Osborne 1
Technical Manual

By Thom Hogan and Mike Iannamico

Part No. 2F00153-01

This manual describes the Osborne 1 computer, its software, and its specifications. The material presented is the proprietary information of Osborne Computer Corporation, and is intended for the personal use of the purchaser. Incorporation of the material described herein in any product or use in any commercial endeavor is strictly prohibited.

copyright © 1982 by Osborne Computer Corporation
all rights reserved

This document may not be copied by any means or using any medium without the express written permission of Osborne Computer Corporation. Violators will be prosecuted to the full extent of the law.

Osborne Computer Corporation
26538 Danti Court
Hayward, CA 94545
415-887-8080

Contents

Perspective

1.0 Perspective

1.1 INTRODUCTION

The Osborne 1 computer is a completely integrated computer unit. Designed to be portable, the entire computer weighs 24 pounds including a weather-resistant case and handle which facilitate moving it from one location to another.

The primary hardware components of the Osborne 1 are:

1. Dual 5 1/4-inch disk drives
2. Built-in 5" black and white monitor
3. Single-board computer
4. Z80A central processor
5. 64K bytes
6. 4K bytes ROM
7. Floppy disk interface
8. IEEE-488 interface
9. RS-232C interface
10. Modem (communications) interface option
11. 32 x 128 character memory-mapped video
12. Battery-pack option
13. 69-key detachable keyboard/numeric keypad
14. Lightweight, switching power supply

The design philosophy used in creating the Osborne 1 was twofold: 1) get all of the user controls and interface options up front where the user can see and manipulate them; and 2) make sure all of the components are integrated in such a way so that the case becomes a protective shell when the unit is closed up for traveling.

Upon receiving an Osborne 1, the user merely lays the unit on a flat surface, plugs it in, unlocks two latches to pull the keyboard unit from the rest of the computer, then begins using the computer. In normal operation, the main computer housing is propped onto the back lip of the keyboard, so the video display is tilted at a comfortable working angle.

1.2 POWER ON

When the Osborne 1 is powered ON, a sign-on message is displayed on the video monitor prompting the user to insert a diskette and press the RETURN key. This same message is displayed immediately following the depression of the RESET button located on the front panel of the Osborne 1.

The current revision of the monitor ROM is identified within the box under "OSBORNE 1" in the sign on message. There have been a number of hardware and software modifications during the ongoing refinement of the computer. Improvements have been made and features have been added in four distinct catagories; the main logic board, the disk interface board, the monitor ROM, and BIOS. Also, the latest Osborne 1 has a blue injection-molded case with a covered vent and fixed lid that covers the power well.

1.3 VARIATIONS IN DESIGN

Since there are some subtle differences between the various releases of the Osborne 1, this manual attempts to point out discrepancies between these versions whenever possible. For instance, as of this writing, there are currently six revisions of the main assembly logic board (A-F) which is further complicated by eight revisions of the board with components (A-H). There have also been numerous revisions of the disk interface board, four revisions of the monitor ROM (A, 1.2, 1.3, 1.4), and five revisions of BIOS (A, 1.2, 1.2.1, 1.3, 1.4). Version A software (ROM and BIOS) were upgraded to 1.2 free of charge by Osborne Computer Corp in November of 81.

The ROM monitor, which takes control of the machine at power on and reset (and displays the "Insert Disk" message described earlier), contains the diskette boot loader and extensions to the CP/M BIOS routines. There are no machine-level byte entry, examination, or modification routines in the ROM monitor. Such routines are unnecessary, given the utility software supplied with the Osborne 1.

Single density versions of the Osborne 1 prior to the 1.3 ROM and BIOS include a set of built-in diagnostics which are accessible by pressing ^D when the sign-on message is displayed.

Further discussion of the main hardware and software will be found in the following detailed descriptions of each module.

Logic Board and CPU

2.0 Logic Board And CPU

2.1 OSBORNE 1 MAIN LOGIC BOARD

The main logic board—as already described—contains all of the electronics necessary to provide a 64K, Z80A-based computer, including the additional interfaces required to run the disk drives, video display, and external communications ports.

2.2 BOARD REVISIONS

Six revisions of the bare main logic board have been made:

Revision A	— prototype, not released
Revision B	— first release, requires wiring changes
Revision C	— incorporates all wiring changes, requires additional changes to correct layout problems
Revision D	— requires no wiring changes
Revision E	— complete relayout of board
Revision F	— revision E board with clock modification added

Boards at levels B through F are in the field. Current production (15 June 1982) is at the Revision F level.

Boards loaded with parts are given a separate revision level:

Revision A	— prototype, not released
Revision B	— memory
Revision C	— first revision shipped, Revision B bare board
Revision D	— incorporates rework required by Revision C bare board
Revision E	— same as Revision D loaded board, but assembled without rework
Revision F	— first incorporation of Revision E bare board, not released
Revision G	— primary production board through 3 November 1981, incorporates Revision E bare board
Revision H-N	— released

NOTE

As of 9/82 a multilayer main logic PC Board which conforms to F.C.C. regulations is in production.

Beginning in January 1982, an international version of the Osborne 1 was in production. The difference in this international version is the character generator ROM and the primary power supply which is configured to accomodate the international voltages being used.

Also, future Osborne 1's which have been upgraded to double density, feature a 1.4 ROM. The double density option includes a small add-on component board mounted above the main logic board. This option will be described more thoroughly, later in this manual.

2.3 BOARD LAYOUT

The layout for the current revision and future multilayer main logic board are both shown in Figure 2.3:

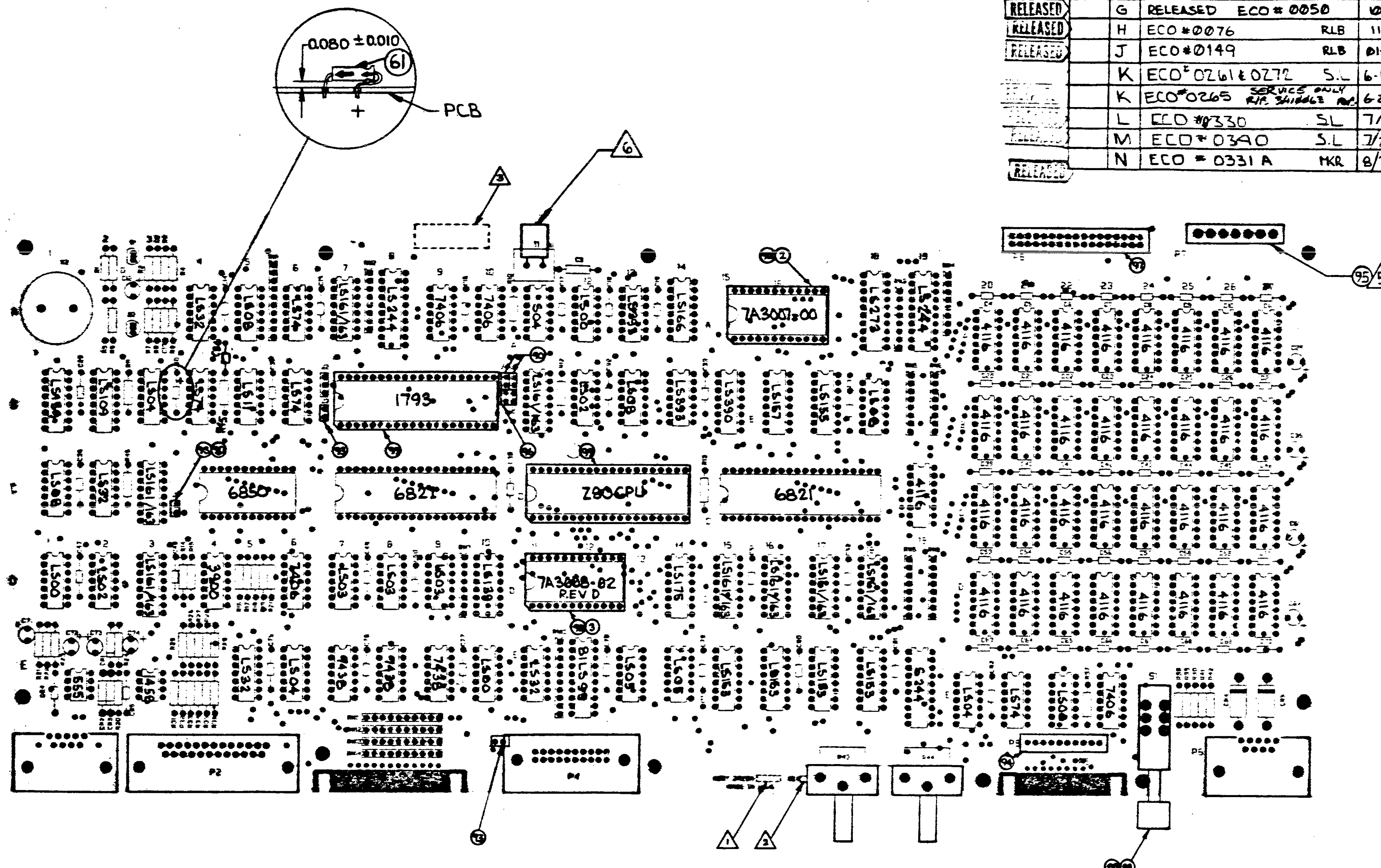
2.4 CENTRAL PROCESSOR

The central processor used is the NEC C780C, which is equivalent to the Zilog Z80A.

The CPU uses a clock of 4-megahertz, generated from a 15.9744-megahertz crystal run through a 74LS161 binary counter.

REVISIONS

ZONE	REV	DESCRIPTION	DATE	APPROVED
	G	RELEASED ECO # 0050	10-13-81	RBM
	H	ECO #0076 RLB	11-06-81	RBM
	J	ECO #0149 RLB	01-03-82	
	K	ECO # 0261 & 0272 S.L	6-15-82	
	K	ECO #0265 SERVICE ONLY R/F SHIELD	6-22-82	
	L	ECO #0330 SL	7/12/82	
	M	ECO # 0390 S.L	7/27/82	
	N	ECO # 0331 A MKR	8/24/82	



NOTES:

1. MARK "02" WITH INK IN AREA SHOWN
2. MARK "N" WITH INK IN AREA SHOWN
3. MARK SERIAL NO. WITH INK IN AREA SHOWN
4. ______________________________
5. CLIP EXCESS PIN LENGTH TO .05 MAX. LENGTH ON CIRCUIT SIDE.
6. CRYSTAL ITEM 50 MAY LAY FLAT OR BE UPRIGHT

Service Level

SEE DETACHED LIST OF MATERIAL 3A02011-02 (A SIZE)

	QTY REQD	FSCM NO.	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	MATERIAL SPECIFICATION	

PARTS LIST

NEXT ASSY	USED ON
2A12054	OCC1
2A01004	OCC1

APPLICATION

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE
FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±

MATERIAL

FINISH

DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS		DATE
DRAWN	RBM	10-9-81
CHECKED	RBM	10-13-81
ISSUED		

OSBORNE COMPUTER CORPORATION
26500 Corporate Avenue
Hayward, California 94545
415-887-8080

MAIN LOGIC P.C.B ASSY FAB

SIZE C | FSCM NO. —— | DWG. NO. 3A02011-C2 | REV N

SCALE FULL | SHEET 1 OF 1

Figure 2.3 — Main PC Board Assembly drawings

Osborne 1 Memory Scheme

3.0 Osborne 1 Memory Scheme

3.1 MEMORY BANKS

Four rows of eight 4116 dynamic RAM chips provide the Osborne 1 with 64K of main memory. The memory on the Osborne 1 main logic board is mapped into three logical banks. The first bank of memory is 64K by 8 bits of dynamic RAM (4116 chips). The second bank of memory consists of 4K of ROM (currently a 2732 chip; formerly two 2716 chips), 16K of RAM, and memory-mapped I/O . A "mimicking" of the first bank's dynamic RAM provides the top 48K. The third bank of memory is 16K by 1 bit worth of dynamic RAM memory used for storing the dim character video attribute. Figure 3.1 below shows the Osborne 1 memory map:

The addresses shown on this memory map pertain to the 1.4 ROM and BIOS. Software released prior to 1.4 has different addresses for BIOS, BDOS, and CCP. Consult the Software section for more information.

3.2 CP/M MEMORY ALLOCATION

0000 - 0002	Jump to BIOS warm start entry
0003	IOBYTE
0004	Drive number/current user
0005 - 0007	Jump to BDOS entry
0008 - 0037	Reserved for interrupts
0038 - 003A	RST7 (used by DDT)
003B - 003F	Reserved for interrupts
0040 - 004F	Scratch area used by BIOS
0050 - 005B	Not used
005C - 007C	File control block
007D - 007F	Random record position
0080 - 00FF	Default DMA buffer area
0100 - CEFF	Transient program area
CB00 - E0FF	CCP/BDOS (CP/M)
E100 - EFFF	BIOS and Osborne buffer area
F000 - FFFF	Memory mapped video display

Note: 1.2 and 1.3 software have CCP/BDOS located from CF00 to E4FF and BIOS from E500 to EFFF.

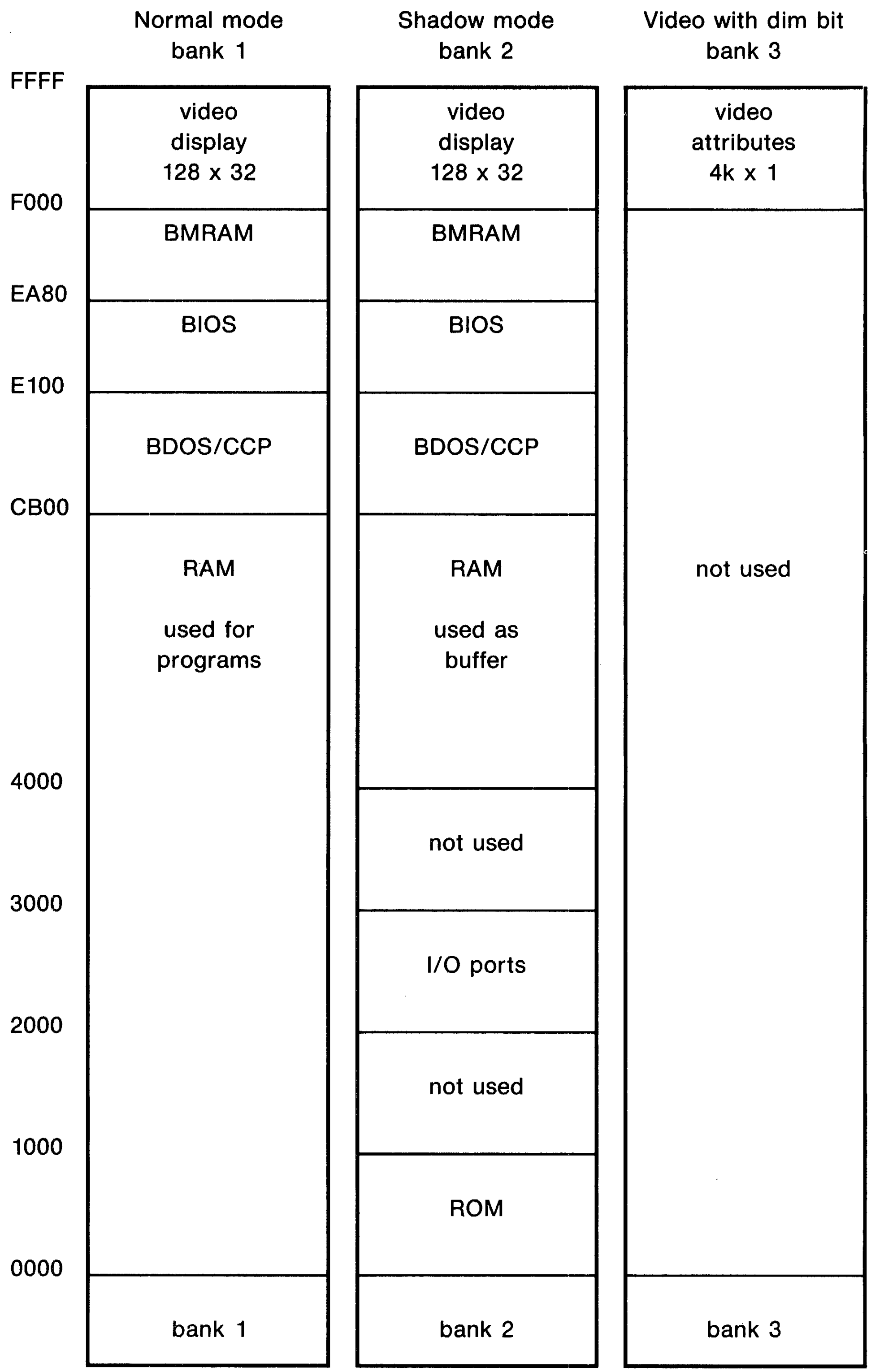


Figure 3.1 Osborne 1 Memory Map

3.3 MAIN MEMORY

Bank 1 of memory is used mainly for programming. CP/M, for instance, loads itself into the uppermost free area—just below the video display memory—with the Basic Input Output System (BIOS), the Basic Disk Operating System (BDOS) and the Console Command Processor (CCP). As with all CP/M systems, the memory area from 0000 hex to 0100—commonly referred to as "page 1" of memory—is reserved for use by CP/M. Overall, about 51K bytes of usable memory are available to the programmer.

As you can see from Figure 3.1, the primary bank of memory is divided into 60K of user memory and 4K of video display memory. The video display memory layout is discussed in an upcoming section.

3.4 MEMORY-MAPPED I/O

The second bank of memory consists of the monitor ROM and memory-mapped I/O. Specifically, the serial/modem port, IEEE-488 port, disk interface, and keyboard are all addressed through the memory-mapped I/O section. The memory-mapped I/O is detailed in Figure 3.4 and 3.4.1.

Use of each of the I/O memory locations will be discussed in conjunction with the specific peripheral being addressed.

3.5 MEMORY ACCESS TIME

The memory access time is 250 nanoseconds for programmable memory and 350 nanoseconds for the read-only memory. Programs in read-only memory execute without delay, while programmable memory has delay times added as follows:

First M1 cycle — 188 nanoseconds
Subsequent consecutive M1 cycles — 0 nanoseconds
Non-M1 cycles — 375 nanoseconds

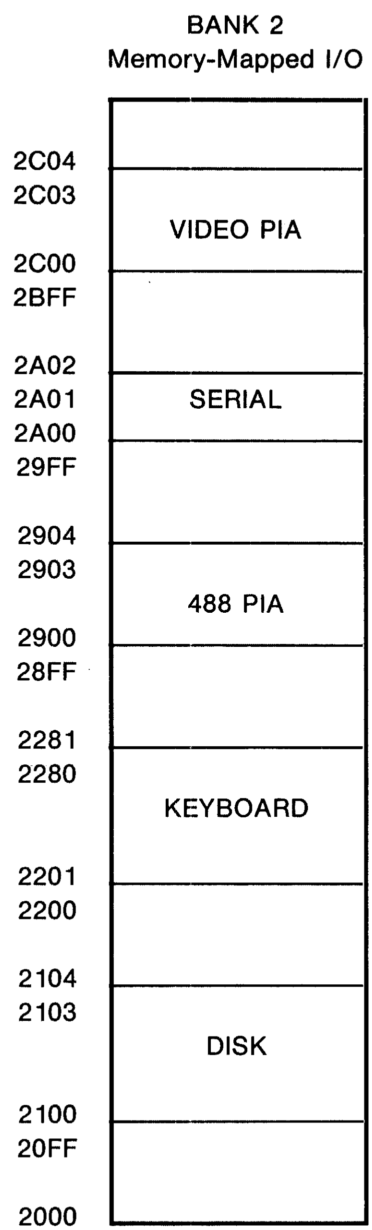


Figure 3.4 I/O memory map

OSBORNE 1 I/O Port Assignment (Shadow Mode)

	Address	Read	Write
Disk	2100	Status Register	Control Register
	2101	Track Register	Track Register
	2102	Sector Register	Sector Register
	2103	Data Register	Data Register
Keyboard	2201	Row 0	
	2202	Row 1	
	2204	Row 2	
	2208	Row 3	
	2210	Row 4	
	2220	Row 5	
	2240	Row 6	
	2280	Row 7	
488 PIA	2900	Port A Direction/Data	Port A Direction/Data
	2901		Port A Control Register
	2902	Port B Direction/Data	Port B Direction/Data
	2903		Port B Control Register
Serial	2A00	Status Register	Control Register
	2A01	Receive Buffer	Transmit Buffer
Videc PIA	2C00	Port A Direction/Data	Port A Direction/Data
	2C01		Port A Control Register
	2C02	Port B Direction/Data	Port B Direction/Data
	2C03		Port B Control

Figure 3.4.1 I/O Port Assignments In Bank 2

Osborne 1 Interface Design

4.0 Osborne 1 Interface Design

4.1 IEEE-488 INTERFACE

The IEEE-488 interface is created using a 6821 PIA. The IEEE-488 implementation, as described in the Osborne 1 User Guide's Appendix, is a subset of the complete IEEE specification. Specifically, no provision has been made for controlling multiple devices on the interface.

4.1.1 IEEE-488 Signal Direction

The Osborne 1 IEEE-488 signal directions are provided here:

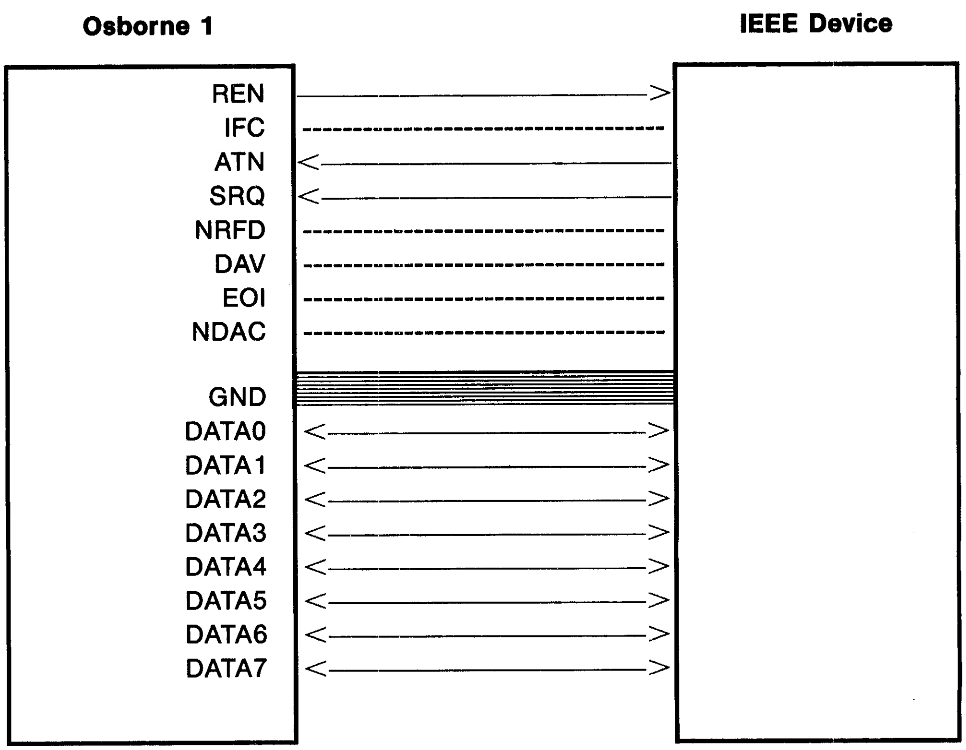


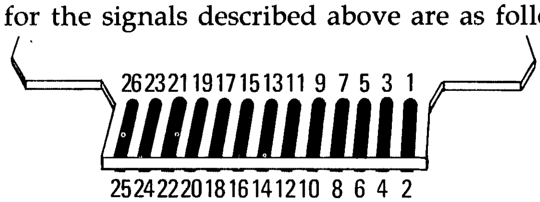
Figure 4.1.1 IEEE-488 Signal Direction

4.1.2 IEEE-488 Pinouts

Any IEEE 488-compatible device can connect to the Osborne 1 through the IEEE connector. Because this port is used for more than just IEEE 488 signals, we've declined to use a standard IEEE connector. The following table shows the pin assignments for both the IEEE standard connector and the Osborne 1 edge connector:

IEEE	OSBORNE	SIGNAL NAME	
1	1	Data bit 1	(DIO1)
2	3	Data bit 2	(DIO2)
3	5	Data bit 3	(DIO3)
4	7	Data bit 4	(DIO4)
5	9	End or Identify	(EOI)
6	11	Data valid	(DAV)
7	13	Not ready for data	(NRFD)
8	15	No data accepted	(NDAC)
9	17	Interface clear	(IFC)
10	19	Service request	(SRQ)
11	21	Attention	(ATN)
12	23	Cable shield + GND	(SHIELD)
13	2	Data bit 5	(DIO5)
14	4	Data bit 6	(DIO6)
15	6	Data bit 7	(DIO7)
16	8	Data bit 8	(DIO8)
17	10	Remote enable	(REN)
18	12	Signal ground	(DAV)
19	14	Signal ground	(NRFD)
20	16	Signal ground	(NDAC)
21	18	Signal ground	(IFC)
22	20	Signal ground	(SRQ)
23	22	Signal ground	(ATN)
24	24	Signal ground	(Logic)

The pinouts for the signals described above are as follows:



26-pin edge connector, looking at front of Osborne 1. Pins 25 and 26 are not used, but provided for compatability.

Figure 4.1.2 IEEE-488 Pinouts

4.1.3 IEEE-488 Jump Vectors

To provide easier access to the routines necessary to use the IEEE-488 interface, the CP/M BIOS jump table has been extended to provide a series of extra jumps specifically for the IEEE-488 programmer. The IEEE-488 routines are offset from the starting address of BIOS as follows:

BIOS + 3FH Control out
BIOS + 42H Status in
BIOS + 45H Go to standby
BIOS + 48H Take control
BIOS + 4BH Output interface message
BIOS + 4EH Output device message
BIOS + 51H Input device message
BIOS + 54H Input parallel poll message

4.1.4 IEEE-488 Communication Protocol

IEEE-488 commands use no RAM other than the stack. Each command routine in BIOS determines status of the port by reading the status of the 6821 PIA chip. The PIA transmits signals in both directions, so to reduce the overhead in determining the current direction the PIA is attempting to communicate, it is always left in one of two modes:

the source handshake mode
or
the acceptor handshake mode

(The PIA specification sheet will be helpful in determining these modes.)

Several of the IEEE commands require that the PIA be in the source handshake mode when called. The PIA is normally in the source handshake mode following the completion of any IEEE- bus information transfer, so this is not a major restriction. For instance, both the Status In and the Parallel Poll commands require that the PIA be in the source mode, which means that you can perform the detection-of-device request using either serial poll or parallel poll only when the interface is idle.

To send data to a device on the IEEE bus, the controller makes the device a LISTENER, assumes the role of TALKER, and sends the data. To receive data from an external device, the controller must first make the device a TALKER and then assume the role of LISTENER. After this, the controller goes on "standby" and allows the two devices to communicate at their own rate.

The controller can regain control asynchronously by setting the ATN signal to true. But if a device-dependent message is true at the same time when ATN becomes true, other devices on the IEEE bus can misinterpret the interrupted byte as an interface message and produce chaos. Avoid the problem by taking control synchronously. If high-speed transfer of data between devices is not required and the computer can be tied up during the transfer, it is better to make the controller listen to the transfer while discarding the data. This procedure allows the controller to count transfers, look for EOI signals, or "time out" the TALKER before regaining control.

The IEEE commands are detailed in the User's Guide's Appendix, with sample programs included to help decipher how we've put the BIOS jumps into effect for the IEEE bus. A listing of the 6821 registers and instruction set is provided as Appendix B of this manual for those who wish to make direct use of the PIA for controlling the IEEE-488 port.

4.1.5 IEEE-488 As A Parallel Port

The IEEE-488 can also be used as a standard parallel port, and software has been added to the BIOS section of CP/M so that users of a Centronics-compatible printer may use their printer as the list device under CP/M. By setting the CP/M IOBYTE equal to BAT:, UR1:, UP1:, or LPT:, the IEEE-488 port is reconfigured by BIOS to be a simple 8-bit parallel input/output port with the following pinouts:

Osborne IEEE Edge Connector		**Centronics-Compatible Connector**
pin 1	data 0	2
2	data 4	6
3	data 1	3
4	data 5	7
5	data 2	4
6	data 6	8
7	data 3	5
8	data 7	9
11	out strobe	1
12	ground	19
15	busy	11
16	ground	29
19	select	13

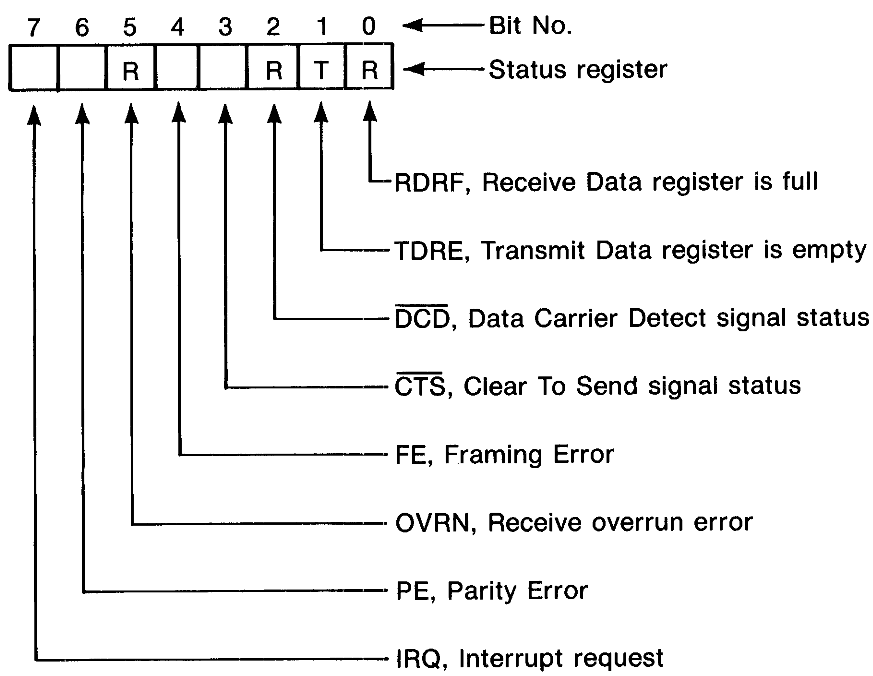
Figure 4.1.5 IEEE-488 Parallel Interface

To use a parallel printer connected to the IEEE interface, you must employ the SETUP program to configure the appropriate protocol for the printer. You need do this only once for each diskette. Alternatively, if you have two printers hooked up, or wish to change the printer being used from within a program, you can reset the IOBYTE as described later.

4.2 SERIAL RS232 INTERFACE

The serial port is configured as a RS-232C-compatible port, though certain of the RS-232C signals are held at +5 volts since they are not needed to control the Osborne 1. A 6850 ACIA chip controls the serial port.

The RS232 status port address is located at 2A00H and the data port at 2A01H in the shadow mode. RS232 status bit assignments are detailed in the following diagram:



NOTE: See pages 9-59 and 9-60 in Volume 2 of An Introduction To Micro computers by Adam Osborne/McGraw-Hill or the 6850 Data sheet for a complete description.

Figure 4.2 RS-232 Status Bit Assignments

4.2.1 RS232 Signal Direction

The serial port is configured as a DTE device. The following signals apply:

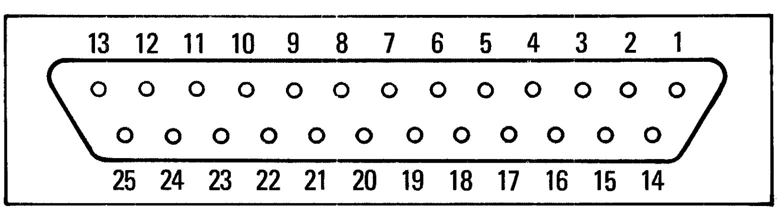
```
        Osborne 1                                          RS-232 device

            GND  ------------------------------------  ND
         TXDATA  <-----------------------------------  TXDATA
         RXDATA  ----------------------------------->  RXDATA
            RTS  ----------------------------------->  RTS
            CTS  +5 volts< --------------------------  CTS
            DSR  +5 volts -------------------------->  DSR
       CARR DET  +5 volts< --------------------------  CARR DET
            DTR  <-----------------------------------  DTR
```

Figure 4.2.1 RS-232 Signal Direction

4.2.2 RS232 Pinouts

Below are the pin assignments for the RS-232 serial interface:

DB-25S	RS-232	Pin Definition
1	AA	Frame ground (optional)
2	BA	Transmitted data (low=1)
3	BB	Received data (low=1)
4	CA	Request to send (high or no connection enables)
5	CB	Clear to send (always high on OCC 1)
6	CC	Data set ready (always high on OCC 1)
7	AB	Signal ground
8	CF	Received line signal detected (always high)
20	CD	Data terminal ready (high or no connection enables)



9,10,11,12,13,14,15,16,17,18,19,21,22,23,24,25 no connections

Figure 4.2.2 RS-232 Serial Pinouts

4.3 MODEM

A close look at the circuitry in the Osborne 1 schematics will show that the modem and RS-232 interfaces are basically one and the same. In addition to the serial port, TTL-level signals may be directly input into the 6850 ACIA using the modem port connection. To read to and from the modem or serial port use the CP/M IOBYTE function.

4.3.1 Modem Signal Direction

The following signals apply to the Modem Port:

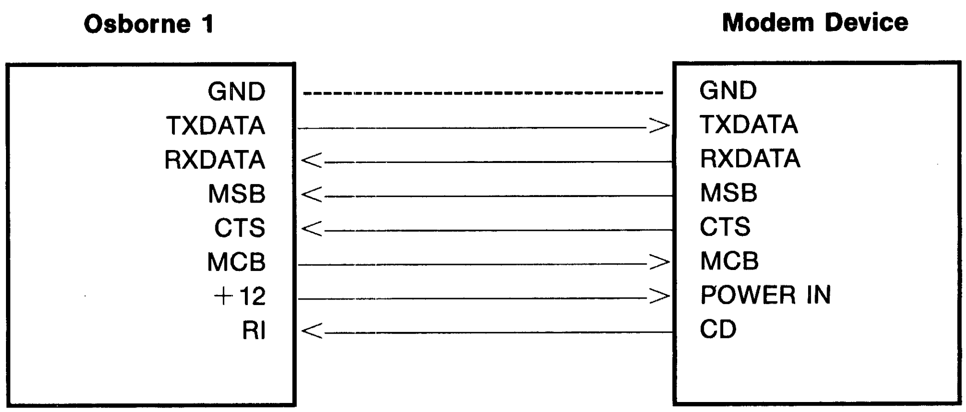


Figure 4.3.1 Modem Signal Direction

4.3.2 Modem Pin Connections

The pin connections on the modem port are as follows (all use the standard numbering of the DE-9P connector).

DE-9P Osborne Modem Definition:

1 **GND** – Signal ground
2 **TXD** – Transmitted data — TTL logic, 1=high
3 – Not used
4 **MSB** – Modem status bit— open collector, 50ua sink=inactive
5 **CTS** – Clear to send
6 **RXD** – Receive data — bipolar input, -0.5v-10v=1
7 **+12v** – Connected to power supply through 22 ohms
8 **MCB** – Modem control bit — TTL, low suppresses output
9 **RI** – Ring indicator — TTL, high-to-low sets flag

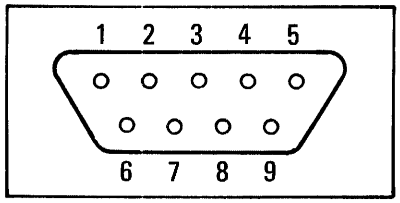


Figure 4.3.2 Modem Pinouts

NOTE: all signals are TTL-level and thus this interface is extremely vulnerable to damage through misuse.

Early versions of the main PC board have a reversal of the modem signals described in this document; the original design called for a female modem socket. Consult the User's Guide which accompanied your Osborne 1 for the applicable modem pinouts. A special cable is required with the Osborne Modem to compensate for the configuration of these earlier connectors.

If you have a modem that uses an RS-232C connector, you may have to use an external adapter box to properly institute all the modem functions. Connecting a modem without using an external adaptor may damage your Osborne 1, as pins 4 and 5 are open collectors and are sensitive to signal-edge transitions. If pin 4 is not connected to the modem, make sure that nothing is connected to pin 4 at the Osborne end; otherwise, adjacent signals may be received inadvertently.

4.3.3 Modem Status

You can determine the output status of the modem port by using the BIOS call LISTST located at 0E12D hex. A value of 0FF hex indicates that the list device is ready; 00 indicates busy.

To find the input status of the modem port, you must first switch to bank 2 of memory and then look at memory location 2A00 hex. To change the status of the modem or serial device directly, you use the same memory location and write a special "control" byte as dictated in the 6850 specification sheet. Memory location 2A01 hex in bank 2 is the data buffer: you read information from external devices by moving the byte to one of the CPU internal registers, you send information to the external device by moving data from the CPU register to the memory location.

4.4 BAUD RATE

Baud rates for the serial and modem ports is software-selectable between 300 or 1200 (use the SETUP program to change the baud rate from 1200 that BIOS assumes to 300). If necessary, the baud rate may be increased (on Revision level E boards and latter) from the 300/1200 baud normally used on the Osborne 1 to 600/2400 baud. To Switch baud rates remove the two-pin jumper from the position "J1" (See Fig. 2.3) on the logic board. Earlier versions require soldering and cutting of traces.

An even faster Baud rate of 1200/2400 or 2400/9600 can be attained with the addition of a few routing wires. The Osborne 1 cannot handle terminal functions above 2400 baud due to the limitation of system calls. However, communication to an external terminal, printer or another computer can be maintained at these higher rates of transmittion.

Note: We are making the procedure for increasing Baud rate available because we are convinced that certain parties are using it successfuly. Osborne Computer Corporation does not officially support these higher Baud rates because the 6850 support driver is potentially unreliable at high speeds. Also, be forwarned that any tampering within the computer will void your waranty.

Here is the procedure for increasing the Baud rate to either 1200/2400 or 2400/9600. Refer to the Figure 2.3 of the main logic board and the illustration below the instructions for more details:

1. Remove LS161/163 from C3 and repace it with a 16 pin socket.
2. Bend up pins 2, 3, 7, and 10.
3. Route a wire between pins 3 and 4, another between pins 4 and 7, and another wire between pins 7 and 10.
4. Place the modified pack in the socket. The modification so far will provide 1200/4800 Baud capability.
5. For 2400/9600 Baud you must connect an extra wire from pin 2 to the middle contact of J2 as illustrated in Figure 4.4.

For those who wish to control the 6850 ACIA directly, Appendix C contains a listing of the registers and instructions the 6850 chip utilizes.

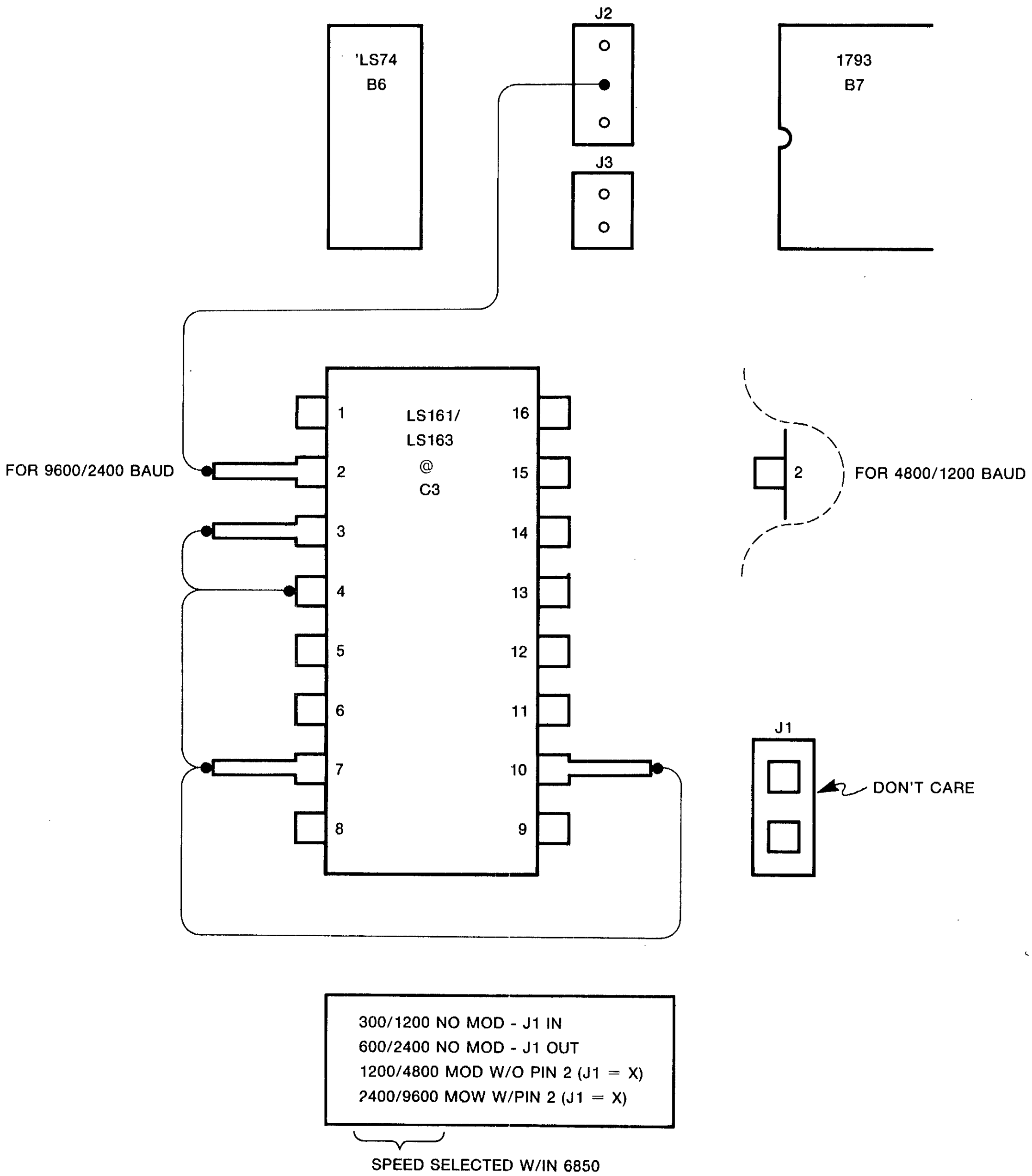


Figure 4.4 Baud Rate Hardware Modification

Video Attributes

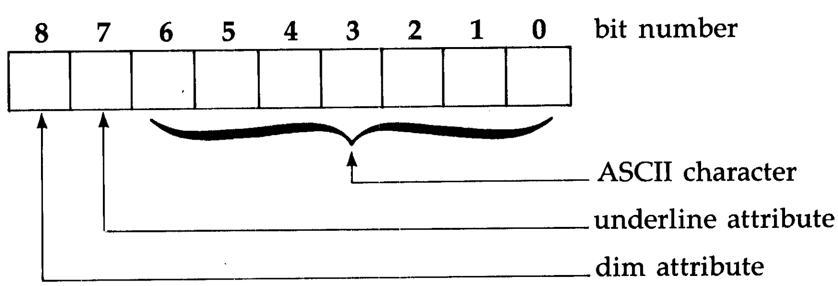
5.0 Video Attributes

5.1 VIDEO DISPLAY

The video display probably has the trickiest design of any of the modules on the main logic board. A 6821 PIA chip is used to control portions of the video display circuitry, while the video display circuitry itself is used to refresh the main memory dynamic RAM. A 2716 ROM is used as a character generator, and an extra 4116 chip is used to store the dim video attribute.

5.2 VIDEO MEMORY

Memory from 0F000 hex to 0FFFF hex is used for the video memory map. That memory is considered to be 32 rows of 128 characters, even though only 24 rows of 52 characters are actually displayed on the monitor screen at a time. The character matrix is 7 × 9 in an 8 × 10 box. The following bits are used to encode the character to be displayed:



5.3 CHARACTER GENERATION

All characters are defined using a dot matrix that is 8 columns wide by 10 rows high. Within the character generator ROM, 128 characters are defined. The top row of each character is defined in the first 128 bytes of the character generator ROM. The second row is subsequently defined in the second 128 bytes, and so on. This is illustrated as follows:

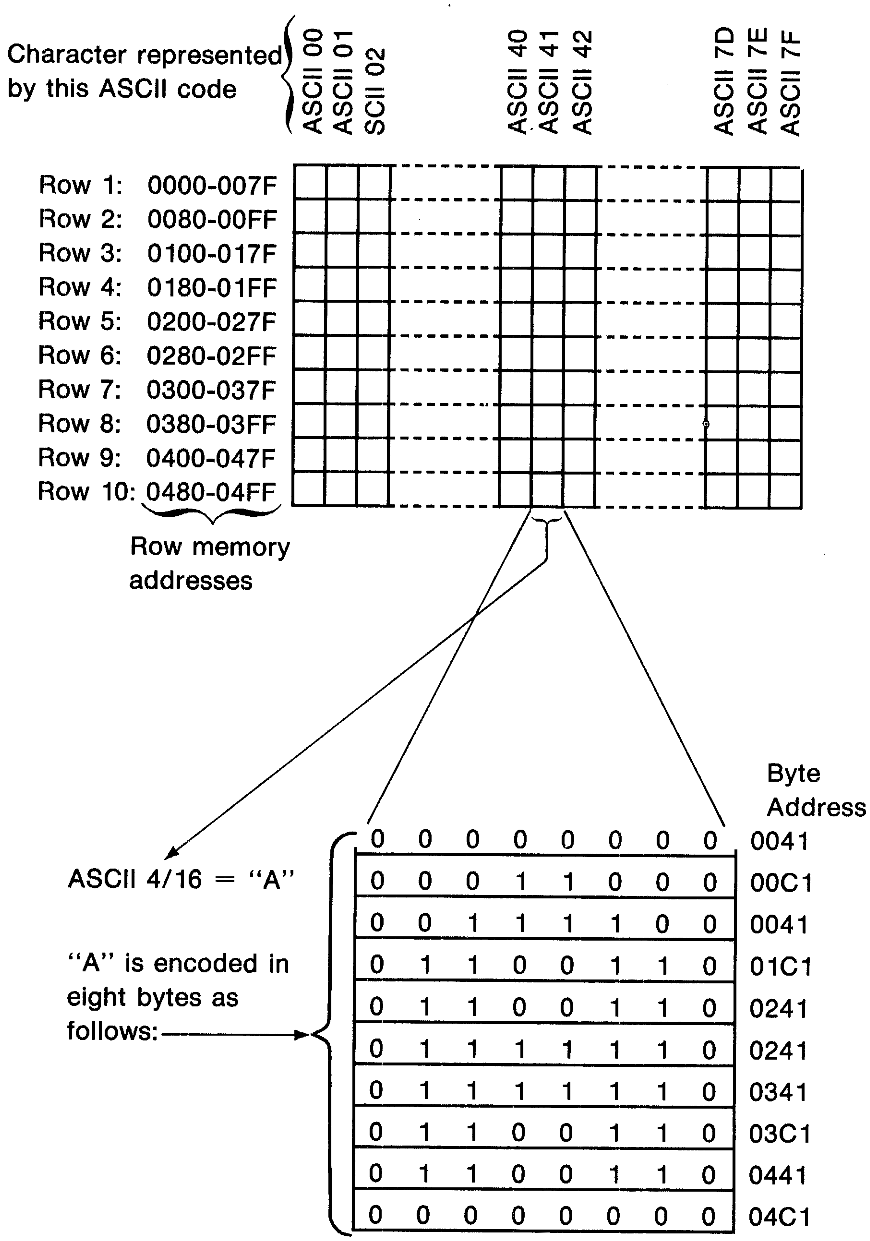


Figure 5.3 Character Generator Encoding

5.4 DISPLAY LOGIC

In order to create displays, ASCII codes are written into screen RAM. Display logic fetches the code in each screen RAM location and displays the character from the character generator ROM corresponding to the screen RAM ASCII code. This may be illustrated as follows:

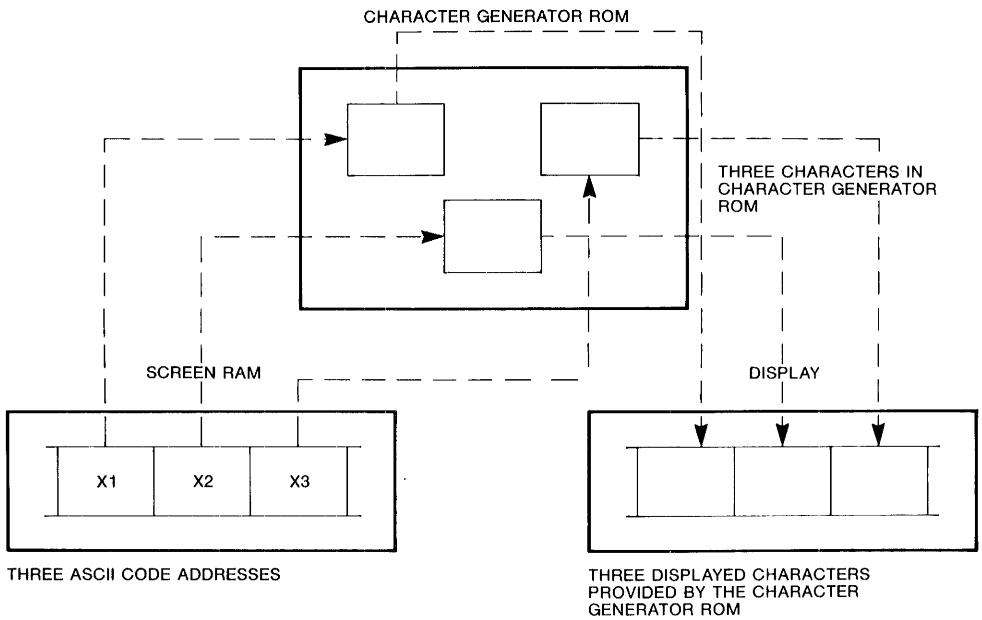


Figure 5.4 Display Logic Diagram

5.5 BLOCK GRAPHICS

The first 32 character values (00 hex through 1F hex) are not displayable ASCII characters and are thus used for block graphics characters on the Osborne 1. The following graphic characters are available:

Hexa Decimal		Binary								ASCII
0	0	0	0	0	0	0	0	0	0	@
0	1	0	0	0	0	0	0	0	1	^A
0	2	0	0	0	0	0	0	1	0	^B
0	3	0	0	0	0	0	0	1	1	^C
0	4	0	0	0	0	0	1	0	0	^D
0	5	0	0	0	0	0	1	0	1	^E
0	6	0	0	0	0	0	1	1	0	^F
0	7	0	0	0	0	0	1	1	1	^G
0	8	0	0	0	0	1	0	0	0	^H
0	9	0	0	0	0	1	0	0	1	^I
0	A	0	0	0	0	1	0	1	0	^J
0	B	0	0	0	0	1	0	1	1	^K
0	C	0	0	0	0	1	1	0	0	^L
0	D	0	0	0	0	1	1	0	1	^M
0	E	0	0	0	0	1	1	1	0	^N
0	F	0	0	0	0	1	1	1	1	^O
1	0	0	0	1	0	0	0	0	0	^P
1	1	0	0	1	0	0	0	0	1	^Q
1	2	0	0	1	0	0	0	1	0	^R
1	3	0	0	1	0	0	0	1	1	^S
1	4	0	0	1	0	0	1	0	0	^T
1	5	0	0	1	0	0	1	0	1	^U
1	6	0	0	1	0	0	1	1	0	^V
1	7	0	0	1	0	0	1	1	1	^W
1	8	0	0	1	0	1	0	0	0	^X
1	9	0	0	1	0	1	0	0	1	^Y
1	A	0	0	1	0	1	0	1	0	^Z
1	B	0	0	1	0	1	0	1	1	ESC
1	C	0	0	1	0	1	1	0	0	—
1	D	0	0	1	0	1	1	0	1	]
1	E	0	0	1	0	1	1	1	0	.
1	F	0	0	1	0	1	1	1	1	;

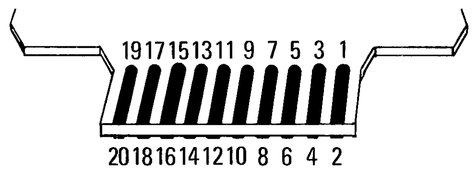
NOTE

Graphic characters must be accessed through an application program. ESC g enters graphic mode. The hex equivalent of each graphic character accesses and displays the graphic character.

Figure 5.5 Graphic Character Chart

5.6 VIDEO SIGNALS AND PINOUTS

The video signals are either directed to the built in monitor, or to an external monitor depending on what's plugged into the PC edge connector on the front panel of the Osborne 1. Normally, a shunt plug connects here which directs the video signals from the bottom of the PC board to the monitor connections on the top edge of the PC board. The following PC board pinouts apply:



pin	**2**	Ground	
	4	Brightness high	
	6	Brightness low	*Brightness potentiometer*
	8	Brightness arm	
	10	Ground	
	12	Horizontal sync	
	14	+12 volts	
	16	Video out	
	18	Vertical sync	
	20	Ground	

Figure 5.6 PC Edge Video Pinouts

5.6.1 Video In-line Connector

Normally, the shunt connects each of the above signals to the top side of the PC board, meaning that pin 2 connects to pin 1, pin 4 to pin 3, etc. In addition, a single inline connector located inside the Osborne 1 on the main logic board becomes active when the shunt is in place. This single in-line connector provides the signals and power to the internal video display monitor. The pinouts for this internal connector are as follows:

pin	**1**	Ground	
	2	Brightness high	
	3	Brightness low	*Brightness Potentiometer*
	4	Brightness arm	
	5	Ground	
	6	Horizontal sync	
	7	+12 volts 8 video out	
	9	Vertical sync	
	10	Ground	

Figure 5.6.1 Video In-line Pinouts

5.7 VIDEO CONNECTIONS AND CIRCUITRY

Figure 5.7A shows the connections between the top and bottom of the PC connector (P9) and the single In-line internal connector (P5):

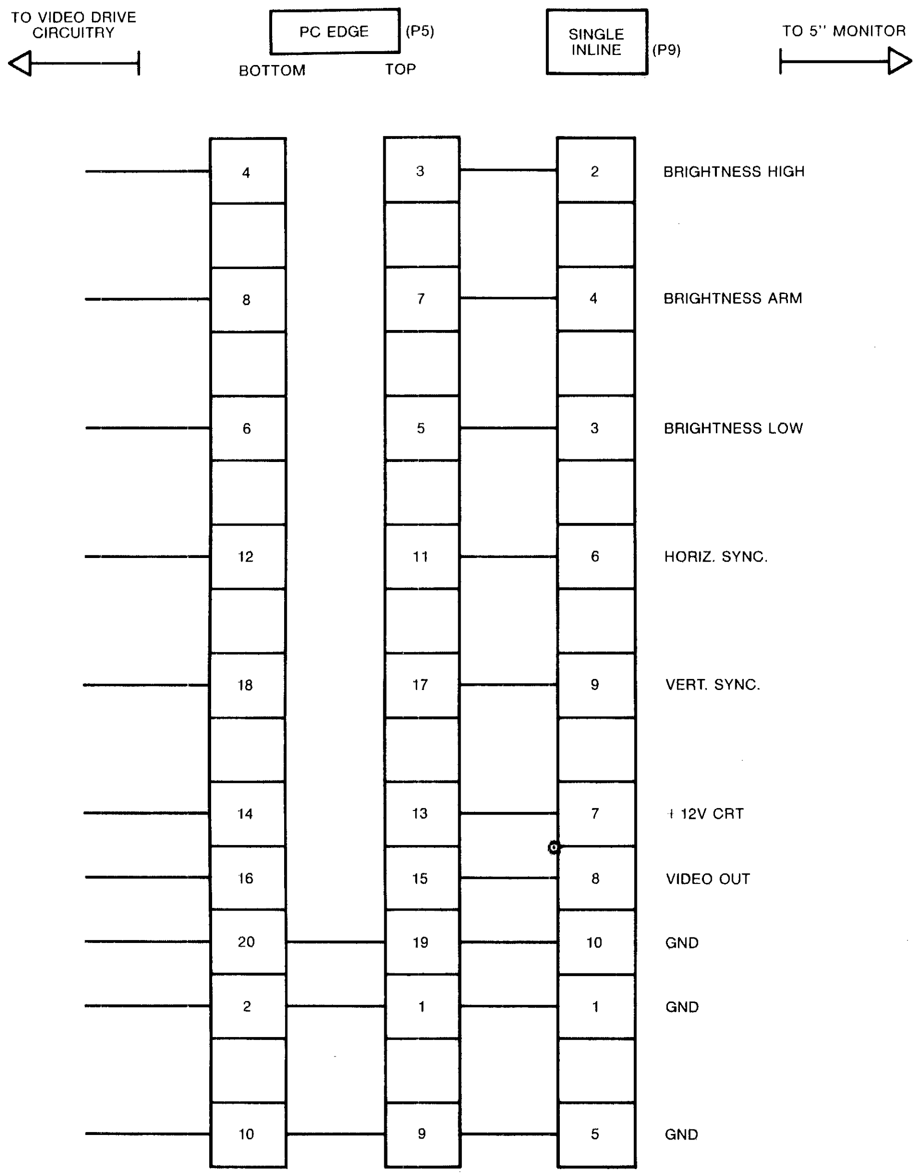


Figure 5.7A PC edge to Single Inline connections

The following diagram illustrates the applicable circuitry connected to this single inline connector:

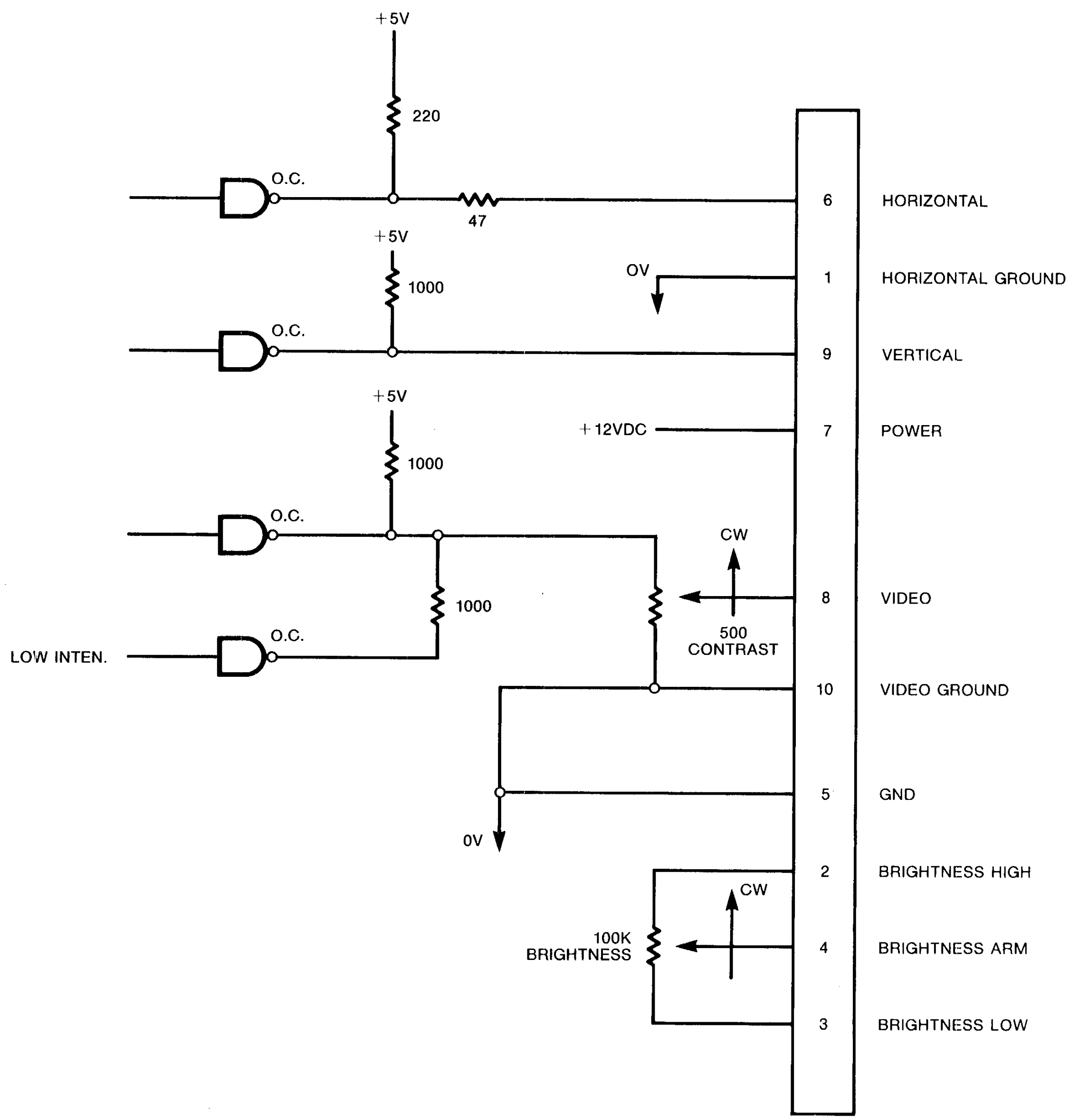


Figure 5.7B Video Drive Circuitry

5.8 VIDEO TIMING

The following illustrations show the horizontal (5.8A) and vertical (5.8B) timing used in the video display interface:

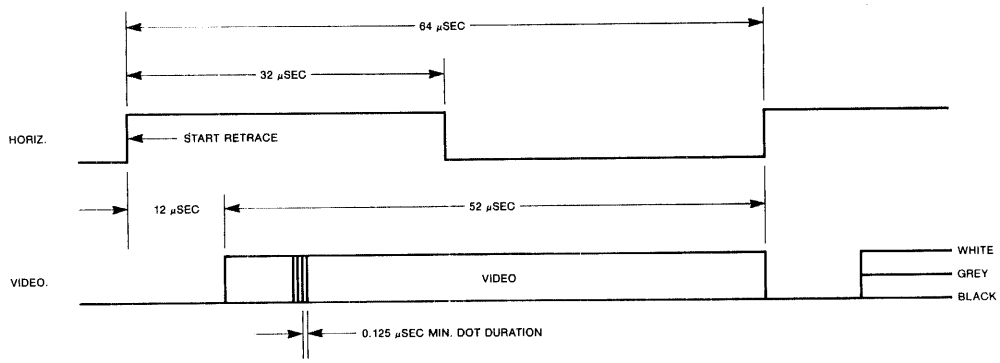


Figure 5.8A Horizontal Timing Diagram

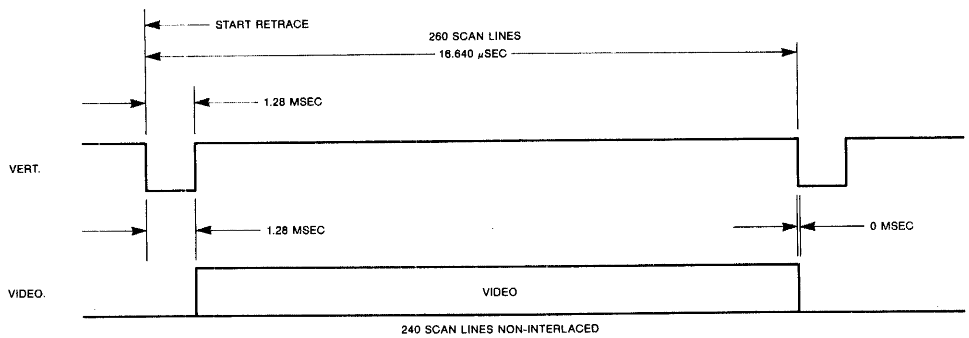


Figure 5.8B Verticle Timing Diagram

Warning: The video connectors contain TTL-level signals along with +12 Volts, and are quite vulnerable to damage by accidental misconnections. Be careful when you attach anything to either video connector.

The internal monitor is a P4 phosphorous monitor with an image area of 3.55 inches horizontally and 2.63 inches vertically. Rated linearity is 10% for adjacent characters, 15% overall.

Power Specifications

6.0 Power Specifications

6.1 POWER SUPPLY

The power supply for the Osborne 1 is an Astec switching power supply providing the following voltages and amperages:

+5 volts	2.5 amps
+12 volts	2.05 amps

Earlier versions of the power supply may be wired for 115v or 230v operation. To switch from one voltage to another, the wire with the bright orange tag indicating the current voltage setting must be removed from the pin it is currently connected to and attached to the adjoining pin. (See Figure 6.3)

6.2 SWITCHING VOLTAGE

Osborne 1 computers with the new blue case can operate at the 110V and 220V. Adapting the system for one of these voltages requires rotation of a fuse card accessable through the rear power well. The circuit breaker button has been replaced by this fuse and fuse card. Here is the procedure for switching from one voltage to another:

1. Disconnect the AC power cord from the power panel.

2. Slide open the transparent fuse box door in the AC power panel. Flip the fuse pull to remove the fuse.

3. The fuse card is located underneath the fuse mounting. The current AC voltage setting of the system is visible as a number printed on the card (100, 120, 220, or 240).

4. If you wish to change the indicated voltage setting, use needle-nose pliers to pull the fuse card from the fuse box.

5. Turn the card and replace it in the fuse box so that the desired voltage rating is the only number visible.

6. Replace the fuse and close the fuse box cover.

7. Reconnect the AC power cord to the power panel.

6.3 POWER BOARD LAYOUT

Here is the layout of the power supply board with the 115v and 230v pins identified:

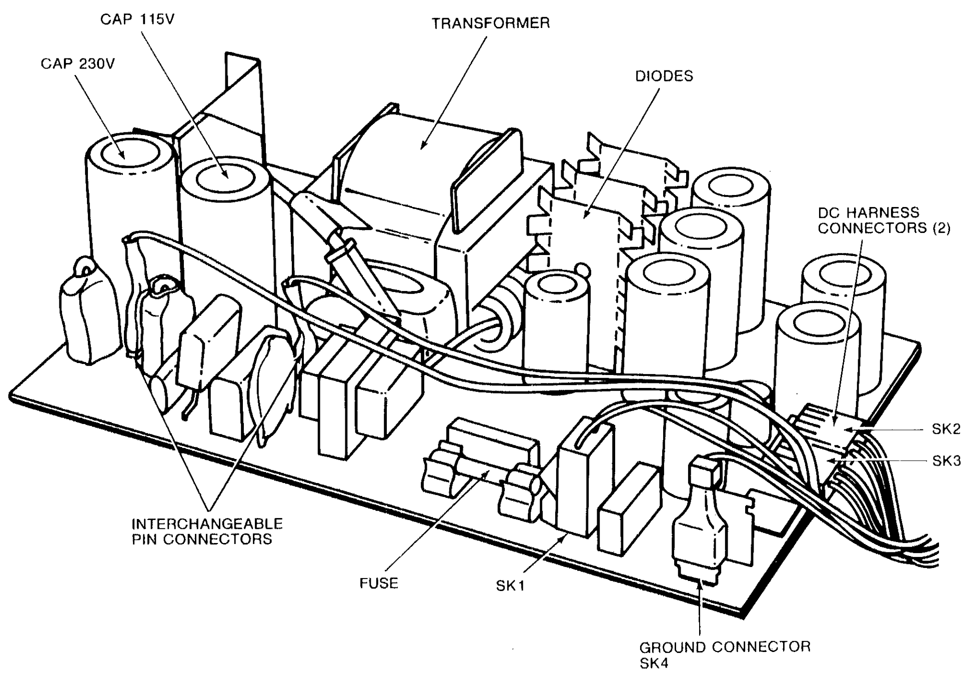


Figure 6.3 Power Board Layout

Pins 1 and 2 on the SK1 connector provide neutral and live in that order. The three connectors SK2, SK3, and SK4 are identical, and are laid out as follows when looked at from above:

power supply

P1	−12	----->
P2	+12	----->
P3	COM	----->
P4	+5	----->

The power supply is fused. If this fuse burns out, replace it with another fuse of the same type (T2A250V).

Disk Drives

7.0 Disk Drives

7.1 DRIVES USED

Two types of disk drives are used in the Osborne 1. The type supplied with your computer depends on when your Osborne 1 was made. Both Siemens and MPI disk drives are used, with the manufacturer-supplied electronics replaced with a Osborne-designed electronics board.

7.2 DISK DRIVE INTERFACE BOARD

As stated earlier, there have been numerous revisions on the disk drive interface board as described below:

Rev A — Original release
Rev B — Added jumper to U3 to stabilize erase current
Rev C — RN2 substituted with discrete resistors
Rev D — Added snubbing resistor and bypass capacitor
Rev E — Incorporation of all changes
Rev F — RN8 jump increases current through index LED
Rev G-J — Snubbing capacitor (R39), discrete resistor (C5) eliminated

Boards currently being shipped are designated Rev B which is distiguished from the earlier Rev B by a new part number 1B10081. Changes between Rev G and the new Rev B disk board are:

- Read filter was optimized for group delay response.
- Servo circuitry optimized and top-adjust speed control pot added.
- Stepper power-down option added.
- Tantallum capacitors and bypass capacitors were added to reduce board noise.

The layout of the disk drive board is shown in Figure 7.2.

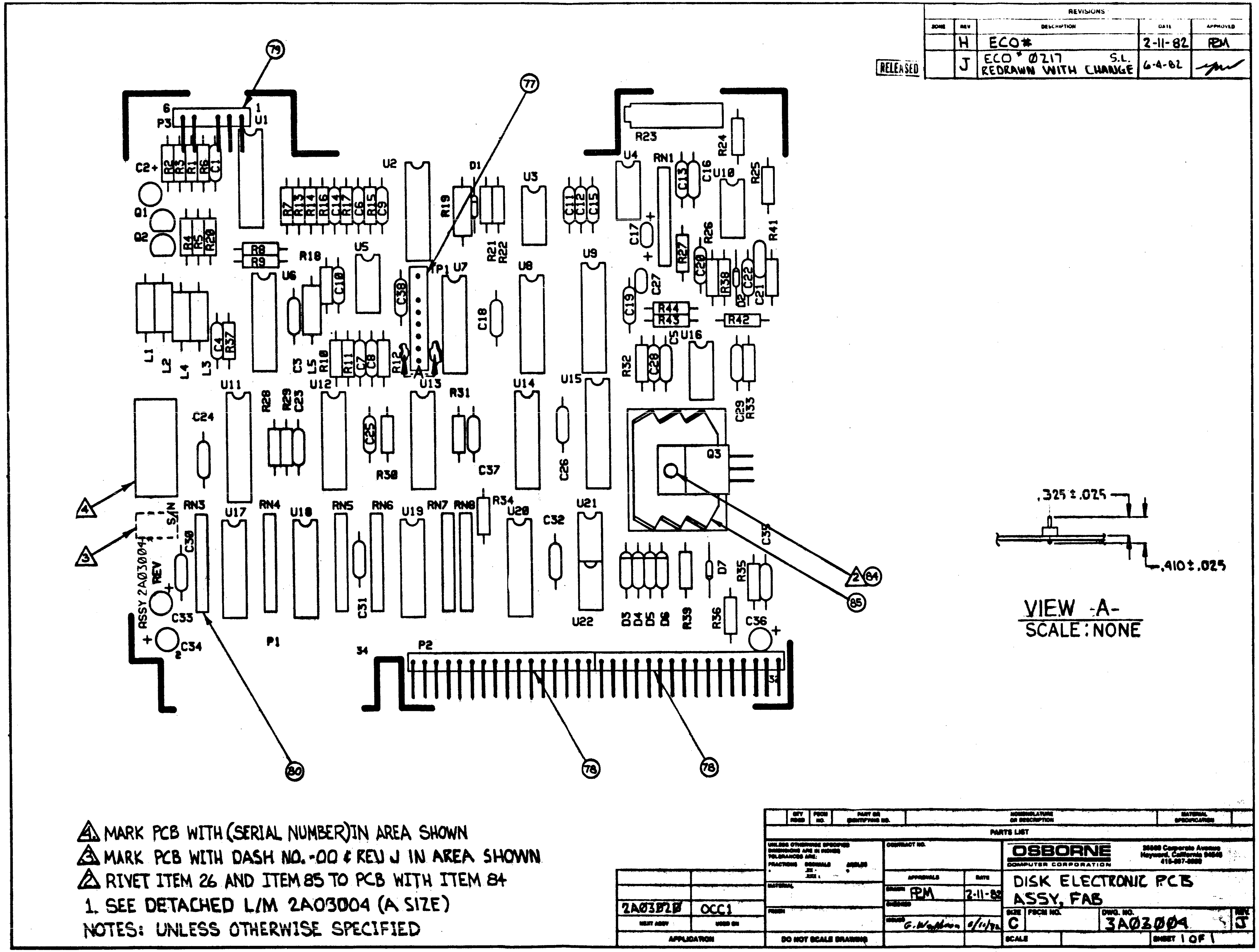


Figure 7.2 Disk Drive Electronics Board Layout

7.3 DISK CONTROLLER AND PINOUTS

The disk controller chip used on the main logic board is the Fujitsu 8877, equivalent to the Western Digital 1793. The following pinouts apply to the disk cable connector on the main logic board:

pin	**1**	GND
	2	GND
	3	GND
	4	GND
	5	GND
	6	GND
	7	GND
	8	INDEX
	9	GND
	10	DRIVE SELECT 1
	11	+12 volts
	12	DRIVE SELECT 2
	13	+12 volts
	14	NC
	15	+12 volts
	16	4mhz clock
	17	+12 volts
	18	DIR
	19	GND
	20	STEP
	21	+5 volts
	22	WRITE DATA
	23	+5 volts
	24	WRITE GATE
	25	+5 volts
	26	TRACK 00
	27	GND
	28	WRITE PROTECT
	29	GND
	30	READ DATA
	31	GND
	32	SIDE SELECT
	33	GND
	34	LATE

Figure 7.3 Main Board Disk Cable Connector Pinouts

7.4 CP/M BIOS DISK ADDRESSING

You can address the disk interface directly, although we strongly discourage this practice. If you must control the disk drives directly, do so through the standard CP/M BIOS calls:

BIOS + 18	Move head to track 0 on selected drive
BIOS + 1B	Select disk-drive number
BIOS + 1E	Set track number
BIOS + 21	Set sector number
BIOS + 24	Set the DMA address
BIOS + 27	Read the selected sector
BIOS + 2A	Write the selected sector
BIOS + 30	Translate the sector

NOTE

These calls are addressed with an offset from the start of BIOS for the particular revision of software being used.

If you're familiar with the way CP/M handles disk I/O, you also know that "BDOS functions" are accessible through memory location 0005 hex.

7.5 DISK INTERFACE

The Osborne 1 does not allow transfers of data directly to memory in the first 16K of memory space because the ROM and I/O in the second bank reside there. Instead, transfer information involving the first 16K of memory by first buffering the information in high memory (above BIOS) and then moving it into position. The opposite procedure occurs when you write information to the diskette from the initial 16K of memory. Use of the Z80 block-memory-move instruction makes this buffering transparent to users, and almost completely cancels any speed penalty involved.

7.6 DISK DRIVE SPECIFICATIONS

- maximum seek time 20 milliseconds track to track
- head load time 0 milliseconds rotation time 200 milliseconds
- utilizes standard FM recording

7.7 DISK FORMAT

The first track used for data is track 4 which contains the directory. Each entry consists of the standard CP/M format: one byte to indicate deletion, 11 bytes for the file name, and 20 bytes representing the "groups" assigned to the file.

There are two unusual aspects of the Osborne 1's use of the disk system. First, even though information is stored on the diskette in 10 sectors, to CP/M there are 20 sectors of 128 bytes each on the diskette. In other words, if you are using the Osborne 1 ROM routines, as documented, you'll be working with 10 physical sectors of 256 bytes, but if you're working with CP/M BIOS or BDOS routines, you'll be dealing with 20 logical sectors of 128 bytes each.

The disk medium used is the single-sided, single-density, soft-sectored 5 1/4-inch diskette. Data is stored on the diskette in 40 tracks of 10 256-byte sectors each, resulting in 102K of data storage per diskette. The Osborne double density option involves the addition of a small electronics board that interfaces between the main logic board and the disk drives. With the double density option installed, the Osborne 1 stores data in a multitude of different formats, including:

40 tracks 10 256-byte sectors (Osborne single density)
40 tracks 5 1024-byte sectors (Osborne double density)
40 tracks 8 512-byte sectors (IBM Personal Computer)
40 tracks 18 128-byte sectors (Xerox 820 Computer)
40 tracks 9 512-byte sectors (DEC 1820 double density)

7.7.1 CP/M File Control Block

Each file being accessed through CP/M must have a corresponding "File Control Block" which provides the name and allocation information for all subsequent file operations. The default FCB is located at BIOS + 05c hex and consists of 36 bytes of information:

5C	drive code (0=current, 1=A, 2=B)
5D 5E 5F 60 61 62 63 64	Filename
65 66 67	File Type
68	Current extent number
69	Reserved
6A	=0 on BDOS call to open, make, or search
6B	extent record count
6C 6D 6E 77 78 79	Disk record map
7A 7B 7C	Random record number

Figure 7.7.1 File Control Block

File Control Blocks are stored in the directory area of the disk and are brought into main memory when a file operation begins. The FCB in memory is updated as file operations occur and its attributes are recorded permanently when the file operation is completed.

Keyboard

8.0 Keyboard

8.1 THE KEYBOARD

The Osborne 1 keyboard contains no electronics--it consists of a column/row matrix of keyswitches which are converted into data and addresses by a 81LS95 and two 74LS05 chips. The data and addresses are then converted into ASCII key sequences by the monitor ROM, which uses a three-key rollover routine and maintains a lookup table for the conversion process.

8.2 KEYBOARD PINOUTS

The pinouts on the keyboard are as follows:

pin	**1**	ground	
	2	row 4	
	3	row 0	
	4	row 3	
	5	row 6	used for address
	6	row 2	
	7	row 5	
	8	row 1	
	9	row 7	
	10	col 0	
	11	col 1	
	12	col 2	
	13	col 3	used for data
	14	col 4	
	15	col 5	
	16	col 6	
	17	col 7	
	18	no connection	
	19	no connection	
	20	ground	

Figure 8.2 Keyboard Pinouts

8.3 KEYBOARD LAYOUT

The keys are laid out as follows (graphic characters are also shown):

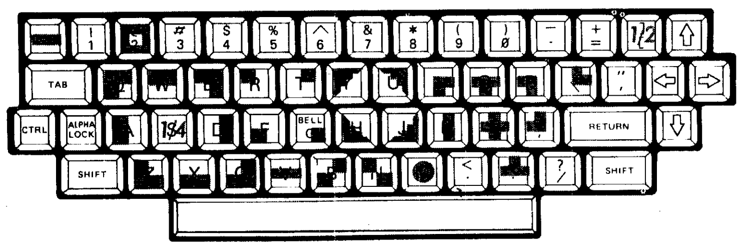


Figure 8.3 Osborne 1 Key Layout

8.4 KEYSWITCH MATRIX

The matrix used to convert the keyswitches to ASCII is as follows:

	P1								
A0	3	ESC	TAB	CTRL		SHIFT	RET / ENT	' ' / '	| / |
A1	8	1 / 1	2 / 2	3 / 3	4 / 4	5 / 5	6 / 6	7 / 7	8 / 8
A5	7	⇧	⇦	0 / 0	SP	> / %	P	O	9 / 9
A2	6	Q	W	E	R	T	Y	U	I
A3	4	A	S	D	F	G	H	J	K
A4	2	Z	X	C	V	B	N	M	< / ,
A6	5	⇨	⇩	— / -	? / /	: / ;	! / \	L	+ / =
A7	9				ALPHA LOCK				
GND	1 / 20	10	11	12	13	14	15	16	17
	GND	DO	D1	D2	D3	D4	D5	D6	D7

RFI AND STATIC SHIELD

Figure 8.4 Key Switch Matrix

8.5 PROGRAMMABLE KEYS

In addition to the normal ASCII keystrokes allowed, 14 programmable keys have been added to the Osborne 1. These keys («0 through «9, and the four arrow keys) are converted by a routine in BIOS to any series of keystrokes up to 96 (for all 14 keys). The function keys (0-9) are programmed through the SETUP program located on the System program diskette.

Assembly And Disassembly

9.0 Assembly And Disassembly

9.1 VARIATIONS IN DESIGN

This section provides the instructions needed to dismantel the Osborne 1 into its various components and re-assemble it. The case in which the computer is currently being shipped has been redesigned. The following instructions pertain to both the original and the new case; except where a slight deviation in procedure or extra step need be noted. Two sections on the Bezel/Chasis and the Video Monitor are provided since the difference between the two cases with regard to these items is substantial enough to warrant separate descriptions. Information in this section was extracted from the Osborne Field Service Manual.

9.2 KEYBOARD

9.2.1 Keyboard Disassembly:

CAUTION: Identify connector orientations before detaching any cable.

1. Disconnect computer from power source.

2. Disconnect the keyboard cable from the front Bezel by spreading the latches of the keyboard connector.

3. Using a pad or soft material to protect the keyboard, turn the keyboard assembly over so the keys face down.

4. Remove the 6/32 Phillips screw located at each latch end of the keyboard case.

5. Remove the keyboard case from the keyboard and keyboard bezel assembly.

6. Carefully remove the keyboard harness from the double back tape located on both the keyboard Bezel and keyboard. (There is no tape on the blue case).

7. Carefully remove the keyboard harness from the keyboard connector.

NOTE

The keyboard harness is aligned with Pin 1 of the keyboard connector. Looking at the underside of the keyboard with the numeric row of keys to the top, Pin 1 is the upper right Pin.

8. Remove four 8/32 Phillips screws which secure the keyboard to the standoffs.

9. Remove the keyswitch array from the keyboard Bezel.

9.2.2 Keyboard Assembly:

1. Place the keyboard Bezel onto a Pad or Soft material (to protect the keyboard), standoffs facing up.

2. Align the keyswitch array face down with cutouts on keyboard Bezel.

3. Install four 8/32 Phillips screws to secure the keyboard to the bezel standoffs.

4. (This step does not apply to the blue case). Locate the notch on the long edge of the keyboard Bezel furthest from the "space" bar. There should be a 1 1/2 - 2 inch length of double-backed tape on the underside of the Bezel at this notch. Affix tape if necessary.

5. (This step does not apply to the blue case). Check that there is also a 1 1/2 - 2 inch length of double-backed tape near the keyboard connector on the underside of the keyboard. Affix tape if necessary.

6. Connect Pin 1 (black stripe or thick ground wire) on the keyboard harness to Pin 1 of the keyboard connector on the underside of the keyboard.

NOTE

To locate Pin 1 of the keyboard connector, position the keyboard with the numeric row of keys to the top. Facing the underside of the keyboard, Pin 1 is in the upper right corner of the keyboard connector. On the keyboard Harness, Pin 1 is identified by a black stripe.

7. Position the keyboard harness in the notch on the long edge of the keyboard bezel and press it flat against both lengths of tape. (No tape on blue case).

8. Lower the keyboard case onto the keyboard and keyboard Bezel assembly so the Case and Bezel are flush.

9. Install the 6/32 Phillips screw located on each latch end of the keyboard case to secure it to the keyboard assembly.

10. Turn keyboard face up and connect the keyboard cable to the keyboard connector below the CRT screen.

9.3 ORIGINAL AND NEW BEZEL/CHASSIS

9.3.1 Original Bezel And Chasis Disassembly

1. Disconnect computer from power source.

2. Disconnect the keyboard cable from the front bezel by spreading the latches on the keyboard connector.

3. Remove all external cables (ie. RS-232, Modem, External Video, etc.).

4. Using a 1/20 inch Allen wrench, remove the video knobs.

5. Remove the 6/32 x 1/4 inch Phillips screws surrounding the bezel.

6. Placing thumbs below CRT screen and fingers in the disk storage pockets, apply a slight pressure inward with fingers, and pull bezel straight forward.

7. With the Osborne 1 resting flat on its rubber feet, turn it so the A/C power panel faces you.

8. Remove the two 6/32 x 1/4 inch Phillips screws from above and below the handle. Mark these screws FOR HANDLE and set aside.

NOTE

Using an incorrect screw may puncture the Disk harness running behind the handle assembly.

---

9. Remove the six 6/32 Phillips screws securing the A/C Power panel to the case. Note that the two upper screws have washers. Mark these two screws FOR A/C POWER PANEL and set aside. Using an incorrect screw may puncture the Drive harness.

10. With the Osborne 1 resting flat on its rubber feet, turn it so the CRT screen faces you.

11. Remove either the two or the four 8/32 x 1/2 inch Phillips screws which hold the chassis to the case. These are located on the left and right inside forward edges of the chassis.

12. Grasp the chassis assembly between the CRT and the Logic board. Lift the assembly slightly and pull it straight forward. Be careful when removing the chassis from the case. The A/C power panel will be dragging behind the chassis by wires only.

13. Pull remaining power cord through case.

9.3.2 Original Case Assembly:

1. Position the chassis assembly with disk drives on top, CRT screen facing you.

2. Place the case behind chassis assembly, AC power panel recess to the rear right.

3. Pull power cord through case until DC harness is taut or AC power panel is at case recess.

4. Grasp the chassis assembly between the CRT and the logic board. Lift the assembly slightly and push it straight back into the case. Be sure the AC Power panel is positioned correctly in its case recess.

NOTE

When installing chassis into case be sure not to pinch, trap or rip harness assemblies.

5. Install either two or four 8/32 x 1/2 inch Phillips screws on the left and right inside front edges of the chassis to secure it to the case.

6. Keeping the Osborne 1 flat on its rubber feet, turn it so the A/C power panel faces you.

7. Install six 6/32 Phillips screws to secure the A/C power panel to the case. The two upper screws MUST be 6/32 x 1/4 inch with washers. These were marked FOR A/C POWER PANEL. An incorrect upper screw may puncture the drive harness.

8. Install the two 6/32 x 1/4 inch Phillips screws above and below the handle. These screws were marked "FOR HANDLE". Using an incorrect screw may puncture the disk harness behind the handle assembly.

9. Keeping the Osborne 1 flat on its rubber feet, turn it so the CRT screen faces you.

10. Grasping bezel with thumbs below CRT screen cutout and fingers in the disk storage pockets, slide bezel completely into case.

11. Install the 6/32 Phillips screws surrounding the bezel to secure it to the case.

12. Using a 1/20 inch Allen wrench, install the brightness and contrast knobs onto their shafts.

13. With the keyboard assembly in front of the Osborne 1, connect the keyboard cable to the keyboard connector below the CRT screen.

9.3.3 New Case Disassembly:

CAUTION

The interior of the case is coated with a special metallic paint. Avoid scratching this coating.

1. Disconnect computer from power source.

2. Disconnect the keyboard cable from the front bezel by spreading the latches on the keyboard connector.

3. Remove all external cables (RS-232, modem, external video, etc.).

4. Pull the video contrast and brightness knobs off their shafts.

5. Remove the 6/32 x 1/4 inch Phillips screws surrounding the bezel.

6. Placing thumbs below CRT screen and fingers in the disk storage pockets, apply a slight pressure inward with fingers to loosen the bezel.

7. Grasp the bezel near the carrying-case latches and carefully pull it straight forward until it is free.

8. Remove the AC power cord from the power cord compartment.

9. Carefully turn the unit over so ventilation slots are on the bottom.

10. Remove five long 6/32 Phillips screws holding the two halves of the carrying case together.

11. Hold down the carrying handle plate and the power cord compartment and lift the upper half of the case free of the unit.

NOTE

The logic board and the power supply unit can be replaced at this point without further disassembly of the chassis. To further dismantle the unit, proceed as follows:

12. Slide the carrying handle plate out of the lower part of the case. Note that the broad area of the plate is below the handle at this point.

13. Lift the power cord compartment slightly and detach the door.

14. Place one hand under the front of the video monitor and the other at the back of the monitor and lift the chassis out of the case. Be sure that the AC power cord compartment lifts freely with the chassis.

9.3.4 New Case Assembly:

1. Position the chassis assembly so the logic-board is up, CRT screen facing technician.

2. Place the top half of the carrying case (the part with the ventilation slots) upside down with latches closest to you.

3. Place one hand under the front of the video monitor and the other at the back of the monitor and lift the chassis into the top half of the carrying case.

NOTE

When installing the chassis into the case, be careful not to pinch, trap, or rip harness assemblies.

4. Align screw holes in the chassis with the five mounting standoffs in the case.

5. Position the video harness and DC cables in the tab on the "B" drive shield.

6. Slide the AC power cord compartment into the case, fuse at the top and facing out.

7. Attach the power cord compartment door.

8. Slide the carrying handle plate into the case. The broad area of the plate is below the handle at this point.

9. Place the lower half of the case onto the chassis assembly. Be sure that the AC power compartment, carrying handle plate, and upper and lower halves of the case align properly.

10. Install five long 6/32 Phillips screws which hold both halves of the case together. Tighten these screws until they are snug. Do not overtighten!

11. Turn the unit over so the ventilation slots are on top and CRT screen facing you.

12. Placing thumbs below CRT screen cutout and fingers in the disk storage pockets, slide the bezel into the chassis.

13. Install the 6/32 Phillips screws which secure the bezel to the front of the chassis. Do not overtighten these screws!

14. Carefully push the video control knobs onto their shafts.

15. With the keyboard assembly in front of the unit, connect the keyboard cable to the keyboard connection below the CRT screen. Facing the screen, the cable connects from the left.

9.4 LOGIC BOARD

9.4.1 Logic Board Disassembly:

1. Position the chassis assembly with the logic board facing up and the CRT screen facing you.

2. Remove the 6/32 Phillips screw at each corner of the logic board. The screw in the right front corner of some logic boards has a nylon insulation washer.

3. Lift the logic board by the front edge and hold it perpendicular to the chassis, video control shafts pointing straight up. Make note of all connector orientations while still connected.

4. Remove the DC harness connector located in the extreme lower left corner of the logic board. The DC harness connector is not keyed. The RED wire on the harness must go to the left. (On the blue case, the connector is keyed by the "lip" on one side. 5. Remove the disk harness connector located in the lower left corner of the logic board at the right of the DC harness connector. Grip the connector and CAREFULLY detach it, being sure not to bend any pins.)

NOTE

The disk drive harness connector is not keyed. The RED stripe on the harness must go to the right.

6. Remove the video harness connector located in the upper left corner of the logic board to the right of the reset button.

NOTE

The Video harness connector is not keyed. The RED wire on the harness must go to the left.

7. Remove external video plug located between the reset button and video contrast shaft on the front edge of the logic board.

8. Remove the logic board.

9.4.2 Logic Board Assembly:

1. Position the chassis assembly with its logic-board side up, and CRT screen facing you.

2. Hold the logic board perpendicular to the chassis, component-side facing you, video control shafts pointing straight up.

3. Connect the external video plug to the connector on the edge of the logic board between the reset button and video contrast shaft.

4. With the video harness running below the logic board, attach the video harness connector to the 10-Pin connection on the logic board between reset button and the contrast shaft, RED WIRE TO THE LEFT.

5. With the DC harness running below the logic board, attach the DC harness connector to the 7-Pin connection in the extreme left lower corner of the logic board, RED WIRE TO THE LEFT.

6. With the disk harness running below the logic board, attach the disk harness connector to the 34 Pin connection located at the lower left, of the logic board's to the right of the DC harness. RED STRIPE TO THE RIGHT.

7. Lower the logic board onto its chassis mounting blocks.

8. Install a 6/32 Phillips screw with star washer at each corner of the logic board. The screw in the right front corner of some older logic boards has a nylon insulation washer to protect the trace.

9.5 ORIGINAL AND NEW VIDEO MONITOR

9.5.1 Original Monitor Disassembly:

1. Disassemble the chasis and bezel following earlier instructions.

2. Position the chassis assembly with the logic-board side up and CRT screen facing you.

3. Remove the four 6/32 Phillips screws securing the monitor to the chassis.

4. Grasp the left and right sides of the chassis assembly and carefully lift away to expose the underside of the monitor.

5. Remove the video harness connector located at the top rear of the video PC board. This is a keyed connector.

9.5.2 Original Monitor Assembly:

1. Position the chassis assembly vertically with handle on work bench and drive shields facing you.

2. If the video harness has been removed, insert the small end-connector of the harness through the left rear slot of the video shield from the inside.

3. Place the video monitor in front of chassis assembly, CRT screen facing you and keyed PC board connection at top rear.

4. Connect the keyed large end-connector of the video harness to the rear of the monitor PC board.

5. Grasp the left and right sides of the chassis assembly. Lift the chassis and carefully lower it onto the video monitor.

6. Align the monitor with the four screw holes in the chassis. install four 6/32 Phillips screws to secure the monitor to the chassis.

9.5.3 Blue Case Monitor Disassembly:

1. Position the chassis assembly with its logic-board side up and CRT screen facing you.

2. Slip the video harness cable from its tab on the chassis.

3. Disconnect the slip-on ground wire from its connection at the back of the video monitor shield.

4. Remove the disk drive from the four fasteners holding it to the back of the video shield.

5. Remove the video harness connector located at the top rear of the video PC board. This is a keyed connector.

6. Remove four 6/32 Phillips screws securing the monitor to the chassis assembly.

7. Remove the video monitor from its shield.

8. Remove the transparent face plate from the monitor screen.

9.5.4 Blue Case Monitor Assembly:

1. Position the video monitor in front of the chassis assembly, CRT screen facing you and keyed video PC board connection at top rear.
2. Lower the video monitor shield onto the monitor and align the monitor and monitor shield screw holes.
3. Connect the keyed, large end-connector of the video harnesss to the rear of the monitor PC board.
4. Grasp the left and right sides of the chassis assembly. Lift the chassis and carefully lower it onto the video monitor. Do not crimp any wires.
5. Align the monitor with the four screw holes in the chassis. Install four 6/32 Phillips screws to secure the monitor to the chassis.
6. Attach the slip-on ground connector to the connection at the rear of the video monitor shield.
7. Slip the disk drive harness into the four fasteners on the back of the video monitor shield.
8. Replace the transparent face plate on the monitor screen.

9.6 POWER SUPPLY

9.6.1 Power Supply Disassembly:

WARNING

Working with Power Supplies is DANGEROUS. Power Supplies can hold an electrical charge for long periods of time. Be careful not to touch any components unnecessarily!

1. Disassemble the Osborne-1 chasis and bezel as described earlier.
2. Position the chassis assembly logic-board-side-up and CRT screen facing away from you.
3. Remove the 6/32 Phillips screw from each corner of the power supply unit.

4. With the wires still attached, carefully lift the power supply out of the chassis.

5. Turn the power supply over left-to-right so the components face you and the five large capacitors are in the lower left corner.

6. Remove the DC harness connector from the power supply. This keyed connector is attached to one of the three identical male connectors on the left side of the power supply.

7. Remove the ground wire connector from the power supply. This is a slip-on connector located in the upper left corner of the power supply.

8. Remove the AC input connector from the power supply. This is a keyed connector located left of the fuse on the upper side of the power supply.

9. (This step applies only to the blue case). Remove both interchangeable pin connectors from the 115V and 230V pins on the power supply.

9.6.2 Power Supply Assembly:

NOTE

Power Supply procedures are with DC harness installed in Chassis.

1. Position the chassis assembly with its logic-board side up and handle facing you.

2. Turn the power supply so the components face you and the five large capacitors are in the lower left corner.

3. Attach the AC input connector to the keyed connector located left of the fuse on the power supply.

4. Attach the slip-on ground wire connector to the ground connection in the upper left corner of the power supply.

5. Attach the keyed DC harness connector to one of the three identical male connectors on the left side of the power supply.

6. With the wires attached, turn the power supply over, from right to left, so the DC connector is to the right. Carefully insert the power supply into chassis.

7. Align the screw holes on the power supply PC board with the chassis assembly standoffs.

8. Install a 6/32 Phillips screw and washer in each corner of the power supply. Use a metal star washer in the upper right corner. Use nylon washers on the other three corners.

9. (This step applies only to the blue case). Slip the power supply harness into its tab on the chassis.

9.7 DISK DRIVES

9.7.1 Disk Drive Disassembly:

NOTE

The A drive has an 8-pin 150-OHM Terminator resistor pack. B DRIVE DOES NOT.

---

1. Disassemble the Osborne 1 chasis and bezel as described earlier.

2. Position the chassis assembly with its logic-board side up and handle facing away from you.

3. Remove the four 6/32 Phillips screws holding the "A" drive to the chassis assembly.

NOTE

Disk drive A is the drive closest to the power supply.

---

4. With the wires still connected, pivot the shielded drive horizontally to the right 90 degrees from its original position.

5. (This step applies only to the blue case). Remove the disk harness connector and the ground connector from drive. The disk harness connector is at the rear of the drive PC board. The slip-on ground

connector is located at the rear of the drive either on the drive frame or the drive shield.

6. Remove the two 6/32 Phillips screws which hold the shield to the drive. These screws are located on the left and right sides of the drive.

7. Hold the drive shield down and lift the drive up enough to access the rear of the drive.

8. Remove the disk harness connector and the ground connector from drive. The disk harness connector is at the rear of the drive PC board. The slip-on ground connector is located at the rear of the drive either on the drive frame or the drive shield.

NOTE

The disk drive harness is not keyed. Facing the back of the drive with the strobe wheel side up, the harness is always connected RED STRIPE to the RIGHT.

9. Remove drive from shield.

10. If drive A is being replaced with another, remove the 8 pin terminator from the drive PC board and KEEP IT for installation on the new drive. The terminator is located at position RN3 on the right rear corner of the PC board.

9.7.2 Disk Drive Assembly:

1. Install terminator resistor pack at position RN3 of PC board. (for Drive A)

2. Place drive in shield with PC board facing down.

3. Position the chassis assembly with its logic-board side up and handle facing away from you.

4. Place shielded drive next to chassis assembly, strobe wheel facing up, drive door facing away from chassis.

5. Hold the drive shield down and move the drive enough to access the rear of the drive.

6. Insert the broad drive connector on the end of the drive harness through the rectangular slot at the rear of the drive shield. Connect the drive connector to the rear of the PC board, RED STRIPE on harness edge closest to you.

7. Connect the narrow slip-on ground connector to the male tab located at the rear of the drive either on the drive frame or the drive shield.

8. Align drive with screw holes in the drive shield. Install one 6/32 Phillips screw with star washer in each side of the drive.

9. Slide drive under chassis assembly with drive door facing you. Install the four 6/32 Phillips screws which secure the "A" drive to the chassis.

9.8 POWER PANEL

Figure 9.8 identifies the DC harness and its connectors:

9.8.1 Power Panel Disassembly:

1. Disassemble the Osborne 1 chasis and bezel as discribed earlier.

2. Remove the ground wire to the video monitor shield.

3. Remove the ground wire from each disk drive shield.

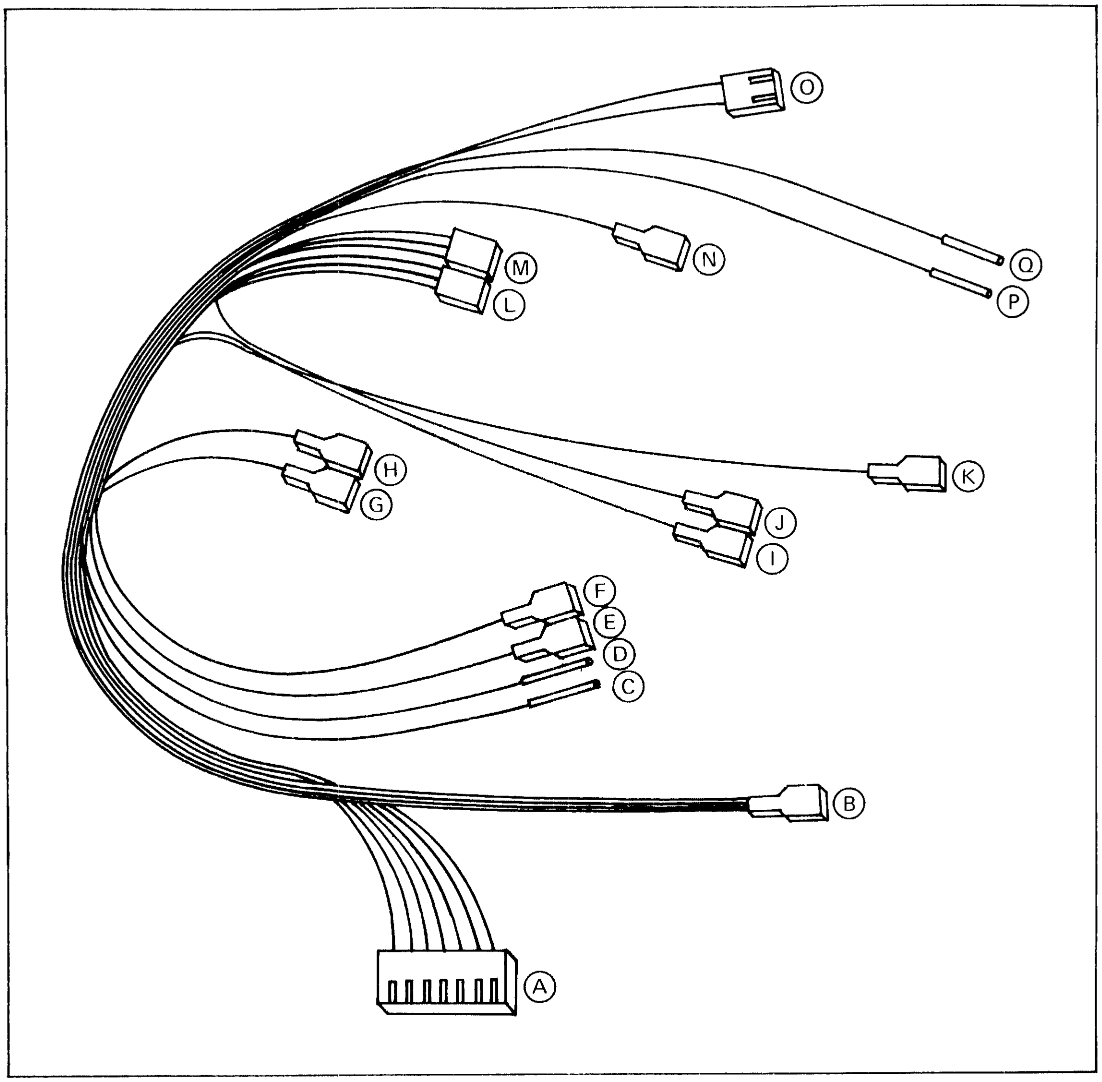
4. Remove the thermal relay from its clip on the chassis. The AC power panel should now be completely disconnected from the chassis.

5. Remove the screws which hold the AC power panel to the back of the power panel compartment.

6. Position the AC power panel with connections facing the technician, switch assembly in lower left corner, ground wires in lower right corner.

7. Disconnect the five-wire slip-on ground connector from the lower right corner of the power panel.

8. Disconnect both interchangeable pin connectors from pins "C" and "D" of the power panel.



A	Logic Board DC Input	K	"A" Drive Ground
B	AC Power Panel Ground (5 wire)	L and M	Interchangeable DC Output Connectors from Power Supply
C and D	Interchangeable 115V and 230V Pin Connectors toAC Panel	N	DC Power Supply Ground
E and F	Power Panel AC Outputs	O	AC Input Connector
G	"B" Drive Ground	P and Q	Interchangeable 115V and 230V Pin Connectors to Power Supply
H	Video Monitor Ground		
I and J	Interchangeable Thermal Cutout Connectors		

Figure 9.8 DC Power Harness

9. Disconnect both lower wires from the switch assembly in the lower left corner of the power panel. Do not disconnect the upper wires.

10. Detach the thermal relay from the wire harness.

9.8.2 Power Panel Assembly

1. Connect two slip-on connectors, I and J, to the thermal relay.

2. Position the AC power panel with connectors facing you, switch assembly in lower left corner, and ground wire connnection at lower right.

3. Connect the five-wire ground cable B to the connector in the lower right corner of the power panel.

4. Connect the AC output wires E and F to the lower connectors on the power panel switch assembly. Connect F, which runs to the thermal relay, to the right of E.

5. Connect two interchangeable pin connectors, C and D, to pins "C" and "D" of the power panel.

6. Align the power panel with the screw holes on the rear of the power panel compartment. Position the fuse box furthest from the door hinge cutouts on the power panel compartment.

7. Install four 6/32 Phillips screws which secure the power panel to the back of the power panel compartment.

8. Fasten the thermal relay to its chassis mounting clip beneath the power supply unit.

9. Place the thermal relay wires I and J in their notch on the chassis. This prevents pinching of the wires between chassis and case.

Osborne 1 Software

10.0 Osborne 1 Software

10.1 SOFTWARE REVISIONS

Following are the monitor ROM and BIOS revision levels to date, and a summary of the modifications that have been made:

ROM revisions

REV A — contains diagnostics, limits boot to one fixed address. This version was subsequently upgraded.

REV 1.2 — supports configurable CP/M system through MOVCPM, adds three key rollover.

REV 1.3 — Clock removed, disk drivers modified.

REV 1.4 — allows multi-density format, I/O functions moved from BIOS to ROM, graphics offset added, and increased size of BMRAM.

REV 1.4.1 — fixes reset jump

BIOS revisions

Rev A — original BIOS; subsequently upgraded.

Rev 1.2 — supports parallel IEEE, adds function keys and printer protocals (XON/XOFF, ETX-ACK).

Rev 1.3 — allows WordStar to run from serial terminal, fix of XON/XOFF and timing of serial port.

Rev 1.4 — adds printer initialization string, IEEE device addressing, BIOS modified to preserve IX and IY registers, added ability for recognition of different disk formats.

Since improvements to ROM and BIOS and subsequent release of this software were made in tandem, both have the same revision numbers. You can determine the address of BDOS and CCP for any version by using the starting address of BIOS for each release shown below:

Version #	start of BIOS address
Rev A	EA00
Rev 1.2	E600
Rev 1.2.1	E500
Rev 1.3	E500
Rev 1.4	E100

10.2 THE MONITOR ROM

The monitor ROM consists of 10 assembly language source modules:

- ROM initialization and startup
- CP/M boot routines
- Console routines
- PIA routines
- ACIA routines
- Disk routines
- Diagnostic routines (not after 1.2)
- Formatting routines
- Utilities
- RAM linkages

The initialization routines set the computer into a known state, display a message to the user, then wait for the user's response. Three responses are allowed:

RETURN	boot from disk drive A
"	boot from disk drive B
^[	homes screen

In addition, a jump table is located at location 0100 hex which parallels the jump table of BIOS.

The bootstrap loader in ROM loads CP/M from the diskette using the following assumptions:

Pre-1.4 ROM single density

track 0	sectors	1 - 8	CCP
	sectors	9 - 10	BDOS
track 1	sectors	1 - 10	BDOS
track 2	sectors	1 - 2	BDOS
	sectors	3 - 10	BIOS

1.4 ROM

track 0	sectors	1 - 2	CCP
	sectors	3 - 5	BDOS
track 1	sectors	1 - 1.5	BDOS
	sectors	1.5 - 2.5	BIOS

The boot routines in BIOS are responsible for displaying the sign-on message (if any), and for setting up the base page parameters.

10.2.1 Console Routines

The console routines include the special commands which the Osborne 1 recognizes as terminal control functions. In particular, the Osborne 1 emulates many of the functions which control the TeleVideo 912 and 920 terminals:

Hex Sequence	ASCII Control Code	Description of Action
07	^G	rings the bell
0B	^K	moves cursor up
0A	^J	moves cursor down
0C	^L	moves cursor right
08	^H	moves cursor left
1A	^Z	clears screen and homes cursor
1E		homes cursor
1B 23	ESC #	locks keyboard
1B 22	ESC "	unlocks keyboard
1B 3D	ESC =	cursor XY positioning
1B 53	ESC S	screen XY positioning
1B 51	ESC Q	insert character
1B 57	ESC W	delete character
1B 45	ESC E	insert line
1B 52	ESC R	delete line
1B 54	ESC T	clear to end of line
1B 29	ESC)	start half intensity display
1B 28	ESC (	end half intensity display
1B 4C	ESC L	start underline display
1B 4D	ESC M	end underline display
1B 67	ESC g	start graphics display
1B 47	ESC G	end graphics display
1B 5B	ESC [	homes screen

The console routines are responsible for keyboard input, console output, bell ringing, updating the real-time clock, and checking to see if the disk drive should be deactivated.

10.2.2 Other Interface Routines

The parallel, serial, and disk interface routines directly control the 6821, 6850, and 1793 chips through memory-mapped I/O.

Diagnostics for the keyboard, memory, and diskette are provided in the ROM, as well as routines to bootstrap from either disk drive, and to load special test programs via the serial port if the disk drives are not functioning. These diagnostics have been removed from the 1.4 ROM and Osborne Computer Corporation will be providing an optional system test to replace the missing diagnostics in the near future.

10.2.3 ROM Listings

The current ROM was coded by several different organizations under the direction of Osborne Computer Corporation. Nevertheless, all maintanence and modification of the ROM is performed by the software department at Osborne Computer Corporation. The listings that follow were created with the Sorcim ACT I assembler and fully utilize Z80 code.

Listings for the 1.3 Monitor ROM, followed by the most current 1.4 Monitor ROM are included at the back of this manual.

10.3 BDOS CALLS

Function Number	Description	Entry Values	Return Values
0	system reset		
1	console read		A=character
2	console write	E=character	
3	reader read		A=character
4	punch write	E=character	
5	list write	E=character	
6	direct I/O	E=type*	A=0 if busy
7	get IOBYTE		A=IOBYTE
8	set IOBYTE	E=IOBYTE	
9	print string	DE=address	
10	read console buffer	DE=address	
11	get console status		A=status
12	get version number		HL=version #
13	reset disk		
14	select disk	E=drive	A=found ***

Function Number	Description	Entry Values	Return Values
15	open file	E=drive	A=found ***
16	close file	DE=FCB address	A=found ***
17	search for file	DE=FCB address	A=found ***
18	search for next		A=found ***
19	delete file	DE=FCB address	A=found ***
20	read next record	DE=FCB address	A=found ***
21	create file	DE=FCB address	A=found ***
22	create file	DE=FCB address	A=found ***
23	rename file	DE=FCB address	A=found ***
24	get login vector		HL=drive
25	get disk number		A=drive
26	set DMA address	DE=DMA address	
27	get allocate vector		HL=allocation
28	write protect		
29	get R/O vector		HL=R/O vector
30	set file attributes	DE=FCB address	A=found ***
31	get disk header address		HL=address
32	set/get user number	D=FF (get) code (set)	A=user #
33	read random	DE=FCB address	A=error ***
34	write random	DE=FCB address	A=error ***
35	compute file size	DE=FCB address	A=record
36	set random record	DE=FCB address	A=record
37	reset drive	DE=drive vector	A=0
38	write random zero fill	DE=FCB address	A=error ***

coded in C register
*type: FF=input from console
all else=character to output to console

**status: 00=not ready
FF=ready

***found: FF=not found
00=valid entry

****error: 1=reading unwritten data
3=cannot close current extent
4=seek to unwritten extent
5=directory overflow
6=seek past physical end of disk

Drive numbers are coded: 0=A 1=B

NOTE

You must set the random record position (7D-7F hex) before reading or writing random records.

10.4 OSBORNE 1 BIOS ROUTINES

The BIOS routines supplied by Osborne Computer Corporation include several additional functions to the standard BIOS CP/M expects.

10.4.1 The IOBYTE

The IOBYTE is a reserved memory location (0003 hex) which defines the current assignment of physical to logical devices. The IOBYTE is divided into four distinct fields for the four logical devices recognized by CP/M. The logical device names are now obsolete (i.e., do not match Osborne physical device names) and are used only for the sake of maintaining the standard CP/M nomenclature. There are four logical devices each of which takes up two bits as follows:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LIST		PUNCH		READER		CONSOLE	

A value in the range 00 to 11 hex (0 to 3 decimal) determines the assignment of each logical device as follows:

Logical Device	Value	CP/M Physical	Osborne Physical
Console	00	TTY:	Keyboard + Screen
(CON:)	01	CRT:	Serial Port
	10	BAT:	Parallel Port
	11	UC1:	IEEE Port
Reader	00	TTY:	Keyboard + Screen
(RDR:)	01	PTR:	Serial Port
	10	UR1:	Parallel Port
	11	UR2:	IEEE Port
Punch	00	TTY:	Keyboard + Screen
(PUN:)	01	PTR:	Serial Port
	10	UP1:	Parallel Port
	11	UP2:	IEEE Port
List	00	TTY:	Keyboard + Screen
(LST:)	01	CRT:	Serial Port
	10	LPT:	Parallel (Centronics)
	11	UL1:	IEEE Port

Besides fully implementing the CP/M IOBYTE, routines are provided to drive printers which require XON/XOFF or ETX/ACK protocols. Automatic horizontal scrolling, screen size for word wrap, default baud rate and programming of the function keys are maintained by the BIOS.

10.4.2 BIOS Listings

The BIOS listings are included at the end of this manual.

Theory Of Operations

11.0 Theory of Operations

11.1 INTRODUCTION

This chapter provides a functional description of the Osborne 1 Computer functions. To understand this chapter, an understanding of digital Logic and TTL (transistor-transistor-Logic) is required.

11.2 FUNCTIONAL OPERATION

The major functional areas of the Osborne 1 Computer as shown on Figure 11.2 are: Basic Timing, CPU, ROM, RAM, Video System, Parallel I/O, Serial I/O, Keyboard, and Disk System.

The basic timing originates with a 16 MHz crystal oscillator. The 15.9744 MHz signal is divided down to provide nominal 8, 4, 2, and 1 MHz signals to control other circuits of the system.

The (CPU) Central Processor Unit controls and communicates with the remainder of the system through an 8-bit data bus and a 16-bit address bus. In addition to the two busses shown on the diagram, the CPU has several additional inputs and outputs that are not shown on the block diagram. These inputs and outputs are control functions and will be described later.

The (ROM) Read Only Memory is a 4 kilobyte Read-Only Memory with tri-state outputs connected to the data bus. When the ROM is enabled, data stored at the address on the address bus is put on the data bus.

The Random Access Memory (RAM) is a 64 kilobyte dynamic memory. The RAM data outputs are tied to the data bus through a gate that is enabled only when the CPU accesses RAM. The RAM can be accessed by the video logic also. RAM output for the video display is latched, then fed into the address inputs of a two kilobyte EPROM, called the character generator. This EPROM has dot patterns stored in it and is not connected with the CPU data or address busses. Its outputs go to a shift register where the dot pattern is shifted out (one at a time) to the video interface, and then to the CRT.

The disk system consists of a disk controller and two disk drives. It receives and transmits data on the CPU data bus and is addressed by the address bus. The disk system uses single density Osborne disks.

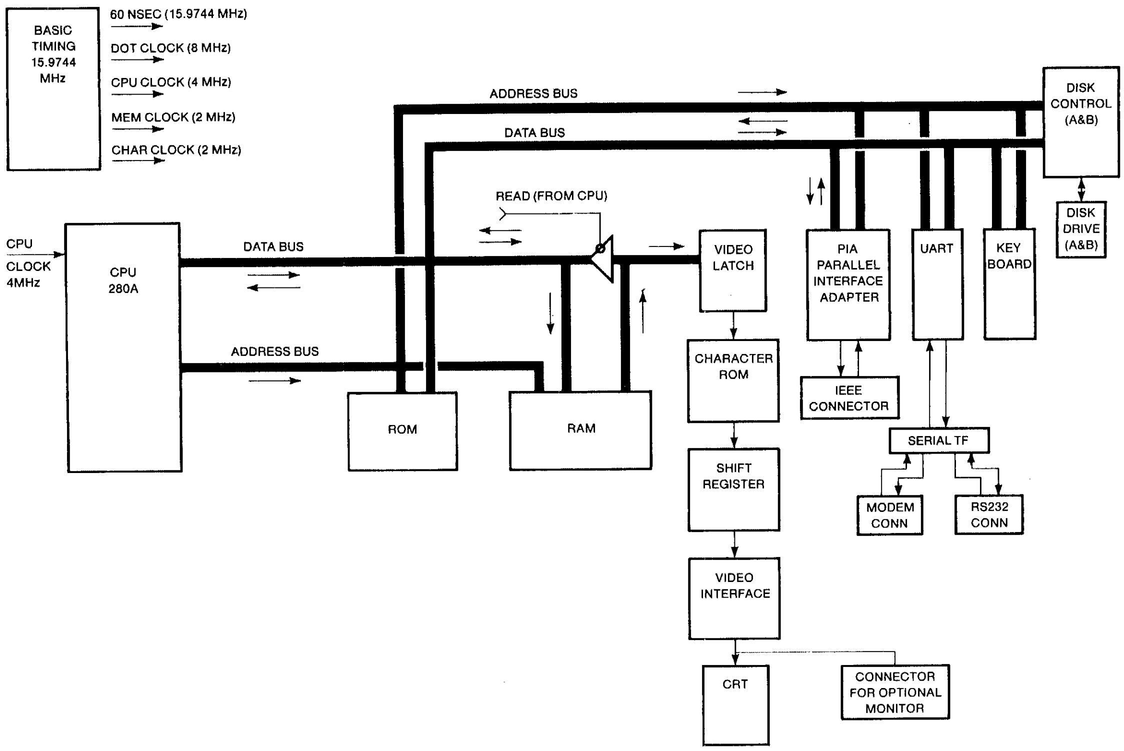


Figure 11.2 — Functional Block Diagram

The Osborne 1 can interface with IEEE 488 and RS-232-C standard external busses. There is also a MODEM port. The RS-232-C serial data communications port uses a universal asynchronous receiver/transmitter (UART). The UART can send and receive, through an external MODEM (MOdulator/DEModulator), over telephone lines.

Another connector on the front panel is for an optional larger monitor; the last connector is for connecting an optional battery pack for operating the computer where no commercial power is available.

11.3 MNEMONIC CODES:

Mnemonic	Description	Function
CAS	Column Address Strobe	Strobes RAM column address
DOT CLK	Dot clock	Used in conjunction with the Character Generator and video logic to produce dot patterns on display
NMI	Non Maskable Interrupt	Restarts ROM program
RAS	Row Address Strobe	Stobes RAM row address

11.4 BASIC TIMING

Crystal controlled oscillator X1 operates at 15.9744 MHz. The oscillator output is buffered by UB13. The oscillator circuit output at a nominal 16 MHz is used as 62 NSEC timing signal and is also used to clock 4-bit synchronous counter UB11 (on the positive edge of the clock) to develop additional timing signals of 8, 4, 2, and 1 MHz (Figure 11.4).

The 8 MHz is inverted by UA11 and becomes the dot clock (DOT CLK). One of the destinations of DOT CLK is Shift Register UA14 (Sheet 4 of the Schematic); UA14 is positive edge sensitive which is the reason for inverting DOT CLK. The active edge of DOT CLK is also the active edge of signals that follow at lower frequencies or lower rates. UA14 will load synchronously with the rising edge of DOT CLK when a load pulse (H COUNT*) Active low is indicated by (*). is applied. H COUNT* is generated by decoder UB1 (Schematic, Sheet 2). UB1 Pin 9 will be active during the counts E to F of counter UB11 which generates H COUNT*.

The Memory Counter is UD3. During the time H COUNT* is active synchronous with the rising of the 8 MHz clock, Shift Register UA14 will load new data from the video circuitry. The character on the screen is bracketed by the H COUNT* pulse which sets an inflexible limit in timing. A horizontal display line consists of 64 pulses including video retrace blanking which leaves 52 characters visible. Each character time is very nearly 1 microsecond, so that the vertical frame rate is in full synchronization with the AC line (60 cycles).

