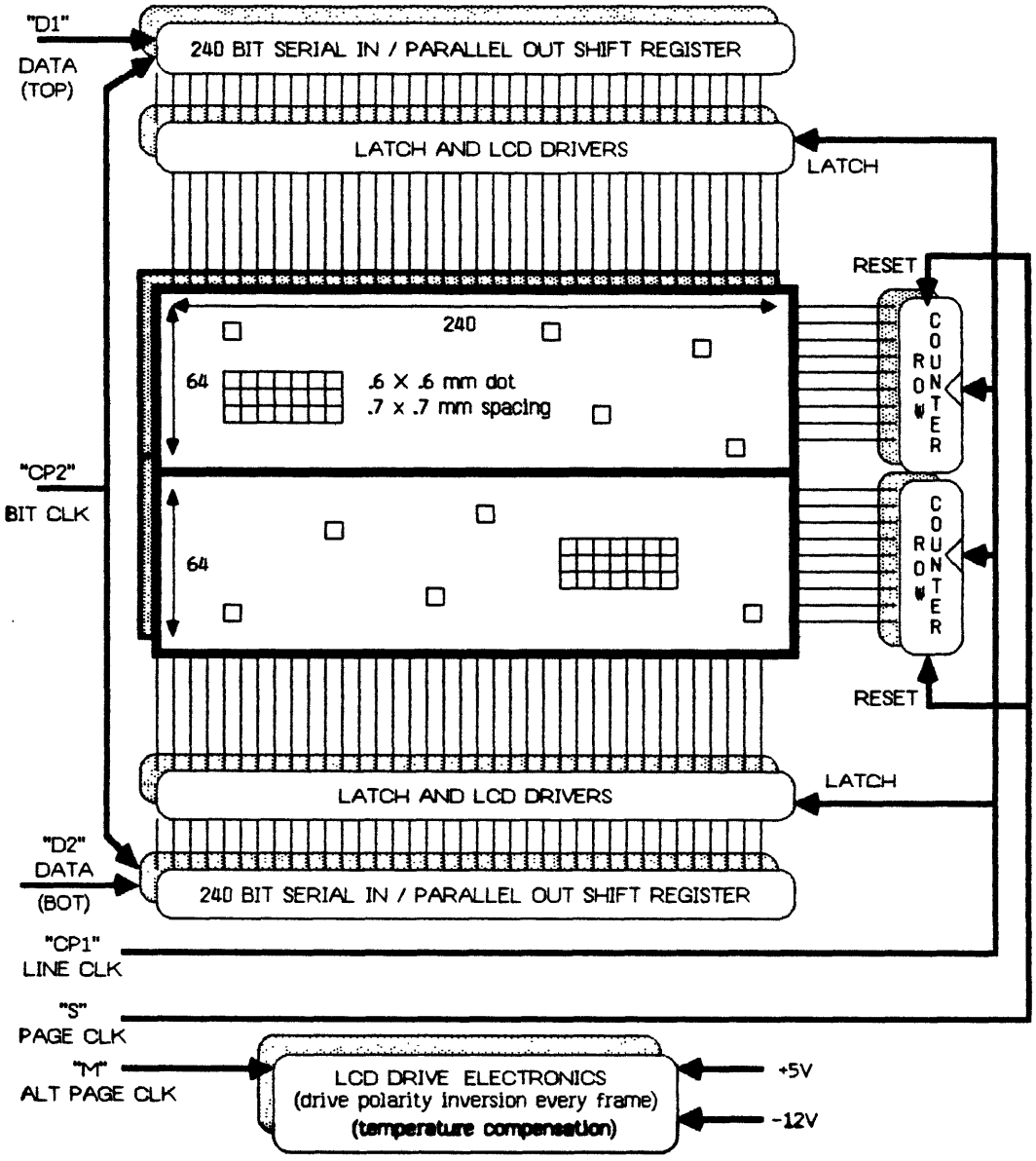


Fig. 8 SHARP DISPLAY

TIMING GENERATOR / ADDRESS COUNTER
BLOCK DIAGRAM

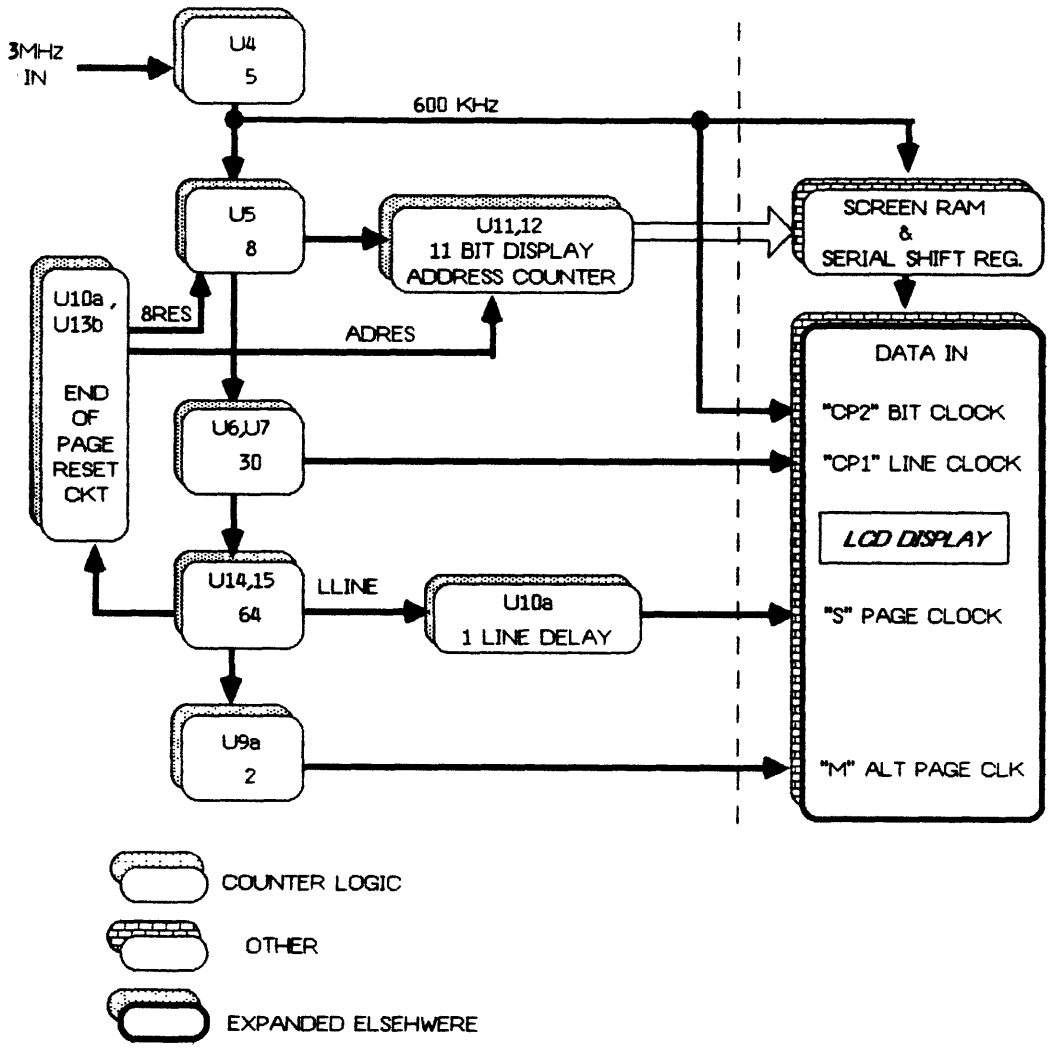


Fig. 9 TIMING GENERATOR

CONTROL LOGIC - BLOCK DIAGRAM

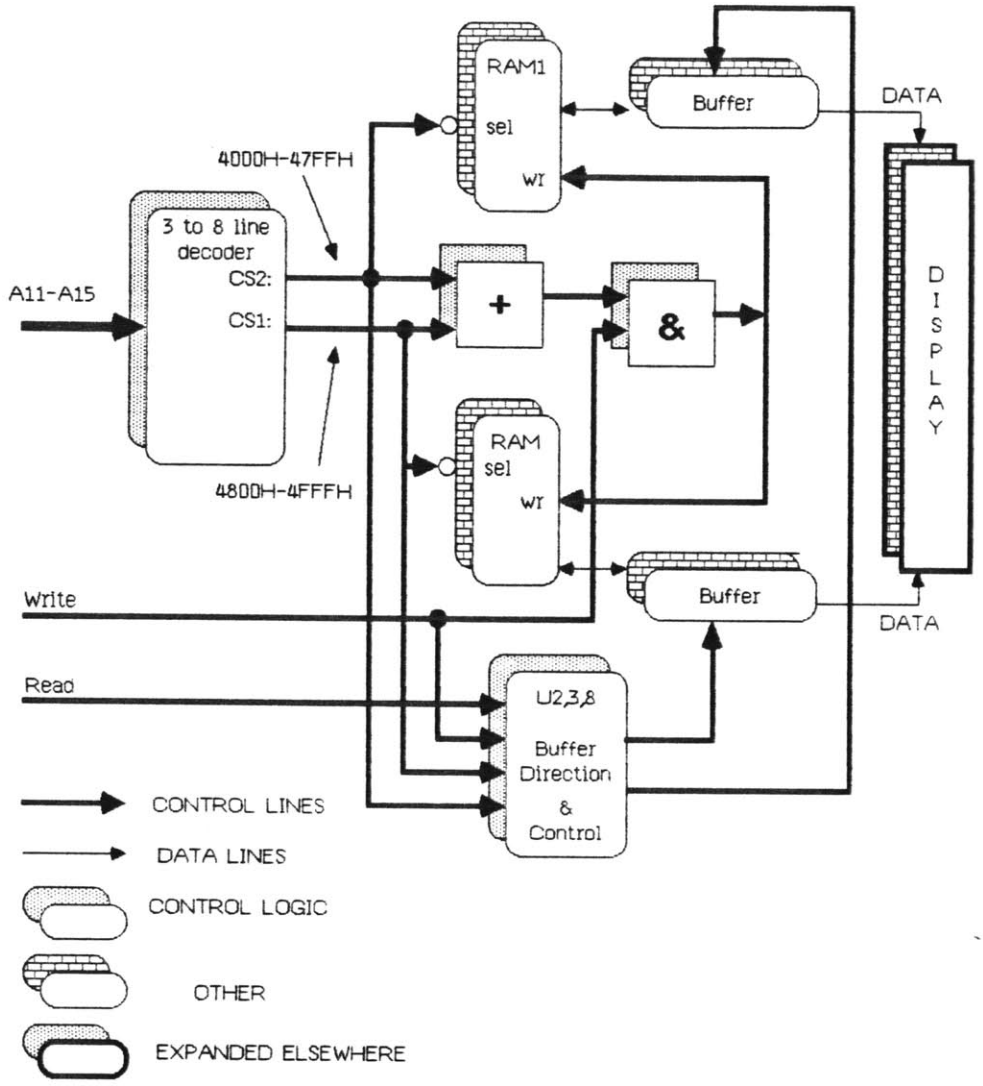


Fig. 10 CONTROL LOGIC

LCD CONTROLLER - BLOCK DIAGRAM

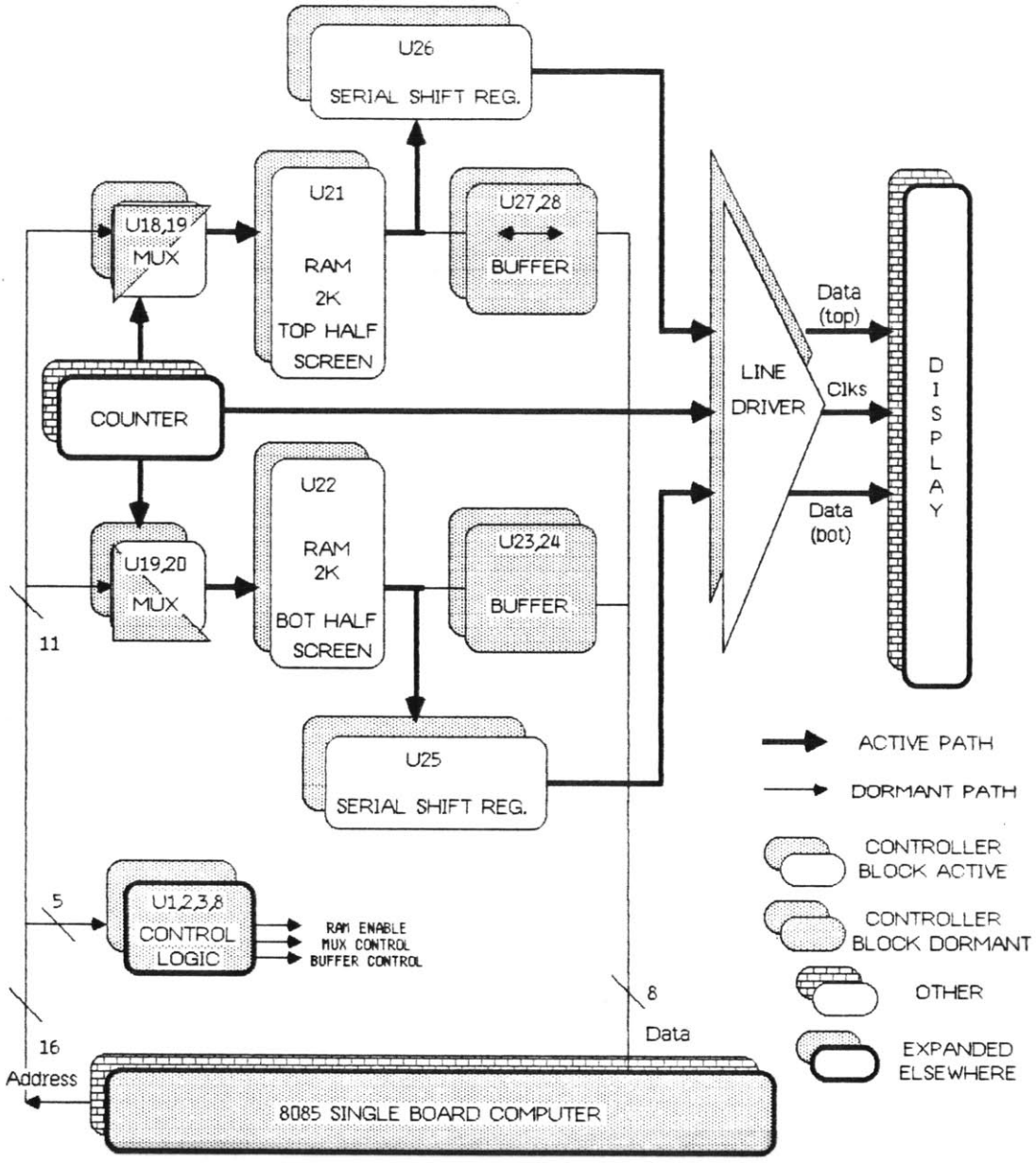


Fig. 11 DATA FLOW DURING DISPLAY REFRESH

LCD CONTROLLER - BLOCK DIAGRAM

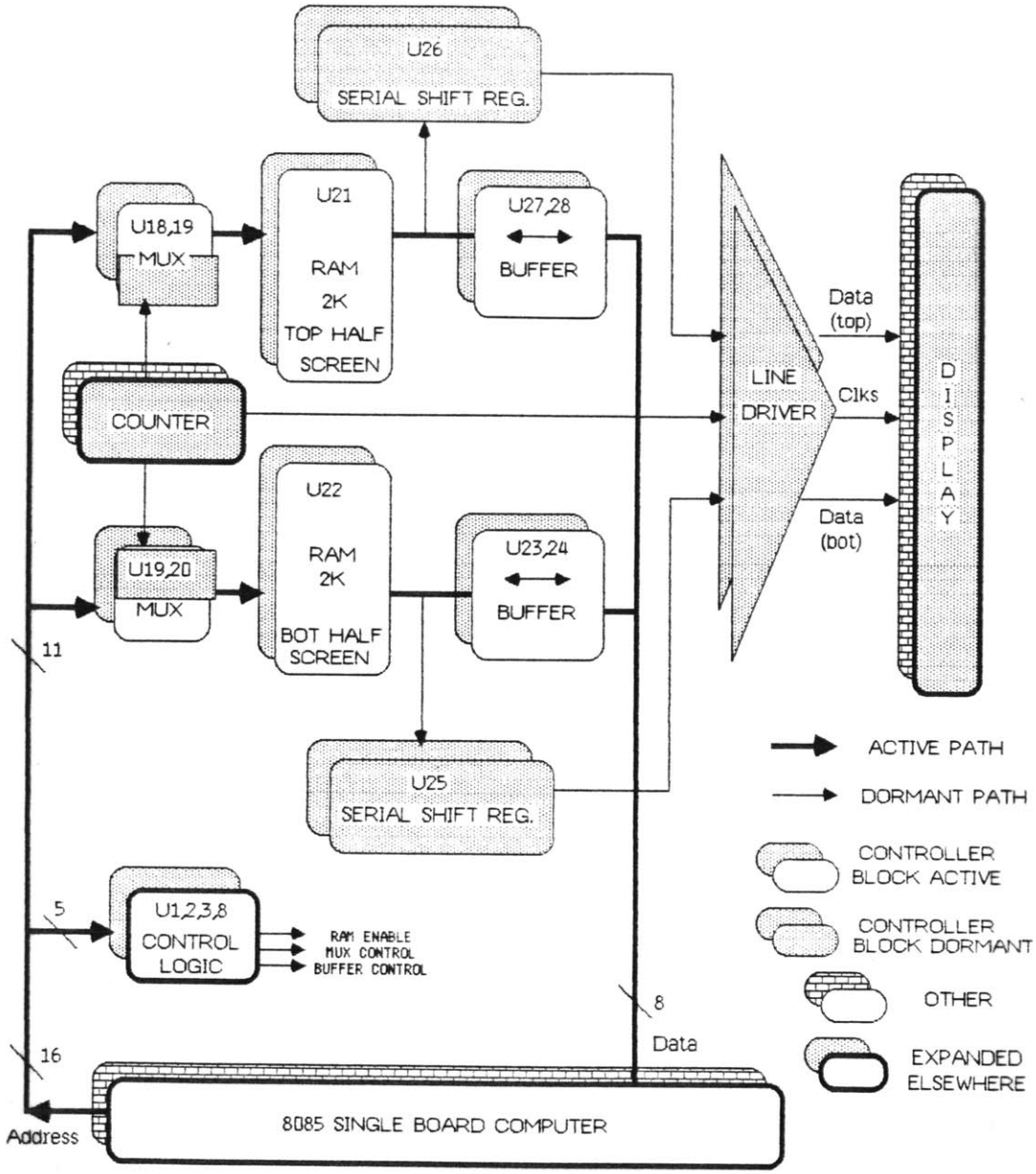


Fig. 11 DATA FLOW DURING RAM ACCESS

3.3 PROGRAM DESCRIPTION

The LCD program is organized around a command recognizer which accepts data from the serial port and calls the appropriate subroutines.

INITALIZATION

INIT: -- Entry point for cold start.

GO: -- Warm start entry point. Does not disturb currently defined symbols.

COMMAND RECOGNIZER

GETCMD Commands are read from the input buffer by executing RDBUF. A lookup is done in the table, CTAB to find the address of the command routine, which is then passed control. An unrecognized command causes a jump to the error routine which sends a '*' to the console.

COMMAND IMPLEMENTING ROUTINES

For each of the characters recognized by GETCMD, a command routine exists to process the data in the command string, and call the appropriate graphics subroutines.

MOVE GRAPHICS ROUTINES

POINT allows any pixel on the screen to be turned on or off.

PUT places a graphics symbol on the screen at the specified coordinates.

PUTX places a graphics symbol on the screen by performing an exclusive-or (XOR) with the data currently on the screen. This allows a symbol to overlap another, and it

may easily be removed by doing a second XOR.

The subroutine DROTR shifts the accumulator right from one to seven places. Carry-in is from the C register and carry-out is placed back in the C register. The value in RMASK must be valid prior to calling DROTR by executing GETMSK. This is done to avoid selecting the correct mask on every call of DROTR.

SCREEN ADDRESSING ROUTINES

At a specified pixel, SCADR calculates the address of display RAM, and location of this pixel within the indicated word.

INCSAD computes the next address of display RAM within the window defined by the hieght, width, and coordinates of a specified symbol.

SYMBOL STORAGE / LOOKUP ROUTINES

SYMADR determines the storage location of a symbol.

GETSYM accepts data defining a graphic symbol, consisting of its height, width, location, and form.

ADDSYM adds a symbol to the symbol table along with the next free address for storage.

STRING STORAGE / LOOKUP ROUTINES

ADDSTR accepts a string and adds it to the symbol table.

PUTSTR places a string on the screen.

CLRSTR erases a string by printing an equivalent number of spaces.

INPUT / OUTPUT ROUTINES

RDBUF reads one character from the bottom of the input buffer and returns it in the accumulator. The circular buffer is 100_H bytes long. If the buffer is empty and the demo program is not running, the console input routine is called. If, however, the demo program is running (FUNBYT = FF) control is passed back to the demo program.

WRBUF writes one character to the top of the circular input buffer. Buffer overflow causes a jump to the error routine.

DEMONSTRATION PROGRAM

The demonstration program was written to test the graphics program, as well as illustrate the simulation a simple aircraft instrument. As such, it writes ASCII command strings to the input buffer, instead of calling the individual routines. Because control of the display over a RS-232 port would involve sending the same sequence of commands, this program is excellent example of how to define and move symbols and strings.

4. CONCLUSIONS

This project has demonstrated that a fairly low cost, modular, flat panel flight instrument is feasable. The LCD display would be the best choice for the general aviation market where price, size, power consumption, and weight are very critical.

The graphics software currently allows many objects to be moved about on the screen, as well as the display of character strings. Symbol rotations could be simulated by storing copies of the different orientations and switching between them. These functions would allow the generation of quite complex flight instrument symbology.

Many functions could be added to the current system such as:

- Drawing a line between two specified points.
- Adding special attributes to symbols such as automatic blink.
- Adding a touch sensitive panel in front of the display so that mode choices may be selected by touching the screen.

5. BIBLIOGRAPHY

Alan R. Miller, 8080/Z80 Assemble Language Programming , John Wiley and Sons, Inc., 1981.

Microprocessor and Peripheral Handbook , Intel Inc., 1983.

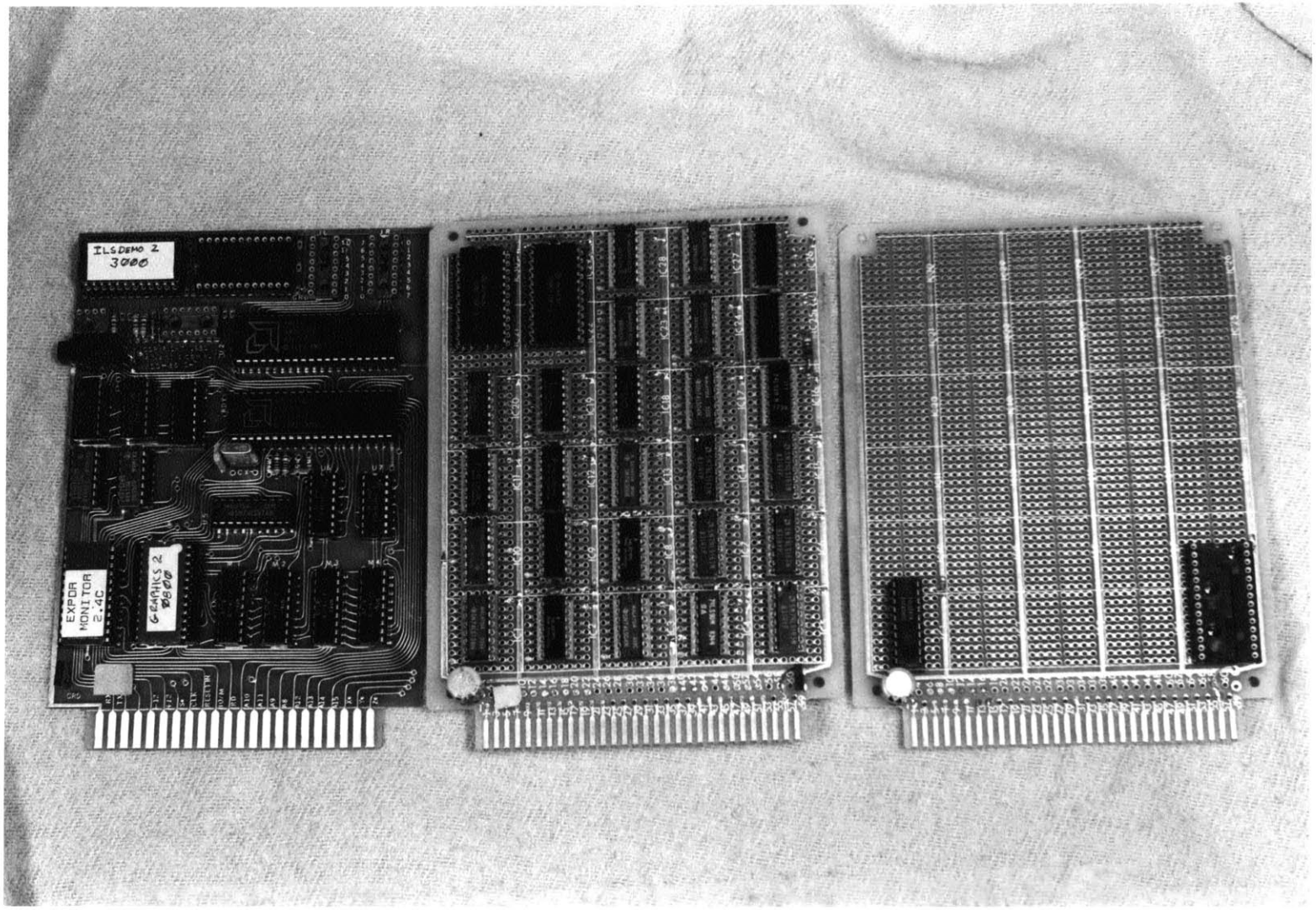
Modern Display Technologies and Applications , AGARD AR-169, North Atlantic Treaty Organization, Advisory Group for Aerospace Reasearch and Development, edited by Prof. Ir.D.Bosman, Twente University of Technology, Netherlands, October 1982.

Motorola Memorey Data Manual , Motorola Inc., 1982.

APPENDIX A.

