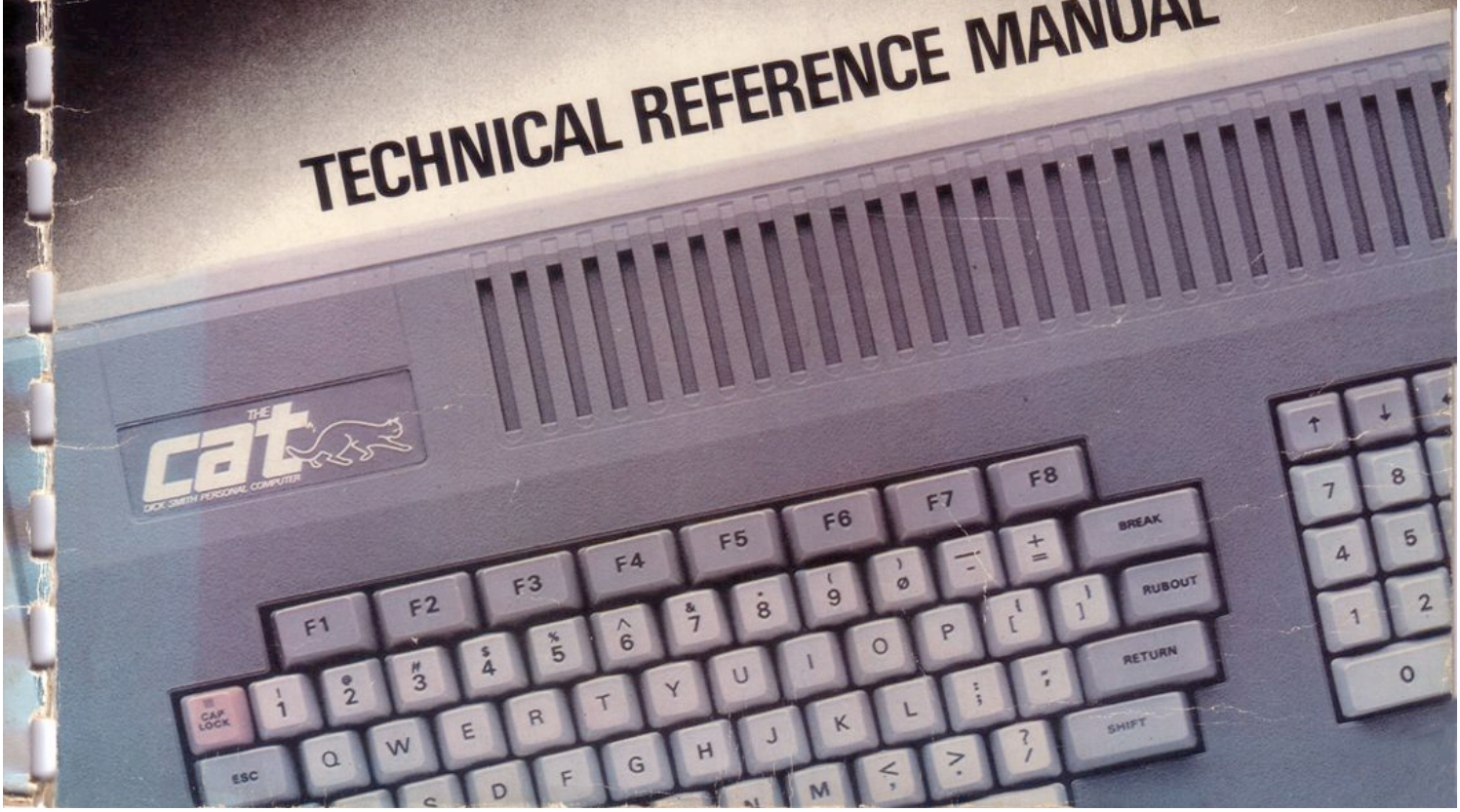
THE cat™

DICK SMITH PERSONAL COMPUTER

TECHNICAL REFERENCE MANUAL



FIRST EDITION – 1984

PREFACE

This technical manual has been written to provide the maximum amount of useful data to those people interested in the internal workings of the computer.

For the computer programmers, this book contains a number of invaluable machine language subroutines that will make software development somewhat easier than expected. It also contains an entire section on memory-mapping procedures and memory locations in BASIC.

We have also included as much information as possible on the system board, the busses, built-in interfaces and power supply. Diagrams, circuits, charts and specifications give a basis for servicing – in the unlikely event of component failure.

The technical engineering data included in this manual gives a full set of parameters for trouble – shooting and repair work. Unless your computers has suffered massive mechanical damage, you should find enough information here to restore any of the features and functions to full operational status.

CONTENTS

CHAPTER 1

THE SYSTEM OVERVIEW

- The system overview
- The board
- On-board peripherals
  (Z-80/Centronics Parallel/Disk drives/RS232)
- Off-board peripherals
  (Everything else.)

THE SYSTEM BOARD OVERVIEW

The computer is built around a 6502A microprocessor chip. The PCB (Printed Circuit Board) which surrounds this chip also has a number of other chips incorporated into it. These include the 8048 which controls the keyboard. Another runs the programmable sound generator. Still others contain the RAM (Random Access Memory), the ROM (Read-Only-Memory) and the unique video and input/output processors.

ON-BOARD PERIPHERALS

The system board offers a number of busses and sockets to which you can attach peripheral equipment. Four types of peripherals can be attached directly onto the system board.

1) Z80 Cartridge (catalogue No X 7570)
2) Centronics parallel printer (catalogue No X 3250 X 3268)
3) Disk Controller (catalogue No X 7510)
4) RS232 Serial Adaptor (catalogue No X 7515)

THE Z80 CARTRIDGE
(catalogue No X 7570)

Containing the Z80A microprocessor chip and interfacing logic, the Z80 cartridge allows you to run CP/M-80®software. This includes a wide range of business applications and a number of personal programs. With the Z80 cartridge, your computer suddenly has a whole library of proven software, ready to run.

HOOKING UP THE Z80 CARTRIDGE (catalogue No X 7570)

Follow these simple steps:

1) Make sure all power is OFF

2) Plug the Z80 cartridge directly into the socket located on the right hand side of your computer.

3) Turn on the power

4) Insert CP/M-80® software into disk drive

5) Load CP/M® software.

CENTRONICS PRINTER

To connect any Centronics printer you must have a CAT Printer Connection cable (catalogue No X 7540). The interface to control the printer is built into your computer, but the connecting cable has special sockets and is easier to buy than to try and make up yourself.

THE BEST PRINTERS TO USE

Although you can use any Centronics - type printer with your computer, we have developed special printers that precisely match the computer and take advantage of all it's high power and special features. These are the printers described below.

THE GRAPHICS PRINTER (BX-80) — Catalogue No X 3268

A dot matrix printer with a speed of 100 characters per second, the Graphics printer gives you a great price/performance package.

It will print graphics. It will print characters in close to letter quality. It allows you to print either 40 or 80 characters per line.

THE 4-COLOUR PRINTER-PLOTTER (Catalogue No X 3248)

This machine offers another great value. It prints both text and graphics. It can do plotting in 3-colours and, at 4 1/2 inches, is very compact and convenient to use.

HOW TO HOOK THEM UP:

1) Turn off all the power to the computer and printer

2) Attach the printer cable to the PRINTER socket on the rear panel of the computer.

3) Attach the other end of the cable to the appropriate socket on the printer.

4) Turn everything back on.

5) Go nuts.

DISK CONTROLLER

(Catalogue No X7510)

Before you can add on a floppy disk drive, you must have a disk controller. It plugs directly onto the board through a socket on the back of the computer.

Once the disk controller unit is in place, it will support one or two 5-1/4 inch floppy disk drives. Each drive will store up to 160Kb of data per disk.

To connect the disk drives to the disk controller, simply plug the flat wires into the sockets on the controller. Note: If only using one drive unit plug into socket marked Drive 1.

THE RS232 SERIAL ADAPTOR

(Catalogue No X 7515)

The interface unit, called a CAT RS232 Serial Adaptor, plug into the RS232 socket in the back of your CAT.

Once it is in place, it can communicate to other computers through a direct cable or over the telephone through a direct connect modem. The Dick Smith Dataphone II (catalogue X 3272) is suitable for this purpose.

The data transmission speeds range from 110 to 9600 bauds and are selected by using the mini switch on the bottom of the RS232 Serial Adaptor.

The Dataphone II operates at 300 baud.

THE OFF-BOARD PERIPHERALS

These include everything not mentioned above. Such as . . . .

GAME PADDLE (catalogue No X 7520)

Two Joysticks are linked together by a 9-pin "D" type connector cable and plug in to the right-side of your computer console. Each Joystick consists of a 4-direction variable resistor and two contact switches as fire buttons.

DATA CASSETTE (catalogue No X 7206)

For low cost storage, you may choose the data cassette or just an ordinary cassette. However, the data cassette narrows down the noise bandwidth and makes parogram loading and saving more reliable.

EMULATOR CARTRIDGE (catalogue No X 7530)

The Emulator Cartridge built in 16K Bytes of RAM, and electronically located at the same address as your built-in BASIC ROM interpreter.

Application Programs and data can be loaded into these RAMs which is later read enabled and control is automatically passed from BASIC to these programs. Addition of this soft emulator thus allows more user memory.

CHAPTER 2

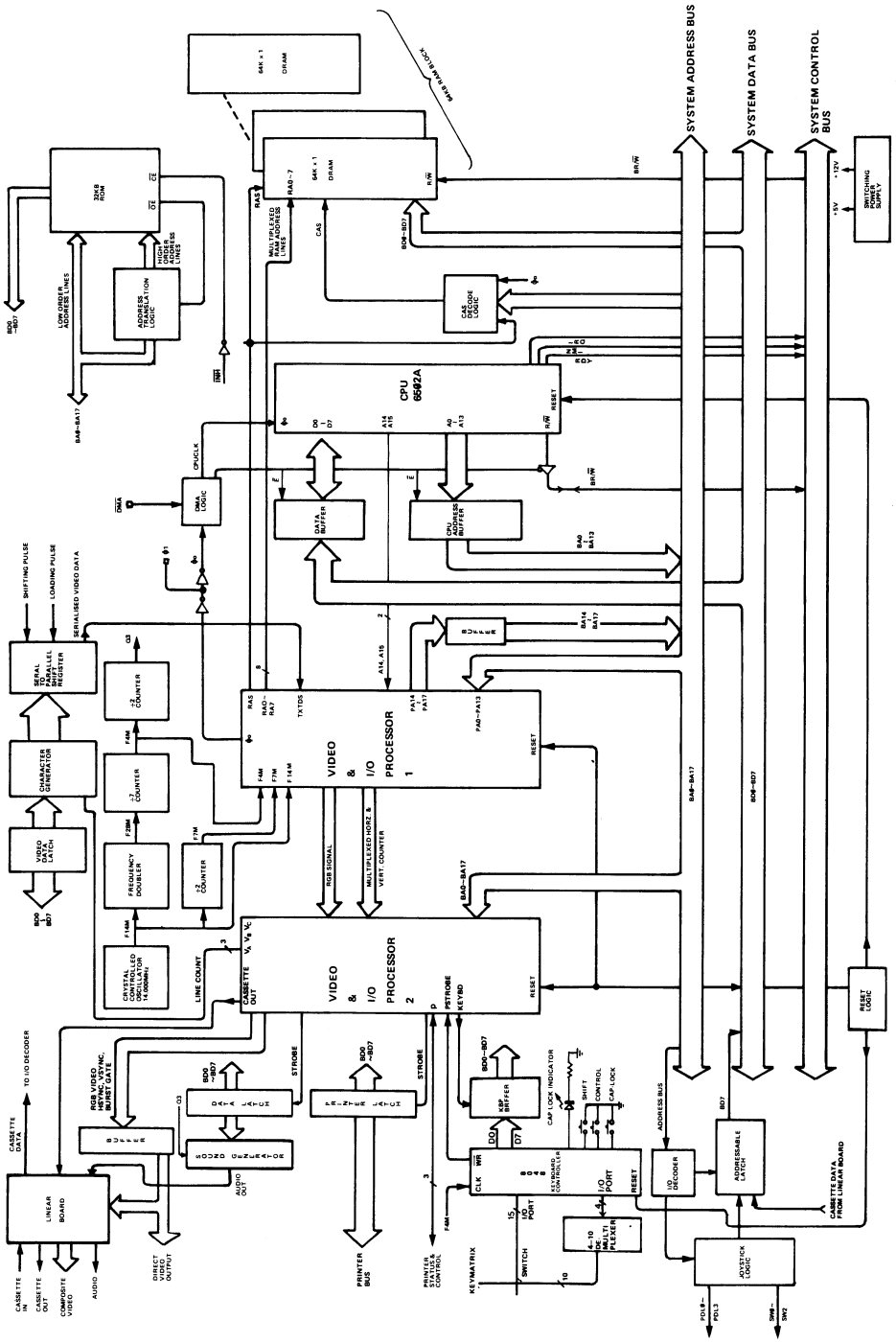
SYSTEM BOARD AND SUBSYSTEM

- The system board
- The processor subsystem
- The Read Only Memory (ROM) subsystem
- The Random Access Memory (RAM) subsystem
- The Master Timing Generator
- The Keyboard Controller subsystem
- The keys and their codes
- The Integrated Video subsystem
- Gate Array pin out descriptions
- Software switches
- Internal I/O
- Screen mapping details

SYSTEM BOARD

The major elements of the System Board are divided into five functional areas. They are, the processor subsystem and its support elements, the Read-Only-Memory subsystem, the Random-Access-Memory subsystem, Integrated video and I/O subsystem, Master timing generator and keyboard controller subsystem. Diagram 2.1 shows the system board block diagram.

SYSTEM DIAGRAM



THE PROCESSOR SUBSYSTEM

The heart of the system board is the 8-bit 6502A microprocessor. It features 16 bit addressing (64K Bytes of storage), memory mapped I/O and operates at 1 MHz or 2 MHz. The clock frequency is derived from a timing circuit inside the video processor. The clock frequency varies with the display mode. Diagram 2.2 below gives the clock timing and display switch relationships.

SWR1	SWR0	Display Mode
0	0	40 column text or graphics
0	1	bit image graphics
1	0	double resolution graphics
1	1	80 column text

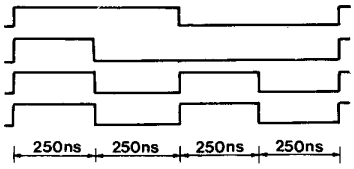


Diagram 2.2 CPU clock patterns

Address buffers 74LS244 and Data Tranceivers 74LS245 are added to enchance CPU driving characteristics. Figure 2.3 shows the block diagram for the processor subsystem.

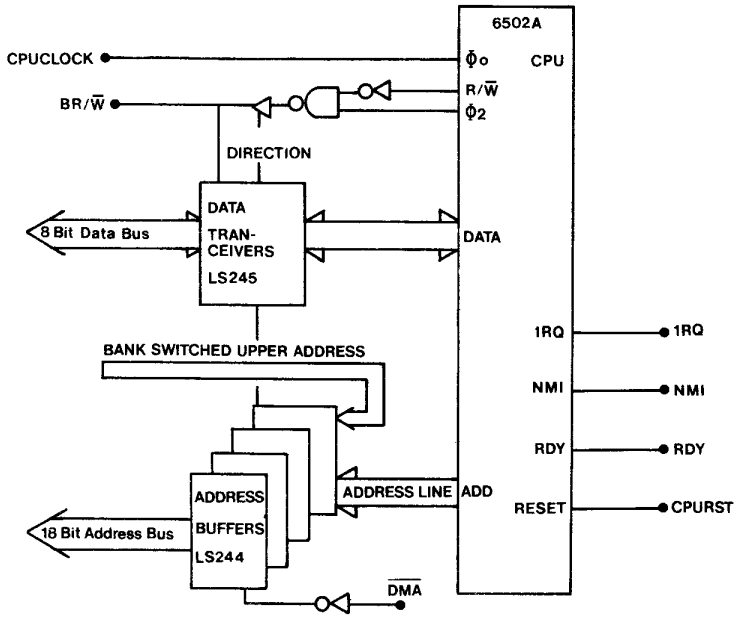


Figure 2.3 block diagram of processor subsystem

System Data Bus is partitioned into two phases Φo and Φ1. In Φo phase, data bus holds read/write data for CPU whereas in Φ1 phase, read data appears for the video processors. Figure 2.4 shows the timing.

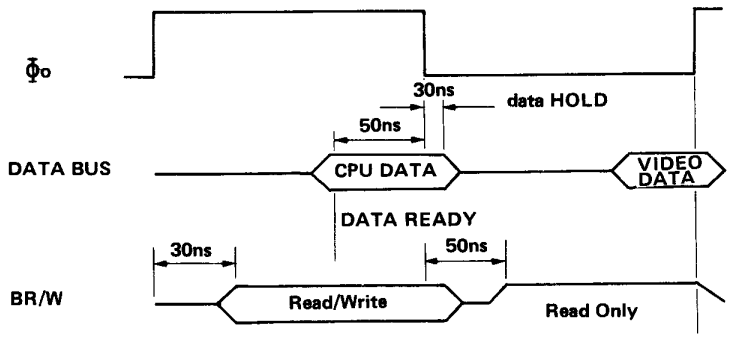


Figure 2.4 data bus timing

Timing for Address lines and interrupt or ready lines are the same as those for 6502A specifications in Rockwell or Synertek data sheets.

READ-ONLY-MEMORY (ROM) SUBSYSTEM

The ROM subsystem consists of the 32K Byte ROM, which holds the enchanced Microsoft BASIC and System Kernel, and the decoding logic.

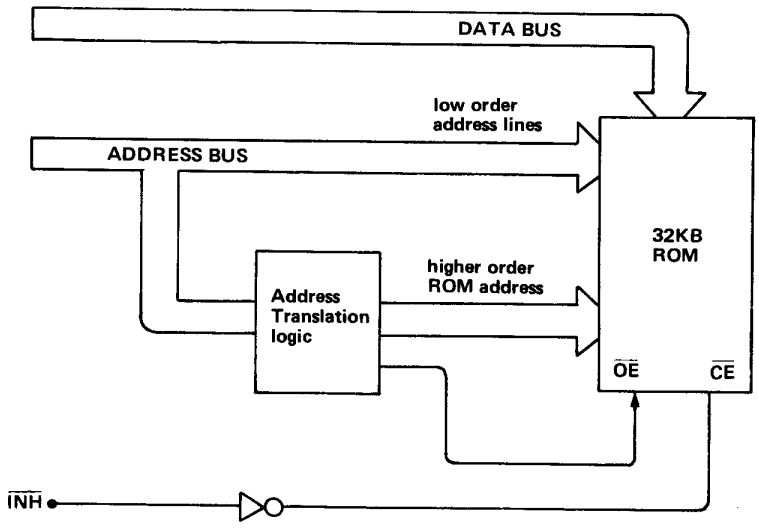


Figure 2.5 ROM subsystem block diagram.

The 32KB ROM is a high speed 250ns access device. Early products consists of four 8KB device further divided by a LS138 and functions exactly the same as the 32KB ROM. For details on ROM specification, please refer to the semi-conductor memory data sheets.

RAMDOM-ACCESS-MEMORY RAM SUBSYSTEM

The RAM subsystem includes eight 64K bit X 1 dynamic RAMs, RAM address buffers and RAM block decode logic. RAM address multiplexing is done inside the video and I/O controllers, Figure 2.6 shows the RAM subsystem block diagram.

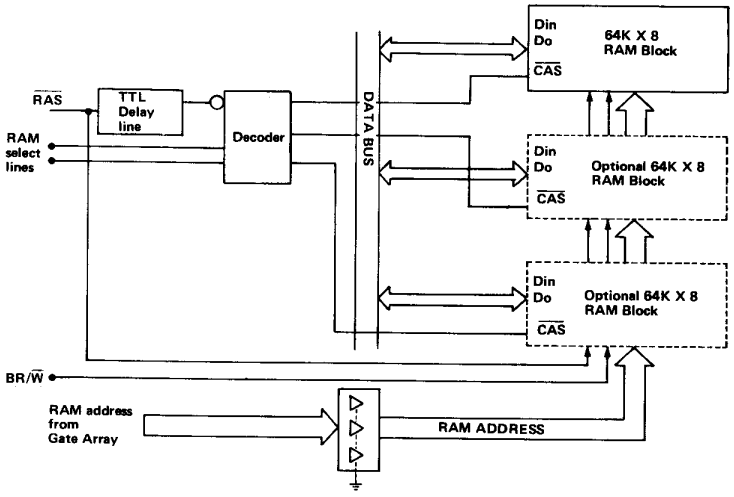


Figure 2.6 RAM subsystem block diagram.

RAM with access time of 150ns are used. Cycle timing of RAM must be less than 270ns. RAM data will be valid 50ns before each CPU clock edges. For details of RAM specification, please refer to 4164 data sheets from semi-conductor manufacturers. Figure 2.7 gives some important RAM timings:

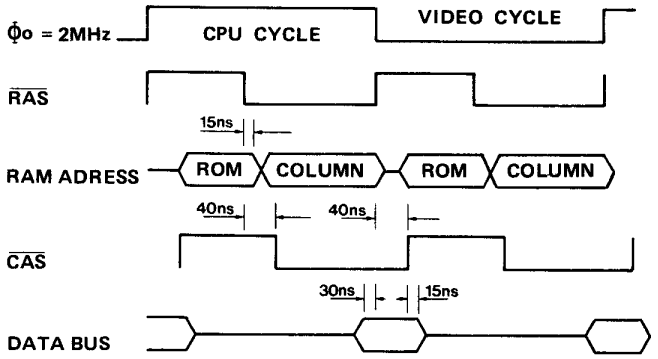


Figure 2.7 dynamic RAM timing

MASTER TIMING GENERATOR

All system timing are referenced to a 14.000MHz crystal controlled timing generator which provides three Basic timings: F4M, F14M and F28M as shown in figure 2.7 below. Notice that the F4M has a 3:4 duty cycle.

F4M provides timing reference for the CPU clock and all RAM/ROM and CPU related logic whereas F14M and its intermediate frequency F7M act as the video dot clock. F28M is used internally to generate F4M and externally for fine timing adjustment.

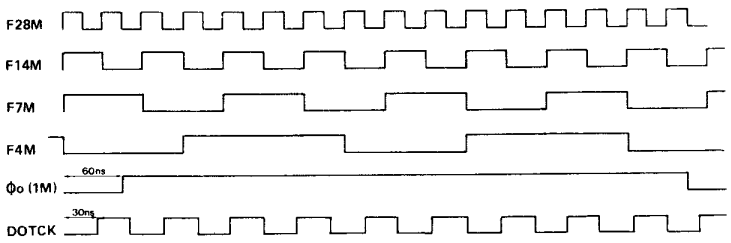


Figure 2.8 master clock timings

KEYBOARD CONTROLLER SUBSYSTEM

The keyboard controller consists of a single chip 8-bit microcomputer (8048) which is responsible for all keyboard scanning, encoding and debouncing. Upon each key closure, a flip-flop will be set by the 8048. To detect a new key depression, the 65Ø2A strobes the keyboard encoding by issuing the keyboard address and reading back the data. The above mentioned key flag is the MSB of keyboard data. The rest 7-bits are the keycodes. The uP then issues another address to clear the keyflag.

Hex address	Function
3C000	Read data
3C010	Clear keyboard flag

Figure 2.9 below shows the block diagram for the keyboard controller section and Figure 2.10 shows the associated key codes.

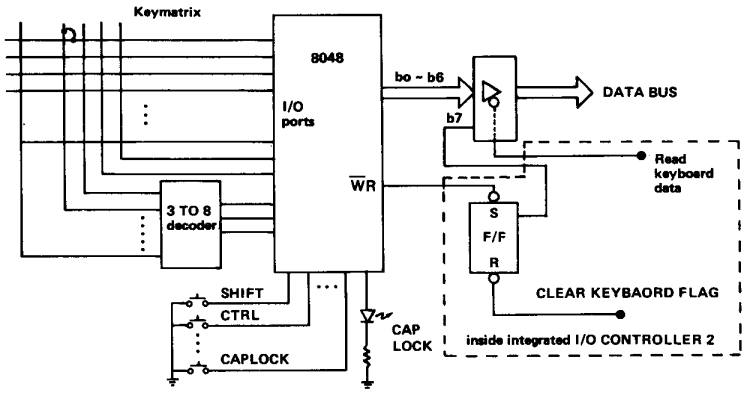


Figure 2.9 keyboard controller block diagram

FIGURE 2.10
KEYS AND THEIR ASSOCIATED CODES

KEY	CTRL	CTRL & SHIFT	SHIFT	CAP. ONLY	LOWER CASE ONLY
SPACE	20	20	20	20	20
0)	30	29	29	30	30
1 !	31	21	21	31	31
2 @	32	00	4Ø	32	32
3 #	33	23	23	33	33
4 $	34	24	24	34	34
5 %	35	25	25	35	35
6 ^	36	1E	5E	36	36
7 &	37	26	26	37	37
8 *	38	2A	2A	38	38
9 (	39	28	28	39	39
- _	2D	5F	5F	2D	2D
= +	3D	2B	2B	3D	3D
[{	5B	7B	7B	5B	5B
] }	5D	7D	7D	5D	5D
; :	3B	3A	3A	3B	3B
' "	27	22	22	27	27
, <	2C	3C	3C	2C	2C
. >	2E	3E	3E	2E	2E
/ ?	2F	3F	3F	2F	2F
A	01	01	41	41	61
B	02	02	42	42	62
C	03	03	43	43	63
D	04	04	44	44	64
E	05	05	45	45	65
F	06	06	46	46	66
G	07	07	47	47	67
H	08	08	48	48	68

KEY	CTRL	CTRL & SHIFT	SHIFT	CAP.ONLY	LOWER CASE ONLY
I	09	09	49	49	69
J	0A	0A	4A	4A	6A
K	0B	0B	4B	4B	6B
L	0C	0C	4C	4C	6C
M	0D	0D	5D	4D	6D
N	0E	0E	4E	4E	6E
O	0F	0F	4F	4F	6F
P	10	10	50	50	70
Q	11	11	51	51	71
R	12	12	52	52	72
S	13	13	53	53	73
T	14	14	54	54	74
U	15	15	55	55	75
V	16	16	56	56	76
W	17	17	57	57	77
X	18	18	58	58	78
Y	19	19	59	59	79
Z	1A	1A	5A	5A	7A
↑	1B 44	1B 44	1B 44	1B 44	1B 44
↓	0A	0A	0A	0A	0A
←	08	08	08	08	08
→	15	15	15	15	15

KEY	CTRL	CTRL & SHIFT	SHIFT	CAP. ONLY	LOWER CASE ONLY
0	30	30	30	30	30
1	31	31	31	31	31
2	32	32	32	32	32
3	33	33	33	33	33
4	34	34	34	34	34
5	35	35	35	35	35
6	36	36	36	36	36
7	37	37	37	37	37
8	38	38	38	38	38
9	39	39	39	39	39
+	2B	2B	2B	2B	2B
–	2D	2D	2D	2D	2D
.	2E	3E	3E	2E	2E
RETURN	0D	0D	0D	0D	0D
ESC	1B	1B	1B	1B	1B
TAB	1C	1C	1C	1C	1C
BREAK	7F	7F	7F	7F	7F
RUBOUT	08 20 08	08 20 08	08 20 08	08 20 08	08 20 08
F1	1B 32 30	1B 32 30	1B 31 30	1B 30 30	1B 30 30
F2	1B 32 31	1B 32 31	1B 31 31	1B 30 31	1B 30 31
F3	1B 32 32	1B 32 32	1B 31 32	1B 30 32	1B 30 32
F4	1B 32 33	1B 32 33	1B 31 33	1B 30 33	1B 30 33
F5	1B 32 34	1B 32 34	1B 31 34	1B 30 34	1B 30 34
F6	1B 32 35	1B 32 35	1B 31 35	1B 30 35	1B 30 35
F7	1B 32 36	1B 32 36	1B 31 36	1B 30 36	1B 30 36
F8	1B 32 37	1B 32 37	1B 31 37	1B 30 37	1B 30 37

INTEGRATE VIDEO AND I/O SUBSYSTEM

The integrated video and I/O subsystem consists of 2 custom designed gate arrays which are responsible for all video memory mapping and control, memory windowing, system synchronization and certain I/O decoding. Additional external circuits furnish the rest of the I/O system, namely, programmable sound generator, printer latch and joystick control. Figure 2.11 below shows the block diagram of the integrated video and I/O subsystem.

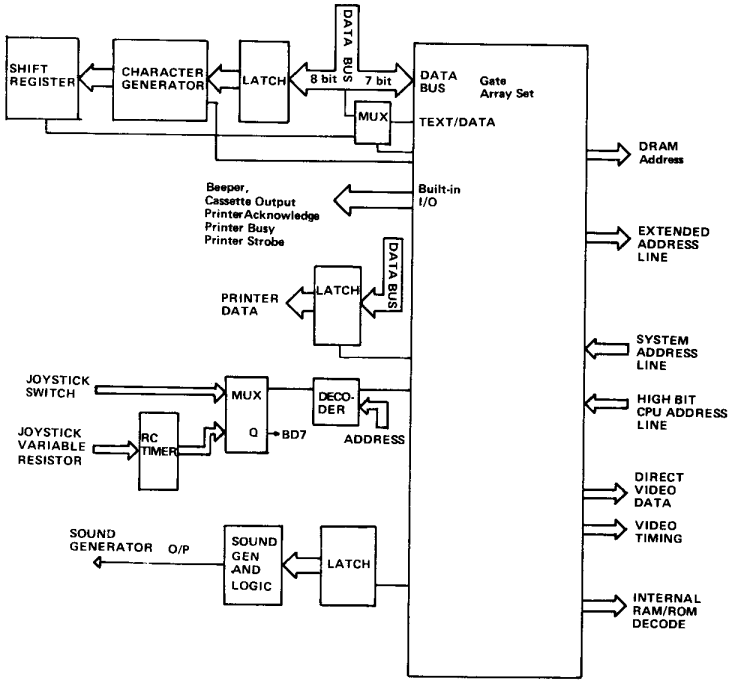


Figure 2.11 block diagram of integrate video and I/O subsystem

Figure 2.12 and 2.13 list the pinout and describe the functions of each gate array pins.

Figure 2.14 summarize all I/O address and Figure 2.15 summarize all video screen memory mapping.

FIGURE 2.12 GATE ARRAY 1 PINOUT

PIN NO.	PIN NAME	DESCRIPTION
1	VSS	GROUND
2	F14M	14MHz CLOCK INPUT
3	F7M	7MHz CLOCK INPUT, SYNCHRONOUS TO RISING EDGE OF F14M
4	F4M	4MHz CLOCK INPUT, 3 LOW: 4 HIGH DUTY CYCLE
5	LOADS	NEGATIVE PULSE INPUT, MUST OCCUR LESS THAN 250NS AFTER CPUCK
6	DOTCK	VIDEO DOT CLOCK OUTPUT
7	RGUN	RED VIDEO O/P
8	GGUN	GREEN VIDEO O/P
9	BGUN	BLUE VIDEO O/P
10	$\overline{RAS}$	DRAM ROW ADDRESS STROBE
11	HOVA	MULTIPLEXED HORIZONTAL AND VERTICAL CTR O/P, $\Phi o=\emptyset$ O/P=H0
12	H1VB	" O/P=H1
13	H2VC	" O/P=H2
14	H3V0	" O/P=H3
15	H4V1	" O/P=H4
16	H5V2	" O/P=H5
17	H6V3	" O/P=H6
18	V_4 V_5	MULTIPLEXED VERTICAL CTRS O/P, $\Phi o=0$, O/P=V4
19	VDD	+5V INPUT

PIN NO.	PIN NAME	DESCRIPTION
20	RAM0	DRAM MULTIPLEXED ADDRESS OUTPUT, LSB
21	RAM1	"
22	RAM2	"
23	RAM3	"
24	RAM4	"
25	RAM5	"
26	RAM6	"
27	RAM7	" , MSB
28	SWR0	DISPLAY MODE SWITCH O/P
29	SWR1	"
30	SWT	TEXT/$\overline{\text{GRAPHICS}}$ DISPLAY MODE SWITCH O/P
31	SWP1	SWP1=0— PRIMARY; SWP1=1 SECONDARY DISPLAY PAGE
32	VSS	GROUND
33	$\overline{\text{RESET}}$	CHIP RESET, ACTIVE LOW
34	A0	ADDRESS LINE INPUT, LSB
35	A1	"
36	A2	"
37	A3	"
38	A4	"
39	A5	"
40	A6	"
41	A7	"
42	A8	"

*ALL I/0 ARE TTL LEVEL, I_{OH} = 400μA at VoH = 2.7V
I_{OL} = 4mA at Vol = 0.4V
I_{IN} = +/— 10μA

PIN NO.	PIN NAME	DESCRIPTION
42	A9	"
44	A10	"
45	A11	"
46	A12	"
47	A13	"
48	A14	"
49	A15	ADDRESS LINE INPUT , MSB
50	BA14	WINDOW ADDRESS LINE OUTPUT
51	BA15	"
52	BA16	"
53	BA17	"
54	FSEL	50/$\overline{60}$Hz VIDEO FRAME SELECT
55	BD0	DATA BUS INPUT, LSB
56	BD1	
57	BD2	
58	BD3	
59	BD4	
60	TXTD5	TEXT/DATA BUS BD5 INPUT, SWT=1, TEXT; SWT=0, DATA INPUT
61	BD6	DATA BUS INPUT
62	BD7	" , MSB
63	CPUCK	CPU CLOCK INPUT 1/2 MHz WHEN SWR1=0/1
64	VDD	+5V SUPPLY

FIGURE 2.13
GATE ARRAY 2 PINOUT

PIN NO.	PIN NAME	DESCRIPTION
1	VSS	GROUND
2	$\overline{\text{ROUT}}$	RED VIDEO OUTPUT, ACTIVE LOW TTL
3	$\overline{\text{GOUT}}$	GREEN "
4	$\overline{\text{BOUT}}$	BLUE "
5	HSYNC	HORIZONAL SYNC, ACTIVE HIGH TTL
6	VSYNC	VERTICAL SYNC, "
7	CBLANK	COMPOSITE BLANKING, "
8	BGATE	BURST GATE, "
9	BD7	DATA BUS, MSB
10	$\overline{\text{IOEN}}$	I/O ENABLE, ACTIVE LOW TTL, $3C000 – 3C07F
11	H0VA	Φ_0 MULTIPLEXED HORI-ZONTAL/VERTICAL CTR INPUT
12	H1VB	"
13	H2VC	"
14	H3V0	"
15	H4V1	"
16	H5V2	"
17	H6V3	"
18	V5V4	Φ_0 MULTIPLEXED VERTICAL/ VERTICAL CTR INPUT
19	VDD	+5V SUPPLY
20	KEYBD	READ KEYBOARD, $3C000

*ALL I/0 ARE TTL LEVEL, I_{OH} = –400μA at V_{OH} = 2.7V
I_{OL} = 4mA at V_{OL} = 0.4V
I_{IN} =+/–10uA

PIN NO.	PIN NAME	DESCRIPTION
21	KEYSTROBE	KEYBOARD STROBE INPUT, SETS, INTERNAL NEW KEY Flip – flop
22	IOSTR	I/O STROBE OUTPUT, $3CFFF
23	$\overline{\text{RAMOE}}$	RAM LATCH OUTPUT ENABLE, N.C. IN THE COMPUTER
24	TTLS	1 BIT TTL OUTPUT, ACCESSS BY $3C030
25	CASOUT	CASSETTE OUTPUT, TTL LEVEL
26	$\overline{\text{PIOSEL}}$	PRE-I/O SELECT, $3C000 – $3CFFF
27	$\overline{\text{PRACK}}$	PRINTER ACKNOWLEDGE
28	PRBUSY	PRINTER BUSY
29	PRSTROBE	PRINTER STROBE
30	PRLATCH	LATCH DATA BUS INTO PRINTER BUFFER LATCH
31	SWR1	DISPLAY MODE SWITCH INPUT
32	VSS	GROUND
33	RESET	CHIP RESET, ACTIVE LOW
34	A0	ADDRESS LINE INPUT, LSB
35	A1	"
36	A2	"
37	A3	"
38	A4	"
39	A5	"
40	A6	"
41	A7	"
42	A8	"
43	A9	"
44	A10	"

PIN NO.	PIN NAME	DESCRIPTION
45	A11	"
46	A12	"
47	A13	"
48	ROMA11	RE-MAPPED ROM ADDRESS LINE A11
49	$\overline{ROM64}$	ENABLE UPPERMOST 64K ROM
50	A14	ADDRESS LINE INPUT
51	A15	"
52	A16	"
53	A17	" , MSB
54	FSEL	50/60Hz VIDEO FRAME SELECT
55	BR/$\overline{W}$	READ/WRITE LINE
56	BIN	BLUE VIDEO INPUT
57	GIN	GREEN "
58	RIN	RED "
59	LOADS	PERIODIC LOW PULSE FOR TIMING REFERENCE
60	VC	VERTICAL LINE CTR
61	VB	"
62	VA	"
63	$\bar{\Phi}o$	$\bar{\Phi}o$ CLOCK INPUT
64	VDD	+5V SUPPLY

FIGURE 2.14
SOFTWARE SWITCHES

3C008	SET BORDER COLOUR TO BLACK
3C009	RED
3C00A	GREEN
3C00B	YELLOW
3C00C	BLUE
3C00D	MAGENTA
3C00E	CYAN
3C00F	WHITE
3C018	SET BACKGROUND COLOUR TO BLACK
3C019	RED
3C01A	GREEN
3C01B	YELLOW
3Co1C	BLUE
3C01D	MAGENTA
3C01E	CYAN
3C01F	WHITE
3C028	ENABLE MULTI COLOUR MODE
3C029	SET TO SINGLE COLOUR MODE OF RED PIXELS
3C02A	GREEN
3C02B	YELLOW
3C02C	BLUE
3C02D	MAGENTA
3C02E	CYAN
3C02F	WHITE
3C04C	SET TO LOW RESOLUTION
3C04D	RGB MODE
3C04E	HIGH RESOLUTION
3C04F	HIGH RESOLUTION
3C050	GRAPHICS MODE

3C051	TEXT MODE
3C052	PURE TEXT OR GRAPHICS MODE
3C053	MIXED TEXT OR GRAPHICS MODE
3C054	DISPLAY PRIMARY PAGE
3C055	SECONDARY PAGE
3C056	TURN OFF EMULATION
3C057	SET EMULATION ONLY
3C07C	WRITE 1ST MEMORY WINDOW
3C07D	2ND
3C07E	3RD
3C07F	4TH

INTERNAL I/O

3C000	READ KEYBOARD DATA
3C010	CLEAR KEYBOARD STROBE
3C020	CASSETTE OUTPUT
3C030	TOGGLE SPEAKER
3C060	CASSETTE INPUT
3C061	BINARY FLAG 1 INPUT
3C062	2
3C063	3
3C064	GAME PADDLE 1 INPUT
3C065	2
3C066	3
3C067	4
3C068	WRITE DATA TO SOUND GENERATOR
3C070	ANALOG CLEAR
3C090	WRITE DATA TO PRINTER
3C1C0	READ PRINTER ACKNOWLEDGE
3C1C1	READ PRINTER BUSY
3C1C2	READ HORIZONTAL BLANKING
3C1C3	READ VERTICAL BLANKING
3C1C4	READ 50/60 Hz STATUS
3C1C5	READ HIGH RESOLUTION SWITCH (SWR1) STATUS

FIGURE 2.15
SCREEN MAPPING

MODE	PAGE	HEX ADDRESS
40 COLUMN TEXT	0	00400 TO 007FF
	1	00800 TO 00BFF
80 COLUMN TEXT	0	01000 TO 017FF
	1	01800 TO 01FFF
LOW RESOLUTION GRAPHICS	0	02000 TO 03FFF
	1	04000 TO 07FFF
RGB GRAPHICS	0	04000 TO 09FFF
	1	0A000 TO0FFFF
DOUBLE RESOLUTION	0	04000 TO 07FFF
	1	08000 TO 0BFFF

CHAPTER 3

EXTERNAL BUS CONNECTIONS

- Unit specifications
- RS232 adaptor bus
- System bus (cartridge connector and internal expansion bus)
- Printer bus
- Game Paddle
- Keyboard connector
- Linear PCB connector
- Gate array 1
- Gate array 2
- Production specifications for Linear PAL board

UNIT SPECIFICATION

Size : Length 495 mm
Depth 250 mm
Height 90 mm

Weight : 3.9kg

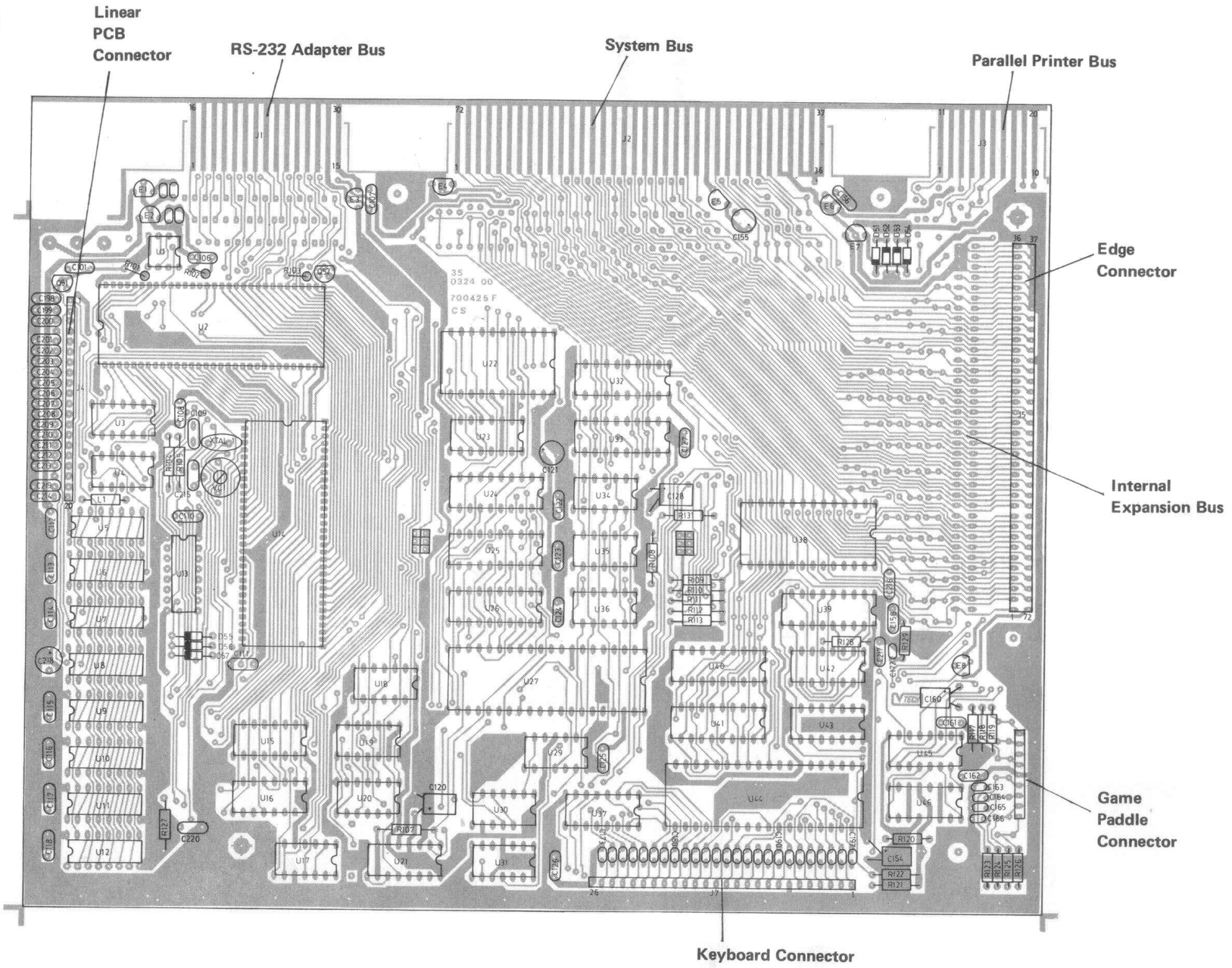
Power cable : Length : 6 feet
Size : AWG 18

Environment: System ON 15 to 30 C, 20% to 80% humidity
System OFF 10 to 45 C, 20% to 80% humidity

Electrical : 90 VAC to 130 VAC, 50W max, 20W without peripherals
180 VAC to 250 VAC, 50W max, 20W without perpherals

Composite cable : Length : 5 feet

Cassette cable : Length : 5 feet


Linear
PCB
Connector
RS-232 Adapter Bus
System Bus
Parallel Printer Bus
Edge
Connector
Internal
Expansion Bus
Game
Paddle
Connector
Keyboard Connector

RS232 ADAPTOR BUS

PIN NO.	PIN NAME	DESCRIPTION
1	BD6	DATA LINE
2	BD7	"
3	BD1	"
4	BD3	"
5	BA0	ADDRESS LINE, LSB
6	BA5	"
7	BA11	"
8	BA10	"
9	BA8	"
10	BA6	"
11	BA4	"
12	BA3	"
13	BA9	"
14	GND	SYSTEM GROUND
15	+5V	+5V
16	BD5	DATA LINE
17	BD0	" , LSB
18	BD4	"
19	BD2	"
20	BA7	ADDRESS LINE
21	BA2	"
22	BA1	"
23	N.C.	
24	N.C.	
25	$\bar{\Phi}$o	1/2 MHz CLOCK

PIN NO.	PIN NAME	DESCRIPTION
26	BR/W	SYSTEM READ/WRITE LINE
27	F4M	4 MHz CLOCK (3 LOW : 4 HIGH DUTY CYCLE)
28	$\overline{\text{PIOSEL}}$	DECODER OUTPUT, TTL LEVEL, $\overline{3CXXX}$
29	N.C.	
30	+12V	+12V

SYSTEM BUS, CARTRIDGE CONNECTOR AND INTERNAL EXPANSION BUS

PIN NO.	PIN NAME	DESCRIPTION
1	$\overline{\text{PIOSEL}}$	$\overline{\text{DECODER}}$ OUTPUT, TTL LEVEL, $\overline{\text{3CXXX}}$
2	BA0	SYSTEM ADDRESSS LINE, LSB
3	BA1	"
4	BA2	"
5	BA3	"
6	BA4	"
7	BA5	"
8	BA6	"
9	BA7	"
10	BA8	"
11	BA9	"
12	BA10	"
13	BA11	"
14	BA12	"
15	BA13	"
16	A14	BUFFERED CPU ADDRESS LINE
17	A15	BUFFERED CPU ADDRESS LINE
18	BR/$\overline{\text{W}}$	SYSTEM READ/WRITE
19	BA16	SYSTEM ADDRESS LINE
20	$\overline{\text{IOSTR}}$	ADDRESS DECODER OUTPUT, TTL LEVEL, 3C800 –> 3CFFF
21	$\overline{\text{RDY}}$	CPU READY INPUT, TTL LEVEL
22	$\overline{\text{DMA}}$	DIRECT MEMORY ACCESS, TTL LEVEL, NORMALLY HIGH
23	INT OUT	DAISY CHAIN INTERRUPT OUT
24	DMA OUT	DMA DAISY CHAIN OUT
25	+5V	

PIN NO.	PIN NAME	DESCRIPTION
26	BA14	SYSTEM ADDRESS LINE
27	BA15	"
28	BA16	"
29	BA17	" , MSB
30	F28M	28MHz SYSTEM CLOCK
31	$\overline{\text{ROM64K}}$	ADDRESS DECODER OUTPUT, TTL LEVEL, ENABLE UPPER-MOST 64K ROM
32	GND	GROUND
33	GND	GROUND
34	+5V	+5V SUPPLY
35	+5V	+5V SUPPLY
36	+5V	+5V SUPPLY
72	+12V	+12V SUPPLY
71	BD0	SYSTEM DATA BUS, LSB
70	BD1	"
69	BD2	"
68	BD3	"
67	BD4	"
66	BD5	"
65	BD6	"
64	BD7	SYSTEM DATA BUS, MSB
63	N.C.	
62	Φ_0	PHI0 CLOCK (=CPU CLOCK WITHOUT DMA)
61	N.C.	
60	Φ_1	$\overline{\text{PHI0}}$
59	Q3	2 MHz CLOCK FOR GENERAL USE, TTL LEVEL
58	F7M	7 MHz INTERMEDIATE CLOCK
57	BA17	SYSTEM ADDRESS LINE, MSB

PIN NO.	PIN NAME	DESCRIPTION
56	F4M	4 MHz CLOCK (3 LOW : 4 HIGH DUTY CYCLE)
55	F14M	14MHz SYSTEM CLOCK
54	$\overline{INH}$	INTERNAL ROM DISABLE
53	CPU RST	RESET CPU AND PERIPHERAL DEVICES
52	$\overline{IRQ}$	INTERRUPT REQUEST, TTL LEVEL, ACTIVE LOW
51	$\overline{NMI}$	NON-MASKABLE REQUEST, TTL LEVEL, ACTIVE LOW
50	INTIN	INTERRUPT DAISY CHAIN INPUT
49	DMAIN	DMA DAISY CHAIN INPUT
48	GND	GROUND
47	SWR0	DISPLAY MODE SWITCH STATUS, LOW ORDER BIT
46	SWR1	" HIGH "
45	SWP1	DISPLAY PAGE SWITCH STATUS: 0=PRIMARY PAGE, 1=SECONDARY PAGE.
44	SWT	TEXT/GRAPHICS DISPLAY MODE SWITCH STATUS
43	GND	GROUND
42	GND	"
41	GND	"
40	GND	"
39	GND	"
38	GND	"
37	GND	GROUND

PRINTER BUS

PIN NO.	PIN NAME	DESCRIPTION
1	$\overline{\text{ACK}}$	PRINTER ACKNOWLEGE, TTL ACTIVE LOW
2	BUSY	PRINTER BUSY, TTL ACTIVE HIGH
3	PR1	PRINTER DATA
4	PR0	" , LSB
5	PR4	"
6	PR5	"
7	PR2	"
8	GND	GROUND
9	PR6	PRINTER DATA
10	PR7	" , MSB
11	PSTROBE	STROBE PRINTER INTO PRINTER
12	N.C.	
13	+5V	
14	+5V	
15	+5V	
16	N.C.	
17	N.C.	
18	PR3	PRINTER DATA
19	N.C.	
20	N.C.	