At the decoder UB1 (Schematic, Sheet 2), there is an active low CHAR CLK* at Pin 15. CHAR CLK is inverted by UE12 (Schematic, sheet 8). The other two

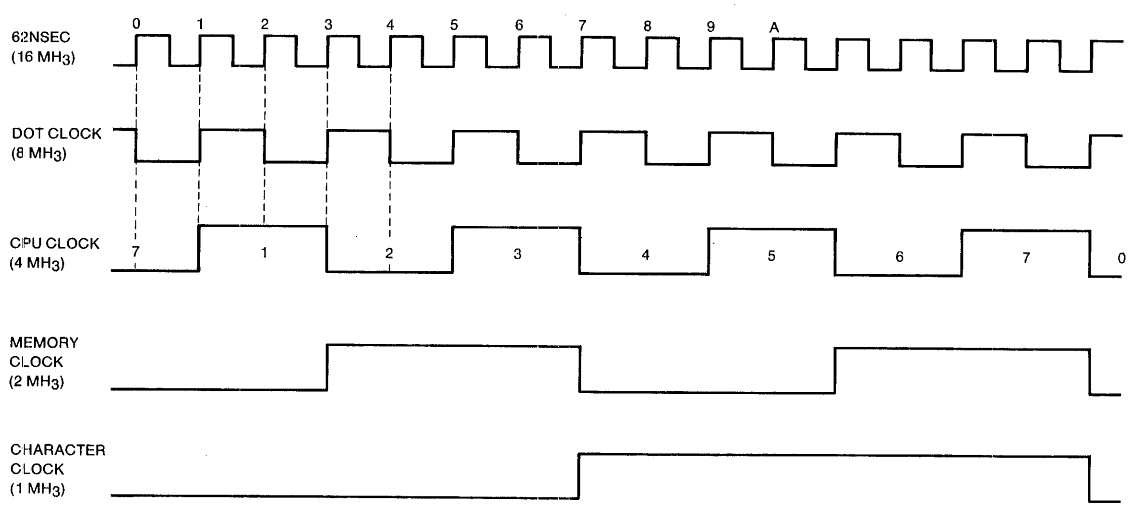


Figure 11.4 — Basic System Timing

inputs to the decoder are CPU CLK and MEM CLK. CPU CLK is a 4 MHz signal that will change synchronous with the low to high transition of DOT CLK. MEM CLK and CHAR CLK also have transitions on the rising edge of the DOT CLK, (the active times that are decoded for the operation of the machine state or the logic timing synchronizing circuits). CARRY is a 62 nanosecond pulse which is fed to a chain of synchronous counters which are clocked with the 62 NSEC clock so that the pulse must be just that long in order to allow for one advance. The CARRY output occurs only during the F count of counter UB11.

The address decoder UB1 (Schematic, Sheet 2) decodes address lines 13 and 14 and the added conditions to UE5 of ROM MODE* and address 15. ROM MODE* comes from a sequence of flip-flops set by various interrupt conditions, by a power-on-reset, or by a particular I/O reference. If ROM MODE* is active (low), address 15 is low, address 14 is low and address 13 is high indicating the second 8K of memory, then one of the UB1 outputs (Y1) will be active. In the bottom 8K of memory with address 13, 14, and 15 all low, Y0 will be active. This output is ROM CE* (chip enable) which allows ROM access.

MREQ* is ANDed in UE11 with RD* to produce ROM RD*. Note that the MREQ* is delayed from the Z80's MREQ* and is not present when a refresh is occurring; MREQ* is different from the MREQ* normally associated with a Z80. ROM RD* enables the ROM output, therefore, ROM access is developed from Chip Enable (address 13, 14, and 15 low together with ROM MODE* low) and the ROM Output Enable going active when MREQ* goes active (as long as read is present). There is no RAM access under these conditions.

If ROM MODE* is not active, decoder UB1 will not have any output low. If ROM MODE* is active, and address 14 or 13 is high (in the ROM MODE but in the upper 3/4 of memory), Y2 or Y3 will be low or no output will be low. This is the condition for a RAM access.

If either of the inputs on pins 4 or 5 of UD1 is low, pin 6 will be high which will prevent a RAM RQ* from being issued from UA4, pin 3. On the other hand, if UD1 pins 4 and 5 are high, pin 6 will be low permitting a RAM request. On an MREQ* signal, a RAM request will be issued. RAM RQ* is applied to gate UC2, pin 12 where it will pass thru if the Memory Counter UD3 is in a finished condition (low output on pin 11). The Memory Counter will lock up in the finished condition because of the UB3 inversion applied to UD3 pin 3 which causes the counter not to count. The Memory Counter will count up to 8 and stop with QD high. If UD3-QD is locked high, then UD2-8 is high which results in a low output (UD2-10) permitting the RAM RQ* to pass thru UC2. It will be the OR'd thru UC1 and applied as a WAIT* to pin 24 of the Z80. This means that if the Memory Counter (UD3) is in the finished state, the RAM request will immediately result in a WAIT* request to the CPU. In order to leave this state, the other destination of RAM RQ* from UA4-3 must be examined.

The timing conditions at UC1-8 are as follows:
(1) Count 8 to 9 from counter UB11 via decoder UB1
(2) Condition of UB1-10, count C to D
(3) M1* which comes direct from the CPU

The M1 memory timing sequence from the Z80 is shorter than the non-M1 memory sequence. Two possible openings for M1 access are allowed, but only one possible opening is allowed for a non-M1 timing sequence. Therefore, either M1 and C-D or 8-9 are the possible times RAM request will be gated thru UC2-8. The output of UC2-8 is called RAM LOAD* which is applied to UD1-12 where it is ORed with C-D from decoder UB1. This means that even if a RAM request is not issued, C-D time will cause Memory Counter UD3 to load. If the Memory Counter is finished counting, a high active signal is applied to UD1-9. Therefore, if a RAM request is not issued on C-D time, UD1-9 will cause the Memory Counter to load. UD3 inputs A, B and C determine what is loaded into the memory counter. Input D will always load to a zero, and the count will start. Bit C indicates no RAM request and the load came from C-D of decoder UB1. This will cause a video access cycle of RAM. Video timing must allow for video access thru memory and to the Video Latch UA18 (Schematic, Sheet 4). The data must be presented to the latch and to Character Generator UA15 in time for the propagation delay (415 nanoseconds). The video deadline is 450 nanoseconds in advance of the H COUNT* load signal to Shift Register UA14 which is the end of the zero count state on Video Counter UB1. Video access will count from 4 to 8 in the Memory Counter. QC will be high and remain high until the count reaches 8. Active low constitutes CPU access. UD3-13 starts at zero and will remain zero for the first count based on the 125 nanosecond dot clock. It will go high for two counts, then it will go low as CPU time changes low. At that point, QD will go high and a finished condition will exist.

11.5 ROM (Read Only Memory)

The ROM (See Figure 11.5) is a device with information pre-written into its memory slots. The contents of the ROM within the Osborne 1 cannot be changed. There are 12 address inputs and 8 data outputs, which means that the ROM has a capacity of 4 kilobytes.

The ROM has tri-state outputs. When pin 20 (Output Enable*, or OE*) is high, the output pins present a high impedance to the data bus lines, permitting other devices to use the lines when ROM is not in use. When pin 20 is high, no information is put on the data bus by the ROM and the address lines have no effect. When pin 20 is low, the data stored in the ROM is connected to the data bus, and the bus will see whatever is stored in the location being addressed at that time.

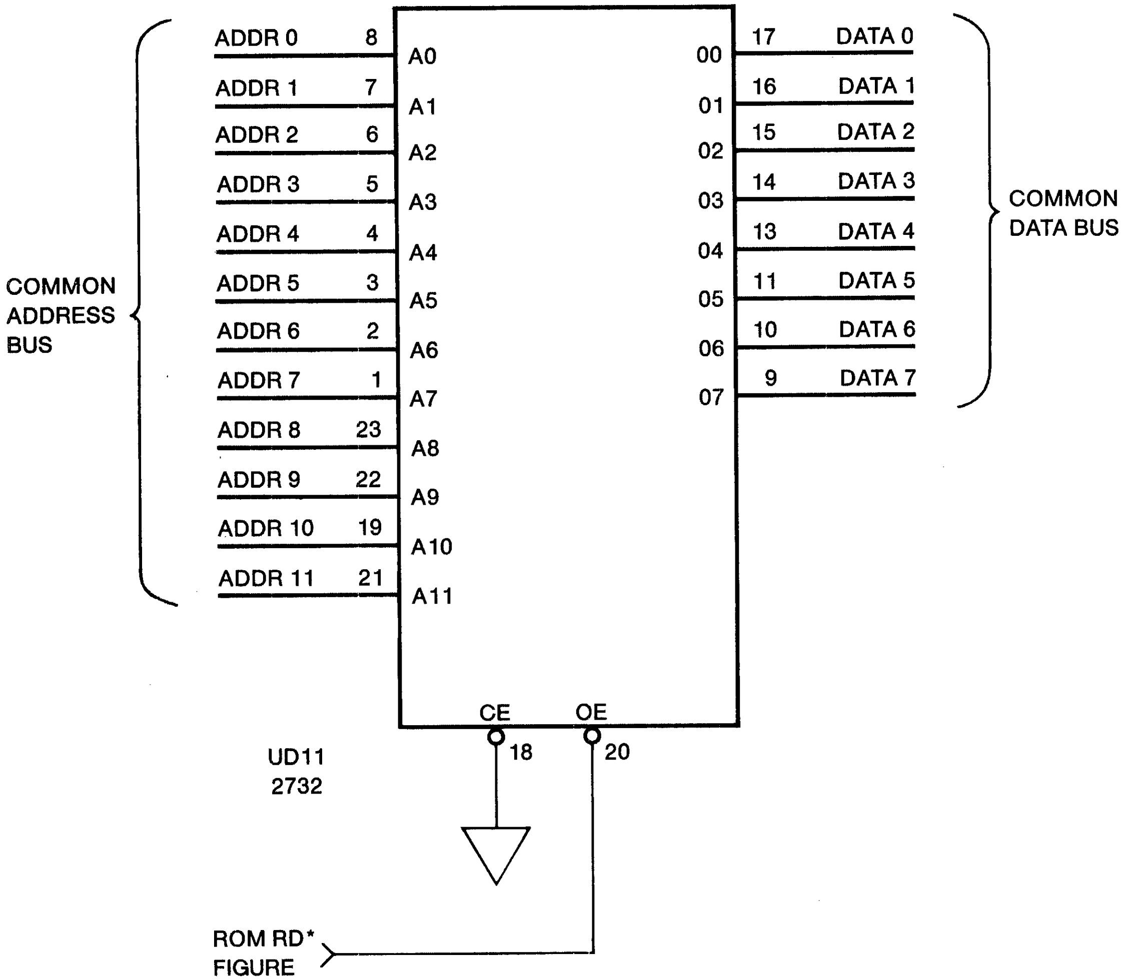


Figure 11.5 — ROM Pinout

Operation of the ROM (Schematic, Sheets 2 and 5) is basically controlled by pin 18 (Chip Enable) and pin 20 (Output Enable). ROM CE* (Chip Enable) and ROM RD* (read) are applied to pins 18 and 20 respectively. ROM CE* and ROM RD* are generated as briefly described below. A detailed discussion is provided under the Timing Function.

ROM CE*: ROM CE* is generated by 2-to-4 line decoder UB1 which is under control of ADR 13, 14, and 15 and ROM MODE*. Address lines 13 thru 15 are used by the CPU to select the ROM chip rather than another device on the bus. The ROM MODE* signal is available to ensure no bus conflict will occur when ROM data is using the bus (UA4 and UB1 act in concert to ensure that RAM and ROM cannot be selected simultaneously.

ROM RD*: ROM RD* (which is active low) is generated by UE11 which allows the ROM (UD11) to access the bus via its tri-state outputs only after causing the RAM to wait.

11.6 RAM (Random Access Memory)

The RAM is a dynamic 64K by 8 bit memory. It is physically structured as a 4×8 matrix (UA20 thru 27, UB20 thru 27, UC20 thru 27, and UD20 thru 27) of integrated circuits (IC), each IC being a 16K by 1 bit RAM. This arrangement results in a 64K by 8 bit memory.

11.6.1 Address Lines

Fourteen address lines are required to specify 1 of 16,348 memory cells. Since the RAM ICs have 7 address inputs, addresses are strobed into the chip seven bits at a time. To read from the RAM, seven row address bits are set up on input addresses A0 through A6, then latched into the chip by a Row Address Strobe (RAS*). Then seven column bits are set up on A0 through A6 and latched in by the Column Address Strobe (CAS*). After both RAS* and CAS* are low, the data output of the cells addressed becomes valid.

11.6.2 RAM Data Output

The outputs are tri-state. Except during a memory cycle by RAS* and CAS*, the outputs present a high impedance to the data lines.

11.6.3 RAM Refresh Operation

It is the nature of this type of memory that the stored data will deteriorate rapidly; therefore, a refresh operation must be performed by applying a RAS* strobe to each row address every 2 milliseconds, or more often. In the Osborne 1 the RAM is refreshed during each video cycle by reading display data from the RAM.

The A0 through A6 pins on all the RAM chips are tied together; that is all pins 5 (A0) are together, all 7 (A1) and so on, so that whatever is on the RAM address bus is applied to all 32 chips. Likewise, all the DI (data in) pins are paralleled. The DO (data out) pins are also tied together.

11.6.4 Write Enable Signal WE*

Write Enable from OR gate UE5 enables data to be put onto the bus. Write Enable is a function of the CPU delayed access input at UE5 pin 2 and the RD* signal. RAM bus access is also controlled by the tri-state line driver UA19, the state of which is controlled by UE5 output pin 11. The output of UE5 is a function of READ*, pin 12, and CPU delayed access, pin 13. The generation and access will be discussed as part of the timing function. Addresses are set on A0 through A6 by the address multiplexers UE15, UE16, UE17, and UE18 and through buffers UE19. The gate inputs to the buffers are tied low, thus the buffer outputs are always enabled. The DO (data out) pins to the RAMs are connected to the data bus through the read buffer UA19. This driver is a tri-state output device gated by UE5.

11.6.5 CPU Write Operations

When RD* and CPU TIME* are low, RAM data outputs are gated out on the data lines. When CPU TIME* is high, the RAM is (probably) in a video cycle and RAM outputs go to the video circuits. The CPU TIME* and RD* signals switch RAM output to the CPU data lines.

To read a memory cell it is necessary to set up address bits A0 through A6 with a row address, strobe with RAS*, set up a column address, strobe with CAS*, and while the RAS* and CAS* strobes are low, data appears at the outputs.

11.6.6 Memory Multiplexing

Memory multiplexing (Schematic, Sheet 4) is done by four ICs, UE15 thru UE18. Pins 14 and 2 (S0 and S1) switch between CPU and video, and between rows and columns, as determined by ACCESS, CPU TIME*, and the 62 NSEC clock. The addressing, reading, writing, and strobing the memory with appropriate delays are a separate function. The paragraphs on timing describe the timing and sequence of these signals.

11.7 CHARACTER GENERATOR

The character generator EPROM is a part of the video section that performs a translation on the character data as it comes out of the RAM, to the data as it is displayed on the screen. The data in RAM is generally stored using the ASCII code (American Standard Code for Information and Interchange) which is a table of 128 bit patterns that relate to typical characters and control characters from the keyboard. The Character Generator EPROM is entirely separate from the Program ROM and performs a different function. The Character Generator EPROM converts the character code in RAM to a pattern of dots for the display.

When a key is pressed on the keyboard, it has to get into the Character Generator EPROM, since that is the path it has to take in order to be displayed in the screen. When it comes from the keyboard, it is read by the processor and temporarily stored in the RAM. The processor reads the RAM to determine the next action. It looks through a table in the EPROM that contains the keyboard decoding table. A part of the table information resides in the RAM so the user may change the coding of the keys. The numeric keys with the control key pressed down result in reference to this table in RAM.

11.8 I/O SERIAL PORT

Divider UC3 (Schematic, Sheet 7) is a divide-by-13 counter that operates from a 2 or 4 microsecond signal that is obtained from the video counter UA13. This works out to a 26 or 52 microsecond clock period. The software registers in UC4 chip can be set to divide by 16 or 64 as necessary. This results in a 1200 baud bit rate or a 300 baud bit rate for the 6850.

11.9 VIDEO DISPLAY

There are 52 microseconds for display across the screen divided into 52 character elements. There are 24 text lines vertically, each line consisting of 10 elements vertically and 8 horizontal dots; the display matrix is therefore 8×10 dots.

The Character Generator outputs an 8-bit element based on the character code from RAM and a line scan number (0–3) from UB15 divider that indicates which line of the character is being displayed. Every time the retrace returns to the beginning and starts a new scan line, the scan counter increments. It starts at 0 and resets to 0 after the count of 9. The resultant line of text, has text address TXADR.

In terms of the actual data coming out of memory, there are 24 text lines, and Counters UD17 and UD18 count from 0 to 23 (24 lines). There are two text line times that are not seen. The 24th and 25th lines occur during the vertical retrace. The visible size of the display is 24 lines. Horizontally, Character Counters UD15 and UD16 keep track of which character is being displayed at a particular location on the screen. The display can be scrolled vertically and horizontally in hardware. This means that the starting point of each of those lines, horizontal and vertical, can be changed.

Registers contain the values of the first horizontal count in memory that will be accessed, and the value of the first vertical count. Any changing if these two reference values will cause the display to appear to move on the screen. This action is implemented as part of a large scale integration (LSI) parallel interface adapter (PIA) that is a 6821 chip. These reference numbers are latched into the PIA from the CPU.

This indicates that there is a horizontal and vertical set of pointers that can be manipulated by manipulating data as presented by the 6821 chip and results in changing the positioning of the display screen relative to the RAM memory.

As previously described, a text line in RAM can go out to a width of 128 characters. The screen displays only 52 characters at any given moment. The portion of the 128 characters to be displayed is determined by the horizontal offset.

11.10 VIDEO SCAN GENERATOR

The video display timing starts with the horizontal timing signal (CHAR CLK) developed by 4-bit Counter UB1-11. CHAR CLK is inverted by UE12 (CHAR CLK*) and is applied to Decoder UB1. UB1 output count of E-F results in H COUNT* which is applied to the Scan Generator input at UA13-1. The input to UA13-1 is divided down to provide 2, 4, 8 and 16 microsecond outputs. The 2 msec and 4 msec outputs are applied to the baud rate generator UC3 which is part of the I/O serial port. The 16 microsecond output is applied to UA13 where it is further divided down to provide two 64 microsecond outputs. One of the 64 microsecond outputs UA13-10 is inverted through UE23D and routed to the external monitor connector. This is the horizontal sync pulse. The other 64 msec output is inverted by UA11 and applied to UA12. The flip-flop formed by two sections of UA12 develops the horizontal blanking pulse.

Flip-flop UD14 develops a one character wide pulse. The Q output (UD14-7) advances Scan Counter UB15, resets Horizontal Counter UA13, and also increments Text Line Counters UD17 and UD18.

The Q* output (UD14-6) reloads Character Counters UD15 and UD16 with the horizontal offset.

Divider UB15 develops the SCAN 0 to 3 signals. The SCAN 0 and SCAN 3 signals are applied to the inputs of AND gate UB13 that increments the text address counters (UB17 and UB18) by one each time scan number nine occurs. The audio alarm is enabled by the PB5 output of parallel interface adapter (PIA) UC15 and is modulated by SCAN 2 (UB15-7).

The QD output from UB15 (SCAN 3) clocks a line counter formed by UB15 and UB14. This line counter develops the vertical blanking and sync signals.

The vertical blanking pulse output from AND gate UB13 is also routed to the external monitor connector through inverter buffer UE23. Vertical blanking is also applied to another section of inverter buffer UE23, mixed with the horizontal blanking output from UE23, and applied to gates UE22. The outputs from the two gates (UE22) are inverted by buffer UE23 and applied to the CRT video via video contrast control R44.

Flip-flop UE21 is used to insert the cursor signal and DIM* level via gate UE22 onto the CRT video line along with the video signal that originates from the video display shift register UA14.

The outputs of counters UD15 and UD16 (CHADR 0 to 6) and UD17 and UD18 (TXADR 0-4) are routed to the RAM address multiplexer circuits.

11.11 KEYBOARD AND ASSOCIATED BUFFERS

Input to the Osborne 1 is made via the keyboard unit. The keyboard is comprised of a series of 69 key switches which connect columns (COL 0 to 7) and rows (ROW 0 to 7).

The keyboard circuitry is an 8×8 matrix with a key switch located at the junction of each pair of lines. The circuitry is enabled when ADR 9 line is set high, I/O SEL* is set low, and RD* line is also set low. This results in enabling UE12 tri-state buffers that transmit the COL 0 through COL 7 levels via the data lines (DATA 0 through DATA 7) to the CPU.

Pressing a key connects the associated ROW and COL lines. The voltage level from the ADR line is transferred to the ROW line via Inverter UE13 or UE14. The keyswitch passes this level to the COL line, and it is then routed through enabled tri-state buffers of UE12 to the data lines to the CPU.

11.12 DISK DRIVE AND DISK CONTROLLER

There are two 5-1/4" Disk Drives provided as an integral part of the Osborne 1. The two units permit recording and playing back information on single sided, single density, soft sectored diskettes and transferring this information from or to RAM.

11.12.1 Disk Drive Controller

The disk drive controller circuit is a 1793 LSI chip that provides all the necessary control of the two disk drive units (1 and 2). The disk drive controller (UB7) is a serial interface that is under software control. The 1793 interfaces with the disk drive electronics via buffer interface gates UA8 thru UA10.

11.12.2 Disk Drive

The Osborne 1 contains two disk drive units that store information on a floppy disk. Each disk has 40 tracks capable of storing 102K bytes for a total of 80 tracks (204K bytes) with two drives. The disks when operating rotate at 300 rpm that results in 200 millisecond period of rotation. The head can move across the 40 tracks taking approximately 20 milliseconds per track step time. The recorded data is retrieved as a serial data stream. This data is examined by the disk controller to decipher which sector is being read.

11.12.3 Disk Drive Electronics

The disk drive electronics board on each drive provides the following functions:

1. Read amplification to amplify signals received from the magnetic read heads
2. Write drive control that is used to drive the recording head
3. Motor control electronics used to maintain the disk drive motor running at a constant speed
4. Stepper motor control used to activate the stepping motor and move the head across the disk to the proper track in response to step commands from the disk controller
5. Track 0 sensors
6. Write protect that detects if a tab has been placed over a notch in the disk envelope prohibiting the ability to write on the disk
7. Index sensing that senses a hole in the disk to provide an index point

11.12.4 Disk Format

The disk track format is such that it provides the writing and reading of magnetic flux densities along a disk track. An electrical pulse, applied to a write head, will result in polarizing the magnetic particles in the disk at a specific spot. During the read process, the spot will cause an electrical read pulse to be developed. These pulses when written, will normally occur at 4 microsecond intervals.

To be assured that the correct pulse train versus noise is read, the pulse is gated with a 4 microsecond clock pulse generated by a counter. In this manner, noise transient effects will be minimized.

The index timing pulse occurs once each rotation of the disk. With a disk present in the drive assembly and the drive door closed, the index sensor optically detects the presence of a disk hole. Only one hole is permitted per disk to identify the beginning of sector 1. Disks that contain additional holes (hard sectored disks) will not operate correctly in the disk drives.

To verify a good recording, the information writing operation must be completed before the second index pulse occurs. A second index pulse will terminate the operation and will cause an error indication to occur. The index pulse received by the 1793 disk drive controller is a significant indicator of operational status.

11.13 RESET AND NMI

Reset and NMI are two signals generated separately that produce similar results.

NMI: Pressing the Reset Switch S1 on the front of the Computer will cause the non-maskable interrupt (NMI) to be generated from the debounce buffer UE20 (Schematic, Sheet 1). The NMI or reset signal causes ROM to be addressed (at location zero) as explained later. When Reset switch S1 is pressed, the following sequence occurs:

1. NMI is applied to the D input of UB6 and to pin 4 of AND gate UA4 (Schematic, Sheet 3).
2. The M1* output of the CPU is inverted by UB12 and clocks flip-flops UB6 and UB4. The Q output of UB6 goes low, and this signal, ANDed with switch signal, applies NMI* to the CPU.

The next M1* pulse switches the Q output of UB4 low, and the pulse after that sets the Q output of UB4 (ROM MODE*) low. The active low Q* output of UB4B prevents further clocking of UB1 and UB4 via OR gate UB4UA4.

UD10B functions as an I/O write cycle decoder. The low active outputs of UD10A perform as follows:

1. Y0 clears flip-flop UB4 and ROM MODE* becomes low active.
2. Y1 presets UB4 and ROM MODE* goes high which enables RAM if MEM REQ* is low.
3. Y2 causes BIT 9 to go high. UE10A serves as a latch so that BIT 9 remains high until Y3 goes low, setting BIT 9 low.

Reset: Before the computer power supply is turned on, Capacitor C12 (Schematic, Sheet 1) is completely discharged. Turning the power supply on results in the Capacitor (C12) being charged to 5V. This charging does not occur instantaneously. The time it takes depends on Resistors R1, R6 and Capacitor C12. While C12 is charging, the input to UB3A is in the low state (between 0 and 2.0 Volts) and this results in the output of UB3B, which is the reset signal, being low. Reset is routed to:

1. Z80 CPU UC11
2. The parallel port interface chip UC7
3. Interface chip UC15 (Figure 5)
4. Clear input of flip-flop UB4

The reset signal resets the parallel port interface and the video interface chip. The effect of the reset signal on the CPU and on flip-flop UB4 is the same as the NMI signal.

Osborne 1 Schematics

12.0 Osborne 1 Schematics

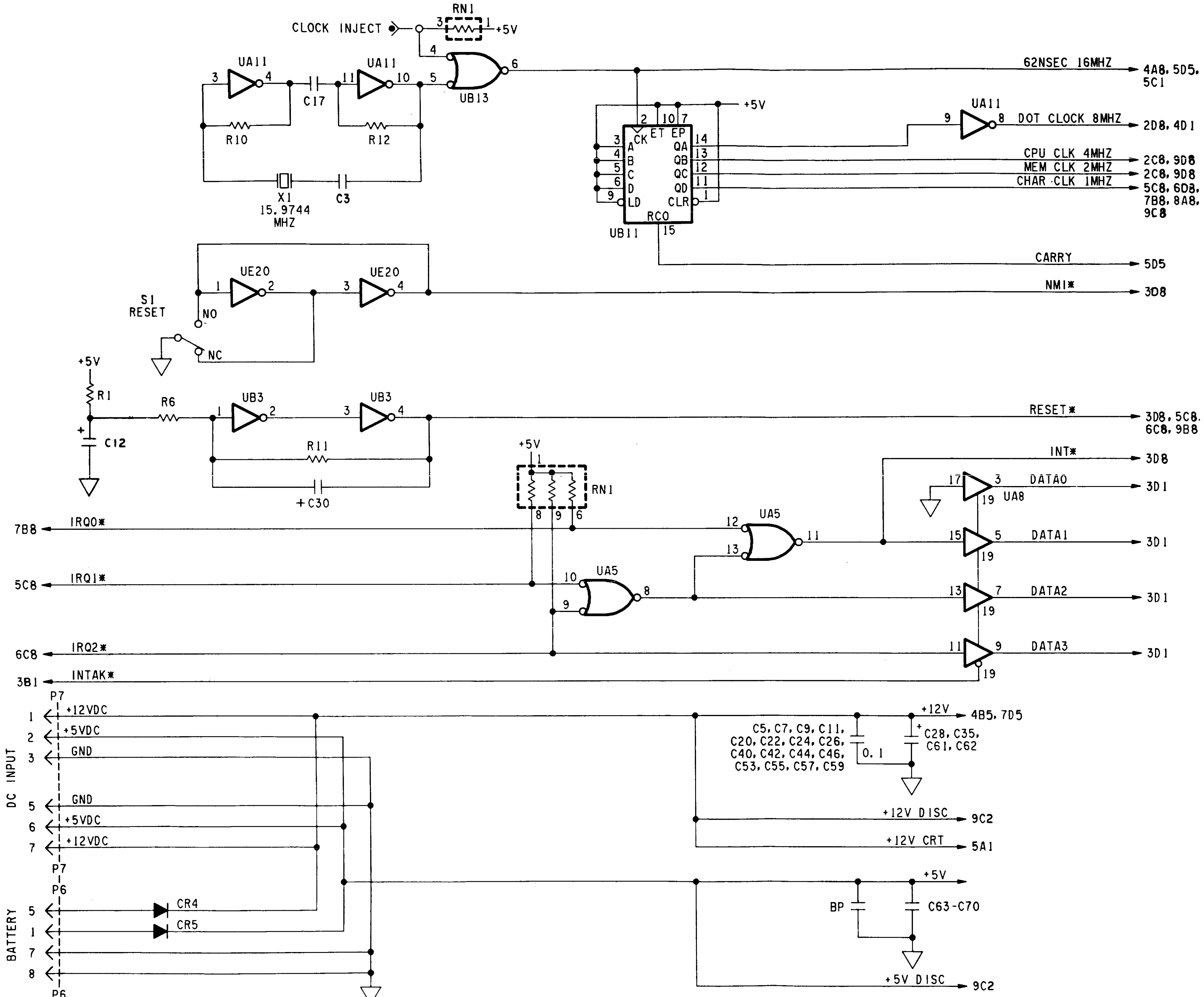
12.1 MAIN LOGIC BOARD SCHEMATICS

Following are the schematics for the main logic printed circuit board:

DWG. NO.	SH	REV
1A10063	1	B

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED
	A	FIRST RELEASE ECO # 0265	6/11/82	
	B	E.C.O. # 0331 A MR	8/18/82	



I.C. IDENTIFICATION TABLE

REF. DESIGNATION	TYPE NO.	GND	+5V	+12V
UA12, UD1, UE10, UE22	LS00	7	14	
UB12, UD2	LS02	7	14	
UD7, UD8, UD9	LS03	7	14	
UE6, UE20	LS04	7	14	
UA11, UB3	S04	7	14	
UE13, UE14	LS05	7	14	
UA9, UA10, UD6, UE23	7406	7	14	
UA5, UB13, UB18, UC1	LS08	7	14	
UB5	LS11	7	14	
UA4, UE5, UE11, UC2	LS32	7	14	
UE7, UE8, UE9	7438	7	14	
UA6, UB4, UB6, UE21	LS74	7	14	
UB2	LS109	8	16	
UB1, UD10	LS139	8	16	
UE15, UE16, UE17, UE18	LS153	8	16	
UB17	LS155	8	16	
UB16	LS157	8	16	
UA7, UB11, UC3, UD3 UD15, UD16, UD17, UD18	LS161	8	16	
UA14	74166	8	16	
UD14	LS175	8	16	
UA8, UA19	LS244	10	20	
UE19	S244	10	20	
UA18	LS273	10	20	
UB15	LS390	8	16	
UA13, UB14	LS393	7	14	
UC11	Z80-A	29	11	
UE1	555	1	8	
UE3	LM1458			
UB7	1793	20	21	40
UA15	2716	12	24	
UD4	LM3900	7	14	
UC7, UC15	6821	1	20	
UC4	6850	1	12	
UE12	81LS95	10	20	
UD11	2732	12	24	
UA20-UA27, UB20-UB27 UC19-UC27, UD20, UD27	4116	16	9	8

REFERENCE DESIGNATION

LAST	NOT USED
C85	C37, C60, C83
CR5	—
P9	P6, 7
Q2	—
R46	—
RN14	R9, 45
S1	—
X2	—

MG0144 BLODGETT 02JUN82

NEXT ASSY	USED ON
2A12054	OCC1

APPLICATION

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±

MATERIAL

FINISH

DO NOT SCALE DRAWING

APPROVALS	DATE
DRAWN	5/82
CHECKED G. Loveland	6-22-82
ISSUED G. Waldron	6/25/82

OSBORNE COMPUTER CORPORATION
26500 Corporate Avenue Hayward, California 94545 415-887-8080

MULTI LAYER MAIN LOGIC PCB SCHEMATIC

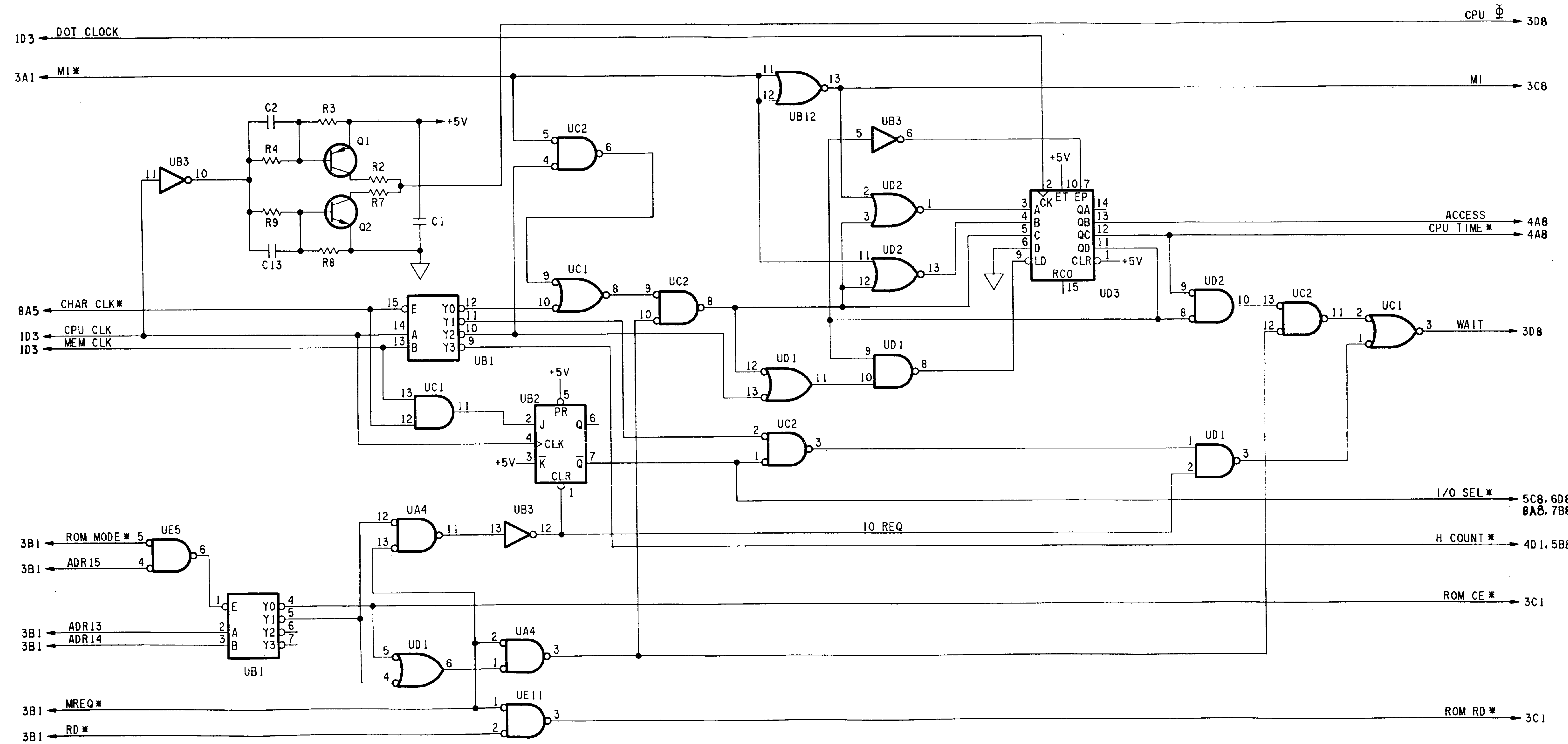
SIZE D | DWG. NO. 1A10063 | REV. B

SCALE NONE | SHEET 1 OF 9

1A1Ø063 SH 2 REV B

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED



MG0145 BLODGETT 02JUN82

NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±

MATERIAL

FINISH

DO NOT SCALE DRAWING

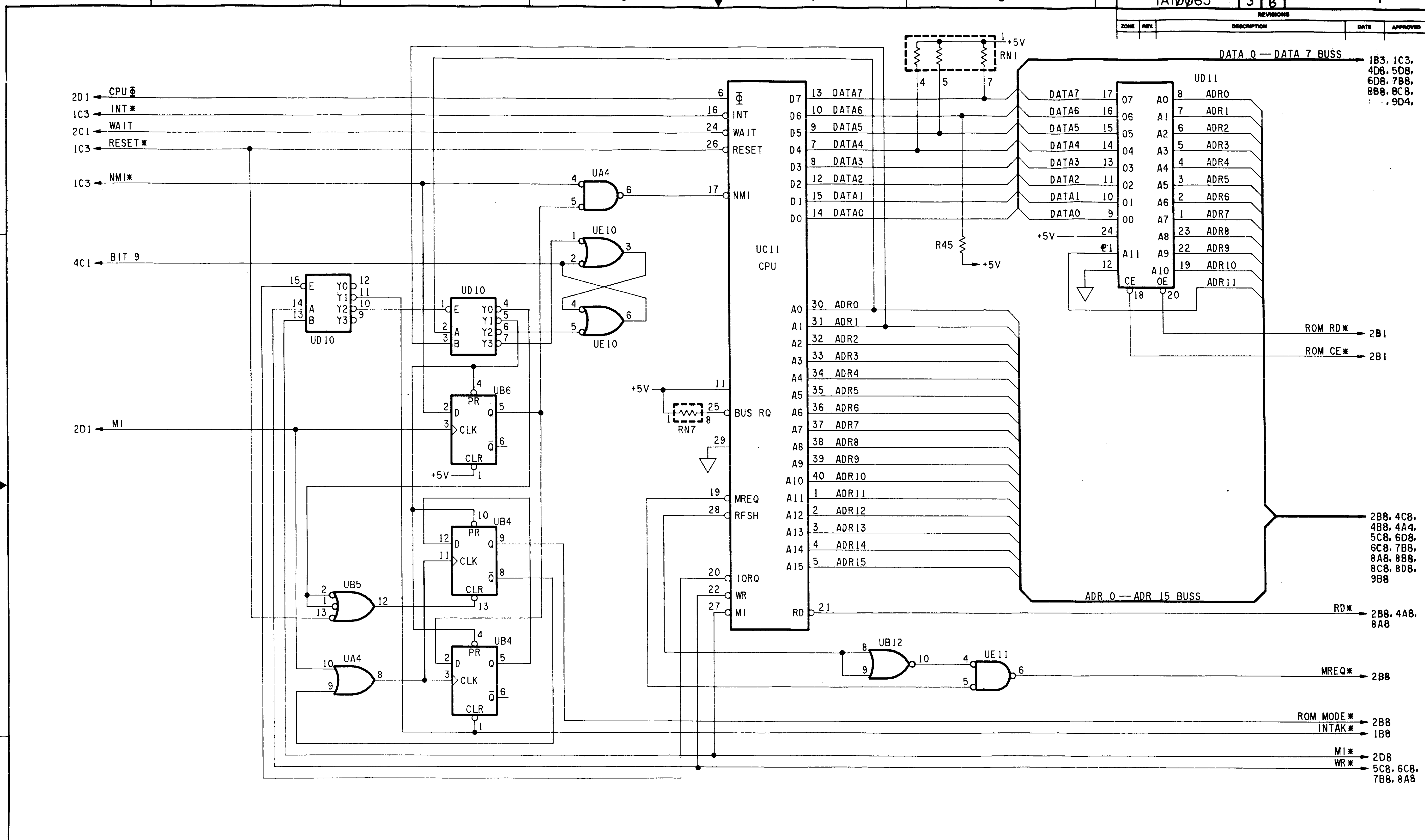
APPROVALS	DATE
DRAWN	5/82
CHECKED	
ISSUED	

OSBORNE COMPUTER CORPORATION
26500 Corporate Avenue
Hayward, California 94545
415-887-8080

MULTI LAYER MAIN LOGIC PCB SCHEMATIC

SIZE D DWG. NO. 1A1Ø063 REV. B

SCALE NONE SHEET 2 OF 9



DWG. NO. 1A1ØØ63 SH. 3 REV. B

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED

MG0146 BLODGETT 02JUN82

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS DECIMALS ANGLES .XX ± .XXX ±

MATERIAL

FINISH

NEXT ASSY | USED ON

APPLICATION

DO NOT SCALE DRAWING

APPROVALS | DATE

DRAWN | 5/82

CHECKED

ISSUED

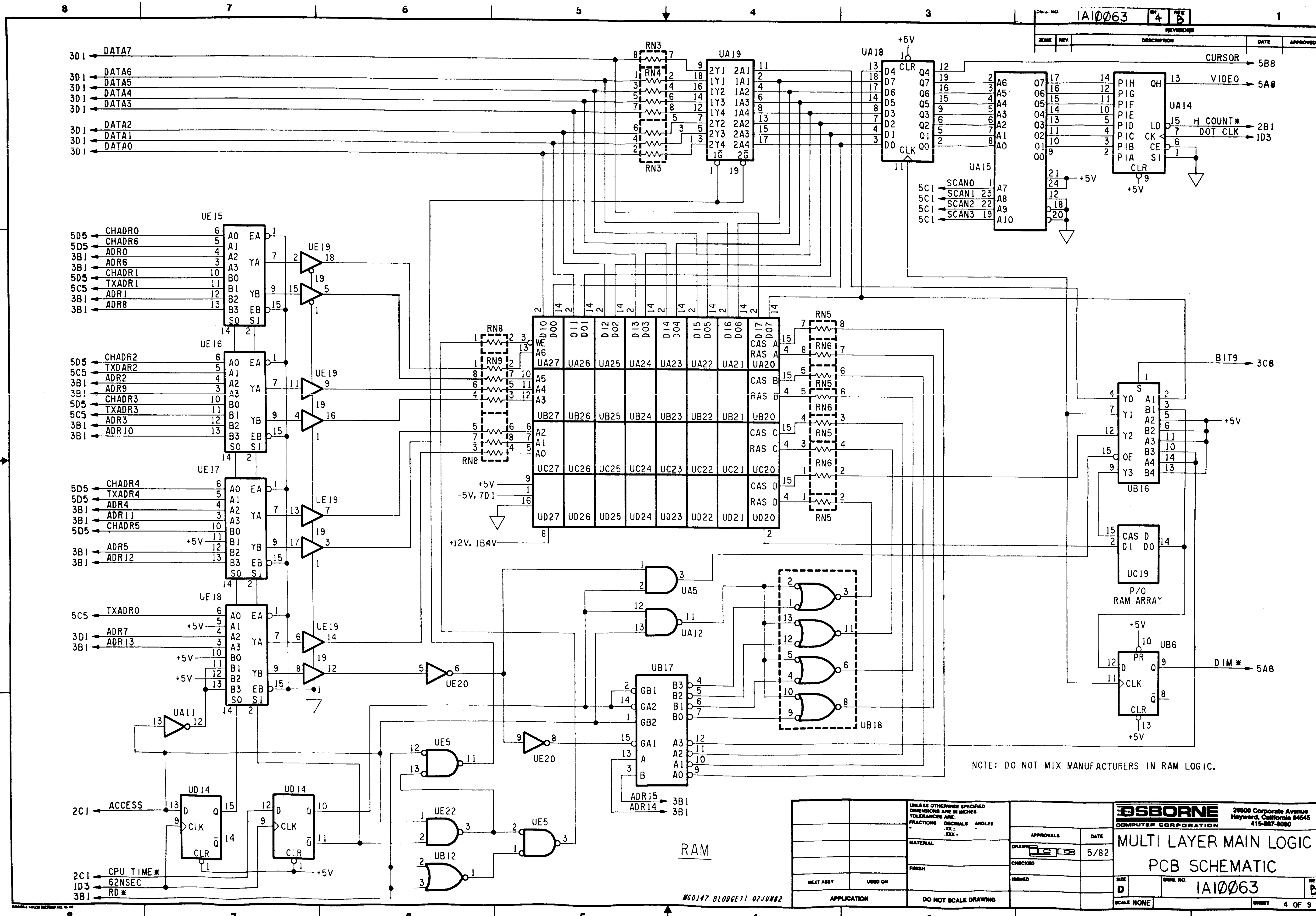
OSBORNE COMPUTER CORPORATION

26500 Corporate Avenue Hayward, California 94545 415-887-8080

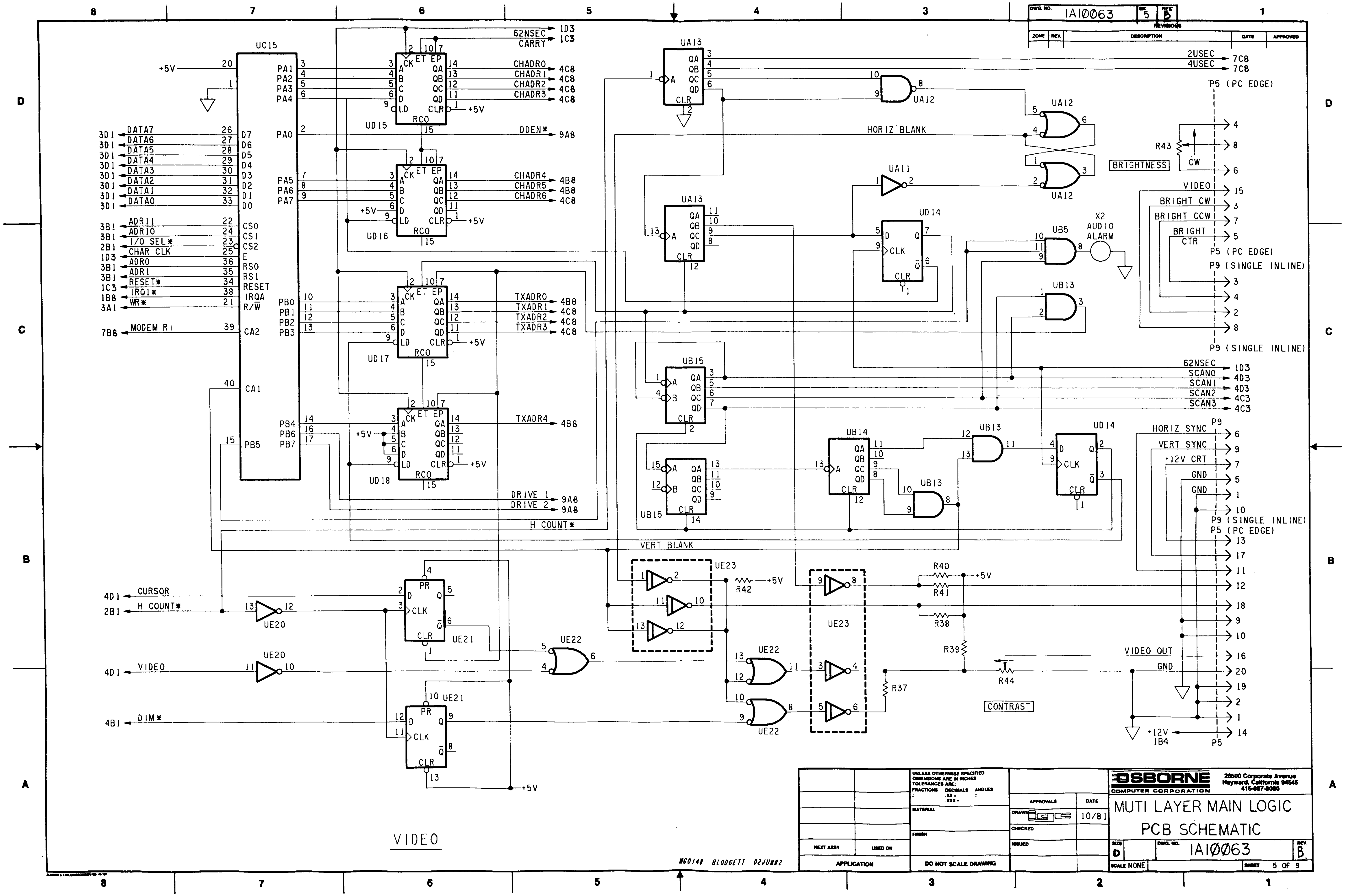
MULTI LAYER MAIN LOGIC PCB SCHEMATIC

SIZE D | DWG. NO. 1A1ØØ63 | REV. B

SCALE NONE | SHEET 3 OF 9



IA10063
REVISIONS
ZONE
REV.
DESCRIPTION
DATE
APPROVED
DATA7
DATA6
DATA5
DATA4
DATA3
DATA2
DATA1
DATA0
RN3
RN4
UA19
UA18
+5V
CLR
CLK
UA15
UA14
CURSOR
VIDEO
H COUNT*
DOT CLK
SCAN0
SCAN1
SCAN2
SCAN3
UE15
UE16
UE17
UE18
UE19
CHADR0
CHADR6
ADR0
ADR6
CHADR1
TXADR1
ADR1
ADR8
CHADR2
TXDAR2
ADR2
ADR9
CHADR3
TXADR3
ADR3
ADR10
CHADR4
TXADR4
ADR4
ADR11
CHADR5
ADR5
ADR12
TXADR0
ADR7
ADR13
RN5
RN6
RN8
RN9
BIT9
UB16
UC19
P/O
RAM ARRAY
UB6
DIM*
UA5
UA12
UB17
UB18
UE20
UE5
UE22
UB12
UA11
UD14
ACCESS
CPU TIME*
62NSEC
RD*
ADR15
ADR14
+12V, 1B4V
-5V, 7D1
NOTE: DO NOT MIX MANUFACTURERS IN RAM LOGIC.
RAM
MG0147 BLODGETT 02JUN82
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE:
FRACTIONS
DECIMALS
ANGLES
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
APPROVALS
DATE
DRAWN
5/82
CHECKED
ISSUED
OSBORNE
COMPUTER CORPORATION
26500 Corporate Avenue
Hayward, California 94545
415-887-8080
MULTI LAYER MAIN LOGIC
PCB SCHEMATIC
SIZE
D
DWG. NO.
IA10063
REV.
B
SCALE NONE
SHEET
4 OF 9



VIDEO

MGO148 BLODGETT 02JUN82

DWG. NO.	1A10063
SH	5
REV	B

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE:

APPLICATION

DO NOT SCALE DRAWING

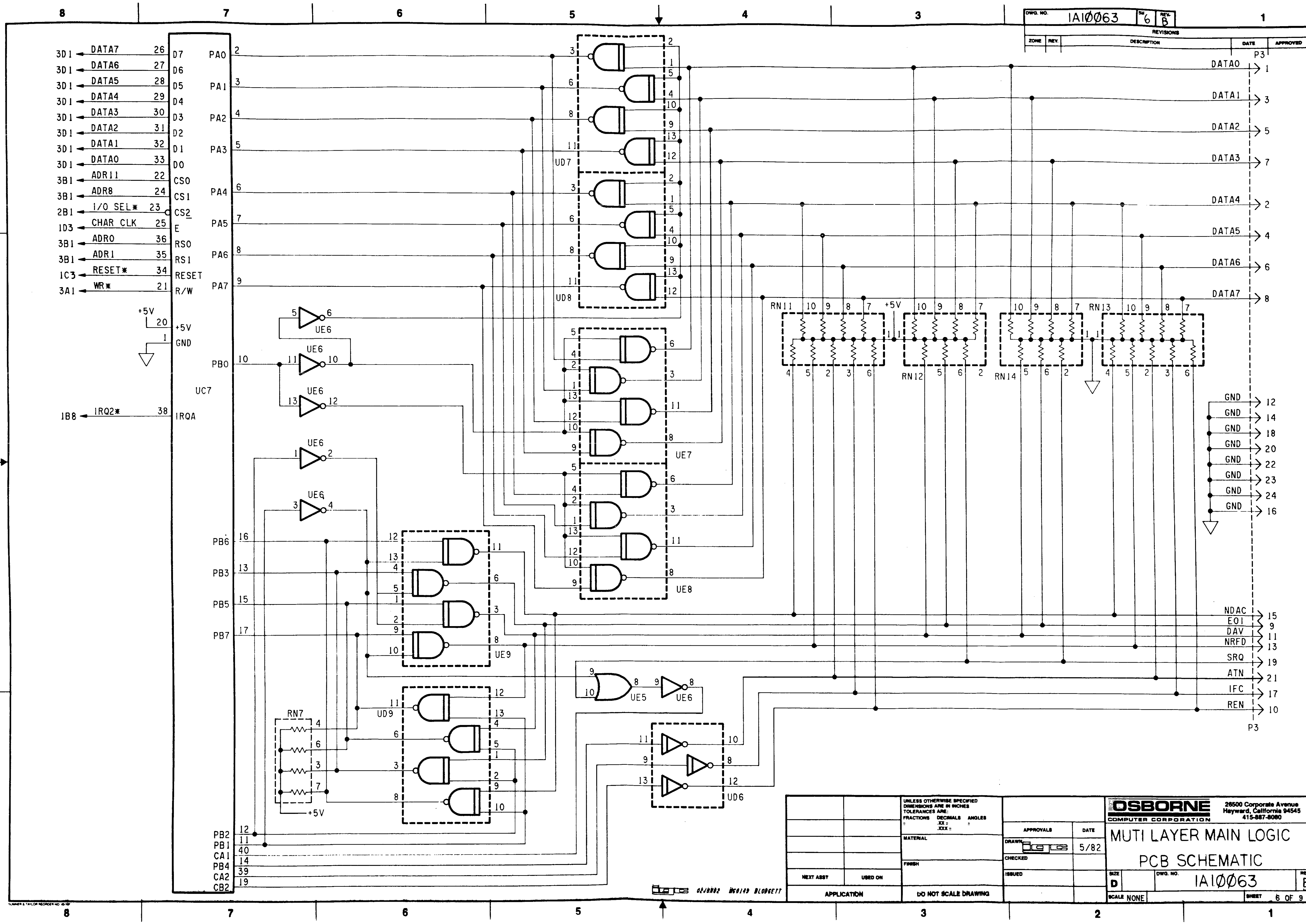
DRAWN 10/81

OSBORNE COMPUTER CORPORATION
26500 Corporate Avenue
Hayward, California 94545
415-887-8080

MUTI LAYER MAIN LOGIC PCB SCHEMATIC

SIZE D DWG. NO. 1A10063 REV. B

SCALE NONE SHEET 5 OF 9



UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS DECIMALS ANGLES .XX = .XXX =

MATERIAL

FINISH

NEXT ASSY | USED ON

APPLICATION

DO NOT SCALE DRAWING

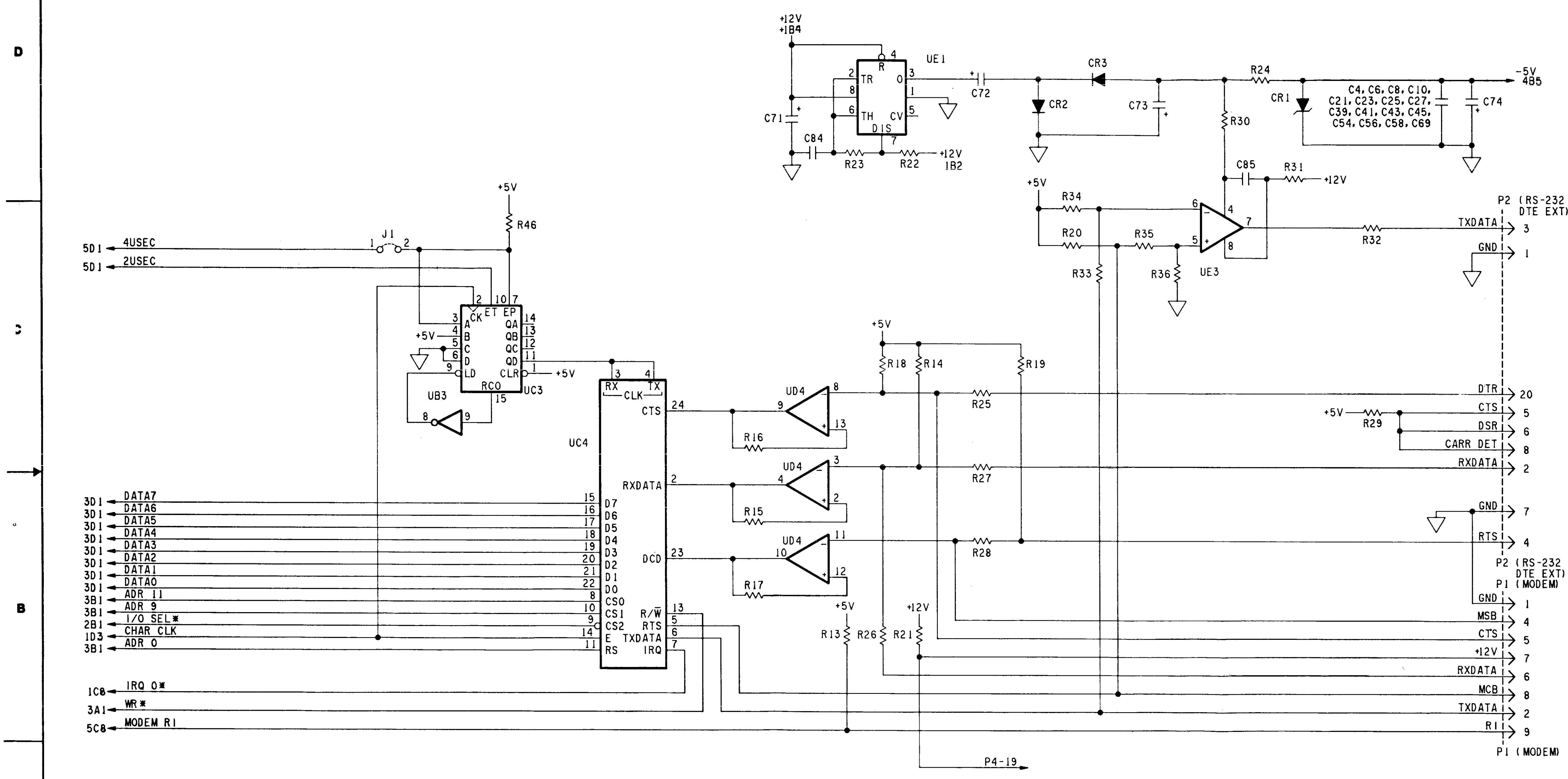
APPROVALS | DATE
DRAWN | 5/82
CHECKED
ISSUED

OSBORNE COMPUTER CORPORATION
26500 Corporate Avenue Hayward, California 94545 415-887-8080

MUTI LAYER MAIN LOGIC PCB SCHEMATIC

SIZE D | DWG. NO. 1A10063 | REV. B

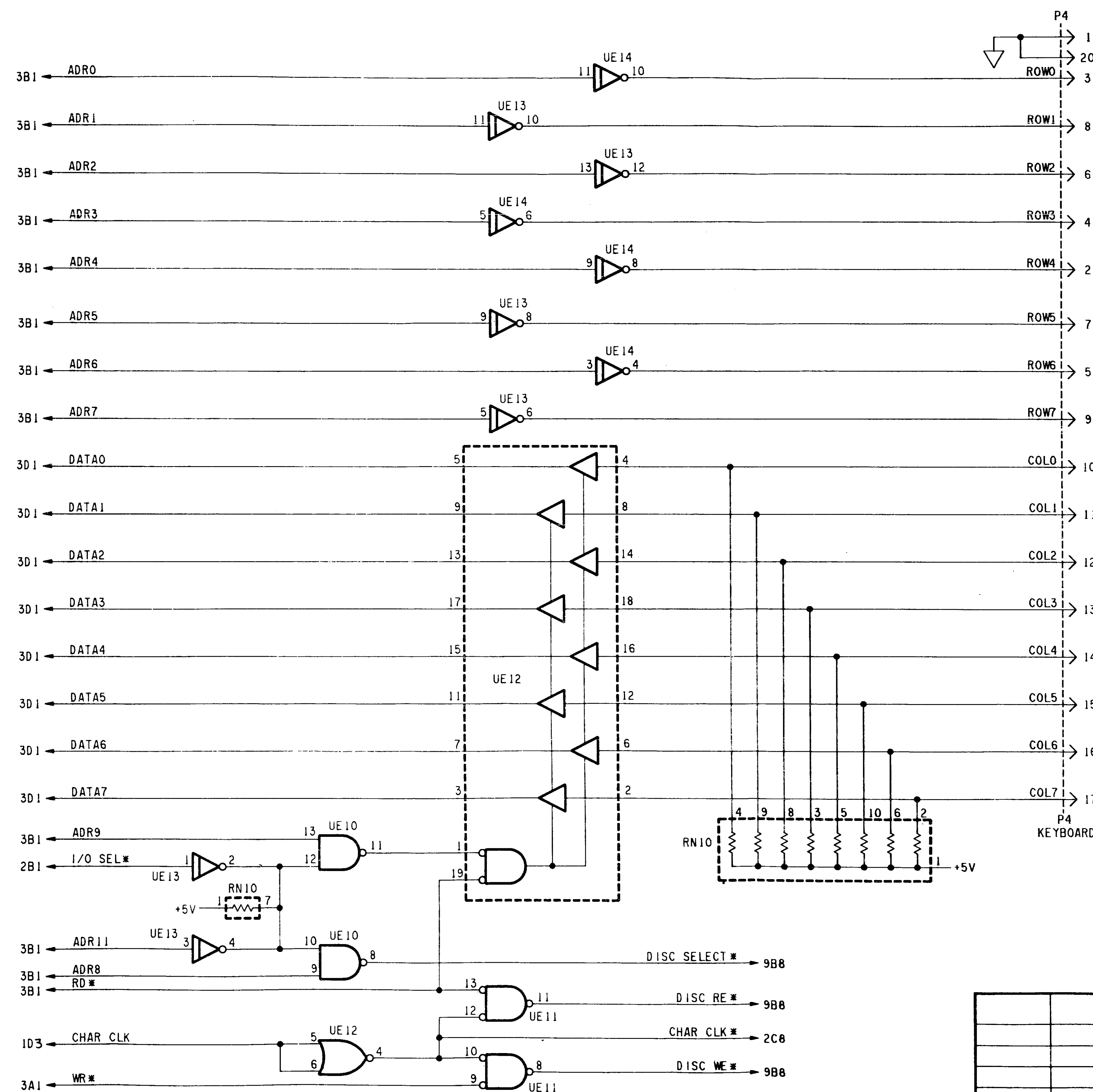
SCALE NONE | SHEET 6 OF 9



SERIAL I/O

M60150 BLODGETT 02JUN82

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±	APPROVALS	DATE	OSBORNE COMPUTER CORPORATION 26500 Corporate Avenue Hayward, California 94545 415-887-8080	
MATERIAL	DRAWN	10/81	MULTI LAYER MAIN LOGIC PCB SCHEMATIC	
FINISH	CHECKED		SIZE D	DWG. NO. 1A10063 REV. B
NEXT ASSY / USED ON / APPLICATION — DO NOT SCALE DRAWING	ISSUED		SCALE NONE	SHEET 7 OF 9



DWG. NO.	SH	REV.
1A1Ø063	8	B

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±

MATERIAL

FINISH

NEXT ASSY | USED ON

APPLICATION

DO NOT SCALE DRAWING

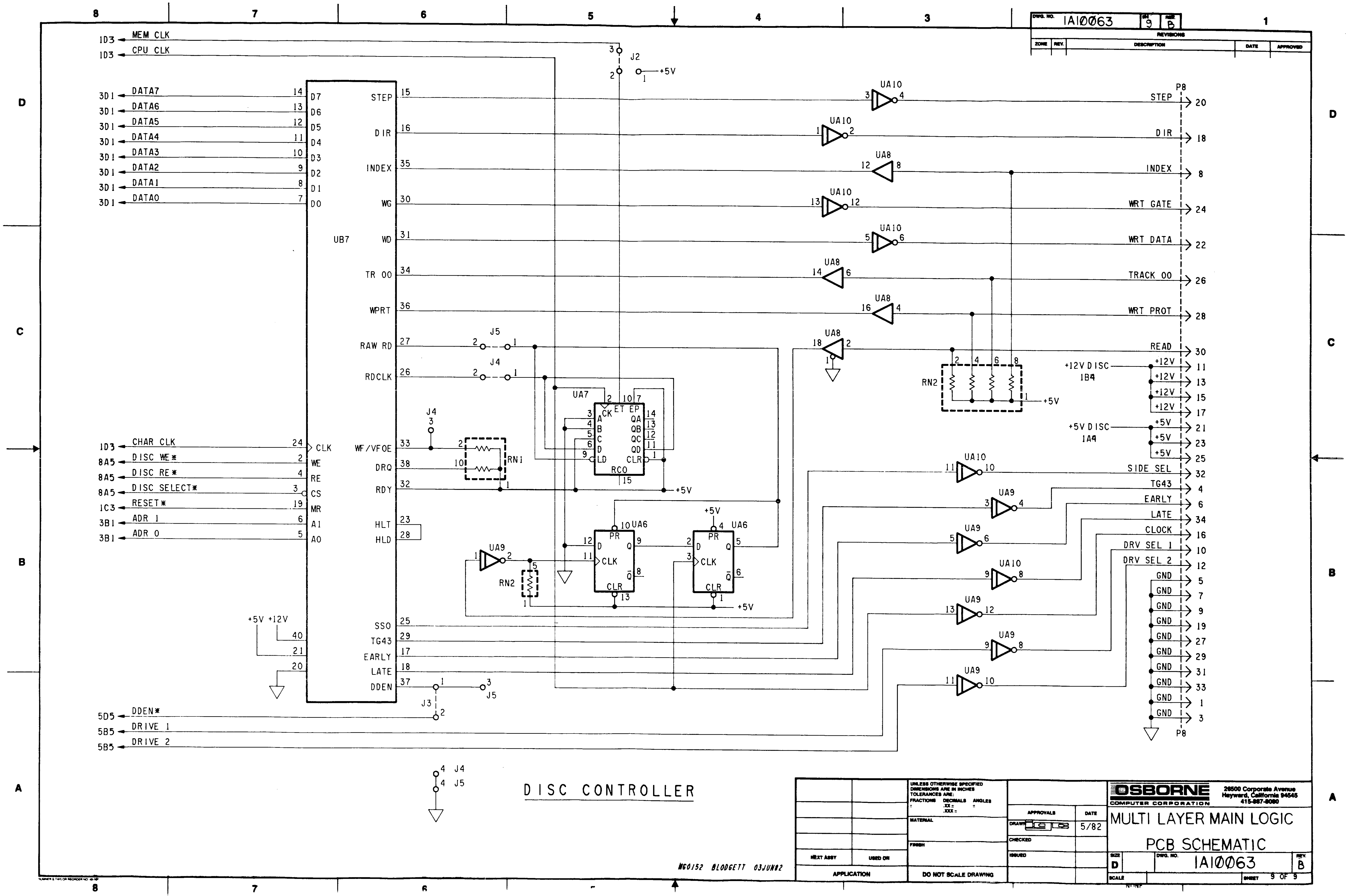
APPROVALS	DATE
DRAWN	5/82
CHECKED	
ISSUED	

OSBORNE COMPUTER CORPORATION
26500 Corporate Avenue Hayward, California 94545 415-887-8080

MULTI LAYER MAIN LOGIC PCB SCHEMATIC

SIZE D DWG. NO. 1A1Ø063 REV. B

SCALE NONE SHEET 8 OF 9



DWG. NO. 1A10063 SH 9 REV B
REVISIONS
ZONE REV. DESCRIPTION DATE APPROVED
MEM CLK
CPU CLK
DATA7
DATA6
DATA5
DATA4
DATA3
DATA2
DATA1
DATA0
UB7
STEP
DIR
INDEX
WG
WD
TR 00
WPRT
RAW RD
RDCLK
WF/VFOE
DRQ
RDY
HLT
HLD
SSO
TG43
EARLY
LATE
DDEN
CLK
WE
RE
CS
MR
A1
A0
CHAR CLK
DISC WE*
DISC RE*
DISC SELECT*
RESET*
ADR 1
ADR 0
DDEN*
DRIVE 1
DRIVE 2
J2
J3
J4
J5
UA6
UA7
UA8
UA9
UA10
RN1
RN2
P8
STEP
DIR
INDEX
WRT GATE
WRT DATA
TRACK 00
WRT PROT
READ
+12V DISC 1B4
+5V DISC 1A4
SIDE SEL
TG43
EARLY
LATE
CLOCK
DRV SEL 1
DRV SEL 2
GND
DISC CONTROLLER
MG0152 BLODGETT 03JUN82
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE:
FRACTIONS DECIMALS ANGLES
MATERIAL
FINISH
NEXT ASSY USED ON
APPLICATION
DO NOT SCALE DRAWING
APPROVALS DATE
DRAWN 5/82
CHECKED
ISSUED
OSBORNE COMPUTER CORPORATION
26500 Corporate Avenue Hayward, California 94545 415-887-8080
MULTI LAYER MAIN LOGIC
PCB SCHEMATIC
SIZE D
DWG. NO. 1A10063
REV. B
SCALE
SHEET 9 OF 9

OSBORNE COMPUTER CORP.	TITLE: D MULTLYR MAINLOGIC PCB ASSY	LM# 3A10063-06
Drawn: KDK / Checked M. [signature]	DATE 9/8/82	Revision H
Appvd: [signature]	ECO#: 0401 / Date:	Released: RELEASED
DETACHED LIST OF MATERIALS		

ITEM	PART NO.	QTY	UM	TITLE	REFERENCE DESIGNATORS
000	1A10063-00	A/R	EA	SCHEM, MULTI LAYER MAIN LOGIC PCB	
001	3P10063-00	001	EA	FAB, MULTI LAYER MAIN LOGIC PCB	
002	3A10256-00	001	EA	D CHAR GEN PRGMD EPROM ASSY	UA15
003	3A10206-00	001	EA	D SYSTEM PRGMD EPROM ASSY	UD11
004	7P11000-00	004	EA	74LS00	UA12, UD1, UE10, UE22
005	7P11002-00	002	EA	74LS02	UB12, UD2
006	7P11003-00	003	EA	74LS03	UD7, UD8, UD9
007	7P11004-00	002	EA	74LS04	UE6, UE20
008	7P11005-00	002	EA	74LS05	UE13, UE14
009	7P11008-00	004	EA	74LS08	UA5, UB13, UB18, UC1
010	7P11010-00	001	EA	74LS11	UB5
011	7P11032-00	004	EA	74LS32	UA4, UC2, UE5, UE11
012	7P11074-00	004	EA	74LS74	UA6, UB4, UB6, UE21
013	7P11109-00	001	EA	74LS109	UB2
014	7P11139-00	002	EA	74LS139	UB1, UD10
015	7P11153-00	004	EA	74LS153	UE15, UE16, UE17, UE18
016	7P11155-00	001	EA	74LS155	UB17
017	7P11157-00	001	EA	74LS157	UB16
018	7P11161-00	008	EA	74LS161	UA7, UB11, UC3, UD3, UD15
					UD16, UD17, UD18
				Alternate P/N 7P11163	
019	7P11166-00	001	EA	74LS166	UA14
020	7P11175-00	001	EA	74LS175	UD14
021	7P11244-00	002	EA	74LS244	UA8, UA19
022	7P11273-00	001	EA	74LS273	UA18
023	7P11390-00	001	EA	74LS390	UB15
024	7P11393-00	002	EA	74LS393	UA13, UB14
025	7P11895-00	001	EA	81LS95	UE12
026	7P11897-00	002	EA	74S04	UA11, UB3
027	7P11898-00	001	EA	74S244	UE19
028	7P11899-00	004	EA	7406	UA9, UA10, UD6, UE23
029	7P11800-00	003	EA	7438	UE7, UE8, UE9
030	7P11808-00	002	EA	6821 PARALLEL I/O	UC7, UC15
031	7P11809-00	001	EA	6850 SERIAL I/O	UC4
032	7P11811-00	001	EA	1793 DISK CONTROLLER	UB7
033	7P11813-00	001	EA	Z80A MICRO PROCESSOR	UC11
034	7P11814-00	033	EA	4116 16K MOS RAM	UA20-27, UB20-27, UC19-
					27, UD20-27

LM CONTINUED

OSBORNE COMPUTER CORP.	TITLE: D MULTLYR MAINLOGIC PCB ASSY	LM# 3A10063-06
	DATE 9/8/82	Revision H
Drawn:KDK Checked M. Bushy	Appvd: [signature] ECO#: 0401 Date:	Released: RELEASED

DETACHED LIST OF MATERIALS

ITEM	PART NO.	QTY	UM	TITLE	REFERENCE DESIGNATORS
035	7P11817-00	001	EA	LM 1458 LINEAR,OP AMP	UE3
036	7P11818-00	001	EA	LM 3900 LINEAR,OP AMP	UD4
037	7P11820-00	001	EA	NE555V LINEAR,TIMER	UE1
040	7P30001-00	002	EA	1N4001 DIODE,1A,SILICON	CR2,CR3
041	7P30002-00	001	EA	1N5231B DIODE,ZENER 5.1V,10%	CR1
042	7P30003-00	002	EA	MR501 DIODE,RECTIFIER 3A	CR4,CR5
045	7P30006-00	001	EA	2N3906 TRANSISTOR,PNP	Q1
046	7P30007-00	001	EA	2N3904 TRANSISTOR,NPN	Q2
050	7P40025-00	001	EA	XTAL 15.9744MHZ +-.005%HC18U	X1
051	7P40026-00	001	EA	ALARM AUDIO,PIEZO PC MOUNT	X2
052	7P45501-00	005	EA	CAP,DIPPED,TANT,1.0 UF 20V	C28,C35,C61,C62,C74
			Alternate P/N 7P45505-00		
053	7P45502-00	003	EA	CAP,DIPPED,TANT,15UF,15V	C71,C72,C73
054	7P45604-00	001	EA	CAP,ELECT.,ALUM 22UF,16V	C12
055	7P46103-00	001	EA	CAP,AXL,CER,.001uf +-10%,50V	C17
056	7P46104-00	028	EA	CAP,AXL,CER,.01UF,+-30%,25V	C1,C14,C15,C16,C18,C19
					C29,C31,C32,C33,C34,
					C36,C38,C47,C48,C49,
					C50,C51,C52,C75,C76,
					C77,C78,C79,C80,C81,
					C82,C84
057	7P46105-00	042	EA	CAP,AXL,CER,0.1UF,+-50%,25V	C4-11,C20-27,C39-46,
					C53-60,C63-70,C85,C86
058	7P46330-00	001	EA	CAP,AXL,CER,33PF +-5%,50V	C3
059	7P46680-00	001	EA	CAP,AXL,CER, 68PF,+-05%,50V	C2
060	7P46188-00	001	EA	CAP,AXL,CER,180pf,+-10%,50V	C13
061	7P46106-00	001	EA	CAP,22UF+-20% 12V ALUM EL AXL	C30
062	7P50006-00	001	EA	RES,VAR,PC MT,100K,30%,1/4W	R43
063	7P50007-00	001	EA	RES,VAR,PC MT,500 OHM,30%,1/4W	R44
064	7P51001-00	001	EA	RES,NWK,150 OHM SIP 8/P P/U	RN2
065	7P51004-00	001	EA	RES,NWK, 3.3K SIP 8 PIN P/U	RN7
067	7P51007-00	001	EA	RES,NWK, 10K,SIP 10 PIN P/U	RN1
068	7P51008-00	003	EA	RES,NWK, 3.3K,SIP 10 PIN P/U	RN10,RN11,RN12
069	7P51009-00	002	EA	RES,NWK, 6.8K,SIP 10 PIN P/U	RN13,RN14
070	7P51011-00	004	EA	RES,NWK,33 OHM SIP 8 PIN ISO	RN5,RN6,RN8,RN9
071	7P51014-00	002	EA	RES,NWK,82 OHM SIP 8 PIN ISO	RN3,RN4
072	7P52101-00	003	EA	RES,FXD,1/4W 5% 100 OHM	R3,R30,R31
073	7P52102-00	005	EA	RES,FXD,1/4W 5% 1.0K	R32,R35,R37,R38,R39
074	7P52103-00	007	EA	RES,FXD,1/4W 5% 10K	R13,R20,R26,R34,R36,

LM CONTINUED

OSBORNE COMPUTER CORP.	TITLE: D MULTLYR MAINLOGIC PCB ASSY	LM# 3A10063-06
	DATE 9/8/82	Revision H
Drawn: KDK Checked M Bailey	Appvd: D Lyman ECO#: 0401 Date:	Released: RELEASED
DETACHED LIST OF MATERIALS		

ITEM	PART NO.	QTY	UM	TITLE	REFERENCE DESIGNATORS
075	7P52104-00	005	EA	RES, FXD, 1/4W 5% 100K	R14, R15, R16, R17, R18
076	7P52181-00	001	EA	RES, FXD, 1/4W 5% 180 OHM	R24
077	7P52220-00	002	EA	RES, FXD, 1/4W 5% 22 OHM	R2, R21
078	7P52221-00	002	EA	RES, FXD, 1/4W 5% 220 OHM	R8, R40
079	7P52222-00	003	EA	RES, FXD, 1/4W 5% 2.2K	R22, R23, R33
080	7P52223-00	001	EA	RES, FXD, 1/4W 5% 22K	R1
081	7P52330-00	001	EA	RES, FXD, 1/4W 5% 33 OHM	R7
082	7P52331-00	003	EA	RES, FXD, 1/4W 5% 330 OHM	R9, R10, R12
083	7P52332-00	002	EA	RES, FXD, 1/4W 5% 3.3K	R29, R42
084	7P52470-00	001	EA	RES, FXD, 1/4W 5% 47 OHM	R41
085	7P52471-00	002	EA	RES, FXD, 1/4W 5% 470 OHM	R6, R4
086	7P52472-00	001	EA	RES, FXD, 1/4W 5% 4.7K	R19
087	7P52473-00	004	EA	RES, FXD, 1/4W 5% 47K	R11, R25, R27, R28
088	7P61106-00	001	EA	SWITCH, DPDT, MOMENTARY, ACTION	S1
089	7P61103-00	001	EA	SWITCH BUTTON	(S1)
090	7P63000-00	004	EA	CONN, SHUNT, 2 PIN	(J1), (J3), (J4), (J5)
091	7P64125-00	001	EA	CONN, PC MT, RT ANGL, DB 25S	P2
092	7P64209-00	002	EA	CONN, PC MT, RT ANGL, DE 9P	P1, P6
093	7P64102-00	002	EA	HDR, PC MT, STR, 100ML SIL 2 PIN	J1, J3
094	7P64410-00	001	EA	HDR, PC MT, STR, .100 SIL 10 PIN	P9
095	7P64207-00	001	EA	HDR, PC MT, STR, .156 SIL 7 PIN	P7
096	7P64408-00	001	EA	HDR, PC MT, STR, DIL 8 PIN	J4, J5
097	7P64520-00	001	EA	HDR, PC MT, RT ANGL DIL 20 PIN	P4
098	7P64534-00	001	EA	HDR, PC MT, RT ANGL, DIL 34 PIN	P8
099	7P65124-00	002	EA	SOCKET LOW PROFILE, DIP 24 PIN	PA15, PD11
100	7P65140-00	002	EA	SOCKET LOW PROFILE, DIP 40 PIN	PB7, PC11

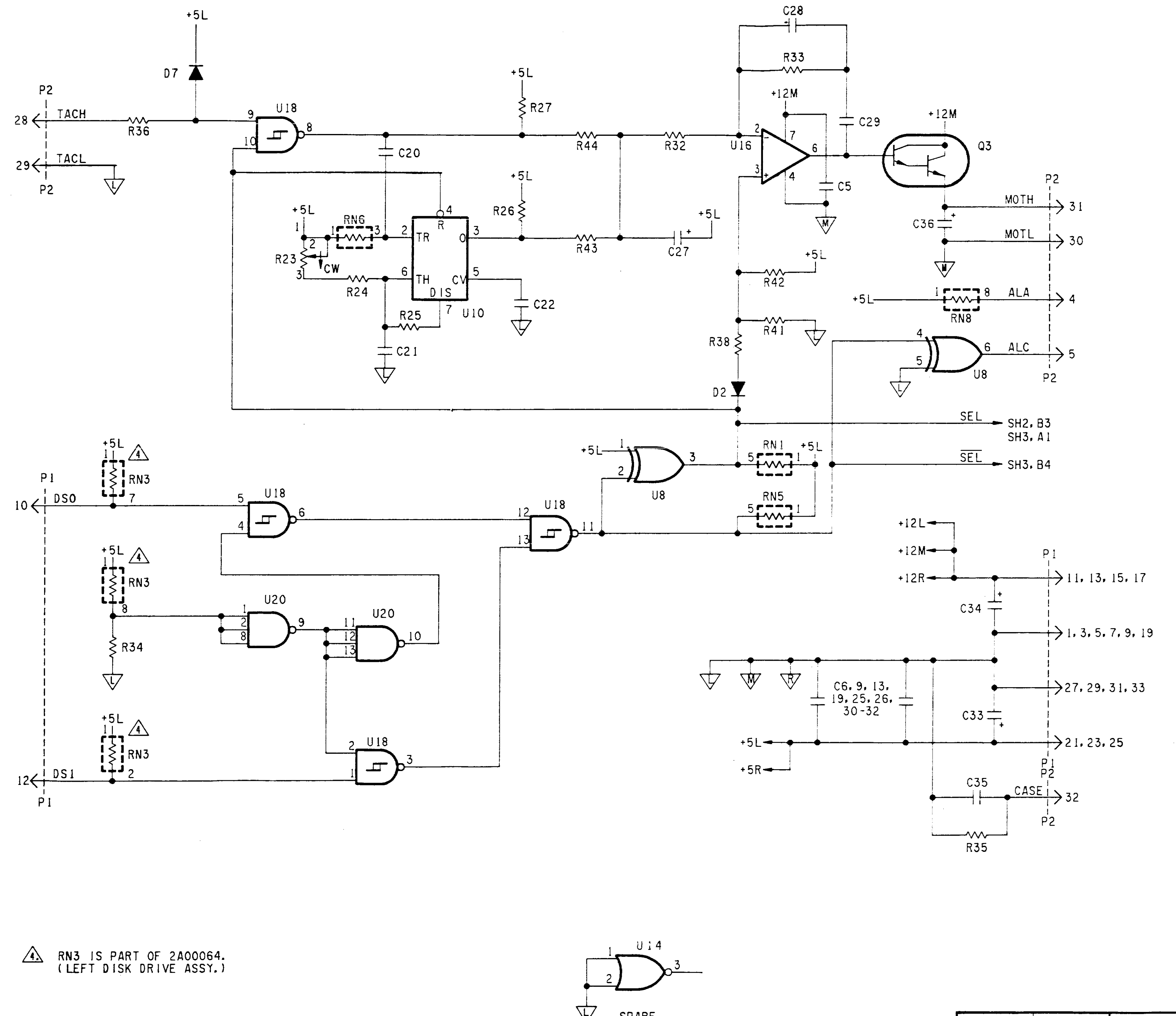
END OF LM

12.2 DISK ELECTRONICS BOARD SCHEMATICS

DWG. NO.	SH	REV.
1A03004	1	J

REVISIONS

	ZONE	REV.	DESCRIPTION	DATE	APPROVED
RELEASED		H	ECO #0090	2/82	PLN
RELEASED		J	ECO #0207	4/82	



4 RN3 IS PART OF 2A00064. (LEFT DISK DRIVE ASSY.)

INTEGRATED CIRCUIT IDENTIFICATION TABLE									
REFERENCE DESIGNATION	TYPE	GND L	GND M	GND R	+5L	+5M	+12L	+12M	+12R
U1	CA3054			5					
U2	4025B	7			14				
U3, 4, 10	555	1			8				
U5	311	1, 4					8		
U6	NE592			5					10
U7	4013B	7			14				
U8	LS86	7			14				
U9	LS112	8			16				
U11	LS123	8			16				
U12	7438	7			14				
U13	4070B	7			14				
U14	4001B	7			14				
U15	LS139	8			16				
U16	TL091		4					7	
U17, 19	LS14	7			14				
U18	LS132	7			14				
U20	4023B	7			14				
U21, 22	75451		4			8			

COMPONENT REF DESIG	
LAST	NOT USED
C38	C17
D6	-
L5	-
P3	-
Q3	-
R39	R23, 24, 40
RN8	RN2
TP8	TP2
U22	-

MG0141 BLODGETT 02JUN82

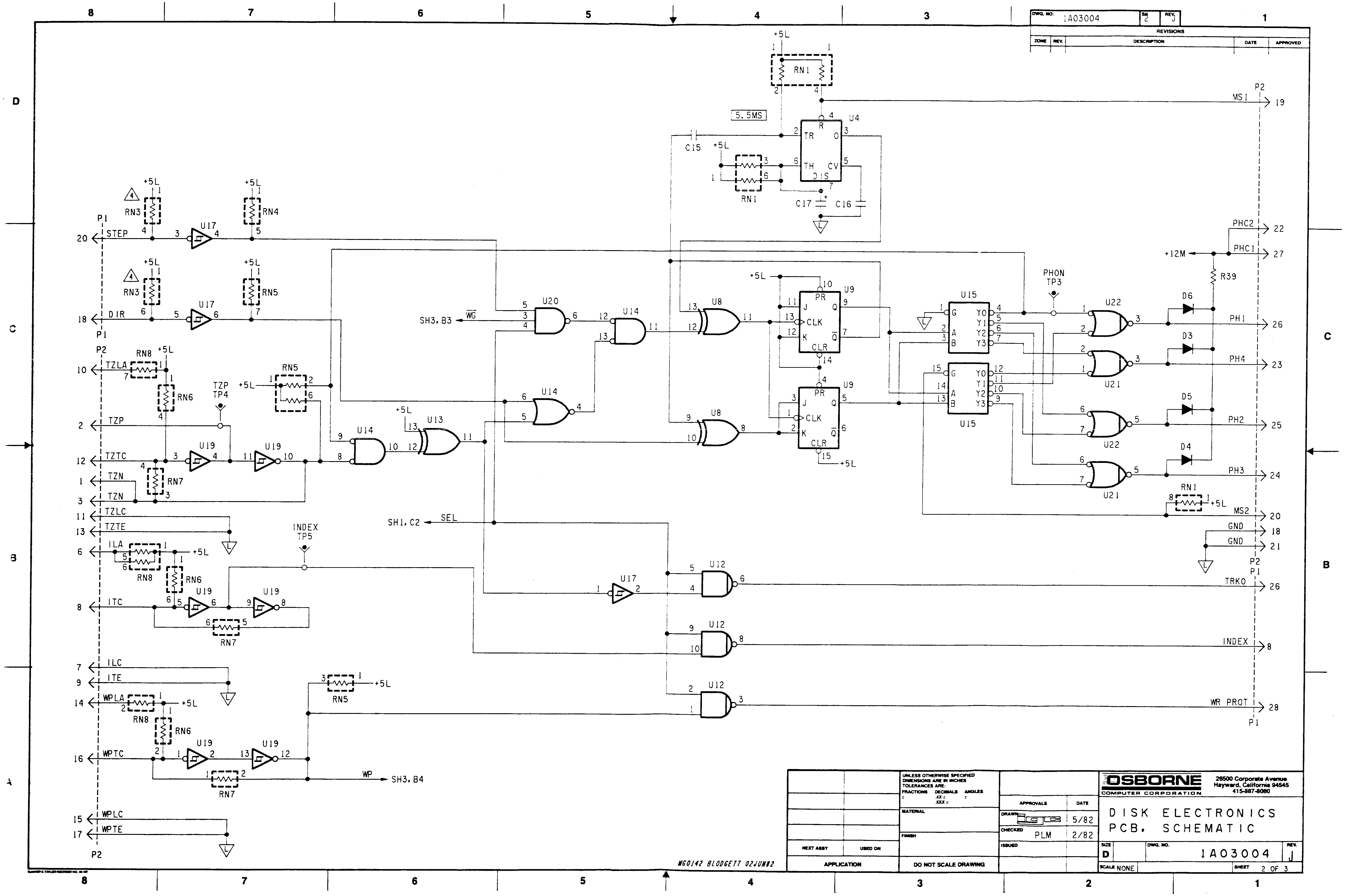
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±	
MATERIAL	
FINISH	
NEXT ASSY / USED ON	
APPLICATION	DO NOT SCALE DRAWING

APPROVALS		DATE
DRAWN		10/81
CHECKED	PLM	2/82
ISSUED	G. Walker	6/9/82

OSBORNE COMPUTER CORPORATION
26500 Corporate Avenue Hayward, California 94545 415-887-8080

DISK ELECTRONICS PCB, SCHEMATIC

SIZE D | DWG. NO. 1A03004 | REV. J
SCALE NONE | SHEET 1 OF 3



DWG. NO. 1A03004 SH 2 REV. J

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED

MG0142 BLODGETT 02JUN82

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS DECIMALS ANGLES

MATERIAL

FINISH

NEXT ASSY | USED ON

APPLICATION

DO NOT SCALE DRAWING

APPROVALS | DATE

DRAWN 5/82

CHECKED PLM 2/82

ISSUED

OSBORNE COMPUTER CORPORATION

26500 Corporate Avenue Hayward, California 94545 415-887-8080

DISK ELECTRONICS PCB. SCHEMATIC

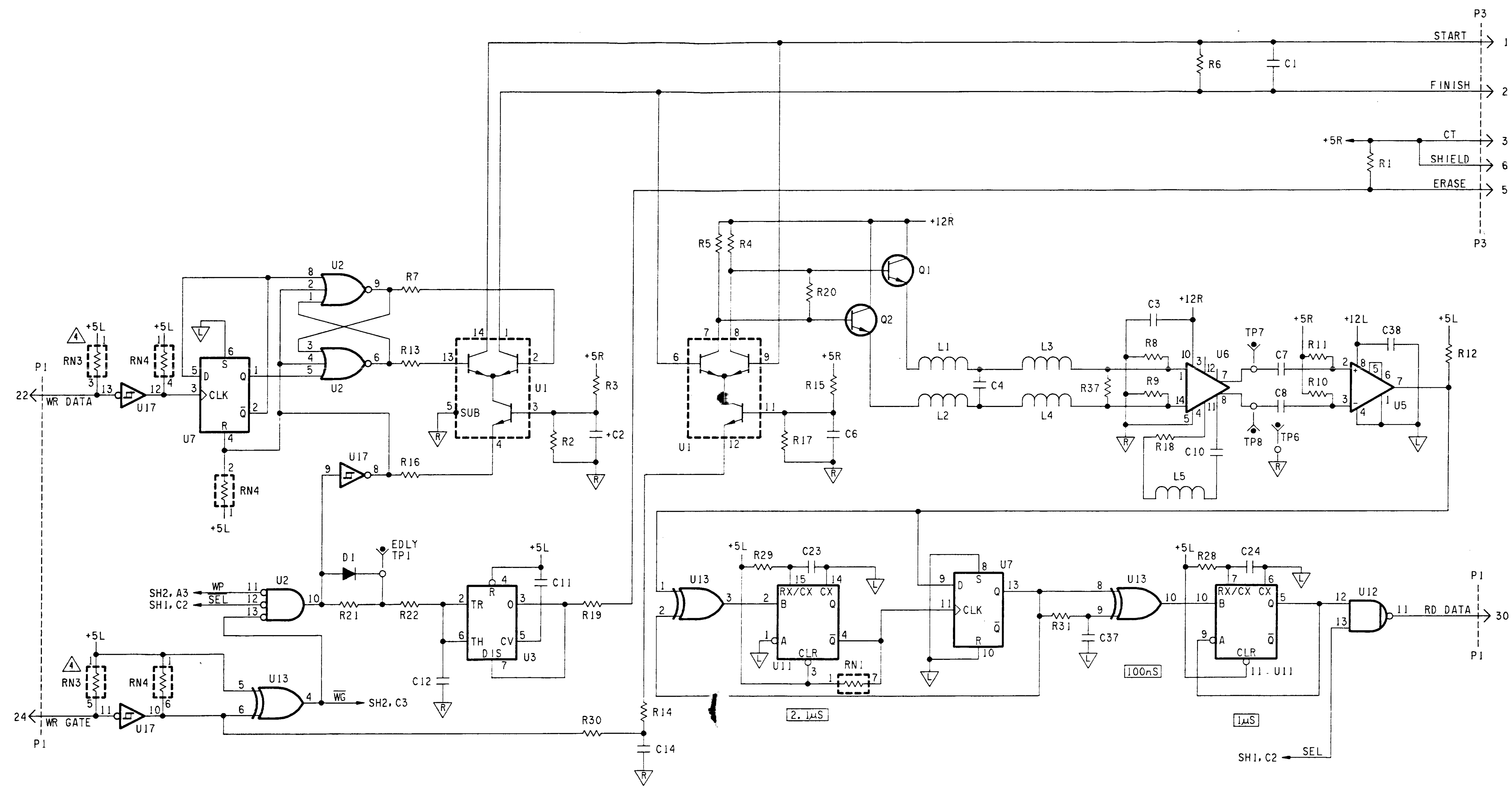
SIZE D DWG. NO. 1A03004 REV. J

SCALE NONE SHEET 2 OF 3

DWG. NO. 1A03004 SH 3 REV J

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED



		UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±	APPROVALS	DATE	OSBORNE COMPUTER CORPORATION 26500 Corporate Avenue Hayward, California 94545 415-887-8080
		MATERIAL	DRAWN	5/82	DISK ELECTRONICS PCB, SCHEMATIC
		FINISH	CHECKED PLM	2/82	
NEXT ASSY	USED ON		ISSUED		SIZE D DWG. NO. 1A03004 REV. J
APPLICATION		DO NOT SCALE DRAWING			SCALE NONE SHEET 3 OF 3

MG0143 BLODGETT 02JUN82

OSBORNE COMPUTER CORP.	TITLE DISK ELECTRONICS PCB ASSY FAB	LM# 3A03004-00
	DATE 06/22/82	Revision J

Drawn: LLB	Appvd:	ECO#:0217	Date:	Released: RELEASED

DETACHED LIST OF MATERIALS

ITEM	PART NO.	QTY	TITLE	REFERENCE DESIGNATORS
046	7P50003-00	001	RES, VARIABLE 20K 20TURN	R23
			RES, NETWK, SIP 8 PIN	
047	7P51002-00	001	47-100K10%, 1/8W BUSSED	RN6
048	7P51003-00	001	150R, 10%, 1/2W, BUSSED	RN8
049	7P51005-00	003	4.7K, 10%, 1/8W, BUSSED	RN1, 4, 5
050	7P51012-00	001	33K, 10%, 1/8W, ISOLATED	RN7
051	7P52101-00	001	RES, C.F., 5%, 1/4W 100R	R1
052	7P52105-00	001	1.0M	R35
054	7P52153-00	002	15K	R7, 13
055	7P52164-00	001	160K	R22
056	7P52203-00	001	20K	R128
057	7P52220-00	001	22R	R39
058	7P52272-00	002	2.7K	R10, 11
059	7P52273-00	001	270K	R32
060	7P52332-00	001	3.3K	R6
061	7P52335-00	001	3.3M	R33
062	7P52393-00	001	39K	R29
063	7P52472-00	010	4.7K	R4, 5, 8, 9, 20, 26, 27, 31, 34,
				37, 38
064	7P52514-00	001	510K	R21
065	7P52682-00	001	6.8K	R36
068	7P52751-00	001	750r	R18
069	7P52821-00	003	820r	R12, 25, 38
071	7P53560-00	001	RES, C.F. 5%, 1/2W 56r	R19
072	7P57153-00	001	RES, M.F. 2%, 1/4W 15K	R42
073	7P53223-00	003	22K	R24, 41, 44
074	7P57331-00	005	330r	R2, 3, 14, 16, 30
075	7P57472-00	002	4.7K	R15, 17
076	7P53912-00	001	9.1K	R43
077	7P64108-00	001	CONN 8PIN STR, TIN	TP1
078	7P64316-00	002	16PIN RT ANGL, TIN	P2
079	7P64906-00	001	6PIN RT ANGL, GOLD	P3
080	7P65201-00	001	SOCKET, IC, SIP, 8PIN	(RN3)
084	7P70041-00	001	RIVET, .125"DIA X .093"THK	
085	7P73001-00	001	HEATSINK	
087	1A03004-00	000	DISK ELEC. PCB SCHEM REV J	
088	3P04001-00	001	DISK ELEC. PCB FAB REV D	

END OF LM

OSBORNE COMPUTER CORP.

TITLE DISK ELECTRONICS PCB ASSY FAB | LM# 3A03004-00

DATE 06/22/82 | Revision J

Drawn:LLB | Check SL | Appvd: | ECO#:0217 | Date: 9/2/82 | Released: RELEASED

DETACHED LIST OF MATERIALS

ITEM	PART NO.	QTY		TITLE	REFERENCE DESIGNATORS
001	7P11014-00	002	IC, TTL	SN74LS14N	U17, 19
002	7P11086-00	001		SN74LS86N	U8
003	7P11112-00	001		SN74LS112N	U9
004	7P11123-00	001		SN74LS123N	U11
005	7P11132-00	001		SN74LS132N	U18
006	7P11139-00	001		SN74LS139N	U15
007	7P11800-00	001		SN7438N	U12
008	7P11801-00	002		SN75451BP	21, 22
009	7P11802-00	001	IC, CMOS	CD4001BE	U14
010	7P11803-00	001		CD4013BE	U7
011	7P11804-00	001		CD4023BE	U20
012	7P11805-00	001		CD4025BE	U2
013	7P11807-00	001		CD4070BE	U13
015	7P21004-00	001	IC, LINEAR	TL091CP	U16
016	7P11820-00	003	IC, LINEAR	LM555CN	U3, 4, 10
017	7P20001-00	001		LM311N	U5
018	7P20002-00	001		CA3054	U1
019	7P20003-00	001		NE592N	U6
022	7P30001-00	004	DIODE	1N4001	D3-6
023	7P30004-00	003		1N4148	D1, 2, 7
025	7P30007-00	002	TRANSISTORS	2N3904	Q1, 2
026	7P30008-00	001		TIP110	Q3
028	7P40101-00	003	INDUCTORS	150uH +-10%	L3-5
029	7P40102-00	002		560uH +-10%	L1, 2
031	7P45503-00	001	CAP, TANTLUM	.33uF +-20%, 6V	C27
032	7P45504-00	001		2.2uF +-10%, 6V	C17
033	7P45604-00	001	CAP, ALUM LO PROFILE	22uF, 16V	C34
034	7P45605-00	001		4.7uF	C36
035	7P45606-00	002		47uF, 10V	C2, 33
036	7P45609-00	001	CAP, MYLAR	68NF +-20%	C21
037	7P46100-00	020	CAP, CER, AXIAL	10NF +80-20%	C3, 5, 6-9, 11, 13-16, 18, 19, 22, C25, 26, 30-32, 35
038	7P46101-00	002	CAP, CER, AXIAL	.100PF+- 5%	C23, 24
040	7P46103-00	001		10NF +-20%	C29,
041	7P46152-00	004		1.5NF +10%	C10, 12, 20, 28
042	7P46330-00	002		33PF+-5%	C1, 37
043	7P46331-00	001		330PF+-10%	C4
044	7P46504-00	001		100NF +80-20%	C38

LM CONTINUED

Appendix A
Z80 Instruction Set

Presented here is the Z80A instruction set:

Z80 Instruction Set

Register Layout:

Main Register Set

Accumulator	Flags
B	C
D	E
H	L

Alternate Registers

Accumulator′	Flags′
B′	C′
D′	E′
H′	L′

interrupt vector I	memory refresh R
index register IX	
index register IY	
stack pointer SP	
program counter PC	

nnnn = hexadecimal 16-bit value
nn = hexadecimal 8-bit value
dd = 8-bit signed displacment
r = register
b = single bit

Z80 instruction	8080 instruction	description
ADC A,(HL)	ADC M	adds byte at HL to A
ADC A,(IX+dd)	none	adds byte indexed by X to A
ADC A,(IY+dd)	none	adds byte indexed by Y to A
ADC A,r	ADC r	adds value in register to A
ADC A,nn	ADI nn	adds value to A
ADC HL,BC	none	adds BC to HL
ADC HL,DE	none	adds DE to HL
ADC HL,HL	none	doubles HL
ADC HL,SP	none	adds stack pointer to HL
ADD A,(HL)	ADD M	adds byte at HL to A
ADD A,(IX+dd)	none	adds byte indexed by X to A
ADD A, (IY+dd)	none	adds byte indexed by Y to A
ADD A,r	ADD r	adds value in regsiter to A
ADD A,nn	ADI nn	adds value to A
ADD HL,BC	DAD B	adds BC to HL
ADD HL,DE	DAD D	adds DE to HL
ADD HL,HL	DAD H	doubles HL
ADD HL,SP	DAD SP	adds stack pointer to HL
ADD IX,BC	none	adds BC to X index
ADD IX,DE	none	adds DE to X index
ADD IX,IX	none	doubles X index
ADD IX,SP	none	adds stack pointer to X index
ADD IY,BC	none	adds BC to Y index
ADD IY,DE	none	adds DE to Y index
ADD IY,IY	none	doubles Y index
ADD IY,SP	none	adds stack pointer to Y index
AND (HL)	ANA M	logical AND with byte and A
AND (IX+dd)	none	logical AND with index and A
AND (IY+dd)	none	logical AND with index and A
AND r	ANA r	logical AND with register and A
AND nn	ANI nn	logical AND with value and A
BIT b,(HL)	none	test bit of byte at HL
BIT b,(IX+dd)	none	test bit of byte at index X
BIT b,(IY+dd)	none	test bit of byte at index Y
BIT b,r	none	test bit of regsiter value
CALL nnnn	CALL nnnn	subroutine call to location
CALL C,nnnn	CC nnnn	subroutine call if carry
CALL M,nnnn	CM nnnn	subroutine call if sign
CALL NC,nnnn	CNC nnnn	subroutine call if carry reset
CALL NZ,nnnn	CNZ nnnn	subroutine call if zero reset
CALL P,nnnn	CP nnnn	subroutine call if sign reset
CALL PE,nnnn	CPE nnnn	subroutine call if parity
CALL PO,nnnn	CPO nnnn	subroutine call if parity reset
CALL Z,nnnn	CZ nnnn	subroutine call if zero
CCF	CMC	complement carry flag
CP (HL)	CMP M	compare byte at HL to A
CP (IX+dd)	none	compare byte at X index to A
CP (IY+dd)	none	compare byte at Y index to A
CP r	CMP r	compare register value to A
CP nn	CPI nn	compare value to A
CPD	none	compare byte at HL and decrement BC

Z80 instruction	8080 instruction	description
CPDR	none	compare byte at HL, decrement and repeat
CPI	none	compare byte at HL, increment BC
CPIR	none	compare byte at HL, increment and repeat
CPL	CMA	complement A
DAA	DAA	decimal adjust A
DEC (HL)	DCR M	decrement byte at HL
DEC (IX+dd)	none	decrement byte at index X
DEC (IY+dd)	none	decrement byte at index Y
DEC r	DCR r	decrement register
DEC BC	DCX B	decrement BC
DEC DE	DCX D	decrement DE
DEC HL	DCX H	decrement HL
DEC SP	DCX SP	decrement stack pointer
DEC IX	none	decrement X index
DEC IY	none	decrement Y index
DI	DI	disable interrupts
DJNZ dd	none	decrement B and jump relative
EI	EI	enable interrupts
EX (SP),HL	XTHL	exchange stack pointer with HL
EX (SP),IX	none	exchange stack pointer with X index
EX (SP),IY	none	exchange stack pointer with Y index
EX AF,AF'	none	exchange AF register sets
EX DE,HL	XCHG	exchange DE and HL
EXX	none	exchange BC,DE,HL register sets
HALT	HLT	suspend operation
IM 0	none	set interrupt mode 0
IM 1	none	set interrupt mode 1
IM 2	none	set interrupt mode 2
IN r,(C)	none	input byte from C port to register
IN A,(nn)	IN nn	input byte from por to A
INC (HL)	INR M	increment byte at HL
INC (IX+dd)	none	increment byte at X index
INC (IY+dd)	none	increment byte at Y index
INC r	INR r	increment byte in register
INC BE	INX B	increment BC
INC DE	INX D	increment DE
INC HL	INX H	increment HL
INC SP	INX SP	increment stack pointer
INC IX	none	increment X index
INC IY	none	increment Y index
IND	none	input from C port to HL byte, decrement B, increment HL
INDR	none	input from C port to HL byte, decrement B, increment HL, repeat
INI	none	input from C port to HL byte, decrement B, decrement HL
INIR	none	input from C port to HL byte, decrement B, decrement HL, repeat
JP (HL)	PCHL	copy HL to PC then jump to it
JP (IX)	none	copy X index to PC then jump

Z80 instruction	8080 instruction	description
JP (IY)	none	copy Y index to PC then jump
JP nnnn	JMP nnnn	jump to location
JP C,nnnn	JC nnnn	jump if carry
JP M,nnnn	JM nnnn	jump if sign
JP NC,nnnn	JNC nnnn	jump if no carry
JP NZ,nnnn	JNZ nnnn	jump if not zero
JP P,nnnn	JP nnnn	jump if sign reset
JP PE,nnnn	JPE nnnn	jump if parity
JP PO,nnnn	JPO nnnn	jump if parity reset
JP Z,nnnn	JZ nnnn	jump if zero
JR dd	none	jump relative using value
JR C,dd	none	jump relative if carry
JR NC,dd	none	jump relative if no carry
JR NZ,dd	none	jump relative if not zero
JR Z,dd	none	jump relative if zero
LD (BC),A	STAX B	move A to byte at BC
LD (DE),A	STAX D	move A to byte at DE
LD (HL),r	MOV M,r	move byte in register to byte at HL
LD (HL),nn	MVI M,nn	move value to byte at HL
LD (IX+dd),r	none	move byte into byte at indexed location
LD (IX+dd),nn	none	move byte into byte at indexed location
LD (IY+dd),r	none	move byte into byte at indexed location
LD (IY+dd),nn	none	move byte into byte at indexed location
LD (nnnn),A	STA nnnn	move A to location
LD (nnnn),BC	none	move BC to location and location + 1
LD (nnnn),DE	none	move DE to location and location + 1
LD (nnnn),HL	SHLD nnnn	move HL to location and location + 1
LD (nnnn),IX	none	move IX to location and location + 1
LD (nnnn),IY	none	move IY to location and location + 1
LD (nnnn),SP	none	move stack pointer to location and location + 1
LD A,(BC)	LDAX B	move byte at BC to A
LD A,(DE)	LDAX D	move byte at DE to A
LD A,I	none	move interrupt vector register to A
LD A,R	none	move memory refresh register to A
LD I,A	none	move A to interrupt vector register
LD R,A	none	move A to memory refresh register
LD r,(HL)	MOV r,M	move byte at HL to register
LD r,(IX+dd)	none	move byte at IX to register
LD r,(IY+dd)	none	move byte at IY to register
LD r,r	MOV r,r	move byte from register to register
LD r,nn	MVI r,nn	move value to register
LD A,(nnnn)	LDA nnnn	load A from location
LD BC,(nnnn)	none	load BC from locations
LD DE,(nnnn)	none	load DE from locations
LD HL,(nnnn)	LHLD nnnn	load HL from locations
LD BC,nnnn	LXI B,nnnn	load BC with value
LD DE,nnnn	LXI D,nnnn	load DE with value
LD HL,nnnn	LXI H,nnnn	load HL with value
LD SP,nnnn	LXI SP,nnnn	load stack pointer with value
LD IX,nnnn	none	load IX with value
LD IY,nnnn	none	load IY with value
LD IX,(nnnn)	none	load IX from locations

Z80 instruction	8080 instruction	description
LD IY,(nnnn)	none	load IY from locations
LD SP,(nnnn)	none	load stack pointer from locations
LD SP,HL	SPHL	load stack pointer from HL
LD SP,IX	none	load stack pointer from IX
LD SP,IY	none	load stack pointer from IY
LDD	none	move byte at HL to DE location, decrement BC, decrement HL and DE
LDDR	none	move byte at HL to DE location, decrement BC, decrement HL and DE, repeat
LDI	none	move byte at HL to DE location, decrement BC, increment HL and DE
LDIR	none	move byte at HL to DE location, decrement BC, increment HL and DE, repeat
NEG	none	two's complement of A
NOP	NOP	no operation
OR (HL)	ORA M	logical OR of A and byte at HL
OR (IX+dd)	none	logical OR of A and byte at IX
OR (IY+dd)	none	logical OR of A and byte at IY
OR r	ORA r	logical OR of A and register
OR nn	ORI nn	logical OR of A and value
OTDR	none	output byte at HL to port C, decrement B, decrement HL, repeat
OTIR	none	output byte at HL to port C, decrement B, increment HL, repeat
OUT (C),r	none	output byte in register to port C
OUT (nn),A	OUT nn	output value to port A
OUTD	none	output byte from HL to port C, decrement B, decrement HL
OUTI	none	output byte from HL to port C, decrement B, increment HL
POP AF	POP PSW	restore AF from stack
POP BC	POP B	restore BC from stack
POP DE	POP D	restore DE from stack
POP HL	POP H	restore HL from stack
POP IX	none	restore IX from stack
POP IY	none	restore IY from stack
PUSH AF	PUSH PSW	store AF on stack
PUSH BC	PUSH B	store BC on stack
PUSH DE	PUSH D	store DE on stack
PUSH HL	PUSH H	store HL on stack
PUSH IX	none	store IX on stack
PUSH IY	none	store IY on stack
RES b,(IX+dd)	none	reset bit in byte at IX
RES b,(IY+dd)	none	reset bit in byte at IY
RES b,r	none	reset bit in register
RET	RET	return from subroutine

Z80 instruction	8080 instruction	description
RET C	RC	return if carry
RET M	RM	return if sign
RET NC	RNC	return if no carry
RET NZ	RNZ	return if not zero
RET P	RP	return if sign reset
RET PE	RPE	return if parity
RET PO	RPO	return if parity reset
RET Z	RZ	return if zero
RETI	none	return from maskable interrupt
RETN	none	return from nonmaskable interrupt
RL (HL)	none	rotate byte at HL left with carry
RL (IX+dd)	none	rotate byte at IX left with carry
RL (IY+dd)	none	rotate byte at IY left with carry
RL r	none	rotate byte in register left with carry
RLA	RAL	rotate byte in A left with carry
RLC (HL)	none	rotate byte at HL circularly left
RLC (IX+dd)	none	rotate byte at IX circularly left
RLC (IY+dd)	none	rotate byte at IY circularly left
RLC r	none	rotate byte in register circularly left
RLCA	RLC	rotate byte in A circularly left
RLD	none	rotate 12 bits n HL 4 bits at time left
RR (HL)	none	rotate byte at HL right with carry
RR (IX+dd)	none	rotate byte at IX right with carry
RR (IY+dd)	none	rotate byte at IY right with carry
RR r	none	rotate byte in register right with carry
RRA	RAR	rotate accumulator right
RRC (HL)	none	rotate byte in HL circularly right
RRC (IX+dd)	none	rotate byte in IX circularly right
RRC (IY+dd)	none	rotate byte in IY circularly right
RRC r	none	rotate byte in register circularly right
RRCA	RRC	rotate byte in A circularly right
RRD	none	rotate 12 bits in HL 4 bits at time right
RST 00h	RST 0	first restart location
RST 08h	RST 1	second restart location
RST 10h	RST 2	third restart location
RST 18h	RST 3	fourth restart location
RST 20h	RST 4	fifth restart location
RST 28h	RST 5	sixth restart location
RST 30h	RST 6	seventh restart location
RST 38h	RST 7	eighth restart location
SBC A,(HL)	SBB M	subtract byte at HL from A
SBC A,(IX+dd)	none	subtract byte at IX from A
SBC A,(IY+dd)	none	subtract byte at IY from A
SBC A,r	SBB r	subtract byte in register from A
SBC A,nn	SBI nn	subtract value from A
SBC HL,BC	none	subtract BC from HL
SBC HL,DE	none	subtract DE from HL
SBC HL,HL	none	subtract HL from HL
SBC HL,SP	none	subtract SP from HL
SCF	STC	set carry flag
SET b,(HL)	none	set big in byte at HL
SET b,(IX+dd)	none	set bit in byte at IX
SET b,(IY+dd)	none	set bit in byte at IY

Z80 instruction	8080 instruction	description
SET b,r	none	set bit in byte in register
SLA (HL)	none	arithmetic shift left on byte at HL
SLA (IX+dd)	none	arithmetic shift left on byte at IX
SLA (IY+dd)	none	arithmetic shift left on byte at IY
SLA r	none	arithmetic shift left on byte in register
SRA (HL)	none	arithmetic shift right on byte at HL
SRA (IX+dd)	none	arithmetic shift right on byte at IX
SRA (IY+dd)	none	arithmetic shift right on byte at IY
SRA r	none	arithmetic shift right on byte in register
SRL (HL)	none	logical shift right on byte at HL
SRL (IX+dd)	none	logical shift right on byte at IX
SRL (IY+dd)	none	logical shift right on byte at IY
SRL r	none	logical shift right on byte in register
SUB (HL)	SUB M	subtract byte at HL from A
SUB (IX+dd)	none	subtract byte at IX from A
SUB (IY+dd)	none	subtract byte at IY from A
SUB r	SUB r	subtract byte in register from A
SUB nn	SUI nn	subtract value from A
XOR (HL)	XRA M	XOR HL register
XOR (IX+dd)	none	XOR IX register
XOR (IY+dd)	none	XOR IY register
XOR r	XRA r	XOR register
XOR nn	XRI nn	XOR value with A

Appendix B
6821 PIA Registers/ Instructions

The following specifications on the 6821 PIA were furnished in full by Motorola, Inc:

MC6821 (1.0 MHz)
MC68A21 (1.5 MHz)
MC68B21 (2.0 MHz)

PERIPHERAL INTERFACE ADAPTER (PIA)

The MC6821 Peripheral Interface Adapter provides the universal means of interfacing peripheral equipment to the M6800 family of microprocessors. This device is capable of interfacing the MPU to peripherals through two 8-bit bidirectional peripheral data buses and four control lines. No external logic is required for interfacing to most peripheral devices.

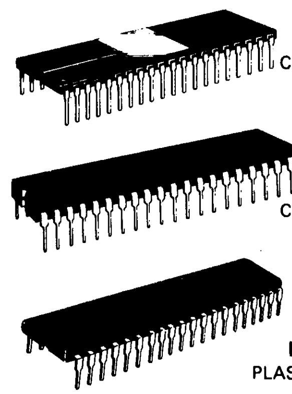
The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control/interrupt lines may be programmed for one of several control modes. This allows a high degree of flexibility in the overall operation of the interface.

- 8-Bit Bidirectional Data Bus for Communication with the MPU
- Two Bidirectional 8-Bit Buses for Interface to Peripherals
- Two Programmable Control Registers
- Two Programmable Data Direction Registers
- Four Individually-Controlled Interrupt Input Lines; Two Usable as Peripheral Control Outputs
- Handshake Control Logic for Input and Output Peripheral Operation
- High-Impedance Three-State and Direct Transistor Drive Peripheral Lines
- Program Controlled Interrupt and Interrupt Disable Capability
- CMOS Drive Capability on Side A Peripheral Lines
- Two TTL Drive Capability on All A and B Side Buffers
- TTL-Compatible
- Static Operation

MOS

(N-CHANNEL, SILICON-GATE, DEPLETION LOAD)

PERIPHERAL INTERFACE ADAPTER



L SUFFIX
CERAMIC PACKAGE
CASE 715

S SUFFIX
CERDIP PACKAGE
CASE 734

P SUFFIX
PLASTIC PACKAGE
CASE 711

MAXIMUM RATINGS

Characteristics	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range MC6821, MC68A21, MC68B21 MC6821C, MC68A21C, MC68B21C	T_A	T_L to T_H 0 to 70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Ceramic Plastic Cerdip	θ_{JA}	 50 100 60	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage (i.e., either V_{SS} or V_{CC}).

PIN ASSIGNMENT

Signal	Pin	Pin	Signal
V_{SS}	1	40	CA1
PA0	2	39	CA2
PA1	3	38	$\overline{IRQA}$
PA2	4	37	$\overline{IRQB}$
PA3	5	36	RS0
PA4	6	35	RS1
PA5	7	34	$\overline{RESET}$
PA6	8	33	D0
PA7	9	32	D1
PB0	10	31	D2
PB1	11	30	D3
PB2	12	29	D4
PB3	13	28	D5
PB4	14	27	D6
PB5	15	26	D7
PB6	16	25	E
PB7	17	24	CS1
CB1	18	23	$\overline{CS2}$
CB2	19	22	CS0
V_{CC}	20	21	R/$\overline{W}$

©MOTOROLA INC., 1981

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \tag{1}$$

Where:

T_A ≡ Ambient Temperature, °C

θ_{JA} ≡ Package Thermal Resistance, Junction-to-Ambient, °C/W

P_D ≡ $P_{INT} + P_{PORT}$

P_{INT} ≡ $I_{CC} \times V_{CC}$, Watts — Chip Internal Power

P_{PORT} ≡ Port Power Dissipation, Watts — User Determined

For most applications $P_{PORT} \ll P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is:

$$P_D = K \div (T_J + 273°C) \tag{2}$$

Solving equations 1 and 2 for K gives:

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2 \tag{3}$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc ±5%, V_{SS} = 0, T_A = T_L to T_H unless otherwise noted).

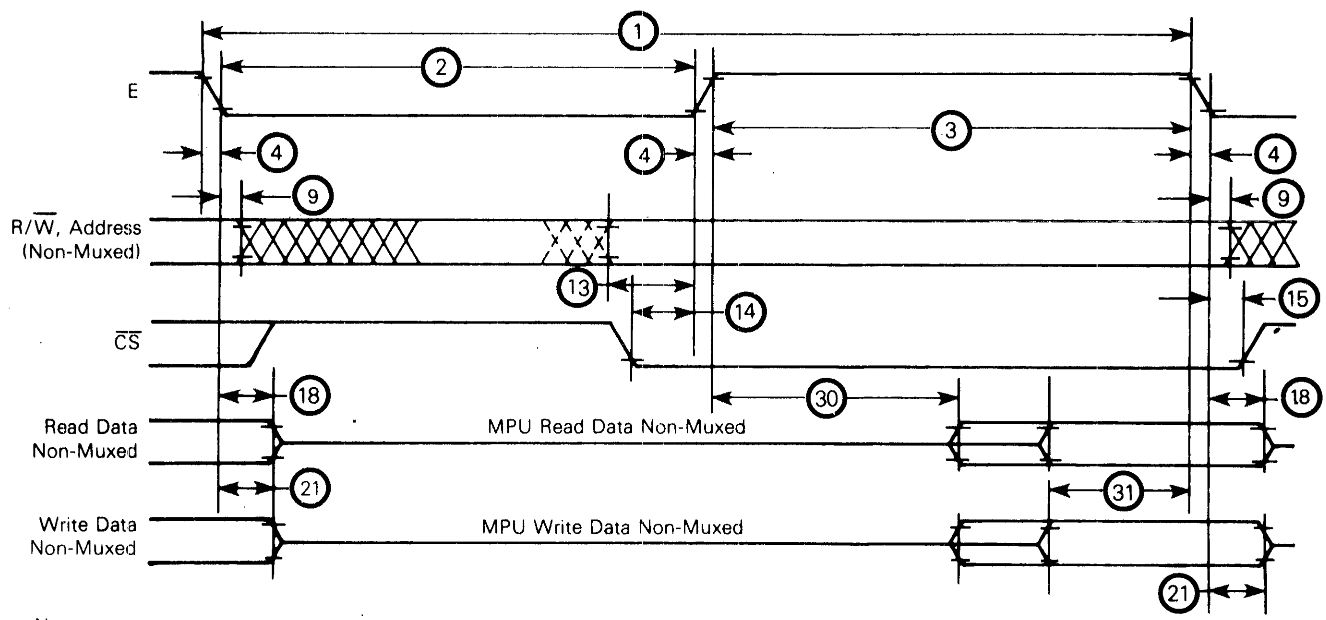
Characteristic		Symbol	Min	Typ	Max	Unit
BUS CONTROL INPUTS (R/$\overline{W}$, Enable, $\overline{RESET}$, RS0, RS1, CS0, CS1, $\overline{CS2}$)						
Input High Voltage		V_{IH}	V_{SS}+2.0	–	V_{CC}	V
Input Low Voltage		V_{IL}	V_{SS}−0.3	–	V_{SS}+0.8	V
Input Leakage Current (V_{in} = 0 to 5.25 V)		I_{in}	–	1.0	2.5	μA
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{in}	–	–	7.5	pF
INTERRUPT OUTPUTS ($\overline{IRQA}$, $\overline{IRQB}$)						
Output Low Voltage (I_{Load} = 3.2 mA)		V_{OL}	–	–	V_{SS}+0.4	V
Three-State Output Leakage Current		I_{OZ}	–	1.0	10	μA
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{out}	–	–	5.0	pF
DATA BUS (D0-D7)						
Input High Voltage		V_{IH}	V_{SS}+2.0	–	V_{CC}	V
Input Low Voltage		V_{IL}	V_{SS}−0.3	–	V_{SS}+0.8	V
Three-State Input Leakage Current (V_{in} = 0.4 to 2.4 V)		I_{IZ}	–	2.0	10	μA
Output High Voltage (I_{Load} = −205 μA)		V_{OH}	V_{SS}+2.4	–	–	V
Output Low Voltage (I_{Load} = 1.6 mA)		V_{OL}	–	–	V_{SS}+0.4	V
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{in}	–	–	12.5	pF
PERIPHERAL BUS (PA0-PA7, PB0-PB7, CA1, CA2, CB1, CB2)						
Input Leakage Current (V_{in} = 0 to 5.25 V)	R/$\overline{W}$, $\overline{RESET}$, RS0, RS1, CS0, CS1, $\overline{CS2}$, CA1, CB1, Enable	I_{in}	–	1.0	2.5	μA
Three-State Input Leakage Current (V_{in} = 0.4 to 2.4 V)	PB0-PB7, CB2	I_{IZ}	–	2.0	10	μA
Input High Current (V_{IH} = 2.4 V)	PA0-PA7, CA2	I_{IH}	−200	−400	–	μA
Darlington Drive Current (V_O = 1.5 V)	PB0-PB7, CB2	I_{OH}	−1.0	–	−10	mA
Input Low Current (V_{IL} = 0.4 V)	PA0-PA7, CA2	I_{IL}	–	−1.3	−2.4	mA
Output High Voltage (I_{Load} = −200 μA)	PA0-PA7, PB0-PB7, CA2, CB2	V_{OH}	V_{SS}+2.4	–	–	V
Output High Voltage (I_{Load} = −10 μA)	PA0-PA7, CA2	V_{OH}	V_{CC}−1.0	–	–	V
Output Low Voltage (I_{Load} = 3.2 mA)		V_{OL}	–	–	V_{SS}+0.4	V
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{in}	–	–	10	pF
POWER REQUIREMENTS						
Internal Power Dissipation (Measured at T_A = T_L)		P_{INT}	–	–	550	mW

BUS TIMING CHARACTERISTICS (See Notes 1 and 2)

Ident. Number	Characteristic	Symbol	MC6821		MC68A21		MC68B21		Unit
			Min	Max	Min	Max	Min	Max	
1	Cycle Time	t_{cyc}	1.0	10	0.67	10	0.5	10	μs
2	Pulse Width, E Low	PW_{EL}	430	–	280	–	210	–	ns
3	Pulse Width, E High	PW_{EH}	450	–	280	–	220	–	ns
4	Clock Rise and Fall Time	t_r, t_f	–	25	–	25	–	20	ns
9	Address Hold Time	t_{AH}	10	–	10	–	10	–	ns
13	Address Setup Time Before E	t_{AS}	80	–	60	–	40	–	ns
14	Chip Select Setup Time Before E	t_{CS}	80	–	60	–	40	–	ns
15	Chip Select Hold Time	t_{CH}	10	–	10	–	10	–	ns
18	Read Data Hold Time	t_{DHR}	20	50*	20	50*	20	50*	ns
21	Write Data Hold Time	t_{DHW}	10	–	10	–	10	–	ms
30	Output Data Delay Time	t_{DDR}	–	290	–	180	–	150	ns
31	Input Data Setup Time	t_{DSW}	165	–	80	–	60	–	ns

*The data bus output buffers are no longer sourcing or sinking current by t_{DHR}max (High Impedance).

