Osborne 1
Technical Manual

By Thom Hogan and Mike Iannamico
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
415-887-9080
Contents

1.0 Perspective 1

1.1 INTRODUCTION 1
1.2 POWER ON 2
1.3 VARIATIONS IN DESIGN 2

2.0 Logic Board And CPU 3

2.1 OSBORNE I MAIN LOGIC BOARD 3
2.2 BOARD REVISIONS 3
2.3 MAIN BOARD LAYOUT 4
2.4 CENTRAL PROCESSOR 4

3.0 Osborne 1 Memory Scheme 7

3.1 MEMORY BANKS 7
3.2 CP/M MEMORY ALLOCATION 7
3.3 MAIN MEMORY 9
3.4 MEMORY-MAPPED I/O 9
3.5 MEMORY ACCESS TIME 9

4.0 Osborne 1 Interface Design 13

4.1 IEEE-488 INTERFACE 13
4.1.1 IEEE-488 Signal Direction 13
4.1.2 IEEE-488 Pinouts 14
4.1.3 IEEE-488 Jump Vectors 15
4.1.4 IEEE-488 Communication Protocol 15
4.1.5 IEEE-488 As A Parallel Port 16

4.2 SERIAL RS232 INTERFACE 17
4.2.1 RS232 Signal Direction 18
4.2.2 RS232 Pinouts 18

4.3 MODEM 19
4.3.1 Modem Signal Direction 19
4.3.2 Modem Pin Connections 19
4.3.3 Modem Status 20

4.4 BAUD RATE 21
5.0 Video Attributes  23
  5.1 VIDEO DISPLAY  23
  5.2 VIDEO MEMORY  23
  5.3 CHARACTER GENERATION  24
  5.4 DISPLAY LOGIC  25
  5.5 BLOCK GRAPHICS  25
  5.6 VIDEO SIGNALS AND PINOUTS  27
    5.6.1 Video In-Line Connector  27
  5.7 VIDEO CONNECTIONS AND CIRCUITRY  28
  5.8 VIDEO TIMING  30

6.0 Power Specifications  31
  6.1 POWER SUPPLY  31
  6.2 SWITCHING VOLTAGE  31
  6.3 POWER BOARD LAYOUT  32

7.0 Disk Drives  33
  7.1 DRIVES USED  33
  7.2 DISK DRIVE INTERFACE BOARD  33
  7.3 DISK CONTROLLER AND PINOUTS  37
  7.4 CP/M BIOS DISK ADDRESSING  38
  7.5 DISK INTERFACE  38
  7.6 DISK DRIVE SPECIFICATIONS  38
  7.7 DISK FORMAT  39
    7.7.1 CP/M File Control Block  39

8.0 Keyboard  41
  8.1 THE KEYBOARD  41
  8.2 KEYBOARD PINOUTS  41
  8.3 KEYBOARD LAYOUT  42
  8.4 KEYSWITCH MATRIX  42
  8.5 PROGRAMMABLE KEYS  43
9.0 Assembly And Disassembly  45

9.1 VARIATIONS IN DESIGN  45

9.2 KEYBOARD  45
9.2.1 Keyboard Disassembly  45
9.2.2 Keyboard Assembly  46

9.3 ORIGINAL AND NEW BEZEL/CHASSIS  47
9.3.1 Original Bezel And Chassis Disassembly  47
9.3.2 Original Case Assembly  48
9.3.3 New Case Disassembly  49
9.3.4 New Case Assembly  50

9.4 LOGIC BOARD  52
9.4.1 Logic Board Disassembly  52
9.4.2 Logic Board Assembly  53

9.5 ORIGINAL AND NEW VIDEO MONITOR  53
9.5.1 Original Monitor Disassembly  53
9.5.2 Original Monitor Assembly  54
9.5.3 Blue Case Monitor Disassembly  54
9.5.4 Blue Case Monitor Assembly  55

9.6 POWER SUPPLY  55
9.6.1 Power Supply Disassembly  55
9.6.2 Power Supply Assembly  56

9.7 DISK DRIVE  57
9.7.1 Disk Drive Disassembly  57
9.7.2 Disk Drive Assembly  58

9.8 POWER PANEL  59
9.8.1 Power Panel Disassembly  59
9.8.2 Power Panel Assembly  61

10.0 Osborne 1 Software  63

10.1 SOFTWARE REVISIONS  63

10.2 THE MONITOR ROM  64
10.2.1 Console Routines  65
10.2.2 Other Interface Routines  66
10.2.3 ROM Listings  66

10.3 BDOS CALLS  66

10.4 OSBORNE 1 BIOS ROUTINES  68
10.4.1 The I0BYTE  68
10.4.2 BIOS Listings  69
11.0 Theory of Operations  71
  11.1 INTRODUCTION  71
  11.2 FUNCTIONAL OPERATION  71
  11.3 MNEMONIC CODES  72
  11.4 BASIC TIMING  73
  11.5 ROM  75
  11.6 RAM  77
    11.6.1 Address Lines  77
    11.6.2 RAM Data Output  77
    11.6.3 RAM Refresh Operation  77
    11.6.4 Write Enable Signal WE*  77
    11.6.5 CPU Write Operations  78
    11.6.6 Memory Multiplexing  78
  11.7 CHARACTER GENERATOR  78
  11.8 I/O SERIAL PORT  79
  11.9 VIDEO DISPLAY  79
  11.10 VIDEO SCAN GENERATOR  80
  11.11 KEYBOARD AND ASSOCIATED BUFFERS  81
  11.12 DISK DRIVE AND DISK CONTROLLER  81
  11.13 RESET AND NMI  83

12.0 Osborne 1 Schematics  85
  12.1 MAIN LOGIC BOARD SCHEMATICS  85
  12.2 DISK ELECTRONICS BOARD SCHEMATICS  105

Appendices
  A. Z80 INSTRUCTION SET  113
  B. 6821 PIA REGISTERS/INSTRUCTIONS  121
  C. 6850 ACIA REGISTERS/INSTRUCTIONS  135
  D. MB8877A DISK CONTROLLER REGISTERS/INSTRUCTIONS  147
  E. OSBORNE 1 SYSTEM SPECIFICATIONS  161

ROM/BIOS 1.3

ROM/BIOS 1.4
Perspective
1.1 INTRODUCTION 1
1.2 POWER ON 2
1.3 VARIATIONS IN DESIGN 2
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 categories; 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
2.1 OSBORNE 1 MAIN LOGIC BOARD 3
2.2 BOARD REVISIONS 3
2.3 MAIN BOARD LAYOUT 4
2.4 CENTRAL PROCESSOR 4
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:

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>prototype, not released</td>
</tr>
<tr>
<td>B</td>
<td>first release, requires wiring changes</td>
</tr>
<tr>
<td>C</td>
<td>incorporates all wiring changes, requires additional changes to correct layout problems</td>
</tr>
<tr>
<td>D</td>
<td>requires no wiring changes</td>
</tr>
<tr>
<td>E</td>
<td>complete relayout of board</td>
</tr>
<tr>
<td>F</td>
<td>revision E board with clock modification added</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>prototype, not released</td>
</tr>
<tr>
<td>B</td>
<td>memory</td>
</tr>
<tr>
<td>C</td>
<td>first revision shipped, Revision B bare board</td>
</tr>
<tr>
<td>D</td>
<td>incorporates rework required by Revision C bare board</td>
</tr>
<tr>
<td>E</td>
<td>same as Revision D loaded board, but assembled without rework</td>
</tr>
<tr>
<td>F</td>
<td>first incorporation of Revision E bare board, not released</td>
</tr>
<tr>
<td>G</td>
<td>primary production board through 3 November 1981, incorporates Revision E bare board</td>
</tr>
<tr>
<td>H-N</td>
<td>released</td>
</tr>
</tbody>
</table>

**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 accommodate 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.
NOTES:

- MARK "02" WITH INK IN AREA SHOWN
- MARK "A" WITH INK IN AREA SHOWN
- MARK SERIAL NO. WITH INK IN AREA SHOWN
- CRYSTAL ITEM SO MAY LAY FLAT OR BE UPRIGHT

Figure 2.3 — Main PC Board Assembly drawings
3.1 MEMORY BANKS 7
3.2 CP/M MEMORY ALLOCATION 7
3.3 MAIN MEMORY 9
3.4 MEMORY-MAPPED I/O 9
3.5 MEMORY ACCESS TIME 9
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

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 - 0002</td>
<td>Jump to BIOS warm start entry</td>
</tr>
<tr>
<td>0003</td>
<td>IOBYTE</td>
</tr>
<tr>
<td>0004</td>
<td>Drive number/current user</td>
</tr>
<tr>
<td>0005 - 0007</td>
<td>Jump to BDOS entry</td>
</tr>
<tr>
<td>0008 - 0037</td>
<td>Reserved for interrupts</td>
</tr>
<tr>
<td>0038 - 003A</td>
<td>RST7 (used by DDT)</td>
</tr>
<tr>
<td>003B - 003F</td>
<td>Reserved for interrupts</td>
</tr>
<tr>
<td>0040 - 004F</td>
<td>Scratch area used by BIOS</td>
</tr>
<tr>
<td>0050 - 005B</td>
<td>Not used</td>
</tr>
<tr>
<td>005C - 007C</td>
<td>File control block</td>
</tr>
<tr>
<td>007D - 007F</td>
<td>Random record position</td>
</tr>
<tr>
<td>0080 - 00FF</td>
<td>Default DMA buffer area</td>
</tr>
<tr>
<td>0100 - C7FF</td>
<td>Transient program area</td>
</tr>
<tr>
<td>C800 - E0FF</td>
<td>CCP/BDOS (CP/M)</td>
</tr>
<tr>
<td>E100 - E3FF</td>
<td>BIOS and Osborne buffer area</td>
</tr>
<tr>
<td>F000 - FFFF</td>
<td>Memory mapped video display</td>
</tr>
</tbody>
</table>

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)

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

**Figure 3.4.1 I/O Port Assignments In Bank 2**
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:

![IEEE-488 Signal Direction Diagram](image)

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:

<table>
<thead>
<tr>
<th>IEEE</th>
<th>OSBORNE</th>
<th>SIGNAL NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Data bit 1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Data bit 2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>Data bit 3</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>Data bit 4</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>End or Identify</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>Data valid</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>Not ready for data</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>No data accepted</td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>Interface clear</td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>Service request</td>
</tr>
<tr>
<td>11</td>
<td>21</td>
<td>Attention</td>
</tr>
<tr>
<td>12</td>
<td>23</td>
<td>Cable shield + GND</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>Data bit 5</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>Data bit 6</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
<td>Data bit 7</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>Data bit 8</td>
</tr>
<tr>
<td>17</td>
<td>10</td>
<td>Remote enable</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Signal ground</td>
</tr>
<tr>
<td>19</td>
<td>14</td>
<td>Signal ground</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>Signal ground</td>
</tr>
<tr>
<td>21</td>
<td>18</td>
<td>Signal ground</td>
</tr>
<tr>
<td>22</td>
<td>20</td>
<td>Signal ground</td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>Signal ground</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
<td>Signal ground</td>
</tr>
</tbody>
</table>

The pinouts for the signals described above are as follows:

![IEEE-488 Pinouts Diagram](image)

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

**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 BAT1; URI1; UP1; or LPT1; the IEEE-488 port is reconfigured by BIOS to be a simple 8-bit parallel input/output port with the following pinouts:

<table>
<thead>
<tr>
<th>Osborne IEEE Edge Connector</th>
<th>Centronics-Compatible Connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>pin 1</td>
<td>data 0</td>
</tr>
<tr>
<td>2</td>
<td>data 4</td>
</tr>
<tr>
<td>3</td>
<td>data 1</td>
</tr>
<tr>
<td>4</td>
<td>data 5</td>
</tr>
<tr>
<td>5</td>
<td>data 2</td>
</tr>
<tr>
<td>6</td>
<td>data 6</td>
</tr>
<tr>
<td>7</td>
<td>data 3</td>
</tr>
<tr>
<td>8</td>
<td>data 7</td>
</tr>
<tr>
<td></td>
<td>out strobe</td>
</tr>
<tr>
<td>11</td>
<td>ground</td>
</tr>
<tr>
<td>12</td>
<td>busy</td>
</tr>
<tr>
<td></td>
<td>ground</td>
</tr>
<tr>
<td>16</td>
<td>select</td>
</tr>
</tbody>
</table>

*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:

```
  7 6 5 4 3 2 1 0 ← Bit No.
   R  R  R  R ← Status register

  RDRF, Receive Data register is full
  TDRE, Transmit Data register is empty
  DCD, Data Carrier Detect signal status
  CTS, Clear To Send signal status
  FE, Framing Error
  OVRN, Receive overrun error
  PE, Parity Error
  IRQ, Interrupt request
```

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:

<table>
<thead>
<tr>
<th>Osborne 1</th>
<th>RS-232 device</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>ND</td>
</tr>
<tr>
<td>TXDATA</td>
<td>TXDATA</td>
</tr>
<tr>
<td>RXDATA</td>
<td>RXDATA</td>
</tr>
<tr>
<td>RTS</td>
<td>RTS</td>
</tr>
<tr>
<td>CTS</td>
<td>CTS</td>
</tr>
<tr>
<td>DSR</td>
<td>DSR</td>
</tr>
<tr>
<td>CARR DET</td>
<td>CARR DET</td>
</tr>
<tr>
<td>DTR</td>
<td>DTR</td>
</tr>
</tbody>
</table>

**Figure 4.2.1 RS-232 Signal Direction**

4.2.2 RS232 Pinouts

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

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

**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:

<table>
<thead>
<tr>
<th>Osborne 1</th>
<th>Modem Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>TXDATA</td>
<td>TXDATA</td>
</tr>
<tr>
<td>RXDATA</td>
<td>RXDATA</td>
</tr>
<tr>
<td>MSB</td>
<td>MSB</td>
</tr>
<tr>
<td>CTS</td>
<td>CTS</td>
</tr>
<tr>
<td>MCB</td>
<td>MCB</td>
</tr>
<tr>
<td>+12</td>
<td>POWER IN</td>
</tr>
<tr>
<td>RI</td>
<td>CD</td>
</tr>
</tbody>
</table>

**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
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 transmission.