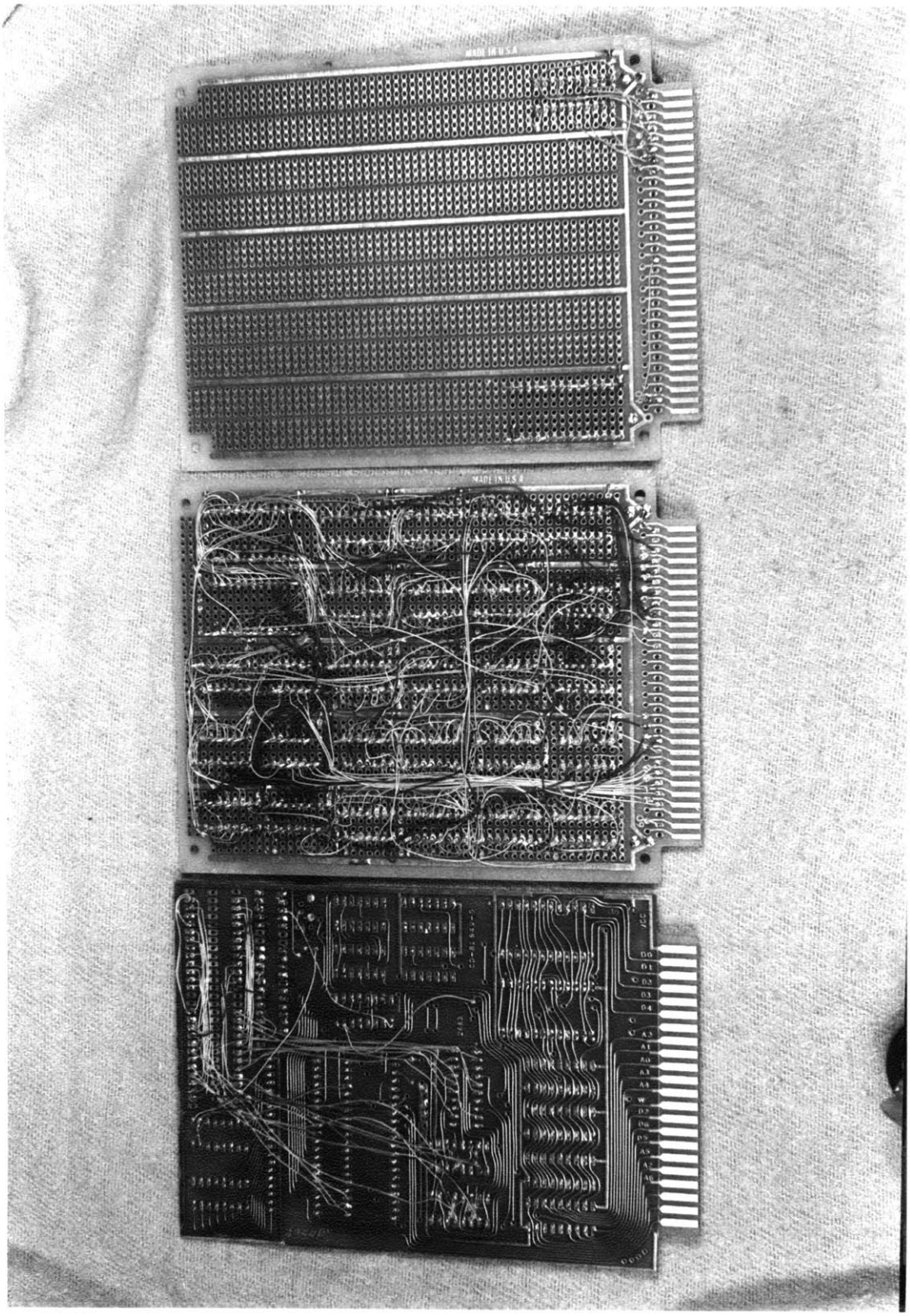
PICTURES

INSIDE GRAPHICS UNIT



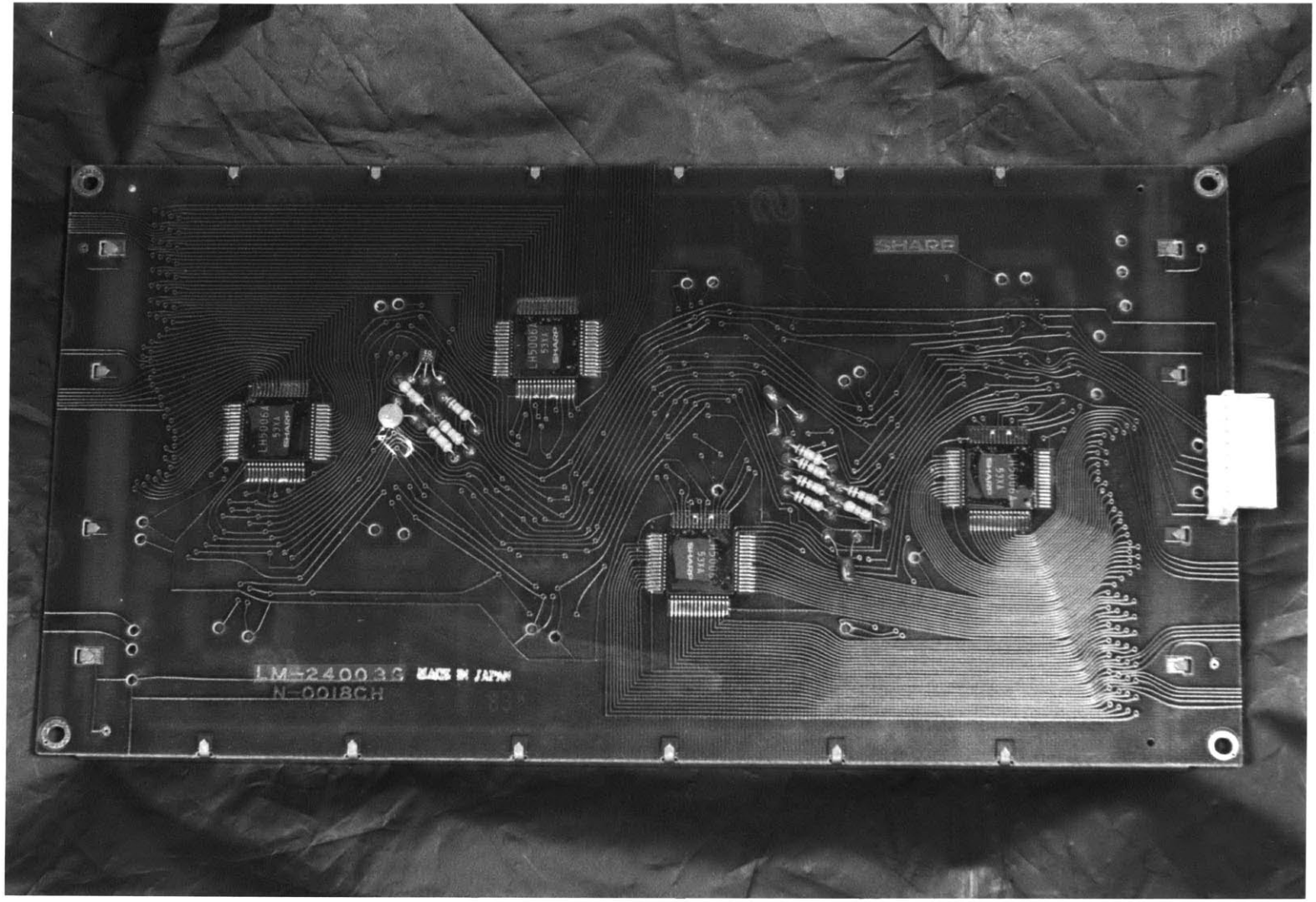
MCG-8085 BOARD A BOARD B

CIRCUIT BOARDS (FRONT)

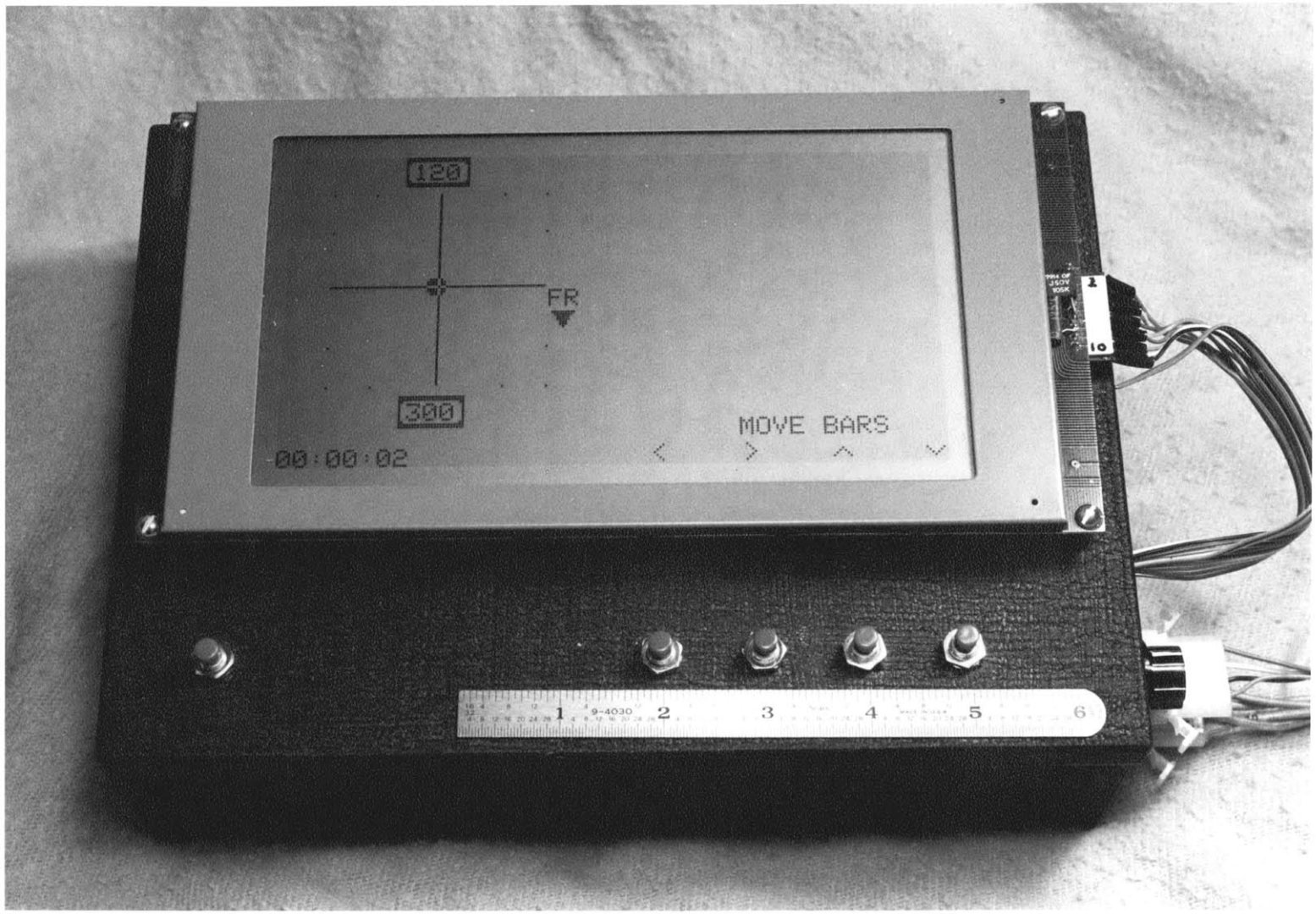


MCG-8085 BOARD A BOARD B

CIRCUIT BOARDS (BACK)



LM-24003G DRIVE ELECTRONICS ON BACK OF BOARD



GRAPHICS DISPLAY UNIT WITH ILS DEMO RUNNING

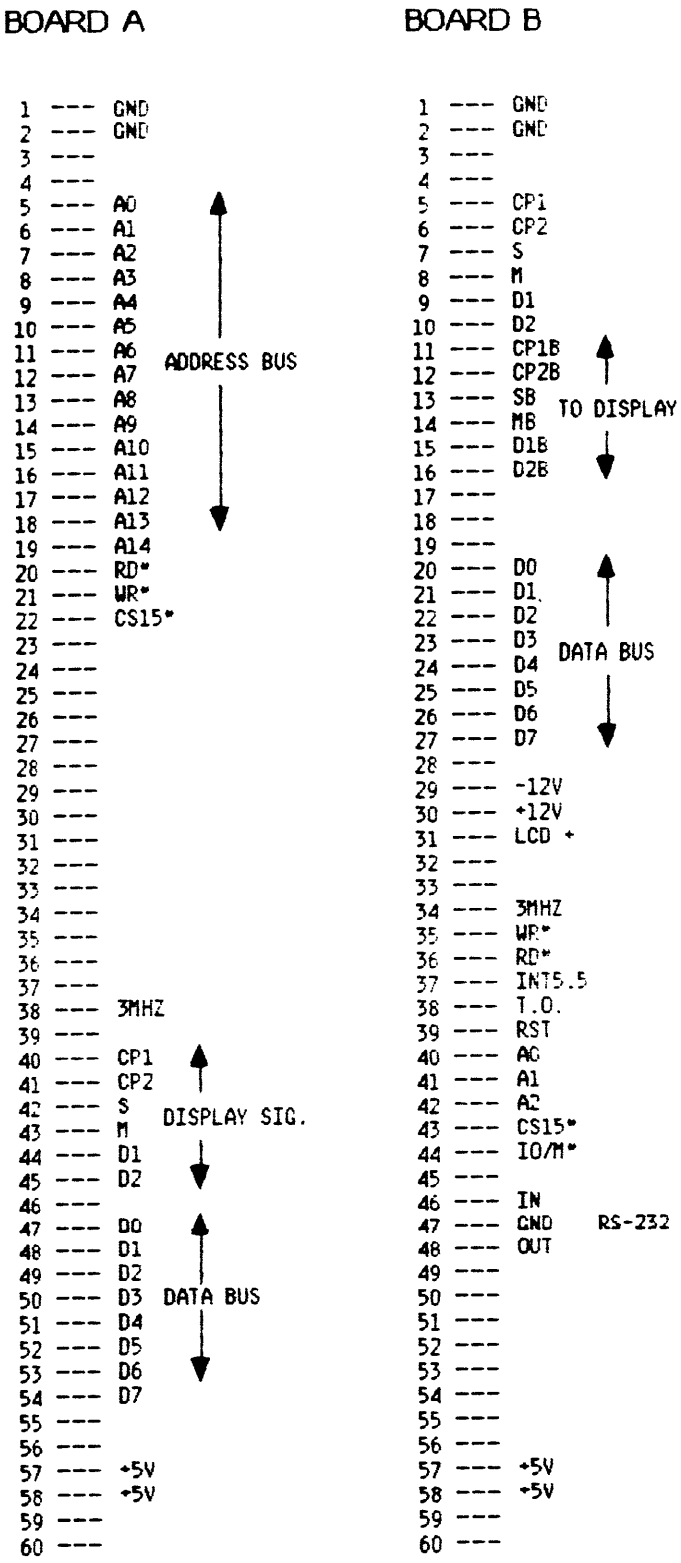
APPENDIX B.

SCHEMATICS AND PARTS LAYOUT

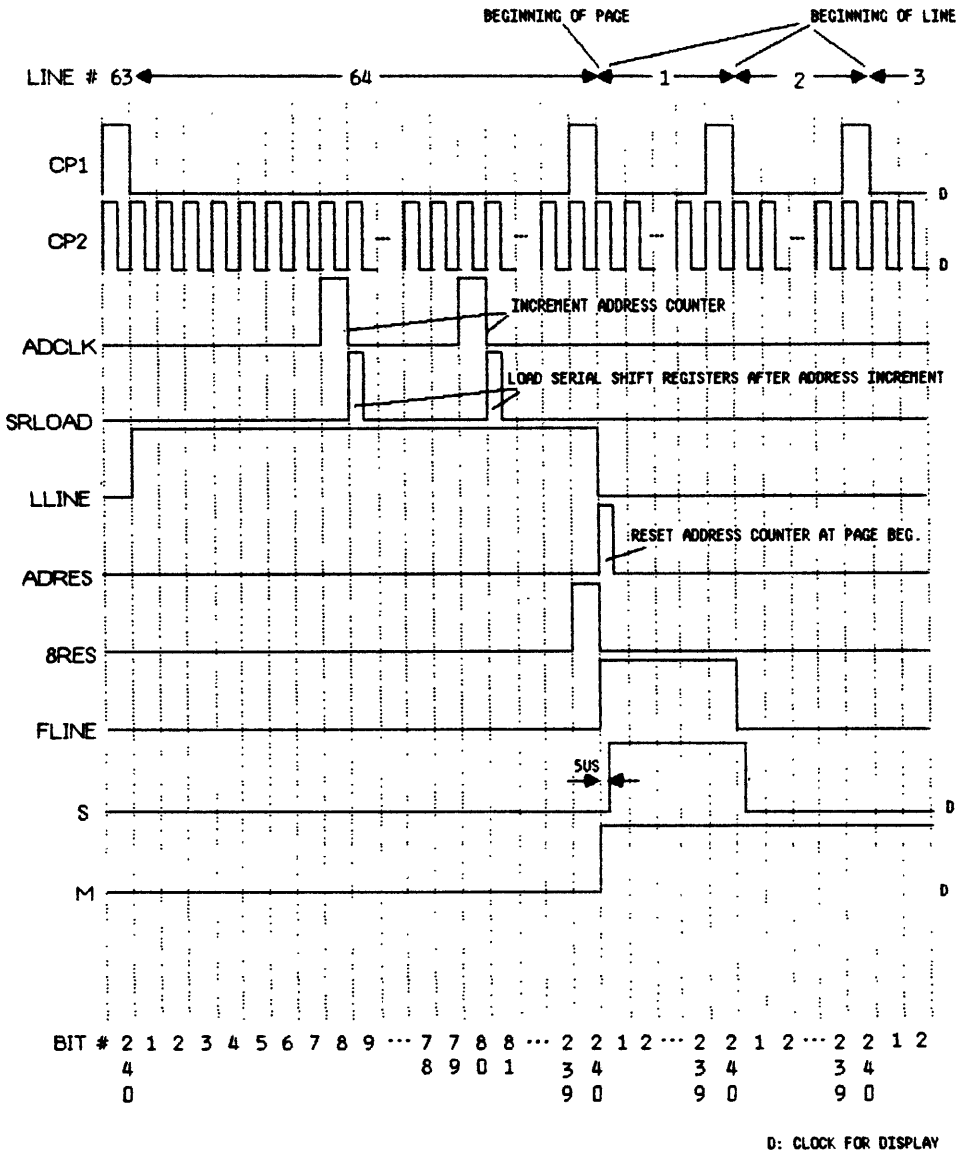
PARTS LIST

U1	- 74S138
U2, 8	- 74LS27
U3, 17	- 74LS04
U4	- 74LS90
U5, 6, 7, 14, 15	- 74LS163
U9	- 74LS74
U10	- 74LS76
U11, 12	- 74LS393
U13	- 74LS00
U16	- 74LS28
U18, 19, 20	- 74LS157
U21, 22	- HM6116
U23, 24, 27, 28	- 74LS243
U25, 26	- 74LS165
U29	- 74LS241
C1	- .001 μF
R1	- 1K Ω

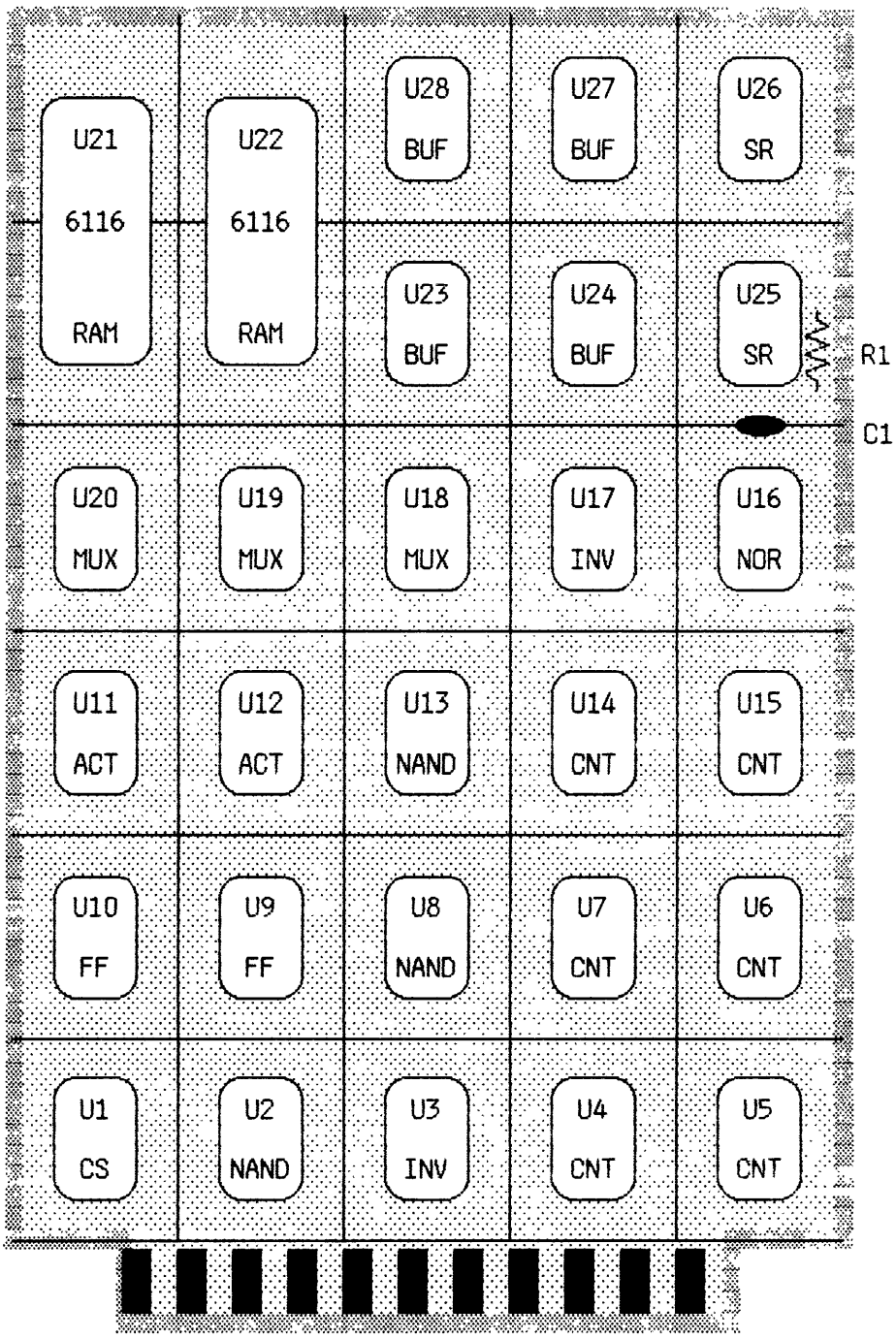
CARD PINOUT


BOARD A
1 --- GND
2 --- GND
3 ---
4 ---
5 --- A0
6 --- A1
7 --- A2
8 --- A3
9 --- A4
10 --- A5
11 --- A6
12 --- A7
13 --- A8
14 --- A9
15 --- A10
16 --- A11
17 --- A12
18 --- A13
19 --- A14
20 --- RD*
21 --- WR*
22 --- CS15*
23 ---
24 ---
25 ---
26 ---
27 ---
28 ---
29 ---
30 ---
31 ---
32 ---
33 ---
34 ---
35 ---
36 ---
37 ---
38 --- 3MHZ
39 ---
40 --- CP1
41 --- CP2
42 --- S
43 --- M
44 --- D1
45 --- D2
46 ---
47 --- D0
48 --- D1
49 --- D2
50 --- D3
51 --- D4
52 --- D5
53 --- D6
54 --- D7
55 ---
56 ---
57 --- +5V
58 --- +5V
59 ---
60 ---
ADDRESS BUS
DISPLAY SIG.
DATA BUS
BOARD B
1 --- GND
2 --- GND
3 ---
4 ---
5 --- CP1
6 --- CP2
7 --- S
8 --- M
9 --- D1
10 --- D2
11 --- CP1B
12 --- CP2B
13 --- SB
14 --- MB
15 --- D1B
16 --- D2B
17 ---
18 ---
19 ---
20 --- D0
21 --- D1.
22 --- D2
23 --- D3
24 --- D4
25 --- D5
26 --- D6
27 --- D7
28 ---
29 --- -12V
30 --- +12V
31 --- LCD +
32 ---
33 ---
34 --- 3MHZ
35 --- WR*
36 --- RD*
37 --- INT5.5
38 --- T.O.
39 --- RST
40 --- A0
41 --- A1
42 --- A2
43 --- CS15*
44 --- IO/M*
45 ---
46 --- IN
47 --- GND
48 --- OUT
49 ---
50 ---
51 ---
52 ---
53 ---
54 ---
55 ---
56 ---
57 --- +5V
58 --- +5V
59 ---
60 ---
TO DISPLAY
DATA BUS
RS-232