FIGURE 1 — BUS TIMING



Notes:

1. Voltage levels shown are $V_L \leq 0.4$ V, $V_H \geq 2.4$ V, unless otherwise specified.
2. Measurement points shown are 0.8 V and 2.0 V, unless otherwise specified.

PERIPHERAL TIMING CHARACTERISTICS (V_{CC}=5.0 V ±5%, V_{SS}=0 V, $T_A=T_L$ to T_H unless otherwise specified)

Characteristic	Symbol	MC6821		MC68A21		MC68B21		Unit	Reference Fig. No.
		Min	Max	Min	Max	Min	Max		
Data Setup Time	t_{PDS}	200	–	135	–	100	–	ns	6
Data Hold Time	t_{PDH}	0	–	0	–	0	–	ns	6
Delay Time, Enable Negative Transition to CA2 Negative Transition	t_{CA2}	–	1.0	–	0.670	–	0.500	μs	3, 7, 8
Delay Time, Enable Negative Transition to CA2 Positive Transition	T_{RS1}	–	1.0	–	0.670	–	0.500	μs	3, 7
Rise and Fall Times for CA1 and CA2 Input Signals	t_r, t_f	–	1.0	–	1.0	–	1.0	μs	8
Delay Time from CA1 Active Transition to CA2 Positive Transition	t_{RS2}	–	2.0	–	1.35	–	1.0	μs	3, 8
Delay Time, Enable Negative Transition to Data Valid	t_{PDW}	–	1.0	–	0.670	–	0.5	μs	3, 9, 10
Delay Time, Enable Negative Transition to CMOS Data Valid PA0-PA7, CA2	t_{CMOS}	–	2.0	–	1.35	–	1.0	μs	4, 9
Delay Time, Enable Positive Transition to CB2 Negative Transition	t_{CB2}	–	1.0	–	0.670	–	0.5	μs	3, 11, 12
Delay Time, Data Valid to CB2 Negative Transition	t_{DC}	20	–	20	–	20	–	ns	3, 10
Delay Time, Enable Positive Transition to CB2 Positive Transition	t_{RS1}	–	1.0	–	0.670	–	0.5	μs	3, 11
Control Output Pulse Width, CA2/CB2	PW_{CT}	500	–	375	–	250	–	ns	3, 11
Rise and Fall Time for CB1 and CB2 Input Signals	t_r, t_f	–	1.0	–	1.0	–	1.0	μ	12
Delay Time, CB1 Active Transition to CB2 Positive Transition	t_{RS2}	–	2.0	–	1.35	–	1.0	μs	3, 12
Interrupt Release Time, $\overline{IRQA}$ and $\overline{IRQB}$	t_{IR}	–	1.60	–	1.10	–	0.85	μs	5, 14
Interrupt Response Time	t_{RS3}	–	1.0	–	1.0	–	1.0	μs	5, 13
Interrupt Input Pulse Time	PW_I	500	–	500	–	500	–	ns	13
$\overline{RESET}$ Low Time*	t_{RL}	1.0	–	0.66	–	0.5	–	μs	15

*The $\overline{RESET}$ line must be high a minimum of 1.0 μs before addressing the PIA.

FIGURE 2 — BUS TIMING TEST LOADS

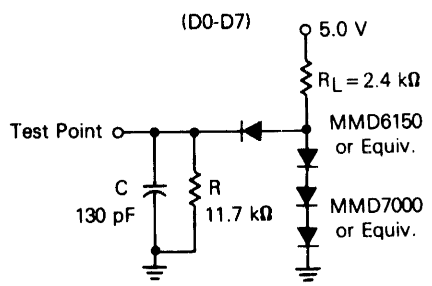


FIGURE 3 — TTL EQUIVALENT TEST LOAD

(PA0–PA7, PB0–PB7, CA2, CB2)

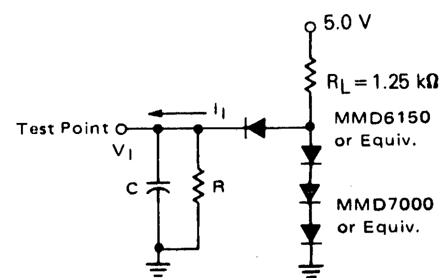


FIGURE 4 — CMOS EQUIVALENT TEST LOAD

(PA0-PA7, CA2)

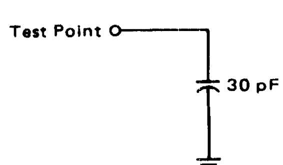


FIGURE 5 — NMOS EQUIVALENT TEST LOAD

($\overline{IRQ}$ Only)

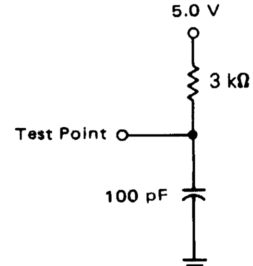


FIGURE 6 — PERIPHERAL DATA SETUP AND HOLD TIMES (Read Mode)

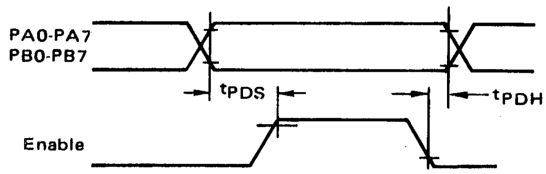
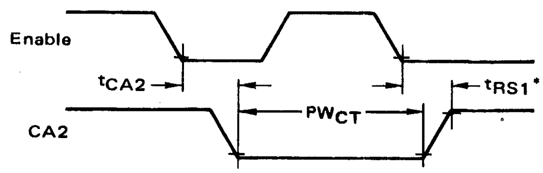


FIGURE 7 — CA2 DELAY TIME (Read Mode; CRA-5 = CRA3 = 1, CRA-4 = 0)



*Assumes part was deselected during the previous E pulse.

FIGURE 8 — CA2 DELAY TIME (Read Mode; CRA-5 = 1, CRA-3 = CRA-4 = 0)

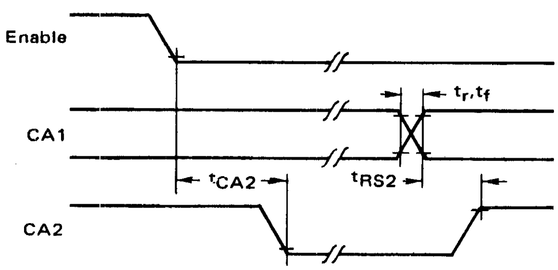


FIGURE 9 — PERIPHERAL CMOS DATA DELAY TIMES (Write Mode; CRA-5 = CRA-3 = 1, CRA-4 = 0)

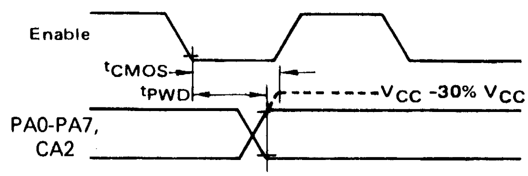
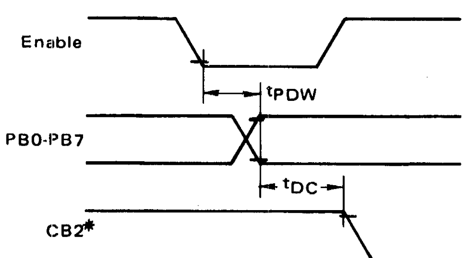
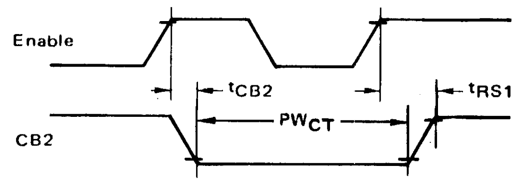


FIGURE 10 — PERIPHERAL DATA AND CB2 DELAY TIMES (Write Mode; CRB-5 = CRB-3 = 1, CRB-4 = 0)



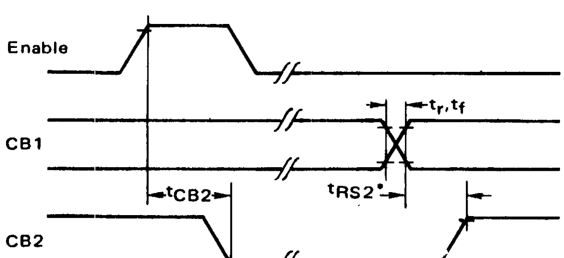
*CB2 goes low as a result of the positive transition of Enable.

FIGURE 11 — CB2 DELAY TIME (Write Mode; CRB-5 = CRB-3 = 1, CRB-4 = 0)



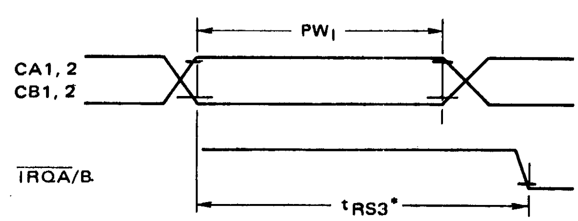
*Assumes part was deselected during the previous E pulse.

FIGURE 12 — CB2 DELAY TIME (Write Mode; CRB-5 = 1, CRB-3 = CRB-4 = 0)



*Assumes part was deselected during any previous E pulse.

FIGURE 13 — INTERRUPT PULSE WIDTH AND $\overline{IRQ}$ RESPONSE



*Assumes Interrupt Enable Bits are set.

Note: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

FIGURE 14 — $\overline{IRQ}$ RELEASE TIME

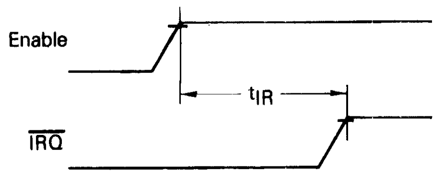
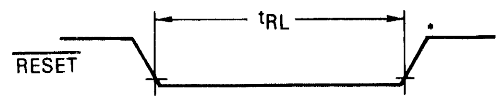


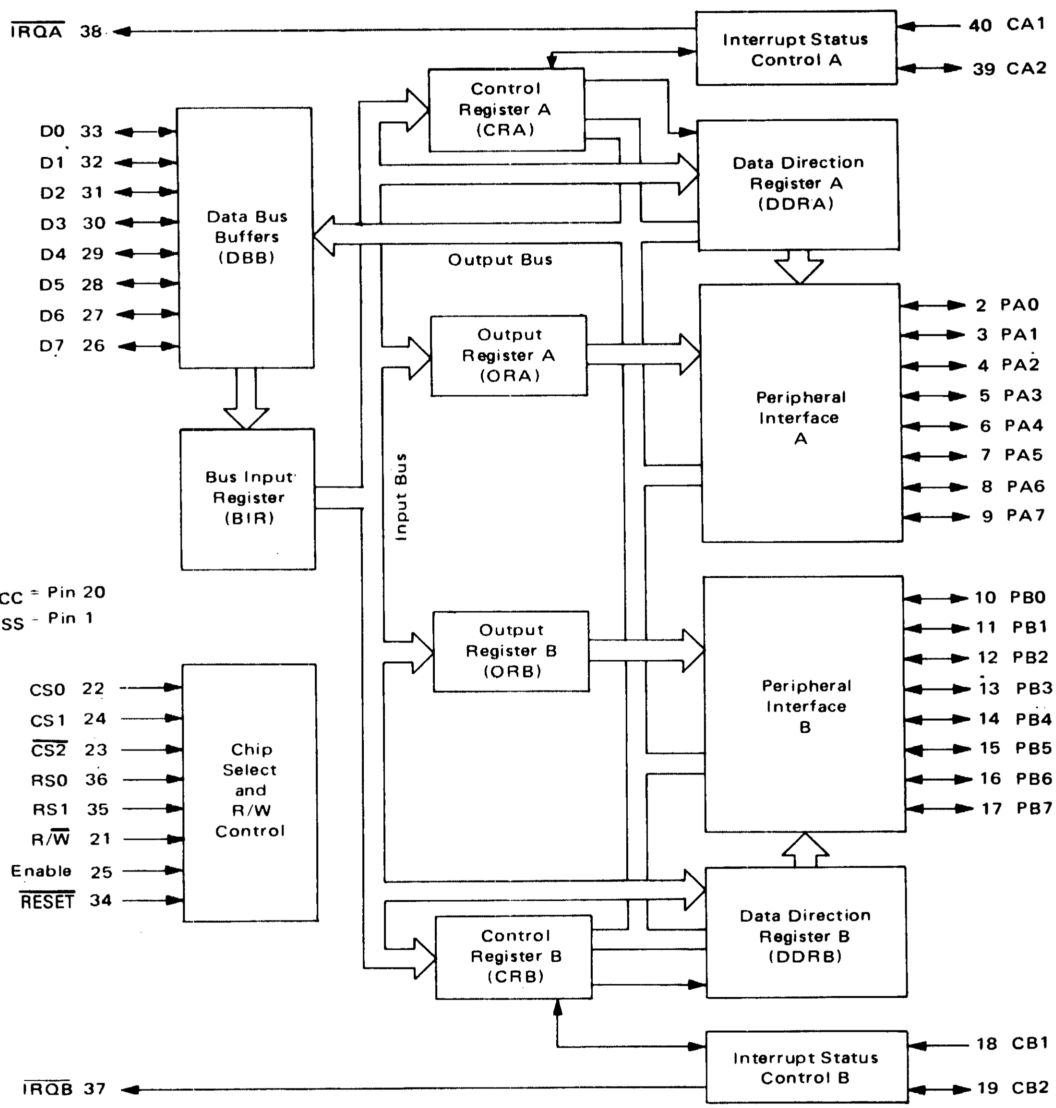
FIGURE 15 — $\overline{RESET}$ LOW TIME



*The $\overline{RESET}$ line must be a V_{IH} for a minimum of 1.0 μs before addressing the PIA.

Note: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

FIGURE 16 — EXPANDED BLOCK DIAGRAM



PIA INTERFACE SIGNALS FOR MPU

The PIA interfaces to the M6800 bus with an 8-bit bidirectional data bus, three chip select lines, two register select lines, two interrupt request lines, a read/write line, an enable line and a reset line. To ensure proper operation with the MC6800, MC6802, or MC6808 microprocessors, VMA should be used as an active part of the address decoding.

Bidirectional Data (D0-D7) — The bidirectional data lines (D0-D7) allow the transfer of data between the MPU and the PIA. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs a PIA read operation. The read/write line is in the read (high) state when the PIA is selected for a read operation.

Enable (E) — The enable pulse, E, is the only timing signal that is supplied to the PIA. Timing of all other signals is referenced to the leading and trailing edges of the E pulse.

Read/Write (R/$\overline{\text{W}}$) — This signal is generated by the MPU to control the direction of data transfers on the data bus. A low state on the PIA read/write line enables the input buffers and data is transferred from the MPU to the PIA on the E signal if the device has been selected. A high on the read/write line sets up the PIA for a transfer of data to the bus. The PIA output buffers are enabled when the proper address and the enable pulse E are present.

$\overline{\text{RESET}}$ — The active low $\overline{\text{RESET}}$ line is used to reset all register bits in the PIA to a logical zero (low). This line can be used as a power-on reset and as a master reset during system operation.

Chip Selects (CS0, CS1, and $\overline{\text{CS2}}$) — These three input signals are used to select the PIA. CS0 and CS1 must be high and $\overline{\text{CS2}}$ must be low for selection of the device. Data transfers are then performed under the control of the enable and read/write signals. The chip select lines must be stable for the duration of the E pulse. The device is deselected when any of the chip selects are in the inactive state.

Register Selects (RS0 and RS1) — The two register select lines are used to select the various registers inside the PIA. These two lines are used in conjunction with internal Control Registers to select a particular register that is to be written or read.

The register and chip select lines should be stable for the duration of the E pulse while in the read or write cycle.

Interrupt Request ($\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$) — The active low Interrupt Request lines ($\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$) act to interrupt the MPU either directly or through interrupt priority circuitry. These lines are "open drain" (no load device on the chip). This permits all interrupt request lines to be tied together in a wire-OR configuration.

Each Interrupt Request line has two internal interrupt flag bits that can cause the Interrupt Request line to go low. Each flag bit is associated with a particular peripheral interrupt line. Also, four interrupt enable bits are provided in the PIA which may be used to inhibit a particular interrupt from a peripheral device.

Servicing an interrupt by the MPU may be accomplished by a software routine that, on a prioritized basis, sequentially reads and tests the two control registers in each PIA for interrupt flag bits that are set.

The interrupt flags are cleared (zeroed) as a result of an MPU Read Peripheral Data Operation of the corresponding data register. After being cleared, the interrupt flag bit cannot be enabled to be set until the PIA is deselected during an E pulse. The E pulse is used to condition the interrupt control lines (CA1, CA2, CB1, CB2). When these lines are used as interrupt inputs, at least one E pulse must occur from the inactive edge to the active edge of the interrupt input signal to condition the edge sense network. If the interrupt flag has been enabled and the edge sense circuit has been properly conditioned, the interrupt flag will be set on the next active transition of the interrupt input pin.

PIA PERIPHERAL INTERFACE LINES

The PIA provides two 8-bit bidirectional data buses and four interrupt/control lines for interfacing to peripheral devices.

Section A Peripheral Data (PA0-PA7) — Each of the peripheral data lines can be programmed to act as an input or output. This is accomplished by setting a "1" in the corresponding Data Direction Register bit for those lines which are to be outputs. A "0" in a bit of the Data Direction Register causes the corresponding peripheral data line to act as an input. During an MPU Read Peripheral Data Operation, the data on peripheral lines programmed to act as inputs appears directly on the corresponding MPU Data Bus lines. In the input mode, the internal pullup resistor on these lines represents a maximum of 1.5 standard TTL loads.

The data in Output Register A will appear on the data lines that are programmed to be outputs. A logical "1" written into the register will cause a "high" on the corresponding data line while a "0" results in a "low." Data in Output Register A may be read by an MPU "Read Peripheral Data A" operation when the corresponding lines are programmed as outputs. This data will be read property if the voltage on the peripheral data lines is greater than 2.0 volts for a logic "1" output and less than 0.8 volt for a logic "0" output. Loading the output lines such that the voltage on these lines does not reach full voltage causes the data transferred into the MPU on a Read operation to differ from that contained in the respective bit of Output Register A.

Section B Peripheral Data (PB0-PB7) — The peripheral data lines in the B Section of the PIA can be programmed to act as either inputs or outputs in a similar manner to PA0-PA7. They have three-state capabiity, allowing them to enter a high-impedance state when the peripheral data line is used as an input. In addition, data on the peripheral data lines

PB0-PB7 will be read properly from those lines programmed as outputs even if the voltages are below 2.0 volts for a "high" or above 0.8 V for a "low". As outputs, these lines are compatible with standard TTL and may also be used as a source of up to 1 milliampere at 1.5 volts to directly drive the base of a transistor switch.

Interrupt Input (CA1 and CB1) — Peripheral input lines CA1 and CB1 are input only lines that set the interrupt flags of the control registers. The active transition for these signals is also programmed by the two control registers.

Peripheral Control (CA2) — The peripheral control line CA2 can be programmed to act as an interrupt input or as a peripheral control output. As an output, this line is compatible with standard TTL; as an input the internal pullup resistor on this line represents 1.5 standard TTL loads. The function of this signal line is programmed with Control Register A.

Peripheral Control (CB2) — Peripheral Control line CB2 may also be programmed to act as an interrupt input or peripheral control output. As an input, this line has high input impedance and is compatible with standard TTL. As an output it is compatible with standard TTL and may also be used as a source of up to 1 milliampere at 1.5 volts to directly drive the base of a transistor switch. This line is programmed by Control Register B.

INTERNAL CONTROLS

INITIALIZATION

A $\overline{\text{RESET}}$ has the effect of zeroing all PIA registers. This will set PA0-PA7, PB0-PB7, CA2 and CB2 as inputs, and all interrupts disabled. The PIA must be configured during the restart program which follows the reset.

There are six locations within the PIA accessible to the MPU data bus: two Peripheral Registers, two Data Direction Registers, and two Control Registers. Selection of these locations is controlled by the RS0 and RS1 inputs together with bit 2 in the Control Register, as shown in Table 1.

Details of possible configurations of the Data Direction and Control Register are as follows:

TABLE 1 — INTERNAL ADDRESSING

RS1	RS0	Control Register Bit		Location Selected
		CRA-2	CRB-2	
0	0	1	X	Peripheral Register A
0	0	0	X	Data Direction Register A
0	1	X	X	Control Register A
1	0	X	1	Peripheral Register B
1	0	X	0	Data Direction Register B
1	1	X	X	Control Register B

X = Don't Care

PORT A-B HARDWARE CHARACTERISTICS

As shown in Figure 17, the MC6821 has a pair of I/O ports whose characteristics differ greatly. The A side is designed to drive CMOS logic to normal 30% to 70% levels, and incorporates an internal pullup device that remains connected even in the input mode. Because of this, the A side requires more drive current in the input mode than Port B. In contrast, the B side uses a normal three-state NMOS buffer which cannot pullup to CMOS levels without external resistors. The B side can drive extra loads such as Darlingtons without problem. When the PIA comes out of reset, the A port represents inputs with pullup resistors, whereas the B side (input mode also) will float high or low, depending upon the load connected to it.

Notice the differences between a Port A and Port B read operation when in the output mode. When reading Port A, the actual pin is read, whereas the B side read comes from an output latch, ahead of the actual pin.

CONTROL REGISTERS (CRA and CRB)

The two Control Registers (CRA and CRB) allow the MPU to control the operation of the four peripheral control lines CA1, CA2, CB1, and CB2. In addition they allow the MPU to enable the interrupt lines and monitor the status of the interrupt flags. Bits 0 through 5 of the two registers may be written or read by the MPU when the proper chip select and register select signals are applied. Bits 6 and 7 of the two registers are read only and are modified by external interrupts occurring on control lines CA1, CA2, CB1, or CB2. The format of the control words is shown in Figure 18.

DATA DIRECTION ACCESS CONTROL BIT (CRA-2 and CRB-2)

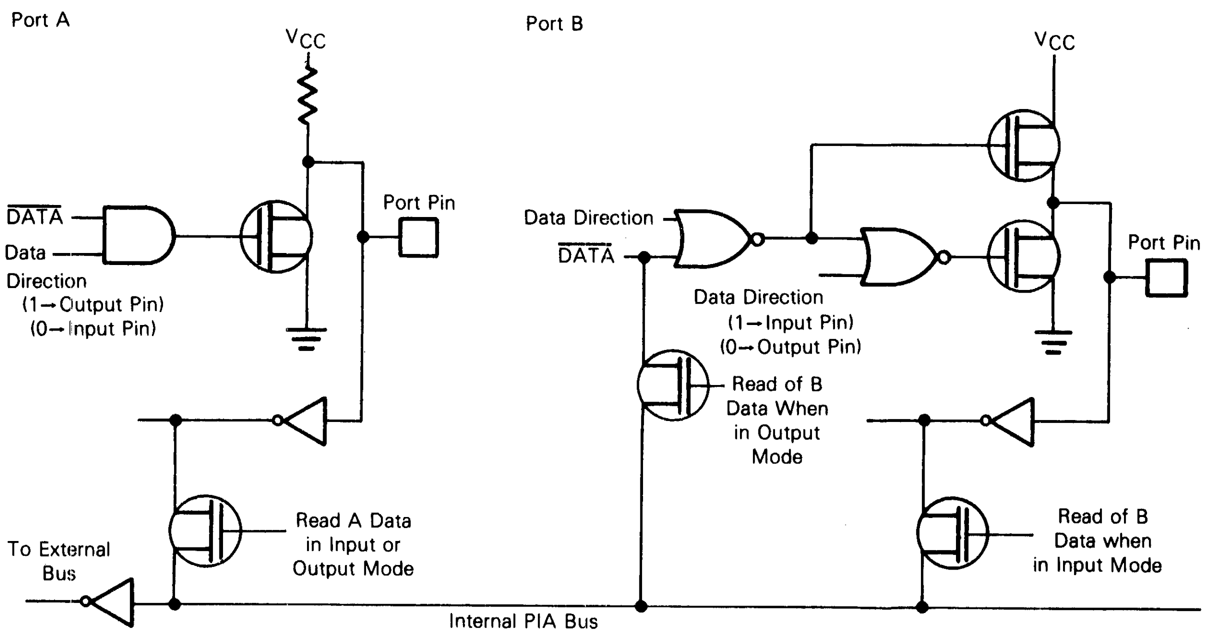
Bit 2, in each Control Register (CRA and CRB), determines selection of either a Peripheral Output Register or the corresponding Data Direction E Register when the proper register select signals are applied to RS0 and RS1. A "1" in bit 2 allows access of the Peripheral Interface Register, while a "0" causes the Data Direction Register to be addressed.

Interrupt Flags (CRA-6, CRA-7, CRB-6, and CRB-7) — The four interrupt flag bits are set by active transitions of signals on the four Interrupt and Peripheral Control lines when those lines are programmed to be inputs. These bits cannot be set directly from the MPU Data Bus and are reset indirectly by a Read Peripheral Data Operation on the appropriate section.

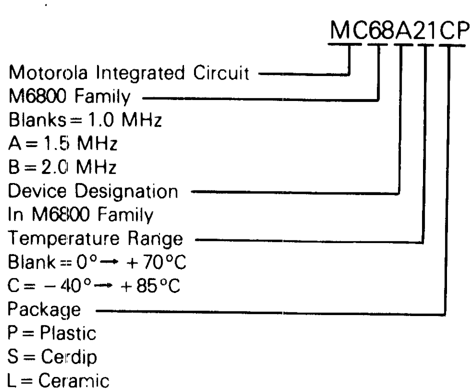
Control of CA2 and CB2 Peripheral Control Lines (CRA-3, CRA-4, CRA-5, CRB-3, CRB-4, and CRB-5) — Bits 3, 4, and 5 of the two control registers are used to control the CA2 and CB2 Peripheral Control lines. These bits determine if the control lines will be an interrupt input or an output control signal. If bit CRA-5 (CRB-5) is low, CA2 (CB2) is an interrupt input line similar to CA1 (CB1). When CRA-5 (CRB-5) is high, CA2 (CB2) becomes an output signal that may be used to control peripheral data transfers. When in the output mode, CA2 and CB2 have slightly different loading characteristics.

Control of CA1 and CB1 Interrupt Input Lines (CRA-0, CRB-1, CRA-1, and CRB-1) — The two lowest-order bits of the control registers are used to control the interrupt input lines CA1 and CB1. Bits CRA-0 and CRB-0 are used to enable the MPU interrupt signals $\overline{IRQA}$ and $\overline{IRQB}$, respectively. Bits CRA-1 and CRB-1 determine the active transition of the interrupt input signals CA1 and CB1.

FIGURE 17 — PORT A AND PORT B EQUIVALENT CIRCUITS



ORDERING INFORMATION



BETTER PROGRAM

Better program processing is available on all types listed. Add suffix letters to part number.

Level 1 add "S" Level 2 add "D" Level 3 add "DS"

Level 1 "S" = 10 Temp Cycles — (−25 to 150°C);
Hi Temp testing at T_A max.
Level 2 "D" = 168 Hour Burn-in at 125°C
Level 3 "DS" = Combination of Level 1 and 2.

Speed	Device	Temperature Range
1.0 MHz	MC6821P,L,S	0 to 70°C
	MC6821CP,CL,CS	−40 to +85°C
1.5 MHz	MC68A21P,L,S	0 to +70°C
	MC68A21CP,CL,CS	−40 to +85°C
2.0 MHz	MC68B21P,L,S	0 to +70°C

FIGURE 18 — CONTROL WORD FORMAT

Determine Active CA1 (CB1) Transition for Setting Interrupt Flag IRQA(B)1 — (bit 7)

b1=0: IRQA(B)1 set by high-to-low transition on CA1 (CB1)

b1=1: IRQA(B)1 set by low-to-high transition on CA1 (CB1).

CA1 (CB1) Interrupt Request Enable/Disable

b0=0: Disables IRQA(B) MPU Interrupt by CA1 (CB1) active transition.[1]

b0=1: Enable IRQA(B) MPU Interrupt by CA1 (CB1) active transition.

1. IRQA(B) will occur on next (MPU generated) positive transition of b0 if CA1 (CB1) active transition occurred while interrupt was disabled.

IRQA(B) 1 Interrupt Flag (bit 7)

Goes high on active transition of CA1 (CB1); Automatically cleared by MPU Read of Output Register A(B). May also be cleared by hardware Reset.

Control Register

b7	b6	b5	b4	b3	b2	b1	b0
IRQA(B)1 Flag	IRQA(B)2 Flag	CA2 (CB2) Control			DDR Access	CA1 (CB1) Control	

IRQA(B)2 Interrupt Flag (bit 6)

When CA2 (CB2) is an input, IRQA(B) goes high on active transition CA2 (CB2); Automatically cleared by MPU Read of Output Register A(B). May also be cleared by hardware Reset.

CA2 (CB2) Established as Output (b5=1): IRQA(B) 2=0, not affected by CA2 (CB2) transitions.

Determines Whether Data Direction Register Or Output Register is Addressed

b2=0: Data Direction Register selected.

b2=1: Output Register selected.

CA2 (CB2) Established as Output by b5=1

(Note that operation of CA2 and CB2 output functions are not identical)

b5 b4 b3

1 0 → CA2

b3=0: **Read Strobe with CA1 Restore**
CA2 goes low on first high-to-low E transition following an MPU read of Output Register A; returned high by next active CA1 transition, as specified by bit 1.

b3=1: **Read Strobe with E Restore**
CA2 goes low on first high-to-low E transition following an MPU read of Output Register A; returned high by next high-to-low E transition during a deselect.

→ CB2

b3=0: **Write Strobe with CB1 Restore**
CB2 goes low on first low-to-high E transition following an MPU write into Output Register B; returned high by the next active CB1 transition as specified by bit 1. CRB-b7 must first be cleared by a read of data.

b3=1: **Write Strobe with E Restore**
CB2 goes low on first low-to-high E transition following an MPU write into Output Register B; returned high by the next low-to-high E transition following an E pulse which occurred while the part was deselected.

b5 b4 b3

1 1 → **Set/Reset CA2 (CB2)**

CA2 (CB2) goes low as MPU writes b3=0 into Control Register.

CA2 (CB2) goes high as MPU writes b3=1 into Control Register.

CA2 (CB2) Established as Input by b5=0

b5 b4 b3

0

→ **CA2 (CB2) Interrupt Request Enable/Disable**

b3=0: Disables IRQA(B) MPU Interrupt by CA2 (CB2) active transition.*

b3=1: Enables IRQA(B) MPU Interrupt by CA2 (CB2) active transition.

*IRQA(B) will occur on next (MPU generatted) positive transition of b3 if CA2 (CB2), active transition occurred while interrupt was disabled.

→ **Determines Active CA2 (CB2) Transition for Setting Interrupt Flag IRQA(B)2 — (Bit b6)**

b4=0: IRQA(B)2 set by high-to-low transition on CA2 (CB2).

b4=1: IRQA(B)2 set by low-to-high transition on CA2 (CB2).

Appendix C
6850 ACIA Registers/ Instructions

The following specifications on the 6850 ACIA were furnished in full by Motorola, Inc:

MC6850 (1.0 MHz)
MC68A50 (1.5 MHz)
MC68B50 (2.0 MHz)

ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER (ACIA)

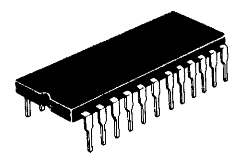
The MC6850 Asynchronous Communications Interface Adapter provides the data formatting and control to interface serial asynchronous data communications information to bus organized systems such as the MC6800 Microprocessing Unit.

The bus interface of the MC6850 includes select, enable, read/write, interrupt and bus interface logic to allow data transfer over an 8-bit bidirectional data bus. The parallel data of the bus system is serially transmitted and received by the asynchronous data interface, with proper formatting and error checking. The functional configuration of the ACIA is programmed via the data bus during system initialization. A programmable Control Register provides variable word lengths, clock division ratios, transmit control, receive control, and interrupt control. For peripheral or modem operation, three control lines are provided. These lines allow the ACIA to interface directly with the MC6860L 0-600 bps digital modem.

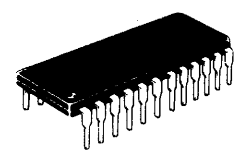
- 8- and 9-Bit Transmission
- Optional Even and Odd Parity
- Parity, Overrun and Framing Error Checking
- Programmable Control Register
- Optional ÷1, ÷16, and ÷64 Clock Modes
- Up to 1.0 Mbps Transmission
- False Start Bit Deletion
- Peripheral/Modem Control Functions
- Double Buffered
- One- or Two-Stop Bit Operation

MOS
(N-CHANNEL, SILICON-GATE)

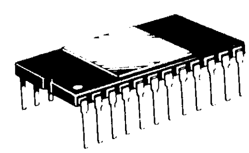
ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER



S SUFFIX
CERDIP PACKAGE
CASE 623

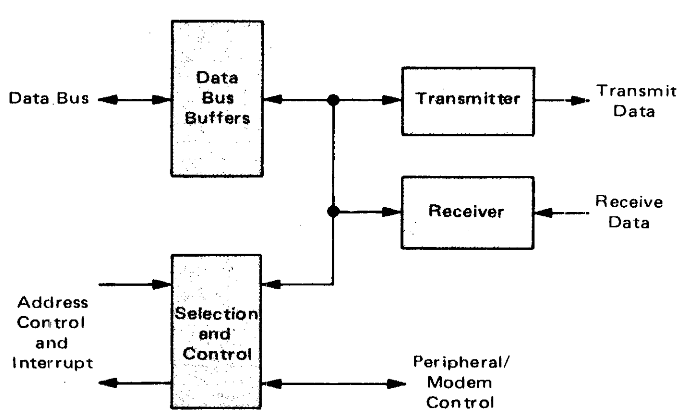


P SUFFIX
PLASTIC PACKAGE
CASE 709

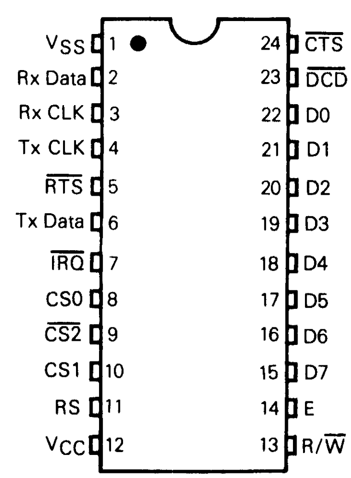


L SUFFIX
CERAMIC PACKAGE
CASE 716

MC6850 ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER BLOCK DIAGRAM



PIN ASSIGNMENT



©MOTOROLA INC., 1981

MAXIMUM RATINGS

Characteristics	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range MC6850, MC68A50, MC68B50 MC6850C, MC68A50C, MC68B50C	T_A	T_L to T_H 0 to 70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Plastic Ceramic Cerdip	θ_{JA}	 120 60 65	°C/W

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \quad (1)$$

Where:

$T_A \equiv$ Ambient Temperature, °C

$\theta_{JA} \equiv$ Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D \equiv P_{INT} + P_{PORT}$

$P_{INT} \equiv I_{CC} \times V_{CC}$, Watts — Chip Internal Power

$P_{PORT} \equiv$ Port Power Dissipation, Watts — User Determined

For most applications $P_{PORT} \ll P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc ±5%, V_{SS} = 0, $T_A = T_L$ to T_H unless otherwise noted.)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage		V_{IH}	$V_{SS}+2.0$	–	V_{CC}	V
Input Low Voltage		V_{IL}	$V_{SS}-0.3$	–	$V_{SS}+0.8$	V
Input Leakage Current (V_{in} = 0 to 5.25 V)	$R/\overline{W}$, CS0, CS1, $\overline{CS2}$, Enable RS, Rx D, Rx C, $\overline{CTS}$, $\overline{DCD}$	I_{in}	–	1.0	2.5	µA
Three-State (Off State) Input Current (V_{in} = 0.4 to 2.4 V)	D0-D7	I_{TSI}	–	2.0	10	µA
Output High Voltage (I_{Load} = −205 µA, Enable Pulse Width < 25 µs) (I_{Load} = −100 µA, Enable Pulse Width < 25 µs)	D0-D7 Tx Data, $\overline{RTS}$	V_{OH}	 $V_{SS}+2.4$ $V_{SS}+2.4$	 – –	 – –	V
Output Low Voltage (I_{Load} = 1.6 mA, Enable Pulse Width < 25 µs)		V_{OL}	–	–	$V_{SS}+0.4$	V
Output Leakage Current (Off State) (V_{OH} = 2.4 V)	$\overline{IRQ}$	I_{LOH}	–	1.0	10	µA
Internal Power Dissipation (Measured at $T_A = T_L$)		P_{INT}	–	300	525	mW
Internal Input Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	D0-D7 E, Tx CLK, Rx CLK, R/W, RS, Rx Data, CS0, CS1, $\overline{CS2}$, $\overline{CTS}$, $\overline{DCD}$	C_{in}	– –	10 7.0	12.5 7.5	pF
Output Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	RTS, Tx Data $\overline{IRQ}$	C_{out}	– –	– –	10 5.0	pF

SERIAL DATA TIMING CHARACTERISTICS

Characteristic		Symbol	MC6850		MC68A50		MC68B50		Unit
			Min	Max	Min	Max	Min	Max	
Data Clock Pulse Width, Low (See Figure 1)	÷16, ÷64 Modes	PW_{CL}	600	–	450	–	280	–	ns
	÷1 Mode		900	–	650	–	500	–	
Data Clock Pulse Width, High (See Figure 2)	÷16, ÷64 Modes	PW_{CH}	600	–	450	–	280	–	ns
	÷1 Mode		900	–	650	–	500	–	
Data Clock Frequency	÷16, ÷64 Modes	f_C	–	0.8	–	1.0	–	1.5	MHz
	÷1 Mode		–	500	–	750	–	1000	kHz
Data Clock-to-Data Delay for Transmitter (See Figure 3)		t_{TDD}	–	600	–	540	–	460	ns
Receive Data Setup Time (See Figure 4)	÷1 Mode	t_{RDS}	250	–	100	–	30	–	ns
Receive Data Hold Time (See Figure 5)	÷1 Mode	t_{RDH}	250	–	100	–	30	–	ns
Interrupt Request Release Time (See Figure 6)		t_{IR}	–	1.2	–	0.9	–	0.7	µs
Request-to-Send Delay Time (See Figure 6)		t_{RTS}	–	560	–	480	–	400	ns
Input Rise and Fall Times (or 10% of the pulse width if smaller)		t_r, t_f	–	1.0	–	0.5	–	0.25	µs

FIGURE 1 — CLOCK PULSE WIDTH, LOW-STATE

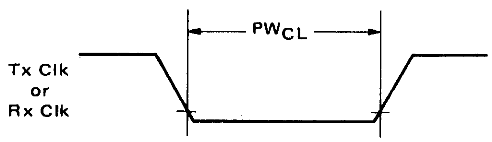


FIGURE 2 — CLOCK PULSE WIDTH, HIGH-STATE

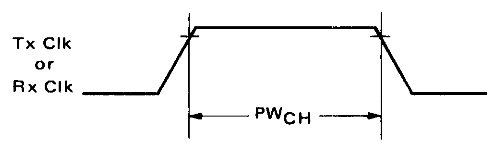


FIGURE 3 — TRANSMIT DATA OUTPUT DELAY

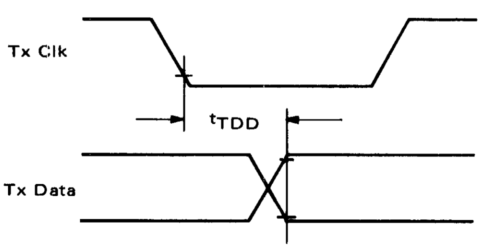


FIGURE 4 — RECEIVE DATA SETUP TIME
(÷1 Mode)

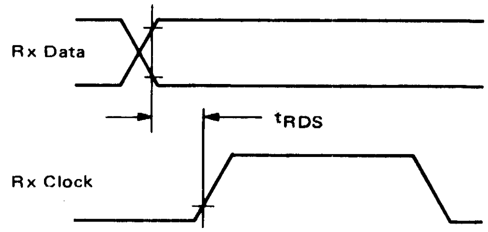


FIGURE 5 — RECEIVE DATA HOLD TIME
(÷1 Mode)

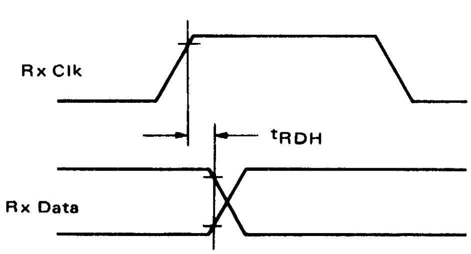
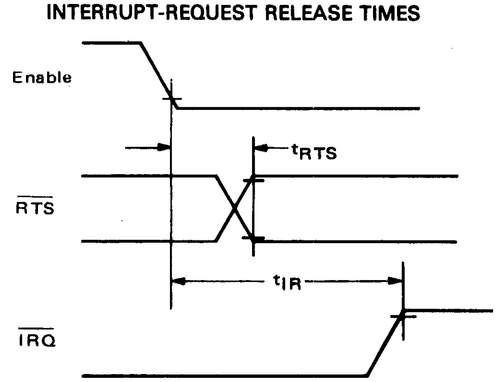


FIGURE 6 — REQUEST-TO-SEND DELAY AND INTERRUPT-REQUEST RELEASE TIMES



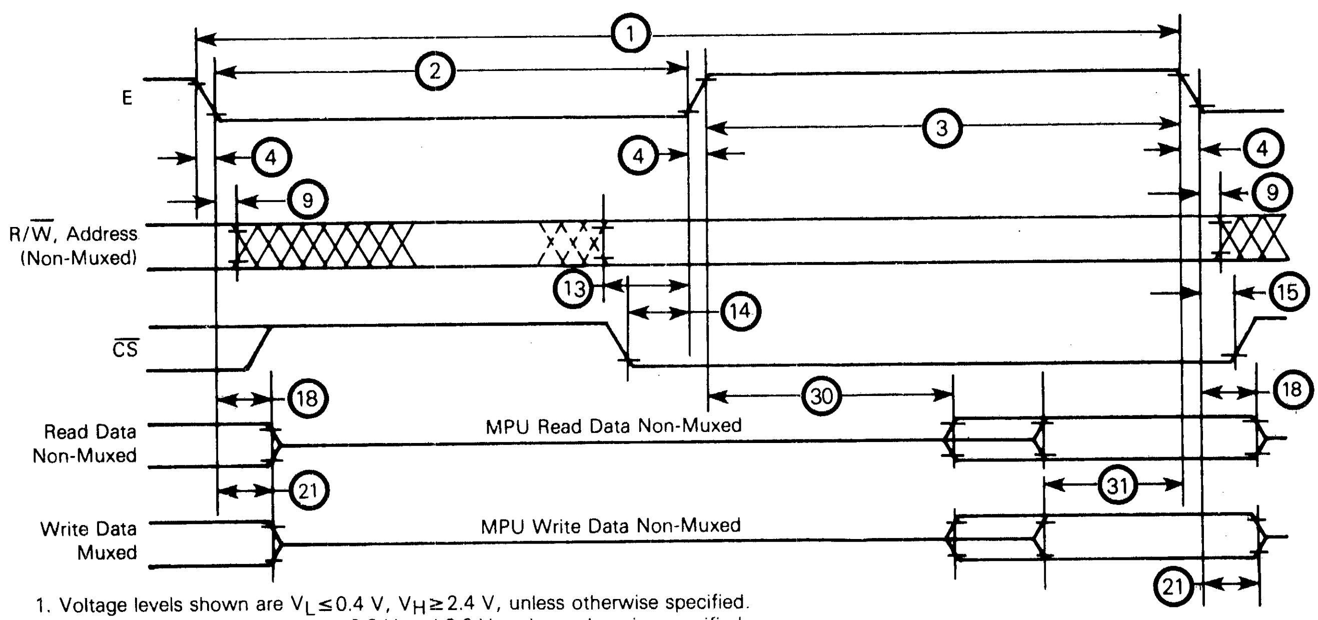
Note: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

BUS TIMING CHARACTERISTICS (See Notes 1 and 2 and Figure 7)

Ident. Number	Characteristic	Symbol	MC6850 Min	MC6850 Max	MC68A50 Min	MC68A50 Max	MC68B50 Min	MC68B50 Max	Unit
1	Cycle Time	t_{cyc}	1.0	10	0.67	10	0.5	10	μs
2	Pulse Width, E Low	PW_{EL}	430	9500	280	9500	210	9500	ns
3	Pulse Width, E High	PW_{EH}	450	9500	280	9500	220	9500	ns
4	Clock Rise and Fall Time	t_r, t_f	–	25	–	25	–	20	ns
9	Address Hold Time	t_{AH}	10	–	10	–	10	–	ns
13	Address Setup Time Before E	t_{AS}	80	–	60	–	40	–	ns
14	Chip Select Setup Time Before E	t_{CS}	80	–	60	–	40	–	ns
15	Chip Select Hold Time	t_{CH}	10	–	10	–	10	–	ns
18	Read Data Hold Time	t_{DHR}	20	50*	20	50*	20	50*	ns
21	Write Data Hold Time	t_{DHW}	10	–	10	–	10	–	ns
30	Output Data Delay Time	t_{DDR}	–	290	–	180	–	150	ns
31	Input Data Setup Time	t_{DSW}	165	–	80	–	60	–	ns

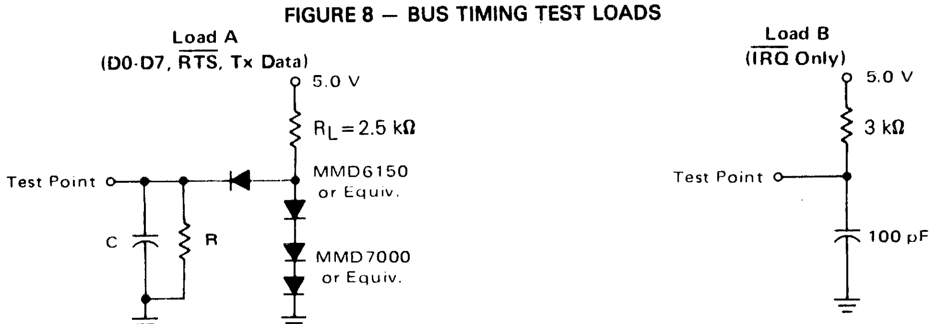
*The data bus output buffers are no longer sourcing or sinking current by t_{DHR}max (High Impedance).

FIGURE 7 — BUS TIMING CHARACTERISTICS



1. Voltage levels shown are $V_L \le 0.4$ V, $V_H \ge 2.4$ V, unless otherwise specified.
2. Measurement points shown are 0.8 V and 2.0 V, unless otherwise specified.

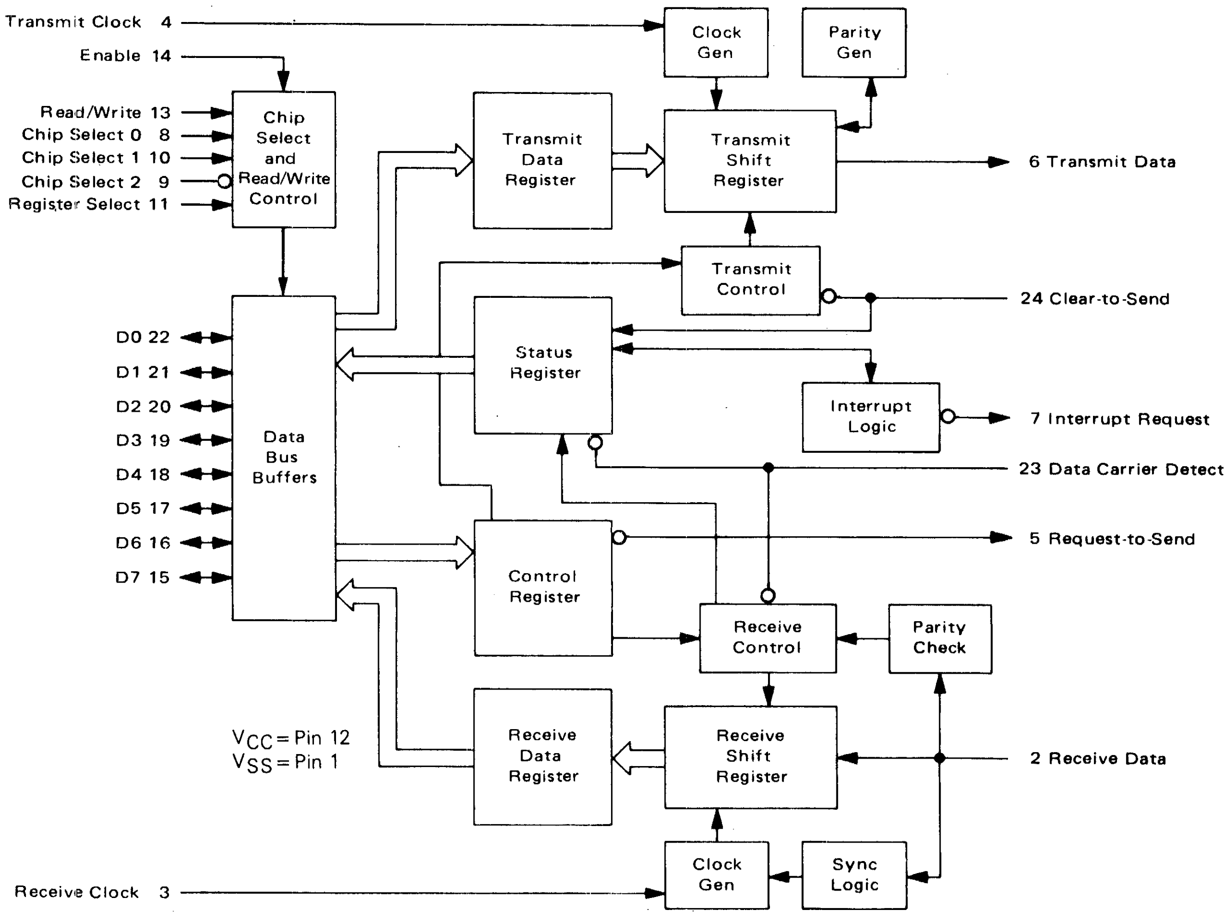
FIGURE 8 — BUS TIMING TEST LOADS



C = 130 pF for D0-D7
= 30 pF for RTS̅ and Tx Data

R = 11.7 kΩ for D0-D7
= 24 kΩ for RTS̅ and Tx Data

FIGURE 9 — EXPANDED BLOCK DIAGRAM



DEVICE OPERATION

At the bus interface, the ACIA appears as two addressable memory locations. Internally, there are four registers: two read-only and two write-only registers. The read-only registers are Status and Receive Data; the write-only registers are Control and Transmit Data. The serial interface consists of serial input and output lines with independent clocks, and three peripheral/modem control lines.

POWER ON/MASTER RESET

The master reset (CR0, CR1) should be set during system initialization to insure the reset condition and prepare for programming the ACIA functional configuration when the communications channel is required. During the first master reset, the $\overline{IRQ}$ and $\overline{RTS}$ outputs are held at level 1. On all other master resets, the $\overline{RTS}$ output can be programmed high or low with the $\overline{IRQ}$ output held high. Control bits CR5 and CR6 should also be programmed to define the state of $\overline{RTS}$ whenever master reset is utilized. The ACIA also contains internal power-on reset logic to detect the power line turn-on transition and hold the chip in a reset state to prevent erroneous output transitions prior to initialization. This circuitry depends on clean power turn-on transitions. The power-on reset is released by means of the bus-programmed master reset which must be applied prior to operating the ACIA. After master resetting the ACIA, the programmable Control Register can be set for a number of options such as variable clock divider ratios, variable word length, one or two stop bits, parity (even, odd, or none), etc.

TRANSMIT

A typical transmitting sequence consists of reading the ACIA Status Register either as a result of an interrupt or in the ACIA's turn in a polling sequence. A character may be written into the Transmit Data Register if the status read operation has indicated that the Transmit Data Register is empty. This character is transferred to a Shift Register where it is serialized and transmitted from the Transmit Data output preceded by a start bit and followed by one or two stop bits. Internal parity (odd or even) can be optionally added to the character and will occur between the last data bit and the first stop bit. After the first character is written in the Data Register, the Status Register can be read again to check for a Transmit Data Register Empty condition and current peripheral status. If the register is empty, another character can be loaded for transmission even though the first character is in the process of being transmitted (because of double buffering). The second character will be automatically transferred into the Shift Register when the first character transmission is completed. This sequence continues until all the characters have been transmitted.

RECEIVE

Data is received from a peripheral by means of the Receive Data input. A divide-by-one clock ratio is provided for an externally synchronized clock (to its data) while the divide-by-16 and 64 ratios are provided for internal synchronization. Bit synchronization in the divide-by-16 and 64 modes is initiated by the detection of 8 or 32 low samples on the receive line in the divide-by-16 and 64 modes respectively. False start bit deletion capability insures that a full half bit of a start bit has been received before the internal clock is synchronized to the bit time. As a character is being received, parity (odd or even) will be checked and the error indication will be available in the Status Register along with framing error, overrun error, and Receive Data Register full. In a typical receiving sequence, the Status Register is read to determine if a character has been received from a peripheral. If the Receiver Data Register is full, the character is placed on the 8-bit ACIA bus when a Read Data command is received from the MPU. When parity has been selected for a 7-bit word (7 bits plus parity), the receiver strips the parity bit (D7=0) so that data alone is transferred to the MPU. This feature reduces MPU programming. The Status Register can continue to be read to determine when another character is available in the Receive Data Register. The receiver is also double buffered so that a character can be read from the data register as another character is being received in the shift register. The above sequence continues until all characters have been received.

INPUT/OUTPUT FUNCTIONS

ACIA INTERFACE SIGNALS FOR MPU

The ACIA interfaces to the M6800 MPU with an 8-bit bidirectional data bus, three chip select lines, a register select line, an interrupt request line, read/write line, and enable line. These signals permit the MPU to have complete control over the ACIA.

ACIA Bidirectional Data (D0-D7) – The bidirectional data lines (D0-D7) allow for data transfer between the ACIA and the MPU. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs an ACIA read operation.

ACIA Enable (E) – The Enable signal, E, is a high-impedance TTL-compatible input that enables the bus input/output data buffers and clocks data to and from the ACIA. This signal will normally be a derivative of the MC6800 $\phi 2$ Clock or MC6809 E clock.

Read/Write (R/$\overline{W}$) – The Read/Write line is a high-impedance input that is TTL compatible and is used to control the direction of data flow through the ACIA's input/output data bus interface. When Read/Write is high (MPU Read cycle), ACIA output drivers are turned on and a selected register is read. When it is low, the ACIA output drivers are turned off and the MPU writes into a selected register. Therefore, the Read/Write signal is used to select read-only or write-only registers within the ACIA.

Chip Select (CS0, CS1, $\overline{\text{CS2}}$) – These three high-impedance TTL-compatible input lines are used to address the ACIA. The ACIA is selected when CS0 and CS1 are high and $\overline{\text{CS2}}$ is low. Transfers of data to and from the ACIA are then performed under the control of the Enable Signal, Read/Write, and Register Select.

Register Select (RS) – The Register Select line is a high-impedance input that is TTL compatible. A high level is used to select the Transmit/Receive Data Registers and a low level the Control/Status Registers. The Read/Write signal line is used in conjunction with Register Select to select the read-only or write-only register in each register pair.

Interrupt Request ($\overline{\text{IRQ}}$) – Interrupt Request is a TTL-compatible, open-drain (no internal pullup), active low output that is used to interrupt the MPU. The $\overline{\text{IRQ}}$ output remains low as long as the cause of the interrupt is present and the appropriate interrupt enable within the ACIA is set. The IRQ status bit, when high, indicates the $\overline{\text{IRQ}}$ output is in the active state.

Interrupts result from conditions in both the transmitter and receiver sections of the ACIA. The transmitter section causes an interrupt when the Transmitter Interrupt Enabled condition is selected (CR5•$\overline{\text{CR6}}$), and the Transmit Data Register Empty (TDRE) status bit is high. The TDRE status bit indicates the current status of the Transmitter Data Register except when inhibited by Clear-to-Send ($\overline{\text{CTS}}$) being high or the ACIA being maintained in the Reset condition. The interrupt is cleared by writing data into the Transmit Data Register. The interrupt is masked by disabling the Transmitter Interrupt via CR5 or CR6 or by the loss of $\overline{\text{CTS}}$ which inhibits the TDRE status bit. The Receiver section causes an interrupt when the Receiver Interrupt Enable is set and the Receive Data Register Full (RDRF) status bit is high, an Overrun has occurred, or Data Carrier Detect ($\overline{\text{DCD}}$) has gone high. An interrupt resulting from the RDRF status bit can be cleared by reading data or resetting the ACIA. Interrupts caused by Overrun or loss of $\overline{\text{DCD}}$ are cleared by reading the status register after the error condition has occurred and then reading the Receive Data Register or resetting the ACIA. The receiver interrupt is masked by resetting the Receiver Interrupt Enable.

CLOCK INPUTS

Separate high-impedance TTL-compatible inputs are provided for clocking of transmitted and received data. Clock frequencies of 1, 16, or 64 times the data rate may be selected.

Transmit Clock (Tx CLK) – The Transmit Clock input is used for the clocking of transmitted data. The transmitter initiates data on the negative transition of the clock.

Receive Clock (Rx CLK) – The Receive Clock input is used for synchronization of received data. (In the ÷1 mode, the clock and data must be synchronized externally.) The receiver samples the data on the positive transition of the clock.

SERIAL INPUT/OUTPUT LINES

Receive Data (Rx Data) — The Receive Data line is a high-impedance TTL-compatible input through which data is received in a serial format. Synchronization with a clock for detection of data is accomplished internally when clock rates of 16 or 64 times the bit rate are used.

Transmit Data (Tx Data) — The Transmit Data output line transfers serial data to a modem or other peripheral.

PERIPHERAL/MODEM CONTROL

The ACIA includes several functions that permit limited control of a peripheral or modem. The functions included are Clear-to-Send, Request-to-Send and Data Carrier Detect.

Clear-to-Send ($\overline{CTS}$) — This high-impedance TTL-compatible input provides automatic control of the transmitting end of a communications link via the modem Clear-to-Send active low output by inhibiting the Transmit Data Register Empty (TDRE) status bit.

Request-to-Send ($\overline{RTS}$) — The Request-to-Send output enables the MPU to control a peripheral or modem via the data bus. The RTS output corresponds to the state of the Control Register bits CR5 and CR6. When CR6=0 or both CR5 and CR6=1, the $\overline{RTS}$ output is low (the active state). This output can also be used for Data Terminal Ready (DTR).

Data Carrier Detect ($\overline{DCD}$) — This high-impedance TTL-compatible input provides automatic control, such as in the receiving end of a communications link by means of a modem Data Carrier Detect output. The $\overline{DCD}$ input inhibits and initializes the receiver section of the ACIA when high. A low-to-high transition of the Data Carrier Detect initiates an interrupt to the MPU to indicate the occurrence of a loss of carrier when the Receive Interrupt Enable bit is set. The Rx CLK must be running for proper $\overline{DCD}$ operation.

ACIA REGISTERS

The expanded block diagram for the ACIA indicates the internal registers on the chip that are used for the status, control, receiving, and transmitting of data. The content of each of the registers is summarized in Table 1.

TRANSMIT DATA REGISTER (TDR)

Data is written in the Transmit Data Register during the negative transition of the enable (E) when the ACIA has been addressed with RS high and R/$\overline{W}$ low. Writing data into the register causes the Transmit Data Register Empty bit in the Status Register to go low. Data can then be transmitted. If the transmitter is idling and no character is being transmitted, then the transfer will take place within 1-bit time of the trailing edge of the Write command. If a character is being transmitted, the new data character will commence as soon as the previous character is complete. The transfer of data causes the Transmit Data Register Empty (TDRE) bit to indicate empty.

RECEIVE DATA REGISTER (RDR)

Data is automatically transferred to the empty Receive Data Register (RDR) from the receiver deserializer (a shift register) upon receiving a complete character. This event causes the Receive Data Register Full bit (RDRF) in the status buffer to go high (full). Data may then be read through the bus by addressing the ACIA and selecting the Receive Data Register with RS and R/$\overline{W}$ high when the ACIA is enabled. The non-destructive read cycle causes the RDRF bit to be cleared to empty although the data is retained in the RDR. The status is maintained by RDRF as to whether or not the data is current. When the Receive Data Register is full, the automatic transfer of data from the Receiver Shift Register to the Data Register is inhibited and the RDR contents remain valid with its current status stored in the Status Register.

TABLE 1 — DEFINITION OF ACIA REGISTER CONTENTS

Data Bus Line Number	Buffer Address			
	$RS \bullet \overline{R/\overline{W}}$ Transmit Data Register	$RS \bullet R/\overline{W}$ Receive Data Register	$\overline{RS} \bullet \overline{R/\overline{W}}$ Control Register	$\overline{RS} \bullet R/\overline{W}$ Status Register
	(Write Only)	(Read Only)	(Write Only)	(Read Only)
0	Data Bit 0*	Data Bit 0	Counter Divide Select 1 (CR0)	Receive Data Register Full (RDRF)
1	Data Bit 1	Data Bit 1	Counter Divide Select 2 (CR1)	Transmit Data Register Empty (TDRE)
2	Data Bit 2	Data Bit 2	Word Select 1 (CR2)	Data Carrier Detect ($\overline{DCD}$)
3	Data Bit 3	Data Bit 3	Word Select 2 (CR3)	Clear-to-Send ($\overline{CTS}$)
4	Data Bit 4	Data Bit 4	Word Select 3 (CR4)	Framing Error (FE)
5	Data Bit 5	Data Bit 5	Transmit Control 1 (CR5)	Receiver Overrun (OVRN)
6	Data Bit 6	Data Bit 6	Transmit Control 2 (CR6)	Parity Error (PE)
7	Data Bit 7***	Data Bit 7**	Receive Interrupt Enable (CR7)	Interrupt Request ($\overline{IRQ}$)

* Leading bit = LSB = Bit 0
** Data bit will be zero in 7-bit plus parity modes.
*** Data bit is "don't care" in 7-bit plus parity modes.

CONTROL REGISTER

The ACIA Control Register consists of eight bits of write-only buffer that are selected when RS and R/$\overline{W}$ are low. This register controls the function of the receiver, transmitter, interrupt enables, and the Request-to-Send peripheral/modem control output.

Counter Divide Select Bits (CR0 and CR1) — The Counter Divide Select Bits (CR0 and CR1) determine the divide ratios utilized in both the transmitter and receiver sections of the ACIA. Additionally, these bits are used to provide a master reset for the ACIA which clears the Status Register (except for external conditions on $\overline{CTS}$ and $\overline{DCD}$) and initializes both the receiver and transmitter. Master reset does not affect other Control Register bits. Note that after power-on or a power fail/restart, these bits must be set high to reset the ACIA. After resetting, the clock divide ratio may be selected. These counter select bits provide for the following clock divide ratios:

CR1	CR0	Function
0	0	÷1
0	1	÷16
1	0	÷64
1	1	Master Reset

Word Select Bits (CR2, CR3, and CR4) — The Word Select bits are used to select word length, parity, and the number of stop bits. The encoding format is as follows:

CR4	CR3	CR2	Function
0	0	0	7 Bits + Even Parity + 2 Stop Bits
0	0	1	7 Bits + Odd Parity + 2 Stop Bits
0	1	0	7 Bits + Even Parity + 1 Stop Bit
0	1	1	7 Bits + Odd Parity + 1 Stop Bit
1	0	0	8 Bits + 2 Stop Bits
1	0	1	8 Bits + 1 Stop Bit
1	1	0	8 Bits + Even parity + 1 Stop Bit
1	1	1	8 Bits + Odd Parity + 1 Stop Bit

Word length, Parity Select, and Stop Bit changes are not buffered and therefore become effective immediately.

Transmitter Control Bits (CR5 and CR6) — Two Transmitter Control bits provide for the control of the interrupt from the Transmit Data Register Empty condition, the Request-to-Send ($\overline{RTS}$) output, and the transmission of a Break level (space). The following encoding format is used:

CR6	CR5	Function
0	0	$\overline{RTS}$ = low, Transmitting Interrupt Disabled.
0	1	$\overline{RTS}$ = low, Transmitting Interrupt Enabled.
1	0	$\overline{RTS}$ = high, Transmitting Interrupt Disabled.
1	1	$\overline{RTS}$ = low, Transmits a Break level on the Transmit Data Output. Transmitting Interrupt Disabled.

Receive Interrupt Enable Bit (CR7) — The following interrupts will be enabled by a high level in bit position 7 of the Control Register (CR7): Receive Data Register Full, Overrun, or a low-to-high transition on the Data Carrier Detect ($\overline{DCD}$) signal line.

STATUS REGISTER

Information on the status of the ACIA is available to the MPU by reading the ACIA Status Register. This read-only register is selected when RS is low and R/$\overline{W}$ is high. Information stored in this register indicates the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral/modem status inputs of the ACIA.

Receive Data Register Full (RDRF), Bit 0 — Receive Data Register Full indicates that received data has been transferred to the Receive Data Register. RDRF is cleared after an MPU read of the Receive Data Register or by a master reset. The cleared or empty state indicates that the contents of the Receive Data Register are not current. Data Carrier Detect being high also causes RDRF to indicate empty.

Transmit Data Register Empty (TDRE), Bit 1 — The Transmit Data Register Empty bit being set high indicates that the Transmit Data Register contents have been transferred and that new data may be entered. The low state indicates that the register is full and that transmission of a new character has not begun since the last write data command.

Data Carrier Detect ($\overline{DCD}$), Bit 2 — The Data Carrier Detect bit will be high when the $\overline{DCD}$ input from a modem has gone high to indicate that a carrier is not present. This bit going high causes an Interrupt Request to be generated when the Receive Interrupt Enable is set. It remains high after the $\overline{DCD}$ input is returned low until cleared by first reading the Status Register and then the Data Register or until a master reset occurs. If the $\overline{DCD}$ input remains high after read status and read data or master reset has occurred, the interrupt is cleared, the $\overline{DCD}$ status bit remains high and will follow the $\overline{DCD}$ input.

Clear-to-Send ($\overline{CTS}$), Bit 3 — The Clear-to-Send bit indicates the state of the Clear-to-Send input from a modem. A low $\overline{CTS}$ indicates that there is a Clear-to-Send from the modem. In the high state, the Transmit Data Register Empty bit is inhibited and the Clear-to-Send status bit will be high. Master reset does not affect the Clear-to-Send status bit.

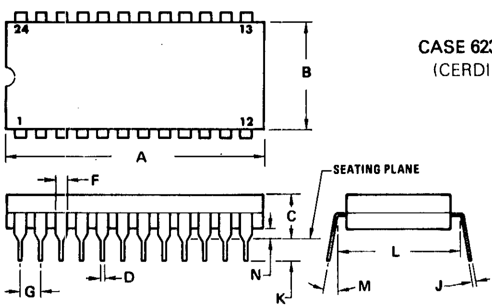
Framing Error (FE), Bit 4 — Framing error indicates that the received character is improperly framed by a start and a stop bit and is detected by the absence of the first stop bit. This error indicates a synchronization error, faulty transmission, or a break condition. The framing error flag is set or reset during the receive data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

Receiver Overrun (OVRN), Bit 5 — Overrun is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receive Data Register (RDR) prior to subsequent characters being received. The overrun condition begins at the midpoint of the last bit of the second character received in succession without a read of the RDR having occurred. The Overrun does not occur in the Status Register until the valid character prior to Overrun has been read. The RDRF bit remains set until the Overrun is reset. Character synchronization is maintained during the Overrun condition. The Overrun indication is reset after the reading of data from the Receive Data Register or by a Master Reset.

Parity Error (PE), Bit 6 — The parity error flag indicates that the number of highs (ones) in the character does not agree with the preselected odd or even parity. Odd parity is defined to be when the total number of ones is odd. The parity error indication will be present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver partiy check results are inhibited.

Interrupt Request ($\overline{\text{IRQ}}$), Bit 7 — The $\overline{\text{IRQ}}$ bit indicates the state of the $\overline{\text{IRQ}}$ output. Any interrupt condition with its applicable enable will be indicated in this status bit. Anytime the $\overline{\text{IRQ}}$ output is low the $\overline{\text{IRQ}}$ bit will be high to indicate the interrupt or service request status. $\overline{\text{IRQ}}$ is cleared by a read operation to the Receive Data Register or a write operation to the Transmit Data Register.

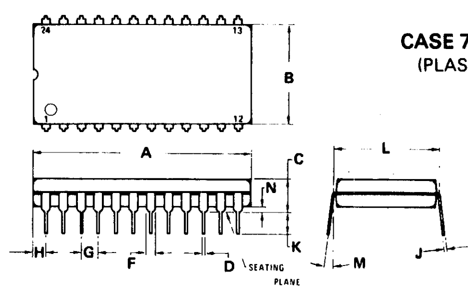
PACKAGE DIMENSIONS



CASE 623-03
(CERDIP)

DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	31.24	32.77	1.230	1.290
B	12.70	15.49	0.500	0.610
C	4.06	5.59	0.160	0.220
D	0.41	0.51	0.016	0.020
F	1.27	1.52	0.050	0.060
G	2.54 BSC		0.100 BSC	
J	0.20	0.30	0.008	0.012
K	2.29	4.06	0.090	0.160
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.27	0.020	0.050

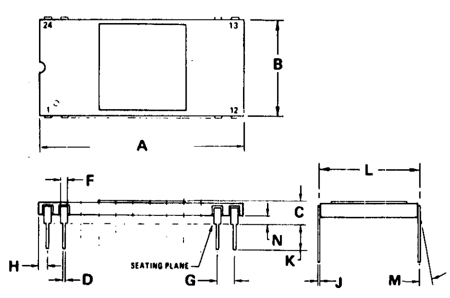
NOTES:
1. DIM "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
2. LEADS WITHIN 0.13 mm (0.005) RADIUS OF TRUE POSITION AT SEATING PLANE AT MAXIMUM MATERIAL CONDITION. (WHEN FORMED PARALLEL)



CASE 709-02
(PLASTIC)

DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	31.37	32.13	1.235	1.265
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.03	0.065	0.080
J	0.20	0.38	0.008	0.015
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.



CASE 716-06
(CERAMIC)

NOTE:
1. LEADS TRUE POSITIONED WITHIN 0.25mm (0.010) DIA (AT SEATING PLANE) AT MAXIMUM MATERIAL CONDITION.
2. DIM "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.

DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	27.64	30.99	1.088	1.220
B	14.73	15.34	0.580	0.604
C	2.67	4.32	0.105	0.170
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.30	0.008	0.012
K	2.54	4.57	0.100	0.180
L	14.99	15.49	0.590	0.610
M	–	10°	–	10°
N	1.02	1.52	0.040	0.060

ORDERING INFORMATION

MC68A50CP

Motorola Integrated Circuit
M6800 Family
Blanks = 1.0 MHz
A = 1.5 MHz
B = 2.0 MHz
Device Designation
In M6800 Family
Temperature Range
Blank = 0° → +70°C
C = −40° → +85°C
Package
P = Plastic
S = Cerdip
L = Ceramic

BETTER PROGRAM

Better program processing is available on all types listed. Add suffix letters to part number.

Level 1 add "S" Level 2 add "D" Level 3 add "DS"

Level 1 "S" = 10 Temp Cycles — (−25 to 150°C);
Hi Temp testing at T_A max.
Level 2 "D" = 168 Hour Burn-in at 125°C
Level 3 "DS" = Combination of Level 1 and 2.

Speed	Device	Temperature Range
1.0 MHz	MC6850P,L,S MC6850CP,CL,CS	0 to 70°C −40 to +85°C
1.5 MHz	MC68A50P,L,S MC68A50CP,CL,CS	0 to +70°C −40 to +85°C
2.0 MHz	MC68B50P,L,S	0 to +70°C

Motorola reserves the right to make changes to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

MOTOROLA Semiconductor Products Inc.

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

Appendix D MB8877A Disk Controller Registers/Instructions

The following specifications on the MB8877A disk controller were furnished in full by Fujitsu Microelectronics:

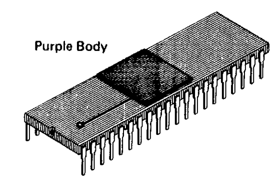
FLOPPY DISK FORMATTER/ CONTROLLER

MB 8876A MB 8877A

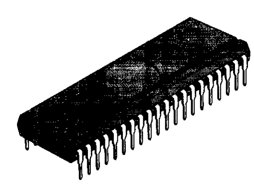
March 1982
Edition 1.0

FLOPPY DISK FORMATTER/CONTROLLER

- One-Chip Floppy Fisk Formatter/Controller.
- Applicable to Single Density Floppy Disk, Double Density Floppy Disk and Mini Floppy Disk.
- Interfaceable to 8-bit Microprocessor.
  MB 8876A: Negative-logic 8-bit Data Bus.
  MB 8877A: Positive-logic 8-bit Data Bus.
- IBM Compatible Sector Format.
- Automatic Track Seeking and Verification.
- Both Single and Double Density Formats.
  a) Single Density in IBM 3740 Format and FM Recording.
  b) Double Density in IBM System-34 Format and MFM Recording.
- Programmable Single Sector/Multiple Sectors/Entire Track Read Operation.
- Programmable Single Sector/Multiple Sectors/Entire Track Wire Operation.
- Programmable Sector Length.
- Programmable Side Compare Function.
- Programmable Head Step Rate.
- Programmable Head Engage/Head Settle Time.
- Double Buffered Data I/O.
- DMA Data Transfer Capability.
- Write Precompensation Capability.
- All TTL Compatible I/O.
- Single +5V Power Supply.
- N-Channel E/D MOS Technology.
- Standard 40-pin Dual-In Line Package.
- MB 8876A: Upward Compatible with Western Digital FD1791-02.
  MB 8877A: Upward Compatible with Western Digital FD1793-02.

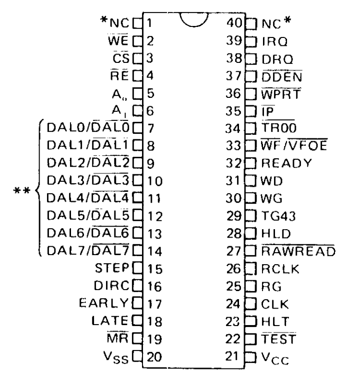


CERAMIC PACKAGE
DIP-40C-A01



PLASTIC PACKAGE
DIP-40P-M01

PIN ASSIGNMENT



* NC: Non Connection
** MB 8876A: Negative Logic; MB 8877A: Positive Logic

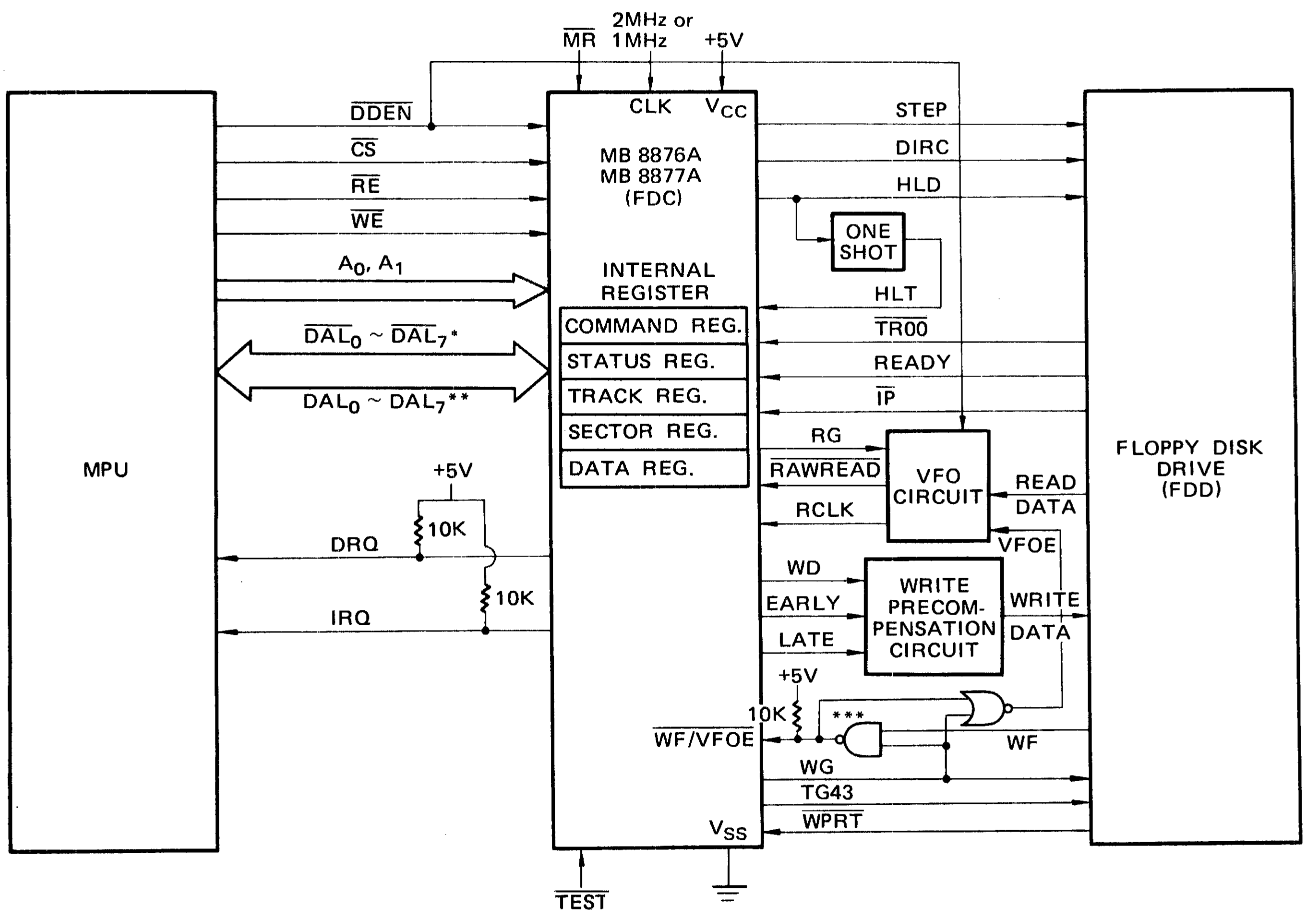
ABSOLUTE MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Voltage on any pin to V_{SS}	V_{CC}, V_I, V_O	-0.3 to +7.0	V
Operating Temperature	T_{OP}	0 to 70	°C
Storage Temperature	T_{stg}	-55 to +150	°C

NOTE: Permanent device damage may occur if ABSOLUTE MAXIMUM RATINGS are exceeded. Functional operation should be restricted to the conditions as detailed in the operational sections of this data sheet. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit.

Fig. 1 — MB 8876A/MB 8877A SYSTEM CONSTRUCTURE



* MB 8876A: Negative Logic
** MB 8877A: Positive Logic
*** Open Collector Output

GUARANTEED OPERATING CONDITIONS
(Referenced to V_{SS})

Parameter	Symbol	Value			Unit	Operating Temperature
		Min.	Typ.	Max.		
Supply Voltage	V_{CC}	4.75	5.00	5.25	V	0°C to +70°C
	V_{SS}		0			
Input High Voltage	V_{IH}	2.0		V_{CC}	V	
Input Low Voltage	V_{IL}	-0.3		0.8	V	

PIN DESCRIPTIONS

Pin No.	Symbol	Pin Name	I/O	Description
20	V_{SS}	Power Supply	I	Ground (GND)
21	V_{CC}			+5V DC supply
24	CLK	Clock	I	2-MHz fixed frequency clock signal (1-MHz for mini-floppy disk)
19	$\overline{MR}$	Master Reset	I	Signal for resetting the FDC
22	$\overline{TEST}$	Test	I	Signal for setting the FDC into a test mode
1, 40	NC	Non Connection	—	These pins are not used.

MPU INTERFACE PINS

37	$\overline{DDEN}$	Double Density	I	Signal for selecting a FDC operation mode: When $\overline{DDEN}$ = 0, the double density operation mode is selected. When $\overline{DDEN}$ = 1, the single density operation mode is selected. This input must be fixed while the FDC is in busy state.
3	$\overline{CS}$	Chip Select	I	Signal for controlling the DALs: When $\overline{CS}$ = 0, the DALs are activated and data transfer between the FDC and the MPU is enabled. When $\overline{CS}$ = 1, the DALs are in high impedance state and data transfer is inhibited. (i.e., $\overline{RE}$ and $\overline{WE}$ are ignored.)
4	$\overline{RE}$	Read Enable	I	Strobe signal provided when data is read from internal registers: When $\overline{CS}$ = $\overline{RE}$ = 0, data can be read from internal registers.
2	$\overline{WE}$	Write Enable	I	Strobe signal provided when data is written into internal registers: When $\overline{CS}$ = $\overline{WE}$ = 0, data can be written into internal registers.
5, 6	A_0, A_1	Register Select Line	I	Signal for addressing an internal register among Command Register (CR), Status Register (STR), Track Register (TR), Sector Register (SCR) and Data Register (DR): Refer to table of REGISTER SELECTION (p. 6)
7 ~ 14	$\overline{DAL_0} \sim \overline{DAL_7}$ $DAL_0 \sim DAL_7$	Data Access Line	I/O	8-bit bidirectional bus for transferring 8-bit data between the FDC and the MPU. MB 8876A: negative logic / MB 8877A: positive logic
38	DRQ	Data Request	O	Signal for informing the MPU of a DR status: Read operation: DRQ = 1 shows the DR is filled with a 8-bit data from a disk, and the FDC is requesting for the MPU to read the data. Write operation: DRQ = 1 shows the DR is empty, and the FDC is requesting for the MPU to write the next data into the DR.
39	IRQ	Interrupt Request	O	Interrupt signal to the MPU: IRQ is set when a Command is completed or the TYPE IV Command is executed. IRQ is reset when the next Command is written or the STR is read.

FLOPPY DISK INTERFACE PINS

Disk Head Control Signal

15	STEP	Step Move	O	Step pulse signal for moving a disk head.
16	DIRC	Direction	O	Signal for indicating a direction of disk head moving to the FDD: DIRC = 0 shows the head moves toward outside. DIRC = 1 shows the head moves toward inside.
28	HLD	Head Load	O	Signal for loading a disk head: When HLD = 1, the head is engaged on the disk. When HLD = 0, the head is released from the disk.

Disk Head Control Signal (cont'd)

Pin No.	Symbol	Pin Name	I/O	Description
23	HLT	Head Load Timing	I	Signal for informing a disk head status: HLT = 1 shows a disk head is in an engaged state. HLT is set when a disk head has been settled or a head settle time pre-determined by one shot circuit has elapsed after HLD = 1.
34	$\overline{\text{TR00}}$	Track 00	I	Signal for informing whether a disk head is positioned on Track No. 00 or not: $\overline{\text{TR00}}$ = 0 shows Track No. 00 is detected during track seaking operation.
32	READY	Ready	I	Signal for informing the FDC of a disk drive status: READY = 1 shows the disk drive is ready for operation, and only when READY = 1, read/write operation for disk can be executed. READY = 0 shows the disk drive is not ready, and neither read/write operation can not be executed. However, seek operation is excuted regardless of this signal.
35	$\overline{\text{IP}}$	Index Pulse	I	Signal for informing the FDC of an index hole of disk being detected in the FDD.

Disk Read Operation Signal

25	RG	Read Gate	O	Signal for informing synchronization between RCLK and $\overline{\text{RAWREAD}}$ to an external VFO circuit: RG = 1 shows the FDC has found out a SYNC byte during disk reading operation.
26	RCLK	Read Clock	I	A data window signal which is generated in an external VFO circuit out of Read Data.
27	$\overline{\text{RAWREAD}}$	Raw Read	I	A raw read data signal transferred from the FDD.

Disk Write Operation Signal

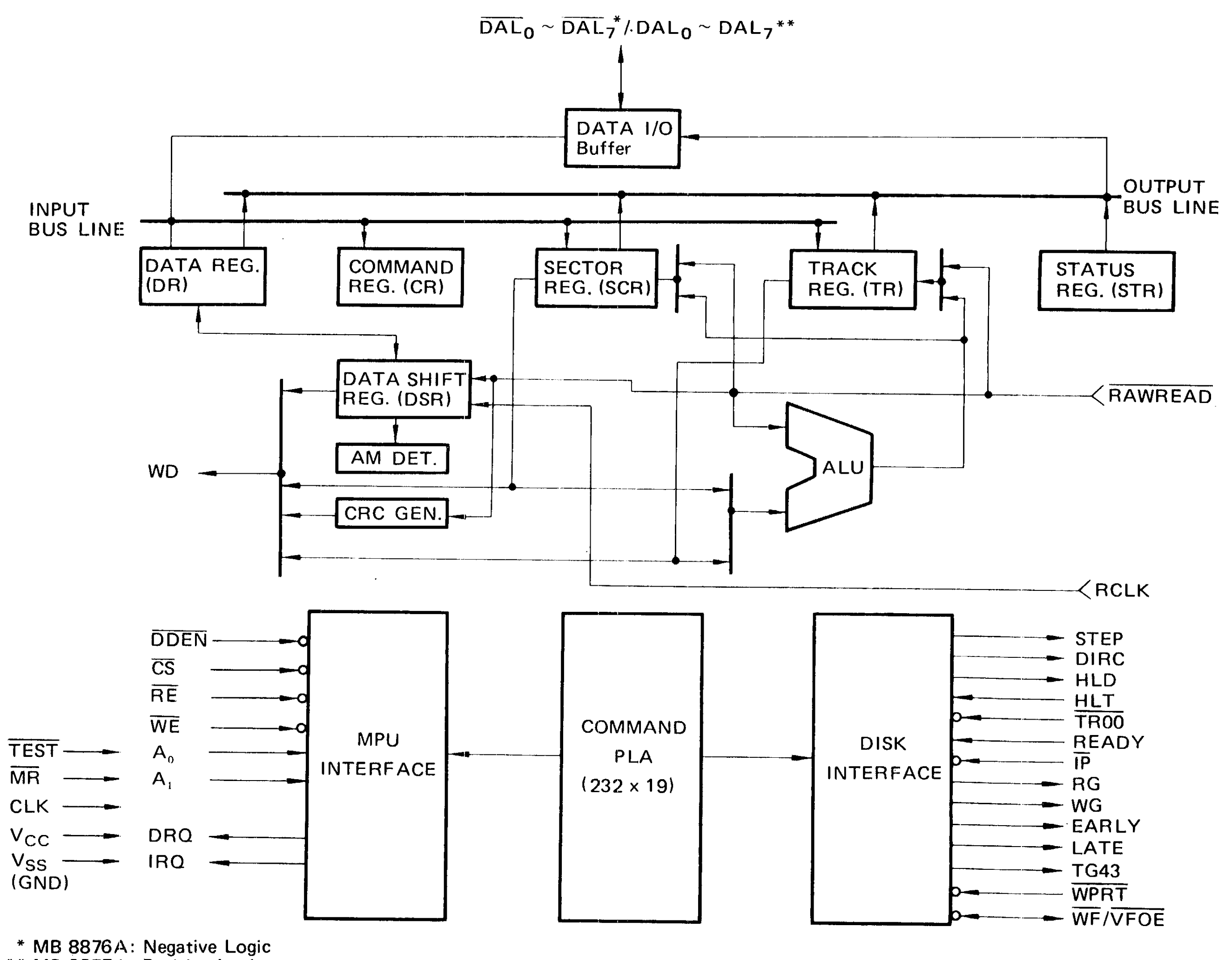
30	WG	Write Gate	O	Signal for indicating data is being written into a disk.
17	EARLY	Early Shift	O	Signal for indicating early pre-compensation of data write timing to a disk: EARLY = 1 shows a serial data to be transmitted via the WD pin to a disk must be shifted earlier.
18	LATE	Late Shift	O	Signal for indicating later pre-compensation of data write timing to a disk: LATE = 1 shows a serial data to be transmitted via the WD pin to a disk must be shifted later.
31	WD	Write Data	O	A write data signal transferred to the FDD.
29	TG43	Track Greater Than 43	O	Signal for indicating a head position of a disk: TG43 = 1 shows the head is located on any of Track No. 44 thru 76. TG43 = 0 shows the head is located on any of Track No. 0 thru 43.
33	$\overline{\text{WF}}/\overline{\text{VFOE}}$	Write Fault/ Variable Frequency Oscillator Enable	I/O	Input signal for informing a fault is detected during write operation for a disk (during WG = 1). Output signal forinforming the FDC is reading a disk (during WG = 0).
34	$\overline{\text{WPRT}}$	Write Protect	I	Signal for inhibiting write operation for disk.

REGISTER SELECTION

Chip Select	Address		Selected Register		Data Access Line Status
$\overline{CS}$	A_1	A_0	Read Mode ($\overline{RE}$ = 0)	Write Mode ($\overline{WE}$ = 0)	$\overline{DAL_7} \sim \overline{DAL_0}$ $DAL_7 \sim DAL_0$
1	*	*	Deselected	Deselected	High Impedance
0	0	0	Status Register (STR)	Command Register (CR)	Enabled
0	0	1	Track Register (TR)	Track Register (TR)	Enabled
0	1	0	Sector Register (SCR)	Sector Register (SCR)	Enabled
0	1	1	Data Register (DR)	Data Register (DR)	Enabled

*: Don't care

Fig. 2 – MB 8876A/MB 8877A BLOCK DIAGRAM



* MB 8876A: Negative Logic
** MB 8877A: Positive Logic

FUNCTIONAL BLOCK DESCRIPTIONS

INTERNAL REGISTERS

Command Register (CR)

An 8-bit write-only register in which a Command is written from the MPU:
A Command must be loaded in the CR during BUSY = 0. (The BUSY flag is Bit 0 of the STR.) After completion of execution for a loaded Command, such a completion is informed to the MPU by setting the IRQ output high (IRQ = 1), and resetting the BUSY flag.

Status Register (STR)

An 8-bit read-only register from which the MPU can read the contents to know a status of the FDC and the FDD:
The contents of STR is automatically changed according to the status of executing Command. After the STR is read by the MPU, the IRQ output is usually reset to low (IRQ = 0) except for the Type IV Command.

Data Register (DR)

An 8-bit read/write register:
In a read operation, an 8-bit parallel data is loaded from the DSR to the DR previously.
For Seek Command, a loaded data in the DR means a track number to be sought.
For Read Data Comand or Write Data Command, a stored data in the DR means a data read from a data field of disk or a data to be written into a data field.

Data Shift Register (DSR)

An 8-bit serial shift-register which can not be accessed directly by the MPU:
In a disk read operation, a series of bit data read from a disk are transferred to the DSR, and then the filled data in the DSR is transferred in parallel to the DR.
In a disk write operation, and 8-bit data in the DR is transferred in parallel to the DSR, and then the filled data in the DSR is serially transferred and written into a disk bit by bit.

Track Register (TR)

An 8-bit read/write shift-register:
For Restore, Seek, Step, Step-In and Step-Out Commands (i.e. TYPE I Command), a content of the TR means a present track number, and is updated during the Command execution.
For Read Data Command and Write Data Command: a content of the TR means a designated track number itself which is written into the TR prior to the Command execution and can not be renewed during the Command execution.

Sector Register (SCR)

An 8-bit read/write shift-register:
For Read Data Command and Write Data Command, a designated sector number is written into the SCR prior to the Command execution.
For Read Address Command, a track number in the ID field is transferred to the SCR.

OTHER FUNCTIONAL BLOCKS

Cycle Redundancy Check (CRC) Circuit

A circuit to check a misread/miswrite on a data which is serially transferred between the FDC and a disk:
In a write operation, a CRC data (16-bit serial bit cell) is automatically generated in the CRC circuit based on the original data and written in the disk together with the original data.
In a read operation, a CRC data is read together with the target data to check an error.

Arithmetic Logic Unit (ALU)

The ALU has the functions of Serial Data Compare, Increment (+1), Decrement (-1) and Through (±0).
The ALU can renew a content of Register and compare data.

Address Mark (AM) Detection Circuit

A circuit to detect specific bit pattern data in serial data from a disk, such as the Index Mark (or Track Mark: IDM), ID Address Mark (or Address Mark: IDAM) and Data Address Mark (Data AM).

Data Modulator

A circuit to modulate data to be written into a disk in the specified recording format:

Single density recording format:	Frequency Modulation (FM)
Double density recording format:	Modified Frequency Modulation (MFM)

Programmable Logic Array (PLA) for Commands

A micro-program to generate control signals (Commands) which control the FDC operation:
The size of micro-program is approximately 232 x 19 bits.

DC CHARACTERISTICS

(Full Guaranteed Operating Conditions unless otherwise noted.)

Parameter	Symbol	Value			Unit
		Min.	Typ.	Max.	
Output High Voltage (I_{OH} = -200μA)	V_{OH}	2.4			V
Output Low Voltage (I_{OL} = 1.8 mA)	V_{OL}			0.4	V
Three-State (Off-State) Input Current (V_{IN} = 0.4V to 2.4V)	I_{TSI}			10	μA
Input Leakage Current See Note 1)	I_{IN1}			2.5	μA
Input Leakage Current See Note 2)	I_{IN2}			100	μA
Output Leakage Current for Off-State (V_{OH} = 2.4V)	I_{LOH}			10	μA
Power Consumption	P_C			350	mW

Note 1) : Except for HLT, $\overline{TEST}$, $\overline{WF}$, $\overline{WPRT}$, and $\overline{DDEN}$. (V_{IN} = 0V to 5.25V)
2) : For HLT, $\overline{TEST}$, $\overline{WF}$, $\overline{WPRT}$, and $\overline{DDEN}$. (V_{IN} = 0V to 5.25V)

AC CHARACTERISTICS

(Full Guaranteed Operating Conditions unless otherwise noted.)

MPU Read Timing (From FDC)

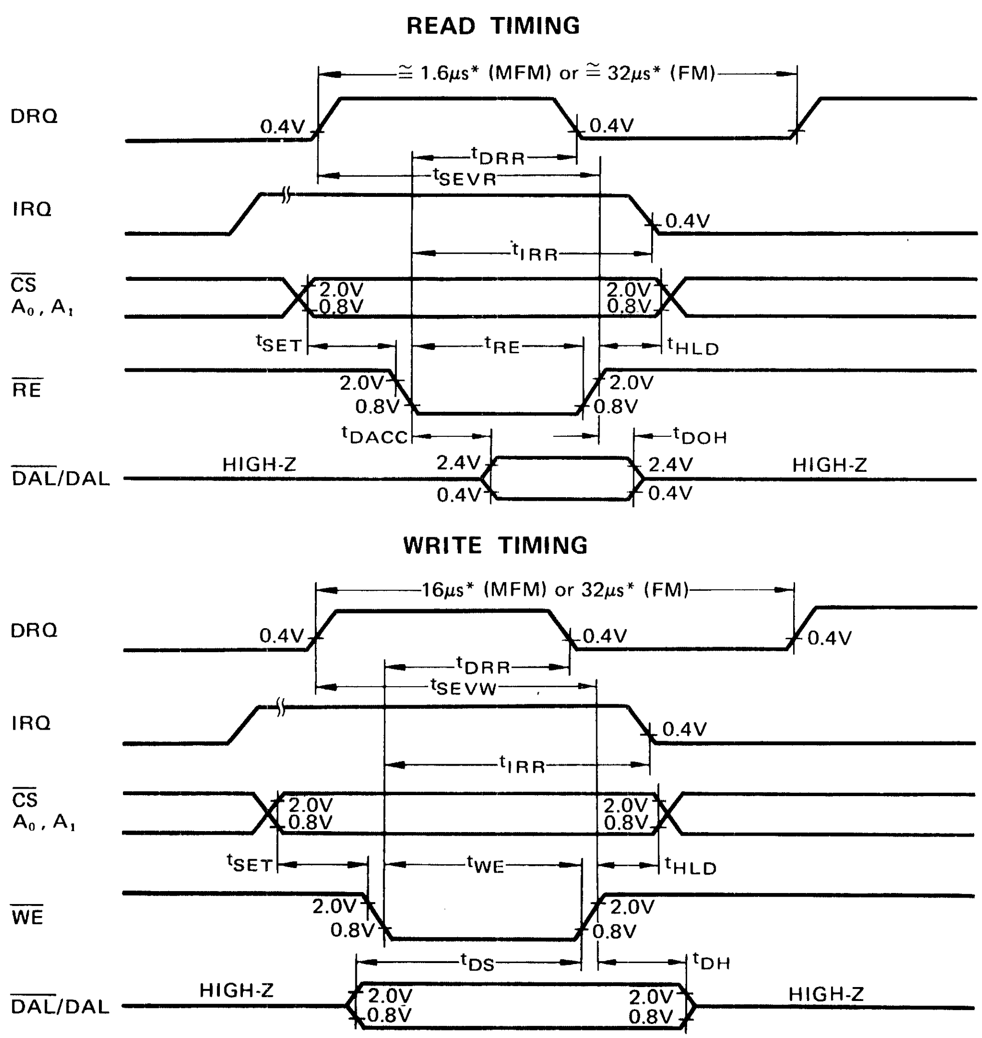
Parameter	Symbol	Value			Unit
		Min.	Typ.	Max.	
Address Setup Time	t_{SET}	50			ns
Address Hold Time	t_{HLD}	10			ns
$\overline{RE}$ Pulse Width	t_{RE}	280			ns
DRQ Reset Time	t_{DRR}			250	ns
IRQ Reset Time	t_{IRR}			500	ns
Data Delay Time (C_L = 25pF)	t_{DACC}			250	ns
Data Hold Time (C_L = 25pF)	t_{DOH}	50		150	ns
DRQ Service Time (RCLK cycle = 2μs)	t_{SEVR}			13.5*	μs

* : These values are doubled when CLK = 1 MHz.

MPU Write Timing (To FDC)

Parameter	Symbol	Value			Unit
		Min.	Typ.	Max.	
Address Setup Time	t_{SET}	50			ns
Address Hold Time	t_{HLD}	10			ns
$\overline{WE}$ Pulse Width	t_{WE}	200			ns
DRQ Reset Time	t_{DRR}			250	ns
IRQ Reset Time	t_{IRR}			500	ns
Data Setup Time	t_{DS}	250			ns
Data Hold Time	t_{DH}	0			ns
DRQ Service Time (DDEN = "L")	t_{SEVW}			11.5*	μs

* : These values are doubled when CLK = 1 MHz.



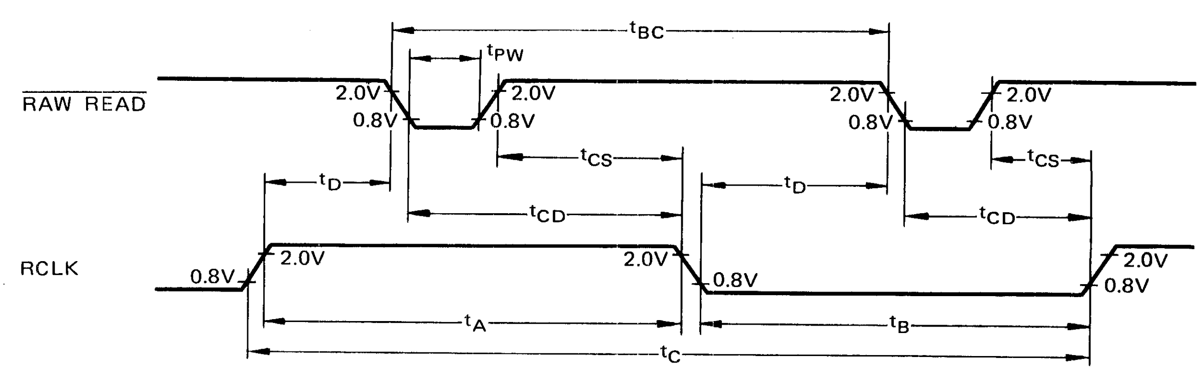
*: These values are doubled when CLK=1MHz.

FDC Read Data Timing (From Disk)

Parameter		Symbol	Value Min.	Value Typ.	Value Max.	Unit
$\overline{\text{RAWREAD}}$ Pulse Width		t_{PW}	100*		250*	ns
Clock Setup Time		t_D	40			ns
Clock Hold Time for MFM		t_{CD}	40			ns
Clock Hold Time for FM		t_{CS}	40			ns
$\overline{\text{RAWREAD}}$ Cycle Time	MFM	t_{BC}		2*, 3* or 4*		μs
	FM			2* or 4*		μs
RCLK High Pulse Width	MFM	t_A	0.8	1*	20	μs
	FM		0.8	2*	20	μs
RCLK Low Pulse Width	MFM	t_B	0.8	1*	20	μs
	FM		0.8	2*	20	μs
RCLK Cycle Time	MFM	t_C		2*		μs
	FM			4*		μs

* : These values are doubled when CLK = 1MHz.

READ DATA TIMING

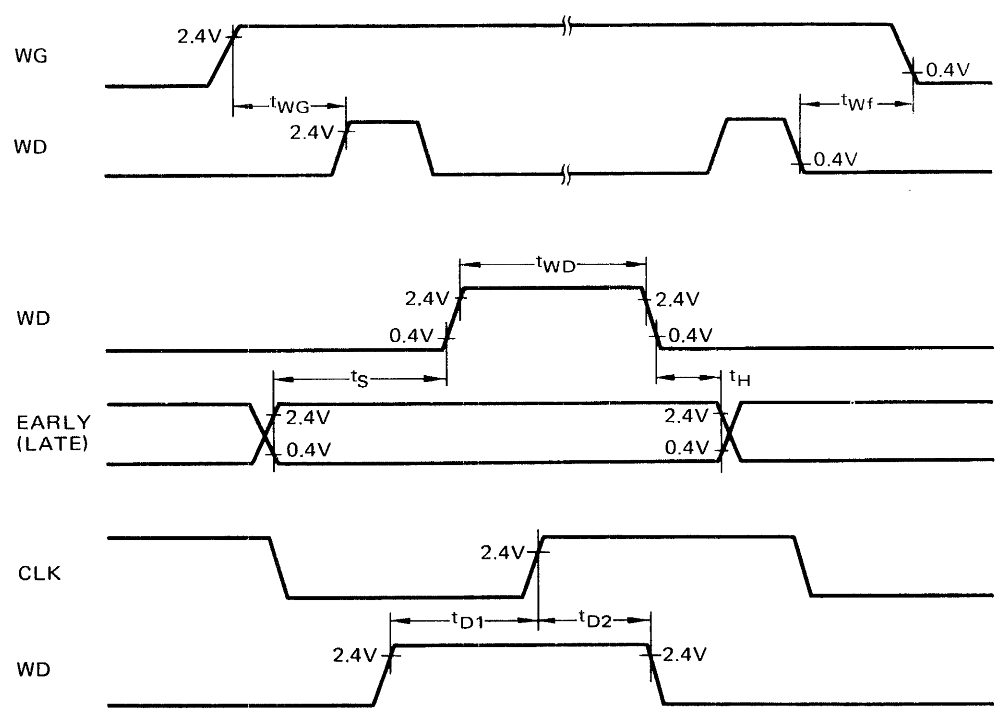


FDC Write Data Timing (To Disk)

Parameter	Symbol	Conditions		Value Min.	Value Typ.	Value Max.	Unit
Write Data Pulse Width	t_{WD}**	CLK = 2 MHz	FM	450	500	550	ns
			MFM	150	200	250	
Write Gate to Write Data	t_{WG}**	CLK = 2 MHz	FM	–	2	–	μs
			MFM	–	1	–	
Write Gate off from WD	t_{WF}**	CLK = 2 MHz	FM	–	2	–	μs
			MFM	1	–	2	
Early (Late) to Write Data	t_S	CLK = 2 MHz	MFM	125	–	–	ns
Early (Late) from Write Data	t_H	CLK = 2 MHz	MFM	-50*	–	–	ns
WD Valid after CLK	t_{D1}	CLK = 1 MHz	MFM	200	–	–	ns
		CLK = 2 MHz	MFM	30	–	–	
WD Valid to CLK	t_{D2}	CLK = 1 MHz	MFM	50	–	–	ns
		CLK = 2 MHz	MFM	50	–	–	

*: This value, -50ns (min) indicates that Early (Late) signal changes 50ns (min) before WD falls down in worst case. See DISK DATA OUTPUT TIMING.
**: All times are doubled when CLK = 1 MHz.

WRITE DATA TIMING

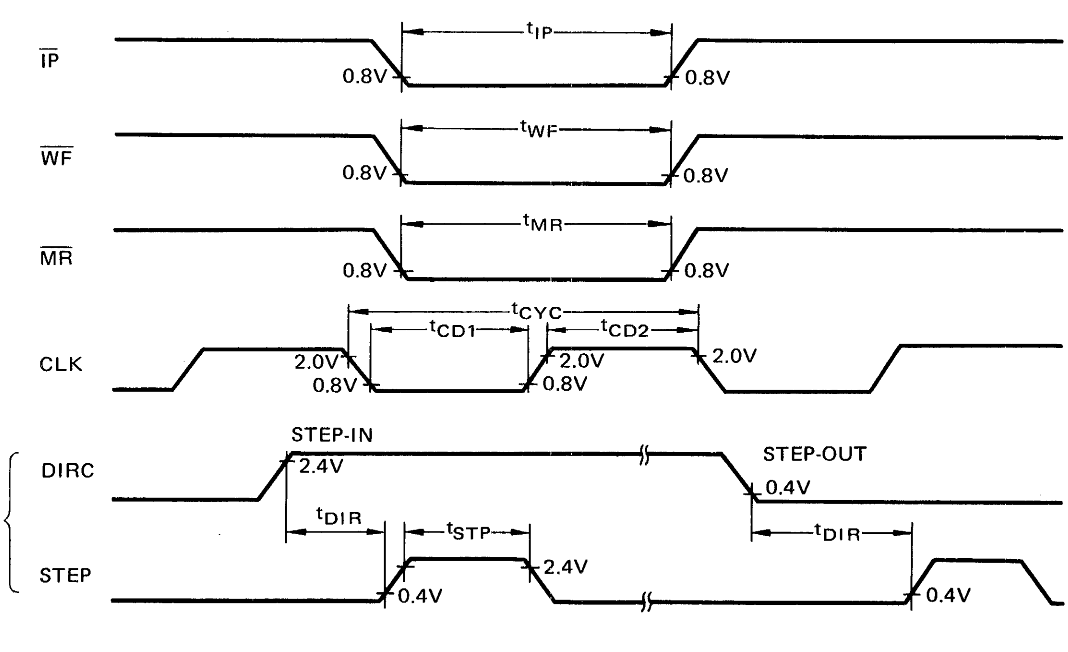


OTHER TIMINGS

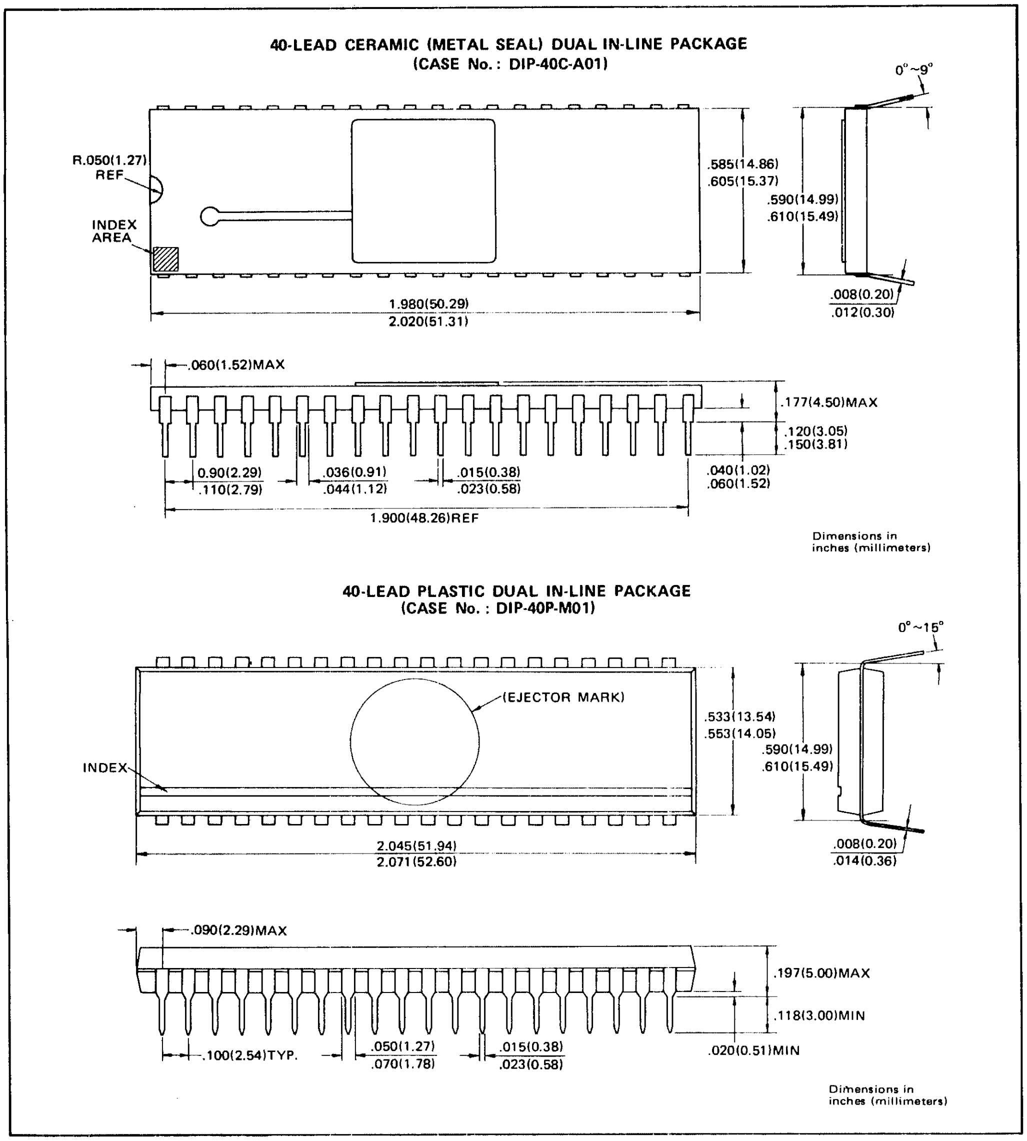
Parameter		Symbol	Value			Unit
			Min.	Typ.	Max.	
CLK Low Pulse Width		t_{CD1}	230		20000	ns
CLK High Pulse Width		t_{CD2}	200		20000	ns
STEP Pulse Width	MFM	t_{STP}	2*			μs
	FM		4*			μs
DIRC Setup Time		t_{DIR}	12*			μs
$\overline{\text{MR}}$ Pulse Width**		t_{MR}	50*			μs
$\overline{\text{IP}}$ Pulse Width		t_{IP}	10*			μs
$\overline{\text{WF}}$ Pulse Width		t_{WF}	10*			μs
CLK Cycle Time		t_{CYC}		0.5*		μs

*: These values are doubled when CLK = 1 MHz.
**: During Master Reset, CLK of more than 10 cycles are required.

OTHER TIMINGS



PACKAGE DIMENSIONS



Circuit diagrams utilizing Fujitsu products are included as a means of illustrating typical semiconductor applications; consequently, complete information sufficient for construction purposes is not necessarily given. The information has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, such information does not convey to the purchaser of the semiconductor devices described herein any license under the patent rights of Fujitsu Limited or others. Fujitsu Limited reserves the right to change device specifications.

FUJITSU LIMITED
Communications and Electronics

6-1, Marunouchi 2-chome, Chiyoda-ku, Tokyo 100, Japan
Phone: National (03) 216-3211 International (Int'l Prefix) 81-3-216-3211 Telex: J22833 Cable: "FUJITSULIMITED TOKYO"

For further information please contact:

FUJITSU LIMITED
Semiconductor Division: No.18 Mori Bldg., 3-13, 2-chome, Toranomon, Minato-ku, Tokyo 105, Japan
Phone: National (03) 502-0161 International (Int'l Prefix) 81-3-502-0161 Telex: 2224361 FTTOR J

FUJITSU MICROELECTRONICS, INC.:
2985 Kifer Road, Santa Clara, CA95051, U.S.A.
Phone: 408-727-1700 Telex: 910-338-0190 FUJITSU SNTA

FUJITSU MIKROELEKTRONIK GmbH.:
Arabella Center 9. OG./A
Lyoner Straße 44-48 D-6000 Frankfurt 71, F.R. Germany
Phone: 0611/66 63 056 Telex: 0411 963

Printed in Japan 0V2067-823C

Appendix E
Osborne 1 System Specifications

SIZE measured with case closed
- 20.5 inches wide
- 13 inches deep
- 9 inches high

WEIGHT
- 24.5 lbs. (shipping weight 32 lbs.)