GAME PADDLE

PIN NO.	PIN NAME	DESCRIPTION
1	+5V	
2	GND	
3	PD2	ANALOG INPUT, 0 – 5V
4	PD3	" "
5	PD1	" "
6	PD0	" "
7	SW0	SWITCH INPUT, TTL LEVEL, NORMALLY HIGH
8	SW1	"
9	SW2	"

KEYBOARD CONNECTOR

PIN NO.	PIN NAME	DESCRIPTION
1	Y7	Y-LINE OF KEYBOARD MATRIX
2	Y8	"
3	Y9	"
4	Y1	"
5	+5V	+5V SUPPLY
6	Y2	Y-LINE
7	Y3	"
8	Y4	"
9	Y5	"
10	Y6	"
11	Y0	"
12	X0	X-LINE OF KEYBOARD MATRIX
13	X1	"
14	X2	"
15	X3	"
16	X4	"
17	X5	"
18	X6	"
19	X7	"
20	Y10	Y-LINE
21	Y11	Y-LINE
22	INDICATOR	CAP-LOCK INDICATOR
23	CTRL	CONTROL KEY
24	SHIFT	SHIFT KEY
25	CAP LOCK	CAP LOCK KEY
26	GROUND	GROUND

LINEAR PCB CONNECTOR

PIN NO.	PIN NAME	DESCRIPTION
1	CASSETTE IN	CASSETTE INPUT, TTL LEVEL
2	+12V	
3	GND	
4	+5V	
5	H/2	1/2 HORIZONTAL CHARACTER FREQUENCY, TTL LEVEL
6	RED	VIDEO GUN RED, TTL LEVEL
7	GREEN	GREEN VIDEO, "
8	BLUE	BLUE " . "
9	$\overline{\text{CBLANK}}$	COMPOSITE BLANKING, TTL LEVEL, ACTIVE LOW
10	$\overline{\text{HSYNC}}$	HORIZONTAL SYNC, TTL LEVEL, ACTIVE LOW
11	$\overline{\text{VSYNC}}$	VERTICAL SYNC, TTL LEVEL, ACTIVE LOW
12	CPURST	CPU AND EXTERNAL PERIPHERALS RESET
13	TTLS	TTL LEVEL SOUND OUTPUT
14	CASSETTE OUT	CASSETTE OUTPUT, TTL LEVEL
15	$\overline{\text{BURST}}$	BURST GATE, TTL LEVEL
16	SGCS	SOUND GENERATOR OUTPUT,150mV TYPICAL
17	GND	GROUND

GATE ARRAY 1 SPECIFICATION

PIN NO.	PIN NAME	DESCRIPTION
1	VSS	GROUND
2	F14M	14MHz CLOCK INPUT
3	F7M	7MHz CLOCK INPUT, SYNCHRONOUS TO RISING EDGE OF F14M
4	F4M	4MHz CLOCK INPUT (3 LOW : 4 HIGH DUTY CYCLE)
5	LOADS	NEGATIVE PULSE INPUT, MUST OCCUR LESS THAN 250NS AFTER CPUCK
6	DOTCK	VIDEO DOT CLOCK OUTPUT
7	RGUN	RED VIDEO O/P
8	GGUN	GREEN VIDEO O/P
9	BGUN	BLUE VIDEO O/P
10	$\overline{RAS}$	DRAM ROW ADDRESS STROBE
11	HOVA	MULTIPLEXED HORIZONTAL AND VERTICAL CTR O/P, Φ= 0, O/P = HO
12	H1VB	" O/P = H1
13	H2VC	" O/P = H2
14	H3V0	" O/P = H3
15	H4V1	" O/P = H4
16	H5V2	" O/P = H5
17	H6V3	" O/P = H6
18	V4V5	MULTIPLEXED VERTICAL CTRS O/P, Φ = 0, O/P=V4
19	VDD	+5V
20	RAM0	DRAM MULTIPLEXED ADDRESS, OUTPUT, LSB

*ALL I/0 ARE TTL LEVEL, I_{OH} =-400μA AT 2.7V
I_{OL} =4mA AT 0.4V
I_{IN} =+/−10uA

PIN NO.	PIN NAME	DESCRIPTION
21	RAM1	"
22	RAM2	"
23	RAM3	"
24	RAM4	"
25	RAM5	"
26	RAM6	"
27	RAM7	" , MSB
28	SWR0	DISPLAY MODE SWITCH O/P
29	SWR1	"
30	SWT	TEXT/$\overline{\text{GRAPHICS}}$ DISPLAY MODE SWITCH O/P
31	SWP1	SWP1=0 –> PRIMARY; SWP1=1 –> SECONDARY DISPLAY PAGE
32	VSS	GROUND
33	$\overline{\text{RESET}}$	CHIP RESET, ACTIVE LOW
34	A0	ADDRESS LINE INPUT, LSB
35	A1	"
36	A2	"
37	A3	"
38	A4	"
39	A5	"
40	A6	"
41	A7	"
42	A8	"
43	A9	"
44	A10	"

45	A11	"
46	A12	"
47	A13	"
48	A14	"
49	A15	" , MSB
50	BA14	WINDOW ADDRESS LINE OUTPUT
51	BA15	"
52	BA16	"
53	BA17	"
54	FSEL	50/60Hz VIDEO FRAME SELECT
55	BD0	DATA BUS INPUT, LSB
56	BD1	"
57	BD2	"
58	BD3	"
59	BD4	"
60	TXTD5	TEXT/DATA BUS BD5 INPUT, SWT=1, TEXT; SWT=0, DATA INPUT
61	BD6	DATA BUS INPUT
62	BD7	" , MSB
63	CPUCK	CPU CLOCK INPUT 1/2 MHz WHEN SWR1=0/1
64	VDD	+5V SUPPLY

GATE ARRAY 2 SPECIFICATION

PIN NO.	PIN NAME	DESCRIPTION
1	VSS	GROUND
2	$\overline{\text{ROUT}}$	RED VIDEO OUTPUT, ACTIVE LOW TTL
3	$\overline{\text{GOUT}}$	GREEN "
4	$\overline{\text{BOUT}}$	BLUE "
5	HSYNC	HORIZONTAL SYNC, ACTIVE HIGH TTL
6	VSYNC	VERTICAL SYNC "
7	CBLANK	COMPOSITE BLANKING, "
8	BGATE	BURST GATE, "
9	BD7	DATA BUS, MSB
10	$\overline{\text{IOEN}}$	I/O ENABLE, ACTIVE LOW TTL, 3C000 –> 3C07F
11	HOVA	Φo MULTIPLEXED HORIZONTAL/VERTICAL CTR INPUT
12	H1VB	"
13	H2VC	"
14	H3VO	"
15	H4V1	"
16	H5V2	"
17	H6V3	"
18	V5V4	Φo MULTIPLEXED VERTICAL/VERTICAL CTR INPUT
19	VDD	+5V SUPPLY
20	KEYBD	READ KEYBOARD, $3C000

*ALL I/O ARE TTL LEVEL, I_{OH} =-400uA AT 2.7V
I_{OL} =4mA AT 0.4V
I_{IN} =+/–10μA

PIN NO.	PIN NAME	DESCRIPTION
21	KEYSTROBE	KEYBAORD STROBE INPUT, SETS INTERNAL KEY Flip – Flop
22	IOSTR	I/O STROBE OUTPUT, $3C800 – $3CFFF
23	$\overline{\text{RAMOE}}$	RAM LATCH OUTPUT ENABLE, N.C. IN the computer
24	TTLS	1 BIT TTL OUTPUT, ACCESS BY $3C030
25	CASOUT	CASSETTE OUTPUT, TTL LEVEL
26	$\overline{\text{PIOSEL}}$	PRE-I/O SELECT, $3C000 –> $3CFFF
27	$\overline{\text{PRACK}}$	PRINTER ACKNOWLEDGE
28	PRBUSY	PRINTER BUSY
29	PRSTROBE	PRINTER STROBE
30	PRLATCH	LATCH DATA BUS INTO PRINTER BUFFER LATCH
31	SWR1	DISPLAY MODE SWITCH INPUT
32	VSS	GROUND
33	RESET	POWER UP RESET
34	A0	ADDRESS LINE INPUT, LSB
35	A1	"
36	A2	"
37	A3	"
38	A4	"
39	A5	"
40	A6	"

PIN NO.	PIN NAME	DESCRIPTION
41	A7	"
42	A8	"
43	A9	"
44	A10	"
45	A11	"
46	A12	"
47	A13	"
48	ROMA11	RE-MAPPED ROM ADDRESS LINE A11
49	$\overline{\text{ROM64}}$	ENABLE UPPER MOST 64K ROM
50	A14	ADDRESS LINE
51	A15	"
52	A16	"
53	A17	" , MSB
54	FSEL	50/60Hz VIDEO FRAME SELECT
55	BR/$\overline{W}$	READ/WRITE LINE
56	BIN	BLUE VIDEO INPUT
57	GIN	GREEN VIDEO INPUT
58	RIN	RED VIDEO INPUT
59	LOADS	PERIODIC LOW PULSE FOR TIMING REFERENCE
60	VC	VERTICAL LINE CTR
61	VB	"
62	VA	" , LSB
63	PHIO	PHIO CLOCK INPUT
64	VDD	+5V SUPPLY

PRODUCTION SPECIFICATION FOR LINEAR BOARD (PAL VERSION)

PERFORMANCE MEASUREMENT

	CONDITION	SPEC
Composite video	75 ohm loaded	
Output voltage (Sync. tip to white)		1 +/–0.1 Volt
Sync. tip d.c. level		0.9 +/–0.1 Volt
Sound output	10K ohm loaded	175mVrms +/–3db
	TTL input (1KHz/5Vp–p)	
Speaker output	8 ohm load/max. volume	1.3Vrms +/–3db
	TTL input (1KHz/ 5Vp–p)	
Cassette output	1K ohm load	> 30mVp-p

CHAPTER 4

OPERATION OF SWITCHING POWER SUPPLY UNIT

- Theory of operation
  - General
  - Input filter and rectifier
  - Bias and switch
  - Output section
  - Regulation control
  - Protection Circuits
- Production specification
- Alignment procedures

THEORY OF OPERATION

GENERAL

The power supply operates on the "Flyback" principle in which energy is stored and released in a cyclic pattern. The ratio of energy storage during the charging portion of the cycle is determined by the circuit constants. The discharging time, and thus the amount of energy delivered to the load is determined by the load requirement.

Essentially the power supply consists of five functional blocks: Input filter and Rectifier, Drive and Switching element, Regulation control, Output section and Protection circuit.

INPUT FILTER AND RECTIFIER

The AC input filter consists of T1, C1, C2, C3 and C5. The purpose of this circuit is to filter out 20KHz and above switching noise, preventing it from being transmitted back out the input line. The fuse F1 is included to protect the PC Board traces and to reduce fire and personal hazard in the event of catastrophic supply failure. In-rush current limiting is accomplished with resistor R1. D1-D4 forms a bridge rectifier which converts the AC supply voltage into DC voltage. The energy is then stored in capacitor C6.

BIAS AND SWITCH

Q1 is the switching element which is driven by Q2 and Q3. RC network C7 and R3 forms a snubber network that absorbs harmful voltage spike. The transformer is so phased to form a blocking oscillator. D5, C9, R4 forms a DC biasing network for the switch Q1.

OUTPUT SECTION

Each output voltage is developed from the transformer output through a single diode rectifier feeding a capacitor input filter.

REGULATION CONTROL

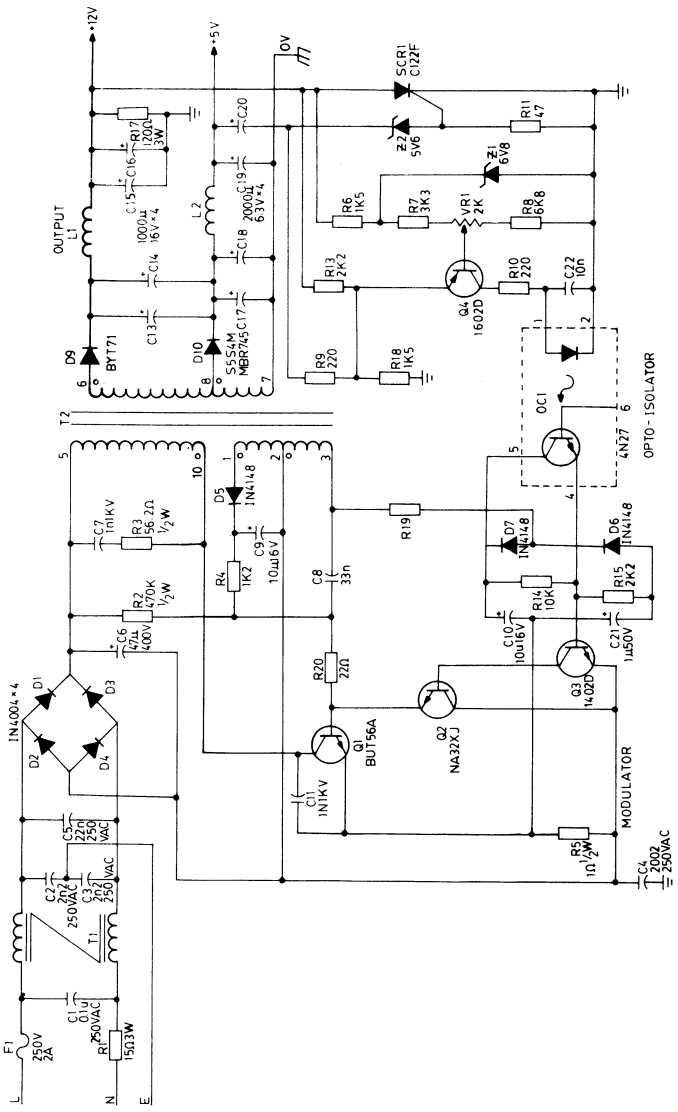
1) Error Amplifier
   The +5V and +12V output is sampled by R9, R18, R13 and is compared to a reference voltage source consists of R6, R7, R8, VR1 and Z1. The output of error amplifier Q4 drives the opto-islator OC1 which in-turn drives Q3 through modulator network.

2) Modulator
   The R.C.D. network D7, D6, R14, R15, C10, C21, forms a modulator. The primary ramp current and the output of the opto-isolator is processed by the modulator to adjust the turn off point of the switch Q1.

PROTECTION CIRCUIT

1) Overcurrent Protection
   If a secondary output is shorted, the primary current, thus the voltage across R5 increases, this causes Q3 to be immediately driven into saturation, thus cutting off Q1. The protective "burping" action is repeated far as long as the short remains.

2) Over voltage protection
   Over voltage protection for the loads is provided by SCR, and Z2. The +12V output is monitored via the anode of SCR1, the +5V output via Z2 in the gate circuit. An over-voltage condition in either output will cause SCR1 to fire, Simulating an overcurrent condition. This "Crow-bar" effect initiates the overcurrent protection circuit described above.


+12V
+5V
0V
SCR1
C122F
Z2
5V6
Z1
6V8
R11
47
R6
1K5
R7
3K3
VR1
2K
R8
6K8
OUTPUT
L1
L2
C15
1000µ
16V*4
C19
2000µ
6.3V*4
C20
R17
120Ω
3W
C16
C14
C18
C13
C17
D9
BYT71
D10
S5S4M
MBR745
R13
2K2
R10
220
C22
10n
Q4
16020
R9
220
R18
1K5
T2
OC1
4N27
OPTO-ISOLATOR
D5
1N4148
C7
1n1KV
R3
56.2Ω
½W
R19
D7
1N4148
D6
1N4148
R4
1K2
C9
10µ16V
C8
33n
R2
470K
½W
R14
10K
R15
2K2
C6
1µ
400V
C10
10µ16V
C21
1µ50V
R20
22Ω
Q3
14020
Q2
NA32XJ
Q1
BUT56A
C11
1N1KV
MODULATOR
R5
1Ω/2W
C4
200Ω
250VAC
D1
D2
D3
D4
1N4004*4
C5
220n
250
VAC
C2
220n
250VAC
C3
250
VAC
T1
C1
0.1µ
250VAC
R1
150Ω3W
F1
250V
2A
L
N
E

PRODUCTION SPECIFICATION FOR THE COMPUTER'S SWITCHING POWER SUPPLY

A. Output Voltage

V01 = 4.80V–5.20V
V02 = 11.40V–12.60V
for Conditions (a) – (f)

AC 110V/220V 50Hz (*) input

a) I01 = 1.0A	I02 = 0.5A
b) I01 = 3.0A	I02 = 0.5A
c) I01 = 1.0A	I02 = 1.5A
d) I01 = 3.0A	I02 = 1.5A
e) I01 = 1.0A	I02 = 0A
f) I01 = 3.0A	I02 = 0A

B. Ripple

Rvo1 = < 100mV pp
Rvo2 = < 200mV pp

for Condition AC 110V/220V 50Hz (*) input
I01 = 3.0A, I02 = 1.5A

C. Short Circuit Output Current

Is1 = < 3.0A
Is2 = < 1.5A

for Condition AC 110V/220V 50Hz (*) input

D. Overvoltage Protection Threshold
V01 = 6.25V +/–0.75V

E. Dielectric Test
2100 VDC for 110V version
2100 VDC for 220V non-VDE version
3000 VDC for 220V CDE version

between (a) Case and primary
(b) Secondary and Primary

Cut off current at 0.5mA, 60 seconds

F. Efficency

>= 70% at I01 = 3.0A, I02 = 1.5A

G. Operation Voltage Range

V01 = 4.8V–5.20V
V02 = 11.40V–12.60V

for AC 110V/220V 50Hz (*) input

and I01 = 3.0A
I02 = 1.5A

* : 110V for 110V models, 220V for 220V models

THE COMPUTER'S SWITCHING POWER SUPPLY ALIGNMENT PROCEDURE

CONDITIONS

AC 110V 50Hz for 110V model

AC 220V 50Hz for 220V model

V01 at 1.0A

V02 at 1.5A

ALIGNMENT PROCEDURE

Adjust VR1 so that

V02 = 11.40V

CHAPTER 5

PERIPHERAL DATA

- Keyboard electrical
  and mechnaical specs
- Printer cable pin assignments
  Installation and test procedure
- Joystick connector signals
  memory locations and installation procedures

THE COMPUTER'S KEYBOARD SPECIFICATION

1. ELECTRICAL SPECIFICATION

- Full 81 key array
- Numeric key pad plus 4 cursor control keys
- Eight user definable function keys
- Single chip microcomputer (8048) as keyboard controller
- Automatic repeat. Repeat frequency 10Hz after 0.6 sec key depression.
- ASCII Encoding
- N-key rollover
- Software cap lock with LED indicator
- 4 modes of operations (Unshifted, Alpha Lock, Control, Shift-control)
- Conductive rubber switching technology. Operation contact resistance less than 1.5K ohm.
- Insulation resistance 30V 10M ohm approx.

2. MECHANICAL SPECIFICATION

- Non glare keytop
- Step sculptured keytops
- Multicolor keytop
- Injection molded enclosure
- Key total travel 4mm nominal
- Pretravel 2.2mm
- Key actuating force 60gm.
- Switch reliability 5 X 10^6 cycles

PRINTER CABLE INSTALLATION TEST PROCEDURE & PIN ASSIGNMENT.

With the Printer Cable, you can now connect your Computer to any Centronics Bus printer.

CAUTION:

Before connecting or disconnecting TURN OFF ALL COMPUTER AND PRINTER POWER.

INSTALLATION

1) Turn off all power.

2) Attach the Printer Cable to the Printer socket in the rear panel of the computer. A polarization slot on the connector make you wouldn't attach the cable in reverse side.

3) Check to ensure that the Printer Cable is fully inserted and firmly attached.

4) Connect the other end of the Printer Cable to the printer. Double check that it is well-connected.

5) Turn on the computer and printer. You are now ready to use the advanced printing capability of the computer.

TEST PROCEDURE

To check whether your printer has been correctly set-up, try the following test.

1) Enter the following program:

```
10  PR#1
20  FOR I=32 TO 128
30  PRINT CHR$(I);
40  NEXT I
50  PR#0
60  END
```

2) Type in RUN, 'RETURN', the printer will print out the entire range of alphanumeric characters.

NOTE:

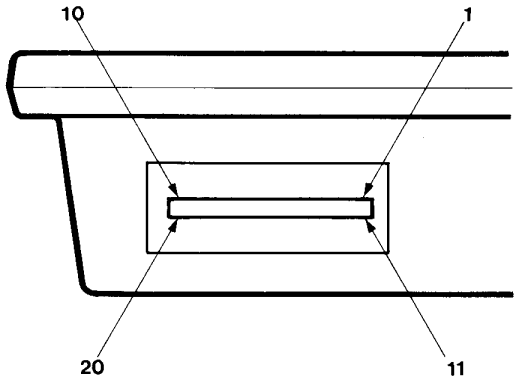
For normal text printing, any type of Centronics Bus printer will work well with the computer. For graphics printing, a BX-80 series, dot matrix printer or compatable printer is required. Refer to computer BASIC REFERENCE MANUAL for detail description.

PIN ASSIGNMENT

1. Computer side

PIN NO.	SIGNAL	PIN NO.	SIGNAL
1	$\overline{\text{ACK}}$	11	STROBE
2	BUSY	12	NC
3	DATA 1	13	+5V
4	DATA 0	14	+5V
5	DATA 4	15	+5V
6	DATA 5	16	NC
7	DATA 2	17	NC
8	GND	18	DATA 3
9	DATA 6	19	NC
10	DATA 7	20	NC

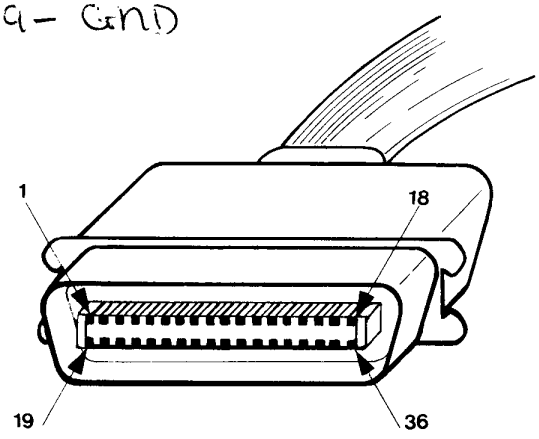
NOTE NC: No Connection



2. Printer side

PIN NO.	SIGNAL
1	STROBE
2	DATA 0
3	DATA 1
4	DATA 2
5	DATA 3
6	DATA 4
7	DATA 5
8	DATA 6
9	DATA 7
10	ACK
11	BUSY
12–18	N.C.
19–27	N.C.

19 – GND

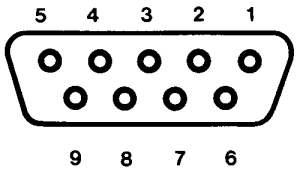


JOYSTICK

JOYSTICK CONNECTOR SIGNALS

A 9-pin D-type miniature connector in the right-hand side of the computer provides signals required for the Joysticks. There are three switch inputs, called SW0–SW2, and four paddle inputs called PDL0–PLD3.

PIN NO.	SIGNAL	DESCRIPTION
2	+5V	+5 volt power
3	GND	Signal ground
5, 8, 4, 9	PDL0–PDL3	Paddle inputs
7, 1, 6	SW0–SW2	Switch inputs



Joystick Connector

The switch inputs are standard low-power schottky TTL inputs. It can be used to detect the status of a switch. To use them, connect each one to a 220 ohm pull-down resistors to ground and through single pole push switches to +5 volt supply.

Each one of the switch input is corresponding to a memory location. Program can detect the status of the switches by read these locations. But only the high-order bit is valid information, the other bit is undefined. For BASIC language, PEEK the location and check the value with 128. If the value is 128 or greater, the switch is on.

The paddle inputs are connected to the timing input of type NE558 quadruple analog timer. It can be connected through a 150K ohm variable resistor to +5 volt supply and form a one-shot timing circuit. The time that the timer changes state is proportional to the corresponding resistance, and hence to the position of the joystick.

To read the paddle inputs, the program must first reset the timing circuit by accessing memory location $C070 (decimal 49264 or −16272). All the four timer is set high and then return to low within about 3 milliseconds.

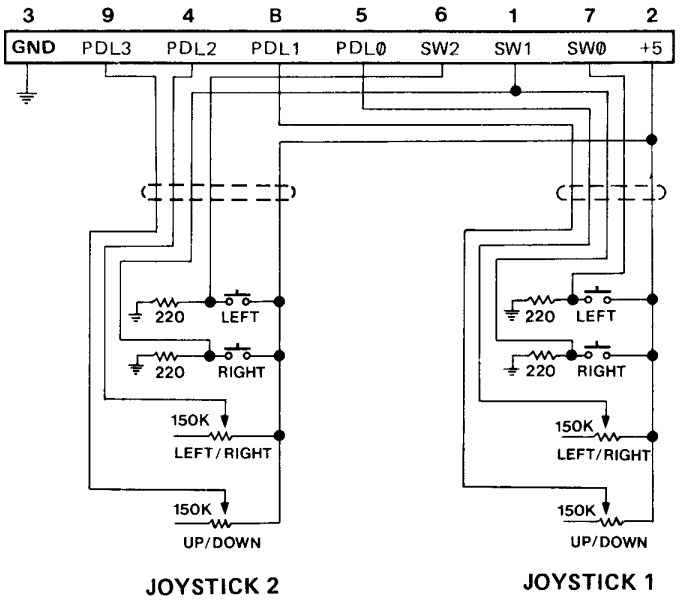
The exact time for the timer to return to low is proportional to the resistance. For timer set high, a read to the appropriate memory location gives a value equal to or greater than 128, or in other words, the high-order bit set. So program can count the time remain high by check the high-order bit and hence determine the position of the variable resistor.

JOYSTICK MEMORY LOCATION

FUNCTION	ADDRESS		
	HEX	DECIMAL	
SW0	$C061	49249	−16287
SW1	$C062	49250	−16286
SW2	$C063	49251	−16285
PDL0	$C064	49252	−16284
PDL1	$C065	49253	−16283
PDL2	$C066	49254	−16282
PDL3	$C067	49255	−16281
PADDLE INPUT RESET	$C070	49264	−16272

Also, the BASIC function PDL can be used to determine the paddle position. Refer to the computer BASIC REFERENCE MANUAL for detail description.

CIRCUIT DIAGRAM



Note: RIGHT fire button of Joystick 1 and Joystick 2 are connected in parallel.

JOYSTICK INSTALLATION PROCEDURE

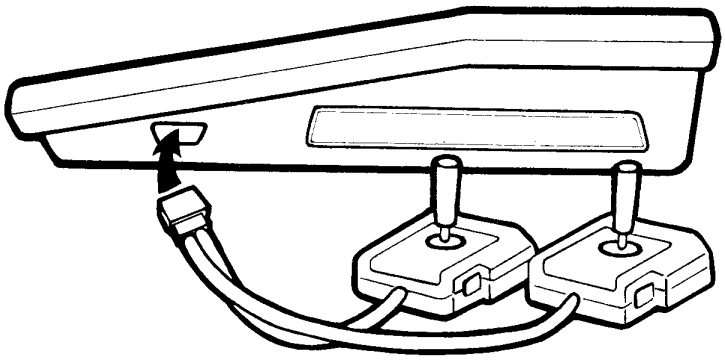
These Joysticks offer you fast direction control and dual fire button facility. Now, you can get more fun from the computer's entertainment programs.

CAUTION:

Before connecting or disconnecting TURN OFF POWER.

INSTALLATION

1) Turn off power.
2) Plug the Joystick cable into the joystick connector in the right hand side of computer.
3) Turn on power, a pair of Joysticks is ready for you.



CHAPTER 6

MEMORY MAPPING

- Memory windows, Window addressing & Window Updates
- The main memory map
- I/O mapping
- I/O device mapping
- BASIC reserved memory locations

MEMORY WINDOWS, WINDOW ADDRESSING & WINDOW UPDATE

Your computer's 6502A micro-processor can address 65,536 (64K) bytes of memory directly. The gate arrays inside your computer further enchance the addressing capability to 256 KB. Four windows of 16KB each maps part of the physical memory of 256 KB into 6502A logical address space. Figure 3.1 below shows an example of the windowing technique.

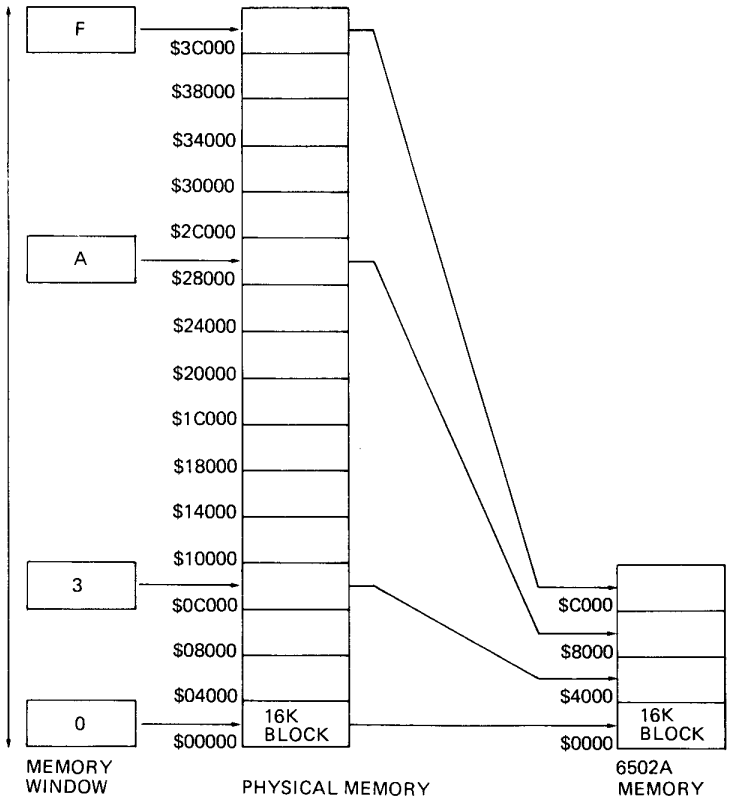


Figure 3.1 Window Addressing

Thus, changing the content of the windows will map different 16K–BLOCK of physical memory into 6502A's memory space. Window updating is done by writing a byte into the window address listed in figure 3.2. Figure 3.3 shows an example of window update.

Physical Address	Window no
$3CØ7C	Ø
$3CØ7D	1
$3CØ7E	2
$3CØ7F	3

Figure 3.2 Window Address

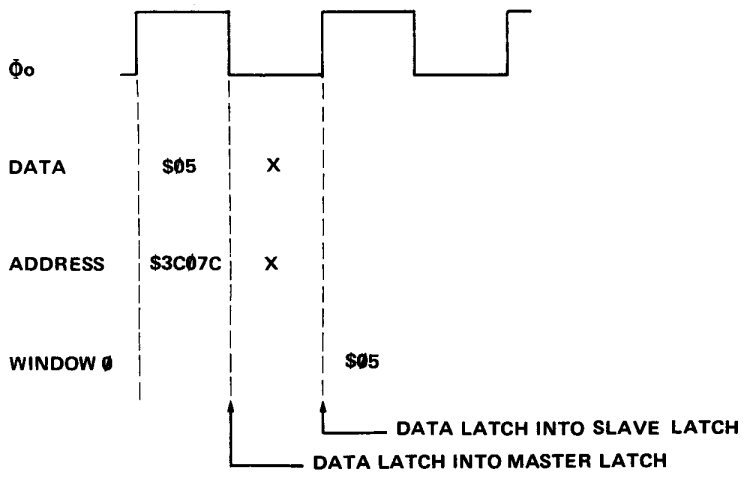


Figure 3.3 Window Update

The window latches are master slave type. The lower 4 data bits are latched into the master latch during falling edge of PHIO and the master content will be transferred to the slave latch at the rising edge of PHIO. Figure 3.4 shows a block diagram for this window structure.

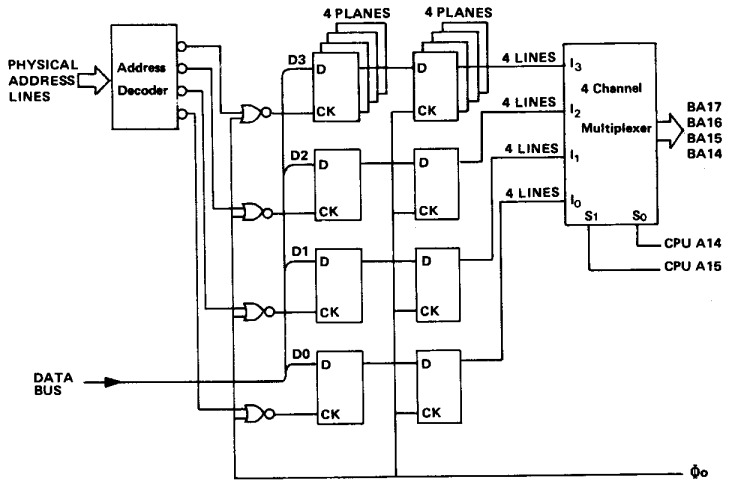


Figure 3.4 Block diagram of window structure.

The CPU address line A14, A15 routes directly to the gate arrays to select which 4-bit group of 4 will be outputed as the upper 4 system address lines. Figure 3.5 shows an example.

Window 0 = 1
Window 1 = 3
Window 2 = 5
Window 3 = E

CPU A15, A14		WINDOW SELECTED	BA17, BA16, BA15, BA14			
0	0	0	0	0	0	1(1)
0	1	2	0	0	1	1(3)
1	0	2	0	1	0	1(5)
1	1	3	1	1	1	0(E)

Figure 3.5 example of window addressing

MAIN MEMORY MAP

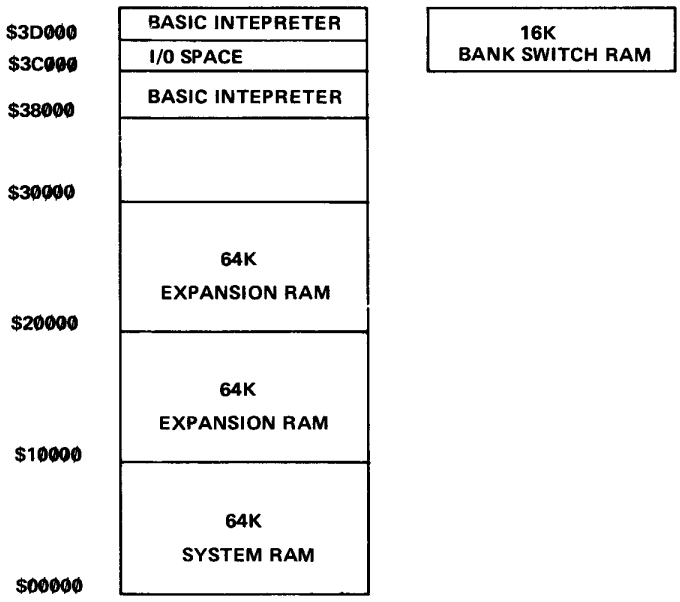


Figure 3.6 Main Memory Map

Internal to your computer, you can add a 128K RAM expansion card which occupies memory space $10000 to $2FFFF, and a 16K Bank switched RAM. Notice that address space $30000 to $37FFF has no device connected and is free for custom expansion. The built in enchanced Microsoft Basic occupies two sections. : $38000 to $3BFFF and $3FFFF. Between the Basic interpreter lies 4 KB of I/O space. Figure 3.6 shows the main memory map.

I/O MAP

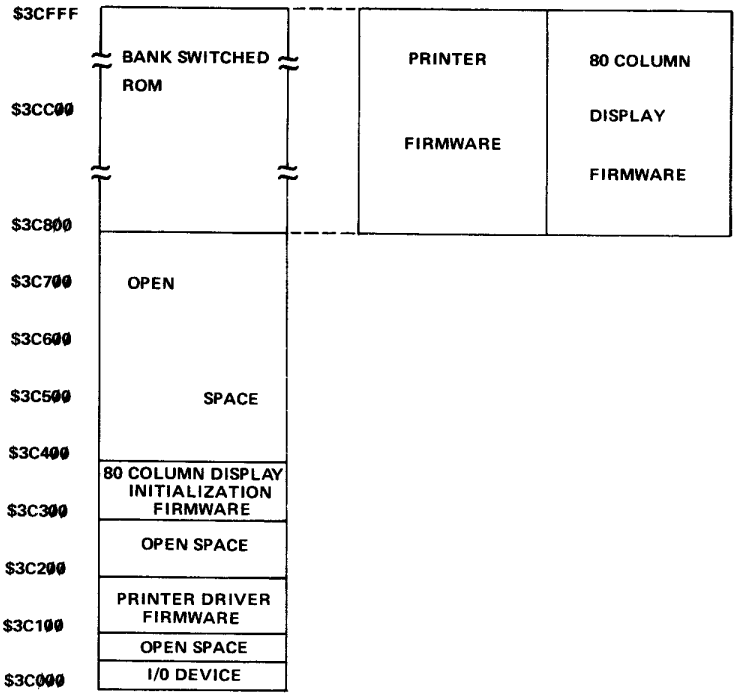


Figure 2.71 I/O mapping

Figure 2.7 shows the I/O mapping. Two 2KB bank switch ROMs houses the PRINTER and 80 COLUMN DISPLAY FIRMWARE. Normally both ROMs are disabled. To access these bank switched ROM, program must branch to the corresponding initialization routine located at $3C100 and $3C300. After execution of the initialization and driver routine, the corresponding bank switched ROM is turned on Figure 3.8 shows the correct sequence for programming using bank switch ROMs. Figure 3.9 gives a schematic for associated hardware required when additional bank switch ROM is to be installed.

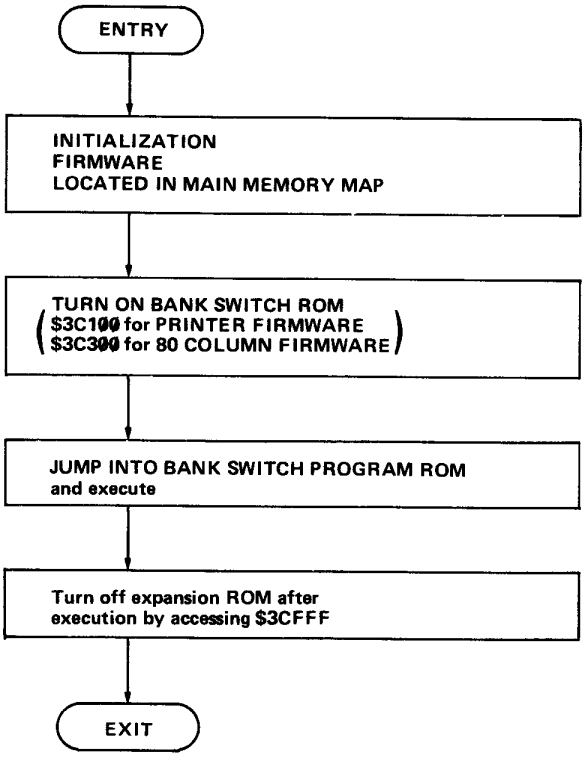


Figure 3.8 sequence for accessing bank switch ROM.

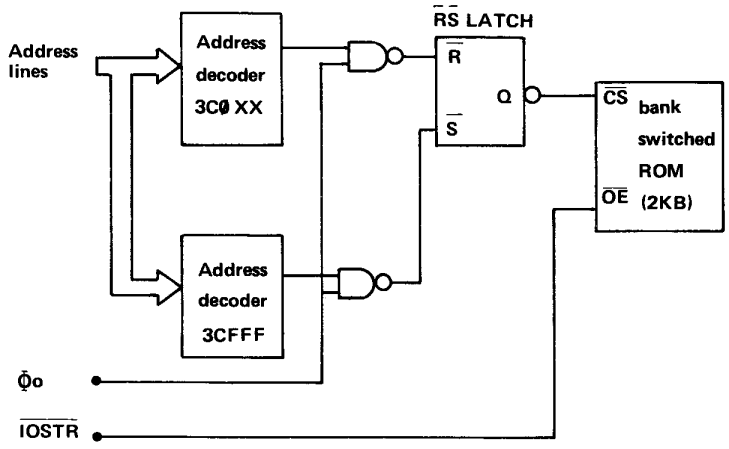


Figure 3.9 supporting hardware for bank switch ROM.

I/O DEVICE MAPPING

Memory space $3C000 to $3C07F houses the Input/Output devices such as sound generator, memory windows, joystick electronics, keybaord data etc. Figure 2.14 shows all the SOFTWARE SWITCHES location and internal I/O device mapping.

CAT BASIC Zero Page Usage

Decimal	Hex	0 $0	1 $1	2 $2	3 $3	4 $4	5 $5	6 $6	7 $7	8 $8	9 $9	10 $A	11 $B	12 $C	13 $D	14 $E	15 $F
0	$00	●	●	●	●	●	●	●	●			●	●	●	●	●	●
16	$10	●	●	●	●	●	●	●	●	●	●						
32	$20	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
48	$30	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
64	$40	●	●	●	●	●	●	●	●	●	●					●	●
80	$50	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
96	$60	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
112	$70	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
128	$80	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
144	$90	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
160	$A0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
176	$B0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
192	$C0	●	●	●	●	●	●	●	●	●	●	●	●	●	●		
208	$D0	●	●	●	●	●	●			●	●	●	●	●	●	●	●
224	$E0	●	●	●	●	●	●	●	●	●	●	●	●				
240	$F0	●	●	●	●	●	●	●	●	●	●	●					

● Used by the BASIC interpreter

Figure 1

CHAPTER 7

THE SYSTEM KERNEL

- System kernel overview
- Entering the kernel
- Communicating with the kernel
- Examining memory
- Examining some more memory
- Altering memory contents
- Moving a range of memory
- Comparing two ranges of memory
- Saving the contents of a memory range on tape
- Reading back a memory range from tape
- Other kernel commands
- Leaving the kernel
- A summary of kernel commands

SYSTEM KERNEL OVERVIEW

Residing deep inside the computer's read-only memory (ROM) is a supervisory program called the Kernel (in some computers, it is called the Monitor). It serves as a link between the system hardware and the high level languages. It is used whether you are programming in machine language, BASIC, PASCAL or almost any other language. Without it, you could not get information into or out of the computer. The keyboard, text display, cassette, and disk drives would all become inactive and unusable.

You can use the Kernel via keyboard commands. Some reasons for doing this would be to look at one or more memory locations; to change the contents of a location, or to write programs in machine language to be executed directly by the computer's microprocessor – the 6502A. Most of the time you will find no need to use it, but the Kernel has some functions which you may find handy at some time.

ENTERING THE KERNEL

The Kernel program of computer starts at memory location $FF69 ($ stands for hexadecimal). So, a subroutine call to $FF69 will bring you there. In BASIC, the interpreter does not recognize hexadecimal numbers and you should type in the following command in order to get into the Kernel.

CALL −151 or CALL 65385

65385 is actually the decimal equivalent of $FF69.

COMMUNICATING WITH THE KERNEL

Once you have entered the Kernel, you will see the prompt (*) and cursor appearing at the left edge of the screen. This indicates that the Kernel is now waiting for your input line. As in BASIC, the Kernel will not respond to any command unless you press the (RETURN) or the (ENTER) key. Each input line to the Kernel may consist of up to 255 characters.