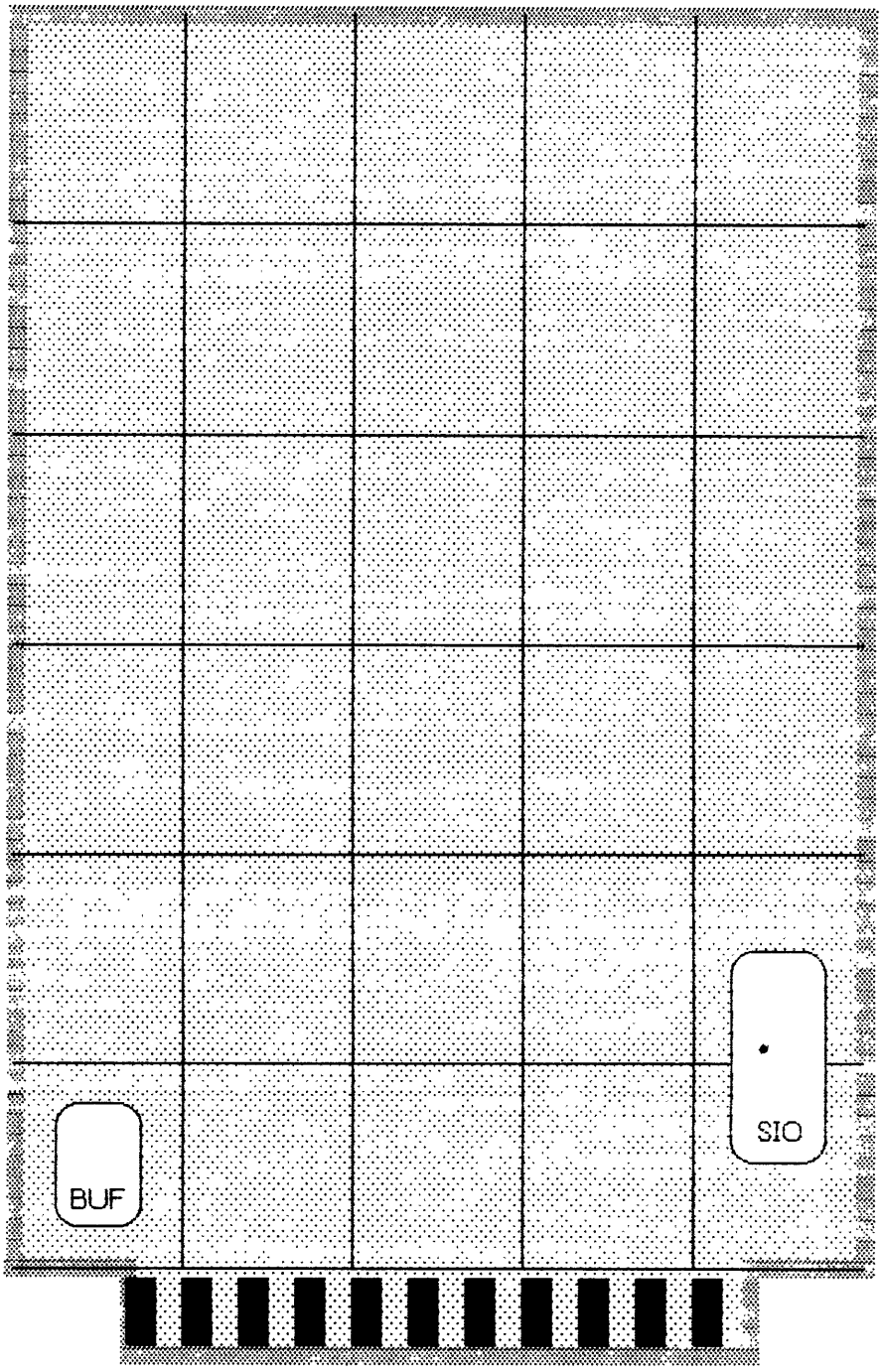
LOGIC TIMING DIAGRAM

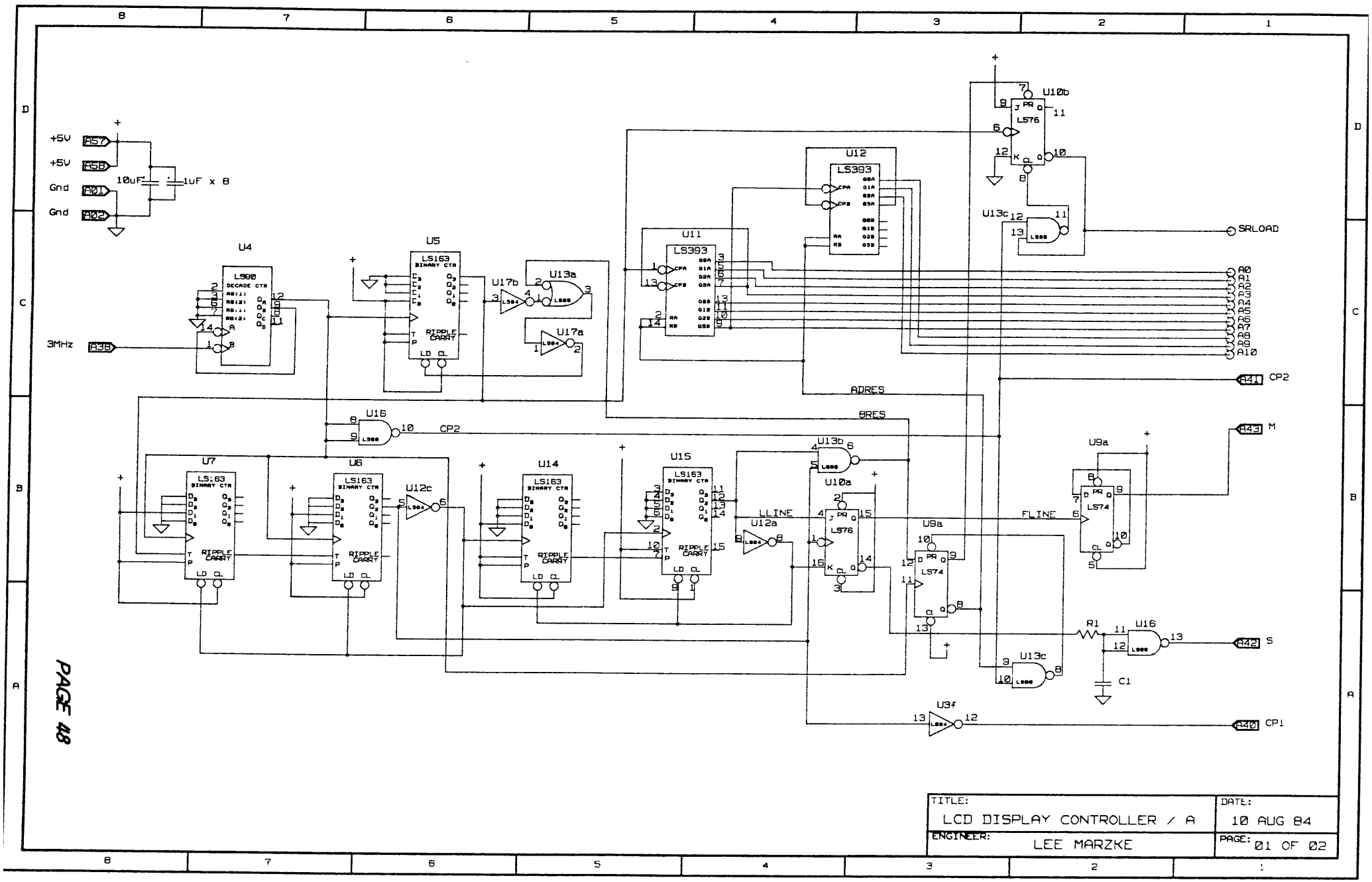


BOARD A - COMPONENT LAYOUT



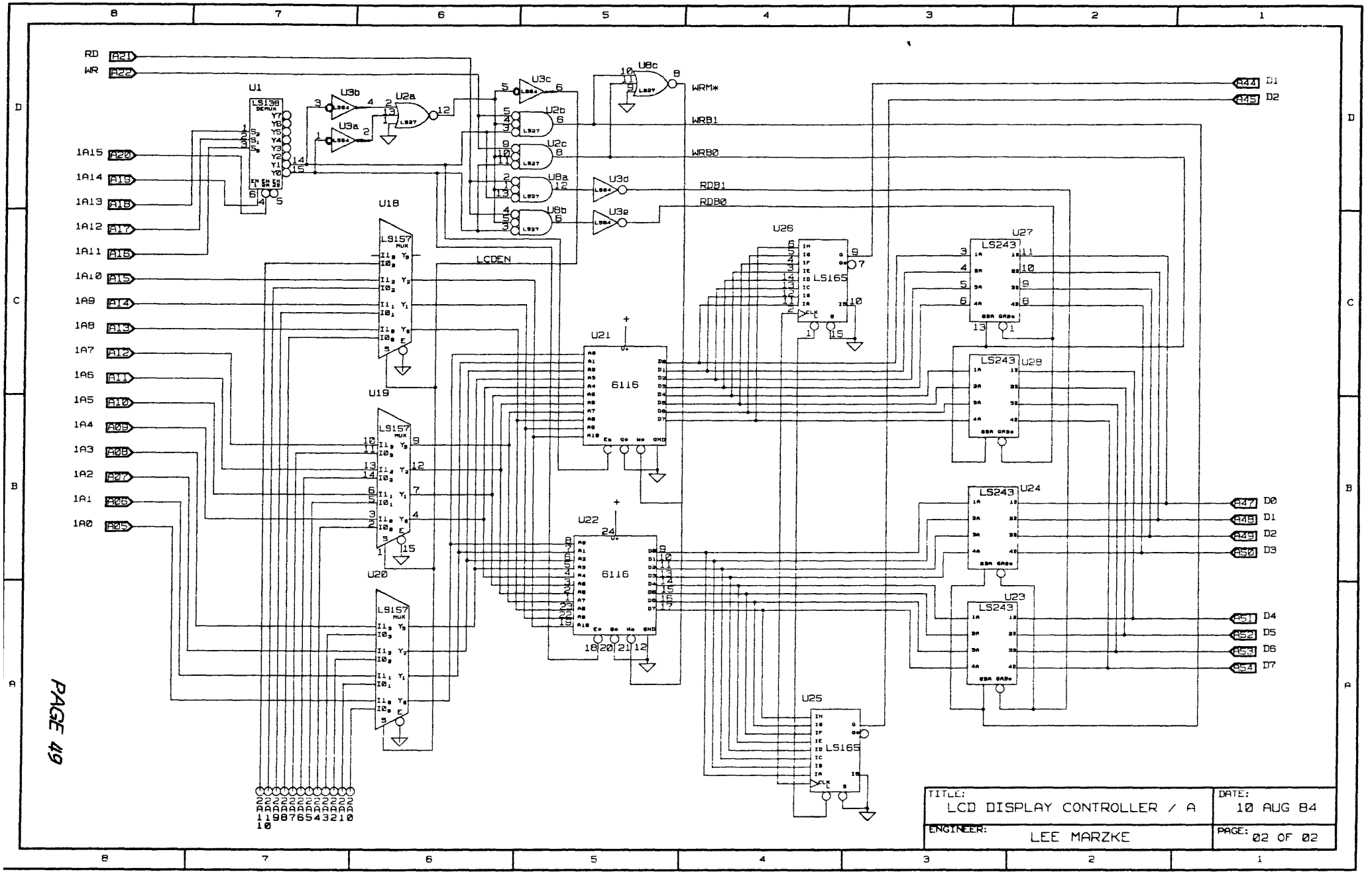
BOARD B - COMPONENT LAYOUT





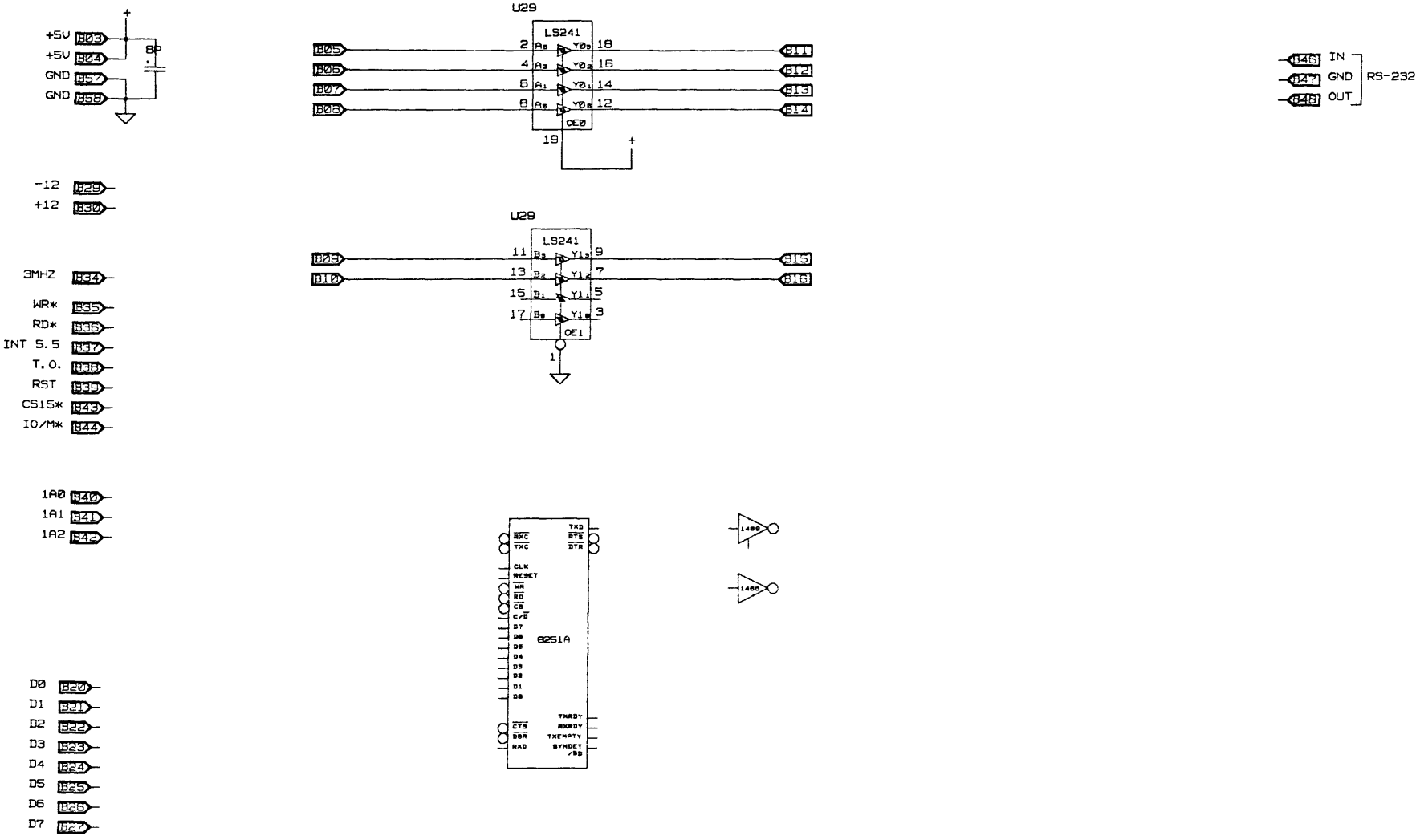
PAGE 48

TITLE:	DATE:
LCD DISPLAY CONTROLLER / A	10 AUG 84
ENGINEER: LEE MARZKE	PAGE: 01 OF 02


RD
WR
A21
A22
U1
LS138
U3b
U3a
U2a
U3c
U2b
U2c
U8a
U8b
U8c
U3d
U3e
WRM*
WRB1
WRB0
RDB1
RDB0
1A15
1A14
1A13
1A12
1A11
1A10
1A9
1A8
1A7
1A6
1A5
1A4
1A3
1A2
1A1
1A0
A20
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A09
A08
A07
A06
A05
U18
LS157
LCDEN
U19
U20
U21
6116
U22
U26
LS165
U27
LS243
U28
U24
U23
U25
D1
D2
D0
D3
D4
D5
D6
D7
A44
A45
A47
A48
A49
A50
A51
A52
A53
A54
PAGE 49
TITLE:
LCD DISPLAY CONTROLLER / A
DATE:
10 AUG 84
ENGINEER:
LEE MARZKE
PAGE:
02 OF 02

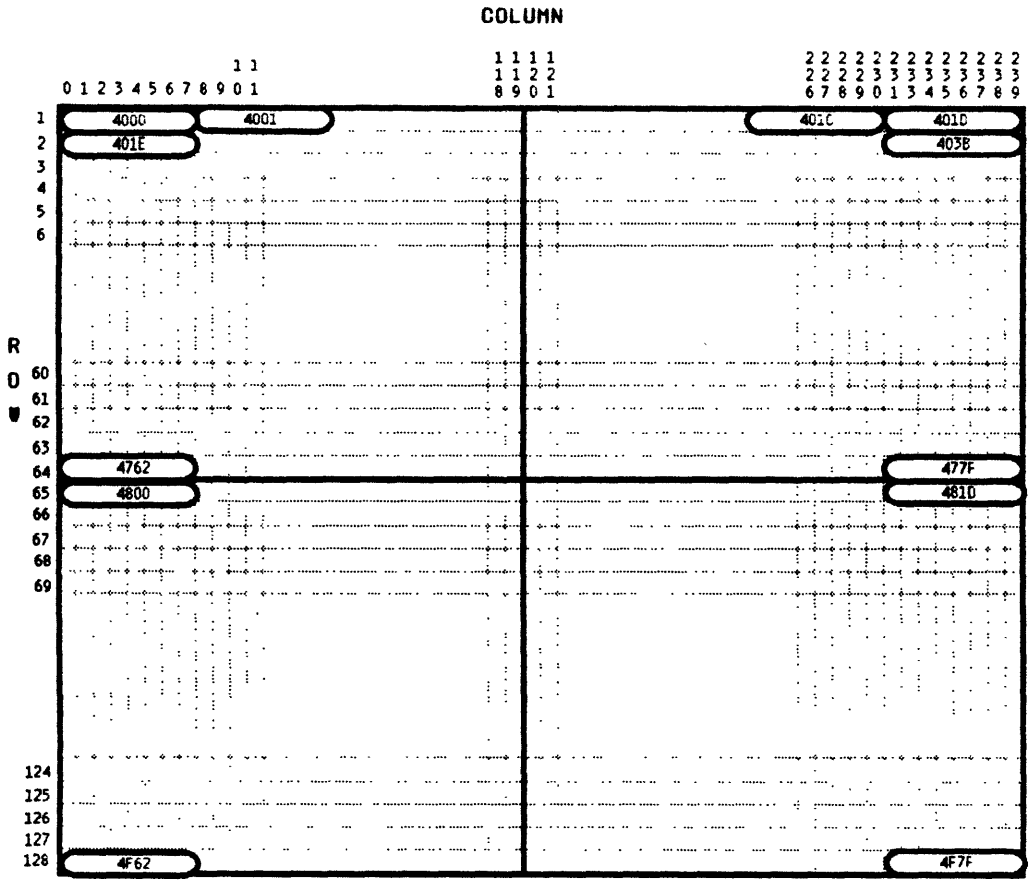
GRAPHIC LCD AIRCRAFT DISPLAY

PAGE 50



TITLE: LCD DISPLAY CONTROLLER / B	DATE: 10 AUG 84
ENGINEER: LEE MARZKE	PAGE: 01 OF 01

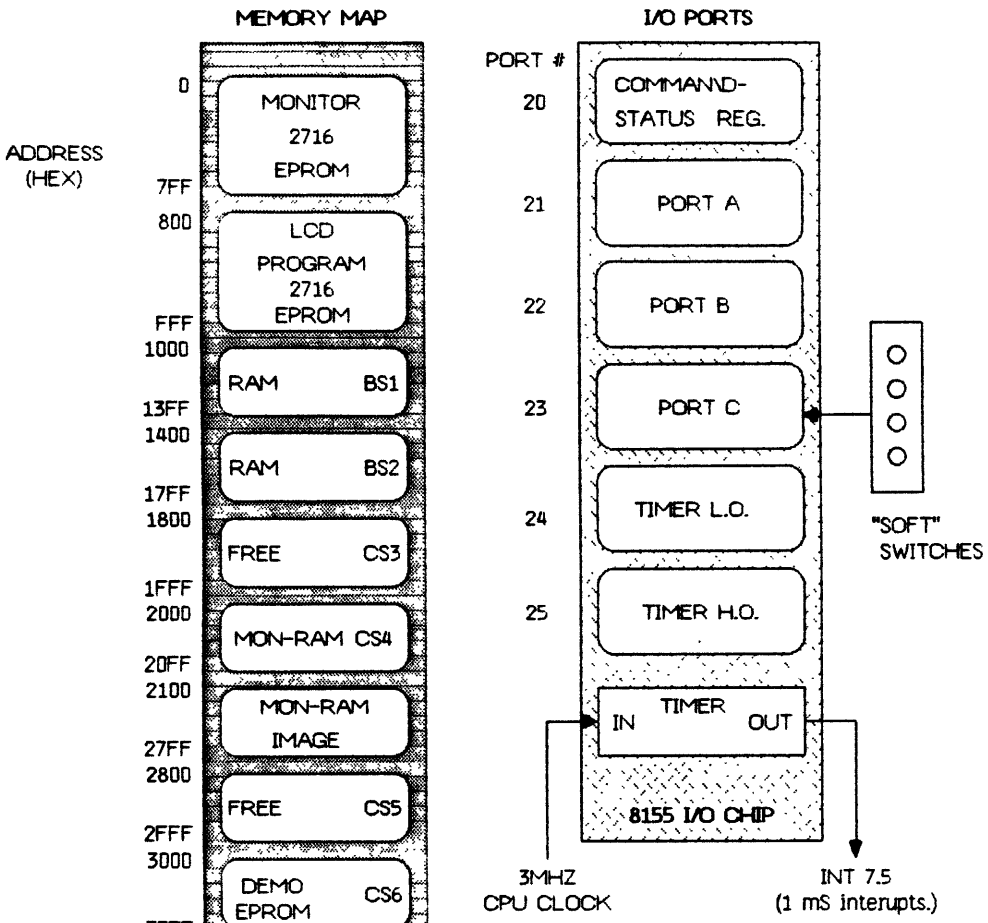
SCREEN MEMORY LOCATIONS



4000 —— MEMORY LOCATION 4000

1 2 3 4 5 6 7 8

8 PIXELS / WORD


MEMORY MAP
ADDRESS (HEX)
0
MONITOR 2716 EPROM
7FF
800
LCD PROGRAM 2716 EPROM
FFF
1000
RAM BS1
13FF
1400
RAM BS2
17FF
1800
FREE CS3
1FFF
2000
MON-RAM CS4
20FF
2100
MON-RAM IMAGE
27FF
2800
FREE CS5
2FFF
3000
DEMO EPROM CS6
37FF
3800
CMOS RAM CS7
3FFF
CPU CARD
4000
DISPLAY RAM
47FF
4800
DISPLAY RAM
4FFF
CONTROLLER
I/O PORTS
PORT #
20
COMMAND-STATUS REG.
21
PORT A
22
PORT B
23
PORT C
24
TIMER L.O.
25
TIMER H.O.
TIMER
IN
OUT
8155 I/O CHIP
"SOFT" SWITCHES
3MHZ CPU CLOCK
INT 7.5 (1 mS interupts.)

APPENDIX C.

SHARP LM-24003G DISPLAY

SHARP LM-24003G
LCD GRAPHIC UNIT

■ Description

The Sharp LM-24003G is a graphic display unit of 240 x 128 full dots which combines a dot-matrix LCD panel and driver C-MOS LSI on a single printed wiring board.

It can display graphs, diagrams and characters as bit mapped patterns It is suitable to various types of equipment, such as, compact OA equipment and measuring instruments which require needs of slim, light weight and low power consumption.

■ Features

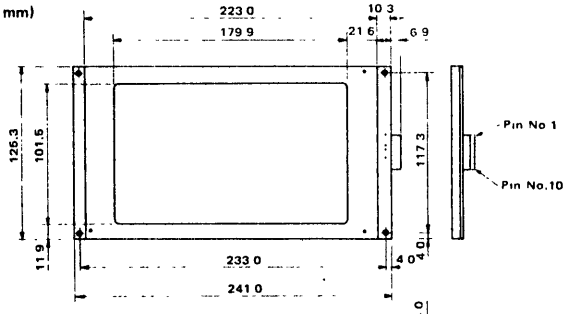
- Displaying graphs and diagrams in addition to characters
- Very easy to be mounted on other equipment due to compact, slim body and low power consumption
- Stable display in wide range of temperature due to built-in temperature compensation circuit
- Easy-to-read display with wide viewing angle and high contrast.
- The unit operates from +5, −12 power supplies.

■ Applications

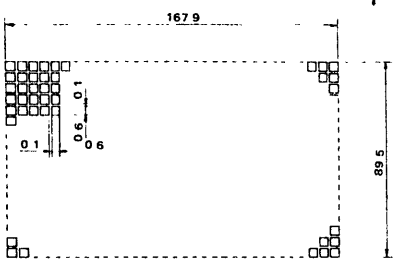
- OA system. Word processor, POS terminal, Computer terminal, etc.
- Measuring instruments: Panel meter, Scaler, Analogue digital tester, etc.

■ Outline Drawing

Outline Dimensions (Units: mm)



Dot Display Details (Units: mm)



■ Mechanical Specifications

Item	Specifications	Unit
Unit outline dimensions	241 (W) x 125.3 (H) x 14 (D)	mm
Effective viewing area	179.9 (W) x 101.5 (H)	mm
Display format	240 (W) x 128 (H) full dots	–
Dot size	0.6 (W) x 0.6 (H)	mm
Dot spacing	0.1	mm
Dot color	Dark blue	–
Background color	White	–
Weight	Approx. 300	gr

■ Absolute Maximum Ratings

Item	Symbol	Min.	Max.	Unit	Remark
Supply voltage (Logic)	$V_{DD}-V_{SS}$	0	7	V	
Supply voltage (LCD driver)	$V_{DD}-V_{EE}$	0	20	V	
Input voltage	V_{IN}	0	V_{DD}	V	
Storage temperature	T_{stg}	−25	+55	°C	
Operating temperature	T_{opr}	0	+50	°C	

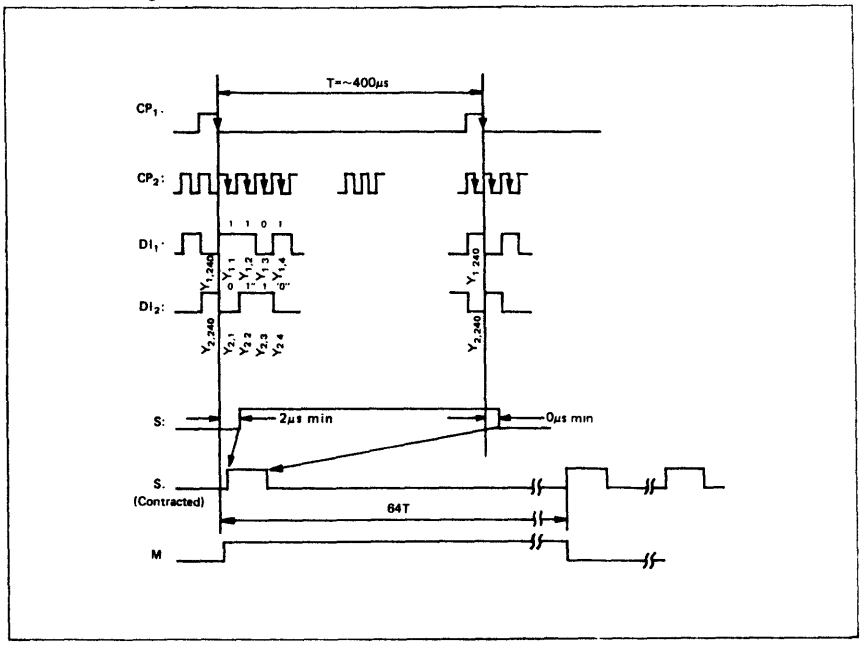
■ *Electro-optical Characteristics

(Ta = 25°C)

Item	Symbol	Min.	Max.	Unit	Conditions
Supply voltage (Logic)	$V_{DD}-V_{SS}$	4.75	5.25	V	
Supply voltage (LCD driver)	$V_{EE}-V_{SS}$	−13	−11	V	
Input signal voltage	V_{IH}	0.7 V_{DD}	V_{DD}	V	"High" level
	V_{IL}	0	0.3 V_{DD}	V	"Low" level
Input leakage current	I_{IL}	–	20	μA	V_{IN}=5V
Power consumption	P_d	–	100	mW	V_{DD}=5V V_{EE}=−12V
Viewing angle	θ_1	40	–	degree	$C_O \geqq 2$
	θ_2	–	15	degree	$C_O \geqq 2$
Contrast ratio	C_o	2.0	**2.5		θ=15°

* Temperature compensation circuit is built-in.
** Typical value

■ Interface Timing Chart



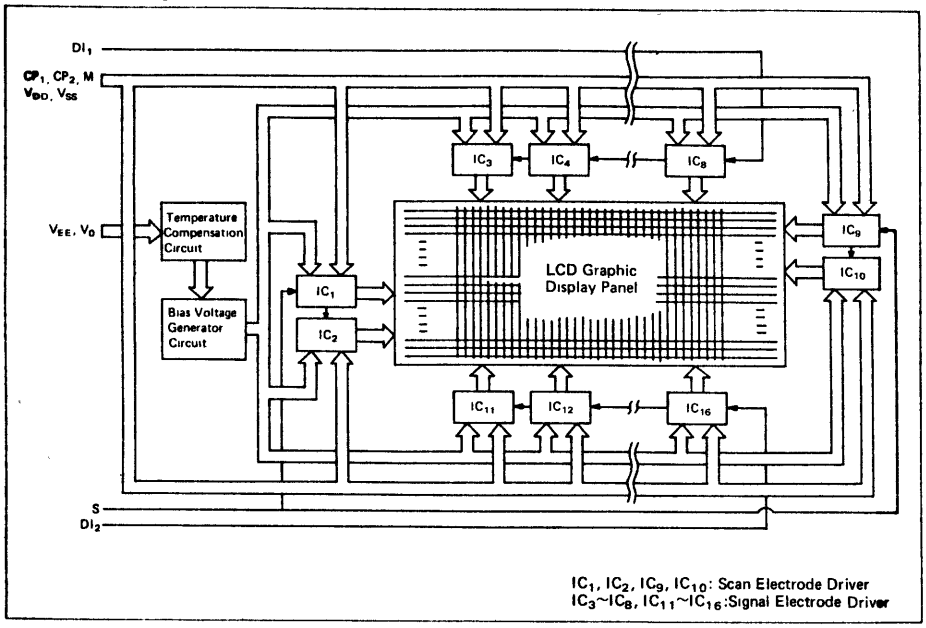
Interface Signals

Connector used: MOLEX 5046-10A
Compatible connector: MOLEX 5051-10

Pin No.*	Symbol	Description	Effective Level
1	S	Scan start-up signal	"H"
2	CP_1	Input data latch signal	H → L
3	CP_2	Data input clock signal	H → L
4	DI_1	Display data signal (Upper half of screen)	H (ON), L (OFF)
5	M	Drive waveform alternating signal	H, L
6	V_O	Power supply for LCD drive(+)	–
7	V_{DD}	Power supply for logic circuit	–
8	V_{SS}	Ground potential	–
9	DI_2	Display data signal (Lower half of screen)	H (ON), L (OFF)
10	V_{EE}	Power supply for LCD drive (–)	–

* For the location of Pin No., refer to outline dimensions.