PORTABILITY
- Leather handle for carrying
- When closed, unit is weatherproof
- Unit sized to fit under standard airline seat
- Power cord stows within case with cover secured by hook-and-loop plastic fasteners or hinged
- High-impact ABS plastic or injection molded case meets UL 94VO flame retardance specifications

POWER CONSUMPTION
- 110 or 220 volts
- 37 watts max.
- Configurable through internal jumper to 230 vac, or through fuse card
- Switching power supply

ENVIRONMENTAL
- Convection cooled; no fan
- Maximum operating ambient air temperature 85 degrees F
- Humidity — 95% relative, non-condensing
- Minimum operating ambient air temperature 32 degrees F

SAFETY
- Three-wire grounded power plug
- Three primary circuit protection points
- Fuse integral to power supply
- Resettable or fused circuit breaker at power switch
- Thermal cutoff opens when internal temperature exceeds 150 degrees F (resets at 130 degrees F)
- Power supply shuts off under overload conditions

CONTROLS

69 key detachable keyboard, full-travel sloped keytops includes 12 key numeric keypad
Power on-off switch in rear
Reset pushbutton on front panel
Brightness and contrast controls on front panel

MAIN PC BOARD

Processor — Z80A®, 4 Mhz CPU clock
Memory size — 64K bytes programmable (RAM)
 4K read-only memory bank-switched
 59K of programmable memory available for software
Memory access time — 250 nsec programmable memory
 350 nsec read-only memory
Program in read-only memory executes without delay
Programmable memory adds average delay times as follows:
 First M1 cycle — 188 nsec
 Subsequent consecutive M1 cycles — 0 nsec
 Non-M1 cycles — 375 nsec

DISPLAY SYSTEM

4K byte memory-mapped display memory in top page of main memory
9 bit wide display memory — 7 bit ASCII plus underline and half-intensity attributes
Scrolling performed by hardware in vertical and horizontal directions
Solid underline cursor
White video on dark background
24 lines of 52 characters visible at any time
32 lines of 128 characters video memory over which screen may be moved
Character set
 96 upper and lower case characters
 32 graphics characters

INTERFACES

Serial RS-232C
 1200 or 300 baud, selectable in software
 DTR (pin 20) handshaking input to control transmission spot rate
Modem — 9 pin plug for use with external modem (DCE)
 Adapter allows connection of modem and printer simultaneously
IEEE-488 (General Purpose instrumentation Bus)
 Controller ony, talker and listener
Keyboard — connects to keyboard through 10 inch ribbon cable removable by user
External video — drives video monitor as option (same format as 24 x 52)
Battery adapter — allows operation with external battery pack

PERIPHERALS — SELF-CONTAINED

- Video monitor
  - Size of video area
    - 3.55″ horiz.
    - 2.63″ vert.
  - P4 phosphor (same as TV)
  - Linearity 10% adjacent characters, 15% overall
  - Display format
    - 24 lines
    - 52 characters
    - 10 scan lines per character line
    - Cursor on lowest scan
  - Timing
    - 64 microseconds per scan
    - 16.666 milliseconds per vertical sweep
    - 12 microseconds horizontal blanking
    - 1.28 milliseconds vertical blanking

DISKS

- Media
  - 5.25 inch diskettes, single sided, soft sectored
  - Storage provided for 25 diskettes max.
- 40 tracks per diskette
- 10 sectors per track
- 256 bytes per sector
- Max. seek time 12 milliseconds track-to-track
- Head load time 0 milliseconds
- Rotation time 200 milliseconds
- Recording standard — FM
- Controller type 1793

ROM/BIOS 1.3

Copyright ©1982 Osborne Computer Corporation

```
        *ABS    0000  EFF0
        *CODE   EFF0  0000
        *DATA   EFF0  0000

                        ;       +------------------------------+
                        ;       !                              !
                        ;       !    SINGLE Density Monitor    !
                        ;       !                              !
                        ;       +------------------------------+

                        ;VER    =       REV A.0
                        ;LABLE  =       REV 1.3
                        ;DATE   =       29

                        ;     This  rom has been modified to accept single or double density.  All the disk driver have  been
                        ;changed. New valriables have been added which are:

                        ;       SAVTYP:
                        ;               BIT 0   =       (0,1)Double density or single density 1=single, 0=double.
                        ;               BIT 1   =       (0,1)SIDE SELECT
                        ;               BIT 2-3 =       (0-3)The sector lenth 0=128, 1=256, 2=512, 3=1024.
                        ;Added Dec. 21,81 by DEB.

                        ;     Density select(SELDEN) has been added to the disk drivers.  This proc senses the density of the
                        ;disk  and reports it by setting SAVTYP with density and sector size.  It also reports the number  of
                        ;sectors on one track in regester B.
                        ;Added Jan. 5,82 by DEB.

                        ;     There  seems to be a problem in the timing of the RSEC AND WSEC routines.  The problem shows up
                        ;when  CP/M reads the disk one sector after another.  The fix is to use RADR only when the  drive  is
                        ;slected and not before every read or write.
                        ;Added Jan. 6,82 by DEB.

        = 0000          CCP     =       0
0000    = 0000                  ORG     0               ;FWA of memory

                        ALPUSH: MACRO
                                PUSH    AF
                                PUSH    BC
                                PUSH    DE
                                PUSH    HL
                                PUSH    IX
                                PUSH    IY
                                ENDM

                        ALLPOP: MACRO
                                POP     IY
                                POP     IX
                                POP     HL
                                POP     DE
                                POP     BC
                                POP     AF
                                ENDM
```

Copyright © 1982 Osborne Computer Corporation

```
;ROM SECTIONS ARE:

;       *[1]    =       ROM STANDARDS
;       *[2]    =       CPM BOOT
;       *[3]    =       CONSOL ROUTINES
;       *[4]    =       PIA ROUTINES
;       *[5]    =       SIA ROUTINES
;       *[6]    =       DISK ROUTINES
;       *[7]    =       FORMAT ROUTINE
;       *[R]    =       RAM LOCATIONS
```

Copyright © 1982 Osborne Computer Corporation

```
                        *[1]
0000                            DISDIM                    ;disable DIM
0002  C33600                    JMP     BMON              ;reset/restart

0005                    USER:
                        ;Exit to user program from 'G' command
0005                            EI
0006  C9                        RET                       ;Exit from G command
0007                    ILINT:
                        ;Here on unknown interrupt
0007                            DI
0008                            DISDIM
000A  ED736FEF  $               STO     SP,IESTK          ;save interrupted proc stack
000E  3199EF                    LDK     SP,ISTK           ;insure stack in RAM
0011                            ALPUSH                    ;save all registers
0019  3E49                      LDK     A,'I'
001B  3206EF                    STO     A,MPCHR           ;set prompt to inter level

001E                    EXITI:
                        ;Exit interrupt code via exiting to RAM and
                        ;       then enable or disable ROM code depending on
                        ;       the value contained in ROMRAM cell.
001E  3A002C                    LD      A,H.VIO           ;clear interrupt
0021  3A08EF                    LD      A,ROMRAM
0024  B7                        OR      A
0025  C2DF00                    JNZ     ROMJP1            ;if return to RAM, CPM
0028                            ALLPOP
0030  ED7B6FEF  $               LD      SP,IESTK          ;get users stack back
0034                            EI
0035  C9                        RET

0036                    BMON:   DI
0037  31C1EF                    LDK     SP,ROMSTK
003A  AF                        XRA     A
003B  3217EF                    STO     A,SDISK           ;set disk to drive 0
003E  32EFEF                    STO     A,VRTOFF          ;OFFSET TO ZERO USED IN COUT
0041  3C                        INC     A
0042  32D0EF                    STO     A,SAVTYP          ;SET SAVTYP TO SINGEL
0045  CD8000                    CALL    HINT              ;Initialize hardware
0048  2100ED                    LDK     HL,(high MRAM)*100h
004B  22EAEF                    STO     HL,SSAVE          ;Initialize User stack in ram
004E  210000                    LDK     HL,0
0051  226DEF                    STO     HL,LDSEL          ;clear select and last track
0054  118601                    LDK     DE,IMSG
0057  CD9701                    CALL    OSTR              ;Output initial message
005A  180E ^006A$               JR      BMON1             ;continue
005C  = 0066                    ORG     NMIA
0066                            DI
0067  C33600                    JMP     BMON              ;NMI
```

```
                        ;       Main start up for Monitor.
                        ;       Get 1st user response.

006A  CD7E03          BMON1:  CALL    CI              ;Get next character
006D  FE0D                    CMP     CR
006F  21C7EF                  LK      HL,DSKSWP       ;disk swap cell
0072  0E00                    LK      C,0             ;boot from A:
0074  71                      STO     C,[hl]          ;set to A=A, B=B
0075  CA6802                  JZ      CBOOT           ;if cold boot
0078  34                      INC     [hl]            ;swap drives: A=B, B=A
0079  FE22                    CMP     '"'
007B  CA6802                  JZ      CBOOT           ;if cold boot off of B
007E  18B6 ^0036$             JR      BMON
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
0080                     HINT:
                         ;Initialize all dependent hardware.
                         ;ENTRY
                         ;None.

                         ;EXIT
                         ;All hardware initialized.
                         ;Set flag indicating in ROM
0080                             PROC
0080                             ENAROM
0089                             DISDIM                   ;disable DIM bit
008B  211F07                     LDK     HL,GKEY
008E  22F8EF                     STO     HL,INTBL+(4*2)   ;set keyboard interrupt
0091  ED5E        $              IM2
0093  3EEF                       LDK     A,high INTBL
0095  ED47        $              MOV     I,A              ;set interrupt page

                         ;       Initialize keyboard
0097  AF                         XRA     A
0098  32F0EF                     STO     A,0EFF0H
009B  3268EF                     STO     A,BELCNT         ;clear bell timer cell
009E  325EEF                     STO     A,LKEY           ;clear last key cell
00A1  3207EF                     STO     A,ECHOP          ;clear echo to list dev
00A4  3260EF                     STO     A,ESCH           ;clear ESC hold flag
00A7  21D4EF                     LDK     HL,KEYLST
00AA  0606                       LDK     B,KL_LEN*KLE_LEN
00AC                     :1:
00AC  77                         STO     A,[HL]
00AD  23                         INC     HL
00AE  10FC ^00AC$                DJNZ    :1

00B0  2F                         CMA
00B1  3263EF                     STO     A,IDAY           ;set date invalid for SETUP.COM
00B4  3259EF                     STO     A,KEYLCK         ;indicate NOT locked
00B7  3E80                       LDK     A,VLL
00B9  326CEF                     STO     A,LLIMIT         ;set max line limit

                         ;       Now function PIA to set starting address
                         ;       and continue to move display window over
                         ;       the first display line.
                         ;       Set cursor as 1-40 on 1st line.

00BC  CD4F09                     CALL    SPAO             ;set up for output
00BF  0EEA                       LDK     C,VFLO
00C1  CD6409                     CALL    OPAD             ;set for -10 char position AND DOUBLE DENSITY

                         ;       Set beginning line to 0

00C4  AF                         XRA     A
00C5  4F                         MOV     C,A
00C6  CD7109                     CALL    OPBD             ;set line

                         ;       Intialize IEEE port

00C9  0E01                       ldk     c,1
00CB  CD7E09                     call    ie.co

                         ;       Reset-Master clear the SIO (ACIA)

00CE  0E55                       LK      C,SI.S16         ;select 16x clock for 1200 baud
```

```
00D0  CDF60A          CALL    SIRST           ;reset

                 ;    Set default seek to as defined by systext

00D3  3E02            LDK     A,SEEKTM        ;defined from systext
00D5  3213EF          STO     A,SEKDEL        ;set seek step rate

                 ;    Set default prompt char

00D8  3E3E            LDK     A,PMCHR
00DA  3206EF          STO     A,MPCHR         ;set up default
00DD                  EI
00DE  C9              RET
```

Copyright © 1982 Osborne Computer Corporation

```
00DF  113316        romjp1: ldk     de,1633h        ;offset in bios jump table
00E2  1803 ^00E7$           jr      biojp
00E4  113616        romjp2: ldk     de,1636h        ;offset in bios jump table
00E7  2AD2EF        biojp:  ld      hl,ccpadr
00EA  19                    add     hl,de           ;form address
00EB  E9                    jmp     [hl]
```

Copyright © 1982 Osborne Computer Corporation

```
00EC  = 0100            ORG     100h

                ;ROM JUMP TABLE
                ;       CBIOS   =       Jmps used mainly by CBIOS
                ;       SC      =       Jmps used mainly by SuperCalc

0100  C36802            JMP     CBOOT           ;CBIOS  cold boot
0103  C3AB02            JMP     WBOOT           ;CBIOS  warm boot
0106  C37103            JMP     SKEY            ;CBIOS  keyboard status
0109  C37E03            JMP     CI              ;CBIOS  keyboard input
010C  C31004            JMP     COUT            ;CBIOS  console output
010F  C31C0B            JMP     LIST            ;CBIOS  list output
0112  C31C0B            JMP     LIST            ;CBIOS  punch output
0115  C3030B            JMP     READER          ;CBIOS  reader input

                ;       Disk I/O

0118  C3400B            JMP     RDRV            ;CBIOS  HOME
011B  C9                RET ! NOP ! NOP         ;CBIOS  SELECT DISK
011E  C3750C            JMP     READ            ;       READ SECTOR
0121  C37D0C            JMP     WRITE           ;       WRITE SECTOR
0124  C3EC0C            JMP     RADR            ;       READ SECTOR ANY SECTOR HEADER
0127  C35B0B            JMP     RSEC            ;CBIOS  DISK SECTOR READ
012A  C36D0B            JMP     WSEC            ;CBIOS  DISK SECTOR WRITE
012D  C3130B            JMP     SLST            ;CBIOS  List device status
0130  C3800B            JMP     SENDEN          ;CBIOS  SENSE THE DENSITY OF DRIVE
0133  C3DF00            JMP     ROMJP1          ;CBIOS
0136  C3E400            JMP     ROMJP2          ;CBIOS
0139  C33F0E            JMP     FORMAT          ;CBIOS  FORMATING ROUTINE

                ;       IEEE

013C  C3F60A            JMP     SIRST           ;CBIOS  SIO reset
013F  C37E09            JMP     IE.CO           ;CBIOS  IEEE    Control Out
0142  C3C109            JMP     IE.SI           ;CBIOS          Status In
0145  C3D409            JMP     IE.GTS          ;CBIOS          Go To Standby
0148  C3E009            JMP     IE.TC           ;CBIOS          Take Control
014B  C3200A            JMP     IE.OIM          ;CBIOS          Output Interface Message
014E  C3430A            JMP     IE.ODM          ;CBIOS          Output Device Message
0151  C3850A            JMP     IE.IDM          ;CBIOS          Input Device Message
0154  C3E10A            JMP     IE.PP           ;CBIOS          Parallel Poll

                ;       SuperCalc

0157  C3F106            JMP     VLDDR           ;SC     VIDEO BLOCK MOVE DEC
015A  C30507            JMP     VLDIR           ;SC     VIDEO BLOCK MOVE INC
015D  C3E906            JMP     STODIM          ;SC     STO reg B IN [HL]

                ;       DISK I/O

0160  C3200E            JMP     DMAWRT          ;       DMA WRITE TO CONTROLER
0163  C30A0E            JMP     DMARD           ;       DMA READ FROM CONTROLER
0166  C3F70B            JMP     HOME            ;       HOME DISK DRIVE
0169  C31A0C            JMP     SEEK            ;       SEEK TO TRACK
016C  C3390C            JMP     STEP            ;       STEP SAME DIRECTION
016F  C34D0C            JMP     STEPIN          ;       STEP IN
0172  C3610C            JMP     STEPOUT         ;       STEP OUT
0175  C3640D            JMP     FORINT          ;       FORCE INTERUPT
0178  C3330D            JMP     READTRK         ;       READ TRACK
017B  C3490D            JMP     FMTTRK          ;       Format one track
```

Copyright © 1982 Osborne Computer Corporation

```
017E  C3A40D          JMP     SELDRV          ;       SELECT DRIVE
0181  C3280B          JMP     ACISTAT         ;CBIOS  SERIAL PORT STATUS
0184  C3F008          JMP     SCTRKR          ;       SET TRACK REGESTER IN CONTROLER CHIP WITH VALUE IN SAVTRK
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0187  0D0A         EMBOOT: DB      CR,LF
0189  424F4F5420           DB      'BOOT ERROR'
0193  A0                   DC      ' '

0194               EBOOT:
                   ;BOOT ERROR MESSAGE ROUTINE
                   ;ENTRY
                   ;NONE

0194                       PROC
0194  118701               LDK     DE,EMBOOT        ;HERE ON BOOT ERROR
                                                    ;FALL THROUGH TO CSTR
```

Copyright © 1982 Osborne Computer Corporation

```
0197                    OSTR:
                        ;OUTPUT STRING TO CONSOLE
                        ;NOTE:  OSTR RECOGNIZES 7F AS AN ESCAPE SEQUENCE TO REPEAT CHAR N TIMES.  FORMAT IS: 7F, REPEAT COUNT, CHAR
                        ;ENTRY
                        ;DE      =       FWA OF SOURCE

0197                             PROC
0197  1A                         LD      A,[DE]
0198  B7                         OR      A
0199  F5                         PUSH    AF
019A  E67F                       AND     07FH
019C  FE7F                       CMP     07FH
019E  4F                         MOV     C,A
019F  200C ^01AD$                JRNZ    :4              ;IF NOT REPEAT
01A1  13                         INC     DE
01A2  1A                         LD      A,[DE]
01A3  3D                         DEC     A
01A4  47                         MOV     B,A             ;REPEAT COUNT
01A5  13                         INC     DE
01A6  1A                         LD      A,[DE]          ;GET REPEAT CHAR
01A7  4F                         MOV     C,A
01A8  CD1004          :2:        CALL    COUT            ;OUTPUT CHAR
01AB  10FB ^01A8$                DJNZ    :2              ;IF NOT DONE
01AD  CD1004          :4:        CALL    COUT            ;OUTPUT IT
01B0  13                         INC     DE
01B1  F1                         POP     AF
01B2  F29701                     JP      OSTR            ;IF NOT DONE
01B5  C9                         RET
```

Copyright © 1982 Osborne Computer Corporation

```
01B6                    IMSG:
01B6    1A0A0A0A0A              DB      'Z'-40h,lf,lf,lf,lf
01BB    7F0B20                  DB      07Fh, 11, ' '
01BE    1B67                    DB      ESC,VSGH
01C0    11                      DB      'Q'-40h
01C1    7F1817                  DB      07Fh, 24, 'W'-40h
01C4    05                      DB      'E'-40h
01C5    1B47                    DB      ESC,VEGH

01C7    0D0A                    DB      cr,lf
01C9    7F0B20                  DB      07Fh, 11, ' '
01CC    1B67011B47              DB      ESC,VSGH,1, ESC,VEGH
01D1    2020                    DB      '  '
01D3    1B6C                    DB      ESC,'l'
01D5    4F53424F52              DB      'OSBORNE System One.'
01E8    1B6D                    DB      ESC,'m'
01EA    202020                  DB      '   '
01ED    1B67041B47              DB      ESC,VSGH,4, ESC,VEGH

01F2    0D0A                    DB      cr,lf
01F4    7F0B20                  DB      07Fh, 11, ' '
01F7    1B67                    DB      ESC,VSGH
01F9    01                      DB      'A'-40h
01FA    7F1820                  DB      07Fh,24,' '
01FD    04                      DB      'D'-40h
01FE    1B47                    DB      ESC,VEGH

0200    0D0A                    DB      cr,lf
0202    7F0B20                  DB      07Fh, 11, ' '
0205    1B67                    DB      ESC,VSGH
0207    01                      DB      'A'-40h
0208    1B29                    DB      ESC,')'
020A    2052657620              DB      ' Rev 1.3    c. 1982 OCC '
0222    1B28                    DB      ESC,'('
0224    04                      DB      'D'-40h
0225    1B47                    DB      ESC,VEGH

0227    0D0A                    DB      cr,lf
0229    7F0B20                  DB      07Fh, 11, ' '
022C    1B67                    DB      ESC,VSGH
022E    1A                      DB      'Z'-40h
022F    7F1818                  DB      07Fh, 24, 'X'-40h
0232    03                      DB      'C'-40h
0233    1B47                    DB      ESC,VEGH

0235    0D0A0A0A                DB      cr,lf,lf,lf
0239    7F0420                  DB      07Fh, 4, ' '
023C    496E736572              DB      'Insert disk in Drive '
0251    1B6C                    DB      ESC,'l'
0253    41                      DB      'A'
0254    1B6D                    DB      ESC,'m'
0256    20616E6420              DB      ' and press RETURN'
0267    AE                      DC      '.'
```

Copyright © 1982 Osborne Computer Corporation

```
*[2]

;    The  CBOOT entry point gets control from the cold start loader and is responsible for the  basic
;system  initialization.  This  includes outputting a sign-on message and initializing the  following
;page zero locations:

;       0,1,2: Set to the warmstart jump vector.
;           3: Set to the initial IOBYTE value.
;           4: Default and logged on drive.
;       5,6,7: Set to a jump to BDOS.

;    The WBOOT entry point gets control when a warm start occurs,  a ^C from the console,  a jump  to
;BDOS  (function 0),  or a jump to location zero.  The WBOOT routine reads the CCP and BDOS from  the
;appropriate disk sectors.
;    WBOOT  must  also re-initialize locations 0,1,2 and 5,6,7.  The WBOOT routines exits with the  C
;register set to the appropriate drive selection value. The exit address is to the CCP routine.

;    Single Density Disk layout Definition:
;
;       Track 0---------------------
;         1 thru 8      CCP     2k
;         9 thru 10     BDOS
;       Track 1---------------------
;         1 thru 10     BDOS
;       Track 2---------------------
;         1 thru 2      BDOS    3.5k
;         3 thru 10     CBIOS   2k
```

Copyright © 1982 Osborne Computer Corporation

```
0268                    CBOOT:
                        ;ENTRY
                        ;C        =       drive to boot from

                        ;EXIT
                        ;A        =       DRIVE TO BOOT FROM

0268                              Proc

                        *SET "SDISK" AND "SAVTYP"

0268  79                          MOV     A,C
0269  3217EF                      STO     A,SDISK         ;SET DRIVE

026C  CD800B            :1:       CALL    SENDEN          ;DETERMINE DENSITY
026F  2805 ^0276$                 JRZ     :2              ;IF GOOD

0271  CD9401                      CALL    EBCOT           ;PRINT ERROR
0274  18F6 ^026C$                 JR      :1

                        *READ AND SET FBA OF CCP

0276  2100D0            :2:       LDK     HL,0D000H
0279  220FEF                      STO     HL,DMADR        ;SET DMA

027C  AF                :3:       XRA     A
027D  3215EF                      STO     A,SAVTRK        ;set track
0280  3C                          INC     A
0281  3214EF                      STO     A,SAVSEC        ;set sector
0284  47                          MOV     B,A
0285  CD5B08                      CALL    RSEC            ;READ SECTOR ONE
0288  2805 ^028F$                 JRZ     :4              ;IF GOOD

028A  CD9401                      CALL    EBCOT           ;PRINT ERROR
028D  18ED ^027C$                 JR      :3

028F  3A02D0            :4:       LD      A,0D002H        ;get ccp address/100h + 3
0292  D603                        SUB     3
0294  2E00                        LDK     L,0
0296  67                          MOV     H,A

                        *SET NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT

0297  063C                        LDK     B,60            ;CCP/BDOS/CBIOS

                        *READ SYSTEM

0299  3A17EF                      LD      A,SDISK         ;DRIVE TO BOOT FROM
029C  F5                          PUSH    AF
029D  E5                          PUSH    HL              ;SAVE FWA FOR "CCPADR"
029E  CDB302                      CALL    BCPM            ;boot system
02A1  E1                          POP     HL
02A2  F1                          POP     AF

                        *JUMP SYSTEM

02A3  22D2EF                      STO     HL,CCPADR
02A6  110016                      LDK     DE,1600h        ;offset for bios
02A9  19                          ADD     HL,DE           ;address of bios in hl
02AA  E9                          JMP     [HL]            ;enter cpm
```

```
02AB                  WBOOT:
                      ;ENTRY
                      ;NONE

                      ;EXIT
                      *NOTE*
                      *:      THIS ROUTINE DOES NOT EXIT. IT ONLY SETS PARAMITERS FOR BCPM:.

                      *;A     =       DRIVE TO BOOT FROM
                      *;B     =       NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT
                      *;HL    =       DMA ADDR FOR CCP

02AB                          Proc
02AB  062C                    LDK     B,44            ;CCP/BDOS and don't read CBIOS
02AD  3A0400                  LD      A,CDISK         ;Current logged in drive
02B0  2AD2EF                  LD      HL,CCPADR
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
02B3                    BCPM:
                        ;Boot CPM from disk
                        ;ENTRY
                        ;A      =       DRIVE TO BOOT FROM
                        ;B      =       NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT
                        ;HL     =       DMA ADDR FOR CCP

                        ;EXIT
                        ;NONE
                        ;LOOPS ON ERROR

02B3                            PROC

                        *SET "SDISK", "DMADR" AND "SAVSEC"

02B3  3217EF                    STO     A,SDISK         ;Set drive to boot from
02B6  220FEF                    STO     HL,DMADR        ;SET DMA
02B9  3E01                      LDK     A,1
02BB  3214EF                    STO     A,SAVSEC        ;set sector
02BE  3D                        DEC     A
02BF  3200EF                    STO     A,TEM           ;MAKE TEM ZERO

                        *SET "SAVTYP" AND GET NUMBER OF SECTORS PER TRACK

02C2  C5                        PUSH    BC

02C3  CD8008            :RLOOP: CALL    SENDEN          ;DETERMINE DENSITY
02C6  2805 ^02CD$               JRZ     :GOOD           ;IF GOOD

02C8  CD9401                    CALL    EBOOT           ;PRINT ERROR
02CB  18F6 ^02C3$               JR      :RLOOP

02CD  D1                :GOOD:  POP     DE              ;D=NUMBER OF 128 BYTE BLOCKS
02CE  C5                        PUSH    BC              ;SAVE NUMBER OF SECTORS IN ONE TRACK

                        *SET NUMBER OF SECTORS TO READ

02CF  3AD0EF                    LD      A,SAVTYP
02D2  CB3F        $             SRL     A
02D4  CB3F        $             SRL     A
02D6  E603                      ANI     0000_0011B      ;A=NUMBER OF BYTES IN ONE SECTOR(0-3)
02D8  FE00                      CMP     0
02DA  2818 ^02F4$               JRZ     :1              ;IF 128 BYTES SECTORS

                                ;GET NUMBER TO DEVIDE BY

02DC  47                        MOV     B,A             ;B=NUMBER OF BYTES IN ONE SECTOR(1-3)
02DD  3E01                      LDK     A,1
02DF  CB27        $     :1LOOP: SLA     A               ;TIMES TWO
02E1  10FC ^02DF$               DJNZ    :1LOOP

02E3  47                        MOV     B,A             ;NUMBER TO DIVIDE BY
02E4  7A                        MOV     A,D             ;A=NUMBER OF 128 BYTE BLOCKS
02E5  1600                      LDK     D,0
02E7  90                :2LOOP: SUB     B               ;SUBTRACK WITH DIVISOR
02E8  08          $             EX      AF              ;SAVE FLAGS
02E9  14                        INC     D               ;COUNT
02EA  08          $             EX      AF              ;RESTORE FLAGS
02EB  2807 ^02F4$               JRZ     :1              ;IF REZULT IS ZERO (NO PARTIAL SECTORS)
02ED  30F8 ^02E7$               JRNC    :2LOOP          ;LOOP
```

Copyright © 1982 Osborne Computer Corporation


```
02EF  ED44       $          NEG     A                ;2 COMP
02F1  3200EF                STO     A,TEM            ;SAVE REMAINDER AND INDICATE A PARTIAL SECTOR

02F4  C1            :1:     POP     BC               ;B=NUMBER OF SECTORS IN ONE TRACK
02F5  4A                    MOV     C,D              ;C=NUMBER OF SECTORS TO READ

                    *READ SYSTEM

02F6  AF                    XRA     A                ;A=0
02F7  3215EF        :TLOOP: STO     A,SAVTRK         ;SET TRACK

                            ;CHECK FOR ALL SECTORS READ

02FA  79                    MOV     A,C              ;SECTORS TO READ
02FB  B7                    ORA     A
02FC  2008 ^0306$           JRNZ    :5               ;IF C IS NOT ZERO CONTINUE

                            ;CHECK FOR NO PARCIAL SECTORS

02FE  3A00EF                LD      A,TEM
0301  B7                    ORA     A
0302  2858 ^035C$           JRZ     :4               ;YOUR DONE IF C=0 AND TEM=0
0304  182A ^0330$           JR      :RLS             ;READ ONE MORE PARCIAL SECTOR

                            ;UPDATE NUMBER OF SECTORS LEFT TO READ

0306  90            :5:     SUB     B                ;SUBTRACK SECTORS IN ONE TRACK
0307  4F                    MOV     C,A              ;SAVE REMAINING SECTORS TO READ
0308  3005 ^030F$           JRNC    :2               ;A>B MORE THAN ONE TRACK LEFT TO READ

                            ;IF THIS IS LAST TRACK ZERO NUMBER OF SECTORS LEFT TO READ

030A  ED44       $          NEG     A                ;2 COMP
030C  47                    MOV     B,A              ;READ ALL THE REMAINING SECTORS
030D  0E00                  LDK     C,0              ;STOP AFTER THIS READ

                            ;CHECK FOR NONZERO VALUE IN TEM AND THE LAST SECTOR TO READ

030F  3A00EF        :2:     LD      A,TEM
0312  B7                    ORA     A
0313  2808 ^031D$           JRZ     :3               ;IF TEM IS ZERO SKIP THIS

0315  AF                    XRA     A
0316  B1                    ORA     C
0317  2004 ^031D$           JRNZ    :3               ;IF REG C IS NOT ZERO SKIP THIS(NOT LAST TRACK)

                            ;READ ONE LESS THAN THE LAST SECTOR

0319  78                    MOV     A,B
031A  3D                    DEC     A                ;A=B-1
031B  2813 ^0330$           JRZ     :RLS             ;ONLY ONE SECTOR LEFT TO READ

                            ;READ ONE TRACK

031D  CD5B0B        :3:     CALL    RSEC             ;READ (BC IS SAVED)
0320  2805 ^0327$           JRZ     :GOO             ;IF GOOD

0322  CD9401                CALL    EBOOT            ;REPORT ERROR
0325  18F6 ^031D$           JR      :3
```

Copyright ©1982 Osborne Computer Corporation

```
                              ;UPDATE DMA

0327  220FEF       :GOO:     STO     HL,DMADR        ;SET DMA

                              ;UPDATE TRACK

032A  3A15EF                  LD      A,SAVTRK
032D  3C                      INC     A
032E  18C7 ^02F7$             JR      :TLOOP          ;TRACK LOOP

                    *READ A PARCIAL SECTOR

0330  E5           :RLS:     PUSH    HL              ;SAVE ADDRESS TO WRITE TO

0331  2180EA                  LDK     HL,0EA80H       ;ADDRESS OF HOST BUFFER IN BIOS
0334  220FEF                  STO     HL,DMADR        ;SET DMA

0337  2115EF                  LDK     HL,SAVTRK
033A  35                      DEC     [HL]            ;SAVTRK = SAVTRK - 1

                              ;READ SECTOR IN HOST BUFF

033B  0601                    LDK     B,1
033D  CD750C       :3LOOP:   CALL    READ            ;READ ONE SECTOR
0340  2805 ^0347$             JRZ     :GO             ;IF GOOD

0342  CD9401                  CALL    EBOOT           ;REPORT ERROR
0345  18F6 ^033D$             JR      :3LOOP

                              ;SET NUMBER OF BYTES TO TRANSFER

0347  3A00EF       :GO:      LD      A,TEM
034A  47                      MOV     B,A             ;B=NUMBER OF 128 BYTE BLOCK TO TRANSFER
034B  210000                  LDK     HL,0
034E  118000                  LDK     DE,128

0351  19           :4LOOP:   ADD     HL,DE
0352  10FD ^0351$             DJNZ    :4LOOP
0354  E5                      PUSH    HL
0355  C1                      POP     BC              ;BC=NUMBER OF BYTES TO TRANSFER

                              ;TRANSFER BYTES

0356  2180EA                  LDK     HL,0EA80H       ;SOURCE
0359  D1                      POP     DE              ;DESTINATION
035A  EDB0         $          LDIR                    ;MOVE

                    *CLEAR BUFFERS SET PARR. AND RETURN TO SYSTEM

035C  2150EF       :4:       LDK     HL,HSTACT
035F  110700                  LDK     DE,(LOGSEC-HSTACT)+1
0362  CD3D09                  CALL    FILLZ           ;clear Host BIOS cells
0365  3EFF                    LDK     A,0FFh
0367  3255EF                  STO     A,UNASEC
036A  3E7F                    LDK     A,VLL-1
036C  326EEF                  STO     A,LDTRK         ;set other drive NOT int
036F  AF                      XRA     A               ;Clear error indicator
0370  C9                      RET
```

```
                    *[3]

                    ;       Assembly constant
   = 0001           @KEY    =       1               ;production keyboard

                    ;       Control keys
   = 0007           CBELL   =       'G'-40h         ;Ring the Bell
   = 000B           MCUP    =       'K'-40h         ;Move cursor up
   = 000A           MCDOWN  =       LF              ;Move cursor down
   = 0008           MCLEFT  =       BKS             ;Move cursor left
   = 000C           MCRIGH  =       'L'-40h         ;Move cursor right
   = 001A           VCLRS   =       'Z'-40h         ;Clear and home cursor
   = 001E           VHOME   =       '^'-40h         ;Home Cursor


                    ;       Escape keys
   = 0023           VLOCK:  =       '#'             ;Lock Keyboard
   = 0022           VUNLK   =       '"'             ;Unlock Keyboard
   = 003D           VCAD:   =       '='             ;Cursor Addressing
   = 0053           VSAD:   =       'S'             ;Screen Addressing
   = 0051           VINC:   =       'Q'             ;Insert Char
   = 0057           VDELC:  =       'W'             ;Delete char
   = 0045           VINL:   =       'E'             ;Insert line
   = 0052           VDELL:  =       'R'             ;Delete line
   = 0054           VCEOL:  =       'T'             ;Clear to end of line
   = 0029           VSHI:   =       ')'             ;Start half intensity
   = 0028           VEHI:   =       '('             ;end
   = 006C           VSUL:   =       'l'             ;Start underline
   = 006D           VEUL:   =       'm'             ;end

   = 0067           VSGH:   =       'g'             ;Start graphics
   = 0047           VEGH:   =       'G'             ;End

                    ;ARROW KEYS

   = 008A           UP      =       8AH
   = 008B           RIGHT   =       8BH
   = 008C           DOWN    =       8CH
   = 008D           LEFT    =       8DH
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0371                SKEY:
                    ;Get status of keyboard

                    ;EXIT
                    ;Cbit set if no data ready
0371  3A59EF                LD      A,KEYLCK
0374  B7                    OR      A
0375  C8                    RZ                      ;if locked keyboard
0376  3A5EEF                LD      A,LKEY
0379  B7                    ORA     A               ;CHECK FOR ZERO
037A  C8                    RZ
037B  F6FF                  ORI     OFFH            ;IF NOT ZERO MAKE OFFH
037D  C9                    RET                     ;IF DATA
```

Copyright © 1982 Osborne Computer Corporation

```
037E                    CI:
037E                    RKEY:
                        ;Read next key from keyboard

                        ;EXIT
                        ;A        =       last key
037E                              PROC
037E  CD7103                      CALL    SKEY
0381  28FB ^037E$                 JRZ     RKEY            ;if NO data
0383                              DI
0384  3A5EEF                      LD      A,LKEY          ;GET CHARACTER
0387  4F                          MOV     C,A
0388  AF                          XRA     A
0389  325EEF                      STO     A,LKEY          ;clear key from hold
038C  79                          MOV     A,C
038D                              EI
038E  C9                          RET
```

Copyright © 1982 Osborne Computer Corporation

```
038F                    NORM:
                        ;Normalize to UPPER case
                        ;ENTRY
                        ;A       =       char

                        ;EXIT
                        ;A       =       CHAR
038F  FE61                       CMP     'a'
0391  D8                         RC                      ;if upper
0392  FE7B                       CMP     'z'+1
0394  D0                         RNC                     ;if not lower
0395  D620                       SUI     'a'-'A'
0397  C9                         RET

      = 002D            REPD     =       45              ;initial rep delay
      = 0005            REPK     =       5               ;repeat constant
```

Copyright © 1982 Osborne Computer Corporation

```
0398                    UPTIM:
                        ;Routine checks to see if the disk drive motor should be turned off by updating DACTIVE...Routine ALSO
                        ;checks to see if bell is currently ringing: if so, decrement counter.  If counter turns zero,
                        ;shut off bell.
0398                            Proc
0398  216BEF                    LDK     HL,BELCNT
039B  AF                        XOR     A
039C  B6                        OR      [hl]     ;cell=zero ?
039D  280C ^03AB$               JRZ     :2       ;if bell now off
039F  35                        DEC     [hl]     ;...bell is on. decrement counter
03A0  2009 ^03AB$               JRNZ    :2       ;if bell should stay on awhile yet
03A2  3A62EF                    LD      A,PIABD
03A5  E6DF                      AND     1101_1111b      ;clear bell bit
03A7  4F                        MOV     C,A
03A8  CD7109                    CALL    OPBD
03AB                    :2:
03AB  2B                        DEC     HL              ;HL => DACTVE
03AC  7E                        LD      A,[hl]
03AD  B7                        OR      A
03AE  C8                        RZ                      ;RETURN if inactive
03AF  35                        DEC     [hl]            ;reset delay
03B0  CC360E                    CZ      DDRV            ;if deselect drive
03B3  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;       Bit definitions for ESCH flag byte
                        ;       Note Bit 7 is currently free.
 = 0040                 EF_X:   =       64              ;B6= extegting X-coordinate
 = 0020                 EF_SCR: =       32              ;B5= Screen/Cursor Addressing
 = 0010                 EF_ADR: =       16              ;B4= expegting address-chr
 = 0008                 EF_ESC: =       8               ;B3=$last char was ESC
 = 0004                 EF_UN:  =       4               ;B2= Underline mode
 = 0002                 EF_HA:  =       2               ;B1= Half Intensity mode
 = 0001                 EF_GR:  =       1               ;B0= Graphics mode
 = 0007                 EF_MSK: =       EF_UN+EF_HA+EF_GR  ;Mask to get mode.
```

```
                        ;       Vector (branch) table for video output mode selection
                        ;       controlled by ESCH mode
0384                    ESCHTS:
03B4  6704                      DW      VNORM           ;0 Normal mode
03B6  8105                      DW      VGRAPH          ;1 Graphics mode
03B8  9504                      DW      VHALF           ;2 Half intensity mode
03BA  9904                      DW      VHA_GR          ;3 Half and graphics
03BC  8704                      DW      VUNDER          ;4 Underline mode
03BE  8B04                      DW      VUN_GR          ;5 Under and graphics
03C0  8F04                      DW      VUN_HA          ;6 Under and half intensity
03C2  9304                      DW      VUN_HA_GR       ;7 Under and half and graphics
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
03C4                VALIDE:
                    ;Valid ESC-Sequence Table
                    ;       3 bytes per entry:ascii char , "DW"-Vector.
                    ;       no. of entries is VALETS
                    ;       Following body of table is 2 byte No-Match adrs
03C4  3D                    DB      VCAD    ! DW    ESCCAD  ;Cursor Addressing
03C7  53                    DB      VSAD    ! DW    ESCSAD  ;Screen Addressing
03CA  67                    DB      VSGH    ! DW    ESCSGR  ;Set graphics mode
03CD  47                    DB      VEGH    ! DW    ESCCGR  ;Clr graphics mode
03D0  29                    DB      VSHI    ! DW    ESCSHA  ;Set half int. mode
03D3  28                    DB      VEHI    ! DW    ESCCHA  ;Clr half int. mode
03D6  6C                    DB      VSUL    ! DW    ESCSUN  ;Set underline mode
03D9  6D                    DB      VEUL    ! DW    ESCCUN  ;Clr underline mode

03DC  1A                    DB      VCLRS   ! DW    ESCZZ   ;Clear screen to blanks
03DF  51                    DB      VINC    ! DW    EINSRT  ;Insert char
03E2  57                    DB      VDELC   ! DW    EDELC   ;Delete char
03E5  45                    DB      VINL    ! DW    ESCEE   ;Insert line
03E8  52                    DB      VDELL   ! DW    ESCRR   ;Delete line
03EB  54                    DB      VCEOL   ! DW    EEOL    ;Clear to end of line
03EE  23                    DB      VLOCK   ! DW    ESCLCK  ;Lock Keyboard
03F1  22                    DB      VUNLK   ! DW    ESCULK  ;Unlock Keyboard
03F4                :end:
03F4  4A04                  DW      COUT2   ;No Match exit
                    ;       Ignore char upon undefined ESC-Sequence
                    ;       (to treat undefined char after ESC as a regular
                    ;        data char, should go to COUT2).
      = 0010        VALETS: =       (:end-VALIDE)/3         ;# of entries in table
```

Copyright ©1982 Osborne Computer Corporation


```
03F6                VALIDC:
                    ;Valid control character table
                    ;       3 bytes per entry: Ascii char , "DW"- Vector
                    ;       no. of entries is VALCTS
                    ;       Following body of table is 2 byte No-Match adrs
03F6  0D                    DB      CR      ! DW  VC_CR     ;carriage return routine
03F9  0A                    DB      LF      ! DW  VC_LF     ;line feed
03FC  08                    DB      BKS     ! DW  VC_BKS    ;back space
03FF  0C                    DB      MCRIGH  ! DW  VC_MCRT   ;move cursor right
0402  0B                    DB      MCUP    ! DW  VC_MCUP   ;move cursor up
0405  07                    DB      CBELL   ! DW  VC_BEL    ;Ring bell
0408  1A                    DB      VCLRS   ! DW  VC_CLRS   ;clear screen
040B  1E                    DB      VHOME   ! DW  VC_HOME   ;Cursor Home

040E  9E05                  DW      VOUT97          ;No match--ignore undef control char

      = 0008        VALCTS: =       ((*-2)-VALIDC)/3        ;Number of valid entries
```

Copyright ©1982 Osborne Computer Corporation

```
0410                    COUT:
                        ;General output routine to Video Screen
                        ;ENTRY
                        ;C=Character,  CURS=Cursor,  ESCH=Flag+Mode

                        ;EXIT
                        ;CURS & ESCH updated,  A=Character
                        ;(bc, de, hl preserved)

                        ;       ESCH is flag + mode byte as follows
                        ;       =00      Normal mode & Last chr Esc flag false
                        ;       =08      Normal mode & Last chr Esc flag True
                        ;       =01,02,04 Mode is Graphics, Half, or Under, respectively
                        ;   , and Last chr Esc flag is False.
                        ;       =3,5,6,7    As above, but mode is combination
                        ;       =9-15     Last chr Esc flag True;otherwise like 1-7.

0410                            PROC

                        ;RESET VETICAL OFFSET WITH VRTOFF

0410  F5                        PUSH    AF
0411  C5                        PUSH    BC
0412  3A62EF                    LD      A,PIABD         ;PRESENT VALUE
0415  E6E0                      AND     11100000B       ;HOUSEKEEPING
0417  47                        MOV     B,A
0418  3AEFEF                    LD      A,VRTOFF        ;LAST VERTICAL OFFSET
041B  E61F                      AND     00011111B       ;ONLY VIDIO
041D  B0                        ORA     B               ;ADD HOUSEKEEPING
041E  4F                        MOV     C,A
041F  CD7109                    CALL    CPBD            ;SET OFFSET
0422  C1                        POP     BC
0423  F1                        POP     AF

0424  E5                        PUSH    HL
0425  D5                        PUSH    DE
0426  C5                        PUSH    BC
0427  2A5AEF                    LD      HL,CURS         ;HL will usually be cursor/
042A  3A60EF                    LD      A,ESCH
042D  47                        MOV     B,A             ;B will be ESCH for a while
042E  E608                      AND     EF_ESC          ;test flag bit
0430  2023 ^0455$               JRNZ    PSTESC          ;IF last chr was ESC

                        ;Current chr is NOT ESCaped. Is this chr ESC?

0432  79                        MOV     A,C             ;Chr
0433  FE1B                      CMP     ESC
0435  78                        MOV     A,B             ;(A=ESCH)
0436  2815 ^044D$               JRZ     :ESC            ;if this chr = ESC
```

```
                        ;Here with A=B  =       ESCH
0438                    :out:
0438  E5                        PUSH    HL
0439  21B403                    LDK     HL,ESCHTB
043C  E607                      AND     EF_MSK          ;Mode bits only
043E  87                        ADD     A,A             ;Times two
043F  5F                        MOV     E,A
0440  1600                      LDK     D,0             ;DE = offset
0442  19                        ADD     HL,DE           ;HL = tbl addrs
0443                    VECTOR:
                                                        ;entry point.  note hl on stack.
0443  7E                        LD      A,[HL]          ;1st byte (low order adrs)
0444  23                        INC     HL
0445  66                        LD      H,[HL]          ;2nd byte (hi order adrs)
0446  6F                        MOV     L,A             ;HL=adrs from table
0447  E3                        XTHL                    ;Restore hl from stack ;stack=tbl adrs
0448  79                        MOV     A,C             ;Chr.  note B=ESCH byte value
0449  C9                        RET                     ;enter routine per table adrs.

044A                    COUT2:
044A  78                        MOV     A,B             ;recall ESCH value
044B  18EB ^0438$               JR      :out            ;output chr per current settings

044D                    :ESC:
                        ;Current chr is ESC. Set flag and exit
044D  F608                      OR      EF_ESC          ;indicate last char= ESC
044F  3260EF                    STO     A,ESCH
0452  C39E05                    JMP     VOUT97          ;Exit
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
0455                PSTESC:
                    ;Last chr was ESC
                    ;Entry
                    ;A      =       EF_ESC
                    ;B      =       ESCH
                    ;C      =       Char to output
                    ;HL     =       curs
0455                        Proc
0455  CB60       $          BIT     4,B             ;is this chr really an address?
0457  2075 ^04CE$           JRNZ    SETXY           ;...if chr is part of an addr

                    ;no cursor/screen addressing in effect:
0459  A8                    XOR     B               ;Clr EF_ESC bit (for next time)
045A  47                    MOV     B,A             ;Set up B = ESCH byte value.
045B  3260EF                STO     A,ESCH
045E  E5                    PUSH    HL              ;save Curs
045F  21C403                LDK     HL,VALIDE       ;Branch table adrs
0462  1E10                  LDK     E,VALETS        ;Table size
0464  79                    MOV     A,C             ;Chr to A
0465  1815 ^047C$           JR      LOOKUPB         ;Go to routine to branch per tbl
```

Copyright © 1982 Osborne Computer Corporation

```
0467                    VNORM:
                        ;NORMAL mode character processing.
                        ;ENTRY
                        ;A       =       char to output
                        ;HL=curs
0467  FE20                      CMP     ' '
0469  380B ^0476$               JRC     :2              ;IF control chr
046B                    VBRIGH:
046B                            DI
046C                            ENADIM          ;9th bit memory
046E  3680                      STO     BRTBIT,[hl]             ;set this chr BRIGHT
0470                            DISDIM
0472                            EI
0473  C38105                    JMP     VOUT80

0476                    :2:
0476  E5                        PUSH    HL              ;Save Curs
0477  21F603                    LDK     HL,VALIDC       ;Branch table adrs
047A  1E08                      LDK     E,VALCTS        ;Table size
                        ;       JMP     LOOKUPB         ;Scan table of valid control chrs
                        ;                               ;and branch to appropriate routine.
```

Copyright © 1982 Osborne Computer Corporation

```
047C                LOOKUPB:
                    ;Logic to scan 3 byte branch table
                    ;       NOT a subroutine---do not CALL.
                    ;ENTRY
                    ;HL =1st byte of table (match code)
                    ;       (2nd,3rd bytes = branch adrs)
                    ;       (table repeats [3 byte entries])
                    ;E is table size (no. of entries)
                    ;       (table body is followed with
                    ;       2 byte "No-Match" adrs)
                    ;Stack has HL saved as top entry.
                    ;C      =       char
                    ;A      =       value to scan for possible match
047C  BE                    CMP     [HL]
047D  23                    INC     HL      ;(2nd byte of this 3 byte entry)
047E  28C3 ^0443$           JRZ     VECTOR  ;If match process
0480  23                    INC     HL      ;(3rd byte of this entry)
0481  23                    INC     HL      ;1st byte of next entry
0482  1D                    DEC     E       ;Dec count of entries remaining
0483  20F7 ^047C$           JRNZ    LOOKUPB ;Continue thru body of table
0485  188C ^0443$           JR      VECTOR  ;No-Match.  hl=points to vector
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;       Processing for modes other than normal.
                        ;       --------------------------------------

                        ;VGRAPH =       VOUT80  ;Normal mode EXCEPT: cntl chrs are printed

0487                    VUNDER:
                        ;Underline only
0487  FE20                      CMP     ' '
0489  38DC ^0467$               JRC     VNORM           ;if cntl-chr, process as normal
                        ;       JR      VUN_GR          ;continue

048B                    VUN_GR:
                        ;Underlined Graphics
048B  F680                      OR      80h             ;underline bit
048D  18DC ^0468$               JR      VBRIGH          ;set this chr BRIGHT

048F                    VUN_HA:
                        ;Underline and Half intensity
048F  FE20                      CMP     ' '
0491  38D4 ^0467$               JRC     VNORM           ;if cntl-chr, process as normal
                        ;       JR      VUN_HA_GR

0493                    VUN_HA_GR:
                        ;Underline, Half Intensity, Graphics
0493  F680                      OR      80h     ;set underline bit
                        ;       JR      VHA_GR

0495                    VHALF:
                        ;Half Intensity
0495  FE20                      CMP     ' '
0497  38CE ^0467$               JRC     VNORM           ;if cntl-chr, process as normal
                        ;       JR      VHA_GR

0499                    VHA_GR:
                        ;C=Chr, HL=Curs
0499                            DI
049A                            ENADIM
049C  3600                      STO     DIMBIT,[hl]             ;set dim field bit
                        ;       LD      E,[hl]                  ;diagnostic
049E                            DISDIM
04A0                            EI
04A1  C38105                    JMP     VOUT80          ;continue
```

Copyright © 1982 Osborne Computer Corporation


```
04A4                    SCREEN:
                        ;SetXY for Screen movement
                        ;ENTRY
                        ;B          =       ESCH
                        ;A          =       new co-ord val, NO OFFSET
04A4                                Proc
04A4  CB70       $                  BIT     6,B
04A6  2012 ^04BA$                   JRNZ    :sX      ;if X-coordinate

04A8                    :sY:
04A8  E61F                          AND     0001_1111b       ;mod 32
04AA  4F                            MOV     C,A
04AB  32EFEF                        STO     A,VRTOFF         ;SET VERTICAL OFFSET FOR COUT
04AE  3A62EF                        LD      A,PIABD
04B1  E6E0                          AND     1110_0000b
04B3  B1                            OR      C
04B4  4F                            MOV     C,A
04B5  CD7109                        CALL    OPBD             ;set Y coordinate
04B8  182E ^04E8$                   JR      :exitY

04BA                    :sX:
04BA  87                            ADD     A,A              ;double A
04BB  C6EA                          ADD     A,VFLO           ;PIA A-reg magic offset constant
04BD  E6FE                          AND     1111_1110b       ;clear bit 0
04BF  4F                            MOV     C,A

                        ;SET DENSITY BIT

04C0  3A61EF                        LD      A,PIAAD          ;GET OLD VALUE
04C3  E601                          ANI     0000_0001B       ;SAVE DENSITY BIT
04C5  B1                            ORA     C                ;OR IN HORIZONTAL OFFSET
04C6  4F                            MOV     C,A
04C7  CD6409                        CALL    OPAD             ;FUNCTION PIA
04CA  CBA8       $                  CBIT    5,B              ;finished screen-addressing
04CC  1824 ^04F2$                   JR      :exitX
```

Copyright ©1982 Osborne Computer Corporation

```
04CE                    SETXY:
                        ;Set X-Y value for Cursor/Screen Addressing
                        ;ENTRY
                        ;HL      =       cursor_addr
                        ;B       =       ESCH
                        ;C       =       chr

                        ;EXIT
                        ;to VOUT90; ESCH updated
04CE  CD1907                    CALL    UN_CUR
04D1  3EE0                      LDK     A,-(' ')
04D3  81                        ADD     C       ;remove cursor bias
04D4  CB68       $              BIT     5,B     ;cursor/screen addressing?
04D6  20CC ^04A4$               JRNZ    SCREEN  ;if screen addressing

                        ;cursor addressing:
04D8  29                        ADD     HL,HL   ;shift HL left
04D9  CB70       $              BIT     6,B     ;X/Y coordinate?
04DB  2010 ^04ED$               JRNZ    :cX     ;if X coordinate

04DD                    :cY:
04DD  67                        MOV     H,A     ;save
04DE  3A62EF                    LD      A,PIABD
04E1  84                        ADD     H       ;offset by start-Y co-ord of video
04E2  1F                        RAR             ;bit0(A) -> CY, shift A right
04E3  CB1D       $              RR      L       ;CY -> bit7(L)
04E5  F6F0                      OR      0F0h    ;turn on upper nybl
04E7  67                        MOV     H,A     ;HL= new cursor addr
                        ;       JR      :exitY

04E8                    :exitY:
04E8  3E40                      LDK     A,0100_0000b    ;next addr-chr will be X-coord
04EA  B0                        OR      B
04EB  1808 ^04F5$               JR      :exit2

04ED                    :cX:
04ED  17                        RAL             ;trash 7th bit
04EE  CB2C       $              SRA     H       ;bit0(H) -> CY, bit7 stays 1
04F0  1F                        RAR             ;... CY -> bit7(A)
04F1  6F                        MOV     L,A
                        ;       JR      :exitX

04F2                    :exitX:
04F2  3E07                      LDK     A,EF_MSK
04F4  A0                        AND     B       ;finished addressing: reset addr bits
04F5                    :exit2:
04F5  3260EF                    STO     A,ESCH
04F8  C39605                    JMP     VOUT90
```

Copyright © 1982 Osborne Computer Corporation


```
                        ;       Control Code character processing

04FB                    VC_HOME: ;Home Cursor
04FB                            Proc
04FB  CD1907                    CALL    UN_CUR
04FE  3A62EF                    LD      A,PIABD
0501  1F                        RAR                     ;bit0 => CY
0502  2E00                      LDK     L,0
0504  CB1D       $              RR      L               ;CY => bit7, trash bit0
0506  67                        MOV     H,A
0507  182F ^0538$               JR      :fixhl          ;HL := HL or F000h
```

Copyright © 1982 Osborne Computer Corporation

```
0509                    VC_MCUP:        ;Move Cursor Up.
                        ;A=C=Chr=MCUP.  HL=Curs.
0509  CD1907                    CALL    UN_CUR
050C  E5                        PUSH    HL              ;old cursor must be on stack
050D  0180FF                    LDK     BC,(-VLL)
0510  1816 ^0528$               JR      :fwa            ;...at this entry point



0512                    VC_BKS:
                        ;HL=Curs=current (old) char
0512  CD1907                    CALL    UN_CUR          ;clear 80h bit
0515  3E7F                      LDK     A,7Fh
0517  A5                        AND     L
0518  2803 ^051D$               JRZ     :wrap           ;if must wrap from col 0 to LLIMIT
051A  2B                        DEC     HL
051B  1879 ^0596$               JR      VCUT90          ;Exit

051D                    :wrap:
051D  E5                        PUSH    HL              ;save old cursor
051E  017FFF                    LDK     BC,-(VLL+1)
0521  09                        ADD     HL,BC           ;HL = prev_line, (-1)st column
0522  3A6CEF                    LD      A,LLIMIT        ;LLIMIT = #columns on screen
0525  4F                        MOV     C,A
0526  0600                      LDK     B,0
0528                    :fwa:
0528  09                        ADD     HL,BC
0529  E3                        XTHL                    ;get old cursor, save new
052A  29                        ADD     HL,HL           ;shift line# into H reg.
052B  3A62EF                    LD      A,PIABD
052E  F6E0                      OR      1110_0000b      ;A = line# of UL corner
0530  BC                        CMP     H               ;set Zflag: @home?
0531  E1                        POP     HL              ;get new cursor...
0532  2004 ^0538$               JRNZ    :fixhl          ;if NOT @video home
0534  01000C                    LDK     BC,(24*VLL)     ;wrap constant
0537  09                        ADD     HL,BC
0538                    :fixhl:
0538  3EF0                      LDK     A,0F0h
053A  B4                        OR      H               ;modulo result: keep cursor
053B  67                        MOV     H,A             ;inside video memory.
053C  1858 ^0596$               JR      VOUT90
```

```
053E                    VC_BEL:
                        ;Ring the bell via setting PIAB 2**5 bit
053E  3A62EF                    LD      A,PIABD
0541  F620                      OR      0010_0000b      ;bell bit
0543  4F                        MOV     C,A
0544  CD7109                    CALL    OPBD            ;function PIAB
0547  3E1E                      LDK     A,30            ;ring bell for 30 ticks
0549  326BEF                    STO     A,BELCNT        ;... = 1/2 second
054C  1850 ^059E$               JR      VOUT97          ;exit no change
```

Copyright ©1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
054E  2100F0       VC_CLRS: LDK     HL,FWAVM
0551  CD8406                CALL    CLRLN           ;clear 1st line
0554  01800F                LDK     BC,LVMEM-VLL
0557  D5                    PUSH    DE
0558  DDE1        $         POP     IX
055A  CD0507                CALL    VLDIR           ;clear remaining lines
055D  3A62EF                LD      A,PIABD         ;Reset for 1st line of display mem
0560  E6E0                  AND     not(1_1111b)
0562  4F                    MOV     C,A
0563  CD7109                CALL    OPBD
0566  AF                    XRA     A               ;ZERO A
0567  32EFEF                STO     A,VRTOFF        ;SET VERTICAL OFFSET FOR COUT
056A  2100F0                LDK     HL,FWAVM        ;new cursor
056D  1827 ^0596$           JR      VCOUT90         ;Exit
```

```
056F                VC_CR:
056F  CD1907                CALL    UN_CUR          ;erase cursor
0572  3E80                  LDK     A,80h           ;Carriage Return
0574  A5                    AND     L
0575  6F                    MOV     L,A
0576  181E ^0596$           JR      VOUT90

0578                VC_LF:
0578  CDA706                CALL    DO_LF           ;Line Feed
057B  1819 ^0596$           JR      VOUT90
```

Copyright © 1982 Osborne Computer Corporation

```
057D                VC_MCRT ;Move Cursor Right
057D  CD1907                CALL    UN_CUR
0580  7E                    LD      A,[hl]
                    ;       JR      VOUT80          ;re-echo current chr
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        ;       Exit points for COUT

                        ;       Here to store new data and to update cursor
0581                    VGRAPH:
0581                    VOUT80:
0581  77                        STO     A,[hl]          ;This exit path stores A (new char)
0582  5D                        MOV     E,L
0583  CBBB      $               CBIT    7,E             ;E = col(cursor)
0585  3A6CEF                    LD      A,LLIMIT
0588  3D                        DEC     A               ;A = last_legal_col
0589  93                        SUB     E               ;A = last_legal_col - col(cur)
058A  2009 ^0595$               JRNZ    VOUT85          ;if not @LLIMIT
058C  3E80                      LDK     A,80h
058E  A5                        AND     L
058F  6F                        MOV     L,A             ;do CR...
0590  CDAA06                    CALL    DO_LF2          ;...and LF.
0593  1801 ^0596$               JR      VOUT90

0595                    VOUT85:
0595  23                        INC     HL              ;move cursor

                        ;       Here if NO cursor update
0596                    VOUT90:
0596  7E                        LD      A,[HL]          ;This exit path turns on 80h bit

                        ;       Here if new data already in A
0597                    VOUT95:
0597  17                        RAL                     ;Make this chr cursor
0598                    VOUT96:
0598  3F                        CMC                     ;invert cursor bit
0599  1F                        RAR
059A  77                        STO     A,[hl]
059B  225AEF                    STO     HL,CURS         ;update cursor

                        ;       Here if no change to cursor, restore reg and exit
059E                    VOUT97:
059E  C1                        POP     BC
059F  D1                        POP     DE
05A0  E1                        POP     HL
05A1  79                        MOV     A,C             ;Exit with A=chr
05A2  C9                        RET                     ;return, end of cout subr.

      = 051D            :First  =       VOUT97 - (127 + 2)      ;earliest possible JR

      = 05FF            :Last   =       VOUT80 + (128 - 2)      ;latest possible JR
```

Copyright © 1982 Osborne Computer Corporation

```
05A3                ESC_LCK: ;Lock Keyboard
05A3                        Proc
05A3  AF                    XOR     A
05A4  1802 ^05A8$           JR      :2

05A6                ESC_ULK: ;Unlock Keyboard
05A6  3EFF                  LDK     A,0FFh
05A8                :2:
05A8  3259EF                STO     A,KEYLCK
05AB  18F1 ^059E$           JR      VOUT97
```

Copyright © 1982 Osborne Computer Corporation

```
05AD                 EEOL:
                     ;Erase to end of line
05AD  E5                     PUSH    HL      ;save cursor
05AE  CD8406                 CALL    CLRLN
05B1  E1                     POP     HL
05B2  18E2 ^0596$            JR      VOUT90
```

Copyright © 1982 Osborne Computer Corporation


```
05B4                    ESC_CAD:
                        ;Cursor Addressing
05B4  3E07                      LDK     A,EF_MSK
05B6  A0                        AND     B
05B7  F618                      OR      EF_ESC or EF_ADR        ;next chr will be Y-coord
05B9                    :exit3:
05B9  3260EF                    STO     A,ESCH
05BC  18E0 ^059E$               JR      VOUT97

05BE                    ESC_SAD:
                        ;Screen Addressing
05BE  3E07                      LDK     A,EF_MSK
05C0  A0                        AND     B
05C1  F638                      OR      EF_ESC or EF_ADR or EF_SCR
05C3  18F4 ^05B9$               JR      :exit3
```

```
                    ;Subroutine for use with EDELC and EINSRT:
                    ;Calculate #chrs to move; if move zero chrs, never return.
05C5                        Proc
05C5                :calc:
05C5  3E7F                  LDK     A,VLL-1 ;A= max #chrs to be moved
05C7  4D                    MOV     C,L
05C8  CBB9      $           CBIT    7,C     ;C = col(cursor)
05CA  91                    SUB     C       ;A = #chrs to move
05CB  2804 ^05D1$           JRZ     :end    ;if move zero characters
05CD  4F                    MOV     C,A
05CE  0600                  LDK     B,0     ;BC = #chrs to move
05D0  C9                    RET

05D1                :end:
05D1  E1                    POP     HL      ;trash return_addr
05D2  E1                    POP     HL      ;cursor_addr
05D3  18C1 ^0596$           JR      VOUT90
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
05D5                      EDELC:
                          ;Delete Character
05D5  E5                          PUSH    HL              ;save cursor_addr
05D6  CDC505                      CALL    :calc           ;calculate BC
                          ;       MOV     D,H
                          ;       MOV     E,L             ;DE = cursor_addr
05D9  E5                          PUSH    HL
05DA  DDE1          $             POP     IX
05DC  23                          INC     HL              ;HL = cursor_addr + 1
05DD  CD0507                      CALL    VLDIR           ;move characters
05E0  3620                        STO     ' ',[hl]        ;last chr becomes blank
05E2                              DI
05E3                              ENADIM          ;enable 9th bit memory
05E5  2B                          DEC     HL              ;HL = last chr on this line
05E6  3680                        STO     BRTBIT,[hl]             ;set chr BRIGHT
05E8                              DISDIM
05EA                              EI              ;main memory
05EB  E1                          POP     HL              ;restore cursor_addr
05EC  18A8 ^0596$                 JR      VOUT90          ;next
```

Copyright © 1982 Osborne Computer Corporation

```
05EE                EINSRT:
                    ;Insert Character

05EE  CD1907                CALL    UN_CUR
05F1  E5                    PUSH    HL      ;save cursor_addr
05F2  CDC505                CALL    :calc   ;calculate BC
05F5  3E7F                  LDK     A,7Fh
05F7  B5                    OR      L
05F8  6F                    MOV     L,A     ;HL = last_chr on this line
05F9  E5                    PUSH    HL
05FA  DDE1        $         POP     IX
05FC  2B                    DEC     HL
05FD  CDF106                CALL    VLDDR   ;do move
0600  E1                    POP     HL      ;restore cursor
0601  7E                    LD      A,[hl]  ;get underline bit of this chr.
0602  17                    RAL             ;into CY
0603  3E40                  LDK     A,' ' shl 1     ;change this chr to ' '
0605  1891 ^0598$           JR      VOUT96          ;exit
```

Copyright © 1982 Osborne Computer Corporation


```
                       ;ESC-Sequence processing.
0607                            Proc
0607                   ESCSGR:
0607  3E01                      LDK     A,EF_GR         ;ESC-g
0609  1806 ^0611$               JR      :125            ;set graphics mode.

060B                   ESCSHA:
060B  3E02                      LDK     A,EF_HA         ;ESC-)  set half intensity
060D  1802 ^0611$               JR      :125            ;go set flag bit

060F  3E04             ESCSUN:  LDK     A,EF_UN         ;ESC-l  set underline
0611                   :125:
0611  B0                        OR      B               ;Reg B is ESCH Byte value
0612                   :130:
0612  3260EF                    STO     A,ESCH          ;store desired value.
0615  1887 ^059E$               JR      VOUT97          ;Exit

0617  3EFE             ESCCGR:  LDK     A,NOT EF_GR     ;ESC-G  Clear graphics mode
0619  1806 ^0621$               JR      :140            ;go clear ESCH bit

061B  3EFD             ESCCHA:  LDK     A,NOT EF_HA     ;ESC-(  Clear half intensity
061D  1802 ^0621$               JR      :140

061F  3EFB             ESCCUN:  LDK     A,NOT EF_UN     ;ESC-m  Clear underline
0621                   :140:
0621  A0                        AND     B               ;Clear bit
0622  18EE ^0612$               JR      :130            ;Go store ESCH byte

      = 054E           ESCZZ:   =       VC_CLRS ;ESC-Z  Clear screen -same as
                                                        ;Control-Z routine.
```

Copyright © 1982 Osborne Computer Corporation

```
0624                    ESCRR:
                        ;Delete Line

0624                    ESCEE:
                        ;Insert Line
                        ;ENTRY
                        ;HL      =       cursor
                        ;C       =       chr

                        ;EXIT
                        ;screen updated
                        ;HL      =       new cursor ...to VOUT90
0624                            Proc
0624  CD1907                    CALL    UN_CUR
0627  3E80                      LDK     A,1000_0000b
0629  A5                        AND     L
062A  6F                        MOV     L,A             ;do CR
062B  E5                        PUSH    HL              ;save new cursor
062C  29                        ADD     HL,HL
062D  3A62EF                    LD      A,PIABD
0630  C618                      ADD     A,24            ;A = addr(25th line)
0632  94                        SUB     H               ;A = lines_to_move + 1
0633  E61F                      AND     0001_1111b              ;mod 32
0635  47                        MOV     B,A
0636  3E52                      LDK     A,VDELL
0638  B9                        CMP     C
0639  78                        MOV     A,B             ;recall #lines to move
063A  2823 ^065F$               JRZ     :delt           ;if deleting a line

063C                    :insrt:
                        ;Insert a line

063C  84                        ADD     H               ;A = addr(25th line)
063D  57                        MOV     D,A
063E  1E00                      LDK     E,0
0640  CB1A       $              RR D
0642  CB1B       $              RR E            ;shift right DE
                                                ;DE = addr(1st_chr_on_1st_line)
0644  1B                        DEC     DE
0645  78                        MOV     A,B             ;A = #lines to move
0646  2180FF                    LDK     HL,-VLL
0649  19                        ADD     HL,DE           ;HL = addr(line above DE)
                                                ;DE := DE or F000h; HL := HL or F000h
064A  1806 ^0652$               JR      :istrt

064C                    :icont:
064C  D5                        PUSH    DE
064D  DDE1       $              POP     IX
064F  CDF106                    CALL    VLDDR           ;move 1 line down
0652                    :istrt:
0652  CD7506                    CALL    :vmod
0655  20F5 ^064C$               JRNZ    :icont          ;if must move more lines
0657  23                        INC     HL              ;HL => 1st chr of new line
```

Copyright © 1982 Osborne Computer Corporation

```
0658                :exit:
0658  CD8406               CALL    CLRLN
065B  E1                   POP     HL              ;recover cursor
065C  C39605               JMP     VCUT90  ;Main Exit

065F                :delt:
065F  D1                   POP     DE              ;recover new cursor
0660  D5                   PUSH    DE
0661  218000               LDK     HL,VLL
0664  19                   ADD     HL,DE           ;HL = line_below_cursor
0665  1806 ^066D$          JR      :dstrt

0667                :dcont:
0667  D5                   PUSH    DE
0668  DDE1      $          POP     IX
066A  CD0507               CALL    VLDIR           ;move 1 line up
066D                :dstrt:
066D  CD7506               CALL    :vmod
0670  20F5 ^0667$          JRNZ    :dcont
0672  EB                   EX      HL,DE           ;get addr of line to clear
0673  18E3 ^0658$          JR      :exit
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;HL := HL or F000h; DE := DE or F000h;
                        ;simple mod-4096 arithmetic to keep pointers INSIDE video memory
0675                    ;vmod:
0675  F5                        PUSH    AF              ;save A = #lines to move
0676  3EF0                      LDK     A,0F0h
0678  B4                        OR      H
0679  67                        MOV     H,A             ;set upper nybl of H
067A  3EF0                      LDK     A,0F0h
067C  B2                        OR      D
067D  57                        MOV     D,A             ;modulo 4096
067E  018000                    LDK     BC,VLL
0681  F1                        POP     AF
0682  3D                        DEC     A               ;decrement line_count
0683  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0684                CLRLN:
                    ;Clear to end of line
                    ;ENTRY
                    ;HL      =       Cursor

                    ;EXIT
                    ;clear to EOL
                    ;        Uses    All.
0684                         PROC
0684  3620                   STO     ' ',[hl]        ;clear cursor...
0686                         DI
0687                         ENADIM
0689  3680                   STO     BRTBIT,[hl]             ;set cursor BRIGHT
068B                         DISDIM
068D                         EI
068E  3A6CEF                 LD      A,LLIMIT
0691  3D                     DEC     A               ;max_#cols => maximum_col_#
0692  5D                     MOV     E,L
0693  CBBB       $           CBIT    7,E
0695  93                     SUB     E               ;A = col(EOL) - col(cursor)
0696  C8                     RZ                      ;if @EOL, done

0697  3004 ^069D$            JRNC    :2              ;if inside logical_video_line

0699  3E80                   LDK     A,VLL
069B  93                     SUB     E               ;...else clr to end of 128-chr line
069C  C8                     RZ                      ;if cursor @ column #127

069D                :2:
069D  4F                     MOV     C,A
069E  0600                   LDK     B,0             ;BC = chrs to move
                    ;        MOV     E,L
                    ;        MOV     D,H
06A0  E5                     PUSH    HL
06A1  DDE1       $           POP     IX
                    ;        INC     DE              ;DE = HL + 1
06A3  DD23       $           INC     IX
06A5  185E ^0705$            JR      VLDIR
```

Copyright © 1982 Osborne Computer Corporation

```
06A7                    DO_LF:
                        ;Do Line Feed processing
                        ;ENTRY
                        ;HL      =        cursor_addr

                        ;EXIT
                        ;Cursor cleared
                        ;HL updated for current cursor pos
                        ;window moved if necessary
06A7                            PROC
06A7    CD1907                  CALL    UN_CUR          ;clear cursor
06AA                    DO_LF2:
06AA    E5                      PUSH    HL      ;save original cursor
06AB    018000                  LDK     BC,VLL  ;line length
06AE    09                      ADD     HL,BC
06AF    3004 ^06B5$             JRNC    :nowap  ;if not wrapping from LWAVM to FWAVM
06B1    0100F0                  LDK     BC,FWAVM
06B4    09                      ADD     HL,BC   ;HL = new cursor @ top of VM
06B5                    :nowap:
06B5    E3                      XTHL            ;save new cursor, get old
06B6    29                      ADD     HL,HL   ;shift HL left
06B7    3A62EF                  LD      A,PIABD
06BA    C617                    ADD     A,23    ;start + 23 = last_video_line
06BC    94                      SUB     H       ;A = l_line - curr_line
06BD    E61F                    AND     0001_1111b      ;modulo 32
063F    2802 ^06C3$             JRZ     :vmov   ;if cursor is on 24th line of screen
06C1                    :end:
06C1    E1                      POP     HL      ;get new cursor
06C2    C9                      RET

06C3                    :vmov:
06C3    3A6CEF                  LD      A,LLIMIT
06C6    CB3D        $           SRL     L       ;unshift L register
06C8    95                      SUB     L       ;A = LLIMIT - col(cursor)
06C9    38F6 ^06C1$             JRC     :end    ;if cursor is outside logical line

                        ;cursor is on last line of screen, inside of logical line.
                        ;must move screen to follow cursor down through video memory.
06CB    E1                      POP     HL
06CC    E5                      PUSH    HL
06CD    3E80                    LDK     A,80h
06CF    A5                      AND     L
06D0    6F                      MOV     L,A     ;HL = beginning of line
06D1    CD8406                  CALL    CLRLN   ;erase to EOL
06D4    3A62EF                  LD      A,PIABD
06D7    47                      MOV     B,A
06D8    E6E0                    AND     not 31  ;A = line zero
06DA    4F                      MOV     C,A     ;C = housekeeping bits 5..7
06DB    3E1F                    LDK     A,31
06DD    04                      INC     B       ;increment line#
06DE    A0                      AND     B       ;A = line#
06DF    32EFEF                  STO     A,VRTOFF        ;SET VERTICAL OFFSET FOR COUT
06E2    B1                      OR      C       ;A = new line# OR housekeeping_bits
06E3    4F                      MOV     C,A     ;C = new value for OPBD
06E4    CD7109                  CALL    OPBD    ;move video screen down 1 line in memory
06E7    E1                      POP     HL
06E8    C9                      RET
```

Copyright © 1982 Osborne Computer Corporation


```
06E9                STODIM:
                    ;STORE THE CONTENTS OF THE B REG IN THE ADDR POINTED TO BY THE HL PAIR
                    ;ENTRY
                    ;B      =       VALUE
                    ;HL     =       ADDRESS

                    ;EXIT
                    ;NONE
06E9                        PROC
06E9                        DI
06EA                        ENADIM                  ;ENABLE DIM
06EC   70                   STO     B,[HL]          ;STORE
06ED                        DISDIM                  ;DISABLE DIM
06EF                        EI
06F0   C9                   RET
```

Copyright © 1982 Osborne Computer Corporation

```
06F1                  VLDDR:
                      ;Video Block Move
                      ;ENTRY
                      ;BC, IX, HL set

                      ;EXIT
                      ;LDDR on main & 9th bit memory
                      ;         Uses     BC, DE, HL,IX

06F1  DDE5       $             PUSH     IX
06F3  D1                       POP      DE
06F4                           PUSHAL
06F7  EDB8       $             LDDR                       ;main memory
06F9                           POPALL
06FC                           DI
06FD                           ENADIM
06FF  EDB8       $             LDDR                       ;9th bit memory
0701                           DISDIM
0703                           EI
0704  C9                       RET

0705                  VLDIR:
                      ;Video Block Move
                      ;ENTRY
                      ;BC, IX, HL set

                      ;EXIT
                      ;LDIR on main & 9th bit memory
                      ;         Uses     BC, DE, HL,IX

0705  DDE5       $             PUSH     IX
0707  D1                       POP      DE
0708                           PUSHAL
070B  EDB0       $             LDIR                       ;main memory
070D                           POPALL
0710                           DI
0711                           ENADIM
0713  EDB0       $             LDIR                       ;9th bit memory
0715                           DISDIM
0717                           EI
0718  C9                       RET
```

Copyright © 1982 Osborne Computer Corporation

```
0719              UN_CUR:
                  ;Undo/Invert Cursor
                  ;ENTRY
                  ;HL      =       cursor_addr

                  ;EXIT
                  ;cursor inverted
                  ;       Uses    A, CY.
0719  7E                  LD      A,[hl]          ;get the chr
071A  17                  RAL                     ;cursor_bit => CY
071B  3F                  CMC                     ;invert it
071C  1F                  RAR
071D  77                  STO     A,[hl]          ;...
071E  C9                  RET
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;KEYBOARD SCANNING & DECODE

071F                    GKEY:
                        ;KEYBOARD INTERRUPT PROCESSOR
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;KEYBOARD PROCESSING DONE, RESULT IN LDKEY.
                        ;SYSTEM CLOCK UPDATED BY UPTIM.

071F                            PROC

071F                            DI
0720  ED736FEF  $               STO     SP,IESTK        ;SAVE INTERRUPTED PROCESS STK
0724  3199EF                    LDK     SP,ISTK         ;SET TO RAM INT STK
0727                            ALPUSH

072F  CD9803                    CALL    UPTIM           ;UPDATE SYSTEM CLOCK

0732  3A59EF                    LD      A,KEYLCK
0735  B7                        OR      A
0736  C43C07                    CNZ     KBDRVR
0739                    GKEYX:
0739  C31E00                    JMP     EXITI           ;EXIT INTERRUPT
```

Copyright ©1982 Osborne Computer Corporation

```
;       This file contains the 2-key roll over keyboard driver for
;       the OSBCRNE 1 comuter.

;       Author:
;       Microcode Corporation.
;       Fremont, CA.
;       Y. N. Sahae
;       September 1981

;       Revisions:

;       2-Key roll over keyboard driver.

;       DESCRIPTION:
;       The keyboard driver gets control via the 60hz interrupt, i.e. once
;       every 16 ms.  It scans the keyboard  to detect any struck keys.  If a
;       key is found, it is entered into the keylist if there is space
;       in the keylist and the key is not already in the list.   At the end of
;       the scan, the keys in the list are proecessed.  If the key is still
;       on,  it is placed in lkey (or special action taken) after translating
;       the keynumber.  A count is also stored in the list and the key will
;       be serviced again at the end of this count if it is still on.  Thus
;       the key will repeat if it is held down.  If a key which is in the
;       list is not on it is removed from the list.

;       The keyboard driver consists of the following routines:

;       KBDRVR - Examines the keylist, calls CHKEY to determine if key
;       is still on.  Removes the key from the list if it is not on.  If
;       key is on, it decrements the count associated with the key.  when
;       the count goes to zero, it calls KBSERV to service the key.  Calls
;       KBSCAN to enter any new keys into the list.

;       KBSCAN - This routine scans the keyboard, detects a struck
;       key and enters it into the keylist.  The key is entered
;       into the keylist if the key is not already present in the keylist
;       and there is an empty slot in the keylist.

;       KBSERV - It calls the routine CHKEY to check if shift/ctl or alphlock
;       keys are on.  It then translates the keynumber into the ASCII
;       code and places it in the LKEY for the CBIOS to read.  For some
;       special cases, it calls ROM resident routines to process the key.

;       CHKEY - It checks if a given key is on.


;       Data structure:
;       The main structure used is the keylist.  The format of each entry is:

;       Byte 0:
;               bit 7 :  Set indicates entry is in use.
;               bit 6 :  Set indicates key has been serviced once.
;               bits 5..3 : contain the row number of struck key.
;               bits 2..0 : contain the column number of struck key.
;       Byte 1:
;               bits 7..0 : contain the repeat count for the key.
```

Copyright © 1982 Osborne Computer Corporation

```
073C                KBDRVR:
                    ;       Routine name: KBDRVR - Detects and processes keystrokes.
                    ;       Input: none
                    ;       Output: LKEY contains keystroke.

073C                        PROC
073C  CD7407                call    kbscan          ;scan keyboard and enter keys into keylist

                    ;       Examine keylist.  If key found in keylist, call CHKEY to see
                    ;       if key is still on.  remove from list when not on.

073F  21D4EF                LDK     hl,keylst       ;point to first entry of keylist
0742  0603                  LDK     b,kl_len
0744                :10:
0744  3A5EEF                ld      a,lkey
0747  B7                    or      a
0748  C0                    RNZ                     ;exit when a key is waiting in lkey

0749  7E                    ld      a,[hl]          ;get byte 0 of entry
074A  CB7F      $           bit     kl_used,a
074C  2821 ^076F$           jrz     :40             ;if entry is in use then
074E  CDD107                call    chkey           ;check if still on
0751  2004 ^0757$           jrnz    :20             ;if key is now off then
0753  3600                  sto     0,[hl]          ;remove key from list
0755  1818 ^076F$           jr      :40
                                                    ;else
0757                :20:
                    ;       key is on.  decrement its repeat count.  If count goes to zero
                    ;       then it is time to service the key.

0757  E5                    push    hl                  ;save ptr to first byte of entry
0758  23                    inc     hl                  ;point to repeat count
0759  35                    dec     [hl]
075A  2012 ^076E$           jrnz    :30                 ;exit when not time to service the key.

                    ;       it is time to service the key.  Set the next repeat count

075C  E3                    ex      [sp],hl             ;point back to the first byte of entry
075D  7E                    ld      a,[hl]
075E  CB77      $           bit     ky_srvd,a           ;check if it is serviced before
0760  CBF6      $           sbit    ky_srvd,[hl]        ;set the serviced once flag
0762  E3                    ex      [sp],hl             ;point back to the repeat count
0763  3618                  sto     irptct,[hl]         ;and store rpt count as per serviced flag
0765  2802 ^0769$           jrz     :22
0767  3606                  sto     srptct,[hl]
0769                :22:
0769  E63F                  and     krow_m+kcol_m
076B  CDF807                call    kbserv              ;call to service the key
076E                :30:
076E  E1                    pop     hl                  ;get ptr to first byte of entry again
                                                    ;endif
                                                ;endif
076F                :40:
                            echo    kle_len
                             inc    hl              ;point to next entry
                            ENDM
0771  10D1 ^0744$           djnz    :10             ;until  complete list scanned

0773  C9                    RET                     ;return to exiti code in rom
```

Copyright ©1982 Osborne Computer Corporation

```
                        ;**
                        ;       sbrt:   KBSCAN - Scan keyboard and enter detected keys in the
                        ;       keylist.
                        ;       input:
                        ;none
                        ;       output:
                        ;keylst = contains any keys detected.

0774                    KBSCAN: proc
0774  2EFF                      LDK     l,0ffh                  ;see if any key pressed
0776  CDEF07                    call    rdrow
0779  C8                        rz                              ;return when none

077A  2E81                      LDK     l,row0_m                ;get row 0
077C  CDEF07                    call    rdrow

                        ;       LDK     L,1
                        ;       CALL    RDROW
                        ;       PUSH    psw
                        ;       LDK     l,80h
                        ;       CALL    rdrow
                        ;       ANI     08h
                        ;       MOV     B,A
                        ;       POP     PSW
                        ;       OR      B

077F  E6E3                      and     11100011b               ;remove ctl/shift and alpha lock
0781  0607                      LDK     b,tot_row

                        ;       in this loop, reg b contains totrow- current row being scanned
0783                    :10:
0783  2844 ^07C9$               jrz     :50                     ;if any key is pressed then

0785  C5                        push    bc                        ;save loop count
0786  5F                        mov     e,a                       ;e = columns
0787  3E07                      LDK     a,tot_row
0789  90                        sub     b
078A  17                        ral
078B  17                        ral
078C  17                        ral
078D  57                        mov     d,a                       ;d = row number * 8
078E  0E00                      LDK     c,0                       ;initialize column counter

                        ;       scan this row from right to left to get the column number
0790                    :15:                                      ;repeat
0790  CB38        $             srl     e                           ;shift column bit into carry
0792  302F ^07C3$               jrnc    :30                         ;if a key is found then

                        ;       enter the key whose column number is in c and row*8 is in d
                        ;       into the keylst provided the key is not already in list and
                        ;       there is an empty slot in the list.

0794  7A                        mov     a,d
0795  81                        add     a,c
0796  C5                        push    bc
0797  4F                        mov     c,a                             ;c = key number
0798  D5                        push    de                              ;save de
0799  E5                        push    hl                              ;save hl
079A  0603                      LDK     b,kl_len                        ;length of keylist
079C  21D4EF                    LDK     hl,keylst
```

Copyright © 1982 Osborne Computer Corporation

```
079F  110000                   LDK    de,0
07A2                  :18:
07A2  7E                       ld     a,[hl]                    ;
07A3  CB7F        $            bit    kl_used,a                 ;if entry is used then
07A5  2807 ^07AE$              jrz    :22
07A7  E63F                     and    krow_m+kcol_m
07A9  B9                       cmp    c                           ;check with current key
07AA  2814 ^07C0$              jrz    :27                         ;exit if this key is in list
07AC  1802 ^07B0$              jr     :25
07AE                  :22:                                      ;else (an empty entry is found)
07AE  5D                       mov    e,l
07AF  54                       mov    d,h                         ;save adrs of empty entry in de
07B0                  :25:                                      ;endif
                               echo   kle_len
                                 inc  hl
                               ENDM                             ;next entry
07B2  10EE ^07A2$              djnz   :18                       ;till list scanned

                      ;        check if an empty entry was found.
07B4  7A                       mov    a,d
07B5  B7                       or     a
07B6  CAC007                   jz     :27                       ;if empty entry was found then
07B9  EB                       ex     de,hl                       ;hl = empty entry
07BA  71                       sto    c,[hl]                      ;store the key in the list
07BB  CBFE        $            sbit   kl_used,[hl]                ;set used flag
07BD  23                       inc    hl
07BE  3601                     sto    DB_ct,[hl]                  ;store debounce delay
                                                                ;endif
07C0                  :27:                                      ;restore all regsters
07C0  E1                       pop    hl
07C1  D1                       pop    de
07C2  C1                       pop    bc
07C3                  :30:                                    ;endif
07C3  0C                       inc    c                       ;increment column number
07C4  AF                       xor    a
07C5  BB                       cmp    e
07C6  20C8 ^0790$              jrnz   :15                   ;until all columns scanned

07C8  C1                       pop    bc                    ;rstore bc
07C9                  :50:                                ;endif
07C9  CB25        $            sla    l                   ;move to next row
07CB  CDEF07                   call   rdrow
07CE  10B3 ^0783$              djnz   :10
07D0  C9                       ret
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;**
                        ;       sbrt:   CHKEY - checks if key number is on.
                        ;       input:
                        ;Reg A  =       Keynumber
                        ;       output:
                        ;Z ind clr      =       Key is off.
                        ;Z ind set      =       Key is on.

07D1                    CHKEY:  proc
07D1    E5                      push    hl              ;save callers hl
07D2    F5                      push    af              ;save keynumber
07D3    1F                      rar
07D4    1F                      rar
07D5    1F                      rar                     ;right justify row number
07D6    CDE507                  call    gtmask
07D9    F1                      pop     af              ;get key number
07DA    D5                      push    de              ;save row mask
07DB    CDE507                  call    gtmask          ;get col mask (col num is in bits 0..2)
07DE    E1                      pop     hl              ;move row mask to l
07DF    CDEF07                  call    rdrow           ;get row of keys adrsed by l
07E2    A3                      and     e               ;z ind  =       value of key
07E3    E1                      pop     hl
07E4    C9                      ret
```

```
                        ;**
                        ;       sbrt:   GTMASK - generates mask with one bit set.
                        ;       input:
                        ;a      =       bit number (0..7)
                        ;       output:
                        ;e      =       mask

07E5                    GTMASK: proc
07E5  1E01                      LDK     e,1
07E7  E607                      and     7
07E9                    :10:
07E9  C8                        rz
07EA  CB23        $             sla     e
07EC  3D                        dec     a
07ED  18FA ^07E9$               jr      :10
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        ;**
                        ;       sbrt:   RDROW - Reads a row of keys
                        ;       input:
                        ;reg L  =       lower 8 bits of adrs to read the row
                        ;       output:
                        ;reg A  =       row value


07EF                    RDROW:  proc

07EF  2622                      LDK     h,high(h.key)           ;hl = prt adrs for given row
07F1  7D                        MOV     A,L
07F2  ED4F          $           MOV     R,A
07F4  7E                        ld      a,[hl]
07F5  EEFF                      xor     0ffh                    ;invert values
07F7  C9                        ret
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;**
                        ;       sbrt:   KBSERV - services the key
                        ;       input:
                        ;           reg A       =       keynumber
                        ;       [SP]-4  =       pointer to keylst entry (used for slide fnc only)
                        ;       preserves reg B

        = 008D          lft_arw equ     8dh
        = 008B          rt_arw  equ     8bh
        = 008A          up_arw  equ     8ah
        = 008C          dn_arw  equ     8ch
        = 005B          hm_scrn equ     '['

07F8                    KBSERV:
07F8                            PROC

                        ;       setup hl to point to keycode table entry for this key
07F8    5F                      mov     e,a
07F9    1600                    LDK     d,0             ;used here and later
07FB    21C308                  LDK     hl,kycdtb
07FE    19                      add     hl,de
07FF    7E                      ld      a,[hl]
0800    FE21                    cmp     ' '+1
0802    381F ^0823$             jrc     :12             ;ignore shift/ctl etc for chars less than 21h

0804    F5                      push    af
0805    2E01                    LDK     l,1             ;row 0 adrs
0807    CDEF07                  call    rdrow           ;get row containing ctl,shift and alpha key

080A    F5                      PUSH    AF
080B    2E80                    LDK     L,80H
080D    CDEF07                  CALL    RDROW
0810    E608                    ANI     8
0812    5F                      MOV     E,A
0813    F1                      POP     AF
0814    B3                      OR      E

0815    5F                      mov     e,a
0816    F1                      pop     af              ;restore keycode
0817    CB53      $             bit     ctl_ky,e
0819    202E ^0849$             jrnz    :50             ;go process ctl key
081B    CB63      $             bit     shft_ky,e
081D    2014 ^0833$             jrnz    :30             ;go process shift key
081F    CB5B      $             bit     alph_ky,e
0821    2004 ^0827$             jrnz    :25             ;go process alpha key

                        ;       store key code into lkey
0823                    :12:
0823    325EEF                  sto     a,lkey
0826    C9                      ret
0827                    :25:                            ;process alpha key
0827    FE61                    cmp     'a'
0829    38F8 ^0823$             jrc     :12             ;exit when less than 'a'.  Alpha has no effect
082B                    :27:
082B    FE80                    cmp     80h
082D    30F4 ^0823$             jrnc    :12             ;or when >= 80h
082F                    :28:
082F    EE20                    xor     20h             ;fold char to upper case
0831    18F0 ^0823$             jr      :12
```

Copyright © 1982 Osborne Computer Corporation


```
0833                 :30:                                   ;process shift key
0833  FE61                     cmp      'a'
0835  30F4 ^0828$              jrnc     :27                 ;goto alpha when char > 'a'
0837  FE5B                     cmp      '['
0839  3806 ^0841$              jrc      :35                 ;goto process shift numerics etc
083B  20F2 ^082F$              jrnz     :28                 ;invert shift bit for '\'
083D  3E5D                     LDK      a,']'
083F  18E2 ^0823$              jr       :12                 ;convert [ to ]
0841                 :35:
                     ;         chars ' to > (ascii codes 27h to 3eh) are converted using
                     ;         the shft_tb

0841  5F                       mov      e,a
                                                            ;d=0 from before
0842  21D408                   LDK      hl,shft_tb - ''''
0845  19                       add      hl,de
0846                 :37:
0846  7E                       ld       a,[hl]
0847  18DA ^0823$              jr       :12


0849                 :50:                                   ;process control key
                     ;         if char is between a..z then turn off the 3 high order
                     ;         bits.
                     ;         if char is between ','..'?' it is translated as per table ctl_tb'
                     ;         if char is the arrow keys or the ']'/'[' key the slide functions
                     ;         are called.

0849  FE8D                     cmp      lft_arw
084B  2833 ^0880$              jrz      slidel
084D  FE8B                     cmp      rt_arw
084F  2833 ^0884$              jrz      slider
0851  FE8A                     cmp      up_arw
0853  2840 ^0895$              jrz      slideu
0855  FE8C                     cmp      dn_arw
0857  2840 ^0899$              jrz      slided
0859  FE5B                     cmp      hm_scrn
085B  2850 ^08AD$              jrz      dohome

085D  CB63        $            BIT      shft_ky,e           ;test for cntl shift
085F  2808 ^0869$              jrz      :52                 ;if not
0861  FE2F                     cmp      '/'                 ;is it ?
0863  2004 ^0869$              jrnz     :52                 ;if not '?'
0865  3E7F                     ldk      a,07fh              ;deleat key
0867  18BA ^0823$              jr       :12

0869  FE40           :52:      cmp      'a'
086B  3808 ^0875$              jrc      :54                 ;goto translate chars from table
086D  FE7B                     cmp      'z'+1
086F  30B2 ^0823$              jrnc     :12
0871  E61F                     and      1fh
0873  18AE ^0823$              jr       :12

0875                 :54:
0875  FE2C                     cmp      ','
0877  38AA ^0823$              jrc      :12                 ;no translation if char below ','
0879  21E708                   LDK      hl,ctl_tb-','
087C  5F                       mov      e,a
                                                            ;d=0 from above
087D  19                       add      hl,de
```

Copyright © 1982 Osborne Computer Corporation

```
087E  18C6 ^0846$           jr       :37

                     ;       slide functions.
0880                 slidel:
0880  0E02                  LDK      c,2
0882  1802 ^0886$           jr       :70
0884                 slider:
0884  0EFE                  LDK      c,-2

0886  3A61EF         :70:   ld       a,piaad          ;get horizontal coord.
0889  81                    add      a,c
088A  4F                    MOV      C,A

088B  CD6409         :72:   CALL     OPAD             ;FUNCTION PIA

088E                 :74:
                     ;       set repeat count for these keys (override count set by the kbdrvr)
088E  D1                    pop      de               ;get return adrs
088F  E1                    pop      hl               ;pointer to repeat entry
0890  3603                  sto      sld_rct,[hl]     ;repeat count for slide keys
0892  E5                    push     hl
0893  D5                    push     de               ;restore stack
0894  C9                    ret

0895                 slideu:
0895  0E01                  LDK      c,1
0897  1802 ^089B$           jr       :76
0899                 slided:
0899  0EFF                  LDK      c,-1
089B                 :76:                             ;merge new vertoffset to lower 5 bits of
                                                      ;PIAB
089B  2162EF                LDK      hl,piabd
089E  7E                    ld       a,[hl]
089F  81                    add      a,c
08A0  E61F                  and      1fh              ;modify current with +1/or-1
08A2  4F                    mov      c,a
08A3  7E                    ld       a,[hl]
08A4  E6E0                  and      0e0h
08A6                 :78:
08A6  B1                    or       c
08A7  4F                    mov      c,a
08A8  CD7109                call     opbd
08AB  18E1 ^088E$           jr       :74

08AD                 dohome:

                            ;SET DENSITY BIT

08AD  3A61EF                LD       A,PIAAD          ;GET OLD VALUE
08B0  E601                  ANI      0000_0001B       ;SAVE DENSITY BIT
08B2  F6EA                  ORI      VFLO             ;OR IN HORIZONTAL OFFSET
08B4  4F                    MOV      C,A
08B5  CD6409                CALL     OPAD             ;FUNCTION PIA

08B8  3A62EF                ld       A,piabd
08BB  E6E0                  AND      0E0H             ;HOUSE KEEPING BITS
08BD  4F                    MOV      C,A
08BE  3AEFEF                LD       A,VRTOFF         ;GET LAST VERTICAL OFFSET
08C1  18E3 ^08A6$           jr       :78              ;and the vert to 0 also
```

Copyright © 1982 Osborne Computer Corporation


```
                        ;       key code translation tables
08C3                    kycdtb:
08C3  1B097F7F                  DB      esc, tab, erc, erc

08C7  7F0D275B                  DB      erc, cr, '''', '['

08CB  31323334                  DB      '1', '2', '3', '4'

08CF  35363738                  DB      '5', '6', '7', '8'

08D3  71776572                  DB      'q', 'w', 'e', 'r'

08D7  74797569                  DB      't', 'y', 'u', 'i'

08DB  61736466                  DB      'a', 's', 'd', 'f'

08DF  67686A6B                  DB      'g', 'h', 'j', 'k'

08E3  7A786376                  DB      'z', 'x', 'c', 'v'

08E7  626E6D2C                  DB      'b', 'n', 'm', ','

08EB  8A8D3020                  DB      8ah, 8dh, '0', ' '

08EF  2E706F39                  DB      '.', 'p', 'o', '9'

08F3  8B8C2D2F                  DB      8bh, 8ch, '-', '/'

08F7  3B5C6C3D                  DB      ';', '\', '|', '='


08FB                    shft_tb:
08FB  2200000000                DB      '"', 00h, 00h, 00h, 00h

0900  3C5F3E3F29                DB      '<', '_', '>', '?', ')'

0905  2140232425                DB      '!', '@', '#', '$', '%'

090A  5E262A2800                DB      '^', '&', '*', '(', 00h

090F  3A002B00                  DB      ':', 00h, '+', 00h

0913                    ctl_tb:
0913  7B1F7D7E                  DB      '{', '_'-40h, '}', '~'

0917  8081828384                DB      80h, 81h, 82h, 83h, 84h

091C  8586878889                DB      85h, 86h, 87h, 88h, 89h

0921  00000060                  DB      00h, 00h, 00H, 60H
```

Copyright © 1982 Osborne Computer Corporation

```
                     *[4]
0925                 CDEHL:
                     ;Compare DE to HL as unsigned integers.
                     ;ENTRY
                     ;DE,HL set to values to compare

                     ;EXIT
                     ;Zbit set if DE =       HL
                     ;Cbit set if DE < HL.
                     ;       Uses    AF.
0925  7A                     MOV     A,D
0926  BC                     CMP     H
0927  C0                     RNZ             ;If D .ne. H
0928  7B                     MOV     A,E
0929  BD                     CMP     L
092A  C0                     RNZ             ;if DE .ne. HL
092B  37                     STC
092C  C9                     RET
```

Copyright © 1982 Osborne Computer Corporation

```
092D                    DELAY:
                        ;"N" Milliseconds
                        ;ENTRY
                        ;A        =     Number of Milliseconds to delay
                        ;SCLFRE =       (Freq/1000)/25

                        ;EXIT
                        ;NONE
092D                            Proc
092D  C5                        PUSH    BC

092E  4F                        MOV     C,A
092F  3E85              :1:     LDK     A,SCLFRE

0931  3D                :MLOOP: DEC     A               ;(4 tics)
0932  40                        MOV     B,B             ;(4 tics)
0933  49                        MOV     C,C             ;(4 tics)
0934  C23109                    JNZ     :MLOOP          ;(10 tics) If 1 ms not elapsed

0937  0D                        DEC     C
0938  C22F09                    JNZ     :1              ;If requested msec not done

093B  C1                        POP     BC              ;restore registers
093C  C9                        RET

                        ;SCLFRE =       2000/22 ;Z80, 2mhz
                        ;...defined in OCCTXT.ast
```

Copyright © 1982 Osborne Computer Corporation


```
093D                 FILLZ:
                     ;Fill block of memory with byte value.
                     ;ENTRY
                     ;DE=LENGTH TO broadcast character
                     ;C =character TO broadcast
                     ;HL=FWA TO START broadcast

                     ;EXIT
                     ;FILL DONE
093D  0E00                   LK      C,0      ;FILL WITH ZERO
093F  7B             FILLC:  MOV     A,E
0940  B2                     OR      D
0941  C8                     RZ               ; return here if broadcast 0 bytes
0942  71                     STO     C,[hl]   ; 1st byte
0943  1B                     DEC     DE
0944  7B                     MOV     A,E
0945  B2                     OR      D
0946  C8                     RZ               ; return here if broadcast 1 byte
0947  42                     MOV     B,D
0948  4B                     MOV     C,E      ; BC := (count)
0949  54                     MOV     D,H
094A  5D                     MOV     E,L
094B  13                     INC     DE       ; DE := HL + 1
094C  EDB0       $           LDIR             ; overlapping move
094E  C9                     RET
```

Copyright © 1982 Osborne Computer Corporation

```
094F                SPAO:
                    ;Set PIA for output
094F  3E03                  LDK     A,03h           ;CRA1-0, Interrupt inputs CA1
0951  32012C                STO     A,H.VIO+1       ;set data direction
0954  3EFF                  LDK     A,0FFh
0956  32002C                STO     A,H.VIO         ;set all A lines as output
0959  3E00                  LDK     A,0
095B  32032C                STO     A,H.VIO+3
095E  3EFF                  LDK     A,0FFh
0960  32022C                STO     A,H.VIO+2       ;set all B lines as output
0963  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0964                    OPAD:
                        ;Output data to pia A register
                        ;       PIA definition.
                        ;         7  6  5  4  3  2  1  0
                        ;       +--+--+--+--+--+--+--+--+
                        ;       !  horizontal offset !DD!
                        ;       +--+--+--+--+--+--+--+--+
                        *NOTE   The DD(double density) bit is inverted and the jumper must be installed on the pc board.
                        ;       If the 0 bit is LOW double density is set if it is HIGH single density is set.
                        *NOTE   Bit 0 of "PIAAD" :
                        ;       set     =       single density
                        ;       reset   =       double density

                        ;ENTRY
                        ;C      =       data

                        ;EXIT
                        ;NONE
0964                            PROC
0964  3E07                      LDK     A,4+3
0966  32012C                    STO     A,H.VIO+1

0969  79                        MOV     A,C
096A  3261EF                    STO     A,PIAAD
096D  32002C                    STO     A,H.VIO         ;send data
0970  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0971                    OPBD:
                        ;Output data to pia B register
                        ;       PIA definition.
                        ;         7  6  5  4  3  2  1  0
                        ;       +--+--+--+--+--+--+--+--+
                        ;       !D1!D0!^G! vert  offset !
                        ;       +--+--+--+--+--+--+--+--+

                        ;ENTRY
                        ;C      =       data

                        ;EXIT
                        ;NONE
0971                            PROC
0971  3E04                      LDK     A,4
0973  32032C                    STO     A,H.VIO+3

0976  79                        MOV     A,C
0977  3262EF                    STO     A,PIABD
097A  32022C                    STO     A,H.VIO+2       ;send data
097D  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
;       +----------------------------------------+
;       ! ENTERED 05/01/81 FROM TNW XEROX, SEH.  !
;       +----------------------------------------+


;LAST EDITED AT 09:29 ON 11 NOV 80

;THERE ARE FOUR COMMANDS TO THE 6821

;       00      PERIPHERAL/DIRECTION REGISTER A         CPDRA
;       01      CONTROL REGISTER A                      CCRA
;       10      PERIPHERAL/DIRECTION REGISTER B         CPDRB
;       11      CONTROL REGISTER B                      CCRB

;BIT 2 OF THE CONTROL REGISTER (A AND B) ALLOWS SELECTION OF EITHER
;A PERIPHERAL INTERFACE REGISTER OR A DATA DIRECTION REGISTER.
;A "1" IN BIT 2 SELECTS THE PERIPHERAL REGISTER.

;THE TWO DATA DIRECTION REGISTERS ALLOW CONTROL OF THE DIRECTION
;OF DATA THROUGH EACH CORRESPONDING PERIPHERAL DATA LINE.
;A DATA DIRECTION REGISTER BIT SET AT "0" CONFIGURES
;THE CORRESPONDING PERIPHERAL DATA LINE AS AN INPUT.

;A RESET AT POWER UP HAS THE EFFECT OF ZEROING ALL PIA REGISTERS.
;THIS WILL SET PA0-PA7, PB0-PB7, CA2, AND CB2 AS INPUTS,
;AND ALL INTERRUPTS DISABLED.
;SIGNALS ATN, REN, AND IFC WILL BE DRIVEN LOW
;UNTIL INITIALIZED BY SOFTWARE.

;DATA DIRECTION IS ALWAYS SET FOR OUTPUT FOR THE DATA REGISTER.
;DATA MUST BE SET TO ALL ONES WHEN INPUTTING.
;THE INTERFACE IS IN SOURCE HANDSHAKE MODE IF DATA ENABLE (PB0)
;IS SET TO "0", AND IN ACCEPTOR HANDSHAKE MODE IF SET TO "1".
;WHEN SWITCHING FROM SOURCE TO ACCEPTOR HANDSHAKE,
;ATN WILL ALWAYS BE LOW.
;TAKE CONTROL CAN ONLY BE CALLED FOLLOWING A GO TO STANDBY.
;AFTER A FATAL ERROR, PERFORM AN IFC RESET.

;STANDARD VALUES USED:

;CCRA   0011(IFC)(DIR)10

;CCRB   0011(REN)(DIR)00

;CPDRA  SOURCE          DIRECTION       1111_1111
;                       DATA            DATA

;CPDRA  ACCEPTOR        DIRECTION       1111_1111
;                       DATA            1111_1111

;CPDRB  SOURCE          DIRECTION       0011_1111
;                       DATA            000A_0010       ;A = ATN

;CPDRB  ACCEPTOR        DIRECTION       1101_0111
;                       DATA            0100_0101
```

```
;PIA SIGNAL DEFINITIONS:
;ALL SIGNALS ARE LOW ON THE IEEE BUS WHEN PIA REGISTER CONTAINS "1".

;       PA0     DIO 1
;       PA1     DIO 2
;       PA2     DIO 3
;       PA3     DIO 4
;       PA4     DIO 5
;       PA5     DIO 6
;       PA6     DIO 7
;       PA7     DIO 8

;       CA1     SRQ
;       CA2     IFC

;       PB0     ENABLE DATA OUT         (ENABLED WHEN "0")
;       PB1     ENABLE NDAC/NRFD        (ENABLED WHEN "0")
;       PB2     ENABLE EOI/DAV          (ENABLED WHEN "0")
;       PB3     EOI
;       PB4     ATN
;       PB5     DAV
;       PB6     NDAC
;       PB7     NRFD

;       CB1     NOT USED
;       CB2     REN

;CONTROL WORD FORMAT

;[  7  ][  6  ][  5  ][  4  ][  3  ][  2  ][  1  ][  0  ]

;[IRQA1][IRQA2][    CA2 CONTROL    ][ DDRA][ CA1 CONTROL]
;[IRQB1][IRQB2][    CB2 CONTROL    ][ DDRB][ CB1 CONTROL]

;       IRQA1   0       INTERRUPT FLAG SET BY FALL OF SRQ
;       IRQA2   0       NOT USED
;       CA2     110     SET IFC HIGH
;               111     SET IFC LOW
;       DDRA    0       R/W DATA DIRECTION REGISTER A
;               1       R/W PERIPHERAL REGISTER A
;       CA1     10      SET IRQA1 HIGH ON RISE OF SRQ

;       IRQB1   0       NOT USED
;       IRQB2   0       NOT USED
;       CB2     110     SET REN HIGH
;               111     SET REN LOW
;       DDRB    0       R/W DATA DIRECTION REGISTER B
;               1       R/W PERIPHERAL REGISTER B
;       CB1     00      NOT USED
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
                        ;BIOS CALL 1:   CONTROL OUT

                        ;       CAN BE CALLED WHILE IN ANY STATE.

                        ;       EXITS IN THE CONTROLLER STANDBY STATE (ATN HIGH),
                        ;       SOURCE HANDSHAKE MODE

                        ;PARAMETER PASSED IN REGISTER C:

                        ;       BIT 0   IF "1", THE IFC SIGNAL IS SET LOW FOR 100 MICRO-SEC
                        ;               AND ALL PIA SIGNALS ARE INITIALIZED

                        ;       BIT 2 1
                        ;           0 X NO ACTION
                        ;           1 0 SETS REN HIGH
                        ;           1 1 SETS REN LOW

097E                    IE.CO:  PROC
097E  F5                        PUSH    AF
097F  E5                        PUSH    HL
0980  CB41       $              BIT     0,C             ;CHECK IFC SUB-COMMAND
0982  282B ^09AF$               JRZ     :B1C20

                        ;INITIALIZE ALL IEEE-488 SIGNALS
0984  210129                    LK      HL,CCRA
0987  363A                      STO     0011_1010B,[HL] ;ENABLE SRQ AND SET IFC-OUT LOW
0989  3EFF                      LK      A,1111_1111B    ;DIRECT DATA OUT
098B  320029                    STO     A,CPDRA
098E  363E                      STO     0011_1110B,[HL]
0990  AF                        XOR     A               ;SET DATA TO ZERO
0991  320029                    STO     A,CPDRA
0994  210329                    LK      HL,CCRB
0997  3630                      STO     0011_0000B,[HL] ;SET REN-OUT HIGH
0999  3E3F                      LK      A,0011_1111B    ;DIRECTION FOR SOURCE HANDSHAKE
099B  320229                    STO     A,CPDRB
099E  3634                      STO     0011_0100B,[HL]
09A0  3E02                      LK      A,0000_0010B    ;VALUES FOR SOURCE HANDSHAKE
09A2  320229                    STO     A,CPDRB

                        ;LEAVE IFC LOW FOR 100 MICRO-SEC
09A5  3E19                      LK      A,25            ;DELAY 100 MICRO-SEC
09A7  3D                :B1C10: DEC     A
09A8  20FD ^09A7$               JRNZ    :B1C10
09AA  3E36                      LK      A,0011_0110B    ;SET IFC HIGH
09AC  320129                    STO     A,CCRA
09AF  CB51       $ :B1C20:      BIT     2,C             ;CHECK REN SUB-COMMAND
09B1  280B ^09BE$               JRZ     :B1C40

                        ;SET/CLEAR REN
09B3  3E30                      LK      A,0011_0000B
09B5  CB49       $              BIT     1,C
09B7  2802 ^09BB$               JRZ     :B1C30
09B9  3E38                      LK      A,0011_1000B
09BB  320329            :B1C30: STO     A,CCRB
09BE  E1                :B1C40: POP     HL
09BF  F1                        POP     AF
09C0  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation


```
                        ;BIOS CALL 2:   STATUS IN

                        ;       CAN BE CALLED ONLY WHILE IN SOURCE HANDSHAKE MODE.

                        ;BIT 0 OF REGISTER A SET IF SRQ IS LOW

09C1                    IE.SI:  PROC
09C1  E5                        PUSH    HL
09C2  3A0029                    LD      A,CPDRA         ;CLEAR IRQA1
09C5  210229                    LK      HL,CPDRB        ;PULSE ENABLE ndac/nrfd
09C8  CB8E        $             CBIT    1,[HL]
09CA  CBCE        $             SBIT    1,[HL]
09CC  3A0129                    LD      A,CCRA          ;SET SRQ VALUE IN A
09CF  E680                      AND     1000_0000B
09D1  07                        RLC     A
09D2  E1                        POP     HL
09D3  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
                    ;BIOS CALL 3: GO TO STANDBY

                    ;       CAN BE CALLED ONLY WHILE IN SOURCE HANDSHAKE MODE

                    ;NO PARAMETERS PASSED

09D4                IE.GTS: PROC
09D4  F5                    PUSH    AF
09D5  3E02                  LK      A,0000_0010B    ;SET ATN HIGH
09D7  320229                STO     A,CPDRB
09DA  AF                    XOR     A               ;FLOAT DATA BUS
09DB  320029                STO     A,CPDRA
09DE  F1                    POP     AF
09DF  C9                    RET
```

```
                    ;BIOS CALL 4:   TAKE CONTROL

                    ;       CAN BE CALLED ONLY WHILE IN THE CONTROLLER STANDBY STATE
                    ;               (ATN HIGH).

                    ;       EXITS IN THE CONTROLLER ACTIVE STATE (ATN LOW),
                    ;       SOURCE HANDSHAKE MODE.

                    ;BIT 0 OF REGISTER C SET TO TAKE CONTROL ASYNCHRONOUS

                    ;ERROR CODE RETURNED IN REGISTER A.

09E0                IE.TC:  PROC
09E0  E5                    PUSH    HL
09E1  210229                LK      HL,CPDRB
09E4  CB41        $         BIT     0,C
09E6  2022 ^0A0A$           JRNZ    :B4C30

                    ;TAKE CONTROL SYNCHRONOUSLY
09E8  3607                  STO     0000_0111B,[HL] ;DISABLE DRIVERS
09EA  3A0329                LD      A,CCRB
09ED  CB97        $         CBIT    2,A
09EF  320329                STO     A,CCRB
09F2  36D7                  STO     1101_0111B,[HL] ;DIRECTION REGISTER
09F4  CBD7        $         SBIT    2,A
09F6  320329                STO     A,CCRB
09F9  3685                  STO     1000_0101B,[HL] ;SET NRFD LOW
09FB  3E19                  LK      A,25
09FD  CB6E        $ :B4C10: BIT     5,[HL]
09FF  2807 ^0A08$           JRZ     :B4C20          ;DATA VALID HAS DROPPED
0A01  3D                    DEC     A
0A02  20F9 ^09FD$           JRNZ    :B4C10          ;WAIT 100 MICRO-SEC
0A04  3E81                  LK      A,1000_0001B    ;SET DATA VALID TIMEOUT ERROR
0A06  1816 ^0A1E$           JR      :B4C40

0A08  36C5          :B4C20: STO     1100_0101B,[HL] ;SET NDAC LOW
0A0A  CBE6        $ :B4C30: SBIT    4,[HL]          ;SET ATN LOW

                    ;SET-UP FOR SOURCE HANDSHAKE
0A0C  3A0329                LD      A,CCRB
0A0F  CB97        $         CBIT    2,A
0A11  320329                STO     A,CCRB
0A14  363F                  STO     0011_1111B,[HL] ;DIRECTION REGISTER
0A16  CBD7        $         SBIT    2,A
0A18  320329                STO     A,CCRB
0A1B  3612                  STO     0001_0010B,[HL] ;CONTROL SIGNAL INITIAL VALUE
0A1D  AF                    XOR     A               ;CLEAR ERROR CODE
0A1E  E1            :B4C40: POP     HL
0A1F  C9                    RET
```

Copyright ©1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
                        ;BIOS CALL 5:   OUTPUT INTERFACE MESSAGE

                        ;       CAN BE CALLED WHILE IN ANY MODE OR STATE

                        ;       EXITS IN THE SOURCE HANDSHAKE MODE WITH ATN LOW.

                        ;MULTI-LINE MESSAGE IN REGISTER C

                        ;ERROR CODE RETURNED IN REGISTER A

0A20                    IE.OIM: PROC
0A20  E5                        PUSH    HL
0A21  210229                    LK      HL,CPDRB
0A24  CBE6        $             SBIT    4,[HL]          ;SET ATN LOW
0A26  CB46        $             BIT     0,[HL]
0A28  2825 ^0A4F$               JRZ     IE.SHK

                        ;SET-UP FOR SOURCE HANDSHAKE
0A2A  3617                      STO     0001_0111B,[HL] ;DISABLE DRIVERS
0A2C  3A0329                    LD      A,CCRB
0A2F  CB97        $             CBIT    2,A
0A31  320329                    STO     A,CCRB
0A34  363F                      STO     0011_1111B,[HL] ;DIRECTION REGISTER
0A36  CBD7        $             SBIT    2,A
0A38  320329                    STO     A,CCRB
0A3B  AF                        XOR     A               ;FLOAT EXTERNAL DATA BUS
0A3C  320029                    STO     A,CPDRA
0A3F  3612                      STO     0001_0010B,[HL] ;CONTROL SIGNAL INITIAL VALUE
0A41  180C ^0A4F$               JR      IE.SHK
```

Copyright © 1982 Osborne Computer Corporation

```
                     ;BIOS CALL 6:   OUTPUT DEVICE MESSAGE

                     ;        CAN BE CALLED ONLY WHILE IN THE SOURCE HANDSHAKE MODE
                     ;        WITH ATN HIGH OR LOW.

                     ;        EXITS IN THE SOURCE HANDSHAKE MODE WITH ATN HIGH.

                     ;MULTI-LINE MESSAGE IN REGISTER C
                     ;EOI REQUEST IN REGISTER B

                     ;ERROR CODE RETURNED IN REGISTER A

0A43                 IE.ODM: PROC
0A43  E5                     PUSH    HL
0A44  210229                 LK      HL,CPDRB
0A47  CBA6       $           CBIT    4,[HL]          ;SET ATN HIGH
0A49  CB40       $           BIT     0,B             ;CHECK IF EOI REQUESTED
0A4B  2802 ^0A4F$            JRZ     IE.SHK
0A4D  CBDE       $           SBIT    3,[HL]
                     ;PERFORM SOURCE HANDSHAKE
0A4F  CB6E       $ IE.SHK: BIT     5,[HL]          ;
0A51  201B ^0A6E$            JRNZ    :B6C50          ;DAC TIMEOUT RE-ENTRY
0A53  79                     MOV     A,C             ;PLACE DATA ON BUS
0A54  320029                 STO     A,CPDRA
0A57  3E0A                   LK      A,10
0A59  CB7E       $ :B6C20: BIT     7,[HL]
0A5B  2807 ^0A64$            JRZ     :B6C30          ;READY FOR DATA
0A5D  3D                     DEC     A
0A5E  20F9 ^0A59$            JRNZ    :B6C20          ;WAIT FOR 100 MICRO-SEC
0A60  3E82                   LK      A,1000_0010B    ;SET RFD TIMEOUT ERROR
0A62  181F ^0A83$            JR      :B6C80

0A64  CB76       $ :B6C30: BIT     6,[HL]
0A66  2004 ^0A6C$            JRNZ    :B6C40          ;DATA ACCEPTED LOW
0A68  3E81                   LK      A,1000_0001B    ;SET DEVICE NOT PRESENT ERROR
0A6A  1817 ^0A83$            JR      :B6C80

0A6C  CBEE       $ :B6C40: SBIT    5,[HL]          ;SET DAV LOW
0A6E  3EFF         :B6C50: LK      A,255
0A70  CB76       $ :B6C60: BIT     6,[HL]
0A72  2807 ^0A7B$            JRZ     :B6C70          ;DATA ACCEPTED
0A74  3D                     DEC     A
0A75  20F9 ^0A70$            JRNZ    :B6C60          ;WAIT 1000 MICRO-SEC
0A77  3E84                   LK      A,1000_0100B    ;SET DAC TIMEOUT ERROR
0A79  1808 ^0A83$            JR      :B6C80

0A7B  CBAE       $ :B6C70: CBIT    5,[HL]          ;SET DAV HIGH
0A7D  CB9E       $           CBIT    3,[HL]          ;SET EOI HIGH
0A7F  AF                     XOR     A               ;REMOVE DATA FROM BUS
0A80  320029                 STO     A,CPDRA
0A83  E1           :B6C80: POP     HL
0A84  C9                     RET
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;BIOS CALL 7:   INPUT DEVICE MESSAGE

                        ;       CAN BE CALLED WHILE IN ANY MODE OR STATE

                        ;       EXITS IN THE ACCEPTOR HANDSHAKE MODE WITH ATN HIGH.

                        ;DEVICE MESSAGE RETURNED IN BOTH REGISTERS A AND H
                        ;ERROR CODE RETURNED IN REGISTER L

0A85                    IE.IDM: PROC
0A85  D5                        PUSH    DE
0A86  EB                        EX      DE,HL           ;SAVE RE-ENTRY DATA
0A87  210229                    LK      HL,CPDRB
0A8A  CB46      $               BIT     0,[HL]
0A8C  201A ^0AA8$               JRNZ    :B7C10

                        ;SET-UP FOR ACCEPTOR HANDSHAKE
0A8E  3617                      STO     0001_0111B,[HL] ;DISABLE DRIVERS
0A90  3A0329                    LD      A,CCRB
0A93  CB97      $               CBIT    2,A
0A95  320329                    STO     A,CCRB
0A98  36D7                      STO     1101_0111B,[HL] ;DIRECTION REGISTER
0A9A  CBD7      $               SBIT    2,A
0A9C  320329                    STO     A,CCRB
0A9F  3EFF                      LK      A,1111_1111B    ;FLOAT INTERNAL DATA BUS
0AA1  320029                    STO     A,CPDRA
0AA4  3655                      STO     0101_0101B,[HL] ;CONTROL SIGNALS INITIAL VALUE
0AA6  3645                      STO     0100_0101B,[HL] ;SET ATN HIGH

                        ;PERFORM ACCEPTOR HANDSHAKE
0AA8  CB76      $ :B7C10: BIT   6,[HL]
0AAA  2820 ^0ACC$               JRZ     :B7C50          ;DATA INVALID TIMEOUT ERROR RE-ENTRY
0AAC  CBBE      $               CBIT    7,[HL]          ;SET NRFD HIGH
0AAE  3E0A                      LK      A,10
0AB0  CB6E      $ :B7C20: BIT   5,[HL]
0AB2  2008 ^0ABC$               JRNZ    :B7C30          ;DATA VALID
0AB4  3D                        DEC     A
0AB5  20F9 ^0AB0$               JRNZ    :B7C20          ;WAIT 100 MICRO-SEC
0AB7  118200                    LK      DE,1000_0010B   ;SET DATA VALID TIMEOUT ERROR
0ABA  1821 ^0ADD$               JR      :B7C80

0ABC  CBFE      $ :B7C30: SBIT  7,[HL]          ;SET NRFD LOW
0ABE  3A0029                    LD      A,CPDRA         ;READ DATA
0AC1  57                        MOV     D,A
0AC2  1E00                      LK      E,0             ;READ EOI
0AC4  CB5E      $               BIT     3,[HL]
0AC6  2802 ^0ACA$               JRZ     :B7C40
0AC8  1E01                      LK      E,1
0ACA  CB86      $ :B7C40: CBIT  6,[HL]          ;SET NDAC HIGH
0ACC  3EFF        :B7C50: LK    A,255
0ACE  CB6E      $ :B7C60: BIT   5,[HL]
0AD0  2809 ^0ADB$               JRZ     :B7C70          ;DATA VALID DROPPED
0AD2  3D                        DEC     A
0AD3  20F9 ^0ACE$               JRNZ    :B7C60          ;WAIT 1000 MICRO-SEC
0AD5  CBD3      $               SBIT    2,E             ;SET DATA INVALID TIMEOUT ERROR
0AD7  CBFB      $               SBIT    7,E
0AD9  1802 ^0ADD$               JR      :B7C80

0ADB  CBF6      $ :B7C70: SBIT  6,[HL]          ;SET NDAC LOW
0ADD  EB          :B7C80: EX    DE,HL           ;MOVE RESULTS TO REGISTERS A AND HL
```

```
0ADE  7C          MOV   A,H
0ADF  D1          POP   DE
0AE0  C9          RET
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;BIOS CALL 8:   PARALLEL POLL

                        ;       CAN BE CALLED ONLY WHILE IN THE SOURCE HANDSHAKE MODE
                        ;       WITH ATN HIGH OR LOW.

                        ;       EXITS IN THE SOURCE HANDSHAKE MODE WITH ATN LOW.

                        ;PARALLEL POLL VALUE RETURNED IN A.