How do you type commands into the Kernel? The Kernel recognizes over ten different COMMAND CHARACTERS, which in appropriate context specify WHAT action is to be taken. In many cases, merely a command itself is neither complete nor grammatically correct, unless additional information in the form of ADDRESS or DATA is also supplied. Address and data typed into the Kernel are always treated as hexadecimal numbers.

In the computer, the address of every memory location is numbered with four hex digits, and the content of each memory location is represented by two hex digits. When the Kernel looks for an address, it is actually looking for a group of four hex digits. If an address has been typed with less than four digits, the Kernel will assume the hex digit group come with leading zero (s). If an address contains more than four hex digits, the Kernel will truncate the number and only accept the last four hex digits as an address. The same procedure is followed when looking for two-digit data values.

EXAMINING MEMORY

EXAMINING THE CONTENTS OF A MEMORY LOCATION

Type in the hexadecimal address of the location you want to see, following by the (RETURN) or (ENTER) key, as in this example:
*DOOO (RETURN)

Remember that any number typed into the Kernel is ALWAYS treated as a hexadecimal number WITHOUT any special designation preceding or following it.

The Kernel responds by displaying the contents of that location:
DOOO=6F

EXAMINING SOME MORE MEMORY

Let's say you examined the single address $DOOO, as in the example above, and you want to look at the next higher locations in memory. You do not have to examine them one by one, as this would be quite time consuming. Instead, you can press (RETURN) or (ENTER) and the Kernel will respond with one line of a memory dump, as shown below:

```
*DOOO (RETURN)
DOOO= 6F
*(RETURN)
DOO1=D8 65 D7 F8 DC 94 D9
```

The contents in memory locations $DOO1 to $D007 have been shown with a simple (RETURN). One more (RETURN) will cause the Kernel to produce one more line of memory dump.

```
*(RETURN)
DOO8= B1 D8 30 F3 D8 DF E1 DB
```

From the above examples, you should have noticed several things about the format of a memory dump. First, a memory dump always begins with the address of teh location FOLLOWING the last referenced location (the reference may be in the form of contents-examination or others). Secondly, memory dump always begins with an address which ends in either zero or eight (i.e. XXX0 or XXX8). Thirdly, in 40 column text mode as in the above examples, there are never more than eight values displayed on a memory dump; while in 80 column text mode, there are never more than sixteen values displayed on a single line in a memory dump.

ALTERING MEMORY CONTENTS

ALTERING THE CONTENTS OF A MEMORY LOCATION

Type in the hexadecimal address of the lcoation you want to alter, followed by a colon (:), then the two-digit hexadecimal number you intend to write into that location, and press (RETURN) or (ENTER) to inform the Kernel. For example:
*6000:13 (RETURN)

places the number $13 into memory location $6000 and overwrites the original value. To check the result, you should type:
6000 (RETURN)
and the Kernel should respond with
6000= 13

If, instead of $13, a different value was shown, then something must have been wrong, a typing error, for example, and you had better try again.

ALTERING THE CONTENTS IN CONSECUTIVE MEMORY LOCATIONS

To change the contents in consecutive memory locations, you don't have to press an address, a colon, and a value for each location. The Kernel allows you to change the contents of up to 85 consecutive memory locations at a time. To do this, you first type in the starting address of the memory range. Next, use the colon to indicate to the Kernel that this is a memory alteration command. Finally, enter the data you want at each consecutive location, seperating them by a space. For example, to place the quantities $00 through $09 in address $6009, enter:
*6000:00 01 02 03 04 05 06 07 08 09 (RETURN)

Again, it is good practice to check these memory alterations if you want the final product, whether a graphics shape table or a series of machine language instructions, to be correct. To do this you will have to use the memory examine commands described earlier in this chapter.

Finally, a few words about memory alteration, only contents in RAM (which stands for RANDOM ACCESS MEMORY) can be changed. Contents in ROM cannot be altered. In the computer, the memory banks can be moved around by accessing certain soft switches. So, it is quite difficult to say strictly where are the RAMs and where are the ROMs. Nevertheless, in most cases, the following assumptions can be made.

(1) From $0000 to $BFFF (a 48K range) are RAMs.
(2) From $C000 to $C7FF (a 2K range) are Input/Output locations.
(3) From $C800 to $FFFF (a 14K range) are ROMs.

MOVING A RANGE OF MEMORY

The MOVE command allows you to move a whole memory range to a new location. To do this, you must supply the following information to the Kernel:

(1) the source start address (i.e. where to move memory from)
(2) the source end address (i.e. the last address you want to be moved)
(3) the destination address (i.e. where you want to move memory to)

The format of the command looks like:
(Destination) < (starting address). (end address)M

As with other Kernel commands, all addresses are hexadecimal numbers. For example, the command:
*4000 < 2000.3FFFM (RETURN)

moves data to address $4000 (the destination address) from the block starting at $2000 (the source start address) and ending with $3FFF (the source end address). If you are familiar with the memory map of the computer, you may probably realize that the above example can be used to copy the picture in HGR1 to HGR2.

Normally, when you specify addresses in the move memory command, te source start address should be equal to or less than the source end address. Otherwise, only the first byte from the source range will be moved to the destination.

When the address of destination is inside the source range, the MOVE command becomes the FILL command, in which one or more bytes of data are repeatedly written into consecutive memory loactions. Suppose you want to place zeros in the memory range $6000–$6FFF, you may first place zeros in the start location by means of the ALTER MEMORY command:
*6000:00 (RETURN)

Now comes to the second step which makes use of the MOVE command:

(1) set the destination address to be one greater than the last byte of the pattern (in this example it would be $6001)

(2) set the source start address to the beginning of the pattern ($6000 in this case)

(3) set the source end address to the last byte which you want fill ($6FFF) MINUS the length of the pattern you want to fill memory with (in this example, the length is one, so it gives $6FFE)

*6001 <6000.6FFEM (RETURN)

You can check the result by examining the addresses $6000 through $6FFF and seeing whether they are indeed filled with zeros.

As another example, the following Kernel commands will fill memory from $6000 through $6FFF with the four-byte pattern:
$00, $11, $22, $33

```
*6000:00 11 22 33  (RETURN)
*6004 <6000.6FFBM  (RETURN)
```

Once again, you may examine memory from $6000 through $6FFF to verify that the pattern repeatedly occurs in this block.

COMPARING TWO RANGES OF MEMORY

The VERIFY command of Kernel compares two blocks of memory against each other, noting differences between them. Its format is nearly the same as for the MOVE memory command, except the last item which is now the letter V, instead of M.

(Destination) < (starting address). (ending address)V

Here is an example:
*2000 < 4000.5FFFV (RETURN)

This instructs the Kernel to start comparing data at address $2000 against address $4000, and to continue the comparison until address $3FFF is compared with $5FFF.

In case the Kernel finds discrepancy during the comparison process, it will dislplay the source address where discrepant value was found and the disagreeing values. If there is no discrepancy, nothing will be shown. Note when discrepancy happens, the displayed address is an address in the SOURCE memory range. For instance, in the example above, assume there is a discrepancy and the Kernel displays:
*4345= 00 (FF)

This message shows that source address $4345 contains $00, while its relative counterpart ($2345) contains $FF.

As in the case of MOVE, if the ending address of the source range is less than the starting address, only the first bytes of the source and destination data blocks will be compared.

SAVING THE CONTENTS OF A MEMORY RANGE ON TAPE

To save memory on tape, use the Kernel's memory WRITE command. You have to inform the Kernel the beginning and ending addresses of the memory range you intend to save. The command format is shown below:
(start address). (end address)W

For example, the command:
*2000.3FFFW
tells the Kernel to write the contents of memory locations, starting at $2000 and ending with address $3FFF, to the cassette recorder.

The memory write command cannot check a recoder is in RECORD mode or whether the tape recorder is actually connected to the cassette port; or the tape is free from jamming, dropouts, and other problem which are inherent in using tape cassettes.

When you press (RETURN) to execute a tape WRITE command, your computer will first write a reference tone onto the tape which lasts for about 10 seconds. Then data are sent out to the cassette output port at a rate of approximate 210 characters per second. After the memory write command has finished, the computer beeps once and the Kernel prompt appears again.

READING BACK A MEMORY RANGE FROM TAPE

To retrieve data stored on tape with the WRITE command, you can use the Kernel's READ command. You should enter the starting address to which data from the cassette tape should begin loading followed immediately by a period, then the ending address (where the last byte of data read from cassette will be stored), and lastly the letter R.
(start address). (end address)R (RETURN)

Unlike the WRITE command, the memory READ command forces the computer to wait until it encounters the reference tone from the cassette recorder. So, if there is no signal present on the tape, your computer will lock in the READ process endlessly. Before you press PLAY on the cassette recorder, make sure you position the tape to where the reference tone begins. You can tell the difference between the reference tone and the actual data on the tape by listening to it. The reference tone is a steady, medium-pitched note while actual data sounds like random noise.

Be sure to adjust the cassette recorder playback volume before using the memory READ command and do not press the (RETURN) key too hastily, let the tape play for three or four seconds first to let it comes to steady.

Note that when using the READ commad, the range of memory contents on tape does not need not to be read back to the same range of locations where it has been saved; however, the amount of data read must be equal in LENGTH to the previous memory WRITE. For example, if you save the data in HGR1's memory on tape first:

*2000.3FFFW

Later, you can use either:
*2000.3FFFR (RETURN) to load them back to HGR1, or use
*4000.5FFFR (RETURN) to load them back to HGR2, but
*4000.4FFFR (RETURN) will not work correctly, and an error message will be displayed on the screen.

OTHER KERNEL COMMANDS

In previous sections, we have talked about the frequently used Kernel commands. Now, it comes to the lesser used ones.

DIVERT OUTPUT FROM SCREEN

Normally, all outputs from the computer are sent to the screen. If you want to change that (e.g. divert all output to a printer), you can type:
n(CTRL)P (RETURN)
where n is a number from 0 to 7.

When using this command, be sure that beginning at $Cn00, there is a control program to receive and control the outputs from the computer, which may be in the form of a printer driver, or a RS–232 firmware, etc. Otherwise, your computer will lock up and the only way to recover from this condition is to press RESET.

With a stand alone computer main unit, you can type 1(CTRL)P (RETURN) to send characters to a printer. To select the computer's screen as the console output device, type 0(CTRL)P, followed with a (RETURN).

DIVERT INPUT FROM KEYBOARD

As in the case of output, you can use the Kernel command n(CTRL)K to accept input from a device other than the computer keyboard. To return control to the keyboard, use 0(CTRL)K.

Again, make sure that in $Cn00, there is a program to handle the input to the computer before using this command.

THE GO COMMAND

The GO command can be used to transfer control of the computer to a machine language program at an address you specify. The format of this command is:
(address)G (RETURN)

For example, E000G, followed with a RETURN, instructs the Kernel to jump to address $E000 in memory and pass program control to the machine language instruction located there (in this case, it is a BASIC cold start).

THE USR COMMAND

If you type (CTRL)Y in the Kernel, the 6502A CPU will jump to location $3F8 to start executing instructions. There is enough room at location $3F8 for one machine language jump insturction. If you have a special machine language program somewhere in memory, (CTRL)Y could initiate a jump to it via location $3F8.

The example below shows how to set up (CTRL)Y to start booting the disk drive without typing the familiar 6(CTRL)P command.

*3F8:4C 00 C6 (RETURN)
Now try (CTRL)Y (RETURN) and you should see the drive boot up.

SETTING DISPLAY MODES

To convert all the computer's output on the screen in inverse video, enter the inverse video command I (RETURN). This will reverse the character and background colours.

To recover from inverse video, enter the normal video command N (RETURN).

LEAVING THE KERNEL

There are three ways to leave the Kernel and put you back in BASIC:

(1) (CTRL)B (RETURN) puts you back in BASIC with a cold start(i.e. any program or variables that you had previously in BASIC will be lost).

(2) (CTRL)C (RETURN) puts you back in BASIC with your program and variables remains intact, i.e. a warm start.

(3) 3DOG (RETURN) returns you back to the disk operating system you were using, with your program and variables intact.

SUMMARY OF KERNEL COMMANDS

Examining memory
(adrs)
Displays the contents in (adrs).
(RETURN)
Displays one line of memory contents of the locations following the last examined location.

Altering the contents of memory
(adrs) : (val) (val) . . . . .
Stores the values specified into the consecutive memory locations starting at (adrs).

Moving and comparing the contents of memory
(dest) ((start). (end)M
Copies the contents in the range (start). (end) into another range starting at (dest).
(dest) ((start). (end)V
Compares the contents in the range (start). (end) to another range beginning at (dest).

Saving and loading information via cassette tape
(start) . (end)W
Writes the contents within the memory range (start) . (end) onto the tape.
(start) . (end)R
Reads the values stored on tape, placing them into the memory locations beginning at (start) and ending at (end).

Running machine language programs
(adrs)G
Transfer program control to the machine language program beginning at (adrs).
(CTRL)Y
Transfer program control to the machine language program beginning at memory location $3F8.

Divert input and output
(n) (CTRL)P
n ranges from 0 through 7. When n=0, the output device is assigned to the screen. When n is not 0, the output control is passed to a routine beginning at $Cn00. The new output device receives characters via this routine.
(n) (CTRL)K
Similiar to (n) (CTRL)P, but this time, input control is diverted. When n=0, the input device is assigned to the keyboard.

Change display mode
I
Set inverse display mode.
N
Set normal display mode.

Enter or re-enter BASIC
(CTRL)B
Enter BASIC with a cold start (i.e. erase the current program and data)
(CTRL)C
Re-enter BASIC with a warm start (i.e. let the current program and data intact).

CHAPTER 8

MACHINE LANGUAGE SUBROUTINES

- Exploring the kernel
- Special locations used by the kernel
- Important kernel routines

EXPLORING THE KERNEL

In this chapter, we will explore the Kernel of the computer revealing the functions of its important subroutines, and explaining the significance of some special locations used by it. This serves two main purposes. First, machine language programmers can find useful Kernel subroutines to be called for their own application programs. Secondly, interested users who want to know more about the internal working mechanism of their computer can retrieve the necessary relevant information.

SPECIAL LOCATIONS USED BY THE KERNEL

$20 : WNDLFT

This location holds the leftmost column position in the text window. In 40 column text mode, this number ranges from 0 to 39 ($27) while in 80 column text mode, it ranges from 0 to 79 ($4F). Normally, this location stores 0 for the extreme left side of the screen.

$21 : WNDWTH

This location holds the width, in columns, of the text window. It ranges from 1 to 40 in 40 column text mode and from 1 to 80 in 80 column text mode. Normally, this location stores either 40 (in 40 column text mode) or 80 (in 80 column text mode).

$22 : WNDTOP

This location stores the number of the top most line of the text window. Its range is from 0 to 22. Normally, it contains 0, indicating the topmost line of the screen.

$23 : WNDBTM

This location stores the number of the bottom most line of the screen plus one. Its range is from 1 to 24. Normally, it contains 24 (i.e. the bottom most line of the screen).

Changing the contents in the above locations can alter the format of the text window. One thing you should notice is that : in many cases, when you alter the value of WNDLFT, you have to alter the value in WNDWTH in order to make sure that (WNDLFT + WNDWTH) is smaller than 40 in 40 column mode; and smaller than 80 in 80 column mode.

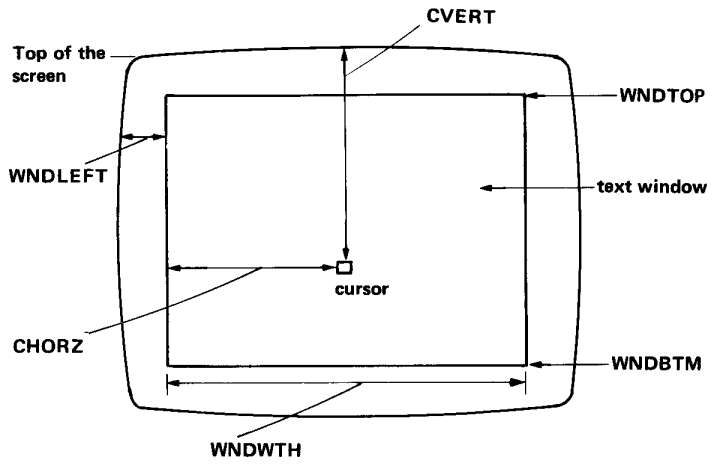
$24 : CHORZ

This location holds the displacement from WNDLFT to the cursor position (i.e. the position where the next output character will be placed on screen). It ranges from 0 to (WNDWTH–1).

$25 : CVERT

This location holds the displacement from THE TOP OF THE SCREEN (not WNDTOP) to the cursor position. It ranges from 0 to (WNDBTM–1).

Figure 1 helps you to understand the meanings of the memory locations mentioned above.



$32 : INVFLG

This is a mask byte used by both the KERNEL and the BASIC interpreter to cause characters displayed on screen to appear as normal (INVFLG=$FF), reversed (INVFLG=$3F), or blinking (INVFLG=$7F) characters. Its content is normally $FF.

When Kernel writes a character onto the screen; it performs a bit to bit AND operation between this flag and the character code of the character to be displayed (whose MSB=1).

When BASIC writes a character, it performs a bit to bit OR between the character code and another flag called FLASH first. The ORed result is then sent to the Kernel's display routine, where the result will be further ANDed with INVFLG. The FLASH flag contains $00 for normal character display and holds $40 for blinking display.

$33 : PROMPT

This location stores the ASCII code (with MSB=1) of the prompt character which is to be displayed onto the screen whenever the computer wants a input line. In BASIC, it contains $DD (]) while in Kernel, it contains $AA (*).

$36, $37 : OUTSWL, OUTSWH

These two locations contains the start address of the routine which is to receive and output characters to a output medium (which may be the computer's screen or may be printers). During initialization, this pointer pair is set to point to MCOUT1 to send output characters to the screen. Either the BASIC command "PR#n" or the Kernel command "n(CTRL)P" sets OUTSWL to $00, OUTSWH to $Cn. The routine at $Cn00 will then receive (in A—reg) each byte 'written' through MCOUT, which is a Jump Indirect instruction : JMP (OUTSWL).

$38, $39 : INSWL, INSWH

These two locations contains the start address of the routine which is to input a character to the computer. On power up, they are set to point to INKEY which gets inputs from the keyboard. The BASIC command "IN#n" or the Kernel command "n(CTRL)K" sets INSWL to $00, INSWH to $Cn. The routine at $Cn00 is then called any time the Kernel or the executing program asks for an input (by calling MRDKEY or one of the subroutines which in turn calls MRDKEY).

$C5–$C8 : PBANK1–PBANK4

In the computer the 256K physical addressing space is divided into 16 windows (or blocks), with each window containing 16K bytes. The addressing capacity of the 6502 CPU, as you probably know, is 64K bytes. So, at any time, no more than four memory windows can be linked to the CPU. Four I/0 locations : SBANK1 ($C07C), SBANK2 ($C07D), SBANK3 ($C07E), and SBANK4 ($C07F) are responsible for these linkages (or what we call – mappings).

By storing different values into these I/O locations, we can control which 16K logical memory of the 6502 corresponds to which 16K physical memory block. For example, when powered up, a memory configuration of 0, 1, 3, F is assigned to the computer: the lowest 16K logical memory ($0000–$3FFF) of 6502 is mapped to the first 16K physical RAM block ($00000–$03FFF); and the highest 16K logical memory space ($C000–$FFFF) of the 6502 is mapped to the last 16K physical memory space ($3C000–$3FFFF) which contains ROM and I/Os.

Storing different values in these I/O locations can change the memory configuration. But once a change has been made, there is, unfortunately, no way to read it back as these I/O locations are WRITE ONLY (i.e. nonreadable). Many Kernel subroutines MUST know the memory configuration at every moment in order to carry out their functions correctly. So, whenever you make a change in the memory mappings, you must inform the Kernel by recording the change in certain memory locations.

These memory locations are: PBANK1 ($C5), PBANK2 ($C6), PBANK3 ($C7), and PBANK4 ($C8). They contain the latest values which have been sent to the four I/O locations – SBANK1 to SBANK4.

For example, then you want to change the memory configuration of your computer from 0, 1, 3, F to, say, 0, 2, 3, F. You should send $02 to SBANK2 ($C07D), as well as PBANK2 ($C6). Failure to place $02 at PBANK2 will invalidate your change in SBANK2.

The reason for the invalidation can be explained as follows. When you store $02 to SBANK2, the memory configuration indeed changes to 0, 2, 3, F immediately. However, when a character is to be written out to the screen later on through the standard Kernel subroutine MCOUT1, something occurs. The text memory area of the computer resides in the first 16K block of the physical memory. When a character is to be sent to the text memory area, an instruction like STA or STA $1XXX may not work at some cases. This is because the above instructions assume that RAM block 1 is mapped to the lowest 16K logical addressing space of the 6502A. This assumption can be invalid due to the existance of SBANK1. So, the Kernel does its task through a harder way. It first maps RAM block 1 to the second 16K logical addressing space ($4000–$7FFF) of the 6502 by sending $00 to SBANK2. This causes the original physical memory block there (how about call it B from now on?) to be ticked out. After RAM block 1 has been taken in, the displayed character is written into a text memory location by means of an instruction like STA $44XX or STA $5XXX. Once finished, RAM block 1 should be removed in order to free the 16K logical addressing space to the original physical memory block B. This can be done by storing another value

into SBANK2 but how can the Kernel subroutines know who is memory block B? The answer is to read PBANK2. Since you haven't changed this location, it still contains one. So, RAM block 1 is brought into $4000–$7FFF, instead of RAM block 2. This explains why you were fail in changing the memory configuration from 0, 1, 3, F to 0, 2, 3, F.

$57B : TXTMOD

In the computer the display RAMs of 40 column text reside in the physical memory range $00400–$007FF while the display RAMs of 80 column text are in the physical memory range $01000–$017FF. Since writing a character on screen is actually a process of storing the character code into one of the display RAM locations, it becomes obvious that there must be a way to inform the Kernel subroutines about what the current text mode is – 40 column or 80 column? Otherwise, these subroutines do not know where to release the character codes – into $00400–$007FF or into $01000–$017FF?

The solution adopted in the computer is to store the information into the memory location $57B (TXTMOD). If 40 column text mode is now being activated, TXTMOD contains $04. If, on the other hand, the present text mode is 80 column, then TXTMOD should contain $10.

You should be aware of the importance of TXTMOD. Don't disturb it or you will see nothing (or something peculiar) on the screen. It might be disastrous and in some cases, if you are unlucky, it will destroy your programs. Even pressing the (RESET) button may not recover.

$3F0, $3F1 : BRKVER

When a BRK instruction is encountered during the execution of a program, the 6502A CPU jumps to the BRK interrupt handler within the Kernel. All reigsters are stored into a set of zero page memory locations : $45 stores X; $47 stores Y; $48 stores P and $49 stores SP. The program counter stacked during the BRK interrupt process is also POPPED out from the stack. Its high order 8 bits are then stored into $3B and the low order 8 bits are stored into $3A. Finally, a jump indirect through BRKVER is executed. On system power up, BRKVER is initiated to contain the address of BREAK1 (which is a routine used to print the address where the BRK instruction was found).

If you have booted the disk operating system into the computer, you should alter the contents in BRKVER ($3F0). Otherwise, instead of showing the message "!BREAK AT $XXXX", the machine will be dead locked when it encounters a BRK instruction.

$3F2, $3F3 : RESTVR

This is the RESET VECTOR used by the Kernel for soft entry (i.e. the reset is caused by pressing the (RESET) button AFTER the machine has been powered up). For more details, please read the description of RESET ($FA62) in the next portion of this chapter.

$3F4

This is the powered-up indicator. Its contents are to be Exclusive ORed with the contents of $3F3 in the RESET handler ($FA62). Depending on the result (i.e. whether it is $A5 or not), either a jump indirect via RESTVR or the power up initialization will be performed. Again, for more details, read the RESET routine.

$3FB–$3FD : NMIADR

When a non-maskable interrupt (NMI) occurs in the computer the 6502 CPU jumps to NMIADR, where a JMP instruction should be placed to pass the control to the user provided NMI handler.

$3FE, $3FF : IRQVER

In the computer, the handling of an IRQ interrupt is as follows:

The contents of the A–reg are stored at $45. A Jump Indirect via location IRQVER is executed to pass the control to the user provided IRQ interrupt handler. The interrupt handler has the responsibility of clearing the "I" flag on exit, so as to allow further interrupts.

IMPORTANT KERNEL ROUTINES

In the following decriptions, when we talk about a Kernel subroutine residing at the range $C800–$CFFF, you should be aware that this routine can only be called after you have enabled the expansion ROM of $C3XX. A standard way to do so is shown below:

```
STA $CFFF ; disselect all expansion ROMs
STA $C300 ; select the expansion ROM of $C3XX
JSR $CXXX ; call the Kernel routine within $C800–$CFFF
```

KEYIN : $C84D

This subroutine reads a single character from either the keyboard or the function-key definition area ($00801–$00FFF). On exit, A–reg contains the ASCII code of the input character with MSB set (i.e. bit 7=1). The ASCII code with MSB cleared is stored at location BYTE ($67B). X–reg is unaffected. Y–reg will be destroyed if a function key has been pressed.

On entry, KEYIN first check KEYFLG ($00800) to see whether the input character should come from the keyboard or from the function-key definition area. If from keyboard, it will wait for the user to press a key, meanwhile creating a 16 bit random number seed in RNDNOH ($4F) and RNDNOL ($4E). You can use this 16 bit number as the base of a random number generator in your program.

If, on the other hand, the input character should come from the function keys' definition area, KEYIN will fetch a character from this area, using the pointer pair FKEYPH ($27) and FKEYPL ($26). If the character fetched is found to be the last one of a definition string (e.g. "N" string), then KEYIN will clear KEYFLG before return. So at the next time when KEYIN is called, the character will be read from the keyboard, instead of from the function keys' definition area.

VIDOUT : $C9F9

This subroutine outputs a character to the screen at the current cursor position (remember? The cursor position is defined by the two zero page locations CHORZ and CVERT). If the character is a screen formatting control code, i.e. a non-displayable character, its associated function will be performed, without placing it into a text memory location. On entry, A–reg should contain the ASCII code of the character, with MSB=0. When exiting from VIDOUT, the contents in A–reg and Y–reg may be affected and the contents in X–reg will kept unchanged.

Note that VIDOUT, as well as all other Kernel's subroutines concerning character output on screen, works with no difference in 40 column and 80 text column modes. It knows when to output characters on the 40 column text and when to output characters on the 80 column text (still remember the marvellous location TXTMOD ?).

After placing a displayable character onto the screen, VIDOUT examines the keyboard. If the CTRLS key is pressed, then VIDOUT will stop until you press another key. This feature is useful when you are listing a long program or text on the screen. You can use CTRLS to examine your program or text portion by portion.

The following is a list showing all the screen formatting control characters recognized by VIDOUT, together with their associated functions.

$07 (CTRLG) : sound the bell at 1K Hz for 0.1 second

$08 (CTRLH) : move the cursor to the left one space by decrementing CHORZ. If CHORZ goes negative, it is set to (WNDWTH-1) and CVERT is decremented. If decrementing CVERT would take it above WNDTOP, CVERT will not be changed.

$0A (CTRLJ) : cause a line feed. The cursor is moved down one line unless this would put it on a line below the dislplay window. In this case, all the contents within the display window will be scrolled up one line, and the cursor stays on the current line.

$0B (CTRLK) : clear the text from the cursor position to the end of the screen. The character at the cursor position will also be deleted but the cursor itself will not move.

$0C (CTRLL) : cause a form feed. All information displayed on the screen will be cleared. As for other format control characters, this does not destroy any information stored internally in your computer, but rather simply starts a new screen.

$0D (CTRLM) : cause a carriage return. Notice that here the cursor will be moved to the leftmost column of the current line but its line position will not be changed.

$19 (CTRLY) : home the cursor. This control character causes the cursor to be placed in the first row, first column WITHOUT clearing the screen. Notice that the "HOME" command in BASIC actually sends the FORM FEED ($0C) code to VIDOUT (not the HOME ($19) code).

$1C : This moves the cursor forward one space on the screen without copying the character into the input line buffer (it is not the code sent when you press the RIGHT ARROW key on the keyboard).

$1D : clear to end of line. This operation erases all characters from the current cursor location to the end of the present line. The character at the cursor location will also be deleted but the cursor itself will not move.

$IF : causes reverse linefeed. This moves the cursor up one line without changing its column position. Once the cursor reaches the top of the scroll window, it will not move anymore.

ENCUR : $CBA0

This subroutine, when called, save the character of the current cursor position (defined by CHORZ & CVERT, remember?) into a temporary location TEMPA($6FB). Then, the character code is manipulated to form either a flashing character or an inversed character (which depends on the character itself and what the current text mode is – 40 or 80 column?). The character formed is then placed on the cursor position.

This routine destroys the contents in the Y–reg.

DECUR : $CBDF

This subroutine should be called whenever you want to recover the original character under the current cursor position (i.e. replacing the flashing or inversed character with a normal one). This routine destroys the contents in the Y—reg.

MRDKEY : $FD0C

This subroutine, when called, enables the cursor on screen first (i.e., calls ENCUR). Then it reads a single character through the vector pair INSWL, INSWH ($38, $39) and returns to the caller with the character code in A—reg (MSB=1). It destroys the contents in the Y—reg.

MINKEY : $FD1B

This subroutine, normally pointed to by INSWL and INSWH, reads a single character by calling KEYIN ($C84D). Before exit, it will disable the cursor on screen (i.e., calling DECUR). On return, A—reg contains the character code of the input key, with MSB=1.

MRDCHR : $FD35

This subroutine, besides reading a key by calling MRDKEY, also allows the users to enter the ESCAPE MODE. The escape mode is entered by pressing the (ESCAPE) key once. Twelve keys then have separate meanings. When you press one of them, the function associated with that key is performed. MRDCHR will either continue or terminate the escape mode, depending upon which escape function has been performed. If in the escape mode, you press any key other than the twelve special ones, that keypress will be ignored and you will leave the escape mode.

The twelve characters which have special meanings in escape mode are:

@, A, B, C, D, E, F, G, I, J, K, M

For a through explanation of these escape functions, please refer to the USER MANUAL.

MGETLZ : $FD67

This is the subroutine used to request an input line. When a carriage return is received, it returns to the calling program. On exit, the input line is stored in the input line buffer (i.e. from $200 to $2FF) and X—reg contains the number of characters on that line.

Every time MGETLZ is called, it first sends a carriage return and a line feed code out. Then the prompt character stored in PROMPT ($33) is printed on the screen, telling the user now the computer wants a input line. MRDCHR is called for inputing each character. As you type, the characters are echoed to the screen at the current cursor position and the curosr is then advanced. Meanwhile, the character codes are stored into the line buffer. When a carriage return code is encountered. MGETLZ places it at the end of the buffer, clearing the remainder of the screen line (How to do that? Simple, send $1D to VIDOUT!).

MGETLZ can receive up to 254 characters a line. As it exceeds 249 characters, teh KERNEL will produce a sound for each character follows, as a warning to the user. If you type over 255 characters, the entire line will be cancelled. You can also cancel your input line at any time by pressing (CTRL)X.

MGETLZ allows you to make corrections for your input line by using the two arrow keys (their uses are mentioned in the USER MANUAL).

If you do not want a carriage return sent before inputing a line, you can call MGETLN ($FD6A).

MCOUT : $FDEF

This is the general character output subroutine. Before entry, the character to be output should be placed in A—reg. MCOUT jumps to the actual character output subroutine (which is normally MCOUT1) through the vector pair OUTSWL and OUTSWH.

MCOUT1 : $FDF0

This subroutine, normally entered through MCOUT, is to display the character in A—reg onto the screen, as well as advancing the cursor. If the character is a format-control one (e.g. a carriage return), MCOUT1 will only perform the control function, without storing the character code into a text memory location. This subroutine, unlike VIDOUT, supports inverse character display by carrying out a bit to bit AND function between INVFLG and the character code. Another special feature about this subroutine is that when a carriage return code is received, two control codes : first the return code, then the line feed code, will be sent out to the output device. On exit, the contents in all registers remain unchanged.

SETINV : $FE80

This subroutine stores $3F into INVFLG ($32) so as to turn on inverse text display. It returns with $3F in A—reg.

SETNOR : $FE84

This subroutine stores $FF into INVFLG so as to recover normal text display. It returns with $FF in A—reg.

F8HOME : $FC58

This subroutine clears the scroll window entirely and moves the cursor to the top-left hand corner (i.e. the HOME position) of the window. You should always use this subroutine to clear the screen. Don't clear the screen by storing $A0 into $400—$7FF (or $1000—$17FF in 80 column text) as this would erase the contents stored in TXTMOD ($57B).

CROUT : $FD8E

This subroitine sends a carriage return to the current output device through MCOUT. On exit, the content in A—reg equals $8D.

PRBYTE : $FDDA

This subroutine prints the contents in A—reg as two hexadecimal digits to the current output device — the high nibble being the first digit and the low nibble being the second digit. The contents in A—reg will be changed.

PRHEX : $FDE3

This prints the contents in A—reg as a single hexadecimal digit. The contents in A—reg should be within the range : $00—$0F. On exit, the contents of A—reg will be changed.

PRNYX : $F940

This subroutine prints the contents in Y—reg and X—reg as four hexadecimal digits. The contents in Y—reg are printed as the two most signigicant digits. The contents in A—reg will be changed on return.

PRNAX : $F941

This subroutine prints the contents in A—reg and X—reg as four hexadecimal digits. The contents in A—reg are printed as the two most significant digits. Again, the contents in A—reg will be destroyed.

PRNX : $F944

The content in X—reg is printed as two hexadecimal digits. Again, it destroys the contents in A—reg.

PRNSPC : $F948

This subroutine outputs three spaces to the current output device. The contents in A—reg will be set to $A0 if the current output device is the computer's screen.

MBELL : $FF3A

This subroutine asks the current output device to beep. If the current output device is the screen, a beep at 1K Hz for 0.1 second will be output from the speaker. Contents in A—reg will be destroyed.

ERROR : $FF2D

This subroutine prints the word "ERROR" to the current output device and followed by a beep. The contents in A—reg will be destroyed.

MBELL1 : $FFD9

This subroutine produces a beep at the computer's speaker (1KHz for 0.1 second). Contents in A—reg will be destroyed.

AUDOUT : $F80C

This subroutine receives the data stored in A—reg and send it to the sound generator (SN76489).

MWAIT : $FCA8

This subroutine causes a delay for a specific period of time. Approximately, the duration of the delay can be related to the contents in A—reg as follows:
Delay time = $(52*A^2 +27*A+26)/2$
where A = the value stored in A—reg
When MWAIT finishes, A—reg contains zeros.

MPREAD : $FB1E

This subroutine returns in Y—reg a number which represents the position of a paddle. This number ranges from 0 to 255 ($FF). Contents in A—reg are destroyed. Before entering here, X—reg should already be set up to contain the paddle number (from 0 to 3).

SAVE : $FF4A

This subroutine can be called to save the contents of the 6502A's registers into a set of zero page locations. The contents in A and X will be changed and the decimal mode will be cleared. The memory locations used to store these contents are:

$45 : stores value in A
$46 : stores value in X
$47 : stores value in Y
$48 : stores value in the status register
$49 : stores value in the stack pointer

GETBRG : $FF3F

This subroutine loads the contents of the 6502's registers back from the memory locations mentioned above.

MOVE : $FE2C

This subroutine is used by the Kernel's "M" command. The contents of memory from the source starting location (pointed to by REG1L and REG1H, i.e. $3C, $3D) through the source ending location (pointed to by REG2L and REG2H, i.e. $3E, $3F) are moved to the memory range beginning at the location pointed to by REG4L and REG4H (i.e. $42, $43). Y—reg must contain zero on entry. Contents in A—reg will be destroyed.

MTSAVE : $FECD

This subroutine is used by the kernel "W" command. A block of memory contents is written out to the cassette tape. The starting address should be stored in REG1L,H ($3C, $3D) and the ending address should be stored in REG2L,H ($3E, $3F). On return, the contents in A, X, Y will all be changed.

MTLOAD : $FEFD

This subroutine is used by the Kernel's "R" command. A block of data is read from the cassette tape. These data are then stored into a memory range. Before entering this routine, the starting address of this memory range should have already been set up in REG1L and REG1H ($3C, $3D), and the ending address should be in REG2L, H ($3E, $3F). Again, the contents in A, X, and Y will be changed on exit.

TEXT80 : $F809

This subroutine when called, turns on the 80 column text display by referencing the appropriate soft switches. Moreover, the contents in TXTMOD are set to $10, which tells all the Kernel's dislplay subroutines to send characters to the 80 column text screen.

The usual way to initiate 80 column text dislplay is shown below:

```
JSR TEXT80 ; turn on the 80 column text
LDA #80 ; adjust the width of the scroll window
STA WNDWTH ;#80 for 80 columns
JMP F8HOME ; clear the screen and home the cursor position
```

TEXT40 : $F806

This subroutine can be used to turn back to the 40 column text display. Again, it references the appropriate soft switches and stores a new value (this time $04) into TXTMOD.

The usual way to initiate 40 column text is as follows:

```
JSR TEXT40 ; turn on the 40 column text
LDA #40 ;adjust the width of the scroll window
STA WNDWTH ;#40 for 40 columns
JMP F8HOME ;clear the screen and home the cursor position
```

RESET : $FA62

A reset interrupt occurs in the 6502 CPU when the computer is first powered up or when you press the (RESET) button later. The 6502 responds to this type of interrupt by transfering control to the reset handling routine through a vector pair at $FFFC and $FFFD. In the computer this vector pair contains the address of RESET. So, RESET is the reset handling routine of the computer.

This routine is described here so that the users know what happens during the reset process. It is not intended to be called by your application programs.

Once RESET is entered, the contents in PBANK4 ($C8) are checked. If $OF is found in this location, the routine assumes that the reset is NOT caused by a power up and execution continues at RESET1. Otherwise, a memory configuration of 0, 1, 3, F is initiated. The first 16K logical addressing space of the 6502 is mapped to the first 16K physical RAM block (we call it Block 0); the third 16K logical addressing space is mapped to the fourth 16K physical memory block (which is also a RAM block and we call it Block 3). After that, the Keyboard is read to see whether the user wants any options. If (ESCAPE) is pressed, 80 column text display is selected, instead of 40 column. If A is pressed, BASIC programs start at $801, instead of $1801.

Now, it comes to RESET1. Here, normal text display (characters appear white on black and the background colour is black) is set. The sound generator is turn off. Current input device is assigned to the keyboard and current output device is assigned to the screen.The cause of the reset is checked again. This time we check the contents of two page three memory locations, instead of PBANK4. If the Exclusive ORed of the contents of these two locations ($3F3 and $3F4) is $A5, then a warm start (i.e. the reset is caused by pressing the (reset) button after power up) is assumed. The 6502 goes to another memory location (whose address is stored in $3F2, $3F3) to continue execution.

If, on the other hand, the result of the Exclusive OR operation is not $A5, then cold start (i.e. the reset is caused by a power up) is assumed. In this case, the logo of the machine is printed on the first few lines of the screen. Next, the RESET and BREAK vectors in page three are established to point to the appropriate addresses (e.g. the BREAK vector points to BREAK1). After this, the routine goes to check if there is a disk controller plugged into the main unit. If there is one, the control is passed to the controller to boot the disk. If no controller is found, the RESET vector at page 3 is changed to $E003 (BASIC warm start entry) and the reset handling routine terminates by jumping to $E000 (BASIC cold start entry).