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

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 replace 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 $7 \times 9$ in an $8 \times 10$ box. The following bits are used to encode the character to be displayed:

```
<table>
<thead>
<tr>
<th>bit number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
</tr>
</tbody>
</table>
```

- ASCII character
- underline attribute
- dim attribute
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:

<table>
<thead>
<tr>
<th>Character represented by this ASCII code</th>
<th>ASCII 00</th>
<th>ASCII 01</th>
<th>ASCII 02</th>
<th>ASCII 10</th>
<th>ASCII 11</th>
<th>ASCII 12</th>
<th>ASCII 70</th>
<th>ASCII 71</th>
<th>ASCII 72</th>
<th>ASCII 73</th>
<th>ASCII 74</th>
<th>ASCII 75</th>
<th>ASCII 76</th>
<th>ASCII 77</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1: 0000-007F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 2: 0080-00FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 3: 0100-017F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 4: 0180-01FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 5: 0200-027F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 6: 0280-02FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 7: 0300-037F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 8: 0380-03FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 9: 0400-047F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 10: 0480-04FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Row memory addresses

ASCII 4/16 = "A"

"A" is encoded in eight bytes as follows:

```
0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 0
0 1 1 1 1 0 0 0
0 1 1 0 0 1 1 0
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0
```

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:

![Display Logic Diagram](image)

**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:
**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 in-line 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):

![Diagram of PC edge to Single Inline connections]

**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 Vertical 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
6.1 POWER SUPPLY 31
6.2 SWITCHING VOLTAGE 31
6.3 POWER BOARD LAYOUT 32
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 accessible 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:

![Power Board Layout Diagram]

**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:

```
<table>
<thead>
<tr>
<th>power supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
</tr>
<tr>
<td>P2</td>
</tr>
<tr>
<td>P3</td>
</tr>
<tr>
<td>P4</td>
</tr>
</tbody>
</table>
```

The power supply is fused. If this fuse burns out, replace it with another fuse of the same type (T2A250V).
Disk Drives
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 DRIVES USED</td>
<td>33</td>
</tr>
<tr>
<td>7.2 DISK DRIVE INTERFACE BOARD</td>
<td>33</td>
</tr>
<tr>
<td>7.3 DISK CONTROLLER AND PINOUTS</td>
<td>37</td>
</tr>
<tr>
<td>7.4 CP/M BIOS DISK ADDRESSING</td>
<td>38</td>
</tr>
<tr>
<td>7.5 DISK INTERFACE</td>
<td>38</td>
</tr>
<tr>
<td>7.6 DISK DRIVE SPECIFICATIONS</td>
<td>38</td>
</tr>
<tr>
<td>7.7 DISK FORMAT</td>
<td>39</td>
</tr>
<tr>
<td>7.7.1 CP/M File Control Block</td>
<td>39</td>
</tr>
</tbody>
</table>
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 distinguished 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.
MARK PCB WITH SERIAL NUMBER IN AREA SHOWN
MARK PCB WITH DASH NO. -00 & REJ J IN AREA SHOWN
RIVET ITEM 26 AND ITEM 85 TO PCB WITH ITEM 84
1. SEE DETACHED L/M 2A03004 (A SIZE)
NOTES: UNLESS OTHERWISE SPECIFIED
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:

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

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-sector 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 Filename
61
62
63
64

65 File Type
66
67

68 Current extent number
69 Reserved

6A = 0 on BDOS call to open, make, or search

6B extent record count
6C
6D

. Disk record map
.

77
78
79

7A
7B Random record number
7C

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
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>THE KEYBOARD</td>
<td>41</td>
</tr>
<tr>
<td>8.2</td>
<td>KEYBOARD PINOUTS</td>
<td>41</td>
</tr>
<tr>
<td>8.3</td>
<td>KEYBOARD LAYOUT</td>
<td>42</td>
</tr>
<tr>
<td>8.4</td>
<td>KEYSWITCH MATRIX</td>
<td>42</td>
</tr>
<tr>
<td>8.5</td>
<td>PROGRAMMABLE KEYS</td>
<td>43</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>pin</th>
<th>ground</th>
<th>2</th>
<th>row 4</th>
<th>3</th>
<th>row 0</th>
<th>4</th>
<th>row 3</th>
<th>5</th>
<th>row 6</th>
<th>used for address</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>row 2</td>
<td>7</td>
<td>row 5</td>
<td>8</td>
<td>row 1</td>
<td>9</td>
<td>row 7</td>
<td>10</td>
<td>col 0</td>
<td>11</td>
</tr>
<tr>
<td>14</td>
<td>col 4</td>
<td>15</td>
<td>col 5</td>
<td>16</td>
<td>col 6</td>
<td>17</td>
<td>col 7</td>
<td>18</td>
<td>no connection</td>
<td>19</td>
</tr>
</tbody>
</table>

Figure 8.2 Keyboard Pinouts
8.3 KEYBOARD LAYOUT

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

![Keyboard Layout](image)

**Figure 8.3 Osborne 1 Key Layout**

8.4 KEYSWITCH MATRIX

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

![Keyswitch Matrix](image)

**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 dismantle 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.*

---

5. 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.*

---

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

7. 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 harness 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 chassis 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 chassis and bezel as described 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.
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 connection 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
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1 SOFTWARE REVISIONS</td>
<td>63</td>
</tr>
<tr>
<td>10.2 THE MONITOR ROM</td>
<td>64</td>
</tr>
<tr>
<td>10.2.1 Console Routines</td>
<td>65</td>
</tr>
<tr>
<td>10.2.2 Other Interface Routines</td>
<td>66</td>
</tr>
<tr>
<td>10.2.3 ROM Listings</td>
<td>66</td>
</tr>
<tr>
<td>10.3 BDOS CALLS</td>
<td>66</td>
</tr>
<tr>
<td>10.4 OSBORNE 1 BIOS ROUTINES</td>
<td>68</td>
</tr>
<tr>
<td>10.4.1 The IOBYTE</td>
<td>68</td>
</tr>
<tr>
<td>10.4.2 BIOS Listings</td>
<td>69</td>
</tr>
</tbody>
</table>
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 protocols (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:
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
^A             boot from disk drive B
^H             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**

<table>
<thead>
<tr>
<th>track</th>
<th>sectors</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 - 8</td>
<td>CCP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9 - 10</td>
<td>BDOS</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1 - 10</td>
<td>BDOS</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1 - 2</td>
<td>BDOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 - 10</td>
<td>BIOS</td>
<td></td>
</tr>
</tbody>
</table>
1.4 ROM

<table>
<thead>
<tr>
<th>Track 0</th>
<th>Sectors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 - 2</td>
<td>CCP</td>
</tr>
<tr>
<td></td>
<td>3 - 5</td>
<td>BDOS</td>
</tr>
<tr>
<td>Track 1</td>
<td>Sectors</td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>1 - 1.5</td>
<td>BDOS</td>
</tr>
<tr>
<td></td>
<td>1.5 - 2.5</td>
<td>BIOS</td>
</tr>
</tbody>
</table>

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:

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

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 maintenance 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

<table>
<thead>
<tr>
<th>Function Number</th>
<th>Description</th>
<th>Entry Values</th>
<th>Return Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>system reset</td>
<td></td>
<td>A=character</td>
</tr>
<tr>
<td>1</td>
<td>console read</td>
<td></td>
<td>A=character</td>
</tr>
<tr>
<td>2</td>
<td>console write</td>
<td>E=character</td>
<td>A=character</td>
</tr>
<tr>
<td>3</td>
<td>reader read</td>
<td>E=character</td>
<td>A=character</td>
</tr>
<tr>
<td>4</td>
<td>punch write</td>
<td>E=character</td>
<td>A=character</td>
</tr>
<tr>
<td>5</td>
<td>list write</td>
<td>E=character</td>
<td>A=character</td>
</tr>
<tr>
<td>6</td>
<td>direct I/O</td>
<td>E=type*</td>
<td>A=0 if busy</td>
</tr>
<tr>
<td>7</td>
<td>get IOBYTE</td>
<td>E=IOBYTE</td>
<td>A=IOBYTE</td>
</tr>
<tr>
<td>8</td>
<td>set IOBYTE</td>
<td>DE=address</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>print string</td>
<td>DE=address</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>read console buffer</td>
<td></td>
<td>A=status</td>
</tr>
<tr>
<td>11</td>
<td>get console status</td>
<td></td>
<td>HL=version #</td>
</tr>
<tr>
<td>12</td>
<td>get version number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>reset disk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>select disk</td>
<td>E=drive</td>
<td>A=found ***</td>
</tr>
<tr>
<td>Function Number</td>
<td>Description</td>
<td>Entry Values</td>
<td>Return Values</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------</td>
<td>--------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>15</td>
<td>open file</td>
<td>E=drive</td>
<td>A=found ***</td>
</tr>
<tr>
<td>16</td>
<td>close file</td>
<td>DE=FCB address</td>
<td>A=found ***</td>
</tr>
<tr>
<td>17</td>
<td>search for file</td>
<td>DE=FCB address</td>
<td>A=found ***</td>
</tr>
<tr>
<td>18</td>
<td>search for next</td>
<td></td>
<td>A=found</td>
</tr>
<tr>
<td>19</td>
<td>delete file</td>
<td>DE=FCB address</td>
<td>A=found ***</td>
</tr>
<tr>
<td>20</td>
<td>read next record</td>
<td>DE=FCB address</td>
<td>A=found ***</td>
</tr>
<tr>
<td>21</td>
<td>create file</td>
<td>DE=FCB address</td>
<td>A=found</td>
</tr>
<tr>
<td>22</td>
<td>create file</td>
<td>DE=FCB address</td>
<td>A=found</td>
</tr>
<tr>
<td>23</td>
<td>rename file</td>
<td>DE=FCB address</td>
<td>A=found</td>
</tr>
<tr>
<td>24</td>
<td>get login vector</td>
<td></td>
<td>HL=drive</td>
</tr>
<tr>
<td>25</td>
<td>get disk number</td>
<td></td>
<td>A=drive</td>
</tr>
<tr>
<td>26</td>
<td>set DMA address</td>
<td>DE=DMA address</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>get allocate vector</td>
<td></td>
<td>HL=allocation</td>
</tr>
<tr>
<td>28</td>
<td>write protect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>get R/O vector</td>
<td></td>
<td>HL=R/O vector</td>
</tr>
<tr>
<td>30</td>
<td>set file attributes</td>
<td>DE=FCB address</td>
<td>A=found ***</td>
</tr>
<tr>
<td>31</td>
<td>get disk header address</td>
<td></td>
<td>HL=address</td>
</tr>
<tr>
<td>32</td>
<td>set/get user number code (set)</td>
<td>D=FF (get)</td>
<td>A=user #</td>
</tr>
<tr>
<td>33</td>
<td>read random</td>
<td>DE=FCB address</td>
<td>A=error ***</td>
</tr>
<tr>
<td>34</td>
<td>write random</td>
<td>DE=FCB address</td>
<td>A=error</td>
</tr>
<tr>
<td>35</td>
<td>compute file size</td>
<td>DE=FCB address</td>
<td>A=record</td>
</tr>
<tr>
<td>36</td>
<td>set random record</td>
<td>DE=FCB address</td>
<td>A=record</td>
</tr>
<tr>
<td>37</td>
<td>reset drive</td>
<td>DE=drive vector</td>
<td>A=0</td>
</tr>
<tr>
<td>38</td>
<td>write random zero fill</td>
<td>DE=FCB address</td>
<td>A=error ***</td>
</tr>
</tbody>
</table>

*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

 coded in C register

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:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST</td>
<td>PUNCH</td>
<td>READER</td>
<td>CONSOLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Logical Device</th>
<th>Value</th>
<th>CP/M Physical</th>
<th>Osborne Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Console (CON:)</td>
<td>00</td>
<td>TTY:</td>
<td>Keyboard + Screen</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>CRT:</td>
<td>Serial Port</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>BAT:</td>
<td>Parallel Port</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>UCI:</td>
<td>IEEE Port</td>
</tr>
<tr>
<td>Reader (RDR:)</td>
<td>00</td>
<td>TTY:</td>
<td>Keyboard + Screen</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>PTR:</td>
<td>Serial Port</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>UR1:</td>
<td>Parallel Port</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>UR2:</td>
<td>IEEE Port</td>
</tr>
<tr>
<td>Punch (PUN:)</td>
<td>00</td>
<td>TTY:</td>
<td>Keyboard + Screen</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>PTR:</td>
<td>Serial Port</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>UP1:</td>
<td>Parallel Port</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>UP2:</td>
<td>IEEE Port</td>
</tr>
<tr>
<td>List (LST:)</td>
<td>00</td>
<td>TTY:</td>
<td>Keyboard + Screen</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>CRT:</td>
<td>Serial Port</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>LPT:</td>
<td>Parallel (Centronics)</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>UL1:</td>
<td>IEEE Port</td>
</tr>
</tbody>
</table>
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.
11.1 INTRODUCTION 71
11.2 FUNCTIONAL OPERATION 71
11.3 MNEMONIC CODES 72
11.4 BASIC TIMING 73
11.5 ROM 75
11.6 RAM 77
   11.6.1 Address Lines 77
   11.6.2 RAM Data Output 77
   11.6.3 RAM Refresh Operation 77
   11.6.4 Write Enable Signal WE* 77
   11.6.5 CPU Write Operations 78
   11.6.6 Memory Multiplexing 78
11.7 CHARACTER GENERATOR 78
11.8 I/O SERIAL PORT 79
11.9 VIDEO DISPLAY 79
11.10 VIDEO SCAN GENERATOR 80
11.11 KEYBOARD AND ASSOCIATED BUFFERS 81
11.12 DISK DRIVE AND DISK CONTROLLER 81
   11.12.1 Disk Drive Controller 81
   11.12.2 Disk Drive 81
   11.12.3 Disk Drive Electronics 82
   11.12.4 Disk Format 82
11.13 RESET AND NMI 83
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 tristate 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.
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:

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS</td>
<td>Column Address Strobe</td>
<td>Strobes RAM column address</td>
</tr>
<tr>
<td>DOT CLK</td>
<td>Dot clock</td>
<td>Used in conjunction with the Character Generator and video logic to produce dot patterns on display</td>
</tr>
<tr>
<td>NMI</td>
<td>Non Maskable Interrupt</td>
<td>Restarts ROM program</td>
</tr>
<tr>
<td>RAS</td>
<td>Row Address Strobe</td>
<td>Stobes RAM row address</td>
</tr>
</tbody>
</table>
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*) 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](image-url)
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 UBI (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 UBI 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 UBI 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 UBI-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.
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 UBI 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 UBI 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 4x8 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 I 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 $8 \times 10$ 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 P85 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:
<table>
<thead>
<tr>
<th>ITEM</th>
<th>PART NO.</th>
<th>QTY.</th>
<th>REFERENCE DESIGNATORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1A1063-00</td>
<td>A/R</td>
<td>EA ISCHEM, MULTI LAYER MAIN LOGIC PCB</td>
</tr>
<tr>
<td>001</td>
<td>3P1065-00</td>
<td>001</td>
<td>EA 1FAR, MULTI LAYER MAIN LOGIC PCB</td>
</tr>
<tr>
<td>002</td>
<td>3A10256-00</td>
<td>001</td>
<td>EA 1D CHAR GEN PRGMD EPROM ASSY</td>
</tr>
<tr>
<td>003</td>
<td>3A10260-00</td>
<td>001</td>
<td>EA 1D SYSTEM PRGMD EPROM ASSY</td>
</tr>
<tr>
<td>004</td>
<td>7P1100-00</td>
<td>004</td>
<td>EA 74LS00</td>
</tr>
<tr>
<td>005</td>
<td>7P1102-00</td>
<td>002</td>
<td>EA 74LS02</td>
</tr>
<tr>
<td>006</td>
<td>7P1103-00</td>
<td>003</td>
<td>EA 74LS03</td>
</tr>
<tr>
<td>007</td>
<td>7P1104-00</td>
<td>002</td>
<td>EA 74LS04</td>
</tr>
<tr>
<td>008</td>
<td>7P1105-00</td>
<td>002</td>
<td>EA 74LS05</td>
</tr>
<tr>
<td>009</td>
<td>7P1106-00</td>
<td>004</td>
<td>EA 74LS06</td>
</tr>
<tr>
<td>010</td>
<td>7P1107-00</td>
<td>001</td>
<td>EA 74LS11</td>
</tr>
<tr>
<td>011</td>
<td>7P1108-00</td>
<td>001</td>
<td>EA 74LS11</td>
</tr>
<tr>
<td>012</td>
<td>7P1109-00</td>
<td>001</td>
<td>EA 74LS109</td>
</tr>
<tr>
<td>013</td>
<td>7P1110-00</td>
<td>001</td>
<td>EA 74LS109</td>
</tr>
<tr>
<td>014</td>
<td>7P1111-00</td>
<td>002</td>
<td>EA 74LS120</td>
</tr>
<tr>
<td>015</td>
<td>7P1112-00</td>
<td>004</td>
<td>EA 74LS153</td>
</tr>
<tr>
<td>016</td>
<td>7P1113-00</td>
<td>001</td>
<td>EA 74LS155</td>
</tr>
<tr>
<td>017</td>
<td>7P1114-00</td>
<td>002</td>
<td>EA 74LS157</td>
</tr>
<tr>
<td>018</td>
<td>7P1115-00</td>
<td>001</td>
<td>EA 74LS157</td>
</tr>
<tr>
<td>019</td>
<td>7P1116-00</td>
<td>001</td>
<td>EA 74LS156</td>
</tr>
<tr>
<td>020</td>
<td>7P1117-00</td>
<td>001</td>
<td>EA 74LS175</td>
</tr>
<tr>
<td>021</td>
<td>7P1124-00</td>
<td>002</td>
<td>EA 74LS244</td>
</tr>
<tr>
<td>022</td>
<td>7P1127-00</td>
<td>001</td>
<td>EA 74LS273</td>
</tr>
<tr>
<td>023</td>
<td>7P1130-00</td>
<td>001</td>
<td>EA 74LS390</td>
</tr>
<tr>
<td>024</td>
<td>7P1139-00</td>
<td>002</td>
<td>EA 74LS393</td>
</tr>
<tr>
<td>025</td>
<td>7P1189-00</td>
<td>001</td>
<td>EA 181S95</td>
</tr>
<tr>
<td>026</td>
<td>7P1197-00</td>
<td>002</td>
<td>EA 74S04</td>
</tr>
<tr>
<td>027</td>
<td>7P1198-00</td>
<td>002</td>
<td>EA 74S244</td>
</tr>
<tr>
<td>028</td>
<td>7P1199-00</td>
<td>001</td>
<td>EA 14116 16K MOS RAM</td>
</tr>
<tr>
<td>029</td>
<td>7P1200-00</td>
<td>001</td>
<td>EA 17438</td>
</tr>
<tr>
<td>030</td>
<td>7P1208-00</td>
<td>002</td>
<td>EA 16821 PARALLEL I/O</td>
</tr>
<tr>
<td>031</td>
<td>7P1209-00</td>
<td>001</td>
<td>EA 16850 SERIAL I/O</td>
</tr>
<tr>
<td>032</td>
<td>7P1211-00</td>
<td>001</td>
<td>EA 11793 DISK CONTROLLER</td>
</tr>
<tr>
<td>033</td>
<td>7P1213-00</td>
<td>001</td>
<td>EA 1280A MICRO PROCESSOR</td>
</tr>
<tr>
<td>034</td>
<td>7P1214-00</td>
<td>003</td>
<td>EA 14116 16K MOS RAM</td>
</tr>
</tbody>
</table>