0AE1                    IE.PP:  PROC
0AE1  E5                        PUSH    HL
0AE2  210029                    LK      HL,CPDRA
0AE5  3E1B                      LK      A,0001_1011B    ;FORM PARALLEL POLL
0AE7  320229                    STO     A,CPDRB
0AEA  36FF                      STO     1111_1111B,[HL] ;FLOAT INTERNAL DATA BUS
0AEC  7E                        LD      A,[HL]          ;READ PARALLEL POLL DATA
0AED  3600                      STO     0,[HL]          ;RE-STORE SOURCE HANDSHAKE MODE
0AEF  210229                    LK      HL,CPDRB
0AF2  3612                      STO     0001_0010B,[HL]
0AF4  E1                        POP     HL
0AF5  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                     *[5]
0AF6                 SIRST:
                     ;Master reset SIO
                     ;ENTRY
                     ;C       =       SI.S16 or SI.S64 for 1200/300 baud

                     ;EXIT
                     ;NONE
0AF6                         PROC
0AF6  3E57                   LDK     A,SI.MRST
0AF8  32002A                 STO     A,H.SCTRL       ;master reset

0AFB  79                     MOV     A,C
0AFC  32C1EF                 STO     A,ACIAD         ;last-command cell
0AFF  32002A                 STO     A,H.SCTRL       ;select SIO
0B02  C9                     RET
```

Copyright ©1982 Osborne Computer Corporation

```
0803                    READER:
                        ;Input one byte from reader port
                        ;ENTRY
                        ;None

                        ;EXIT
                        ;C       =       character read

0803                            PROC
0803  CD280B                    CALL    ACISTAT
0806  E601                      ANI     SI.RRDY
0808  28F9 ^0803$               JRZ     READER          ;if not ready

080A  AF                        XRA     A
080B  32DAEF                    STO     A,SERFLG        ;SET FLAG

080E  3A012A                    LD      A,H.SREC        ;get data
0811  4F                        MOV     C,A             ;C=A
0812  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0B13                 SLST:
                     ;Get list device status
                     ;ENTRY
                     ;NONE

                     ;EXIT
                     ;A       =       0, IF NOT READY
                     ;A       =       OFFh IF READY
                     ;ZBIT    =       SET IF NOT READY FOR OUTPUT

0B13                         PROC
0B13  CD280B                 CALL    ACISTAT
0B16  E602                   ANI     SI.TRDY
0B18  C8                     RZ

0B19  F6FF                   ORI     TRUE
0B1B  C9                     RET
```

Copyright © 1982 Osborne Computer Corporation

```
0B1C                LIST:
                    ;Output one byte to list port
                    ;ENTRY
                    ;C       =       character to output

                    ;EXIT
                    ;NONE
0B1C                        PROC
0B1C  C5                    PUSH    BC              ;SAVE CHARACTER

0B1D  CD130B        :1:     CALL    SLST            ;GET STATUS
0B20  28FB ^0B1D$           JRZ     :1

0B22  C1                    POP     BC              ;RESTORE CHARACTER
0B23  79                    MOV     A,C
0B24  32012A                STO     A,H.SXMT        ;send chr
0B27  C9                    RET
```

```
0B28                 ACISTAT:
                     ;RETURN STATUS OF THE SERIAL PORT
                     ;ENTRY
                     ;NONE

                     ;EXIT
                     ;A          =       STATUS REG

0B28                            PROC
0B28  3A012C                    LD      A,H.VIO+1
0B2B  0F                        RRC     A
0B2C  E620                      ANI     20h
0B2E  4F                        MOV     C,A
0B2F  3A002A                    LD      A,H.SSTS
0B32  E6DF                      ANI     0DFh
0B34  B1                        ORA     C
0B35  4F                        MOV     C,A
0B36  3ADAEF                    LD      A,SERFLG
0B39  E601                      ANI     01
0B3B  B1                        ORA     C
0B3C  32DAEF                    STO     A,SERFLG
0B3F  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                *[6]

   = 000A       NRETRY: =       10              ;NUMBER OF RETRYS MUST BE AT LEAST 2
   = EFD0       SAVTYP  =       0EFD0H          ;DISK TYPE
   = EFD1       RDT_WRTS:       =       0EFD1H          ;NUMBER OF SECTORS TO READ OR WRITE
   = EFD2       CCPADR: =       0EFD2H          ;This location is assigned in BIOS
                                                ;and filled in by loader in ROM
   = EFD4       KEYLST: =       0EFD4H          ;KEY LIST GOES HERE
```

Copyright © 1982 Osborne Computer Corporation

```
 = 0010       MSECB:   =        010H            ;MULTI SECTOR BIT

 = 0000       D.RES:   =        000H            ;RESTORE
 = 0010       D.SEK:   =        010H            ;SEEK
 = 0020       D.STP:   =        020H            ;STEP
 = 0040       D.STPI:  =        040H            ;STEP IN
 = 0060       D.STPO:  =        060H            ;STEP OUT
 = 0080       D.RDS:   =        080H            ;READ SECTOR
 = 00A0       D.WRTS:  =        0A0H            ;WRITE SECTOR
 = 00C0       D.RDA:   =        0C0H            ;READ ADDRESS
 = 00E0       D.RDT:   =        0E0H            ;READ TRACK
 = 00F0       D.WRTT:  =        0F0H            ;WRITE TRACK
 = 00D0       D.FINT:  =        0D0H            ;FORCE INTERRUPT

              ;DISK REGISTERS

 = 2100       D.CMDR:  =        H.FDC           ;DISK COMMAND REG        (WRITE)
 = 2100       D.STSR:  =        H.FDC           ;STATUS REG              (READ)
 = 2101       D.TRKR:  =        D.CMDR+1        ;TRACK REG
 = 2102       D.SECR:  =        D.CMDR+2        ;SECTOR REG
 = 2103       D.DATR:  =        D.CMDR+3        ;DATA REG                (R/W)

              ;STATUS DEFINITIONS

 = 0000       BS.BSY:  =        0               ;BUSY
 = 0001       DS.BSY:  =        1 SHL BS.BSY

 = 0001       BS.DRQ:  =        1
 = 0002       DS.INX:  =        1 SHL BS.DRQ    ;INDEX MARK DETECTED
 = 0002       DS.DRQ:  =        DS.INX          ;DR IS FULL ON READ, EMPTY ON WRITE

 = 0002       BS.TKO:  =        2               ;TRACK ZERO
 = 0004       DS.TKO:  =        1 SHL BS.TKO
 = 0004       DS.LSD:  =        DS.TKO          ;LOST DATA

 = 0008       DS.CRC:  =        08H             ;CRC ERROR IN ID FIELD

 = 0004       BS.SEK:  =        4
 = 0010       DS.SEK:  =        1 SHL BS.SEK    ;SEEK ERROR
 = 0010       DS.RNF:  =        DS.SEK          ;RECORD NOT FOUND

 = 0020       DS.HDL:  =        20H             ;HEAD LOADED
 = 0020       DS.WTF:  =        DS.HDL          ;WRITE FAULT

 = 0040       DS.WTP:  =        40H             ;WRITE PROTECTED
 = 0080       DS.NRY:  =        80H             ;DRIVE NOT READY

              ;DISK TIMING COUNTS

 = 0014       D.DEL:   =        20              ;DELAY AFTER FUNCTION
```

Copyright © 1982 Osborne Computer Corporation

```
ENAROM  MACRO
        DI
        OUT     0
        LDK     A,0
        STO     A,ROMRAM
        EI
        ENDM

DISROM  MACRO
        DI
        OUT     1
        LDK     A,1
        STO     A,ROMRAM
        EI
        ENDM

PUSHAL  MACRO
        PUSH    BC
        PUSH    DE
        PUSH    HL
        ENDM

POPALL  MACRO
        POP     HL
        POP     DE
        POP     BC
        ENDM
```

```
0B40                    RDRV:
                        ;RESET DRIVE
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;CBIT     =       SET IF ERROR

0B40                              PROC

0B40  3E0A                        LDK     A,NRETRY
0B42  3205EF            :LOOP:    STO     A,RTRY
0B45  CDA40D                      CALL    SELDRV          ;SELECT DRIVE
0B48  3804 ^0B4E$                 JRC     :1
0B4A  CDF70B                      CALL    HOME            ;HOME DRIVE
0B4D  D0                          RNC                     ;IF GOOD

0B4E  3A05EF            :1:       LD      A,RTRY
0B51  3D                          DEC     A
0B52  FE01                        CMP     1
0B54  20EC ^0B42$                 JRNZ    :LOOP

0B56  3E01                        LDK     A,1             ;A = 0FFH FOR CBIOS
0B58  B7                          ORA     A
0B59  37                          STC                     ;INDICATE ERROR
0B5A  C9                          RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0858                RSEC:
                    ;READ SECTOR
                    ;*NOTE*
                    ;       No retries are performed at this level
                    ;ENTRY
                    ;B      =       NUMBER OF SECTORS

                    ;EXIT
                    ;HL     =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                    ;ZBIT   =       RESET IF ERROR
                    ;A      =       NONZERO IF ERROR
                            ;RTRY   =       1 IF ERROR(SO OLD CBIOS DOESN'T DO RETRYS)

0858                        PROC

0858  FD21750C  $           LDK     IY,READ
085F  CD840B                CALL    R_WSEC
0862  3802 ^0866$           JRC     :1              ;IF ERROR

0864  AF                    XRA     A               ;INDICATE GOOD TO CBIOS
0865  C9                    RET

0866  3E01          :1:     LDK     A,1
0868  3205EF                STO     A,RTRY          ;SET RETRY TO 1 FOR OLD CBIOS
086B  B7                    ORA     A               ;RESET ZERO FLAG TO INDICATE ERROR
086C  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0B6D                    WSEC:
                        ;WRITE A SECTOR
                        ;*NOTE*
                        ;       No retries are performed at this level
                        ;ENTRY
                        ;B      =       NUMBER OF SECTORS

                        ;EXIT
                        ;HL     =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                        ;ZBIT   =       RESET IF ERROR
                        ;A      =       NONZERO IF ERROR
                                ;RTRY   =       1 IF ERROR(SO OLD CBIOS DOESN'T DO RETRYS)

0B6D                            PROC

0B6D  FD217D0C  $               LDK     IY,WRITE
0B71  CD840B                    CALL    R_WSEC
0B74  3802 ^0B78$               JRC     :1              ;IF ERROR

0B76  AF                        XRA     A               ;INDICATE GOOD TO CBIOS
0B77  C9                        RET

0B78  3E01              :1:     LDK     A,1
0B7A  3205EF                    STO     A,RTRY          ;SET RETRY TO 1 FOR OLD CBIOS
0B7D  B7                        ORA     A               ;RESET ZERO FLAG TO INDICATE ERROR
0B7E  C9                        RET
0B7F  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0B80                    SENDEN:
                        ;READ THE ADDRESS AND SET CHECK FOR SINGLE DENSITY
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;A      =       NONZERO IF ERROR
                        ;B      =       NUMBER OF SECTORS ON ONE TRACK
                        ;ZBIT   =       RESET IF ERROR
                                ;SAVTYP IS SET WITH DENSITY AND SECTOR SIZE

0B80                            PROC

                        *CHECK DENSITY

                                ;DENSITY LOOP(CHECK PRESENT DENSITY FIRST)

0B80  3E02                      LDK     A,2             ;HOME DRIVE JUST ONCE
0B82  3205EF            :RL1:   STO     A,RTRY

                                ;RETRY LOOP BEFORE AND AFTER HOME

0B85  0605                      LDK     B,NRETRY/2      ;HALF BEFORE HOME AND HALF AFTER

0B87  C5                :RL2:   PUSH    BC              ;SAVE RETRY COUNT
0B88  CDA40D                    CALL    SELDRV          ;SELECT DRIVE
                        ;       JRC     ?               ;NO ERROR CHECKING FOR SELDRV BECAUSE NO ERRORS ARE RETURNED AT THIS TIME
0B8B  CDEC0C                    CALL    RADR            ;READ ADDRESS
0B8E  C1                        POP     BC              ;RESTORE RETRY
0B8F  3004 ^0B95$               JRNC    :1              ;IF GOOD
0B91  10F4 ^0B87$               DJNZ    :RL2            ;RETRY IN THIS DENSITY

0B93  1812 ^0BA7$               JR      :ERR1           ;IF ERROR HOME DRIVE AND TRY AGAIN

                        *SET "SAVTYP"

0B95  3A0CEF            :1:     LD      A,DSTSB+3       ;SECTOR LENTH STATUS BYTE
0B98  E603                      ANI     0000_0011B      ;0-3
0B9A  FE01                      CPI     0000_0001B      ;CHECK FOR 256
0B9C  2009 ^0BA7$               JRNZ    :ERR1

0B9E  3E05                      LDK     A,0000_0101B    ;SINGLE DENSITY 256
0BA0  32D0EF                    STO     A,SAVTYP

0BA3  060A                      LDK     B,10            ;SET FOR SINGLE DENSITY
0BA5  AF                        XRA     A               ;RESET FLAGS
0BA6  C9                        RET                     ;GOOD RETURN

                        *IF DENSITY ERROR CHANGE DENSITY AND RETRY TO :RL1:

0BA7  CDF70B            :ERR1:  CALL    HOME    ;HOME DRIVE
0BAA  3A05EF                    LD      A,RTRY
0BAD  3D                        DEC     A
0BAE  20D2 ^0B82$               JRNZ    :RL1            ;RETRY LOOP

0BB0  3E01                      LDK     A,1
0BB2  B7                        ORA     A               ;FLAGS TO NONZERO
0BB3  C9                        RET                     ;ERROR RETURN
```

Copyright © 1982 Osborne Computer Corporation


```
0BB4                    R_WSEC:
                        ;READ OR WRITE SEGMENT
                        ;ENTRY
                        ;B      =       NUMBER OF SECTORS TO READ OR WRITE
                        ;IY     =       "READ" ADDR OR "WRITE" ADDR

                        ;EXIT
                        ;HL     =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                        ;CBIT   =       SET IF ERROR

0BB4                            PROC

0BB4  3E0A                      LDK     A,NRETRY
0BB6  3205EF                    STO     A,RTRY

0BB9  C5                :RLOOP: PUSH    BC

0BBA  C5                        PUSH    BC
0BBB  FDE5      $               PUSH    IY

0BBD  CDA40D                    CALL    SELDRV          ;TURN DRIVE ON
0BC0  2805 ^0BC7$               JRZ     :1              ;IF DRIVE WAS SELECTED DON'T READ ADDRESS

                        ;SET "D.TRKR" TO HEAD POSITION

0BC2  CD800B                    CALL    SENDEN          ;READ ADDRESS AND SET CONTROLLER
0BC5  2022 ^0BE9$               JRNZ    :ERR1           ;STOP

0BC7  CD1A0C            :1:     CALL    SEEK            ;SEEK TO TRACK
0BCA  381D ^0BE9$               JRC     :ERR1           ;STOP

0BCC  DDE1      $               POP     IX              ;ADDRESS TO CALL IN IX
0BCE  C1                        POP     BC              ;NUMBER OF SECTORS TO R/W
0BCF  21D50B                    LDK     HL,:RTRY1
0BD2  E5                        PUSH    HL              ;FOR RETURN
0BD3  DDE9      $               JMP     [IX]            ;CALL AND RETURN TO :2

0BD5  C1                :RTRY1: POP     BC              ;RESTORE NUMBER OF SECTORS TO READ
0BD6  D0                        RNC                     ;RETURN IF GOOD

0BD7  2105EF                    LDK     HL,RTRY
0BDA  35                        DEC     [HL]
0BDB  2811 ^0BEE$               JRZ     :FIN            ;NO MORE RETRYS

0BDD  7E                        LD      A,[HL]          ;GET NUMBER OF RETRY
0BDE  FE09                      CMP     NRETRY-1
0BE0  20D7 ^0BB9$               JRNZ    :RLOOP          ;LOOP

0BE2  CD800B                    CALL    SENDEN          ;CHECK TRACK ON THE SECOND RETRY
0BE5  2007 ^0BEE$               JRNZ    :FIN            ;STOP IF ERROR

0BE7  18D0 ^0BB9$               JR      :RLOOP          ;LOOP

0BE9  FDE1      $ :ERR1:        POP     IY
0BEB  C1                        POP     BC              ;IF ERROR BEFORE READ OR WRITE

0BEC  C1                        POP     BC
0BED  C9                        RET

0BEE  37                :FIN:   STC
```

```
0BEF  C9              RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright ©1982 Osborne Computer Corporation


```
0BF0                    SCTRKR:
                        ;SET CONTROLER TRACK REGESTER
                        ;D.TRKR <= SAVTRK
                        ;USED IN FORMATING WHEN YOU DON'T KNOW WHERE THE HEAD IS
                        ;ENTRY
                        ;SAVTRK =        TRACK

                        ;EXIT
                        ;D.TRKR =        SAVTRK

0BF0                            PROC
0BF0  3A15EF                    LD      A,SAVTRK
0BF3  320121                    STO     A,D.TRKR
0BF6  C9                        RET
```

Copyright ©1982 Osborne Computer Corporation

```
0BF7                    HOME:
                        ;HOME DISK DRIVE
                        ;DRIVE IS ALREADY SELECTED AND READY
                        ;If "SEKDEL" has the verify bit set this proc will check for seek and crc errors
                        ;ENTRY
                        ;SDISK  =        DRIVE

                        ;EXIT
                        ;CBIT   =        SET IF ERROR

0BF7                            PROC
0BF7  3A13EF                    LD      A,SEKDEL        ;SEEK DELAY
0BFA  E607                      ANI     0000_0111B      ;SPEAD & VERIFY BITS ONLY
0BFC  3213EF                    STO     A,SEKDEL

0BFF  3E00                      LDK     A,D.RES
0C01  CD760D                    CALL    PSEKC           ;PERFORM HOME COMMAND
0C04  D8                        RC                      ;IF ERROR

0C05  3A0021                    LD      A,D.STSR
0C08  CB57         $            BIT     2,A
0C0A  280C ^0C18$               JRZ     :1              ;IF NOT ON TRACK ZERO

0C0C  3A13EF                    LD      A,SEKDEL
0C0F  E604                      ANI     0000_0100B      ;VERIFY?
0C11  C8                        RZ                      ;NO VERIFY GOOD RETURN

0C12  3A0021                    LD      A,D.STSR
0C15  E618                      ANI     0001_1000B      ;TEST SEEK AND CRC
0C17  C8                        RZ                      ;GOOD RETURN

0C18  37                :1:     STC                     ;IF ERROR
0C19  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation


```
0C1A                       SEEK:
                           ;SEEK TO TRACK DEFINED BY SAVTRK
                           ;TRACK REG UPDATED AND VERIFIED
                           ;ENTRY
                           ;SAVTRK SET TO DESIRED TRACK

                           ;EXIT
                           ;CBIT    =       SET IF ERROR
                           ;         IF NO ERROR CONTROLER TRACK = SAVTRK

0C1A                             PROC
0C1A  210121                     LDK     HL,D.TRKR
0C1D  3A15EF                     LD      A,SAVTRK
0C20  BE                         CMP     [HL]
0C21  C8                         RZ                      ;RETURN

0C22  320321                     STO     A,D.DATR        ;SET TRACK WANTED
0C25  3E10                       LDK     A,D.SEK
0C27  CD760D                     CALL    PSEKC           ;PERFORM SEEK COMMAND
0C2A  D8                         RC                      ;IF ERROR

0C2B  3A13EF                     LD      A,SEKDEL
0C2E  E604                       ANI     0000_0100B      ;VERIFY?
0C30  C8                         RZ                      ;NO VERIFY GOOD RETURN

0C31  3A0021                     LD      A,D.STSR
0C34  E618                       ANI     0001_1000B      ;TEST SEEK AND CRC
0C36  C8                         RZ                      ;GOOD RETURN

0C37  37                         STC                     ;IF ERROR
0C38  C9                         RET
```

Copyright © 1982 Osborne Computer Corporation

```
0C39                 STEP:
                     ;STEP ONE TRACK
                     ;SAVTRK IS NOT USED IN THIS PROC
                     ;CONTROLER TRK REG IS UPDATED
                     ;VERIFY IS PERFORMED
                     ;ENTRY
                     ;NONE

                     ;EXIT
                     ;CBIT    =       SET IF ERROR
                     ;        IF NO ERROR CONTROLER TRACK = TRACK +/- 1

0C39                          PROC
0C39  3E20                    LDK     A,D.STP
0C3B  CD760D                  CALL    PSEKC           ;PERFORM STEP COMMAND
0C3E  D8                      RC                      ;IF ERROR

0C3F  3A13EF                  LD      A,SEKDEL
0C42  E604                    ANI     0000_0100B      ;VERIFY?
0C44  C8                      RZ                      ;NO VERIFY GOOD RETURN

0C45  3A0021                  LD      A,D.STSR
0C48  E618                    ANI     0001_1000B      ;TEST SEEK AND CRC
0C4A  C8                      RZ                      ;GOOD RETURN

0C4B  37                      STC                     ;IF ERROR
0C4C  C9                      RET
```

Copyright © 1982 Osborne Computer Corporation


```
0C4D                    STEPIN:
                        ;STEP IN ONE TRACK
                        ;SAVTRK IS NOT USED IN THIS PROC
                        ;CONTROLER URK REG IS UPDATED
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;CBIT    =       SET IF ERROR
                        ;        IF NO ERROR CONTROLER TRACK = TRACK + 1

0C4D                             PROC
0C4D  3E40                       LDK     A,D.STPI
0C4F  CD760D                     CALL    PSEKC           ;PERFORM STEP-IN COMMAND
0C52  D8                         RC                      ;IF ERROR

0C53  3A13EF                     LD      A,SEKDEL
0C56  E604                       ANI     0000_0100B      ;VERIFY?
0C58  C8                         RZ                      ;NO VERIFY GOOD RETURN

0C59  3A0021                     LD      A,D.STSR
0C5C  E618                       ANI     0001_1000B      ;TEST SEEK AND CRC
0C5E  C8                         RZ                      ;GOOD RETURN

0C5F  37                         STC                     ;IF ERROR
0C60  C9                         RET
```

Copyright © 1982 Osborne Computer Corporation


```
0C61                STEPOUT:
                    ;STEP OUT ONE TRACK
                    ;SAVTRK IS NOT USED IN THIS PROC
                    ;CONTROLER TRK REG IS UPDATED
                    ;VERIFY IS PERFORMED
                    ;ENTRY
                    ;NONE

                    ;EXIT
                    ;CBIT    =       SET IF ERROR
                    ;        IF NO ERROR CONTROLER TRACK = TRACK - 1

0C61                        PROC
0C61  3E60                  LDK     A,D.STPO
0C63  CD760D                CALL    PSEKC           ;PERFORM STEP-OUT COMMAND
0C66  D8                    RC                      ;IF ERROR

0C67  3A13EF                LD      A,SEKDEL
0C6A  E604                  ANI     0000_0100B      ;VERIFY?
0C6C  C8                    RZ                      ;NO VERIFY GOOD RETURN

0C6D  3A0021                LD      A,D.STSR
0C70  E618                  ANI     0001_1000B      ;TEST SEEK AND CRC
0C72  C8                    RZ                      ;GOOD RETURN

0C73  37                    STC                     ;IF ERROR
0C74  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation


```
0C75                    READ:
                        ;ENTRY
                        ;B        =       NUMB OF SECTORS TO READ

                        ;EXIT
                        ;HL       =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                        ;CBIT     =       SET IF ERROR

0C75                              PROC

0C75  3E80                        LDK     A,D.RDS
0C77  32D1EF                      STO     A,RDT_WRTS

0C7A  C3820C                      JMP     RD_WRT          ;JMP AND RETURN TO CALLING PROC
```

Copyright © 1982 Osborne Computer Corporation

```
0C7D                WRITE:
                    ;ENTRY
                    ;B          =       NUMB OF SECTORS TO WRITE

                    ;EXIT
                    ;HL         =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                    ;CBIT       =       SET IF ERROR

0C7D                            PROC

0C7D  3EA0                      LDK     A,D.WRTS
0C7F  32D1EF                    STO     A,RDT_WRTS

                    ;           JMP     RD_WRT          ;FALLS THROUGH TO RD_WRT
```

Copyright © 1982 Osborne Computer Corporation

```
0C82                    RD_WRT:
                        ;READ OR WRITE
                        ;ENTRY
                        ;B      =       NUMB OF SECTORS TO READ OR WRITE
                                ;RDT_WRTS       =       D.RDS OR D.WRTS

                        ;EXIT
                        ;HL     =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                        ;CBIT   =       SET IF ERROR

0C82                            PROC

                        ;SET SECTOR REG

0C82  3A14EF                    LD      A,SAVSEC
0C85  320221                    STO     A,D.SECR

                        ;SET DE TO NUMBER OF BYTES IN ONE SECTOR

0C88  C5                        PUSH    BC              ;SAVE NUMBER OF SECTORS TO READ OR WRITE

0C89  218000                    LDK     HL,128
0C8C  3AD0EF                    LD      A,SAVTYP        ;DISK TYPE
0C8F  CB3F        $             SRL     A               ;DUMP TWO BITS
0C91  CB3F        $             SRL     A
0C93  E603                      ANI     0000_0011B      ;SIZE ONLY
0C95  B7                        ORA     A               ;SET FLAGS
0C96  2804 ^0C9C$               JRZ     :2              ;IF 128

0C98  47                        MOV     B,A
0C99  29                :BLOOP: ADD     HL,HL           ;SHIFT LEFT ONE BIT
0C9A  10FD ^0C99$               DJNZ    :BLOOP

0C9C  EB                :2:     EX      DE,HL           ;DE=HL

0C9D  C1                        POP     BC              ;RESTORE NUMBER OF SECTORS TO READ OR WRITE
0C9E  C5                        PUSH    BC              ;SAVE NUMBER OF SECTORS TO READ OR WRITE

                        ;GET COMMAND AND CHECK FOR MULTI-SECTOR

0C9F  78                        MOV     A,B             ;GET NUMBER OF SECTORS
0CA0  0E00                      LDK     C,0             ;MAKE NONMULTI-SECTOR
0CA2  FE02                      CMP     2
0CA4  3802 ^0CA8$               JRC     :1              ;IF LESS THAN TWO SECTORS
0CA6  0E10                      LDK     C,10H           ;MAKE MULTI-SECTOR

0CA8  3AD1EF            :1:     LD      A,RDT_WRTS      ;GET D.RDS OR D.WRTS
0CAB  B1                        OR      C               ;MAKE MULTI-SECTOR OR NONMULTI-SECTOR

                        ;SET HL TO NUMBER OF BYTES TO TRANSFER

0CAC  210000                    LDK     HL,0
0CAF  19                :LOOP:  ADD     HL,DE
0CB0  10FD ^0CAF$               DJNZ    :LOOP
0CB2  E5                        PUSH    HL              ;SAVE LENTH

                        ;GIVE COMMAND

0CB3                            DI
0CB4  CD840D                    CALL    FDSK
```

Copyright © 1982 Osborne Computer Corporation

```
0CB7  C1                       POP    BC            ;RESTORE LENTH
0CB8  3003 ^0CBD$              JRNC   :7            ;IF GOOD

0CBA  D1                       POP    DE            ;RESTORE STACK
0CBB                           EI
0CBC  C9                       RET                  ;ERROR RETURN

                   ;DO DMA

0CBD  2A0FEF       :7:         LD     HL,DMADR      ;HL = DMA ADDRESS

0CC0  11D10C                   LDK    DE,:6
0CC3  D5                       PUSH   DE            ;FOR RETURN

0CC4  3AD1EF                   LD     A,RDT_WRTS    ;GET COMMAND
0CC7  FE80                     CMP    D.RDS
0CC9  2003 ^0CCE$              JRNZ   :5            ;IF WRITE

0CCB  C30A0E                   JMP    DMARD         ;READ DMA RETURNS TO :6
0CCE  C3200E       :5:         JMP    DMAWRT        ;WRITE DMA RETURNS TO :6

                   ;CHECK FOR BUSY AND RESET

0CD1  1A           :6:         LD     A,[DE]        ;GET STATUS
0CD2  CB47       $             BIT    0,A
0CD4  C1                       POP    BC            ;RESTORE NUMBER OF SECTORS
0CD5  280E ^0CE5$              JRZ    :4            ;IF NOT BUSY
0CD7  05                       DEC    B             ;SUBTRACK ONE FROM THE NUMBER OF SECTORS AND SET THE ZERO FLAG
0CD8  2005 ^0CDF$              JRNZ   :A1           ;IF MORE THAN ONE, FORCE INTERUPT

0CDA  CDE60D                   CALL   WBUSY         ;IF NON MULTI-SECTOR R/W WAIT FOR BUSY TO DROP
0CDD  1803 ^0CE2$              JR     :A2

0CDF  CD640D       :A1:        CALL   FORINT        ;CLEAR BUSY
0CE2  3A0021       :A2:        LD     A,D.STSR      ;GET STATUS

                   ;CHECK FOR ERRORS

0CE5  E65C         :4:         ANI    0101_1100B    ;TEST write protect, rnf, crc, and lost data
0CE7  2801 ^0CEA$              JRZ    :3            ;IF GOOD

                   *NOTE*
                   ;      If this was a mult-sector function their will not be a Record Not Found error, as the manual
                   ;states, because we left the chip before it got to the next sector to find it wasn't there.

0CE9  37                       STC                  ;IF ERROR RECORD CONTROLER REGESTERS

0CEA               :3:         EI
0CEB  C9                       RET                  ;RETURN
```

Copyright © 1982 Osborne Computer Corporation

```
0CEC                    RADR:
                        ;Read Address info.
                        ;READS SIX BYTES INTO "DSTSB"

                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;A      =       OFFH IF TIME OUT ERROR
                        ;CBIT   =       SET IF ERROR
                        ;D.TRKR =       HEAD POSITION
                        *NOTE*
                        ;       SETS TRACK REG IN CONTROLER IF GOOD

0CEC                            PROC
0CEC  3EC0                      LDK     A,D.RDA
0CEE                            DI
0CEF  CD840D                    CALL    FDSK            ;function disk
0CF2  3830 ^0D31$               JRC     :1

                        ;WAIT FOR FIRST DRQ OR TIME OUT

                                ;SET REGESTERS FOR DMA TRANSFER

0CF4  010600                    LDK     BC,6            ;SIX BYTES TO READ
0CF7  2109EF                    LDK     HL,DSTSB        ;FBA FOR DMA

                                ;WAIT FOR 1/4 OF A TRACK(60MS) OR DRQ

0CFA  110811                    LDK     DE,4363
0CFD  3A0021            :LOOP:  LD      A,D.STSR        ;(13) GET STATUS
0D00  1F                        RAR                     ;(4)
0D01  1F                        RAR                     ;(4)
0D02  DA120D                    JC      :3              ;(10) GOT DRQ

0D05  1B                        DEC     DE              ;(6)
0D06  7A                        MOV     A,D             ;(4)
0D07  B3                        ORA     E               ;(4)
0D08  C2FD0C                    JNZ     :LOOP           ;(10)

                                ;INDICATE TIME CUT ERROR

0D0B  CD640D                    CALL    FORINT          ;CLEAR BUSY
0D0E  3EFF                      LDK     A,OFFH          ;A=OFFH
0D10  181E ^0D30$               JR      :2              ;INDICATE A TIME OUT ERROR

                                ;TRANSFER FIRST BYTE AND CALL DMARD FOR LAST FIVE BYTES

0D12  3A0321            :3:     LD      A,D.DATR        ;(13) GET BYTE
0D15  77                        STO     A,[HL]          ;(7) STORE BYTE
0D16  23                        INC     HL              ;(6)
0D17  0B                        DEC     BC              ;(6)
0D18  CD0A0E                    CALL    DMARD           ;(17) CALL DMARD

                        ;RETURN FROM DMARD AND WAIT FOR BUSY TO BE RESET

0D1B  CDE60D                    CALL    WBUSY
0D1E  3811 ^0D31$               JRC     :1              ;IF TIME OUT ERROR

0D20  3A0021                    LD      A,D.STSR        ;GET STATUS
```

Copyright © 1982 Osborne Computer Corporation

```
                   ;CHECK FOR ERRORS

                   ;        ANI     0001_1000B      ;TEST RNF AND CRC
0D23  E61C                  ANI     0001_1100B      ;TEST RNF, CRC AND DATA LOST
0D25  2009 ^0D30$           JRNZ    :2              ;IF ERROR

                   ;SET TRACK REGESTER

0D27  3A0221                LD      A,D.SECR        ;GET TRACK
0D2A  320121                STO     A,D.TRKR        ;SET TRACK
0D2D  AF                    XRA     A               ;RESET CARRY
0D2E  1801 ^0D31$           JR      :1

0D30  37           :2:      STC                     ;SET CBIT

0D31               :1:      EI
0D32  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0D33                  READTRK:
                      ;READ ONE TRACK FROM THE DRIVE
                      ;ENTRY
                      ;DMADR   =       FWA OF BUFFER

                      ;EXIT
                      ;CBIT    =       SET IF ERROR

0D33                          PROC
0D33  3EE0                    LDK     A,D.RDT
0D35                          DI
0D36  CD840D                  CALL    FDSK
0D39  380C ^0D47$             JRC     :1              ;IF ERROR

                      ;DO DMA

0D3B  01FFFF                  LDK     BC,0FFFFH       ;FOR ROM 1.2
0D3E  2A0FEF                  LD      HL,DMADR
0D41  CD0A0E                  CALL    DMARD           ;IN ROM
0D44                          EI
0D45  AF                      XRA     A
0D46  C9                      RET

0D47                  :1:     EI
0D48  C9                      RET
```

```
0D49                FMTTRK:
                    ;FORMAT ONE TRACK
                    ;ENTRY
                            ;DMADR  =       FWA OF BUFFER

                    ;EXIT
                    ;CBIT   =       SET IF ERROR
0D49                        PROC
0D49  3EF0                  LDK     A,D.WRTT
0D4B                        DI
0D4C  CD840D                CALL    FDSK
0D4F  3810 ^0D61$           JRC     :1              ;IF ERROR

                    ;DO DMA

0D51  01FFFF                LDK     BC,0FFFFH       ;DEFEAT COUNTER
0D54  2A0FEF                LD      HL,DMADR
0D57  CD200E                CALL    DMAWRT

                    ;CHECK FOR ERROR

0D5A  3A0021                LD      A,D.STSR        ;GET STATUS
0D5D  E644                  ANI     0100_0100B      ;TEST write protect, and data lost
0D5F  2801 ^0D62$           JRZ     :2              ;IF GOOD

0D61  37            :1:     STC

0D62                :2:     EI
0D63  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0D64                    FORINT:
                        ;INTERRUPT DISK CONTROLLER
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;BUSY CLEARED.
0D64                            PROC
0D64  F5                        PUSH    AF
0D65  C5                        PUSH    BC

0D66  3ED0                      LDK     A,D.FINT
0D68  320021                    STO     A,D.CMDR

                        ;WAIT FOR AT LEAST 28 MICROSECONDS

0D6B  B7                        ORA     A               ;(4)
0D6C  0607                      LDK     B,7             ;(7)
0D6E  10FE ^0D6E$ :WLOOP: DJNZ  :WLOOP          ;(91) = (13*7) WAIT

                        ;CHECK FOR BUSY DROP

0D70  CDE60D                    CALL    WBUSY

0D73  C1                        POP     BC
0D74  F1                        POP     AF
0D75  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation


```
0076                    PSEKC:
                        ;OR IN SEKDEL AND PERFORM SEEK TYPE COMMAND
                        ;ENTRY
                        ;A      =       SEEK TYPE COMMAND

                        ;EXIT
                        ;CBIT   =       SET IF ERROR
                        ;COMMAND SENT TO CONTROLER IF NO ERROR

0076                            PROC
0076  47                        MOV     B,A             ;B = A
0077  3A13EF                    LD      A,SEKDEL
007A  E617                      ANI     0001_0111B      ;ONLY UPDATE,VERIFY, & SPEAD
007C  B0                        ORA     B               ;OR IN COMMAND
007D  CD840D                    CALL    FDSK            ;FUNCTION DISK
0080  D8                        RC                      ;IF ERROR
0081  C3E60D                    JMP     WBUSY
```

Copyright © 1982 Osborne Computer Corporation


```
0D84                    FDSK:
                        ;FUNCTION DISK ROUTINE
                        ;THIS IS THE ONLY ROUTINE THAT WRITES TO THE COMMAND REGESTER OF THE CONTROLER CHIP
                        ;THIS ROUTINE HAS A BUILT IN DELAY OF AT LEAST 28 MICRO SEC. BEFORE READING THE STATUS ON THE CHIP
                        ;ENTRY
                        ;A      =       FUNCTION CODE

                        ;EXIT
                        ;A      =       OFFH IS TIME CUT ERROR
                        ;CBIT   =       SET IF ERROR

0D84                            PROC
0D84  210021                    LDK     HL,D.STSR           ;STATUS AND COMMAND REGESTER
0D87  CB46        $             BIT     0,[HL]
0D89  2803 ^0D8E$               JRZ     :1                  ;IF NOT BUSY

0D8B  CD640D                    CALL    FORINT              ;RESET BUSY

0D8E  77                :1:     STO     A,[HL]              ;FUNCTION DRIVE(WRITE COMMAND TO CONTROLER)

                        ;WAIT FOR AT LEAST 28 MICROSECONDS

0D8F  B7                        ORA     A                   ;(4)
0D90  0607                      LDK     B,7                 ;(7)
0D92  10FE ^0D92$ :WLOOP:       DJNZ    :WLOOP              ;(91) = (13*7) WAIT

                        ;WAIT FOR BUSY TO BE SET

0D94  3EFF                      LDK     A,0FFH              ;(7)
0D96  47                        MOV     B,A                 ;(4) 256 LOOPS

0D97  CB46        $ :LOOP:      BIT     BS.BSY,[HL]         ;BS.BSY
0D99  2004 ^0D9F$               JRNZ    :3                  ;IF CHIP WENT BUSY
0D9B  10FA ^0D97$               DJNZ    :LOOP               ;IF NOT TIME-OUT

0D9D  37                        STC                         ;IF ERROR A=FF AND CBIT = SET
0D9E  C9                        RET

0D9F  326AEF            :3:     STO     A,DACTVE            ;SET DRIVE ACTIVE COUNTER
0DA2  AF                        XRA     A                   ;RESET CARRY FLAG
0DA3  C9                        RET
```

```
0DA4                    SELDRV:
                        ;SELECT DRIVE
                        ;ENTRY
                        ;SDISK  =       DRIVE TO SELECT

                        ;EXIT
                        ;ZBIT   =       SET IF PIABD WAS THE SAME AS SDISK
                        ;ZBIT   =       RESET IF PIABD WAS DIFFERENT THAN SDISK
                        ;CBIT   =       SET IF THERE ARE NO INDEX PULSES

0DA4                            PROC
0DA4  CDD40D                    CALL    SELDEN          ;SELECT DENSITY

0DA7  3A17EF                    LD      A,SDISK
0DAA  21C7EF                    LDK     HL,DSKSWP       ;DISK DRIVE SWAP CELL
0DAD  AE                        XOR     [HL]            ;SWAP A FOR B IF DSKSWP=1
0DAE  E601                      AND     1               ;CAN ONLY BE 0 OR 1
0DB0  FE01                      CMP     1
0DB2  2002 ^0DB6$               JRNZ    :1              ;IF NOT DRIVE 1
0DB4  3E40                      LDK     A,40H

0DB6  C640              :1:     ADI     40H
0DB8  4F                        MOV     C,A
0DB9  3A62EF                    LD      A,PIABD
0DBC  47                        MOV     B,A
0DBD  E6C0                      ANI     1100_0000B      ;GET DRIVE BITS ONLY
0DBF  B9                        CMP     C
0DC0  280B ^0DCD$               JRZ     :2              ;IF DRIVE ALREADY SELECTED

                        ;SELECT DRIVE

0DC2  CDFF0D                    CALL    RDSKD           ;TURN DRIVE ON
0DC5  3EFA                      LDK     A,250
0DC7  CD2D09                    CALL    DELAY           ;WAIT FOR MOTOR SPIN UP
                        ;       LDK     A,250
                        ;       CALL    DELAY           ;2ND DELAY

0DCA  3E01                      LDK     A,1             ;INDICATED DRIVE WAS NOT SELECTED
0DCC  B7                        ORA     A               ;SET FLAGS

0DCD  216AEF            :2:     LDK     HL,DACTVE
0DD0  36FF                      STO     0FFH,[HL]
0DD2                            EI
0DD3  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
0DD4                    SELDEN:
                        ;SELECT SINGLE OR DOUBLE DENSITY
                        ;ENTRY
                        ;SAVTYP =        BIT 0:
                        ;                1 = SINGLE, 0 = DOUBLE

                        ;EXIT
                        ;NONE
                        *NOTE   Bit 0 of "PIAAD" :
                        ;       set     =       single density
                        ;       reset   =       double density

0DD4                            PROC
0DD4  3A61EF                    LD      A,PIAAD         ;PRESENT VALUE OF PIA REG
0DD7  E6FE                      ANI     1111_1110B      ;CLEAR BIT 0
0DD9  4F                        MOV     C,A

                        ;SET DENSITY BIT

0DDA  3AD0EF                    LD      A,SAVTYP        ;GET DISK TYPE INFO
0DDD  0F                        RRC                     ;CBIT <= BIT 0
0DDE  3002 ^0DE2$               JRNC    :1              ;IF "SAVTYP" BIT0 IS 0

0DE0  CBC1       $              SBIT    0,C             ;SET BIT 0 OF REG C

0DE2  CD6409     :1:            CALL    OPAD            ;FUNCTION PIA
0DE5  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0DE6                    WBUSY:
                        ;WAIT FOR BUSY TO CLEAR
                        ;This routine must wait for 2 seconds
                        ;2 seconds is the time it takes for the chip to seek 39 tracks and have five index holes go by.
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;A       =       0FFH IF TIME OUT OCCURRED
                        ;CBIT    =       SET  "    "    "      "

0DE6                            PROC
0DE6  010000                    LDK     BC,0

0DE9  3A0021            :LOOP:  LD      A,D.STSR        ;(13)
0DEC  CB47        $             BIT     0,A             ;(8) DS.BSY
0DEE  280E ^0DFE$               JRZ     :1              ;(7) GOOD RETURN
0DF0  E3                        EX      [SP],HL         ;(23) DELAY
0DF1  E3                        EX      [SP],HL         ;(23) DELAY
0DF2  2B                        DEC     HL              ;(6) DELAY
0DF3  23                        INC     HL              ;(6) DELAY
0DF4  2B                        DEC     HL              ;(6) DELAY
0DF5  23                        INC     HL              ;(6) DELAY
0DF6  0B                        DEC     BC              ;(6)
0DF7  78                        MOV     A,B             ;(4)
0DF8  B1                        OR      C               ;(4)
0DF9  20EE ^0DE9$               JRNZ    :LOOP           ;(12) IF NOT TIME-OUT

0DFB  3EFF                      LDK     A,0FFH          ;TIME OUT ERROR
                        ;       CALL    FORINT          ;RESET BUSY
0DFD  37                        STC                     ;SET ERROR

0DFE  C9                :1:     RET
```

Copyright © 1982 Osborne Computer Corporation


```
0DFF                    RDSKD:
                        ;SELECT DRIVE BY SETING THE "PIA" WITH THE VALUE SPECIFIED BY C
                        ;ENTRY
                        ;C      =       DRIVE

                        ;EXIT
                        ;NONE
0DFF                            PROC
0DFF  3A62EF                    LD      A,PIABD
0E02  E63F                      ANI     0011_1111B      ;GET VIO OFFSET AND BELL
0E04  B1                        OR      C
0E05  4F                        MOV     C,A
0E06  CD7109                    CALL    OPBD            ;FUNCTION PIO-B
0E09  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation


```
0E0A                    DMARD:
                        ;TRANSFER DATA FROM CONTROLER TO MEMORY
                        ;ENTRY
                        ;BC      =       BYTES TO TRANSFER
                        ;HL      =       FWA OF BUFFER

                        ;EXIT
                        ;HL      =       NEXT ADDRESS
                        ;DE      =       D.STSR

0E0A                            PROC
0E0A  110021                    LDK     DE,D.STSR       ;(10)

0E0D  1A                :LOOP:  LD      A,[DE]          ;(7) GET STATUS
0E0E  1F                        RAR                     ;(4)
0E0F  D0                        RNC                     ;(5) RETURN IF NO BUSY
0E10  1F                        RAR                     ;(4)
0E11  D20D0E                    JNC     :LOOP           ;(10) IF NO DRQ

0E14  3A0321                    LD      A,D.DATR        ;(13) GET BYTE
0E17  77                        STO     A,[HL]          ;(7) STORE BYTE
0E18  23                        INC     HL              ;(6)
0E19  0B                        DEC     BC              ;(6)
0E1A  78                        MOV     A,B             ;(4)
0E1B  B1                        ORA     C               ;(4)
0E1C  C20D0E                    JNZ     :LOOP           ;(10)
0E1F  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation


```
0E20                DMAWRT:
                    ;Xfer data from memory to disk
                    ;ENTRY
                    ;BC      =       BYTES TO TRANSFER
                    ;HL      =       FWA OF BUFFER

                    ;EXIT
                    ;HL      =       NEXT ADDRESS

0E20                        PROC
0E20  110021                LDK     DE,D.STSR

0E23  1A            :LOOP:  LD      A,[DE]          ;GET STATUS
0E24  1F                    RAR
0E25  D0                    RNC                     ;RETURN IF NO BUSY
0E26  1F                    RAR
0E27  D2230E                JNC     :LOOP           ;IF NO DRQ

0E2A  7E                    LD      A,[HL]          ;GET BYTE
0E2B  320321                STO     A,D.DATR        ;STORE BYTE
0E2E  23                    INC     HL
0E2F  0B                    DEC     BC
0E30  78                    MOV     A,B
0E31  B1                    ORA     C
0E32  C2230E                JNZ     :LOOP
0E35  C9                    RET
```

```
0E36                DDRV:
                    ;Deselect drive
                    ;Entry
                    ;SDISK  =       current disk drive
0E36                        PROC
0E36  3A62EF                LD      A,PIABD
0E39  E61F                  AND     1_1111b
0E3B  4F                    MOV     C,A
0E3C  C37109                JMP     OPBD            ;deselect last drive
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        *[7]

0E3F                    FORMAT:
                        ;       This proc will format the next track in IBM 3740 format consisting of 40 tracks, with each
                        ;track containing 10 sectors.
                        ;Entry
                        ;BC      =       FWA of buffer
                                ;BUF+0   =       DW length
                                ;BUF+2   =       beginning of data
                                ;SAVTRK  =       THE TRACK TO BE FORMATED

                        ;EXIT
                        ;NONE
0E3F                            PROC

                        *SET UP BUFFER AND "DMADR"

                                ;SET DMADR AND GET LBA+1 OF PRESENT BUFFER

0E3F  C5                        PUSH    BC
0E40  E1                        POP     HL              ;HL = BC
0E41  4E                        LD      C,[HL]
0E42  23                        INC     HL
0E43  46                        LD      B,[HL]          ;BC = LENTH OF FORMAT DATA
0E44  23                        INC     HL
0E45  220FEF                    STO     HL,DMADR        ;SET DMA
0E48  09                        ADD     HL,BC           ;HL = HL + BC

                                ;TEST DENSITY AND SET REG D TO 04EH OR OFFH

0E49  012003                    LDK     BC,800          ;800 BYTES OF PAD FOR END OF TRACK

0E4C  3AD0EF                    LD      A,SAVTYP
0E4F  164E                      LDK     D,04EH          ;DOUBLE
0E51  0F                        RRC     A
0E52  3002 ^0E56$               JRNC    :PLOOP          ;IF DOUBLE
0E54  16FF                      LDK     D,0FFH          ;SINGLE

                                ;PAD REST OF BUFFER

0E56  72                :PLOOP: STO     D,[HL]
0E57  23                        INC     HL
0E58  0B                        DEC     BC
0E59  78                        MOV     A,B
0E5A  B1                        ORA     C
0E5B  20F9 ^0E56$               JRNZ    :PLOOP

                        *FUNCTION DRIVE

                                ;SELECT DRIVE

0E5D  CDA40D                    CALL    SELDRV
0E60  3827 ^0E89$               JRC     :ERROR

                                ;TEST FOR STEP OR NO-STEP

0E62  2115EF                    LDK     HL,SAVTRK
0E65  3A0121                    LD      A,D.TRKR        ;TRACK REG
0E68  BE                        CMP     [HL]
0E69  2817 ^0E82$               JRZ     :1              ;IF SAVTRK AND TRACK REG ARE THE SAME SKIP THE STEP
```

Copyright © 1982 Osborne Computer Corporation


```
                                ;STEP IN ONE TRACK

0E68  3A13EF                    LD      A,SEKDEL
0E6E  F610                      ORI     0001_0000B      ;UPDATE
0E70  3213EF                    STO     A,SEKDEL        ;SET UP SEKDEL

0E73  CD4D0C                    CALL    STEPIN

0E76  F5                        PUSH    AF              ;SAVE FLAGS
0E77  3A13EF                    LD      A,SEKDEL
0E7A  E603                      ANI     0000_0011B      ;ONLY SPEAD LEFT
0E7C  3213EF                    STO     A,SEKDEL        ;RESET SEKDEL
0E7F  F1                        POP     AF              ;RESTORE

0E80  3807 ^0E89$               JRC     :ERROR

                                ;FORMAT TRACK

0E82  CD490D          :1:       CALL    FMTTRK
0E85  3802 ^0E89$               JRC     :ERROR

0E87  AF                        XRA     A
0E88  C9                        RET

0E89  3EFF            :ERROR:   LDK     A,0FFH
0E8B  B7                        ORA     A
0E8C  C9                        RET


      = 0E8C          RLWA      =       *-1             ;LWA OF ROM RESIDENT CODE
                                MSG     'LENGTH OF THIS ROM IS = ',RLWA
      'LENGTH OF THIS ROM IS = 0E8C'
      = 0000                    IF      RLWA > 0FFCH
   -                  .9        ERROR   CODE TOO LARGE..
                                ENDIF

0E8D  = 0FFD                    org     0ffdh
0FFD  = 0003          SERNO:    ds      3
```

Copyright © 1982 Osborne Computer Corporation


```
                         *[R]

1000   = ED80                     ORG      MRAM
       = EFDA            SERFLG   =        0EFDAh

                         ;        Host disk xfer buffer and...
                         ;        Format track template holding buffer
ED80                     HSTBUF:
ED80   = 0180                     DS       256+128

                         ;        Directory Buffer
       = EE80            DIRBUF:  =        HSTBUF+256

EF00   = 0006            TEM      DS       6
       = EF01            RNDV     =        TEM+1     ;random number seed
       = EF02            ERCNT    =        RNDV+1    ;DW ERCNT
       = EF04            RTRC     =        ERCNT+2   ;retry count
       = EF05            RTRY     =        RTRC+1
EF06   = 0001            MPCHR    DS       1         ;prompt character
EF07   = 0001            ECHOP    DS       1         ;=0, list ehco off
EF08   = 0001            ROMRAM   DS       1         ;0= RAM, 1= ROM
EF09   = 0006            DSTSB    DS       6         ;Disk status bytes

                         ;        Disk operation temps and control
EF0F   = 0002            DMADR    DS       2         ;Address for read/write Disk
EF11   = 0002            DMAADR   DS       2         ;CBIOS, users DMA

                         ;        Note order of xxxSEC,xxxTRK,xxxDSK must be maintained
                         ;        along with length (1,2,1).
EF13   = 0001            SEKDEL:  DS       1         ;Set for seek-restore command in ROM
                                                     ;depends on disk type. Siemens = 3h, MPI = 0h
EF14   = 0001            SAVSEC   DS       1         ;last sector requested
EF15   = 0002            SAVTRK   DS       2         ;last track requested
EF17   = 0001            SDISK    DS       1         ;Selected disk drive (0,1)

       = EF14            ACTSEC   =        SAVSEC
       = EF15            ACTTRK   =        SAVTRK
       = EF17            ACTDSK   =        SDISK

EF18   = 0001            SEKSEC   DS       1
EF19   = 0002            SEKTRK   DS       2
EF1B   = 0001            SEKDSK   DS       1

EF1C   = 0001            HSTSEC   DS       1
EF1D   = 0002            HSTTRK   DS       2
EF1F   = 0001            HSTDSK   DS       1

EF20   = 0001            TEMSEC   DS       1         ;Used in bios only
EF21   = 0001            RDFLAG   DS       1         ;Read flag
EF22   = 0001            ERFLAG   DS       1         ;Error reporting
EF23   = 0001            WRTYPE   DS       1         ;Write operation type

EF24   = 000C            ALV:     DS       ALVS
EF30   = 0020            CSV:     DS       CSVS

                         ;        BIOS blocking-deblocking flags
EF50   = 0001            HSTACT:  DS       1         ;host active flag
EF51   = 0001            HSTWRT:  DS       1         ;Host written flag
EF52   = 0001            UNACNT:  DS       1         ;Unalloc rec count
EF53   = 0002            UNATRK:  DS       2         ;Track
```

Copyright © 1982 Osborne Computer Corporation


```
EF55  = 0001       UNASEC: DS      1       ;Sector
EF56  = 0001       LOGSEC: DS      1       ;Logical sector


EF57  = 0002       LDADR   DS      2
EF59  = 0001       KEYLCK  DS      1       ;Zero if locked keyboard
EF5A  = 0002       CURS    DS      2       ;current cursor position

                   ;       Keyboard scan temporaries
EF5C  = 0001       TKEY    DS      1       ;Tem holding key
EF5D  = 0001       HKCNT   DS      1       ;Debounce key
EF5E  = 0001       LKEY    DS      1       ;Last valid keystroke
EF5F  = 0001       CKEY    DS      1       ;Last control key
EF60  = 0001       ESCH    DS      1       ;ESC holding flag

                   ;PIAAD and PIABD must be kept sequential,PIAAD first
                   ;dependency in VC_HOME of BMKEY.asm
EF61  = 0001       PIAAD:  DS      1       ;Holds last PIA-A data
EF62  = 0001       PIABD:  DS      1       ;Holds last PIA-B data

                   ;       Calendar month, day year
EF63  = 0003       IDAY    DS      3
      = EF64       IMONTH  =       IDAY+1
      = EF65       IYR     =       IDAY+2

                   ;       Wall clock time cells and disk active
                   ;       see UPTIM: in BMKEY.asm
EF66  = 0006       HOURS:  DS      6
      = EF67       MINS:   =       HOURS+1
      = EF68       SECS:   =       HOURS+2
      = EF69       SEC6:   =       HOURS+3

                   ;       Used to deselect drive when there is NO activity
                   ;       on drive for n seconds.  See FDSK routine
      = EF6A       DACTVE: =       HOURS+4 ;=0 by FDSK, Used by UPTIM

      = EF6B       BELCNT: =       HOURS+5 ;^G bell timer cell


EF6C  = 0001       LLIMIT  DS      1       ;max #columns in a logical line
                   ;       MSG     'LLIMIT = ',LLIMIT,'h.'

                   ;       Disk drive current positions
EF6D  = 0002       LDSEL:  DS      2       ;Last selected drive
      = EF6E       LDTRK   =       LDSEL+1 ;Last track used for non-selected drive

EF6F  = 0002       IESTK:  DS      2       ;save current stk ptr


                   ;       Interrupt stack
EF71  = 0028               DS      20*2
EF99  = 0000       ISTK:   DS      0

                   ;       Stack entry
EF99  = 0028               DS      20*2
EFC1               BIOSTK:
EFC1  = 0000       ROMSTK: DS      0

EFC1  = 0001       ACIAD:  DS      1       ;last command byte written to ACIA
```

Copyright © 1982 Osborne Computer Corporation

```
EFC2  = 0004      R179x:  DS     4        ;179x register save area
EFC6  = 0001      KBDLY:  DS     1        ;keyboard debounce-delay cell

                  ;since CP/M CANNOT boot off B:, this cell is used
                  ;to invert the names of the 2 drives:
                  ;       =0, all normal, A=A:, B=B:
                  ;       =1, all inverted, A=B:, B=A:
EFC7  = 0001      DSKSWP  DS     1


                  ;       Z80 Alternate Register Set
EFC8                      ALIGN  10h
EFD0              RAGS:
EFD0  = 0002      DESAX:  DS     2        ;DE'
EFD2  = 0002      BCSAX:  DS     2        ;BC'
EFD4  = 0002      AFSAX:  DS     2        ;AF'
EFD6  = 0002      HLSAX:  DS     2        ;HL'

EFD8  = 0002      IXSAX:  DS     2        ;IX
EFDA  = 0002      IYSAX:  DS     2        ;IY
EFDC  = 0002      IVSAX:  DS     2        ;Interrupt page register


                  ;       8080 Register Save Area.
EFDE                      ALIGN  10h
EFE0              REGS:
EFE0  = 0001      ESAVE:  DS     1        ;E Register save location
EFE1  = 0001      DSAVE:  DS     1        ;D Register save location
EFE2  = 0001      CSAVE:  DS     1        ;C Register save location
EFE3  = 0001      BSAVE:  DS     1        ;B Register save location
EFE4  = 0001      FSAVE:  DS     1        ;FLAGS save location
EFE5  = 0001      ASAVE:  DS     1        ;A Register save location
EFE6  = 0001      LSAVE:  DS     1        ;L Register save location
EFE7  = 0001      HSAVE:  DS     1        ;H Register save location
EFE8  = 0002      PSAVE:  DS     2        ;PGM COUNTER save location
EFEA  = 0002      SSAVE:  DS     2        ;USER STACK pointer save location

EFEC  = 0002      BKPA:   DS     2        ;last breakpoint address
EFEE  = 0001      BKPC:   DS     1        ;Contents of bkp

EFEF  = 0001      VRTOFF  DS     1        ;LAST VERTICAL OFFSET TAKEN FROM COUT
                  ;
                  ;
                  ;       Interrupt Jump Vector is between EFF8, EFFF.
                  ;       Endx    MRAM


no ERRORs, 687 Labels, 5EDFh bytes not used.     Program LWA = EFF0h.
```

Copyright © 1982 Osborne Computer Corporation

```
s  @FREQ  0FA0
 n @KEY   0001     19# 4
   ACIAD  EFC1     87/14   128#59
   ACISTA 0B28      9/ 2    88/10    89/12    90#18
 n ACTDSK EF17    127#38
 n ACTSEC EF14    127#36
 n ACTTRK EF15    127#37
 n AFSAX  EFD4    129#16
s  ALIGN  mac     129/13   129/26
   ALLPOP mac       1#46     3/28
s  ALPHKY 0003     66/44
   ALPUSH mac       1#37     3/16    58/18
 n ALV    EF24    127#53
s  ALVS   000C    127/53
 n ASAVE  EFE5    129#32
   BCPM   02B3     14/51    15#17
 n BCSAX  EFD2    129#15
   BELCNT EF68      5/21    23/ 6    38/ 8   128#37
   BIOJP  00E7      7/ 2     7# 4
 n BIOSTK EFC1    128#56
 n BKPA   EFEC    129#38
 n BKPC   EFEE    129#39
s  BKS    0008     19/10    27/ 8
   BMON   0036      3/ 3     3#32     3/49     4/13
   BMON1  006A      3/46     4# 4
s  BRTBIT 0080     31/11    47/15    53/13
   BS.BSY 0000     93#27    93/28   117/32
   BS.DRQ 0001     93#30    93/31
   BS.SEK 0004     93#40    93/41
   BS.TK0 0002     93#34    93/35
 n BSAVE  EFE3    129#30
   CBELL  0007     19# 7    27/11
   CBOOT  0268      4/ 9     4/12     8/ 7    13#28
 n CCP    0000      1#34
   CCPADR EFD2      7/ 4    14/57    15/16    92# 6
s  CCRA   2901     78/25    78/45    79/13
s  CCRB   2903     78/32    78/54    81/21    81/23    81/26    81/40    81/42
                   81/45    82/20    82/22    82/25    84/19    84/21    84/24
 n CDEHL  0925     70# 2
s  CDISK  0004     15/15
   CHKEY  07D1     60/22    63# 9
   CI     037E      4/ 4     8/10    20#14
 n CKEY   EF5F    128#13
   CLRLN  0684     39/ 2    44/ 4    51/ 2    52#15    54/44
   COUT   0410      8/11    11/22    11/24    27#18
   COUT2  044A     26/24    29#20
s  CPDRA  2900     78/28    78/31    79/ 9    80/12    82/27    83/24    83/50
                   84/26    84/43    86/12
s  CPDRB  2902     78/35    78/38    79/10    80/10    81/15    82/13    83/15
                   84/13    86/14    86/18
s  CR     000D      4/ 5    10/ 2    12/10    12/20    12/28    12/38    12/46
                   27/ 6    69/ 5
 n CSAVE  EFE2    129#29
 n CSV    EF30    127#54
s  CSVS   0020    127/54
s  CTLKY  0002     66/40
   CTLTB  0913     67/57    69#43
   CURS   EF5A     28/38    42/33   128# 7
   D.CMDR 2100     93#19    93/21    93/22    93/23   115/13
   D.DATR 2103     93#23   103/17   111/49   122/20   123/20
```

Copyright © 1982 Osborne Computer Corporation

```
 n D.DEL   0014     93#52
   D.FINT  00D0     93#15   115/12
   D.RDA   00C0     93#12   111/16
   D.RDS   0080     93#10   107/11   110/16
   D.RDT   00E0     93#13   113/10
   D.RES   0000     93# 5   102/16
   D.SECR  2102     93#22   109/16   112/10
   D.SEK   0010     93# 6   103/18
   D.STP   0020     93# 7   104/14
   D.STPI  0040     93# 8   105/13
   D.STPO  0060     93# 9   106/14
   D.STSR  2100     93#20   102/20   102/28   103/26   104/22   105/21   106/22
                   110/35   111/31   111/60   114/22   117/13   120/15   122/12
                   123/11
   D.TRKR  2101     93#21   101/13   103/12   112/11   125/58
   D.WRTS  00A0     93#11   108/11
   D.WRTT  00F0     93#14   114/ 9
   DACTVE  EF6A    117/39   118/41   128#35
 s DBCT    0001     62/27
   DDRV    0E36     23/22   123#27
   DELAY   092D     70#19   118/34
 n DESAX   EFD0    129#14
sD DI      mac       3/12     3/33     3/49     5/11    21/11    31/10    33/37
                    47/13    53/12    55/11    56/16    56/38    58/15   109/60
                   111/18   113/12   114/11
 s DIMBIT  0000     33/38
 n DIRBUF  EE80    127#12
 s DISDIM  mac       3/ 3     3/13     5/12    31/13    33/41    47/17    53/15
                    55/14    56/19    56/41
 s DMA     0080
 n DMAADR  EF11    127#26
   DMADR   EF0F     14/24    16/17    18/ 4    18/17   110/10   113/18   114/17
                   125/26   127#25
   DMARD   0E0A      8/52   110/19   111/53   113/19   121#15
   DMAWRT  0E20      8/51   110/20   114/18   122#28
   DNARW   008C     66#11    67/35
   DOHOME  08AD     67/38    68#45
   DOLF    06A7     40/ 9    53#39
   DOLF2   06AA     42/16    54#12
 n DOWN    008C     19#38
 n DS.BSY  0001     93#28
 n DS.CRC  0008     93#38
 n DS.DRQ  0002     93#32
   DS.HDL  0020     93#44    93/45
   DS.INX  0002     93#31    93/32
 n DS.LSD  0004     93#36
 n DS.NRY  0080     93#48
 n DS.RNF  0010     93#42
   DS.SEK  0010     93#41    93/42
   DS.TKO  0004     93#35    93/36
 n DS.WTF  0020     93#45
 n DS.WTP  0040     93#47
 n DSAVE   EFE1    129#28
   DSKSWP  EFC7      4/ 6   118/15   129# 8
   DSTSB   EF09     98/37   111/26   127#22
   EBOOT   0194     10# 6    14/18    14/34    16/30    17/59    18/28
   ECHOP   EF07      5/23   127#20
   EDELC   05D5     26/18    46#18
   EEOL    05AD     26/21    43#11
   EFADR   0010     24# 6    45/ 5    45/14
```

```
   EFESC  0008     24# 7   28/41   29/26   45/ 5   45/14
   EFGR   0001     24#10   24/11   49/ 4   49/18
   EFHA   0002     24# 9   24/11   49/ 8   49/21
   EFMSK  0007     24#11   29/ 5   35/44   45/ 3   45/12
   EFSCR  0020     24# 5   45/14
   EFUN   0004     24# 8   24/11   49/11   49/24
 n EFX    0040     24# 4
sD EI     mac       3/ 8    3/30    5/11    6/13   21/17   31/14   33/42
                   47/18   53/16   55/15   56/20   56/42  110/ 6  113/21
                  118/44
   EINSRT 05EE     26/17   47#20
   EMBOOT 0187     10# 2   10/12
s  ENADIM mac      31/11   33/38   47/14   53/13   55/12   56/17   56/39
s  ENAROM mac       5/11   94# 2
s  ERC    007F     69/ 3   69/ 3   69/ 5
   ERCNT  EF02    127#16  127/17
 n ERFLAG EF22    127#50
 n ESAVE  EFE0    129#27
s  ESC    0018     12/ 4   12/ 8   12/12   12/12   12/14   12/16   12/18
                   12/18   12/22   12/26   12/30   12/32   12/34   12/36
                   12/40   12/44   12/49   12/51   28/47   69/ 3
   ESCCAD 05B4     26/ 7   44# 7
   ESCCGR 0617     26/10   49#18
   ESCCHA 061B     26/12   49#21
   ESCCUN 061F     26/14   49#24
   ESCEE  0624     26/19   50# 4
   ESCH   EF60      5/24   28/39   29/27   30/15   35/47   45/ 7   49/15
                  128#14
   ESCHTB 03B4     25# 3   29/ 4
   ESCLCK 05A3     26/22   42#46
   ESCRR  0624     26/20   49#31
   ESCSAD 05BE     26/ 8   45#10
   ESCSGR 0607     26/ 9   49# 3
   ESCSHA 060B     26/11   49# 7
   ESCSUN 060F     26/13   49#11
   ESCULK 05A6     26/23   43# 6
   ESCZZ  054E     26/16   49#29
   EXITI  001E      3#19   58/25
s  FALSE  0000
s  FCB    005C
   FDSK   0D84    109/60  111/18  113/12  114/11  116/15  116#18
 n FILLC  093F     72#11
   FILLZ  093D     18/53   71#28
   FMTTRK 0D49      8/60  113#26  126/20
   FORINT 0D64      8/58  110/34  111/43  114#30  117/17
   FORMAT 0E3F      8/29  125# 3
 n FSAVE  EFE4    129#31
s  FWAVM  F000     38/10   39/13   54/17
   GKEY   071F      5/12   58# 3
 n GKEYX  0739     58#24
   GTMASK 07E5     63/15   63/18   64# 8
s  H.FDC  2100     93/19   93/20
s  H.IEEE 2900
s  H.KEY  2200     65/11
s  H.SCTR 2A00     87/11   87/15
s  H.SIO  2A00
s  H.SREC 2A01     88/17
s  H.SSTS 2A00     91/14
s  H.SXMT 2A01     90/16
s  H.VIO  2C00      3/23   73/ 4   73/ 6   73/ 8   73/10   74/21   74/25
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
                     75/16    75/20    91/10
   HINT    0080       3/39     4#14
 n HKCNT   EF5D     128#11
 n HLSAX   EFD6     129#17
   HMSCRN  005B      66#12    67/37
   HOME    08F7       8/53    95/15    98/51   101#15
   HOURS   EF66     128#28   128/29   128/30   128/31   128/35   128/37
 n HSAVE   EFE7     129#34
   HSTACT  EF50      18/51    18/52   127#57
   HSTBUF  ED80     127# 8   127/12
 n HSTDSK  EF1F     127#46
 n HSTSEC  EF1C     127#44
 n HSTTRK  EF1D     127#45
 n HSTWRT  EF51     127#58
   IDAY    EF63       5/33   128#22   128/23   128/24
   IE.CO   097E       5/56     8/34    78#18
   IE.GTS  09D4       8/36    80# 7
   IE.IDM  0A85       8/40    84#10
   IE.ODM  0A43       8/39    83#13
   IE.OIM  0A20       8/38    82#11
   IE.PP   0AE1       8/41    86#10
   IE.SHK  0A4F      82/16    82/29    83/18    83#21
   IE.SI   09C1       8/35    79# 7
   IE.TC   09E0       8/37    81#13
   IESTK   EF6F       3/13     3/28    58/15   128#47
 n ILINT   0007       3# 9
 n IMONTH  EF64     128#23
   IMSG    0186       3/44    11#29
s  INTBL   EFF0       5/13     5/15
s  IRPTCT  0018      60/42
   ISTK    EF99       3/14    58/16   128#52
 n IVSAX   EFDC     129#21
 n IXSAX   EFD8     129#19
 n IYR     EF65     128#24
 n IYSAX   EFDA     129#20
 n KBDLY   EFC6     129# 2
   KBDRVR  073C      58/23    59#57
   KBSCAN  0774      60/ 7    61# 9
   KBSERV  07F8      60/47    66#14
s  KCOLM   0007      60/46    62/ 6
   KEYLCK  EF59       5/34    20/ 6    43/ 9    58/21   128# 6
   KEYLST  EFD4       5/25    60/12    61/60    92# 8
s  KLELEN  0002       5/26    60/52    62/14
s  KLLEN   0003       5/26    60/13    61/59
s  KLUSED  0007      60/20    62/ 4    62/25
s  KROWM   0038      60/46    62/ 6
   KYCDTB  08C3      66/20    69# 2
s  KYSRVD  0006      60/39    60/40
 n LDADR   EF57     128# 5
   LDSEL   EF6D       3/43   128#44   128/45
   LDTRK   EF6E      18/57   128#45
 n LEFT    008D      19#39
s  LF      000A      10/ 2    12/ 2    12/ 2    12/ 2    12/ 2    12/10    12/20
                     12/28    12/38    12/46    12/46    12/46    19/ 9    27/ 7
   LFTARW  008D      66# 8    67/29
   LIST    081C       8/12     8/13    89#18
   LKEY    EF5E       5/22    20/ 9    21/11    21/14    60/15    66/49   128#12
   LLIMIT  EF6C       5/36    37/24    42/ 9    53/16    54/32   128#40
   LOGSEC  EF56      18/52   128# 2
   LOOKUP  047C      30/20    31#22    32/20
```

Copyright ©1982 Osborne Computer Corporation

```
 n LSAVE   EFE6    129#33
s  LVMEM   1000     39/ 3
 n MCDOWN  000A     19# 9
 n MCLEFT  0008     19#1C
   MCRIGH  000C     19#11    27/ 9
   MCUP    000B     19# 8    27/10
 n MINS    EF67    128#29
   MPCHR   EF06      3/17     6/11   127#19
s  MRAM    ED80      3/4C   127/ 3
 n MSECB   0010     93# 3
s  NMIA    0066      3/47
 n NORM    038F     21#18
   NRETRY  000A     92# 3    95/11    98/23    99/13    99/46
s  NVDL    0018
   OPAD    0964      5/45    34/33    68/14    68/53    73#12   119/26
   OPBD    0971      5/51    23/15    28/31    34/18    38/ 6    39/10    54/55
                    68/42    74#27   121/13   124/ 9
   OSTR    0197      3/45    10#15    11/27
   PIAAD   EF61     34/29    68/10    68/49    74/24   119/14   128#18
   PIABD   EF62     23/12    28/24    34/14    35/23    36/ 6    37/31    38/ 3
                    39/ 7    50/20    54/22    54/45    68/32    68/55    75/19
                   118/24   121/ 9   124/ 6   128#19
s  PMCHR   003E      6/1C
s  POPALL  mac      56/15    56/37    94#24
 n PSAVE   EFE8    129#35
   PSEKC   0D76    102/17   103/19   104/15   105/14   106/15   115#28
   PSTESC  0455     28/42    29#29
s  PUSHAL  mac      56/13    56/35    94#18
 n R179X   EFC2    128#61
   RADR    0CEC      8/22    98/28   110#50
 n RAGS    EFD0    129#13
 n RDFLAG  EF21    127#49
   RDROW   07EF     61/11    61/15    62/42    63/20    65# 9    66/28    66/32
   RDRV    0B40      8/16    94#29
   RDSKD   0DFF    118/32   120#34
   RDTWRT  EFD1     92# 5   107/12   108/12   109/47   110/15
   RDWRT   0C82    107/14   108#15
   READ    0C75      8/2C    18/25    96/16   106#28
   READER  0B03      8/14    87#17    88/12
   READTR  0D33      8/59   112#19
 n REGS    EFE0    129#26
 n REPD    002D     22#15
 n REPK    0005     22#16
 n RIGHT   0088     19#37
   RKEY    037E     21# 2    21/ 9
   RLWA    0E8C    126#33   126/35   126/36
   RNDV    EF01    127#15   127/16
   ROMJP1  00DF      3/26     6#14     8/27
   ROMJP2  00E4      7# 3     8/28
   ROMRAM  EF08      3/24     5/11   127#21
   ROMSTK  EFC1      3/33   128#57
s  ROWOM   0081     61/14
   RSEC    0B5B      8/23    14/31    17/56    95#27
   RTARW   008B     66# 9    67/31
   RTRC    EF04    127#17   127/18
   RTRY    EF05     95/12    95/18    96/24    97/24    98/19    98/52    99/14
                    99/41   127#18
   RWSEC   0BB4     96/17    97/17    98#59
   SAVSEC  EF14     14/29    16/19   109/15   127#32   127/36
   SAVTRK  EF15     14/27    17/11    18/ 8    18/19   101/12   103/13   125/57
```

Copyright © 1982 Osborne Computer Corporation

```
                          127#33  127/37
   SAVTYP EFD0              3/38   16/38   92# 4   98/43  109/23  119/20  125/33
s  SCLFRE 0085             71/13
   SCREEN 04A4             33#43   35/14
   SCTRKR 08F0              9/ 3  100# 2
   SDISK  EF17              3/35   14/13   14/48   16/16  118/14  127#34  127/38
 n SEC6   EF69            128#31
 n SECS   EF68            128#30
   SEEK   0C1A              8/54   99/29  102#34
s  SEEKTM 0002              6/ 5
   SEKDEL EF13              6/ 6  102/12  102/14  102/24  103/22  104/18  105/17
                          106/18  116/12  126/ 4  126/ 6  126/11  126/13  127#30
 n SEKDSK EF1B            127#42
 n SEKSEC EF18            127#40
 n SEKTRK EF19            127#41
   SELDEN 0DD4            118/12  118#45
   SELDRV 0DA4              8/61   95/13   98/26   99/21  117#42  125/52
   SENDEN 0880              8/26   14/15   16/27   97#28   99/26   99/49
   SERFLG EFDA             88/15   91/18   91/21  127# 4
 n SERNO  0FFD            126#41
   SETXY  04CE             30/10   34#36
s  SHFTKY 0004             66/42   67/40
   SHFTTB 08FB             67/15   69#32
s  SI.MRS 0057             87/10
s  SI.RRD 0001             88/11
s  SI.S16 0055              5/60
s  SI.S64 0056
s  SI.TRD 0002             89/13
   SIRST  0AF6              5/61    8/33   87# 2
   SKEY   0371              8/ 9   19#40   21/ 8
s  SLDRCT 0003             68/20
   SLIDED 0899             67/36   68#28
   SLIDEL 0880             67/30   68# 4
   SLIDER 0884             67/32   68# 7
   SLIDEU 0895             67/34   68#25
   SLST   0813              8/25   88#20   90/11
   SPAD   094F              5/43   72#26
s  SRPTCT 0006             60/44
   SSAVE  EFEA              3/41  129#36
   STEP   0C39              8/55  103#32
   STEPIN 0C4D              8/56  104#28  126/ 8
   STEPOU 0C61              8/57  105#27
   STODIM 06E9              8/47   54#58
s  SVER   0001
s  SYS    0005
s  SYSDAT 0010
s  SYSL   0006
s  TAB    0009             69/ 3
   TEM    EF00             16/21   17/ 3   17/21   17/40   18/33  127#14  127/15
 n TEMSEC EF20            127#48
 n TKEY   EF5C            128#10
s  TOTROW 0007             61/28   61/36
s  TRUE   FFFF             89/16
 n UNACNT EF52            127#59
   UNASEC EF55             18/55  127#61
 n UNATRK EF53            127#60
   UNCUR  0719             35/10   36/ 5   37/ 3   37/13   40/ 2   41/ 2   48/ 4
                           50/14   54/11   56#43
 n UP     008A             19#36
   UPARW  008A             66#10   67/33
```

```
   UPTIM  0398     22#17    58/19
 n USER   0005      3# 5
   VALCTS 0008     27#17    31/19
   VALETS 0010     26#28    30/18
   VALIDC 03F6     26#29    27/17    31/18
   VALIDE 03C4     25#12    26/28    30/17
   VBRIGH 046B     31# 8    33/15
   VCAD   003D     19#19    26/ 6
   VCBEL  053E     27/12    37#43
   VCBKS  0512     27/ 9    37#11
   VCCLRS 054E     27/13    38#10    49/29
   VCCR   056F     27/ 7    39#15
   VCEOL  0054     19#25    26/20
   VCHOME 04FB     27/14    36# 3
   VCLF   0578     27/ 8    40# 8
   VCLRS  001A     19#12    26/15    27/12
   VCMCRT 057D     27/10    40#11
   VCMCUP 0509     27/11    36#12
   VDELC  0057     19#22    26/17
   VDELL  0052     19#24    26/19    50/25
   VECTOR 0443     29#10    32/16    32/21
   VEGH   0047     12/ 8    12/12    12/18    12/26    12/36    12/44    19#32
                   26/ 9
   VEHI   0028     19#27    26/11
   VEUL   006D     19#29    26/13
 s VFLO   FFEA      5/44    34/23    68/51
   VGRAPH 0581     25/ 5    42# 4
   VHAGR  0499     25/ 7    33#34
   VHALF  0495     25/ 6    33#28
   VHOME  001E     19#13    27/13
   VINC   0051     19#21    26/16
   VINL   0045     19#23    26/18
   VLDDR  06F1      8/45    48/13    50/48    55#16
   VLDIR  0705      8/46    39/ 6    47/10    51/16    53/38    56#23
 s VLDL   0034
 s VLL    0080      5/35    18/56    37/ 5    37/22    37/36    39/ 3    46/ 5
                   50/40    51/ 9    52/11    53/25    54/14
   VLOCK  0023     19#17    26/21
   VNORM  0467     25/ 4    30#21    33/ 9    33/20    33/31
   VOUT80 0581     31/14    33/42    42# 5    42/45
   VOUT85 0595     42/12    42#19
   VOUT90 0596     35/48    37/18    37/42    39/14    40/ 6    40/10    42/17
                   42#23    44/ 6    46/17    47/19    51/ 4
 n VOUT95 0597     42#27
   VOUT96 0598     42#29    48/18
   VOUT97 059E     27/15    29/28    38/ 9    42#36    42/43    43/10    45/ 8
                   49/16
   VRTOFF EFEF      3/36    28/27    34/13    39/12    54/52    68/58   129#41
   VSAD   0053     19#20    26/ 7
   VSGH   0067     12/ 4    12/12    12/18    12/22    12/30    12/40    19#31
                   26/ 8
   VSHI   0029     19#26    26/10
   VSUL   006C     19#28    26/12
   VUNDER 0487     25/ 8    33# 6
   VUNGR  048B     25/ 9    33#12
   VUNHA  048F     25/10    33#17
   VUNHAG 0493     25/11    33#23
   VUNLK  0022     19#18    26/22
   WBOOT  02AB      8/ 8    14#61
   WBUSY  0DE6    110/31   111/57   115/23   116/17   119#28
```

Copyright © 1982 Osborne Computer Corporation

```
  WRITE  0C7D      8/21   97/16  107#15
n WRTYPE EF23    127#51
  WSEC   0B6D      8/24   96#27
```

Copyright © 1982 Osborne Computer Corporation

Copyright ©1982 Osborne Computer Corporation

```
        *ABS    0000  EFFO
        *CODE   EFFO  0000
        *DATA   EFFO  0000
                    ;OCCBIC03.ASM

                    *NOTE*  FOR USE WITH OCCTXT6.AST ONLY

                    *       4D2007-00       MASTER  .ASM
                    *       2D2007-00       ASSY    .ASM
                    *       1D2007-00       LISTING .PRN
                    *       4D1007-00       MASTER  .COM
                    *       2D1007-00       ASSY    .COM

                    ;       +-----------------------+
                    ;       !                       !
                    ;       !       C B I O S       !
                    ;       !                       !
                    ;       +-----------------------+

                    ;       Copyright 1982, Osborne.

                    ;       This product is a copyright program product of
                    ;       Osborne and is supplied for use with the Osborne.

                    ;REV    =       1.3
                    ;DATE   =       FEB 14 1982
                    ;RWC

                    ;       Revisions:
                    ;       1.      Extensions to CBIOS added by:
                    ;               Microcode Corporation.
                    ;               Fremont, Ca
                    ;               Y. N. Sahae
                    ;               August 1981
                    ;
                    ;       2.      Programmable function keys added by:
                    ;               Roger W. Chapman
                    ;               October 1981
                    ;
                    ;       3.      Printer protocols added by:
                    ;               Roger W. Chapman
                    ;               October 1981
                    ;
                    ;       4.      Extensions added to BIOS and jump table standardized by:
                    ;               Roger W. Chapman
                    ;               October 1981
       = 0016       VERS:   =       22
```

Copyright © 1982 Osborne Computer Corporation


```
                LINK    OCCBIO13.ASM    ;Jump Table
                LINK    OCCBIO23.ASM    ;CP/M disk definitions
                LINK    OCCBIO33.ASM    ;Unit record I/O
                LINK    OCCBIO43.ASM    ;Non data transfer disk
                LINK    OCCBIO53.ASM    ;cold and warm boot
                LINK    OCCBIO63.ASM    ;Disk data transfer I/O
                LINK    OCCBIO73.ASM    ;Utility routines
                LINK    OCCBIO83.ASM    ;Utility routines
                LINK    OCCRAM2.ASM     ;Commom ram definitions

        ;       END     OCCBIO03.ASM
```

Copyright © 1982 Osborne Computer Corporation


```
                ;*****
                ;       Revisions:
                ;       YNS     18AUG81         change bios jump vector to call new routines
                ;       RWC     05OCT81         standardize cpm jump table (change calls to jumps)
                ;       RWC     16OCT81         added function key table and user defined switches
                ;
                ;*****


   = 0005       MRTRY:  =       5               ;Maximum number of retries.
   = 003C       msize:  =       60
                ;ccp:   =       (msize-8)*1024  ;location of ccp
   = CF00       CCP:    =       0CF00H
   = E500       bios:   =       ccp+1600h
   = D706       bdos:   =       ccp+806h


                        MSG     'Assemblying BIOS for LWA of ', LWAMEM,'h.'
   'Assemblying BIOS for LWA of FFFFh.'

                ;       CP/M to host disk constants

   = 0100       HSTSIZ: =       256             ;Blocking/Deblocking buffer size
   = 0010       FPYSIB: =       2048/128        ;Sectors in floppy disk block

                ;       CP/M disk control block equates which define the
                ;       disk types and maximum storage capability of each
                ;       disk type.

   = 0000       DSKS1:  =       0       ;Single density, single sided.

   = 002E       S1DSM:  =       ((40-3)*2*10)/FPYSIB

                ;       BDOS constants on entry to write

   = 0000       WRALL:  =       0               ;write to allocated
   = 0001       WRDIR:  =       1               ;write to directory
   = 0002       WRUAL:  =       2               ;write to unallocated



                ;       ROM equates.

   = 0000       ENROM:  =       0               ;Port to enable ROM
   = 0001       DIROM:  =       1               ;Port to disable ROM
```

Copyright © 1982 Osborne Computer Corporation

```
                    ;       Macro for generating Control Blocks for disk drives
                    ;       The format of these disk control blocks are as follows:
                    ;       16 bits = -> translation table.
                    ;       48 bits = Work area for CP/M.
                    ;       16 bits = -> DIRBUF.
                    ;       16 bits = -> Parameter block.
                    ;       16 bits = -> check vector.
                    ;       16 bits = -> allocation vector.

     = 0000         NDSK:   SET     0               ;Number of disk drives
     = 0000         NOFDD:  SET     0               ;Number of floppy disk drives
     = 0000         ALVSZ:  SET     0               ;Allocation vector size
     = 0000         CSVSZ:  SET     C               ;Check vector size

                            LIST    D,G,M
                    DPHGEN  MACRO   TYPE,XLATE,DIRBUF,DPBADR
                    NDSK:   SET     NDSK+1
                            DW      %2
                            DW      0,0,0
                            DW      %3
                            DW      %4
                            DW      CSV+CSVSZ
                            DW      ALV+ALVSZ
                    NOFDD:  SET     NOFDD+1
                    CSVSZ:  SET     CSVSZ+(64/4)
                    ALVSZ:  SET     ALVSZ+((SIDSM+7)/8)
                            ENDM

                    ;       Make sure Systext agrees with assembled size




                    ;       Macro for generating the Disk Parameter Blocks.
                    ;
                    ;       Disk type definition blocks for each particular mode.
                    ;       The format of these areas are as follow:
                    ;       8 bit = disk type code
                    ;       16 bit = Sectors per track
                    ;       8 bit  = Block shift
                    ;       8 bit  = BS mask
                    ;       8 bit  = Extent mask
                    ;       16 bit = Disk size/1024 - 1.
                    ;       16 bit = Directory size.
                    ;       16 bit = Allocation for directory.
                    ;       16 bit = check area size.
                    ;       16 bit = offset to first track.

                    DPBGEN  MACRO   TYPE,SPT,BSH,BSM,EXM,DSM,DIRSIZ,ALVMSK,OFFSET
                            DB      %1
                            DW      %2
                            DB      %3,%4,%5
                            DW      %6-1,%7-1,REV (%8)
                            DW      (%7+3)/4
                            DW      %9
                            ENDM
```

Copyright © 1982 Osborne Computer Corporation


```
                        ;       The following jump table defines the entry points
                        ;       into the CBIOS for use by CP/M and other external
                        ;       routines; therfore the order of these jump cannot
                        ;       be modified.  The location of these jumps can only
                        ;       be modified by 400h locations, which is a restriction
                        ;       of MCVCPM.

0000  = E500                    ORG     BIOS

E500  C30CE7                    jmp     CBOOT           ;Cold boot
E503  C334E7                    jmp     WBOOT           ;Warm boot
E506  C30BE9                    jmp     cnsta           ;Console status (input)
E509  C310E9            CONIN:  jmp     cnin            ;Console input
E50C  C315E9            CONOUT: jmp     cnout           ;Console output
E50F  C31DE9            LIST:   jmp     lst             ;List output
E512  C325E9            PUNCH:  jmp     pnch            ;Punch output
E515  C32DE9            READER: jmp     rdr             ;Reader input
E518  C39FE6                    JMP     HOME            ;Set track to zero
E51B  C380E6            RSELDK: JMP     SELDSK          ;Select disk unit
E51E  C3DDE6                    JMP     SETTRK          ;Set track
E521  C3F0E6                    JMP     SETSEC          ;Set sector
E524  C3F5E6                    JMP     SETDMA          ;Set Disk Memory Address
E527  C3D3E7            RRDK:   JMP     READ            ;Read from disk
E52A  C3F0E7            RWDK:   JMP     WRITE           ;Write onto disk
E52D  C335E9            LISTST: jmp     lstst           ;Return LST: device status
E530  C3FAE6                    JMP     SECTRN          ;Sector translation routine

                        ;       Extensions
E533  C34FE6            RRI:    JMP     ROMRI
E536  C36AE6                    JMP     ROMJMP
E539  CD63E6            FMTJ:   CALL    ROMCDE          ;Rom resident call
E53C  CD63E6            SBAUD:  CALL    ROMCDE

                        ;       IEEE-488 vectors
E53F  CD63E6            ieb1c:  CALL    ROMCDE          ;Control Out
E542  CD63E6            ieb2c:  CALL    ROMCDE          ;Status In
E545  CD63E6            ieb3c:  CALL    ROMCDE          ;Go To Standby
E548  CD63E6            ieb4c:  CALL    ROMCDE          ;Take Control
E54B  CD63E6            ieb5c:  CALL    ROMCDE          ;Output Interface Message
E54E  CD63E6            ieb6c:  CALL    ROMCDE          ;Output Device Message
E551  CD63E6            ieb7c:  CALL    ROMCDE          ;Input Device Message
E554  CD63E6            ieb8c:  CALL    ROMCDE          ;Parallel Poll
E557  CD63E6                    call    romcde          ;extensions
E55A  CD63E6                    call    romcde          ;  for
E55D  CD63E6                    call    romcde          ;     memory-mapped video
E560  C3CEEA                    jmp     acictl          ;hook for serial command port write
E563  C3D9EA                    jmp     acistat         ;hook for serial status port read
                        ;
                        ;;;;;;;;;;;;;;;;;;;;;;;
                        ;
                        ;       This area is reserved data storage area for
                        ;       the set-up program to install printer drivers,
                        ;       function keys, auto boot command, iobyte value,
                        ;       and auto horizontal scroll flag
                        ;                                                   RWC
                        ;;;;;;;;;;;;;;;;;;;;;;;
                        ;
E566  40                iobite: db      40h             ;default to serial printer=40h
                                                        ;         parallel printer=80h
                                                        ;             IEEE printer=c0h
```

Copyright ©1982 Osborne Computer Corporation


```
E567  00            prnter: db      00h            ;default to standard serial=0
                                                   ;Qume          ETX/ACK     =1
                                                   ;Diablo        XON/XOFF    =2
E568  FF            ahsenb: db      TRUE           ;auto horizontal scroll enable
E569  55            brate:  db      si.120         ;default baud rate = 120C
E56A  80            scrsze: db      128            ;default screen size = 128
                    ;
E56B  92E5          xltbl:  dw      cntrl0         ;Fixed length table
E56D  93E5                  dw      cntrl1         ;contains pointers
E56F  94E5                  dw      cntrl2         ;to strings
E571  95E5                  dw      cntrl3         ;to decode
E573  96E5                  dw      cntrl4         ;function keys
E575  97E5                  dw      cntrl5
E577  98E5                  dw      cntrl6
E579  99E5                  dw      cntrl7
E57B  9AE5                  dw      cntrl8
E57D  9BE5                  dw      cntrl9
E57F  9CE5                  dw      up
E581  9DE5                  dw      right
E583  9EE5                  dw      down
E585  9FE5                  dw      left
E587  A0E5                  dw      eotbl          ;end of table address
                    ;
E589  01            acmd:   db      1              ;auto command = 0 ignore auto boot
                                                   ;              = 1 auto on cold boot
                                                   ;              = 2 auto on warm boot
                                                   ;              = 3 auto on both
E58A  07            cauto:  db      cautol         ;length of auto command here
E58B  4155544F53            db      'AUTCST '      ;auto command goes here
      5420
      = 0007        cautol: =       *-cauto-1
                    ;
E592  30            cntrl0: db      '0'            ;Variable length table
E593  31            cntrl1: db      '1'            ;is placed here by set-up
E594  32            cntrl2: db      '2'            ;program, with xlttbl
E595  33            cntrl3: db      '3'            ;pointing to the entries
E596  34            cntrl4: db      '4'            ;
E597  35            cntrl5: db      '5'            ;Default values for the
E598  36            cntrl6: db      '6'            ;control numerics
E599  37            cntrl7: db      '7'            ;are the numbers on the keys
E59A  38            cntrl8: db      '8'
E59B  39            cntrl9: db      '9'            ;
E59C  0B            up:     db      'K'-40h        ;Default values
E59D  0C            right:  db      'L'-40h        ;for the cusor
E59E  0A            down:   db      'J'-40h        ;keys are standard
E59F  08            left:   db      'H'-40h        ;values for CP/M
      = E5A0        eotbl:  =       *
                    ;

                                                   ;space reseverd for full function
                                                   ;key decoding and 16 byte auto
                                                   ;boot command

                    ;       Sector Translation Tables.
E5A0  = E600                org     BIOS+256

E600                XLTS:   ;Translation table 2 to 1
E600  0001040508            DB      0, 1,   4, 5,   8, 9,  12,13, 16,17
      090C0D1011
E60A  020306070A            DB      2, 3,   6, 7,  10,11,  14,15, 18,19
      0B0E0F1213
```

Copyright ©1982 Osborne Computer Corporation

```
                        ;       Translation 3 to 1
                        ;       DB      0, 1,   6, 7,   12,13,  18,19
                        ;       DB      2, 3,   8, 9,   14,15
                        ;       DB      4, 5,   10,11,  16,17
                        ;       Translation 4 to 1
                        ;       DB      0, 1,   3, 9,   16,17
                        ;       DB      2, 3,   10, 11, 18,19
                        ;       DB      4, 5,   12, 13
                        ;       DB      6, 7,   14, 15
       = 0000                   IF      (*-XLTS) <> (MSEC*2)
  -                             MSG     'Translation table error',ERROR
                                ENDIF


                        ;       Control Blocks for disk drives

E614                    DPBASE:
E614                            DPHGEN  DSKS1,XLTS,DIRBUF,DPBS1+1       ;Drive A:
     += 0001            NDSK:   SET     NDSK+1
E614 +00E6                      DW      XLTS
E616 +0000000000                DW      0,0,0
      00
E61C +80EE                      DW      DIRBUF
E61E +35E6                      DW      DPBS1+1
E620 +30EF                      DW      CSV+CSVSZ
E622 +24EF                      DW      ALV+ALVSZ
     += 0001            NOFDD:  SET     NOFDD+1
     += 0010            CSVSZ:  SET     CSVSZ+(64/4)
     += 0006            ALVSZ:  SET     ALVSZ+((S1DSM+7)/8)

E624                            DPHGEN  DSKS1,XLTS,DIRBUF,DPBS1+1       ;Drive B:
     += 0002            NDSK:   SET     NDSK+1
E624 +00E6                      DW      XLTS
E626 +0000000000                DW      0,0,0
      00
E62C +80EE                      DW      DIRBUF
E62E +35E6                      DW      DPBS1+1
E630 +40EF                      DW      CSV+CSVSZ
E632 +2AEF                      DW      ALV+ALVSZ
     += 0002            NOFDD:  SET     NOFDD+1
     += 0020            CSVSZ:  SET     CSVSZ+(64/4)
     += 000C            ALVSZ:  SET     ALVSZ+((S1DSM+7)/8)


                        ;       Disk type definition blocks for each particular mode.

E634                    DPBS1:          ;Single density, single sided.

E634                            DPBGEN  DSKS1,2*MSEC,4,15,1,S1DSM,64,1000000000000000b,3
E634 +00                        DB      DSKS1
E635 +1400                      DW      2*MSEC
E637 +040F01                    DB      4,15,1
E63A +2D003F0080                DW      S1DSM-1,64-1,REV (1000000000000000B)
      00
E640 +1000                      DW      (64+3)/4
E642 +0300                      DW      3
```

Copyright © 1982 Osborne Computer Corporation

```
    = 0000      .AL     IF      ALVSZ <> ALVS
 -                      MSG     'Allocation problems ALVS<> ALVSZ, ALVSZ= ',ALVSZ
 -              .9      ERROR
                .AL     ENDIF

                ;       endx    OSBBIOS2
```

Copyright ©1982 Osborne Computer Corporation

```
                        ;*****
                        ;       Revisions:
                        ;       YNS     18AUG81         added iobyte function to listst
                        ;                               modified romcde routine
                        ;       YNS     30AUG81         moved const and listst to osbbios9.asm
                        ;       YNS     23SEP81         saved hl in ROMCDE routine when returning
                        ;                               to RAM as IEEE routines return status in
                        ;                               HL
                        ;       RWC     C9NOV81         Preserved value of HL when calling ROM
                        ;                               routines, HL used in VLDDR, VLDIR, & STODIM
                        ;                               preserved HL & DE on exit
                        ;
                        ;*****


E644                    SETRRM: ;Set ROM-RAM flag
                        ;       Entry   A= port to output (setting ROM or RAM enable)
                        ;       Exit    none.
                        ;       Uses    A
E644                            DI
E644 +F3                        DB      0F3h
E645  C5                        PUSH    BC
E646  4F                        MOV     C,A
E647  ED79          $           OT,C    A
E649  3208EF                    STO     A,ROMRAM
E64C  C1                        POP     BC
E64D                            EI ! RET
E64D +FB                        DB      0FBh
E64E +C9                 RET



E64F                    ROMRI:  ;Exit ROM resident Interrupt routine.
E64F  3A08EF                    LD      A,ROMRAM
E652  4F                        MOV     C,A             ;port
E653  ED79          $           OT,C    A               ;set ROM or RAM enabled
E655  FDE1          $           POP     IY
E657  DDE1          $           POP     IX
E659  E1                        POP     HL
E65A  D1                        POP     DE
E65B  C1                        POP     BC
E65C  F1                        POP     AF
E65D  ED7B6FEF      $           LD      SP,IESTK        ;reset to interrupt entry stk
E661                            EI ! RET
E661 +FB                        DB      0FBh
E662 +C9                 RET
```

Copyright © 1982 Osborne Computer Corporation

```
E663                    ROMCDE: ;Call ROM resident processor
                        ;       Entry   DE = resident processor to call biased
                        ;               by CBICS jump vector.
                        ;       NOTE: ROM jump vector must match CBICS vector
                        ;
                        ;       Entry at ROMCD1 with low digit of CBIOS vector in reg E

E663  D1                        POP     DE              ;Get calling address
E664  7B                        MOV     A,E
E665  D603                      SUI     3
E667  5F                        MOV     E,A
E668                    ROMCD1:
E668  1601                      LDK     D,high (ROMVEC)

E66A                    ROMJMP: ;Entry here to jump to ROM function code directly
                        ;       Entry   DE = ROM jump address
                        ;               BC, HL, IX = any parameters

E66A  229DE6                    sto     hl,stohl
E66D                            DI
E66D +F3                        DB      0F3h
E66E  210000                    LDK     HL,0
E671  39                        ADD     HL,SP           ;Old stack to HL
E672  31C1EF                    LDK     SP,BIOSTK
E675  E5                        PUSH    HL              ;Save old stack pointer
E676  2A9DE6                    ld      hl,stohl        ;get user hl
E679                            ENAROM
E679 +                          DI
E679 +F3                        DB      0F3h
E67A +D300                      OUT     0
E67C +3E00                      LDK     A,0
E67E +3208EF                    STO     A,ROMRAM
E681 +                          EI
E681 +FB                        DB      0FBh
E682  D5                        push    de              ;ROM jump address to IY
E683  FDE1          $           pop     IY
E685  CD9AE6                    CALL    GOROM
E688                            DI
E688 +F3                        DB      0F3h
E689  F5                        PUSH    AF              ;save status returned
E68A                            DISROM
E68A +                          DI
E68A +F3                        DB      0F3h
E68B +D301                      OUT     1
E68D +3E01                      LDK     A,1
E68F +3208EF                    STO     A,ROMRAM
E692 +                          EI
E692 +FB                        DB      0FBh
E693  F1                        POP     AF
E694  FDE1          $           pop     iy              ;Restore old stack pointer
E696  FDF9          $           MOV     SP,iy
E698                            EI ! RET
E698 +FB                        DB      0FBh
E699 +C9                  RET

E69A                     GOROM: EI
E69A +FB                        DB      0FBh
E69B  FDE9          $           JMP     [IY]

E69D  = 0002             STOHL: DS      2
```

Copyright © 1982 Osborne Computer Corporation


```
                        ;       H O M E
                        ;
                        ;       Return disk to home.  This routine sets the track number
                        ;       to zero.  The current host disk buffer is flushed to the
                        ;       disk.

E69F                    HOME:
E69F  CDAAE8                    CALL    FLUSH           ;Flush host buffer
E6A2  AF                        XRA     A
E6A3  3250EF                    STO     A,HSTACT                ;Clear host active flag
E6A6  3252EF                    STO     A,UNACNT                ;Clear sector count
E6A9  3219EF                    STO     A,SEKTRK
E6AC  321AEF                    STO     A,SEKTRK+1
E6AF  C9                        RET



E6B0                    SELDSK: ;Select disk drive for next transfer.
                        ;
                        ;       ENTRY   C  = disk selection value (0..15).
                        ;               DE and 1 = 0, first call for this disk.
                        ;
                        ;       EXIT    HL = 0, if drive not selectable.
                        ;               HL = DPH address if drive is selected.

E6B0  79                        MOV     A,C
E6B1  FE02                      CPI     NDSK
E6B3  D2D1E6                    JNC     SELD1           ;If invalid drive number
E6B6  321BEF                    STO     A,SEKDSK
E6B9  69                        MOV     L,C
E6BA  2600                      MVI     H,0
E6BC  29                        ADD     HL,HL           ;*2
E6BD  29                        ADD     HL,HL           ;*4
E6BE  29                        ADD     HL,HL           ;*8
E6BF  29                        ADD     HL,HL           ;*16
E6C0  43                        MOV     B,E             ;save initial bit
E6C1  1114E6                    LDK     DE,DPBASE
E6C4  19                        ADD     HL,DE           ;HL = DPH address
E6C5  E5                        PUSH    HL
E6C6  111301                    LDK     DE,100h + (low RSELDK)
E6C9  78                        MOV     A,B
E6CA  E601                      AND     1
E6CC  CC6AE6                    CZ      ROMJMP          ;Select the disk in reg C
E6CF  E1                        POP     HL
E6D0  C9                        RET

E6D1  210000            SELD1:  LDK     HL,0
E6D4  3A0400                    LDA     CDISK
E6D7  91                        SUB     C
E6D8  C0                        RNZ                     ;If default drive not in error
E6D9  320400                    STO     A,CDISK
E6DC  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;       S E T   T R A C K.
                        ;
                        ;       Set track number.  The track number is saved for later
                        ;       use during a disk transfer operation.
                        ;
                        ;       ENTRY   BC = track number.

E6DD                    SETTRK:
E6DD  ED4319EF   $              STD     BC,SEKTRK       ;Set track
E6E1  2A53EF                    LHLD    UNATRK
E6E4  7D                        MOV     A,L
E6E5  A9                        XRA     C
E6E6  4F                        MOV     C,A
E6E7  7C                        MOV     A,H
E6E8  A8                        XRA     B
E6E9  B1                        ORA     C
E6EA  C8                        RZ                      ;If same track
                        ;       JMP     CUNACT


                        ;       Clear Unallocated block count (force pre-reads).

E6EB  AF                CUNACT: XRA     A               ;A = 0
E6EC  3252EF                    STD     A,UNACNT                ;Clear unallocated block count
E6EF  C9                        RET


                        ;       Set the sector for later use in the disk transfer.  No
                        ;       actual disk operations are perfomed.
                        ;
                        ;       Entry   BC = sector number.

E6F0  79                SETSEC: MOV     A,C
E6F1  3220EF                    STD     A,TEMSEC        ;sector to seek
E6F4  C9                        RET


                        ;       Set Disk memory address for subsequent disk read or
                        ;       write routines.  This address is saved in DMAADR until
                        ;       the disk transfer is performed.
                        ;
                        ;       ENTRY   BC = Disk memory address.
                        ;
                        ;       EXIT    DMAADR = BC.

E6F5                    SETDMA:
E6F5  ED4311EF   $              STD     BC,DMAADR
E6F9  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation


```
                        ;       Translate sector number from logical to physical.
                        ;
                        ;       ENTRY   DE = 0, no translation required.
                        ;               DE = translation table address.
                        ;               BC = sector number to translate.
                        ;
                        ;       EXIT    HL = translated sector.

E6FA                    SECTRN:
E6FA  3A55EF                    LDA     UNASEC
E6FD  B9                        CMP     C
E6FE  C4EBE6                    CNZ     CUNACT          ;If sectors do not match
E701  79                        MOV     A,C
E702  3256EF                    STA     A,LOGSEC
E705  69                        MOV     L,C
E706  60                        MOV     H,B
E707  19                        ADD     HL,DE
E708  6E                        MOV     L,M
E709  2600                      MVI     H,0
E70B  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
                         ;*****
                         ;       Revisions:
                         ;       RWC     20OCT81         warm boot makes dir 2 files across
                         ;       RWC     19OCT81         included baud rate and screen size initalization
                         ;       RWC     14OCT81         change iobyte to variable
                         ;                               modified auto start feature
                         ;       YNS     20AUG81         change iobyte defaults
                         ;
                         ;*****

                                 ;       B o o t   C P / M   f r o m   d i s k.
                         ;
                         ;       The CBOOT entry point gets control from the cold start
                         ;       loader and is responsible for the basic system initial-
                         ;       ization.  This includes outputting a sign-on message and
                         ;       initializing the following page zero locations:
                         ;
                         ;          0,1,2: Set to the warmstart jump vector.
                         ;              3: Set to the initial IOBYTE value.
                         ;              4: Default and logged on drive.
                         ;          5,6,7: Set to a jump to BDOS.
                         ;          40-41: Points to where Date and Time are kept
                         ;
                         ;       Register C must contain the selected drive, which is
                         ;       zero to select the A drive.  The exit address is to
                         ;       the CCP routine.
                         ;
                         ;
                         ;       The WBOOT entry point gets control when a warm start
                         ;       occurs, a ^C from the console, a jump to BDOS (function
                         ;       0), or a jump to location zero.  The WBOOT routine reads
                         ;       the CCP and BDOS from the appropriate disk sectors.
                         ;       WBOOT must also re-initialize locations 0,1,2 and 5,6,7.
                         ;       The WBOOT routines exits with the C register set to the
                         ;       appropriate drive selection value.  The exit address
                         ;       is to the CCP routine.
                         ;
E70C                     CBOOT:  ;Entry  A= drive to boot off of.
E70C  47                         MOV     B,A             ;save requested drive
E70D                             DISROM
E70D +                           DI
E70D +F3                         DB      0F3h
E70E +D301                       OUT     1
E710 +3E01                       LDK     A,1
E712 +3208EF                     STO     A,ROMRAM
E715 +                           EI
E715 +FB                         DB      0FBh
E716  78                         MOV     A,B             ;restore requested drive
E717  320400                     STO     A,CDISK         ;force requested drive
E71A  3100CF                     LDK     SP,CCP

                          ;**********************

E71D  3A66E5                     ld      a,iobite        ;get iobyte value
E720  320300                     STO     A,IOBYTE        ;Set I/O byte to default
E723  3A69E5                     ld      a,brate
E726  4F                         mov     c,a
E727  CD3CE5                     call    sbaud           ;set baud rate
E72A  3A6AE5                     lda     scrsze
```

Copyright © 1982 Osborne Computer Corporation

```
E72D  326CEF                    sta     llimit          ;set screen size
E730  3E01                      ldk     a,1
E732  1814 ^E748$               JR      BCCP            ;Do CP/M

E734                    WBOOT:                          ;Warm boot
E734  3100CF                    LDK     SP,CCP
E737  CD9FE6                    CALL    HOME            ;flush any buffer

E73A  110301            BCPM:   LDK     DE,ROMVEC+3*1   ;Set ROM vector address
E73D  0100CF                    LDK     BC,CCP
E740  CD6AE6                    CALL    ROMJMP
E743  B7                        ORA     A
E744  20F4 ^E73A$               JRNZ    BCPM            ;If error in read
E746  3E02                      mvi     a,2             ;indicate warm boot

E748                    BCCP:   ;Entry  A = 01, if cold boot
                        ;               A = 02, if warm boot
E748  F5                        PUSH    AF              ;Save flags
E749  018000                    LDK     BC,DBUF         ;Set default data transfer address
E74C  CDF5E6                    CALL    SETDMA
E74F  2180ED                    LDK     HL,HSTBUF
E752  220FEF                    STO     HL,DMADR        ;set ROM DMA address

                        ;       Clear console control ESC cell
E755  AF                        XRA     A
E756  3260EF                    STO     A,ESCH          ;clear ESC
E759  3C                        inc     a               ;a=1  make directory list two files across
E75A  32B2D3                    sto     a,ccp+04b2h     ;change to ccp

                        ;       Set-up low core pointer cells
E75D  3EC3                      LDK     A,0C3h          ;Store jumps in low memory
E75F  320000                    STO     A,0
E762  320500                    STO     A,5
E765  2103E5                    LDK     HL,BIOS+3
E768  220100                    STO     HL,1
E76B  2106D7                    LDK     HL,BDOS
E76E  220600                    STO     HL,6
E771  2163EF                    LDK     HL,IDAY
E774  224000                    STO     HL,TIMPTR       ;set date-time pointer
E777  218AE5                    LDK     HL,CAUTO

                        ;Digital Research has informed SORCIM that CP/M 2.2
                        ;CANNOT be booted off B:.
                        ;Manipulation of CDISK is kept here to avoid the bugs
                        ;which would appear with its disappearance.
                        ;Booting off B: is accomplished with the RAM cell DSKSWP.
E77A  C1                        pop     bc              ;cold/warm indicator in b
E77B  3A89E5                    ld      a,acmd
E77E  A0                        and     b
E77F  283B ^E7BC$               jrz     done
E781  7E                        ld      a,[hl]
E782  B7                        ORA     A
E783  2837 ^E7BC$               JRZ     DONE
E785  1107CF                    LDK     DE,CCP+7
E788  0600                      LDK     B,0
E78A  4F                        MOV     C,A
E78B  EDB0        $             LDIR                    ;Move command line to buffer
E78D  110000                    ldk     de,0
E790  1838 ^E7CA$               jr      done1
```

Copyright © 1982 Osborne Computer Corporation


```
E792                    SIGNON:
E792  1A                        DB      'Z'-40h
E793  4F73626F72                DB      'Osborne Computer System'
      6E6520436F
      6D70757465
      7220537973
      74656D
E7AA  0D0A3630                  DB      CR,LF,MSIZE/10+'0',MSIZE mod 10 + '0'
E7AE  4B2043502F                DB      'K CP/M ',VERS/10+'0','.',VERS mod 10 + '0'
      4D20322E32
E7B8  410D0A00                  DB      CBIOSV+'a',CR,LF,0

E7BC                    done:
E7BC  3E02                      mvi     a,2
E7BE  B8                        cmp     b
E7BF  2806 ^E7C7$               jrz     done0
E7C1  2192E7                    ldk     hl,signon
E7C4  CDFDE8                    call    print
E7C7                    done0:
E7C7  110300                    ldk     de,3
E7CA                    done1:
E7CA  2100CF                    ldk     hl,ccp
E7CD  19                        add     hl,de
E7CE  3A0400                    ld      a,cdisk
E7D1  4F                        mov     c,a
E7D2  E9                        jmp     [hl]

                        ;       Endx    OSBBIOS5.asm
```

Copyright © 1982 Osborne Computer Corporation

```
E7D3                   READ:   ;a CP/M 128 byte sector.
                       ;
                       ;       EXIT    A = 0, successful read operation.
                       ;               A = 1, unsucessful read operation.
                       ;               Z bit = 1, successful read operation.
                       ;               Z bit = 0, unsuccessful read operation.

E7D3  CDF2E8                   CALL    MVINFO          ;Move information for transfer
E7D6  AF                       XRA     A               ;Set flag to force a read
E7D7  3252EF                   STO     A,UNACNT                ;Clear sector counter
E7DA  CD4BE8                   CALL    FILL            ;Fill buffer with data
E7DD  E1                       POP     HL
E7DE  D1                       POP     DE
E7DF  018000                   LDK     BC,128          ;Move 128 bytes
E7E2  EDB0        $            LDIR
E7E4  3A22EF                   LD      A,ERFLAG
E7E7  B7                       ORA     A
E7E8  C8                       RZ                      ;If no error
E7E9  AF                       XRA     A
E7EA  3250EF                   STO     A,HSTACT        ;Clear host active
E7ED  F601                     ORI     001h            ;Set error flag
E7EF  C9                       RET


E7F0                   WRITE:  ;the selected 128 byte CP/M sector.
                       ;
                       ;       ENTRY   C = 0, write to a previously allocated block.
                       ;               C = 1, write to the directory.
                       ;               C = 2, write to the first sector of unallocated
                       ;               data block.
                       ;
                       ;       EXIT    A = 0, write was successful.
                       ;               A = 1, write was unsucessful.
                       ;               Z bit = 1, write was successful.
                       ;               Z bit = 0, write was unsucessful.

E7F0  CDF2E8                   CALL    MVINFO          ;Move information for transfer
E7F3  79                       MOV     A,C             ;Write type in c
E7F4  3223EF                   STO     A,WRTYPE
E7F7  FE02                     CPI     WRUAL
E7F9  2011 ^E80C$              JRNZ    WRIT2           ;If write to allocated
E7FB  3E10                     LDK     A,2048/128
E7FD  3252EF                   STO     A,UNACNT
E800  2A19EF                   LD      HL,SEKTRK
E803  2253EF                   STO     HL,UNATRK       ;UNATRK = SEKTRK
E806  3A56EF                   LD      A,LOGSEC
E809  3C                       INC     A
E80A  1819 ^E825$              JR      WRIT3

E80C  2152EF           WRIT2:  LDK     HL,UNACNT
E80F  7E                       LD      A,[hl]
E810  B7                       ORA     A
E811  CA2AE8                   JZ      WRIT4           ;If no unallocated records
E814  35                       DEC     [hl]            ;dec unalloc record count
E815  3A55EF                   LD      A,UNASEC        ;Increment logical sector
E818  3C                       INC     A
E819  FE14                     CPI     2*10
E81B  2008 ^E825$              JRNZ    WRIT3           ;If not end of track
```

```
E81D  2A53EF                LD      HL,UNATRK
E820  23                    INC     HL
E821  2253EF                STO     HL,UNATRK
E824  AF                    XRA     A

E825  3255EF       WRIT3:   STO     A,UNASEC
E828  3EFF                  LDK     A,0FFh

E82A  CD4BE8       WRIT4:   CALL    FILL
E82D  D1                    POP     DE
E82E  E1                    POP     HL
E82F  018000                LDK     BC,128
E832  EDB0       $          LDIR
E834  3E01                  LDK     A,1
E836  3251EF                STO     A,HSTWRT        ;HSTWRT = 1
E839  3A22EF                LD      A,ERFLAG
E83C  B7                    ORA     A
E83D  C0                    RNZ                     ;If any errors occurred

E83E  3A23EF                LD      A,WRTYPE        ;write type
E841  FE01                  CPI     WRDIR           ;to directory?
E843  CCAAE8                CZ      FLUSH           ;Force write of directory
E846  3A22EF                LD      A,ERFLAG
E849  B7                    ORA     A
E84A  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
                      ;       FILL - fill host buffer with approprite host sector.
                      ;
                      ;       ENTRY   A = 0, Read required if not in buffer.
                      ;               Otherwise read not required.
                      ;
                      ;       EXIT    On exit the stack will contain the following
                      ;               values:
                      ;               POP     x         ;x = host record address.
                      ;               POP     y         ;y = caller's buffer address.

E84B  3221EF          FILL:   STO     A,RDFLAG          ;Save read flag
E84E  1180ED                  LDK     DE,HSTBUF         ;initial offset
E851  218000                  LDK     HL,128            ;128 byte records
E854  3A18EF                  LD      A,SEKSEC          ;Get logical sector
E857  EB                      EX      DE,HL
E858  0F                      RRC
E859  3001 ^E85C$             JRNC    FILL3             ;If low bit not set
E85B  19                      ADD     HL,DE             ;Add bias to offset
E85C  EB              FILL3:  EX      DE,HL
E85D  29                      ADD     HL,HL
E85E  E67F                    ANI     07Fh              ;Mask sector
E860  3218EF                  STO     A,SEKSEC
E863  2A11EF                  LD      HL,DMAADR
E866  E3                      XTHL                      ;Set return parameters
E867  D5                      PUSH    DE
E868  E5                      PUSH    HL                ;Set return address

E869  2150EF                  LDK     HL,HSTACT         ;host active flag
E86C  7E                      LD      A,[hl]
E86D  3601                    STO     1,[hl]            ;always becomes 1
E86F  B7                      ORA     A
E870  2814 ^E886$             JRZ     FILL6             ;If host buffer inactive
E872  211CEF                  LDK     HL,HSTSEC
E875  1118EF                  LDK     DE,SEKSEC
E878  0604                    LDK     B,SEKDSK-SEKSEC+1
E87A  1A              FILL4:  LD      A,[de]
E87B  BE                      CMP     [hl]
E87C  2005 ^E883$             JRNZ    FILL5             ;If mis-match
E87E  23                      INC     HL
E87F  13                      INC     DE
E880  10F8 ^E87A$             DJNZ    FILL4             ;If all bytes not checked
E882  C9                      RET

E883  CDAAE8          FILL5:  CALL    FLUSH             ;Flush host buffer

E886  3A1BEF          FILL6:  LD      A,SEKDSK          ;Move disk and type
E889  321FEF                  STO     A,HSTDSK
E88C  3217EF                  STO     A,ACTDSK
E88F  2A19EF                  LD      HL,SEKTRK
E892  221DEF                  STO     HL,HSTTRK
E895  2215EF                  STO     HL,ACTTRK
E898  3A18EF                  LD      A,SEKSEC
E89B  321CEF                  STO     A,HSTSEC
E89E  3214EF                  STO     A,ACTSEC
E8A1  3A21EF                  LD      A,RDFLAG
E8A4  B7                      ORA     A
E8A5  C0                      RNZ                       ;If no read required

E8A6  3E00                    LDK     A,0               ;Read
E8A8  181C ^E8C6$             JR      FINAL
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                    ;       FLUSH - Write out active host buffer onto disk.

E8AA                FLUSH:
E8AA  2151EF                LDK     HL,HSTWRT
E8AD  7E                    LD      A,[hl]
E8AE  B7                    ORA     A
E8AF  C8                    RZ                      ;If host buffer already on disk
E8B0  3600                  STO     0,[hl]
E8B2  3A1FEF                LD      A,HSTDSK        ;Move disk and type
E8B5  3217EF                STO     A,ACTDSK
E8B8  2A1DEF                LD      HL,HSTTRK
E8BB  2215EF                STO     HL,ACTTRK
E8BE  3A1CEF                LD      A,HSTSEC
E8C1  3214EF                STO     A,ACTSEC
E8C4  3E03                  LDK     A,3             ;Write flag
                    ;       JMP     FINAL



E8C6                FINAL:          ;Preform final transfer processing.
                    ;
                    ;       ENTRY   A = 0 .. read disk.
                    ;                 = 3 .. write disk.
                    ;       Calls: Rom resident routine to read/write ONE
                    ;               sector only.

E8C6  5F                    MOV     E,A
E8C7  1600                  LDK     D,0
E8C9  212701                LDK     HL,ROMVEC+3*13
E8CC  19                    ADD     HL,DE
E8CD  2209E9                STO     HL,SAVADR
E8D0  2114EF                LDK     HL,ACTSEC
E8D3  34                    INC     [hl]            ;update sector+1
E8D4  3E05                  LDK     A,MRTRY         ;Set retry count
E8D6  3205EF        FNL1:   STO     A,RTRY          ;Clear retry count
E8D9  2A09E9                LD      HL,SAVADR
E8DC  EB                    EX      DE,HL
E8DD  0601                  LDK     B,1             ;indicate one sector xfer
E8DF  CD6AE6                CALL    ROMJMP          ;process read or write
E8E2  3222EF                STO     A,ERFLAG        ;set possible error flag
E8E5  C8                    RZ                      ;If no errors

E8E6  3A05EF                LD      A,RTRY          ;Get retry counter
E8E9  3D                    DEC     A
E8EA  20EA ^E8D6            JRNZ    FNL1            ;If not permanent error
E8EC  F601                  ORI     01h
E8EE  3222EF                STO     A,ERFLAG        ;Set error flag
E8F1  C9                    RET

                    ;       endx    OSBBIOS6.asm
```

```
                        ;       MVINFO  Move information necessary for transfer.

E8F2                    MVINFO:
E8F2    AF                      XRA     A
E8F3    3222EF                  STO     A,ERFLAG                ;Clear error flag
E8F6    3A20EF                  LD      A,TEMSEC
E8F9    3218EF                  STO     A,SEKSEC
E8FC    C9                      RET


                        ;       Print message terminated by zero byte.
                        ;
                        ;       ENTRY   HL -> message buffer, terminated by zero.
                        ;
                        ;       EXIT    HL -> zero byte + 1.
                        ;               A = 0.
                        ;               Z bit set.
                        ;
                        ;       Destroys only HL, Flags, and A registers.

E8FD    7E              PRINT:  LD      A,[hl]          ;Get a character
E8FE    B7                      ORA     A
E8FF    23                      INC     HL
E900    C8                      RZ                      ;If zero the terminate
E901    E5                      PUSH    HL
E902    4F                      MOV     C,A
E903    CD0CE5                  CALL    CONOUT          ;Output to the console
E906    E1                      POP     HL
E907    18F4 ^E8FD$             JR      PRINT

E909    = 0002          SAVADR: DS      2               ;Disk transfer routine vector
```

Copyright ©1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        ;*****
                        ;
                        ;       Revisions:
                        ;       YNS     20AUG81         Initial release
                        ;       YNS     29AUG81         Expanded trans. table to include ctl/numerics
                        ;       YNS     01SEP81         Added parallel port support
                        ;       YNS     05SEP81         Invert data to/from parallel port
                        ;       YNS     06SEP81         Return character in C as well as A from
                        ;                               all input routines.
                        ;       YNS     08SEP81         Fixed "some" ieee bugs
                        ;       YNS     20SEP81         Recode ieee drivers to send untalk and
                        ;                               unlisten commands after each char xfr.
                        ;       YNS     29SEP81         Add auto horiz. scroll
                        ;       RWC     14OCT81         Reassignment in dispatch table
                        ;                               Redefinition of iobyte assignments
                        ;                               Printer protocols added to serial list device
                        ;                               Function key decoding added
                        ;       RWC     20OCT81         Horizontal scroll toggle added
                        ;       RWC     24OCT81         Horizontal scroll modified
                        ;
                        ;*****