Circuit Block Diagram



IC_1, IC_2, IC_9, IC_{10}: Scan Electrode Driver
IC_3~IC_8, IC_{11}~IC_{16}:Signal Electrode Driver

(Remarks) This is tentative information and subject to be changed without notice. For detail specifications, contact our sales department.

SHARP
SHARP CORPORATION OSAKA JAPAN
INTERNATIONAL BUSINESS GROUP
ELECTRONIC COMPONENTS SALES DEPT.
22-22, NAGAIKE-CHO ABENO KU, OSAKA 545, JAPAN
PHONE OSAKA (06) 621 1221
TELEX NO AAB LABOMETA J63428(OSKPA)

NORTH AMERICA SHARP ELECTRONICS CORPORATION
10 Sharp Plaza Paramus NJ 07652, USA
Electronic Components Division
PHONE (201) 265-5600
TELEX NO 426903 (SHARPAM PARA)

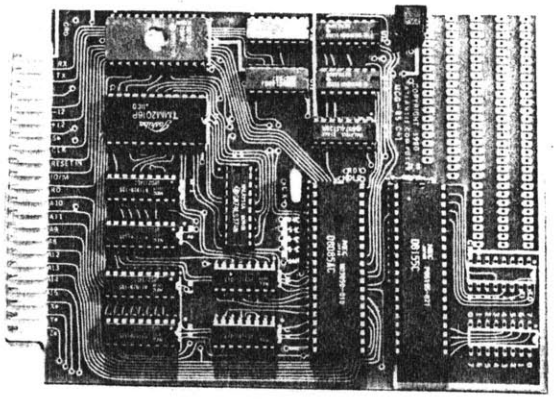
EUROPE SHARP ELECTRONICS (EUROPE) GMBH
Sonninstraße 3 2000 Hamburg 1 F R Germany
Electronic Components Division
PHONE (040) 23775 286/350
TELEX NO 21 61 867 (HEEG D)

Distributed by

APPENDIX D.

MCG-8085 COMPUTER

MCG-85™



THE GENERAL

STANDARD UNIT FEATURES

HARDWARE

- 8085A CPU
- 1 Serial RS232C Port
- 2 Programmable 8 Bit Parallel Ports
- 1 Programmable 6 Bit Parallel Port
- 5 Interrupts, 3 with Maskable Priority
- 2304 Byte RAM Capacity On-Board (2048 can be CMOS RAM)
- 256 Byte RAM Included
- 4096 Byte of ROM Capacity On-Board
- 2048 Byte of RAM Capacity Included (Programmed with Expeditor System Monitor)
- 2048 Bytes of User Selectable RAM or ROM Capacity On-Board
- Programmable 14 Bit Counter/Timer
- Power-On Reset
- External Push-Button Reset Switch
- 6.144 MHz Crystal (Provided with Board)
- Operates on any Crystal Frequency (1 MHz - 10 MHz)
- Data Bus Buffer Capacity On-Board
- 44 Pin Edge Card
- All Chips On Sockets
- Parallel Ports Available on On-Board Sockets
- Address, Data and Control Bus Available at Edge Connector
- Ample On-Board Prototyping Space
- 4 Spare Edge Connector Pins Available
- 6 Spare Pins Available at one of the On-Board Parallel I/O Sockets

SOFTWARE

"EXPEDITOR" 2K SYSTEM MONITOR

- Programmed on 2716 EPROM
- Automatic Baud Rate Detection (50-19.2K Baud)
- CP/M® or ISIS® Systems can be used for Program Development

SYSTEM COMMANDS

- Substitute Memory
- Move Memory
- Display Memory
- Execute
- Fill Memory with Constant
- Kill Echo
- Examine Registers
- Insert
- String Search and Replace
- Assemble
- List Disassemble Code

DOWNLOAD COMMANDS

- Read Hex Paper Tape Format
- Write Hex Paper Format
- Hex Sequential Load
- Ascii Sequential Load

- 9 Terminal I/O and Utility Routines
- 6 Test and Compare Routines
- 4 Code Check or Convert Routines
- Optional Line Assembler on 2716
- Optional Disassembler on 2716

ATLANTIS COMPUTERS DIVISION OF ATLANTIS COMPUTERIZED SERVICES CORP.

31-14 BROADWAY, ASTORIA, N.Y. 11106 (212) 728-6700

HARDWARE DESCRIPTION

THE 8085A CPU

The 8085A CPU is 100% software compatible with the 8080A while offering the benefits of a single power supply, higher integration, higher performance and improved system timing. The 8085A CPU is fully described in the Intel MCS-85 User's Manual. The 8085A derives its timing inputs from the crystal. In addition, the 8085A drives the system with control signals available on-chip. No additional status decoding circuitry is required for most systems. The data bus is multiplexed with the 8 low bits to drive the system's memory components. This is not true in the case of the 8155 which is internally latched and is therefore driven directly from the 8085A.

8085A INTERRUPTS

Four vectored interrupt inputs are available in addition to the standard 8080A type interrupt, (TRAP, INT5.5, INT6.5, INT7.5 and INT). There is also a serial input (SID) and serial output (SOD) data line pair that is used as the system's software uart.

8155

The 8155 is a highly integrated chip designed for compatibility with the 8085A's bus structure. It contains 256 bytes of RAM, 22 programmable parallel I/O lines and a 14 bit counter/timer.

74LS138 (Address Decoder)

The MCG-85 contains a 74LS138 chip that decodes the 8085A's memory address bits to provide the chip enables for the 8155, 2716, 5516, 2114s, 6514s and the 2016.

MCG-85 MEMORY ADDRESSING

Each memory I/O on the basic MCG-85 is enabled from the 74LS138 address decoder. The memory map lists each chip output enable accompanied by the address space over which it is active and the MCG-85 device that is selected. Furthermore, any areas marked open are free for expansion; the 74LS138 has 4 uncommitted chip select lines to allow for the addition of four 2048 byte memory blocks without additional decoding circuitry.

MEMORY CAPACITY AND DESCRIPTION

The total on-board capacity of the MCG-85 is 6K bytes, in addition to the 256 bytes of Static RAM contained on the 8155. The General can accommodate the following memory chips:

2716	24 pin 2048 X 8	5V, 2K EPROM or equivalent
2016	24 pin 2048 X 8	2K Static RAM or equivalent
5516	24 pin 2048 X 8	2K CMOS RAM or equivalent
2114	18 pin 1024 X 4	Static RAMS or equivalent
6514	18 pin 1024 X 4	CMOS RAM or equivalent

For the exact on-board combinations, please reference the MCG-85 Block Diagram. The MCG-85 will accept any memory speed. Memory supplied as a standard feature with each unit is a 2K EPROM (2716) containing the "Expeditor Monitor" and the 256 bytes of RAM available on the 8155.

RS232 SERIAL PORT

The 1488 and 1489 provide the RS232 signal levels for the General. They are the RS232 Driver and receiver respectively. One gate of the 1488 is used for TX data and one gate of the 1489 is used for RX data. This leaves three gates available for future expansion.

PARALLEL I/O PORTS

Included on the MCG-85 are 22 parallel I/O lines (on 8155). They are configured as follows: two 8 bit ports and one 6 bit port (please reference parallel port connector diagram). The ports are available on two on-board 16 pin sockets. Connections can easily be made using 16 pin headers.

8216

The 8216 are data bus buffers (two used) which are installed if the user wishes to drive additional off-card memory peripherals. They are not necessary for system operation.

ON-BOARD PROTOTYPING SPACE

The MCG-85 has a small area on-board dedicated for prototyping. There are 122 pre-drilled holes with solder mask on both sides that can accommodate chips with different pin configurations.

SOFTWARE DESCRIPTION

All the software for communication, program development and program storage, is provided standard with each unit, in the Expeditor System Monitor. The software features are listed on the front cover. For the system's operation an RS232 type terminal and a power supply are required. When properly interfaced to the aforementioned hardware, the General, during "bootup" will automatically calculate the baud rate of the terminal (50 - 19.2K baud) and transfer the program to the command recognizing routine. At this point, the General will accept any valid command.

SOFTWARE DEVELOPMENT

Knowledge of 8085 Machine language is required, in order to program the General directly, using the commands and utility routines on the Expeditor Monitor. In our User's Manual, we provide useful examples of sample programs (parallel to serial, serial to parallel conversions, data acquisition, interrupt driven A/D, remote control of the MCG-85 from host computer, etc.) which will suggest program development methods and potential uses for the MCG-85.

*PROGRAM DEVELOPMENT UNDER CP/M®

The General, through the download commands (R&W), will enable the user to develop programs on a CP/M based computer. The majority, if not all of the CP/M disks existing contain an 8080 assembler. This assembler is 100% compatible with the 8085 instruction set. By using this assembler and taking advantage of the more powerful editor of the CP/M based computer, programs can be developed quickly, without the need of a development system. Subsequently, at each stage of development, the code can be downloaded into the MCG-85 by using the "PIP" CP/M command and the "R" (Read Hex Paper Tape Format) command of the MCG-85 and tested without the need for an EPROM to be burned each time. After each work session, the programs developed for the General can be stored on the floppy disk of the CP/M based system for safe keeping and recall for later use thus eliminating the need for retyping the code into the General from the beginning each time.

UPLOADING

Using the "W" (Write Hex Paper Tape Format) command will enable the user to save programs developed on the General on a system that supports a disk drive. Through this method, machine language programs can be saved and later retrieved through downloading thus eliminating the need for reentering code.

THE GENERAL IN DEDICATED SYSTEMS

In the event that the MCG-85 is intended to be used as a Dedicated Controller, and because it has a software UART, not a hardware UART, the UART part of our monitor may be copied and incorporated into the Dedicated Controller program. It is about 100 bytes long and can very easily be integrated into the software.

This method of program development can also be used with an ISIS based or any other computer which will generate 8080 code in Hex Paper Tape Format

THE GENERAL AT A GLANCE

NAME: The Micro Computer General 85 — MCG-85.
PHYSICAL DIMENSIONS: 4.5 x 6.5 inches.
PRINTED CIRCUIT BOARD MATERIAL: Fire Retardant Glass Epoxy.
SOLDER MASK: Both Sides.
TYPE OF CHIPS USED: 1st Grade Commercial.
SUPPLIERS: Only Franchised Distributors. References on Request.

ELECTRICAL CHARACTERISTICS

CURRENT REQUIREMENTS FOR FULLY CONFIGURED BOARD	+5V at 1A ± 12 at 20MA

FUNCTIONAL CHARACTERISTICS

STANDARD UNIT	Completely functional computer card monitor software in ROM and illustrative User's Manual.
ADDITIONAL REQUIREMENTS FOR OPERATION	3 voltage power supply and RS232 terminal.
BAUD RATE	50-19.2K baud rate. The baud rate of the terminal used is calculated automatically by the Expeditor System software.
CRYSTAL	6.144 MHz crystal provided with the MCG-85 (can operate with crystal frequencies from 1-10MHz with no modifications).
PARALLEL I/O	Dip socket access for the two 8 bit and one 6 bit parallel ports.
SERIAL PORT	RS232 port available at edge connector.
TIMER PORT	Dip socket access to the 8155 timer.
EDGE CONNECTOR BUS	ATLANTIS COMPUTERS 44 Pin MCG Bus™.
CONNECTOR USED	44 pin edge connector; .156 centers.
APPLICATIONS	Industrial controls; dedicated test and monitoring systems; communication subsystems; small scale data processing; A to D, D to A conversions and processing; data logging; data acquisition; prototyping and experimentation.
SUPPORT CARDS AND RELATED PERIPHERALS	The MCG-85 is supported by a full line of cards, all on the MCG Bus™ which is the standard we have established; see inserts for all available cards.

CALL US FOR DETAILED INFORMATION

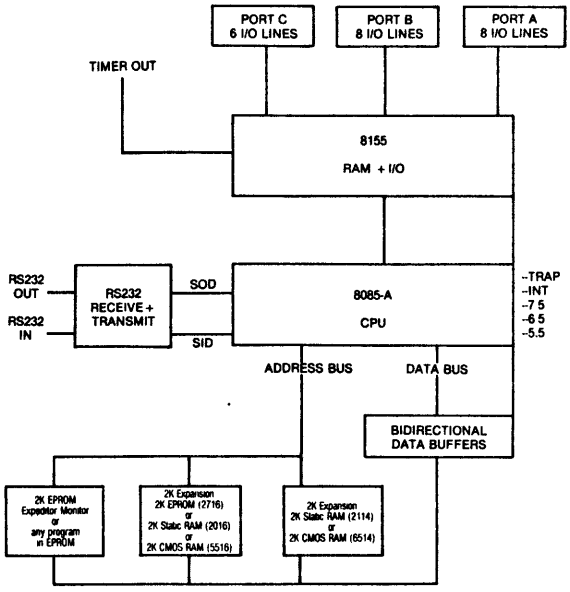
ATLANTIS COMPUTERS' design engineers will be glad to work with you and investigate your specific applications and requirements. Call direct (212) 728-6700 or write ATLANTIS at the address below.

PRODUCT DEVELOPMENT A PROBLEM?

Before you hire expensive outside consultants, give us a call. We can design, prototype and develop customized software for any project involving the MCG-85.

ATLANTIS COMPUTERS DIVISION OF ATLANTIS COMPUTERIZED SERVICES CORP
31-14 BROADWAY, ASTORIA, N.Y. 11106 (212) 728-6700

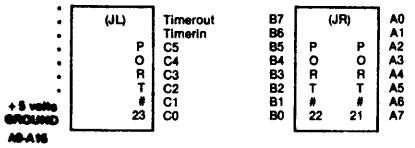
MCG-85 SINGLE BOARD COMPUTER



A. EDGE CONNECTOR PINOUT

FUNCTION	CODE	PIN	PIN	CODE	FUNCTION
Ground	GRD	1	A	Vcc	+5 Volts
Receive data	RX	2	B	D0	
Transmit data	TX	3	C	D1	
EPROM Programming Voltage	EPV	4	D	D2	
−12 volts	−12v	5	E	D3	
+12 volts	+12v	6	F	D4	
	*	7	H	*	
Clock	CLK	8	J	A2	
	RESET IN	9	K	A1	
	IO/M	10	L	A0	
Read	RD	11	M	A3	
	A10	12	N	A4	
	A11	13	P	W	Write
	A9	14	R	D7	
	A8	15	S	D6	
	A12	16	T	D5	
	A13	17	U	A5	
	A14	18	V	A7	
	A15	19	W	A6	
	*	20	X	INT7.5	
	*	21	Y	INT6.5	
	RDY	22	Z	INT5.5	

B. PORT CONNECTOR PINOUT

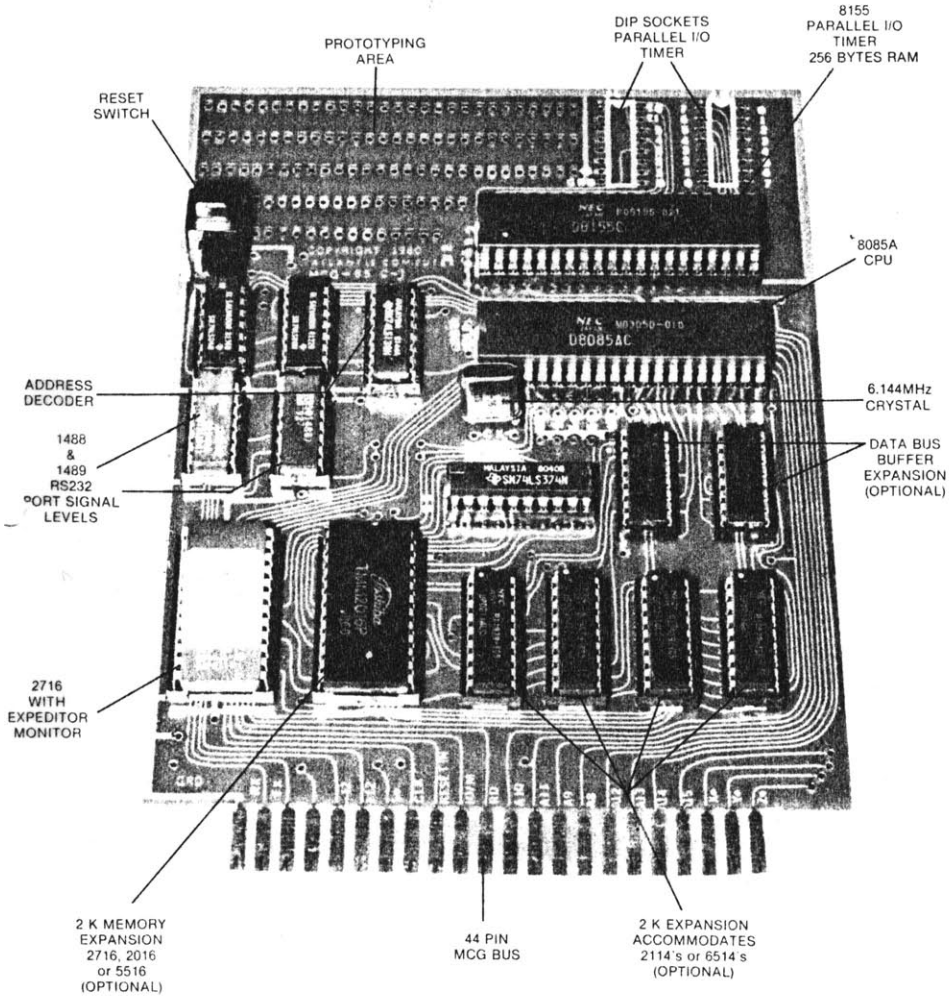


A0-A15 ADDRESS BUS, D0-D7 BIDIRECTIONAL DATA BUS
*Available pins for user modification

MCG-85 MEMORY MAP

0000	MONITOR ROM (2K)	CS0
07FF		
0800	EXP EPROM (2K)	CS1
0FFF		
1000	EXP RAM1 (1K)	BS1
13FF		
1400	EXP RAM2 (1K)	BS2
17FF		
2000	BASIC RAM (256 BYTE) CS4	
20FF		
	BASIC RAM IMAGE	
27FF		
1800	OPEN (2K)	CS3
1FFF		
2800	OPEN (2K)	CS5
2FFF		
3000	OPEN (2K)	CS6
37FF		
3800	OPEN (2K)	CS7
4000		

MCG-85™



ATLANTIS COMPUTERS DIVISION OF ATLANTIS COMPUTERIZED SERVICES CORP
31-14 BROADWAY, ASTORIA, N.Y. 11106 (212) 728-6700

A. GENERAL COMMANDS

*Substitute command. SXXXX (SP) :

The substitute command allows the user to modify memory locations individually. The command works as follows.
a) Type an S followed by a 4 hex digit address of the first memory location you wish to examine, followed by a space.
b) The contents of that memory location are displayed, followed by a dash.
c) To change the contents of the location displayed, type in the new data, followed by a space. The contents of the next higher memory location will automatically be displayed and be available for modification.
d) Type a carriage return. the S command will automatically be terminated.

*Move memory command. MXXXX,YYYY,ZZZZ (cr) :

The move memory command moves the contents of memory between hex locations XXXX and YYYY to destination field starting at address ZZZZ.

*Display memory command. DXXXX,YYYY (cr) :

The display memory command will produce a formatted listing of the memory contents between XXXX and YYYY. Each line of the listing begins with the address of the first memory location displayed on that line, represented by four hex digits followed by up to sixteen memory locations each one represented by two hex digits.

*Execute command. GXXXX (cr) :

Control is transferred from the monitor to the program starting at address XXXX. If XXXX is not specified the monitor uses as an address the value on top of the stack.

*Fill memory with constant. FXXXX,YYYY,DD (cr) :

The fill command will put a single byte constant denoted by DD in all the memory locations between XXXX and YYYY.

*Kill echo command. K (cr) :

Initially the MCG-85 will echo characters that are received from the console device. Upon issuing the K command the MCG 85 will toggle the echoing on and off. MCG-85 will not echo received characters after the K command has been issued. This will allow communication with devices that operate in HALF DUPLEX mode, also this command will insure trouble free operation (no missing characters) when downloading.

*EXamine registers command. X (register identifier)

Display and modification of the CPU registers is possible with this command. The X command uses a register identifier to select the particular register to be examined. A register identifier is a single alphabetic character (A,B,C,D,E,F,I,H,L,S,P,M) .

*Insert command. IXXXX (cr) :

Insert command can be used to insert a single instruction or an entire proram in memory. The monitor waits for the user to type in a string of hex digits. Each digit in the string is converted to its binary value, and then loaded into memory. Two hex digits are loaded into each byte of memory.

*sTring search and replace T XXXX (cr).
Promts user for search string, then finds the first occurance of that string starting at address XXXX. Displays the address of the string occurence at which time the user has the option of replacing the string by typing in the new string or the user can go on to the next occurence of the search string by hitting (cr). The command terminates when user types (esc).
[e.g. You wish to change all calls to a subroutine which was originally at 2080 but is now at 20A4. If your program starts at 2000 you would type T2000(cr). The monitor would promt you for the search string in this case that would be CD8020. The monitor would then search for the first occurance of the string, at which time it would print its address on the console and allow you to change contents of memory starting at that address. To continue search type (cr), to end search type (esc)]

*Assemble AXXXX(cr).
If the optional line assembler is mounted on the MCG-85, then assemble user lines starting at address XXXX.

C. TERMINAL I/O ROUTINES + UTILITY ROUTINES

*CI (Character input) [0065H]
Waits for an ASCII character to be received from the console and returns its value in the A register.

*CO (Character output) [0068H]
Transmits the ASCII character in C to the console. The character is printed at the current print position.

*HEXOUT (Hex number printer) [006BH]
The 8 bit quantity in A register is printed at the console as two hex digits.

*CROUT (Carriage return line feed) [006EH]
Sends carriage return and line feed to the console.

*CRSEND (Send characters until carriage return) [0071H]
The contents of H,L contain the address of the characters to be sent to the console sequentially until a carriage return character is found in the string.

*BSEND (Send characters until B=0) [0074]
Register B contains the number of characters to be send to the console, H,L contains the beginning address of the string.

*CSEND (send character in C reg. B times) [0077H]
Register B contains the number of times the character in register C will be sent to the console.

*ASCGET (Input characters until B=0) [007AH]
Input characters from the console until content of B register is equal to zero. Register H,L contain address of where string will be stored. The CARRY is set if routine terminated by carriage return before B=0.

*GETBYT (Get two characters and form byte) [007DH]
First character from input stream becomes lower hex digit, second character becomes high hex digit of the byte returned in A.

DOWNLOAD COMMANDS

*Read hex paper tape format. R (cr)
Accepts information in the hex paper tape format. It starts out by looking for the colon leader, then it picks out the record length and the address of the record. After this parameters have been acquired incoming ASCII characters are stripped down to 4 bit HEX digit format. It takes two incoming ASCII characters to form a byte which is stored in memory. The command terminates when a record length of 00 is detected. The format it expects follows.
[:10203000CD40203E25....23]
This record is 16 bytes long and it is to be placed starting at address 2030. The first byte of the record is CD, the checkum is 23.

*Write hex paper tape format. W XXXX,YYYY,ZZ (cr)
Writes 16 byte records in the hex paper tape format.
Records start at address XXXX and continue until all data has been sent up to address YYYY.
The parameter ZZ (if ZZ=DD then delay is on) determines if a delay of 12 seconds should occur before the W command starts sending out formatted data. At the end of each line there is a .5 second pause to allow for software delays.

*** NOTE *** BY USING THIS COMMAND A USER CAN SAVE A MACHINE LANGUAGE PROGRAM ON A COMPUTER THAT HAS A STORAGE FACILITY.

*Hex sequential load. H XXXX,YYYY (cr)
This command will consider any incoming characters as 8 bit quantities that are to be stored in successive memory locations starting at address XXXX until the command terminates when we reach the address YYYY.

*ASCII sequential load. C XXXX,YYYY (cr)
ASCII load expects its input to be ASCII encoded hex digits which are converted to bytes and stored in memory locations starting at XXXX until YYYY. This command accepts groups of two ASCII characters only if they are preceeded by a space.

[This command is used when we want to transfer hex data from any computer system that has a display memory command) to the MCG-85].

E. CODE CHECK OR CONVERT.

***VALDIG** (Check for valid hex digit) [008FH]
Checks if register A contains a valid hex digit. CARRY is set if A contains valid hex digit, carry is cleared otherwise.

***PRVAL** (Converts HEX to ASCII) [0092H]
Converts the hexadecimal digit in register C to an ASCII character in register C.

***CNVNIBBLE** (Converts ASCII to HEX digit) [0095H]
Converts the ASCII character in A to a HEX digit in A.

***CNVBYTE** (Convert ASCII to two HEX digits) [0098H]
Assumes the contents of B,A registers contain two ASCII characters. From these characters it forms two HEX digits as a byte of data returned in the A register.

D. TEST + COMPARE ROUTINES.

***HILO** (Compare 16 bit integers) [0080H]
Compares the 16 bit integers in HL and DE. The CARRY flag is set if the contents of HL are larger than than or equal to the contents of DE; the CARRY is cleared if the contents of HL are less than the contents of DE.

***STRING** (String compare) [0083H]
Register pair DE contain the address of STRING1; HL contain the address of STRING2. Register B contains the length of the string. The CARRY is set if the strings are equal, otherwise the CARRY flag is cleared.

***STRUE** (Set carry and return) [0086H]
Strue is JUMPED to by routines wishing to indicate success. Strue sets the CARRY flag to indicate success and then returns to the calling program.

***SFALSE** (Clear carry and return) [0089H]
Sfalse is JUMPED to by routines wishing to indicate failure. Sfalse clears the CARRY flag to indicate failure and then returns to the calling program.

***BCDADD** (Binary coded decimal addition) [008CH]
Register pair HL contains the address of the FIRST number, while register pair DE contains address of SECOND number. Register B contains 1/2 the number of BCD digits to be added. The sum replaces the FIRST number.