LM CONTINUED
<table>
<thead>
<tr>
<th>ITEM</th>
<th>PART NO.</th>
<th>QTY</th>
<th>UM</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>035</td>
<td>7P11817-00</td>
<td>001</td>
<td>EA</td>
<td>ILM 1458 LINEAR, OP AMP</td>
</tr>
<tr>
<td>036</td>
<td>7P11818-00</td>
<td>001</td>
<td>EA</td>
<td>ILM 3900 LINEAR, OP AMP</td>
</tr>
<tr>
<td>037</td>
<td>7P11820-00</td>
<td>001</td>
<td>EA</td>
<td>INE555V LINEAR, TIMER</td>
</tr>
<tr>
<td>040</td>
<td>7P30001-00</td>
<td>002</td>
<td>EA</td>
<td>IN4001 DIODE, 1A, SILICON</td>
</tr>
<tr>
<td>041</td>
<td>7P30002-00</td>
<td>001</td>
<td>EA</td>
<td>IN5231B DIODE, ZENER 5.1V, 10%</td>
</tr>
<tr>
<td>042</td>
<td>7P30003-00</td>
<td>002</td>
<td>EA</td>
<td>MR501 DIODE, RECTIFIER 3A</td>
</tr>
<tr>
<td>045</td>
<td>7P30006-00</td>
<td>001</td>
<td>EA</td>
<td>2N3906 TRANSISTOR, PNP</td>
</tr>
<tr>
<td>046</td>
<td>7P30007-00</td>
<td>001</td>
<td>EA</td>
<td>2N3904 TRANSISTOR, NPN</td>
</tr>
<tr>
<td>050</td>
<td>7P40002-00</td>
<td>001</td>
<td>EA</td>
<td>XTAL 15.9744MHZ +-.005%CH18U</td>
</tr>
<tr>
<td>051</td>
<td>7P40026-00</td>
<td>001</td>
<td>EA</td>
<td>ALARM AUDIO, PIEZO PC MOUNT</td>
</tr>
<tr>
<td>052</td>
<td>7P45501-00</td>
<td>005</td>
<td>EA</td>
<td>ICAP, DIPPED, TANT, 1.0 UF 20V</td>
</tr>
<tr>
<td>053</td>
<td>7P45502-00</td>
<td>003</td>
<td>EA</td>
<td>ICAP, DIPPED, TANT, 15UF, 15V</td>
</tr>
<tr>
<td>054</td>
<td>7P45604-00</td>
<td>001</td>
<td>EA</td>
<td>ICAP, ELECT., ALUM 22UF, 16V</td>
</tr>
<tr>
<td>055</td>
<td>7P46103-00</td>
<td>001</td>
<td>EA</td>
<td>ICAP, AXL, CER, .001uf +10%, 50V</td>
</tr>
<tr>
<td>056</td>
<td>7P46104-00</td>
<td>028</td>
<td>EA</td>
<td>ICAP, AXL, CER, .01UF, +30%, 25V</td>
</tr>
<tr>
<td>057</td>
<td>7P46105-00</td>
<td>042</td>
<td>EA</td>
<td>ICAP, AXL, CER, 0.1UF, +50%, 25V</td>
</tr>
<tr>
<td>058</td>
<td>7P46330-00</td>
<td>001</td>
<td>EA</td>
<td>ICAP, AXL, CER, 33PF +5%, 50V</td>
</tr>
<tr>
<td>059</td>
<td>7P46680-00</td>
<td>001</td>
<td>EA</td>
<td>ICAP, AXL, CER, 68PF, +05%, 50V</td>
</tr>
<tr>
<td>060</td>
<td>7P46180-00</td>
<td>001</td>
<td>EA</td>
<td>ICAP, AXL, CER, 180PF, +10%, 50V</td>
</tr>
<tr>
<td>061</td>
<td>7P46106-00</td>
<td>001</td>
<td>EA</td>
<td>ICAP, 22UF, +20%, 12V ALUM EL AXL</td>
</tr>
<tr>
<td>062</td>
<td>7P50006-00</td>
<td>001</td>
<td>EA</td>
<td>IRES, VAR, PC MT, 100K, 30%, 1/4W</td>
</tr>
<tr>
<td>063</td>
<td>7P50007-00</td>
<td>001</td>
<td>EA</td>
<td>IRES, VAR, PC MT, 500 OHM, 30%, 1/4W</td>
</tr>
<tr>
<td>064</td>
<td>7P51001-00</td>
<td>001</td>
<td>EA</td>
<td>IRES, NWK, 150 OHM SIP 8/P P/U</td>
</tr>
<tr>
<td>065</td>
<td>7P51004-00</td>
<td>001</td>
<td>EA</td>
<td>IRES, NWK, 3.3K SIP 8 PIN P/U</td>
</tr>
<tr>
<td>066</td>
<td>7P51007-00</td>
<td>001</td>
<td>EA</td>
<td>IRES, NWK, 10K, SIP 10 PIN P/U</td>
</tr>
<tr>
<td>067</td>
<td>7P51008-00</td>
<td>003</td>
<td>EA</td>
<td>IRES, NWK, 3.3K, SIP 10 PIN P/U</td>
</tr>
<tr>
<td>068</td>
<td>7P51009-00</td>
<td>002</td>
<td>EA</td>
<td>IRES, NWK, 6.8K, SIP 10 PIN P/U</td>
</tr>
<tr>
<td>069</td>
<td>7P51011-00</td>
<td>004</td>
<td>EA</td>
<td>IRES, NWK, 33 OHM SIP 8 PIN ISO</td>
</tr>
<tr>
<td>070</td>
<td>7P51014-00</td>
<td>002</td>
<td>EA</td>
<td>IRES, NWK, 82 OHM SIP 8 PIN ISO</td>
</tr>
<tr>
<td>071</td>
<td>7P52101-00</td>
<td>003</td>
<td>EA</td>
<td>IRES, FXD, 1/4W 5X 100 OHM</td>
</tr>
<tr>
<td>072</td>
<td>7P52102-00</td>
<td>005</td>
<td>EA</td>
<td>IRES, FXD, 1/4W 5X 1.0K</td>
</tr>
<tr>
<td>073</td>
<td>7P52103-00</td>
<td>007</td>
<td>EA</td>
<td>IRES, FXD, 1/4W 5X 10K</td>
</tr>
<tr>
<td>ITEM</td>
<td>PART NO.</td>
<td>QTY</td>
<td>UM</td>
<td>TITLE</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-----</td>
<td>----</td>
<td>-------</td>
</tr>
<tr>
<td>1075</td>
<td>7P52104-00</td>
<td>005</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 100K</td>
</tr>
<tr>
<td>1076</td>
<td>7P52181-00</td>
<td>001</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 100 OHM</td>
</tr>
<tr>
<td>1077</td>
<td>7P52220-00</td>
<td>002</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 22 OHM</td>
</tr>
<tr>
<td>1078</td>
<td>7P52221-00</td>
<td>002</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 220 OHM</td>
</tr>
<tr>
<td>1079</td>
<td>7P52222-00</td>
<td>003</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 2.2K</td>
</tr>
<tr>
<td>1080</td>
<td>7P52223-00</td>
<td>001</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 22K</td>
</tr>
<tr>
<td>1081</td>
<td>7P52330-00</td>
<td>001</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 33 OHM</td>
</tr>
<tr>
<td>1082</td>
<td>7P52331-00</td>
<td>003</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 330 OHM</td>
</tr>
<tr>
<td>1083</td>
<td>7P52332-00</td>
<td>002</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 3.3K</td>
</tr>
<tr>
<td>1084</td>
<td>7P52470-00</td>
<td>001</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 47 OHM</td>
</tr>
<tr>
<td>1085</td>
<td>7P52471-00</td>
<td>002</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 470 OHM</td>
</tr>
<tr>
<td>1086</td>
<td>7P52472-00</td>
<td>001</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 4.7K</td>
</tr>
<tr>
<td>1087</td>
<td>7P52473-00</td>
<td>001</td>
<td>EA</td>
<td>RES, FXD, 1/4W 5% 47K</td>
</tr>
<tr>
<td>1088</td>
<td>7P61106-00</td>
<td>001</td>
<td>EA</td>
<td>SWITCH, DPDT, MOMENTARY, ACTION</td>
</tr>
<tr>
<td>1089</td>
<td>7P61103-00</td>
<td>001</td>
<td>EA</td>
<td>SWITCH BUTTON</td>
</tr>
<tr>
<td>1090</td>
<td>7P63000-00</td>
<td>004</td>
<td>EA</td>
<td>CONN, SHUNT, 2 PIN</td>
</tr>
<tr>
<td>1091</td>
<td>7P64125-00</td>
<td>001</td>
<td>EA</td>
<td>CONN, PC MT, RT ANGL, DB 25S</td>
</tr>
<tr>
<td>1092</td>
<td>7P64209-00</td>
<td>002</td>
<td>EA</td>
<td>CONN, PC MT, RT ANGL, DE 9P</td>
</tr>
<tr>
<td>1093</td>
<td>7P64100-00</td>
<td>002</td>
<td>EA</td>
<td>HDR, PC MT, STR, 100ML SIL 2 PIN</td>
</tr>
<tr>
<td>1094</td>
<td>7P64410-00</td>
<td>001</td>
<td>EA</td>
<td>HDR, PC MT, STR, 100 SIL 10 PIN</td>
</tr>
<tr>
<td>1095</td>
<td>7P64207-00</td>
<td>001</td>
<td>EA</td>
<td>HDR, PC MT, STR, 156 SIL 7 PIN</td>
</tr>
<tr>
<td>1096</td>
<td>7P64400-00</td>
<td>001</td>
<td>EA</td>
<td>HDR, PC MT, STR, DIL 8 PIN</td>
</tr>
<tr>
<td>1097</td>
<td>7P64520-00</td>
<td>001</td>
<td>EA</td>
<td>HDR, PC MT, RT ANGL DIL 20 PIN</td>
</tr>
<tr>
<td>1098</td>
<td>7P64534-00</td>
<td>001</td>
<td>EA</td>
<td>HDR, PC MT, RT ANGL, DIL 34 PIN</td>
</tr>
<tr>
<td>1099</td>
<td>7P65124-00</td>
<td>002</td>
<td>EA</td>
<td>SOCKET LOW PROFILE, DIP 24 PIN</td>
</tr>
<tr>
<td>1100</td>
<td>7P65140-00</td>
<td>002</td>
<td>EA</td>
<td>SOCKET LOW PROFILE, DIP 40 PIN</td>
</tr>
</tbody>
</table>
12.2 DISK ELECTRONICS BOARD
SCHEMATICS
<table>
<thead>
<tr>
<th>ITEM</th>
<th>PART NO.</th>
<th>QTY</th>
<th>TITLE</th>
<th>REFERENCE DESIGNATORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1046</td>
<td>17550003-00</td>
<td>1</td>
<td>RES, VARIABLE</td>
<td>20K 20 TURN</td>
</tr>
<tr>
<td>1047</td>
<td>17551002-00</td>
<td>1</td>
<td>RES, NETWORK, SIP &amp; PIN</td>
<td></td>
</tr>
<tr>
<td>1048</td>
<td>17551003-00</td>
<td>1</td>
<td>47-10K10%, 1/8W BUSS</td>
<td>RN6</td>
</tr>
<tr>
<td>1049</td>
<td>17551003-00</td>
<td>1</td>
<td>150R, 10%, 1/2W, BUSS</td>
<td>RN8</td>
</tr>
<tr>
<td>1050</td>
<td>17551005-00</td>
<td>1</td>
<td>47K, 10%, 1/8W, BUSS</td>
<td>RN1, 4, 5</td>
</tr>
<tr>
<td>1051</td>
<td>17551012-00</td>
<td>1</td>
<td>33K, 10%, 1/8W, ISOLATED</td>
<td>RN7</td>
</tr>
<tr>
<td>1052</td>
<td>17551021-00</td>
<td>1</td>
<td>RES, C.F., 5%, 1/4W</td>
<td>R1</td>
</tr>
<tr>
<td>1053</td>
<td>17551025-00</td>
<td>1</td>
<td>1.0K</td>
<td>R15</td>
</tr>
<tr>
<td>1054</td>
<td>17552153-00</td>
<td>1</td>
<td>BUSS</td>
<td>R7, 13</td>
</tr>
<tr>
<td>1055</td>
<td>17552164-00</td>
<td>1</td>
<td>15K</td>
<td>R22</td>
</tr>
<tr>
<td>1056</td>
<td>17552203-00</td>
<td>1</td>
<td>20K</td>
<td>R37</td>
</tr>
<tr>
<td>1057</td>
<td>17552220-00</td>
<td>1</td>
<td>10R</td>
<td>R39</td>
</tr>
<tr>
<td>1058</td>
<td>17552270-00</td>
<td>1</td>
<td>2.2K</td>
<td>R10, 11</td>
</tr>
<tr>
<td>1059</td>
<td>17552273-00</td>
<td>1</td>
<td>2.7K</td>
<td>R32</td>
</tr>
<tr>
<td>1060</td>
<td>17552332-00</td>
<td>1</td>
<td>3.3K</td>
<td>R6</td>
</tr>
<tr>
<td>1061</td>
<td>17552335-00</td>
<td>1</td>
<td>3.3K</td>
<td>R33</td>
</tr>
<tr>
<td>1062</td>
<td>17552339-00</td>
<td>1</td>
<td>39K</td>
<td>R29</td>
</tr>
<tr>
<td>1063</td>
<td>17552472-00</td>
<td>1</td>
<td>4.7K</td>
<td>R4, 5, 8, 9, 20, 26, 27, 31, 34,</td>
</tr>
<tr>
<td>1064</td>
<td>17552514-00</td>
<td>1</td>
<td>10K</td>
<td>R37, 38</td>
</tr>
<tr>
<td>1065</td>
<td>17552662-00</td>
<td>1</td>
<td>6.8K</td>
<td>R36</td>
</tr>
<tr>
<td>1066</td>
<td>17552751-00</td>
<td>1</td>
<td>750K</td>
<td>R18</td>
</tr>
<tr>
<td>1067</td>
<td>17552821-00</td>
<td>1</td>
<td>820K</td>
<td>R10, 12, 25, 38</td>
</tr>
<tr>
<td>1071</td>
<td>17553560-00</td>
<td>1</td>
<td>RES, C.F. 5%, 1/2W</td>
<td>56r</td>
</tr>
<tr>
<td>1072</td>
<td>17557152-00</td>
<td>1</td>
<td>RES, M.F. 2%, 1/4W</td>
<td>R19</td>
</tr>
<tr>
<td>1073</td>
<td>17553223-00</td>
<td>1</td>
<td>15K</td>
<td>R42</td>
</tr>
<tr>
<td>1074</td>
<td>17557331-00</td>
<td>1</td>
<td>22K</td>
<td>R24, 41, 44</td>
</tr>
<tr>
<td>1075</td>
<td>17557472-00</td>
<td>1</td>
<td>330K</td>
<td>R2, 3, 14, 16, 32</td>
</tr>
<tr>
<td>1076</td>
<td>17553912-00</td>
<td>1</td>
<td>4.7K</td>
<td>R15, 17</td>
</tr>
<tr>
<td>1077</td>
<td>17554108-00</td>
<td>1</td>
<td>9.1K</td>
<td>R43</td>
</tr>
<tr>
<td>1078</td>
<td>17564316-00</td>
<td>1</td>
<td>8PIN STR, TIN</td>
<td>TTP</td>
</tr>
<tr>
<td>1079</td>
<td>17564906-00</td>
<td>1</td>
<td>16PIN AT ANGL, TIN</td>
<td>IP2</td>
</tr>
<tr>
<td>1080</td>
<td>17565261-00</td>
<td>1</td>
<td>SOCKET, IC, SIP,</td>
<td>8PIN</td>
</tr>
<tr>
<td>1081</td>
<td>17565261-00</td>
<td>1</td>
<td>25&quot;DIA X .093&quot;THK</td>
<td>RN3</td>
</tr>
<tr>
<td>1082</td>
<td>1757041-00</td>
<td>1</td>
<td>RIVET, 125&quot;DIA X 093&quot;THK</td>
<td></td>
</tr>
<tr>
<td>1083</td>
<td>17573021-00</td>
<td>1</td>
<td>HEATSINK</td>
<td></td>
</tr>
<tr>
<td>1084</td>
<td>1030064-00</td>
<td>1</td>
<td>DISK ELECT. PCB SCHEM REV J</td>
<td></td>
</tr>
<tr>
<td>1085</td>
<td>3P0401-00</td>
<td>1</td>
<td>DISK ELECT. PCB FAB REV D</td>
<td></td>
</tr>
<tr>
<td>ITEM</td>
<td>PART NO.</td>
<td>QTY</td>
<td>TITLE</td>
<td>REFERENCE DESIGNATOR</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-----</td>
<td>-------</td>
<td>---------------------</td>
</tr>
<tr>
<td>001</td>
<td>17P11014-00</td>
<td>1002</td>
<td>IC, TTL</td>
<td>SN74LS14N</td>
</tr>
<tr>
<td>002</td>
<td>17P11036-00</td>
<td>1001</td>
<td>SN74LS14N</td>
<td>U8</td>
</tr>
<tr>
<td>003</td>
<td>17P11112-00</td>
<td>1001</td>
<td>SN74LS12N</td>
<td>U9</td>
</tr>
<tr>
<td>004</td>
<td>17P11112-00</td>
<td>1001</td>
<td>SN74LS123N</td>
<td>U13</td>
</tr>
<tr>
<td>005</td>
<td>17P111132-00</td>
<td>1001</td>
<td>SN74LS132N</td>
<td>U18</td>
</tr>
<tr>
<td>006</td>
<td>17P111139-00</td>
<td>1001</td>
<td>SN74LS139N</td>
<td>U15</td>
</tr>
<tr>
<td>007</td>
<td>17P11800-00</td>
<td>1001</td>
<td>SN7438N</td>
<td>U12</td>
</tr>
<tr>
<td>008</td>
<td>17P11801-00</td>
<td>1001</td>
<td>SN75451BP</td>
<td>U21, 22</td>
</tr>
<tr>
<td>009</td>
<td>17P11802-00</td>
<td>1001</td>
<td>IC, CMOS</td>
<td>CD4001BE</td>
</tr>
<tr>
<td>010</td>
<td>17P11803-00</td>
<td>1001</td>
<td>CD4013BE</td>
<td>U7</td>
</tr>
<tr>
<td>011</td>
<td>17P11804-00</td>
<td>1001</td>
<td>CD4023BE</td>
<td>U20</td>
</tr>
<tr>
<td>012</td>
<td>17P11805-00</td>
<td>1001</td>
<td>CD4025BE</td>
<td>U8</td>
</tr>
<tr>
<td>013</td>
<td>17P11807-00</td>
<td>1001</td>
<td>CD4070BE</td>
<td>U13</td>
</tr>
<tr>
<td>015</td>
<td>17P21004-00</td>
<td>1001</td>
<td>IC, LINEAR</td>
<td>TL081CP</td>
</tr>
<tr>
<td>016</td>
<td>17P11820-00</td>
<td>1001</td>
<td>IC, LINEAR</td>
<td>LM555CN</td>
</tr>
<tr>
<td>017</td>
<td>17P20011-00</td>
<td>1001</td>
<td>LM311N</td>
<td>U5</td>
</tr>
<tr>
<td>018</td>
<td>17P20021-00</td>
<td>1001</td>
<td>CA3054</td>
<td>U1</td>
</tr>
<tr>
<td>019</td>
<td>17P20031-00</td>
<td>1001</td>
<td>NE5520N</td>
<td>U6</td>
</tr>
<tr>
<td>020</td>
<td>17P20041-00</td>
<td>1001</td>
<td>DIODE</td>
<td>1N4001</td>
</tr>
<tr>
<td>021</td>
<td>17P20051-00</td>
<td>1001</td>
<td>1N4148</td>
<td>D1, 2, 7</td>
</tr>
<tr>
<td>022</td>
<td>17P20071-00</td>
<td>1001</td>
<td>TRANSISTORS</td>
<td>2N3904</td>
</tr>
<tr>
<td>023</td>
<td>17P30001-00</td>
<td>1001</td>
<td>TIP110</td>
<td>C3</td>
</tr>
<tr>
<td>024</td>
<td>17P40101-00</td>
<td>1001</td>
<td>INDUCTORS</td>
<td>150uH +10%</td>
</tr>
<tr>
<td>025</td>
<td>17P40102-00</td>
<td>1001</td>
<td>560uH +10%</td>
<td>L1, 2</td>
</tr>
<tr>
<td>026</td>
<td>17P45505-00</td>
<td>1001</td>
<td>CAP, TANTUM</td>
<td>22uF +20%, 6V</td>
</tr>
<tr>
<td>027</td>
<td>17P45506-00</td>
<td>1001</td>
<td>2.2uF +10%, 6V</td>
<td>C17</td>
</tr>
<tr>
<td>028</td>
<td>17P45624-00</td>
<td>1001</td>
<td>CAP, ALL.</td>
<td>22uF, 16V</td>
</tr>
<tr>
<td>029</td>
<td>17P45605-00</td>
<td>1001</td>
<td>4.7uF</td>
<td>C36</td>
</tr>
<tr>
<td>030</td>
<td>17P45606-00</td>
<td>1001</td>
<td>47uF, 10V</td>
<td>C2, 33</td>
</tr>
<tr>
<td>031</td>
<td>17P45609-00</td>
<td>1001</td>
<td>CAP, MYLAR</td>
<td>68NF +20%</td>
</tr>
<tr>
<td>032</td>
<td>17P46100-00</td>
<td>1001</td>
<td>CAP, CER, AXIAL</td>
<td>100PF +5%</td>
</tr>
<tr>
<td>033</td>
<td>17P46101-00</td>
<td>1001</td>
<td>CAP, CER, AXIAL</td>
<td>100PF +5%</td>
</tr>
<tr>
<td>034</td>
<td>17P46103-00</td>
<td>1001</td>
<td>10NF +20%</td>
<td>C29</td>
</tr>
<tr>
<td>035</td>
<td>17P46152-00</td>
<td>1001</td>
<td>1.5NF +10%</td>
<td>C10, 12, 18, 28</td>
</tr>
<tr>
<td>036</td>
<td>17P46330-00</td>
<td>1001</td>
<td>33pF +5%</td>
<td>C1, 37</td>
</tr>
<tr>
<td>037</td>
<td>17P46331-00</td>
<td>1001</td>
<td>33pF +5%</td>
<td>C4</td>
</tr>
<tr>
<td>038</td>
<td>17P46504-00</td>
<td>1001</td>
<td>100NF +80-20%</td>
<td>C38</td>
</tr>
</tbody>
</table>