                        ;
                        ; the following routines will use the IOBYTE to transfer
                        ; control to the appropriate device driver
                        ;
                        ;--------------
                        ; return console status
                        ;
E90B                    CNSTA:  proc
E90B  2179E9                    ldk     hl,ptr+cstat    ; status table
E90E  1808 ^E918$               jr      godispch        ; call appropriate rtn
                        ;------------
                        ; read input character from device
                        ;
E910                    CNIN:   proc
E910  2181E9                    ldk     hl,ptr+cinp     ; table of input rtns
E913  1803 ^E918$               jr      godispch
                        ;-------------
                        ; put output character to device
                        ; c contains output character
                        ;
E915                    CNOUT:  proc
E915  2189E9                    ldk     hl,ptr+cout     ; table of output rtns
E918                    godispch:
E918  0601                      mvi     b,1             ;number of shifts required to align
                                                        ;CONSOLE field
E91A  C33DE9                    jmp     dispch
                        ;---------------
                        ; list device character output
                        ;
E91D                    LST:    proc
E91D  0604                      mvi     b,4
E91F  2199E9                    ldk     hl,ptr+list     ;table of list routines
E922  C33DE9                    jmp     dispch
                        ;---------------
                        ; output to punch
                        ;
E925                    PNCH:   proc
```

Copyright © 1982 Osborne Computer Corporation

```
E925  0606                      mvi     b,6
E927  2189E9                    ldk     hl,ptr+pnch      ; punch routines
E92A  C33DE9                    jmp     dispch
                        ;----------------
                        ; reader inout
                        ;
E92D                    RDR:    proc
E92D  0608                      mvi     b,8
E92F  2181E9                    ldk     hl,ptr+rdr       ; reader routines
E932  C33DE9                    jmp     dispch



                        ;       L i s t   S t a t u s.
                        ;
                        ;       Return the ready status for the list device.
                        ;
                        ;       EXIT    A = 0 (zero), list device is not ready to
                        ;               accept another character.
                        ;               A = FFh (255), list device is ready to accept
                        ;               a character.
                        ;       The list status is returned depending upon the iobyte fields
                        ;       LIST field (bits 6,7)
                        ;               =0:     status of crt. (always ready)
                        ;               =1:     status of serial printer
                        ;               =2:     status of parallel printer
                        ;               =3:     status of ieee port (always ready)
                        ;
E935                    LSTST:  proc
E935  0604                      mvi     b,4              ;number of left shifts thru carry
                                                         ;to align LIST field of IOBYTE
E937  2191E9                    ldk     hl,ptr+lst       ;list status routines
E93A  C33DE9                    jmp     dispch
```

Copyright © 1982 Osborne Computer Corporation

```
E93D                  dispcn: proc
                      ;       on entry here reg B contains the left shift count
                      ;       required to align the iobyte field to bit 1 position.
                      ;       and reg HL contains address of select table
                      ;
                      ;       A special stack is used to avoid overflows (eg when
                      ;       called by WordStar)

E93D  229DE6                  sto     hl,stohl        ;Save hl
E940  210000                  ldk     hl,0
E943  39                      add     hl,sp           ;Old stack to hl
E944  3179E9                  ldk     sp,disstk
E947  E5                      push    hl              ;Save old stack
E948  2A9DE6                  ld      hl,stohl        ;Restore hl

E94B  3A0300                  lda     iobyte
E94E                  dspchl:
E94E  17                      ral
E94F  10FD ^E94E$             djnz    dspchl
E951  E606                    ani     6                ;get select field*2
E953  5F                      mov     e,a
E954  1600                    ldk     d,0              ; de = iobyte field * 2
E956  19                      dad     de
E957  5E                      mov     e,m
E958  23                      inx     hl
E959  66                      mov     h,m              ; get the routine address
E95A  6B                      mov     l,e              ; into hl for xchange with pc

E95B  1160E9                  ldk     de,dspret
E95E  D5                      push    de               ;push return address
E95F  E9                      pchl                     ;jump to routine

E960  FDE1          $ dspret: pop     iy
E962  FDF9          $         mov     sp,iy            ;Restore old stack
E964  C9                      ret


E965  = 0014                  ds      10*2
E979  = 0000          disstk: ds      0

                      ;
                      ; addresses of routines
                      ;
E979                  ptr+cstat:
E979  A1E9                    dw      const            ; keyboard status
E97B  6BEA                    dw      sistat           ; serial port input status
E97D  D4EB                    dw      pistat           ; parallel input status
E97F  F8EA                    dw      ieinstat         ; status of input device on IEEE port
E981                  ptr+rdr:
E981                  ptr+cinp:
E981  32E9                    dw      keyinp           ; get input from keyboard
E983  74EA                    dw      spinp            ; serial port input
E985  F0EB                    dw      parinp           ; parallel input
E987  2FEB                    dw      ieinp            ; ieee port input
E989                  ptr+pnch:
E989                  ptr+cout:
E989  F5E9                    dw      crtout           ; output character to crt
E98B  87EA                    dw      spout            ; serial port output
E98D  04EC                    dw      parout           ; parallel output
E98F  33EB                    dw      ieout            ; ieee port output
```

```
E991                 ptr←lst:
E991   F3E9                   dw      crstat
E993   62EA                   dw      sostat          ; serial output status
E995   C1EB                   dw      postat          ; parallel output status
E997   F3E9                   dw      ieostat         ; ieee output status
E999                 ptr←list:
E999   F6E9                   dw      crtout
E99B   98EA                   dw      prtout
E99D   04EC                   dw      parout
E99F   3BEB                   dw      ieout
                     ;
```

Copyright ©1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        ;       C O N S O L   S T A T U S
                        ;
                        ;       This routine samples the Console status and returns the
                        ;       following values in the A register.
                        ;
                        ;       EXIT    A = 0 (zero), means no character
                        ;               currently ready to read.
                        ;
                        ;               A = FFh (255), means character
                        ;               currently ready to read.

      = 0000            nokey   equ     00h             ;indicates key not present

E9A1                    CONST:  proc
                        ;       check if any translated keys are pending
E9A1  3AC2E9                    ld      a,count
E9A4  B7                        ora     a
E9A5  2C06 ^E9AD$               jrnz    const5
                        ;       if no xlated keys pending, check keyhit flag

E9A7  3A5EEF                    LD      A,LKEY          ;Get Key hit flag
E9AA  EE00                      xri     nokey
E9AC  C8                        RZ                      ;If data not available
E9AD                    const5:
E9AD  F6FF                      ORI     0FFh
E9AF  C9                        RET
```

Copyright ©1982 Osborne Computer Corporation

```
                        ;**
                        ; Routine name: KEYINP - gets keystroke from rom kbd driver.  Translates
                        ;                        the codes 80h to 8fh as per table.
                        ;
                        ; Outputs: A = translated code in ASCII
                        ; All registers are destroyed
                        ;

       = 0080           basvl0  equ     80h                     ;lowest value of translatable keys
E9B0   = 0002           xltkey  ds      2
                        ;
E9B2                    KEYINP: proc
                        ;       if there are no xlated keys waiting then
                        ;       call the keyboard driver in rom

E9B2   CD0FEA                   call    ahscrl
E9B5   21C2E9                   ldk     hl,count
E9B8   7E                       ld      a,[hl]                  ;get number of xlated keys
E9B9   B7                       ora     a
E9BA   2809 ^E9C5$              jrz     kinl0                   ;if keys pending then
E9BC   DD2AB0E9  $              ld      ix,xltkey               ;get base address in IX reg
E9C0   DD7E                     db      0DDh,7Eh                ;simulate LD  A,(IX+COUNT)
E9C2   00               count:  db      0                       ;to get next key from table
E9C3   34                       inc     [hl]                    ;increment count
E9C4   C9                       ret

E9C5                    kinl0:
E9C5   1E09                     ldk     e,low(conin)
E9C7   CD68E6                   call    romcdl

                        ;       When console has returned this code will check
                        ;       for function key and preform some translation

E9CA   FE80                     cpi     80h             ;function keys have value
E9CC   D8                       rc                      ;80h-8dh
E9CD   FE8E                     cpi     8En             ;do a shift to make pointer
E9CF   D0                       rnc                     ;into table and return if not function key
E9D0   CB27      $              sla     a               ;
E9D2   5F                       mov     e,a             ;
E9D3   1600                     ldk     d,0             ;
E9D5   DD216BE5  $              ldk     ix,xltbl        ;
E9D9   DD19      $              add     ix,de           ;
E9DB   DD6E00    $              ld      l,[ix+0]
E9DE   DD6601    $              ld      h,[ix+1]
E9E1   DD5E02    $              ld      e,[ix+2]
E9E4   DD5603    $              ld      d,[ix+3]
E9E7   ED53B0E9  $              sto     de,xltkey
E9EB   ED52      $              sbc     hl,de
E9ED   7D                       mov     a,l
E9EE   32C2E9                   sto     a,count         ;
E9F1   18BF ^E9B2$              jr      keyinp          ;
                        ;
                        ;
```

Copyright ©1982 Osborne Computer Corporation

```
                        ;**
                        ; routine:       CRSTAT
                        ;                returns status of crt.
                        ;                crt is always ready
                        ;                IEOSTAT returns status of ieee
                        ;                ieee always appears to be ready

E9F3                    ieostat:
E9F3                    CRSTAT: proc
E9F3  F6FF                      ori     0FFh
E9F5  C9                        ret


                        ;**
                        ; routine name: CRTOUT
                        ;
                        ; inputs: C: output character

      = 0008            EF←ESC: =       8               ;escape flag bit definitions
      = 0001            EF←GR:  =       1
                        ;
E9F6                    CRTOUT: proc
E9F6  3A60EF                    lda     esch
E9F9  E609                      and     ef←esc+ef←gr
                        ;       push    af
E9FB  200D ^EA0A$               jrnz    crt10
E9FD  79                        mov     a,c
E9FE  FE14                      cpi     14h
EA00  2008 ^EA0A$               jrnz    crt10
EA02  3A68E5                    ld      a,ahsenb
EA05  2F                        cma
EA06  3268E5                    sto     a,ahsenb
                        ;       pop     af
EA09  C9                        ret
EA0A                    crt10:
                        ;       call    outch
                        ;       pop     af
                        ;       rnz
                        ;       mvi     a,cr
                        ;       cmp     c
                        ;       cz      ahscrl
                        ;       ret


                        ;       sbrt: outch - calls rom cout routine
EA0A                    outch:
EA0A  1E0C                      ldk     e,low(conout)     ;output to crt
EA0C  C368E6                    jmp     romcdl

                        ;       sbrt: ahscrl - does auto horizontal scroll if required.
EA0F                    ahscrl: proc
EA0F  3A68E5                    ld      a,ahsenb
EA12  B7                        or      a
EA13  C8                        rz

EA14  2A5AEF                    ld      hl,curs         ;get cursor
EA17  29                        add     hl,hl

                        ;       check for cursor in home window
EA18  3E64                      ldk     a,100
```

Copyright © 1982 Osborne Computer Corporation

```
EA1A  BD                          cmp     l
EA1B  3809 ^EA26$                 jrc     rhc          ;jump if cursor not in home window
EA1D  3A61EF                      ld      a,piaad      ;check for screen at home
EA20  D6EA                        sub     vflo
EA22  C8                          rz                   ;screen at home
EA23  AF                          xra     a            ;home screen
EA24  1818 ^EA3E$                 jr      scrl

EA26                    rhc:
                        ;         check right-hand margin
EA26  3A61EF                      ld      a,piaad      ;a=horizontal screen position
EA29  D6EA                        sub     vflo
EA2B  C664                        add     a,100        ;window size*2 (49)
EA2D  BD                          cmp     l
EA2E  DA3BEA                      jc      :30          ;move screen when cursor about to go off
                                                       ;the right hand margin
                        ;         check left hand margin
EA31  D65A                        sub     90
EA33  BD                          cmp     l            ;check left margin
EA34  D8                          rc                   ;cursor in window return
EA35  7D                          mov     a,l
EA36  D60A                        sub     10
EA38  C8                          rz                   ;return if cusor at column 2
EA39  1803 ^EA3E$                 jr      scrl

EA3B                    :30:
EA3B  7D                          mov     a,l
EA3C  D664                        sub     100
EA3E                    scrl:
EA3E  1F                          rar
EA3F  C620                        add     a,' '
EA41  2161EA                      lx      hl,escsq+3
EA44  77                          sto     a,[hl]
EA45  3A62EF                      ld      a,piabd
EA48  E61F                        and     1fh
EA4A  C620                        add     a,' '
EA4C  2B                          dec     hl
EA4D  77                          sto     a,[hl]               ;escsq+2 = vert. coords
EA4E  2B                          dec     hl
EA4F  2B                          dec     hl                   ;point to start of esc seq
EA50  0604                        lx      b,4
EA52                    :50:
EA52  C5                          push    bc
EA53  E5                          push    hl
EA54  4E                          ld      c,[hl]
EA55  CD0AEA                      call    outch
EA58  E1                          pop     hl
EA59  C1                          pop     bc
EA5A  23                          inc     hl
EA5B  10F5 ^EA52$                 djnz    :50
EA5D  C9                          ret

EA5E  1B                escsq     db      esc                  ;set screen coord escape sequence
EA5F  53                          db      'S'
EA60  00                          db      0                    ;** y coord
EA61  00                          db      0                    ;** x coord
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;**
                        ; routine:      SOSTAT
                        ;               gets status of output device attached to serial port
EA62                    SOSTAT: proc
EA62   CDD9EA                   call    acistat         ;get 6850 status
EA65   E602                     ani     si.trdy
EA67   C8                       rz                      ;return with not ready status
EA68   F6FF                     ori     true
EA6A   C9                       ret

                        ;**
                        ; routine:      SISTAT
                        ;               gets status of input device attached to serial port
EA6B                    SISTAT: proc
EA6B   CDD9EA                   call    acistat
EA6E   E601                     ani     si.rrdy
EA70   C8                       rz                      ;return with not ready status
EA71   F6FF                     ori     true
EA73   C9                       ret

                        ;**
                        ; routine:      SPINP
                        ;               Inputs a character from the serial port
                        ;               Resets serflg to show Receive Data Register has
                        ;               been read (RDRF bit = 0).
                        ;               A ROM call isn't used to input since the ROM
                        ;               doesn't fudge status to get around the hardware timing
                        ;               problem
EA74                    SPINP:  proc
EA74   CD6BEA                   call    sistat
EA77   28FB ^EA74$              jrz     SPINP                   ;if not ready

EA79   CD1EEC                   call    sw2rom
EA7C   AF                       xra     a
EA7D   32D8EA                   sto     a,serflg                ;reset RDRF bit
EA80   3A012A                   ld      a,h.srec                ;get data
EA83   4F                       mov     c,a
EA84   C32CEC                   jmp     sw2ram


                        ;**
                        ; routine:      SPOUT
                        ;               Outputs character in reg c to the serial port (list device)
                        ;               Since the ROM doesn't fudge status to get around the
                        ;               hardware timing problem, output is done directly
                        ;
EA87                    SPOUT:  proc
EA87   C5                       push    bc                      ;save character
EA88   CD62EA           :1:     call    sostat
EA8B   28FB ^EA88$              jrz     :1                      ;if not ready
EA8D   C1                       pop     bc

EA8E   CD1EEC                   call    sw2rom
EA91   79                       mov     a,c
EA92   32012A                   sto     a,h.sxmt                ;send chr
EA95   C32CEC                   jmp     sw2ram

                        ;**
                        ; routine does serial i/o and does printer protocols
                        ;
```

Copyright ©1982 Osborne Computer Corporation


```
        = 0011        xon:     =       11h
        = 0013        xoff:    =       13h
        = 0003        etx:     =       3
        = 0006        ack:     =       6
                      ;
EA98                  prtout:
EA98    CD87EA                 call    spout
EA9B    3A67E5                 ld      a,prnter
EA9E    B7                     ora     a
EA9F    C8                     rz
EAA0    E602                   ani     2
EAA2    2013 ^EAB7$            jrnz    xonxof
EAA4    3E0D                   mvi     a,0dh
EAA6    B9                     cmp     c
EAA7    C0                     rnz
EAA8    0E03                   ldk     c,etx
EAAA    CD87EA                 call    spout
EAAD                  prt10:
EAAD    CD74EA                 call    spinp
EAB0    E67F                   ani     7fh              ;mask out parity bit
EAB2    FE06                   cpi     ack
EAB4    20F7 ^EAAD$            jrnz    prt10
EAB6    C9                     ret
EAB7                  xonxof:
EAB7    CD6BEA                 call    sistat
EABA    B7                     ora     a
EABB    C8                     rz
EABC    CD74EA                 call    spinp
EABF    E67F                   ani     7fh              ;mask out parity bit
EAC1    FE13                   cpi     xoff
EAC3    C0                     rnz
EAC4                  prt20:
EAC4    CD74EA                 call    spinp
EAC7    E67F                   ani     7fh              ;mask out parity bit
EAC9    FE11                   cpi     xon
EACB    20F7 ^EAC4$            jrnz    prt20
EACD    C9                     ret


                      ;**
                      ; routine:        ACICTL
                      ;                 outputs character in c to the ACIA CTL port.
                      ;
EACE                  ACICTL: proc
EACE    CD1EEC                 call    sw2rom
EAD1    79                     mov     a,c
EAD2    32002A                 sta     h.sctrl
EAD5    C32CEC                 jmp     sw2ram

                      ;**
                      ; routine:        acistat
                      ;                 returns usart status in A
                      ;        The ringin signal is read from the pia at address 2c00
                      ;        and merged in at the overrun error bit of the acia status
                      ;
                      ;        A hardware timing problem results in the RDRF (Receive Data
                      ;        Register Full) sometimes being reset before the Receive Data
                      ;        Register has been read.  To get around this, a flag (serflg) keeps
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;       track of the correct status.  Every time the status register
                        ;       is read, serflg is updated.  If the RDRF bit is ever set in the
                        ;       status register, that bit remains high in serflg until the
                        ;       Receive Data Register is read.  The RDRF bit from serflg is
                        ;       ORed in with the status register to give the correct status.

     = 2C01             riport  equ     h.vio+1
     = 0040             ribit   equ     040h
     = 0020             ormask  equ     020h
     = 0001             RDRFm   equ     001h                    ;masks out all bits except RDRF
EAD8 00                 serflg  db      00                      ;flag to keep track of RDRF bit
                                                                ;(to correct for hardware timing problem)

EAD9                    acistat: proc
EAD9 CD1EEC                     call    sw2rom
EADC 3A012C                     lda     riport                  ;read in ringin signal
EADF 0F                         rrc                             ;shift ring in to overrun err
                                                                ;position
EAE0 E620                       ani     ormask                  ;get ring in
EAE2 4F                         mov     c,a
EAE3 3A002A                     lda     h.ssts
EAE6 E6DF                       ani     0ffh-ormask             ;remove overrun error bit
EAE8 B1                         ora     c                       ;merge ring in bit

EAE9 4F                         mov     c,a
EAEA 3AD8EA                     lda     serflg
EAED E601                       ani     RDRFm                   ;get RDRF bit
EAEF B1                         ora     c
EAF0 32D8EA                     sta     serflg                  ;save new status
EAF3 C32CEC                     jmp     sw2ram
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;***
                        ;       IEEE drivers:
                        ;
                        ;       The routines IEINSTAT, IEINP and IEOUT are used to
                        ;       transfer characters to and from an IEEE device attached to the
                        ;       OSBORNE IEEE port.  The address of the device is specified in
                        ;       the cell IE←ADRS.  No facility is provided at present to allow a
                        ;       transient programme to specify the device address.  Thus the device
                        ;       attached must be addressable as 0 (zero).
                        ;       The function IEINSTAT returns the status of the input device.
                        ;       Unfortunately there is no standard way by which an IEEE device
                        ;       indicates that it has a character.  In order to determine this, one
                        ;       has to read the character device.  As a CP/M transient can call
                        ;       IEINSTAT many times before calling IEINP to read a char, and IEINSTAT
                        ;       has to read the char to determine the status, the character read has to
                        ;       be buffered till call to IEINP is made.  IEINSTAT reads the device
                        ;       only when the buffer is empty.  As zeros are used to indicate
                        ;       that the bfr is empty, a null character can not be read from the
                        ;       IEEE device.

                        ;       IEEE control codes
       = 0040           ie←talk equ     40h                     ;make talker
       = 005F           ie←utlk equ     5fh                     ;make untalk
       = 0020           ie←lstn equ     20h                     ;make listener
       = 003F           ie←ulstn equ    3fh                     ;make unlisten

EAF6   00               ie←adrs db      0                       ;device address (can be moved to
                                                                ;BMRAM)
EAF7   00               ie←char db      0                       ;ie inp char buffer

                        ;**
                        ; routine:      IEINSTAT
                        ;               gets status of the input device attached to ieee port
                        ;
                        ;               if a char is present in ie←char then
                        ;                       return with true status
                        ;               else
                        ;                       make device talker
                        ;                       read the device
                        ;                       if char read then
                        ;                         store in bfr
                        ;                       endif
                        ;                       make untalk
                        ;                       return with status of buffer
                        ;               endif
EAF8                    IEINSTAT: proc
EAF8   3AF7EA                   lda     ie←char
EAFB   B7                       ora     a
EAFC   CA02EB                   jz      ieil0                   ;if char present then
EAFF   F6FF                     ori     0ffh                    ;  ;return with true status
EB01   C9                       ret
EB02                    ieil0:                                  ;endif
                                                                ;  ;make talker
EB02   3AF6EA                   lda     ie←adrs
EB05   C640                     adi     ie←talk                 ;  ;get primary address
EB07   4F                       mov     c,a
EB08   1E4B                     ldk     e,low(ieb5c)
```

Copyright © 1982 Osborne Computer Corporation

```
EB0A  CD68E6                call    romcdl              ;output interface message
EB0D  B7                    ora     a
EB0E  20F2 ^EB02$           jrnz    iei10               ;try again if error
                    ;
                    ;       ***secondary address output may be added here***

EB10                iei20:                              ;read a char.
EB10  1E51                  ldk     e,low(ieb7c)
EB12  CD68E6                call    romcdl
EB15  CB7D      $           bit     7,l
EB17  2801 ^EB1A$           jrz     iei30               ;if error then
EB19  AF                    xra     a                     ;inicate no char recvd
EB1A                iei30:
EB1A  32F7EA                sta     ie+char             ;stor the char
                    ;
                    ;
EB1D                iei40:                  ;make untalk
EB1D  0E5F                  mvi     c,ie+utlk
EB1F  1E4B                  ldk     e,low(ieb5c)
EB21  CD68E6                call    romcdl
EB24  B7                    ora     a
EB25  20F6 ^EB1D$           jrnz    iei40

                    ;       return with status of the char
EB27  3AF7EA                lda     ie+char
EB2A  B7                    ora     a
EB2B  C8                    rz
EB2C  F6FF                  ori     0ffh
EB2E  C9                    ret

                    ;**
                    ; routine:      IEINP
                    ;               Reads a character from IEEE port

EB2F                IEINP:  proc
EB2F  CDF8EA                call    ieinstat
EB32  28FB ^EB2F$           jrz     ieinp               ;wait till char avail
EB34  21F7EA                ldk     hl,ie+char
EB37  7E                    ld      a,[hl]
EB38  3600                  sto     0,[hl]              ;clear the buffer
EB3A  C9                    ret


                    ;**
                    ; routine:      IEOUT
                    ;               Outputs the character in reg C to IEEE port
                    ;               Uses ROM resident primitives.
                    ;
EB3B                IEOUT:  proc
EB3B  C5                    push    bc                  ;save the char
                                                          ;make listener
EB3C                ieo05:
EB3C  3AF6EA                lda     ie+adrs
EB3F  C620                  adi     ie+lstn               ;get primary address
EB41  4F                    mov     c,a
EB42  1E4B                  ldk     e,low(ieb5c)
EB44  CD68E6                call    romcdl                ;output interface message
EB47  B7                    ora     a
EB48  20F2 ^EB3C$           jrnz    ieo05                 ;try again if error
                    ;
```

Copyright © 1982 Osborne Computer Corporation

```
                    ;       ***secondary address output may be added here***
                    ;
EB4A                ieo20:                                  ;
EB4A  0600                  mvi     b,0                     ;do not send eoi
EB4C  C1                    pop     b
EB4D                ieo22:
EB4D  C5                    push    b                       ;save char again in case of retry
EB4E  1E4E                  ldk     e,low(ieb6c)
EB50  CD68E6                call    romcdl
EB53  C1                    pop     b
EB54  B7                    ora     a
EB55  20F6 ^EB4D$           jrnz    ieo22                   ;try again if error
                    ;
                    ;                                       ;make unlisten
EB57                ieo40:
EB57  0E3F                  mvi     c,ie+ulstn
EB59  1E4B                  ldk     e,low(ieb5c)
EB5B  CD68E6                call    romcdl
EB5E  B7                    ora     a
EB5F  20F6 ^EB57$           jrnz    ieo40

EB61  C9                    ret
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;***
                        ;       The Parallel port is actually the IEEE port driven with the centronix
                        ;       protocol.  The bit assignements of the PIA and PIB are as follows:
                        ;       PIA0-7 = data bus
                        ;       PIB0 = 0, data bus is output. 1, data bus is input
                        ;       PIB1 = set to 1.
                        ;       PIB2 = set to 0.
                        ;       PIB3 = 0 output, 1 Input
                        ;       PIB4 = not used
                        ;       PIB5 = output strobe. Active = 1.
                        ;       PIB6 = 0, printer busy. 1, printer is ready.
                        ;       PIB7 = not used.
                        ;
                        ;       CA2 = going low indicates to device that we are busy.
                        ;       CA1 = low to high transition gates input data to port a.
                        ;
                        ;       The port is bidirectional but only one direction
                        ;       can be active at any time. The direction of port is determined
                        ;       by which routines are called.  If postat or parout are
                        ;       called, it is made an output port and an input port if
                        ;       pistat or parinp are called.

                        ;       port registers
         = 2900         pa.dta  equ     h.ieee+0
         = 2900         pa.dir  equ     pa.dta
         = 2901         pa.ctl  equ     h.ieee+1
         = 2902         pb.dta  equ     h.ieee+2
         = 2902         pb.dir  equ     pb.dta
         = 2903         pb.ctl  equ     h.ieee+3

                        ;       port ctl register constants.
         = 002A         pa.cdr  equ     00101010b       ;to address port a direction
         = 002E         pa.cdt  equ     00101110b       ;to address port a data and set
                                                        ;port a in input program handshake mode.
         = 0000         pb.cdr  equ     00000000b       ;to address port b direction
         = 0004         pb.cdt  equ     00000100b       ;to address port b data

                        ;       direction register constants
         = 00FF         pa.dro  equ     0ffh            ;port a output mode
         = 0000         pa.dri  equ     00h             ;port a input mode
         = 00BF         pb.dr   equ     0bfh            ;port b direction
         = 0002         pb.dto  equ     00000010b       ;port b data for output
         = 000B         pb.dti  equ     00001011b       ;port b data for input

         = 0040         pp.ordy equ     01000000b       ;output rdy bit in pib
         = 0080         pp.irdy equ     10000000b       ;input rdy bit in pia ctl reg
         = 0020         strb    equ     00100000b       ;strobe bit in port b

                        ;       port modes
         = 0000         pp.undef equ    0
         = 0001         pp.out  equ     1
         = 0002         pp.in   equ     2

EB62    00              pp.mode db      pp.undef

                        ;***
                        ; sbrt:         CV2OP.          initializes the port to a parallel output
                        ;                               port.
EB63                    cv2op:  proc
EB63    3A62EB                  lda     pp.mode
```

Copyright © 1982 Osborne Computer Corporation

```
EB66  FE01              cpi     pp.out
EB68  C8                rz                      ;return when in output mode
                ;       set port a to output on all lines
EB69  3E2A              lk      a,pa.cdr
EB6B  320129            sta     pa.ctl          ;select direction reg
EB6E  3EFF              lk      a,pa.dro
EB70  320029            sta     pa.dir          ;output constant to dir. reg to put
                ;                                a port in output mode
EB73  3E2E              lk      a,pa.cdt
EB75  320129            sta     pa.ctl          ;select port a data reg.

EB78  3E00              lk      a,pb.cdr
EB7A  320329            sta     pb.ctl          ;select port b direction
EB7D  3EBF              lk      a,pb.dr
EB7F  320229            sta     pb.dir          ;all lines are output except the output
                                                ;busy signal on bit 6
EB82  3E04              lk      a,pb.cdt
EB84  320329            sta     pb.ctl          ;select data register
EB87  3E02              lk      a,pb.dto
EB89  320229            sta     pb.dta          ;initialize port b data

EB8C  3E01              lk      a,pp.out
EB8E  3262EB            sta     pp.mode
EB91  C9                ret
                ;***
                ; sbrt:         CV2IP.          initializes the port to a parallel input
                ;                               port.
EB92            cv2ip:  proc
EB92  3A62EB            lda     pp.mode
EB95  FE02              cpi     pp.in
EB97  C8                rz                      ;return when in input mode
                ;       set port a to input on all lines
EB98  3E2A              lk      a,pa.cdr
EB9A  320129            sta     pa.ctl          ;select direction reg
EB9D  3E00              lk      a,pa.dri
EB9F  320029            sta     pa.dir          ;output constant to dir. reg to put
                ;                                a port in input mode
EBA2  3E2E              lk      a,pa.cdt
EBA4  320129            sta     pa.ctl          ;select port a data reg.

EBA7  3E00              lk      a,pb.cdr
EBA9  320329            sta     pb.ctl          ;select port b direction
EBAC  3EBF              lk      a,pb.dr
EBAE  320229            sta     pb.dir          ;all lines are output except the output
                                                ;busy signal on bit 6
EBB1  3E04              lk      a,pb.cdt
EBB3  320329            sta     pb.ctl          ;select data register
EBB6  3E0B              lk      a,pb.dti
EBB8  320229            sta     pb.dta          ;initialize port b data

EBBB  3E02              lk      a,pp.in
EBBD  3262EB            sta     pp.mode
EBC0  C9                ret


                ;**
                ; routine:      POSTAT
                ;               gets status of the parallel (centronix) printer
                ;               attached to the ieee port
                ;
```

Copyright © 1982 Osborne Computer Corporation

```
EBC1                POSTAT: proc
EBC1  CD1EEC                call    sw2rom
EBC4  CD63EB                call    cv2op           ;convert to output
EBC7  3A0229                lda     pb.dta          ;get port b data
EBCA  E640                  ani     pp.ordy
EBCC  2802 ^EBD0$           jrz     pos10
EBCE  F6FF                  ori     true
EBD0                pos10:
EBD0  C32CEC                jmp     sw2ram


                    ;**
                    ; routine:      PISTAT
                    ;               gets status of the input device attached to the
                    ;               parallel port
                    ;
EBD3  00            piactl  db      0

EBD4                PISTAT: proc
EBD4  CD1EEC                call    sw2rom
EBD7  CD92EB                call    cv2ip
EBDA  3AD3EB                lda     piactl
EBDD  E680                  ani     pp.irdy
EBDF  200A ^EBEB$           jrnz    pis20           ;if saved status indicates there is a char
                                                    ;in the PIA
EBE1  3A0129                lda     pa.ctl
EBE4  32D3EB                sta     piactl          ;this is saved as reading the
                                                    ;pia clears the status
EBE7  E680                  ani     pp.irdy
EBE9  2802 ^EBED$           jrz     pis30
EBEB                pis20:
EBEB  F6FF                  ori     true
EBED                pis30:
EBED  C32CEC                jmp     sw2ram


                    ;**
                    ; routine:      PARINP
                    ;               inputs a character from parallel port.
EBF0                PARINP: proc
EBF0  CDD4EB                call    pistat
EBF3  28FB ^EBF0$           jrz     parinp          ;wait till char in pia
EBF5  CD1EEC                call    sw2rom
EBF8  AF                    xra     a
EBF9  32D3EB                sta     piactl          ;clear saved status
EBFC  3A0029                lda     pa.dta
EBFF  2F                    cma                     ;invert data
EC00  4F                    mov     c,a             ;also in c
EC01  C32CEC                jmp     sw2ram



                    ;**
                    ; routine:      PAROUT
                    ;               outputs the character in c to the IEEE port treating the
                    ;               port as a parallel port.
EC04                PAROUT: proc
EC04  CDC1EB                call    postat
EC07  28FB ^EC04$           jrz     parout
```

```
EC09  CD1EEC          call    sw2rom
EC0C  79              mov     a,c
EC0D  2F              cma                     ;invert data
EC0E  320029          sta     pa.dta
EC11  3E22            lk      a,pb.dto+strb
EC13  320229          sta     pb.dta          ;set strobe
EC16  3E02            lk      a,pb.dto
EC18  320229          sta     pb.dta          ;clear strobe
EC1B  C32CEC          jmp     sw2ram
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                             ;***
                             ; sbrt:           SW2ROM
                             ;                 switches to rom
                             ;                 saves all registers
EC1E                         SW2ROM: proc
EC1E                                 di
EC1E +F3                             DB       0F3h
EC1F  F5                             push     af
EC20                                 enarom
EC20 +                               DI
EC20 +F3                             DB       0F3h
EC21 +D300                           OUT      0
EC23 +3E00                           LDK      A,0
EC25 +3208EF                         STO      A,ROMRAM
EC28 +                               EI
EC28 +FB                             DB       0FBh
EC29  F1                             pop      af
EC2A                                 ei
EC2A +FB                             DB       0FBh
EC2B  C9                             ret

                             ;***
                             ; sbrt:           SW2RAM
                             ;                 switches to ram
                             ;                 preserves all registers
EC2C                         SW2RAM: proc
EC2C                                 di
EC2C +F3                             DB       0F3h
EC2D  F5                             push     af
EC2E                                 disrom
EC2E +                               DI
EC2E +F3                             DB       0F3h
EC2F +D301                           OUT      1
EC31 +3E01                           LDK      A,1
EC33 +3208EF                         STO      A,ROMRAM
EC36 +                               EI
EC36 +FB                             DB       0FBh
EC37  F1                             pop      af
EC38                                 ei
EC38 +FB                             DB       0FBh
EC39  C9                             ret

      = EC3A                 xxx:    =        *
      = 0000                         if       xxx>MRAMEX
   -                         .9      error    CODE TOO LARGE..
                                     endif


EC3A  = EC70                         ORG      BIOS+770H            ;This is location ED70h in 60K system
EC70  00CF                           DW       CCP                  ;This location needed by ROM for
                                                                   ;relocating boot loader
                                                                   ;last sector read bombs out part
                                                                   ;of BMRAM in 60K system

                             ;OCCRAM2.ASM
```

Copyright ©1982 Osborne Computer Corporation

```
                        ;       Used to assembly ROM resident and CBIOS

EC72  = ED70                    ORG     MRAMEXT
ED70  = 0002            copadr: DS      2                       ;This location is assigned in BIOS
                                                                ;and filled in by loader in ROM
ED72  = 0006            KEYLST: DS      KL+LEN*KLE+LEN

                        ;
ED78  = ED80                    ORG     MRAM

                        ;       Host disk xfer buffer and...
                        ;       Format track template holding buffer
ED80                    HSTBUF:
ED80  = 0180                    DS      256+128

                        ;       Directory Buffer
      = EE80            DIRBUF: =       HSTBUF+256

EF00  = 0006            TEM     DS      6
      = EF01            RNDV    =       TEM+1   ;random number seed
      = EF02            ERCNT   =       RNDV+1  ;DW ERCNT
      = EF04            RTRC    =       ERCNT+2 ;retry count
      = EF05            RTRY    =       RTRC+1
EF06  = 0001            MPCHR   DS      1       ;prompt character
EF07  = 0001            ECHOP   DS      1       ;=0, list ehco off
EF08  = 0001            ROMRAM  DS      1       ;0= RAM, 1= ROM
EF09  = 0006            DSTS8   DS      6       ;Disk status bytes

                        ;       Disk operation temps and control
EF0F  = 0002            DMADR   DS      2       ;Address for read/write Disk
EF11  = 0002            DMAADR  DS      2       ;CBIOS, users DMA

                        ;       Note order of xxxSEC,xxxTRK,xxxDSK must be maintained
                        ;       along with length (1,2,1).
EF13  = 0001            SEKDEL: DS      1       ;Set for seek-restore command in ROM
                                                ;depends on disk type. Siemens = 3h, MPI = 0h
EF14  = 0001            SAVSEC  DS      1       ;last sector requested
EF15  = 0002            SAVTRK  DS      2       ;last track requested
EF17  = 0001            SDISK   DS      1       ;Selected disk drive (0,1)

      = EF14            ACTSEC  =       SAVSEC
      = EF15            ACTTRK  =       SAVTRK
      = EF17            ACTDSK  =       SDISK

EF18  = 0001            SEKSEC  DS      1
EF19  = 0002            SEKTRK  DS      2
EF1B  = 0001            SEKDSK  DS      1

EF1C  = 0001            HSTSEC  DS      1
EF1D  = 0002            HSTTRK  DS      2
EF1F  = 0001            HSTDSK  DS      1

EF20  = 0001            TEMSEC  DS      1       ;Used in bios only
EF21  = 0001            RDFLAG  DS      1       ;Read flag
EF22  = 0001            ERFLAG  DS      1       ;Error reporting
EF23  = 0001            WRTYPE  DS      1       ;Write operation type

EF24  = 000C            ALV:    DS      ALVS
EF30  = 0020            CSV:    DS      CSVS
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;       BIOS blocking-deblocking flags
EF50  = 0001            HSTACT: DS      1       ;host active flag
EF51  = 0001            HSTWRT: DS      1       ;Host written flag
EF52  = 0001            UNACNT: DS      1       ;Unalloc rec count
EF53  = 0002            UNATRK: DS      2       ;Track
EF55  = 0001            UNASEC: DS      1       ;Sector
EF56  = 0001            LOGSEC: DS      1       ;Logical sector


EF57  = 0002            LOADR   DS      2
EF59  = 0001            KEYLCK  DS      1       ;Zero if locked keyboard
EF5A  = 0002            CURS    DS      2       ;current cursor position

                        ;       Keyboard scan temporaries
EF5C  = 0001            TKEY    DS      1       ;Tem holding key
EF5D  = 0001            HKCNT   DS      1       ;Debounce key
EF5E  = 0001            LKEY    DS      1       ;Last valid keystroke
EF5F  = 0001            CKEY    DS      1       ;Last control key
EF60  = 0001            ESCH    DS      1       ;ESC holding flag

                        ;PIAAD and PIABD must be kept sequential, PIAAD first
                        ;dependency in VC+HOME of BMKEY.asm
EF61  = 0001            PIAAD:  DS      1       ;Holds last PIA-A data
EF62  = 0001            PIABD:  DS      1       ;Holds last PIA-B data

                        ;       Calendar month, day year
EF63  = 0003            IDAY    DS      3
      = EF64            IMONTH  =       IDAY+1
      = EF65            IYR     =       IDAY+2

                        ;       Wall clock time cells and disk active
                        ;       see UPTIM: in BMKEY.asm
EF66  = 0006            HOURS:  DS      6
      = EF67            MINS:   =       HOURS+1
      = EF68            SECS:   =       HOURS+2
      = EF69            SEC6:   =       HOURS+3

                        ;       Used to deselect drive when there is NO activity
                        ;       on drive for n seconds.  See FDSK routine
      = EF6A            DACTVE: =       HOURS+4 ;=0 by FDSK, Used by UPTIM

      = EF6B            BELCNT: =       HOURS+5 ;^G bell timer cell


EF6C  = 0001            LLIMIT  DS      1       ;max #columns in a logical line
                        ;       MSG     'LLIMIT = ',LLIMIT,'h.'

                        ;       Disk drive current positions
EF6D  = 0002            LDSEL:  DS      2       ;Last selected drive
      = EF6E            LDTRK   =       LDSEL+1 ;Last track used for non-selected drive

EF6F  = 0002            IESTK:  DS      2       ;save current stk ptr


                        ;       Interrupt stack
EF71  = 0028                    DS      20*2
EF99  = 0000            ISTK:   DS      0


                        ;       Stack entry
```

Copyright ©1982 Osborne Computer Corporation

```
EF99  = 0028            DS      20*2
EFC1             BIOSTK:
EFC1  = 0000     ROMSTK: DS      0

EFC1  = 0001     ACIAD:  DS      1       ;last command byte written to ACIA

EFC2  = 0004     R179x:  DS      4       ;179x register save area
EFC6  = 0001     KBDLY:  DS      1       ;keyboard debounce-delay cell

                 ;since CP/M CANNOT boot off B:, this cell is used
                 ;to invert the names of the 2 drives:
                 ;       =0, all normal, A=A:, B=B:
                 ;       =1, all inverted, A=B:, B=A:
EFC7  = 0001     DSKSWP  DS      1


                 ;       Z80 Alternate Register Set
EFC8                     ALIGN   10h
EFC8 += 0008             DS      (*+(10H)-1)/(10H)*(10H)-*
EFD0             RAGS:
EFD0  = 0002     DESAX:  DS      2       ;DE'
EFD2  = 0002     BCSAX:  DS      2       ;BC'
EFD4  = 0002     AFSAX:  DS      2       ;AF'
EFD6  = 0002     HLSAX:  DS      2       ;HL'

EFD8  = 0002     IXSAX:  DS      2       ;IX
EFDA  = 0002     IYSAX:  DS      2       ;IY
EFDC  = 0002     IVSAX:  DS      2       ;Interrupt page register


                 ;       8080 Register Save Area.
EFDE                     ALIGN   10h
EFDE += 0002             DS      (*+(10H)-1)/(10H)*(10H)-*
EFE0             REGS:
EFE0  = 0001     ESAVE:  DS      1       ;E Register save location
EFE1  = 0001     DSAVE:  DS      1       ;D Register save location
EFE2  = 0001     CSAVE:  DS      1       ;C Register save location
EFE3  = 0001     BSAVE:  DS      1       ;B Register save location
EFE4  = 0001     FSAVE:  DS      1       ;FLAGS save location
EFE5  = 0001     ASAVE:  DS      1       ;A Register save location
EFE6  = 0001     LSAVE:  DS      1       ;L Register save location
EFE7  = 0001     HSAVE:  DS      1       ;H Register save location
EFE8  = 0002     PSAVE:  DS      2       ;PGM COUNTER save location
EFEA  = 0002     SSAVE:  DS      2       ;USER STACK pointer save location

EFEC  = 0002     BKPA:   DS      2       ;last breakpoint address
EFEE  = 0001     BKPC:   DS      1       ;Contents of bkp

EFEF  = 0001     VRTOFF  DS      1       ;LAST VERTICAL OFFSET TAKEN FROM COUT
                 ;
                 ;
                 ;       Interrupt Jump Vector is between EFF8, EFFF.
                 ;       Endx    MRAM


no ERRORs, 517 Labels, 6AD5h bytes not used.     Program LWA = EFF0h.
```

```
s  @FREQ   0FA0
 n ACIAD   EFC1      43# 5
   ACICTL  EACE       5/46   31#46
   ACISTA  EAD9       5/47   30/ 5    30/15    32#14
   ACK     0006      31# 5   31/22
   ACMD    E589       6#24   15/48
   ACTDSK  EF17      19/48   20/10    41#43
   ACTSEC  EF14      19/54   20/14    20/33    41#41
   ACTTRK  EF15      19/51   20/12    41#42
 n AFSAX   EFD4      43#23
   AHSCRL  EA0F      27/16   28#51
   AHSENB  E568       6# 4   28/30    28/32    28/52
s  ALIGN   mac       43/19   43/33
   ALV     EF24       7/28    7/41    41#58
s  ALVS    000C       8/ 2   41/58
   ALVSZ   000C       4#12    7/28     7/31     7#31     7/41     7/44     7#44
                      8/ 2
 n ASAVE   EFE5      43#40
 n BASVLO  0080      27# 9
   BCCP    E748      15/ 3   15#16
   BCPM    E73A      15# 9   15/13
 n BCSAX   EFD2      43#22
   BDOS    D706       3#15   15/36
 n BELCNT  EF68      42#42
   BIOS    E500       3#14    5/ 8     6/54    15/34    40/49
   BIOSTK  EFC1      10/24   43# 2
 n BKPA    EFEC      43#46
 n BKPC    EFEE      43#47
   BRATE   E569       6# 5   14/57
 n BSAVE   EFE3      43#38
   CAUTO   E58A       6#28    6/31    15/40
   CAUTOL  0007       6/28    6#31
s  CBIOSV  0001      16/11
   CBOOT   E70C       5/10   14#39
   CCP     CF00       3#13    3/14     3/15    14/51    15/ 6    15/10    15/28
                     15/54   16/22    40/50
 n CCPADR  ED70      41# 4
s  CDISK   0004      11/49   11/52    14/50    16/24
 n CKEY    EF5F      42#18
   CNIN    E910       5/13   22#37
   CNOUT   E915       5/14   22#44
   CNSTA   E90B       5/12   22#31
   CNTRL0  E592       6/ 8    6#33
   CNTRL1  E593       6/ 9    6#34
   CNTRL2  E594       6/10    6#35
   CNTRL3  E595       6/11    6#36
   CNTRL4  E596       6/12    6#37
   CNTRL5  E597       6/13    6#38
   CNTRL6  E598       6/14    6#39
   CNTRL7  E599       6/15    6#40
   CNTRL8  E59A       6/16    6#41
   CNTRL9  E59B       6/17    6#42
   CONIN   E509       5#13   27/28
   CONOUT  E50C       5#14   21/29    28/47
   CONST   E9A1      24/45   26#14
   CONST5  E9AD      26/18   26#24
   COUNT   E9C2      26/16   27/17    27#23    27/50
s  CR      000D      16/ 8   16/11
   CRSTAT  E9F3      25/ 2   28# 9
   CRT10   EA0A      28/26   28/29    28#35
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
    CRTOUT E9F6    24/57   25/ 7   28#22
  n CSAVE  EFE2    43#37
    CSV    EF30     7/27    7/40   41#59
 s  CSVS   0020    41/59
    CSVSZ  0020     4#13    7/27    7/30    7#30    7/40    7/43    7#43
    CUNACT E6EB    12#25   13/12
    CURS   EF5A    28/56   42#12
    CV2IP  EB92    37#28   38/21
    CV2OP  EB63    36#59   38/ 3
  n DACTVE EF6A    42#40
 s  DBUF   0080    15/19
  n DESAX  EFD0    43#21
 sD DI     mac      9/21   10/21   10/29   10/39   10/43   14/43   40/ 7
                   40/11   40/28   40/32
    DIRBUF EE80     7/25    7/38   41#17
  n DIROM  0001     3#46
    DISPCH E93D    22/49   22/56   23/ 3   23/10   23/34   23#38
 s  DISROM mac     10/42   14/42   40/31
    DISSTK E979    24/12   24#39
 s  DMA    0080
    DMAADR EF11    12/53   19/23   41#31
    DMADR  EF0F    15/22   41#30
    DONE   E7BC    15/50   15/53   16#13
    DONE0  E7C7    16/16   16#19
    DONE1  E7CA    15/59   16#21
    DOWN   E59E     6/20    6#45
    DPBASE E614     7#19   11/38
    DPBGEN mac      4#50    7/54
    DPBS1  E634     7/26    7/39    7#51
    DPHGEN mac      4#16    7/21    7/34
  n DSAVE  EFE1    43#36
    DSKS1  0000     3#30    7/54
  n DSKSWP EFC7    43#14
    DSPCH1 E94E    24#17   24/19
    DSPRET E960    24/29   24#33
  n DSTSB  EF09    41#27
  n ECHOP  EF07    41#25
    EFESC  0008    28#19   28/24
    EFGR   0001    28#20   28/24
 sD EI     mac      9/28    9/46   10/34   10/48   10/53   10/57   14/48
                   40/16   40/19   40/37   40/40
 s  ENAROM mac     10/28   40/10
  n ENROM  0000     3#45
    EOTBL  E5A0     6/22    6#47
    ERCNT  EF02    41#21   41/22
    ERFLAG EF22    17/16   18/16   18/23   20/41   20/48   21/ 5   41#55
  n ESAVE  EFE0    43#35
 s  ESC    001B    29/53
    ESCH   EF60    15/26   28/23   42#19
    ESCSQ  EA5E    29/32   29#53
    ETX    0003    31# 4   31/17
 s  FALSE  0000
 s  FCB    005C
    FILL   E84B    17/11   18/ 9   19#11
    FILL3  E85C    19/17   19#19
    FILL4  E87A    19#36   19/41
    FILL5  E883    19/38   19#44
    FILL6  E886    19/32   19#46
    FINAL  E8C6    19/60   20#21
    FLUSH  E8AA    11/ 8   18/22   19/44   20# 3
```

```
n FMTJ    E539     5#31
  FNL1    E3D6    20#36   20/46
  FPYSIZ  0010     3#24    3/32
n FSAVE   EFE4    43#39
s FWAVM   F000
  GODISP  E918    22/33   22/39   22#46
  GOROM   E69A    10/37   10#56
s H.IEEE  2900    36/24   36/26   36/27   36/29
s H.SCTR  2A00    31/49
s H.SIO   2A00
s H.SREC  2A01    30/36
s H.SSTS  2A00    32/21
s H.SXMT  2A01    30/55
s H.VIO   2C00    32/ 7
n HKCNT   EF5D    42#16
n HLSAX   EFD6    43#24
  HOME    E69F     5/18   11# 7   15/ 7
  HOURS   EF66    42#33   42/34   42/35   42/35   42/40   42/42
n HSAVE   EFE7    43#42
  HSTACT  EF50    11/10   17/20   19/28   42# 2
  HSTBUF  ED80    15/21   19/12   41#13   41/17
  HSTDSK  EF1F    19/47   20/ 9   41#51
  HSTSEC  EF1C    19/33   19/53   20/13   41#49
n HSTSIZ  0100     3#23
  HSTTRK  EF1D    19/50   20/11   41#50
  HSTWRT  EF51    18/15   20/ 4   42# 3
  IDAY    EF63    15/38   42#27   42/28   42/29
  IEADRS  EAF6    33#29   33/57   34/53
n IEB1C   E53F     5#35
n IEB2C   E542     5#36
n IEB3C   E545     5#37
n IEB4C   E548     5#38
  IEB5C   E54B     5#39   33/60   34/19   34/55   35/17
  IEB6C   E54E     5#40   35/ 8
  IEB7C   E551     5#41   34/ 8
n IEB8C   E554     5#42
  IECHAR  EAF7    33#31   33/50   34/14   34/25   34/38
  IEI10   EB02    33/52   33#55   34/ 3
n IEI20   EB10    34# 7
  IEI30   EB1A    34/11   34#13
  IEI40   EB1D    34#17   34/22
  IEINP   EB2F    24/54   34#35   34/37
  IEINST  EAF8    24/48   33#49   34/36
  IELSTN  0020    33#25   34/54
  IEO05   EB3C    34#52   34/59
n IEO20   EB4A    35# 3
  IEO22   EB4D    35# 6   35/12
  IEO40   EB57    35#15   35/20
  IEOSTA  E9F3    25/ 5   28# 3
  IEOUT   EB3B    24/60   25/10   34#49
  IESTK   EF6F     9/44   42#52
  IETALK  0040    33#23   33/53
  IEULST  003F    33#26   35/16
  IEUTLK  005F    33#24   34/18
n IMONTH  EF64    42#23
  IOBITE  E566     5#58   14/55
s IOBYTE  0003    14/56   24/16
n ISTK    EF99    42#57
n IVSAX   EFDC    43#28
n IXSAX   EFD8    43#26
```

Copyright © 1982 Osborne Computer Corporation

Copyright ©1982 Osborne Computer Corporation


```
 n IYR     EF65   42#29
 n IYSAX   EFDA   43#27
 n KBDLY   EFC6   43# 8
   KEYINP  E982   24/51   27#12   27/51
 n KEYLCK  EF59   42#11
 n KEYLST  ED72   41# 6
   KIN10   E9C5   27/20   27#27
s  KLELEN  0002   41/ 6
s  KLLEN   0003   41/ 6
 n LDADR   EF57   42#10
   LDSEL   EF6D   42#49   42/50
 n LDTRK   EF6E   42#50
   LEFT    E59F    6/21    6#46
s  LF      000A   16/ 8   16/11
 n LIST    E50F    5#15
 n LISTST  E52D    5#25
   LKEY    EF5E   26/21   42#17
   LLIMIT  EF6C   14/61   42#45
   LOGSEC  EF56   13/14   17/48   42# 7
 n LSAVE   EFE6   43#41
   LST     E91D    5/15   22#53
   LSTST   E935    5/25   23#30
s  LVMEM   1000
s  LWAMEM  FFFF    3/19
 n MINS    EF67   42#34
 n MPCHR   EF06   41#24
s  MRAM    ED80   41/ 9
s  MRAMEX  ED70   40/44   41/ 3
   MRTRY   0005    3#10   20/35
s  MSEC    000A    7/10    7/55
   MSIZE   003C    3#11   16/ 8   16/ 8
   MVINFO  E8F2   17/ 8   17/39   21# 3
   NDSK    0002    4#10    7/21    7#21    7/34    7#34   11/28
   NOFDD   0002    4#11    7/29    7#29    7/42    7#42
   NOKEY   0000   26#12   26/22
s  NVDL    0018
   ORMASK  0020   32# 9   32/19   32/22
   OUTCH   EA0A   28#46   29/46
   PA.CDR  002A   36#32   37/ 4   37/33
   PA.CDT  002E   36#33   37/ 9   37/38
   PA.CTL  2901   36#26   37/ 5   37/10   37/34   37/39   38/26
   PA.DIR  2900   36#25   37/ 7   37/36
   PA.DRI  0000   36#40   37/35
   PA.DRO  00FF   36#39   37/ 6
   PA.DTA  2900   36#24   36/25   38/46   39/ 4
   PARINP  EBF0   24/53   38#40   38/42
   PAROUT  EC04   24/59   25/ 9   38#58   38/60
   PB.CDR  0000   36#35   37/12   37/41
   PB.CDT  0004   36#36   37/17   37/46
   PB.CTL  2903   36#29   37/13   37/18   37/42   37/47
   PB.DIR  2902   36#28   37/15   37/44
   PB.DR   00BF   36#41   37/14   37/43
   PB.DTA  2902   36#27   36/28   37/20   37/49   38/ 4   39/ 6   39/ 8
   PB.DTI  000B   36#43   37/48
   PB.DTO  0002   36#42   37/19   39/ 5   39/ 7
   PIAAD   EF61   29/ 3   29/11   42#23
   PIABD   EF62   29/34   42#24
   PIACTL  EBD3   38#17   38/22   38/27   38/45
   PIS20   EBEB   38/24   38#31
   PIS30   EBED   38/30   38#33
```

Copyright © 1982 Osborne Computer Corporation

```
   PISTAT EBD4     24/47   38#19   38/41
   PNCH   E925      5/16   22#60
   POS10  EBD0     38/ 6   38# 8
   POSTAT EBC1     25/ 4   37#61   38/59
   PP.IN  0002     36#52   37/30   37/51
   PP.IRD 0080     36#45   38/23   38/29
   PP.MOD EB62     36#54   36/60   37/23   37/29   37/52
   PP.ORD 0040     36#45   38/ 5
   PP.OUT 0001     36#51   36/61   37/22
   PP.UND 0000     36#50   36/54
   PRINT  E8FD     16/18   21#23   21/31
   PRNTER E567      5#61   31/ 9
   PRT10  EAAD     31#19   31/23
   PRT20  EAC4     31#33   31/37
   PRTOUT EA98     25/ 8   31# 7
 n PSAVE  EFE8     43#43
   PTRCIN E981     22/38   24#50
   PTRCOU E989     22/45   24#56
   PTRCST E979     22/32   24#44
   PTRLIS E999     22/55   25# 6
   PTRLST E991     23/33   24#61
   PTRPNC E989     23/ 2   24#55
   PTRRDR E981     23/ 9   24#49
 n PUNCH  E512      5#16
 n R179X  EFC2     43# 7
 n RAGS   EFD0     43#20
   RDFLAG EF21     19/11   19/55   41#54
   RDR    E92D      5/17   23# 7
   RDRFM  0001     32#10   32/27
   READ   E7D3      5/23   16#29
 n READER E515      5#17
 n REGS   EFE0     43#34
   RHC    EA26     29/ 2   29# 9
 n RIBIT  0040     32# 8
   RIGHT  E59D      6/19    6#44
   RIPORT 2C01     32# 7   32/16
   RNDV   EF01     41#20   41/21
   ROMCD1 E668     10#12   27/29   28/48   33/61   34/ 9   34/20   34/57
                   35/ 9   35/19
   ROMCDE E663      5/31    5/32    5/35    5/36    5/37    5/38    5/39
                    5/40    5/41    5/42    5/43    5/44    5/45    9#48
   ROMJMP E66A      5/30   10#15   11/44   15/11   20/40
   ROMRAM EF08      9/25    9/35   10/32   10/46   14/46   40/14   40/35
                   41#26
   ROMRI  E64F      5/29    9#34
 n ROMSTK EFC1     43# 3
s  ROMVEC 0100     10/13   15/ 9   20/30
 n RRDK   E527      5#23
 n RRI    E533      5#29
   RSELDK E518      5#19   11/41
   RTRC   EF04     41#22   41/23
   RTRY   EF05     20/36   20/44   41#23
 n RWDK   E52A      5#24
   S1DSM  002E      3#32    7/31    7/44    7/57
   SAVADR E909     20/32   20/37   21#33
   SAVSEC EF14     41#37   41/41
   SAVTRK EF15     41#38   41/42
   SBAUD  E53C      5#32   14/59
   SCRL   EA3E     29/ 7   29/24   29#29
   SCRSZE E56A      6# 6   14/60
```

Copyright ©1982 Osborne Computer Corporation

```
  SDISK  EF17   41#39   41/43
n SEC6   EF69   42#36
n SECS   EF68   42#35
  SECTRN E6FA    5/26   13# 9
n SEKDEL EF13   41#35
  SEKDSK EF1B   11/30   19/35   19/46   41#47
  SEKSEC EF18   19/14   19/22   19/34   19/35   19/52   21/ 7   41#45
  SEKTRK EF19   11/12   11/13   12/ 9   17/46   19/49   41#46
  SELD1  E6D1   11/29   11#43
  SELDSK E6B0    5/19   11#19
  SERFLG EAD8   30/35   32#11   32/25   32/29
  SETDMA E6F5    5/22   12#52   15/20
n SETRRM E644    9#16
  SETSEC E6F0    5/21   12#37
  SETTRK E6DD    5/20   12# 8
s SI.120 0055    6/ 5
s SI.RRD 0001   30/16
s SI.S16 0055
s SI.S64 0056
s SI.TRD 0002   30/ 6
  SIGNON E792   15#61   16/17
  SISTAT EA6B   24/45   30#14   30/30   31/25
  SOSTAT EA62   25/ 3   30# 4   30/49
  SPINP  EA74   24/52   30#29   30/31   31/20   31/29   31/34
  SPOUT  EA87   24/58   30#47   31/ 3   31/18
n SSAVE  EFEA   43#44
  STOHL  E69D   10/19   10/26   10#60   24/ 9   24/14
  STRB   0020   36#47   39/ 5
s SVER   0001
  SW2RAM EC2C   30/38   30/56   31/50   32/30   38/ 9   38/34   38/49
                39/ 9   40#26
  SW2ROM EC1E   30/33   30/53   31/47   32/15   38/ 2   38/20   38/43
                38/61   40# 5
s SYS    0005
s SYSDAT 0010
s SYSL   0006
  TEM    EF00   41#19   41/20
  TEMSEC EF20   12/38   21/ 6   41#53
s TIMPTR 0040   15/39
n TKEY   EF5C   42#15
s TRUE   FFFF    6/ 4   30/ 8   30/18   38/ 7   38/32
  UNACNT EF52   11/11   12/25   17/10   17/45   17/52   42# 4
  UNASEC EF55   13/10   17/57   18/ 6   42# 6
  UNATRK EF53   12/10   17/47   17/61   18/ 3   42# 5
  UP     E59C    6/18    6#43
  VERS   0016    1#48   16/ 9   16/ 9
s VFLO   FFEA   29/ 4   29/12
s VLDL   0034
n VRTOFF EFEF   43#49
  WBOOT  E734    5/11   15# 5
n WRALL  0000    3#36
  WRDIR  0001    3#37   18/21
  WRIT2  E80C   17/43   17#52
  WRIT3  E825   17/50   17/60   18# 6
  WRIT4  E82A   17/55   18# 9
  WRITE  E7F0    5/24   17#27
  WRTYPE EF23   17/41   18/20   41#56
  WRUAL  0002    3#38   17/42
  XLTBL  E568    6# 8   27/41
  XLTKEY E9B0   27#10   27/21   27/47
```

```
XLTS    E600      6#56    7/10    7/22    7/35
XOFF    0013     31# 3   31/31
XON     0011     31# 2   31/36
XONXOF  EAB7     31/13   31#25
XXX     EC3A     40#43   40/44
```

Copyright © 1982 Osborne Computer Corporation

ROM/BIOS 1.4

Copyright © 1982 Osborne Computer Corporation

```
          *ABS    0000  0FFF
          *CODE   0FFF  0000
          *DATA   0FFF  0000
                          ;.Date: 9/13/82
                     ;.Author: Daniel E. Brown & Roger W. Chapman
                     ;.Title: DOUBLE DENSITY ROM : REV 1.41
                     ;.Comments:

                     ;       +--------------------------------+
                     ;       !                                !
                     ;       !     Double Density Monitor     !
                     ;       !                                !
                     ;       +--------------------------------+

0000  = 0000                 ORG     0               ;FWA of memory

      = 0007         CBELL:  =       'G'-40h         ;Ring the Bell
      = 000B         MCUP:   =       'K'-40h         ;Move cursor up
      = 000C         MCRIGH: =       'L'-40h         ;Move cursor right
      = 001A         VCLRS:  =       'Z'-40h         ;Clear and home cursor
      = 001E         VHOME:  =       '^'-40h         ;Home Cursor

      = 0023         VLOCK:  =       '#'             ;Lock Keyboard
      = 0022         VUNLK:  =       '"'             ;Unlock Keyboard
      = 003D         VCAD:   =       '='             ;Cursor Addressing
      = 0053         VSAD:   =       'S'             ;Screen Addressing
      = 0051         VINC:   =       'Q'             ;Insert Char
      = 0057         VDELC:  =       'W'             ;Delete char
      = 0045         VINL:   =       'E'             ;Insert line
      = 0052         VDELL:  =       'R'             ;Delete line
      = 0054         VCEOL:  =       'T'             ;Clear to end of line
      = 0029         VSHI:   =       ')'             ;Start half intensity
      = 0028         VEHI:   =       '('             ;end
      = 006C         VSUL:   =       'l'             ;Start underline
      = 006D         VEUL:   =       'm'             ;end
      = 0067         VSGH:   =       'g'             ;Start graphics
      = 0047         VEGH:   =       'G'             ;End

      = 0081         ROWOM:  =       081H
      = 0057         SI.MRST =       0_10_101_11b    ;Master reset
      = 0055         SI.S16: =       0_10_101_01b    ;Select 16x clock, xmit/rec
      = 0001         SI.RRDY =       01              ;Receiver ready
      = 0002         SI.TRDY =       02              ;Transmit ready
      = 0066         NMIA:   =       66h             ;NMI address
      = 1000         LVMEM:  =       128*32          ;Length of video memory
      = FFEA         VFLO:   =       -22             ;First line video offset
      = 0080         VLL:    =       128             ;Length of one video line
      = 0085         SCLFRE: =       0B5H            ;for DELAY routine
      = 000A         LF:     =       0Ah             ;^J, LF = Line Feed
      = 000D         CR:     =       0Dh             ;^M, CR = Carriage Return
      = 001B         ESC:    =       1Bh             ;^[, ESC = Escape
      = 007F         ERC:    =       7Fh             ;illegal key
      = 0008         BKS:    =       08h             ;BACKSPACE
      = 0009         TAB:    =       09h             ;TAB

      = 0003         KL_LEN: =       3               ;KEY LIST LENTH
      = 0002         KLE_LEN =       2               ;KEY LIST ENTRY LENTH
      = 0007         KL_USED =       7               ;KEYLIST ENTRY USED
```

Copyright © 1982 Osborne Computer Corporation

```
= 0006   KY_SRVD =     6          ;KEY SERVICED ONCE
= 0038   KROW_M: =     38H        ;ROW NUMBER MASK
= 0007   KCOL_M: =     7H         ;COL NUMBER MASK
= 0001   DB_CT:  =     1          ;DEBOUNCE COUNT
= 0018   IRPTCT: =     24         ;INITIAL REPEAT COUNT (400MS)
= 0006   SRPTCT: =     6          ;SECOND REPEAT COUNT (100MS)
= 0007   TOT_ROW =     7          ;TOTAL ROWS
= 0002   CTL_KY: =     2          ;COLUMN NUMBER OF CTL,ALPHA AND SHIFT KEYS
= 0003   ALPH_KY =     3
= 0004   SHFT_KY =     4
= 0003   SLD_RCT =     3          ;REPEAT COUNT FOR SLIDE KEYS (50MS APPROX)
= 0080   BRTBIT: =     80h        ;set brt/dim memory BRIGHT
= 0000   DIMBIT: =     00h        ;set brt/dim memory DIM
= 000A   NRETRY: =     10         ;NUMBER OF RETRYS

         ;MEMORY MAPPED I/O

= 2100   D.CMDR: =     02100H     ;Floppy disk DISK COMMAND REG   (WRITE)
= 2100   D.STSR: =     D.CMDR     ;STATUS REG              (READ)
= 2101   D.TRKR: =     D.CMDR+1   ;TRACK REG
= 2102   D.SECR: =     D.CMDR+2   ;SECTOR REG
= 2103   D.DATR: =     D.CMDR+3   ;DATA REG                (R/W)
= 2200   H.KEY:  =     02200H     ;Keyboard
= 2900   CPDRA:  =     02900H     ;Peripheral/Direction register A
= 2901   CCRA:   =     CPDRA+1    ;Control register A
= 2902   CPDRB:  =     CPDRA+2    ;Peripheral/Direction register B
= 2903   CCRB:   =     CPDRA+3    ;Control register B
= 2900   PA.DTA  =     CPDRA+0
= 2900   PA.DIR  =     PA.DTA
= 2901   PA.CTL  =     CPDRA+1
= 2902   PB.DTA  =     CPDRA+2
= 2902   PB.DIR  =     PB.DTA
= 2903   PB.CTL  =     CPDRA+3
= 2A00   H.SCTRL =     02A00H     ;Set control reg         (write)
= 2A00   H.SSTS: =     02A00H     ;Status reg              (read)
= 2A01   H.SXMT: =     02A00H+1   ;Transmit address
= 2A01   H.SREC: =     02A00H+1   ;Receive (read from address)
= 2C00   H.VID:  =     02C00H     ;Video memory controls

         ;RAM MEMORY LOCATIONS

= EF00   TEM:    =     0EF00H     ;(1) USED IN BOOT ROUTINES
= EF05   RTRY:   =     0EF05H     ;(1) RETRY COUNTER
= EF08   ROMRAM: =     0EF08H     ;(1) ROM/RAM FLAG
= EF09   DSTSB:  =     0EF09H     ;(1) SIX BYTES FOR DISK INFO
= EF0F   DMADR:  =     0EF0FH     ;(1) DISK DMA ADDRESS
= EF13   SEKDEL: =     0EF13H     ;(1) DISK STEP DELAY
= EF14   SAVSEC: =     0EF14H     ;(1) SECTOR
= EF15   SAVTRK: =     0EF15H     ;(1) TRACK
= EF17   SDISK:  =     0EF17H     ;(1) DISK
= EF50   HSTACT: =     0EF50H
= EF55   UNASEC: =     0EF55H
= EF56   LOGSEC: =     0EF56H
= EF59   KEYLCK: =     0EF59H     ;(1) KEYBOARD LOCKED CELL
= EF5A   CURS:   =     0EF5AH
= EF5E   LKEY:   =     0EF5EH     ;(1)
= EF60   ESCH:   =     0EF60H
= EF61   PIAAD:  =     0EF61H     ;(1)
= EF62   PIABD:  =     0EF62H     ;(1)
= EF6A   DACTVE: =     0EF6AH     ;(1) DISK ACTIVE FLAG
```

Copyright © 1982 Osborne Computer Corporation

```
= EF6B     BELCNT: =      0EF6BH        ;(1) IS BELL RINGING
= EF6C     LLIMIT: =      0EF6CH
= EF6E     LDTRK:  =      0EF6EH
= EF6F     IESTK:  =      0EF6FH        ;(2) SAVE STACK POINTER HERE
= EF99     ISTK:   =      0EF99H        ;(-40) INTERUPT STACK
= EFC1     ACIAD:  =      0EFC1H
= EFC1     ROMSTK: =      0EFC1H        ;(-40) ROM STACK
= EFC7     DSKSWP: =      0EFC7H        ;(1) DISK SWAPED CELL
= EFCA     NUMSEC: =      0EFCAH        ;(2) NUMBER OF SECTORS TO R/W
= EFCC     SEQ:    =      0EFCCH        ;(2) COUNTER
= EFD0     SAVTYP  =      0EFD0H        ;(1) DISK TYPE
= EFD1     R_WCOM: =      0EFD1H        ;(1) NUMBER OF SECTORS TO READ OR WRITE
= EFD2     CCPADR: =      0EFD2H        ;(2) This location is assigned in BIOS and filled in by loader in ROM
= EFD4     KEYLST: =      0EFD4H        ;(6) KEY LIST GOES HERE
= EFDA     SERFLG: =      0EFDAH        ;(1)
= EFDB     IE_ADRS =      0EFDBH        ;(1) device address
= EFDC     IE_CHAR =      0EFDCH        ;(1) IE inp char buffer
= EFDD     PIACTL  =      0EFDDH        ;(1)
= EFDE     PP.MODE =      0EFDEH        ;(1) Parallel port input, output, undefined
= EFEF     VRTOFF: =      0EFEFH        ;(1) LAST VERICAL OFFSET
= EFF0     INTBL:  =      0EFF0h        ;(16) interrupt vector table
= F000     FWAVM:  =      0F000H        ;FIRST ADDRESS OF MEMORY MAP
```

```
                    ;port ctl register constants.

 = 002A             PA.CDR   =        00101010b        ;to address port a direction
 = 002E             PA.CDT   =        00101110b        ;to address port a data and set
                                                       ;port a in input program handshake mode.
 = 0000             PB.CDR   =        00000000b        ;to address port b direction
 = 0004             PB.CDT   =        00000100b        ;to address port b data

                    ;direction register constants

 = 00FF             PA.DRO   =        0FFh             ;port a output mode
 = 0000             PA.DRI   =        00h              ;port a input mode
 = 00BF             PB.DR    =        0BFh             ;port b direction
 = 0002             PB.DTO   =        00000010b        ;port b data for output
 = 000B             PB.DTI   =        00001011b        ;port b data for input

 = 0040             PP.ORDY =         01000000b        ;output rdy bit in pib
 = 0080             PP.IRDY =         10000000b        ;input rdy bit in pia ctl reg
 = 0020             STRB     =        00100000b        ;strobe bit in port b

                    ;port modes

 = 0001             PP.OUT   =        1
 = 0002             PP.IN    =        2

                    ;IEEE control codes

 = 0040             IE_TALK =         40h              ;make talker
 = 005F             IE_UTLK =         5Fh              ;make untalk
 = 0020             IE_LSTN =         20h              ;make listener
 = 003F             IE_ULST =         3Fh              ;make unlisten
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
 = 0010          D.SEK:  =       010H            ;SEEK
 = 0020          D.STP:  =       020H            ;STEP
 = 0040          D.STPI: =       040H            ;STEP IN
 = 0060          D.STPO: =       060H            ;STEP OUT
 = 0080          D.RDS:  =       080H            ;READ SECTOR
 = 00A0          D.WRTS: =       0A0H            ;WRITE SECTOR
 = 00C0          D.RDA:  =       0C0H            ;READ ADDRESS
 = 00E0          D.RDT:  =       0E0H            ;READ TRACK
 = 00F0          D.WRTT: =       0F0H            ;WRITE TRACK
 = 00D0          D.FINT: =       0D0H            ;FORCE INTERRUPT

                 ;MACRO DEFINITION

                 ENADIM: MACRO
                         OUT     2
                         ENDM

                 DISDIM: MACRO
                         OUT     3
                         ENDM
```

Copyright © 1982 Osborne Computer Corporation

```
                        *[1]

0000                    START:
                        ;Initialize all dependent hardware.

0000                            PROC
0000  31C1EF                    LDK     SP,ROMSTK       ;SET STACK
0003  CD360E                    CALL    FORINT          ;DISABLE CURRENT DISK COMMAND, IF ANY

                        ;DISABLE DIM BIT

0006                            DISDIM

                        ;SET INTERUPTS

                                ;SET MODE 2 INTERUPTS

0008  ED5E        $             IM2

                                ;SET INTERRUPT REGISTER

000A  3EEF                      LDK     A,high INTBL
000C  ED47        $             MOV     I,A

                                ;SET KEYBOARD VECTOR

000E  21FC06                    LDK     HL,GKEY
0011  22F8EF                    STO     HL,INTBL+(4*2)  ;set keyboard interrupt

                                ;SET SERIAL VECTOR

                                ;SET IEEE VECTOR

                        *INITIALIZE MEMORY

                        ;CLEAR ALL BUT INTERUPT VECTORS

0014  2100EF                    LDK     HL,0EF00H       ;START
0017  06F0                      LDK     B,0F0H          ;LENTH

0019  AF                        XRA     A

001A  77                :CLOOP: STO     A,[HL]
001B  23                        INC     HL
001C  10FC ^001A$               DJNZ    :CLOOP          ;CLEAR LOOP

                        ;SET VALUES OF ONE

001E  3C                        INC     A               ;A = ONE
001F  3259EF                    STO     A,KEYLCK        ;indicate NOT locked

0022  4F                        MOV     C,A             ;FOR "IE.CO"

                        ;SET VALUES OF TWO

0023  3C                        INC     A               ;A = TWO
0024  3213EF                    STO     A,SEKDEL        ;set seek step rate FOR SEMIENS

0027  CD6800                    CALL    SPAO            ;set up for output (SAVES REG C)
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;Intialize IEEE port

002A  CD3609                 CALL    IE.CD          ;REG C=1 (SAVES REG C)

                        ;Set beginning line to 0

002D  0D                     DEC     C              ;C=0
002E  CD8600                 CALL    DPBD

                        ;set for -10 char position AND DOUBLE DENSITY

0031  0EEA                   LDK     C,VFLO
0033  CD7900                 CALL    DPAD

0036  3E80                   LDK     A,VLL
0038  326CEF                 STO     A,LLIMIT       ;set max line limit

                        ;Reset-Master clear the SIO (ACIA)

003B  0E55                   LK      C,SI.S16       ;select 16x clock for 1200 baud
003D  CDB20B                 CALL    SIRST          ;reset

                        *SIGN ON PROMPT

0040  119F01                 LDK     DE,IMSG
0043  CDB000                 CALL    OSTR           ;Output initial message

0046  FB                     EI                     ;ENABLE INTERUPTS

0047  CD7303                 CALL    CI             ;Get character
004A  FE1B                   CMP     ESC
004C  CA4D0F                 JZ      HCBOOT         ;IF COLD BOOT OFF OF HARD DISK

004F  21C7EF                 LK      HL,DSKSWP      ;disk swap cell
0052  FE0D                   CMP     CR
0054  CA5502                 JZ      CBOOT          ;if cold boot off of A

0057  34                     INC     [hl]           ;swap drives: A=B, B=A
0058  FE22                   CMP     '"'
005A  CA5502                 JZ      CBOOT          ;if cold boot off of B

005D  18A1 ^0000$            JR      START          ;LOOP
```

Copyright © 1982 Osborne Computer Corporation

```
                    ;INTERUPT VECTOR FOR RESET BUTTON (NON MASKABLE INTERUPT)

005F  = 0066            ORG     NMIA

0066  1898 ^0000$       JR      START           ;START
```

Copyright © 1982 Osborne Computer Corporation

```
0068                    SPAO:
                        ;Set PIA for output
                        ;ENTRY
                        ;NONE

                        ;EXIT
                                ;HL     =       H.VIO+2

                        ;CHANGE
                                ;HL

0068                            PROC

0068    21012C                  LDK     HL,H.VIO+1      ;H.VIO+1

006B    3603                    STO     3,[HL]          ;set data direction

006D    2B                      DEC     HL              ;H.VIO

006E    36FF                    STO     0FFh,[HL]       ;set all A lines as output

0070    23                      INC     HL              ;H.VIO+1
0071    23                      INC     HL              ;H.VIO+2
0072    23                      INC     HL              ;H.VIO+3

0073    3600                    STO     0,[HL]

0075    2B                      DEC     HL              ;H.VIO+2

0076    36FF                    STO     0FFH,[HL]       ;set all B lines as output

0078    C9                      RET
```

Copyright © 1982 Osborne Computer Corporation

```
0079                    OPAD:
                        ;Output data to pia A register
                        ;       PIA definition.
                        ;        7  6  5  4  3  2  1  0
                        ;       +--+--+--+--+--+--+--+--+
                        ;       !  horizontal offset !DD!
                        ;       +--+--+--+--+--+--+--+--+

                        *NOTE   The DD(double density) bit is inverted and the jumper must be installed on the pc board.
                        ;       If the 0 bit is LOW double density is set if it is HIGH single density is set.

                        *NOTE   Bit 0 of "PIAAD" :
                        ;       set     =       single density
                        ;       reset   =       double density

                        *NOTE*
                        ;SAVE HL

                        ;ENTRY
                        ;C      =       data

                        ;EXIT
                        ;NONE
0079                            PROC

0079  3E07                      LDK     A,4+3
007B  32012C                    STO     A,H.VID+1

007E  79                        MOV     A,C
007F  3261EF                    STO     A,PIAAD
0082  32002C                    STO     A,H.VID         ;send data
0085  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation


```
0086                    OPBD:
                        ;Output data to pia B register
                        ;        PIA definition.
                        ;          7  6  5  4  3  2  1  0
                        ;        +--+--+--+--+--+--+--+--+
                        ;        !D1!D0!^G! vert  offset !
                        ;        +--+--+--+--+--+--+--+--+

                        ;ENTRY
                        ;C       =       data

                        ;EXIT
                        ;NONE
0086                             PROC

0086   3E04                      LDK     A,4
0088   32032C                    STO     A,H.VIO+3

008B   79                        MOV     A,C
008C   3262EF                    STO     A,PIABD
008F   32022C                    STO     A,H.VIO+2       ;send data
0092   C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0093  113316      ROMJP1: LDK    DE,1633h        ;offset in bios jump table
0096  1803 ^009B$         JR     BIOJP

0098  113616      ROMJP2: LDK    DE,1636h        ;offset in bios jump table

009B  2AD2EF      BIOJP:  LD     HL,CCPADR
009E  19                  ADD    HL,DE           ;form address
009F  E9                  JMP    [HL]
```

```
00A0  000A         EMBOOT: DB      CR,LF
00A2  424F4F5420           DB      'BOOT ERROR'
00AC  A0                   DC      ' '

00AD               EBOOT:
                   ;BOOT ERROR MESSAGE ROUTINE
                   ;ENTRY
                   ;NONE

00AD                       PROC

00AD  11A000               LDK     DE,EMBOOT       ;HERE ON BOOT ERROR

                   ;FALL THROUGH TO OSTR
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0080                 OSTR:
                     ;OUTPUT STRING TO CONSOLE
                     ;NOTE:   OSTR RECOGNIZES 7F AS AN ESCAPE SEQUENCE TO REPEAT CHAR N TIMES.  FORMAT IS: 7F, REPEAT COUNT, CHAR
                     ;ENTRY
                     ;DE      =       FWA OF SOURCE

                     ;EXIT
                     ;NONE
0080                          PROC

0080  1A                      LD      A,[DE]
0081  B7                      OR      A
0082  F5                      PUSH    AF

0083  E67F                    AND     07FH
0085  FE7F                    CMP     07FH
0087  4F                      MOV     C,A
0088  200C ^00C6$             JRNZ    :4              ;IF NOT REPEAT

008A  13                      INC     DE
008B  1A                      LD      A,[DE]
008C  3D                      DEC     A
008D  47                      MOV     B,A             ;REPEAT COUNT
008E  13                      INC     DE
008F  1A                      LD      A,[DE]          ;GET REPEAT CHAR
00C0  4F                      MOV     C,A

00C1  CDE003         :2:      CALL    COUT            ;OUTPUT CHAR
00C4  10FB ^00C1$             DJNZ    :2              ;IF NOT DONE

00C6  CDE003         :4:      CALL    COUT            ;OUTPUT IT
00C9  13                      INC     DE

00CA  F1                      POP     AF
00CB  F28000                  JP      OSTR            ;IF NOT DONE

00CE  C9                      RET
```

Copyright ©1982 Osborne Computer Corporation

```
00CF                  DELAY:
                      ;'N' Milliseconds
                      ;ENTRY
                      ;A      =      Number of Milliseconds to delay
                      ;SCLFRE =      (Freq/1000)/25

                      ;EXIT
                      ;NONE
00CF                          PROC

00CF  4F                      MOV    C,A
00D0  3E35            :1:     LDK    A,SCLFRE

00D2  3D              :MLOOP: DEC    A              ;(4 tics)
00D3  40                      MOV    B,B            ;(4 tics)
00D4  49                      MOV    C,C            ;(4 tics)
00D5  20FB ^00D2$             JRNZ   :MLOOP         ;(10 tics) If 1 ms not elapsed

00D7  0D                      DEC    C
00D8  20F6 ^00D0$             JRNZ   :1             ;If requested msec not done

00DA  C9                      RET

                      ;SCLFRE =      2000/22 ;Z80, 2mhz
                      ;...defined in OCCTXT.ast
```

Copyright © 1982 Osborne Computer Corporation


```
00DB  = 0100            ORG     100h

              ;ROM JUMP TABLE
              ;         CBIOS   =       Jmps used mainly by CBIOS
              ;         SC      =       Jmps used mainly by SuperCalc

0100  C35502            JMP     CBOOT           ;CBIOS  cold boot
0103  C39C02            JMP     WBOOT           ;CBIOS  warm boot
0106  C36603            JMP     SKEY            ;CBIOS  keyboard status
0109  C37303            JMP     CI              ;CBIOS  keyboard input
010C  C3E003            JMP     COUT            ;CBIOS  console output
010F  C3D90B            JMP     LIST            ;CBIOS  list output
0112  C3D90B            JMP     LIST            ;CBIOS  punch output
0115  C3BF0B            JMP     READER          ;CBIOS  reader input

              ;         Disk I/O

0118  C3020C            JMP     RDRV            ;CBIOS  HOME
011B  C9                RET                     ;CBIOS  SELECT DISK
011C  00                NOP
011D  00                NOP
011E  C3300D            JMP     READ            ;       READ SECTOR
0121  C3370D            JMP     WRITE           ;       WRITE SECTOR
0124  C3A50D            JMP     RADR            ;       READ SECTOR ANY SECTOR HEADER
0127  C31C0C            JMP     RSEC            ;CBIOS  DISK SECTOR READ
012A  C3230C            JMP     WSEC            ;CBIOS  DISK SECTOR WRITE
012D  C3D00B            JMP     SLST            ;CBIOS  List device status
0130  C3690C            JMP     SENDEN          ;CBIOS  SENSE THE DENSITY OF DRIVE
0133  C39300            JMP     ROMJP1          ;CBIOS
0136  C39800            JMP     ROMJP2          ;CBIOS
0139  C30F0F            JMP     FORMAT          ;CBIOS  FORMATING ROUTINE

              ;         IEEE

013C  C3B20B            JMP     SIRST           ;CBIOS  SIO reset
013F  C33609            JMP     IE.CO           ;CBIOS  IEEE    Control Out
0142  C37909            JMP     IE.SI           ;CBIOS          Status In
0145  C38C09            JMP     IE.GTS          ;CBIOS          Go To Standby
0148  C39809            JMP     IE.TC           ;CBIOS          Take Control
014B  C3D809            JMP     IE.OIM          ;CBIOS          Output Interface Message
014E  C3F809            JMP     IE.ODM          ;CBIOS          Output Device Message
0151  C3300A            JMP     IE.IDM          ;CBIOS          Input Device Message
0154  C3990A            JMP     IE.PP           ;CBIOS          Parallel Poll

              ;         SuperCalc

0157  C3CE06            JMP     VLDDR           ;SC     VIDEO BLOCK MOVE DEC
015A  C3E206            JMP     VLDIR           ;SC     VIDEO BLOCK MOVE INC
015D  C3C606            JMP     STODIM          ;SC     STO reg B IN [HL]

              ;         DISK I/O

0160  C3F00E            JMP     DMAWRT          ;       DMA WRITE TO CONTROLER
0163  C3DA0E            JMP     DMARD           ;       DMA READ FROM CONTROLER
0166  C3D80C            JMP     HOME            ;       HOME DISK DRIVE
0169  C3FA0C            JMP     SEEK            ;       SEEK TO TRACK
016C  C3090D            JMP     STEP            ;       STEP SAME DIRECTION
016F  C30D0D            JMP     STEPIN          ;       STEP IN
0172  C3110D            JMP     STEPOUT         ;       STEP OUT
```

```
0175 C3360E          JMP     FORINT          :       FORCE INTERUPT
0178 C3E80D          JMP     READTRK         :       READ TRACK
017B C3010E          JMP     FMTTRK          :       Format one track
017E C3710E          JMP     SELDRV          :       SELECT DRIVE
0181 C3E80B          JMP     ACISTAT         ;CBIOS  SERIAL PORT STATUS
0184 C3D10C          JMP     SCTRKR          :       SET TRACK REGESTER IN CONTROLER CHIP WITH VALUE IN SAVTRK

0187 C3B10A          JMP     IEINSTAT        ;CBIOS  IEEE
018A C3AE0A          JMP     IEOSTAT         ;CBIOS  IEEE
018D C3E10A          JMP     IEINP           ;CBIOS  IEEE
0190 C3ED0A          JMP     IEOUT           ;CBIOS  IEEE

0193 C3780B          JMP     PISTAT          ;CBIOS  IEEE
0196 C36C0B          JMP     POSTAT          ;CBIOS  IEEE
0199 C38E0B          JMP     PARINP          ;CBIOS  IEEE
019C C39D0B          JMP     PAROUT          ;CBIOS  IEEE
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
019F  1A0A0A0A0A  IMSG:   DB    'Z'-40h,lf,lf,lf,lf
01A4  7F0B20              DB    07Fh, 11, ' '
01A7  1B67                DB    ESC,VSGH
01A9  11                  DB    'Q'-40h
01AA  7F1817              DB    07Fh, 24, 'W'-40h
01AD  05                  DB    'E'-40h
01AE  1B47                DB    ESC,VEGH

01B0  0D0A                DB    cr,lf
01B2  7F0B20              DB    07Fh, 11, ' '
01B5  1B67011B47          DB    ESC,VSGH,1, ESC,VEGH
01BA  2020202020          DB    '      '
01C1  1B6C                DB    ESC,'l'
01C3  4F53424F52          DB    'OSBORNE 1'
01CC  1B6D                DB    ESC,'m'
01CE  2020202020          DB    '         '
01D6  1B67041B47          DB    ESC,VSGH,4, ESC,VEGH

01DB  0D0A                DB    cr,lf
01DD  7F0B20              DB    07Fh, 11, ' '
01E0  1B67                DB    ESC,VSGH
01E2  01                  DB    'A'-40h
01E3  7F1820              DB    07Fh,24,' '
01E6  04                  DB    'D'-40h
01E7  1B47                DB    ESC,VEGH

01E9  0D0A                DB    cr,lf
01EB  7F0B20              DB    07Fh, 11, ' '
01EE  1B67                DB    ESC,VSGH
01F0  01                  DB    'A'-40h
01F1  1B47                DB    ESC,VEGH
01F3  1B29                DB    ESC,')'
01F5  2020526576          DB    '  Rev 1.41 (c) 1982 OCC '
020D  1B28                DB    ESC,'('
020F  1B67                DB    ESC,VSGH
0211  04                  DB    'D'-40h
0212  1B47                DB    ESC,VEGH

0214  0D0A                DB    cr,lf
0216  7F0B20              DB    07Fh, 11, ' '
0219  1B67                DB    ESC,VSGH
021B  1A                  DB    'Z'-40h
021C  7F1818              DB    07Fh, 24, 'X'-40h
021F  03                  DB    'C'-40h
0220  1B47                DB    ESC,VEGH

0222  0D0A0A0A            DB    cr,lf,lf,lf
0226  7F0420              DB    07Fh, 4, ' '
0229  496E736572          DB    'Insert disk in Drive '
023E  1B6C                DB    ESC,'l'
0240  41                  DB    'A'
0241  1B6D                DB    ESC,'m'
0243  20616E6420          DB    ' and press RETURN'
0254  AE                  DC    '.'
```

```
                        *[2]

0255                    CBOOT:
                        ;LOAD ALL THE OPERATING SYSTEM INCLUDING THE CBIOS
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;A      =       DRIVE TO BOOT FROM

0255                            PROC

                        *SET "SAVTYP"

0255  CD020C            :1:     CALL    RDRV            ;HOME DRIVE
0258  CD690C                    CALL    SENDEN          ;DETERMINE DENSITY
025B  2805 ^0262$               JRZ     :2              ;IF GOOD

025D  CDAD00            :ERR:   CALL    EBOOT           ;PRINT ERROR
0260  18F3 ^0255$               JR      :1

                        *READ AND SET FBA OF CCP

0262  210000            :2:     LDK     HL,0D000H
0265  220FEF                    STO     HL,DMADR        ;SET DMA

0268  E5                        PUSH    HL

0269  3E01              :3:     LDK     A,1
026B  3214EF                    STO     A,SAVSEC        ;SET SECTOR
026E  47                        MOV     B,A
026F  CD1C0C                    CALL    RSEC            ;READ SECTOR ONE
0272  2805 ^0279$               JRZ     :4              ;IF GOOD

0274  CDAD00                    CALL    EBOOT           ;PRINT ERROR
0277  18F0 ^0269$               JR      :3

                        *CHECK FIRST TWO BYTES OF THE CCP

0279  E1                :4:     POP     HL              ;HL = 0D000H

027A  7E                        LD      A,[HL]          ;FIRST BYTE
027B  FEC3                      CMP     0C3H
027D  20DE ^025D$               JRNZ    :ERR            ;IF NOT THE SAME

027F  23                        INC     HL              ;HL = 0D001H
0280  7E                        LD      A,[HL]          ;SECOND BYTE
0281  FE5C                      CMP     05CH
0283  20D8 ^025D$               JRNZ    :ERR            ;IF NOT THE SAME

0285  23                        INC     HL              ;HL = 0D002H

                        ;SET LOAD ADDRESS

0286  7E                        LD      A,[HL]          ;get ccp address/100h + 3
0287  D603                      SUB     3
0289  2E00                      LDK     L,0
028B  67                        MOV     H,A
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        *SET NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT

028C  063C                  LDK     B,60            ;CCP/BDOS/CBIOS

                        *READ SYSTEM

028E  E5                    PUSH    HL              ;SAVE FWA FOR "CCPADR"
028F  CDA102                CALL    BCPM            ;boot system
0292  E1                    POP     HL

                        *JUMP SYSTEM

0293  AF                    XRA     A               ;DRIVE BOOTED FROM
0294  2202EF                STO     HL,CCPADR       ;SET "CCPADR"

0297  110016                LDK     DE,1600h        ;offset for bios
029A  19                    ADD     HL,DE           ;address of bios in hl
029B  E9                    JMP     [HL]            ;enter com
```

```
029C                    WBOOT:
                        ;LOAD ONLY THE CCP AND THE BDOS FROM DRIVE A
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        *NOTE*
                        *:      THIS ROUTINE DOES NOT EXIT. IT ONLY SETS PARAMITERS FOR BCPM:.

                        *:B     =       NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT
                        *:HL    =       DMA ADDR FOR CCP

029C                            PROC

029C  062C                      LDK     B,44            ;CCP/BDOS and don't read CBIOS
029E  2AD2EF                    LD      HL,CCPADR
```

Copyright © 1982 Osborne Computer Corporation

```
02A1                    BCPM:
                        ;LOAD ALL OR PART OF CPM FROM THE DISK

                        *NOTE*
                        ;       This loader will load single or double density and any number of sectors per track or bytes
                        ;per sector. TEM is not zero if there are an uneven number of sectors to read. If this is true the
                        ;last sector is read into a temporary buffer and the part needed is moved into the memory.

                        ;ENTRY
                        ;B      =       NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT
                        ;HL     =       DMA ADDR FOR CCP

                        ;EXIT
                        ;NONE

02A1                            PROC

                        *SET "SDISK", "DMADR" AND "SAVSEC"

02A1  220FEF                    STO     HL,DMADR        ;SET DMA

02A4  AF                        XRA     A
02A5  3217EF                    STO     A,SDISK         ;BOOT ONLY FROM DRIVE A
02A8  3200EF                    STO     A,TEM           ;MAKE TEM ZERO

02AB  3C                        INC     A
02AC  3214EF                    STO     A,SAVSEC        ;set sector

                        *SET "SAVTYP" AND GET NUMBER OF SECTORS PER TRACK

02AF  C5                        PUSH    BC              ;SAVE NUMBER OF 128 BLOCKS

02B0  CD020C            :RL1:   CALL    RDRV            ;HOME DRIVE
02B3  CD690C                    CALL    SENDEN          ;DETERMINE DENSITY
02B6  2805 ^02BD$               JRZ     :1              ;IF GOOD

02B8  CDAD00                    CALL    EBOOT           ;PRINT ERROR
02BB  18F3 ^02B0$               JR      :RL1

02BD  D1                :1:     POP     DE              ;D=NUMBER OF 128 BYTE BLOCKS
02BE  C5                        PUSH    BC              ;SAVE NUMBER OF SECTORS IN ONE TRACK

                        *SET NUMBER OF SECTORS TO READ

02BF  3AD0EF                    LD      A,SAVTYP
02C2  CB3F       $              SRL     A
02C4  CB3F       $              SRL     A
02C6  E603                      ANI     0000_0011B      ;A=NUMBER OF BYTES IN ONE SECTOR(0-3)
02C8  2818 ^02E2$               JRZ     :2              ;IF 128 BYTES SECTORS

                                ;GET NUMBER TO DEVIDE BY

02CA  47                        MOV     B,A             ;B=NUMBER OF BYTES IN ONE SECTOR(1-3)
02CB  3E01                      LDK     A,1

02CD  CB27       $ :1LOOP:      SLA     A               ;TIMES TWO
02CF  10FC ^02CD$               DJNZ    :1LOOP

02D1  47                        MOV     B,A             ;NUMBER TO DIVIDE BY
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
02D2  7A                        MOV   A,D        ;A=NUMBER OF 128 BYTE BLOCKS
02D3  1600                      LDK   D,0

02D5  90              :2LOOP:   SUB   B          ;SUBTRACK WITH DIVISOR
02D6  08          $             EX    AF         ;SAVE FLAGS
02D7  14                        INC   D          ;COUNT
02D8  08          $             EX    AF         ;RESTORE FLAGS
02D9  2807 ^02E2$               JRZ   :2         ;IF REZULT IS ZERO (NO PARTIAL SECTORS)

02DB  30F8 ^02D5$               JRNC  :2LOOP     ;LOOP

02DD  ED44        $             NEG   A          ;2 COMP
02DF  3200EF                    STO   A,TEM      ;SAVE REMAINDER AND INDICATE A PARTIAL SECTOR

02E2  C1              :2:       POP   BC         ;B=NUMBER OF SECTORS IN ONE TRACK
02E3  4A                        MOV   C,D        ;C=NUMBER OF SECTORS TO READ
```

Copyright © 1982 Osborne Computer Corporation

```
                  *READ SYSTEM

02E4  AF                     XRA    A               ;A=0
02E5  3215EF        :TLOOP:  STO    A,SAVTRK        ;SET TRACK

                             ;CHECK FOR ALL SECTORS READ

02E8  79                     MOV    A,C             ;SECTORS TO READ
02E9  B7                     ORA    A
02EA  2008 ^02F4$            JRNZ   :3              ;IF C IS NOT ZERO CONTINUE

                                    ;CHECK FOR NO PARCIAL SECTOR

02EC  3A00EF                 LD     A,TEM
02EF  B7                     ORA    A
02F0  285B ^034D$            JRZ    :9              ;STOP IF C=0 AND TEM=0

02F2  1826 ^031A$            JR     :7              ;READ PARCIAL SECTOR

                             ;UPDATE NUMBER OF SECTORS LEFT TO READ

02F4  90            :3:      SUB    B               ;SUBTRACK SECTORS IN ONE TRACK
02F5  3002 ^02F9$            JRNC   :4              ;A>B MORE THAN ONE TRACK LEFT TO READ

                             ;IF THIS IS LAST TRACK ZERO NUMBER OF SECTORS LEFT TO READ

02F7  41                     MOV    B,C             ;READ ALL THE REMAINING SECTORS
02F8  AF                     XRA    A               ;THIS WILL ZERO REG C

                             ;CHECK FOR NONZERO VALUE IN TEM AND THE LAST SECTOR TO READ

02F9  4F            :4:      MOV    C,A             ;SAVE REMAINING SECTORS TO READ
02FA  3A00EF                 LD     A,TEM
02FD  B7                     ORA    A
02FE  2807 ^0307$            JRZ    :5              ;IF TEM IS ZERO SKIP THIS

0300  AF                     XRA    A
0301  B1                     ORA    C
0302  2003 ^0307$            JRNZ   :5              ;IF REG C IS NOT ZERO SKIP THIS(NOT LAST TRACK)

                                    ;READ ONE LESS THAN THE LAST SECTOR

0304  05                     DEC    B               ;B=B-1
0305  2813 ^031A$            JRZ    :7              ;IF ONLY ONE SECTOR LEFT TO READ

                             ;READ ONE TRACK

0307  CD1C0C        :5:      CALL   RSEC            ;READ (BC IS SAVED)
030A  2805 ^0311$            JRZ    :6              ;IF GOOD

030C  CDAD00                 CALL   EBOOT           ;REPORT ERROR
030F  18F6 ^0307$            JR     :5

                             ;UPDATE DMA

0311  220FEF        :6:      STO    HL,DMADR        ;SET DMA

                             ;UPDATE TRACK
```

```
0314  3A15EF          LD     A,SAVTRK
0317  3C              INC    A
0318  18CB ^02E5$     JR     :TLOOP          ;TRACK LOOP
```

Copyright ©1982 Osborne Computer Corporation

```
                    ;READ A PARTIAL SECTOR

031A  E5            :7:      PUSH   HL            ;SAVE ADDRESS TO WRITE TO

031B  2180EA                 LDK    HL,0EA80H     ;ADDRESS OF HOST BUFFER IN BIOS
031E  220FEF                 STO    HL,DMADR      ;SET DMA

                             ;SET TRACK IF NEEDED

0321  2804 ^0327$            JRZ    :10           ;IF B=0 THEN SAVTRK WAS NOT INCREMENTED

0323  2115EF                 LDK    HL,SAVTRK
0326  35                     DEC    [HL]          ;SAVTRK = SAVTRK - 1

                             ;SET THE SECTOR TO B + 1

0327  04            :10:     INC    B
0328  78                     MOV    A,B
0329  3214EF                 STO    A,SAVSEC      ;SET SECTOR

                             ;READ SECTOR INTO HOST BUFF

032C  0601                   LDK    B,1
032E  CD1C0C        :RL2:    CALL   RSEC          ;READ ONE SECTOR
0331  2805 ^0338$            JRZ    :8            ;IF GOOD

0333  CDAD00                 CALL   EBOOT         ;REPORT ERROR
0336  18F6 ^032E$            JR     :RL2

                             ;SET NUMBER OF BYTES TO TRANSFER

0338  3A00EF        :8:      LD     A,TEM
033B  47                     MOV    B,A           ;B=NUMBER OF 128 BYTE BLOCK TO TRANSFER
033C  210000                 LDK    HL,0
033F  118000                 LDK    DE,128

0342  19            :3LOOP:  ADD    HL,DE
0343  10FD ^0342$            DJNZ   :3LOOP

0345  E5                     PUSH   HL
0346  C1                     POP    BC            ;BC=NUMBER OF BYTES TO TRANSFER

                             ;TRANSFER BYTES

0347  2180EA                 LDK    HL,0EA80H     ;SOURCE
034A  D1                     POP    DE            ;DESTINATION
034B  EDB0        $          LDIR                 ;MOVE
```

Copyright © 1982 Osborne Computer Corporation

```
                      *CLEAR BUFFERS SET PARR. AND RETURN TO SYSTEM

034D  2150EF          :9:     LDK     HL,HSTACT

0350  3600                    STO     0,[hl]             ; 1st byte

0352  010600                  LDK     BC,(LOGSEC-HSTACT)
0355  1151EF                  LDK     DE,HSTACT+1        ;DE = HL + 1

0358  ED30          $         LDIR                       ; overlapping move

035A  3EFF                    LDK     A,0FFh
035C  3255EF                  STO     A,UNASEC

035F  3E7F                    LDK     A,VLL-1
0361  326EEF                  STO     A,LDTRK            ;set other drive NOT int

0364  AF                      XRA     A                  ;Clear error indicator
0365  C9                      RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        *[3]

0366                    SKEY:
                        ;Get status of keyboard

                        ;EXIT
                        ;Cbit set if no data ready

0366  3A59EF                    LD      A,KEYLCK
0369  B7                        OR      A
036A  C8                        RZ                      ;if locked keyboard

036B  3A5EEF                    LD      A,LKEY
036E  B7                        ORA     A               ;CHECK FOR ZERO
036F  C8                        RZ

0370  F6FF                      ORI     0FFH            ;IF NOT ZERO MAKE 0FFH
0372  C9                        RET                     ;IF DATA
```

Copyright © 1982 Osborne Computer Corporation

```
0373                  CI:
0373                  RKEY:
                      ;Read next key from keyboard

                      ;EXIT
                      ;A       =       last key

0373                          PROC

0373  CD6603                  CALL    SKEY
0376  28FB ^0373$             JRZ     RKEY            ;if NO data

0378  F3                      DI
0379  3A5EEF                  LD      A,LKEY          ;GET CHARACTER
037C  4F                      MOV     C,A
037D  AF                      XRA     A
037E  325EEF                  STD     A,LKEY          ;clear key from hold
0381  79                      MOV     A,C
0382  FB                      EI

0383  C9                      RET
```

Copyright © 1982 Osborne Computer Corporation

```
                ;Bit definitions for ESCH flag byte
                ;Note Bit 7 is currently free.

 = 0020         EF_SCR: =       32              ;B5= Screen/Cursor Addressing
 = 0010         EF_ADR: =       16              ;B4= expecting address-chr
 = 0008         EF_ESC: =       8               ;B3=$last char was ESC
 = 0004         EF_UN:  =       4               ;B2= Underline mode
 = 0002         EF_HA:  =       2               ;B1= Half Intensity mode
 = 0001         EF_GR:  =       1               ;B0= Graphics mode
 = 0007         EF_MSK: =       EF_UN+EF_HA+EF_GR       ;Mask to get mode.
```

```
                        ;Vector (branch) table for video output mode selection
                        ;controlled by ESCH mode

0384                    ESCHTB:
0384  3704                      DW      VNORM           ;0 Normal mode
0386  5405                      DW      VGRAPH          ;1 Graphics mode
0388  6504                      DW      VHALF           ;2 Half intensity mode
038A  6904                      DW      VHA_GR          ;3 Half and graphics
038C  5704                      DW      VUNDER          ;4 Underline mode
038E  5B04                      DW      VUN_GR          ;5 Under and graphics
0390  5F04                      DW      VUN_HA          ;6 Under and half intensity
0392  6304                      DW      VUN_HA_GR       ;7 Under and half and graphics
```

Copyright ©1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0394                VALIDE:
                    ;Valid ESC-Sequence Table
                    ;3 bytes per entry:ascii char , "DW"-Vector. no. of entries is VALETS
                    ;Following body of table is 2 byte No-Match adrs

0394  3D                    DB    VCAD    ! DW   ESCCAD  ;Cursor Addressing
0397  53                    DB    VSAD    ! DW   ESCSAD  ;Screen Addressing
039A  67                    DB    VSGH    ! DW   ESCSGR  ;Set graphics mode
039D  47                    DB    VEGH    ! DW   ESCCGR  ;Clr graphics mode
03A0  29                    DB    VSHI    ! DW   ESCSHA  ;Set half int. mode
03A3  28                    DB    VEHI    ! DW   ESCCHA  ;Clr half int. mode
03A6  6C                    DB    VSUL    ! DW   ESCSUN  ;Set underline mode
03A9  6D                    DB    VEUL    ! DW   ESCCUN  ;Clr underline mode

03AC  1A                    DB    VCLRS   ! DW   ESCZZ   ;Clear screen to blanks
03AF  51                    DB    VINC    ! DW   EINSRT  ;Insert char
03B2  57                    DB    VDELC   ! DW   EDELC   ;Delete char
03B5  45                    DB    VINL    ! DW   ESCEE   ;Insert line
03B8  52                    DB    VDELL   ! DW   ESCRR   ;Delete line
03BB  54                    DB    VCEOL   ! DW   EEOL    ;Clear to end of line
03BE  23                    DB    VLOCK   ! DW   ESCLCK  ;Lock Keyboard
03C1  22                    DB    VUNLK   ! DW   ESCULK  ;Unlock Keyboard

03C4  1A04          :end:   DW    COUT2   ;No Match exit

                    ;Ignore char upon undefined ESC-Sequence (to treat undefined char after ESC as a regular
                    ;data char, should go to COUT2).

      = 0010        VALETS: =     (:end-VALIDE)/3         ;# of entries in table
```

Copyright © 1982 Osborne Computer Corporation


```
03C6                    VALIDC:
                        ;Valid control character table
                        ;3 bytes per entry: Ascii char , "DW"- Vector no. of entries is VALCTS
                        ;Following body of table is 2 byte No-Match adrs

03C6  0D                        DB      CR       ! DW    VC_CR    ;carriage return routine
03C9  0A                        DB      LF       ! DW    VC_LF    ;line feed
03CC  08                        DB      BKS      ! DW    VC_BKS   ;back space
03CF  0C                        DB      MCRIGH   ! DW    VC_MCRT  ;move cursor right
03D2  0B                        DB      MCUP     ! DW    VC_MCUP  ;move cursor up
03D5  07                        DB      CBELL    ! DW    VC_BEL   ;Ring bell
03D8  1A                        DB      VCLRS    ! DW    VC_CLRS  ;clear screen
03DB  1E                        DB      VHOME    ! DW    VC_HOME  ;Cursor Home

03DE  7B05                      DW      VOUT97                    ;No match--ignore undef control char

      = 0008            VALCTS: = ((*-2)-VALIDC)/3                ;Number of valid entries
```

Copyright © 1982 Osborne Computer Corporation

```
03E0                    COUT:
                        ;General output routine to Video Screen

                        ;ENTRY
                        ;C       =        Character
                                ;CURS    =        Cursor
                                ;ESCH    =        Flag+Mode

                        ;CURS & ESCH updated, A=Character
                        ;(bc, de, hl preserved)

                        ;        ESCH is flag + mode byte as follows
                        ;        =00      Normal mode & Last chr Esc flag false
                        ;        =08      Normal mode & Last chr Esc flag True
                        ;        =01,02,04 Mode is Graphics, Half, or Under, respectively and Last chr Esc flag is False.
                        ;        =3,5,6,7         As above, but mode is combination
                        ;        =9-15            Last chr Esc flag True;otherwise like 1-7.

03E0                            PROC

                        ;RESET VETICAL OFFSET WITH VRTOFF

03E0  F5                        PUSH    AF
03E1  C5                        PUSH    BC
03E2  3A62EF                    LD      A,PIABD         ;PRESENT VALUE
03E5  E6E0                      AND     11100000B       ;HOUSEKEEPING
03E7  47                        MOV     B,A
03E8  3AEFEF                    LD      A,VRTOFF        ;LAST VERTICAL OFFSET
03EB  E61F                      AND     00011111B       ;ONLY VIDIO
03ED  B0                        ORA     B               ;ADD HOUSEKEEPING
03EE  4F                        MOV     C,A
03EF  CD8600                    CALL    OPBD            ;SET OFFSET
03F2  C1                        POP     BC
03F3  F1                        POP     AF

03F4  E5                        PUSH    HL
03F5  D5                        PUSH    DE
03F6  C5                        PUSH    BC
03F7  2A5AEF                    LD      HL,CURS         ;HL will usually be cursor/
03FA  3A60EF                    LD      A,ESCH
03FD  47                        MOV     B,A             ;B will be ESCH for a while
03FE  E608                      AND     EF_ESC          ;test flag bit
0400  2023 ^0425$               JRNZ    PSTESC          ;IF last chr was ESC

                        ;Current chr is NOT ESCaped. Is this chr ESC?

0402  79                        MOV     A,C             ;Chr
0403  FE1B                      CMP     ESC
0405  78                        MOV     A,B             ;(A=ESCH)
0406  2815 ^041D$               JRZ     :ESC            ;if this chr = ESC
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;Here with A=3  =       ESCH

0408  E5                :out:   PUSH    HL
0409  218403                    LDK     HL,ESCHT3
040C  E607                      AND     EF_MSK          ;Mode bits only
040E  87                        ADD     A,A             ;Times two
040F  5F                        MOV     E,A
0410  1600                      LDK     D,0             ;DE = offset
0412  19                        ADD     HL,DE           ;HL = tbl addrs

0413                    VECTOR:                         ;entry point. note hl on stack.
0413  7E                        LD      A,[HL]          ;1st byte (low order adrs)
0414  23                        INC     HL
0415  66                        LD      H,[HL]          ;2nd byte (hi order adrs)
0416  6F                        MOV     L,A             ;HL=adrs from table
0417  E3                        XTHL                    ;Restore hl from stack stack=tbl adrs
0418  79                        MOV     A,C             ;Chr. note B=ESCH byte value
0419  C9                        RET                     ;enter routine per table adrs.
```

```
041A                COUT2:
041A  78                       MOV    A,B           ;recall ESCH value
041B  18EB ^0408$              JR     :out          ;output chr per current settings

041D                :ESC:
                    ;Current chr is ESC. Set flag and exit

041D  F608                     OR     EF_ESC        ;indicate last char= ESC
041F  3260EF                   STO    A,ESCH
0422  C37B05                   JMP    VOUT97        ;Exit
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0425                    PSTESC:
                        ;Last chr was ESC
                        ;Entry
                        ;A       =       EF_ESC
                        ;B       =       ESCH
                        ;C       =       Char to output
                        ;HL      =       curs

0425                            PROC

0425  CB60       $              BIT     4,B             ;is this chr really an address?
0427  2075 ^049E$               JRNZ    SETXY           ;...if chr is part of an addr

                        ;no cursor/screen addressing in effect:

0429  A8                        XOR     B               ;Clr EF_ESC bit (for next time)
042A  47                        MOV     B,A             ;Set up B = ESCH byte value.
042B  3260EF                    STO     A,ESCH
042E  E5                        PUSH    HL              ;save Curs
042F  219403                    LDK     HL,VALIDE       ;Branch table adrs
0432  1E10                      LDK     E,VALETS        ;Table size
0434  79                        MOV     A,C             ;Chr to A
0435  1815 ^044C$               JR      LOOKUP3         ;Go to routine to branch per tbl
```

Copyright © 1982 Osborne Computer Corporation

```
0437                  VNORM:
                      ;NORMAL mode character processing.
                      ;ENTRY
                      ;A        =       char to output
                      ;HL       =       curs

0437  FE20                      CMP     ' '
0439  3808 ^0446$               JRC     :2              ;IF control chr

043B  F3              VBRIGH:   DI
043C                            ENADIM          ;9th bit memory
043E  3680                      STB     BRTBIT,[hl]     ;set this chr BRIGHT
0440                            DISDIM
0442  FB                        EI

0443  C35E05                    JMP     VOUT80

0446  E5              :2:       PUSH    HL              ;Save Curs
0447  21C603                    LDK     HL,VALIDC       ;Branch table adrs
044A  1E0B                      LDK     E,VALCTS        ;Table size

                      ;         JMP     LOOKJPB         ;Scan table of valid control chrs and branch to appropriate routine.
```

Copyright © 1982 Osborne Computer Corporation

```
044C                    LOOKUPB:
                        ;Logic to scan 3 byte branch table
                        ;       NOT a subroutine---do not CALL.
                        ;ENTRY
                        ;HL     =       1st byte of table (match code)
                                ;(2nd,3rd bytes = branch adrs)
                                ;(table repeats [3 byte entries])
                        ;E      =       is table size (no. of entries)
                                ;(table body is followed with
                                ;2 byte "No-Match" adrs)
                                ;Stack has HL saved as top entry.
                        ;C      =       char
                        ;A      =       value to scan for possible match

044C  BE                        CMP     [HL]
044D  23                        INC     HL      ;(2nd byte of this 3 byte entry)
044E  2803 ^0413$               JRZ     VECTOR  ;If match process

0450  23                        INC     HL      ;(3rd byte of this entry)
0451  23                        INC     HL      ;1st byte of next entry
0452  1D                        DEC     E       ;Dec count of entries remaining
0453  20F7 ^044C$               JRNZ    LOOKUPB ;Continue thru body of table

0455  18BC ^0413$               JR      VECTOR  ;No-Match. hl=points to vector
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;PROCessing for modes other than normal.

                        ;VGRAPH                         ;Normal mode EXCEPT: cntl chrs are printed

0457                    VUNDER:
                        ;Underline only
0457  FE20                      CMP     ' '
0459  38DC ^0437$               JRC     VNORM           ;if cntl-chr, process as normal
                        ;       JR      VUN_GR          ;continue


045B                    VUN_GR:
                        ;Underlined Graphics
045B  F680                      OR      80h             ;underline bit
045D  18DC ^043B$               JR      VBRIGH          ;set this chr BRIGHT


045F                    VUN_HA:
                        ;Underline and Half intensity
045F  FE20                      CMP     ' '
0461  38D4 ^0437$               JRC     VNORM           ;if cntl-chr, process as normal
                        ;       JR      VUN_HA_GR


0463                    VUN_HA_GR:
                        ;Underline, Half Intensity, Graphics
0463  F680                      OR      80h             ;set underline bit
                        ;       JR      VHA_GR


0465                    VHALF:
                        ;Half Intensity
0465  FE20                      CMP     ' '
0467  38CE ^0437$               JRC     VNORM           ;if cntl-chr, process as normal
                        ;       JR      VHA_GR


0469                    VHA_GR:
                        ;C=Chr, HL=Curs
0469  F3                        DI
046A                            ENADIM
046C  3600                      STO     DIMBIT,[hl]     ;set dim field bit
046E                            DISDIM
0470  FB                        EI
0471  C35E05                    JMP     VOUT80          ;continue
```

Copyright © 1982 Osborne Computer Corporation

```
0474                    SCREEN:
                        ;SetXY for Screen movement
                        ;ENTRY
                        ;B      =       ESCH
                        ;A      =       new co-ord val, NO OFFSET

0474                            PROC

0474  CB70       $              BIT     6,B
0476  2012 ^048A$               JRNZ    :sX             ;if X-coordinate

0478  E61F              :sY:    AND     0001_1111b      ;mod 32
047A  4F                        MOV     C,A
047B  32EFEF                    STO     A,VRTOFF        ;SET VERTICAL OFFSET FOR COUT
047E  3A62EF                    LD      A,PIABD
0481  E6E0                      AND     1110_0000b
0483  B1                        OR      C
0484  4F                        MOV     C,A
0485  CD8600                    CALL    OPBD            ;set Y coordinate
0488  182E ^0438$               JR      :exitY

048A  87                :sX:    ADD     A,A             ;double A
048B  C6EA                      ADD     A,VFLO          ;PIA A-reg magic offset constant
048D  E6FE                      AND     1111_1110b      ;clear bit 0
048F  4F                        MOV     C,A

                        ;SET DENSITY BIT

0490  3A61EF                    LD      A,PIAAD         ;GET OLD VALUE
0493  E601                      ANI     0000_0001B      ;SAVE DENSITY BIT
0495  B1                        ORA     C               ;OR IN HORIZONTAL OFFSET
0496  4F                        MOV     C,A
0497  CD7900                    CALL    OPAD            ;FUNCTION PIA
049A  CBA8       $              CBIT    5,B             ;finished screen-addressing
049C  1824 ^04C2$               JR      :exitX
```

Copyright © 1982 Osborne Computer Corporation

```
049E                    SETXY:
                        ;Set X-Y value for Cursor/Screen Addressing
                        ;ENTRY
                        ;HL      =       cursor_addr
                        ;B       =       ESCH
                        ;C       =       chr

                        ;EXIT
                        ;to VOUT90; ESCH updated

049E  CDF606                    CALL    UN_CUR
04A1  3EE0                      LDK     A,-(' ')
04A3  81                        ADD     C               ;remove cursor bias
04A4  CB68        $             BIT     5,B             ;cursor/screen addressing?
04A6  20CC ^0474$               JRNZ    SCREEN          ;if screen addressing

                        ;cursor addressing:

04A8  29                        ADD     HL,HL           ;shift HL left
04A9  CB70        $             BIT     6,B             ;X/Y coordinate?
04AB  2010 ^04BD$               JRNZ    :cX             ;if X coordinate

04AD  67                :cY:    MOV     H,A             ;save
04AE  3A62EF                    LD      A,PIABD
04B1  84                        ADD     H               ;offset by start-Y co-ord of video
04B2  1F                        RAR                     ;bit0(A) -> CY, shift A right
04B3  CB1D        $             RR      L               ;CY -> bit7(L)
04B5  F6F0                      OR      0F0h            ;turn on upper nybl
04B7  67                        MOV     H,A             ;HL= new cursor addr

04B8  3E40              :exitY: LDK     A,0100_0000b    ;next addr-chr will be X-coord
04BA  B0                        OR      B
04BB  1808 ^04C5$               JR      :exit2

04BD  17                :cX:    RAL                     ;trash 7th bit
04BE  CB2C        $             SRA     H               ;bit0(H) -> CY, bit7 stays 1
04C0  1F                        RAR                     ;... CY -> bit7(A)
04C1  6F                        MOV     L,A

04C2  3E07              :exitX: LDK     A,EF_MSK
04C4  A0                        AND     B               ;finished addressing: reset addr bits

04C5  3260EF            :exit2: STO     A,ESCH
04C8  C37305                    JMP     VOUT90
```

```
                    ;       Control Code character processing

04CB                VC_HOME:                            ;Home Cursor

04CB                        PROC

04CB  CDF606                CALL    UN_CUR
04CE  3A62EF                LD      A,PIABD
04D1  1F                    RAR                         ;bit0 => CY
04D2  2E00                  LDK     L,0
04D4  CB1D       $          RR      L                   ;CY => bit7, trash bit0
04D6  67                    MOV     H,A
04D7  1830 ^0509$           JR      :fixhl              ;HL := HL or F000h
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
04D9                  VC_MCUP:                            ;Move Cursor Up.
                      ;A=C=Chr=MCUP. HL=Curs.

04D9  CDF606                   CALL    UN_CUR
04DC  E5                       PUSH    HL                 ;old cursor must be on stack
04DD  0180FF                   LDK     BC,(-VLL)
04E0  1817 ^04F9$              JR      :fwa               ;...at this entry point


04E2                  VC_BKS:
                      ;HL=Curs=current (old) char

04E2  CDF606                   CALL    UN_CUR             ;clear 80h bit
04E5  3E7F                     LDK     A,7Fh
04E7  A5                       AND     L
04E8  2804 ^04EE$              JRZ     :wrap              ;if must wrap from col 0 to LLIMIT

04EA  2B                       DEC     HL
04EB  C37305                   JMP     VOUT90             ;Exit

04EE  E5              :wrap:   PUSH    HL                 ;save old cursor
04EF  017FFF                   LDK     BC,-(VLL+1)
04F2  09                       ADD     HL,BC              ;HL = prev_line, (-1)st column
04F3  3A6CEF                   LD      A,LLIMIT           ;LLIMIT = #columns on screen
04F6  4F                       MOV     C,A
04F7  0600                     LDK     B,0

04F9  09              :fwa:    ADD     HL,BC
04FA  E3                       XTHL                       ;get old cursor, save new
04FB  29                       ADD     HL,HL              ;shift line# into H reg.
04FC  3A62EF                   LD      A,PIA8D
04FF  F6E0                     OR      1110_0000b         ;A = line# of UL corner
0501  BC                       CMP     H                  ;set Zflag: @home?
0502  E1                       POP     HL                 ;get new cursor...
0503  2004 ^0509$              JRNZ    :fixhl             ;if NOT @video home

0505  01000C                   LDK     BC,(24*VLL)        ;wrap constant
0508  09                       ADD     HL,BC

0509  3EF0            :fixhl:  LDK     A,0F0h
050B  84                       OR      H                  ;modulo result: keep cursor
050C  67                       MOV     H,A                ;inside video memory.
050D  1864 ^0573$              JR      VOUT90
```