***DELAY** (Delay proportional to content of DE) [009BH]
The contents of register pair DE are decremented to produce a delay of a few microseconds to approximately 1/2 second.

F. RESTART INSTRUCTIONS

RST 1, RST 2, RST 3, RST 4, RST 6, RST 7 all cause a "warm start". Upon executing any of the previous restart instructions control is transferred to location 0008H. The contents of all registers are saved and then displayed on the screen, control is then transferred to the command recognizing routine.

RST 5 will sound the bell at the console and then transfer control to the command recognizing routine. (RST 5 will not cause the contents of the registers to be saved.)

The following is a summary of the instruction set

8080/85 CPU INSTRUCTIONS IN OPERATION CODE SEQUENCE

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	28	DCX H	56	MOV D,M	81	ADD C	AC	XRA H	D7	RST 2
01	LXI B,D16	2C	INR L	57	MOV D,A	82	ADD D	AD	XRA L	D8	RC
02	STAX B	2D	DCR L	58	MOV E,B	83	ADD E	AE	XRA M	D9	–
03	INX B	2E	MVI L,D8	59	MOV E,C	84	ADD H	AF	XRA A	DA	JC Adr
04	INR B	2F	CMA	5A	MOV E,D	85	ADD L	B0	ORA B	DB	IN D8
05	DCR B	30	SIM	5B	MOV E,E	86	ADD M	B1	ORA C	DC	CC Adr
06	MVI B,D8	31	LXI SPD16	5C	MOV E,H	87	ADD A	B2	ORA D	DD	–
07	RLC	32	STA Adr	5D	MOV E,L	88	ADC B	B3	ORA E	DE	SBI D8
08	–	33	INX SP	5E	MOV E,M	89	ADC C	B4	ORA H	DF	RST 3
09	DAD B	34	INR M	5F	MOV E,A	8A	ADC D	B5	ORA L	E0	RPO
0A	LDAX B	35	DCR M	60	MOV H,B	8B	ADC E	B6	ORA M	E1	POP H
0B	DCX B	36	MVI M,D8	61	MOV H,C	8C	ADC H	B7	ORA A	E2	JPO Adr
0C	INR C	37	STC	62	MOV H,D	8D	ADC L	B8	CMP B	E3	XTHL
0D	DCR C	38	–	63	MOV H,E	8E	ADC M	B9	CMP C	E4	CPO Adr
0E	MVI C,D8	39	DAD SP	64	MOV H,H	8F	ADC A	BA	CMP D	E5	PUSH H
0F	RRC	3A	LDA Adr	65	MOV H,L	90	SUB B	BB	CMP E	E6	ANI D8
10	–	3B	DCX SP	66	MOV H,M	91	SUB C	BC	CMP H	E7	RST 4
11	LXI D,D16	3C	INR A	67	MOV H,A	92	SUB D	BD	CMP L	E8	RPE
12	STAX D	3D	DCR A	68	MOV L,B	93	SUB E	BE	CMP M	E9	PCHL
13	INX D	3E	MVI A,D8	69	MOV L,C	94	SUB H	BF	CMP A	EA	JPE Adr
14	INR D	3F	CMC	6A	MOV L,D	95	SUB L	C0	RNZ	EB	XCHG
15	DCR D	40	MOV B,B	6B	MOV L,E	96	SUB M	C1	POP B	EC	CPE Adr
16	MVI D,D8	41	MOV B,C	6C	MOV L,H	97	SUB A	C2	JNZ Adr	ED	–
17	RAL	42	MOV B,D	6D	MOV L,L	98	SBB B	C3	JMP Adr	EE	XRI D8
18	–	43	MOV B,E	6E	MOV L,M	99	SBB C	C4	CNZ Adr	EF	RST 5
19	DAD D	44	MOV B,H	6F	MOV L,A	9A	SBB D	C5	PUSH B	F0	RP
1A	LDAX D	45	MOV B,L	70	MOV M,B	9B	SBB E	C6	ADI D8	F1	POP PSW
1B	DCX D	46	MOV B,M	71	MOV M,C	9C	SBB H	C7	RST 0	F2	JP Adr
1C	INR E	47	MOV B,A	72	MOV M,D	9D	SBB L	C8	RZ	F3	DI
1D	DRC E	48	MOV C,B	73	MOV M,E	9E	SBB M	C9	RET Adr	F4	CP Adr
1E	MVI E,D8	49	MOV C,C	74	MOV M,H	9F	SBB A	CA	JZ Adr	F5	PUSH PSW
1F	RAR	4A	MOV C,D	75	MOV M,L	A0	ANA B	CB	–	F6	ORI D8
20	RIM	4B	MOV C,E	76	HLT	A1	ANA C	CC	CZ Adr	F7	RST 6
21	LXI H,D16	4C	MOV C,H	77	MOV M,A	A2	ANA D	CD	CALL Adr	F8	RM
22	SHLD Adr	4D	MOV C,L	78	MOV A,B	A3	ANA E	CE	ACI D8	F9	SPHL
23	INX H	4E	MOV C,M	79	MOV A,C	A4	ANA H	CF	RST 1	FA	JM Adr
24	INR H	4F	MOV C,A	7A	MOV A,D	A5	ANA L	D0	RNC	FB	EI
25	DCR H	50	MOV D,B	7B	MOV A,E	A6	ANA M	D1	POP D	FC	CM Adr
26	MVI H,D8	51	MOV D,C	7C	MOV A,H	A7	ANA A	D2	JNC Adr	FD	–
27	DAA	52	MOV D,D	7D	MOV A,L	A8	XRA B	D3	OUT D8	FE	CPI D8
28	–	53	MOV D,E	7E	MOV A,M	A9	XRA C	D4	CNC Adr	FF	RST 7
29	DAD H	54	MOV D,H	7F	MOV A,A	AA	XRA D	D5	PUSH D		
2A	LHLD Adr	55	MOV D,L	80	ADD B	AB	XRA E	D6	SUI D8		

D8 = constant, or logical/arithmetic expression that evaluates to an 8 bit data quantity.

D16 = constant, or logical/arithmetic expression that evaluates to a 16 bit data quantity

Adr = 16 bit address

ASCII CODES

The 8080 and 8085 use the seven-bit ASCII code, with the high-order eighth bit (parity bit) always reset.

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
NUL	00
SOH	01
STX	02
ETX	03
EOT	04
ENQ	05
ACK	06
BEL	07
BS	08
HT	09
LF	0A
VT	0B
FF	0C
CR	0D
SO	0E
SI	0F
DLE	10
DC1 (X-ON)	11
DC2 (TAPE)	12
DC3 (X-OFF)	13
DC4 (TAPE)	14
NAK	15
SYN	16
ETB	17
CAN	18
EM	19
SUB	1A
ESC	1B
FS	1C
GS	1D
RS	1E
US	1F
SP	20
!	21
"	22
#	23
$	24
%	25
&	26
'	27
(	28
)	29
*	2A

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
+	2B
,	2C
-	2D
.	2E
/	2F
0	30
1	31
2	32
3	33
4	34
5	35
6	36
7	37
8	38
9	39
:	3A
;	3B
<	3C
=	3D
>	3E
?	3F
@	40
A	41
B	42
C	43
D	44
E	45
F	46
G	47
H	48
I	49
J	4A
K	4B
L	4C
M	4D
N	4E
O	4F
P	50
Q	51
R	52
S	53
T	54
U	55

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
V	56
W	57
X	58
Y	59
Z	5A
[	5B
\	5C
]	5D
^ (↑)	5E
_ (←)	5F
`	60
a	61
b	62
c	63
d	64
e	65
f	66
g	67
h	68
i	69
j	6A
k	6B
l	6C
m	6D
n	6E
o	6F
p	70
q	71
r	72
s	73
t	74
u	75
v	76
w	77
x	78
y	79
z	7A
{	7B
|	7C
} (ALT MODE)	7D
~	7E
DEL (RUB OUT)	7F

APPENDIX E.

PROGRAM LISTINGS

```
                ;*****************************************************
                ;*****************************************************
                ;*                                                   *
                ;*   L      CCCCC  DDDDD      AAA   SSSSS  MMMMM     *
                ;*   L      CC     DD  D     A   A  SS     M M M     *
                ;*   L      CC     DD  D     AAAAA   SS    M M M     *
                ;*   L      CC     DD D      A   A     SS  M   M     *
                ;*   LLLLL  CCCCC  DDD   O   A   A  SSSSS  M   M     *
                ;*                                                   *
                ;*****************************************************
                ;*****************************************************
                ;REVISION HISTORY:              VERSION: 1.2
                ;
                ;DATE:          CHANGES:
                ;23 APR 84      WRITTEN BY LEE MARZKE
                ;13 MAY 84      FIXED MANY BUGS IN PUT,PUTX, AND DROTR
                ;16 MAY 84      ADDED GETBAUD CALL, AUTOBAR
                ;*****************************************************
                ;  THIS IS A GRAPHICS AND TEXT DISPLAY PROGRAM FOR THE
                ;SHARPE LM-24003G LCD DISPLAY.  IT IS WRITTEN TO RUN
                ;ON THE MCG-8085 COMPUTER HAVING A LHM-24000 DISPLAY
                ;CONTROLLER BOARD.
                ;  TEXT STRINGS AND GRAPHIC SYMBOLS ARE DEFINED AT
                ;INITILIZATION. THE TEXT STRINGS MAY THEN BE CHANGED
                ;AND BOTH STRINGS AND SYMBOLS MAY BE MOVED TO ANY
                ;COORDINATE ON THE SCREEN.
                ;
                ;*****************************************************
                ;               EQUATES
000D =          CR      EQU     0DH     ;CARRIAGE RETURN
000A =          LF      EQU     0AH     ;LINE FEED
4000 =          SSAD    EQU     4000H   ;START OF DISPLAY RAM
003F =          ASCMSK  EQU     03FH    ;ASCII MASK
0080 =          SYMMSK  EQU     080H    ;SYMBOL MASK
3000 =          DEMO    EQU     03000H  ;DEMO, FUNCTION EPROM
3050 =          FUNCT   EQU     03050H  ;ENTRY FOR FUNCTION COMMAND
                ;
                ;*****************************************************
                ;
                ; MONITOR CALLS
                ;
                ;*****************************************************
                ;
0000 =          MRES    EQU     0000H   ;MONITOR COLD START
0008 =          MCO     EQU     0008H   ;MONITER WARM START (MUST BE
                                        ; CALLED)
0065 =          MCI     EQU     0065H   ;MONITOR INPUT ROUTINE
                                        ; RETURNS ASCII CHAR. IN A.
0068 =          MCO     EQU     0068H   ;MONITOR OUTPUT ROUTINE
                                        ; SENDS ASCII CHAR IN C.
006E =          MCRLF   EQU     006EH   ;SENDS CR,LF TO CONSOLE
0080 =          HILO    EQU     0080H   ;16 BIT COMPARE OF DE,HL
```



```
                                        ; CARRY SET IF HL>=DE
0524 =          CONV    EQU     0524H   ;CONVERTS TWO ASCII CHAR. IN
                                        ; A,B TO HEX DIGIT IN A
0466 =          VALDL   EQU     0466H   ;RET WITH CARRY SET IF CHAR.
                                        ; IN C. IS A VALID DELIMITER
009B =          DELAY   EQU     009BH   ;DELAY PROPIONAL TO D,E
0479 =          GETBAUD EQU     0479H   ;CALCULATE BAUD FROM SPACE CHR
                ;
0800            ORG     0800H
                ;
0800 AA                 DB      0AAH    ;AUTOSTART BYTE
                ;
                ;COLD START HERE
0801 AF         INIT:   XRA     A       ;A=0
0802 321611             STA     EFLAG   ;ECHO ON
0805 320211             STA     FUNBYT  ;FUNCTION OFF
0808 3E07               MVI     A,07H
080A 321711             STA     CSPACE  ;CHARACTER SPACING
080D CD1308             CALL    INSTAB  ;INIT SYM. TABLE
0810 C32D08             JMP     GO
                ;
0813 AF         INSTAB: XRA     A       ;INIT SYMBOL TABLE
0814 321111             STA     NUMSYM
0817 211211             LXI     H,NEWSYM
081A 111811             LXI     D,SYMTAB
081D 73                 MOV     M,E
081E 23                 INX     H
081F 72                 MOV     M,D
0820 211411             LXI     H,NEWDAT
0823 110013             LXI     D,DATA
0826 73                 MOV     M,E
0827 23                 INX     H
0828 72                 MOV     M,D
0829 CDE508             CALL    CLR
082C C9                 RET
                ;
                ;
                ; WARM START ENTRY HERE
                ;
082D 31A811     GO:     LXI     SP,STACK;INIT STACK
0830 AF                 XRA     A       ;A=0
0831 320011             STA     BUFTOP
0834 320111             STA     BUFBOT
0837 CD170D             CALL    CRLF
083A 3A0030             LDA     DEMO    ;SEE IF DEMO ROM PRESENT
083D FEBB               CPI     0BBH    ;DEMO PRESENT BYTE
083F CA0130             JZ      DEMO+1  ;YES, JUMP TO DEMO ROM
0842 CD7904             CALL    GETBAUD
0845 C35308             JMP     GETCMD  ;NO
                ;
                ;*****************************************************
                ;
                ; JUMP TABLE
                ;
                ;*****************************************************
```

```
                ;
0850            ORG     0850H
0850 C3AF0C     WRBUF:  JMP     WRBUFR
0853 C35608     GETCMD: JMP     GETCM
                ;
                ;*****************************************************
                ;
                ; COMMAND RECOGNIZER ROUTINE
                ;
                ;*****************************************************
                ;
                ;
                ; FUNCTION: GET COMMAND
                ; INPUTS: NONE
                ; OUTPUTS: NONE
                ; CALLS: RDBUF, ECHO, ERROR
                ; DESTROYS: A,B,C,H,L,F
                ; DESCRIPTION: GETCMD RECEIVES AN INPUT CHARACTER FROM
                ;              THE INPUT BUFFER AND ATTEMPTS TO LOCATE
                ;              IT IN THE COMMAND CHARACTER TABLE.
                ;              IF SUCCESSFUL THE ROUTINE CORRESPONDING
                ;              TO THIS CHARACTER IS SELECTED FROM A
                ;              TABLE OF COMMAND ROUTINE ADDRESSES, AND
                ;              CONTROL IS TRANSFERRED TO THIS ROUTINE.
                ;
                ;
0856 31A811     GETCM:  LXI     SP,STACK;LCD STACK POINTER SET
0859 0E23               MVI     C,'#'
085B CD030D             CALL    ECHO    ;PROMPT CHARACTER
085E CD7D0C     GTC03:  CALL    RDBUF
0861 010E00             LXI     B,NCMDS ;NUMBER OF COMMANDS
0864 21380D             LXI     H,CTAB  ;COMMAND TABLE
0867 BE         GTC05:  CMP     M       ;COMPARE TABLE ENTRY AND CHAR.
0868 CA7308             JZ      GTC10   ;COMD. RECOGNIZED
086B 23                 INX     H       ;ELSE, INC TABLE POINTER
086C 0D                 DCR     C       ;DEC LOOP COUNT
086D C26708             JNZ     GTC05   ;BRANCH IF NOT AT TABLE END
0870 C37D08             JMP     ERROR
0873 21460D     GTC10:  LXI     H,CADR  ;ADR OF TABLE OF COM. ROUTINE
0876 09                 DAD     B       ; ADDRESSES
0877 09                 DAD     B       ;ADD REMAINDER OF LOOP COUNT
0878 7E                 MOV     A,M     ; TWICE BECAUSE ENTRY 2 WORDS
0879 23                 INX     H
087A 66                 MOV     H,M
087B 6F                 MOV     L,A
087C E9                 PCHL
                ;
                ;
087D 0E2A       ERROR:  MVI     C,'*'
087F CD030D             CALL    ECHO
0882 CD170D             CALL    CRLF
0885 C35308             JMP     GETCMD
                ;
                ;
                ;*****************************************************
```



```
                ;
                ; COMMAND IMPLEMENTING ROUTINES
                ;
                ;*****************************************************
                ;
                ; THE FOLLOWING SHORTHAND IS USED BELOW:
                ; ## — SYMBOL IDENT (TWO HEX DIGITS)
                ; XX — X POSITION      "    "
                ; YY — Y POSITION      "    "
                ; HH — HEIGHT IN PIXELS
                ; WW — WIDTH IN WORDS (UNITS OF 8 PIXELS)
                ; BBB.. — HEX DIGITS (IN GROUPS OF TWO) REPRESENTING
                ;          THE SYMBOL TO BE STORED.
                ; STRING — ASCII STRING IN THE RANGE OF 20H TO 60H
                ; <CR>  — CARRIAGE RETURN
                ;
                ;NOTE: THE UPPER TWO BITS OF ## ARE MASKED AND SET TO:
                ;       00 - FOR ASCII STRINGS
                ;       01 - FOR SYMBOLS
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: A##/STRING <CR>  (ENTERS OR CHANGES STRING)
                ;         A##,XX,YY <CR>    (CHANGES X,Y POSITION)
                ;
0888 CDD00C     ACMD:   CALL    GETBYT
088B E63F               ANI     ASCMSK
088D F5                 PUSH    PSW     ;SAVE SYM #
088E CDDF0C             CALL    GETCHR
0891 FE2F               CPI     '/'     ;CHANGE STRING VALUE?
0893 CABF08             JZ      AC10    ;YES
0896 4F                 MOV     C,A
0897 CD6604             CALL    VALDL   ;NO, MOVE STRING LOCATION
089A D27D08             JNC     ERROR
089D F1                 POP     PSW     ;SYM #
089E CD610B             CALL    SYMADR  ;GET STORAGE ADR
08A1 D27D08             JNC     ERROR   ;STRING NOT FOUND
08A4 E5                 PUSH    H       ;SAVE IT
08A5 CD580C             CALL    CLRSTR  ;ERASE OLD STRING
08A8 E1                 POP     H
08A9 CDD00C             CALL    GETBYT  ;X POS
08AC 77                 MOV     M,A
08AD 23                 INX     H
08AE CDF30C             CALL    GETDL
08B1 CDD00C             CALL    GETBYT  ;Y POS
08B4 77                 MOV     M,A
08B5 2B                 DCX     H
08B6 CDE50C             CALL    GETCR
08B9 CD030C             CALL    PUTSTR
08BC C35308             JMP     GETCMD
08BF F1         AC10:   POP     PSW     ;GET SYM #
08C0 F5                 PUSH    PSW     ;SAVE AGAIN
08C1 CD610B             CALL    SYMADR
08C4 D2D508             JNC     AC15    ;SYMBOL FOUND?
08C7 E5                 PUSH    H       ;YES, SAVE ADR
08C8 23                 INX     H
```

```
08C9 23             INX     H
08CA CD220C         CALL    GETSTR
08CD E1             POP     H       ;SYM ADR
08CE F1             POP     PSW     ;SYM #
08CF CD030C         CALL    PUTSTR
08D2 C35308         JMP     GETCMD
08D5 F1       AC15: POP     PSW     ;NO
08D6 CDF40B         CALL    ADDSTR
08D9 C35308         JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: C <CR>        CLEAR SCREEN
08DC CDE50C   CCMD: CALL    GETCR
08DF CDE508         CALL    CLR
08E2 C35308         JMP     GETCMD
                ;
08E5 0E00     CLR:  MVI     C,00H
08E7 210040         LXI     H,04000H
08EA 110050         LXI     D,05000H
08ED 71             MOV     M,C
08EE 23       CL5:  INX     H
08EF 71             MOV     M,C
08F0 CD8000         CALL    HILO
08F3 D2EE08         JNC     CL5
08F6 C9             RET
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: D##,HHWWBBBBBBBB...  <CR>
                ;
08F7 CDD00C   DCMD: CALL    GETBYT  ;SYMBOL IDENT
08FA E63F           ANI     ASCMSK
08FC F680           ORI     SYMMSK
08FE F5             PUSH    PSW     ;SAVE IDENT
08FF CD610B         CALL    SYMADR  ;LOOK UP ADDRESS
0902 DA7D08         JC      ERROR   ;ERROR IF ALREADY DEFINED
0905 CDF30C         CALL    GETDL   ;GET COMMA
0908 F1             POP     PSW     ;GET IDENT
0909 CD850B         CALL    GETSYM  ;GET SYMBOL FROM INPUT BUFFER
090C CDE50C         CALL    GETCR
090F C35308         JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: E <CR>        ECHO ON
0912 AF       ECMD: XRA     A       ;ZERO A
0913 321611         STA     EFLAG   ;TURN ON ECHO
0916 CDE50C         CALL    GETCR
0919 C35308         JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: F## <CR>
091C CDD00C   FCMD: CALL    GETBYT
091F F5             PUSH    PSW
0920 CDE50C         CALL    GETCR
0923 F1             POP     PSW
0924 CD5030         CALL    FUNCT
```



```
0927 C35308         JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: I <CR>        ;INIT SYMBOL TABLE
092A CDE50C   ICMD: CALL    GETCR
092D CD1308         CALL    INSTAB
0930 C35308         JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: K <CR>        KILL ECHO
0933 3EFF     KCMD: MVI     A,0FFH
0935 321611         STA     EFLAG   ;TURN OFF ECHO
0938 CDE50C         CALL    GETCR
093B C35308         JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: LXXYY,XXYY,XXYY,... <CR>
                ;
093E CDD00C   LCMD: CALL    GETBYT  ;GET X CORD.
0941 57             MOV     D,A
0942 CDD00C         CALL    GETBYT  ;GET Y CORD.
0945 4F             MOV     C,A
0946 42             MOV     B,D
0947 3EFF           MVI     A,0FFH  ;SET POINT
0949 CDF009         CALL    POINT
094C CDDF0C         CALL    GETCHR
094F FE0D           CPI     CR
0951 CA5E09         JZ      LMD05   ;END?
0954 4F             MOV     C,A     ;NO
0955 CD6604         CALL    VALDL   ;SEE IF COMMA
0958 D27D08         JNC     ERROR   ;NOT COMMA
095B C33E09         JMP     LCMD    ;GET NEXT POINT
095E CD170D   LMD05: CALL   CRLF
0961 C35308         JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: N <CR>        NARROW CHARACTER SPACING
0964 3E06     NCMD: MVI     A,06H   ;CHARACTER SPACING = 6 PIXELS
0966 321711         STA     CSPACE
0969 CDE50C         CALL    GETCR
096C C35308         JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; PUT SYMBOL AT X,Y COORDINATES
                ; FORMAT: P##,XX,YY <CR>
096F CDB609   PCMD: CALL    GXYHL   ;GET COORDINATES, HEIGHT,WIDTH
0972 CD060A         CALL    PUT
0975 C35308         JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: W <CR>        WIDE CHARACTER SPACING
0978 3E07     WCMD: MVI     A,07H   ;CHARACTER SPACING = 7 PIXELS
097A 321711         STA     CSPACE
097D CDE50C         CALL    GETCR
0980 C35308         JMP     GETCMD
```

```
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; XOR SYMBOL WITH SCREEN AT X,Y COORDINATES
                ; FORMAT: X##,XX,YY <CR>
                ;         X##/<CR> - USES OLD COORDS
0983 CDB609     XCMD:   CALL    GXYHL
0986 CD380A             CALL    PUTX
0989 C35308             JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; MOVE SYMBOL FROM OLD POSITION TO X,Y COORDINATES
                ; FORMAT: M##,XX,YY <CR>
098C CDD00C     MCMD:   CALL    GETBYT  ;GET ADDRESS
098F E63F               ANI     ASCMSK
0991 F680               ORI     SYMMSK
0993 CD610B             CALL    SYMADR
0996 D27D08             JNC     ERROR
0999 E5                 PUSH    H       ;SAVE ADDRESS
099A 46                 MOV     B,M     ;OLD X POS
099B 23                 INX     H
099C 4E                 MOV     C,M     ;OLD Y POS
099D 23                 INX     H
099E 56                 MOV     D,M     ;GET HEIGHT
099F 23                 INX     H
09A0 5E                 MOV     E,M     ;GET WIDTH
09A1 23                 INX     H
09A2 EB                 XCHG
09A3 CD380A             CALL    PUTX    ;ERASE OLD
09A6 E1                 POP     H       ;GET ADRESS
09A7 CDC309             CALL    XYHL    ;XOR NEW
09AA CD380A             CALL    PUTX
09AD C35308             JMP     GETCMD
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
                ; FORMAT: Z <CR>        JUMP TO MONITOR
09B0 CDE50C     ZCMD:   CALL    GETCR
09B3 CD0800             CALL    MGO
                ;
                ;*****************************************************
                ; INPUTS: INPUT BUFFER [##,XX,YY]
                ; OUTPUTS: H = HEIGHT, L = WIDTH
                ;          D,E = SYM ADR + 4
                ;          B = X POS, C = Y POS
                ; DESCRIPTION:
                ; GET X,Y POSITION AND STORE WITH SYMBOL
                ; RETRIEVE HIEGHT, WIDTH FROM SYMBOL
                ;
09B6 CDD00C     GXYHL:  CALL    GETBYT  ;SYM IDENT TO A
09B9 E63F               ANI     ASCMSK
09BB F680               ORI     SYMMSK
09BD CD610B             CALL    SYMADR  ;LOOKUP STORAGE ADR.
09C0 D27D08             JNC     ERROR
09C3 CDDF0C     XYHL:   CALL    GETCHR  ;CHECK FOR DELIMITER
09C6 FE2F               CPI     '/'
09C8 CAE909             JZ      XY      ;IF /, USE OLD X.Y CORDS
```



```
09CB CDFB0C             CALL    GDL
09CE CDD00C             CALL    GETBYT  ;GET X POS.
09D1 77                 MOV     M,A     ;STORE IN SYMBOL
09D2 23                 INX     H
09D3 57                 MOV     D,A     ;SAVE X
09D4 CDF30C             CALL    GETDL   ;GET DL
09D7 CDD00C             CALL    GETBYT  ;GET Y POS.
09DA 77                 MOV     M,A     ;STORE IT
09DB 23                 INX     H
09DC 4F                 MOV     C,A     ;Y POS
09DD 42                 MOV     B,D     ;X POS
09DE C5         XYH:    PUSH    B
09DF 56                 MOV     D,M     ;GET HEIGHT
09E0 23                 INX     H
09E1 5E                 MOV     E,M     ;GET WIDTH
09E2 23                 INX     H       ;SYM ADR + 4
09E3 CDE50C             CALL    GETCR   ;CHECK FOR CR
09E6 C1                 POP     B       ;X,Y POS
09E7 EB                 XCHG
09E8 C9                 RET
09E9 46         XY:     MOV     B,M     ;X POS
09EA 23                 INX     H
09EB 4E                 MOV     C,M     ;Y POS
09EC 23                 INX     H
09ED C3DE09             JMP     XYH
                ;
                ;*****************************************************
                ;
                ; MOVE GRAPHICS ROUTINES
                ;
                ;*****************************************************
                ;
                ;FUNCTION: POINT
                ;INPUTS:  A - 0,RESET POINT; ELSE SET POINT
                ;         B - X COORDINATE
                ;         C - Y COORDINATE
                ;DESCRIPTION: SETS OR CLEARS SPECIFIED POINT ON SCREEN
                ;
09F0 F5         POINT:  PUSH    PSW
09F1 CDAF0A             CALL    SCADR   ;GET ADDRESS,B=SBITS
09F4 3E80               MVI     A,080H  ;MASK
09F6 CD960A             CALL    ROTR    ;ROTATE B TIMES
09F9 47                 MOV     B,A     ;SAVE MASK
09FA 2F                 CMA
09FB A6                 ANA     M       ;MASK OUT POINT
09FC 4F                 MOV     C,A     ;SAVE
09FD F1                 POP     PSW
09FE B7                 ORA     A       ;SET FLAGS
09FF 79                 MOV     A,C     ;DOESN'T CHANGE FLAGS
0A00 CA040A             JZ      PNT05   ;LEAVE POINT RESET
0A03 B0                 ORA     B       ;SET POINT
0A04 77         PNT05:  MOV     M,A     ;WRITE TO SCREEN MEM
0A05 C9                 RET
                ;
                ;*****************************************************
```