LM CONTINUED
**Appendix A**  
**Z80 Instruction Set**

Presented here is the Z80A instruction set:

### Z80 Instruction Set

**Register Layout:**

<table>
<thead>
<tr>
<th>Main Register Set</th>
<th>Alternate Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accumulator</td>
<td>Flags</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>Accumulator'</td>
<td>Flags'</td>
</tr>
<tr>
<td>B'</td>
<td>C'</td>
</tr>
<tr>
<td>D'</td>
<td>E'</td>
</tr>
<tr>
<td>H'</td>
<td>L'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>interrupt vector I</th>
<th>memory refresh R</th>
</tr>
</thead>
<tbody>
<tr>
<td>index register IX</td>
<td></td>
</tr>
<tr>
<td>index register IY</td>
<td></td>
</tr>
<tr>
<td>stack pointer SP</td>
<td></td>
</tr>
<tr>
<td>program counter PC</td>
<td></td>
</tr>
</tbody>
</table>

*nnnn = hexadecimal 16-bit value*

*nn = hexadecimal 8-bit value*

*dd = 8-bit signed displacement*

*r = register*

*b = single bit*
<table>
<thead>
<tr>
<th>Z80 instruction</th>
<th>8080 instruction</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC A,(HL)</td>
<td>ADC M</td>
<td>adds byte at HL to A</td>
</tr>
<tr>
<td>ADC A,(IX+dd)</td>
<td>none</td>
<td>adds byte indexed by X to A</td>
</tr>
<tr>
<td>ADC A,(IY+dd)</td>
<td>none</td>
<td>adds byte indexed by Y to A</td>
</tr>
<tr>
<td>ADC A,r</td>
<td>ADC r</td>
<td>adds value in register to A</td>
</tr>
<tr>
<td>ADC A,nn</td>
<td>ADI nn</td>
<td>adds value to A</td>
</tr>
<tr>
<td>ADC HL,BC</td>
<td>none</td>
<td>adds BC to HL</td>
</tr>
<tr>
<td>ADC HL,DE</td>
<td>none</td>
<td>adds DE to HL</td>
</tr>
<tr>
<td>ADC HL,HL</td>
<td>none</td>
<td>doubles HL</td>
</tr>
<tr>
<td>ADC HL,SP</td>
<td>none</td>
<td>adds stack pointer to HL</td>
</tr>
<tr>
<td>ADD A,(HL)</td>
<td>ADD M</td>
<td>adds byte at HL to A</td>
</tr>
<tr>
<td>ADD A,(IX+dd)</td>
<td>none</td>
<td>adds byte indexed by X to A</td>
</tr>
<tr>
<td>ADD A, (IY+dd)</td>
<td>none</td>
<td>adds byte indexed by Y to A</td>
</tr>
<tr>
<td>ADD A,r</td>
<td>ADD r</td>
<td>adds value in register to A</td>
</tr>
<tr>
<td>ADD A,nn</td>
<td>ADI nn</td>
<td>adds value to A</td>
</tr>
<tr>
<td>ADD HL,BC</td>
<td>DAD B</td>
<td>adds BC to HL</td>
</tr>
<tr>
<td>ADD HL,DE</td>
<td>DAD D</td>
<td>adds DE to HL</td>
</tr>
<tr>
<td>ADD HL,HL</td>
<td>DAD H</td>
<td>doubles HL</td>
</tr>
<tr>
<td>ADD HL,SP</td>
<td>DAD SP</td>
<td>adds stack pointer to HL</td>
</tr>
<tr>
<td>ADD IX,BC</td>
<td>none</td>
<td>adds BC to X index</td>
</tr>
<tr>
<td>ADD IX,DE</td>
<td>none</td>
<td>adds DE to X index</td>
</tr>
<tr>
<td>ADD IX,IX</td>
<td>none</td>
<td>doubles X index</td>
</tr>
<tr>
<td>ADD IX,SP</td>
<td>none</td>
<td>adds stack pointer to X index</td>
</tr>
<tr>
<td>ADD IY,BC</td>
<td>none</td>
<td>adds BC to Y index</td>
</tr>
<tr>
<td>ADD IY,DE</td>
<td>none</td>
<td>adds DE to Y index</td>
</tr>
<tr>
<td>ADD IY,II</td>
<td>none</td>
<td>doubles Y index</td>
</tr>
<tr>
<td>ADD IY,SP</td>
<td>none</td>
<td>adds stack pointer to Y index</td>
</tr>
<tr>
<td>AND (HL)</td>
<td>ANA M</td>
<td>logical AND with byte and A</td>
</tr>
<tr>
<td>AND (IX+dd)</td>
<td>none</td>
<td>logical AND with index and A</td>
</tr>
<tr>
<td>AND (IY+dd)</td>
<td>none</td>
<td>logical AND with index and A</td>
</tr>
<tr>
<td>AND r</td>
<td>ANA r</td>
<td>logical AND with register and A</td>
</tr>
<tr>
<td>AND nn</td>
<td>ANI nn</td>
<td>logical AND with value and A</td>
</tr>
<tr>
<td>BIT b,(HL)</td>
<td>none</td>
<td>test bit of byte at HL</td>
</tr>
<tr>
<td>BIT b,(IX+dd)</td>
<td>none</td>
<td>test bit of byte at index X</td>
</tr>
<tr>
<td>BIT b,(IY+dd)</td>
<td>none</td>
<td>test bit of byte at index Y</td>
</tr>
<tr>
<td>BIT b,r</td>
<td>none</td>
<td>test bit of register value</td>
</tr>
<tr>
<td>CALL nnnn</td>
<td>CALL nnnn</td>
<td>subroutine call to location</td>
</tr>
<tr>
<td>CALL C,nnnn</td>
<td>CC nnnn</td>
<td>subroutine call if carry</td>
</tr>
<tr>
<td>CALL M,nnnn</td>
<td>CM nnnn</td>
<td>subroutine call if sign</td>
</tr>
<tr>
<td>CALL NC,nnnn</td>
<td>CNC nnnn</td>
<td>subroutine call if carry reset</td>
</tr>
<tr>
<td>CALL NZ,nnnn</td>
<td>CNZ nnnn</td>
<td>subroutine call if zero reset</td>
</tr>
<tr>
<td>CALL P,nnnn</td>
<td>CP nnnn</td>
<td>subroutine call if sign reset</td>
</tr>
<tr>
<td>CALL PE,nnnn</td>
<td>CPE nnnn</td>
<td>subroutine call if parity</td>
</tr>
<tr>
<td>CALL PO,nnnn</td>
<td>CPO nnnn</td>
<td>subroutine call if parity reset</td>
</tr>
<tr>
<td>CALL Z,nnnn</td>
<td>CZ nnnn</td>
<td>subroutine call if zero</td>
</tr>
<tr>
<td>CCF</td>
<td>CMC</td>
<td>complement carry flag</td>
</tr>
<tr>
<td>CP (HL)</td>
<td>CMP M</td>
<td>compare byte at HL to A</td>
</tr>
<tr>
<td>CP (IX+dd)</td>
<td>none</td>
<td>compare byte at X index to A</td>
</tr>
<tr>
<td>CP (IY+dd)</td>
<td>none</td>
<td>compare byte at Y index to A</td>
</tr>
<tr>
<td>CP r</td>
<td>CMP r</td>
<td>compare register value to A</td>
</tr>
<tr>
<td>CP nn</td>
<td>CPI nn</td>
<td>compare value to A</td>
</tr>
<tr>
<td>CPD</td>
<td>none</td>
<td>compare byte at HL and decrement BC</td>
</tr>
<tr>
<td>Z80 instruction</td>
<td>8080 instruction</td>
<td>description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>CPDR</td>
<td>none</td>
<td>compare byte at HL, decrement and repeat</td>
</tr>
<tr>
<td>CPI</td>
<td>none</td>
<td>compare byte at HL, increment BC</td>
</tr>
<tr>
<td>CPIR</td>
<td>none</td>
<td>compare byte at HL, increment and repeat</td>
</tr>
<tr>
<td>CPL</td>
<td>CMA</td>
<td>complement A</td>
</tr>
<tr>
<td>DAA</td>
<td>DAA</td>
<td>decimal adjust A</td>
</tr>
<tr>
<td>DEC (HL)</td>
<td>DCR M</td>
<td>decrement byte at HL</td>
</tr>
<tr>
<td>DEC (IX+dd)</td>
<td>none</td>
<td>decrement byte at index X</td>
</tr>
<tr>
<td>DEC (IY+dd)</td>
<td>none</td>
<td>decrement byte at index Y</td>
</tr>
<tr>
<td>DEC r</td>
<td>DCR r</td>
<td>decrement register</td>
</tr>
<tr>
<td>DEC BC</td>
<td>DCX B</td>
<td>decrement BC</td>
</tr>
<tr>
<td>DEC DE</td>
<td>DCX D</td>
<td>decrement DE</td>
</tr>
<tr>
<td>DEC HL</td>
<td>DCX H</td>
<td>decrement HL</td>
</tr>
<tr>
<td>DEC SP</td>
<td>DCX SP</td>
<td>decrement stack pointer</td>
</tr>
<tr>
<td>DEC IX</td>
<td>none</td>
<td>decrement X index</td>
</tr>
<tr>
<td>DEC IY</td>
<td>none</td>
<td>decrement Y index</td>
</tr>
<tr>
<td>DI</td>
<td>DI</td>
<td>disable interrupts</td>
</tr>
<tr>
<td>DJNZ dd</td>
<td>none</td>
<td>decrement B and jump relative</td>
</tr>
<tr>
<td>EI</td>
<td>EI</td>
<td>enable interrupts</td>
</tr>
<tr>
<td>EX (SP),HL</td>
<td>XTHL</td>
<td>exchange stack pointer with HL</td>
</tr>
<tr>
<td>EX (SP),IX</td>
<td>none</td>
<td>exchange stack pointer with X index</td>
</tr>
<tr>
<td>EX (SP),IY</td>
<td>none</td>
<td>exchange stack pointer with Y index</td>
</tr>
<tr>
<td>EX AF,AF</td>
<td>none</td>
<td>exchange AF register sets</td>
</tr>
<tr>
<td>EX DE,HL</td>
<td>XCHG</td>
<td>exchange DE and HL</td>
</tr>
<tr>
<td>EXX</td>
<td>none</td>
<td>exchange BC,DE,HL register sets</td>
</tr>
<tr>
<td>HALT</td>
<td>HLT</td>
<td>suspend operation</td>
</tr>
<tr>
<td>IM 0</td>
<td>none</td>
<td>set interrupt mode 0</td>
</tr>
<tr>
<td>IM 1</td>
<td>none</td>
<td>set interrupt mode 1</td>
</tr>
<tr>
<td>IM 2</td>
<td>none</td>
<td>set interrupt mode 2</td>
</tr>
<tr>
<td>IN r(C)</td>
<td>none</td>
<td>input byte from C port to register</td>
</tr>
<tr>
<td>IN A,(nn)</td>
<td>IN nn</td>
<td>input byte from port to A</td>
</tr>
<tr>
<td>INC (HL)</td>
<td>INR M</td>
<td>increment byte at HL</td>
</tr>
<tr>
<td>INC (IX+dd)</td>
<td>none</td>
<td>increment byte at X index</td>
</tr>
<tr>
<td>INC (IY+dd)</td>
<td>none</td>
<td>increment byte at Y index</td>
</tr>
<tr>
<td>INC r</td>
<td>INR r</td>
<td>increment byte in register</td>
</tr>
<tr>
<td>INC BE</td>
<td>INX B</td>
<td>increment BC</td>
</tr>
<tr>
<td>INC DE</td>
<td>INX D</td>
<td>increment DE</td>
</tr>
<tr>
<td>INC HL</td>
<td>INX H</td>
<td>increment HL</td>
</tr>
<tr>
<td>INC SP</td>
<td>INX SP</td>
<td>decrement stack pointer</td>
</tr>
<tr>
<td>INC IX</td>
<td>none</td>
<td>increment X index</td>
</tr>
<tr>
<td>INC IY</td>
<td>none</td>
<td>increment Y index</td>
</tr>
<tr>
<td>IND</td>
<td>none</td>
<td>input from C port to HL byte, decrement B, increment HL</td>
</tr>
<tr>
<td>INDR</td>
<td>none</td>
<td>input from C port to HL byte, decrement B, increment HL, repeat</td>
</tr>
<tr>
<td>INI</td>
<td>none</td>
<td>input from C port to HL byte, decrement B, decrement HL</td>
</tr>
<tr>
<td>INIR</td>
<td>none</td>
<td>input from C port to HL byte, decrement B, decrement HL, repeat</td>
</tr>
<tr>
<td>JP (HL)</td>
<td>PCHL</td>
<td>copy HL to PC then jump to it</td>
</tr>
<tr>
<td>JP (IX)</td>
<td>none</td>
<td>copy X index to PC then jump</td>
</tr>
<tr>
<td>Z80 instruction</td>
<td>8080 instruction</td>
<td>description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>JP (IY)</td>
<td>none</td>
<td>copy Y index to PC then jump</td>
</tr>
<tr>
<td>JP nnnn</td>
<td>JMP nnnn</td>
<td>jump to location</td>
</tr>
<tr>
<td>JP C,nnnn</td>
<td>JC nnnn</td>
<td>jump if carry</td>
</tr>
<tr>
<td>JP M,nnnn</td>
<td>JM nnnn</td>
<td>jump if sign</td>
</tr>
<tr>
<td>JP NC,nnnn</td>
<td>JNC nnnn</td>
<td>jump if no carry</td>
</tr>
<tr>
<td>JP NZ,nnnn</td>
<td>JNZ nnnn</td>
<td>jump if not zero</td>
</tr>
<tr>
<td>JP P,nnnn</td>
<td>JP nnnn</td>
<td>jump if sign reset</td>
</tr>
<tr>
<td>JP PE,nnnn</td>
<td>JPE nnnn</td>
<td>jump if parity</td>
</tr>
<tr>
<td>JP PO,nnnn</td>
<td>JPO nnnn</td>
<td>jump if parity reset</td>
</tr>
<tr>
<td>JP Z,nnnn</td>
<td>JZ nnnn</td>
<td>jump if zero</td>
</tr>
<tr>
<td>JR dd</td>
<td>none</td>
<td>jump relative using value</td>
</tr>
<tr>
<td>JR C,dd</td>
<td>none</td>
<td>jump relative if carry</td>
</tr>
<tr>
<td>JR NC,dd</td>
<td>none</td>
<td>jump relative if no carry</td>
</tr>
<tr>
<td>JR NZ,dd</td>
<td>none</td>
<td>jump relative if not zero</td>
</tr>
<tr>
<td>JR Z,dd</td>
<td>none</td>
<td>jump relative if zero</td>
</tr>
<tr>
<td>LD (BC),A</td>
<td>STAX B</td>
<td>move A to byte at BC</td>
</tr>
<tr>
<td>LD (DE),A</td>
<td>STAX D</td>
<td>move A to byte at DE</td>
</tr>
<tr>
<td>LD (HL),r</td>
<td>MOV M,r</td>
<td>move byte in register to byte at HL</td>
</tr>
<tr>
<td>LD (HL),nn</td>
<td>MVI M,nn</td>
<td>move value to byte at HL</td>
</tr>
<tr>
<td>LD (IX+dd),r</td>
<td>none</td>
<td>move byte into byte at indexed location</td>
</tr>
<tr>
<td>LD (IX+dd),nn</td>
<td>none</td>
<td>move byte into byte at indexed location</td>
</tr>
<tr>
<td>LD (IY+dd),r</td>
<td>none</td>
<td>move byte into byte at indexed location</td>
</tr>
<tr>
<td>LD (IY+dd),nn</td>
<td>none</td>
<td>move byte into byte at indexed location</td>
</tr>
<tr>
<td>LD (nnnn),A</td>
<td>STA nnnn</td>
<td>move A to location</td>
</tr>
<tr>
<td>LD (nnnn),BC</td>
<td>none</td>
<td>move BC to location and location + 1</td>
</tr>
<tr>
<td>LD (nnnn),DE</td>
<td>none</td>
<td>move DE to location and location + 1</td>
</tr>
<tr>
<td>LD (nnnn),HL</td>
<td>ShLD nnnn</td>
<td>move HL to location and location + 1</td>
</tr>
<tr>
<td>LD (nnnn),IX</td>
<td>none</td>
<td>move IX to location and location + 1</td>
</tr>
<tr>
<td>LD (nnnn),IY</td>
<td>none</td>
<td>move IY to location and location + 1</td>
</tr>
<tr>
<td>LD (nnnn),SP</td>
<td>none</td>
<td>move stack pointer to location and location + 1</td>
</tr>
<tr>
<td>LD A,(BC)</td>
<td>LDAX B</td>
<td>move byte at BC to A</td>
</tr>
<tr>