APPENDIX A

KERNEL LISTING

```
                        ;***********************************************
                        ;
                        ;       SYSTEM KERNEL
                        ;
                        ;       (C) COPYRIGHT
                        ;       1984 :
                        ;       V.T.L.
                        ;
                        ;***********************************************

                                .6502
                        ;
                        ; ZERO PAGE EQUATES
                        ;
0020                    WNDLFT  EQU     $20
0021                    WNDWTH  EQU     $21
0022                    WNDTOP  EQU     $22
0023                    WNDBTM  EQU     $23
0024                    CHORZ   EQU     $24
0025                    CVERT   EQU     $25
0026                    FKEYPL  EQU     $26
0027                    FKEYPH  EQU     $27
0028                    SBASL   EQU     $28
0029                    SBASH   EQU     $29
002A                    SBAS2L  EQU     $2A
002B                    SBAS2H  EQU     $2B
002E                    CHKSUM  EQU     $2E
002F                    OPCODL  EQU     $2F
002F                    LASTBI  EQU     $2F
0031                    STOFLG  EQU     $31
0032                    INVFLG  EQU     $32
0033                    PROMPT  EQU     $33
0034                    SAVEX   EQU     $34
0035                    SAVEY   EQU     $35
0036                    OUTSWL  EQU     $36
0037                    OUTSWH  EQU     $37
0038                    INSWL   EQU     $38
0039                    INSWH   EQU     $39
003A                    PCL     EQU     $3A
003B                    PCH     EQU     $3B
003C                    REG1L   EQU     $3C
003D                    REG1H   EQU     $3D
003E                    REG2L   EQU     $3E
003F                    REG2H   EQU     $3F
0042                    REG4L   EQU     $42
0043                    REG4H   EQU     $43
0045                    ACC     EQU     $45
0046                    REGX    EQU     $46
0047                    REGY    EQU     $47
0048                    STATUS  EQU     $48
0049                    STACKP  EQU     $49
004E                    RNDNOL  EQU     $4E
004F                    RNDNOH  EQU     $4F
0067                    TXTABL  EQU     $67
```

```
0068                    TXTABH  EQU     $68
00C5                    PBANK1  EQU     $C5
00C6                    PBANK2  EQU     $C6
00C7                    PBANK3  EQU     $C7
00C8                    PBANK4  EQU     $C8
                        ;
                        ;
                        ;
                        ; SLOT 0 EQUATES
                        ;
0778                    SAVE1   EQU     $778
                        ;
                        ;
                        ;
                        ; SLOT 3 EQUATES
                        ;
04FB                    TEMPY   EQU     $4FB
057B                    TXTMOD  EQU     $57B
05FB                    TEMPX   EQU     $5FB
067B                    BYTE    EQU     $67B
06FB                    TEMPA   EQU     $6FB
077B                    POWER   EQU     $77B
047B                    CHWHO   EQU     $47B
07FB                    CVWHO   EQU     $7FB
                        ;
                        ;
                        ; OTHER RAM LOC. EQUATES
                        ;
0100                    STACK   EQU     $100
0200                    KEYBUF  EQU     $200
03F0                    BRKVER  EQU     $3F0
03F2                    RESTVR  EQU     $3F2
03F8                    USRADR  EQU     $3F8
03FB                    NMIADR  EQU     $3FB
03FE                    IROVER  EQU     $3FE
4800                    KEYFLG  EQU     $4800
                        ;
                        ;
                        ; ROM EQUATES
                        ;
E000                    BASICC  EQU     $E000
E003                    BASICW  EQU     $E003
F1BD                    HRSEXT  EQU     $F1BD
F229                    RENEW   EQU     $F229
F23C                    NORMAL  EQU     $F23C
                        ;
                        ;
                        ; I/O EQUATES
                        ;
C000                    KEYBRD  EQU     $C000
C010                    KEYSTR  EQU     $C010
C008                    BKDROP  EQU     $C008
C018                    BKGRND  EQU     $C018
C020                    TAPEOU  EQU     $C020
C028                    TEXTCR  EQU     $C028
C030                    SPEAKR  EQU     $C030
```

```
C04C                    VZTX40  EQU     $C04C
C04F                    VZTX80  EQU     $C04F
C051                    VZTEXT  EQU     $C051
C054                    VZPAG1  EQU     $C054
C056                    VZSELF  EQU     $C056
C060                    TAPEIN  EQU     $C060
C064                    PADDL0  EQU     $C064
C068                    SONGEN  EQU     $C068
C070                    PDLRES  EQU     $C070
C078                    SYSTEM  EQU     $C078
C07C                    SBANK1  EQU     $C07C
C07D                    SBANK2  EQU     $C07D
C07E                    SBANK3  EQU     $C07E
C07F                    SBANK4  EQU     $C07F
C1C2                    HORZSC  EQU     $C1C2
C1C3                    VERTSC  EQU     $C1C3
C1C5                    TWOMHZ  EQU     $C1C5
CFFF                    ROMCLR  EQU     $CFFF
                        ;
                        ;
                                ORG     $300
                                .PHASE  $C300
                        ;
                        ;SOME SIGNITURE BYTES EXIST HERE
                        ;THEY ARE RECOGNIZED BY CP/M AND PASCAL
                        ;REMARKS:
                        ;CP/M AND PASCAL DISTINGUISH DEVICES BY
                        ;CHECKING CN05 AND CN07
                        ;
                        ;
                        ;
C300    2C C30F                 BIT     IORTS   ;ENTER HERE AT THE FIRST TIME
C303    70 04                   BVS     ENTER   ;BRANCH ALWAYS
C305    38              INENT   SEC             ;FROM SECOND TIME ON
C306    90                      DB      $90
C307    18              OUTENT  CLC             ;FROM SECOND TIME ON
C308    B8                      CLV
C309    8D CFFF         ENTER   STA     ROMCLR  ;BRING C800 IN
C30C    20 C310                 JSR     ROAD    ;MUST GO THROUGH THIS WAY
C30F    60              IORTS   RTS
                        ;
                        ;
C310    48              ROAD    PHA             ;SAVE EVERYTHING
C311    8A                      TXA
C312    48                      PHA
C313    98                      TYA
C314    48                      PHA
C315    08                      PHP             ;INCLUDING STATUS
C316    BA                      TSX             ;USED TO GET A FROM STACK
C317    BD 0104                 LDA     STACK+4,X
C31A    28                      PLP             ;RECOVER STATUS
C31B    48                      PHA             ;SAVE CHARACTER
C31C    70 03                   BVS     *+5     ;FIRST TIME?
C31E    4C CC30                 JMP     IO      ;NO

C321    AD FB4A                 LDA     SIGNAT  ;WHO ARE YOU?
```

```
C324  D0 2A           BNE   WHO      ;VISITOR?
C326  20 C801         JSR   SETUP
C329  A9 00           LDA   #$00     ;BASIC, NOT
C32B  8D 077B         STA   POWER    ;PASCAL OR CP/M
C32E  A9 02           LDA   #2       ;INFORM BASIC THE CHANGE
C330  85 C5           STA   PBANK1
C332  A9 01           LDA   #1
C334  85 C6           STA   PBANK2
C336  A9 C3           LDA   #)OUTMED;FORM MEDIA
C338  85 39           STA   INSWH
C33A  85 37           STA   OUTSWH
C33C  A9 4D           LDA   #(INMED
C33E  85 38           STA   INSWL
C340  A9 4A           LDA   #(OUTMED
C342  85 36           STA   OUTSWL
C344  68              PLA             ;RELEASE STACK
C345  68              PLA
C346  A8              TAY
C347  68              PLA
C348  AA              TAX
C349  68              PLA             ;GET BACK CHARACTER
                      ;
                      ;
C34A  4C FDF0   OUTMED JMP   MCOUT1   ;MEDIA ONLY
C34D  4C FD1B   INMED  JMP   MINKEY
                      ;
                      ;
                      ;
C350  20 CCCA   WHO    JSR   TUGGLE   ;DO INITIALIZATION
C353  20 C801          JSR   SETUP
C356  20 CCCA          JSR   TUGGLE
C359  A9 C3            LDA   #)INENT
C35B  85 39            STA   INSWH
C35D  85 37            STA   OUTSWH
C35F  A9 05            LDA   #(INENT
C361  85 38            STA   INSWL
C363  A9 07            LDA   #(OUTENT
C365  85 36            STA   OUTSWL
C367  18               CLC             ;THEN DO OUTPUT
C368  4C CC30          JMP   IO
                      ;
                      ;
                       PAGE
```

```
                        .DEPHASE
                        ORG     $36E
                        .PHASE  $C36E
                    ;
                    ;
                    ;*****************************************
                    ;*** FOR RS232 INTERFACE USAGE          ***
                    ;*** ICHRDIS: DISPLAY CHARACTER ON SCREEN***
                    ;*** IENCUR: TURN ON CURSOR            ***
                    ;*** DON'T RELOCATE THE FOLLOWING CODES ***
                    ;*****************************************
                    ;
                    ;
C36E    08          ICHRDIS PHP             ;SAVE STATUS
C36F    8D CFFF             STA     ROMCLR  ;ENABLE C8
C372    20 CB54             JSR     CHRDIS  ;DISPLAY IT
C375    28                  PLP             ;RESUME STATUS
C376    60                  RTS
                    ;
                    ;
C377    08          IENCUR  PHP
C378    8D CFFF             STA     ROMCLR  ;ENABLE C8
C37B    20 CBA0             JSR     ENCUR   ;TURN CURSOR ON
C37E    28                  PLP
C37F    60                  RTS
                    ;
                    ;
                        PAGE
```

```
                        ;
                                .DEPHASE
                                ORG     $800
                                .PHASE  $C800
                        ;
C800                            DS      $800,$FF
                        ;
                        ;
                                .DEPHASE
                                ORG     $800
                                .PHASE  $C800
                        ;
C800    60                      RTS
                        ;
C801    20 C92D         SETUP   JSR     WBANK   ;REFORM MEMORY ASSIGNMENT
C804    20 C822                 JSR     SETWND  ;SET UP SCREEN SIZE
C807    A9 90                   LDA     #$90
C809    8D 077B                 STA     POWER   ;SETUP ONCE IS ENOUGH
C80C    A9 A0                   LDA     #$A0    ;CLEAR TEMPA
C80E    8D 06FB                 STA     TEMPA
C811    2C C1C3                 BIT     VERTSC  ;CHANGE SCREEN DURING THE
C814    30 FB                   BMI     *-3     ;VERTICAL RETRACE PERIOD
C816    AD C056                 LDA     VZSELF  ;TURN TO
C819    AD C051                 LDA     VZTEXT  ;80 COLUMN TEXT DISPLAY
C81C    AD C04F                 LDA     VZTX80
C81F    4C CA76                 JMP     CLSCRN  ;CLEAR SCREEN
                        ;
                        ;
C822    A9 00           SETWND  LDA     #0      ;FULL SCREEN SIZE: 24*80
C824    85 22                   STA     WNDTOP
C826    85 20                   STA     WNDLFT
C828    A9 18                   LDA     #24
C82A    85 23                   STA     WNDBTM
C82C    A9 50                   LDA     #80
C82E    85 21                   STA     WNDWTH
C830    A9 10                   LDA     #$10
C832    8D 057B                 STA     TXTMOD
C835    60                      RTS
                        ;
                        ;
C836    08              COMPAT  PHP             ;ALWAYS BE SAFE
C837    48                      PHA
C838    A5 29                   LDA     SBASH
C83A    49 40                   EOR     #$40
C83C    85 29                   STA     SBASH
C83E    68                      PLA
C83F    28                      PLP
C840    60                      RTS
                        ;
                        ;
                                .DEPHASE
                                ORG     $847
                                .PHASE  $C847
                        ;
```

```
                        ;IN#8 ENTRY POINTS
C847    4C CBF6                 JMP     ESCX
C84A    4C CA97                 JMP     CLREOL
                        ;
                        ;
                        ;PASCAL AND CP/M INPUT ENTRY POINT
                        ;
C84D    20 CC0D         KEYIN   JSR     RAMOIN  ;CHECK FUNCTION KEY FLAG
C850    AD 4800                 LDA     KEYFLG
C853    20 CC17                 JSR     RAMOOU
C856    C9 99                   CMP     #$99    ;GET FROM FUNCTION KEY BUFFER?
C858    D0 06                   BNE     KEYINH
C85A    20 C8FE                 JSR     KEYINB  ;YES
C85D    4C C870                 JMP     KEYHAB  ;TO BE CONTINUE
C860    20 C878         KEYINH  JSR     POLKBD  ;IF NO, GET FROM KEYBOARD
C863    C9 9C                   CMP     #$9C    ;TAB?
C865    D0 02                   BNE     *+4
C867    A9 89                   LDA     #$89    ;YES, USE CTRL-I
C869    C9 9B                   CMP     #$9B    ;IS IT ESC?
C86B    D0 03                   BNE     *+5
C86D    20 C88A                 JSR     ESCHK   ;IF YES, GO ON FURTHER
C870    48              KEYHAB  PHA             ;SAVE THE CHARACTER GOT FIRST
C871    29 7F                   AND     #$7F    ;PASCAL LIKES MSB=0
C873    8D 067B                 STA     BYTE
C876    68                      PLA             ;RECOVER CHARACTER WITH MSB=1
C877    60                      RTS             ;FOR BASIC
                        ;
C878    AD C000         POLKBD  LDA     KEYBRD  ;POLL KEYBOARD UNTILL KEY GOT
C87B    30 09                   BMI     POLRTS
C87D    E6 4E                   INC     RNDNOL  ;MEANWHILE CREATE A RANDOM NO.
C87F    D0 F7                   BNE     POLKBD
C881    E6 4F                   INC     RNDNOH
C883    4C C878                 JMP     POLKBD
C886    2C C010         POLRTS  BIT     KEYSTR  ;KEY GOT, CLEAR KEYBOARD
C889    60                      RTS
                        ;
                        ;
C88A    A9 04           ESCHK   LDA     #4      ;SET UP TIMER
C88C    85 4F                   STA     RNDNOH
C88E    AD C000         ESCHK1  LDA     KEYBRD  ;ANY KEY FOLLOWING THE ESC?
C891    30 0B                   BMI     ESCHK2
C893    E6 4E                   INC     RNDNOL  ;A STRAIGHT TIMER
C895    D0 F7                   BNE     ESCHK1
C897    C6 4F                   DEC     RNDNOH
C899    D0 F3                   BNE     ESCHK1
C89B    A9 9B           ESCKRT  LDA     #$9B    ;TIME IS UP, MUST ESC ONLY
C89D    60                      RTS
                        ;
                        ;
C89E    2C C010         ESCHK2  BIT     KEYSTR  ;CLEAR KEYBOARD
C8A1    C9 C4                   CMP     #$C4    ;UP ARROW?
C8A3    D0 03                   BNE     *+5
C8A5    A9 9F                   LDA     #$9F    ;YES, REPLACE IT WITH A CTRL-KEY
C8A7    60                      RTS             ;AND RETURN
C8A8    C9 B0                   CMP     #$B0    ;0,1,2?
C8AA    90 EF                   BLT     ESCKRT  ;IF NOT, MUST BE ESC ONLY
```

```
C8AC  C9 B3           CMP   #$B3
C8AE  B0 EB           BGE   ESCKRT
C8B0  29 07           AND   #$07
C8B2  0A              ASL   A
C8B3  0A              ASL   A
C8B4  0A              ASL   A
C8B5  85 26           STA   FKEYPL  ;PREPARE COUNTER
C8B7  20 C878         JSR   POLKBD
C8BA  C9 B0           CMP   #$B0    ;0 TO 7?
C8BC  90 DD           BLT   ESCKRT
C8BE  C9 B8           CMP   #$B8
C8C0  B0 D9           BGE   ESCKRT
C8C2  29 07           AND   #$07
C8C4  05 26           ORA   FKEYPL
C8C6  85 2A           STA   SBAS2L  ;COUNTER COMPLETED
C8C8  85 2B           STA   SBAS2H  ;THIS IS FOR IN#8

C8CA  A9 00           LDA   #$00    ;SET UP POINTER NOW
C8CC  A8              TAY
C8CD  85 26           STA   FKEYPL
C8CF  A9 48           LDA   #$48
C8D1  85 27           STA   FKEYPH  ;FUNCTION KEY STORED FROM $4800
C8D3  20 CC0D         JSR   RAMOIN
C8D6  C6 2A     FKFND1 DEC  SBAS2L
C8D8  30 0C           BMI   FKFND4  ;REACH?
C8DA  E6 26     FKFND2 INC  FKEYPL  ;IF NOT, INCREMENT POINTER
C8DC  D0 02           BNE   FKFND3
C8DE  E6 27           INC   FKEYPH
C8E0  B1 26     FKFND3 LDA  (FKEYPL),Y
C8E2  10 F6           BPL   FKFND2  ;END OF A FUNCTION KEY?
C8E4  30 F0           BMI   FKFND1  ;YES, UPDATE COUNTER
C8E6  AD 4800   FKFND4 LDA  KEYFLG  ;HAS IN#8 HAPPENED?
C8E9  C9 66           CMP   #$66
C8EB  D0 09           BNE   FKFND5  ;NO, GOOD!
C8ED  20 F229         JSR   RENEW   ;YES, GOTO BASIC
C8F0  20 CBA0         JSR   ENCUR
C8F3  4C C84D         JMP   KEYIN
                      ;
C8F6  A9 99     FKFND5 LDA  #$99    ;BEFORE EXIT, MAKE FUNCTION
C8F8  8D 4800         STA   KEYFLG  ;KEY ACTIVE
C8FB  20 CC17         JSR   RAMOOU
                      ;
                      ;
                      ;GET THE FIRST CHARACTER FROM THE FUNCTION KEY BUFFER
                      ;
C8FE  E6 26     KEYINB INC  FKEYPL  ;FUNCTION KEY POINTER READY
C900  D0 02           BNE   *+4
C902  E6 27           INC   FKEYPH
C904  20 CC0D         JSR   RAMOIN  ;MOVE RAM BLOCK 0 IN
C907  A0 00           LDY   #$00
C909  B1 26           LDA   (FKEYPL),Y      ;READ KEY FORM BUFFER
C90B  10 03           BPL   KEYBRT  ;END OF A FUNCTION KEY STRING?
C90D  8C 4800         STY   KEYFLG  ;IF YES, DISABLE FUNCTION KEY
C910  20 CC17   KEYBRT JSR  RAMOOU  ;FUNCTION KEY BUFFER AREA
C913  09 80           ORA   #$80    ;ENSURE MSB=1
```

```
C915  60              RTS
                      ;
                      ;
                      ;
C916  AD C000   VIDWAI  LDA     KEYBRD  ;CHECK STOP LIST
C919  C9 93             CMP     #$93    ;CTRL-S ?
C91B  D0 0F             BNE     VWDONE  ;IF NOT, EXIT
C91D  2C C010           BIT     KEYSTR  ;IF YES, CLEAR KEYBOARD
C920  AD C000   VWLOOP  LDA     KEYBRD  ;WAIT FOR ANOTHER KEY
C923  10 FB             BPL     VWLOOP
C925  C9 83             CMP     #$83    ;IS THE NEXT KEY CTRL-C?
C927  F0 03             BEQ     VWDONE  ;IF YES, DON'T CLEAR IT
C929  2C C010           BIT     KEYSTR  ;OTHERWISE, CLEAR KEYBOARD
C92C  60        VWDONE  RTS
                      ;
                      ;
                      ;
                      ;ROUTINE 'WBANK' IS USED TO CONVERT THE MEMORY
                      ;ASSIGNMENT WHEN RUNNING 80 COLUMN CP/M OR PASCAL
                      ;NEW MEMORY ASSIGNMENT WILL BE 2,1,X,F
                      ;RAM0, WHICH CONTAINS THE SCREEN MEMORY OF THE 80
                      ;COLUMN TEXT, MUST BE OUTSIDE THE VIRTUAL MEMORY
                      ;AREA
                      ;
C92D  A9 02     WBANK   LDA     #2      ;COPY BLOCK 1 INTO RAM 2
C92F  8D C07D           STA     SBANK2
C932  A0 00             LDY     #0      ;SAVE ZERO PAGE FIRST
C934  B9 0000   WBANK1  LDA     $0,Y
C937  99 4000           STA     $4000,Y
C93A  C8                INY
C93B  D0 F7             BNE     WBANK1
                      ;
C93D  84 00             STY     $00     ;NOW WE CAN USE ZERO PAGE
C93F  84 02             STY     $02     ;LOCATIONS
C941  A9 01             LDA     #$01
C943  85 01             STA     $01
C945  A9 41             LDA     #$41
C947  85 03             STA     $03
                      ;
C949  B1 00     WBANK2  LDA     ($00),Y ;NOW FOR THE NON-ZERO PAGE
C94B  91 02             STA     ($02),Y ;REGION
C94D  C8                INY
C94E  D0 F9             BNE     WBANK2
C950  E6 01             INC     $01
C952  E6 03             INC     $03
C954  A5 03             LDA     $03
C956  C9 80             CMP     #$80    ;THE WHOLE 16K FINISHED?
C958  90 EF             BLT     WBANK2  ;IF NOT, CONTINUE
                      ;
C95A  B9 4000   WBANK3  LDA     $4000,Y ;NOW RECOVER THE ZERO PAGE
C95D  99 0000           STA     $0,Y
C960  C8                INY
C961  D0 F7             BNE     WBANK3
                      ;
C963  A9 02             LDA     #2      ;EVERY O.K., SHOT!
C965  8D C07C           STA     SBANK1
```

```
C968  A9 01           LDA   #1
C96A  8D C07D         STA   SBANK2
C96D  60              RTS
                      ;
                      ;
C96E  2D              SUBTBL  DB    LOW BELL-1      ;CTRL-G
C96F  53                      DB    LOW BS-1        ;CTRL-H
C970  16                      DB    LOW VIDRTS-1
C971  A6                      DB    LOW LF-1        ;CTRL-J
C972  7E                      DB    LOW CLREOP-1    ;CTRL-K
C973  75                      DB    LOW CLSCRN-1    ;CTRL-L
C974  28                      DB    LOW CR-1        ;CTRL-M
                      ;
C975  6A                      DB    LOW HOME-1      ;CTRL-Y
C976  16                      DB    LOW VIDRTS-1
C977  16                      DB    LOW VIDRTS-1
C978  1F                      DB    LOW ADVANC-1
C979  96                      DB    LOW CLREOL-1
C97A  4A                      DB    LOW GOTOXY-1
C97B  5D                      DB    LOW UP-1
                      ;
                      ;
                      ;
C97C  75              ESCTBL  DB    LOW CLSCRN-1    ;ESC @
C97D  1F                      DB    LOW ADVANC-1    ;ESC A
C97E  53                      DB    LOW BS-1        ;ESC B
C97F  A6                      DB    LOW LF-1        ;ESC C
C980  5D                      DB    LOW UP-1        ;ESC D
C981  96                      DB    LOW CLREOL-1    ;ESC E
C982  7E                      DB    LOW CLREOP-1    ;ESC F
C983  6A                      DB    LOW HOME-1      ;ESC G
                      ;
                      ;
C984  C4              ESCTB1  DB    $C4     ;ESC I = ESC D
C985  C2                      DB    $C2     ;ESC J = ESC B
C986  C1                      DB    $C1     ;ESC K = ESC A
C987  C8                      DB    $C8     ;ESC L = NOP
C988  C3                      DB    $C3     ;ESC M = ESC C
                      ;
                      ;
                      ;
                      ;TABLE OF SCREEN BASE ADDRESSES (LOW ORDER BYTES ONLY)
C989  00 80           ADRESL  DB    $00,$80
C98B  00 80                   DB    $00,$80
C98D  00 80                   DB    $00,$80
C98F  00 80                   DB    $00,$80
C991  28 A8                   DB    $28,$A8
C993  28 A8                   DB    $28,$A8
C995  28 A8                   DB    $28,$A8
C997  28 A8                   DB    $28,$A8
C999  50 D0                   DB    $50,$D0
C99B  50 D0                   DB    $50,$D0
C99D  50 D0                   DB    $50,$D0
C99F  50 D0                   DB    $50,$D0
                      ;
                      ;
```

```
                                .DEPHASE
                                ORG     $9AA
                                .PHASE  $C9AA
                        ;PASCAL AND CP/M OUTPUT ENTRY POINT
                        ;
C9AA    AD 077B                 LDA     POWER   ;IS IT THE FIRST TIME?
C9AD    29 FC                   AND     #$FC
C9AF    C9 90                   CMP     #$90
C9B1    F0 03                   BEQ     *+5
C9B3    20 C801                 JSR     SETUP   ;INITIALIZE 80 COLUMN DISPLAY
C9B6    20 CBDF                 JSR     DECUR   ;DISABLE THE CURSOR FIRST
C9B9    AD 077B                 LDA     POWER   ;THEN CHECK IF IT IS GOTO XY
C9BC    29 03                   AND     #$03
C9BE    D0 09                   BNE     GOXY    ;2 OR 1 IF IT IS GOTO XY
C9C0    AD 067B                 LDA     BYTE    ;IF NOT, DISPLAY THE CHARACTER
C9C3    20 C9F9                 JSR     VIDOUT
C9C6    4C CBA0         PSCORT  JMP     ENCUR   ;WHEN ALL FINISHED, TURN ON THE
                                                ;CURSOR BEFORE LEAVING
                        ;
                        ;
C9C9    20 C9CF         GOXY    JSR     GOXY1
C9CC    4C C9C6                 JMP     PSCORT
                        ;
C9CF    AD 067B         GOXY1   LDA     BYTE    ;GOTO WHERE?
C9D2    29 7F                   AND     #$7F    ;SAFETY
C9D4    E9 20                   SBC     #$20    ;ASCII TO NUMBER
C9D6    48                      PHA             ;SAVE THIS NO. FIRST
C9D7    CE 077B                 DEC     POWER   ;WHAT IS THIS NO, X OR Y?
C9DA    AD 077B                 LDA     POWER
C9DD    29 03                   AND     #$03
C9DF    D0 13                   BNE     GOTOX
C9E1    68                      PLA             ;IT IS Y!
C9E2    C5 23                   CMP     WNDBTM  ;Y > WINDOW BOTTOM?
C9E4    B0 02                   BGE     POORY   ;IF YES, KEEP CVERT UNCHANGED
C9E6    85 25                   STA     CVERT   ;OTHERWISE, PERFORM GOTOY
C9E8    AD 05FB         POORY   LDA     TEMPX   ;NOW FOR GOTOX
C9EB    C5 21                   CMP     WNDWTH  ;X > WINDOW WIDTH?
C9ED    B0 02                   BGE     POORX   ;IF YES, KEEP CHORZ UNCHANGED
C9EF    85 24                   STA     CHORZ   ;OTHERWISE, PERFORM GOTOX
C9F1    4C CB3F         POORX   JMP     VTAB    ;FINIALISE GOTOXY
                        ;
C9F4    68              GOTOX   PLA             ;THE NO. IS X!
C9F5    8D 05FB                 STA     TEMPX   ;SAVE IT UNTIL Y GOT
C9F8    60                      RTS
                        ;
                        ;
C9F9    C9 20           VIDOUT  CMP     #$20    ;A DISPLAYABLE CHARACTER?
C9FB    90 04                   BLT     VIDOU1
C9FD    09 80                   ORA     #$80    ;YES, THEN SET MSB=1 FISRT
C9FF    30 17                   BMI     STORAD  ;ALWAYS
CA01    C9 07           VIDOU1  CMP     #$07    ;CTRL-@ TO CTRL-F DONT CARE
CA03    90 12                   BLT     VIDRTS
CA05    C9 0E                   CMP     #$0E    ;TAKE CARE OF CTRL-G TO M
CA07    90 06                   BLT     VIDCON
CA09    C9 19                   CMP     #$19    ;CTRL-E TO X ARE DON'T CARES
CA0B    90 0A                   BLT     VIDRTS
```

```
CA0D  E9 0B             SBC   #11       ;PACK THE NO.
CA0F  A8          VIDCON TAY            ;USE Y AS A POINTER TO GET THE
CA10  A9 CA             LDA   #>BELL-1  ;ADDRESSES OF THE CTRL-ROUTINES
CA12  48                PHA
CA13  B9 C967           LDA   SUBTBL-7,Y
CA16  48                PHA
CA17  60          VIDRTS RTS
                  ;
                  ;
CA18  A4 24       STORAD LDY    CHORZ   ;DISPLAY CHARACTER
CA1A  20 CB54           JSR   CHRDIS
CA1D  20 C916           JSR   VIDWAI  ;SEE IF CTRL-S PRESSED
CA20  E6 24       ADVANC INC    CHORZ   ;AND ADVANCE CURSOR
CA22  A5 24             LDA   CHORZ
CA24  C5 21             CMP   WNDWTH  ;CURSOR EXCEEDS SCREEN?
CA26  B0 7C             BGE   CRLF    ;IF YES, LINE FEED
CA28  60                RTS
                  ;
                  ;
CA29  A9 00       CR     LDA   #0      ;CARRIAGE RETURN ONLY
CA2B  85 24             STA   CHORZ
CA2D  60                RTS
                  ;
                  ;
CA2E  A9 C0       BELL   LDA   #$C0    ;BELL THE SPEAKER AT 1KHZ FOR
CA30  85 2A             STA   SBAS2L  ;0.1 SECOND
CA32  38                SEC
CA33  A9 08       BELL1  LDA   #8       ; 8 * 64 =1024/2
CA35  2C C1C2     BELL2  BIT    HORZSC  ;HORIZONTAL SYNC PERIOD=64US
CA38  30 FB             BMI   *-3
CA3A  2C C1C2           BIT   HORZSC
CA3D  10 FB             BPL   *-3
CA3F  E9 01             SBC   #1
CA41  D0 F2             BNE   BELL2
CA43  AD C030           LDA   SPEAKR  ;TOGGLE THE SPEAKER
CA46  C6 2A             DEC   SBAS2L  ;12*16/2=96
CA48  D0 E9             BNE   BELL1
CA4A  60                RTS
                  ;
                  ;
                  ;
CA4B  A9 02       GOTOXY LDA    #$02
CA4D  0D 077B           ORA   POWER
CA50  8D 077B           STA   POWER
CA53  60                RTS
                  ;
                  ;
CA54  C6 24       BS     DEC    CHORZ
CA56  10 BF             BPL   VIDRTS  ;SHOULD NOT EXCEED THE LEFT EDGE
CA58  A5 21             LDA   WNDWTH  ;IF PASS, GO UP ONE LINE
CA5A  85 24             STA   CHORZ
CA5C  C6 24             DEC   CHORZ   ;NATRLAL VS. INTERGER
                  ;
                  ;
CA5E  A5 25       UP     LDA    CVERT
CA60  C5 22             CMP   WNDTOP  ;CURSOR SHOULD NOT OVER THE
```

```
CA62  90 B3           BLT   VIDRTS  ;TOP OF THE SCREEN WINDOW
CA64  F0 B1           BEQ   VIDRTS
CA66  C6 25           DEC   CVERT   ;IF NOT, WE CAN GO UP
CA68  4C CB3F         JMP   VTAB    ;ONE LINE
                      ;
                      ;
CA6B  A5 22     HOME  LDA   WNDTOP  ;TO POSITION 'HOME'
CA6D  85 25           STA   CVERT
CA6F  A9 00           LDA   #0
CA71  85 24           STA   CHORZ
CA73  4C CB3F         JMP   VTAB
                      ;
                      ;
CA76  20 CA6B   CLSCRN JSR  HOME    ;CLEAR THE WHOLE SCREEN
CA79  A5 22           LDA   WNDTOP
CA7B  A0 00           LDY   #0
CA7D  F0 04           BEQ   CLEOP1  ;ALWAYS
                      ;
                      ;
CA7F  A4 24     CLREOP LDY  CHORZ   ;CLEAR TO END OF PAGE
CA81  A5 25           LDA   CVERT
CA83  48        CLEOP1 PHA          ;CLSCRN ENTER HERE
CA84  20 CB41         JSR   ADRCAL  ;CLEAR LINE BY LINE
CA87  20 CA99         JSR   CLEOLZ
CA8A  A0 00           LDY   #0      ;STARTING FROM THE SECOND LINE
CA8C  68              PLA           ;CLEAR FROM THE LEFT EDGE
CA8D  18              CLC
CA8E  69 01           ADC   #1      ;NEXT LINE
CA90  C5 23           CMP   WNDBTM  ;DOWN TO THE BOTTOM LINE?
CA92  90 EF           BLT   CLEOP1
CA94  4C CB3F         JMP   VTAB
                      ;
                      ;
CA97  A4 24     CLREOL LDY  CHORZ   ;CLEAR TO END OF LINE
                      ;
                      ;CLEOP1 ENTER HERE
CA99  A9 A0     CLEOLZ LDA  #$A0    ;CLEAR = FILL WITH SPACE
CA9B  20 CB54   CLEOL2 JSR  CHRDIS  ;DISPALY THE CHARACTER
CA9E  C8              INY
CA9F  C4 21           CPY   WNDWTH  ;REACH THE END OF A LINE?
CAA1  90 F8           BLT   CLEOL2  ;IF NOT, CONTINUE
CAA3  60              RTS
                      ;
                      ;
CAA4  20 CA29   CRLF  JSR   CR      ;CARRIAGE RETURN + LINE FEED
CAA7  E6 25     LF    INC   CVERT
CAA9  A5 25           LDA   CVERT   ;CURSOR SHOULD NOT GO BEYOND
CAAB  C5 23           CMP   WNDBTM  ;THE BOTTOM OF THE SCREEN
CAAD  B0 03           BGE   *+5     ;IF EXCEED, PERFORM SCROLLING
CAAF  4C CB41         JMP   ADRCAL  ;IF NOT, GOOD!
CAB2  C6 25           DEC   CVERT
                      ;
CAB4  A5 21     SCROLL LDA  WNDWTH  ;PREPARE FOR SCROLLING
CAB6  48              PHA           ;SAVE IT FIRST
CAB7  18              CLC
CAB8  65 20           ADC   WNDLFT
```

```
CABA  A8                TAY
CABB  88                DEY                     ;NATRUAL V.S. INTEGER
CABC  84 21             STY     WNDWTH  ;CREATE "NEW" WINDOW WIDTH
CABE  20 CC0D           JSR     RAMOIN  ;GET THE DISPLAY BANK IN
                ;
CAC1  A5 22     SCROL0  LDA     WNDTOP  ;SCROLL TEXT SCREEN
CAC3  48                PHA
CAC4  20 CB41           JSR     ADRCAL  ;CALCULATE A BASE ADDRESS
CAC7  20 C836           JSR     COMPAT
CACA  A5 28     SCROL1  LDA     SBASL
CACC  85 2A             STA     SBAS2L
CACE  A5 29             LDA     SBASH
CAD0  85 2B             STA     SBAS2H
CAD2  68                PLA                     ;GET THE LINE COUNT BACK
CAD3  18                CLC
CAD4  69 01             ADC     #$01
CAD6  C5 23             CMP     WNDBTM  ;REACH THE LAST LINE?
CAD8  B0 57             BGE     SCROL6
CADA  48                PHA                     ;SAVE IT FOR THE NEXT CYCLE
CADB  20 CB41           JSR     ADRCAL  ;CALCULATE THE NEXT BASE ADDRESS
CADE  20 C836           JSR     COMPAT
CAE1  A4 21             LDY     WNDWTH
CAE3  C0 28             CPY     #40     ;40 OR 80 COLUMN MODE?
CAE5  B0 0B             BGE     SCROL3
CAE7  B1 28     SCROL2  LDA     (SBASL),Y       ;MOVE UP ONE LINE
CAE9  91 2A             STA     (SBAS2L),Y
CAEB  88                DEY
CAEC  C4 20             CPY     WNDLFT  ;ONE LINE FINISHED?
CAEE  10 F7             BPL     SCROL2  ;IF NO, CONTINUE
CAF0  30 D8             BMI     SCROL1  ;IF YES, GO FOR THE NEXT LINE
                ;
CAF2  A5 29     SCROL3  LDA     SBASH   ;DO SOME TRANSFORMATION
CAF4  09 04             ORA     #$04    ;FOR 80 COLUMN TEXT MODE
CAF6  85 29             STA     SBASH
CAF8  A5 2B             LDA     SBAS2H
CAFA  09 04             ORA     #$04
CAFC  85 2B             STA     SBAS2H
CAFE  A5 20             LDA     WNDLFT
CB00  38                SEC
CB01  E9 28             SBC     #40
CB03  85 20             STA     WNDLFT
CB05  98                TYA
CB06  38                SEC
CB07  E9 28             SBC     #40
CB09  A8                TAY
                ;
CB0A  B1 28     SCROL4  LDA     (SBASL),Y       ;MOVE UP THE RIGHT
CB0C  91 2A             STA     (SBAS2L),Y      ;HALF PAGE
CB0E  88                DEY
CB0F  C4 20             CPY     WNDLFT  ;ONE LINE FINISHED?
CB11  30 03             BMI     SCROL5  ;IF YES, SKIP
CB13  98                TYA
CB14  10 F4             BPL     SCROL4  ;AN EXACT HALF LINE GONE?
                ;
CB16  A5 29     SCROL5  LDA     SBASH   ;INVERSE TRANSFORM
CB18  49 04             EOR     #$04
```

```
CB1A  85 29           STA   SBASH
CB1C  A5 2B           LDA   SBAS2H
CB1E  49 04           EOR   #$04
CB20  85 2B           STA   SBAS2H
CB22  A0 27           LDY   #39
CB24  A5 20           LDA   WNDLFT
CB26  18              CLC
CB27  69 28           ADC   #40
CB29  85 20           STA   WNDLFT
CB2B  C9 28           CMP   #40
CB2D  B0 9B           BGE   SCROL1  ;IF LEFT EDGE )=40
                                    ;WE HAVE FINISHED
CB2F  90 B6           BLT   SCROL2  ;IF LEFT EDGE (40,
                ;
CB31  20 C836   SCROL6  JSR   COMPAT
CB34  20 CC17         JSR   RAMOOU  ;TICK DISPLAY BANK OUT
CB37  68              PLA
CB38  85 21           STA   WNDWTH  ;RECOVER WINDOW WIDTH
CB3A  A0 00           LDY   #0
CB3C  20 CA99         JSR   CLEOLZ  ;CLEAR THE BOTTOM LINE
                ;
                ;
CB3F  A5 25     VTAB  LDA   CVERT   ;PREPARE SCREEN BASE ADDRESS
                ;
                ;ROUTINE 'ADRCAL' CALCULATES THE TEXT SCREEN BASE
                ;ADDRESS
                ;INPUT : A=LINE NUMBER
                ;OUTPUT : SBASL,H=SCREEN BASE ADDRESS FOR THIS LINE
                ;
                ;
CB41  84 28     ADRCAL  STY   SBASL   ;SAVE Y
CB43  A8                TAY          ;SAVE A
CB44  4A                LSR   A
CB45  29 03             AND   #$03
CB47  0D 057B           ORA   TXTMOD
CB4A  85 29             STA   SBASH
CB4C  B9 C989           LDA   ADRESL,Y
CB4F  A4 28             LDY   SBASL   ;RECOVER Y
CB51  85 28             STA   SBASL
CB53  60                RTS
                ;
                ;
                ;
CB54  20 CB81   CHRDIS  JSR   TST40C  ;REQUIRE ADDRESS MODIFICATION?
CB57  B0 11             BGE   CHRDS1
CB59  20 CC0D           JSR   RAMOIN  ;NO, THEN SIMPLE
CB5C  20 C836           JSR   COMPAT
CB5F  91 28             STA   (SBASL),Y
CB61  20 C836           JSR   COMPAT
CB64  AC 04FB           LDY   TEMPY   ;GET BACK ORIGINAL Y
CB67  4C CC17           JMP   RAMOOU  ;FINISHED
                ;
CB6A  20 CB8E   CHRDS1  JSR   SUBY40  ;MODIFY BASE ADDRESS
CB6D  20 CC0D           JSR   RAMOIN
CB70  20 C836           JSR   COMPAT
CB73  91 28             STA   (SBASL),Y
```

```
CB75  20 C836          JSR   COMPAT
CB78  20 CC17          JSR   RAMOOU
CB7B  AC 04FB          LDY   TEMPY    ;GET BACK ORIGINAL Y
CB7E  4C CB97          JMP   ADDY40   ;CURE THE MODIFICATION
                ;
                ;
CB81  48       TST40C  PHA            ;SAVE A
CB82  8C 04FB          STY   TEMPY    ;SAVE Y
CB85  98               TYA
CB86  18               CLC            ;Y = Y + WINDOW LEFT
CB87  65 20            ADC   WNDLFT
CB89  A8               TAY
CB8A  C9 28            CMP   #40      ;SET CARRY FLAG
CB8C  68               PLA            ;RECOVER A
CB8D  60               RTS
                ;
                ;
CB8E  48       SUBY40  PHA            ;SAVE CHARACTER FIRST
CB8F  98               TYA            ;Y=Y-40
CB90  38               SEC
CB91  E9 28            SBC   #40
CB93  A8               TAY
CB94  4C CB98          JMP   TOGGSH   ;MODIFY BASE ADDRESS
                ;
                ;
CB97  48       ADDY40  PHA            ;SAVE CHARACTER
                ;
CB98  A5 29    TOGGSH  LDA   SBASH
CB9A  49 04            EOR   #$04
CB9C  85 29            STA   SBASH
CB9E  68               PLA            ;RETAIN CHARACTER
CB9F  60               RTS
                ;
                ;
CBA0  48       ENCUR   PHA            ;SAVE CHARACTER FIRST
CBA1  A4 24            LDY   CHORZ
CBA3  20 CB81          JSR   TST40C   ;MODIFYING ADDRESS REQUIRED?
CBA6  B0 0A            BGE   ENCUR1
CBA8  20 CC0D          JSR   RAMOIN   ;NO, THEN SIMPLE
CBAB  20 CBC2          JSR   ENCUR3
CBAE  68               PLA            ;GET BACK THE CHARACTER
CBAF  4C CC17          JMP   RAMOOU   ;FINISHED
                ;
CBB2  20 CB8E  ENCUR1  JSR   SUBY40   ;MODIFY BASE ADDRESS
CBB5  20 CC0D          JSR   RAMOIN
CBB8  20 CBC2          JSR   ENCUR3
CBBB  20 CC17          JSR   RAMOOU
CBBE  68               PLA
CBBF  4C CB97          JMP   ADDY40   ;FINISHED
                ;
                ;
CBC2  20 C836  ENCUR3  JSR   COMPAT
CBC5  B1 28            LDA   (SBASL),Y
CBC7  8D 06FB          STA   TEMPA    ;SAVE FOR DECUR
CBCA  2C C1C5          BIT   TWOMHZ   ;TWO KINDS OF CURSOR
CBCD  30 06            BMI   *+8
```

```
CBCF  29 3F           AND   #$3F    ;FOR 40 COLUMN TEXT
CBD1  09 40           ORA   #$40
CBD3  D0 02           BNE   *+4     ;ALWAYS
CBD5  29 7F           AND   #$7F    ;FOR 80 COLUMN TEXT
CBD7  91 28           STA   (SBASL),Y
CBD9  AC 04FB         LDY   TEMPY   ;GET ORIGINAL Y
CBDC  4C C836         JMP   COMPAT
                      ;
                      ;
CBDF  08              DECUR PHP           ;TURN OFF THE CURSOR
CBE0  48              PHA                 ;SAVE THINGS FOR SAFETY
CBE1  A4 24           LDY   CHORZ
CBE3  AD 06FB         LDA   TEMPA
CBE6  20 CB54         JSR   CHRDIS
CBE9  68              PLA
CBEA  28              PLP
CBEB  60              RTS   ;DONE
                      ;
                      ;
CBEC  A8              ESCX2 TAY                   ;USE Y AS A POINTER
CBED  B9 C8BB         LDA   ESCTB1-$C9,Y          ;TRANSFORM X
CBF0  20 CBFE         JSR   ESCX1                 ;THEN PROCESS IT
CBF3  20 FD0C         JSR   MRDKEY
                      ;
                      ;
CBF6  C9 CE           ESCX  CMP   #$CE    ; )= M?
CBF8  B0 12           BGE   ESCXRT
CBFA  C9 C9           CMP   #$C9    ; ( I?
CBFC  B0 EE           BGE   ESCX2   ; IF FALSE, THEN MUST
                                    ; BE ESC I,J,K, OR M
CBFE  29 3F           ESCX1 AND   #$3F    ;SKIP OFF HIGH ORDER BITS
CC00  C9 08           CMP   #$08    ;)=H?
CC02  B0 08           BGE   ESCXRT  ;IF YES, DO NOTHING
CC04  A8              TAY           ;USE Y AS A POINTER
CC05  A9 CA           LDA   #)BELL
CC07  48              PHA
CC08  B9 C97C         LDA   ESCTBL,Y
CC0B  48              PHA
CC0C  60              ESCXRT RTS
                      ;
                      ;
                      ;
                      ;RAMOIN AND RAMODU ARE TWO VERY, VERY IMPORTANT
                      ;ROUTINES
                      ;IT IS CALLED BY THE KERNEL (BOTH C8 AND F8) AND
                      ;THE BASIC
                      ;
CC0D  08              RAMOIN PHP          ;SAFTEY IS MOST IMPORTANT
CC0E  48              PHA
CC0F  A9 00           LDA   #$00
CC11  8D C07D         STA   SBANK2  ;BRING RAM 0 IN
CC14  68              PLA
CC15  28              PLP
CC16  60              RTS
                      ;
                      ;
```

```
CC17  08          RAMOOU  PHP
CC18  48                  PHA
CC19  AD 077B             LDA     POWER   ;UNDER CP/M OR PASCAL?
CC1C  29 FC               AND     #$FC
CC1E  C9 90               CMP     #$90
CC20  F0 06               BEQ     MUSTR1  ;IF YES, MOVE RAM 1 IN
CC22  A5 C6               LDA     PBANK2  ;MOVE WHICH BLOCK IN?
CC24  C9 10               CMP     #$10    ;A REASONABLE NUMBER?
CC26  90 02               BLT     RAMOO1
CC28  A9 01       MUSTR1  LDA     #$01    ;IF NOT, ASSUME RAM 1
CC2A  8D C07D     RAMOO1  STA     SBANK2  ;STORE IT INTO BLOCK 1
CC2D  68                  PLA
CC2E  28                  PLP
CC2F  60                  RTS
                  ;
                  ;
CC30  20 CCCA     IO      JSR     TUGGLE  ;PERFORM INTERCHANGE
CC33  20 CBDF             JSR     DECUR   ;DO THIS BEFORE EVERYTHING
CC36  B0 63               BCS     BASINP  ;INPUT OR OUTPUT?
                  ;
                  ;
CC38  AD 07FB     BASOUT  LDA     CVWHO
CC3B  C5 25               CMP     CVERT
CC3D  F0 05               BEQ     CVOK
CC3F  85 25               STA     CVERT
CC41  20 CB3F             JSR     VTAB
CC44  AD 047B     CVOK    LDA     CHWHO
CC47  C5 24               CMP     CHORZ
CC49  90 02               BLT     CHOK
CC4B  85 24               STA     CHORZ
CC4D  68          CHOK    PLA             ;GET BACK CHARACTER
CC4E  C9 A0               CMP     #$A0    ;CONTROL CHARACTER?
CC50  90 08               BLT     BASOU1
CC52  25 32               AND     INVFLG  ;NO, DISPLAYABLE
CC54  20 CA18             JSR     STORAD
CC57  4C CC68             JMP     BASOU2  ;GATHER
                  ;
CC5A  29 7F       BASOU1  AND     #$7F    ;FOR CONTROL CHARACTER
CC5C  C9 0D               CMP     #$0D    ;CARRAIGE RETURN?
CC5E  D0 05               BNE     BASOU3
CC60  20 CA01             JSR     VIDOU1  ;YES
CC63  A9 0A               LDA     #$0A    ;ADD A LINE-FEED
CC65  20 CA01     BASOU3  JSR     VIDOU1
                  ;
CC68  A5 24       BASOU2  LDA     CHORZ   ;CHARACTER HAS BEEN SENT
CC6A  F0 06               BEQ     CURECH
CC6C  E9 47               SBC     #$47
CC6E  90 05               BCC     DONE
CC70  69 1F               ADC     #$1F
CC72  8D 047B     CURECH  STA     CHWHO
                  ;
CC75  A5 25       DONE    LDA     CVERT   ;ALL FINISHED, GO BACK!
CC77  8D 07FB             STA     CVWHO
CC7A  20 CBA0             JSR     ENCUR   ;DO THIS BEFORE EXIT
CC7D  20 CCCA             JSR     TUGGLE  ;ALSO DO THIS
CC80  68                  PLA
```

```
CC81  A8              TAY
CC82  68              PLA
CC83  AA              TAX
CC84  68              PLA                 ;RECOVER CHARACTER
CC85  60       DONRTS RTS
               ;
               ;
CC86  A8       NEWESC TAY
CC87  B9 C8BB         LDA   ESCTB1-$C9,Y
CC8A  20 CBFE         JSR   ESCX1
               ;
CC8D  20 CCC1  ESCWHO JSR   RDKWHO  ;ESC WHAT?
CC90  C9 CE           CMP   #$CE    ; >= M?
CC92  B0 F1           BGE   DONRTS
CC94  C9 C9           CMP   #$C9    ; < I ?
CC96  B0 EE           BGE   NEWESC
CC98  20 CBFE         JSR   ESCX1   ;YES
               ;
CC9B  20 CCC1  BASINP JSR   RDKWHO  ;READ KEY
CC9E  C9 9B           CMP   #$9B
CCA0  F0 EB           BEQ   ESCWHO  ;ESC?
CCA2  C9 8D           CMP   #$8D
CCA4  D0 05           BNE   NOTCRW  ;CARIAGE RETURN?
CCA6  48              PHA           ;YES,
CCA7  20 CA97         JSR   CLREOL  ;CLEAR TO END OF LINE
CCAA  68              PLA
CCAB  C9 95    NOTCRW CMP   #$95    ;RIGHT ARROW?
CCAD  D0 05           BNE   NOPICK
CCAF  AD 06FB         LDA   TEMPA   ;YES, PICK UP CHARACTER
CCB2  09 80           ORA   #$80    ;MSB MUST =1
CCB4  BA       NOPICK TSX           ;REPLACE CHARACTER ON STACK
CCB5  9D 0104         STA   STACK+4,X
CCB8  A9 00           LDA   #0
CCBA  8D 0478         STA   CHWHO
CCBD  68              PLA           ;RELEASE THE DUMMY CHARACTER
CCBE  4C CC75         JMP   DONE    ;FINISHED
               ;
               ;
CCC1  20 CBA0  RDKWHO JSR   ENCUR   ;TURN CURSOR ON
CCC4  20 C84D         JSR   KEYIN   ;WHILE WAITING KEY INPUT
CCC7  4C CBDF         JMP   DECUR   ;KEY GOT, TURN OFF CURSOR
               ;
               ;
               ;
               ;ROUTINE 'TUGGLE' IS USED TO INTERCHANGE A SET
               ;OF ZERO PAGE LOCATIONS WITH A SET OF SLOT 3
               ;LOCATIONS
               ;THIS ALLOWS OUR READ-KEY AND CHARACTER-DISPLAY
               ;ROUTINES TO BE SHARED BY DIFFERENT TYPE OF
               ;OPERATING SYSTEMS
               ;
CCCA  08       TUGGLE PHP           ;SAFETY
CCCB  48              PHA
               ;
CCCC  A5 24           LDA   CHORZ
CCCE  48              PHA
```

```
CCCF    AD 047B         LDA     CHWHO
CCD2    85 24           STA     CHORZ
CCD4    68              PLA
CCD5    8D 047B         STA     CHWHO
                ;
CCD8    A5 25           LDA     CVERT
CCDA    48              PHA
CCDB    AD 07FB         LDA     CVWHO
GCDE    85 25           STA     CVERT
CCE0    68              PLA
CCE1    8D 07FB         STA     CVWHO
                ;
CCE4    A5 21           LDA     WNDWTH
CCE6    48              PHA
CCE7    AD 0778         LDA     SAVE1
CCEA    85 21           STA     WNDWTH
CCEC    68              PLA
CCED    8D 0778         STA     SAVE1
                ;
CCF0    68              PLA
CCF1    28              PLP
CCF2    60              RTS
                ;
                ;
                ;
                PAGE
```

```
                        C           INCLUDE VZF800
                        C           .DEPHASE
                        C           ORG     $1571
                        C           .PHASE  $D571
                        C ;
                        C ;THIS IS THE RESET HANDLER
                        C ;
D571    D8              C RESETO    CLD
D572    A2 0F           C           LDX     #$0F
D574    E4 C8           C           CPX     PBANK4
D576    F0 56           C           BEQ     RESET1  ;IF NOT POWER UP, SKIP
                        C ;
D578    A0 A0           C DELAY     LDY     #$A0    ;WAIT TILL SYSTEM SETTLES
D57A    69 01           C DELAY1    ADC     #$01
D57C    90 FC           C           BCC     DELAY1
D57E    88              C           DEY
D57F    D0 F9           C           BNE     DELAY1
                        C ;
D581    8E C07F         C ASSIGN    STX     SBANK4  ;ASSIGN MEMORY CONFIGURATION
D584    8D C078         C           STA     SYSTEM  ;AS : 0,1,3,F
D587    8C C07C         C           STY     SBANK1
D58A    84 C5           C           STY     PBANK1
D58C    86 C8           C           STX     PBANK4
D58E    C8              C           INY
D58F    8C C07D         C           STY     SBANK2
D592    84 C6           C           STY     PBANK2
D594    C8              C           INY
D595    C8              C           INY
D596    8C C07E         C           STY     SBANK3
D599    84 C7           C           STY     PBANK3
                        C ;
D59B    A0 10           C           LDY     #$10
D59D    84 68           C           STY     TXTABH
D59F    85 67           C           STA     TXTABL  ;REMEMBER (A)=$00?
D5A1    85 48           C           STA     STATUS  ;INIT STATUS
D5A3    A8              C           TAY
D5A4    A9 A0           C CLEAR     LDA     #$A0    ;CLEAR SCREEN
D5A6    91 67           C           STA     (TXTABL),Y
D5A8    C8              C           INY
D5A9    D0 F9           C           BNE     CLEAR
D5AB    E6 68           C           INC     TXTABH
D5AD    A5 68           C           LDA     TXTABH
D5AF    C9 18           C           CMP     #$18
D5B1    D0 F1           C           BNE     CLEAR
                        C ;
D5B3    AE C000         C CHOICE    LDX     KEYBRD  ;BASIC STARTS AT $1800?
D5B6    E0 C1           C           CPX     #"A"
D5B8    D0 02           C           BNE     USUAL
D5BA    A9 08           C           LDA     #$08    ;OR AT $800?
D5BC    85 68           C USUAL     STA     TXTABH
                        C ;
D5BE    2C C010         C           BIT     KEYSTR  ;CLEAR KEYBOARD
                        C ;
D5C1    E0 9B           C           CPX     #$9B    ;HAS ESC BEEN PRESSED?
```

```
D5C3    D0 06       C           BNE     SETT40
D5C5    20 FB81     C           JSR     TEXT80  ;CAN BE 80 COLUMN TEXT
D5C8    4C D5CE     C           JMP     RESET1  ;TO BE CONTINUE
D5CB    20 FB78     C SETT40    JSR     TEXT40  ;DEFAULT 40 COLUMN TEXT
                    C ;
D5CE    20 F1BD     C RESET1    JSR     HRSEXT  ;FORCE MEMORY ASSIGNMENTS
D5D1    20 FBCD     C           JSR     FRAM00  ;RIGHT
D5D4    20 F23C     C           JSR     NORMAL  ;SET NORMAL DISPLAY
D5D7    AD C028     C           LDA     TEXTCR  ;WHITE, BLACK, BLACK
D5DA    AD C018     C           LDA     BKGRND
D5DD    AD C008     C           LDA     BKDROP
D5E0    20 FBFA     C           JSR     SONINT  ;TURN OFF SOUND GEN.
D5E3    20 FB2F     C           JSR     VZINIT  ;INIT TEXT DISPLAY
D5E6    20 FE39     C           JSR     SCREEN  ;INIT INPUT/OUTPUT
D5E9    20 FE42     C           JSR     KBDBRD
D5EC    20 FF3A     C           JSR     MBELL   ;INFORM THE USER
D5EF    AD 03F4     C           LDA     RESTVR+2;CHECK RESET VECTOR
D5F2    49 A5       C           EOR     #$A5
D5F4    CD 03F3     C           CMP     RESTVR+1
D5F7    D0 03       C           BNE     FSTIME  ;POWER UP?
D5F9    4C FCF1     C           JMP     DEBUG1  ;NO
D5FC    20 FB99     C FSTIME    JSR     TITLE   ;SHOW OUR LOG
D5FF    20 FBAA     C           JSR     INFKEY  ;CLEAR THE FUNCTION KEYS
D602    20 FB61     C           JSR     PRESP3  ;PRESET PAGE 3 VECTORS
D605    20 FD43     C           JSR     CHKDIS  ;CHECK DRIVE CONTROLLER
D608    A9 03       C NODRIV    LDA     #(BASICW;SET UP RESET VECTOR
D60A    8D 03F2     C           STA     RESTVR
D60D    20 FB6F     C           JSR     SRESTV
D610    4C E000     C           JMP     BASICC
                    C ;
                    C           .DEPHASE
                    C           ORG     $1800
                    C           .PHASE  $F800
                    C ;
F800                C           DS      $800,$60
                    C ;
                    C ;
                    C           .DEPHASE
                    C           ORG     $1800
                    C           .PHASE  $F800
                    C ;
F800    4C FBC7     C           JMP     FRAM01
F803    4C FBCD     C           JMP     FRAM00
F806    4C FB78     C           JMP     TEXT40
F809    4C FB81     C           JMP     TEXT80
F80C    4C FC2A     C           JMP     AUDOUT
F80F    4C FC4E     C           JMP     MOUTS1
                    C ;
                    C ;
F812    AD 057B     C SETWTH    LDA     TXTMOD  ;40 OR 80 COLUMN MODE?
F815    C9 10       C           CMP     #$10
F817    D0 08       C           BNE     SETW40
F819    A9 50       C           LDA     #80     ;80 COLUMN TEXT MODE
F81B    8D C04F     C           STA     VZTX80
F81E    85 21       C SETW40    STA     WNDWTH
F820    60          C           RTS
```

```
F821  A9 28     C SETW40  LDA   #40      ;40 COLUMN TEXT MODE
F823  8D C04C   C         STA   VZTX40
F826  D0 F6     C         BNE   SETW48    ;ALWAYS
                C ;
                C ;
F828  A5 3E     C DOT     LDA   REG2L
F82A  85 3C     C         STA   REG1L
F82C  A5 3F     C         LDA   REG2H
F82E  85 3D     C         STA   REG1H
F830  60        C         RTS
                C ;
                C ;
F831  07        C DATA07  DB    $07
                C ;
                C ;
F832  60        C         RTS
                C ;
                C ;
F833  A5 34     C RETURN  LDA   SAVEX    ;RETURN ONLY?
F835  F0 0F     C         BEQ   XMEM8
F837  20 FCDF   C         JSR   SPACE    ;IF NOT, DO A SPACE FUNCTION
F83A  20 F828   C         JSR   DOT      ;REGISTER 1=2
F83D  68        C RETRTS  PLA            ;POP RETURN ADDRESS
F83E  68        C         PLA
F83F  A9 00     C         LDA   #$00     ;CLEAR STORE MODE
F841  85 31     C         STA   STOFLG
F843  4C FF69   C         JMP   MON1     ;END OF KERNEL COMMAND
                C                        ;INTERPRETATION
                C ;
                C ;
F846  AD 057B   C XMEM8   LDA   TXTMOD   ;40 OR 80 COLUMN MODE?
F849  C9 10     C         CMP   #$10
F84B  D0 03     C         BNE   *+5
F84D  A9 0F     C         LDA   #$0F     ;80 COLUMN MODE
F84F  2C        C         DB    $2C      ;$2C='BIT'
F850  A9 07     C         LDA   #$07     ;40 COLUMN MODE
F852  85 2E     C         STA   CHKSUM   ;USE CHKSUM AS A GENERAL
                C                        ;REGISTER
                C ;
F854  20 FEC5   C         JSR   CROUT    ;CARRIAGE RETURN
F857  20 FC85   C         JSR   INCRE2   ;INCREMENT REGISTER 1
F85A  A5 3D     C         LDA   REG1H    ;PRINT ADDRESS
F85C  20 FDC1   C         JSR   PRBYTE
F85F  A5 3C     C         LDA   REG1L
F861  20 FDC1   C         JSR   PRBYTE
F864  A9 BD     C         LDA   #"="     ;FOLLOWED BY =
F866  20 FDED   C         JSR   MCOUT
                C ;
F869  A0 00     C         LDY   #0       ;CLEAR OFFSET
F86B  A9 A0     C XMEM81  LDA   #" "     ;WITH A SPACE BEHIND
F86D  20 FDED   C         JSR   MCOUT
F870  B1 3C     C         LDA   (REG1L),Y        ;THEN ITS CONTENTS
F872  20 FDC1   C         JSR   PRBYTE
F875  A5 3C     C         LDA   REG1L
F877  25 2E     C         AND   CHKSUM   ;EITHER MOD 8 OR
F879  C5 2E     C         CMP   CHKSUM   ;MOD 16
```

```
F87B  F0 C0     C          BEQ   RETRTS  ;FINISHED?
F87D  20 FC85   C          JSR   INCRE2  ;INCREMENT REGISTER 1
F880  4C F86B   C          JMP   XMEM81  ;CONTINUE
                C ;
                C ;
F883  C5 D2 D2  C ERRORM   ASC   "ERROR "
F886  CF D2 A0  C
                C ;
                C ;
F889  A9 00     C OPERR    LDA   #0      ;FOR INVALID OPCODES
F88B  85 2F     C          STA   OPCODL  ;SET LENGHT=0
F88D  60        C          RTS
                C ;
F88E  4A        C LENGTH   LSR           ;START CHECKING OPCODE
F88F  B0 14     C          BCS   ODD     ;IF ODD, DO MORE
F891  4A        C ODDEVN   LSR           ;SELECT NIBBLE
F892  A8        C          TAY           ;USE Y AS POINTER
F893  B9 F9F6   C          LDA   TABLE1,Y;GET LENGTH
F896  90 04     C          BCC   LOWNIB  ;WHICH NIBBLE?
F898  4A        C          LSR           ;HIGH NIBBLE
F899  4A        C          LSR
F89A  4A        C          LSR
F89B  4A        C          LSR
F89C  29 0F     C LOWNIB   AND   #$0F     ;TICK OUT HIGH BITS
F89E  C9 06     C          CMP   #6      ;INVALID OPCODE?
F8A0  F0 E7     C          BEQ   OPERR
F8A2  85 2F     C          STA   OPCODL  ;SAVE THE LENGHT
F8A4  60        C          RTS           ;FINISHED
                C ;
F8A5  6A        C ODD      ROR
F8A6  B0 E1     C          BCS   OPERR   ;NO XXXXXX11 OPCODES
F8A8  49 FF     C          EOR   #$FF
F8AA  C9 5D     C          CMP   #$5D
F8AC  F0 DB     C          BEQ   OPERR   ;NO STA #---
F8AE  49 FF     C          EOR   #$FF
F8B0  29 87     C          AND   #$87    ;MASK BITS
F8B2  4C F891   C          JMP   ODDEVN
                C ;
                C ;
F8B5  A9 40     C MTSAVE   LDA   #64
F8B7  20 F8F0   C          JSR   LEADER
F8BA  2C C1C5   C          BIT   TWOMHZ
F8BD  10 03     C          BPL   *+5
F8BF  A0 54     C          LDY   #84     ;FIRST BYTE
F8C1  2C        C          DB    $2C
F8C2  A0 25     C          LDY   #37
F8C4  A2 00     C TSAVE1   LDX   #0
F8C6  41 3C     C          EOR   (REG1L,X)
F8C8  85 2E     C          STA   CHKSUM
F8CA  A1 3C     C          LDA   (REG1L,X)
F8CC  20 FE18   C          JSR   WRBYTE
F8CF  20 FC85   C          JSR   INCRE2
F8D2  2C C1C5   C          BIT   TWOMHZ
F8D5  10 03     C          BPL   *+5
F8D7  A0 3E     C          LDY   #62     ;REST BYTES
F8D9  2C        C          DB    $2C
```

```
F8DA  A0 1B     C          LDY   #27
F8DC  A5 2E     C          LDA   CHKSUM
F8DE  90 E4     C          BLT   TSAVE1
F8E0  2C C1C5   C          BIT   TWOMHZ
F8E3  10 03     C          BPL   *+5
F8E5  A0 44     C          LDY   #68      ;THE LAST: CHECK-SUM
F8E7  2C        C          DB    $2C
F8E8  A0 20     C          LDY   #32
F8EA  20 FE18   C          JSR   WRBYTE
F8ED  4C FF3A   C          JMP   MBELL
                C ;
                C ;
                C ;
F8F0  2C C1C5   C LEADER   BIT   TWOMHZ   ;ONE OR TWO MHZ?
F8F3  10 03     C          BPL   *+5      ;NORMALLY IT IS 1MHZ
F8F5  A0 9B     C          LDY   #155     ;BUT CAN BE 2MHZ
F8F7  2C        C          DB    $2C      ;$2C = 'BIT'
F8F8  A0 49     C          LDY   #73      ;GOOD, 1MHZ
F8FA  20 F912   C          JSR   ZDELAY
F8FD  D0 F1     C          BNE   LEADER
F8FF  69 FE     C          ADC   #$FE
F901  B0 ED     C          BCS   LEADER
F903  2C C1C5   C          BIT   TWOMHZ
F906  10 03     C          BPL   *+5
F908  A0 44     C          LDY   #68
F90A  2C        C          DB    $2C
F90B  A0 1F     C          LDY   #31
F90D  20 F912   C WRTBIT   JSR   ZDELAY
F910  C8        C          INY
F911  C8        C          INY
F912  88        C ZDELAY   DEY
F913  D0 FD     C          BNE   ZDELAY
F915  90 10     C          BCC   WTAPE    ;ZERO IS SHORTER THAN ONE
F917  2C C1C5   C          BIT   TWOMHZ
F91A  10 03     C          BPL   *+5
F91C  A0 65     C          LDY   #101
F91E  2C        C          DB    $2C
F91F  A0 2F     C          LDY   #47
F921  88        C ODELAY   DEY            ;EXTRA DELAY FOR ONE
F922  D0 FD     C          BNE   ODELAY
F924  4C F927   C          JMP   WTAPE    ;TIME COMPENSATION
F927  AC C020   C WTAPE    LDY   TAPEOU   ;TOGGLE TAPE OUTPUT
F92A  2C C1C5   C          BIT   TWOMHZ
F92D  10 03     C          BPL   *+5
F92F  A0 5A     C          LDY   #90
F931  2C        C          DB    $2C
F932  A0 2A     C          LDY   #42
F934  CA        C          DEX
F935  60        C          RTS
                C ;
                C ;
F936            C          DS    10
                C ;
                C ;
F940  98        C PRNYX    TYA            ;PRINT Y
F941  20 FDC1   C PRNAX    JSR   PRBYTE   ;PRINT A
```