INTENTIONALLY LEFT BLANK

```
                ;
                ; FUNCTION: PUT (PUT SYMBOL AT X,Y LOCATION)
                ; INPUTS:  B - X POS
                ;          C - Y POS
                ;        D,E - SYM. ADDRESS + 4
                ;          H - HEIGHT
                ;          L - WIDTH
                ; OUTPUTS: D,E - LAST SYM. ADDRESS
                ; CALLS: DROTR, GETMSK, INCSAD, SCADR
                ;
                ;
0A06 220E11     PUT:    SHLD    SYMXCT
0A09 220C11             SHLD    SYMWD
0A0C D5                 PUSH    D       ;SAVE SYMBOL ADDRESS
0A0D CDAF0A             CALL    SCADR   ;GET SCREEN ADDRESS
0A10 CD9C0A             CALL    GETMSK  ;SET MASK CORRECTLY
0A13 D1                 POP     D       ;SYM. ADDRESS
0A14 4E         PUT01:  MOV     C,M     ;SAVE 1ST WORD FOR CARRY IN
0A15 EB         PUT02:  XCHG
0A16 7E                 MOV     A,M     ;GET SYMBOL WORD
0A17 CD710A             CALL    DROTR   ;SHIFT IT RIGHT
0A1A EB                 XCHG
0A1B 77                 MOV     M,A     ;WRITE TO SCREEN
0A1C 13                 INX     D       ;NEXT SYM. ADDRESS
0A1D CD020B             CALL    INCSAD  ;NEXT SCR. ADDRESS
0A20 D2150A             JNC     PUT02   ;RIGHT EDGE OF SYM? NO
0A23 F5                 PUSH    PSW     ;YES, STORE A
0A24 E5                 PUSH    H       ;SAVE HL
0A25 2A0511             LHLD    SADRE   ;GET SAD RIGHT EDGE + 1
0A28 7E                 MOV     A,M     ;GET NEXT SCR. WORD
0A29 E5                 PUSH    H
0A2A 2A0A11             LHLD    RMASK   ;GET ROTATE MASK
0A2D A5                 ANA     L       ;MASK OUT NO BITS
0A2E B1                 ORA     C       ;OR CARRY OUT BITS
0A2F E1                 POP     H
0A30 77                 MOV     M,A     ;PUT BACK ON SCREEN
0A31 E1                 POP     H       ;RESTORE LAST ADR. TO H,L
0A32 F1                 POP     PSW     ;RESTORE A (=0, DONE)
0A33 B7                 ORA     A       ;SET FLAGS
0A34 C2140A             JNZ     PUT01   ;DONE? NO
0A37 C9                 RET             ;YES
                ;
                ;
                ;*****************************************************
                ;
                ; FUNCTION: PUTX (XOR SYMBOL WITH SCREEN AT X.Y LOC)
                ; INPUTS: B  -  X POS
                ;         C  -  Y POS
                ;       D,E  -  SYM ADRESS + 4
                ;         H  -  HEIGHT
                ;         L  -  WIDTH
                ; OUTPUTS: D,E - LAST SYM ADDRESS
                ; CALLS:  DROTR, GETMSK, INCSAD, SCADR
                ;
0A38 220E11     PUTX:   SHLD    SYMXCT
```



```
0A3B 220C11             SHLD    SYMWD
0A3E D5                 PUSH    D       ;SAVE SYMBOL ADDRESS
0A3F CDAF0A             CALL    SCADR   ;GET SCREEN ADDRESS
0A42 CD9C0A             CALL    GETMSK  ;SET MASK CORRECTLY
0A45 D1                 POP     D       ;SYM. ADDRESS
0A46 0E00       PUTX01: MVI     C,00H   ;CARRY IN NOT NEEDED
0A48 EB         PUTX02: XCHG
0A49 7E                 MOV     A,M     ;GET SYMBOL WORD
0A4A CD710A             CALL    DROTR   ;SHIFT IT RIGHT
0A4D 47                 MOV     B,A
0A4E 3A0A11             LDA     RMASK   ;MASK UPPER N BITS
0A51 A0                 ANA     B
0A52 EB                 XCHG
0A53 AE                 XRA     M
0A54 77                 MOV     M,A     ;WRITE TO SCREEN
0A55 13                 INX     D       ;NEXT SYM. ADDRESS
0A56 CD020B             CALL    INCSAD  ;NEXT SCR. ADDRESS
0A59 D26B0A             JNC     PUTX03  ;RIGHT EDGE OF SYM? NO
0A5C F5                 PUSH    PSW     ;STORE DONE BYTE
0A5D E5                 PUSH    H       ;SAVE HL
0A5E 79                 MOV     A,C     ;GET CARRY OUT
0A5F 2A0511             LHLD    SADRE   ;YES
0A62 AE                 XRA     M
0A63 77                 MOV     M,A
0A64 E1                 POP     H
0A65 F1                 POP     PSW     ;RESTORE A (=0, DONE)
0A66 B7                 ORA     A       ;SET FLAGS
0A67 C2460A             JNZ     PUTX01  ;DONE? NO
0A6A C9                 RET             ;YES
0A6B 79         PUTX03: MOV     A,C     ;CARRY
0A6C AE                 XRA     M       ;XOR IN NEXT SCREEN LOCATION
0A6D 77                 MOV     M,A
0A6E C3460A             JMP     PUTX01
                ;
                ;
                ;*****************************************************
                ;
                ;
                ;       FUNCTION: DROTR (DOUBLE WIDE SHIFT RIGHT)
                ;       INPUTS:     A    - SHIFT REGISTER
                ;                   C    - OVERFLOW BITS IN
                ;                          (LEFT JUSTIFIED)
                ;                 SBITS  - # OF BITS TO SHIFT (N)
                ;                RMASK   - MASK (8-N OF THE LSB'S SET)
                ;                RMASK+1 - MASK (N OF THE MSB'S SET)
                ;       OUTPUTS:    A    - SHIFT REGESTER
                ;                   C    - OVERFLOW BITS OUT
                ;                          (LEFT JUSTIFIED)
                ;       CALLS: ROTR
                ;       DESTROYS: A,C,F
                ;       DESCRIPTION: THE BITS IN THE A. REG.  ARE
                ;                    SHIFTED RIGHT N TIMES, WITH
                ;                    INFLOW FROM REG. C AND OVERFLOW
                ;                    TO REG. C.    RMASK MUST BE SET
                ;                    (BY EXECUTING GETMSK:) PRIOR TO
```

```
                ;                       ENTRY.
                ;
0A71 F5         DROTR:  PUSH    PSW     ;SAVE A
0A72 3A0911             LDA     SBITS   ;CHECK #
0A75 47                 MOV     B,A
0A76 FE00               CPI     00H     ;=0?
0A78 C27E0A             JNZ     DROT05  ;NO
0A7B 4F                 MOV     C,A     ;CLEAR CARRY OUT
0A7C F1                 POP     PSW     ;YES, DONE
0A7D C9                 RET
0A7E F1         DROT05: POP     PSW
0A7F D5                 PUSH    D       ;SAVE DE
0A80 E5                 PUSH    H       ;SAVE HL
0A81 C5                 PUSH    B       ;SAVE CARRY IN
0A82 2A0A11             LHLD    RMASK   ;GET ROTATE MASK
0A85 CD960A             CALL    ROTR    ;SHIFT A REG.
0A88 47                 MOV     B,A     ;SAVE
0A89 A4                 ANA     H       ;GET OVERFLOW
0A8A 4F                 MOV     C,A     ;PUT IN C
0A8B 78                 MOV     A,B
0A8C A5                 ANA     L       ;MASK OVERFLOW
0A8D D1                 POP     D       ;CARRY IN TO E
0A8E 42                 MOV     B,D     ;RESTORE # IN TO B
0A8F 57                 MOV     D,A
0A90 7B                 MOV     A,E
0A91 A4                 ANA     H       ;MASK
0A92 B2                 ORA     D       ;OR WITH SHIFTED BITS
0A93 E1                 POP     H       ;RESTORE HL
0A94 D1                 POP     D       ;RESTORE DE
0A95 C9                 RET
                ;
                ; ROTATE A RIGHT B TIMES
                ;DESTROYS: A,B,F
                ;
0A96 05         ROTR:   DCR     B       ;DONE?
0A97 F8                 RM              ;YES
0A98 0F                 RRC             ;NO, ROTATE RIGHT
0A99 C3960A             JMP     ROTR
                ;
                ;*****************************************************
                ;
                ;DESTROYS: A,F
                ;
0A9C C5         GETMSK: PUSH    B       ;SAVE B
0A9D E5                 PUSH    H       ;SAVE H
0A9E 212F0D             LXI     H,MSKTAB
0AA1 48                 MOV     C,B
0AA2 0600               MVI     B,00H   ;GET N'TH ENTRY
0AA4 09                 DAD     B       ;IN TABLE
0AA5 66                 MOV     H,M
0AA6 7C                 MOV     A,H
0AA7 2F                 CMA             ;COMPLEMENT IT
0AA8 6F                 MOV     L,A
0AA9 220A11             SHLD    RMASK
```



```
0AAC E1                 POP     H       ;RESTORE H
0AAD C1                 POP     B       ;RESTORE B
0AAE C9                 RET
                ;
                ;*****************************************************
                ;
                ;SCREEN ADDRESSING ROUTINES
                ;
                ;*****************************************************
                ;
                ;
                ; FUNCTION: SCADR
                ; INPUTS:    B - X POSITION
                ;            C - Y POSITION
                ;         SSAD - START OF DISPLAY RAM
                ; OUTPUTS: H,L - ADDRESS OF SCREEN LOCATION = SADSV
                ;            B   - BITS TO SHIFT RIGHT = SBITS
                ; CALLS: FIXADD
                ; DESTROYS: A,B,C,D,E,F,H,L
                ; DESCRIPTION: FROM THE X AND Y COORDINATES THIS
                ;              ROUTINE CALCULATES THE CORRESPONDING
                ;              ADDRESS OF DISPLAY RAM, AND THE BIT
                ;              OFFSET.
                ;              SSAD = 30*Y + INT( X/8 )
                ;              SBITS = 8 * FRC( X/8 )
0AAF 210040     SCADR:  LXI     H,SSAD
0AB2 78                 MOV     A,B     ;X LOCATION
0AB3 CDDA0A             CALL    DIV8
0AB6 F5                 PUSH    PSW     ;SAVE X/8
0AB7 AF                 XRA     A       ;ZERO
0AB8 321011             STA     SCRBYT  ;INIT TO 0
0ABB 57                 MOV     D,A
0ABC 19                 DAD     D       ;ADD TO START SCREEN ADD.
0ABD 220711             SHLD    SADSV   ;SAVE IT
0AC0 79                 MOV     A,C     ;Y LOCATION
0AC1 E67F               ANI     07FH    ;MASK
0AC3 CDE50A             CALL    MULT30
0AC6 EB                 XCHG
0AC7 2A0711             LHLD    SADSV   ;GET SCR ADDRESS
0ACA 19                 DAD     D       ;ADD 30 Y
0ACB CD430B             CALL    FIXADR  ;CORRECT FOR GAP IN DISP RAM
0ACE 220711             SHLD    SADSV   ;FINAL SCR ADDRESS
0AD1 F1                 POP     PSW     ;GET X/8
0AD2 07                 RLC
0AD3 07                 RLC
0AD4 07                 RLC             ;NO OF SHIFTS
0AD5 320911             STA     SBITS
0AD8 47                 MOV     B,A
0AD9 C9                 RET
                ;
                ;*****************************************************
                ;
                ; FUNCTION: DIV8
```

```
                ; INPUTS:     A - 8 BIT INTEGER
                ; OUTPUTS:    A - FRACTION PART
                ;             D - INTEGER PART
                ; CALLS: NOTHING
                ; DESTROYS: A,D,E,F
                ; DESCRIPTION:  THE 8 BIT NUMBER IN A. IS DIVIDED BY
                ;               8 WITH THE INTEGER PART AND FRACTION
                ;               PART STORED IN D. AND A. RESPECTIVELY.
                ;
0ADA 0F         DIV8:   RRC
0ADB 0F                 RRC
0ADC 0F                 RRC             ;DIVIDE BY 8
0ADD 57                 MOV     D,A     ;SAVE
0ADE E61F               ANI     01FH    ;INT PART
0AE0 5F                 MOV     E,A     ;PUT IN E
0AE1 7A                 MOV     A,D     ;GET ROT BITS
0AE2 E6E0               ANI     0E0H    ;FRC PART
0AE4 C9                 RET
                ;
                ;*****************************************************
                ; FUNCTION: MULTPLY A BY 30
                ; INPUT:  A   -  8 BIT NUMBER
                ; OUTPUT: H,L - 16 BIT NUMBER
                ; DESTROYS: A,B,C,H,L,F

0AE5 F5         MULT30: PUSH    PSW     ;SAVE IT
0AE6 07                 RLC
0AE7 07                 RLC
0AE8 07                 RLC
0AE9 07                 RLC
0AEA 07                 RLC             ;32*Y
0AEB 4F                 MOV     C,A
0AEC E61F               ANI     01FH    ;GET CARRY
0AEE 47                 MOV     B,A
0AEF 79                 MOV     A,C
0AF0 E6E0               ANI     0E0H    ;MASK CARRY
0AF2 4F                 MOV     C,A     ;B,C NOW = 32Y
0AF3 F1                 POP     PSW     ;GET Y
0AF4 07                 RLC             ;2*Y
0AF5 6F                 MOV     L,A
0AF6 E601               ANI     01H     ;GET CARRY
0AF8 2F                 CMA             ;COMPLEMEMT IT
0AF9 67                 MOV     H,A
0AFA 7D                 MOV     A,L
0AFB E6FE               ANI     0FEH    ;MASK CARRY
0AFD 2F                 CMA
0AFE 6F                 MOV     L,A
0AFF 23                 INX     H       ;H,L NOW = -2Y
0B00 09                 DAD     B       ;H,L = 30 Y
0B01 C9                 RET
                ;
                ;*****************************************************
                ;
                ;
```



```
                ; FUNCTION: INCREMENT SCREEN ADDRESS OF SYMBOL
                ; INPUTS:   H,L    - CURRENT SCREEN ADDRESS
                ;           SYMXCT - SYMBOL X COUNTER
                ;           SYMXD  - SYMBOL WIDTH
                ;           SYMHT  - SYMBOL HEIGHT
                ;           SYMYCT - SYMBOL Y COUNTER
                ; OUTPUTS:  A      - 0 = DONE (0 FLAG SET)
                ;           H,L    - NEXT SCREEN ADD. = SADSV
                ;           SADRE  - SCR ADR (SYM RIGHT EDGE) + 1
                ;           CARRY  - SET = CURRENTLY AT SYM RIGHT EDGE
                ; CALLS: FIXADD
                ; DESTROYS: A,F,H,L
                ; DESCRIPTION:  THE NEXT SCREEN ADDRESS IS DETERMINED
                ;               AND PLACED IN SADSV. RETURNS WITH CARRY
                ;               SET IF AT RIGHT EDGE OF SYMBOL AND WITH
                ;               ZERO FLAG SET IF AT END OF SYMBOL.
                ;               ON INITIAL CALL SET   SYMXCT = SYMWD,
                ;               SYMYCT = SYMHT , AND SADSV = SCREEN ADR.
                ;
0B02 3A0E11     INCSAD: LDA     SYMXCT
0B05 3D                 DCR     A       ;DEC X COUNTER
0B06 320E11             STA     SYMXCT
0B09 CA100B             JZ      INC05   ;AT RIGHT EDGE ?
0B0C 23                 INX     H       ;NO, INC. SAD
0B0D C33E0B             JMP     FRET
0B10 3A0C11     INC05:  LDA     SYMWD   ;GET SYMBOL WIDTH
0B13 320E11             STA     SYMXCT  ;PUT IN X COUNTER
0B16 3A0F11             LDA     SYMYCT  ;GET Y COUNT
0B19 3D                 DCR     A       ;DECREMENT IT
0B1A 320F11             STA     SYMYCT
0B1D CA370B             JZ      INC10   ;ALL DONE
0B20 D5                 PUSH    D       ;SAVE DE
0B21 23                 INX     H       ;SCR ADR RIGHT EDGE + 1
0B22 220511             SHLD    SADRE   ;STORE IT
0B25 2A0711             LHLD    SADSV   ;GET ADD OF BEG OF LINE
0B28 111E00             LXI     D,01EH  ;JUMP DOWN TO NEXT LINE (+1EH)
0B2B 19                 DAD     D
0B2C CD430B             CALL    FIXADR  ;CORRECT FOR GAP IN DISP. RAM
0B2F 220711             SHLD    SADSV
0B32 D1                 POP     D       ;RESTORE DE
0B33 B7                 ORA     A       ;SET FLAGS (ZERO F CLR)
0B34 C3410B             JMP     SRET    ;DONE
0B37 23         INC10:  INX     H       ;LAST ADD+1
0B38 220511             SHLD    SADRE   ;SAVE IT
0B3B C3410B             JMP     SRET    ;(ZERO F SET)
                ;
                ;
                ;
0B3E 37         FRET:   STC     ;RETURN WITH CARRY CLEAR
0B3F 3F                 CMC
0B40 C9                 RET
                ;
                ;
0B41 37         SRET:   STC     ;RETURN WITH CARRY SET
0B42 C9                 RET
```

```
                ;
                ;*****************************************************
                ;
                ; FUNCTION: FIX SCREEN ADDRESS
                ; INPUTS:   H,L - SCREEN ADDRESS
                ;       SCRBYT  - SCREEN BYTE (00 = TOP,FF = BOT)
                ; OUTPUTS:  H,L - SCREEN ADDRESS (CORRECTED)
                ;       SCRBYT  - SCREEN BYTE
                ; CALLS: NOTHING
                ; DESTROYS: NOTHING
                ; DESCRIPTION:  FIXADR IS CALLED BETWEEN EACH
                ;               INCREMENT OF THE SCREEN ADDRESS.
                ;               80 HEX (128 D) IS ADDED TO FIRST
                ;               SCREEN ADDRESS GREATER THAN 477FH TO
                ;               CORRECT FOR THE GAP IN DISPLAY RAM IN
                ;               THIS AREA. THE WORD SCRBYT IS THEN
                ;               TOGGLED TO FF HEX TO STOP FURTHER
                ;               CORRECTIONS.
                ;
0B43 F5         FIXADR: PUSH    PSW
0B44 3A1011             LDA     SCRBYT
0B47 B7                 ORA     A       ;SET FLAGS
0B48 FA5F0B             JM      FIX10   ;LOWER HALF OF SCREEN? YES
0B4B D5                 PUSH    D       ;DONT KNOW
0B4C 118047             LXI     D,4780H ;LOWER
0B4F CD8000             CALL    HILO    ;HALF OF SCREEN?
0B52 D25E0B             JNC     FIX05   ;
0B55 118000             LXI     D,080H  ;YES
0B58 19                 DAD     D       ;ADD OFFSET
0B59 3EFF               MVI     A,0FFH
0B5B 321011             STA     SCRBYT  ;TOGGLE SCRBYT TO FF
0B5E D1         FIX05:  POP     D
0B5F F1         FIX10:  POP     PSW
0B60 C9                 RET
                ;
                ;*****************************************************
                ;
                ; SYMBOL STORAGE / LOOKUP ROUTINES
                ;
                ;*****************************************************
                ;
                ;
                ; FUNCTION: LOOKUP SYMBOL ADDRESS
                ; INPUTS:  A -  SYMBOL IDENT
                ; OUTPUTS: H,L - ADDRESS OF SYMBOL
                ;          CARRY - CLEAR, SYMBOL NOT FOUND
                ; DESTROYS: A,B,D,E,H,L,F
                ; DESCRIPTION: LOCATES SYMBOL IDENT IN SYMTAB AND
                ;               RETURNS THE ADDRESS OF THE STORED SYMBOL.
                ;               CARRY IS SET IF FOUND.
                ;
0B61 67         SYMADR: MOV     H,A     ;SAVE IDENT
0B62 3A1111             LDA     NUMSYM  ;# OF SYMBOLS
```



```
0B65 B7                 ORA     A       ;SET FLAGS
0B66 CA3E0B             JZ      FRET    ;DONE IF # SYMBOLS = 0
0B69 47                 MOV     B,A     ;PUT IN B
0B6A 7C                 MOV     A,H     ;GET IDENT
0B6B 211811             LXI     H,SYMTAB        ;ADR OF SYMBOL TABLE
0B6E 110300             LXI     D,03H   ;LENGTH OF TABLE ENTRY
0B71 BE         SYM05:  CMP     M
0B72 CA7D0B             JZ      SYM10
0B75 19                 DAD     D       ;NEXT TABLE ENTRY
0B76 05                 DCR     B       ;DEC LOOP COUNT
0B77 C2710B             JNZ     SYM05
0B7A C33E0B             JMP     FRET
0B7D 23         SYM10:  INX     H       ;GET ADDRESS
0B7E 7E                 MOV     A,M     ;OF SYMBOL
0B7F 23                 INX     H
0B80 66                 MOV     H,M
0B81 6F                 MOV     L,A     ;PUT IN H,L
0B82 C3410B             JMP     SRET
                ;
                ;*****************************************************
                ;
                ; FUNCTION: GET SYMBOL
                ; INPUTS: A  - SYMBOL IDENT
                ; OUTPUTS: H,L - STORAGE ADDRESS
                ; DESTROYS: A,B,C,D,H,L,F
                ; DESCRIPTION:  ADDS SYMBOL IDENT AND STORAGE ADDRESS
                ;               TO SYMBOL TABLE. THE BYTES DEFINING THE
                ;               SYMBOL ARE THEN PLUCKED FROM THE INPUT
                ;               BUFFER AND STORED IN RAM.
                ;
                ;
0B85 CDDB0B     GETSYM: CALL    ADDSYM
0B88 2A1411             LHLD    NEWDAT
0B8B 3EFF               MVI     A,0FFH
0B8D 77                 MOV     M,A     ;STORE X POS = FF
0B8E 23                 INX     H
0B8F 3EFF               MVI     A,0FFH
0B91 77                 MOV     M,A     ;STORE Y POS = FF
0B92 23                 INX     H
0B93 CDD00C             CALL    GETBYT  ;GET HEIGHT OF SYM
0B96 4F                 MOV     C,A
0B97 320F11             STA     SYMYCT  ;PUT IN Y COUNTER
0B9A B7                 ORA     A
0B9B FACD0B             JM      GTS15   ;ERROR - TOO WIDE
0B9E 77                 MOV     M,A     ;STORE HEIGHT
0B9F 23                 INX     H
0BA0 CDD00C             CALL    GETBYT  ;GET WIDTH
0BA3 320C11             STA     SYMWD
0BA6 57                 MOV     D,A
0BA7 3C                 INR     A
0BA8 3C                 INR     A
0BA9 E6E0               ANI     0E0H
0BAB C2CD0B             JNZ     GTS15   ;ERROR - TOO HIGH
0BAE 72                 MOV     M,D     ;STORE WIDTH
0BAF 23         GTS05:  INX     H
```

```
0BB0 CDD00C          CALL    GETBYT  ;GET SYM DATA
0BB3 77              MOV     M,A     ;STORE IT
0BB4 15              DCR     D
0BB5 C2AF0B          JNZ     GTS05   ;END OF DATA ?
0BB8 3A0C11          LDA     SYMWD
0BBB 57              MOV     D,A
0BBC 3A0F11          LDA     SYMYCT
0BBF 3D              DCR     A
0BC0 320F11          STA     SYMYCT
0BC3 C2AF0B          JNZ     GTS05
0BC6 23        GTS10: INX    H
0BC7 221411          SHLD    NEWDAT
0BCA C3410B          JMP     SRET
0BCD 2A1111    GTS15: LHLD   NUMSYM  ;ERROR - DELETE ENTRY
0BD0 35              DCR     M
0BD1 2A1211          LHLD    NEWSYM
0BD4 35              DCR     M
0BD5 35              DCR     M
0BD6 35              DCR     M
0BD7 35              DCR     M
0BD8 C33E0B          JMP     FRET
                ;
                ;*****************************************************
                ;
                ; FUNCTION: ADD SYMBOL
                ; INPUTS: A  - SYMBOL IDENT
                ; OUTPUTS: H,L - STORAGE ADDRESS
                ; DESTROYS: A,B,C,D,H,L,F
                ; DESCRIPTION:  ADDS SYMBOL IDENT AND STORAGE ADDRESS
                ;               TO SYMBOL TABLE.
                ;
0BDB 211111    ADDSYM: LXI    H,NUMSYM
0BDE 34               INR     M       ;INC # OF SYMBOLS
0BDF 2A1211           LHLD    NEWSYM  ;ADR FOR NEW TABLE ENTRY
0BE2 77               MOV     M,A     ;STORE SYM IDENT
0BE3 23               INX     H
0BE4 EB               XCHG
0BE5 2A1411           LHLD    NEWDAT  ;ADR FOR NEW SYM DATA
0BE8 7D               MOV     A,L
0BE9 12               STAX    D       ;PUT NEWDAT POINTER IN SYMTAB
0BEA 13               INX     D
0BEB 7C               MOV     A,H
0BEC 12               STAX    D
0BED 13               INX     D
0BEE EB               XCHG
0BEF 221211           SHLD    NEWSYM  ;INC TABLE POINTER
0BF2 EB               XCHG
0BF3 C9               RET
                ;
                ;
                ;*****************************************************
                ;
                ; STRING STORAGE / LOOKUP ROUTINES
                ;
                ;*****************************************************
```



```
                ;FUNCTION: ADDSTR  (ADD STRING)
                ;INPUTS:    A - SYMBOL IDENT
                ;OUTPUTS: H,L - STORAGE ADDRESS
                ;CALLS: GETSTR, ADDSYM
                ;DESTROYS: A,B,C,D,E,F,H,L
                ;DESCRIPTION: ADDS STRING TO SYMBOL TABLE, THEN GETS
                ;             STRING FROM THE INPUT BUFFER AND STORES
                ;             IT.
                ;
0BF4 CDDB0B    ADDSTR: CALL    ADDSYM
0BF7 E5               PUSH    H       ;SAVE STORAGE ADR
0BF8 3EFF             MVI     A,0FFH
0BFA 77               MOV     M,A
0BFB 23               INX     H
0BFC 77               MOV     M,A
0BFD 23               INX     H
0BFE CD220C           CALL    GETSTR
0C01 E1               POP     H       ;RESTORE STORAGE ADR
0C02 C9               RET
                ;
                ;*****************************************************
                ; FUNCTION : PUTSTR (PUT STRING ON SCREEN AT X,Y CORD)
                ; INPUTS: H,L - ADDRESS OF STRING
                ; OUTPUTS: B,C - X,Y COORDINATES
                ; DESCRIPTION: TAKES THE CHARACTER STRING POINTED TO
                ;              BY  H,L AND PLACES IT ON THE SCREEN
                ;              STARTING AT THE SPECIFIED X,Y POSITION.
                ;
0C03 46        PUTSTR: MOV     B,M
0C04 23               INX     H
0C05 4E               MOV     C,M
0C06 23               INX     H
0C07 C5               PUSH    B       ;SAVE X,Y COORDINATES
0C08 7E        PST05:  MOV     A,M
0C09 FE0D             CPI     CR      ;END OF STRING?
0C0B CA200C           JZ      PST10
0C0E EB               XCHG            ;STR POINTER TO DE
0C0F 210107           LXI     H,0701H ;HEIGHT=7, WIDTH=1
0C12 CD060A           CALL    PUT
0C15 EB               XCHG
0C16 C1               POP     B       ;OLD X,Y
0C17 3A1711           LDA     CSPACE  ;GET CHAR. SPACING
0C1A 80               ADD     B
0C1B 47               MOV     B,A
0C1C C5               PUSH    B       ;SAVE NEW X,Y
0C1D C3080C           JMP     PST05
0C20 C1        PST10:  POP     B
0C21 C9               RET
                ;
                ;
                ;
                ;*****************************************************
                ; FUNCTION: GETSTR
                ; INPUTS: H,L - STORAGE ADDRESS