<td>LD A,(DE)</td>
<td>LDAX D</td>
<td>move byte at DE to A</td>
</tr>
<tr>
<td>LD A,I</td>
<td>none</td>
<td>move interrupt vector register to A</td>
</tr>
<tr>
<td>LD A,R</td>
<td>none</td>
<td>move memory refresh register to A</td>
</tr>
<tr>
<td>LD I,A</td>
<td>none</td>
<td>move A to interrupt vector register</td>
</tr>
<tr>
<td>LD R,A</td>
<td>none</td>
<td>move A to memory refresh register</td>
</tr>
<tr>
<td>LD r,(HL)</td>
<td>MOV r,M</td>
<td>move byte at HL to register</td>
</tr>
<tr>
<td>LD r,(IX+dd)</td>
<td>none</td>
<td>move byte at IX to register</td>
</tr>
<tr>
<td>LD r,(IY+dd)</td>
<td>none</td>
<td>move byte at IY to register</td>
</tr>
<tr>
<td>LD r,r</td>
<td>MOV r,r</td>
<td>move byte from register to register</td>
</tr>
<tr>
<td>LD r,nn</td>
<td>MVI r,nn</td>
<td>move value to register</td>
</tr>
<tr>
<td>LD A,(nnnn)</td>
<td>LDA nnnn</td>
<td>load A from location</td>
</tr>
<tr>
<td>LD BC,(nnnn)</td>
<td>none</td>
<td>load BC from locations</td>
</tr>
<tr>
<td>LD DE,(nnnn)</td>
<td>none</td>
<td>load DE from locations</td>
</tr>
<tr>
<td>LD HL,(nnnn)</td>
<td>LHLD nnnn</td>
<td>load HL from locations</td>
</tr>
<tr>
<td>LD BC,nnnn</td>
<td>LXI B,nnnn</td>
<td>load BC with value</td>
</tr>
<tr>
<td>LD DE,nnnn</td>
<td>LXI D,nnnn</td>
<td>load DE with value</td>
</tr>
<tr>
<td>LD HL,nnnn</td>
<td>LXI H,nnnn</td>
<td>load HL with value</td>
</tr>
<tr>
<td>LD SP,nnnn</td>
<td>LXI SP,nnnn</td>
<td>load stack pointer with value</td>
</tr>
<tr>
<td>LD IX,nnnn</td>
<td>none</td>
<td>load IX with value</td>
</tr>
<tr>
<td>LD IY,nnnn</td>
<td>none</td>
<td>load IY with value</td>
</tr>
<tr>
<td>LD IX,(nnnn)</td>
<td>none</td>
<td>load IX from locations</td>
</tr>
<tr>
<td>Z80 instruction</td>
<td>8080 instruction</td>
<td>description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>LD IY,(nnnn)</td>
<td>none</td>
<td>load IY from locations</td>
</tr>
<tr>
<td>LD SP,(nnnn)</td>
<td>none</td>
<td>load stack pointer from locations</td>
</tr>
<tr>
<td>LD SP,HL</td>
<td>SPHL</td>
<td>load stack pointer from HL</td>
</tr>
<tr>
<td>LD SP,IX</td>
<td>none</td>
<td>load stack pointer from IX</td>
</tr>
<tr>
<td>LD SP,IY</td>
<td>none</td>
<td>load stack pointer from IY</td>
</tr>
<tr>
<td>LDD</td>
<td>none</td>
<td>move byte at HL to DE location, decrement BC, decrement HL and DE</td>
</tr>
<tr>
<td>LDDR</td>
<td>none</td>
<td>move byte at HL to DE location, decrement BC, decrement HL and DE, repeat</td>
</tr>
<tr>
<td>LDI</td>
<td>none</td>
<td>move byte at HL to DE location, decrement BC, increment HL and DE</td>
</tr>
<tr>
<td>LDIR</td>
<td>none</td>
<td>move byte at HL to DE location, decrement BC, increment HL and DE, repeat</td>
</tr>
<tr>
<td>NEG</td>
<td>none</td>
<td>two's complement of A</td>
</tr>
<tr>
<td>NOP</td>
<td>NOP</td>
<td>no operation</td>
</tr>
<tr>
<td>OR (HL)</td>
<td>ORA M</td>
<td>logical OR of A and byte at HL</td>
</tr>
<tr>
<td>OR (IX+dd)</td>
<td>none</td>
<td>logical OR of A and byte at IX</td>
</tr>
<tr>
<td>OR (IY+dd)</td>
<td>none</td>
<td>logical OR of A and byte at IY</td>
</tr>
<tr>
<td>OR r</td>
<td>ORA r</td>
<td>logical OR of A and register</td>
</tr>
<tr>
<td>OR nn</td>
<td>ORI nn</td>
<td>logical OR of A and value</td>
</tr>
<tr>
<td>OTDR</td>
<td>none</td>
<td>output byte at HL to port C, decrement B, decrement HL, repeat</td>
</tr>
<tr>
<td>OTIR</td>
<td>none</td>
<td>output byte at HL to port C, decrement B, increment HL, repeat</td>
</tr>
<tr>
<td>OUT (C),r</td>
<td>none</td>
<td>output byte in register to port C</td>
</tr>
<tr>
<td>OUT (nn),A</td>
<td>OUT nn</td>
<td>output value to port A</td>
</tr>
<tr>
<td>OUTD</td>
<td>none</td>
<td>output byte from HL to port C, decrement B, decrement HL</td>
</tr>
<tr>
<td>OUTI</td>
<td>none</td>
<td>output byte from HL to port C, decrement B, increment HL</td>
</tr>
<tr>
<td>POP AF</td>
<td>POP PSW</td>
<td>restore AF from stack</td>
</tr>
<tr>
<td>POP BC</td>
<td>POP B</td>
<td>restore BC from stack</td>
</tr>
<tr>
<td>POP DE</td>
<td>POP D</td>
<td>restore DE from stack</td>
</tr>
<tr>
<td>POP HL</td>
<td>POP H</td>
<td>restore HL from stack</td>
</tr>
<tr>
<td>POP IX</td>
<td>none</td>
<td>restore IX from stack</td>
</tr>
<tr>
<td>POP IY</td>
<td>none</td>
<td>restore IY from stack</td>
</tr>
<tr>
<td>PUSH AF</td>
<td>PUSH PSW</td>
<td>store AF on stack</td>
</tr>
<tr>
<td>PUSH BC</td>
<td>PUSH B</td>
<td>store BC on stack</td>
</tr>
<tr>
<td>PUSH DE</td>
<td>PUSH D</td>
<td>store DE on stack</td>
</tr>
<tr>
<td>PUSH HL</td>
<td>PUSH H</td>
<td>store HL on stack</td>
</tr>
<tr>
<td>PUSH IX</td>
<td>none</td>
<td>store IX on stack</td>
</tr>
<tr>
<td>PUSH IY</td>
<td>none</td>
<td>store IY on stack</td>
</tr>
<tr>
<td>RES b,(IX+dd)</td>
<td>none</td>
<td>reset bit in byte at IX</td>
</tr>
<tr>
<td>RES b,(IY+dd)</td>
<td>none</td>
<td>reset bit in byte at IY</td>
</tr>
<tr>
<td>RES b,r</td>
<td>none</td>
<td>reset bit in register</td>
</tr>
<tr>
<td>RET</td>
<td>RET</td>
<td>return from subroutine</td>
</tr>
<tr>
<td>Z80 instruction</td>
<td>8080 instruction</td>
<td>description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>RET C</td>
<td>RC</td>
<td>return if carry</td>
</tr>
<tr>
<td>RET M</td>
<td>RM</td>
<td>return if sign</td>
</tr>
<tr>
<td>RET NC</td>
<td>RNC</td>
<td>return if no carry</td>
</tr>
<tr>
<td>RET NZ</td>
<td>RNZ</td>
<td>return if not zero</td>
</tr>
<tr>
<td>RET P</td>
<td>RP</td>
<td>return if sign reset</td>
</tr>
<tr>
<td>RET PE</td>
<td>RPE</td>
<td>return if parity</td>
</tr>
<tr>
<td>RET PO</td>
<td>RPO</td>
<td>return if parity reset</td>
</tr>
<tr>
<td>RET Z</td>
<td>RZ</td>
<td>return if zero</td>
</tr>
<tr>
<td>RETI</td>
<td>none</td>
<td>return from maskable interrupt</td>
</tr>
<tr>
<td>RETN</td>
<td>none</td>
<td>return from nonmaskable interrupt</td>
</tr>
<tr>
<td>RL (HL)</td>
<td>none</td>
<td>rotate byte at HL left with carry</td>
</tr>
<tr>
<td>RL (IX+dd)</td>
<td>none</td>
<td>rotate byte at IX left with carry</td>
</tr>
<tr>
<td>RL (IY+dd)</td>
<td>none</td>
<td>rotate byte at IY left with carry</td>
</tr>
<tr>
<td>RL r</td>
<td>none</td>
<td>rotate byte in register left with carry</td>
</tr>
<tr>
<td>RLA</td>
<td>RAL</td>
<td>rotate byte in A left with carry</td>
</tr>
<tr>
<td>RLC (HL)</td>
<td>none</td>
<td>rotate byte at HL circularly left</td>
</tr>
<tr>
<td>RLC (IX+dd)</td>
<td>none</td>
<td>rotate byte at IX circularly left</td>
</tr>
<tr>
<td>RLC (IY+dd)</td>
<td>none</td>
<td>rotate byte at IY circularly left</td>
</tr>
<tr>
<td>RLC r</td>
<td>none</td>
<td>rotate byte in register circularly left</td>
</tr>
<tr>
<td>RLC A</td>
<td>RLC</td>
<td>rotate byte in A circularly left</td>
</tr>
<tr>
<td>RLD</td>
<td>none</td>
<td>rotate 12 bits n HL 4 bits at time left</td>
</tr>
<tr>
<td>RR (HL)</td>
<td>none</td>
<td>rotate byte at HL right with carry</td>
</tr>
<tr>
<td>RR (IX+dd)</td>
<td>none</td>
<td>rotate byte at IX right with carry</td>
</tr>
<tr>
<td>RR (IY+dd)</td>
<td>none</td>
<td>rotate byte at IY right with carry</td>
</tr>
<tr>
<td>RR r</td>
<td>none</td>
<td>rotate byte in register right with carry</td>
</tr>
<tr>
<td>RRA</td>
<td>RAR</td>
<td>rotate accumulator right</td>
</tr>
<tr>
<td>RRC (HL)</td>
<td>none</td>
<td>rotate byte in HL circularly right</td>
</tr>
<tr>
<td>RRC (IX+dd)</td>
<td>none</td>
<td>rotate byte in IX circularly right</td>
</tr>
<tr>
<td>RRC (IY+dd)</td>
<td>none</td>
<td>rotate byte in IY circularly right</td>
</tr>
<tr>
<td>RRC r</td>
<td>none</td>
<td>rotate byte in register circularly right</td>
</tr>
<tr>
<td>RRC A</td>
<td>RRC</td>
<td>rotate byte in A circularly right</td>
</tr>
<tr>
<td>RRD</td>
<td>none</td>
<td>rotate 12 bits in HL 4 bits at time right</td>
</tr>
<tr>
<td>RST 00h</td>
<td>RST 0</td>
<td>first restart location</td>
</tr>
<tr>
<td>RST 08h</td>
<td>RST 1</td>
<td>second restart location</td>
</tr>
<tr>
<td>RST 10h</td>
<td>RST 2</td>
<td>third restart location</td>
</tr>
<tr>
<td>RST 18h</td>
<td>RST 3</td>
<td>fourth restart location</td>
</tr>
<tr>
<td>RST 20h</td>
<td>RST 4</td>
<td>fifth restart location</td>
</tr>
<tr>
<td>RST 28h</td>
<td>RST 5</td>
<td>sixth restart location</td>
</tr>
<tr>
<td>RST 30h</td>
<td>RST 6</td>
<td>seventh restart location</td>
</tr>
<tr>
<td>RST 38h</td>
<td>RST 7</td>
<td>eighth restart location</td>
</tr>
<tr>
<td>SBC A,(HL)</td>
<td>SBB M</td>
<td>subtract byte at HL from A</td>
</tr>
<tr>
<td>SBC A,(IX+dd)</td>
<td>none</td>
<td>subtract byte at IX from A</td>
</tr>
<tr>
<td>SBC A,(IY+dd)</td>
<td>none</td>
<td>subtract byte at IY from A</td>
</tr>
<tr>
<td>SBC A,r</td>
<td>SBB r</td>
<td>subtract byte in register from A</td>
</tr>
<tr>
<td>SBC A,nn</td>
<td>SBI nn</td>
<td>subtract value from A</td>
</tr>
<tr>
<td>SBC HL,BC</td>
<td>none</td>
<td>subtract BC from HL</td>
</tr>
<tr>
<td>SBC HL,DE</td>
<td>none</td>
<td>subtract DE from HL</td>
</tr>
<tr>
<td>SBC HL,HLS</td>
<td>none</td>
<td>subtract HL from HL</td>
</tr>
<tr>
<td>SBC HL,SP</td>
<td>none</td>
<td>subtract SP from HL</td>
</tr>
<tr>
<td>SCF</td>
<td>STC</td>
<td>set carry flag</td>
</tr>
<tr>
<td>SET b,(HL)</td>
<td>none</td>
<td>set big in byte at HL</td>
</tr>
<tr>
<td>SET b,(IX+dd)</td>
<td>none</td>
<td>set bit in byte at IX</td>
</tr>
<tr>
<td>SET b,(IY+dd)</td>
<td>none</td>
<td>set bit in byte at IY</td>
</tr>
<tr>
<td>Z80 instruction</td>
<td>8080 instruction</td>
<td>description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SET b,r</td>
<td>none</td>
<td>set bit in byte in register</td>
</tr>
<tr>
<td>SLA (HL)</td>
<td>none</td>
<td>arithmetic shift left on byte at HL</td>
</tr>
<tr>
<td>SLA (IX+dd)</td>
<td>none</td>
<td>arithmetic shift left on byte at IX</td>
</tr>
<tr>
<td>SLA (IY+dd)</td>
<td>none</td>
<td>arithmetic shift left on byte at IY</td>
</tr>
<tr>
<td>SLA r</td>
<td>none</td>
<td>arithmetic shift left on byte in register</td>
</tr>
<tr>
<td>SRA (HL)</td>
<td>none</td>
<td>arithmetic shift right on byte at HL</td>
</tr>
<tr>
<td>SRA (IX+dd)</td>
<td>none</td>
<td>arithmetic shift right on byte at IX</td>
</tr>
<tr>
<td>SRA (IY+dd)</td>
<td>none</td>
<td>arithmetic shift right on byte at IY</td>
</tr>
<tr>
<td>SRA r</td>
<td>none</td>
<td>arithmetic shift right on byte in register</td>
</tr>
<tr>
<td>SRL (HL)</td>
<td>none</td>
<td>logical shift right on byte at HL</td>
</tr>
<tr>
<td>SRL (IX+dd)</td>
<td>none</td>
<td>logical shift right on byte at IX</td>
</tr>
<tr>
<td>SRL (IY+dd)</td>
<td>none</td>
<td>logical shift right on byte at IY</td>
</tr>
<tr>
<td>SRL r</td>
<td>none</td>
<td>logical shift right on byte in register</td>
</tr>
<tr>
<td>SUB (HL)</td>
<td>SUB M</td>
<td>subtract byte at HL from A</td>
</tr>
<tr>
<td>SUB (IX+dd)</td>
<td>none</td>
<td>subtract byte at IX from A</td>
</tr>
<tr>
<td>SUB (IY+dd)</td>
<td>none</td>
<td>subtract byte at IY from A</td>
</tr>
<tr>
<td>SUB r</td>
<td>SUB r</td>
<td>subtract byte in register from A</td>
</tr>
<tr>
<td>SUB nn</td>
<td>SUI nn</td>
<td>subtract value from A</td>
</tr>
<tr>
<td>XOR (HL)</td>
<td>XRA M</td>
<td>XOR HL register</td>
</tr>
<tr>
<td>XOR (IX+dd)</td>
<td>none</td>
<td>XOR IX register</td>
</tr>
<tr>
<td>XOR (IY+dd)</td>
<td>none</td>
<td>XOR IY register</td>
</tr>
<tr>
<td>XOR r</td>
<td>XRA r</td>
<td>XOR register</td>
</tr>
<tr>
<td>XOR nn</td>
<td>XRI nn</td>
<td>XOR value with A</td>
</tr>
</tbody>
</table>
Appendix B
6821 PIA Registers/
Instructions