ADDENDUM SHEET

FOR P. 186


```
F944   8A          C PRNX    TXA             ;PRINT X
F945   4C FDC1     C         JMP   PRRYTE
                   C ;
                   C ;
F948   A9 03       C PRNSPC  LDA   #3        ;SET UP COUNTER
F94A   85 4E       C         STA   RNDNOL
F94C   A9 A0       C         LDA   #$A0      ;PRINT SPACES
F94E   20 FDED     C PRNSP1  JSR   MCOUT
F951   C6 4E       C         DEC   RNDNOL    ;FINISHED?
F953   D0 F9       C         BNE   PRNSP1
F955   60          C         RTS
                   C .
                   C ;
F956   A2 09       C RTBBIT  LDX   #9        ;ONE BYTE HAVE 8 BITS
F958   CA          C         DEX
F959   48          C RDBYTE  PHA             ;SAVE A FIRST
F95A   20 F965     C         JSR   READPS    ;READ A FULL CYCLE
F95D   68          C         PLA
F95E   2A          C         ROL             ;SHIFT THE READ BIT IN
F95F   A0 39       C         LDY   #57
F961   CA          C         DEX             ;A BYTE FINISHED?
F962   D0 F5       C         BNE   RDDYT2
F964   60          C         RTS             ;YES.  EXIT
                   C ;
                   C ;
F965   20 F968     C READPS  JSR   RT1BIT
F968   88          C RT1BIT  DEY             ;DOUBLE DECREMENT
F969   88          C         DEY
F96A   2C C1C5     C         BIT   TWOMHZ    ;1 OR 2MHZ?
F96D   10 0A       C         BPL   RTBIT1    ;NORMALLY 1MHZ
F96F   E6 4E       C         INC   RNDNOL    ;5 CLOCK CYCLES EACH
F971   C6 4E       C         DEC   RNDNOL    ;AS DOUBLE DELAY
F973   E6 4E       C         INC   RNDNOL
F975   C6 4E       C         DEC   RNDNOL
F977   EA          C         NOP             ;2 CLOCK CYCLES EACH.
F978   EA          C         NOP             ;TOTAL DELAY =24 CLOCK CYCLE
F979   AD C060     C RTBIT1  LDA   TAPEIN    ;READ TAPE, 4 CYCLES
F97C   45 2F       C         EOR   LASTBI    ;ANOTHER 3 CYCLES
F97E   10 07       C         BPL   RTBIT2    ;AGAIN, 3 CYCLES
F980   45 2F       C         EOR   LASTBI
F982   85 2F       C         STA   LASTBI
F984   C0 80       C         CPY   #$80      ;$80 IS AN AVERAGE NUMBER
F986   60          C         RTS             ;ONLY, NO RELATIONSHIP TO MSB.
F987   4C F968     C RTBIT2  JMP   RT1BIT    ;COMPENSATE FOR 3 CYCLES
                   C ;
                   C ;
F98A   A0 00       C VERIFY  LDY   #0        ;NO OFFSET
F98C   B1 3C       C         LDA   (REG1L),Y
F98E   D1 42       C         CMP   (REG4L),Y
F990   D0 06       C         BNE   VERERR    ;SAME?
F992   20 FC7F     C VERCOM  JSR   INCRE1    ;YES, NEXT PAIR
F995   90 F3       C         BLT   VERIFY    ;FINISHED?
F997   60          C         RTS             ;YES
                   C;
F99B   20 FEC5     C VERERR JSR    CROUT     ;CARRIAGE RETURN FIRST
F99B   A5 3D       C        LDA    REG1H     ;PRINT ADDRESS
```

```
F99D  20 FDC1   C          JSR   PRBYTE
F9A0  A5 3C     C          LDA   REG1L
F9A2  20 FDC1   C          JSR   PRBYTE
F9A5  A9 BD     C          LDA   #"="     ;PRINT CONTENT
F9A7  20 FDED   C          JSR   MCOUT
F9AA  A0 00     C          LDY   #0
F9AC  B1 3C     C          LDA   (REG1L),Y
F9AE  20 FDC1   C          JSR   PRBYTE
F9B1  A9 A0     C          LDA   #" "     ;THEN SPACE
F9B3  20 FDED   C          JSR   MCOUT
F9B6  A9 A8     C          LDA   #"("     ;THEN THE UNMATCHED VALUE
F9B8  20 FDED   C          JSR   MCOUT
F9BB  A0 00     C          LDY   #0
F9BD  B1 42     C          LDA   (REG4L),Y
F9BF  20 FDC1   C          JSR   PRBYTE
F9C2  A9 A9     C          LDA   #")"
F9C4  20 FDED   C          JSR   MCOUT
F9C7  4C F992   C          JMP   VERCOM   ;NEXT
                C ;
                C ;
F9CA  E0 F8     C TAB      CPX   #248     ;NEAR END OF INPUT LINE?
F9CC  B0 25     C          BGE   TABJMP   ;IF YES, DO NOTHING
F9CE  A5 24     C          LDA   CHORZ    ;TRY TO DO TAB
F9D0  18        C          CLC
F9D1  69 08     C          ADC   #8       ;TAB = ADVANCE 8 POS.
F9D3  C5 21     C          CMP   WNDWTH   ;EXCEED THE RIGHT END?
F9D5  B0 1C     C          BGE   TABJMP   ;IF YES, CANCEL
                C ;
F9D7  20 FDE6   C          JSR   C800IN   ;FOR DECUR AND ENCUR
F9DA  AD 06FB   C TAB1     LDA   TEMPA    ;NOW, DO TAB
F9DD  9D 0200   C          STA   KEYBUF,X;COPY THINGS ON SCREEN
F9E0  E8        C          INX            ;AND ADVANCE
F9E1  E6 24     C          INC   CHORZ
F9E3  A5 24     C          LDA   CHORZ
F9E5  2C F831   C          BIT   DATA07   ;TAB POS. ARE QUANTIZED
F9E8  F0 09     C          BEQ   TABJMP   ;FINISHED?
F9EA  20 CBA0   C          JSR   ENCUR    ;IF NOT, CONTINUE
F9ED  20 CBDF   C          JSR   DECUR    ;COPYING CHARACTERS
F9F0  4C F9DA   C          JMP   TAB1
                C ;
F9F3  4C FD71   C TABJMP   JMP   GETLN1   ;GO BACK
                C ;
                C ;
                C ;
                C ;TABLE1 CONTAINS:
                C ;1. 128 4-BIT LENGTHS FOR XXXXXXX0 TYPE OF OPCODES
                C ;2. 8 4-BIT LENGTHS FOR XXXXXX01 TYPE OF OPCODES
                C ;    I.E., ORA,AND,EOR,ADC,STA,LDA,CMP,SBC
                C ;
F9F6  60 16 00  C TABLE1   DB    $60,$16,$00,$26,$61,$16,$60,$26
F9F9  26 61 16  C
F9FC  60 26     C
F9FE  62 11 00  C          DB    $62,$11,$00,$22,$61,$16,$60,$26
FA01  22 61 16  C
FA04  60 26     C
FA06  60 16 00  C          DB    $60,$16,$00,$22,$61,$16,$60,$26
```

```
FA09  22 61 16   C
FA0C  60 26      C
FA0E  60 16 00   C          DB      $60,$16,$00,$22,$61,$16,$60,$26
FA11  22 61 16   C
FA14  60 26      C
FA16  66 11 00   C          DB      $66,$11,$00,$22,$61,$16,$60,$26
FA19  22 61 16   C
FA1C  60 26      C
FA1E  11 11 00   C          DB      $11,$11,$00,$22,$61,$11,$00,$22
FA21  22 61 11   C
FA24  00 22      C
FA26  61 11 00   C          DB      $61,$11,$00,$22,$61,$16,$60,$26
FA29  22 61 16   C
FA2C  60 26      C
FA2E  61 11 00   C          DB      $61,$11,$00,$22,$61,$16,$60,$26
FA31  22 61 16   C
FA34  60 26      C
FA36  11 21 11   C          DB      $11,$21,$11,$22
FA39  22         C
                 C ;
                 C ;
                 C          .DEPHASE
                 C          ORG     $1A40
                 C          .PHASE  $FA40
                 C ;
FA40  85 45      C IRQBRK   STA     ACC      ;SAVE A
FA42  68         C          PLA              ;GET STATUS REGISTER
FA43  48         C          PHA
FA44  2C FB49    C          BIT     DATA10   ;INTERUPT OR BREAK?
FA47  D0 03      C          BNE     BREAK
FA49  6C 03FE    C          JMP     (IRQVER)
FA4C  68         C BREAK    PLA              ;GET BACK STATUS REGISTER
FA4D  85 48      C          STA     STATUS   ;SAVE IT
FA4F  68         C          PLA              ;SAVE RETURN ADDRESS
FA50  85 3A      C          STA     PCL
FA52  68         C          PLA
FA53  85 3B      C          STA     PCH
FA55  86 46      C          STX     REGX     ;SAVE REGISTERS
FA57  84 47      C          STY     REGY
FA59  BA         C          TSX
FA5A  86 49      C          STX     STACKP
FA5C  6C 03F0    C          JMP     (BRKVER)
                 C ;
FA5F  4C FF65    C          JMP     MON      ;DUMMY
                 C ;
                 C ;
FA62  4C D571    C RESET    JMP     RESET0
                 C ;
                 C ;
FA65  20 FEC5    C BREAK1   JSR     CROUT    ;CARRIAGE RETURN FIRST
FA68  A2 00      C          LDX     ##$00    ;INIT POINTER & COUNTER
FA6A  BD FCBD    C BREAK2   LDA     BRKMES,X;PRINT BREAK MESSAGE
FA6D  20 FDED    C          JSR     MCOUT
FA70  E8         C          INX              ;FINISHED?
FA71  E0 0B      C          CPX     #11
FA73  D0 F5      C          BNE     BREAK2
```

```
FA75    A5 3A       C           LDA     PCL     ;PRINT ADDRESS
FA77    E9 02       C           SBC     #2
FA79    48          C           PHA
FA7A    A5 3B       C           LDA     PCH
FA7C    E9 00       C           SBC     #0
FA7E    20 FDC1     C           JSR     PRBYTE
FA81    68          C           PLA
FA82    20 FDC1     C           JSR     PRBYTE
FA85    4C FF65     C           JMP     MON     ;THEN GOTO KERNEL
                    C ;
                    C ;
FA88    83          C COMTBL    DB      $83     ;CTRL-C
FA89    82          C           DB      $82     ;CTRL-B
FA8A    99          C           DB      $99     ;CTRL-Y
FA8B    8B          C           DB      $8B     ;CTRL-K
FA8C    90          C           DB      $90     ;CTRL-P
FA8D    CE          C           DB      $CE     ;N
FA8E    C9          C           DB      $C9     ;I
FA8F    AE          C           DB      $AE     ;.
FA90    B3          C           DB      $B3     ;** : **
FA91    B5          C           DB      $B5     ;** ( **
FA92    C7          C           DB      $C7     ;G
FA93    D2          C           DB      $D2     ;R
FA94    D7          C           DB      $D7     ;W
FA95    CD          C           DB      $CD     ;M
FA96    D6          C           DB      $D6     ;V
FA97    8D          C           DB      $8D     ;RETURN
FA98    A0          C           DB      $A0     ;SPACE
                    C ;
                    C ;
FA99    E002        C ADRTBL    DW      BASICW-1
FA9B    DFFF        C           DW      BASICC-1
FA9D    03F7        C           DW      USRADR-1
FA9F    FE8C        C           DW      SETIN-1
FAA1    FE96        C           DW      SETOU-1
FAA3    FE83        C           DW      SETNOR-1
FAA5    FE7F        C           DW      SETINV-1
FAA7    F827        C           DW      DOT-1
FAA9    FC05        C           DW      COLON-1
FAAB    FB3F        C           DW      MODE3-1
FAAD    FECF        C           DW      GO-1
FAAF    FEFC        C           DW      TLOAD-1
FAB1    F8B4        C           DW      MTSAVE-1
FAB3    FE2B        C           DW      MOVE-1
FAB5    F989        C           DW      VERIFY-1
FAB7    F832        C           DW      RETURN-1
FAB9    FCDE        C           DW      SPACE-1
                    C ;
                    C ;
FABB    CD C9 C3    C LOGO      ASC     "MICROSOFT BASIC V.T. VERSION 2.2"
FABE    D2 CF D3    C
FAC1    CF C6 D4    C
FAC4    A0 C2 C1    C
FAC7    D3 C9 C3    C
FACA    A0 D6 AE    C
FACD    D4 AE A0    C
```

```
FAD0    D6 C5 D2    C
FAD3    D3 C9 CF    C
FAD6    CE A0 B2    C
FAD9    AE B2       C
FADB    8D 8D       C           DB      $8D,$8D
FADD    A8 C3 A9    C           ASC     "(C) COPYRIGHT V.T. 1984"
FAE0    A0 C3 CF    C
FAE3    D0 D9 D2    C
FAE6    C9 C7 C8    C
FAE9    D4 A0 D6    C
FAEC    AE D4 AE    C
FAEF    A0 B1 B9    C
FAF2    B8 B4       C
FAF4    8D 8D 00    C           DB      $8D,$8D,$00
                    C ;
                    C ;
                    C ;
                    C           .DEPHASE
                    C           ORG     $1B02
                    C           .PHASE  $FB02
                    C ;
FB02    BD C064     C PREAD1    LDA     PADDLO,X;3 CYCLES
FB05    10 04       C           BPL     PDLRTS  ;2 CYCLES
FB07    C8          C           INY             ;2 CYCLES
FB08    D0 F8       C           BNE     PREAD1  ;3 CYCLES
FB0A    88          C           DEY
FB0B    60          C PDLRTS    RTS
                    C ;
                    C ;
FB0C    BD C064     C PREAD2    LDA     PADDLO,X;3 CYCLES
FB0F    BD C064     C           LDA     PADDLO,X;3 CYCLES
FB12    10 F7       C           BPL     PDLRTS  ;2 CYCLES
FB14    10 F5       C           BPL     PDLRTS  ;2 CYCLES
FB16    C8          C           INY             ;2 CYCLES
FB17    EA          C           NOP             ;2 CYCLES
FB18    85 2A       C           STA     SBAS2L  ;3 CYCLES
FB1A    D0 F0       C           BNE     PREAD2  ;3 CYCLES
FB1C    88          C           DEY
FB1D    60          C           RTS
                    C ;
                    C ;
FB1E    A0 00       C MPREAD    LDY     #$00
FB20    2C C1C5     C           BIT     TWOMHZ  ;1 OR 2 MHZ?
FB23    8D C070     C           STA     PDLRES  ;RESET TIMER
FB26    10 DA       C           BPL     PREAD1  ;FOR 1MHZ
FB28    30 E2       C           BMI     PREAD2  ;FOR 2MHZ
                    C ;
                    C ;
FB2A                C           DS      5
                    C ;
                    C ;
FB2F    EA          C VZINIT    NOP             ;SPACE FILLERS
FB30    EA          C           NOP
FB31    EA          C           NOP
FB32    EA          C           NOP
FB33    AD C056     C           LDA     VZSELF
```

```
FB36  AD C054   C          LDA   VZPAG1
FB39  AD C051   C SETTXT   LDA   VZTEXT
FB3C  A9 00     C          LDA   #$00
FB3E  F0 0B     C          BEQ   MSETWN  ;ALWAYS
                C ;
                C ;
FB40  A5 3E     C MODE3    LDA   REG2L
FB42  85 42     C          STA   REG4L
FB44  A5 3F     C          LDA   REG2H
FB46  85 43     C          STA   REG4H
FB48  60        C          RTS
                C ;
                C ;
FB49  10        C DATA10   DB    $10
FB4A  00        C SIGNAT   DB    $00      ;SIGNATURE BYTE
                C ;
                C ;
FB4B  85 22     C MSETWN   STA   WNDTOP   ;SET FULL SCREEN SIZE
FB4D  A9 00     C          LDA   #0       ;24*40 OR 24*80
FB4F  85 20     C          STA   WNDLFT
FB51  A9 18     C          LDA   #24
FB53  85 23     C          STA   WNDBTM
FB55  EA        C          NOP
FB56  20 F812   C          JSR   SETWTH   ;40 OR 80?
FB59  A9 17     C          LDA   #23      ;PLACE THE CURSOR AT THE
                C                         ;BOTTOM LINE
FB5B  85 25     C MTABV    STA   CVERT
FB5D  20 FC22   C          JSR   MVTAB
FB60  60        C          RTS
                C ;
                C ;
FB61  A0 04     C PRESP3   LDY   #4
FB63  B9 FBD2   C LOOP1    LDA   P3VECT-1,Y
FB66  99 03EF   C          STA   BRKVER-1,Y
FB69  88        C          DEY
FB6A  D0 F7     C          BNE   LOOP1
FB6C  4C FB6F   C          JMP   SRESTV
                C ;
                C ;
FB6F  AD 03F3   C SRESTV   LDA   RESTVR+1
FB72  49 A5     C          EOR   #$A5
FB74  8D 03F4   C          STA   RESTVR+2
FB77  60        C          RTS
                C ;
                C ;
FB78  A9 04     C TEXT40   LDA   #$04     ;ENABLE 40 COLUMN TEXT
FB7A  20 FB8A   C          JSR   BEAUTI
FB7D  2C C04C   C          BIT   VZTX40
FB80  60        C          RTS
                C ;
                C ;
FB81  A9 10     C TEXT80   LDA   #$10     ;ENABLE 80 COLUMN TEXT
FB83  20 FB8A   C          JSR   BEAUTI
FB86  2C C04F   C          BIT   VZTX80
FB89  60        C          RTS
                C ;
```

```
                        C ;
FB8A    8D 057B         C BEAUTI  STA     TXTMOD
FB8D    2C C1C3         C         BIT     VERTSC  ;CHANGE DURING VERTICAL
FB90    30 FB           C         BMI     *-3     ;RETRACE PERIOD
FB92    2C C056         C         BIT     VZSELF
FB95    2C C051         C         BIT     VZTEXT
FB98    60              C         RTS
                        C ;
                        C ;
FB99    20 FC58         C TITLE   JSR     F8HOME
FB9C    A2 00           C         LDX     #0      ;USE X AS COUNTER AND POINTER
FB9E    BD FABB         C TITLE1  LDA     LOGO,X
FBA1    F0 06           C         BEQ     TITRTS  ;END?
FBA3    20 FDED         C         JSR     MCOUT   ;DISPLAY OUR LOGO
FBA6    E8              C         INX
FBA7    D0 F5           C         BNE     TITLE1  ;ALWAYS
FBA9    60              C TITRTS  RTS
                        C ;
                        C ;
FBAA    20 FBC7         C INFKEY  JSR     FRAMOI  ;GET FUNCTION KEY BUFFER
FBAD    A2 18           C         LDX     #24
FBAF    A9 A0           C         LDA     #$A0    ;FILL WITH SPACES
FBB1    9D 4800         C INFKE1  STA     KEYFLG,X;REMEMBER X=24?
FBB4    CA              C         DEX
FBB5    10 FA           C         BPL     INFKE1  ;ALSO DEACTIVATE FUNCTION
                        C                         ;KEY
FBB7    4C FBCD         C         JMP     FRAMOO  ;O.K., FINISHED!
                        C ;
                        C ;
                        C ;
                        C         .DEPHASE
                        C         ORG     $1BC1
                        C         .PHASE  $FBC1
                        C ;
FBC1    20 FDE6         C         JSR     C800IN
FBC4    4C CB41         C         JMP     ADRCAL
                        C ;
                        C ;
                        C ;FRAMOI AND FRAMOO SHOULD BE CALLED WHEN YOU
                        C ;ARE OUTSIDE C8
                        C ;WHEN IN C8, YOU CAN SIMPLY CALL RAMOIN AND RAMOOU
FBC7    20 FDE6         C FRAMOI  JSR     C800IN
FBCA    4C CC0D         C         JMP     RAMOIN
                        C ;
                        C ;
FBCD    20 FDE6         C FRAMOO  JSR     C800IN
FBD0    4C CC17         C         JMP     RAMOOU
                        C ;
                        C ;
FBD3    FA65            C P3VECT  DW      BREAK1
FBD5    E000            C         DW      BASICC
                        C ;
                        C ;
FBD7                    C         DS      2
                        C ;
                        C ;
```

```
FBD9  C9 87     C          CMP    #$87     ;CTRL-G
FBDB  D0 F0     C          BNE    FRAMOD
FBDD  EA        C          NOP
FBDE  EA        C          NOP
FBDF  EA        C          NOP
FBE0  EA        C          NOP
FBE1  EA        C          NOP
FBE2  EA        C          NOP             ;FILL SPACE
FBE3  EA        C          NOP
FBE4  20 FDE6   C          JSR    C800IN
FBE7  4C CA2E   C          JMP    BELL
                C ;
                C ;
FBEA  20 FDE6   C ENTRY1   JSR    C800IN
FBED  4C CA18   C          JMP    STORAD
                C ;
                C ;
FBF0  20 FBEA   C          JSR    ENTRY1
FBF3  60        C          RTS
                C ;
                C ;
FBF4  20 FDE6   C          JSR    C800IN
FBF7  4C CA20   C          JMP    ADVANC
                C ;
                C ;
FBFA  A0 03     C SONINT   LDY    #3       ;TURN SOUND GEN. OFF
FBFC  B9 FC16   C SOUND1   LDA    DEAF,Y
FBFF  20 FC2A   C          JSR    AUDOUT
FC02  88        C          DEY
FC03  10 F7     C          BPL    SOUND1
FC05  60        C          RTS
                C ;
                C ;
FC06  A9 99     C COLON    LDA    #$99
FC08  85 31     C          STA    STOFLG
FC0A  4C F828   C          JMP    DOT      ;REGISTER 1=2
                C ;
                C ;
FC0D            C          DS     3
                C ;
                C ;
FC10  20 FDE6   C          JSR    C800IN
FC13  4C CA54   C          JMP    BS
                C ;
                C ;
                C ;DATA USED TO TURN THE SOUND CHANNELS OFF
FC16  9F BF DF  C DEAF     DB     $9F,$BF,$DF,$FF
FC19  FF        C
                C ;
                C ;
FC1A  20 FDE6   C          JSR    C800IN
FC1D  4C CA5E   C          JMP    UP
                C ;
                C ;
FC20            C          DS     2
                C ;
```

```
                        C ;
FC22    A5 25           C MVTAB   LDA     CVERT
FC24    20 FDE6         C         JSR     C800IN
FC27    4C CB41         C         JMP     ADRCAL
                        C ;
                        C ;
FC2A    8D C068         C AUDOUT  STA     SONGEN  ;SEND DATA TO SOUND GEN.
FC2D    2C C1C2         C         BIT     HORZSC  ;WAIT FOR 1 HORIZONTAL
FC30    30 FB           C         BMI     *-3     ;SYNC. PERIOD
FC32    2C C1C2         C         BIT     HORZSC
FC35    10 FB           C         BPL     *-3
FC37    60              C         RTS
                        C ;
                        C ;
FC38    A0 24           C FNDSTB  LDY     #36     ;READ THE SYNC PERIOD
FC3A    20 F968         C         JSR     RT1BIT
FC3D    B0 F9           C         BCS     FNDSTB
FC3F    60              C         RTS
                        C ;
                        C ;
FC40                    C         DS      2
                        C ;
                        C ;
                        C
FC42    20 FDE6         C         JSR     C800IN
FC45    4C CA7F         C         JMP     CLREOP
                        C ;
                        C ;
FC48    20 FDE6         C ENTRY2  JSR     C800IN
FC4B    4C CAA4         C         JMP     CRLF
                        C ;
                        C ;
FC4E    20 FDF0         C MOUTS1  JSR     MCOUT1  ;FOR PRINTER DRIVER
FC51    8D CFFF         C         STA     ROMCLR
FC54    8D C100         C         STA     $C100   ;ENABLE PRINTER DRIVER
FC57    60              C         RTS
                        C ;
                        C ;
FC58    20 FDE6         C F8HOME  JSR     C800IN
FC5B    4C CA76         C         JMP     CLSCRN  ;HOME OF BASIC
                        C                         ;=CLEAR SCREEN OF PASCAL
                        C ;
                        C ;
FC5E                    C         DS      4       ;FILL SPACE
                        C ;
                        C ;
FC62    20 FC48         C         JSR     ENTRY2
FC65    60              C         RTS
                        C ;
                        C ;
FC66    20 FDE6         C         JSR     C800IN
FC69    4C CAA7         C         JMP     LF
                        C ;
                        C ;
FC6C    48              C MWAIT1  PHA             ;SAVE MAJOR TIMER
FC6D    69 01           C MWAIT2  ADC     #1      ;MINOR TIMER
```

```
FC6F  D0 FC     C          BNE    MWAIT2  ;CARRY=0 FOR NON-EQUAL
FC71  68        C          PLA            ;GET BACK MAJOR TIMER
FC72  69 00     C          ADC    #0      ;N.B. CARRY=1 NOW
FC74  D0 F6     C          BNE    MWAIT1  ;CARRY=0 FOR NON-EQUAL
FC76  60        C          RTS            ;ALL FINISHED!
FC77  49 FF     C MWAIT    EOR    #$FF    ;FORM 2'S COMPLEMENT
FC79  18        C          CLC
FC7A  69 01     C          ADC    #1
FC7C  4C FC6C   C          JMP    MWAIT1
                C ;
                C ;
FC7F  E6 42     C INCRE1   INC    REG4L   ;INCREMENT REG4L,43
FC81  D0 02     C          BNE    *+4
FC83  E6 43     C          INC    REG4H
FC85  A5 3D     C INCRE2   LDA    REG1H   ;COMPARE REG1L,3D WITH
FC87  C5 3F     C          CMP    REG2H   ;REG2L,3F
FC89  90 06     C          BLT    INCRE3
FC8B  D0 04     C          BNE    INCRE3
FC8D  A5 3C     C          LDA    REG1L
FC8F  C5 3E     C          CMP    REG2L
FC91  E6 3C     C INCRE3   INC    REG1L   ;CARRY HAS MEANING NOW
FC93  D0 02     C          BNE    *+4     ;INCREMENT REG1L,3D
FC95  E6 3D     C          INC    REG1H
FC97  60        C          RTS
                C ;
                C ;
FC98            C          DS     4
                C ;
                C ;
FC9C  A4 24     C          LDY    CHORZ
FC9E  20 FDE6   C          JSR    C800IN
FCA1  4C CA99   C          JMP    CLEOLZ
                C ;
                C ;
FCA4            C          DS     4
                C ;
                C ;
FCA8  4C FC77   C          JMP    MWAIT
                C ;
                C ;
FCAB            C          DS     9
                C ;
                C ;
FCB4  E6 42     C          INC    REG4L
FCB6  D0 02     C          BNE    *+4
FCB8  E6 43     C          INC    REG4H
FCBA  4C FC85   C          JMP    INCRE2
                C ;
                C ;
FCBD  A1 C2 D2  C BRKMES   ASC    "!BREAK AT $"
FCC0  C5 C1 CB  C
FCC3  A0 C1 D4  C
FCC6  A0 A4     C
                C ;
                C ;
FCC8  60        C          RTS
```

```
                        C ;
                        C ;
FCC9    38              C HDELAY  SEC             ;BYPASS SOME OF THE HEADER
FCCA    A9 B4           C         LDA     #180    ;ABOUT 3 SECONDS
FCCC    2C C1C3         C HDELA1  BIT     VERTSC  ;REFERENCE CLOCK
FCCF    30 FB           C         BMI     *-3
FCD1    2C C1C3         C         BIT     VERTSC
FCD4    10 FB           C         BPL     *-3
FCD6    E9 01           C         SBC     #1
FCD8    D0 F2           C         BNE     HDELA1
FCDA    A9 FF           C         LDA     ##$FF   ;PREPARE CHECK-SUM
FCDC    85 2E           C         STA     CHKSUM
FCDE    60              C         RTS
                        C ;
                        C ;
FCDF    A5 31           C SPACE   LDA     STOFLG  ;STORE OR EXAMINE?
FCE1    C9 99           C         CMP     ##$99
FCE3    D0 09           C         BNE     EXAMIN
FCE5    A0 00           C         LDY     #0      ;STORE
FCE7    A5 3E           C         LDA     REG2L
FCE9    91 3C           C         STA     (REG1L),Y
FCEB    4C FC85         C         JMP     INCRE2  ;UPDATA POINTER
FCEE    4C FE4B         C EXAMIN  JMP     MEMXM
                        C ;
                        C ;
FCF1    AD 03F3         C DEBUG1  LDA     RESTVR+1
FCF4    C9 E0           C         CMP     ##$E0
FCF6    D0 08           C         BNE     NONRST
FCF8    AD 03F2         C         LDA     RESTVR
FCFB    D0 03           C         BNE     NONRST
FCFD    4C D608         C         JMP     NODRIV
FD00    6C 03F2         C NONRST  JMP     (RESTVR)
                        C ;
                        C ;
                        C         .DEPHASE
                        C         ORG     $1D0C
                        C         .PHASE  $FD0C
                        C ;
FD0C    20 FDE6         C MRDKEY  JSR     C800IN
FD0F    20 CBA0         C         JSR     ENCUR
FD12    AD 06FB         C         LDA     TEMPA
FD15    6C 0038         C         JMP     (INSWL)
                        C ;
                        C ;
FD18                    C         DS      3
                        C ;
                        C ;
FD1B    20 FDE6         C MINKEY  JSR     C800IN
FD1E    84 35           C         STY     SAVEY   ;SAVE REGISTER Y
FD20    20 C84D         C         JSR     KEYIN   ;USE PASCAL READ KEY ROUTINE
FD23    20 CBDF         C         JSR     DECUR   ;ON EXIT DO THIS
FD26    A4 35           C         LDY     SAVEY   ;RECOVER Y
FD28    60              C         RTS
                        C ;
                        C ;
FD29                    C         DS      3       ;FILL SPACE
```

```
                        C ;
                        C ;
FD2C    20 FD0C         C MESC    JSR     MRDKEY  ;ESC WHAT?
FD2F    20 FDE6         C         JSR     C800IN  ;FOR NON-KEYBOARD INPUT CASE
FD32    20 CBF6         C         JSR     ESCX
                        C ;
                        C ;
FD35    20 FD0C         C MRDCHR  JSR     MRDKEY
FD38    C9 9B           C         CMP     #$9B    ;ESC ?
FD3A    F0 F0           C         BEQ     MESC
FD3C    C9 FF           C         CMP     #$FF    ;BREAK?
FD3E    D0 02           C         BNE     *+4
FD40    A9 83           C         LDA     #$83    ;YES, REPLACE WITH CTRL-C
FD42    60              C         RTS
                        C ;
                        C ;
FD43    AD C607         C CHKDIS  LDA     $C607   ;CHECK DISK CONTROLLER
FD46    C9 3C           C         CMP     #$3C
FD48    D0 0A           C         BNE     CHKDI1  ;IF NO, RETURN
FD4A    AD C603         C         LDA     $C603
FD4D    D0 05           C         BNE     CHKDI1
FD4F    68              C         PLA             ;IF THERE IS, BOOT DISK
FD50    68              C         PLA
FD51    4C C600         C         JMP     $C600
FD54    AD C507         C CHKDI1  LDA     $C507
FD57    C9 3C           C         CMP     #$3C
FD59    D0 0A           C         BNE     CHKDRT
FD5B    AD C503         C         LDA     $C503
FD5E    D0 05           C         BNE     CHKDRT
FD60    68              C         PLA
FD61    68              C         PLA
FD62    4C C500         C         JMP     $C500
FD65    60              C CHKDRT  RTS
                        C ;
                        C ;
FD66    60              C         RTS
                        C ;
                        C ;
FD67    20 FEC5         C MGETLZ  JSR     CROUT
FD6A    A5 33           C MGETLN  LDA     PROMPT  ;DISPLAY PROMPT SIGN
FD6C    20 FDED         C         JSR     MCOUT
FD6F    A2 00           C         LDX     #0      ;SET UP CHARACTER COUNTER
FD71    20 FD35         C GETLN1  JSR     MRDCHR  ;READ A CHARACTER
FD74    C9 98           C         CMP     #$98    ;CTRL-X?
FD76    F0 41           C         BEQ     CANCEL  ;IF YES, CANCEL THE LINE
FD78    C9 89           C         CMP     #$89    ;TAB?
FD7A    D0 03           C         BNE     *+5
FD7C    4C F9CA         C         JMP     TAB     ;YES
FD7F    C9 95           C         CMP     #$95    ;RIGHT ARROW?
FD81    D0 05           C         BNE     NRIGHT
FD83    AD 06FB         C         LDA     TEMPA   ;IF YES, PICK UP THE
                        C                         ;CHARACTER UNDER CURSOR POS.
FD86    09 80           C         ORA     #$80    ;ENSURE MSB=1
FD88    9D 0200         C NRIGHT  STA     KEYBUF,X
                        C ;
FD8B    4C FD91         C         JMP     *+6
```

```
FD8E   4C FEC5    C           JMP     CROUT
                  C ;
FD91   C9 88      C           CMP     #$88    ;LEFT ARROW?
FD93   D0 06      C           BNE     NOLEFT
FD95   CA         C           DEX             ;YES, BACK SPACE
FD96   CA         C           DEX
FD97   E0 FE      C           CPX     #$FE    ;BACK TOO MUCH?
FD99   F0 CC      C           BEQ     MGETLZ
FD9B   C9 8D      C NOLEFT    CMP     #$8D    ;CARRIAGE RETURN?
FD9D   D0 0B      C           BNE     NOTCR
FD9F   20 FDE6    C           JSR     C800IN
FDA2   48         C           PHA             ;IF YES, CLEAR TO END OF
FDA3   20 CA97    C           JSR     CLREOL  ;LINE
FDA6   68         C           PLA
FDA7   4C FDED    C           JMP     MCOUT   ;END OF GET-LINE!
FDAA   20 FDED    C NOTCR     JSR     MCOUT   ;RECALL CHARACTER ON SCREEN
FDAD   E8         C           INX             ;NEXT CHARACTER
FDAE   E0 F9      C           CPX     #249    ;EAT TOO MUCH?
FDB0   90 BF      C           BLT     GETLN1
FDB2   20 FF3A    C           JSR     MBELL   ;TAKE CARE OF HEALTH
FDB5   E0 FF      C           CPX     #255
FDB7   D0 B8      C           BNE     GETLN1
FDB9   A9 AF      C CANCEL    LDA     #"/"    ;DANGER
FDBB   20 FDED    C           JSR     MCOUT   ;STOP IMMEDIATELY
FDBE   4C FD67    C           JMP     MGETLZ
                  C ;
                  C ;
FDC1   85 4E      C PRBYTE    STA     RNDNOL  ;SAVE A FIRST
FDC3   4A         C           LSR
FDC4   4A         C           LSR
FDC5   4A         C           LSR
FDC6   4A         C           LSR
FDC7   20 FEB9    C           JSR     PRNHEX  ;PRINT HIGH NIBBLE
FDCA   A5 4E      C           LDA     RNDNOL  ;RESUME A
FDCC   29 0F      C           AND     #$0F
FDCE   4C FEB9    C           JMP     PRNHEX  ;PRINT LOW NIBBLE
                  C ;
                  C ;
                  C           .DEPHASE
                  C           ORG     $1DDA
                  C           .PHASE  $FDDA
                  C ;
FDDA   4C FDC1    C           JMP     PRBYTE
                  C ;
                  C ;
FDDD               C          DS      6
                  C ;
                  C ;
FDE3   4C FEB9    C           JMP     PRNHEX
                  C ;
                  C ;
                  C ;ROUTINE 'C800IN' MUST BE CALLED FIRST BEFORE ANY
                  C ;ROUTINE IN C8 IS CALLED
FDE6   8D CFFF    C C800IN    STA     ROMCLR  ;WE WANT "SLOT 3"'S C8
FDE9   8D C300    C           STA     $C300
FDEC   60         C           RTS
```

```
                        C ;
                        C ;
FDED    6C 0036         C MCOUT   JMP     (OUTSWL)
FDF0    20 FDE6         C MCOUT1  JSR     C800IN  ;EVERYTHING IS IN C8
FDF3    48              C         PHA             ;SAVE CHARACTER FIRST
FDF4    84 35           C         STY     SAVEY   ;SAVE Y
FDF6    29 FF           C         AND     #$FF    ;GET N FLAG
FDF8    10 06           C         BPL     MCOUT5  ;FOR CHARACTERS WITH MSB=0
FDFA    C9 A0           C         CMP     #$A0    ;CONTROL CHARACTER?
FDFC    90 08           C         BLT     MCOUT2
FDFE    25 32           C         AND     INVFLG  ;NO
FE00    20 CA18         C MCOUT5  JSR     STORAD  ;IT IS A DISPLAYABLE CHAR.
FE03    4C FE14         C         JMP     MCOUT3
                        C ;
FE06    29 7F           C MCOUT2  AND     #$7F    ;DO THIS FOR CONTROL CHAR.
FE08    C9 0D           C         CMP     #$0D    ;CARRIAGE RETURN?
FE0A    D0 05           C         BNE     MCOUT4
FE0C    20 CA01         C         JSR     VIDOU1
FE0F    A9 0A           C         LDA     #$0A    ;YES, ADD A LINE-FEED
FE11    20 CA01         C MCOUT4  JSR     VIDOU1
FE14    68              C MCOUT3  PLA             ;NOW, RESUME CHARACTER
FE15    A4 35           C         LDY     SAVEY
FE17    60              C         RTS
                        C ;
                        C ;
                        C ;
FE18    A2 10           C WRBYTE  LDX     #16     ;2 * 8 = 16
FE1A    0A              C WBYTE1  ASL
FE1B    20 F90D         C         JSR     WRTBIT
FE1E    D0 FA           C         BNE     WBYTE1
FE20    60              C         RTS
                        C ;
                        C ;
FE21                    C         DS      11
                        C ;
                        C ;
                        C ;BEFORE ENTERING, PLEASE SET Y=0
                        C ;
FE2C    B1 3C           C MOVE    LDA     (REG1L),Y
FE2E    91 42           C         STA     (REG4L),Y
FE30    20 FC7F         C         JSR     INCRE1
FE33    90 F7           C         BLT     MOVE    ;FINISHED YET?
FE35    60              C         RTS
                        C ;
                        C ;
FE36    4C F98A         C         JMP     VERIFY
                        C ;
                        C ;
FE39    A9 FD           C SCREEN  LDA     #)MCOUT1
FE3B    A0 F0           C         LDY     #(MCOUT1
FE3D    85 37           C         STA     OUTSWH
FE3F    84 36           C         STY     OUTSWL
FE41    60              C         RTS
                        C ;
                        C ;
FE42    A9 FD           C KBDBRD  LDA     #)MINKEY
```

```
FE44  A0 1B     C          LDY     #(MINKEY
FE46  85 39     C          STA     INSWH
FE48  84 38     C          STY     INSWL
FE4A  60        C KBDRTS   RTS
                C ;
                C ;
                C ;PRINT THE ADDRESS AND ITS CONTENTS
FE4B  A6 34     C MEMXM    LDX     SAVEX    ;CHECK LAST KEY
FE4D  F0 FB     C          BEQ     KBDRTS   ;DO NOTHING IF THE FIRST
                C                           ;KEY IS A SPACE
FE4F  CA        C          DEX
FE50  BD 0200   C          LDA     KEYBUF,X
FE53  49 B0     C          EOR     #$B0     ;A HEXADECIMAL DIGIT ?
FE55  C9 0A     C          CMP     #$0A
FE57  90 06     C          BLT     MEMXM1   ;IF YES, EXAMINE MEMORY
                C                           ;CONTENTS
FE59  69 88     C          ADC     #$88
FE5B  C9 FA     C          CMP     #$FA
FE5D  90 EB     C          BLT     KBDRTS   ;IF NO, EXIT
FE5F  A5 3F     C MEMXM1   LDA     REG2H
FE61  20 FDC1   C          JSR     PRBYTE
FE64  A5 3E     C          LDA     REG2L
FE66  20 FDC1   C          JSR     PRBYTE
FE69  A9 BD     C          LDA     #"="
FE6B  20 FDED   C          JSR     MCOUT
FE6E  A9 A0     C          LDA     #" "
FE70  20 FDED   C          JSR     MCOUT
FE73  A0 00     C          LDY     #0
FE75  B1 3E     C          LDA     (REG2L),Y
FE77  20 FDC1   C          JSR     PRBYTE
FE7A  4C FEC5   C          JMP     CROUT    ;FOLLOWS WITH A RETURN
                C ;
                C ;
FE7D            C          DS      3        ;FILL SPACE
                C ;
FE80  A9 3F     C SETINV   LDA     #$3F     ;TURN ON INVERSE MODE
FE82  EA        C          NOP              ;FILL SPACE
FE83  2C        C          DB      $2C      ;$2C='BIT'
FE84  A9 FF     C SETNOR   LDA     #$FF     ;NORMAL VIDEO MODE
FE86  85 32     C          STA     INVFLG
FE88  60        C          RTS
                C ;
                C ;
FE89  A9 00     C SETIN    LDA     #$00
FE8B  85 3E     C INPOT    STA     REG2L
FE8D  4C FEA6   C SETIN    JMP     INPUT
FE90  60        C          RTS
FE91  60        C          RTS
FE92  60        C          RTS
                C ;
                C ;
FE93  A9 00     C SETOU    LDA     #$00
FE95  85 3E     C OUTPOT   STA     REG2L
FE97  A5 3E     C SETOU    LDA     REG2L
FE99  F0 9E     C          BEQ     SCREEN
FE9B  29 07     C          AND     #$07
```

```
FE9D  09 C0     C           ORA     #$C0
FE9F  85 37     C           STA     OUTSWH
FEA1  A0 00     C           LDY     #$00
FEA3  84 36     C           STY     OUTSWL
FEA5  60        C           RTS
                C ;
FEA6  A5 3E     C INPUT     LDA     REG2L
FEA8  F0 98     C           BEQ     KBDBRD
FEAA  29 07     C           AND     #$07
FEAC  09 C0     C           ORA     #$C0
FEAE  85 39     C           STA     INSWH
FEB0  A0 00     C           LDY     #$00
FEB2  84 38     C           STY     INSWL
FEB4  60        C           RTS
FEB5  60        C           RTS
                C ;
                C ;
FEB6  4C FED0   C           JMP     GO
                C ;
                C ;
                C ;
FEB9  C9 0A     C PRNHEX    CMP     #10
FEBB  90 03     C           BLT     PRNHE1  ;0 TO 9?
FEBD  18        C           CLC
FEBE  69 07     C           ADC     #7      ;FOR A TO F
FEC0  69 B0     C PRNHE1    ADC     #$B0    ;FROM NO. TO CHAR.
FEC2  4C FDED   C           JMP     MCOUT   ;PRINT IT!
                C ;
                C ;
                C ;
FEC5  A9 8D     C CROUT     LDA     #$8D
FEC7  4C FDED   C           JMP     MCOUT
                C ;
                C ;
FECA            C           DS      3       ;FILL SPACE
                C ;
FECD  4C F8B5   C           JMP     MTSAVE
                C ;
                C ;
FED0  20 FF3F   C GO        JSR     GETBRG  ;RESUME REGISTER CONTENTS
FED3  A5 3E     C           LDA     REG2L   ;DESINATION
FED5  85 3A     C           STA     PCL
FED7  A5 3F     C           LDA     REG2H
FED9  85 3B     C           STA     PCH
FEDB  6C 003A   C           JMP     (PCL)   ;FLY!
                C ;
                C ;
FEDE  8A        C ERROR     TXA             ;SAVE X
FEDF  48        C           PHA
FEE0  A2 00     C           LDX     #0      ;THEN USE X AS COUNTER
FEE2  BD F883   C ERROR1    LDA     ERRORM,X;PRINT 'ERROR '
FEE5  20 FDED   C           JSR     MCOUT
FEE8  E8        C           INX
FEE9  E0 06     C           CPX     #6
FEEB  D0 F5     C           BNE     ERROR1
FEED  68        C           PLA
```

```
FEEE  AA        C          TAX
FEEF  4C FF3A   C          JMP     MBELL   ;FOLLOWS WITH A BEEP
                C ;
                C ;
                C          .DEPHASE
                C          ORG     $1EFD
                C          .PHASE  $FEFD
                C ;                .
FEFD  20 F965   C TLOAD    JSR     READPS  ;FIND LEADING SIGNAL
FF00  20 FCC9   C          JSR     HDELAY  ;BY PASS THE HEADER
FF03  20 F965   C          JSR     READPS  ;FIND SIGNAL AGAIN
FF06  20 FC38   C          JSR     FNDSTB  ;FIND STARTING BIT
FF09  20 F968   C          JSR     RT1BIT
FF0C  A0 39     C          LDY     #57     ;READ FIRST BYTE IN
FF0E  20 F956   C TLOAD1   JSR     RT8BIT
FF11  81 3C     C          STA     (REG1L,X)
FF13  45 2E     C          EOR     CHKSUM  ;UPDATE CHECK-SUM
FF15  85 2E     C          STA     CHKSUM
FF17  20 FC85   C          JSR     INCRE2  ;END OF TAPE READ?
FF1A  A0 33     C          LDY     #51     ;FOR THE REST BYTES
FF1C  90 F0     C          BLT     TLOAD1  ;NO, CONTINUE
FF1E  20 F956   C          JSR     RT8BIT  ;READ THE LAST BYTE IN
FF21  C5 2E     C          CMP     CHKSUM  ;THIS SHOULD BE THE CHECK-
                C                          ;SUM
FF23  F0 15     C          BEQ     MBELL   ;YES, CONGRATULATION!
FF25  D0 B7     C          BNE     ERROR
                C ;
                C ;
FF27            C          DS      6
                C ;
                C ;
FF2D  4C FEDE   C          JMP     ERROR
                C ;
                C ;
FF30            C          DS      10
                C ;
                C ;
FF3A  A9 87     C MBELL    LDA     #$87    ;CTRL-G
FF3C  4C FDED   C          JMP     MCOUT
                C ;
                C ;
FF3F  A6 46     C GETBRG   LDX     REGX
FF41  A4 47     C          LDY     REGY
FF43  A5 48     C          LDA     STATUS
FF45  48        C          PHA
FF46  A5 45     C          LDA     ACC
FF48  28        C          PLP
FF49  60        C          RTS
                C ;
                C ;
FF4A  86 46     C SAVE     STX     REGX    ;SAVE ALL REGISTERS
FF4C  84 47     C          STY     REGY
FF4E  85 45     C          STA     ACC
FF50  08        C          PHP
FF51  68        C          PLA
FF52  85 48     C          STA     STATUS
```

```
FF54  BA          C          TSX
FF55  86 49       C          STX     STACKP
FF57  D8          C          CLD
FF58  60          C          RTS
                  C ;
                  C ;
FF59  4C FF65     C          JMP     MON
                  C ;
                  C ;
FF5C              C          DS      9          ;FILL SPACE
                  C ;
FF65  20 FF3A     C MON      JSR     MBELL      ;BEEP!
FF68  EA          C          NOP                ;FILL SPACE
FF69  A9 AA       C MON1     LDA     #"*"       ;KERNEL PROMPT SIGN
FF6B  85 33       C          STA     PROMPT
FF6D  20 FD67     C          JSR     MGETLZ     ;GET A LINE
FF70  A2 00       C          LDX     #0         ;CLEAR REGISTER FIRST
FF72  CA          C          DEX
FF73  20 FFA7     C MON2     JSR     GETNUM     ;GET A NO. INTO REG2
FF76  20 FF8F     C          JSR     SEARCH     ;GET A NON-NUMBER
                  C                             ;IS IT A COMMAND?
FF79  90 EA       C          BCC     MON        ;NO, INFORM THE USER
FF7B  86 34       C          STX     SAVEX      ;STORE X FIRST
FF7D  A2 00       C          LDX     #$00       ;SUBROUTINES LIKE THIS
FF7F  C0 10       C          CPY     #16        ;SPACE OR RETURN?
FF81  B0 04       C          BGE     READY      ;IF YES, SKIP
FF83  A9 00       C          LDA     #$00       ;FOR OTHERS, CLEAR FLAG
FF85  85 31       C          STA     STOFLG
FF87  20 FFD6     C READY    JSR     GOSUB      ;GOTO THE ROUTINE
FF8A  A6 34       C          LDX     SAVEX      ;RESUME X
FF8C  4C FF73     C          JMP     MON2
                  C ;
                  C ;
FF8F  A0 11       C SEARCH   LDY     #ADRTBL-COMTBL ;SET UP POINTER
FF91  D9 FA87     C SEARC1   CMP     COMTBL-1,Y
FF94  D0 01       C          BNE     SEARC2
FF96  60          C          RTS
FF97  88          C SEARC2   DEY                ;UPDATE POINTER
FF98  D0 F7       C          BNE     SEARC1     ;END OF TABLE?
FF9A  18          C          CLC                ;SET UP CARRY FLAG
FF9B  60          C          RTS
                  C ;
                  C ;
                  C          .DEPHASE
                  C          ORG     $1FA7
                  C          .PHASE  $FFA7
                  C ;
FFA7  A9 00       C GETNUM   LDA     #$00
FFA9  85 3E       C          STA     REG2L
FFAB  85 3F       C          STA     REG2H
FFAD  E8          C GETNU1   INX                ;UPDATE POINTER
FFAE  BD 0200     C          LDA     KEYBUF,X   ;GET CHARACTER
FFB1  C9 B0       C          CMP     #$B0       ;IS IT A HEX NO.?
FFB3  90 20       C          BLT     NONNUM
FFB5  C9 C7       C          CMP     #$C7
FFB7  B0 1C       C          BGE     NONNUM
```

```
FFB9  C9 BA     C          CMP    #$BA
FFBB  30 06     C          BLT    HEXNUM
FFBD  E9 07     C          SBC    #7        ;A SHOULD FOLLOW 9
FFBF  C9 BA     C          CMP    #$BA
FFC1  30 12     C          BLT    NONNUM    ;BE CAREFUL!
                C                            ;: NOW BECOMES 3
FFC3  A0 04     C HEXNUM   LDY    #4        ;SET UP COUNTER
FFC5  49 B0     C          EOR    #$B0      ;TICK OUT THE HIGH BITS
FFC7  0A        C          ASL              ;SHIFT THIS NUMBER INTO
FFC8  0A        C          ASL              ;REGISTER 2
FFC9  0A        C          ASL
FFCA  0A        C          ASL
FFCB  0A        C NUMSHF   ASL
FFCC  26 3E     C          ROL    REG2L
FFCE  26 3F     C          ROL    REG2H
FFD0  88        C          DEY              ;SHIFT FINISHED?
FFD1  D0 F8     C          BNE    NUMSHF    ;NO, CONTINUE
FFD3  F0 D8     C          BEQ    GETNU1    ;YES, GO FOR THE NEXT
                C                            ;DIGIT
FFD5  60        C NONNUM   RTS
                C ;
                C ;
FFD6  88        C GOSUB    DEY      ;MODIFY POINTER
FFD7  98        C          TYA
FFD8  0A        C          ASL
FFD9  A8        C          TAY
FFDA  B9 FA9A   C          LDA    ADRTBL+1,Y       ;GET ROUTINE ADDRESS
FFDD  48        C          PHA
FFDE  B9 FA99   C          LDA    ADRTBL,Y
FFE1  48        C          PHA
FFE2  A0 00     C          LDY    #0
FFE4  60        C          RTS              ;GOTO THE ROUTINE
                C ;
                C ;
                C          .DEPHASE
                C          ORG    $1FFA
                C          .PHASE $FFFA
                C ;
                C ;
                C ;INTERRUPT VECTERS
                C ;
FFFA  03FB      C          DW     NMIADR
FFFC  FA62      C          DW     RESET
FFFE  FA40      C          DW     IR@BRK
                C ;
                C ;
                C          END
```