Copyright © 1982 Osborne Computer Corporation

```
050F                    VC_BEL:
                        ;Ring the bell via setting PIAB 2**5 bit

050F  3A62EF                    LD      A,PIABD
0512  F620                      OR      0010_0000b      ;bell bit
0514  4F                        MOV     C,A
0515  CD8600                    CALL    OPBD            ;function PIAB
0518  3E1E                      LDK     A,30            ;ring bell for 30 ticks
051A  326BEF                    STO     A,BELCNT        ;... = 1/2 second
051D  185C ^057B$               JR      VOUT97          ;exit no change
```

Copyright © 1982 Osborne Computer Corporation

```
051F                VC_CLRS:

051F  2100F0                LDK     HL,FWAVM
0522  CD6106                CALL    CLRLN           ;clear 1st line
0525  01800F                LDK     BC,LVMEM-VLL
0528  D5                    PUSH    DE
0529  DDE1        $         POP     IX
052B  CDE206                CALL    VLDIR           ;clear remaining lines
052E  3A62EF                LD      A,PIABD         ;Reset for 1st line of display mem
0531  E6E0                  AND     not(1_1111b)
0533  4F                    MOV     C,A
0534  CD8600                CALL    OPBD
0537  AF                    XRA     A               ;ZERO A
0538  32EFEF                STO     A,VRTOFF        ;SET VERTICAL OFFSET FOR COUT
053B  2100F0                LDK     HL,FWAVM        ;new cursor
053E  1833 ^0573$           JR      VOUT90          ;Exit
```

Copyright © 1982 Osborne Computer Corporation

```
0540                    VC_CR:
0540  CDF606                    CALL    UN_CUR          ;erase cursor
0543  3E80                      LDK     A,80h           ;Carriage Return
0545  A5                        AND     L
0546  6F                        MOV     L,A
0547  182A ^0573$               JR      VOUT90

0549                    VC_LF:
0549  CD8406                    CALL    DO_LF           ;Line Feed
054C  1825 ^0573$               JR      VOUT90
```

Copyright © 1982 Osborne Computer Corporation

```
054E                VC_MCRT
                    ;Move Cursor Right

054E  CDF606                CALL    UN_CUR
0551  7E                    LD      A,[hl]
0552  180A ^055E$           JR      VOUT80          ;re-echo current chr
```

Copyright © 1982 Osborne Computer Corporation

```
0554                    VGRAPH:
0554  CB6F       $              BIT     5,A
0556  2806 ^055E$               JRZ     VOUT80
0558  CB77       $              BIT     6,A
055A  2802 ^055E$               JRZ     VOUT80
055C  E69F                      ANI     9FH

055E                    VOUT80:
                        ;Exit points for COUT
                        ;Here to store new data and to update cursor

055E  77                        STO     A,[hl]          ;This exit path stores A (new char)
055F  5D                        MOV     E,L
0560  CBBB       $              CBIT    7,E             ;E = col(cursor)
0562  3A6CEF                    LD      A,LLIMIT
0565  3D                        DEC     A               ;A = last_legal_col
0566  93                        SUB     E               ;A = last_legal_col - col(cur)
0567  2009 ^0572$               JRNZ    VOUT85          ;if not @LLIMIT

0569  3E80                      LDK     A,80h
056B  A5                        AND     L
056C  6F                        MOV     L,A             ;do CR...
056D  CD8706                    CALL    DO_LF2          ;...and LF.
0570  1801 ^0573$               JR      VOUT90

0572  23                VOUT85: INC     HL              ;move cursor

                        ;Here if NO cursor update

0573  7E                VOUT90: LD      A,[HL]          ;This exit path turns on 80h bit

                        ;Here if new data already in A

0574  17                        RAL                     ;Make this chr cursor

0575  3F                VOUT96: CMC                     ;invert cursor bit
0576  1F                        RAR
0577  77                        STO     A,[hl]
0578  225AEF                    STO     HL,CURS         ;update cursor

                        ;Here if no change to cursor, restore reg and exit

057B  C1                VOUT97: POP     BC
057C  D1                        POP     DE
057D  E1                        POP     HL
057E  79                        MOV     A,C             ;Exit with A=chr
057F  C9                        RET                     ;return, end of cout subr.

      = 04FA            ;First  = VOUT97 - (127 + 2)    ;earliest possible JR

      = 05DC            ;Last   = VOUT80 + (128 - 2)    ;latest possible JR
```

Copyright © 1982 Osborne Computer Corporation

```
0580                ESC_LCK:                        ;Lock Keyboard

0580                         PROC

0580  AF                     XOR     A
0581  1802 ^0585$            JR      :2

0583  3EFF          ESC_ULK  LDK     A,0FFh         ;Unlock Keyboard

0585  3259EF        :2:      STD     A,KEYLCK
0588  18F1 ^0578$            JR      VOUT97
```

Copyright © 1982 Osborne Computer Corporation

```
058A                  EEOL:
                      ;Erase to end of line

058A  E5                      PUSH    HL              ;save cursor
058B  CD6106                  CALL    CLRLN
058E  E1                      POP     HL

058F  18E2 ^0573$             JR      VOUT90
```

Copyright © 1982 Osborne Computer Corporation

```
0591                    ESC_CAD:
                        ;Cursor Addressing

0591  3E07                      LDK     A,EF_MSK
0593  A0                        AND     B
0594  F618                      OR      EF_ESC or EF_ADR        ;next chr will be Y-coord

0596  3260EF            :exit3: STO     A,ESCH
0599  18E0 ^057B$               JR      VOUT97

059B                    ESC_SAD:
                        ;Screen Addressing

059B  3E07                      LDK     A,EF_MSK
059D  A0                        AND     B
059E  F638                      OR      EF_ESC or EF_ADR or EF_SCR
05A0  18F4 ^0596$               JR      :exit3
```

Copyright © 1982 Osborne Computer Corporation

```
05A2                    EDELC:
                        ;Delete Character

05A2  E5                        PUSH    HL              ;save cursor_addr

05A3  CDD405                    CALL    CALC            ;calculate BC
05A6  E5                        PUSH    HL
05A7  DDE1        $             POP     IX
05A9  23                        INC     HL              ;HL = cursor_addr + 1
05AA  CDE206                    CALL    VLDIR           ;move characters
05AD  3620                      STO     ' ',[hl]        ;last chr becomes blank

05AF  F3                        DI
05B0                            ENADIM          ;enable 9th bit memory
05B2  2B                        DEC     HL              ;HL = last chr on this line
05B3  3680                      STO     BRTBIT,[hl]             ;set chr BRIGHT
05B5                            DISDIM
05B7  FB                        EI                      ;main memory

05B8  E1                        POP     HL              ;restore cursor_addr
05B9  1888 ^0573$               JR      VOUT90          ;next
```

Copyright © 1982 Osborne Computer Corporation

```
05B8                     EINSRT:
                         ;Insert Character

05B8  CDF606                       CALL    UN_CUR
05BE  E5                           PUSH    HL              ;save cursor_addr
05BF  CDD405                       CALL    CALC            ;calculate BC
05C2  3E7F                         LDK     A,7Fh
05C4  B5                           OR      L
05C5  6F                           MOV     L,A             ;HL = last_chr on this line
05C6  E5                           PUSH    HL
05C7  DDE1          $              POP     IX
05C9  2B                           DEC     HL
05CA  CDCE06                       CALL    VLDDR           ;do move
05CD  E1                           POP     HL              ;restore cursor
05CE  7E                           LD      A,[hl]          ;get underline bit of this chr.
05CF  17                           RAL                     ;into CY
05D0  3E40                         LDK     A,' ' shl 1     ;change this chr to ' '
05D2  18A1 ^0575$                  JR      VOUT96          ;exit
```

Copyright © 1982 Osborne Computer Corporation

```
05D4                   CALC:
                       ;Subroutine for use with EDELC and EINSRT:
                       ;Calculate #chrs to move; if move zero chrs, never return.

05D4  3E7F                     LDK     A,VLL-1          ;A= max #chrs to be moved
05D6  4D                       MOV     C,L
05D7  CBB9        $            CBIT    7,C              ;C = col(cursor)
05D9  91                       SUB     C                ;A = #chrs to move
05DA  2804 ^05E0$              JRZ     :end             ;if move zero characters

05DC  4F                       MOV     C,A
05DD  0600                     LDK     B,0              ;BC = #chrs to move
05DF  C9                       RET

05E0  E1               :end:   POP     HL               ;trash return_addr
05E1  E1                       POP     HL               ;cursor_addr
05E2  188F ^0573$              JR      VOUT90
```

Copyright © 1982 Osborne Computer Corporation


```
05E4                    ESCSGR:
                        ;ESC-Sequence processing.

05E4                            PROC

05E4  3E01                      LDK     A,EF_GR         ;ESC-g
05E6  1806 ^05EE$               JR      :125            ;set graphics mode.

05E8  3E02              ESCSHA: LDK     A,EF_HA         ;ESC-) set half intensity
05EA  1802 ^05EE$               JR      :125            ;go set flag bit

05EC  3E04              ESCSUN: LDK     A,EF_UN         ;ESC-l set underline

05EE  B0                :125:   OR      B               ;Reg B is ESCH Byte value

05EF  3260EF            :130:   STO     A,ESCH          ;store desired value.
05F2  1887 ^0578$               JR      VOUT97          ;Exit

05F4  3EFE              ESCCGR: LDK     A,NOT EF_GR     ;ESC-G Clear graphics mode
05F6  1806 ^05FE$               JR      :140            ;go clear ESCH bit

05F8  3EFD              ESCCHA: LDK     A,NOT EF_HA     ;ESC-( Clear half intensity
05FA  1802 ^05FE$               JR      :140

05FC  3EFB              ESCCUN: LDK     A,NOT EF_UN     ;ESC-m Clear underline

05FE  A0                :140:   AND     B               ;Clear bit
05FF  18EE ^05EF$               JR      :130            ;Go store ESCH byte

      = 051F            ESCZZ:  =       VC_CLRS         ;ESC-Z Clear screen -same as
                                                        ;Control-Z routine.
```

Copyright © 1982 Osborne Computer Corporation


```
0601                    ESCRR:
                        ;Delete Line

0601                    ESCEE:
                        ;Insert Line
                        ;ENTRY
                        ;HL      =       cursor
                        ;C       =       chr

                        ;EXIT
                        ;screen updated
                        ;HL      =       new cursor ...to VOUT90

0601                            PROC

0601  CDF606                    CALL    UN_CUR
0604  3E80                      LDK     A,1000_0000b
0606  A5                        AND     L
0607  6F                        MOV     L,A             ;do CR
0608  E5                        PUSH    HL              ;save new cursor
0609  29                        ADD     HL,HL
060A  3A62EF                    LD      A,PIABD
060D  C618                      ADD     A,24            ;A = addr(25th line)
060F  94                        SUB     H               ;A = lines_to_move + 1
0610  E61F                      AND     0001_1111b      ;mod 32
0612  47                        MOV     B,A
0613  3E52                      LDK     A,VDELL
0615  B9                        CMP     C
0616  78                        MOV     A,B             ;recall #lines to move
0617  2823 ^063C$               JRZ     :delt           ;if deleting a line

                        ;Insert a line

0619  84                :insrt: ADD     H               ;A = addr(25th line)
061A  57                        MOV     D,A
061B  1E00                      LDK     E,0
061D  CB1A          $           RR D
061F  CB1B          $           RR E                    ;shift right DE
0621  1B                        DEC     DE              ;DE = addr(1st_chr_on_1st_line)
0622  78                        MOV     A,B             ;A = #lines to move
0623  2180FF                    LDK     HL,-VLL
0626  19                        ADD     HL,DE           ;HL = addr(line above DE)
                                                        ;DE := DE or F000h; HL := HL or F000h
0627  1806 ^062F$               JR      :istrt

0629  D5                :icont: PUSH    DE
062A  DDE1          $           POP     IX
062C  CDCE06                    CALL    VLDDR           ;move 1 line down

062F  CD5206            :istrt: CALL    :vmod
0632  20F5 ^0629$               JRNZ    :icont          ;if must move more lines

0634  23                        INC     HL              ;HL => 1st chr of new line
```

Copyright © 1982 Osborne Computer Corporation

```
0635                :exit:
0635  CD6106               CALL    CLRLN
0638  E1                   POP     HL              ;recover cursor
0639  C37305               JMP     VOUT90          ;Main Exit

063C  D1            :delt: POP     DE              ;recover new cursor
063D  D5                   PUSH    DE
063E  218000               LDK     HL,VLL
0641  19                   ADD     HL,DE           ;HL = line_below_cursor
0642  1806 ^064A$          JR      :dstrt

0644  D5            :dcont: PUSH   DE
0645  DDE1      $          POP     IX
0647  CDE206               CALL    VLDIR           ;move 1 line up

064A  CD5206        :dstrt: CALL   :vmod
064D  20F5 ^0644$          JRNZ    :dcont

064F  EB                   EX      HL,DE           ;get addr of line to clear
0650  18E3 ^0635$          JR      :exit
```

Copyright © 1982 Osborne Computer Corporation

```
0652                    ;vmod:
                        ;HL = HL or F000h; DE = DE or F000h
                        ;simple mod-4096 arithmetic to keep pointers INSIDE video memory

0652  F5                        PUSH    AF              ;save A = #lines to move
0653  3EF0                      LDK     A,0F0h
0655  B4                        OR      H
0656  67                        MOV     H,A             ;set upper nybl of H
0657  3EF0                      LDK     A,0F0h
0659  B2                        OR      D
065A  57                        MOV     D,A             ;modulo 4096
065B  018000                    LDK     BC,VLL
065E  F1                        POP     AF
065F  3D                        DEC     A               ;decrement line_count
0660  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0661                    CLRLN:
                        ;Clear to end of line
                        ;ENTRY
                        ;HL      =       Cursor

                        ;EXIT
                        ;clear to EOL
                        ;        Uses    All.

0661                             PROC

0661  3620                       STO     ' ',[hl]        ;clear cursor...
0663  F3                         DI
0664                             ENADIM
0666  3680                       STO     BRTBIT,[hl]     ;set cursor BRIGHT
0668                             DISDIM
066A  FB                         EI
066B  3A6CEF                     LD      A,LLIMIT
066E  3D                         DEC     A               ;max_#cols => maximum_col_#
066F  5D                         MOV     E,L
0670  CBBB      $                CBIT    7,E
0672  93                         SUB     E               ;A = col(EOL) - col(cursor)
0673  C8                         RZ                      ;if @EOL, done

0674  3004 ^067A$                JRNC    :2              ;if inside logical_video_line

0676  3E80                       LDK     A,VLL
0678  93                         SUB     E               ;...else clr to end of 128-chr line
0679  C8                         RZ                      ;if cursor @ column #127

067A  4F                :2:      MOV     C,A
067B  0600                       LDK     B,0             ;BC = chrs to move
067D  E5                         PUSH    HL
067E  DDE1      $                POP     IX
0680  DD23      $                INC     IX
0682  185E ^06E2$                JR      VLDIR
```

Copyright © 1982 Osborne Computer Corporation

```
0684                    DO_LF:
                        ;Do Line Feed processing
                        ;ENTRY
                        ;HL      =       cursor_addr

                        ;EXIT
                        ;Cursor cleared
                        ;HL updated for current cursor pos
                        ;window moved if necessary

0684                            PROC

0684  CDF606                    CALL    UN_CUR          ;clear cursor

0687  E5                DO_LF2: PUSH    HL              ;save original cursor
0688  018000                    LDK     BC,VLL          ;line length
068B  09                        ADD     HL,BC
068C  3004 ^0692$               JRNC    :nowap          ;if not wrapping from LWAVM to FWAVM

068E  0100F0                    LDK     BC,FWAVM
0691  09                        ADD     HL,BC           ;HL = new cursor @ top of VM

0692  E3                :nowap: XTHL                    ;save new cursor, get old
0693  29                        ADD     HL,HL           ;shift HL left
0694  3A62EF                    LD      A,PIABD
0697  C617                      ADD     A,23            ;start + 23 = last_video_line
0699  94                        SUB     H               ;A = l_line - curr_line
069A  E61F                      AND     0001_1111b      ;modulo 32
069C  2802 ^06A0$               JRZ     :vmov           ;if cursor is on 24th line of screen

069E  E1                :end:   POP     HL              ;get new cursor
069F  C9                        RET

06A0  3A6CEF            :vmov:  LD      A,LLIMIT
06A3  CB3D         $            SRL     L               ;unshift L register
06A5  95                        SUB     L               ;A = LLIMIT - col(cursor)
06A6  38F6 ^069E$               JRC     :end            ;if cursor is outside logical line

                        ;cursor is on last line of screen, inside of logical line.
                        ;must move screen to follow cursor down through video memory.

06A8  E1                        POP     HL
06A9  E5                        PUSH    HL
06AA  3E80                      LDK     A,80h
06AC  A5                        AND     L
06AD  6F                        MOV     L,A             ;HL = beginning of line
06AE  CD6106                    CALL    CLRLN           ;erase to EOL
06B1  3A62EF                    LD      A,PIABD
06B4  47                        MOV     B,A
06B5  E6E0                      AND     not 31          ;A = line zero
06B7  4F                        MOV     C,A             ;C = housekeeping bits 5..7
06B8  3E1F                      LDK     A,31
06BA  04                        INC     B               ;increment line#
06BB  A0                        AND     B               ;A = line#
06BC  32EFEF                    STO     A,VRTOFF        ;SET VERTICAL OFFSET FOR COUT
06BF  B1                        OR      C               ;A = new line# OR housekeeping_bits
06C0  4F                        MOV     C,A             ;C = new value for OPBD
06C1  CD8600                    CALL    OPBD            ;move video screen down 1 line in memory
06C4  E1                        POP     HL
```

```
06C5  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright ©1982 Osborne Computer Corporation

```
06C6                STODIM:
                    ;STORE THE CONTENTS OF THE B REG IN THE ADDR POINTED TO BY THE HL PAIR
                    ;ENTRY
                    ;B       =       VALUE
                    ;HL      =       ADDRESS

                    ;EXIT
                    ;NONE

06C6                        PROC

06C6  F3                    DI
06C7                        ENADIM                  ;ENABLE DIM
06C9  70                    STO      B,[HL]         ;STORE
06CA                        DISDIM                  ;DISABLE DIM
06CC  FB                    EI
06CD  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
06CE                    VLDDR:
                        ;Video Block Move
                        ;ENTRY
                        ;BC, IX, HL set

                        ;EXIT
                        ;LDDR on main & 9th bit memory
                        ;        Uses    BC, DE, HL,IX

06CE  DDE5        $              PUSH    IX
06D0  D1                         POP     DE

06D1  C5                         PUSH    BC
06D2  D5                         PUSH    DE
06D3  E5                         PUSH    HL

06D4  EDB8        $              LDDR                     ;main memory

06D6  E1                         POP     HL
06D7  D1                         POP     DE
06D8  C1                         POP     BC

06D9  F3                         DI
06DA                             ENADIM
06DC  EDB8        $              LDDR                     ;9th bit memory
06DE                             DISDIM
06E0  FB                         EI
06E1  C9                         RET
```

Copyright © 1982 Osborne Computer Corporation

```
06E2                VLDIR:
                    ;Video Block Move
                    ;ENTRY
                    ;BC, IX, HL set

                    ;EXIT
                    ;LDIR on main & 9th bit memory
                    ;        Uses    BC, DE, HL,IX

06E2  DDE5      $            PUSH    IX
06E4  D1                     POP     DE

06E5  C5                     PUSH    BC
06E6  D5                     PUSH    DE
06E7  E5                     PUSH    HL

06E8  EDB0      $            LDIR                    ;main memory

06EA  E1                     POP     HL
06EB  D1                     POP     DE
06EC  C1                     POP     BC

06ED  F3                     DI
06EE                         ENADIM
06F0  EDB0      $            LDIR                    ;9th bit memory
06F2                         DISDIM
06F4  FB                     EI
06F5  C9                     RET
```

Copyright © 1982 Osborne Computer Corporation

```
06F6                   UN_CUR:
                       ;Undo/Invert Cursor
                       ;ENTRY
                       ;HL      =       cursor_addr

                       ;EXIT
                       ;cursor inverted
                       ;        Uses    A, CY.

06F6  7E                       LD      A,[hl]          ;get the chr
06F7  17                       RAL                     ;cursor_bit => CY
06F8  3F                       CMC                     ;invert it
06F9  1F                       RAR
06FA  77                       STO     A,[hl]          ;...
06FB  C9                       RET
```

Copyright © 1982 Osborne Computer Corporation

```
                        *[ ]

06FC                    GKEY:
                        ;KEYBOARD INTERRUPT PROCESSOR
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;KEYBOARD PROCESSING DONE, RESULT IN LDKEY.

06FC                            PROC

06FC  F3                        DI
06FD  ED736FEF  $               STO     SP,IESTK        ;SAVE INTERRUPTED PROCESS STK
0701  3199EF                    LDK     SP,ISTK         ;SET TO RAM INT STK

0704  F5                        PUSH    AF              ;SAVE ALL REGESTERS
0705  C5                        PUSH    BC
0706  D5                        PUSH    DE
0707  E5                        PUSH    HL
0708  DDE5      $               PUSH    IX
070A  FDE5      $               PUSH    IY

                        ;Routine checks to see if the disk drive motor should be turned off by updating DACTIVE...Routine ALSO
                        ;checks to see if bell is currently ringing: if so, decrement counter. If counter turns zero,
                        ;shut off bell.

                                ;CHECK BELL

070C  216BEF                    LDK     HL,BELCNT
070F  AF                        XOR     A
0710  B6                        OR      [hl]            ;cell=zero ?
0711  280C ^071F$               JRZ     :1              ;if bell now off

0713  35                        DEC     [hl]            ;...bell is on. decrement counter
0714  2009 ^071F$               JRNZ    :1              ;if bell should stay on awhile yet

                                ;TURN BELL OFF

0716  3A62EF                    LD      A,PIABD
0719  E6DF                      AND     1101_1111b      ;clear bell bit
071B  4F                        MOV     C,A
071C  CD8600                    CALL    OPBD

071F  2B                :1:     DEC     HL              ;HL = HL - 1 DACTVE
0720  7E                        LD      A,[hl]
0721  B7                        OR      A
0722  2804 ^0728$               JRZ     :2              ;RETURN if inactive

                                ;TURN DRIVE OFF IF DACTVE = 1

0724  35                        DEC     [hl]            ;reset delay
0725  CC060F                    CZ      DDRV            ;if deselect drive

                        ;UPDATE COUNTER

0728  2ACCEF            :2:     LD      HL,SEQ          ;GET LOW TWO BYTES
072B  23                        INC     HL              ;+1
072C  22CCEF                    STO     HL,SEQ          ;STORE
```

```
                        ;READ KEYBOARD

072F  3A59EF                LD      A,KEYLCK
0732  B7                    OR      A
0733  C44E07                CNZ     KBDRVR          ;READ KEYBOARD IF KEYLOCK NOT ACTIVE

                        ;Exit interrupt code via exiting to RAM and then enable or disable ROM code depending on the value
                                ;contained in ROMRAM cell.

0736  3A002C                LD      A,H.VIO         ;clear interrupt

0739  3A08EF                LD      A,ROMRAM        ;IS ROM IN OR OUT?
073C  B7                    OR      A
073D  C29300                JNZ     ROMJP1          ;if return to RAM

0740  FDE1      $           POP     IY              ;RESTORE REGESTERS
0742  DDE1      $           POP     IX
0744  E1                    POP     HL
0745  D1                    POP     DE
0746  C1                    POP     BC
0747  F1                    POP     AF

0748  ED7B6FEF  $           LD      SP,IESTK        ;get users stack back
074C  FB                    EI
074D  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
;       This file contains the 2-key roll over keyboard driver for
;       the OSBORNE 1 comuter.

;       Author:
;       Microcode Corporation.
;       Fremont, CA.
;       Y. N. Sahae
;       September 1981

;       Revisions:

;       2-Key roll over keyboard driver.

;       DESCRIPTION:
;       The keyboard driver gets control via the 60hz interrupt, i.e. once
;       every 16 ms. It scans the keyboard to detect any struck keys. If a
;       key is found, it is entered into the keylist if there is space
;       in the keylist and the key is not already in the list.  At the end of
;       the scan, the keys in the list are proecessed. If the key is still
;       on, it is placed in lkey (or special action taken) after translating
;       the keynumber. A count is also stored in the list and the key will
;       be serviced again at the end of this count if it is still on. Thus
;       the key will repeat if it is held down. If a key which is in the
;       list is not on it is removed from the list.

;       The keyboard driver consists of the following routines:

;       KBDRVR - Examines the keylist, calls CHKEY to determine if key
;       is still on. Removes the key from the list if it is not on. If
;       key is on, it decrements the count associated with the key. when
;       the count goes to zero, it calls KBSERV to service the key. Calls
;       KBSCAN to enter any new keys into the list.

;       KBSCAN - This routine scans the keyboard, detects a struck
;       key and enters it into the keylist. The key is entered
;       into the keylist if the key is not already present in the keylist
;       and there is an empty slot in the keylist.

;       KBSERV - It calls the routine CHKEY to check if shift/ctl or alphlock
;       keys are on. It then translates the keynumber into the ASCII
;       code and places it in the LKEY for the CBIOS to read. For some
;       special cases, it calls ROM resident routines to process the key.

;       CHKEY - It checks if a given key is on.


;       Data structure:
;       The main structure used is the keylist. The format of each entry is:

;       Byte 0:
;               bit 7 : Set indicates entry is in use.
;               bit 6 : Set indicates key has been serviced once.
;               bits 5..3 : contain the row number of struck key.
;               bits 2..0 : contain the column number of struck key.
;       Byte 1:
;               bits 7..0 : contain the repeat count for the key.
```

Copyright © 1982 Osborne Computer Corporation

```
074E                    KBDRVR:
                        ;DETECTS AND PROCESSES KEYSTROKES.
                        ;ENTRY
                        ;NONE

                        ;EXIT
                                ;LKEY   =       KEYSTROKE

074E                            PROC

074E  CD8607                    CALL    KBSCAN          ;SCAN KEYBOARD AND ENTER KEYS INTO KEYLIST

                        ;EXAMINE KEYLIST. IF KEY FOUND IN KEYLIST, CALL CHKEY TO SEE IF KEY IS STILL ON. REMOVE FROM LIST WHEN NOT ON.

0751  214EF                     LDK     HL,KEYLST       ;POINT TO FIRST ENTRY OF KEYLIST
0754  0603                      LDK     B,KL_LEN

0756  3A5EEF            :1:     LD      A,LKEY
0759  37                        OR      A
075A  C0                        RNZ                     ;RETURN WHEN A KEY IS WAITING IN LKEY

075B  7E                        LD      A,[HL]          ;GET BYTE 0 OF ENTRY
075C  CB7F        $             BIT     KL_USED,A
075E  2821 ^0781$               JRZ     :5              ;IF ENTRY IS IN USE THEN

0760  CDE207                    CALL    CHKEY           ;CHECK IF STILL ON
0763  2004 ^0769$               JRNZ    :2              ;IF KEY IS NOW OFF THEN

0765  3600                      STO     0,[HL]          ;REMOVE KEY FROM LIST
0767  1818 ^0781$               JR      :5

                        ;KEY IS ON. DECREMENT ITS REPEAT COUNT. IF COUNT GOES TO ZERO THEN IT IS TIME TO SERVICE THE KEY.

0769  E5                :2:     PUSH    HL              ;SAVE PTR TO FIRST BYTE OF ENTRY
076A  23                        INC     HL              ;POINT TO REPEAT COUNT
076B  35                        DEC     [HL]
076C  2012 ^0780$               JRNZ    :4              ;EXIT WHEN NOT TIME TO SERVICE THE KEY.

                        ;IT IS TIME TO SERVICE THE KEY. SET THE NEXT REPEAT COUNT

076E  E3                        EX      [SP],HL         ;POINT BACK TO THE FIRST BYTE OF ENTRY
076F  7E                        LD      A,[HL]
0770  CB77        $             BIT     KY_SRVD,A       ;CHECK IF IT IS SERVICED BEFORE
0772  CBF6        $             SBIT    KY_SRVD,[HL]    ;SET THE SERVICED ONCE FLAG
0774  E3                        EX      [SP],HL         ;POINT BACK TO THE REPEAT COUNT
0775  3618                      STO     IRPTCT,[HL]     ;AND STORE RPT COUNT AS PER SERVICED FLAG
0777  2802 ^077B$               JRZ     :3

0779  3606                      STO     SRPTCT,[HL]

077B  E63F              :3:     AND     KROW_M+KCOL_M
077D  CD0908                    CALL    KBSERV          ;CALL TO SERVICE THE KEY

0780  E1                :4:     POP     HL              ;GET PTR TO FIRST BYTE OF ENTRY AGAIN

0781                    :5:
                                ECHO    KLE_LEN
                                INC     HL              ;POINT TO NEXT ENTRY
                                ENDM
```

Copyright © 1982 Osborne Computer Corporation

```
0783  10D1 ^0756$        DJNZ    :1               ;UNTIL  COMPLETE LIST SCANNED

0785  C9                 RET                      ;RETURN
```

Copyright © 1982 Osborne Computer Corporation

```
0786                    KBSCAN:
                        ;SCAN KEYBOARD AND ENTER DETECTED KEYS IN THE KEYLIST.
                        ;ENTRY
                        ;NONE

                        ;EXIT
                                ;KEYLST =       CONTAINS ANY KEYS DETECTED.

0786                            PROC

0786  2EFF                      LDK     L,0FFH          ;SEE IF ANY KEY PRESSED
0788  CD0008                    CALL    RDROW
078B  C8                        RZ                      ;RETURN WHEN NONE

078C  2E81                      LDK     L,ROW0_M        ;GET ROW 0
078E  CD0008                    CALL    RDROW

0791  E6E3                      AND     11100011B       ;REMOVE CTL/SHIFT AND ALPHA LOCK
0793  0607                      LDK     B,TOT_ROW

                        ;IN THIS LOOP, REG B CONTAINS TOTROW CURRENT ROW BEING SCANNED

0795  2843 ^07DA$ :1:           JRZ     :8              ;IF ANY KEY IS PRESSED THEN

0797  C5                        PUSH    BC              ;SAVE LOOP COUNT
0798  5F                        MOV     E,A             ;E = COLUMNS
0799  3E07                      LDK     A,TOT_ROW
079B  90                        SUB     B
079C  17                        RAL
079D  17                        RAL
079E  17                        RAL
079F  57                        MOV     D,A             ;D = ROW NUMBER * 8
07A0  0E00                      LDK     C,0             ;INITIALIZE COLUMN COUNTER

                        ;SCAN THIS ROW FROM RIGHT TO LEFT TO GET THE COLUMN NUMBER

07A2  CB3B        $ :2:         SRL     E               ;SHIFT COLUMN BIT INTO CARRY
07A4  302E ^07D4$               JRNC    :7              ;IF A KEY IS FOUND THEN

                        ;ENTER THE KEY WHOSE COLUMN NUMBER IS IN C AND ROW*8 IS IN D INTO THE KEYLST PROVIDED THE KEY IS NOT
                        ;ALREADY IN LIST AND THERE IS AN EMPTY SLOT IN THE LIST.

07A6  7A                        MOV     A,D
07A7  81                        ADD     A,C
07A8  C5                        PUSH    BC
07A9  4F                        MOV     C,A             ;C = KEY NUMBER
07AA  D5                        PUSH    DE              ;SAVE DE
07AB  E5                        PUSH    HL              ;SAVE HL
07AC  0603                      LDK     B,KL_LEN        ;LENGTH OF KEYLIST
07AE  21D4EF                    LDK     HL,KEYLST
07B1  110000                    LDK     DE,0

07B4  7E          :3:           LD      A,[HL]
07B5  CB7F        $             BIT     KL_USED,A
07B7  2807 ^07C0$               JRZ     :4              ;IF ENTRY IS USED THEN

07B9  E63F                      AND     KROW_M+KCOL_M
07BB  B9                        CMP     C               ;CHECK WITH CURRENT KEY
07BC  2813 ^07D1$               JRZ     :6              ;EXIT IF THIS KEY IS IN LIST
```

```
07BE  1802 ^07C2$          JR      :5

07C0  5D          :4:      MOV     E,L              ;ELSE (AN EMPTY ENTRY IS FOUND)
07C1  54                   MOV     D,H              ;SAVE ADRS OF EMPTY ENTRY IN DE

07C2              :5:
                           ECHO    KLE_LEN
                           INC     HL
                           ENDM                     ;NEXT ENTRY

07C4  10EE ^07B4$          DJNZ    :3               ;TILL LIST SCANNED

                  ;CHECK IF AN EMPTY ENTRY WAS FOUND.

07C6  7A                   MOV     A,D
07C7  B7                   OR      A
07C8  2807 ^07D1$          JRZ     :6               ;IF EMPTY ENTRY WAS FOUND THEN

07CA  EB                   EX      DE,HL            ;HL = EMPTY ENTRY
07CB  71                   STO     C,[HL]           ;STORE THE KEY IN THE LIST
07CC  CBFE        $        SBIT    KL_USED,[HL]     ;SET USED FLAG
07CE  23                   INC     HL
07CF  3601                 STO     DB_CT,[HL]       ;STORE DEBOUNCE DELAY

07D1  E1          :6:      POP     HL               ;RESTORE ALL REGSTERS
07D2  D1                   POP     DE
07D3  C1                   POP     BC

07D4  0C          :7:      INC     C                ;INCREMENT COLUMN NUMBER
07D5  AF                   XOR     A
07D6  BB                   CMP     E
07D7  20C9 ^07A2$          JRNZ    :2               ;UNTIL ALL COLUMNS SCANNED

07D9  C1                   POP     BC               ;RSTORE BC

07DA  CB25        $ :8:    SLA     L                ;MOVE TO NEXT ROW
07DC  CD0008               CALL    RDROW
07DF  10B4 ^0795$          DJNZ    :1

07E1  C9                   RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
07E2                CHKEY:
                    ;CHECKS IF KEY NUMBER IS ON.
                    ;ENTRY
                    ;A       =       KEYNUMBER
                    ;EXIT
                    ;Z CLR   =       KEY IS OFF.
                    ;Z SET   =       KEY IS ON.

07E2                        PROC

07E2  E5                    PUSH    HL              ;SAVE CALLERS HL
07E3  F5                    PUSH    AF              ;SAVE KEYNUMBER
07E4  1F                    RAR
07E5  1F                    RAR
07E6  1F                    RAR                     ;RIGHT JUSTIFY ROW NUMBER
07E7  CDF607                CALL    GTMASK
07EA  F1                    POP     AF              ;GET KEY NUMBER

07EB  D5                    PUSH    DE              ;SAVE ROW MASK
07EC  CDF607                CALL    GTMASK          ;GET COL MASK (COL NUM IS IN BITS 0..2)
07EF  E1                    POP     HL              ;MOVE ROW MASK TO L

07F0  CD0008                CALL    RDROW           ;GET ROW OF KEYS ADRSED BY L
07F3  A3                    AND     E               ;Z IND  =       VALUE OF KEY
07F4  E1                    POP     HL

07F5  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
07F6                    GTMASK:
                        ;GENERATES MASK WITH ONE BIT SET.
                        ;ENTRY
                        ;A      =       BIT NUMBER (0..7)

                        ;EXIT
                        ;E      =       MASK

07F6                            PROC

07F6  1E01                      LDK     E,1
07F8  E607                      AND     7

07FA  C8                :1:     RZ
07FB  CB23        $             SLA     E
07FD  3D                        DEC     A
07FE  18FA ^07FA$               JR      :1
```

Copyright © 1982 Osborne Computer Corporation

```
0800                    RDROW:
                        ;READS A ROW OF KEYS
                        ;ENTRY
                        ;L        =       LOWER 8 BITS OF ADRS TO READ THE ROW

                        ;EXIT
                        ;A        =       ROW VALUE

0800                              PROC

0800  2622                        LDK     H,HIGH(H.KEY)           ;HL = PRT ADRS FOR GIVEN ROW
0802  7D                          MOV     A,L
0803  ED4F         $              MOV     R,A
0805  7E                          LD      A,[HL]
0806  EEFF                        XOR     0FFH                    ;INVERT VALUES
0808  C9                          RET
```

Copyright © 1982 Osborne Computer Corporation

```
      = 008D      LFT_ARW =        8DH
      = 008B      RT_ARW  =        8BH
      = 008A      UP_ARW  =        8AH
      = 008C      DN_ARW  =        8CH
      = 005B      HM_SCRN =        '['

0809              KBSERV:
                  ;SERVICES THE KEY
                  ;ENTRY
                  ;A       =        KEYNUMBER
                  ;[SP]-4 =        POINTER TO KEYLST ENTRY (USED FOR SLIDE FNC ONLY)

                  ;EXIT
                  ;NONE

                  ;PRESERVES REG B

0809                      PROC

                  ;SETUP HL TO POINT TO KEYCODE TABLE ENTRY FOR THIS KEY

0809  5F                  MOV     E,A
080A  1600                LDK     D,0              ;USED HERE AND LATER
080C  21D408              LDK     HL,KYCDTB
080F  19                  ADD     HL,DE
0810  7E                  LD      A,[HL]
0811  FE21                CMP     ' '+1
0813  381F ^0834$         JRC     KEYE             ;IGNORE SHIFT/CTL ETC FOR CHARS LESS THAN 21H

0815  F5                  PUSH    AF

0816  2E01                LDK     L,1              ;ROW 0 ADRS
0818  CD0008              CALL    RDROW            ;GET ROW CONTAINING CTL,SHIFT AND ALPHA KEY

081B  F5                  PUSH    AF

081C  2E80                LDK     L,80H
081E  CD0008              CALL    RDROW
0821  E608                ANI     8
0823  5F                  MOV     E,A
0824  F1                  POP     AF

0825  B3                  OR      E
0826  5F                  MOV     E,A
0827  F1                  POP     AF               ;RESTORE KEYCODE

0828  CB53       $        BIT     CTL_KY,E
082A  202E ^085A$         JRNZ    KEY4             ;GO PROCESS CTL KEY

082C  CB63       $        BIT     SHFT_KY,E
082E  2014 ^0844$         JRNZ    KEY2             ;GO PROCESS SHIFT KEY

0830  CB5B       $        BIT     ALPH_KY,E
0832  2004 ^0838$         JRNZ    KEY1             ;GO PROCESS ALPHA KEY

                  ;FALL THROUGH TO "KEYE"
```

Copyright © 1982 Osborne Computer Corporation

```
0834                KEYE:
                    ;STORE KEY CODE INTO "LKEY" AND RETURN

0834  325EEF                STO     A,LKEY
0837  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0838                KEY1:
0838  FE61                  CMP     'a'         ;PROCESS ALPHA KEY
083A  38F8 ^0834$           JRC     KEYE        ;EXIT WHEN LESS THAN 'a'. ALPHA HAS NO EFFECT

083C  FE80          :27:    CMP     80H
083E  30F4 ^0834$           JRNC    KEYE        ;OR WHEN >= 80H

0840  EE20          :28:    XOR     20H         ;FOLD CHAR TO UPPER CASE
0842  18F0 ^0834$           JR      KEYE
```

```
0844              KEY2:
0844  FE61                  CMP     'a'             ;PROCESS SHIFT KEY
0846  30F4 ^083C$           JRNC    :27             ;GOTO ALPHA WHEN CHAR > 'a'

0848  FE5B                  CMP     '['
084A  3806 ^0852$           JRC     KEY3            ;GOTO PROCESS SHIFT NUMERICS ETC

084C  20F2 ^0840$           JRNZ    :28             ;INVERT SHIFT BIT FOR '\'

084E  3E5D                  LDK     A,']'
0850  18E2 ^0834$           JR      KEYE            ;CONVERT [ TO ]
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
0852                KEY3:
                    ;CHARS ' TO > (ASCII CODES 27H TO 3EH) ARE CONVERTED USING
                    ;THE SHFT_TB. D=0 FROM BEFORE

0852  5F                    MOV     E,A
0853  21E508                LDK     HL,SHFT_TB - ''''
0856  19                    ADD     HL,DE

0857  7E            KEY3A:  LD      A,[HL]
0858  18DA ^0834$           JR      KEYE
```

Copyright © 1982 Osborne Computer Corporation

```
085A                    KEY4:
                        ;PROCESS CONTROL KEY
                        ;IF CHAR IS BETWEEN A..Z THEN TURN OFF THE 3 HIGH ORDER
                        ;BITS.
                        ;IF CHAR IS BETWEEN ','..'?' IT IS TRANSLATED AS PER TABLE CTL_TB'
                        ;IF CHAR IS THE ARROW KEYS OR THE ']'/'[' KEY THE SLIDE FUNCTIONS
                        ;ARE CALLED.

085A  FE8D                      CMP     LFT_ARW
085C  2833 ^0891$               JRZ     SLIDEL

085E  FE8B                      CMP     RT_ARW
0860  2833 ^0895$               JRZ     SLIDER

0862  FE8A                      CMP     UP_ARW
0864  2840 ^08A6$               JRZ     SLIDEU

0866  FE8C                      CMP     DN_ARW
0868  2840 ^08AA$               JRZ     SLIDED

086A  FE5B                      CMP     HM_SCRN
086C  2850 ^083E$               JRZ     DOHOME

086E  CB63       $              BIT     SHFT_KY,E       ;TEST FOR CNTL SHIFT
0870  2808 ^087A$               JRZ     KEY5            ;IF NOT

0872  FE2F                      CMP     '/'             ;IS IT ?
0874  2004 ^087A$               JRNZ    KEY5            ;IF NOT '?'

0876  3E7F                      LDK     A,07FH          ;DELEAT KEY
0878  188A ^0834$               JR      KEYE
```

Copyright © 1982 Osborne Computer Corporation

```
087A                    KEY5:
087A  FE40                      CMP     '@'
087C  3808 ^0886$               JRC     KEY6            ;GOTO TRANSLATE CHARS FROM TABLE

087E  FE7B                      CMP     'z'+1
0880  30B2 ^0834$               JRNC    KEYE

0882  E61F                      AND     1FH
0884  18AE ^0834$               JR      KEYE
```

Copyright © 1982 Osborne Computer Corporation

```
0886                    KEY6:
0886   FE2C                     CMP     ','
0888   38AA ^0834$              JRC     KEYE            ;NO TRANSLATION IF CHAR BELOW ','

088A   21F808                   LDK     HL,CTL_TB-','
088D   5F                       MOV     E,A
                                                        ;D=0 FROM ABOVE
088E   19                       ADD     HL,DE
088F   18C6 ^0857$              JR      KEY3A
```

```
                        ;SLIDE FUNCTIONS.

0891                    SLIDEL:
0891  0E02                      LDK     C,2
0893  1802 ^0897$               JR      SLR1


0895                    SLIDER:
0895  0EFE                      LDK     C,-2

0897  3A61EF            SLR1:   LD      A,PIAAD         ;GET HORIZONTAL COORD.
089A  81                        ADD     A,C
089B  4F                        MOV     C,A

089C  CD7900                    CALL    OPAD            ;FUNCTION PIA

                        ;SET REPEAT COUNT FOR THESE KEYS (OVERRIDE COUNT SET BY THE KBDRVR)

089F  D1                SLR2:   POP     DE              ;GET RETURN ADRS
08A0  E1                        POP     HL              ;POINTER TO REPEAT ENTRY

08A1  3603                      STO     SLD_RCT,[HL]    ;REPEAT COUNT FOR SLIDE KEYS

08A3  E5                        PUSH    HL
08A4  D5                        PUSH    DE              ;RESTORE STACK

08A5  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
08A6  0E01           SLIDEU: LDK     C,1
08A8  1802 ^08AC$            JR      SLD1

08AA  0EFF           SLIDED: LDK     C,-1

08AC  2162EF         SLD1:   LDK     HL,PIABD        ;MERGE NEW VERTOFFSET TO LOWER 5 BITS OF PIAB
08AF  7E                     LD      A,[HL]
08B0  81                     ADD     A,C
08B1  E61F                   AND     1FH             ;MODIFY CURRENT WITH +1/OR-1
08B3  4F                     MOV     C,A
08B4  7E                     LD      A,[HL]
08B5  E6E0                   AND     0E0H

08B7  B1             SLD2:   OR      C
08B8  4F                     MOV     C,A
08B9  CD8600                 CALL    OPBD
08BC  18E1 ^089F$            JR      SLR2
```

Copyright © 1982 Osborne Computer Corporation


```
08BE                    DOHOME:

                        ;SET DENSITY BIT

08BE  3A61EF                    LD      A,PIAAD         ;GET OLD VALUE
08C1  E601                      ANI     0000_0001B      ;SAVE DENSITY BIT
08C3  F6EA                      ORI     VFLO            ;OR IN HORIZONTAL OFFSET
08C5  4F                        MOV     C,A
08C6  CD7900                    CALL    OPAD            ;FUNCTION PIA

08C9  3A62EF                    LD      A,PIABD
08CC  E6E0                      AND     0E0H            ;HOUSE KEEPING BITS
08CE  4F                        MOV     C,A
08CF  3AEFEF                    LD      A,VRTOFF        ;GET LAST VERTICAL OFFSET
08D2  18E3 ^08B7$               JR      SLD2            ;AND THE VERT TO 0 ALSO
```

Copyright © 1982 Osborne Computer Corporation

```
                        ;KEY CODE TRANSLATION TABLES

08D4                    KYCDTB:
08D4  1B097F7F                  DB      esc, tab, erc, erc

08D8  7F0D275B                  DB      erc, cr, '''', '['

08DC  31323334                  DB      '1', '2', '3', '4'

08E0  35363738                  DB      '5', '6', '7', '8'

08E4  71776572                  DB      'q', 'w', 'e', 'r'

08E8  74797569                  DB      't', 'y', 'u', 'i'

08EC  61736466                  DB      'a', 's', 'd', 'f'

08F0  67686A6B                  DB      'g', 'h', 'j', 'k'

08F4  7A786376                  DB      'z', 'x', 'c', 'v'

08F8  626E6D2C                  DB      'b', 'n', 'm', ','

08FC  8A8D3020                  DB      8ah, 8dh, '0', ' '

0900  2E706F39                  DB      '.', 'p', 'o', '9'

0904  8B8C2D2F                  DB      8bh, 8ch, '-', '/'

0908  3B5C6C3D                  DB      ';', '\', 'l', '='

090C                    SHFT_TB:
090C  2200000000                DB      '"', 00h, 00h, 00h, 00h

0911  3C5F3E3F29                DB      '<', '_', '>', '?', ')'

0916  2140232425                DB      '!', '@', '#', '$', '%'

091B  5E262A2800                DB      '^', '&', '*', '(', 00h

0920  3A002B00                  DB      ':', 00h, '+', 00h

0924  7B1F7D7E          CTL_TB: DB      '{', '_'-40h, '}', '~'

0928  8081828384                DB      80h, 81h, 82h, 83h, 84h

092D  8586878889                DB      85h, 86h, 87h, 88h, 89h

0932  00000060                  DB      00h, 00h, 00H, 60H
```

Copyright ©1982 Osborne Computer Corporation

```
*[ ]

;        +-------------------------------------+
;        ! ENTERED 05/01/81 FROM TNW XEROX, SEH. !
;        +-------------------------------------+

;LAST EDITED AT 09:29 ON 11 NOV 80

;THERE ARE FOUR COMMANDS TO THE 6821

;        00      PERIPHERAL/DIRECTION REGISTER A         CPDRA
;        01      CONTROL REGISTER A                      CCRA
;        10      PERIPHERAL/DIRECTION REGISTER B         CPDRB
;        11      CONTROL REGISTER B                      CCRB

;BIT 2 OF THE CONTROL REGISTER (A AND B) ALLOWS SELECTION OF EITHER
;A PERIPHERAL INTERFACE REGISTER OR A DATA DIRECTION REGISTER.
;A "1" IN BIT 2 SELECTS THE PERIPHERAL REGISTER.

;THE TWO DATA DIRECTION REGISTERS ALLOW CONTROL OF THE DIRECTION
;OF DATA THROUGH EACH CORRESPONDING PERIPHERAL DATA LINE.
;A DATA DIRECTION REGISTER BIT SET AT "0" CONFIGURES
;THE CORRESPONDING PERIPHERAL DATA LINE AS AN INPUT.

;A RESET AT POWER UP HAS THE EFFECT OF ZEROING ALL PIA REGISTERS.
;THIS WILL SET PA0-PA7, PB0-PB7, CA2, AND CB2 AS INPUTS,
;AND ALL INTERRUPTS DISABLED.
;SIGNALS ATN, REN, AND IFC WILL BE DRIVEN LOW
;UNTIL INITIALIZED BY SOFTWARE.

;DATA DIRECTION IS ALWAYS SET FOR OUTPUT FOR THE DATA REGISTER.
;DATA MUST BE SET TO ALL ONES WHEN INPUTTING.
;THE INTERFACE IS IN SOURCE HANDSHAKE MODE IF DATA ENABLE (PB0)
;IS SET TO "0", AND IN ACCEPTOR HANDSHAKE MODE IF SET TO "1".
;WHEN SWITCHING FROM SOURCE TO ACCEPTOR HANDSHAKE,
;ATN WILL ALWAYS BE LOW.
;TAKE CONTROL CAN ONLY BE CALLED FOLLOWING A GO TO STANDBY.
;AFTER A FATAL ERROR, PERFORM AN IFC RESET.

;STANDARD VALUES USED:

;CCRA    0011(IFC)(DIR)10

;CCRB    0011(REN)(DIR)00

;CPDRA   SOURCE          DIRECTION       1111_1111
;                        DATA            DATA

;CPDRA   ACCEPTOR        DIRECTION       1111_1111
;                        DATA            1111_1111

;CPDRB   SOURCE          DIRECTION       0011_1111
;                        DATA            000A_0010       ;A = ATN

;CPDRB   ACCEPTOR        DIRECTION       1101_0111
;                        DATA            0100_0101
```

Copyright © 1982 Osborne Computer Corporation

```
;PIA SIGNAL DEFINITIONS:
;ALL SIGNALS ARE LOW ON THE IEEE BUS WHEN PIA REGISTER CONTAINS "1".

;       PA0     DIO 1
;       PA1     DIO 2
;       PA2     DIO 3
;       PA3     DIO 4
;       PA4     DIO 5
;       PA5     DIO 6
;       PA6     DIO 7
;       PA7     DIO 8

;       CA1     SRQ
;       CA2     IFC

;       PB0     ENABLE DATA OUT         (ENABLED WHEN "0")
;       PB1     ENABLE NDAC/NRFD        (ENABLED WHEN "0")
;       PB2     ENABLE EOI/DAV          (ENABLED WHEN "0")
;       PB3     EOI
;       PB4     ATN
;       PB5     DAV
;       PB6     NDAC
;       PB7     NRFD

;       CB1     NOT USED
;       CB2     REN

;CONTROL WORD FORMAT

;[  7  ][  6  ][  5  ][  4  ][  3  ][  2  ][  1  ][  0  ]

;[IRQA1][IRQA2][    CA2 CONTROL    ][ DDRA][ CA1 CONTROL]
;[IRQB1][IRQB2][    CB2 CONTROL    ][ DDRB][ CB1 CONTROL]

;       IRQA1   0       INTERRUPT FLAG SET BY FALL OF SRQ
;       IRQA2   0       NOT USED
;       CA2     110     SET IFC HIGH
;               111     SET IFC LOW
;       DDRA    0       R/W DATA DIRECTION REGISTER A
;               1       R/W PERIPHERAL REGISTER A
;       CA1     10      SET IRQA1 HIGH ON RISE OF SRQ

;       IRQB1   0       NOT USED
;       IRQB2   0       NOT USED
;       CB2     110     SET REN HIGH
;               111     SET REN LOW
;       DDRB    0       R/W DATA DIRECTION REGISTER B
;               1       R/W PERIPHERAL REGISTER B
;       CB1     00      NOT USED
```

```
;BIOS CALL 1:   CONTROL OUT

;       CAN BE CALLED WHILE IN ANY STATE.

;       EXITS IN THE CONTROLLER STANDBY STATE (ATN HIGH),
;       SOURCE HANDSHAKE MODE

;PARAMETER PASSED IN REGISTER C:

;       BIT 0   IF "1", THE IFC SIGNAL IS SET LOW FOR 100 MICRO-SEC
;               AND ALL PIA SIGNALS ARE INITIALIZED

;       BIT 2 1
;       0 X     NO ACTION
;       1 0 SETS REN HIGH
;       1 1 SETS REN LOW
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0936                    IE.CO:
0936                            PROC

0936  F5                        PUSH    AF
0937  E5                        PUSH    HL

0938  CB41       $              BIT     0,C             ;CHECK IFC SUB-COMMAND
093A  282B ^0967$               JRZ     :B1C20

                        ;INITIALIZE ALL IEEE-488 SIGNALS

093C  210129                    LK      HL,CCRA
093F  363A                      STO     0011_1010B,[HL] ;ENABLE SRQ AND SET IFC-OUT LOW

0941  3EFF                      LK      A,1111_1111B    ;DIRECT DATA OUT
0943  320029                    STO     A,CPDRA
0946  363E                      STO     0011_1110B,[HL]

0948  AF                        XRA     A
0949  320029                    STO     A,CPDRA
094C  210329                    LK      HL,CCRB
094F  3630                      STO     0011_0000B,[HL] ;SET REN-OUT HIGH

0951  3E3F                      LK      A,0011_1111B    ;DIRECTION FOR SOURCE HANDSHAKE
0953  320229                    STO     A,CPDRB
0956  3634                      STO     0011_0100B,[HL]

0958  3E02                      LK      A,0000_0010B    ;VALUES FOR SOURCE HANDSHAKE
095A  320229                    STO     A,CPDRB

                        ;LEAVE IFC LOW FOR 100 MICRO-SEC

095D  3E19                      LK      A,25            ;DELAY 100 MICRO-SEC

095F  3D                :B1C10: DEC     A
0960  20FD ^095F$               JRNZ    :B1C10

0962  3E36                      LK      A,0011_0110B    ;SET IFC HIGH
0964  320129                    STO     A,CCRA

0967  CB51       $ :B1C20:      BIT     2,C             ;CHECK REN SUB-COMMAND
0969  280B ^0976$               JRZ     :B1C40

                        ;SET/CLEAR REN

096B  3E34                      LK      A,0011_0100B
096D  CB49       $              BIT     1,C
096F  2802 ^0973$               JRZ     :B1C30

0971  3E3C                      LK      A,0011_1100B

0973  320329            :B1C30: STO     A,CCRB

0976  E1                :B1C40: POP     HL
0977  F1                        POP     AF
0978  C9                        RET
```

```
0979                    IE.SI:
                        ;BIOS CALL 2. STATUS IN
                        ;CAN BE CALLED ONLY WHILE IN SOURCE HANDSHAKE MODE.
                        ;BIT 0 OF REGISTER A SET IF SRQ IS LOW

0979                            PROC

0979  E5                        PUSH    HL

097A  3A0029                    LD      A,CPDRA         ;CLEAR IRQA1
097D  210229                    LK      HL,CPDRB        ;PULSE ENABLE ndac/nrfd
0980  CB8E        $             CBIT    1,[HL]
0982  CBCE        $             SBIT    1,[HL]
0984  3A0129                    LD      A,CCRA          ;SET SRQ VALUE IN A
0987  E680                      AND     1000_0000B
0989  07                        RLC     A

098A  E1                        POP     HL
098B  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
098C                IE.GTS:
                    ;BIOS CALL 3 GO TO STANDBY
                    ;CAN BE CALLED ONLY WHILE IN SOURCE HANDSHAKE MODE
                    ;ENTRY
                    ;NONE

098C                        PROC

098C  F5                    PUSH    AF

098D  3E02                  LK      A,0000_0010B    ;SET ATN HIGH
098F  320229                STO     A,CPDRB
0992  AF                    XOR     A               ;FLOAT DATA BUS
0993  320029                STO     A,CPDRA

0996  F1                    POP     AF
0997  C9                    RET
```

```
0998                    IE.TC:
                        ;BIOS CALL 4 TAKE CONTROL
                        ;CAN BE CALLED ONLY WHILE IN THE CONTROLLER STANDBY STATE (ATN HIGH).
                        ;EXITS IN THE CONTROLLER ACTIVE STATE (ATN LOW), SOURCE HANDSHAKE MODE.
                        ;BIT 0 OF REGISTER C SET TO TAKE CONTROL ASYNCHRONOUS

                        ;EXIT
                                ;A      =       ;ERROR CODE

0998                            PROC

0998  E5                        PUSH    HL

0999  210229                    LK      HL,CPDRB
099C  CB41       $              BIT     0,C
099E  2022 ^09C2$               JRNZ    :B4C30

                        ;TAKE CONTROL SYNCHRONOUSLY

09A0  3607                      STO     0000_0111B,[HL] ;DISABLE DRIVERS
09A2  3A0329                    LD      A,CCRB
09A5  CB97       $              CBIT    2,A
09A7  320329                    STO     A,CCRB
09AA  3607                      STO     1101_0111B,[HL] ;DIRECTION REGISTER

09AC  CBD7       $              SBIT    2,A
09AE  320329                    STO     A,CCRB

09B1  3685                      STO     1000_0101B,[HL] ;SET NRFD LOW
09B3  3E19                      LK      A,25

09B5  CB6E       $ :B4C10:      BIT     5,[HL]
09B7  2807 ^09C0$               JRZ     :B4C20          ;DATA VALID HAS DROPPED

09B9  3D                        DEC     A
09BA  20F9 ^09B5$               JRNZ    :B4C10          ;WAIT 100 MICRO-SEC

09BC  3E81                      LK      A,1000_0001B    ;SET DATA VALID TIMEOUT ERROR
09BE  1816 ^09D6$               JR      :B4C40

09C0  36C5         :B4C20:      STO     1100_0101B,[HL] ;SET NDAC LOW

09C2  CBE6       $ :B4C30:      SBIT    4,[HL]          ;SET ATN LOW

                        ;SET-UP FOR SOURCE HANDSHAKE

09C4  3A0329                    LD      A,CCRB
09C7  CB97       $              CBIT    2,A
09C9  320329                    STO     A,CCRB
09CC  363F                      STO     0011_1111B,[HL] ;DIRECTION REGISTER

09CE  CBD7       $              SBIT    2,A
09D0  320329                    STO     A,CCRB
09D3  3612                      STO     0001_0010B,[HL] ;CONTROL SIGNAL INITIAL VALUE

09D5  AF                        XOR     A               ;CLEAR ERROR CODE

09D6  E1           :B4C40:      POP     HL
09D7  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright ©1982 Osborne Computer Corporation

```
09D8                    IE.OIM:
                        ;BIOS CALL 5 OUTPUT INTERFACE MESSAGE
                        ;CAN BE CALLED WHILE IN ANY MODE OR STATE
                        ;EXITS IN THE SOURCE HANDSHAKE MODE WITH ATN LOW.

                        ;EXIT
                                ;A      =       ERROR CODE
                                ;C      =       MULTI-LINE MESSAGE

09D8                            PROC

09D8  E5                        PUSH    HL

09D9  210229                    LK      HL,CPDRB
09DC  CBE6        $             SBIT    4,[HL]          ;SET ATN LOW
09DE  CB46        $             BIT     0,[HL]
09E0  2825 ^0A07$               JRZ     IE.SHK

                        ;SET-UP FOR SOURCE HANDSHAKE

09E2  3617                      STO     0001_0111B,[HL] ;DISABLE DRIVERS
09E4  3A0329                    LD      A,CCRB
09E7  CB97        $             CBIT    2,A
09E9  320329                    STO     A,CCRB
09EC  363F                      STO     0011_1111B,[HL] ;DIRECTION REGISTER

09EE  CBD7        $             SBIT    2,A
09F0  320329                    STO     A,CCRB

                        ;FLOAT EXTERNAL DATA BUS

09F3  AF                        XOR     A
09F4  320029                    STO     A,CPDRA
09F7  3612                      STO     0001_0010B,[HL] ;CONTROL SIGNAL INITIAL VALUE

09F9  180C ^0A07$               JR      IE.SHK
```

Copyright © 1982 Osborne Computer Corporation

```
09FB                    IE.ODM:
                        ;BIOS CALL 6 OUTPUT DEVICE MESSAGE
                        ;CAN BE CALLED ONLY WHILE IN THE SOURCE HANDSHAKE MODE WITH ATN HIGH OR LOW.
                        ;EXITS IN THE SOURCE HANDSHAKE MODE WITH ATN HIGH.

                        ;EXIT
                                ;C      =       MULTI-LINE MESSAGE
                                ;B      =       EOI REQUEST
                                ;A      =       ERROR CODE

09FB                            PROC

09FB  E5                        PUSH    HL

09FC  210229                    LK      HL,CPDRB
09FF  CBA6       $              CBIT    4,[HL]          ;SET ATN HIGH
0A01  CB40       $              BIT     0,B             ;CHECK IF EOI REQUESTED
0A03  2802 ^0A07$               JRZ     IE.SHK

0A05  CBDE       $              SBIT    3,[HL]

                        ;PERFORM SOURCE HANDSHAKE

0A07  CB6E       $ IE.SHK:      BIT     5,[HL]          ;
0A09  201B ^0A26$               JRNZ    :B6C50          ;DAC TIMEOUT RE-ENTRY

0A0B  79                        MOV     A,C             ;PLACE DATA ON BUS
0A0C  320029                    STD     A,CPDRA
0A0F  3E0A                      LK      A,10

0A11  CB7E       $ :B6C20:      BIT     7,[HL]
0A13  2807 ^0A1C$               JRZ     :B6C30          ;READY FOR DATA

0A15  3D                        DEC     A
0A16  20F9 ^0A11$               JRNZ    :B6C20          ;WAIT FOR 100 MICRO-SEC

0A18  3E82                      LK      A,1000_0010B    ;SET RFD TIMEOUT ERROR
0A1A  181F ^0A3B$               JR      :B6C80

0A1C  CB76       $ :B6C30:      BIT     6,[HL]
0A1E  2004 ^0A24$               JRNZ    :B6C40          ;DATA ACCEPTED LOW

0A20  3E81                      LK      A,1000_0001B    ;SET DEVICE NOT PRESENT ERROR
0A22  1817 ^0A3B$               JR      :B6C80

0A24  CBEE       $ :B6C40:      SBIT    5,[HL]          ;SET DAV LOW

0A26  3EFF         :B6C50:      LK      A,255

0A28  CB76       $ :B6C50:      BIT     6,[HL]
0A2A  2807 ^0A33$               JRZ     :B6C70          ;DATA ACCEPTED

0A2C  3D                        DEC     A
0A2D  20F9 ^0A28$               JRNZ    :B6C50          ;WAIT 1000 MICRO-SEC

0A2F  3E84                      LK      A,1000_0100B    ;SET DAC TIMEOUT ERROR
0A31  1808 ^0A3B$               JR      :B6C80

0A33  CBAE       $ :B6C70:      CBIT    5,[HL]          ;SET DAV HIGH
```

```
0A35  CB9E       $          CBIT    3,[HL]          ;SET EOI HIGH
0A37  AF                    XOR     A               ;REMOVE DATA FROM BUS
0A38  320029                STO     A,CPORA

0A3B  E1           :B6C80:  POP     HL
0A3C  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
0A3D                    IE.IDM:
                        ;BIOS CALL 7 INPUT DEVICE MESSAGE
                        ;CAN BE CALLED WHILE IN ANY MODE OR STATE
                        ;EXITS IN THE ACCEPTOR HANDSHAKE MODE WITH ATN HIGH.
                        ;EXIT
                                ;L      =       ERROR CODE
                                ;A      =       DEVICE MESSAGE
                                ;H      =       DEVICE MESSAGE

0A3D                            PROC

0A3D  D5                        PUSH    DE

0A3E  EB                        EX      DE,HL           ;SAVE RE-ENTRY DATA
0A3F  210229                    LK      HL,CPDRB
0A42  CB46        $             BIT     0,[HL]
0A44  201A ^0A60$               JRNZ    :B7C10

                        ;SET-UP FOR ACCEPTOR HANDSHAKE

0A46  3617                      STO     0001_0111B,[HL] ;DISABLE DRIVERS

0A48  3A0329                    LD      A,CCRB
0A4B  CB97        $             CBIT    2,A
0A4D  320329                    STO     A,CCRB
0A50  3607                      STO     1101_0111B,[HL] ;DIRECTION REGISTER

0A52  CBD7        $             SBIT    2,A
0A54  320329                    STO     A,CCRB

0A57  3EFF                      LK      A,1111_1111B    ;FLOAT INTERNAL DATA BUS
0A59  320029                    STO     A,CPDRA
0A5C  3655                      STO     0101_0101B,[HL] ;CONTROL SIGNALS INITIAL VALUE
0A5E  3645                      STO     0100_0101B,[HL] ;SET ATN HIGH

                        ;PERFORM ACCEPTOR HANDSHAKE

0A60  CB76        $ :B7C10: BIT     6,[HL]
0A62  2820 ^0A84$               JRZ     :B7C50          ;DATA INVALID TIMEOUT ERROR RE-ENTRY

0A64  CBBE        $             CBIT    7,[HL]          ;SET NRFD HIGH
0A66  3E0A                      LK      A,10

0A68  CB6E        $ :B7C20: BIT     5,[HL]
0A6A  2008 ^0A74$               JRNZ    :B7C30          ;DATA VALID

0A6C  3D                        DEC     A
0A6D  20F9 ^0A68$               JRNZ    :B7C20          ;WAIT 100 MICRO-SEC

0A6F  118200                    LK      DE,1000_0010B   ;SET DATA VALID TIMEOUT ERROR
0A72  1821 ^0A95$               JR      :B7C80

0A74  CBFE        $ :B7C30: SBIT    7,[HL]          ;SET NRFD LOW
0A76  3A0029                    LD      A,CPDRA         ;READ DATA
0A79  57                        MOV     D,A
0A7A  1E00                      LK      E,0             ;READ EOI
0A7C  CB5E        $             BIT     3,[HL]
0A7E  2802 ^0A82$               JRZ     :B7C40
```

Copyright © 1982 Osborne Computer Corporation

```
0A80  1E01                    LK     E,1

0A82  CB86         $ :B7C40: CBIT   6,[HL]       ;SET NDAC HIGH

0A84  3EFF           :B7C50: LK     A,255

0A86  CB6E         $ :B7C60: BIT    5,[HL]
0A88  2809 ^0A93$            JRZ    :B7C70       ;DATA VALID DROPPED

0A8A  3D                     DEC    A
0A8B  20F9 ^0A86$            JRNZ   :B7C60       ;WAIT 1000 MICRO-SEC

0A8D  CBD3         $         SBIT   2,E          ;SET DATA INVALID TIMEOUT ERROR
0A8F  CBFB         $         SBIT   7,E
0A91  1802 ^0A95$            JR     :B7C80

0A93  CBF6         $ :B7C70: SBIT   6,[HL]       ;SET NDAC LOW

0A95  EB             :B7C80: EX     DE,HL        ;MOVE RESULTS TO REGISTERS A AND HL
0A96  7C                     MOV    A,H

0A97  D1                     POP    DE
0A98  C9                     RET
```

Copyright © 1982 Osborne Computer Corporation

```
0A99                    IE.PP:
                        ;BIOS CALL 8 PARALLEL POLL
                        ;CAN BE CALLED ONLY WHILE IN THE SOURCE HANDSHAKE MODE WITH ATN HIGH OR LOW.
                        ;EXITS IN THE SOURCE HANDSHAKE MODE WITH ATN LOW.

                        ;EXIT
                                ;A      =       PARALLEL POLL VALUE

0A99                            PROC

0A99   E5                       PUSH    HL

0A9A   210029                   LK      HL,CPDRA
0A9D   3E1B                     LK      A,0001_1011B     ;FORM PARALLEL POLL
0A9F   320229                   STO     A,CPDRB
0AA2   36FF                     STO     1111_1111B,[HL] ;FLOAT INTERNAL DATA BUS

0AA4   7E                       LD      A,[HL]           ;READ PARALLEL POLL DATA
0AA5   3600                     STO     0,[HL]           ;RE-STORE SOURCE HANDSHAKE MODE

0AA7   210229                   LK      HL,CPDRB
0AAA   3612                     STO     0001_0010B,[HL]

0AAC   E1                       POP     HL
0AAD   C9                       RET
```

Copyright © 1982 Osborne Computer Corporation

```
*[ ]

;IEEE drivers:

;The routines IEINSTAT, IEINP and IEOUT are used to
;transfer characters to and from an IEEE device attached to the
;OSBORNE IEEE port.  The address of the device is specified in
;the cell IE_ADRS.
;The function IEINSTAT returns the status of the input device.
;Unfortunately there is no standard way by which an IEEE device
;indicates that it has a character.  In order to determine this, one
;has to read the character device.  As a CP/M transient can call
;IEINSTAT many times before calling IEINP to read a char, and IEINSTAT
;has to read the char to determine the status, the character read has to
;be buffered until call to IEINP is made.  IEINSTAT reads the device
;only when the buffer is empty.  As zeros are used to indicate
;that the bfr is empty, a null character can not be read from the
;IEEE device.
```

Copyright ©1982 Osborne Computer Corporation


```
0AAE                IEOSTAT:
                    ;returns status of IEEE
                    ;IEEE always appears to be ready

0AAE                        PROC

0AAE  F6FF                  ORI     0FFh
0AB0  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0AB1                IEINSTAT:
                    ;gets status of the input device attached to IEEE port
                    ;if a char is present in IE_char then return with 0FFH status
                    ;else
                    ;make device talker
                    ;Read the device
                    ;if char read then
                    ; store in bfr
                    ;make untalk
                    ;return with status of buffer

0AB1                        PROC

0AB1  3ADCEF                LDA     IE_CHAR
0AB4  B7                    ORA     A
0AB5  2803 ^0ABA$           JRZ     IEI10                   ;if char present then

0AB7  F6FF                  ORI     0FFh                    ;return with 0FFH status
0AB9  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0ABA                  IEI10:
                      ;make talker

0ABA  3ADBEF                  LDA     IE_ADRS
0ABD  C640                    ADI     IE_TALK              ;get primary address
0ABF  4F                      MOV     C,A
0AC0  CD0809                  CALL    IE.OIM               ;output interface message
0AC3  B7                      ORA     A
0AC4  20F4 ^0ABA$             JRNZ    IEI10                ;try again if error

0AC6                  IEI20:
                      ;read a char.

0AC6  CD3D0A                  CALL    IE.IDM
0AC9  CB7D       $            BIT     7,L
0ACB  2801 ^0ACE$             JRZ     IEI30                ;if error then

0ACD  AF                      XRA     A                    ;inicate no char recvd

0ACE  32DCEF          IEI30:  STA     IE_CHAR              ;stor the char

0AD1                  IEI40:
                      ;make untalk

0AD1  0E5F                    LDK     C,IE_UTLK
0AD3  CD0809                  CALL    IE.OIM
0AD6  B7                      ORA     A
0AD7  20F8 ^0AD1$             JRNZ    IEI40

                      ;return with status of the char

0AD9  3ADCEF                  LDA     IE_CHAR
0ADC  B7                      ORA     A
0ADD  C8                      RZ

0ADE  F6FF                    ORI     0FFh
0AE0  C9                      RET
```

```
0AE1                IEINP:
                    ;Reads a character from IEEE port

0AE1                        PROC

0AE1  CD810A                CALL    IEINSTAT
0AE4  28FB ^0AE1$           JRZ     IEINP           ;wait till char avail

0AE6  21DCEF                LDK     HL,IE_CHAR
0AE9  7E                    LD      A,[HL]
0AEA  3600                  STO     0,[HL]          ;clear the buffer
0AEC  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0AED                    IEOUT:
                        ;Outputs the character in reg C to IEEE port
                        ;Uses ROM resident primitives.

0AED                            PROC

0AED  C5                        PUSH    BC              ;save the char
                                                        ;make listener
0AEE  3ADBEF            IE005:  LDA     IE_ADRS
0AF1  C620                      ADI     IE_LSTN         ;get primary address
0AF3  4F                        MOV     C,A
0AF4  CDD809                    CALL    IE.DIM          ;output interface message
0AF7  B7                        ORA     A
0AF8  20F4 ^0AEE$               JRNZ    IE005           ;try again if error

0AFA  C1                        POP     B
0AFB  0600                      LDK     B,0             ;do not send eoi

0AFD  C5                IE022:  PUSH    B               ;save char again in case of retry
0AFE  CDFB09                    CALL    IE.ODM
0B01  C1                        POP     B
0B02  B7                        ORA     A
0B03  20F8 ^0AFD$               JRNZ    IE022           ;try again if error

0B05                    IE040:
                        ;make unlisten

0B05  0E3F                      LDK     C,IE_ULST
0B07  CDD809                    CALL    IE.DIM
0B0A  B7                        ORA     A
0B0B  20F8 ^0B05$               JRNZ    IE040

0B0D  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
;The Parallel port is actually the IEEE port driven with the centronix
;protocol.  The bit assignements of the PIA and PIB are as follows:
;PIA0-7 = data bus
;PIB0 = 0, data bus is output. 1, data bus is input
;PIB1 = set to 1.
;PIB2 = set to 0.
;PIB3 = 0 output, 1 Input
;PIB4 = not used
;PIB5 = output strobe. Active = 1.
;PIB6 = 0, printer busy. 1, printer is ready.
;PIB7 = not used.

;CA2 = going low indicates to device that we are busy.
;CA1 = low to high transition gates input data to port a.

;The port is bidirectional but only one direction
;can be active at any time. The direction of port is determined
;by which routines are called.  If postat or parout are
;called, it is made an output port and an input port if
;pistat or parinp are called.
```

Copyright © 1982 Osborne Computer Corporation

```
0B0E                 CV2OP:
                     ;initializes the port to a PArallel output port.

0B0E                          PROC

0B0E   3A0EEF                 LDA     PP.MODE
0B11   FE01                   CPI     PP.OUT
0B13   C8                     RZ                      ;return when in output mode

                     ;set port a to output on all lines

0B14   3E2A                   LDK     A,PA.CDR
0B16   320129                 STA     PA.CTL          ;select direction reg
0B19   3EFF                   LDK     A,PA.DRD
0B1B   320029                 STA     PA.DIR          ;output constant to dir. reg to put a port in output mode

0B1E   3E2E                   LK      A,PA.CDT
0B20   320129                 STA     PA.CTL          ;select port a data reg.

0B23   3E00                   LK      A,PB.CDR
0B25   320329                 STA     PB.CTL          ;select port b direction
0B28   3EBF                   LK      A,PB.DR
0B2A   320229                 STA     PB.DIR          ;all lines are output except the output busy signal on bit  6

0B2D   3E04                   LK      A,PB.CDT
0B2F   320329                 STA     PB.CTL          ;select data register
0B32   3E02                   LK      A,PB.DTD
0B34   320229                 STA     PB.DTA          ;initialize port b data

0B37   3E01                   LK      A,PP.OUT
0B39   320EEF                 STA     PP.MODE
0B3C   C9                     RET
```

Copyright © 1982 Osborne Computer Corporation

```
0B3D                    CV2IP:
                        ;initializes the port to a parallel input port.

0B3D                                PROC

0B3D  3A0EEF                        LDA     PP.MODE
0B40  FE02                          CPI     PP.IN
0B42  C8                            RZ                      ;return when in input mode

                        ;set port a to input on all lines

0B43  3E2A                          LK      A,PA.CDR
0B45  320129                        STA     PA.CTL          ;select direction reg
0B48  3E00                          LK      A,PA.DRI
0B4A  320029                        STA     PA.DIR          ;output constant to dir. reg to put a port in input mode

0B4D  3E2E                          LK      A,PA.CDT
0B4F  320129                        STA     PA.CTL          ;select port a data reg.

0B52  3E00                          LK      A,PB.CDR
0B54  320329                        STA     PB.CTL          ;select port b direction
0B57  3E3F                          LK      A,PB.DR
0B59  320229                        STA     PB.DIR          ;all lines are output except the output busy signal on bit  6

0B5C  3E04                          LK      A,PB.CDT
0B5E  320329                        STA     PB.CTL          ;select data register
0B61  3E08                          LK      A,PB.DTI
0B63  320229                        STA     PB.DTA          ;initialize port b data

0B66  3E02                          LK      A,PP.IN
0B68  320EEF                        STA     PP.MODE
0B6B  C9                            RET
```

Copyright ©1982 Osborne Computer Corporation


```
0B6C                POSTAT:
                    ;gets status of the parallel (centronix) printer attached to the IEEE port

0B6C                        PROC

0B6C  CD0E0B                CALL    CV2OP           ;convert to output
0B6F  3A0229                LDA     PB.DTA          ;get port b data
0B72  E640                  ANI     PP.DRDY
0B74  C8                    RZ

0B75  F6FF                  ORI     0FFH

0B77  C9            POS10:  RET
```

Copyright © 1982 Osborne Computer Corporation

```
0B78                PISTAT:
                    ;gets status of the input device attached to the parallel port

0B78                        PROC

0B78  CD3D0B                CALL    CV2IP
0B7B  3ADDEF                LDA     PIACTL
0B7E  E680                  ANI     PP.IRDY
0B80  2009 ^0B8B$           JRNZ    PIS20           ;if saved status indicates there is a char in the PIA

0B82  3A0129                LDA     PA.CTL
0B85  32DDEF                STA     PIACTL          ;this is saved as reading the
                                                    ;pia clears the status

0B88  E680                  ANI     PP.IRDY
0B8A  C8                    RZ

0B8B  F6FF          PIS20:  ORI     0FFH
0B8D  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0B8E                PARINP:
                    ;inputs a character from PArallel port.

0B8E                        PROC

0B8E  CD780B                CALL    PISTAT
0B91  28FB ^0B8E$           JRZ     PARINP          ;wait till char in pia

0B93  AF                    XRA     A
0B94  32DDEF                STA     PIACTL          ;clear saved status
0B97  3A0029                LDA     PA.DTA
0B9A  2F                    CMA                     ;invert data
0B9B  4F                    MOV     C,A             ;also in c
0B9C  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0B9D                 PAROUT:
                     ;outputs the character in c to the IEEE port treating the port as a parallel port.

0B9D                           PROC

0B9D  CD6C0B                   CALL    POSTAT
0BA0  28FB ^0B9D$              JRZ     PAROUT

0BA2  79                       MOV     A,C
0BA3  2F                       CMA                          ;invert data
0BA4  320029                   STA     PA.DTA
0BA7  3E22                     LK      A,PB.DTO+STRB
0BA9  320229                   STA     PB.DTA               ;set strobe
0BAC  3E02                     LK      A,PB.DTO
0BAE  320229                   STA     PB.DTA               ;clear strobe
0BB1  C9                       RET
```

```
                         *[5]
0BB2                     SIRST:
                         ;Master reset SIO
                         ;ENTRY
                         ;C      =       SI.S16 or SI.S64 for 1200/300 baud

                         ;EXIT
                         ;NONE
0BB2                             PROC

0BB2  3E57                       LDK     A,SI.MRST
0BB4  32002A                     STO     A,H.SCTRL       ;master reset

0BB7  79                         MOV     A,C
0BB8  32C1EF                     STO     A,ACIAD         ;last-command cell
0BBB  32002A                     STO     A,H.SCTRL       ;select SIO

0BBE  C9                         RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0BBF                READER:
                    ;Input one byte from reader port
                    ;ENTRY
                    ;None

                    ;EXIT
                    ;C       =       character read

0BBF                        PROC

0BBF  CDE80B                CALL    ACISTAT
0BC2  E601                  ANI     SI.RRDY
0BC4  28F9 ^0BBF$           JRZ     READER          ;if not ready

0BC6  21DAEF                LDK     HL,SERFLG
0BC9  CB86      $           CBIT    0,[HL]

0BCB  3A012A                LD      A,H.SREC        ;get data
0BCE  4F                    MOV     C,A             ;C=A
0BCF  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0BD0                SLST:
                    ;Get list device status
                    ;ENTRY
                    ;NONE

                    ;EXIT
                    ;A       =       0, IF NOT READY
                    ;A       =       0FFh IF READY
                    ;ZBIT    =       SET IF NOT READY FOR OUTPUT

0BD0                        PROC

0BD0  CDE80B                CALL    ACISTAT
0BD3  E602                  ANI     SI.TRDY
0BD5  C8                    RZ                      ;RETURN

0BD6  F6FF                  ORI     0FFH
0BD8  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0BD9                    LIST:
                        ;Output one byte to list port
                        ;ENTRY
                        ;C       =       character to outout

                        ;EXIT
                        ;NONE
0BD9                            PROC

0BD9  CD000B                    CALL    SLST            ;GET STATUS
0BDC  28FB ^0BD9$               JRZ     LIST            ;LOOP

0BDE  79                        MOV     A,C
0BDF  32012A                    STO     A,H.SXMT        ;send chr

0BE2  21DAEF                    LDK     HL,SERFLG
0BE5  CB8E      $               CBIT    1,[HL]

0BE7  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0BE8                  ACISTAT:
                      ;RETURN STATUS OF THE SERIAL PORT
                      ;ENTRY
                      ;NONE

                      ;EXIT
                      ;A       =       STATUS REG

0BE8                          PROC

0BE8  C5                      PUSH    BC

0BE9  3A012C                  LD      A,H.VID+1
0BEC  0F                      RRC     A
0BED  E620                    ANI     20h
0BEF  4F                      MOV     C,A

0BF0  3A002A                  LD      A,H.SSTS
0BF3  E60F                    ANI     0DFh
0BF5  B1                      ORA     C
0BF6  4F                      MOV     C,A

0BF7  3ADAEF                  LD      A,SERFLG
0BFA  E603                    ANI     03
0BFC  B1                      ORA     C
0BFD  32DAEF                  STO     A,SERFLG

0C00  C1                      POP     BC

0C01  C9                      RET
```

Copyright © 1982 Osborne Computer Corporation

```
                        *[6]

0C02                    RDRV:
                        ;RESET DRIVE
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;ZBIT   =       RESET IF ERROR

0C02                            PROC

0C02  3E0A                      LDK     A,NRETRY
0C04  3205EF            :LOOP:  STO     A,RTRY
0C07  CD710E                    CALL    SELDRV          ;SELECT DRIVE
0C0A  3805 ^0C11$               JRC     :1

0C0C  CD080C                    CALL    HOME            ;HOME DRIVE
0C0F  3009 ^0C1A$               JRNC    :END            ;IF GOOD

0C11  3A05EF            :1:     LD      A,RTRY
0C14  3D                        DEC     A
0C15  20ED ^0C04$               JRNZ    :LOOP

0C17  3C                        INC     A               ;MAKE NOT ZERO
0C18  37                        STC                     ;INDICATE ERROR
0C19  C9                        RET

0C1A  AF                :END:   XRA     A
0C1B  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0C1C                    RSEC:
                        ;READ SECTOR
                        ;*NOTE*
                        ;       No retries are performed at this level
                        ;ENTRY
                        ;B      =       NUMBER OF SECTORS

                        ;EXIT
                        ;HL     =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                        ;ZBIT   =       RESET IF ERROR
                        ;A      =       NONZERO IF ERROR
                                ;RTRY   =       1 IF ERROR(SO OLD CBIOS DOESN'T DO RETRYS)

0C1C                            PROC

0C1C  3E80                      LDK     A,D.RDS
0C1E  32D1EF                    STD     A,R_WCOM

0C21  1805 ^0C28$               JR      R_WSEC
```

Copyright © 1982 Osborne Computer Corporation

```
0C23                    WSEC:
                        ;WRITE A SECTOR
                        ;*NOTE*
                        ;       No retries are performed at this level
                        ;ENTRY
                        ;B      =       NUMBER OF SECTORS

                        ;EXIT
                        ;HL     =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                        ;ZBIT   =       RESET IF ERROR
                        ;A      =       NONZERO IF ERROR
                                ;RTRY   =       1 IF ERROR(SO OLD CBIOS DOESN'T DO RETRYS)

0C23                            PROC

0C23    3EA0                    LDK     A,D.WRTS
0C25    32D1EF                  STO     A,R_WCOM

                        ;FALLS THROUGH TO "R_WSEC"
```

Copyright ©1982 Osborne Computer Corporation


```
0C28                    R_WSEC:
                        ;READ OR WRITE SEGMENT
                        ;ENTRY
                        ;B       =       NUMBER OF SECTORS TO READ OR WRITE
                                ;R_WCOM =       D.RDS OR D.WRTS

                        ;EXIT
                        ;HL      =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                        ;ZBIT    =       RESET IF ERROR
                                ;RTRY    =       0 IF GOOD, 1 IF ERROR

0C28                            PROC

0C28  ED43CAEF  $               STO     BC,NUMSEC       ;SAVE BC

0C2C  3E0A                      LDK     A,NRETRY
0C2E  3205EF                    STO     A,RTRY          ;SET RETRY NUMBER

                        ;SELECT DRIVE

0C31  CD710E            :RLOOP: CALL    SELDRV          ;TURN DRIVE ON
0C34  2805 ^0C3B$               JRZ     :1              ;IF DRIVE ON DON'T READ ADDRESS

                        ;SET "D.TRKR" TO HEAD POSITION

0C36  CDA50D                    CALL    RADR            ;READ ADDRESS AND SET CONTROLLER
0C39  3820 ^0C5B$               JRC     :ERR            ;STOP

                        ;SEEK

0C3B  CDFA0C            :1:     CALL    SEEK            ;SEEK TO TRACK
0C3E  381B ^0C5B$               JRC     :ERR            ;STOP

                        ;READ OR WRITE

0C40  ED4BCAEF  $               LD      BC,NUMSEC       ;B = NUMBER OF SECTORS TO R/W
0C44  CD3C0D                    CALL    RD_WRT          ;READ/WRITE PER R_WCOM
0C47  3017 ^0C60$               JRNC    :END            ;IF GOOD

                        ;RETRY?

0C49  2105EF                    LDK     HL,RTRY         ;GET RETRYS
0C4C  35                        DEC     [HL]
0C4D  280C ^0C5B$               JRZ     :ERR            ;NO MORE RETRYS

0C4F  7E                        LD      A,[HL]          ;GET NUMBER OF RETRY
0C50  FE09                      CMP     NRETRY-1
0C52  200D ^0C31$               JRNZ    :RLOOP          ;LOOP IF NOT FIRST RETRY

0C54  CDA50D                    CALL    RADR            ;CHECK TRACK ON THE FIRST RETRY
0C57  3802 ^0C5B$               JRC     :ERR            ;STOP IF ERROR

0C59  1806 ^0C31$               JR      :RLOOP          ;LOOP

0C5B  3E01              :ERR:   LDK     A,1             ;INDICATE ERROR
0C5D  B7                        ORA     A
0C5E  1801 ^0C61$               JR      :2              ;EXIT

0C60  AF                :END:   XRA     A               ;INDICATE GOOD
```

```
                ;SET "RTRY" AND REGESTER B

0C61  3205EF     :2:    STO     A,RTRY          ;SET RETRY TO 1 FOR ERROR AND 0 FOR GOOD

0C64  ED4BCAEF  $       LD      BC,NUMSEC       ;BC = RESTORE BC

0C68  C9                RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright ©1982 Osborne Computer Corporation

```
0C69                    SENDEN:
                        ;DETERMINE THE DENSITY AND NUMBER OF SECTORS PER TRACK OF THIS DISK DRIVE
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;B       =       NUMBER OF SECTORS ON ONE TRACK
                        ;ZBIT    =       RESET IF ERROR
                                ;SAVTYP IS SET WITH DENSITY AND SECTOR SIZE

0C69                            PROC

                        *CHECK DENSITY

                                ;HOME LOOP

0C69  3E02                      LDK     A,2
0C6B  3205EF                    STO     A,RTRY

                                ;DENSITY LOOP(CHECK PRESENT DENSITY FIRST)

0C6E  0602              :RL1:   LDK     B,2             ;CHECK BOTH DENSITYS

0C70  C5                :RL2:   PUSH    BC              ;SAVE COUNT

                                ;CHECK THIS DENSTIY

0C71  CD710E                    CALL    SELDRV          ;SELECT DRIVE
                        ;       JRC     ?               ;NO ERROR CHECKING FOR SELDRV BECAUSE NO ERRORS ARE RETURNED AT THIS TIME

0C74  CDA50D                    CALL    RADR            ;READ ADDRESS
0C77  C1                        POP     BC              ;RESTORE DENSITY RETRY
0C78  3017 ^0C91$               JRNC    :1              ;IF GOOD

                        *IF DENSITY ERROR CHANGE DENSITY AND LOOP TO :RL2:

0C7A  3AD0EF                    LD      A,SAVTYP        ;PRESENT DENSITY
0C7D  EE01                      XRI     1               ;CHANGE DENSITY BIT
0C7F  32D0EF                    STO     A,SAVTYP        ;NEW DENSITY

0C82  10EC ^0C70$               DJNZ    :RL2            ;DENSITY LOOP

                        *IF BOTH DENSITYS FAIL HOME DRIVE AND TRY AGAIN

0C84  2105EF                    LDK     HL,RTRY
0C87  35                        DEC     [HL]
0C88  2843 ^0CCD$               JRZ     :ERET           ;END IF SECOND TIME THROUGH

0C8A  CDD80C                    CALL    HOME            ;HOME DRIVE
0C8D  383E ^0CCD$               JRC     :ERET           ;END IF ERROR IN HOME

0C8F  18DD ^0C6E$               JR      :RL1            ;TRY AGAIN

                        *SET "SAVTYP"

0C91  3A0CEF            :1:     LD      A,DSTSB+3       ;SECTOR LENTH STATUS BYTE
0C94  E603                      ANI     0000_0011B      ;0-3
0C96  CB27         $            SLA     A
0C98  CB27         $            SLA     A               ;NOW IS 0000_XX00B WAS 0000_00XXB
```

Copyright © 1982 Osborne Computer Corporation

```
0C9A  47                     MOV    B,A              ;SAVE

0C9B  3AD0EF                 LD     A,SAVTYP
0C9E  E6F3                   ANI    1111_0011B       ;CLEAR BITS
0CA0  B0                     ORA    B                ;OR IN SECTOR LENTH
0CA1  32D0EF                 STO    A,SAVTYP

                 *READ ADDRESS AND SET NUMBER OF SECTORS AND RETURN

0CA4  3A08EF                 LD     A,DSTS3+2        ;SECTOR JUST READ
0CA7  57                     MOV    D,A

                             ;RETRY LOOP

0CA8  3E03                   LDK    A,3
0CAA  3205EF      :RL3:      STO    A,RTRY           ;SET RETRYS

                             ;READ HEADER LOOP

0CAD  D5          :LOOP:     PUSH   DE               ;SAVE LAST SECTOR ADDR
0CAE  CDA500                 CALL   RADR             ;READ ADDRESS
0CB1  D1                     POP    DE
0CB2  3811 ^0CC5$            JRC    :ERR             ;IF ERROR IN RADR

                             ;CHECK FOR LAST HEADER

0CB4  3A08EF                 LD     A,DSTS3+2
0CB7  47                     MOV    B,A              ;B=PRESENT SECTOR
0CB8  7A                     MOV    A,D              ;A=LAST SECTOR
0CB9  90                     SUB    B                ;LAST SECTOR - PRESENT SECTOR
0CBA  28F1 ^0CAD$            JRZ    :LOOP            ;IF THE LAST SECTOR = THE PRESENT SECTOR(THIS SHOULD HAPPEN ONLY ON ERROR RETRY)

0CBC  3003 ^0CC1$            JRNC   :2               ;IF THE VALUE IN A WAS THE LAST SECTOR OF THE TRACK

0CBE  50                     MOV    D,B              ;D=LAST SECTOR READ
0CBF  18EC ^0CAD$            JR     :LOOP            ;LOOP TILL YOU FIND THE LAST TRACK

                             ;SET NUMBER OF SECTORS PER TRACK

0CC1  3C          :2:        INC    A                ;A=NUMBER OF SECTORS PER TRACK
0CC2  47                     MOV    B,A
0CC3  AF                     XRA    A                ;RESET FLAGS
0CC4  C9                     RET                     ;GOOD RETURN

                 *IF NUMB. SEC. ERROR MAKE LAST SECTOR READ ZERO AND RETRY TO :LOOP1

0CC5  1600        :ERR:      LDK    D,0
0CC7  3A05EF                 LD     A,RTRY
0CCA  3D                     DEC    A
0CCB  20DD ^0CAA$            JRNZ   :RL3             ;RETRY

0CCD  3E01        :ERET:     LDK    A,1
0CCF  B7                     ORA    A                ;FLAGS TO NONZERO
0CD0  C9                     RET                     ;ERROR RETURN
```

Copyright © 1982 Osborne Computer Corporation

```
0CD1                  SCTRKR:
                      ;SET CONTROLER TRACK REGESTER
                      ;D.TRKR <= SAVTRK
                      ;USED IN FORMATING WHEN YOU DON'T KNOW WHERE THE HEAD IS
                      ;ENTRY
                      ;SAVTRK =        TRACK

                      ;EXIT
                      ;D.TRKR =        SAVTRK

0CD1                          PROC

0CD1  3A15EF                  LD      A,SAVTRK
0CD4  320121                  STD     A,D.TRKR
0CD7  C9                      RET
```

Copyright © 1982 Osborne Computer Corporation

```
0CD8                HOME:
                    ;HOME DISK DRIVE
                    ;DRIVE IS ALREADY SELECTED AND READY
                    ;If "SEKDEL" has the verify bit set this proc will check for seek and crc errors
                    ;ENTRY
                    ;SDISK  =       DRIVE

                    ;EXIT
                    ;CBIT   =       SET IF ERROR

0CD8                        PROC

0CD8  3A13EF                LD      A,SEKDEL        ;GET SEEK DELAY
0CDB  E607                  ANI     0000_0111B      ;SPEAD & VERIFY BITS ONLY

0CDD  CD480E                CALL    FDSK            ;FUNCTION DISK
0CE0  D8                    RC                      ;IF ERROR

0CE1  CD880E                CALL    WBUSY           ;WAIT FOR BUSY TO DROP
0CE4  D8                    RC

0CE5  3A0021                LD      A,D.STSR
0CE8  CB57        $         BIT     2,A
0CEA  280C ^0CF8$           JRZ     :1              ;IF NOT ON TRACK ZERO

0CEC  3A13EF                LD      A,SEKDEL
0CEF  E604                  ANI     0000_0100B      ;VERIFY?
0CF1  C8                    RZ                      ;NO VERIFY GOOD RETURN

0CF2  3A0021                LD      A,D.STSR
0CF5  E618                  ANI     0001_1000B      ;TEST SEEK AND CRC
0CF7  C8                    RZ                      ;GOOD RETURN

0CF8  37            :1:     STC                     ;IF ERROR
0CF9  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
0CFA                   SEEK:
                       ;SEEK TO TRACK DEFINED BY SAVTRK
                       ;TRACK REG UPDATED AND VERIFIED
                       ;ENTRY
                       ;SAVTRK SET TO DESIRED TRACK

                       ;EXIT
                       ;CBIT   =       SET IF ERROR
                       ;        IF NO ERROR CONTROLER TRACK = SAVTRK

0CFA                            PROC

0CFA  210121                    LDK     HL,D.TRKR
0CFD  3A15EF                    LD      A,SAVTRK
0D00  BE                        CMP     [HL]
0D01  C8                        RZ                      ;RETURN

0D02  320321                    STO     A,D.DATR        ;SET TRACK WANTED

0D05  0610                      LDK     B,D.SEK
0D07  180C ^0D15$               JR      PSEKC           ;PERFORM SEEK COMMAND
```

Copyright © 1982 Osborne Computer Corporation

```
0D09                    STEP:
                        ;STEP ONE TRACK
                        ;SAVTRK IS NOT USED IN THIS PROC
                        ;CONTROLER TRK RES IS UPDATED
                        ;VERIFY IS PERFORMED
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;CBIT    =       SET IF ERROR
                        ;        IF NO ERROR CONTROLER TRACK = TRACK +/- 1

0D09                            PROC

0D09  0620                      LDK     B,D.STP         ;PERFORM STEP COMMAND
0D0B  1808 ^0D15$               JR      PSEKC
```

Copyright © 1982 Osborne Computer Corporation

```
0D0D                    STEPIN:
                        ;STEP IN ONE TRACK
                        ;SAVTRK IS NOT USED IN THIS PROC
                        ;CONTROLER URK REG IS UPDATED
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;CBIT    =       SET IF ERROR
                        ;        IF NO ERROR CONTROLER TRACK = TRACK + 1

0D0D                            PROC

0D0D  0640                      LDK     B,D.STPI
0D0F  1804 ^0D15$               JR      PSEKC           ;PERFORM STEP-IN COMMAND
```

Copyright © 1982 Osborne Computer Corporation

```
0D11                    STEPOUT:
                        ;STEP OUT ONE TRACK
                        ;SAVTRK IS NOT USED IN THIS PROC
                        ;CONTROLER TRK REG IS UPDATED
                        ;VERIFY IS PERFORMED
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;CBIT    =       SET IF ERROR
                        ;        IF NO ERROR CONTROLER TRACK = TRACK - 1

0D11                            PROC

0D11  0660                      LDK     B,D.STPO                ;PERFORM STEP-OUT COMMAND
0D13  1800 ^0D15$               JR      PSEKC
```

```
0D15                    PSEKC:
                        ;OR IN SEKDEL AND PERFORM SEEK TYPE COMMAND
                        ;ENTRY
                        ;B       =       SEEK TYPE COMMAND

                        ;EXIT
                        ;CBIT    =       SET IF ERROR

0D15                            PROC

0D15  3A13EF                    LD      A,SEKDEL
0D18  E617                      ANI     0001_0111B      ;ONLY UPDATE,VERIFY, & SPEAD
0D1A  B0                        ORA     B               ;OR IN COMMAND

0D1B  CD480E                    CALL    FDSK            ;FUNCTION DISK
0D1E  D8                        RC                      ;IF ERROR

0D1F  CDB80E                    CALL    WBUSY           ;WAIT FOR BUSY TO DROP

                        ;CHECK FOR ERRORS

0D22  3A13EF                    LD      A,SEKDEL
0D25  E604                      ANI     0000_0100B      ;VERIFY?
0D27  C8                        RZ                      ;NO VERIFY GOOD RETURN

0D28  3A0021                    LD      A,D.STSR
0D2B  E618                      ANI     0001_1000B      ;TEST SEEK AND CRC
0D2D  C8                        RZ                      ;GOOD RETURN

0D2E  37                        STC                     ;IF ERROR
0D2F  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0D30                    READ:
                        ;ENTRY
                        ;B          =       NUMB OF SECTORS TO READ

                        ;EXIT
                        ;HL         =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                        ;CBIT       =       SET IF ERROR

0D30                                PROC

0D30  3E80                          LDK     A,D.RDS
0D32  32D1EF                        STO     A,R_WCOM

0D35  1805 ^0D3C$                   JR      RD_WRT          ;JMP AND RETURN TO CALLING PROC
```

Copyright © 1982 Osborne Computer Corporation

```
0D37                    WRITE:
                        ;ENTRY
                        ;B       =       NUMB OF SECTORS TO WRITE

                        ;EXIT
                        ;HL      =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                        ;CBIT    =       SET IF ERROR

0D37                             PROC

0D37  3EA0                       LDK     A,D.WRTS
0D39  32D1EF                     STO     A,R_WCOM

                        ;FALLS THROUGH TO RD_WRT
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
0D3C                RD_WRT:
                    ;READ OR WRITE A SECTOR

                    ;ENTRY
                    ;B       =       NUMB OF SECTORS TO READ OR WRITE
                            ;R_WCOM =       D.RDS OR D.WRTS

                    ;EXIT
                    ;HL      =       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
                    ;CBIT    =       SET IF ERROR

0D3C                        PROC

                    ;SET SECTOR REG

0D3C  3A14EF                LD      A,SAVSEC
0D3F  320221                STO     A,D.SECR

0D42  C5                    PUSH    BC              ;SAVE NUMBER OF SECTORS TO R/W

                    ;SET DE TO NUMBER OF BYTES IN ONE SECTOR

0D43  218000                LDK     HL,128
0D46  3AD0EF                LD      A,SAVTYP        ;DISK TYPE
0D49  CB3F        $         SRL     A               ;DUMP TWO BITS
0D4B  CB3F        $         SRL     A
0D4D  E603                  ANI     0000_0011B      ;SIZE ONLY
0D4F  B7                    ORA     A               ;SET FLAGS
0D50  2804 ^0D56$           JRZ     :1              ;IF 128

0D52  47                    MOV     B,A
0D53  29            :BLOOP: ADD     HL,HL           ;SHIFT LEFT ONE BIT
0D54  10FD ^0D53$           DJNZ    :BLOOP

0D56  EB            :1:     EX      DE,HL           ;DE=NUMBER OF BYTES IN ONE SECTOR

0D57  C1                    POP     BC              ;RESTORE
0D58  C5                    PUSH    BC              ;SAVE NUMBER OF SECTORS TO R/W

                    ;GET COMMAND AND CHECK FOR MULTI-SECTOR

0D59  78                    MOV     A,B             ;GET NUMBER OF SECTORS
0D5A  0E00                  LDK     C,0             ;MAKE NONMULTI-SECTOR
0D5C  FE02                  CMP     2
0D5E  3802 ^0D62$           JRC     :2              ;IF LESS THAN TWO SECTORS

0D60  0E10                  LDK     C,10H           ;MAKE MULTI-SECTOR

0D62  3AD1EF        :2:     LD      A,R_WCOM        ;GET D.RDS OR D.WRTS
0D65  B1                    OR      C               ;MAKE MULTI-SECTOR OR NONMULTI-SECTOR

                    ;SET HL TO NUMBER OF BYTES TO TRANSFER

0D66  210000                LDK     HL,0
0D69  19            :LOOP1: ADD     HL,DE
0D6A  10FD ^0D69$           DJNZ    :LOOP1

0D6C  E5                    PUSH    HL              ;SAVE LENTH
```

```
                        ;GIVE COMMAND

0D6D  F3                        DI
0D6E  CD480E                    CALL    FDSK
0D71  3004 ^0D77$               JRNC    :3              ;IF GOOD

                        ;IF ABORT BEFORE DMA

0D73  D1                        POP     DE              ;RESTORE STACK
0D74  D1                        POP     DE
0D75  FB                        EI
0D76  C9                        RET                     ;RETURN IF ERROR IN FDSK

                        ;SET RETURN FROM DMA, DMA ADDR AND NUMBER OF BYTES TO TRANSFER

0D77  C1                :3:     POP     BC              ;RESTORE LENTH
0D78  2A0FEF                    LD      HL,DMADR        ;HL = DMA ADDRESS

0D7B  118A0D                    LDK     DE,:RET
0D7E  D5                        PUSH    DE              ;FOR RETURN

                        ;DO DMA

0D7F  3AD1EF                    LD      A,R_WCOM        ;(13) GET COMMAND
0D82  FE80                      CMP     D.RDS           ;(7)

0D84  CAD0DE                    JZ      DMARD           ;(10) READ DMA RETURNS TO :RET

0D87  C3F00E                    JMP     DMAWRT          ;(10) WRITE DMA RETURNS TO :RET

                        ;RETURN FROM DMA AND CHECK FOR BUSY AND RESET

0D8A  C1                :RET:   POP     BC              ;RESTORE NUMBER OF SECTORS

0D8B  1A                        LD      A,[DE]          ;GET STATUS
0D8C  CB47        $             BIT     0,A
0D8E  280E ^0D9E$               JRZ     :4              ;IF NOT BUSY

0D90  119B0D                    LDK     DE,:RET1
0D93  D5                        PUSH    DE              ;FOR RETURN

0D94  05                        DEC     B               ;SUBTRACK ONE FROM THE NUMBER OF SECTORS AND SET THE ZERO FLAG

0D95  CA380E                    JZ      WBUSY           ;IF NON MULTI-SECTOR R/W WAIT FOR BUSY TO DROP

0D98  C3360E                    JMP     FDRINT          ;CLEAR BUSY

0D9B  3A0021            :RET1:  LD      A,D.STSR        ;RETURN AND GET STATUS

                        ;CHECK FOR ERRORS

0D9E  E65C              :4:     ANI     0101_1100B      ;TEST write protect, rnf, crc, and lost data
0DA0  2801 ^0DA3$               JRZ     :5              ;IF GOOD

0DA2  37                        STC                     ;IF ERROR RECORD CONTROLER REGESTERS

0DA3  FB                :5:     EI
0DA4  C9                        RET                     ;RETURN
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0DA5                    RADR:
                        ;Read Address info.
                        ;READS SIX BYTES INTO "DSTSB"

                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;A       =       OFFH IF TIME OUT ERROR
                        ;CBIT    =       SET IF ERROR
                        ;D.TRKR  =       HEAD POSITION
                        *NOTE*
                        ;       SETS TRACK REG IN CONTROLER IF GOOD

0DA5                            PROC

0DA5  3EC0                      LDK     A,D.RDA
0DA7  F3                        DI
0DA8  CD430E                    CALL    FDSK            ;function disk
0DAB  383C ^0DE9$               JRC     :1

                        ;WAIT FOR FIRST DRQ OR TIME OUT

                                ;SET REGESTERS FOR DMA TRANSFER

0DAD  010500                    LDK     BC,5            ;SIX BYTES TO READ
0DB0  2109EF                    LDK     HL,DSTSB        ;FBA FOR DMA

                                ;WAIT FOR 1/4 OF A TRACK(50MS) OR DRQ

0DB3  110311                    LDK     DE,4363
0DB6  3A0021            :LOOP:  LD      A,D.STSR        ;(13) GET STATUS
0DB9  1F                        RAR                     ;(4)
0DBA  1F                        RAR                     ;(4)
0DBB  DACB0D                    JC      :3              ;(10) GOT DRQ

0DBE  1B                        DEC     DE              ;(6)
0DBF  7A                        MOV     A,D             ;(4)
0DC0  B3                        ORA     E               ;(4)
0DC1  C2B60D                    JNZ     :LOOP           ;(10)

                                ;INDICATE TIME OUT ERROR

0DC4  CD360E                    CALL    FORINT          ;CLEAR BUSY
0DC7  3EFF                      LDK     A,0FFH          ;A=0FFH
0DC9  181D ^0DE8$               JR      :2              ;INDICATE A TIME OUT ERROR

                                ;TRANSFER FIRST BYTE AND CALL DMARD FOR LAST FIVE BYTES

0DCB  3A0321            :3:     LD      A,D.DATR        ;(13) GET BYTE
0DCE  77                        STO     A,[HL]          ;(7) STORE BYTE
0DCF  23                        INC     HL              ;(6)
                                                        ;BC = 5
0DD0  CDDA0E                    CALL    DMARD           ;(17) CALL DMARD

                        ;RETURN FROM DMARD AND WAIT FOR BUSY TO BE RESET

0DD3  CDB80E                    CALL    WBUSY
0DD6  3811 ^0DE9$               JRC     :1              ;IF TIME OUT ERROR
```

```
0DD8  3A0021              LD      A,D.STSR        ;GET STATUS

                 ;CHECK FOR ERRORS

0DDB  E61C                ANI     0001_1100B      ;TEST RNF, CRC AND DATA LOST
0DDD  2009 ^0DE8$         JRNZ    :2              ;IF ERROR

                 ;SET TRACK REGESTER

0DDF  3A0221              LD      A,D.SECR        ;GET TRACK
0DE2  320121              STD     A,D.TRKR        ;SET TRACK
0DE5  AF                  XRA     A               ;RESET CARRY
0DE6  1801 ^0DE9$         JR      :1

0DE8  37          :2:     STC                     ;SET CBIT

0DE9  FB          :1:     EI
0DEA  C9                  RET
```

Copyright © 1982 Osborne Computer Corporation

```
0DEB                    READTRK:
                        ;READ ONE TRACK FROM THE DRIVE
                        ;ENTRY
                        ;DMADR  =       FWA OF BUFFER

                        ;EXIT
                        ;CBIT   =       SET IF ERROR

0DEB                            PROC

0DEB  3EE0                      LDK     A,D.RDT
0DED  F3                        DI
0DEE  CD430E                    CALL    FDSK
0DF1  380C ^0DFF$               JRC     :1              ;IF ERROR

                        ;DO DMA

0DF3  01FFFF                    LDK     BC,0FFFFH       ;FOR ROM 1.2
0DF6  2A0FEF                    LD      HL,DMADR
0DF9  CD0A0E                    CALL    DMARD           ;IN ROM
0DFC  FB                        EI
0DFD  AF                        XRA     A
0DFE  C9                        RET

0DFF  FB                :1:     EI
0E00  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0E01                    FMTTRK:
                        ;FORMAT ONE TRACK
                        ;ENTRY
                                ;BC      =       LENTH
                                ;DMADR   =       FWA OF BUFFER

                        ;EXIT
                        ;CBIT    =       SET IF ERROR

0E01                            PROC

                        ;TEST DENSITY AND SET REG D TO 04EH OR 0FFH

0E01   3A00EF                   LD      A,SAVTYP
0E04   164E                     LDK     D,04EH          ;DOUBLE
0E06   0F                       RRC     A
0E07   3002 ^0E0B$              JRNC    :1              ;IF DOUBLE

0E09   16FF                     LDK     D,0FFH          ;SINGLE

                        ;GIVE COMMAND

0E0B   3EF0             :1:     LDK     A,D.WRTT
0E0D   F3                       DI
0E0E   D5                       PUSH    DE              ;FILL BYTE
0E0F   C5                       PUSH    BC              ;LENTH

0E10   CD480E                   CALL    FDSK
0E13   C1                       POP     BC              ;LENTH
0E14   D1                       POP     DE              ;FILL BYTE
0E15   381C ^0E33$              JRC     :3              ;IF ERROR

                        ;DO DMA

0E17   D5                       PUSH    DE              ;FILL BYTE

0E18   2A0FEF                   LD      HL,DMADR
0E1B   CDF00E                   CALL    DMAWRT

                        ;PAD REST OF TRACK

0E1E   C1                       POP     BC              ;B = FILL BYTE
0E1F   210321                   LDK     HL,02103H       ;DATA REGESTER

0E22   1A               :LOOP:  LD      A,[DE]          ;GET STATUS
0E23   1F                       RAR
0E24   3006 ^0E2C$              JRNC    :2              ;FINISHED IF NO BUSY
0E26   1F                       RAR
0E27   30F9 ^0E22$              JRNC    :LOOP           ;IF NO DRQ

0E29   70                       STO     B,[HL]          ;STORE BYTE
0E2A   18F6 ^0E22$              JR      :LOOP

                        ;CHECK FOR ERROR

0E2C   3A0021           :2:     LD      A,D.STSR        ;GET STATUS
0E2F   E644                     ANI     0100_0100B      ;TEST write protect, and data lost
0E31   2801 ^0E34$              JRZ     :4              ;IF GOOD
```

```
0E33  37          :3:     STC

0E34  FB          :4:     EI
0E35  C9                  RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
0E36                    FDRINT:
                        ;INTERRUPT DISK CONTROLLER
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;BUSY CLEARED.
0E36                            PROC

0E36  F5                        PUSH    AF
0E37  C5                        PUSH    BC

0E38  3ED0                      LDK     A,D.FINT
0E3A  320021                    STO     A,D.CMDR

                        ;WAIT FOR AT LEAST 28 MICROSECONDS

0E3D  B7                        ORA     A               ;(4)
0E3E  0607                      LDK     B,7             ;(7)
0E40  10FE ^0E40$ :WLOOP:       DJNZ    :WLOOP          ;(91) = (13*7) WAIT

                        ;CHECK FOR BUSY DROP

0E42  CDB80E                    CALL    WBUSY

0E45  C1                        POP     BC
0E46  F1                        POP     AF
0E47  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation


```
0E48               FDSK:
                   ;FUNCTION DISK ROUTINE
                   ;THIS IS THE ONLY ROUTINE THAT WRITES TO THE COMMAND REGESTER OF THE CONTROLER CHIP
                   ;THIS ROUTINE HAS A BUILT IN DELAY OF AT LEAST 28 MICRO SEC. BEFORE READING THE STATUS ON THE CHIP
                   ;ENTRY
                   ;A       =       FUNCTION CODE

                   ;EXIT
                   ;A       =       OFFH IS TIME OUT ERROR
                   ;CBIT    =       SET IF ERROR

0E48                        PROC

0E48  210021                LDK     HL,D.STSR       ;STATUS AND COMMAND REGESTER
0E4B  CB46       $          BIT     0,[HL]
0E4D  2803 ^0E52$           JRZ     :1              ;IF NOT BUSY

0E4F  CD360E                CALL    FORINT          ;RESET BUSY

0E52  77           :1:      STO     A,[HL]          ;FUNCTION DRIVE(WRITE COMMAND TO CONTROLER)

                   ;WAIT FOR 56 SINGLE AND 28 DOUBLE

0E53  3AD0EF                LD      A,SAVTYP        ;(13) DISK TYPE
0E56  0605                  LDK     B,5             ;(7)
0E58  0F                    RRC     A               ;(4)
0E59  D25F0E                JNC     :WLOOP          ;(10) IF DOUBLE DENSITY

0E5C  AF                    XRA     A               ;(4) RESET CARRY FLAG
0E5D  060D                  LDK     B,13            ;(7)

0E5F  10FE ^0E5F$ :WLOOP: DJNZ    :WLOOP          ;(13) WAIT

                   ;WAIT FOR BUSY TO BE SET

0E61  3EFF                  LDK     A,0FFH          ;(7)
0E63  47                    MOV     B,A             ;(4) 256 LOOPS

0E64  CB46       $ :LOOP:   BIT     0,[HL]          ;TEST BUSY BIT
0E66  2004 ^0E6C$           JRNZ    :3              ;IF CHIP WENT BUSY

0E68  10FA ^0E64$           DJNZ    :LOOP           ;IF NOT TIME-OUT

0E6A  37                    STC                     ;IF ERROR A=FF AND CBIT = SET
0E6B  C9                    RET

0E6C  326AEF       :3:      STO     A,DACTVE        ;SET DRIVE ACTIVE COUNTER
0E6F  AF                    XRA     A               ;RESET CARRY FLAG
0E70  C9                    RET
```

Copyright ©1982 Osborne Computer Corporation

```
0E71                    SELDRV:
                        ;SELECT DRIVE
                        ;ENTRY
                        ;SDISK  =       DRIVE TO SELECT

                        ;EXIT
                        ;ZBIT   =       SET IF PIABD WAS THE SAME AS SDISK
                        ;ZBIT   =       RESET IF PIABD WAS DIFFERENT THAN SDISK
                        ;CBIT   =       SET IF THERE ARE NO INDEX PULSES

0E71                            PROC

0E71  CDA60E                    CALL    SELDEN          ;SELECT DENSITY

0E74  3A17EF                    LD      A,SDISK
0E77  21C7EF                    LDK     HL,DSKSWP       ;DISK DRIVE SWAP CELL
0E7A  AE                        XOR     [HL]            ;SWAP A FOR B IF DSKSWP=1
0E7B  E601                      AND     1               ;CAN ONLY BE 0 OR 1
0E7D  FE01                      CMP     1
0E7F  2002 ^0E83$               JRNZ    :1              ;IF NOT DRIVE 1

0E81  3E40                      LDK     A,40H

0E83  C640              :1:     ADI     40H
0E85  4F                        MOV     C,A
0E86  3A62EF                    LD      A,PIABD
0E89  47                        MOV     B,A
0E8A  E6C0                      ANI     1100_0000B      ;GET DRIVE BITS ONLY
0E8C  B9                        CMP     C
0E8D  2810 ^0E9F$               JRZ     :2              ;IF DRIVE ALREADY SELECTED

                        ;SELECT DRIVE

0E8F  CDCF0E                    CALL    RDSKD           ;TURN DRIVE ON
0E92  3EFA                      LDK     A,250
0E94  CDCF00                    CALL    DELAY           ;WAIT FOR MOTOR SPIN UP
0E97  3E14                      LDK     A,20
0E99  CDCF00                    CALL    DELAY           ;2ND DELAY

0E9C  3E01                      LDK     A,1             ;INDICATED DRIVE WAS NOT SELECTED
0E9E  B7                        ORA     A               ;SET FLAGS

0E9F  216AEF            :2:     LDK     HL,DACTVE
0EA2  36FF                      STO     0FFH,[HL]
0EA4  FB                        EI
0EA5  C9                        RET
```

Copyright ©1982 Osborne Computer Corporation

```
0EA6                    SELDEN:
                        ;SELECT SINGLE OR DOUBLE DENSITY
                        ;ENTRY
                        ;SAVTYP =        BIT 0:
                        ;                1 = SINGLE, 0 = DOUBLE

                        ;EXIT
                        ;NONE
                        ;NOTE    Bit 0 of "PIAAD" :
                        ;        set     =       single density
                        ;        reset   =       double density

0EA6                            PROC

0EA6  3A61EF                    LD      A,PIAAD         ;PRESENT VALUE OF PIA REG
0EA9  E6FE                      ANI     1111_1110B      ;CLEAR BIT 0
0EAB  4F                        MOV     C,A

                        ;SET DENSITY BIT

0EAC  3AD0EF                    LD      A,SAVTYP        ;GET DISK TYPE INFO
0EAF  0F                        RRC                     ;CBIT <= BIT 0
0EB0  3002 ^0EB4$               JRNC    :1              ;IF "SAVTYP" BIT0 IS 0

0EB2  CBC1          $           SBIT    0,C             ;SET BIT 0 OF REG C

0EB4  CD7900        :1:         CALL    OPAD            ;FUNCTION PIA
0EB7  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0EB8                    WBUSY:
                        ;WAIT FOR BUSY TO CLEAR
                        ;This routine must wait for 2 seconds
                        ;2 seconds is the time it takes for the chip to seek 39 tracks and have five index holes go by.
                        ;ENTRY
                        ;NONE

                        ;EXIT
                        ;A      =       0FFH IF TIME OUT OCCURRED
                        ;CBIT   =       SET "  "  "   "

0EB8                            PROC

0EB8  010000                    LDK     BC,0

0EBB  3A0021            :LOOP:  LD      A,D.STSR        ;(13)
0EBE  CB47        $             BIT     0,A             ;(8) DS.BSY
0EC0  280C ^0ECE$               JRZ     :1              ;(7) GOOD RETURN

0EC2  E3                        EX      [SP],HL         ;(23) DELAY
0EC3  E3                        EX      [SP],HL         ;(23) DELAY
0EC4  E3                        EX      [SP],HL         ;(23) DELAY
0EC5  E3                        EX      [SP],HL         ;(23) DELAY
0EC6  0B                        DEC     BC              ;(6)
0EC7  78                        MOV     A,B             ;(4)
0EC8  B1                        OR      C               ;(4)
0EC9  20F0 ^0EBB$               JRNZ    :LOOP           ;(12) IF NOT TIME-OUT

0ECB  3EFF                      LDK     A,0FFH          ;TIME OUT ERROR
                        ;       CALL    FORINT          ;RESET BUSY
0ECD  37                        STC                     ;SET ERROR

0ECE  C9                :1:     RET
```

Copyright ©1982 Osborne Computer Corporation

```
0ECF                    RDSKD:
                        ;SELECT DRIVE BY SETING THE "PIA" WITH THE VALUE SPECIFIED BY C
                        ;ENTRY
                        ;C        =       DRIVE

                        ;EXIT
                        ;NONE
0ECF                            PROC

0ECF  3A62EF                    LD      A,PIABD
0ED2  E63F                      ANI     0011_1111B      ;GET VIO OFFSET AND BELL
0ED4  B1                        OR      C
0ED5  4F                        MOV     C,A
0ED6  CD8600                    CALL    OPBD            ;FUNCTION PIO-3
0ED9  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
0EDA                 DMARD:
                     ;TRANSFER DATA FROM CONTROLER TO MEMORY
                     ;ENTRY
                     ;BC      =       BYTES TO TRANSFER
                     ;HL      =       FWA OF BUFFER

                     ;EXIT
                     ;HL      =       NEXT ADDRESS
                     ;DE      =       D.STSR

0EDA                          PROC

0EDA  110021                  LDK     DE,D.STSR       ;(10)

0EDD  1A             :LOOP:   LD      A,[DE]          ;(7) GET STATUS
0EDE  1F                      RAR                     ;(4)
0EDF  D0                      RNC                     ;(5) RETURN IF NO BUSY
0EE0  1F                      RAR                     ;(4)
0EE1  D2DD0E                  JNC     :LOOP           ;(10) IF NO DRQ

0EE4  3A0321                  LD      A,D.DATR        ;(13) GET BYTE
0EE7  77                      STO     A,[HL]          ;(7) STORE BYTE
0EE8  23                      INC     HL              ;(6)
0EE9  0B                      DEC     BC              ;(6)
0EEA  78                      MOV     A,B             ;(4)
0EEB  B1                      ORA     C               ;(4)
0EEC  C2DD0E                  JNZ     :LOOP           ;(10)

0EEF  C9                      RET
```

Copyright © 1982 Osborne Computer Corporation

```
0EF0                    DMAWRT:
                        ;Xfer data from memory to disk
                        ;ENTRY
                        ;BC      =       BYTES TO TRANSFER
                        ;HL      =       FWA OF BUFFER

                        ;EXIT
                        ;HL      =       NEXT ADDRESS

0EF0                            PROC

0EF0  110021                    LDK     DE,D.STSR

0EF3  1A                :LOOP:  LD      A,[DE]          ;GET STATUS
0EF4  1F                        RAR
0EF5  D0                        RNC                     ;RETURN IF NO BUSY
0EF6  1F                        RAR
0EF7  D2F30E                    JNC     :LOOP           ;IF NO DRQ

0EFA  7E                        LD      A,[HL]          ;GET BYTE
0EFB  320321                    STO     A,D.DATR        ;STORE BYTE
0EFE  23                        INC     HL
0EFF  0B                        DEC     BC
0F00  78                        MOV     A,B
0F01  B1                        ORA     C
0F02  C2F30E                    JNZ     :LOOP

0F05  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation


```
0F06                    DDRV:
                        ;Deselect drive
                        ;Entry
                        ;SDISK  =       current disk drive
0F06                            PROC

0F06  3A62EF                    LD      A,PIABD
0F09  E61F                      AND     1_1111b
0F0B  4F                        MOV     C,A
0F0C  C38600                    JMP     OPBD            ;deselect last drive
```

Copyright ©1982 Osborne Computer Corporation

```
                        *[7]

0F0F                    FORMAT:
                        ;       This proc will format the next track in IBM 3740 format consisting of 40 tracks, with each
                        ;track containing 10 sectors.