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

MAXIMUM RATINGS

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>VCC</td>
<td>0.3 to +7.0</td>
<td>V</td>
</tr>
<tr>
<td>Input Voltage</td>
<td>Vin</td>
<td>0.3 to +7.0</td>
<td>V</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>TA</td>
<td>T_L to T_H</td>
<td>°C</td>
</tr>
<tr>
<td>MC6821, MC68A21, MC68B21</td>
<td></td>
<td>0 to 70</td>
<td>°C</td>
</tr>
<tr>
<td>MC6821C, MC68A21C, MC68B21C</td>
<td></td>
<td>-40 to +85</td>
<td>°C</td>
</tr>
<tr>
<td>Storage Temperature Range</td>
<td>Tstg</td>
<td>-56 to +150</td>
<td>°C</td>
</tr>
</tbody>
</table>

THERMAL CHARACTERISTICS

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal Resistance</td>
<td>θJA</td>
<td>50</td>
<td>°C/W</td>
</tr>
<tr>
<td>Ceramic</td>
<td></td>
<td>100</td>
<td>°C/W</td>
</tr>
<tr>
<td>Plastic</td>
<td></td>
<td>60</td>
<td>°C/W</td>
</tr>
<tr>
<td>Cerdip</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 VSS or VCC).
POWER CONSIDERATIONS

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

\[
T_J = T_A + (P_D \cdot \theta_JA)
\]

(1)

Where:
- \( T_A \) = Ambient Temperature, °C
- \( \theta_JA \) = Package Thermal Resistance, Junction-to-Ambient, °C/W
- \( P_D \) = \( P_{INT} + P_{PORT} \)
- \( P_{INT} \) = \( I_{CC} \cdot V_{CC} \), 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:

\[
P_D = K \cdot (T_J + 273°C)
\]

(2)

Solving equations 1 and 2 for \( K \) gives:

\[
K = P_D \cdot (T_A + 273°C) + \theta_JA \cdot P_D^2
\]

(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 (\( V_{CC} = 5.0 \text{ Vdc} \pm 5\% \), \( V_{SS} = 0 \), \( T_A = T_L \) to \( T_H \) unless otherwise noted).