Macros:

Symbols:

0045	ACC	CB97	ADDY40	CB41	ADRCAL
C989	ADRESL	FA99	ADRTBL	CA20	ADVANC
D581	ASSIGN	FC2A	AUDOUT	E000	BASICC
E003	BASICW	CC9B	BASINP	CC5A	BASOU1
CC68	BASOU2	CC65	BASOU3	CC38	BASOUT
FB8A	BEAUTI	CA2E	BELL	CA33	BELL1
CA35	BELL2	C008	BKDROP	C018	BKGRND
FA4C	BREAK	FA65	BREAK1	FA6A	BREAK2
FCBD	BRKMES	03F0	BRKVER	CA54	BS
067B	BYTE	FDE6	C800IN	FDB9	CANCEL
FD54	CHKDI1	FD43	CHKDIS	FD65	CHKDRT
002E	CHKSUM	D5B3	CHOICE	CC4D	CHOK
0024	CHORZ	CB54	CHRDIS	CB6A	CHRDS1
047B	CHWHO	D5A4	CLEAR	CA9B	CLEOL2
CA99	CLEOLZ	CA83	CLEOP1	CA97	CLREOL
CA7F	CLREOP	CA76	CLSCRN	FC06	COLON
C836	COMPAT	FA88	COMTBL	CA29	CR
CAA4	CRLF	FEC5	CROUT	CC72	CURECH
0025	CVERT	CC44	CVOK	07FB	CVWHO
F831	DATA07	FB49	DATA10	FC16	DEAF
FCF1	DEBUG1	CBDF	DECUR	D578	DELAY
D57A	DELAY1	CC75	DONE	CC85	DONRTS
F828	DOT	CBA0	ENCUR	CBB2	ENCUR1
CBC2	ENCUR3	C309	ENTER	FBEA	ENTRY1
FC48	ENTRY2	FEDE	ERROR	FEE2	ERROR1
F883	ERRORM	C88A	ESCHK	C88E	ESCHK1
C89E	ESCHK2	C89B	ESCKRT	C984	ESCTB1
C97C	ESCTBL	CC8D	ESCWHO	CBF6	ESCX
CBFE	ESCX1	CBEC	ESCX2	CC0C	ESCXRT
FCEE	EXAMIN	FC58	F8HOME	0027	FKEYPH
0026	FKEYPL	C8D6	FKEND1	C8DA	FKEND2
C8E0	FKEND3	C8E6	FKEND4	C8F6	FKEND5
FC38	FNDSTB	FBC7	FRAMOI	FBCD	FRAMOO
D5FC	FSTIME	FF3F	GETBRG	FD71	GETLN1
FFAD	GETNU1	FFA7	GETNUM	FED0	GO
FFD6	GOSUB	C9F4	GOTOX	CA4B	GOTOXY
C9C9	GOXY	C9CF	GOXY1	FCCC	HDELA1
FCC9	HDELAY	FFC3	HEXNUM	CA6B	HOME
C1C2	HORZSC	F1BD	HRSEXT	C36E	ICHRDIS
C377	IENCUR	FC7F	INCRE1	FC85	INCRE2
FC91	INCRE3	C305	INENT	FBB1	INFKE1
FBAA	INFKEY	C34D	INMED	FE8B	INPOT
FEA6	INPUT	0039	INSWH	0038	INSWL
0032	INVFLG	CC30	IO	C30F	IORTS
FA40	IRQBRK	03FE	IRQVER	FE42	KBDBRD
FE4A	KBDRTS	C000	KEYBRD	C910	KEYBRT
0200	KEYBUF	4800	KEYFLG	C870	KEYHAN
C84D	KEYIN	C8FE	KEYINB	C860	KEYINH
C010	KEYSTR	002F	LASTBI	F8F0	LEADER
F88E	LENGTH	CAA7	LF	FABB	LOGO
FB63	LOOP1	F89C	LOWNIB	FF3A	MBELL
FDED	MCOUT	FDF0	MCOUT1	FE06	MCOUT2

FE14	MCOUT3	FE11	MCOUT4	FE00	MCOUT5
FE4B	MEMXM	FE5F	MEMXM1	FD2C	MESC
FD6A	MGETLN	FD67	MGETLZ	FD1B	MINKEY
FB40	MODE3	FF65	MON	FF69	MON1
FF73	MON2	FC4E	MOUTS1	FE2C	MOVE
FB1E	MPREAD	FD35	MRDCHR	FD0C	MRDKEY
FB4B	MSETWN	FB5B	MTABV	F8B5	MTSAVE
CC28	MUSTR1	FC22	MVTAB	FC77	MWAIT
FC6C	MWAIT1	FC6D	MWAIT2	CC86	NEWESC
03FB	NMIADR	D608	NODRIV	FD9B	NOLEFT
FFD5	NONNUM	FD00	NONRST	CCB4	NOP1CK
F23C	NORMAL	FDAA	NOTCR	CCAB	NOTCRW
FD88	NRIGHT	FFCB	NUMSHF	F8A5	ODD
F891	ODDEVN	F921	ODELAY	002F	OPCODL
F889	OPERR	C307	OUTENT	C34A	OUTMED
FE95	OUTPOT	0037	OUTSWH	0036	OUTSWL
FBD3	P3VECT	C064	PADDL0	00C5	PBANK1
00C6	PBANK2	00C7	PBANK3	00C8	PBANK4
003B	PCH	003A	PCL	C070	PDLRES
FB0B	PDLRTS	C878	POLKBD	C886	POLRTS
C9F1	POORX	C9E8	POORY	077B	POWER
FDC1	PRBYTE	FB02	PREAD1	FB0C	PREAD2
FB61	PRESP3	F941	PRNAX	FEC0	PRNHE1
FEB9	PRNHEX	F94E	PRNSP1	F948	PRNSPC
F944	PRNX	F940	PRNYX	0033	PROMPT
C9C6	PSCORT	CC0D	RAMOIN	CC2A	RAMOO1
CC17	RAMOOU	F959	RDBYT2	CCC1	RDKWHO
F965	READPS	FF87	READY	003D	REG1H
003C	REG1L	003F	REG2H	003E	REG2L
0043	REG4H	0042	REG4L	0046	REGX
0047	REGY	F229	RENEW	FA62	RESET
D571	RESETO	D5CE	RESET1	03F2	RESTVR
F83D	RETRTS	F833	RETURN	004F	RNDNOH
004E	RNDNOL	C310	ROAD	CFFF	ROMCLR
F968	RT1BIT	F956	RT8BIT	F979	RTBIT1
F987	RTBIT2	FF4A	SAVE	0778	SAVE1
0034	SAVEX	0035	SAVEY	C07C	SBANK1
C07D	SBANK2	C07E	SBANK3	C07F	SBANK4
002B	SBAS2H	002A	SBAS2L	0029	SBASH
0028	SBASL	FE39	SCREEN	CAC1	SCROL0
CACA	SCROL1	CAE7	SCROL2	CAF2	SCROL3
CB0A	SCROL4	CB16	SCROL5	CB31	SCROL6
CAB4	SCROLL	FF91	SEARC1	FF97	SEARC2
FF8F	SEARCH	FE89	SETCIN	FE93	SETCOU
FE8D	SETIN	FE80	SETINV	FE84	SETNOR
FE97	SETOU	D5CB	SETT40	FB39	SETTXT
C801	SETUP	F821	SETW40	F81E	SETW48
C822	SETWND	F812	SETWTH	FB4A	SIGNAT
C068	SONGEN	FBFA	SONINT	FBFC	SOUND1
FCDF	SPACE	C030	SPEAKR	FB6F	SRESTV
0100	STACK	0049	STACKP	0048	STATUS
0031	STOFLG	CA18	STORAD	C96E	SUBTBL
CB6E	SUBY40	C078	SYSTEM	F9CA	TAB
F9DA	TAB1	F9F3	TABJMP	F9F6	TABLE1
C060	TAPEIN	C020	TAPEOU	06FB	TEMPA
05FB	TEMPX	04FB	TEMPY	FB78	TEXT40

FB81	TEXT80	C028	TEXTCR	FB99	TITLE
FB9E	TITLE1	FBA9	TITRTS	FEFD	TLOAD
FF0E	TLOAD1	CB98	TOGGSK	F8C4	TSAVE1
CB81	TST40C	CCCA	TUGGLE	C1C5	TWOMHZ
0068	TXTABH	0067	TXTABL	057B	TXTMOD
CA5E	UP	03F8	USRADR	D5BC	USUAL
F992	VERCOM	F998	VERERR	F98A	VERIFY
C1C3	VERTSC	CA0F	VIDCON	CA01	VIDOLI
C9F9	VIDOUT	CA17	VIDRTS	C916	VIDWAI
CB3F	VTAB	C92C	VWDONE	C920	/WLOOP
FB2F	VZINIT	C054	VZPAG1	C056	VZSELF
C051	VZTEXT	C04C	VZTX40	C04F	VZTX80
C92D	WBANK	C934	WBANK1	C949	WBANK2
C95A	WBANK3	FE1A	WBYTE1	C350	WHO
0023	WNDBTM	0020	WNDLFT	0022	WNDTOP
0021	WNDWTH	FE18	WRBYTE	F90D	WRTBIT
F927	WTAPE	F846	XMEM8	F86B	XMEM81
F912	ZDELAY				

No Fatal error(s)

APPENDIX B

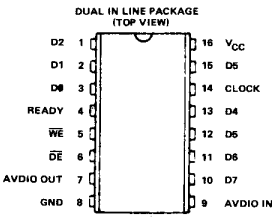
76489 SPECIFICATION

DESCRIPTION

The SN76489AN digital complex sound generator is an I^2L/Bipolar IC designed to provide low cost tone/noise generation capability in microprocessor systems. The SN76489AN is a data bus based I/O peripheral.

FEATURES

- 3 Programmable Tone Generators
- Programmable White Noise Generator
- Programmable Attenuation
- Simultaneous Sounds
- TTL Compatible
- Up to 4MHz Clock Input*
- Audio Summer



RECOMMENDED OPERATING CONDITIONS

PARAMETER	MIN	TYP	MAX	UNITS
Supply Voltage, V_{CC}	4.5	5.0	5.5	V
High Level Output Voltage, V_{OH} (pin 4)			5.5	V
Low Level Output Current, I_{OL} (pin 4)			2	mA
Operating Free-Air Temperature, T_A	0		70	°C

*Part SN76494N is identical to the SN76489A except that the maximum clock input frequency is 500kHz. A "divide-by-eight" stage is deleted from the input circuitry and only 4 clock pulses are required to load the data, compared to 32 pulses for the SN76489AN.

®Ackn : Mos Technology, Inc.

OPERATION

1. TONE GENERATORS

Each tone generator consists of a frequency synthesis section and an attenuation section. The frequency synthesis section requires 10 bits of information (F0-F9) to define half the period of the desired frequency (n). F0 is the most significant bit and F9 is the least significant bit. This information is loaded into a 10 stage tone counter, which is decremented at a N/16 rate where N is the input clock frequency. When the tone counter decrements to zero, a borrow signal is produced. This borrow signal toggles the frequency flip-flop and also reloads the tone counter. Thus, the period of the desired frequency is twice the value of the period register.

The frequency can be calculated by the following:

$$f = \frac{N}{32n}$$

where N = ref clock in Hz
n = 10 bit binary number

The output of the frequency flip-flop feeds into a four stage attenuator. The attenuator values, along with their bit position in the data word, are shown in Table 1. Multiple attenuation control bits may be true simultaneously. Thus, the maximum attenuation is 28 db.

Table 1 ATTENUATION CONTROL

BIT POSITION				
A0	A1	A2	A3	WEIGHT
0	0	0	1	2 db
0	0	1	0	4 db
0	1	0	0	8 db
1	0	0	0	16 db
1	1	1	1	OFF

2. NOISE GENERATOR

The Noise Generator consists of a noise source and an attenuator. The noise source is a shift register with an exclusive OR feedback network. The feedback network has provisions to protect the shift register from being locked in the zero state.

TABLE 2 NOISE FEEDBACK CONTROL

FB	CONFIGURATION
0	"Periodic" Noise
1	"White" Noise

Whenever the noise control register is changed, the shift register is cleared. The shift register will shift at one of four rates as determined by the two NF bits. The fixed shift rates are derived from the input clock.

TABLE 3 NOISE GENERATOR FREQUENCY CONTROL

BITS		
NFO	NFI	SHIFT RATE
0	0	N/512
0	1	N/1024
1	0	N/2048
1	1	Tone Generator #3 Out.

The output of the noise source is connected to a programmable attenuator as shown in Figure 4.

3. AUDIO SUMMER/OUTPUT BUFFER

The summer is a conventional operational amplifier summing circuit. It will sum the three tone generator outputs, noise generator output, and an external audio source. The output buffer will generate up to 10mA (see figure 2).

4. CPU to SN76489AN INTERFACE

The microprocessor interfaces with the SN76489AN by means of the 8 data liness and 3 control lines ($\overline{WE}$, $\overline{CE}$ and READY). Each tone generator requires 10 bits of information to select the frequency and 4 bits of information to select the attenuation. A frequency update requires a double byte transfer, while an attenuator update requires a single byte transfer.

If no other control registers on the chip are accessed, a tone generator may be rapidly updated by initially sending both bytes of frequency and register data, followed by just the second byte of data for succeeding values. The register address is latched on the chip, so the data will continue going into the same register. This allows the 6 most significant bits to be quickly modified for frequency sweeps.

5. CONTROL REGISTERS

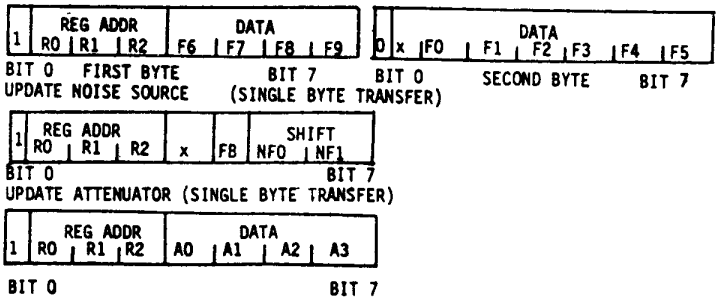
The SN76489AN has 8 internal registers which are used to control the 3 tone generators and the noise source. During all data transfers to the SN76489AN, the first byte contains a three bit field which determines the destination control register. The register address codes are shown in Table 4.

TABLE 4 REGISTER ADDRESS FIELD

R0	R1	R2	DESTINATION CONTROL REGISTER
0	0	0	Tone 1 Frequency
0	0	1	Tone 1 Attenuation
0	1	0	Tone 2 Frequency
0	1	1	Tone 2 Attenuation
1	0	0	Tone 3 Frequency
1	0	1	Tone 3 Attenuation
1	1	0	Noise Control
1	1	1	Noise Attenuation

6. DATA FORMATS

The formats required to transfer data are shown below.



electrical characteristics over recommended operating conditions (unless otherwise noted)						
PARAMETER	TEST COND.		MIN	TYP	MAX	UNITS
I_I Input Current	V_{IN} = GND to V_{CC}	$\overline{CE}$		-25	-175	µA
		DØ-D7, $\overline{WE}$, CLK		-10	-70	µA
V_{OL} Low Level Output Voltage	I_{OUT} = 2mA	READY		.25	.4	Volts
I_{CC} Supply Current	Outputs Open			30	50	mA
C_I Input Capacitance					15	pF
I_{OH} High Level Output Current	READY	Vcc≤5.0V			20	µA
		5.0V≤ Vcc≤5.5V			300	
V_{IH} High Level Input Voltage	DØ-D7, $\overline{WE}$, $\overline{CE}$, CLK		2			Volts
V_{IL} Low Level Input Voltage	DØ, D7, $\overline{WE}$, $\overline{CE}$, CLK				.8	Volts
R_M Trans-Impedance Amplifier Gain			12	17	22	Volts/mA
V_{BIAS} Audio Amplifier Input DC Bias Voltage	Pin 9 Open, All Attenuators Off			1.5		Volts
I_{EXT} External Audio Input	I_{EXT} as defined in Figure 2 I_{INT} = 0		-160		+15	µA
2dB Attenuation			1	2	3	dB
4dB Attenuation			3	4	5	dB
8dB Attenuation			7	8	9	dB
16dB Attenuation			15	16	17	dB

7. The microprocessor selects the SN76489AN by placing $\overline{CE}$ into the true state (low voltage). Unless $\overline{CE}$ is true, no data can occur. When $\overline{CE}$ is true, the $\overline{WE}$ signal strobes the contents of the data bus to the appropriate control register. The data bus contents must be valid at this time.

   The SN76489AN requires approximately 32 clock cycles to load the data into the control register. The open collector READY output is used to synchronize the microprocessor to this transfer and is pulled to the false state (low voltage) immediately following the leading edge of $\overline{CE}$. It is released to go to the true state (external pullup) when the data transfer is completed.

The data transfer timing is shown below.

DATA TRANSFER TIMING

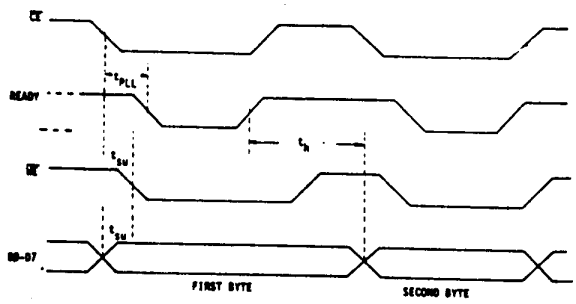


Figure 1.

TABLE 5 FUNCTION TABLE*

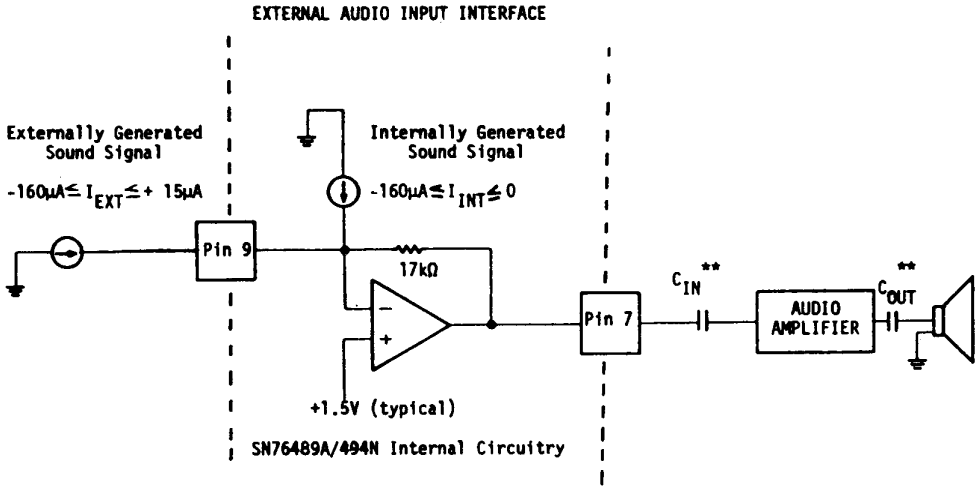
Inputs		Output
$\overline{CE}$	$\overline{WE}$	READY
L	L	L
L	H	L
H	L	H
H	H	H

* This table is valid when the device is:
(1) not being clocked, and
(2) is initialized by pulling $\overline{WE}$ and $\overline{CE}$ high.

8. PIN ASSIGNMENT

The table below defines the SN76489AN pin assignment and describes the function of each pin.

SIGNATURE	PIN	I/O	DESCRIPTION
$\overline{CE}$	6	IN	Chip Enable - when active (low) data may be transferred from CPU to the SN76489AN.
D0(MSB)	3	IN	D0 through D7 - Input data bus through which the control data is input.
D1	2	IN	
D2	1	IN	
D3	15	IN	
D4	13	IN	
D5	12	IN	
D6	11	IN	
D7	10	IN	
VCC	16	IN	Supply Voltage (5V nom)
GND	8	OUT	Ground Reference
CLOCK	14	IN	Input Clock
$\overline{WE}$	5	IN	Write Enable - when active (low), $\overline{WE}$ indicates that data is available from the CPU to the SN76489AN.
READY	4	OUT	When active (high), READY indicates that the data has been read. When READY is low, the microprocessor should enter a wait state until READY is high.
AIN	9	IN	Audio Signal In
AOUT	7	OUT	Audio Drive Out



*-160μA ≤ I_{EXT} + I_{INT} ≤ +15μA Where minimum I_{INT} = -40μA/tone (or noise) generator at 0 dB attenuation

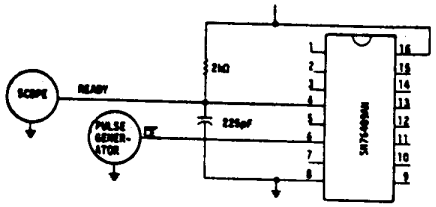
** These capacitance values are determined by the frequency response desired and the audio amplifier used.

Figure 2.

Switching Characteristics, V_{CC} = 5V, T_A = 25°C

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNITS
* $\overline{CE}$ to READY	C_L = 225pF				
t_{PLL}, 50% to 50%	R_L = 2K to V_{CC}		90	150	mS
f_{clock}, Input Clock Frequency	Clock Transition Time (10% to 90%) 10μS	DC	3.579	4	MHz
Setup Time, t_{su}	DATA W.R.T. $\overline{WE}$	0			nS
(see Figure 1)	$\overline{CE}$ W.R.T. $\overline{WE}$	0			nS
Hold Time, t_h (see Figure 1)	DATA W.R.T. READY	0			nS

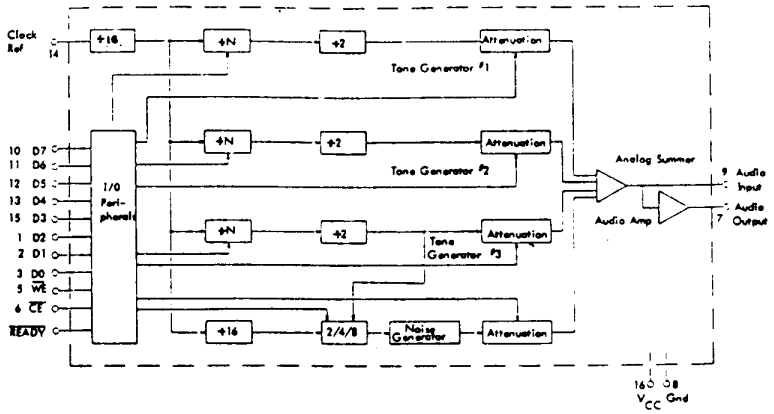
*$\overline{CE}$ Pulse: 0-3V, $t_{rise} \leq 7nS$, $t_{fall} \leq 7nS$



t_{PLL} TEST CIRCUIT

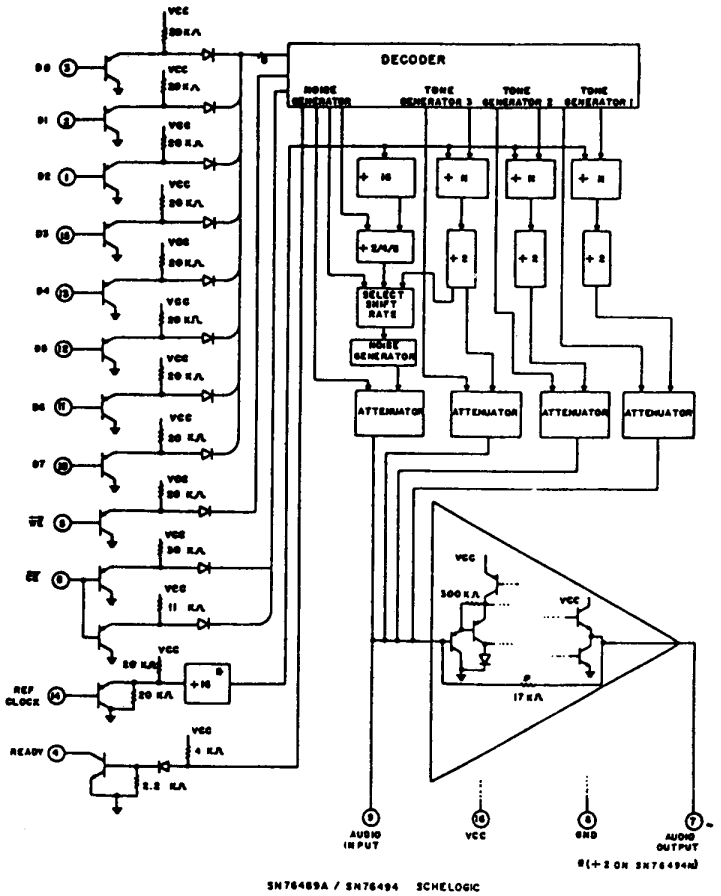
Figure 3.

BLOCK DIAGRAM



BLOCK DIAGRAM DESCRIPTION

This device consists of three programmable tone generators, a programmable noise generator, a clock scaler, individual generator attenuators and an audio summer output buffer. The SN76489AN has a parallel 8 bit interface through which the microprocessor transfers the data which controls the audio output.



SN76489A / SN76494 SCHELOGIC

Figure 4.

APPENDIX C

6502A SPECIFICATION

MCS6500 Microprocessors

- **Single +5V Supply**
- **N-Channel, Silicon-Gate, Depletion-Load Technology**
- **8-Bit Parallel Processing**
- **56 Instructions**
- **Decimal and Binary Arithmetic**
- **13 Addressing Modes**
- **Programmable Stack Pointer and Variable-Length Stack**
- **Usable With Any Type or Speed Memory**
- **1 or 2 MHz Operation**
- **Pipelined Architecture**

DESCRIPTION

The MCS6500 Series microprocessors represent the first totally software-compatible microprocessor family. This family of products includes a range of software-compatible microprocessors which provide a selection of addressable memory range, interrupt input options and on-chip clock oscillators and drivers. All of the microprocessors in the MCS6500 group are software-compatible within the group and are bus compatible with the M6800 product offering.

The family includes five microprocessors with on-board clock oscillators and drivers and four microprocessors driven by external clocks. The on-chip clock versions are aimed at high-performance, low-cost applications where single-phase inputs, crystal or RC inputs provide the time base. The external clock versions are geared for multi-processor system applications where maximum timing control is mandatory. All versions of the microprocessors are available in 1 MHz and 2 MHz ("A" suffix on product numbers) maximum operating frequencies.

MEMBERS OF THE FAMILY

Part Numbers		Clocks	Pins	$\overline{IRQ}$	$\overline{NMI}$	RDY	Addressing
Plastic	Ceramic						
MCS6502	MCS6502	On-Chip	40	√	√	√	16 (64 K)
MCS6503	MCS6503	"	28	√	√		12 (4 K)
MCS6504	MCS6504	"	28	√			13 (8 K)
MCS6505	MCS6505	"	28	√		√	12 (4 K)
MCS6506	MCS6506	"	28	√			12 (4 K)
MCS6507	MCS6507	"	28			√	13 (8 K)
MCS6512	MCS6512	External	40	√	√	√	16 (64 K)
MCS6513	MCS6513	"	28	√	√		12 (4 K)
MCS6514	MCS6514	"	28	√			13 (8 K)
MCS6515	MCS6515	"	28	√		√	12 (4 K)

PIN FUNCTIONS

Clocks (Φ1 and Φ2)
The MCS651X requires a two-phase, non-overlapping clock that runs at the V_{CC} voltage level.

The MCS650X clocks are supplied with an internal clock generator. The frequency of these clocks is externally controlled. Details of this feature are discussed in the MCS6502 portion of this data sheet.

Address Bus (A0-A15)
(See sections on each processor for respective address lines on those devices.)

These outputs are TTL-compatible, capable of driving one standard TTL load and 130pF

Data Bus (D0-D7)
Eight pins are used for the data bus. This is a bi-directional bus, transferring data to and from the device and peripherals. The outputs are three-state buffers capable of driving one standard TTL load and 130pF.

Data Bus Enable (DBE)
This TTL-compatible input allows external control of the three-state data output buffers and will enable the microprocessor bus driver when in the high state. In normal operation, DBE would be driven by the phase two (Φ2) clock, thus allowing data input from microprocessor only during Φ2. During the read cycle, the data bus drivers are internally disabled, becoming essentially an open circuit. To disable data bus drivers externally, DBE should be held low.

Ready (RDY)
This input signal allows the user to single-cycle the microprocessor on all cycles except write cycles. A negative transition to the low state during or coincident with phase one ($\Phi 1$) will halt the microprocessor with the output address lines reflecting the current address being fetched. This condition will remain through a subsequent phase two ($\Phi 2$) in which the Ready signal is low. This feature allows microprocessor interfacing with low-speed PROMS as well as fast (max. 2 cycle) Direct Memory Access (DMA). If Ready is low during a write cycle, it is ignored until the following read operation.