                        ;Entry
                        ;BC      =       FWA of buffer
                                ;BUF+0   =       DW length
                                ;BUF+2   =       beginning of data
                                ;SAVTRK  =       THE TRACK TO BE FORMATED

                        ;EXIT
                        ;NONE

0F0F                            PROC

0F0F  ED430FEF  $               STO     BC,DMADR        ;SAVE BC

                        ;SELECT DRIVE

0F13  CD710E                    CALL    SELDRV
0F16  3831 ^0F49$               JRC     :ERROR

                        ;TEST FOR STEP OR NO-STEP

0F18  2115EF                    LDK     HL,SAVTRK
0F1B  3A0121                    LD      A,D.TRKR        ;TRACK REG
0F1E  BE                        CMP     [HL]
0F1F  2817 ^0F38$               JRZ     :1              ;IF SAVTRK AND TRACK REG ARE THE SAME SKIP THE STEP

                        ;STEP IN ONE TRACK

0F21  3A13EF                    LD      A,SEKDEL
0F24  F610                      ORI     0001_0000B      ;UPDATE
0F26  3213EF                    STO     A,SEKDEL        ;SET UP SEKDEL

0F29  CD000D                    CALL    STEPIN

0F2C  F5                        PUSH    AF              ;SAVE FLAGS
0F2D  3A13EF                    LD      A,SEKDEL
0F30  E603                      ANI     0000_0011B      ;ONLY SPEAD LEFT
0F32  3213EF                    STO     A,SEKDEL        ;RESET SEKDEL
0F35  F1                        POP     AF              ;RESTORE

0F36  3811 ^0F49$               JRC     :ERROR

                        ;SET "DMADR"

0F38  2A0FEF            :1:     LD      HL,DMADR        ;GET ADDRESS

0F3B  4E                        LD      C,[HL]
0F3C  23                        INC     HL
0F3D  46                        LD      B,[HL]          ;BC = LENTH OF FORMAT DATA

0F3E  23                        INC     HL
0F3F  220FEF                    STO     HL,DMADR        ;SET DMA

                        ;FORMAT TRACK
```

Copyright © 1982 Osborne Computer Corporation


```
0F42  CD010E                CALL    FMTTRK
0F45  3802 ^0F49$           JRC     :ERROR

0F47  AF                    XRA     A
0F48  C9                    RET

0F49  3EFF         :ERROR:  LDK     A,0FFH
0F4B  B7                    ORA     A
0F4C  C9                    RET


0F4D               HCBOOT:
                   ;IF "ESC" IS PRESSED IN RESPONCE TO THE FIRST PROMPT THIS IS THE PLACE TO PUT A COLD BOOT LOADER OTHER
                   ;THAN THE FLOPPY ONE SUPPLIED.
                   ;ENTRY
                   ;NONE

                   ;EXIT
                   ;NONE

0F4D                        PROC

0F4D  C30000                JMP     START


      = 0F50       RLWA     =       *               ;LWA OF ROM RESIDENT CODE

                            MSG     'LENGTH OF THIS ROM IS = ',RLWA-1
      'LENGTH OF THIS ROM IS = 0F4F'
      = 0000                IF      RLWA > 0FFBH
   -               .9       ERROR   CODE TOO LARGE..
                            ENDIF

                            ECHO    0FFFH-RLWA  !   DB   0FFH
                            ENDM

                   ;        END


no ERRORs,  472 Labels, 6770h bytes not used.     Program LWA = 0FFFh.
```

```
ACIAD   EFC1      3# 6   115/15
ACISTA  0BE8     17/ 5   116/12   117/14   119# 2
ALPHKY  0003      2# 9    77/55
BCPM    02A1     20/ 8    22# 2
BELCNT  EF6B      2#51    45/10    67/30
BIOJP   0098     12/ 3    12# 8
BKS     0008      1#55    33/ 9
BRTBIT  0080      2#12    38/13    53/17    60/16
CALC    0504     53/ 7    54/ 7    55# 2
CBELL   0007      1#19    33/12
CBOOT   0255      7/36     7/40    16/ 8    19# 3
CCPADR  EFD2      3#13    12/ 8    20/14    21/17
CCRA    2901      2#25    92/13    92/40    93/15
CCRB    2903      2#27    92/22    92/53    95/22    95/24    95/28    95/43
                 95/50    95/54    96/23    96/25    96/29    99/24    99/26
                 99/30
CHKEY   07E2     70/27    74# 2
CI      0373      7/30    16/11    29# 2
CLRLN   0651     46/ 5    51/ 6    58/ 3    60# 2    61/48
COUT    03E0     14/29    14/32    16/12    34# 2
COUT2   041A     32/25    36# 2
CPDRA   2900      2#24     2/25     2/26     2/27     2/28     2/30     2/31
                  2/33    92/17    92/21    93/11    94/15    96/34    97/29
                 98/ 3    99/33    99/55   101/14
CPDRB   2902      2#26    92/26    92/30    93/12    94/13    95/15    96/15
                 97/16    99/16   101/16   101/22
CR      000D      1#52     7/35    13/ 2    18/10    18/20    18/28    18/40
                 18/48    33/ 7    88/ 7
CTLKY   0002      2# 8    77/49
CTLTB   0924     84/ 6    88#46
CURS    EF5A      2#55    34/40    49/40
CV2IP   0B3D    110# 2   112/ 7
CV2OP   0B0E    109# 2   111/ 7
D.CMDR  2100      2#18     2/19     2/20     2/21     2/22   143/15
D.DATR  2103      2#22   129/19   138/51   149/22   150/22
D.FINT  00D0      5#11   143/14
D.RDA   00C0      5# 8   138/18
D.RDS   0080      5# 6   121/17   134/12   137/25
D.RDT   00E0      5# 9   140/12
D.SECR  2102      2#21   136/18   139/11
D.SEK   0010      5# 2   129/21
D.STP   0020      5# 3   130/16
D.STPI  0040      5# 4   131/15
D.STPO  0060      5# 5   132/16
D.STSR  2100      2#19   128/23   128/31   133/27   137/48   138/33   139/ 2
                141/57   144/15   147/17   149/14   150/13
D.TRKR  2101      2#20   127/15   129/14   139/12   152/28
D.WRTS  00A0      5# 7   122/17   135/12
D.WRTT  00F0      5#10   141/24
DACTVE  EF6A      2#60   144/48   145/44
DBCT    0001      2# 4    73/24
DDRV    0F06     67/53   151# 2
DELAY   00CF     15# 2   145/37   145/39
DIMBIT  0000      2#13    40/43
DISDIM  mac       5#21     6/13    38/15    40/45    53/19    60/18    63/17
                 64/28    65/28
DMADR   EF0F      2#46    19/26    22/21    24/57    26/ 7   137/17   140/20
                141/33   152/18   152/50   152/57
DMARD   0EDA     16/55   137/27   138/55   140/21   149# 2
DMAWRT  0EF0     16/54   137/29   141/39   150# 2
```

Copyright ©1982 Osborne Computer Corporation

Copyright ©1982 Osborne Computer Corporation


```
DNARW   008C    77# 5   82/19
DOHOME  083E    82/23   87# 2
DOLF    0684    47/10   61# 2
DOLF2   0687    49/24   61#16
DSKSWP  EFC7     3# 8    7/34  145/17
DSTSB   EFD9     2#45  125/57  126/10  126/27  138/28
EBOOT   00AD    13# 6   19/20   19/36   22/38   24/52   26/28
EDELC   05A2    32/19   53# 2
EEOL    058A    32/22   51# 2
EFADR   0010    30# 6   52/ 7   52/17
EFESC   0008    30# 7   34/43   36/ 9   52/ 7   52/17
EFGR    0001    30#10   30/11   56/ 7   56/20
EFHA    0002    30# 9   30/11   56/10   56/23
EFMSK   0007    30#11   35/ 6   42/41   52/ 5   52/15
EFSCR   0020    30# 5   52/17
EFUN    0004    30# 8   30/11   56/13   56/26
EINSRT  053B    32/13   54# 2
EMBOOT  00A0    13# 2   13/13
ENADIM  mac      5#16   38/13   40/43   53/16   60/16   63/15   64/26
                65/26
ERC     007F     1#54   88/ 5   88/ 5   88/ 7
ESC     001B     1#53    7/31   18/ 4   18/ 8   18/12   18/12   18/14
                18/16   18/18   18/18   18/22   18/26   18/30   18/32
                18/33   18/35   18/36   18/38   18/42   18/46   18/51
                18/53   34/49   88/ 5
ESCCAD  0591    32/ 3   52# 2
ESCCGR  05F4    32/11   56#20
ESCCHA  05F8    32/13   56#23
ESCCUN  05FC    32/15   56#26
ESCEE   0601    32/20   57# 5
ESCH    EF60     2#57   34/41   36/10   37/19   42/44   52/ 9   56/17
ESCHTB  0384    31# 5   35/ 5
ESCLCK  0580    32/23   50# 2
ESCRR   0601    32/21   57# 2
ESCSAD  0598    32/ 9   52#12
ESCSGR  05E4    32/10   56# 2
ESCSHA  05E8    32/12   56#10
ESCSUN  05EC    32/14   56#13
ESCULK  0583    32/24   50# 9
ESCZZ   051F    32/17   56#31
FDSK    0E48   128/17  133/16  137/ 4  138/20  140/14  141/29  144# 2
FMTTRK  0ED1    17/ 3  141# 2  152/61
FORINT  0E36     6/ 8   16/61  137/45  138/45  143# 2  144/19
FORMAT  0F0F    16/32  152# 3
FWAVM   F000     3#22   46/ 4   46/16   61/21
GKEY    06FC     6/27   67# 3
GTMASK  07F6    74/17   74/21   75# 2
H.KEY   2200     2#23   76/12
H.SCTR  2A00     2#34  115/12  115/16
H.SREC  2A01     2#37  116/19
H.SSTS  2A00     2#35  119/19
H.SXMT  2A01     2#36  118/15
H.VIO   2C00     2#38    9/15   10/28   10/32   11/18   11/22   68/10
               119/14
HDBOOT  0F4D     7/32  153#13
HMSCRN  005B    77# 6   82/22
HOME    0C08    15/56  120/19  125/50  128# 2
HSTACT  EF50     2#51   27/ 4   27/ 8   27/ 9
IE.CO   0936     7/ 3   16/37   92# 2
IE.STS  098C    15/39   94# 2
```

Copyright © 1982 Osborne Computer Corporation

```
IE.IDM  0A3D     16/43    99# 2   105/15
IE.ODM  09FB     16/42    97# 2   107/21
IE.OIM  09D8     16/41    96# 2   105/ 8   105/27   107/13   107/30
IE.PP   0A99     16/44   101# 2
IE.SHK  0AD7     96/18    96/37    97/19    97#25
IE.SI   0979     16/38    93# 2
IE.TC   0998     16/40    95# 2
IEADRS  EFD8      3#16   105/ 5   107/10
IECHAR  EFDC      3#17   104/15   105/21   105/33   106/10
IEI10   0ABA    104/17   105# 2   105/10
IEI20   0AC6    105#12
IEI30   0ACE    105/17   105#21
IEI40   0AD1    105#23   105/29
IEINP   0AE1     17/10   106# 2   106/ 8
IEINST  0A31     17/ 8   104# 2   106/ 7
IELSTN  0020      4#31   107/11
IE005   0AEE    107#10   107/15
IE022   0AFD    107#20   107/24
IE040   0B05    107#26   107/32
IEOSTA  0AAE     17/ 9   103# 2
IEOUT   0AED     17/11   107# 2
IESTK   EF6F      3# 4    67/14    68/23
IETALK  0040      4#29   105/ 6
IEULST  003F      4#32   107/29
IEUTLK  005F      4#30   105/26
IMSG    019F      7/25    18# 2
INTBL   EFF0      3#21     6/22     6/28
IRPTCT  0018      2# 5    70/47
ISTK    EF99      3# 5    67/15
KBDRVR  074E     68/ 5    70# 2
KBSCAN  0786     70/12    72# 2
KBSERV  0809     70/53    77# 9
KCOLM   0007      2# 3    70/52    72/58
KEY1    0838     77/56    79# 2
KEY2    0844     77/53    80# 2
KEY3    0852     80/ 7    81# 2
KEY3A   0857     81#10    84/10
KEY4    085A     77/50    82# 2
KEY5    087A     82/26    82/29    83# 2
KEY6    0886     83/ 4    84# 2
KEYE    0834     77/30    78# 2    79/ 4    79/ 7    79/10    80/12    81/11
                 82/32    83/ 7    83/10    84/ 4
KEYLCK  EF59      2#54     6/50    28/ 9    50/11    68/ 3
KEYLST  EFD4      3#14    70/16    72/51
KLELEN  0002      1#59    70/58    73/ 3
KLLEN   0003      1#58    70/17    72/50
KLUSED  0007      1#60    70/24    72/55    73/22
KROWM   0038      2# 2    70/52    72/58
KYCDTB  0834     77/26    88# 4
KYSRVD  0006      1#61    70/44    70/45
LDTRK   EF5E      3# 3    27/17
LF      000A      1#51    13/ 2    18/ 2    18/ 2    18/ 2    18/ 2    18/10
                 18/20    18/23    18/40    18/48    18/48    13/48    33/ 8
LFTARW  008D     77# 2    82/10
LIST    0BD9     16/13    16/14   118# 2   118/12
LKEY    EF5E      2#56    28/13    29/15    29/18    70/19    78/ 5
LLIMIT  EF6C      3# 2     7/16    44/25    49/16    60/19    61/35
LOGSEC  EF56      2#53    27/ 8
LOOKUP  044C     37/24    39# 2    39/23
LVMEM   1000      1#47    46/ 6
```

Copyright ©1982 Osborne Computer Corporation

```
  MCRIGH 000C     1#21   33/10
  MCUP   000B     1#20   33/11
  NMIA   0066     1#46    8/ 4
  NRETRY 000A     2#14  120/14  123/17  123/48
  NUMSEC EFCA     3# 9  123/15  123/37  124/ 6
  OPAD   0079     7/13   10# 2   41/34   85/16   87/10  146/28
  OPBD   0086     7/ 8   11# 2   34/33   41/20   45/ 8   46/13   61/59
                 67/43   86/17  148/15  151/11
  OSTR   0030     7/26   14# 2   14/36
  PA.CDR 002A     4# 4  109/13  110/13
  PA.CDT 002E     4# 5  109/18  110/18
  PA.CTL 2901     2#30  109/14  109/19  110/14  110/19  112/12
  PA.DIR 2900     2#29  109/16  110/16
  PA.DRI 0000     4#13  110/15
  PA.DRD 00FF     4#12  109/15
  PA.DTA 2900     2#28    2/29  113/12  114/12
  PARINP 088E    17/15  113# 2  113/ 8
  PAROUT 089D    17/16  114# 2  114/ 8
  PB.CDR 0000     4# 7  109/21  110/21
  PB.CDT 0004     4# 8  109/26  110/26
  PB.CTL 2903     2#33  109/22  109/27  110/22  110/27
  PB.DIR 2902     2#32  109/24  110/24
  PB.DR  003F     4#14  109/23  110/23
  PB.DTA 2902     2#31    2/32  109/29  110/29  111/ 8  114/14  114/16
  PB.DTI 003B     4#16  110/28
  PB.DTO 0002     4#15  109/28  114/13  114/15
  PIAAD  EF61     2#58   10/31   41/30   85/12   87/ 6  146/16
  PIABD  EF62     2#59   11/21   34/26   41/16   42/25   43/ 9   44/32
                  45/ 5   46/10   57/23   61/26   61/49   67/40   85/ 7
                  87/12  145/27  148/11  151/ 8
  PIACTL EFDD     3#18  112/ 8  112/13  113/11
  PIS20  0BBB   112/10  112#18
  PISTAT 0878    17/13  112# 2  113/ 7
n POS10  0B77   111#14
  POSTAT 086C    17/14  111# 2  114/ 7
  PP.IN  0002     4#25  110/ 8  110/31
  PP.IRD 0080     4#19  112/ 9  112/15
  PP.MOD EFDE     3#19  109/ 7  109/32  110/ 7  110/32
  PP.ORD 0040     4#18  111/ 9
  PP.OUT 0001     4#24  109/ 8  109/31
  PSEKC  0D15   129/22  130/17  131/16  132/17  133# 2
  PSTESC 0425    34/44   37# 2
  RADR   0DA5    16/25  123/27  123/51  125/32  126/21  138# 2
  RDROW  0800    72/13   72/17   73/38   74/24   76# 2   77/35   77/40
  RDRV   0C02    16/19   19/16   22/34  120# 4
  RDSKD  0ECF   145/35  148# 2
  RDWRT  0D3C   123/38  134/15  136# 2
  READ   0D30    16/23  134# 2
  READER 033F    15/15  116# 2  116/14
  READTR 0DEB    17/ 2  140# 2
  RKEY   0373    29# 3   29/12
  RLWA   0F50   153#27  153/30  153/31  153/35
  ROMJP1 0093    12# 2   16/30   68/14
  ROMJP2 0098    12# 6   16/31
  ROMRAM EF08     2#44   68/12
  ROMSTK EFC1     3# 7    6/ 7
  ROWOM  0081     1#41   72/16
  RSEC   0C1C    16/26   19/33   24/49   26/25  121# 2
  RTARW  0088    77# 3   82/13
  RTRY   EF05     2#43  120/15  120/22  123/18  123/43  124/ 4  125/19
```

Copyright © 1982 Osborne Computer Corporation

```
                    125/46  126/16  126/49
RWCOM  EF01           3#12  121/18  122/18  134/13  135/13  136/50  137/24
RWSEC  0C28         121/20  123# 2
SAVSEC EF14           2#48   19/31   22/28   26/20  136/17
SAVTRK EF15           2#49   24/ 5   24/61   26/13  127/14  129/15  152/27
SAVTYP EF00           3#11   22/46  125/38  125/40  126/ 3  126/ 6  136/25
                    141/15  144/25  146/22
SCLFRE 0035           1#50   15/13
SCREEN 0474          41# 2   42/16
SCTRKR 0C01          17/ 6  127# 2
SDISK  EF17           2#50   22/24  145/16
SEEK   0CFA          16/57  123/32  129# 2
SEKDEL EF13           2#47    6/57  128/14  128/27  133/12  133/23  152/34
                    152/36  152/41  152/43
SELDEN 0EA6         145/14  146# 2
SELDRV 0E71          17/ 4  120/16  123/22  125/29  145# 2  152/22
SENDEN 0C69          16/29   19/17   22/35  125# 2
SEQ    EFCC           3#10   67/57   67/59
SERFLG EF0A           3#15  116/16  118/17  119/24  119/27
SETXY  049E          37/13   42# 2
SHFTKY 0004           2#10   77/52   82/25
SHFTTB 090C          81/ 7   88#34
SI.MRS 0057           1#42  115/11
SI.RRD 0001           1#44  116/13
SI.S16 0055           1#43    7/20
SI.TRD 0002           1#45  117/15
SIRST  0B32           7/21   16/36  115# 2
SKEY   0366          16/10   28# 3   29/11
SLD1   08AC          86/ 3   86# 7
SLD2   0837          86#15   87/16
SLDRCT 0033           2#11   85/23
SLIDED 08AA          82/20   86# 5
SLIDEL 0891          82/11   85# 4
SLIDER 0895          82/14   85# 9
SLIDEU 08A6          82/17   86# 2
SLR1   0897          85/ 6   85#12
SLR2   089F          85#20   86/18
SLST   0B00          16/28  117# 2  118/11
SPAD   0068           6/59    9# 2
SRPTCT 0006           2# 6   70/50
START  0000           6# 3    7/42    8/ 6  153/24
STEP   0009          16/58  130# 2
STEPIN 000D          16/59  131# 2  152/38
STEPOU 0011          16/60  132# 2
STODIM 0626          16/50   63# 2
STR3   0020           4#20  114/13
TAB    0009           1#56   88/ 5
TEM    EF00           2#42   22/25   23/13   24/15   24/34   26/33
TOTROW 0007           2# 7   72/20   72/28
UNASEC EF55           2#52   27/14
UNCUR  06F6          42/12   43/ 8   44/ 5   44/14   47/ 3   48/ 5   54/ 5
                     57/17   61/14   66# 2
UPARW  008A          77# 4   82/16
VALCTS 0008          33#18   38/21
VALETS 0010          32#30   37/22
VALIDC 0336          33# 2   33/18   38/20
VALIDE 0394          32# 2   32/30   37/21
VBRIGH 043B          38#11   40/16
VCAD   003D           1#27   32/ 7
VCBEL  050F          33/13   45# 2
```

Copyright © 1982 Osborne Computer Corporation

```
VCBKS  04E2    33/10   44#11
VCCLRS 051F    33/14   46# 2   56/31
VCCR   0540    33/ 8   47# 2
VCEOL  0054     1#33   32/21
VCHOME 04CB    33/15   43# 4
VCLF   0549    33/ 9   47# 9
VCLRS  001A     1#22   32/16   33/13
VCMCRT 054E    33/11   48# 2
VCMCUP 04D9    33/12   44# 2
VDELC  0057     1#30   32/18
VDELL  0052     1#32   32/20   57/28
VECTOR 0413    35#12   39/18   39/25
VEGH   0047     1#39   18/ 8   18/12   18/18   18/26   18/32   18/33
               18/46   32/10
VEHI   0028     1#35   32/12
VEUL   006D     1#37   32/14
VFLO   FFEA     1#48    7/12   41/24   87/ 8
VGRAPH 0554    31/ 7   49# 2
VHAGR  0469    31/ 9   40#39
VHALF  0465    31/ 8   40#32
VHOME  001E     1#23   33/14
VINC   0051     1#29   32/17
VINL   0045     1#31   32/19
VLDOR  06CE    16/48   54/14   57/49   64# 2
VLDIR  06E2    16/49   46/ 9   53/11   58/15   60/37   65# 2
VLL    0080     1#49    7/15   27/16   44/ 7   44/23   44/38   46/ 6
               55/ 6   57/42   58/ 9   59/13   60/28   61/17
VLOCK  0023     1#25   32/22
VNORM  0437    31/ 6   38# 2   40/ 9   40/22   40/35
VOUT80 055E    38/17   40/46   48/ 7   49/ 4   49/ 6   49# 9   49/52
VOUT85 0572    49/19   49#27
VOUT90 0573    42/45   44/20   44/44   46/17   47/ 7   47/11   49/25
               49#31   51/ 9   53/22   55/18   58/ 5
VOUT96 0575    49#37   54/19
VOUT97 0578    33/16   36/11   45/11   49#44   49/50   50/12   52/10
               56/18
VRTOFF EFEF     3#20   34/29   41/15   46/15   61/56   87/15
VSAD   0053     1#28   32/ 8
VSGH   0067     1#38   18/ 4   18/12   18/18   18/22   18/30   18/36
               18/42   32/ 9
VSHI   0029     1#34   32/11
VSUL   006C     1#36   32/13
VUNDER 0457    31/10   40# 6
VUNGR  045B    31/11   40#13
VUNHA  045F    31/12   40#19
VUNHAG 0463    31/13   40#26
VUNLK  0022     1#26   32/23
WBOOT  029C    15/ 9   21# 2
WBUSY  0EB8   128/20  133/19  137/44  138/59  143/25  147# 2
WRITE  0D37    16/24  135# 2
WSEC   0C23    16/27  122# 2
```

Copyright © 1982 Osborne Computer Corporation


```
        *ABS     0000  EFF0
        *CODE    EFF0  0000
        *DATA    EFF0  0000
                       ;.Date: 6/4/82
                       ;.Author: Roger W. Chapman
                       ;.Title: Osborne CP/M 2.2 CBIOS Rev 1.4
                       ;.Comments:

                       *NOTE*  FOR USE WITH OCCTXT6.AST ONLY

                       *       4D2007-00       MASTER  .ASM
                       *       2D2007-00       ASSY    .ASM
                       *       1D2007-00       LISTING .PRN
                       *       4D1007-00       MASTER  .COM
                       *       2D1007-00       ASSY    .COM

                       ;       +-----------------------+
                       ;       :                       :
                       ;       :        O C C          :
                       ;       :                       :
                       ;       :      C B I O S        :
                       ;       :                       :
                       ;       :  by Roger W. Chapman  :
                       ;       :                       :
                       ;       +-----------------------+

                       ;       Copyright 1982, Osborne.

                       ;       This product is a copyright program product of
                       ;       Osborne and is supplied for use with the Osborne.

      = 0016           VERS:   =       22



                               LINK    OCCBIO15.ASM    ;Jump Table
                               LINK    OCCBIO25.ASM    ;Key translation & initialization values
                               LINK    OCCBIO95.ASM    ;CP/M disk tables
                               LINK    OCCBIO35.ASM    ;ROM call interface
                               LINK    OCCBIO45.ASM    ;Non data transfer disk
                               LINK    OCCBIO55.ASM    ;cold and warm boot
                               LINK    OCCBIO65.ASM    ;Disk data transfer I/O
                               LINK    OCCBIO75.ASM    ;Utility routines
                               LINK    OCCBIO85.ASM    ;Iobyte dispatch table
                               LINK    OCCRAM15.ASM    ;Bios ram definitions
                               LINK    OCCRAM25.ASM    ;Common ram definitions
```

Copyright © 1982 Osborne Computer Corporation

```
 = 003B         MSIZE:   =        59
 = C300         CCP:     =        0C300H          ;location of ccp
 = E100         BIOS:    =        ccp+1600h
 = D306         BDOS:    =        ccp+806h

                         MSG      'DDT R (relocation value) = ',1F80H-BIOS
 'DDT R (relocation value) = 3E80'
                         MSG      'Assembling BIOS for LWA of ', LWAMEM,'h.'
 'Assembling BIOS for LWA of FFFFh.'

                ;        CP/M to host disk constants

 = 0400         HSTSIZ: =         1024            ;Blocking/Deblocking buffer size
 = 0010         FPYSIB: =         2048/128        ;Sectors in floppy disk block (Osborne single density block size = 2K)
 = 0008         FPYDIB: =         1024/128        ;Sectors in floppy disk block (Osborne double density block size = 1K)

                ;        CP/M disk control block equates which define the
                ;        disk types and maximum storage capability of each
                ;        disk type.

 = 0005         DSKS1:   =        5       ;Single density ( 256), single sided.
 = 000C         DSKD1:   =        0Ch     ;Double density (1024), single sided.
 = 0001         XEROX:   =        1       ;Single density ( 128), single sided.
 = 0008         IBM:     =        8       ;Double density ( 512), single sided.
 = 0008         DEC:     =        8

 = 002E         S1DSM:   =        ((40-3)*2*10)/FPYSIB
 = 0039         D1DSM:   =        ((40-3)*8*5 )/FPYDIB
 = 0053         XXDSM:   =        ((40-3)*1*18)/FPYDIB
 = 009C         IBMDSM: =         ((40-1)*4*8 )/FPYDIB
 = 00AB         DECDSM: =         ((40-2)*4*9 )/FPYDIB

                ;        BDOS constants on entry to write

 = 0000         WRALL:   =        0               ;write to allocated
 = 0001         WRDIR:   =        1               ;write to directory
 = 0002         WRUAL:   =        2               ;write to unallocated



                ;        ROM equates.

 = 0000         ENROM:   =        0               ;Port to enable ROM
 = 0001         DIROM:   =        1               ;Port to disable ROM
```

Copyright © 1982 Osborne Computer Corporation


```
                ;       Macro for generating Control Blocks for disk drives
                ;       The format of these disk control blocks are as follows:
                ;       16 bits = -> translation table.
                ;       48 bits = Work area for CP/M.
                ;       16 bits = -> DIRBUF.
                ;       16 bits = -> Parameter block.
                ;       16 bits = -> check vector.
                ;       16 bits = -> allocation vector.
 = 0000         NDSK:   SET     0                       ;Number of disk drives
 = 0000         NOFDD:  SET     0                       ;Number of floppy disk drives
 = 0000         ALVSZ:  SET     0                       ;Allocation vector size
 = 0000         CSVSZ:  SET     0                       ;Check vector size

                        LIST    M
                DPHGEN  MACRO   TYPE,XLATE,DIRBUF,DPBADR
                NDSK:   SET     NDSK+1
                        DW      %2
                        DW      0,0,0
                        DW      %3
                        DW      %4
                        DW      CSV+CSVSZ
                        DW      ALV+ALVSZ
                NOFDD:  SET     NOFDD+1
                CSVSZ:  SET     CSVSZ+(64/4)
                ALVSZ:  SET     ALVSZ+((DIDSM+7)/8)
                        ENDM



                ;       Macro for generating the Disk Parameter Blocks.
                ;
                ;       Disk type definition blocks for each particular mode.
                ;       The format of these areas are as follow:
                ;       8 bit = disk type code
                ;       16 bit = Sectors per track
                ;       8 bit  = Block shift
                ;       8 bit  = BS mask
                ;       8 bit  = Extent mask
                ;       16 bit = Disk size/1024 - 1.
                ;       16 bit = Directory size.
                ;       16 bit = Allocation for directory.
                ;       16 bit = check area size.
                ;       16 bit = offset to first track.

                DPBGEN  MACRO   TYPE,SPT,BSH,BSM,EXM,DSM,DIRSIZ,ALVMSK,OFFSET
                        DB      %1
                        DW      %2
                        DB      %3,%4,%5
                        DW      %6-1,%7-1,REV (%8)
                        DW      (%7+3)/4
                        DW      %9
                        ENDM
```

```
                        ;       The following jump table defines the entry points
                        ;       into the CBIOS for use by CP/M and other external
                        ;       routines; therfore the order of these jump cannot
                        ;       be modified.  The location of these jumps can only
                        ;       be modified by 400h locations, which is a restriction
                        ;       of MOVCPM.

0000  = E100                    ORG     BIOS

E100  C3EDE4                    JMP     CBOOT           ;Cold boot
E103  C313E5                    JMP     WBOOT           ;Warm boot
E106  C32EE7                    JMP     CONST           ;Console status (input)
E109  C333E7                    JMP     CONIN           ;Console input
E10C  C338E7                    JMP     CONOUT          ;Console output
E10F  C33FE7                    JMP     LIST            ;List output
E112  C355E7                    JMP     PUNCH           ;Punch output
E115  C35CE7                    JMP     READER          ;Reader input
E118  C33FE3                    JMP     HOME            ;Set track to zero
E11B  C350E3                    JMP     SELDSK          ;Select disk unit
E11E  C3BBE4                    JMP     SETTRK          ;Set track
E121  C3CEE4                    JMP     SETSEC          ;Set sector
E124  C3D3E4                    JMP     SETDMA          ;Set Disk Memory Address
E127  C381E5                    JMP     READ            ;Read from disk
E12A  C3CEE5                    JMP     WRITE           ;Write onto disk
E12D  C363E7                    JMP     LISTST          ;Return LST: device status
E130  C3D8E4                    JMP     SECTRN          ;Sector translation routine

                        ;       Extensions
E133  C3FEE2            RRI:    JMP     ROMRI
E136  C319E3                    JMP     ROMJMP
E139  CD12E3            FMTJ:   CALL    ROMCDE          ;Rom resident call
E13C  CD12E3            SBAUD:  CALL    ROMCDE

                        ;       IEEE-488 vectors
E13F  CD12E3            IEB1C:  CALL    ROMCDE          ;Control Out
E142  CD12E3            IEB2C:  CALL    ROMCDE          ;Status In
E145  CD12E3            IEB3C:  CALL    ROMCDE          ;Go To Standby
E148  CD12E3            IEB4C:  CALL    ROMCDE          ;Take Control
E14B  CD12E3            IEB5C:  CALL    ROMCDE          ;Output Interface Message
E14E  CD12E3            IEB6C:  CALL    ROMCDE          ;Output Device Message
E151  CD12E3            IEB7C:  CALL    ROMCDE          ;Input Device Message
E154  CD12E3            IEB8C:  CALL    ROMCDE          ;Parallel Poll

E157  CD12E3                    CALL    ROMCDE          ;extensions
E15A  CD12E3                    CALL    ROMCDE          ;  for
E15D  CD12E3                    CALL    ROMCDE          ;    memory-mapped video

E160  C33CE1                    JMP     ACICTL          ;hook for serial command port write
E163  C3B3E8                    JMP     ACISTAT         ;hook for serial status port read
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        ;;;;;;;;;;;;;;;;;;;;;;;
                        ;
                        ;       This area is reserved data storage area for
                        ;       the set-up program to install printer drivers,
                        ;       function keys, auto boot command, iobyte value,
                        ;       and auto horizontal scroll flag
                        ;                                               RWC
                        ;;;;;;;;;;;;;;;;;;;;;;;

E166  40                IOBITE: DB      40h             ;default to serial printer=40h
                                                        ;          parallel printer=80h
                                                        ;              IEEE printer=c0h
E167  00                PRNTER: DB      00h             ;default to standard serial=0
                                                        ;Qume           ETX/ACK    =1
                                                        ;Diablo         XON/XOFF   =2

E168  FF                AHSENB: DB      TRUE            ;auto horizontal scroll enable

E169  55                BRATE:  DB      SI.120          ;default baud rate = 1200

E16A  80                SCRSZE: DB      128             ;default screen size = 128


E16B  A5E1              XLTBL:  DW      CNTRL0          ;Fixed length table
E16D  A6E1                      DW      CNTRL1          ;contains pointers
E16F  A7E1                      DW      CNTRL2          ;to strings
E171  A8E1                      DW      CNTRL3          ;to decode
E173  A9E1                      DW      CNTRL4          ;function keys
E175  AAE1                      DW      CNTRL5
E177  ABE1                      DW      CNTRL6
E179  ACE1                      DW      CNTRL7
E17B  ADE1                      DW      CNTRL8
E17D  AEE1                      DW      CNTRL9
E17F  AFE1                      DW      UP
E181  B0E1                      DW      RIGHT
E183  B1E1                      DW      DOWN
E185  B2E1                      DW      LEFT
E187  B3E1                      DW      EOTBL           ;end of table address

E189  04                IEEEAD: DB      4               ;IEEE device address

E18A  00                PINTFG: DB      0               ;Flag indicates if printer has been initialized

E18B  00                PINIT:  DB      0               ;Length of string
E18C  = 0010                    DS      16              ;Printer initialization string

E19C  01                ACMD:   DB      1               ;auto command = 0 ignore auto boot
                                                        ;             = 1 auto on cold boot
                                                        ;             = 2 auto on warm boot
                                                        ;             = 3 auto on both
E19D  07                CAUTO:  DB      CAUTOL          ;length of auto command here
E19E  4155544F53                DB      'AUTOST '       ;auto command goes here
      = 0007            CAUTOL: =       *-CAUTO-1
```

```
E1A5  30          CNTRL0: DB      '0'          ;Variable length table
E1A6  31          CNTRL1: DB      '1'          ;is placed here by set-up
E1A7  32          CNTRL2: DB      '2'          ;program, with xlttbl
E1A8  33          CNTRL3: DB      '3'          ;pointing to the entries
E1A9  34          CNTRL4: DB      '4'
E1AA  35          CNTRL5: DB      '5'          ;Default values for the
E1AB  36          CNTRL6: DB      '6'          ;control numerics
E1AC  37          CNTRL7: DB      '7'          ;are the numbers on the keys
E1AD  38          CNTRL8: DB      '8'
E1AE  39          CNTRL9: DB      '9'
E1AF  0B          UP:     DB      'K'-40h      ;Default values
E1B0  0C          RIGHT:  DB      'L'-40h      ;for the cusor
E1B1  0A          DOWN:   DB      'J'-40h      ;keys are standard
E1B2  08          LEFT:   DB      'H'-40h      ;values for CP/M
      = E1B3      EOTBL:  =       *



E1B3  = E200              ORG     BIOS+256     ;space reseverd for full function
                                               ;key decoding and 16 byte auto
                                               ;boot command
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        ;       Control Blocks for disk drives

E200                    DPBASE:
E200                            DPHGEN  DSKD1,DDXLTS,DIRBUF,DPBD1+1     ;Drive A:
     += 0001            NDSK:   SET     NDSK+1
E200 +0000                      DW      DDXLTS
E202 +0000000000                DW      0,0,0
E208 +80EE                      DW      DIRBUF
E20A +9FE2                      DW      DPBD1+1
E20C +3CE9                      DW      CSV+CSVSZ
E20E +0CE9                      DW      ALV+ALVSZ
     += 0001            NOFDD:  SET     NOFDD+1
     += 0010            CSVSZ:  SET     CSVSZ+(64/4)
     += 0018            ALVSZ:  SET     ALVSZ+((D1DSM+7)/8)

E210                            DPHGEN  DSKD1,DDXLTS,DIRBUF,DPBD1+1     ;Drive B:
     += 0002            NDSK:   SET     NDSK+1
E210 +0000                      DW      DDXLTS
E212 +0000000000                DW      0,0,0
E218 +80EE                      DW      DIRBUF
E21A +9FE2                      DW      DPBD1+1
E21C +4CE9                      DW      CSV+CSVSZ
E21E +24E9                      DW      ALV+ALVSZ
     += 0002            NOFDD:  SET     NOFDD+1
     += 0020            CSVSZ:  SET     CSVSZ+(64/4)
     += 0030            ALVSZ:  SET     ALVSZ+((D1DSM+7)/8)
```

Copyright ©1982 Osborne Computer Corporation

```
E220                XTAB:   ;Translation table addresses

E220  0000                  DW      DDXLTS          ;Double density Osborne tranlation table address
E222  2CE2                  DW      XLTS            ;Single density Osborne
E224  40E2                  DW      XXXLTS          ;Xerox translation table address
E226  0000                  DW      IBMXLT          ;IBM translation table address
E228  52E2                  DW      DECXLT          ;DEC translation table address
E22A  76E2                  DW      XTRXLT          ;User defined translation table address

      = 0000        DDXLTS: =       0               ;Translation table for DOUBLE DENSITY OSBORNE  2 to 1

E22C                XLTS:   ;Translation table 2 to 1

E22C  0001040508            DB      0, 1,    4, 5,    8, 9,    12,13,    16,17
E236  020306070A            DB      2, 3,    6, 7,    10,11,   14,15,    18,19

E240                XXXLTS: ;XEROX TRANSLATION TABLE  5 to 1

E240  00050A0F              DB      0,    5,    10,    15
E244  02070C11              DB      2,    7,    12,    17
E248  04090E                DB      4,    9,    14
E24B  01060B10              DB      1,    6,    11,    16
E24F  030800                DB      3,    8,    13

      = 0000        IBMXLT: =       0               ;IBM TRANSLATION TABLE, No translation  1 to 1

E252                DECXLT: ;DEC TRANSLATION TABLE  2 to 1
E252  0001020308            DB      0, 1, 2, 3,    8, 9,10,11,    16,17,18,19,    24,25,26,27,    32,33,34,35
E266  040506070C            DB      4, 5, 6, 7,    12,13,14,15,   20,21,22,23,    28,29,30,31

E276  = 0028        XTRXLT: DS      40              ;Space for user defined expansion
```

Copyright ©1982 Osborne Computer Corporation

```
                   ;       Disk type definition blocks for each particular mode.

E29E               DPBSTART:                           ;Start of Disk parameter blocks, used by select disk routine

E29E               DPBD1:                              ;Osborne Double density, single sided
E29E                       DPBGEN  DSKD1,8*5 ,3, 7,0,D1DSM ,64,1100000000000000B,3
E29E +0C                   DB      DSKD1
E29F +2800                 DW      8*5
E2A1 +030700               DB      3,7,0
E2A4 +B8003F00C0           DW      D1DSM-1,64-1,REV (1100000000000000B)
E2AA +1000                 DW      (64+3)/4
E2AC +0300                 DW      3

E2AE               DPBS1:                              ;Osborne Single density, single sided.
E2AE                       DPBGEN  DSKS1,2*10,4,15,1,S1DSM ,64,1000000000000000B,3
E2AE +05                   DB      DSKS1
E2AF +1400                 DW      2*10
E2B1 +040F01               DB      4,15,1
E2B4 +2D003F0080           DW      S1DSM-1,64-1,REV (1000000000000000B)
E2BA +1000                 DW      (64+3)/4
E2BC +0300                 DW      3

E2BE               DPBX0:
E2BE                       DPBGEN  XEROX,1*18,3, 7,0,XXDSM ,32,1000000000000000B,3
E2BE +01                   DB      XEROX
E2BF +1200                 DW      1*18
E2C1 +030700               DB      3,7,0
E2C4 +52001F0080           DW      XXDSM-1,32-1,REV (1000000000000000B)
E2CA +0800                 DW      (32+3)/4
E2CC +0300                 DW      3

E2CE               DPBIBM:
E2CE                       DPBGEN  IBM   ,4*8 ,3, 7,0,IBMDSM,64,1100000000000000B,1
E2CE +08                   DB      IBM
E2CF +2000                 DW      4*8
E2D1 +030700               DB      3,7,0
E2D4 +9B003F00C0           DW      IBMDSM-1,64-1,REV (1100000000000000B)
E2DA +1000                 DW      (64+3)/4
E2DC +0100                 DW      1

E2DE               DPBDEC:
E2DE                       DPBGEN  DEC   ,4*9 ,3, 7,0,DECDSM,64,1100000000000000B,2
E2DE +08                   DB      DEC
E2DF +2400                 DW      4*9
E2E1 +030700               DB      3,7,0
E2E4 +AA003F00C0           DW      DECDSM-1,64-1,REV (1100000000000000B)
E2EA +1000                 DW      (64+3)/4
E2EC +0200                 DW      2

E2EE               DPBXTR:
E2EE  = 0010               DS      16


      = 0006       NUMDPB: =       (*-DPBSTART)/10H
```

Copyright © 1982 Osborne Computer Corporation

```
E2FE                 ROMRI:
                     ;        Exits ROM resident Interrupt routine.

E2FE  3A08EF                  LD      A,ROMRAM
E301  4F                      MOV     C,A              ;port
E302  ED79         $          OT,C    A                ;set ROM or RAM enabled
E304  FDE1         $          POP     IY
E306  DDE1         $          POP     IX
E308  E1                      POP     HL
E309  D1                      POP     DE
E30A  C1                      POP     BC
E30B  F1                      POP     AF
E30C  ED7B6FEF     $          LD      SP,IESTK         ;reset to interrupt entry stk
E310  FB                      EI
E311  C9                      RET
```

Copyright © 1982 Osborne Computer Corporation

```
E312                    ROMCDE:
                        ;       Calls ROM resident processor
                        ;
                        ;       Entry   DE = resident processor to call biased
                        ;               by CBIOS jump vector.
                        ;       NOTE: ROM jump vector must match CBIOS vector
                        ;
                        ;       Entry at ROMCD1 with low digit of CBIOS vector in reg E

E312  D1                        POP     DE                  ;Get calling address
E313  7B                        MOV     A,E
E314  D603                      SUI     3
E316  5F                        MOV     E,A

E317  1601              ROMCD1: LDK     D,high (ROMVEC)

E319                    ROMJMP:
                        ;       Entry here to jump to ROM function code directly
                        ;
                        ;       Entry   DE = ROM jump address
                        ;               BC, HL, IX = any parameters

E319  FDE5          $           PUSH    IY                  ;Save user Index registers
E31B  DDE5          $           PUSH    IX

E31D  F3                        DI                          ;Set up local stack for ROM
E31E  D9            $           EXX
E31F  210000                    LDK     HL,0
E322  39                        ADD     HL,SP               ;Old stack to HL
E323  31C1EF                    LDK     SP,BIOSTK
E326  E5                        PUSH    HL                  ;Save old stack pointer
E327  D9            $           EXX

E328  CDD7E8                    CALL    SW2ROM
E32B  CD3AE3                    CALL    GOROM               ;Go to ROM address (DE) and return to next instruction
E32E  CDE2E8                    CALL    SW2RAM

E331  FDE1          $           POP     IY                  ;Restore old stack pointer
E333  FDF9          $           MOV     SP,IY

E335  DDE1          $           POP     IX                  ;Restore user Index registers
E337  FDE1          $           POP     IY

E339  C9                        RET


E33A                    GOROM:
                        ;       This routine is used to simulate a CALL (DE) instruction

E33A  D5                        PUSH    DE                  ;ROM jump address to IY
E33B  FDE1          $           POP     IY
E33D  FDE9          $           JMP     [IY]                ;Go to ROM, ROM will RET to next instruction
```

Copyright © 1982 Osborne Computer Corporation

```
E33F                  HOME:
                      ;
                      ;       Returns disk to home.  This routine sets the track number
                      ;       to zero.  The current host disk buffer is flushed to the
                      ;       disk.

E33F  CDC1E6                  CALL    FLUSH                   ;Flush host buffer
E342  AF                      XRA     A
E343  3250EF                  STO     A,HSTACT                ;Clear host active flag
E346  3252EF                  STO     A,UNACNT                ;Clear sector count
E349  3219EF                  STO     A,SEKTRK
E34C  321AEF                  STO     A,SEKTRK+1
E34F  C9                      RET
```

Copyright © 1982 Osborne Computer Corporation


```
E350                   SELDSK:
                       ;       Selects disk drive for next transfer.
                       ;
                       ;       ENTRY   C  = disk selection value (0..15).
                       ;               DE and 1 = 0, first call for this disk.
                       ;
                       ;       EXIT    HL = 0, if drive not selectable.
                       ;               HL = DPH address if drive is selected.

E350                           PROC
E350  DDE5         $           PUSH    IX              ;Save user IX
E352  79                       MOV     A,C
E353  FE02                     CPI     NDSK
E355  D2A4E3                   JNC     SELD            ;If invalid drive number
E358  3213EF                   STD     A,SEKDSK
E35B  69                       MOV     L,C
E35C  2600                     MVI     H,0
E35E  29                       ADD     HL,HL           ;*2
E35F  29                       ADD     HL,HL           ;*4
E360  29                       ADD     HL,HL           ;*8
E361  29                       ADD     HL,HL           ;*16
E362  7B                       MOV     A,E             ;get initial bit
E363  1100E2                   LDK     DE,DPBASE
E366  19                       ADD     HL,DE           ;HL = DPH address
E367  2206E9                   STD     HL,SAVDPH

E36A  5F                       MOV     E,A             ;Restore initial bit
E36B  CDFAE3                   CALL    CHKSEL          ;Check to see if density needs to be determined
E36E  C28CE3                   JNZ     SELER           ;Unable to determine density, error return

E371  110A00                   LDK     DE,10           ;form dob address
E374  19                       ADD     HL,DE           ;to get type
E375  5E                       LD      E,[HL]
E376  23                       INC     HL
E377  56                       LD      D,[HL]          ;dpb address in DE
E378  1B                       DEC     DE
E379  1A                       LD      A,[DE]          ;get disk type
E37A  325CE9                   STD     A,SEKTYP        ;and store value

E37D  2ACCEF                   LD      HL,SEQ          ;Get current sequence count
E380  DD7500       $           STD     L,[IX+0]
E383  DD7401       $           STD     H,[IX+1]        ;Store seq # in LASTA or LASTB

E386  2A06E9                   LD      HL,SAVDPH
E389  DDE1         $           POP     IX              ;Restore user IX
E38B  C9                       RET
```

```
                        ;       Select disk error handling

E38C  11D8E3            SELER:  LDK     DE,FORERR
E38F  E601                      ANI     1               ;Is diskette unformatted?
E391  2003 ^E396$               JRNZ    SELER1          ;Yes
E393  1132E3                    LDK     DE,DENERR       ;No, get density error message

E396  3A1BEF            SELER1: LD      A,SEKDSK
E399  C641                      ADD     A,'A'
E39B  32D7E3                    STO     A,DRV           ;Indicate drive in message
E39E  32F9E3                    STO     A,DRV1
E3A1  CD11E7                    CALL    PRINT           ;Print appropriate message on console

E3A4  DDE1            $ SELD:   POP     IX              ;Restore user IX
E3A6  210000                    LDK     HL,0
E3A9  3A0400                    LDA     CDISK
E3AC  91                        SUB     C
E3AD  C0                        RNZ                     ;If default drive not in error
E3AE  320400                    STO     A,CDISK
E3B1  C9                        RET

E3B2  25                DENERR: DB      DENL
E3B3  0D0A43616E                DB      CR,LF,'Can''t recognize diskette on drive '
E3D7  = 0001            DRV:    DS      1
      = 0025            DENL:   =       *-DENERR-1

E3D8  21                FORERR: DB      FORL
E3D9  0D0A556E66                DB      CR,LF,'Unformatted diskette on drive '
E3F9  = 0001            DRV1:   DS      1
      = 0021            FORL:   =       *-FORERR-1
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation


```
E3FA                 CHKSEL:
                     ;       Determines if new DPB should be established
                     ;
                     ;       ENTRY   C = disk selection value (0..15)
                     ;               E & 01 = 0, first call for this disk
                     ;
                     ;       EXIT    IX = address of drive sequence number
                     ;               Z status bit set, good return
                     ;               Z status bit clear, error return

E3FA                         PROC
E3FA  E5                     PUSH    HL              ;Save user HL
E3FB  C5                     PUSH    BC
E3FC  DD2100E9   $           LDK     IX,LASTA        ;Get last count for selected drive
E400  79                     MOV     A,C             ;Current drive to A - reg
E401  B7                     ORA     A               ;Is this drive A?
E402  2804 ^E408$            JRZ     CHKSEQ          ;Yes, check sequence number
E404  DD23       $           INC     IX              ;No, form address for B
E406  DD23       $           INC     IX

E408  DD4E00     $ CHKSEQ: LD      C,[IX+0]        ;Get last sequence number for this drive
E40B  DD4601     $           LD      B,[IX+1]
E40E  2ACCEF                 LD      HL,SEQ          ;Get current sequence number
E411  ED42       $           SBC     HL,BC           ;Compare seq #'s
E413  C1                     POP     BC
E414  DDE5       $           PUSH    IX

E416  CB43       $           BIT     0,E             ;First call for this disk?
E418  2800 ^E427$            JRZ     :10             ;Yes, fill in DPB

E41A  7C                     MOV     A,H             ;No, check elasped time
E41B  B7                     ORA     A               ;Elasped time < 4 sec
E41C  280E ^E42C$            JRZ     GRET            ;Yes, good return
E41E  FE02                   CPI     02
E420  3005 ^E427$            JRNC    :10
E422  7D                     MOV     A,L
E423  FEA0                   CPI     0A0H            ;Elasped time > 6 sec
E425  3805 ^E42C$            JRC     GRET            ;No, indicate good return

E427  CD31E4       :10:      CALL    GDEN
E42A  1801 ^E42D$            JR      RETURN

E42C  AF           GRET:     XRA     A

E42D  DDE1       $ RETURN:   POP     IX
E42F  E1                     POP     HL
E430  C9                     RET
```

Copyright © 1982 Osborne Computer Corporation


```
E47F  DD7E00    $       LD      A,[IX+0]
E482  77                STO     A,[HL]
E483  23                INX     HL
E484  DD7E01    $       LD      A,[IX+1]
E487  77                STO     A,[HL]

E488  AF                XRA     A               ;Indicate good return
E489  C9                RET


                ;       Error return section

E48A  3E01      SELD1:  MVI     A,1             ;Indicate unformatted disk Z-flag = clear, A = 1
E48C  C9                RET

E48D  F602      SELD2:  ORI     02              ;Indicate unrecognizable disk Z-flag = clear, A = 2
E48F  C9                RET
```

Copyright © 1982 Osborne Computer Corporation


```
E490                GETDEN:
                    ;       Gets density of selected disk
                    ;
                    ;       ENTRY   SEKDSK = Current drive
                    ;
                    ;       EXIT    C = TYPE            7   6   5   4   3   2   1   0
                    ;                                   !   !   !   !   !   !   !   !
                    ;               Undef = 0 <---------+---+---+---+   !   !   !   !
                    ;               Bytes/sector <----------------------+---+   !   !
                    ;               Sides <-------------------------------------+   !
                    ;               Density <---------------------------------------+
                    ;
                    ;               B = # of physical sectors per track
                    ;               A = 0, good return
                    ;               A <> 0, error return
                    ;               Z-bit set, good return
                    ;               Z-bit clear, error return

      = 0130        SENDEN: =       130H

E490                        PROC
E490  3A17EF                LD      A,SDISK         ;Save current value
E493  3208E9                STO     A,TEMDSK        ;of SDISK
E496  3AD0EF                LD      A,SAVTYP        ;and SAVTYP
E499  3209E9                STO     A,TEMTYP        ;(SDISK & SAVTYP are used by SENDEN)

E49C  3A18EF                LD      A,SEKDSK        ;Disk to be select
E49F  3217EF                STO     A,SDISK         ;in SDISK (parameter to SENDEN)

E4A2  113001                LDK     DE,SENDEN       ;Call SENDEN
E4A5  CD19E3                CALL    ROMJMP

E4A8  3A08E9                LD      A,TEMDSK        ;Restore caller's value of SDISK
E4AB  3217EF                STO     A,SDISK

E4AE  3AD0EF                LD      A,SAVTYP        ;Exit TYPE parameter
E4B1  4F                    MOV     C,A             ;into C - Reg

E4B2  3A09E9                LD      A,TEMTYP        ;Restore caller's SAVTYP
E4B5  32D0EF                STO     A,SAVTYP

E4B8  C0                    RNZ     ERRET           ;Error return, flag set by SENDEN

E4B9  AF                    XRA     A               ;Indicate good return
E4BA  C9                    RET
```

```
E4BB                    SETTRK:
                        ;       Sets track number.  The track number is saved for later
                        ;       use during a disk transfer operation.
                        ;
                        ;       ENTRY   BC = track number.

E4BB  ED4319EF  $               STD     BC,SEKTRK       ;Set track
E4BF  2A53EF                    LHLD    UNATRK
E4C2  7D                        MOV     A,L
E4C3  A9                        XRA     C
E4C4  4F                        MOV     C,A
E4C5  7C                        MOV     A,H
E4C6  A8                        XRA     B
E4C7  B1                        ORA     C
E4C8  C8                        RZ                      ;If same track
                        ;       JMP     CUNACT


E4C9                    CUNACT:
                        ;       Clear Unallocated block count (force pre-reads).

E4C9  AF                        XRA     A               ;A = 0
E4CA  3252EF                    STD     A,UNACNT                ;Clear unallocated block count
E4CD  C9                        RET


E4CE                    SETSEC:
                        ;       Set the sector for later use in the disk transfer.  No
                        ;       actual disk operations are perfomed.
                        ;
                        ;       Entry   BC = sector number.

E4CE  79                        MOV     A,C
E4CF  3220EF                    STD     A,TEMSEC        ;sector to seek
E4D2  C9                        RET


E4D3                    SETDMA:
                        ;       Sets Disk memory address for subsequent disk read or
                        ;       write routines.  This address is saved in DMAADR until
                        ;       the disk transfer is performed.
                        ;
                        ;       ENTRY   BC = Disk memory address.
                        ;
                        ;       EXIT    DMAADR = BC.

E4D3  ED4311EF  $               STD     BC,DMAADR
E4D7  C9                        RET
```

Copyright ©1982 Osborne Computer Corporation

```
E4D8                SECTRN:
                    ;       Translates sector number from logical to physical.
                    ;
                    ;       ENTRY   DE = 0, no translation required.
                    ;               DE = translation table address.
                    ;               BC = sector number to translate.
                    ;
                    ;       EXIT    HL = translated sector.

E4D8  3A55EF                LDA     UNASEC
E4DB  B9                    CMP     C
E4DC  C4C9E4                CNZ     CUNACT          ;If sectors do not match
E4DF  79                    MOV     A,C
E4E0  3256EF                STO     A,LOGSEC
E4E3  69                    MOV     L,C
E4E4  60                    MOV     H,B

E4E5  7B                    MOV     A,E             ;Check if translation is required
E4E6  B2                    ORA     D
E4E7  C8                    RZ                      ;None required, return

E4E8  19            TRAN:   ADD     HL,DE           ;Translation required
E4E9  6E                    MOV     L,M
E4EA  2600                  MVI     H,0
E4EC  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        ;       B o o t   C P / M   f r o m   d i s k.
                        ;
                        ;       The CBOOT entry point gets control from the cold start
                        ;       loader and is responsible for the basic system initial-
                        ;       ization.  This includes outputting a sign-on message and
                        ;       initializing the following page zero locations:
                        ;
                        ;          0,1,2: Set to the warmstart jump vector.
                        ;              3: Set to the initial IOBYTE value.
                        ;              4: Default and logged on drive.
                        ;          5,6,7: Set to a jump to BDOS.
                        ;
                        ;       The WBOOT entry point gets control when a warm start
                        ;       occurs, a ^C from the console, a jump to BDOS (function
                        ;       0), or a jump to location zero.  The WBOOT routine reads
                        ;       the CCP and BDOS from the appropriate disk sectors.
                        ;       WBOOT must also re-initialize locations 0,1,2 and 5,6,7.
                        ;       The WBOOT routines exits with the C register set to the
                        ;       appropriate drive selection value.  The exit address
                        ;       is to the CCP routine.
E4ED                    CBOOT:
E4ED  3100CB                    LDK     SP,CCP

E4F0  CDE2E8                    CALL    SW2RAM

E4F3  3A66E1                    LD      A,IOBITE        ;get iobyte value
E4F6  320300                    STO     A,IOBYTE        ;Set I/O byte to default

E4F9  3A69E1                    LD      A,BRATE
E4FC  4F                        MOV     C,A
E4FD  CD3CE1                    CALL    SBAUD           ;set baud rate

E500  3A6AE1                    LDA     SCRSZE
E503  326CEF                    STA     LLIMIT          ;set screen size

E506  3A89E1                    LD      A,IEEEAD        ;Get IEEE device address
E509  320BEF                    STO     A,IEADR         ;and save it in BMRAM

E50C  AF                        XRA     A
E50D  320400                    STO     A,CDISK         ;Set current drive to A
E510  3C                        INC     A

E511  180E ^E521$               JR      BCCP            ;Do CP/M

E513                    WBOOT:                          ;Warm boot
E513  3100CB                    LDK     SP,CCP
E516  CD3FE3                    CALL    HOME            ;flush any buffer

E519  110301            BCPM:   LDK     DE,ROMVEC+3*1   ;Set ROM vector address
E51C  CD19E3                    CALL    ROMJMP

E51F  3E02                      MVI     A,2             ;indicate warm boot

E521                    BCCP:   ;Entry  A = 01, if cold boot
                        ;               A = 02, if warm boot

E521  F5                        PUSH    AF              ;Save entry
E522  018000                    LDK     BC,DBUF         ;Set default data transfer address
```

Copyright © 1982 Osborne Computer Corporation

```
E525  CDD3E4                 CALL    SETDMA
E528  2180EA                 LDK     HL,HSTBUF
E52B  220FEF                 STO     HL,DMADR         ;set ROM DMA address

                   ;         Clear console control ESC cell
E52E  AF                     XRA     A
E52F  32605F                 STO     A,ESCH           ;clear ESC

                   ;         Set-up low core pointer cells
E532  3EC3                   LDK     A,0C3h           ;Store jumps in low memory
E534  320000                 STO     A,0
E537  320500                 STO     A,5
E53A  2103E1                 LDK     HL,BIOS+3
E53D  220100                 STO     HL,1
E540  2106D3                 LDK     HL,BDOS
E543  220600                 STO     HL,6

E546  219DE1                 LDK     HL,CAUTO
E549  C1                     POP     BC               ;cold/warm indicator in b
E54A  3A9CE1                 LD      A,ACMD
E54D  A0                     AND     B
E54E  284A ^E59A$            JRZ     DONE

E550  7E                     LD      A,[HL]
E551  B7                     ORA     A
E552  2846 ^E59A$            JRZ     DONE

E554  1107CB                 LDK     DE,CCP+7
E557  0600                   LDK     B,0
E559  4F                     MOV     C,A
E55A  EDB0      $            LDIR                     ;Move command line to buffer

E55C  110000                 LDK     DE,0
E55F  1847 ^E5A8$            JR      DONE1

E561  38           SIGNON:  DB      SIGNL            ;Length of signon message
E562  1A                     DB      'Z'-40h
E563  4F73626F72             DB      'Osborne Computer System'
E57A  0D0A                   DB      CR,LF
E57C  3539                   DB      MSIZE/10+'0',MSIZE mod 10 + '0'
E57E  6B2043502F             DB      'k CP/M vers ',VERS/10+'0','.',VERS mod 10 + '0'
E58D  0D0A434249             DB      CR,LF,'CBIOS 1.4',CR,LF
      = 0038       SIGNL:    =       *-SIGNON-1

E59A  3E02         DONE:     LDK     A,2
E59C  B8                     CMP     B
E59D  2806 ^E5A5$            JRZ     DONEO

E59F  1161E5                 LDK     DE,SIGNON
E5A2  CD11E7                 CALL    PRINT

E5A5  110300       DONEO:    LDK     DE,3

E5A8  2100CB       DONE1:    LDK     HL,CCP
E5AB  19                     ADD     HL,DE
E5AC  3A0400                 LD      A,CDISK
E5AF  4F                     MOV     C,A
E5B0  E9                     JMP     [HL]
```

Copyright © 1982 Osborne Computer Corporation

```
E5B1                    READ:
                        ;       a CP/M 128 byte sector.
                        ;
                        ;       EXIT    A = 0, successful read operation.
                        ;               A = 1, unsucessful read operation.
                        ;               Z bit = 1, successful read operation.
                        ;               Z bit = 0, unsuccessful read operation.

E5B1                            PROC
E5B1  CD06E7                    CALL    MVINFO          ;Move information for transfer
E5B4  AF                        XRA     A               ;Set flag to force a read
E5B5  3252EF                    STO     A,UNACNT                ;Clear sector counter
E5B8  CD44E6                    CALL    FILL            ;Fill buffer with data
E5BB  E1                        POP     HL
E5BC  D1                        POP     DE
E5BD  018000                    LDK     BC,128          ;Move 128 bytes
E5C0  EDB0          $           LDIR
E5C2  3A22EF                    LD      A,ERFLAG
E5C5  B7                        ORA     A
E5C6  C8                        RZ                      ;If no error

E5C7  AF                        XRA     A
E5C8  3250EF                    STO     A,HSTACT        ;Clear host active (A = 0)
E5CB  F601                      ORI     1               ;Indicate error
E5CD  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
E5CE                    WRITE:
                        ;       the selected 128 byte CP/M sector.
                        ;
                        ;       ENTRY   C = 0, write to a previously allocated block.
                        ;               C = 1, write to the directory.
                        ;               C = 2, write to the first sector of unallocated
                        ;               data block.
                        ;
                        ;       EXIT    A = 0, write was successful.
                        ;               A = 1, write was unsucessful.
                        ;               Z bit = 1, write was successful.
                        ;               Z bit = 0, write was unsucessful.

E5CE                            PROC
E5CE  CD05E7                    CALL    MVINFO          ;Move information for transfer
E5D1  79                        MOV     A,C             ;Write type in c
E5D2  3223EF                    STO     A,WRTYPE
E5D5  FE02                      CPI     WRUAL
E5D7  201A ^E5F3$               JRNZ    WRIT2           ;If write to allocated
E5D9  3A5CE9                    LD      A,SEKTYP
E5DC  FE05                      CPI     5               ;Check for 2K block size
E5DE  3E10                      LDK     A,2048/128
E5E0  2802 ^E5E4$               JRZ     WRIT1           ;Type = Osborne single density (2K block size)
E5E2  3E08                      LDK     A,1024/128      ;Otherwise 1K block size

E5E4  3252EF            WRIT1:  STO     A,UNACNT
E5E7  2A19EF                    LD      HL,SEKTRK
E5EA  2253EF                    STO     HL,UNATRK       ;UNATRK = SEKTRK
E5ED  3A56EF                    LD      A,LOGSEC
E5F0  3C                        INC     A
E5F1  1828 ^E61E$               JR      WRIT3

E5F3  2152EF            WRIT2:  LDK     HL,UNACNT
E5F6  7E                        LD      A,[hl]
E5F7  B7                        ORA     A
E5F8  CA23E6                    JZ      WRIT4           ;If no unallocated records
E5FB  35                        DEC     [hl]            ;dec unalloc record count

E5FC  21FAE1                    LDK     HL,DPBASE-16+10 ;Get number of sectors per track
E5FF  111000                    LDK     DE,16           ;To point to next DPB
E602  3A1BEF                    LD      A,SEKDSK
E605  47                        MOV     B,A
E606  04                        INC     B

E607  19                WRIT25: ADD     HL,DE
E608  10FD ^E607$               DJNZ    WRIT25

E60A  5E                        LD      E,[HL]
E60B  23                        INC     HL
E60C  56                        LD      D,[HL]
E60D  1A                        LD      A,[DE]          ;Number of sectors per track in A reg
E60E  47                        MOV     B,A
E60F  3A55EF                    LD      A,UNASEC        ;Increment logical sector
E612  3C                        INC     A
E613  B8                        CMP     B
E614  2008 ^E61E$               JRNZ    WRIT3           ;If not end of track
E616  2A53EF                    LD      HL,UNATRK
E619  23                        INC     HL
E61A  2253EF                    STO     HL,UNATRK
```

Copyright © 1982 Osborne Computer Corporation


```
E61D  AF                    XRA     A

E61E  3255EF        WRIT3:  STO     A,UNASEC
E621  3EFF                  LDK     A,0FFh

E623  CD44E6        WRIT4:  CALL    FILL
E626  D1                    POP     DE
E627  E1                    POP     HL
E628  018000                LDK     BC,128
E62B  EDB0      $           LDIR

E62D  3E01                  LDK     A,1
E62F  3251EF                STO     A,HSTWRT        ;HSTWRT = 1
E632  3A22EF                LD      A,ERFLAG
E635  B7                    ORA     A
E636  C0                    RNZ                     ;If any errors occurred

E637  3A23EF                LD      A,WRTYPE        ;write type
E63A  FE01                  CPI     WRDIR           ;to directory?
E63C  CCC1E6                CZ      FLUSH           ;Force write of directory
E63F  3A22EF                LD      A,ERFLAG
E642  B7                    ORA     A
E643  C9                    RET
```

```
E644                  FILL:
                      ;        Fills host buffer with approprite host sector.
                      ;
                      ;        ENTRY   A = 0, Read required if not in buffer.
                      ;                Otherwise read not required.
                      ;
                      ;        EXIT    On exit the stack will contain the following
                      ;                values:
                      ;              POP     x         ;x = host record address.
                      ;              POP     y         ;y = caller's buffer address.

E644                           proc
E644  3221EF                   STO     A,RDFLAG        ;Save read flag

E647  3A5CE9                   LD      A,SEKTYP
E64A  0F                       RRC
E64B  0F                       RRC
E64C  E603                     ANI     03
E64E  47                       MOV     B,A
E64F  1180EA                   LDK     DE,HSTBUF       ;initial offset
E652  3A18EF                   LD      A,SEKSEC        ;Get logical sector
E655  218000                   LDK     HL,128          ;128 byte records
E658  2808 ^E665$              JRZ     FILL35          ;Jump when sector size <> 128, no deblocking necessary

E65A  EB              FILL2:   EX      DE,HL
E65B  0F                       RRC
E65C  3001 ^E65F$              JRNC    FILL3           ;If low bit not set
E65E  19                       ADD     HL,DE           ;Add bias to offset

E65F  EB              FILL3:   EX      DE,HL
E660  29                       ADD     HL,HL
E661  E67F                     ANI     07Fh            ;Mask sector
E663  10F5 ^E65A$              DJNZ    FILL2

E665  3218EF          FILL35:  STO     A,SEKSEC        ;SEKSEC = physical sector - 1
E668  2A11EF                   LD      HL,DMAADR
                                                       ;Set return parameters
E66B  E3                       XTHL
E66C  D5                       PUSH    DE
E66D  E5                       PUSH    HL              ;Set return address

E66E  2150EF                   LDK     HL,HSTACT       ;host active flag
E671  7E                       LD      A,[HL]
E672  3601                     STO     1,[HL]          ;always becomes 1
E674  B7                       ORA     A
E675  281D ^E694$              JRZ     FILL6           ;If host buffer inactive
E677  215DE9                   LDK     HL,HSTTYP
E67A  3A5CE9                   LD      A,SEKTYP
E67D  BE                       CMP     [HL]
E67E  2011 ^E691$              JRNZ    FILL5
E680  211CEF                   LDK     HL,HSTSEC
E683  1118EF                   LDK     DE,SEKSEC
E686  0604                     LDK     B,SEKDSK-SEKSEC+1

E688  1A              FILL4:   LD      A,[DE]
E689  BE                       CMP     [HL]
E68A  2005 ^E691$              JRNZ    FILL5           ;If mis-match
E68C  23                       INC     HL
E68D  13                       INC     DE
E68E  10F8 ^E688$              DJNZ    FILL4           ;If all bytes not checked
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
E690  C9                    RET

E691  CDC1E6       FILL5:   CALL   FLUSH          ;Flush host buffer

E694  3A1BEF       FILL6:   LD     A,SEKDSK       ;Move disk
E697  321FEF                STO    A,HSTDSK
E69A  3217EF                STO    A,ACTDSK
E69D  2A19EF                LD     HL,SEKTRK
E6A0  221DEF                STO    HL,HSTTRK
E6A3  2215EF                STO    HL,ACTTRK
E6A6  3A18EF                LD     A,SEKSEC
E6A9  321CEF                STO    A,HSTSEC
E6AC  3214EF                STO    A,ACTSEC
E6AF  3A5CE9                LD     A,SEKTYP
E6B2  325DE9                STO    A,HSTTYP
E6B5  32D0EF                STO    A,ACTTYP
E6B8  3A21EF                LD     A,RDFLAG
E6BB  B7                    ORA    A
E6BC  C0                    RNZ                   ;If no read required

E6BD  3E00                  LDK    A,0            ;Read
E6BF  1822 ^E6E3$           JR     FINAL
```

Copyright © 1982 Osborne Computer Corporation

```
E6C1                FLUSH:
                    ;       Writes out active host buffer onto disk.

E6C1                        proc
E6C1  2151EF                LDK     HL,HSTWRT
E6C4  7E                    LD      A,[hl]
E6C5  B7                    ORA     A
E6C6  C8                    RZ                      ;If host buffer already on disk
E6C7  3600                  STO     0,[hl]
E6C9  3A1FEF                LD      A,HSTDSK        ;Move disk
E6CC  3217EF                STO     A,ACTDSK
E6CF  2A1DEF                LD      HL,HSTTRK
E6D2  2215EF                STO     HL,ACTTRK
E6D5  3A1CEF                LD      A,HSTSEC
E6D8  3214EF                STO     A,ACTSEC
E6DB  3A5DE9                LD      A,HSTTYP
E6DE  3200EF                STO     A,ACTTYP
E6E1  3E03                  LDK     A,3             ;Write flag
                    ;       JMP     FINAL
```

Copyright ©1982 Osborne Computer Corporation

```
E6E3                    FINAL:
                        ;       Preforms final transfer processing.
                        ;
                        ;       ENTRY   A = 0 .. read disk.
                        ;                 = 3 .. write disk.
                        ;       Calls: Rom resident routine to read/write ONE
                        ;               sector only.

E6E3  5F                        MOV     E,A
E6E4  1600                      LDK     D,0
E6E6  212701                    LDK     HL,ROMVEC+3*13
E6E9  19                        ADD     HL,DE
E6EA  220AE9                    STO     HL,SAVADR

E6ED  2114EF                    LDK     HL,ACTSEC
E6F0  34                        INC     [hl]            ;update sector+1

E6F1  2A0AE9                    LD      HL,SAVADR
E6F4  EB                        EX      DE,HL
E6F5  0601                      LDK     B,1             ;indicate one sector xfer
E6F7  CD19E3                    CALL    ROMJMP          ;process read or write
E6FA  3222EF                    STO     A,ERFLAG        ;set possible error flag
E6FD  C8                        RZ                      ;If no errors

E6FE  2122EF                    LDK     HL,ERFLAG
E701  7E                        LD      A,[HL]
E702  F601                      ORI     01h
E704  77                        STO     A,[HL]  ;Set error flag
E705  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
E706                MVINFO:
                    ;       Move information necessary for transfer.

E706  AF                    XRA     A
E707  3222EF                STO     A,ERFLAG                ;Clear error flag
E70A  3A20EF                LD      A,TEMSEC
E70D  3218EF                STO     A,SEKSEC
E710  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
E711                PRINT:
                    ;       Print message string to CONSOLE device

E711  2138E7                LDK     HL,CONOUT
E714  1803 ^E719$           JR      STROUT



E716                PSTR:
                    ;       Print message string to LIST device

E716  213FE7                LDK     HL,LIST
                    ;       JR      STROUT



E719                STROUT:
                    ;       Print message terminated by zero byte.
                    ;
                    ;       ENTRY   DE -> message buffer, first byte = length
                    ;
                    ;       EXIT    DE -> DE + length
                    ;               A = 0.
                    ;               Z bit set.
                    ;
                    ;       USES A, BC, DE, HL

E719                        PROC
E719  1A                    LD      A,[DE]          ;Get a length of print string
E71A  B7                    ORA     A
E71B  C8                    RZ                      ;If zero then terminate

E71C  47                    MOV     B,A             ;Length to B reg
E71D  13            NEXTC:  INC     DE
E71E  1A                    LD      A,[DE]          ;Get character
E71F  4F                    MOV     C,A

E720  D5                    PUSH    DE              ;Save print string address
E721  C5                    PUSH    BC              ;Save loop counter
E722  E5                    PUSH    HL              ;Save output routine address

E723  1128E7                LDK     DE,NEXT
E726  D5                    PUSH    DE              ;Return address to stack
E727  E9                    JMP     [HL]            ;Output

E728  E1            NEXT:   POP     HL
E729  C1                    POP     BC
E72A  D1                    POP     DE

E72B  10F0 ^E71D$           DJNZ    NEXTC           ;Print next character if not done

E72D  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
                        ; The following routines will use the IOBYTE to transfer
                        ; control to the appropriate device driver

E72E                    CONST:
                        ;       Returns console status

E72E                            proc
E72E  2179E7                    LDK     HL,PTR_CSTAT      ;Status table
E731  1808 ^E733$               JR      GODISPCH          ;Call appropriate rtn



E733                    CONIN:
                        ;       Reads input character from device

E733                            proc
E733  2181E7                    LDK     HL,PTR_CINP       ;Table of input rtns
E736  1803 ^E73B$               JR      GODISPCH



E738                    CONOUT:
                        ;       Puts output character to device
                        ;       C contains output character

E738                            proc
E738  2189E7                    LDK     HL,PTR_COUT       ;Table of output rtns



E73B                    GODISPCH:
E73B  0601                      LDK     B,1               ;number of shifts required to align
                                                          ;CONSOLE field
E73D  1829 ^E768$               JR      DISPCH
```

Copyright © 1982 Osborne Computer Corporation

```
E73F                LIST:
                    ;       List device character output

E73F                        proc
E73F  213AE1                LDK     HL,PINTFG       ;Get printer initialiation flag
E742  7E                    LD      A,[HL]
E743  B7                    ORA     A
E744  2008 ^E74E$           JRNZ    LIST1           ;Printer previously initialized

E746  35                    DEC     [HL]            ;Non-zero value to PINTFG
E747  C5                    PUSH    BC              ;Save character
E748  23                    INC     HL
E749  EB                    EX      HL,DE           ;Get initialization string
E74A  CD16E7                CALL    PSTR            ;and print it
E74D  C1                    POP     BC              ;Restore character

E74E  0604          LIST1:  LDK     B,4
E750  2199E7                LDK     HL,PTR_LIST     ;table of list routines
E753  1813 ^E768$           JR      DISPCH



E755                PUNCH:
                    ;       Output to punch

E755                        proc
E755  0606                  LDK     B,6
E757  2189E7                LDK     HL,PTR_PNCH     ;Punch routines
E75A  180C ^E768$           JR      DISPCH



E75C                READER:
                    ;       Reader input

E75C                        proc
E75C  0608                  LDK     B,8
E75E  2181E7                LDK     HL,PTR_RDR      ; reader routines
E761  1805 ^E768$           JR      DISPCH



E763                LISTST:
                    ;       Return the ready status for the list device.
                    ;
                    ;       EXIT    A = 0 (zero), list device is not ready to
                    ;               accept another character.
                    ;               A = FFh (255), list device is ready to accept
                    ;               a character.

E763                        proc
E763  0604                  LDK     B,4             ;number of left shifts thru carry
                                                    ;to align LIST field of IOBYTE
E765  2191E7                LDK     HL,PTR_LST      ;list status routines
                    ;       JR      DISPCH
```

Copyright © 1982 Osborne Computer Corporation

```
E768                DISPCH:
                    ;       on entry here reg B contains the left shift count
                    ;       required to align the iobyte field to bit 1 position.
                    ;       and reg HL contains address of select table

E768                        proc
E768  3A0300                LD      A,IOBYTE

E76B                DSPCH1:
E76B  17                    RAL
E76C  10FD ^E76B$           DJNZ    DSPCH1
E76E  E606                  ANI     6               ;get select field*2
E770  5F                    MOV     E,A
E771  1600                  LDK     D,0             ;DE = iobyte field * 2
E773  19                    DAD     DE
E774  5E                    MOV     E,[HL]
E775  23                    INC     HL
E776  66                    MOV     H,[HL]          ;get the routine address
E777  6B                    MOV     L,E             ;into hl and xchange with pc
E778  E9                    JMP     [HL]
```

Copyright ©1982 Osborne Computer Corporation

```
                      ;       Dispatch Table

E779                  PTR_CSTAT:
E779  A1E7                    DW      CNST            ; keyboard status
E77B  69E8                    DW      SISTAT          ; serial port input status
E77D  CBE8                    DW      PISTAT          ; parallel input status
E77F  B7E8                    DW      IEINSTAT        ; status of input device on IEEE port

E781                  PTR_RDR:
E781                  PTR_CINP:
E781  AFE7                    DW      KEYINP          ; get input from keyboard
E783  72E8                    DW      SPINP           ; serial port input
E785  CFE8                    DW      PARINP          ; parallel input
E787  BFE8                    DW      IEINP           ; ieee port input

E789                  PTR_PNCH:
E789                  PTR_COUT:
E789  4CE8                    DW      CRTOUT          ; output character to crt
E78B  76E8                    DW      SPOUT           ; serial port output
E78D  D3E8                    DW      PAROUT          ; parallel output
E78F  C3E8                    DW      IEOUT           ; ieee port output

E791                  PTR_LST:
E791  49E8                    DW      CRSTAT
E793  65E8                    DW      SOSTAT          ; serial output status
E795  C7E8                    DW      POSTAT          ; parallel output status
E797  BBE8                    DW      IEOSTAT         ; ieee output status

E799                  PTR_LIST:
E799  4CE8                    DW      CRTOUT
E79B  7AE8                    DW      PRTOUT
E79D  D3E8                    DW      PAROUT
E79F  C3E8                    DW      IEOUT
```

Copyright © 1982 Osborne Computer Corporation

```
E7A1                CNST:
                    ;       C O N S O L   S T A T U S
                    ;
                    ;       This routine samples the Console status and returns the
                    ;       following values in the A register.
                    ;
                    ;       EXIT    A = 0 (zero), means no character
                    ;               currently ready to read.
                    ;
                    ;               A = FFh (255), means character
                    ;               currently ready to read.

                    ;       check if any translated keys are pending

E7A1                        proc
E7A1  3AC1E7                LD      A,COUNT
E7A4  B7                    ORA     A
E7A5  2005 ^E7AC$           JRNZ    CNST5

                    ;       if no xlated keys pending, check keyhit flag

E7A7  3A5EEF                LD      A,LKEY          ;Get Key hit flag
E7AA  B7                    ORA     A
E7AB  C8                    RZ                      ;If data not available

E7AC                CNST5:
E7AC  F6FF                  ORI     OFFh
E7AE  C9                    RET
```

Copyright © 1982 Osborne Computer Corporation

```
E7AF                    KEYINP:
                        ;       Gets keystroke from ROM keyboard driver.
                        ;       Translates the codes 80h to 8fh as per table.
                        ;
                        ;       EXIT    A = translated code in ASCII

     = 0080             BASVLO: =       80h                     ;lowest value of translatable keys
E7AF                            proc
E7AF  DDE5       $              PUSH    IX                      ;Save user IX

E7B1  CDF6E7            KINO:   CALL    AHSCRL
E7B4  21C1E7                    LDK     HL,COUNT
E7B7  7E                        LD      A,[HL]                  ;get number of xlated keys
E7B8  B7                        ORA     A
E7B9  280B ^E7C6$               JRZ     KIN10                   ;if keys pending then
E7BB  DD2A5EE9   $              LD      IX,XLTKEY               ;get base address in IX reg

E7BF  DD7E00     $              LD      A,[IX+0]                ;simulate LD  A,(IX+COUNT)
     = E7C1             COUNT:  =       *-1                     ;to get next key from table

E7C2  34                        INC     [HL]                    ;increment count

E7C3  DDE1       $ KRET:        POP     IX                      ;Restore user IX
E7C5  C9                        RET

E7C6                    KIN10:
E7C6  1E09                      LDK     E,09
E7C8  CD17E3                    CALL    ROMCO1

                        ;       When console has returned this code will check
                        ;       for function key and preform some translation

E7CB  FE80                      CPI     80h             ;function keys have value
E7CD  38F4 ^E7C3$               JRC     KRET            ;80h-8dh
E7CF  FE8E                      CPI     8Eh             ;do a shift to make pointer
E7D1  30F0 ^E7C3$               JRNC    KRET            ;into table and return if not function key
E7D3  CB27       $              SLA     A
E7D5  5F                        MOV     E,A
E7D6  1600                      LDK     D,0
E7D8  DD216BE1   $              LDK     IX,XLTBL
E7DC  DD19       $              ADD     IX,DE
E7DE  DD6E00     $              LD      L,[IX+0]
E7E1  DD6601     $              LD      H,[IX+1]
E7E4  DD5E02     $              LD      E,[IX+2]
E7E7  DD5603     $              LD      D,[IX+3]
E7EA  ED535EE9   $              STO     DE,XLTKEY
E7EE  ED52       $              SBC     HL,DE
E7F0  7D                        MOV     A,L
E7F1  32C1E7                    STO     A,COUNT
E7F4  18BB ^E7B1$               JR      KINO
```

Copyright ©1982 Osborne Computer Corporation

```
E7F6                AHSCRL:
                    ;       ahscrl - does auto horizontal scroll if required.

E7F6                        proc
E7F6  3A63E1                LD      A,AHSENB
E7F9  37                    OR      A
E7FA  C8                    RZ

E7FB  2A5AEF                LD      HL,CURS         ;get cursor
E7FE  29                    ADD     HL,HL

                    ;       check for cursor in home window

E7FF  3E64                  LDK     A,100
E801  BD                    CMP     L
E802  3809 ^E80D$           JRC     RHC             ;jump if cursor not in home window
E804  3A61EF                LD      A,PIAAD         ;check for screen at home
E807  D6EA                  SUB     VFLO
E809  C8                    RZ                      ;screen at home
E80A  AF                    XRA     A               ;home screen
E80B  1818 ^E825$           JR      SCRL

E80D                RHC:
                    ;       check right-hand margin

E80D  3A61EF                LD      A,PIAAD         ;a=horizontal screen position
E810  D6EA                  SUB     VFLO
E812  C664                  ADD     A,100           ;window size*2 (50)
E814  BD                    CMP     L
E815  DA22E8                JC      :30             ;move screen when cursor about to go off
                                                    ;the right hand margin
                    ;       check left hand margin

E818  D65A                  SUB     90
E81A  BD                    CMP     L               ;check left margin
E81B  D8                    RC                      ;cursor in window return
E81C  7D                    MOV     A,L
E81D  D60A                  SUB     10
E81F  C8                    RZ                      ;return if cusor at column 2
E820  1803 ^E825$           JR      SCRL

E822  7D            :30:    MOV     A,L
E823  D664                  SUB     100

E825  1F            SCRL:   RAR
E826  C620                  ADD     A,' '
E828  2148E8                LK      HL,ESCSQ+3
E82B  77                    STO     A,[HL]
E82C  3A62EF                LD      A,PIABD
E82F  E61F                  AND     1Fh
E831  C620                  ADD     A,' '
E833  2B                    DEC     HL
E834  77                    STO     A,[HL]                  ;escsq+2 = vert. coords
E835  2B                    DEC     HL
E836  2B                    DEC     HL                      ;point to start of esc seq
E837  0604                  LK      B,4

E839  C5            :50:    PUSH    BC
E83A  E5                    PUSH    HL
```

```
E83B  4E                    LD      C,[HL]
E83C  CD60E8                CALL    CRT10
E83F  E1                    POP     HL
E840  C1                    POP     BC
E841  23                    INC     HL
E842  10F5 ^E839$           DJNZ    :50
E844  C9                    RET

E845  1B          ESCSQ:    DB      ESC                 ;set screen coord escape sequence
E846  53                    DB      'S'
E847  00                    DB      0                   ;** y coord
E848  00                    DB      0                   ;** x coord
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
E849                    CRSTAT:
                        ;       Returns status of crt.
                        ;       crt is always ready

E849                            proc
E849  F6FF                      ORI     0FFh
E84B  C9                        RET



E84C                    CRTOUT:
                        ;       ENTRY    C = output character

      = 0008            EF_ESC: =       8               ;escape flag bit definitions
      = 0001            EF_GR:  =       1

E84C                            proc
E84C  3A60EF                    LD      A,ESCH
E84F  E609                      AND     EF_ESC+EF_GR
E851  200D ^E860$               JRNZ    CRT10
E853  79                        MOV     A,C
E854  FE14                      CPI     14h
E856  2008 ^E860$               JRNZ    CRT10
E858  3A68E1                    LD      A,AHSENB
E85B  2F                        CMA
E85C  3268E1                    STD     A,AHSENB
E85F  C9                        RET

E860                    CRT10:
E860  1E0C                      LDK     E,0Ch           ;output to crt
E862  C317E3                    JMP     ROMCO1
```

Copyright © 1982 Osborne Computer Corporation

```
E865                    SOSTAT:
                        ;       Gets status of output device attached to serial port

E865                            proc
E865  1E2D                      LDK     E,2Dh
E867  1847 ^E830$               JR      JMPROM          ;Call rom driver



E869                    SISTAT:
                        ;       Gets status of input device attached to serial port

E869                            proc
E869  CDB3E8                    CALL    ACISTAT
E86C  E601                      ANI     SI.RRDY
E86E  C8                        RZ                      ;return with not ready status
E86F  F6FF                      ORI     TRUE
E871  C9                        RET



E872                    SPINP:
                        ;       Inputs a character from the serial port

E872                            proc
E872  1E15                      LDK     E,15h
E874  183A ^E830$               JR      JMPROM



E876                    SPOUT:
                        ;       Outputs character in reg c to the serial port (list device)

E876                            proc
E876  1E0F                      LDK     E,0Fh
E878  1836 ^E830$               JR      JMPROM
```

Copyright © 1982 Osborne Computer Corporation

```
E87A                  PRTOUT:
                      ;        routine does LIST output and printer protocols

     = 0011           XON:     =       11h
     = 0013           XOFF:    =       13h
     = 0003           ETX:     =       3
     = 0006           ACK:     =       6

E87A                           proc
E87A  CD76E8                   CALL    SPOUT
E87D  3A67E1                   LD      A,PRNTER
E880  B7                       ORA     A
E881  C8                       RZ
E882  E602                     ANI     2
E884  2013 ^E899$              JRNZ    XONXOF
E886  3E0D                     LDK     A,0Dh
E888  B9                       CMP     C
E889  C0                       RNZ

E88A  0E03                     LDK     C,ETX
E88C  CD76E8                   CALL    SPOUT

E88F                  PRT10:
E88F  CD72E8                   CALL    SPINP
E892  E67F                     ANI     7Fh            ;mask out parity bit
E894  FE06                     CPI     ACK
E896  20F7 ^E88F$              JRNZ    PRT10
E898  C9                       RET

E899                  XONXOF:
E899  CD69E8                   CALL    SISTAT
E89C  B7                       ORA     A
E89D  C8                       RZ
E89E  CD72E8                   CALL    SPINP
E8A1  E67F                     ANI     7Fh            ;mask out parity bit
E8A3  FE13                     CPI     XOFF
E8A5  C0                       RNZ

E8A6                  PRT20:
E8A6  CD72E8                   CALL    SPINP
E8A9  E67F                     ANI     7Fh            ;mask out parity bit
E8AB  FE11                     CPI     XON
E8AD  20F7 ^E8A6$              JRNZ    PRT20
E8AF  C9                       RET
```

Copyright © 1982 Osborne Computer Corporation

```
E880  C317E3         JMPROM: JMP     ROMCD1



      = E13C         ACICTL: =       SBAUD
                     ;       Outputs character in c to the ACIA CTL port.



E883                 ACISTAT:
                     ;       Returns usart status in A

E883                         PROC
E883  1E81                   LDK     E,81H          ;Low order byte of ACISTAT routine in ROM
E885  18F9 ^E880$            JR      JMPROM
```

Copyright © 1982 Osborne Computer Corporation

```
E8B7                    IEINSTAT:
                        ;       Returns status of IEEE input port
                        ;
                        ;       EXIT    A = 0   character not available
                        ;               A = FFH character available

E8B7                            proc
E8B7  1E87                      LDK     E,87H
E8B9  18F5 ^E8B0$               JR      JMPROM



E8B3                    IEOSTAT:
                        ;       Returns status of IEEE output port
                        ;
                        ;       EXIT    A = 0   transmitter not ready
                        ;               A = FFH transmitter ready

E8BB                            proc
E8BB  1E8A                      LDK     E,8AH
E8BD  18F1 ^E8B0$               JR      JMPROM



E8BF                    IEINP:
                        ;       Reads a character from IEEE port

E8BF  1E8D                      LDK     E,8DH
E8C1  18ED ^E8B0$               JR      JMPROM



E8C3                    IEOUT:
                        ;       Outputs the character in reg C to IEEE port

E8C3                            proc
E8C3  1E90                      LDK     E,90H
E8C5  18E9 ^E8B0$               JR      JMPROM
```

Copyright © 1982 Osborne Computer Corporation

```
E8C7                    POSTAT:
                        ;       Gets status of the parallel (centronix) printer
                        ;       attached to the ieee port

E8C7                            proc
E8C7  1E96                      LDK     E,96H
E8C9  18E5 ^E880$               JR      JMPROM



E8CB                    PISTAT:
                        ;       Gets status of the input device attached to the
                        ;       parallel port

E8CB                            proc
E8CB  1E93                      LDK     E,93H
E8CD  18E1 ^E880$               JR      JMPROM



E8CF                    PARINP:
                        ;       Inputs a character from parallel port.

E8CF                            proc
E8CF  1E99                      LDK     E,99H
E8D1  18DD ^E880$               JR      JMPROM



E8D3                    PAROUT:
                        ;       Outputs the character in c to the IEEE port treating the
                        ;       port as a parallel port.

E8D3                            proc
E8D3  1E9C                      LDK     E,9CH
E8D5  18D9 ^E880$               JR      JMPROM
```

```
E8D7                    SW2ROM:
                        ;       Switches to rom
                        ;       saves all registers

E8D7                            PROC
E8D7  F3                        DI
E8D8  F5                        PUSH    AF
E8D9  AF                        XRA     A
E8DA  D300                      OUT     0
E8DC  3208EF                    STO     A,ROMRAM
E8DF  F1                        POP     AF
E8E0  FB                        EI
E8E1  C9                        RET



E8E2                    SW2RAM:
                        ;       Switches to ram
                        ;       preserves all registers

E8E2                            PROC
E8E2  F3                        DI
E8E3  F5                        PUSH    AF
E8E4  3E01                      LDK     A,1
E8E6  D301                      OUT     1
E8E8  3208EF                    STO     A,ROMRAM
E8EB  F1                        POP     AF
E8EC  FB                        EI
E8ED  C9                        RET
```

Copyright © 1982 Osborne Computer Corporation

```
      = E8EE      XXX:    =       *
      = 0000              IF      XXX > CCP + 1E00H
  -               .9      ERROR   SYSTEM SIZE TOO LARGE
                          ENDIF

                          MSG     'SYSTEM SPACE AVAILABLE = ',1E00H - (XXX - CCP)
      'SYSTEM SPACE AVAILABLE = 0012'



E8EE  = E900              ORG     CCP + 1E00H

E900  = 0004      LASTA:  DS      4               ;used by occpio4?, # drives * 2
E904  = 0002      DPB:    DS      2
E906  = 0002      SAVDPH: DS      2
E908  = 0001      TEMDSK: DS      1
E909  = 0001      TEMTYP: DS      1

E90A  = 0002      SAVADR: DS      2

E90C  = 0030      ALV:    DS      ALVSZ
E93C  = 0020      CSV:    DS      CSVSZ

E95C  = 0001      SEKTYP: DS      1
E95D  = 0001      HSTTYP: DS      1

E95E  = 0002      XLTKEY: DS      2



      = E960      YYY:    =       *
      = 0000              IF      YYY > MRAM
  -               .9      ERROR   CODE HIT BMRAM
                          ENDIF
                  ;OCCRAM2.ASM
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
                        ;       Used to assembly ROM resident and CBIOS

E960  = EA80                    ORG     MRAM

                        ;       Host disk xfer buffer and...
                        ;       Format track template holding buffer
EA80                    HSTBUF:
EA80  = 0480                    DS      1024+128

                        ;       Directory Buffer
      = EE80            DIRBUF: =       HSTBUF+1024


EF00  = 0006            TEM     DS      6
      = EF01            RNDV    =       TEM+1   ;random number seed
      = EF02            ERCNT   =       RNDV+1  ;DW ERCNT
      = EF04            RTRC    =       ERCNT+2 ;retry count
      = EF05            RTRY    =       RTRC+1
EF06  = 0001            MPCHR   DS      1       ;prompt character
EF07  = 0001            ECHOP   DS      1       ;=0, list ehco off
EF08  = 0001            ROMRAM  DS      1       ;0= RAM, 1= ROM
EF09  = 0006            DSTS3   DS      6       ;Disk status bytes

                        ;       Disk operation temps and control
EF0F  = 0002            DMADR   DS      2       ;Address for read/write Disk
EF11  = 0002            DMAADR  DS      2       ;CBIOS, users DMA

                        ;       Note order of xxxSEC,xxxTRK,xxxDSK must be maintained
                        ;       along with length (1,2,1).
EF13  = 0001            SEKDEL: DS      1       ;Set for seek-restore command in ROM
                                                ;depends on disk type. Siemens = 3h, MPI = 0h
EF14  = 0001            SAVSEC  DS      1       ;last sector requested
EF15  = 0002            SAVTRK  DS      2       ;last track requested
EF17  = 0001            SDISK   DS      1       ;Selected disk drive (0,1)
                        ;SAVTYP =       0EFD0H  ;SELECTED TYPE (sector size)

      = EF14            ACTSEC  =       SAVSEC
      = EF15            ACTTRK  =       SAVTRK
      = EF17            ACTDSK  =       SDISK

EF18  = 0001            SEKSEC  DS      1
EF19  = 0002            SEKTRK  DS      2
EF1B  = 0001            SEKDSK  DS      1

EF1C  = 0001            HSTSEC  DS      1
EF1D  = 0002            HSTTRK  DS      2
EF1F  = 0001            HSTDSK  DS      1

EF20  = 0001            TEMSEC  DS      1       ;Used in bios only
EF21  = 0001            RDFLAG  DS      1       ;Read flag
EF22  = 0001            ERFLAG  DS      1       ;Error reporting
EF23  = 0001            WRTYPE  DS      1       ;Write operation type

                        ;ALV:    DS      ALVS
                        ;CSV:    DS      CSVS

EF24  = 000C                    DS      ALVS
EF30  = 0020                    DS      CSVS

                        ;       BIOS blocking-deblocking flags
```

Copyright © 1982 Osborne Computer Corporation

```
EF50  = 0001        HSTACT: DS      1       ;host active flag
EF51  = 0001        HSTWRT: DS      1       ;Host written flag
EF52  = 0001        UNACNT: DS      1       ;Unalloc rec count
EF53  = 0002        UNATRK: DS      2       ;Track
EF55  = 0001        UNASEC: DS      1       ;Sector
EF56  = 0001        LOGSEC: DS      1       ;Logical sector


EF57  = 0002        LDADR   DS      2
EF59  = 0001        KEYLCK  DS      1       ;Zero if locked keyboard
EF5A  = 0002        CURS    DS      2       ;current cursor position

                    ;       Keyboard scan temporaries
EF5C  = 0001        TKEY    DS      1       ;Tem holding key
EF5D  = 0001        HKCNT   DS      1       ;Debounce key
EF5E  = 0001        LKEY    DS      1       ;Last valid keystroke
EF5F  = 0001        CKEY    DS      1       ;Last control key
EF60  = 0001        ESCH    DS      1       ;ESC holding flag

                    ;PIAAD and PIABD must be kept sequential, PIAAD first
                    ;dependency in VC_HOME of BMKEY.asm
EF61  = 0001        PIAAD:  DS      1       ;Holds last PIA-A data
EF62  = 0001        PIABD:  DS      1       ;Holds last PIA-B data

                    ;       Calendar month, day year
EF63  = 0003        IDAY    DS      3
      = EF64        IMONTH  =       IDAY+1
      = EF65        IYR     =       IDAY+2

                    ;       Wall clock time cells and disk active
                    ;       see UPTIM: in BMKEY.asm
EF66  = 0006        HOURS:  DS      6
      = EF67        MINS:   =       HOURS+1
      = EF68        SECS:   =       HOURS+2
      = EF69        SEC6:   =       HOURS+3

                    ;       Used to deselect drive when there is NO activity
                    ;       on drive for n seconds.  See FDSK routine
      = EF6A        DACTVE: =       HOURS+4 ;=0 by FDSK, Used by UPTIM

      = EF6B        BELCNT: =       HOURS+5 ;^G bell timer cell


EF6C  = 0001        LLIMIT  DS      1       ;max #columns in a logical line
                    ;       MSG     'LLIMIT = ',LLIMIT,'h.'

                    ;       Disk drive current positions
EF6D  = 0002        LDSEL:  DS      2       ;Last selected drive
      = EF6E        LDTRK   =       LDSEL+1 ;Last track used for non-selected drive

EF6F  = 0002        IESTK:  DS      2       ;save current stk ptr


                    ;       Interrupt stack
EF71  = 0028                DS      20*2
EF99  = 0000        ISTK:   DS      0

                    ;       Stack entry
EF99  = 0028                DS      20*2
EFC1                BIOSTK:
```

```
EFC1   = 0000      ROMSTK: DS      0

EFC1   = 0001      ACIAD:  DS      1       ;last command byte written to ACIA

EFC2   = 0004      R179x:  DS      4       ;179x register save area
EFC6   = 0001      KBDLY:  DS      1       ;keyboard debounce-delay cell

                   ;since CP/M CANNOT boot off B:, this cell is used
                   ;to invert the names of the 2 drives:
                   ;       =0, all normal, A=A:, B=B:
                   ;       =1, all inverted, A=B:, B=A:
EFC7   = 0001      DSKSWP  DS      1

EFC8                       ALIGN   10h
EFC8 += 0008               DS      (*+(10H)-1)/(10H)*(10H)-*

       = EFCC      SEQ:    =       *-4
EFD0               ACTTYP:
EFD0   = 0001      SAVTYP: DS      1
EFD1   = 0001      RDT_WRTS: DS    1
EFD2   = 0002      CCPADR: DS      2
EFD4   = 0006      KEYLST: DS      6

EFDA   = 0001      SERFLG: DS      1
EFDB   = 0001      IE_ADR: DS      1
EFDC   = 0001      IE_CHAR: DS     1

EFDD   = 0001      PIACTL: DS      1
EFDE   = 0001      PP.MODE: DS     1


                   ;       8080 Register Save Area.
EFDF                       ALIGN   10h
EFDF += 0001               DS      (*+(10H)-1)/(10H)*(10H)-*
EFE0               REGS:
EFE0   = 0001      ESAVE:  DS      1       ;E Register save location
EFE1   = 0001      DSAVE:  DS      1       ;D Register save location
EFE2   = 0001      CSAVE:  DS      1       ;C Register save location
EFE3   = 0001      BSAVE:  DS      1       ;B Register save location
EFE4   = 0001      FSAVE:  DS      1       ;FLAGS save location
EFE5   = 0001      ASAVE:  DS      1       ;A Register save location
EFE6   = 0001      LSAVE:  DS      1       ;L Register save location
EFE7   = 0001      HSAVE:  DS      1       ;H Register save location
EFE8   = 0002      PSAVE:  DS      2       ;PGM COUNTER save location
EFEA   = 0002      SSAVE:  DS      2       ;USER STACK pointer save location

EFEC   = 0002      BKPA:   DS      2       ;last breakpoint address
EFEE   = 0001      BKPC:   DS      1       ;Contents of bkp

EFEF   = 0001      VRTOFF  DS      1       ;LAST VERTICAL OFFSET TAKEN FROM COUT
                   ;
                   ;
                   ;       Interrupt Jump Vector is between EFF8, EFFF.
                   ;       Endx    MRAM


no ERRORS,  523 Labels, 67C3h bytes not used.     Program LWA = EFF0h.
```

Copyright © 1982 Osborne Computer Corporation

Copyright © 1982 Osborne Computer Corporation

```
s  @FREQ   0FA0
 n ACIAD   EFC1     50# 3
   ACICTL  E13C      4/49   43# 7
   ACISTA  E883      4/50   41/16   43#13
   ACK     0006     42# 8   42/27
   ACMD    E19C      5#49   22/20
   ACTDSK  EF17     27/ 7   28/12   48#39
   ACTSEC  EF14     27/13   28/16   29/17   48#37
   ACTTRK  EF15     27/10   28/14   48#38
   ACTTYP  EF00     27/16   28/13   50#18
   AHSCRL  E7F6     37/12   38# 2
   AHSENB  E168      5#18   38/ 6   40/26   40/28
s  ALIGN   mac      50/15   50/35
   ALV     E90C      7/12    7/24   47#23
s  ALVS    000C     48/57
   ALVSZ   0030      3#13    7/12    7/15    7#15    7/24    7/27    7#27
                    47/23
 n ASAVE   EFE5     50#42
 n BASVLO  0080     37# 8
   BCCP    E521     21/45   21#56
 n BCPM    E519     21#51
   BDOS    D306      2# 4   22/15
 n BELCNT  EF6B     49#41
   BIOS    E100      2# 3    2/ 7    4/ 9    5/20   22/13
   BIOSTK  EFC1     11/31   49#60
 n BKPA    EFEC     50#48
 n BKPC    EFEE     50#49
   BRATE   E169      5#20   21/31
 n BSAVE   EFE3     50#40
   CAUTO   E19D      5#53    5/55   22/18
   CAUTOL  0007      5/53    5#55
   CBOOT   E4ED      4/11   21#23
   CCP     CB00      2# 2    2/ 3    2/ 4   21/24   21/48   22/28   22/54
                    47/ 3   47/ 8   47/13
 n CCPADR  EFD2     50#21
s  CDISK   0004     14/17   14/20   21/42   22/56
   CHKSEL  E3FA     13/29   15# 2
   CHKSEQ  E408     15/18   15#22
 n CKEY    EF5F     49#17
   CNST    E7A1     35/ 5   36# 2
   CNST5   E7AC     36/19   36#27
   CNTRL0  E1A5      5/26    5#56
   CNTRL1  E1A6      5/27    6# 2
   CNTRL2  E1A7      5/28    6# 3
   CNTRL3  E1A8      5/29    6# 4
   CNTRL4  E1A9      5/30    6# 5
   CNTRL5  E1AA      5/31    6# 6
   CNTRL6  E1AB      5/32    6# 7
   CNTRL7  E1AC      5/33    6# 8
   CNTRL8  E1AD      5/34    6# 9
   CNTRL9  E1AE      5/35    6#10
   CONIN   E733      4/14   32#16
   CONOUT  E738      4/15   31/ 5   32#26
   CONST   E72E      4/13   32# 6
   COUNT   E7C1     36/17   37/13   37#20   37/50
s  CR      000D     14/24   14/29   22/39   22/42   22/42
   CRSTAT  E849     35/25   40# 2
   CRT10   E860     39/ 2   40/22   40/25   40#31
   CRTOUT  E84C     35/19   35/31   40#13
 n CSAVE   EFE2     50#39
```

Copyright © 1982 Osborne Computer Corporation

```
   CSV     E93C    7/11    7/23    47#24
s  CSVS    0020    48/58
   CSVSZ   0020    3#14    7/11    7/14    7#14    7/23    7/26    7#26
                   47/24
   CUNACT  E4C9    19#22   20/12
   CURS    EF5A    38/10   49#11
   D1DSM   0039    2#28    7/15    7/27    9/11
n  DACTVE  EF6A    49#39
s  DBUF    0080    21/60
   DDXLTS  0000    7/ 7    7/19    8/ 4    8#11
   DEC     0008    2#25    9/44
   DECDSM  00AB    2#31    9/47
   DECXLT  E252    8/ 8    8#28
   DENERR  F332    14/ 7   14#23   14/26
   DENL    0025    14/23   14#26
   DIRBUF  EE80    7/ 9    7/21    48#11
n  DIROM   0001    2#45
   DISPCH  E768    32/39   33/19   33/30   33/41   34# 2
s  DMA     0080
   DMAADR  EF11    19/54   26/37   48#26
   DMADR   EF0F    22/ 3   48#25
   DONE    E59A    22/22   22/26   22#45
   DONE0   E5A5    22/47   22#52
   DONE1   E5A8    22/34   22#54
   DOWN    E1B1    5/38    6#13
   DPB     E904    16/40   16/60   47#16
   DPBASE  E200    7# 4    13/24   24/40
   DPBD1   E29E    7/10    7/22    9# 6
n  DPBDEC  E20E    9#42
   DPBGEN  mac     3#48    9/ 8    9/17    9/26    9/35    9/44
n  DPBIBM  E2CE    9#33
n  DPBS1   E2AE    9#15
   DPBSTA  E29E    9# 4    9/56    16/27
n  DPBX0   E2BE    9#24
n  DPBXTR  E2EE    9#51
   DPHGEN  mac     3#17    7/ 6    7/18
   DRV     E307    14/11   14#25
   DRV1    E3F9    14/12   14#30
n  DSAVE   EFE1    50#38
   DSKD1   000C    2#22    9/ 3
   DSKS1   0005    2#21    9/17
n  DSKSWP  EFC7    50#12
   DSPCH1  E76B    34#10   34/12
n  DSTSB   EF09    48#22
n  ECHOP   EF07    48#20
   EFESC   0008    40#15   40/21
   EFGR    0001    40#17   40/21
n  ENROM   0000    2#44
   EOTBL   E133    5/40    6#15
   ERCNT   EF02    48#16   48/17
   ERFLAG  EF22    23/18   25/14   25/21   29/24   29/27   30/ 6   48#51
n  ESAVE   EFE0    50#37
s  ESC     0013    39/ 9
   ESCH    EF50    22/ 7   40/20   49#18
   ESCSQ   E845    38/48   39# 9
   ETX     0003    42# 7   42/21
s  FALSE   0000
s  FCB     005C
   FILL    E644    23/13   25/ 6   26# 2
   FILL2   E65A    26#26   26/34
```

Copyright © 1982 Osborne Computer Corporation

```
   FILL3  E65F    26/28   26#31
   FILL35 E665    26/24   26#36
   FILL4  E688    26#55   26/60
   FILL5  E691    26/50   26/57   27# 3
   FILL6  E694    26/46   27# 5
   FINAL  E6E3    27/22   29# 2
   FLUSH  E6C1    12/ 8   25/20   27/ 3   28# 2
 n FMTJ   E139     4#32
   FORERR E308    14/ 4   14#28   14/31
   FORL   0021    14/23   14#31
   FPYDIB 0008     2#15    2/28    2/29    2/30    2/31
   FPYSIB 0010     2#14    2/27
 n FSAVE  EFE4    50#41
s  FWAVM  F000
   GDEN   E431    15/41   16# 2
   GETDEN E490    16/13   18# 2
   GDDISP E738    32/11   32/21   32#36
   GOROM  E33A    11/36   11#50
   GRET   E42C    15/34   15/39   15#44
s  H.IEEE 2900
s  H.SIO  2A00
 n HKCNT  EF5D    49#15
   HOME   E33F     4/19   12# 2   21/49
   HOURS  EF66    49#32   49/33   49/34   49/35   49/39   49/41
 n HSAVE  EFE7    50#44
   HSTACT EF50    12/10   23/23   26/42   48#61
   HSTBUF EA80    22/ 2   26/21   48# 7   48/11
   HSTDSK EF1F    27/ 5   28/11   48#47
   HSTSEC EF1C    26/51   27/12   28/15   48#45
 n HSTSIZ 0400     2#13
   HSTTRK EF1D    27/ 9   28/13   48#46
   HSTTYP E95D    26/47   27/15   28/17   47#27
   HSTWRT EF51    25/13   28/ 6   49# 2
   IBM    0008     2#24    9/35
   IBMDSM 009C     2#30    9/38
   IBMXLT 0000     8/ 7    8#26
   IDAY   EF63    49#26   49/27   49/28
   IEADR  EFDB    21/39   50#25
 n IEB1C  E13F     4#36
 n IEB2C  E142     4#37
 n IEB3C  E145     4#38
 n IEB4C  E148     4#39
 n IEB5C  E14B     4#40
 n IEB6C  E14E     4#41
 n IEB7C  E151     4#42
 n IEB8C  E154     4#43
 n IECHAR EFDC    50#26
   IEEEAD E189     5#42   21/33
   IEINP  E83F    35/15   44#28
   IEINST E8B7    35/ 8   44# 2
   IEOSTA E838    35/28   44#15
   IEOUT  E8C3    35/22   35/34   44#37
   IESTK  EF5F    10/14   49#51
 n IMONTH EF64    49#27
   IOBITE E166     5#11   21/28
s  IOBYTE 0003    21/29   34/ 8
 n ISTK   EF99    49#56
 n IYR    EF65    49#23
   JMPROM E830    41/ 7   41/30   41/40   43# 2   43/18   44/10   44/23
                  44/32   44/42   45/ 8   45/19   45/29   45/40
```

Copyright ©1982 Osborne Computer Corporation


```
n KBDLY   EFC6     50# 6
  KEYINP  E7AF     35/12    37# 2
n KEYLCK  EF59     49#10
n KEYLST  EFD4     50#22
  KIN0    E731     37#12    37/51
  KIN10   E7C6     37/16    37#27
  KRET    E7C3     37#24    37/35    37/37
  LASTA   E900     15/15    47#15
n LDADR   EF57     49# 9
  LDSEL   EF6D     49#48    49/49
n LDTRK   EF6E     49#49
  LEFT    E132      5/39     6#14
s LF      000A     14/24    14/29    22/39    22/42    22/42
  LIST    E73F      4/16    31/14    32#40
  LIST1   E74E     33/ 8    33#17
  LISTST  E763      4/26    33#45
  LKEY    EF5E     36/23    49#16
  LLIMIT  EF6C     21/36    49#44
  LOGSEC  EF56     20/14    24/30    49# 6
n LSAVE   EFE6     50#43
s LVMEM   1000
s LWAMEM  FFFF      2/ 9
n MINS    EF67     49#33
n MPCHR   EFD6     48#19
s MRAM    E480     47/37    48/ 3
  MSIZE   003B      1#50    22/40    22/40
  MVINFO  E706     23/10    24/16    30# 2
  NDSK    0002      3#11     7/ 6     7# 6     7/18     7#18    13/14
  NEXT    E728     31/45    31#49
  NEXTC   E71D     31#37    31/53
  NOFDD   0002      3#12     7/13     7#13     7/25     7#25
  NUMDPB  0006      9#56    16/26
s NVDL    0018
  PARINP  E8CF     35/14    45#24
  PAROUT  E8D3     35/21    35/33    45#34
  PIAAD   EF51     38/13    38/27    49#22
  PIABD   EF62     38/50    49#23
n PIACTL  EFDD     50#28
n PINIT   E13B      5#46
  PINTFS  E18A      5#44    33/ 5
  PISTAT  E8C3     35/ 7    45#13
  POSTAT  E8C7     35/27    45# 2
n PP.MOD  EFDE     50#29
  PRINT   E711     14/13    22/50    31# 2
  PRNTER  E167      5#14    42/12
  PRT10   E88F     42#24    42/28
  PRT20   E8A6     42#40    42/44
  PRTOUT  E87A     35/32    42# 2
n PSAVE   EFE8     50#45
  PSTR    E716     31#11    33/14
  PTRCIN  E781     32/20    35#11
  PTRCOU  E789     32/31    35#18
  PTRCST  E779     32/10    35# 4
  PTRLIS  E799     33/18    35#30
  PTRLST  E791     33/57    35#24
  PTRPNC  E789     33/29    35#17
  PTRRDR  E781     33/40    35#10
  PUNCH   E755      4/17    33#24
n R179X   EFC2     50# 5
  RDFLAG  EF21     26/14    27/17    48#50
```

```
 n RDTWRT EFD1     50#20
   READ   E5B1      4/24   22#59
   READER E75C      4/18   33#35
 n REGS   EFE0     50#36
   RETURN E42D     15/42   15#46
   RHC    E80D     38/17   38#24
   RIGHT  E130      5/37    5#12
   RNDV   EFD1     48#15   48/16
   ROMCD1 E317     11#15   37/29   40/33   43/ 2
   ROMCDE E312      4/32    4/33    4/36    4/37    4/38    4/39    4/40
                    4/41    4/42    4/43    4/45    4/46    4/47   11# 2
   ROMJMP E319      4/31   11#18   18/32   21/52   29/23
   ROMRAM EFD8     10/ 5   46/11   46/28   48#21
   ROMRI  E2FE      4/30   10# 2
 n ROMSTK EFC1     49#61
 ; ROMVEC 0100     11/16   21/51   29/13
 n RRI    E133      4#30
   RTRC   EFD4     48#17   48/18
 n RTRY   EFD5     48#18
   S1DSM  002E      2#27    9/20
   SAVADR E9DA     29/15   29/20   47#21
   SAVDPH E9D6     13/26   13/45   16/51   47#17
   SAVSEC EF14     48#32   48/37
   SAVTRK EF15     48#33   48/38
   SAVTYP EFD0     18/25   18/37   18/41   50#19
   SBAUD  E13C      4#33   21/33   43/ 7
   SCRL   E825     38/22   38/41   38#46
   SCRSZE E16A      5#22   21/35
   SDISK  EF17     18/23   18/29   18/35   48#34   48/39
 n SEC6   EF69     49#35
 n SECS   EF68     49#34
   SECTRN E4D8      4/27   19#56
 n SEKDEL EF13     48#30
   SEKDSK EF1B     13/16   14/ 9   18/28   24/42   26/53   27/ 5   48#43
   SEKSEC EF18     26/22   26/36   26/52   26/53   27/11   30/ 8   48#41
   SEKTRK EF19     12/12   12/13   19/ 8   24/28   27/ 8   48#42
   SEKTYP E95C     13/39   24/21   26/16   26/48   27/14   47#26
   SELD   E3A4     13/15   14#15
   SELD1  E48A     16/14   17#14
   SELD2  E48D     16/37   17#17
   SELDSK E350      4/20   13# 2
   SELER  E38C     13/30   14# 4
   SELER1 E396     14/ 6   14# 9
   SENDEN D130     18#20   18/31
   SEQ    EFCC     13/41   15/24   50#17
 n SERFLG EFDA     50#24
   SETDMA E4D3      4/23   19#45   21/61
   SETSEC E4CE      4/22   19#32
   SETTRK E4BB      4/21   19# 2
s  SI.120 0055      5/20
s  SI.RRD 0001     41/17
s  SI.S16 0055
s  SI.S64 0056
   SIGNL  0038     22/36   22#43
   SIGNON E561     22#36   22/43   22/49
   SISTAT E859     35/ 6   41#12   42/32
   SOSTAT E865     35/26   41# 2
   SPINP  E872     35/13   41#25   42/25   42/35   42/41
   SPOUT  E876     35/20   41#35   42/11   42/22
 n SSAVE  EFEA     50#46
```

Copyright ©1982 Osborne Computer Corporation

Copyright ©1982 Osborne Computer Corporation

```
   STROUT E719     31/ 6    31#20
s  SVER   000E
   SW2RAM E8E2     11/37    21/26    46#19
   SW2ROM E8D7     11/35    46# 2
s  SYS    0005
s  SYSDAT 0010
s  SYSL   0006
   TEM    EF00     43#14    48/15
   TEMDSK E908     18/24    18/34    47#18
   TEMSEC EF20     19/39    30/ 7    48#49
   TEMTYP E909     18/26    18/40    47#19
 n TKEY   EF5C     49#14
 n TRAN   E4E8     20#22
s  TRUE   FFFF      5/18    41/19
   UNACNT EF52     12/11    19/26    23/12    24/27    24/34    49# 3
   UNASEC EF55     20/10    24/54    25/ 3    49# 5
   UNATRK EF53     19/ 9    24/29    24/58    24/60    49# 4
   UP     E1AF      5/36     6#11
   VERS   0016      1#33    22/41    22/41
s  VFLD   FFEA     38/19    38/28
s  VLDL   0034
 n VRTOFF EFEF     50#51
   WBOOT  E513      4/12    21#47
 n WRALL  0000      2#35
   WRDIR  0001      2#36    25/19
   WRIT1  E5E4     24/24    24#27
   WRIT2  E5F3     24/20    24#34
   WRIT25 E607     24#46    24/47
   WRIT3  E61E     24/32    24/57    25# 3
   WRIT4  E623     24/37    25# 6
   WRITE  E5CE      4/25    24# 2
   WRTYPE EF23     24/18    25/18    48#52
   WRUAL  0002      2#37    24/19
   XEROX  0001      2#23     9/26
   XLTBL  E158      5#25    37/41
   XLTKEY E95E     37/17    37/47    47#29
   XLTS   E22C      8/ 5     8#13
   XOFF   0013     42# 6    42/37
   XON    0011     42# 5    42/43
   XONXOF E899     42/16    42#31
   XTAB   E220      8# 2    16/29
   XTRXLT E276      8/ 9     8#32
   XXDSM  0053      2#29     9/29
   XXX    E8EE     47# 2    47/ 3    47/ 8
   XXXLTS E240      8/ 6     8#18
   YYY    E960     47#36    47/37
```