```

```
                ; DESCRIPTION: ACCEPTS A STRING OF CHARACTERS UP TO
                ;              THE FIRST CARRAGE RETURN AND STORES THE
                ;              CHARACTER DATA STARTING AT THE LOCATION
                ;              POINTED TO BY H,L
                ;
                ;
0C22 EB         GETSTR: XCHG
0C23 CDDF0C     STR05:  CALL    GETCHR  ;GET CHAR. IN A
0C26 FE0D               CPI     CR      ;CHECK FOR END OF INPUT
0C28 CA4E0C             JZ      STR10
0C2B D620               SUI     020H    ;ASCII 20H FIRST CHAR. IN TABLE
                ;
                ;       MULT A BY 7 TO GET LOCATION IN TABLE
                ;
0C2D E63F               ANI     03FH    ;MASK
0C2F 21640D             LXI     H,CHRTAB
0C32 0600               MVI     B,0H
0C34 F5                 PUSH    PSW
0C35 07                 RLC
0C36 07                 RLC
0C37 4F                 MOV     C,A
0C38 09                 DAD     B
0C39 F1                 POP     PSW
0C3A 4F                 MOV     C,A
0C3B 09                 DAD     B
0C3C 07                 RLC
0C3D 4F                 MOV     C,A
0C3E 09                 DAD     B       ;H,L NOW POINTS TO CHAR.
0C3F 0607               MVI     B,07H   ;BYTE COUNTER = 7
                ;
                ;       LOOP TO GET 7 LINES OF DATA FOR EACH CHARACTER.
                ;
0C41 7E         STR07:  MOV     A,M     ;GET BYTE FROM TABLE
0C42 EB                 XCHG
0C43 77                 MOV     M,A     ;STORE IT
0C44 EB                 XCHG
0C45 23                 INX     H       ;NEXT TABLE LOCATION
0C46 13                 INX     D       ;NEXT MEMORY LOCATION
0C47 05                 DCR     B       ;DONE WITH CHAR. YET?
0C48 C2410C             JNZ     STR07   ;NO
0C4B C3230C             JMP     STR05   ;YES, GET NEXT CHAR.
0C4E EB         STR10:  XCHG
0C4F 77                 MOV     M,A     ;PUT CR IN MEMORY AS END MARK
0C50 23                 INX     H
0C51 221411             SHLD    NEWDAT
0C54 CD170D             CALL    CRLF
0C57 C9                 RET
                ;
                ;*****************************************************
                ; FUNCTION: CLEAR STRING
                ; INPUTS: H,L - ADDRESS OF STRING
                ;
                ; DESCRIPTION: CLEARS STRING POINTED TO BY H,L BY
                ;              WRITING SPACES TO SCREEN FOR EVERY
```



```
                ;              CHARACTER IN STRING.
                ;
0C58 46         CLRSTR: MOV     B,M     ;X POS
0C59 23                 INX     H
0C5A 4E                 MOV     C,M     ;Y POS
0C5B 23                 INX     H
0C5C 78                 MOV     A,B
0C5D FEFF               CPI     0FFH    ;IF NOT ON SCREEN ALREADY
0C5F C8                 RZ              ;WERE DONE
0C60 7E         CST05:  MOV     A,M     ;GET DATA
0C61 FE0D               CPI     CR      ;END?
0C63 C8                 RZ              ;YES
0C64 E5                 PUSH    H       ;SAVE DATA POINTER
0C65 210107             LXI     H,0701H ;HEIGHT=7,WIDTH=1
0C68 11640D             LXI     D,CHRTAB;SPACE CHARACTER
0C6B C5                 PUSH    B       ;SAVE X,Y POSITION
0C6C CD060A             CALL    PUT     ;ERASE OLD STRING
0C6F C1                 POP     B       ;GET X,Y POS
0C70 3A1711             LDA     CSPACE
0C73 80                 ADD     B       ;ADD CSPACE TO X POS
0C74 47                 MOV     B,A
0C75 E1                 POP     H       ;GET DATA POINTER
0C76 110700             LXI     D,0007H ;NEXT CHARACTER
0C79 19                 DAD     D
0C7A C3600C             JMP     CST05
                ;
                ;
                ;*****************************************************
                ;
                ; INPUT / OUTPUT ROUTINES
                ;
                ;*****************************************************
                ;
                ; FUNCTION: READ FROM CHARACTER INPUT BUFFER
                ; OUT - A
                ; DESTROYS A,F
                ; DESCRIPTION:  READS ONE CHARACTER FROM BOTTOM OF
                ;               INPUT BUFFER (BUFBOT).
                ;               IF BUFFER EMPTY AND
                ;       FUNBYT = 00 THEN CALL CI ROUTINE
                ;       FUNBYT = FF THEN RETURN TO DEMO PROGRAM
                ;
0C7D E5         RDBUF:  PUSH    H
0C7E C5                 PUSH    B
0C7F 210010             LXI     H,BUFF
0C82 3A0011             LDA     BUFTOP
0C85 47                 MOV     B,A
0C86 3A0111     RD05:   LDA     BUFBOT
0C89 B8                 CMP     B
0C8A C2A10C             JNZ     RD10    ;BUFFER EMPTY? *
0C8D 3A0211             LDA     FUNBYT  ;FUNCTION RUNNING?
0C90 B7                 ORA     A       ;SET FLAGS
0C91 CA990C             JZ      RD06    ;NO
0C94 2A0311             LHLD    STKSAV  ;YES
0C97 F9                 SPHL            ;RESTORE OLD STACK
```

```
0C98 C9                 RET             ;RET TO FUNCTION
0C99 CD200D     RD06:   CALL    CI      ;*
0C9C E67F               ANI     07FH    ;MASK PARITY BIT*
0C9E C3AA0C             JMP     RD15
0CA1 6F         RD10:   MOV     L,A     ; NO
0CA2 F3                 DI
0CA3 46                 MOV     B,M     ; CHAR FROM BOT OF BUFF
0CA4 3C                 INR     A
0CA5 320111             STA     BUFBOT  ; INC BUFBOT
0CA8 78                 MOV     A,B
0CA9 FB                 EI
0CAA C1         RD15:   POP     B
0CAB E1                 POP     H
0CAC C3410B             JMP     SRET
                ;
                ;*****************************************************
                ;
                ; FUNCTION: WRITE TO CHARACTER INPUT BUFFER
                ; INPUT: A
                ; DESTROYS: A,F
                ; DESCRIPTION:  WRITES ONE CHARACTER TO TOP OF INPUT
                ;               BUFFER (BUFTOP).
0CAF C5         WRBUFR: PUSH    B
0CB0 E5                 PUSH    H
0CB1 4F                 MOV     C,A     ;STORE A TEMP
0CB2 210010             LXI     H,BUFF  ;ADR OF BUFFER
0CB5 3A0111             LDA     BUFBOT
0CB8 47                 MOV     B,A
0CB9 3A0011             LDA     BUFTOP
0CBC 6F                 MOV     L,A     ;H,L NOW POINTS TO DATA
0CBD 3C                 INR     A
0CBE B8                 CMP     B       ;BUFFER OVERFLOW?
0CBF CACB0C             JZ      BUFFUL  ;YES
0CC2 F3                 DI
0CC3 320011             STA     BUFTOP  ;NO, INC BUFTOP
0CC6 71                 MOV     M,C     ;CHAR TO TOP OF BUFF
0CC7 FB                 EI
0CC8 E1                 POP     H
0CC9 C1                 POP     B
0CCA C9                 RET
                ;
                ;
0CCB 3E01       BUFFUL: MVI     A,01H   ;ERROR #1
0CCD CD0800             CALL    MGO     ;JUMP TO MONITOR
                ;
                ;
                ;
                ;*****************************************************
                ; RETURNS WITH HEX BYTE IN REG. A EQUAL TO THE VALUE OF
                ; THE FIRST TWO ASCII DIGITS IN THE INPUT BUFFER.
                ;
                ;
0CD0 CD7D0C     GETBYT: CALL    RDBUF   ;DESTROYS: A,B,C,F
0CD3 E67F               ANI     07FH    ;MASK PARITY BIT
0CD5 47                 MOV     B,A
```



```
0CD6 CD7D0C             CALL    RDBUF
0CD9 E67F               ANI     07FH    ;MASK PARITY
0CDB CD2405             CALL    CONV
0CDE C9                 RET
                ;
0CDF CD7D0C     GETCHR: CALL    RDBUF   ;DESTROYS: A,F
0CE2 E67F               ANI     07FH    ; MASK PARITY
0CE4 C9                 RET
                ;
                ;
0CE5 CDDF0C     GETCR:  CALL    GETCHR  ;DESTROYS: A,C,F
0CE8 4F                 MOV     C,A
0CE9 CD030D             CALL    ECHO
0CEC 79                 MOV     A,C
0CED FE0D               CPI     CR
0CEF C27D08             JNZ     ERROR
0CF2 C9                 RET
                ;
                ; GET NEXT CHARACTER - ERROR IF NOT DELIMITER.
                ;
0CF3 CDDF0C     GETDL:  CALL    GETCHR  ;DESTROYS: A,C,F
0CF6 FE0D               CPI     CR
0CF8 CA7D08             JZ      ERROR
0CFB 4F         GDL:    MOV     C,A
0CFC CD6604             CALL    VALDL
0CFF D27D08             JNC     ERROR
0D02 C9                 RET
                ;
                ;
                ;
                ;       DESTROYS: A,C,F
0D03 3A1611     ECHO:   LDA     EFLAG   ;INHIBIT ECHO
0D06 B7                 ORA     A
0D07 C0                 RNZ             ;IF NOT = 0
0D08 CD2B0D             CALL    CO      ;INPUT REG. C
0D0B 3E0D               MVI     A,CR
0D0D B9                 CMP     C
0D0E C0                 RNZ
0D0F 0E0A               MVI     C,LF    ;ECHO CR AS CRLF
0D11 CD2B0D             CALL    CO
0D14 0E0D               MVI     C,CR
0D16 C9                 RET             ;RET WITH CHAR IN C
                ;
                ;
0D17 3A1611     CRLF:   LDA     EFLAG   ;ECHO SUPRESSED?
0D1A B7                 ORA     A
0D1B C0                 RNZ             ;YES
0D1C CD6E00             CALL    MCRLF   ;NO
0D1F C9                 RET
                ;
                ;
0D20 CD6500     CI:     CALL    MCI     ;CHAR. INPUT IN A
0D23 C5                 PUSH    B       ;DESTROYS ONLY A,F
0D24 4F                 MOV     C,A
0D25 CD030D             CALL    ECHO
```

```
0D28 79             MOV     A,C
0D29 C1             POP     B
0D2A C9             RET
                ;
                ;
0D2B CD6800     CO:     CALL    MCO
0D2E C9             RET
                ;
                ;***************************************************
                ;
                ; TABLES
                ;
                ;***************************************************
                ;
                ;
                ;
0D2F 00         MSKTAB: DB      00000000B       ;USED BY GETMSK
0D30 80                 DB      10000000B
0D31 C0                 DB      11000000B
0D32 E0                 DB      11100000B
0D33 F0                 DB      11110000B
0D34 F8                 DB      11111000B
0D35 FC                 DB      11111100B
0D36 FE                 DB      11111110B
0D37 FF                 DB      11111111B
0D38            CTAB:   DS      0H      ;COMMAND CHAR TABLE
0D38 41                 DB      'A'     ;ASCII STRING
0D39 43                 DB      'C'     ;CLEAR SCREEN
0D3A 44                 DB      'D'     ;DEFINE SYMBOL
0D3B 45                 DB      'E'     ;ECHO ON
0D3C 46                 DB      'F'     ;FUNCTION
0D3D 49                 DB      'I'     ;INIT SYM. TABLE
0D3E 4B                 DB      'K'     ;KILL ECHO
0D3F 4C                 DB      'L'     ;LIST OF POINTS
0D40 4D                 DB      'M'     ;MOVE SYMBOL
0D41 4E                 DB      'N'     ;NARROW TEXT
0D42 50                 DB      'P'     ;PUT SYMBOL
0D43 57                 DB      'W'     ;WIDE TEXT
0D44 58                 DB      'X'     ;XOR SYMBOL
0D45 5A                 DB      'Z'     ;JUMP TO MONITOR
000E =          NCMDS   EQU     14D
0D46            CADR:   DS      0H      ;COMMAND ADDRESS TABLE
0D46 0000               DW      00H     ;DUMMY
0D48 B009               DW      ZCMD
0D4A 8309               DW      XCMD
0D4C 7809               DW      WCMD
0D4E 6F09               DW      PCMD
0D50 6409               DW      NCMD
0D52 8C09               DW      MCMD
0D54 3E09               DW      LCMD
0D56 3309               DW      KCMD
0D58 2A09               DW      ICMD
0D5A 1C09               DW      FCMD
0D5C 1209               DW      ECMD
0D5E F708               DW      DCMD
```



```
0D60 DC08               DW      CCMD
0D62 8808               DW      ACMD
                ;
                ;***************************************************
                ;
                ; CHARACTER GENERATOR TABLES
                ;
                ;***************************************************
                ;
                ; 5 BY 7 DOT MATRIX
                ; ASCII VALUES:  20H (SP),   TO   60H (')
                ;                NUMBERS, UPPER CASE LETTERS, SYMBOLS
                ;
0D64 0000000000CHRTAB:  DB      000H,000H,000H,000H,000H        ;SP
0D69 0000               DB      000H,000H                       ;(=20H)
0D6B 2020202020         DB      020H,020H,020H,020H,020H        ;
0D70 0020               DB      000H,020H
0D72 5050500000         DB      050H,050H,050H,000H,000H        ;"
0D77 0000               DB      000H,000H
0D79 5050F850F8         DB      050H,050H,0F8H,050H,0F8H        ;#
0D7E 5050               DB      050H,050H
0D80 2078A07028         DB      020H,078H,0A0H,070H,028H        ;$
0D85 F020               DB      0F0H,020H
0D87 C0C8102040         DB      0C0H,0C8H,010H,020H,040H        ;%
0D8C 9818               DB      098H,018H
0D8E 40A0A040A8         DB      040H,0A0H,0A0H,040H,0A8H        ;&
0D93 9068               DB      090H,068H
0D95 2020200000         DB      020H,020H,020H,000H,000H        ;'
0D9A 0000               DB      000H,000H
0D9C 2040808080         DB      020H,040H,080H,080H,080H        ;(
0DA1 4020               DB      040H,020H
0DA3 2010080808         DB      020H,010H,008H,008H,008H        ;)
0DA8 1020               DB      010H,020H
0DAA 20A8702070         DB      020H,0A8H,070H,020H,070H        ;*
0DAF A820               DB      0A8H,020H
0DB1 002020F820         DB      000H,020H,020H,0F8H,020H        ;+
0DB6 2000               DB      020H,000H
0DB8 0000000020         DB      000H,000H,000H,000H,020H        ;,
0DBD 2040               DB      020H,040H
0DBF 000000FE00         DB      000H,000H,000H,0FEH,000H        ;-
0DC4 0000               DB      000H,000H
0DC6 0000000000         DB      000H,000H,000H,000H,000H        ;.
0DCB 0020               DB      000H,020H
0DCD 0608102040         DB      006H,008H,010H,020H,040H        ;/
0DD2 8000               DB      080H,000H
0DD4 708898A8C8         DB      070H,088H,098H,0A8H,0C8H        ;0
0DD9 8870               DB      088H,070H
0DDB 2060202020         DB      020H,060H,020H,020H,020H        ;1
0DE0 2070               DB      020H,070H
0DE2 7088087080         DB      070H,088H,008H,070H,080H        ;2
0DE7 80F8               DB      080H,0F8H
0DE9 F808103008         DB      0F8H,008H,010H,030H,008H        ;3
0DEE 8870               DB      088H,070H
0DF0 10305090F8         DB      010H,030H,050H,090H,0F8H        ;4
0DF5 1010               DB      010H,010H
```

```
0DF7 F880F00808          DB      0F8H,080H,0F0H,008H,008H        ;5
0DFC 8870                DB      088H,070H
0DFE 384080F088          DB      038H,040H,080H,0F0H,088H        ;6
0E03 8870                DB      088H,070H
0E05 F808102020          DB      0F8H,008H,010H,020H,020H        ;7
0E0A 2020                DB      020H,020H
0E0C 7088887088          DB      070H,088H,088H,070H,088H        ;8
0E11 8870                DB      088H,070H
0E13 7088887808          DB      070H,088H,088H,078H,008H        ;9
0E18 10E0                DB      010H,0E0H
0E1A 0000200020          DB      000H,000H,020H,000H,020H        ;:
0E1F 0000                DB      000H,000H
0E21 0000200020          DB      000H,000H,020H,000H,020H        ;;
0E26 2040                DB      020H,040H
0E28 1020408040          DB      010H,020H,040H,080H,040H        ;<
0E2D 2010                DB      020H,010H
0E2F 0000F800F8          DB      000H,000H,0F8H,000H,0F8H        ;=
0E34 0000                DB      000H,000H
0E36 4020100810          DB      040H,020H,010H,008H,010H        ;>
0E3B 2040                DB      020H,040H
0E3D 7088083020          DB      070H,088H,008H,030H,020H        ;?
0E42 0020                DB      000H,020H
0E44 7088A8B8B0          DB      070H,088H,0A8H,0B8H,0B0H        ;@
0E49 8078                DB      080H,078H
0E4B 20508888F8          DB      020H,050H,088H,088H,0F8H        ;A
0E50 8888                DB      088H,088H
0E52 F08888F088          DB      0F0H,088H,088H,0F0H,088H        ;B
0E57 88F0                DB      088H,0F0H
0E59 7088808080          DB      070H,088H,080H,080H,080H        ;C
0E5E 8870                DB      088H,070H
0E60 F088888888          DB      0F0H,088H,088H,088H,088H        ;D
0E65 88F0                DB      088H,0F0H
0E67 F88080F080          DB      0F8H,080H,080H,0F0H,080H        ;E
0E6C 80F8                DB      080H,0F8H
0E6E F88080F080          DB      0F8H,080H,080H,0F0H,080H        ;F
0E73 8080                DB      080H,080H
0E75 7888808098          DB      078H,088H,080H,080H,098H        ;G
0E7A 8878                DB      088H,078H
0E7C 888888F888          DB      088H,088H,088H,0F8H,088H        ;H
0E81 8888                DB      088H,088H
0E83 7020202020          DB      070H,020H,020H,020H,020H        ;I
0E88 2070                DB      020H,070H
0E8A 0808080808          DB      008H,008H,008H,008H,008H        ;J
0E8F 8870                DB      088H,070H
0E91 8890A0C0A0          DB      088H,090H,0A0H,0C0H,0A0H        ;K
0E96 9088                DB      090H,088H
0E98 8080808080          DB      080H,080H,080H,080H,080H        ;L
0E9D 80F8                DB      080H,0F8H
0E9F 88D8A8A8A8          DB      088H,0D8H,0A8H,0A8H,0A8H        ;M
0EA4 8888                DB      088H,088H
0EA6 8888C8A898          DB      088H,088H,0C8H,0A8H,098H        ;N
0EAB 8888                DB      088H,088H
0EAD 7088888888          DB      070H,088H,088H,088H,088H        ;O
0EB2 8870                DB      088H,070H
0EB4 F08888F080          DB      0F0H,088H,088H,0F0H,080H        ;P
```



```
0EB9 8080                DB      080H,080H
0EBB 70888888A8          DB      070H,088H,088H,088H,0A8H        ;Q
0EC0 9068                DB      090H,068H
0EC2 F08888F0A0          DB      0F0H,088H,088H,0F0H,0A0H        ;R
0EC7 9088                DB      090H,088H
0EC9 7088807008          DB      070H,088H,080H,070H,008H        ;S
0ECE 8870                DB      088H,070H
0ED0 F8A8202020          DB      0F8H,0A8H,020H,020H,020H        ;T
0ED5 2020                DB      020H,020H
0ED7 8888888888          DB      088H,088H,088H,088H,088H        ;U
0EDC 8870                DB      088H,070H
0EDE 8888885050          DB      088H,088H,088H,050H,050H        ;V
0EE3 2020                DB      020H,020H
0EE5 888888A8A8          DB      088H,088H,088H,0A8H,0A8H        ;W
0EEA A850                DB      0A8H,050H
0EEC 8888702070          DB      088H,088H,070H,020H,070H        ;X
0EF1 8888                DB      088H,088H
0EF3 8888702020          DB      088H,088H,070H,020H,020H        ;Y
0EF8 2020                DB      020H,020H
0EFA F808102040          DB      0F8H,008H,010H,020H,040H        ;Z
0EFF 80F8                DB      080H,0F8H
0F01 F8C0C0C0C0          DB      0F8H,0C0H,0C0H,0C0H,0C0H        ;[
0F06 C0F8                DB      0C0H,0F8H
0F08 0080402010          DB      000H,080H,040H,020H,010H        ;\
0F0D 0800                DB      008H,000H
0F0F F818181818          DB      0F8H,018H,018H,018H,018H        ;]
0F14 18F8                DB      018H,0F8H
0F16 0000205088          DB      000H,000H,020H,050H,088H        ;^
0F1B 0000                DB      000H,000H
0F1D 0000000000          DB      000H,000H,000H,000H,000H        ;_
0F22 00F8                DB      000H,0F8H
0F24 4020100000          DB      040H,020H,010H,000H,000H        ;'
0F29 0000                DB      000H,000H                       ;(=60
                ;
                ;
                ;*************************************************
                ;
                ; VARIABLES
                ;
                ;*************************************************
                ;
1000            ORG     01000H
                ;
1000            BUFF:   DS      0100H   ;START ADR OF INPUT BUFFER
1100            BUFTOP: DS      01H     ;TOP OF BUFFER
1101            BUFBOT: DS      01H     ;BOT OF BUFFER
1102            FUNBYT: DS      01H     ;FUNCTION RUNNING BYTE
1103            STKSAV: DS      02H     ;SP SAVE LOCATION
1105            SADRE:  DS      02H     ;SCREEN ADR END OF LINE
1107            SADSV:  DS      02H     ;SCREEN ADR TEMP SAVE
1109            SBITS:  DS      01H     ;SCREEN BIT OFFSET
110A            RMASK:  DS      01H     ;8-N LSB'S SET
110B                    DS      01H     ;N MSB'S SET
110C            SYMWD:  DS      01H     ;SYM WIDTH
110D            SYMHT:  DS      01H     ;SYM HEIGHT
```

```
110E            SYMXCT: DS      01H     ;SYM X COUNTER
110F            SYMYCT: DS      01H     ;SYM Y COUNTER
1110            SCRBYT: DS      01H     ;SCREEN FLAG 0=TOP, FF=BOT
1111            NUMSYM: DS      01H     ;NUMBER OF SYMBOLS IN SYMTAB
1112            NEWSYM: DS      02H     ;NEXT ADR IN SYMTAB
1114            NEWDAT: DS      02H     ;NEXT DATA STORAGE ADR
1116            EFLAG:  DS      01H     ;ECHO SUPRESSION FLAG
1117            CSPACE: DS      01H     ;CHAR. SPACING (6H OR 7H)
1118            SYMTAB: DS      01H     ;SYMBOL IDENT
1119                    DS      02H     ;STORAGE ADRESS
111B                    DS      3DH     ;ECT. FOR ALL SYMBOLS
                ;
                ;       STACK GOES IN HERE
1158                    DS      050H
11A8            STACK:  DS      0H
                ;
1300            ORG     01300H
1300            DATA:   DS      0H      ;SYMBOL DATA
                ;
1300            END
```

```
                ;*****************************************************
                ;
                ;ILS INDICATOR DEMONSTRATION PROGRAM
                ;
                ;*****************************************************
                ;
                ;       EQUATES
00C3 =          JMPINST EQU     0C3H    ;JUMP INSTRUCTION
20F0 =          RST7P5  EQU     020F0H  ;RST 7.5 INTERUPT LOCATION
0030 =          SIM     EQU     030H    ;SIM, RIM  NOT RECOGNIZED BY
0020 =          RIM     EQU     020H    ;THIS ASSEMBLER
000D =          CR      EQU     0DH     ;CARRIAGE RETURN
0850 =          WRBUF   EQU     0850H   ;CHAR IN A TO INPUT BUFFER
0853 =          GETCMD  EQU     0853H   ;LCD GET COMMAND ROUTINE
0092 =          PRVAL   EQU     0092H   ;CONVERT C. FROM HEX TO ASCII
009B =          DELAY   EQU     009BH   ;DELAY PROP TO CONTENTS OF D,E
05C8 =          BCDADD  EQU     05C8H   ;BCD ADD, HL > NUM1, DE > NUM2
                ;                        RESULT REPLACES NUM1
0479 =          GETBAUD EQU     0479H   ;SET BAUD FROM SPACE CHAR
00FF =          ENDCMD  EQU     0FFH    ;END OF COMMAND STRING MARK
00FF =          FUNON   EQU     0FFH
0000 =          FUNOFF  EQU     00H
00FF =          TIMEON  EQU     0FFH    ;ENABLE TIME DISPLAY
                ;
3000            ORG     03000H
3000 BB                 DB      0BBH            ;AUTOSTART BYTE
                ;*****************************************************
                ; INITILIZATION
                ;*****************************************************
                ;
                ;THIS PROGRAM INITILIZES THE TIMER FOR 1MS INTERUPTS
                ;AND PORT A = OUTPUT, PORT B = INPUT, PORT C = INPUT
                ;
3001 F3         INIT:   DI
3002 315012             LXI     SP,FSTACK       ;INIT STACK
                ;SET UP INTERUPTS
3005 21F020             LXI     H,RST7P5
3008 36C3               MVI     M,JMPINST       ;CODE FOR JUMP
300A 217C30             LXI     H,CLOCK         ;ADR OF CLOCK ROUTINE
300D 22F120             SHLD    RST7P5+1
                ;SET UP TIMER AND PORTS
3010 3E00               MVI     A,00H   ;TIMER L.O. COUNT
3012 D324               OUT     024H
3014 3E4C               MVI     A,04CH  ;TIMER H.O. COUNT
3016 D325               OUT     025H    ;SQUARE WAVE OUT
3018 3EC1               MVI     A,0C1H  ;START TIMER
301A D320               OUT     020H    ;PORT A OUT
                                        ;PORT B IN
                ;INIT VARIABLES
301C 3E0A               MVI     A,010D
301E 325012             STA     THSCNT
3021 325112             STA     HNDCNT
3024 325212             STA     TENCNT
```



```
3027 3EFF               MVI     A,0FFH
3029 320211             STA     FUNBYT  ;FUNCTION ON
302C AF                 XRA     A       ;ZERO
302D 325D12             STA     TIME    ;TIME = 0
3030 325E12             STA     TIME+1
3033 325F12             STA     TIME+2
3036 3E1F               MVI     A,01FH  ;MASK ALL INTERUPTS
3038 30                 DB      SIM
3039 21FA32             LXI     H,CMDS1
303C CDDB30             CALL    SNDCMD  ;SIGN ON MESSAGE
303F CDEF30             CALL    DOCMD   ;PROCESS COMMANDS
3042 C35330             JMP     SW
                ;
3050            ORG     03050H
                ;*****************************************************
                ; JUMP TABLE
                ;*****************************************************