Interrupt Request ($\overline{IRQ}$)
This TTL-compatible signal requests that an interrupt sequence begin within the microprocessor. The microprocessor will complete the current instruction being executed before recognizing the request. At that time, the interrupt mask bit in the Status Code Register will be examined. If the interrupt mask flag is not set, the microprocessor will begin an interrupt sequence. The Program Counter and Processor Status Register are stored in the stack. The microprocessor will then set the interrupt mask flag high so that no further interrupts may occur. At the end of this cycle, the program counter low will be loaded from address FFFE, and program counter high from location FFFF, transferring program control to the memory vector located at these addresses. The RDY signal must be in the high state for any interrupt to be recognized. A 3KΩ external resistor should be used for proper wire-OR operation.

Non-Maskable Interrupt ($\overline{NMI}$)
A negative-going edge on this input requests that a non-maskable interrupt sequence be generated within the microprocessor.

$\overline{NMI}$ is an unconditional interrupt. Following completion of the current instruction, the sequence of operations defined for $\overline{IRQ}$ will be performed, regardless of the state of the interrupt mask flag. The vector address loaded into the program counter, low and high, are locations FFFA and FFFB respectively, transferring program control to the memory vector located at these addresses. The instructions loaded at these locations cause the microprocessor to branch to a non-maskable interrupt routine in memory.

$\overline{NMI}$ also requires an external 3KΩ register to V_{CC} for proper wire-OR operations.

Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupts lines that are sampled during $\Phi 2$ and will begin the appropriate interrupt routine on the $\Phi 1$ following the completion of the current instruction.

Set Overflow Flag (S.O.)
A NEGATIVE-going edge on this input sets the overflow bit in the Status Code Register. This signal is sampled on the trailing edge of $\Phi 1$.

SYNC
This output line is provided to identify those cycles during which the microprocessor is doing an OP CODE fetch. The SYNC line goes high during $\Phi 1$ of an OP CODE fetch and stays high for the remainder of that cycle. If the RDY line is pulled low during the $\Phi 1$ clock pulse in which SYNC went high, the processor will stop in its current state and will remain in the state until the RDY line goes high. In this manner, the SYNC signal can be used to control RDY to cause single instruction execution.

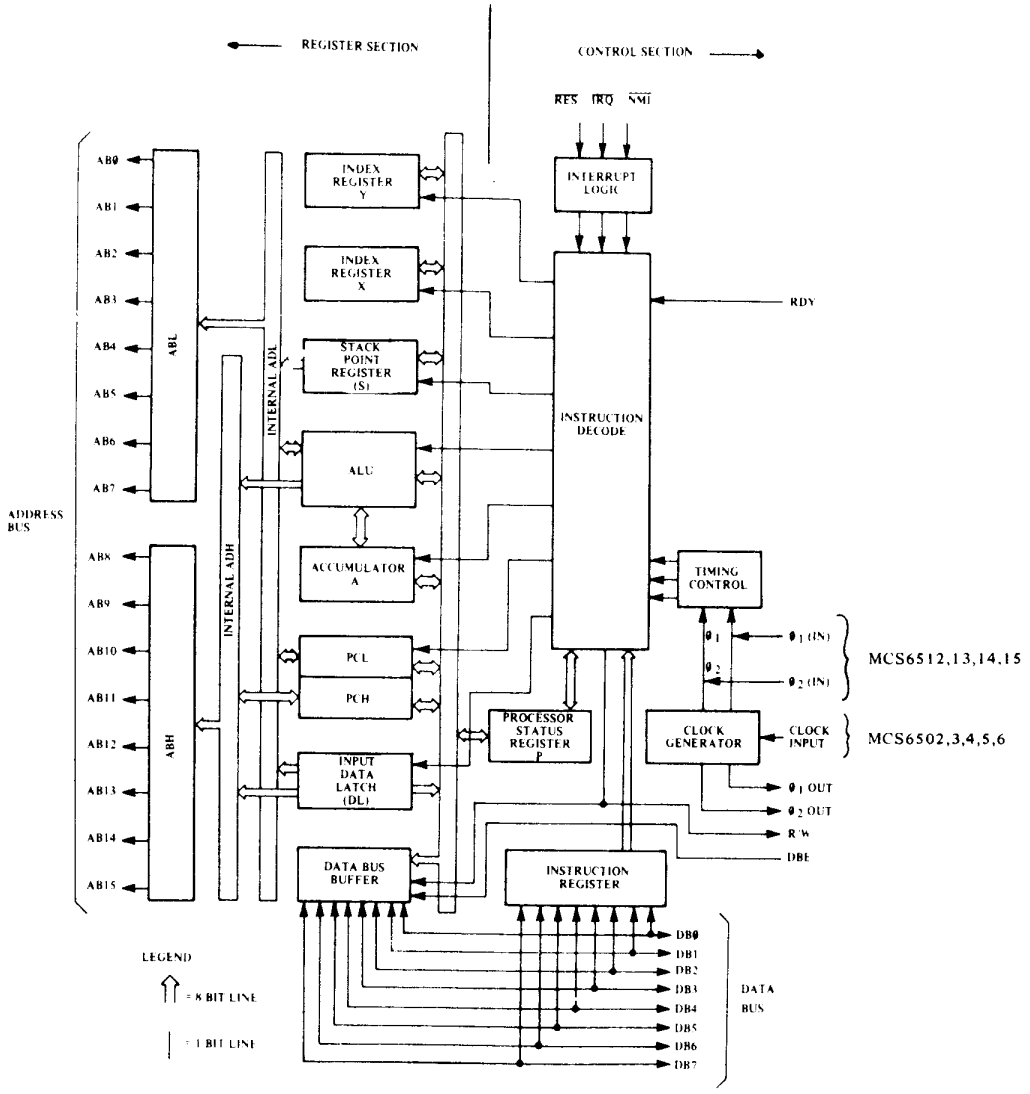
Reset
This input is used to reset or start the microprocessor from a power-down condition. During the time that this line is held low, writing to or from the microprocessor is inhibited. When a positive edge is detected on the input, the microprocessor will immediately begin the reset sequence.

After a system initialization time of six clock cycles, the mask interrupt flag will be set and the microprocessor will load the program counter from memory vector locations FFFC and FFFD. This is the start location for program control.

After V_{CC} reaches 4.75 volts in a power up routine, reset must be held low for at least two clock cycles. At this time the R/W and (SYNC) signal will become valid.

When the reset signal goes high following these two clock cycles, the microprocessor will proceed with the normal reset procedure detailed above.

INTERNAL ARCHITECTURE



NOTES

1. Clock Generator is not included on MCS6512, 13, 14, 15
2. Addressing Capability and control options vary with each of the MCS650(X) Products.

INSTRUCTION SET—OP CODES, Execution Time, Memory Requirements

Instructions		Immediate			Absolute			Zero Page			Accum.			Implied			(IND, X)			(IND), Y			Z, Page, X			ABS, X			ABS, Y			Relative			Indirect			Z, Page, Y			Condition Codes						
Mnemonic	Operation	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
A D C	A+M+C → A (4) (1)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										✓	✓	✓	-	-	✓
A N D	A ∧ M → A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										✓	✓	-	-	-	-
A S L					0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													✓	✓	✓	-	-	-
B C C	BRANCH ON C=0 (2)																														90	2	2							-	-	-	-	-	-
B C S	BRANCH ON C=1 (2)																														B0	2	2							-	-	-	-	-	-
B E Q	BRANCH ON Z=1 (2)																														F0	2	2							-	-	-	-	-	-
B I T	A ∧ M				2C	4	3	24	3	2																														M7	✓	-	-	-	M6
B M I	BRANCH ON N=1 (2)																														30	2	2							-	-	-	-	-	-
B N E	BRANCH ON Z=0 (2)																														D0	2	2							-	-	-	-	-	-
B P L	BRANCH ON N=0 (2)																														10	2	2							-	-	-	-	-	-
B R K	(See Fig. 1)													00	7	1																								-	-	-	-	-	-
B V C	BRANCH ON V=0 (2)																														50	2	2							-	-	-	-	-	-
B V S	BRANCH ON V=1 (2)																														70	2	2							-	-	-	-	-	-
C L C	0 → C													18	2	1																								-	-	0	-	-	-
C L D	0 → D													D8	2	1																								-	-	-	-	0	-
C L I	0 → I													58	2	1																								-	-	-	0	-	-
C L V	0 → V													B8	2	1																								-	-	-	-	-	0
C M P	A-M (1)	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										✓	✓	✓	-	-	-
C P X	X-M	E0	2	2	EC	4	3	E4	3	2																														✓	✓	✓	-	-	-
C P Y	Y-M	C0	2	2	CC	4	3	C4	3	2																														✓	✓	✓	-	-	-
D E C	M-1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3													✓	✓	-	-	-	-
D E X	X-1 → X													CA																										✓	✓	-	-	-	-
D E Y	Y-1 → Y													88																										✓	✓	-	-	-	-
E O R	A ∨ M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										✓	✓	-	-	-	-
I N C	M + 1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													✓	✓	-	-	-	-
I N X	X + 1 → X													E8	2	1																								✓	✓	-	-	-	-
I N Y	Y + 1 → Y													C8	2	1																								✓	✓	-	-	-	-
J M P	JUMP TO NEW LOC				4C	3	3																												6C	5	3				-	-	-	-
J S R	(See Fig. 2) JUMP SUB				20	6	3																																	-	-	-	-	-	-
L D A	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										✓	✓	-	-	-	-

Mnemonic	Operation	Immediate			Absolute			Zero Page			Accum.			Implied			(IND, X)			(IND), Y			Z, Page, X			ABS, X			ABS, Y			Relative			Indirect			Z, Page, Y			Condition Codes						
		OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
L D X	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3							B6	4	2	✓	✓	-	-	-	-
L D Y	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													✓	✓	-	-	-	-
L S R	0 → [7] → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													0	✓	✓	-	-	-
N O P	NO OPERATION													EA	2	1																								-	-	-	-	-	-
O R A	A ∨ M → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										✓	✓	-	-	-	-
P H A	A → M_S S-1 → S													48	3	1																								-	-	-	-	-	-
P H P	P → M_S S-1 → S													08	3	1																								-	-	-	-	-	-
P L A	S + 1 → S M_S → A													68	4	1																								✓	✓	-	-	-	-
P L P	S + 1 → S M_S → P													28	4	1																								(RESTORED)					
R O L	[7] ← [0] ← C ←				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													✓	✓	✓	-	-	-
R O R	→ C → [7] [0] →				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3													✓	✓	✓	-	-	-
R T I	(See Fig. 1) RTRN INT													40	6	1																								(RESTORED)					
R T S	(See Fig. 2) RTRN SUB													60	6	1																								-	-	-	-	-	-
S B C	A-M-$\overline{C}$ → A (1)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										✓	✓	(3)	-	-	✓
S E C	1 → C													38	2	1																								-	-	1	-	-	-
S E D	1 → D													F8	2	1																								-	-	-	-	1	-
S E I	1 → I													78	2	1																								-	-	-	1	-	-
S T A	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	90	5	3	99	5	3										-	-	-	-	-	-
S T X	X → M				8E	4	3	86	3	2																															-	-	-	-	-	-
S T Y	Y → M				8C	4	3	84	3	2													94	4	2													96	4	2	-	-	-	-	-	-
T A X	A → X													AA	2	1																								✓	✓	-	-	-	-
T A Y	A → Y													A8	2	1																								✓	✓	-	-	-	-
T S X	S → X													BA	2	1																								✓	✓	-	-	-	-
T X A	X → A													8A	2	1																								✓	✓	-	-	-	-
T X S	X → S													9A	2	1																								-	-	-	-	-	-
T Y A	Y → A													98	2	1																								✓	✓	-	-	-	-

NOTES

1. Add 1 to "N" if Page Boundary is Crossed
2. Add 1 to "N" if Branch Occurs to Same Page
   Add 2 to "N" if Branch Occurs to Different Page
3. Carry Not = Borrow
4. If in Decimal Mode Z Flag is Invalid
   Accumulator Must be Checked for Zero Result

X Index X

Y Index Y

A Accumulator

M Memory Per Effective Address

M_S Memory Per Stack Pointer

\+ Add

− Subtract

∧ AND

∨ OR

∀ Exclusive OR

✓ Modified

\- Not Modified

M_7 Memory Bit 7

M_6 Memory Bit 6

N No Cycles

No Bytes

INSTRUCTION SET—ALPHABETICAL SEQUENCE

ADC Add Memory to Accumulator with Carry
AND "AND" Memory with Accumulator
ASL Shift left One Bit (Memory or Accumulator)

BCC Branch on Carry Clear
BCS Branch on Carry Set
BEQ Branch on Result Zero
BIT Test Bits in Memory with Accumulator
BMI Branch on Result Minus
BNE Branch on Result not Zero
BPL Branch on Result Plus
BRK Force Break
BVC Branch on Overflow Clear
BVS Branch on Overflow Set

CLC Clear Carry Flag
CLD Clear Decimal Mode
CLI Clear Interrupt Disable Bit
CLV Clear Overflow Flag
CMP Compare Memory and Accumulator
CPX Compare Memory and Index X
CPY Compare Memory and Index Y

DEC Decrement Memory by One
DEX Decrement Index X by One
DEY Decrement Index Y by One

EOR "Exclusive-or Memory with Accumulator

INC Increment Memory by One
INX Increment Index by One
INY Increment Index Y by One

JMP Jump to New Location
JSR Jump to New Location Saving Return Address

LDA Load Accumulator with Memory
LDX Load Index X with Memory
LDY Load Index Y with Memory
LSR Shift One Bit Right (Memory or Accumulator)

NOP No Operation

ORA "OR" Memory with Accumulator

PHA Push Accumulator on Stack
PHP Push Processor Status on Stack
PLA Pull Accumulator from Stack
PLP Pull Processor Status from Stack

ROL Rotate One Bit Left (Memory or Accumulator)
ROR Rotate One Bit Right (Memory or Accumulator)
RTI Return from Interrupt
RTS Return from Subroutine

SBC Subtract Memory from Accumulator with Borrow
SEC Set Carry Flag
SED Set Decimal Mode
SEI Set Interrupt Disable Status
STA Store Accumulator in Memory
STX Store Index X in Memory
STY Store Index Y in Memory

TAX Transfer Accumulator to Index X
TAY Transfer Accumulator to Index Y
TSX Transfer Stack Pointer to Index X
TXA Transfer Index X to Accumulator
TXS Transfer Index X to Stack Pointer
TYA Transfer Index Y to Accumulator

ADDRESSING MODES

Accumulator Addressing. This form of addressing is represented with a one-byte instruction, implying an operation on the accumulator.

Immediate Addressing. In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

Absolute Addressing. In absolute addressing, the second byte of the instruction specifies the eight low-order bits of the effective address while the third byte specifies the eight high-order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

Zero Page Addressing. The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high-address byte. Careful use of the zero page can result in significant increase in code efficiency.

Indexed Zero Page Addressing. (X, Y indexing) – This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

Indexed Absolute Addressing. (X, Y indexing) – This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X", and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

Implied Addressing. In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

Relative Addressing. Relative addressing is used only with branch instructions and establishes a destination for the conditional branch. The second byte of the instruction becomes the operand which is an offset added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is −128 to +127 bytes from the next instruction.

Indexed Indirect Addressing. In indexed indirect addressing (referred to as Indirect, X), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents is the low-order eight bits of the effective address. The next memory location in page zero contains the high-order eight bits of the effective address. Both memory locations specifying the high and low-order bytes of the effective address must be in page zero.

Indirect Indexed Addressing. In indirect indexed addressing (referred to as Indirect, Y), the second byte of the instruction points to a memory location in page zero. The contents on this memory location is added to the contents of the Y index register, the result being the low-order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high-order eight bits of the effective address.

Absolute Indirect. The second byte of the instruction contains the low-order eight bits of a memory location. The high-order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low-order byte of the effective address. The next memory location contains the high-order byte of the effective address which is loaded into the 16-bit program counter.

ABSOLUTE MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	Vdc
Input Voltage	V_{IN}	−0.3 to +7.0	Vdc
Operating Temperature	T_A	0 to +70	°C
Storage Temperature	T_{STG}	−55 to +150	°C

CAUTION

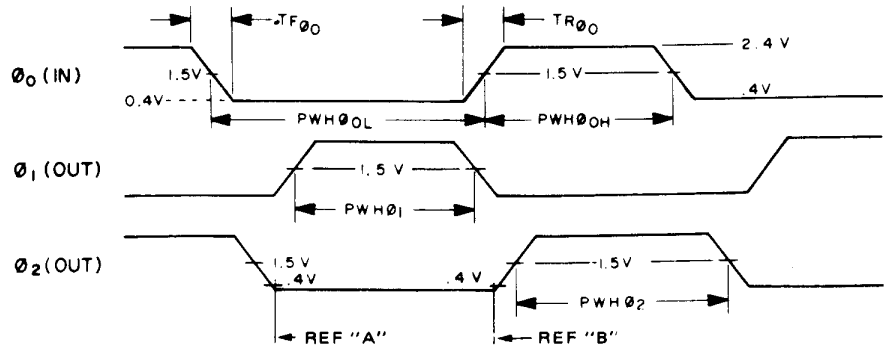
This device contains input protection against damage due to high static voltages or electric fields; however, precautions should be taken to avoid application of voltages higher than the maximum rating.

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0V ± 5%, V_{SS} = 0, T_A = 25°C)
$\emptyset_1$, $\emptyset_2$ applies to MCS6512, 13, 14, 15, $\emptyset_{o\,(in)}$ applies to MCS6502, 03, 04, 05 and 06

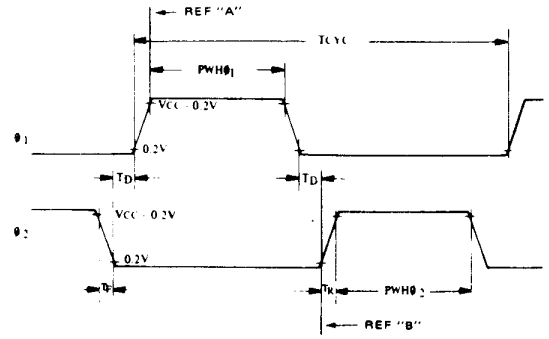
Symbol	Parameter	Min	Typ	Max	Unit	Test Condition
V_{IH}	Input High Voltage	V_{SS} + 2.4 V_{CC} − 0.2		V_{CC} V_{CC} + 0.25	Vdc	Logic, $\emptyset_{o\,(in)}$ $\emptyset_1$, $\emptyset_2$
V_{IL}	Input Low Voltage	V_{SS} − 0.3 V_{SS} − 0.3		V_{SS} + 0.4 V_{SS} + 0.2	Vdc	Logic, $\emptyset_{o\,(in)}$ $\emptyset_1$, $\emptyset_2$
V_{IHT}	Input High Threshold Voltage	V_{SS} + 2.0			Vdc	$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O.
V_{ILT}	Input Low Threshold Voltage			V_{SS} + 0.8	Vdc	$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O.
I_{IN}	Input Leakage Current			 2.5 100 10.0	 μA μA μA	(V_{IN} = 0 to 5.25V, V_{CC} = 0) Logic (Excl. RDY, S.O.) $\emptyset_1$, $\emptyset_2$ $\emptyset_{o\,(in)}$
I_{TSI}	Three-State (Off State) Input Current			10	μA	(V_{IN} = 0.4 to 2.4V, V_{CC} = 5.25V) Data Lines
V_{OH}	Output High Voltage	V_{SS} + 2.4			Vdc	(I_{LOAD} = −100μAdc, V_{CC} = 4.75V) SYNC, Data, A0-A15, R/W
V_{OL}	Output Low Voltage			V_{SS} + 0.4	Vdc	(I_{LOAD} = 1.6mAdc, V_{CC} = 4.75V) SYNC, Data, A0-A15, R/W
P_D	Power Dissipation		.25	.70	W	
C C_{IN} C_{OUT} $C_{\emptyset o\,(in)}$ $C_{\emptyset 1}$ $C_{\emptyset 2}$	Capacitance		 30 50	 10 15 12 50 50 80	pF	(V_{IN} = 0, T_A = 25°C, f = 1MHz) Logic Data A0-A15, R/W, SYNC $\emptyset_{o\,(in)}$ $\emptyset_1$ $\emptyset_2$

NOTE
$\overline{IRQ}$ and $\overline{NMI}$ require 3K pull-up resistors.

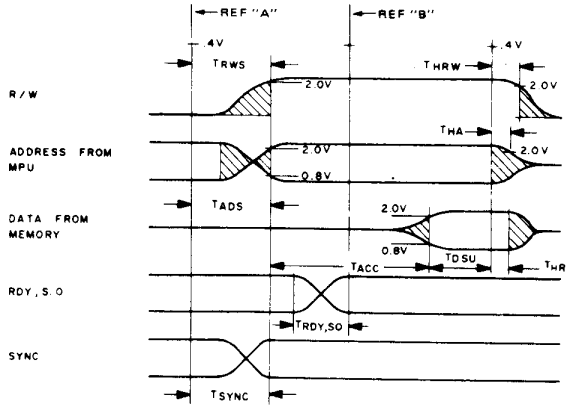
CLOCK TIMING—MCS6502, 03, 04, 05, 06



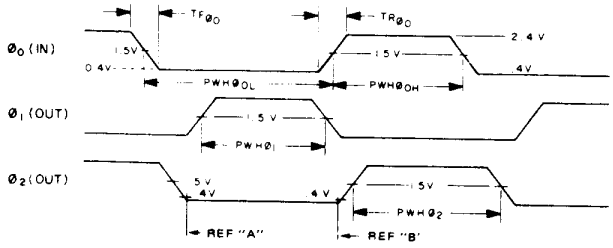
CLOCK TIMING—MCS6512, 13, 14, 15



TIMING FOR READING DATA FROM MEMORY OR PERIPHERALS



TIMING FOR WRITING DATA TO MEMORY OR PERIPHERALS



NOTE
"REF " means Reference Points on clocks.

1 MHz TIMING

CLOCK TIMING—MCS6512, 13, 14, 15

Symbol	Characteristic		Min	Typ	Max	Unit
T_{CYC}	Cycle Time		1000			nsec
PWH $\phi 1$ PWH $\phi 2$	Clock Pulse Width (Measured at V_{CC} − 0.2 V)	$\phi 1$ $\phi 2$	430 470			nsec
T_F	Fall Time (Measured from 0.2 V to V_{CC} − 0.2 V)				25	nsec
T_D	Delay Time Between Clocks (Measured at 0.2 V)		0			nsec

CLOCK TIMING—MCS6502, 03, 04, 05, 06

Symbol	Characteristic	Min	Typ	Max	Unit
T_{CYC}	Cycle Time	1000			ns
PWHϕ_o	$\phi_{o\,(IN)}$ Pulse Width (measured at 1.5 V)	460		520	ns
TRϕ_o, TFϕ_o	$\phi_{o\,(IN)}$ Rise, Fall Time			10	ns
T_D	Delay Time Between Clocks (measured at 1.5 V)	5			ns
PWHϕ_1	$\phi_{1\,(OUT)}$ Pulse Width (measured at 1.5 V)	PWHϕ_{oL}−20		PWHϕ_{oL}	ns
PWHϕ_2	$\phi_{1\,(OUT)}$ Pulse Width (measured at 1.5 V)	PWHϕ_{oH}−40		PWHϕ_{oH}−10	ns
T_R, T_F	$\phi_{1\,(OUT)}$, $\phi_{2\,(OUT)}$ Rise, Fall Time (measured .8 V to 2.0 V) (Load = 30pF + 1 TTL)			25	ns

READ/WRITE TIMING

Symbol	Characteristic	Min	Typ	Max	Unit
T_{RWS}	Read/Write Setup Time From MCS6500		100	300	ns
T_{ADS}	Address Setup Time From MCS6500		100	300	ns
T_{ACC}	Memory Read Access Time			575	ns
T_{DSU}	Data Stability Time Period	100			ns
T_{HR}	Data Hold Time − Read	10			ns
T_{HW}	Data Hold Time − Write	30	60		ns
T_{MDS}	Data Setup Time From MCS6500		150	200	ns
T_{RDY}	RDY, S.O. Setup Time	100			ns
T_{SYNC}	SYNC Setup Time From MCS6500			350	ns
T_{HA}	Address Hold Time	30	60		ns
T_{HRW}	R/W Hold Time	30	60		ns

2 MHz TIMING

CLOCK TIMING—MCS6512, 13, 14, 15, 16

Symbol	Characteristic		Min	Typ	Max	Unit
T_{CYC}	Cycle Time		500			nsec
PWH $\phi 1$ PWH $\phi 2$	Clock Pulse Width (Measured at V_{CC} − 0.2 V)	$\phi 1$ $\phi 2$	215 235			nsec
T_F	Fall Time (Measured from 0.2 V to V_{CC} − 0.2 V)				12	nsec
T_D	Delay Time Between Clocks (Measured at 0.2 V)		0			nsec

CLOCK TIMING—MCS6502, 03, 04, 05, 06

Symbol	Characteristic	Min	Typ	Max	Unit
T_{CYC}	Cycle Time	500			ns
$PWH\phi_o$	$\phi_{o\,(IN)}$ Pulse Width (measured at 1.5 V)	240		260	ns
$TR\phi_o$, $TF\phi_o$	$\phi_{o\,(IN)}$ Rise, Fall Time			10	ns
T_D	Delay Time Between Clocks (measured at 1.5 V)	5			ns
$PWH\phi_1$	$\phi_{1\,(OUT)}$ Pulse Width (measured at 1.5 V)	$PWH\phi_{oL}-20$		$PWH\phi_{oL}$	ns
$PWH\phi_2$	$\phi_{2\,(OUT)}$ Pulse Width (measured at 1.5 V)	$PWH\phi_{oH}-40$		$PWH\phi_{oH}-10$	ns
T_R, T_F	$\phi_{1\,(OUT)}$, $\phi_{2\,(OUT)}$ Rise, Fall Time (measured .8 V to 2.0 V) (Load = 30pF + 1 TTL)			25	ns

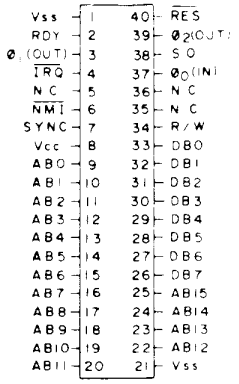
READ/WRITE TIMING

Symbol	Characteristic	Min	Typ	Max	Unit
T_{RWS}	Read/Write Setup Time From MCS6500A		100	150	ns
T_{ADS}	Address Setup Time From MCS6500A		100	150	ns
T_{ACC}	Memory Read Access Time			300	ns
T_{DSU}	Data Stability Time Period	50			ns
T_{HR}	Data Hold Time – Read	10			ns
T_{HW}	Data Hold Time – Write	30	60		ns
T_{MDS}	Data Setup Time From MCS6500A		75	100	ns
T_{RDY}	RDY, S.O. Setup Time	50			ns
T_{SYNC}	SYNC Setup Time From MCS6500A			175	ns
T_{HA}	Address Hold Time	30	60		ns
T_{HRW}	R/W Hold Time	30	60		ns

SPECIFIC VERSION FEATURES

(40 Pin Package)

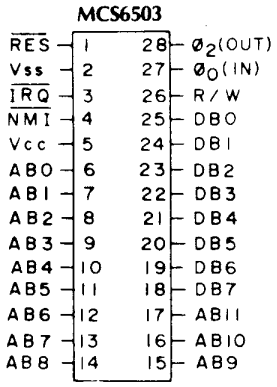
MCS6502



FEATURES
- 65K Addressable Bytes of Memory
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- On-the-chip Clock
  - ✓ TTL Level Single Phase Input
  - ✓ RC Time Base Input
  - ✓ Crystal Time Base Input
- SYNC Signal
  (can be used for single instruction execution)
- RDY Signal
  (can be used for single cycle execution)
- Two Phase Output Clock for Timing of Support Chips

(28 Pin Package)

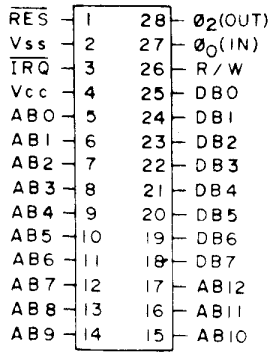
MCS6503



FEATURES
- 4K Addressable Bytes of Memory (AB00-AB11)
- On-the-chip Clock
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- 8 Bit Bi-Directional Data Bus

(28 Pin Package)

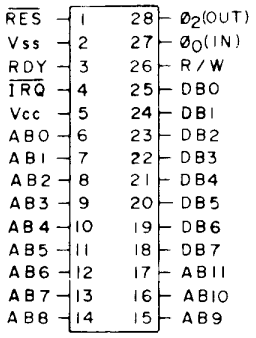
MCS6504



FEATURES
- 8K Addressable Bytes of Memory (AB00–AB12)
- On-the-chip Clock
- $\overline{IRQ}$ Interrupt
- 8 Bit Bi-Directional Data Bus

(28 Pin Package)

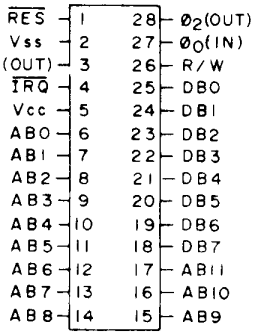
MCS6505



FEATURES
- **4K Addressable Bytes of Memory (AB00–AB11)**
- **On-the-chip Clock**
- **$\overline{IRQ}$ Interrupt**
- **RDY Signal**
- **8 Bit Bi-Directional Data Bus**

(28 Pin Package)

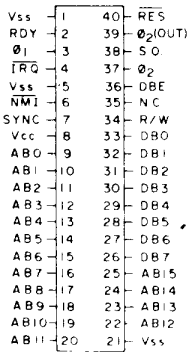
MCS6506



FEATURES
- **4K Addressable Bytes of Memory (AB00–AB11)**
- **On-the-chip Clock**
- **$\overline{IRQ}$ Interrupt**
- **Two phases off**
- **8 Bit Bi-Directional Data Bus**

(40 Pin Package)

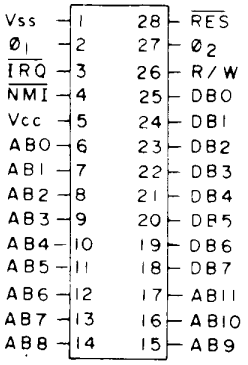
MCS6512



FEATURES
- **65K Addressable Bytes of Memory**
- **$\overline{IRQ}$ Interrupt**
- **$\overline{NMI}$ Interrupt**
- **RDY Signal**
- **8 Bit Bi-Directional Data Bus**
- **SYNC Signal**
- **Two phase input**
- **Data Bus Enable**

(28 Pin Package)

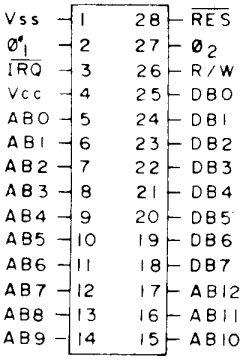
MCS6513



FEATURES
- 4K Addressable Bytes of Memory (AB00–AB11)
- Two phase clock input
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- 8 Bit Bi-Directional Data Bus

(28 Pin Package)

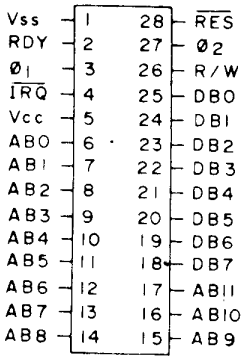
MCS6514



FEATURES
- 8K Addressable Bytes of Memory (AB00–AB12)
- Two phase clock input
- $\overline{IRQ}$ Interrupt
- 8 Bit Bi-Directional Data Bus

(28 Pin Package)

MCS6515

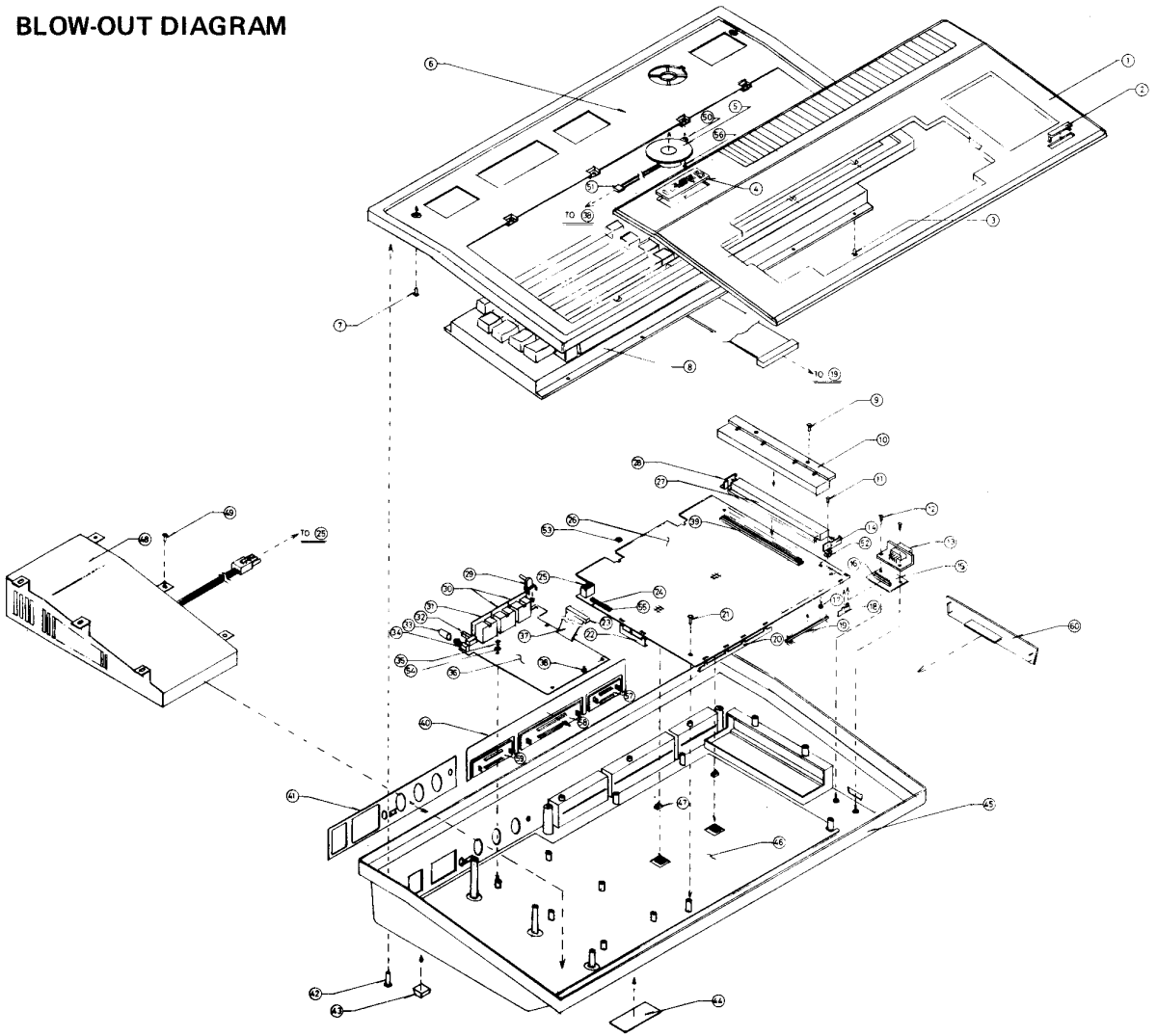


FEATURES
- 4K Addressable Bytes of Memory (AB00–AB11)
- Two phase clock input
- $\overline{IRQ}$ Interrupt
- 8 Bit Bi-Directional Data Bus

APPENDIX D

BLOW-OUT DIAGRAM

BLOW-OUT DIAGRAM

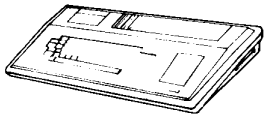


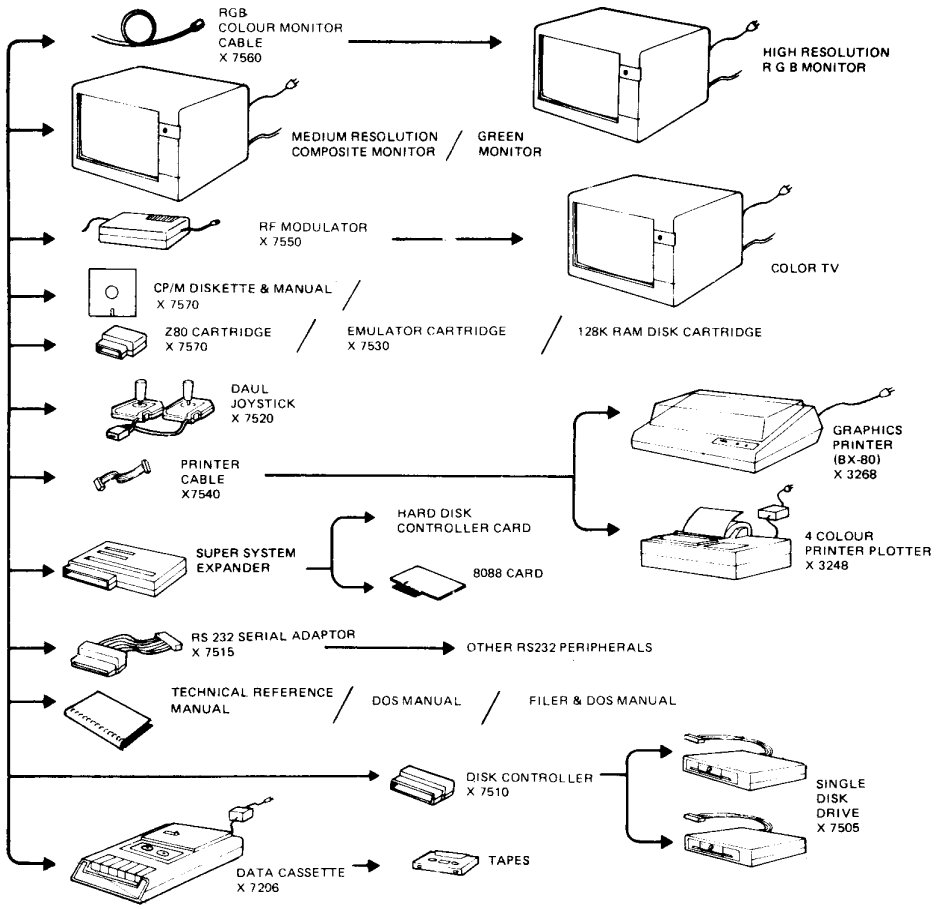
APPENDIX E

SYSTEM DIAGRAM

SYSTEM DIAGRAM

MAIN UNIT

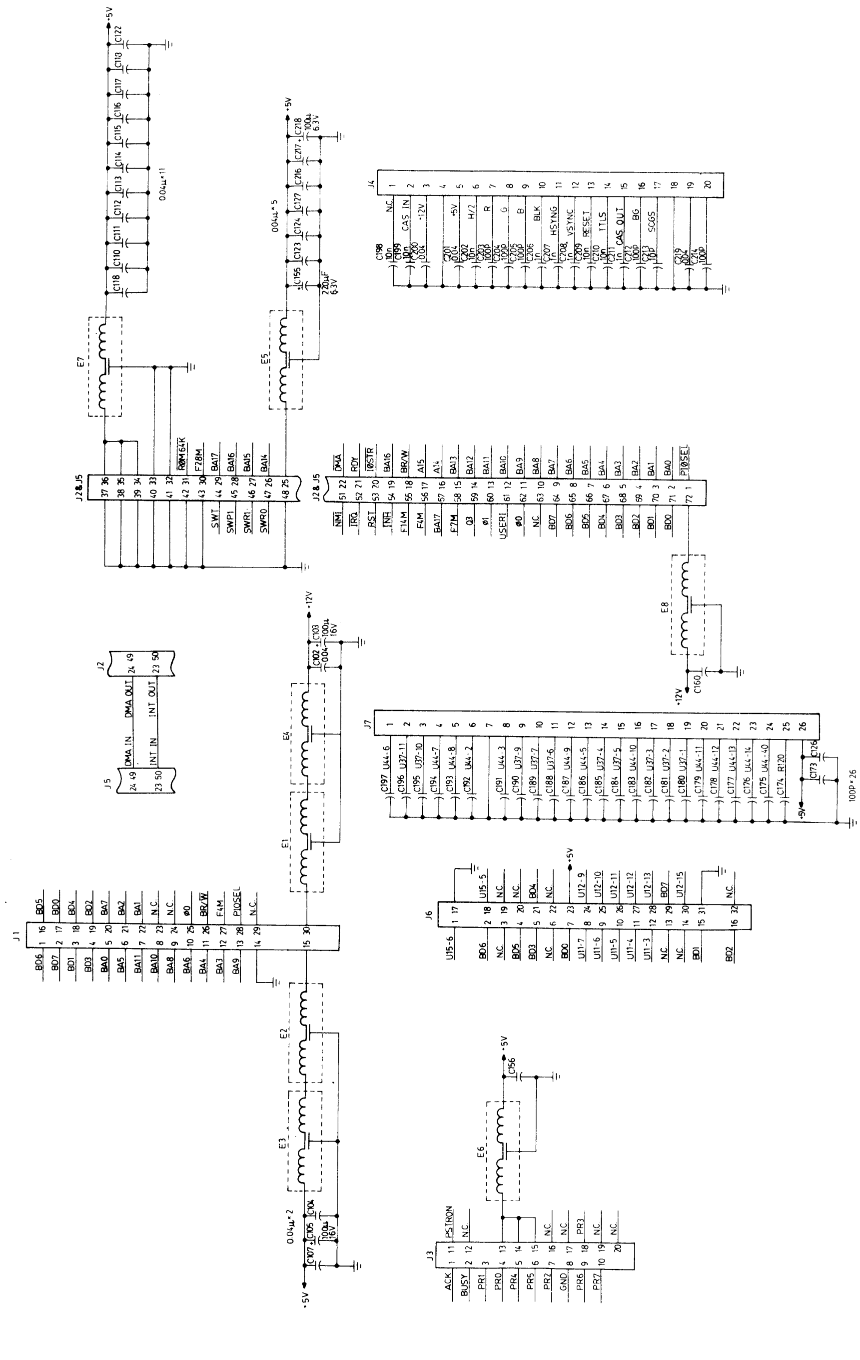




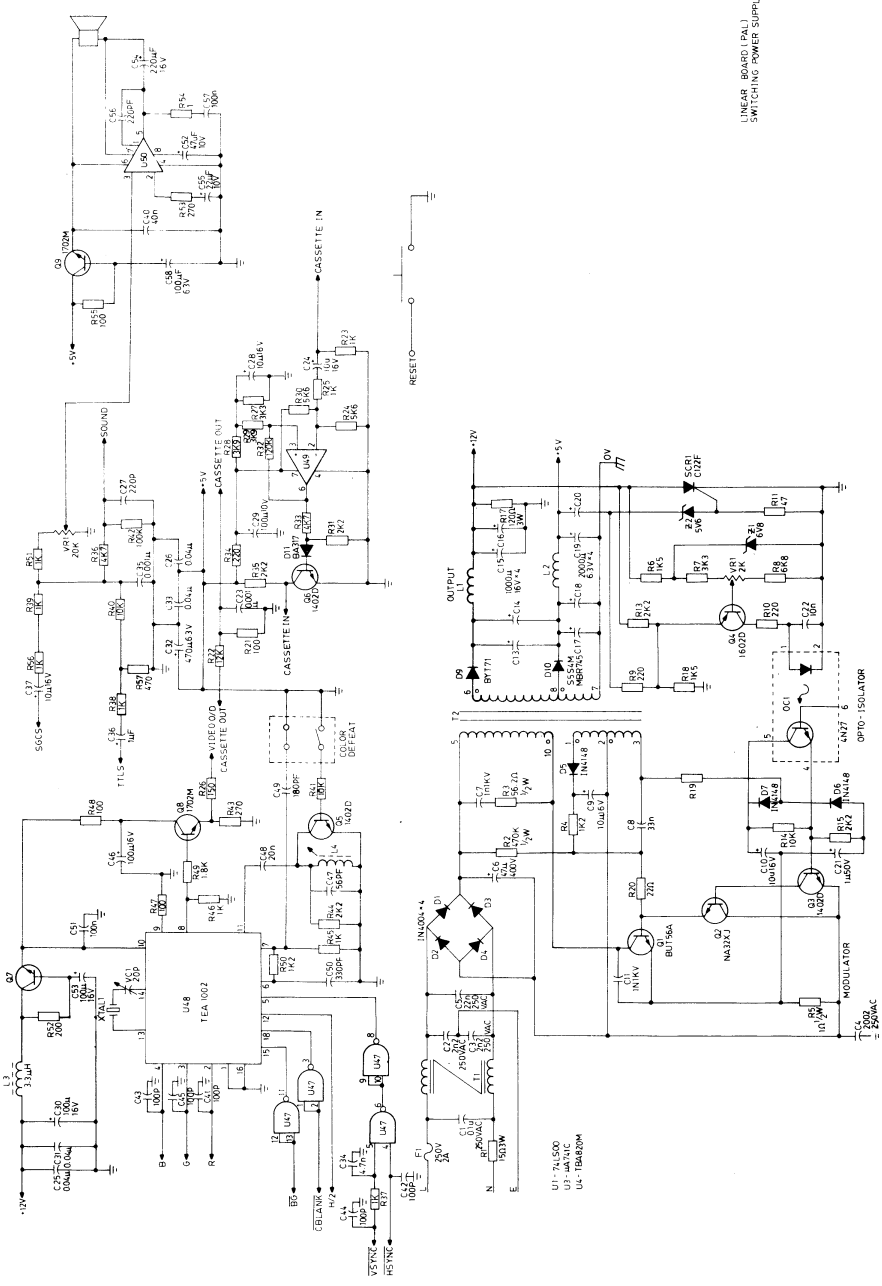
APPENDIX F

CIRCUIT DIAGRAMS (PAL)