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BUS CONTROL INPUTS</strong> (R/W, Enable, RESET, RS0, RS1, CS0, CS1, CS2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input High Voltage</td>
<td>( V_{IH} )</td>
<td>( V_{SS} + 2.0 )</td>
<td>–</td>
<td>( V_{CC} )</td>
<td>V</td>
</tr>
<tr>
<td>Input Low Voltage</td>
<td>( V_{IL} )</td>
<td>( V_{SS} - 0.3 )</td>
<td>–</td>
<td>( V_{SS} + 0.8 )</td>
<td>V</td>
</tr>
<tr>
<td>Input Leakage Current (( V_{IN} = 0 ) to 5.25 V)</td>
<td>( I_{IN} )</td>
<td>–</td>
<td>1.0</td>
<td>2.5</td>
<td>( \mu )A</td>
</tr>
<tr>
<td>Capacitance (( V_{IN} = 0 ), ( T_A = 25°C ), ( f = 1.0 \text{ MHz} ))</td>
<td>( C_{IN} )</td>
<td>–</td>
<td>7.5</td>
<td></td>
<td>pF</td>
</tr>
<tr>
<td><strong>INTERRUPT OUTPUTS</strong> (IRQA, IRQB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Low Voltage (( I_{LOAD} = 3.2 \text{ mA} ))</td>
<td>( V_{OL} )</td>
<td>–</td>
<td>–</td>
<td>( V_{SS} + 0.4 )</td>
<td>V</td>
</tr>
<tr>
<td>Three-State Output Leakage Current</td>
<td>( I_{OZ} )</td>
<td>–</td>
<td>1.0</td>
<td>10</td>
<td>( \mu )A</td>
</tr>
<tr>
<td>Capacitance (( V_{IN} = 0 ), ( T_A = 25°C ), ( f = 1.0 \text{ MHz} ))</td>
<td>( C_{OUT} )</td>
<td>–</td>
<td>5.0</td>
<td></td>
<td>pF</td>
</tr>
<tr>
<td><strong>DATA BUS</strong> (D0-D7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input High Voltage</td>
<td>( V_{IH} )</td>
<td>( V_{SS} + 2.0 )</td>
<td>–</td>
<td>( V_{CC} )</td>
<td>V</td>
</tr>
<tr>
<td>Input Low Voltage</td>
<td>( V_{IL} )</td>
<td>( V_{SS} - 0.3 )</td>
<td>–</td>
<td>( V_{SS} + 0.8 )</td>
<td>V</td>
</tr>
<tr>
<td>Three-State Input Leakage Current (( V_{IN} = 0.4 ) to 2.4 V)</td>
<td>( I_{IZ} )</td>
<td>–</td>
<td>2.0</td>
<td>10</td>
<td>( \mu )A</td>
</tr>
<tr>
<td>Output High Voltage (( I_{LOAD} = -200 \mu A ))</td>
<td>( V_{OH} )</td>
<td>( V_{SS} + 2.4 )</td>
<td>–</td>
<td>–</td>
<td>V</td>
</tr>
<tr>
<td>Output Low Voltage (( I_{LOAD} = 1.6 \text{ mA} ))</td>
<td>( V_{OL} )</td>
<td>–</td>
<td>–</td>
<td>( V_{SS} + 0.4 )</td>
<td>V</td>
</tr>
<tr>
<td>Capacitance (( V_{IN} = 0 ), ( T_A = 25°C ), ( f = 1.0 \text{ MHz} ))</td>
<td>( C_{IN} )</td>
<td>–</td>
<td>12.5</td>
<td></td>
<td>pF</td>
</tr>
<tr>
<td><strong>PERIPHERAL BUS</strong> (PA0-PA7, PB0-PB7, CA1, CA2, CB1, CB2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Leakage Current (( V_{IN} = 0 ) to 5.25 V)</td>
<td>( I_{IN} )</td>
<td>–</td>
<td>1.0</td>
<td>2.5</td>
<td>( \mu )A</td>
</tr>
<tr>
<td>Three-State Input Leakage Current (( V_{IN} = 0.4 ) to 2.4 V)</td>
<td>( I_{IZ} )</td>
<td>–</td>
<td>2.0</td>
<td>10</td>
<td>( \mu )A</td>
</tr>
<tr>
<td>Input High Current (( V_{IH} = 2.4 \text{ V} ))</td>
<td>( I_{IH} )</td>
<td>–</td>
<td>200</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>Darlington Drive Current (( V_O = 1.5 \text{ V} ))</td>
<td>( I_{OH} )</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>Input Low Current (( V_{IL} = 0.4 \text{ V} ))</td>
<td>( I_{IL} )</td>
<td>–</td>
<td>–</td>
<td>1.3</td>
<td>2.4</td>
</tr>
<tr>
<td>Output High Voltage (( I_{LOAD} = -200 \mu A ))</td>
<td>( V_{OH} )</td>
<td>( V_{SS} + 2.4 )</td>
<td>–</td>
<td>–</td>
<td>V</td>
</tr>
<tr>
<td>Output Low Voltage (( I_{LOAD} = 3.2 \text{ mA} ))</td>
<td>( V_{OL} )</td>
<td>–</td>
<td>–</td>
<td>( V_{SS} + 0.4 )</td>
<td>V</td>
</tr>
<tr>
<td>Capacitance (( V_{IN} = 0 ), ( T_A = 25°C ), ( f = 1.0 \text{ MHz} ))</td>
<td>( C_{IN} )</td>
<td>–</td>
<td>–</td>
<td>10</td>
<td>pF</td>
</tr>
<tr>
<td><strong>POWER REQUIREMENTS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Power Dissipation (Measured at ( T_A = T_L ))</td>
<td>( P_{INT} )</td>
<td>–</td>
<td>–</td>
<td>560</td>
<td>mW</td>
</tr>
<tr>
<td>Id. Number</td>
<td>Characteristic</td>
<td>Symbol</td>
<td>MC6821 Min</td>
<td>MC6821 Max</td>
<td>MC68A21 Min</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------</td>
<td>--------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>1</td>
<td>Cycle Time</td>
<td>t_Cyc</td>
<td>1.0</td>
<td>10</td>
<td>0.67</td>
</tr>
<tr>
<td>2</td>
<td>Pulse Width, E Low</td>
<td>PW_EL</td>
<td>430</td>
<td></td>
<td>280</td>
</tr>
<tr>
<td>3</td>
<td>Pulse Width, E High</td>
<td>PW_EH</td>
<td>450</td>
<td></td>
<td>280</td>
</tr>
<tr>
<td>4</td>
<td>Clock Rise and Fall Time</td>
<td>t_H, t_L</td>
<td></td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Address Hold Time</td>
<td>t_AH</td>
<td>10</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>Address Setup Time Before E</td>
<td>t_AS</td>
<td>80</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>14</td>
<td>Chip Select Setup Time Before E</td>
<td>t_CS</td>
<td>80</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>15</td>
<td>Chip Select Hold Time</td>
<td>t_CH</td>
<td>10</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>18</td>
<td>Read Data Hold Time</td>
<td>t_HR</td>
<td>20</td>
<td>50*</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>Write Data Hold Time</td>
<td>t_HW</td>
<td>10</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>Output Data Delay Time</td>
<td>t_DDR</td>
<td></td>
<td>290</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Input Data Setup Time</td>
<td>t_DSW</td>
<td>165</td>
<td></td>
<td>80</td>
</tr>
</tbody>
</table>

*The data bus output buffers are no longer sourcing or sinking current by t\_HR\_max (High Impedance).

**FIGURE 1 — BUS TIMING**

**Notes:**
1. Voltage levels shown are V\_L ≤ 0.4 V, V\_H ≥ 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<sub>CC</sub>=5.0 V ± 5%, V<sub>SS</sub>=0 V, T<sub>A</sub>=T<sub>L</sub> to T<sub>H</sub> unless otherwise specified)

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Symbol</th>
<th>MC6821</th>
<th>MC68A21</th>
<th>MC68B21</th>
<th>Unit</th>
<th>Reference Fig. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Setup Time</td>
<td>t&lt;sub&gt;PDS&lt;/sub&gt;</td>
<td>200 – 135</td>
<td>100 – ns</td>
<td>0 – ns</td>
<td>–</td>
<td>6</td>
</tr>
<tr>
<td>Data Hold Time</td>
<td>t&lt;sub&gt;PDH&lt;/sub&gt;</td>
<td>0 – 0</td>
<td>0 – 0</td>
<td>0 – 0</td>
<td>–</td>
<td>6</td>
</tr>
<tr>
<td>Delay Time, Enable Negative Transition to CA2 Negative Transition</td>
<td>t&lt;sub&gt;CA2&lt;/sub&gt;</td>
<td>1.0 – 0.670</td>
<td>0.500 μs</td>
<td>3, 7, 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay Time, Enable Negative Transition to CA2 Positive Transition</td>
<td>t&lt;sub&gt;Rs1&lt;/sub&gt;</td>
<td>1.0 – 0.670</td>
<td>0.500 μs</td>
<td>3, 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rise and Fall Times for CA1 and CA2 Input Signals</td>
<td>t&lt;sub&gt;r, tf&lt;/sub&gt;</td>
<td>1.0 – 1.0</td>
<td>1.0 μs</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay Time from CA1 Active Transition to CA2 Positive Transition</td>
<td>t&lt;sub&gt;Rs2&lt;/sub&gt;</td>
<td>2.0 – 1.35</td>
<td>1.0 μs</td>
<td>3, 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay Time, Enable Negative Transition to Data Valid</td>
<td>t&lt;sub&gt;PDW&lt;/sub&gt;</td>
<td>1.0 – 0.670</td>
<td>0.5 μs</td>
<td>3, 9, 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay Time, Enable Negative Transition to CMOS Data Valid PA0-PA7, CA2</td>
<td>t&lt;sub&gt;CMOS&lt;/sub&gt;</td>
<td>2.0 – 1.35</td>
<td>1.0 μs</td>
<td>4, 9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay Time, Enable Positive Transition to CB2 Negative Transition</td>
<td>t&lt;sub&gt;Cb2&lt;/sub&gt;</td>
<td>1.0 – 0.670</td>
<td>0.5 μs</td>
<td>3, 11, 12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay Time, Data Valid to CB2 Negative Transition</td>
<td>t&lt;sub&gt;Dc&lt;/sub&gt;</td>
<td>20 – 20</td>
<td>20 – 20</td>
<td>ns</td>
<td>3, 10</td>
<td></td>
</tr>
<tr>
<td>Delay Time, Enable Positive Transition to CB2 Positive Transition</td>
<td>t&lt;sub&gt;Rs1&lt;/sub&gt;</td>
<td>1.0 – 0.670</td>
<td>0.5 μs</td>
<td>3, 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control Output Pulse Width, CA2/CB2</td>
<td>PW&lt;sub&gt;ct&lt;/sub&gt;</td>
<td>500 – 375</td>
<td>250 – ns</td>
<td>3, 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rise and Fall Time for CB1 and CB2 Input Signals</td>
<td>t&lt;sub&gt;r, tf&lt;/sub&gt;</td>
<td>1.0 – 1.0</td>
<td>1.0 μs</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay Time, CB1 Active Transition to CB2 Positive Transition</td>
<td>t&lt;sub&gt;Rs2&lt;/sub&gt;</td>
<td>2.0 – 1.35</td>
<td>1.0 μs</td>
<td>3, 12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt Release Time, IRQA and IRQB</td>
<td>t&lt;sub&gt;Ir&lt;/sub&gt;</td>
<td>1.80 – 1.10</td>
<td>0.85 μs</td>
<td>5, 14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt Response Time</td>
<td>t&lt;sub&gt;Rs3&lt;/sub&gt;</td>
<td>1.0 – 1.0</td>
<td>1.0 μs</td>
<td>5, 13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt Input Pulse Time</td>
<td>PW&lt;sub&gt;I&lt;/sub&gt;</td>
<td>500 – 500</td>
<td>500 – ns</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESET Low Time*</td>
<td>t&lt;sub&gt;IrL&lt;/sub&gt;</td>
<td>1.0 – 0.66</td>
<td>0.5 – μs</td>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*The RESET line must be high a minimum of 1.0 μs before addressing the PIA.

**FIGURE 2 — BUS TIMING TEST LOADS**

(D0-D7) 5.0 V

Test Point

C = 130 pF

R = 11.7 kΩ

MMO6150 or Equiv.

MMO7000 or Equiv.

**FIGURE 3 — TTL EQUIVALENT TEST LOAD**

(PA0-PA7, PB0-PB7, CA2, CB2)

5.0 V

R<sub>L</sub> = 2.4 kΩ

Test Point

V<sub>I</sub>

C = 30 pF, R = 12 k

**FIGURE 4 — CMOS EQUIVALENT TEST LOAD**

(PA0-PA7, CA2)

5.0 V

Test Point

R = 11.7 kΩ

**FIGURE 5 — NMOS EQUIVALENT TEST LOAD**

(IRG Only) 5.0 V

Test Point

R = 3 kΩ

MMO6150 or Equiv.
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.
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.

*The RESET line must be a V_H for a minimum of 1.0 μs before addressing the PIA.
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 (DO-D7) — The bidirectional data lines (DO-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/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.

RESET — The active low 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 CS2) — These three input signals are used to select the PIA. CS0 and CS1 must be high and 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 (IRQA and IRQB) — The active low Interrupt Request lines (IRQA and 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 of the active edge of the interrupt input signal to condition the edge sense circuitry. 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 properly 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 capability, 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 RESET has the effect of zeroing all PIA registers. This will set P0-P7, PB0-PB7, CA2 and CB2 as inputs, and all interrupts disabled. The PIA must be configured during the reset 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>
<thead>
<tr>
<th>TABLE 1 — INTERNAL ADDRESSING</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS1</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

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 Darlington's 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 IRQA and 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**

<table>
<thead>
<tr>
<th>Speed</th>
<th>Device</th>
<th>Temperature Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 MHz</td>
<td>MC6821P.L.S</td>
<td>0 to +70°C</td>
</tr>
<tr>
<td>1.5 MHz</td>
<td>MC6821CP,CL.CS</td>
<td>-40 to +85°C</td>
</tr>
<tr>
<td>2.0 MHz</td>
<td>MC6821CP,CL.CS</td>
<td>-40 to +85°C</td>
</tr>
<tr>
<td></td>
<td>MC68B21P.L.S</td>
<td>0 to +70°C</td>
</tr>
</tbody>
</table>

**BEETTER 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 TA max.
- Level 2 “D” = 168 hour Burn-in at 125°C
- Level 3 “DS” = Combination of Level 1 and 2.
Determine Active CA1 (CB1) Transition for Setting Interrupt Flag IRQA(B1) — (bit 7)

- \( b_1 = 0 \): IRQA(B1) set by high-to-low transition on CA1 (CB1)
- \( b_1 = 1 \): IRQA(B1) set by low-to-high transition on CA1 (CB1)

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

<table>
<thead>
<tr>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRQA(B1) Flag</td>
<td>IRQA(B) Flag</td>
<td>CA2 (CB2) Control</td>
<td>DDR Access</td>
<td>CA1 (CB1) Control</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 (\( b_5 = 1 \)): IRQA(B) 2=0, not affected by CA2 (CB2) transitions.

Determines Whether Data Direction Register Or Output Register is Addressed
\( b_2 = 0 \): Data Direction Register selected.
\( b_2 = 1 \): Output Register selected.

CA2 (CB2) Established as Output by \( b_5 = 1 \)
(Note that operation of CA2 and CB2 output functions are not identical)

- \( b_5 \ b_4 \ b_3 = 0 \): 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 = 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.

- \( b_5 \ b_4 \ b_3 = 1 \): 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 \( b_5 = 0 \)

- \( b_5 \ b_4 \ b_3 = 0 \): CA2 (CB2) Interrupt Request Enable/Disable
  b3 = 0: Enables IRQA(B) MPU Interrupt by CA2 (CB2) active transition.
  b3 = 1: Enables IRQA(B) MPU Interrupt by CA2 (CB2) active transition.

- \( b_5 \ b_4 \ b_3 = 1 \): 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:
ASYMMONOUS 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 MC6880L 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

MC6850 ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER 
BLOCK DIAGRAM

PIN ASSIGNMENT

©MOTOROLA INC., 1981
DS 9493-R2
MAXIMUM RATINGS

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>VCC</td>
<td>-0.3 to +7.0</td>
<td>V</td>
</tr>
<tr>
<td>Input Voltage</td>
<td>Vin</td>
<td>-0.3 to +7.0</td>
<td>V</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>TA</td>
<td>T_L to T_H</td>
<td>°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 to 70</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-40 to +85</td>
<td></td>
</tr>
<tr>
<td>Storage Temperature Range</td>
<td>T_stg</td>
<td>-55 to +150</td>
<td>°C</td>
</tr>
</tbody>
</table>

THERMAL CHARACTERISTICS

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal Resistance</td>
<td>θ_JA</td>
<td>120</td>
<td>°C/W</td>
</tr>
<tr>
<td>Plastic</td>
<td></td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Ceramic</td>
<td></td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>Cerdip</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

POWER CONSIDERATIONS

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

\[ T_J = T_A + (P_D \cdot \theta_JA) \]  

Where:

- \( T_A \) = Ambient Temperature, °C
- \( \theta_JA \) = Package Thermal Resistance, Junction-to-Ambient, °C/W
- \( P_D \) = P_INT + P_PORT
- P_INT = ICC \cdot V_CC, Watts - Chip Internal Power
- P_PORT = Port Power Dissipation, Watts - User Determined

For most applications P_PORT < 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:

\[ P_D = K \cdot (T_J + 273°C) \]  

Solving equations 1 and 2 for K gives:

\[ K = P_D \cdot (T_A + 273°C) + \theta_JA \cdot P_D \]  

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.)