                ;
3050 C3BF31     TFUNCT: JMP     FUNCT
                ;
                ;*****************************************************
                ;
                ;WAIT FOR USER TO PRESS BUTTON
                ;
3053 CDB331     SW:     CALL    GETSW   ;GET SWITCH PRESS
3056 FE02               CPI     02H     ;DEMO OFF MODE?
3058 CA6930             JZ      DOFF    ;YES
305B FE01               CPI     01H     ;DEMO MODE?
305D C25330             JNZ     SW      ;NOT VALID SWITCH
3060 3EFF               MVI     A,FUNON
3062 320211             STA     FUNBYT  ;FUNCT. ON
3065 FB                 EI              ;ENABLE INTERUPTS FOR CLOCK
3066 C3C031             JMP     DEMO1   ;AIRCRAFT INSTURUMENT DEMO
3069 3E00       DOFF:   MVI     A,FUNOFF;ZERO
306B 320211             STA     FUNBYT  ;FUNCT OFF
306E 3E0F               MVI     A,0FH   ;MASK INTERUPTS
3070 30                 DB      SIM
3071 3E45               MVI     A,'E'
3073 CDE630             CALL    SSCMD   ;TURN ON ECHO
3076 CD7904             CALL    GETBAUD
3079 C35308             JMP     GETCMD
                ;
                ;*****************************************************
                ;INTERUPT SERVICE ROUTINES
                ;*****************************************************
                ;
                ; FUNCTION: CLOCK
                ; ENTRY:    INTERUPT DRIVEN
                ; FUNCTION: SCEDULES PROGRAMS
                ;
307C F5         CLOCK:  PUSH    PSW     ;SAVE ENVIRONMENT
307D C5                 PUSH    B
307E D5                 PUSH    D
307F E5                 PUSH    H
3080 3A5012             LDA     THSCNT  ;THOUSANTS COUNT
```

```
3083 3D              DCR     A
3084 CA8D30          JZ      HNDTH
3087 325012          STA     THSCNT
                     ;
                     ;PROGRAMS SCEDULED EVERY 1MS BELOW
                     ;
                     ;NONE
                     ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
308A C3B930          JMP     GOBACK
308D 3E0A        HNDTH:  MVI     A,010D
308F 325012          STA     THSCNT
                     ;
                     ;PROGRAMS SCEDULED EVERY 10MS BELOW
                     ;
                     ;NONE
                     ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
3092 3A5112          LDA     HNDCNT
3095 3D              DCR     A
3096 CA9F30          JZ      TENTH
3099 325112          STA     HNDCNT
309C C3B930          JMP     GOBACK
309F 3E0A        TENTH:  MVI     A,010D
30A1 325112          STA     HNDCNT
                     ;
                     ;PROGRAMS SCEDULED EVERY 100MS BELOW
                     ;
                     ;NONE
                     ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
30A4 3A5212          LDA     TENCNT
30A7 3D              DCR     A
30A8 CAB130          JZ      SECOND
30AB 325212          STA     TENCNT
30AE C3B930          JMP     GOBACK
30B1 3E0A        SECOND: MVI     A,010D
30B3 325212          STA     TENCNT
                     ;
                     ;PROGRAMS SCEDULES EVERY SECOND BELOW
                     ;
30B6 CDFC30          CALL    ADDSEC
                     ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
30B9 E1          GOBACK: POP     H       ;RESTORE ENVIRONMENT
30BA D1              POP     D
30BB C1              POP     B
30BC F1              POP     PSW
30BD FB              EI
30BE C9              RET
                     ;*****************************************************
                     ;
                     ;FUNCTION: NEWTIME
                     ;DESCRIPTION: UPDATES TIME STRING ON SCREEN
                     ;
30BF 3A6012      NEWTIM: LDA     TDBYTE
30C2 B7              ORA     A
30C3 C8              RZ              ;TIME IS CURRENT IF 0
30C4 217734          LXI     H,CMDS5
```



```
30C7 CDDB30          CALL    SNDCMD
30CA 215312          LXI     H,TIMSTR
30CD CDDB30          CALL    SNDCMD
30D0 CDEF30          CALL    DOCMD   ;UPDATE TIME
30D3 AF              XRA     A
30D4 326012          STA     TDBYTE  ;TIME IS NOW CURRENT
30D7 CD9331          CALL    FLIP
30DA C9              RET
                     ;
                     ;*****************************************************
                     ;
                     ;FUNCTION: SEND COMMAND
                     ;INPUTS: H,L - POINTER TO COMMAND STRING
                     ;DESCRIPTION: WRITES MESAGE TO INPUT BUFFER
                     ;
30DB 7E          SNDCMD: MOV     A,M     ;GET CHR
30DC FEFF            CPI     0FFH    ;END?
30DE C8              RZ
30DF CD5008          CALL    WRBUF
30E2 23              INX     H
30E3 C3DB30          JMP     SNDCMD
                     ;
                     ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                     ;
30E6 CD5008      SSCMD:  CALL    WRBUF   ;SEND SINGLE LETTER COMMAND
30E9 3E0D            MVI     A,CR
30EB CD5008          CALL    WRBUF
30EE C9              RET
                     ;
                     ;*****************************************************
                     ;
                     ;THIS ROUTINE CALLS GETCMD TO PROCESS THE COMMANDS IN
                     ;THE INPUT BUFFER. WHEN THE BUFFER IS EMPTY, A RETURN
                     ;TO THIS ROUTINE WILL OCCUR.
                     ;
30EF 210000      DOCMD:  LXI     H,0H
30F2 39              DAD     SP      ;GET SP
30F3 2B              DCX     H       ;COMPENSATE SAVED STACK POINTR
30F4 2B              DCX     H       ;FOR RETURN ADR FROM GETCMD
30F5 220311          SHLD    STKSAV  ;SAVE STACK POINTER
30F8 CD5308          CALL    GETCMD  ;DO COMMANDS
30FB C9              RET             ;GETCMD RETURNS HERE IF
                                     ;FUNBYT = FF
                     ;
                     ;*****************************************************
                     ;
                     ; FUNCTION: ADD SECOND
                     ; DESCRIPTION: UPDATES TIME REGISTER BY ONE SECOND
                     ;
30FC 215D12      ADDSEC: LXI     H,TIME  ;SECONDS
30FF CD3531          CALL    BCDAD1  ;ADD 1 SEC
3102 FE60            CPI     060H    ;= 60 SEC?
3104 CA0A31          JZ      ADDMIN
3107 C33F31          JMP     GETIME
310A AF          ADDMIN: XRA     A       ;0
```

```
310B 325D12            STA     TIME
310E 215E12            LXI     H,TIME+1;MINETS
3111 CD3531            CALL    BCDAD1  ;ADD 1 MIN
3114 FE60              CPI     060H    ;=60 MIN?
3116 CA1C31            JZ      ADDHR
3119 C33F31            JMP     GETIME
311C AF          ADDHR: XRA     A
311D 325E12            STA     TIME+1
3120 215F12            LXI     H,TIME+2;HOURS
3123 CD3531            CALL    BCDAD1  ;ADD 1 HOUR
3126 FE24              CPI     024H    ;= 24 HR?
3128 CA2E31            JZ      ADDAY
312B C33F31            JMP     GETIME
312E AF          ADDAY: XRA     A
312F 325F12            STA     TIME+2
3132 C33F31            JMP     GETIME
                       ;
                       ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                       ;
3135 113E31      BCDAD1: LXI     D,ONE   ;ADD A ONE
3138 0601              MVI     B,01H   ;NUMBER OF DIG. TO ADD
313A CDC805            CALL    BCDADD
313D C9                RET
                       ;
313E 01          ONE:  DB      01H     ;CONSTANT = 1
                       ;
                       ;*****************************************************
                       ;
                       ;FUNCTION: GET TIME
                       ;DESCRIPTION: CONVERTS TIME IN REGISTER TO ASCII
                       ;             STRING AND STORES IT IN TIMSTR.
                       ;
313F 3A5F12      GETIME: LDA     TIME+2
3142 CD7E31            CALL    CONVIM
3145 325312            STA     TIMSTR
3148 79                MOV     A,C
3149 325412            STA     TIMSTR+1
314C 3E3A              MVI     A,':'
314E 325512            STA     TIMSTR+2
3151 3A5E12            LDA     TIME+1
3154 CD7E31            CALL    CONVIM
3157 325612            STA     TIMSTR+3
315A 79                MOV     A,C
315B 325712            STA     TIMSTR+4
315E 3E3A              MVI     A,':'
3160 325812            STA     TIMSTR+5
3163 3A5D12            LDA     TIME
3166 CD7E31            CALL    CONVIM
3169 325912            STA     TIMSTR+6
316C 79                MOV     A,C
316D 325A12            STA     TIMSTR+7
3170 3E0D              MVI     A,CR
3172 325B12            STA     TIMSTR+8
3175 3EFF              MVI     A,ENDCMD
3177 325C12            STA     TIMSTR+9
```



```
317A 326012            STA     TDBYTE
317D C9                RET
                       ;
                       ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                       ;
317E F5          CONVIM: PUSH    PSW     ;IN   A -  HEX BYTE
317F E6F0              ANI     0F0H    ;OUT  C - ASCII DIG 1
3181 0F                RRC             ;     A - ASCII DIG 2
3182 0F                RRC
3183 0F                RRC
3184 0F                RRC
3185 4F                MOV     C,A
3186 CD9200            CALL    PRVAL
3189 F1                POP     PSW
318A E60F              ANI     0FH
318C 41                MOV     B,C
318D 4F                MOV     C,A
318E 78                MOV     A,B
318F CD9200            CALL    PRVAL
3192 C9                RET
                       ;
3193 21A835      FLIP: LXI     H,CMDS8
3196 CDDB30            CALL    SNDCMD
3199 CDEF30            CALL    DOCMD
319C C9                RET
                       ;
                       ;*****************************************************
                       ; READ "SOFT" SWITCHES ROUTINES.
                       ;*****************************************************
                       ;
                       ;FUNCTION: READ SWITCHES
                       ;INPUTS: PARALLEL PORT 'C'
                       ;OUTPUTS: A -   BIT 0 = SW2 DOWN, BIT 1 = SW3
                       ;              BIT 2 = SW4, BIT 3 = SW5
                       ;DESCRIPTION: THIS PROGRAM READS AND DEBOUNCES THE
                       ;             FOUR 'SOFT' SWITCHES ON THE PANEL.
                       ;
319D DB23        RDSW: IN      023H    ;READ SWITCHES
319F E60F              ANI     0FH     ;MASK ALL BUT SW BITS
31A1 F5                PUSH    PSW     ;SAVE
31A2 11FF00            LXI     D,0FFH
31A5 CD9B00            CALL    DELAY   ;DEBOUNCE DELAY
31A8 DB23              IN      023H    ;READ AGAIN
31AA E60F              ANI     0FH
31AC 47                MOV     B,A
31AD F1                POP     PSW
31AE B8                CMP     B       ;SAME AS BEFORE DELAY?
31AF C8                RZ              ;YES, GOOD DATA
31B0 C39D31            JMP     RDSW    ;NO, TRY AGAIN
                       ;
                       ;*****************************************************
                       ;
                       ;FUNCTION: GET SWITCH PRESS
                       ;DESCRIPTION: WAITS FOR SWITCH TO BE DEPRESSED THEN
                       ;             RETURNS WITH VALUE IN A
```

```
                        ;
31B3 CDBF30     GETSW:  CALL    NEWTIM  ;UPDATE TIME
31B6 CD9D31             CALL    RDSW    ;READ SWITCH REGISTER
31B9 FE00               CPI     0H
31BB C0                 RNZ             ;RET IF SWITCH DEPRESSED
31BC C3B331             JMP     GETSW
                        ;
                        ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                        ;
31BF 00         FUNCT:  NOP     ;ROUTINE TO BRANCH TO CORRECT FUNCTION
                        ;               NOT IMPLEMENTED YET
                        ;
                        ;*************************************************
                        ;DEMO1 - SIMULATION OF AN ILS TYPE AIRCRAFT INSTRUMENT
                        ;*************************************************
                        ;
31C0 21C233     DEMO1:  LXI     H,CMDS2 ;HOR. BAR, AND CENTER MARK
31C3 CDDB30             CALL    SNDCMD
31C6 CDEF30             CALL    DOCMD
31C9 21D334             LXI     H,CMDS3 ;VERT BAR
31CC CDDB30             CALL    SNDCMD
31CF CD4936             CALL    CD3
31D2 CDEF30             CALL    DOCMD
31D5 210C34             LXI     H,CMDS4 ;CORDS OF BARS, TIME STRING
31D8 CDDB30             CALL    SNDCMD
31DB CDEF30             CALL    DOCMD
31DE 217C34             LXI     H,CMDS6 ;HOLLOW BOX
31E1 CDDB30             CALL    SNDCMD
31E4 CDEF30             CALL    DOCMD
31E7 210C35             LXI     H,CMDS7 ;TRIANGLES
31EA CDDB30             CALL    SNDCMD
31ED CDEF30             CALL    DOCMD
31F0 21B835             LXI     H,CMDS9 ;POINTERS
31F3 CDDB30             CALL    SNDCMD
31F6 CDEF30             CALL    DOCMD
31F9 3E37               MVI     A,037H
31FB 326112             STA     XBAR    ;INITIAL CORD OF XBAR
31FE 3E37               MVI     A,037H
3200 326212             STA     YBAR    ;INITIAL CORD OF YBAR
3203 3E08               MVI     A,08H   ;UNMASK INTERUPTS
3205 30                 DB      SIM
3206 CD0C32             CALL    MOVBAR  ;LET USER MOVE BARS
3209 CD0800             CALL    0008H   ;MONITOR
                        ;
                        ;*************************************************
                        ;
                        ;FUNCTION: MOVE BARS
                        ;DESCRIPTION: WHEN SW2 THROUGH SW5 PRESSED THE CORRECT
                        ;               ROUTINE TO MOVE A BAR IS CALLED.
                        ;
320C CDB331     MOVBAR: CALL    GETSW
320F FE0F               CPI     0FH     ;EXIT?
3211 CC6930             CZ      DOFF
3214 FE03               CPI     03H
3216 CA6A32             JZ      AUTOBAR
```



```
3219 1F                 RAR
321A F5                 PUSH    PSW
321B DC3232             CC      LEFT    ;MOVE LEFT?
321E F1                 POP     PSW
321F 1F                 RAR
3220 F5                 PUSH    PSW
3221 DC4032             CC      RIGHT   ;MOVE RIGHT?
3224 F1                 POP     PSW
3225 1F                 RAR
3226 F5                 PUSH    PSW
3227 DC4E32             CC      UP      ;MOVE UP?
322A F1                 POP     PSW
322B 1F                 RAR
322C DC5C32             CC      DOWN    ;MOVE DOWN?
322F C30C32             JMP     MOVBAR
3232 3A6112     LEFT:   LDA     XBAR
3235 FE10               CPI     010H
3237 C8                 RZ
3238 3D                 DCR     A
3239 326112             STA     XBAR
323C CDA632             CALL    SNDXBAR
323F C9                 RET
3240 3A6112     RIGHT:  LDA     XBAR
3243 FE60               CPI     060H
3245 C8                 RZ
3246 3C                 INR     A
3247 326112             STA     XBAR
324A CDA632             CALL    SNDXBAR
324D C9                 RET
324E 3A6212     UP:     LDA     YBAR
3251 FE10               CPI     010H
3253 C8                 RZ
3254 3D                 DCR     A
3255 326212             STA     YBAR
3258 CDC632             CALL    SNDYBAR
325B C9                 RET
325C 3A6212     DOWN:   LDA     YBAR
325F FE60               CPI     060H
3261 C8                 RZ
3262 3C                 INR     A
3263 326212             STA     YBAR
3266 CDC632             CALL    SNDYBAR
3269 C9                 RET
                        ;
                        ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                        ;
326A 0610       AUTOBAR:MVI     B,010H
326C C5         AB1:    PUSH    B
326D CD4E32             CALL    UP
3270 CD3232             CALL    LEFT
3273 C1                 POP     B
3274 05                 DCR     B
3275 C26C32             JNZ     AB1
3278 0610               MVI     B,010H
327A C5         AB2:    PUSH    B
```

```
327B CD4032            CALL    RIGHT
327E C1                POP     B
327F 05                DCR     B
3280 C27A32            JNZ     AB2
3283 0610              MVI     B,010H
3285 C5        AB3:    PUSH    B
3286 CD5C32            CALL    DOWN
3289 CD3232            CALL    LEFT
328C C1                POP     B
328D 05                DCR     B
328E C28532            JNZ     AB3
3291 0610              MVI     B,010H
3293 C5        AB4:    PUSH    B
3294 CD4032            CALL    RIGHT
3297 C1                POP     B
3298 05                DCR     B
3299 C29332            JNZ     AB4
329C CD9D31            CALL    RDSW
329F B7                ORA     A
32A0 C20C32            JNZ     MOVBAR
32A3 C36A32            JMP     AUTOBAR
                ;
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
32A6 21F032    SNDXBAR:LXI     H,BCMD2 ;SEND NEW COORDS OF XBAR
32A9 CDDB30            CALL    SNDCMD
32AC 3A6112            LDA     XBAR
32AF CDDD32            CALL    SNDBYT
32B2 21F532            LXI     H,BCMD3
32B5 CDDB30            CALL    SNDCMD
32B8 CDEF30            CALL    DOCMD
32BB 210070    DXBAR:  LXI     H,07000H;BAR MOVE DELAY
32BE EB                XCHG
32BF CD9B00            CALL    DELAY
32C2 CDBF30            CALL    NEWTIM
32C5 C9                RET
                ;
                ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ;
32C6 21E832    SNDYBAR:LXI     H,BCMD1 ;SEND NEW COORDS OF YBAR
32C9 CDDB30            CALL    SNDCMD
32CC 3A6212            LDA     YBAR
32CF CDDD32            CALL    SNDBYT
32D2 3E0D              MVI     A,CR
32D4 CD5008            CALL    WRBUF
32D7 CDEF30            CALL    DOCMD
32DA C3BB32            JMP     DXBAR
                ;
32DD CD7E31    SNDBYT: CALL    CONVTM
32E0 CD5008            CALL    WRBUF
32E3 79                MOV     A,C
32E4 CD5008            CALL    WRBUF
32E7 C9                RET
                ;
32E8 4D30312C31BCMD1:  DB      'M01,10,'
```



```
32EF FF                DB      ENDCMD
32F0 4D30322C  BCMD2:  DB      'M02,'
32F4 FF                DB      ENDCMD
32F5 2C3130    BCMD3:  DB      ',10'
32F8 0D                DB      CR
32F9 FF                DB      ENDCMD
                ;
                ;******************************************************
                ;COMMAND STRINGS
                ;******************************************************
                ;
32FA 4B        CMDS1:  DB      'K'
32FB 0D                DB      CR
32FC 4130312F41        DB      'A01/AIRCRAFT LCD DISPLAY'
3314 0D                DB      CR
3315 4130322F53        DB      'A02/SELECT MODE'
3324 0D                DB      CR
3325 4130312C33        DB      'A01,30,00'
332E 0D                DB      CR
332F 4130322C39        DB      'A02,90,45'
3338 0D                DB      CR
3339 4130352F53        DB      'A05/S.B. THESIS, LEE MARZKE, 5/84'
335A 0D                DB      CR
335B 4130352C31        DB      'A05,10,20'
3364 0D                DB      CR
3365 4130332F44        DB      'A03/DEMO'
336D 0D                DB      CR
336E 4130332C37        DB      'A03,7A,5F'
3377 0D                DB      CR
3378 4130342F44        DB      'A04/DISPLAY'
3383 0D                DB      CR
3384 4130342C39        DB      'A04,95,6A'
338D 0D                DB      CR
338E 4430312C30        DB      'D01,0B011818181818181899DB7E18'
33AC 0D                DB      CR      ;DOWN ARROW SYMBOL
33AD 5030312C38        DB      'P01,85,75'
33B6 0D                DB      CR
33B7 5030312C41        DB      'P01,A6,75'
33C0 0D                DB      CR
33C1 FF                DB      ENDCMD
                ;
33C2 49        CMDS2:  DB      'I'
33C3 0D                DB      CR      ;HOROZONTAL BAR
33C4 4430312C30        DB      'D01,010AFFFFFFFFFFFFFFFFFF'
33E0 0D                DB      CR
33E1 4430352C30        DB      'D05,07013E7F7F7F7F7F3E'
33F7 0D                DB      CR      ;CENTER MARK
33F8 5030352C33        DB      'P05,33,34'
3401 0D                DB      CR
3402 FF                DB      ENDCMD
                ;
3403 4430322C35CMDS3:  DB      'D02,5001';VERTICAL BAR
340B FF                DB      ENDCMD
                ;
                ;
```

```
340C 5830312C31CMDS4:  DB      'X01,10,37';CORDS OF BARS
3415 0D                DB      CR
3416 5830322C33        DB      'X02,37,10'
341F 0D                DB      CR
3420 4E                DB      'N'
3421 0D                DB      CR
3422 4130312F54        DB      'A01/TIMESTRI';TIME STRING
342E 0D                DB      CR
342F 4130312C30        DB      'A01,00,79';LOCATION
3438 0D                DB      CR
3439 4C31303130        DB      'L1010,6010,1060,6060,'
344E 323031302C        DB      '2010,5010,2060,5060,'
3462 313032302C        DB      '1020,6020,1050,6050'
3475 0D                DB      CR
3476 FF                DB      ENDCMD
                ;
3477 4130312F  CMDS5:  DB      'A01/'
347B FF                DB      ENDCMD
347C 4430332C30CMDS6:  DB      'D03,0D03FFFFFFFFFFFF'
3490 4330303030        DB      'C00003800001800001'
34A2 3830303030        DB      '800001800001800001'
34B4 3830303030        DB      '800001800001'
34C0 4330303030        DB      'C00003FFFFFFFFFFFF'
34D2 0D                DB      CR      ;HOLLOW BOX
34D3 5030332C32        DB      'P03,2B,00'
34DC 0D                DB      CR
34DD 5030332C32        DB      'P03,2B,E4'
34E6 0D                DB      CR
34E7 4130322F31        DB      'A02/120'
34EE 0D                DB      CR
34EF 4130332F33        DB      'A03/300'
34F6 0D                DB      CR
34F7 4130322C32        DB      'A02,2E,03'
3500 0D                DB      CR
3501 4130332C32        DB      'A03,2E,67'
350A 0D                DB      CR
350B FF                DB      ENDCMD
                ;
350C 4430342C31CMDS7:  DB      'D04,100204000E000E001F00'
3524 3146303033        DB      '1F003F807FC000000000F9C0'
353C 4141323032        DB      'AA20222022202220222021C0'
3554 0D                DB      CR      ;UP TRIANGLE 'TO'
3555 4430362C31        DB      'D06,1002FBC082208220F3C0'
356D 3832383038        DB      '828082408220000000007FC0'
3585 3346383031        DB      '3F801F001F000E000E000400'
359D 0D                DB      CR      ;DOWN TRIANGLE 'FR'
359E 5030342C36        DB      'P04,61,26'
35A7 0D                DB      CR
35A8 5830342F  CMDS8:  DB      'X04/';FLIP TO/FR
35AC 0D                DB      CR
35AD 584F362C36        DB      'X06,61,39'
35B6 0D                DB      CR
35B7 FF                DB      ENDCMD
                ;
35B8 4430372C30CMDS9:  DB      'D07,040110284482';POINTER,UP
```



```
35C8 0D                DB      CR
35C9 4430382C30        DB      'D08,040182442810';DOWN
35D9 0D                DB      CR
35DA 4430392C30        DB      'D09,070120100804081020';RIGHT
35F0 0D                DB      CR
35F1 4430412C30        DB      'D0A,070104081020100804';LEFT
3607 0D                DB      CR
3608 5030412C38        DB      'P0A,85,78'
3611 0D                DB      CR
3612 5030392C41        DB      'P09,A6,78'
361B 0D                DB      CR
361C 5030372C43        DB      'P07,C7,7A'
3625 0D                DB      CR
3626 5030382C45        DB      'P08,E8,7A'
362F 0D                DB      CR
3630 4130342F4D        DB      'A04/MOVE BARS'
363D 0D                DB      CR
363E 4130342C41        DB      'A04,A6,6E'
3647 0D                DB      CR
3648 FF                DB      ENDCMD
                ;
                ;*****************************************************
                ;COMMAND STRING SUBROUTINES
                ;*****************************************************
                ;
3649 0650       CD3:   MVI     B,80D   ;VERTICAL BAR
364B 3E38       C3:    MVI     A,'8'
364D CD5008            CALL    WRBUF
3650 3E30              MVI     A,'0'
3652 CD5008            CALL    WRBUF
3655 05                DCR     B
3656 C24B36            JNZ     C3
3659 3E0D              MVI     A,CR
365B CD5008            CALL    WRBUF
365E C9                RET
                ;
1102            ORG     01102H
1102            FUNBYT: DS      01H     ;FUNCTION RUNNING BYTE
1103            STKSAV: DS      02H     ;STACK POINTER SAVE LOC.
                ;
                ;*****************************************************
                ;VARIABLES
                ;*****************************************************
1200            ORG     01200H
1200                    DS      050H    ;STACK BELOW
1250            FSTACK: DS      0H
1250            THSCNT: DS      01H
1251            HNDCNT: DS      01H
1252            TENCNT: DS      01H
1253            TIMSTR: DS      0AH     ;ASCII TIME, CR, ENDCMD
125D            TIME:   DS      03H     ;TIME IN HEX (S,M,H)
1260            TDBYTE: DS      01H     ;=FF, TIME NEEDS UPDATING
1261            XBAR:   DS      01H     ;X COORDINATE OF VERT BAR
1262            YBAR:   DS      01H     ;Y COORDINATE OF HOR BAR
1263            BARDLY: DS      02H     ;BAR MOVE DELAY VALUE
1265            END
```