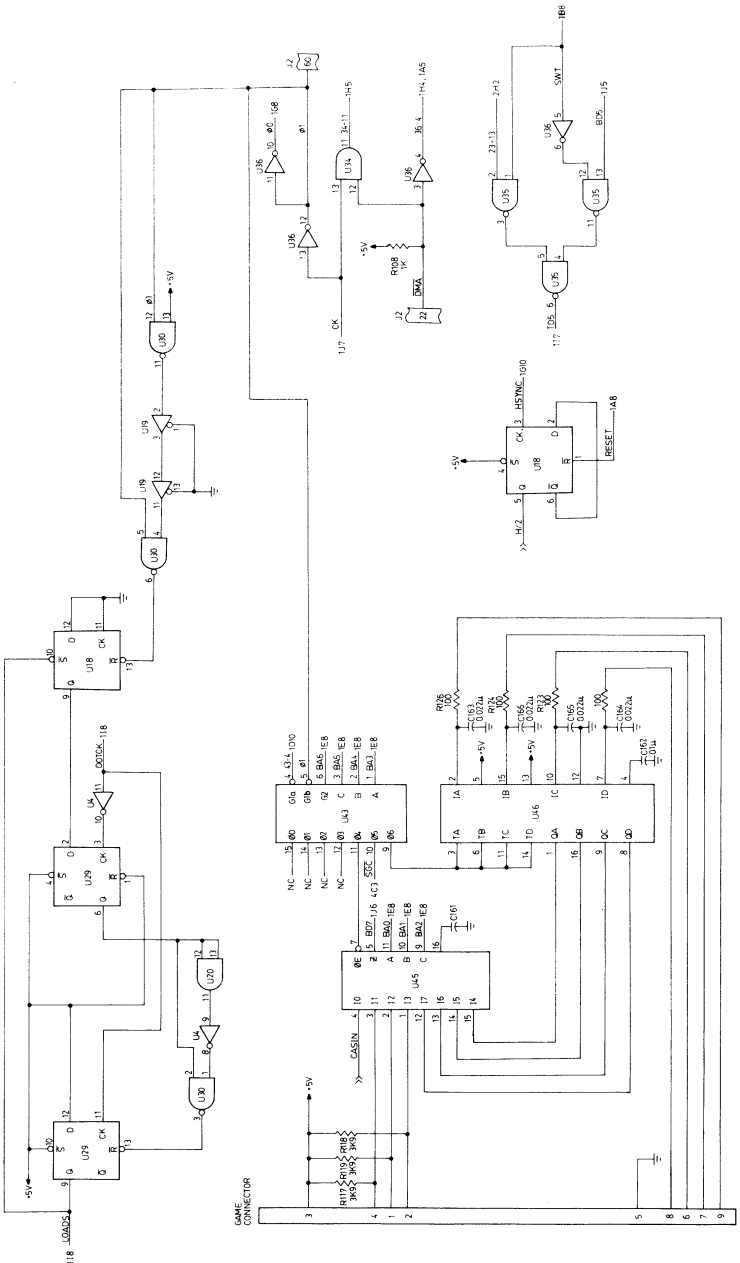
CONNECTOR



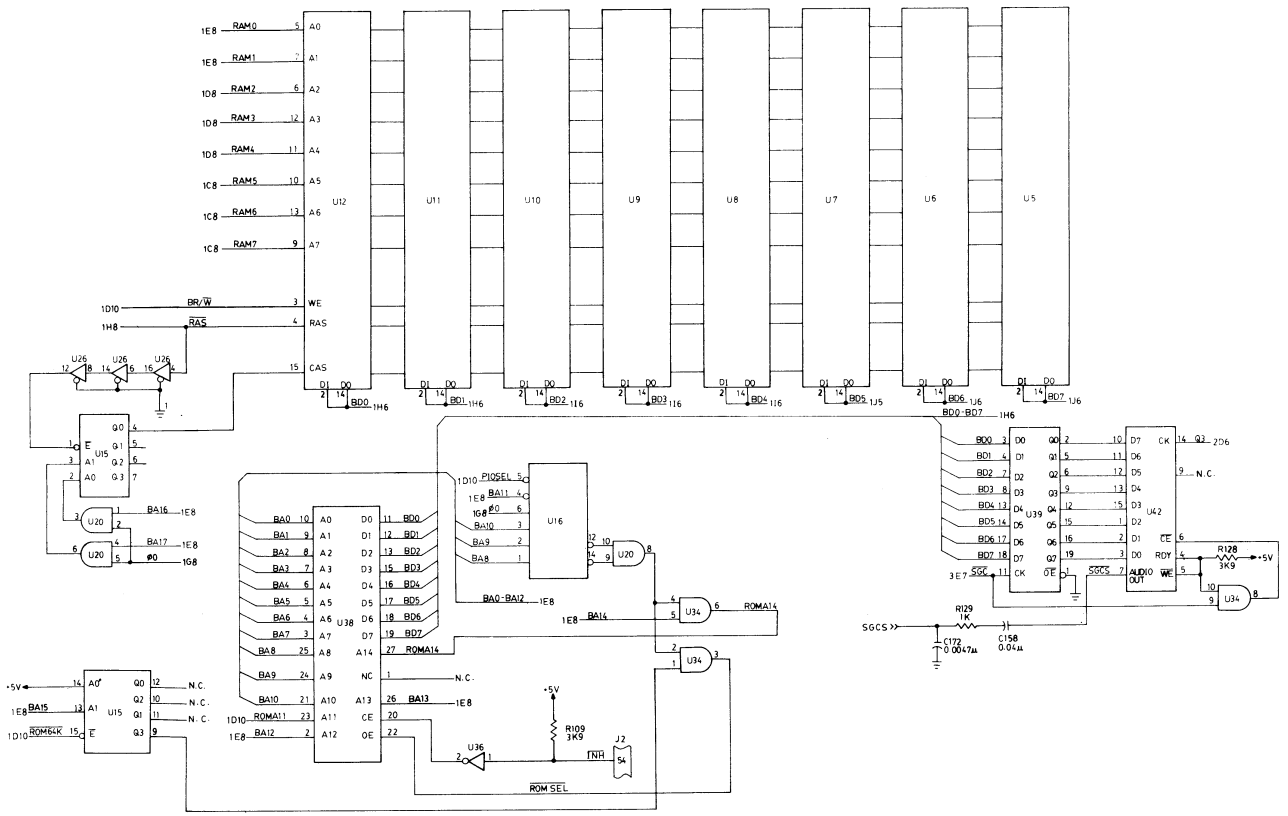
CIRCUIT DIAGRAM



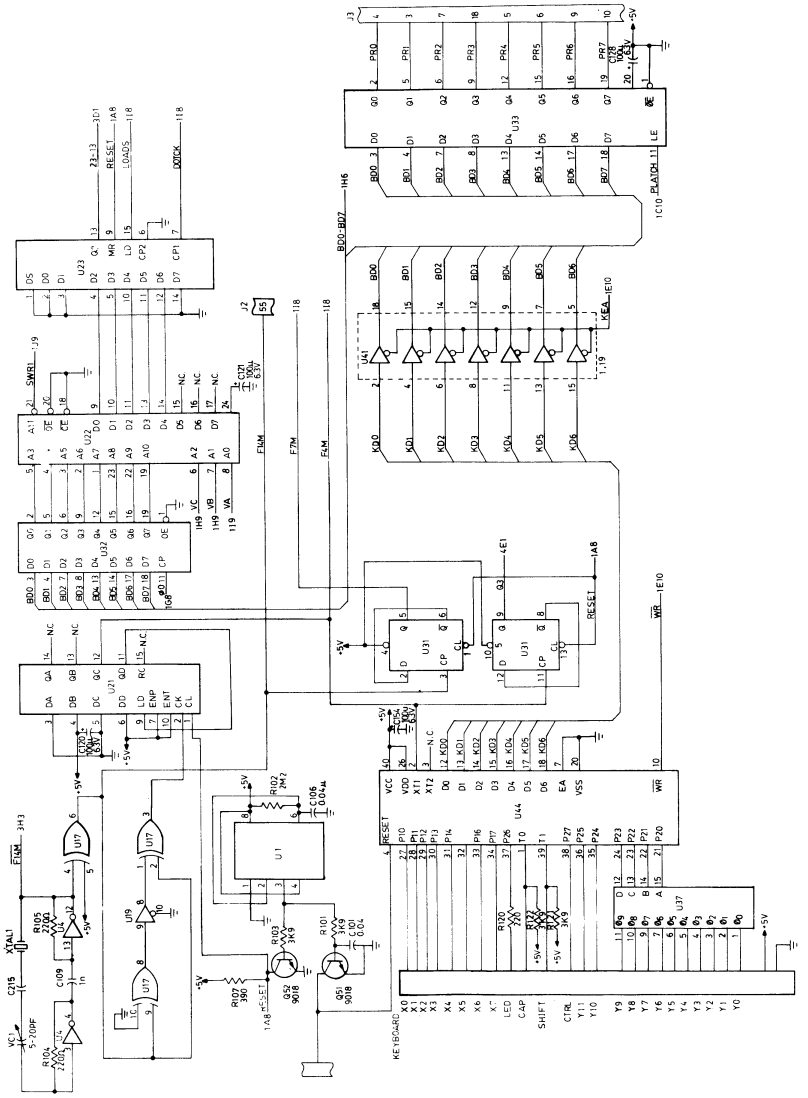
CIRCUIT DIAGRAM



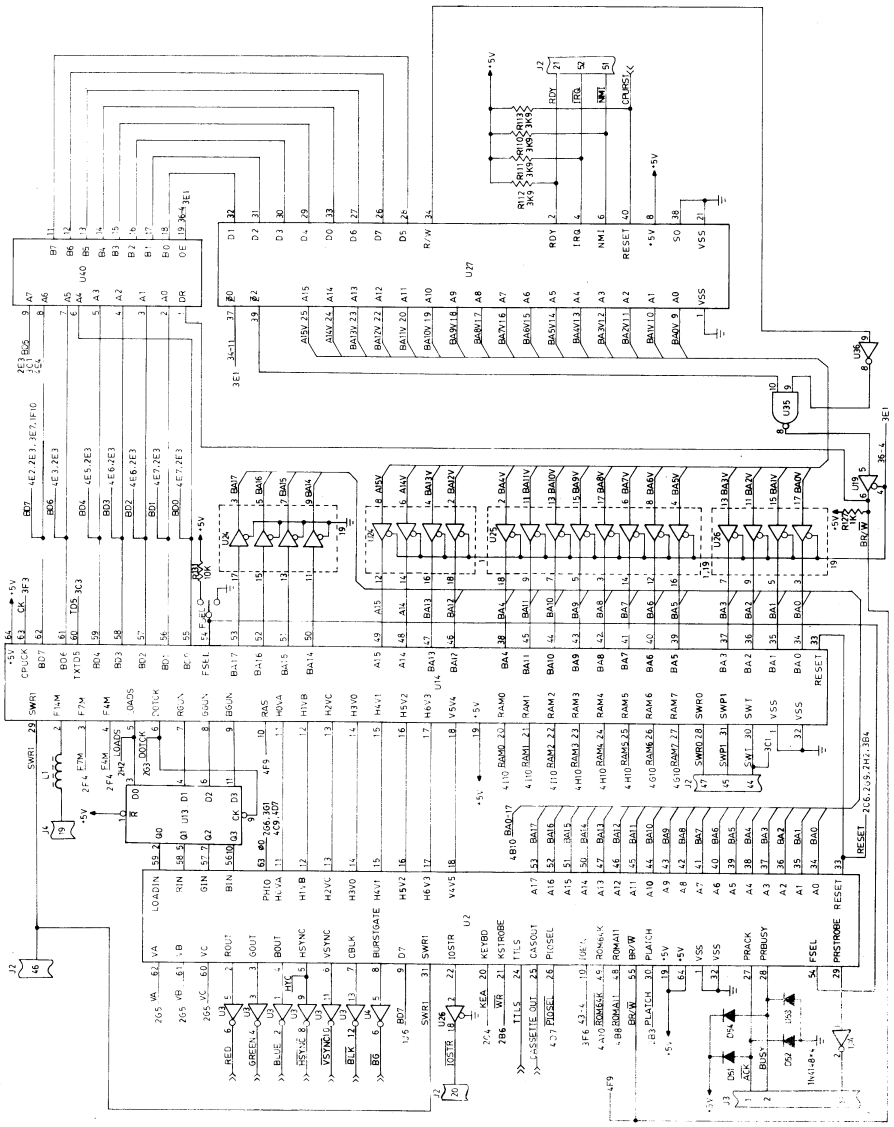
CIRCUIT DIAGRAM



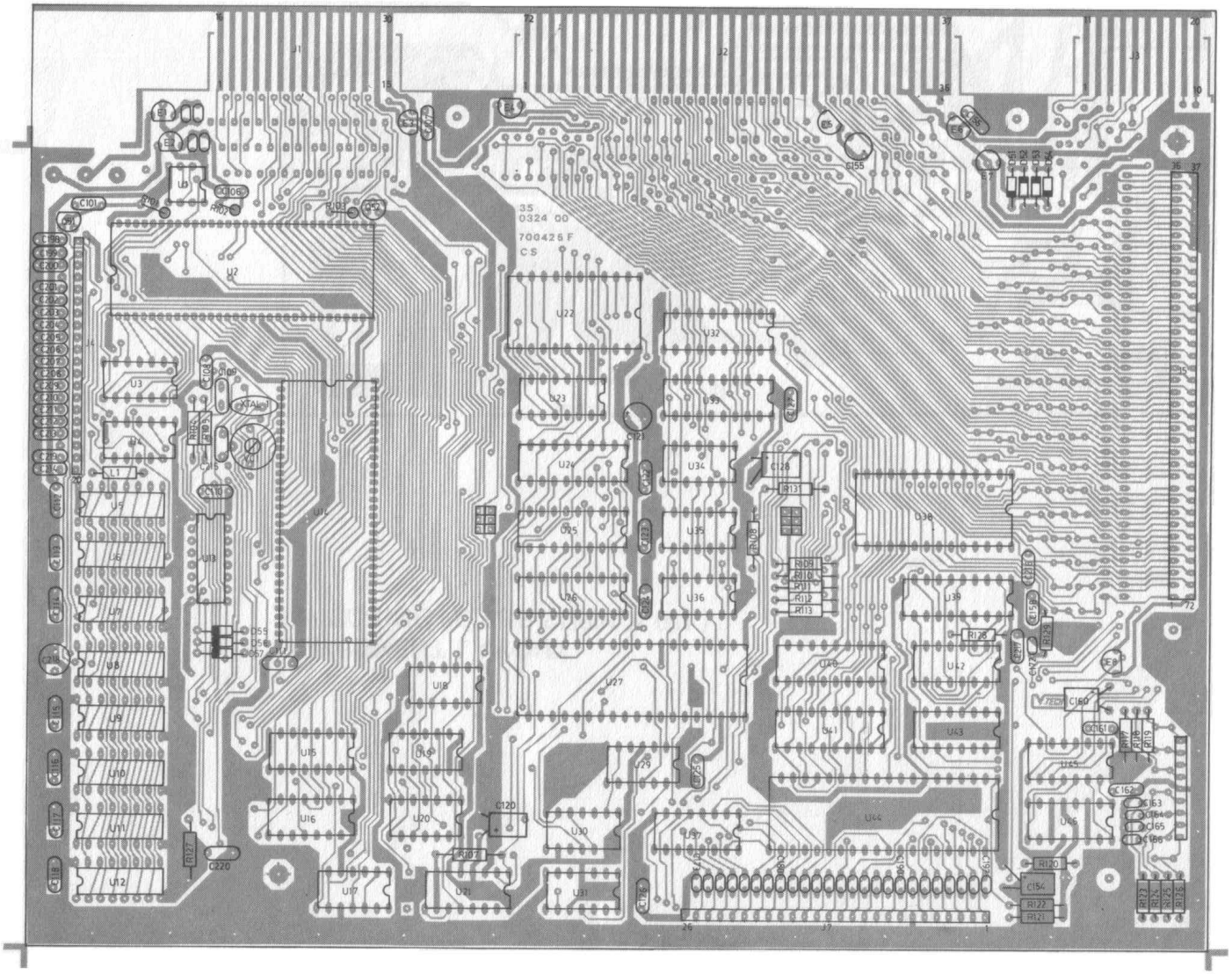
CIRCUIT DIAGRAM



CIRCUIT DIAGRAM



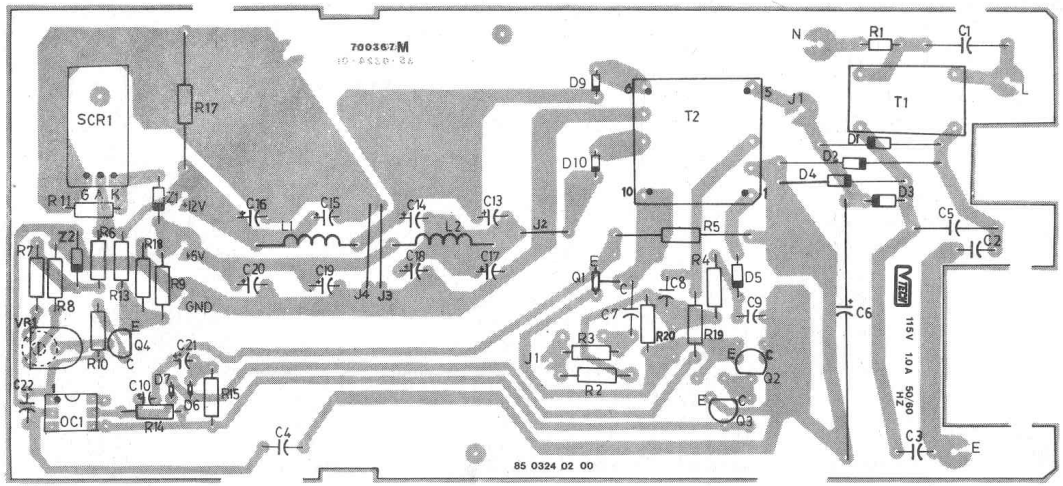
COMPONENT LAYOUT



COMPONENT SIDE

COMPONENT LAYOUT
SWITCHING POWER SUPPLY (UL)

700367

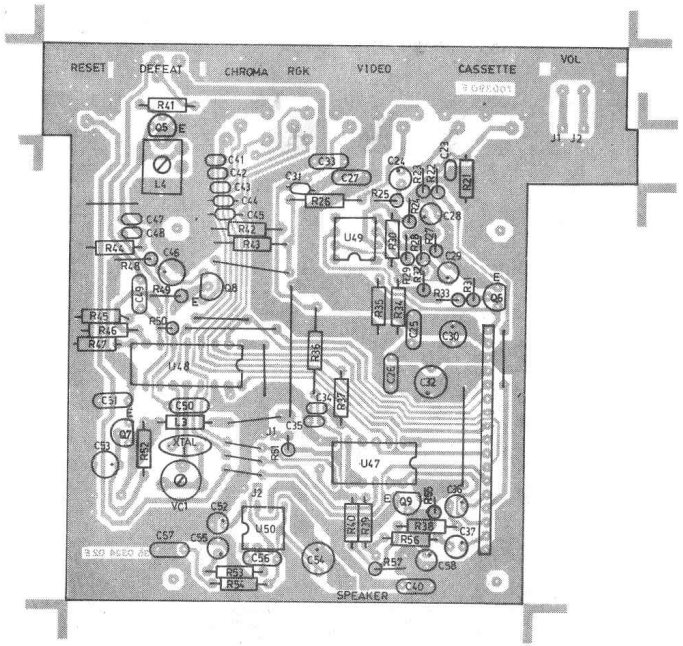


COMPONENT SIDE

COMPONENT LAYOUT
LINEAR BOARD (PAL)

SOLDER SIDE

REWORK

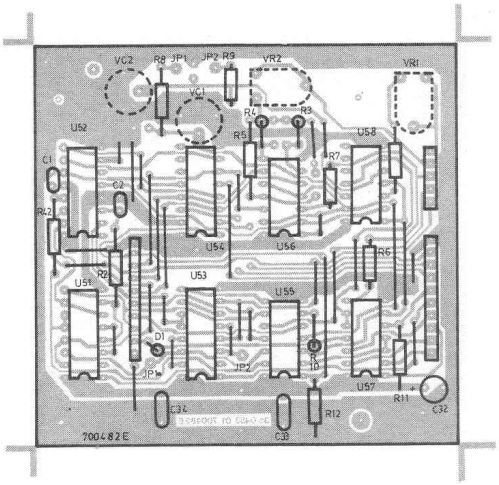


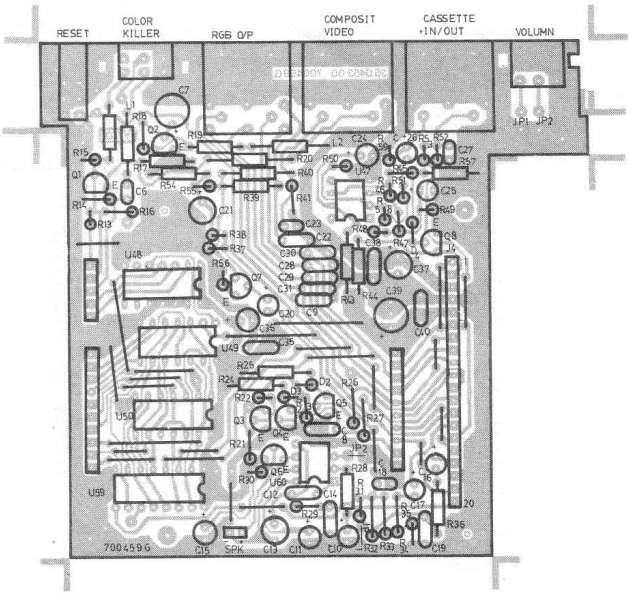
COMPONENT SIDE

APPENDIX G

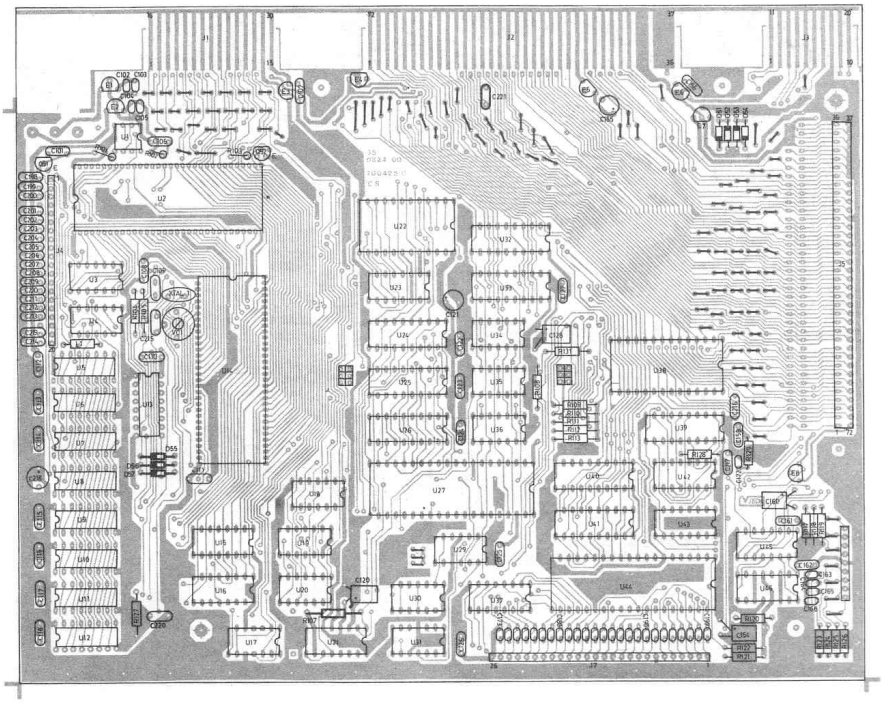
CIRCUIT DIAGRAMS (NTSC)

LINEAR BOARD COMPONENT LAYOUT



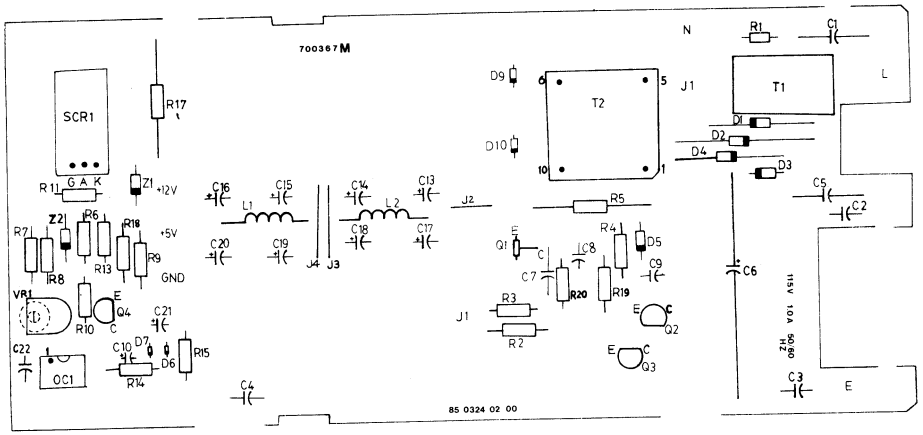


MAIN BOARD COMPONENT LAYOUT



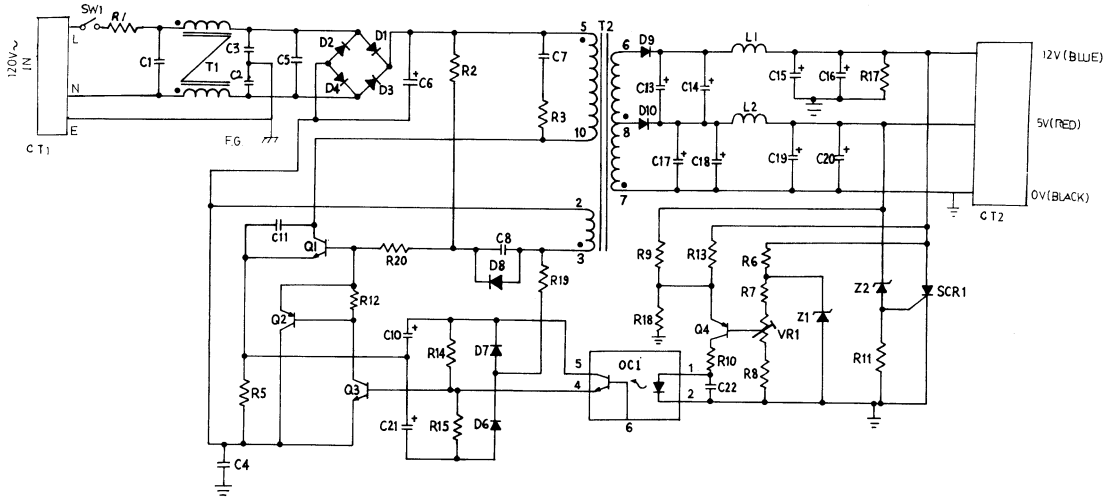
COMPONENT SIDE

COMPONENT LAYOUT
SWITCHING POWER SUPPLY (UL)



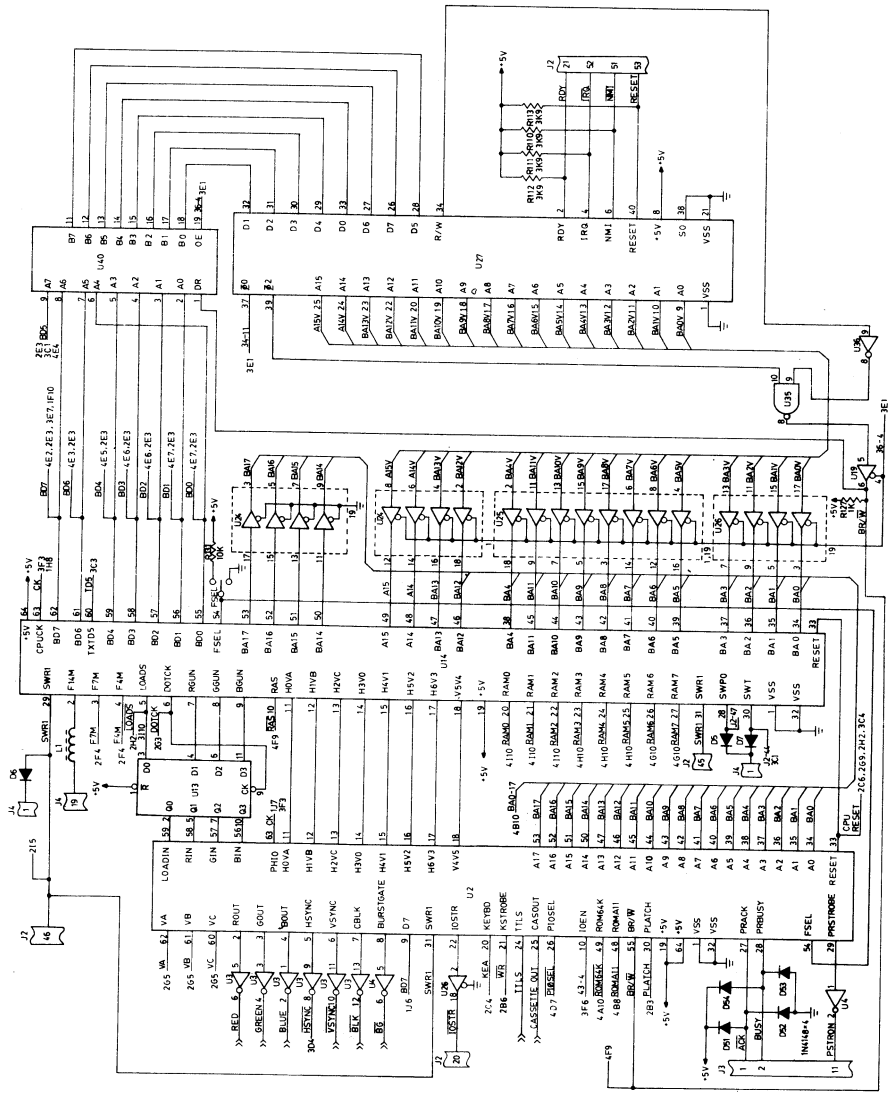
COMPONENT SIDE

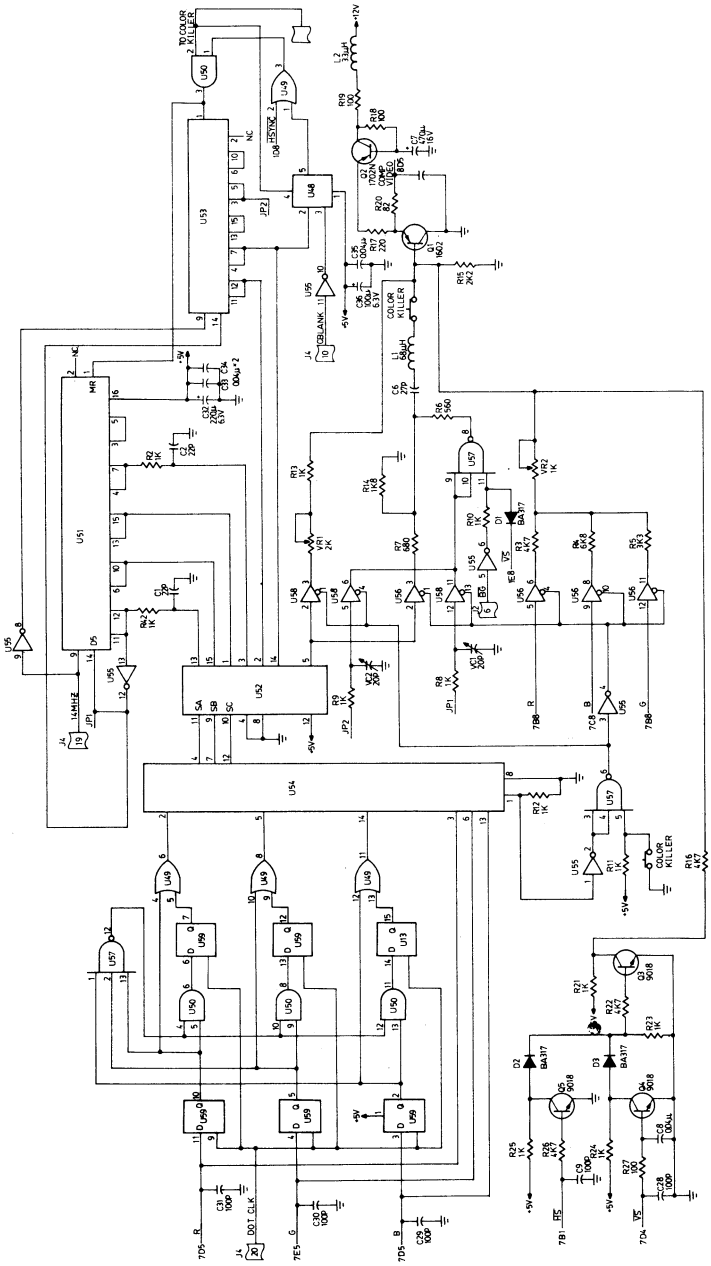
SWITCHING POWER SUPPLY FOR THE COMPUTER



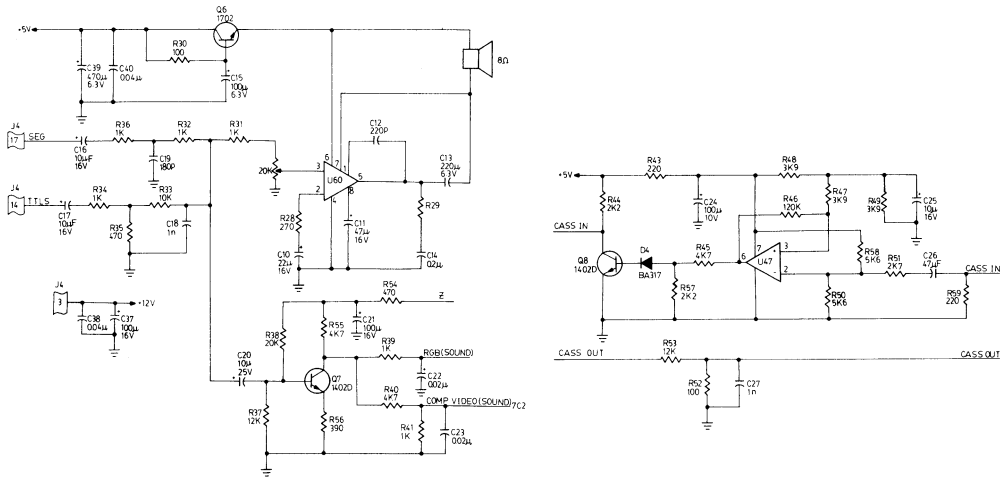
FRAME GROUND

MAIN BOARD CIRCUIT DIAGRAM

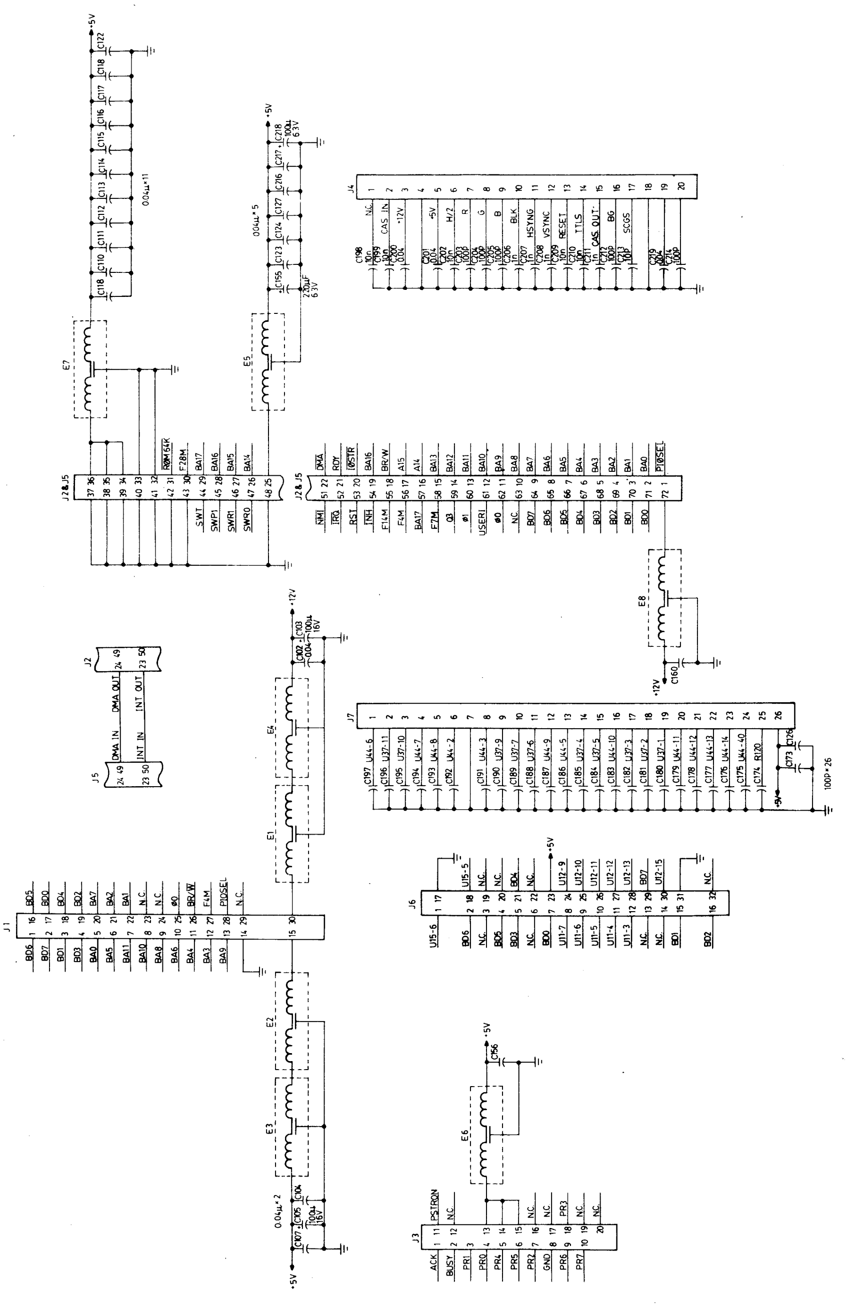




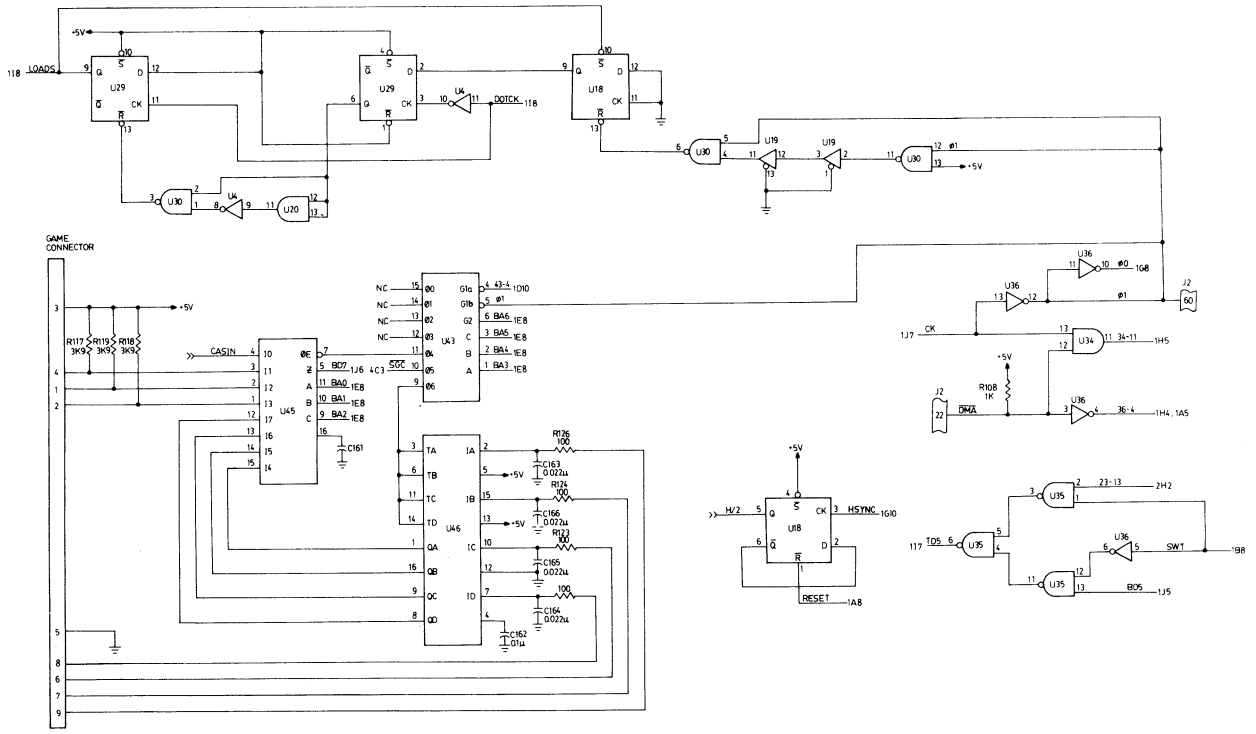
LINEAR BOARD CIRCUIT DIAGRAM



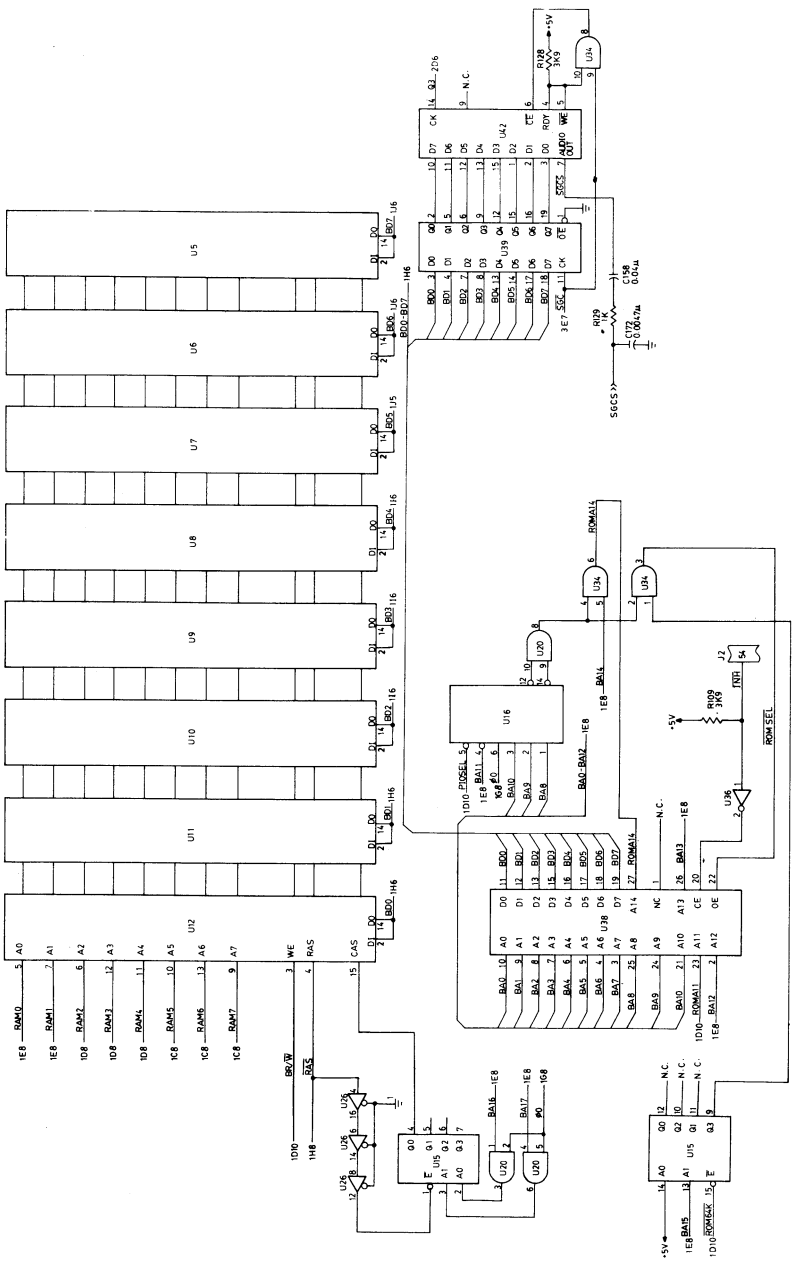
CONNECTOR



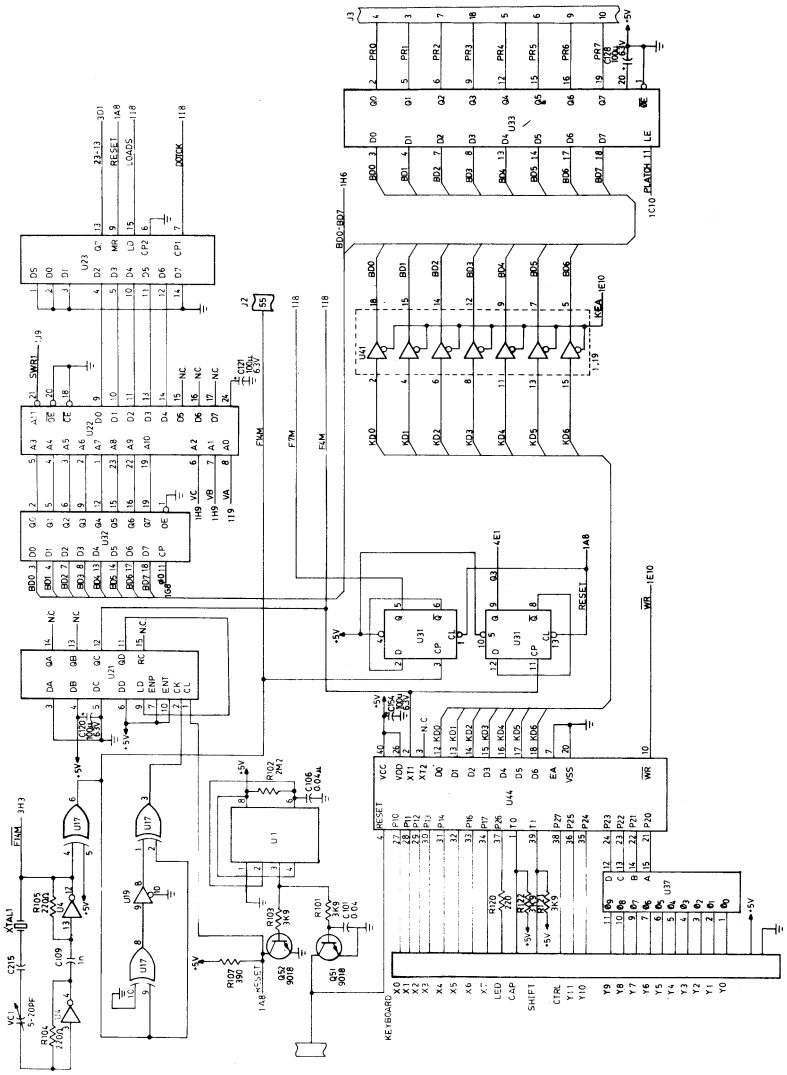
CIRCUIT DIAGRAM



CIRCUIT DIAGRAM



CIRCUIT DIAGRAM



CP/M is a registered trademark of Digital Research Inc.

MADE IN HONG KONG

91-0243-02