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input High Voltage</td>
<td>VIH</td>
<td>VSS+2.0</td>
<td>-</td>
<td>VCC</td>
<td>V</td>
</tr>
<tr>
<td>Input Low Voltage</td>
<td>VIL</td>
<td>VSS-0.3</td>
<td>-</td>
<td>VSS+0.8</td>
<td>V</td>
</tr>
<tr>
<td>Input Leakage Current</td>
<td>IN</td>
<td>1.0</td>
<td>2.5</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>(V_IN = 0 to 5.25 V)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Three-State (Off State) Input Current</td>
<td>D0-D7</td>
<td>TDSL</td>
<td>2.0</td>
<td>10</td>
<td>μA</td>
</tr>
<tr>
<td>(V_IN = 0.4 to 2.4 V)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output High Voltage</td>
<td>VOH</td>
<td>VSS+2.4</td>
<td>-</td>
<td>-</td>
<td>V</td>
</tr>
<tr>
<td>(I_LOAD = -205 μA, Enable Pulse Width &lt; 25 μs)</td>
<td>D0-D7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Low Voltage</td>
<td>VOL</td>
<td>-</td>
<td>VSS+0.4</td>
<td>-</td>
<td>V</td>
</tr>
<tr>
<td>(I_LOAD = 1.6 mA, Enable Pulse Width &lt; 25 μs)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Leakage Current</td>
<td>ILOH</td>
<td>1.0</td>
<td>10</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>(V_OH = 2.4 V)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Power Dissipation (Measured at T_A = T_L)</td>
<td>PINT</td>
<td>300</td>
<td>525</td>
<td>mW</td>
<td></td>
</tr>
<tr>
<td>Internal Input Capacitance</td>
<td>Cin</td>
<td>10</td>
<td>12.5</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>(V_IN = 0, T_A = 25°C, f = 1.0 MHz)</td>
<td>D0-D7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.0</td>
<td>7.5</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>Output Capacitance</td>
<td>COUT</td>
<td>10</td>
<td>5.0</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>(V_IN = 0, T_A = 25°C, f = 1.0 MHz)</td>
<td>RTS_Tx Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SERIAL DATA TIMING CHARACTERISTICS

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Symbol</th>
<th>MC6850 Min</th>
<th>MC6850 Max</th>
<th>MC68A50 Min</th>
<th>MC68A50 Max</th>
<th>MC68B50 Min</th>
<th>MC68B50 Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Clock Pulse Width, Low</td>
<td>PWCL</td>
<td>600</td>
<td>450</td>
<td>260</td>
<td>ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(See Figure 1)</td>
<td></td>
<td>900</td>
<td>650</td>
<td>550</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Clock Pulse Width, High</td>
<td>PWCH</td>
<td>600</td>
<td>450</td>
<td>260</td>
<td>ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(See Figure 2)</td>
<td></td>
<td>900</td>
<td>650</td>
<td>550</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Clock Frequency</td>
<td>TC</td>
<td>0.8</td>
<td>1.0</td>
<td>1.5</td>
<td>MHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(See Figure 3)</td>
<td></td>
<td>500</td>
<td>750</td>
<td>1000</td>
<td>kHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Clock-to-Data Delay for Transmitter (See Figure 3)</td>
<td>TDD</td>
<td>600</td>
<td>540</td>
<td>460</td>
<td>ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive Data Setup Time (See Figure 4)</td>
<td>TRDS</td>
<td>250</td>
<td>100</td>
<td>30</td>
<td>ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive Data Hold Time (See Figure 5)</td>
<td>TRDH</td>
<td>250</td>
<td>100</td>
<td>30</td>
<td>ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt Request Release Time (See Figure 6)</td>
<td>TRR</td>
<td>1.2</td>
<td>0.9</td>
<td>0.7</td>
<td>µs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request-to-Send Delay Time (See Figure 6)</td>
<td>TRTS</td>
<td>560</td>
<td>480</td>
<td>400</td>
<td>ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Rise and Fall Times (or 10% of the pulse width if smaller)</td>
<td>trc, tf</td>
<td>1.0</td>
<td>0.5</td>
<td>0.25</td>
<td>µs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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)

<table>
<thead>
<tr>
<th>Ident. Number</th>
<th>Characteristic</th>
<th>Symbol</th>
<th>MC8850</th>
<th>MC88A50</th>
<th>MC88B50</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cycle Time</td>
<td>t_{cy}</td>
<td>1.0</td>
<td>10</td>
<td>0.67</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Pulse Width, E Low</td>
<td>PW_{EL}</td>
<td>450</td>
<td>9500</td>
<td>280</td>
<td>9500</td>
</tr>
<tr>
<td>3</td>
<td>Pulse Width, E High</td>
<td>PW_{EH}</td>
<td>450</td>
<td>9500</td>
<td>280</td>
<td>9500</td>
</tr>
<tr>
<td>4</td>
<td>Clock Rise and Fall Time</td>
<td>t_{r}, t_{f}</td>
<td>25</td>
<td>25</td>
<td>20</td>
<td>ns</td>
</tr>
<tr>
<td>9</td>
<td>Address Hold Time</td>
<td>t_{AH}</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>Address Setup Time Before E</td>
<td>t_{AS}</td>
<td>80</td>
<td>60</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>14</td>
<td>Chip Select Setup Time Before E</td>
<td>t_{CS}</td>
<td>80</td>
<td>60</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>15</td>
<td>Chip Select Hold Time</td>
<td>t_{CH}</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>18</td>
<td>Read Data Hold Time</td>
<td>t_{DHR}</td>
<td>20</td>
<td>50*</td>
<td>20</td>
<td>50*</td>
</tr>
<tr>
<td>21</td>
<td>Write Data Hold Time</td>
<td>t_{DHW}</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>Output Data Delay Time</td>
<td>t_{DDR}</td>
<td>290</td>
<td>180</td>
<td>180</td>
<td>150</td>
</tr>
<tr>
<td>31</td>
<td>Input Data Setup Time</td>
<td>t_{DSW}</td>
<td>165</td>
<td>80</td>
<td>60</td>
<td>60</td>
</tr>
</tbody>
</table>

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

FIGURE 7 – BUS TIMING CHARACTERISTICS

1. Voltage levels shown are V_{L} ≤ 0.4 V, V_{H} ≥ 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

Load A
(D0-D7, RTS, Tx Data)

- 5.0 V
- R_{L} = 2.5 kΩ
- Test Point
- C = 130 pF for D0-D7
- 30 pF for RTS and Tx Data

Load B
(IRG Only)

- 5.0 V
- 3 kΩ
- Test Point
- R = 11.7 kΩ for D0-D7
- 24 kΩ for RTS and Tx Data
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 IRQ and RTS outputs are held at level 1. On all other master resets, the RTS output can be programmed high or low with the IRQ output held high. Control bits CR5 and CR6 should also be programmed to define the state of 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 Status Register is full, the character is placed on the 8-bit ACIA bus when a Read 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 buffer and clocks data to and from the ACIA. This signal will normally be a derivative of the M6800 ♦2 Clock or M6809 E clock.

Read/Write (R/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, 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 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 (IRQ) — Interrupt Request is a TTL-compatible, open-drain (no internal pullup, active low output) that is used to interrupt the MPU. The 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 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 Transmit Interrupt Enabled condition is selected (CR5•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 (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 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, or an Overrun has occurred, Data Carrier Detect (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 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 (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 (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 RTS output is low (the active state). This output can also be used for Data Terminal Ready (DTR).

Data Carrier Detect (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 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 DCD operation.

ACIA REGISTERS

The expanded block diagram for the ACIA indicates the internal registers and chip functions 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/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/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>
<thead>
<tr>
<th>Table 1 — Definition of ACIA Register Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Bus Line Number</strong></td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

* 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/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 CTS and 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 to 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:

<table>
<thead>
<tr>
<th>CR1</th>
<th>CR0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-16</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-64</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Master Reset</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>CR4</th>
<th>CR3</th>
<th>CR2</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7 Bits + Even Parity + 2 Stop Bits</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7 Bits + Odd Parity + 2 Stop Bits</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7 Bits + Even Parity + 1 Stop Bit</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>7 Bits + Odd Parity + 1 Stop Bit</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>8 Bits + 2 Stop Bits</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>8 Bits + 1 Stop Bit</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8 Bits + Even parity + 1 Stop Bit</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8 Bits + Odd parity + 1 Stop Bit</td>
</tr>
</tbody>
</table>

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 (RTS) output, and the transmission of a Break level (space). The following encoding format is used:

<table>
<thead>
<tr>
<th>CR6</th>
<th>CR5</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>RTS = low, Transmitting Interrupt Disabled.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>RTS = low, Transmitting Interrupt Enabled.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>RTS = high, Transmitting Interrupt Disabled.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>RTS = low, Transmit a Break level on the Transmit Data Output. Transmitting Interrupt Disabled.</td>
</tr>
</tbody>
</table>

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 (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/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 (DCD), Bit 2 — The Data Carrier Detect bit will be high when the 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 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 DCD input remains high after read status and read data or master reset has occurred, the interrupt is cleared, the DCD status bit remains high and will follow the DCD input.

Clear-to-Send (CTS), Bit 3 — The Clear-to-Send bit indicates the state of the Clear-to-Send input from a modem. A low 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 parity check results are inhibited.

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

PACKAGE DIMENSIONS

CASE 716-06
(CERAMIC)

CASE 623-03
(CERDIP)

CASE 709-02
(POLYESTER)

NOTES:
1. DIM "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
2. LEADS WITHIN 0.13 mm (0.006) RADIUS OF TRUE POSITION AT SEATING PLANE AT MAXIMUM MATERIAL CONDITION. WHEN FORMED PARALLEL.

CASE 716-06
(CERAMIC)

NOTE:
1. LEADS TRUE POSITIONED WITHIN 0.35 mm (0.014) DIA AT SEATING PLANE AT MAXIMUM MATERIAL CONDITION.
2. DIM "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.

MOTOROLA Semiconductor Products Inc.
ORDERING INFORMATION

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° to +70°C
C = -40° to +85°C
Package
P = Plastic
S = Cerdip
L = Ceramic

BEetter 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 - I (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.

<table>
<thead>
<tr>
<th>Speed</th>
<th>Device</th>
<th>Temperature Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 MHz</td>
<td>MC6860P.L,S</td>
<td>0 to 70°C, -40 to +85°C</td>
</tr>
<tr>
<td></td>
<td>MC6880CP.CL,CS</td>
<td></td>
</tr>
<tr>
<td>1.5 MHz</td>
<td>MC68A50P.L,S</td>
<td>0 to +70°C, -40 to +85°C</td>
</tr>
<tr>
<td></td>
<td>MC68A50CP.CL,CS</td>
<td></td>
</tr>
<tr>
<td>2.0 MHz</td>
<td>MC68B50P.L,S</td>
<td>0 to +70°C</td>
</tr>
</tbody>
</table>

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

- One-Chip Floppy Disk 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.

ABSOLUTE MAXIMUM RATINGS

<table>
<thead>
<tr>
<th>Rating</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage on any pin to VSS</td>
<td>VCC, V1, VO</td>
<td>-0.3 to +7.0</td>
<td>V</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>TOP</td>
<td>0 to 70</td>
<td>°C</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>Tstg</td>
<td>-55 to +150</td>
<td>°C</td>
</tr>
</tbody>
</table>

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.
**GUARANTEED OPERATING CONDITIONS**

(Referenced to $V_{SS}$)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
<th>Operating Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min.</td>
<td>Typ.</td>
<td>Max.</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>$V_{CC}$</td>
<td>4.75</td>
<td>5.00</td>
<td>5.25</td>
</tr>
<tr>
<td></td>
<td>$V_{SS}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Input High Voltage</td>
<td>$V_{IH}$</td>
<td>2.0</td>
<td></td>
<td>$V_{CC}$</td>
</tr>
<tr>
<td>Input Low Voltage</td>
<td>$V_{IL}$</td>
<td>-0.3</td>
<td>0</td>
<td>0.8</td>
</tr>
</tbody>
</table>
## PIN DESCRIPTIONS

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Symbol</th>
<th>Pin Name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>VSS</td>
<td>Power Supply</td>
<td>I</td>
<td>Ground (GND)</td>
</tr>
<tr>
<td>21</td>
<td>VCC</td>
<td></td>
<td></td>
<td>+5V DC supply</td>
</tr>
<tr>
<td>24</td>
<td>CLK</td>
<td>Clock</td>
<td>I</td>
<td>2-MHz fixed frequency clock signal (1-MHz for mini-floppy disk)</td>
</tr>
<tr>
<td>19</td>
<td>MR</td>
<td>Master Reset</td>
<td>I</td>
<td>Signal for resetting the FDC</td>
</tr>
<tr>
<td>22</td>
<td>TEST</td>
<td>Test</td>
<td>I</td>
<td>Signal for setting the FDC into a test mode</td>
</tr>
<tr>
<td>1, 40</td>
<td>NC</td>
<td>Non Connection</td>
<td></td>
<td>These pins are not used.</td>
</tr>
</tbody>
</table>

## MPU INTERFACE PINS

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Symbol</th>
<th>Pin Name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>DDEN</td>
<td>Double Density</td>
<td>I</td>
<td>Signal for selecting a FDC operation mode: When DDEN = 0, the double density operation mode is selected. When DDEN = 1, the single density operation mode is selected. This input must be fixed while the FDC is in busy state.</td>
</tr>
<tr>
<td>3</td>
<td>CS</td>
<td>Chip Select</td>
<td>I</td>
<td>Signal for controlling the DALs: When CS = 0, the DALs are activated and data transfer between the FDC and the MPU is enabled. When CS = 1, the DALs are in high impedance state and data transfer is inhibited. (i.e., RE and WE are ignored.)</td>
</tr>
<tr>
<td>4</td>
<td>RE</td>
<td>Read Enable</td>
<td>I</td>
<td>Strobe signal provided when data is read from internal registers: When CS = RE = 0, data can be read from internal registers.</td>
</tr>
<tr>
<td>2</td>
<td>WE</td>
<td>Write Enable</td>
<td>I</td>
<td>Strobe signal provided when data is written into internal registers: When CS = WE = 0, data can be written into internal registers.</td>
</tr>
<tr>
<td>5, 6</td>
<td>A₀, A₁</td>
<td>Register Select</td>
<td>I</td>
<td>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)</td>
</tr>
<tr>
<td>7 ~ 14</td>
<td>DAL₀~DAL₇</td>
<td>Data Access Line</td>
<td>I/O</td>
<td>8-bit bidirectional bus for transferring 8-bit data between the FDC and the MPU. MB 8876A: negative logic / MB 8877A: positive logic</td>
</tr>
<tr>
<td>38</td>
<td>DRQ</td>
<td>Data Request</td>
<td>O</td>
<td>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.</td>
</tr>
<tr>
<td>39</td>
<td>IRQ</td>
<td>Interrupt Request</td>
<td>O</td>
<td>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.</td>
</tr>
</tbody>
</table>

## FLOPPY DISK INTERFACE PINS

**Disk Head Control Signal**

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Symbol</th>
<th>Pin Name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>STEP</td>
<td>Step Move</td>
<td>O</td>
<td>Step pulse signal for moving a disk head.</td>
</tr>
<tr>
<td>16</td>
<td>DIRC</td>
<td>Direction</td>
<td>O</td>
<td>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.</td>
</tr>
<tr>
<td>28</td>
<td>HLD</td>
<td>Head Load</td>
<td>O</td>
<td>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.</td>
</tr>
</tbody>
</table>
### Disk Head Control Signal (cont'd)

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Symbol</th>
<th>Pin Name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>HLT</td>
<td>Head Load Timing</td>
<td>I</td>
<td>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.</td>
</tr>
<tr>
<td>34</td>
<td>TR00</td>
<td>Track 00</td>
<td>I</td>
<td>Signal for informing whether a disk head is positioned on Track No. 00 or not: TR00 = 0 shows Track No. 00 is detected during track seeking operation.</td>
</tr>
<tr>
<td>32</td>
<td>READY</td>
<td>Ready</td>
<td>I</td>
<td>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 executed regardless of this signal.</td>
</tr>
<tr>
<td>35</td>
<td>IP</td>
<td>Index Pulse</td>
<td>I</td>
<td>Signal for informing the FDC of an index hole of disk being detected in the FDD.</td>
</tr>
</tbody>
</table>

### Disk Read Operation Signal

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Symbol</th>
<th>Pin Name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>RG</td>
<td>Read Gate</td>
<td>O</td>
<td>Signal for informing synchronization between RCLK and RAWREAD to an external VFO circuit: RG = 1 shows the FDC has found out a SYNC byte during disk reading operation.</td>
</tr>
<tr>
<td>26</td>
<td>RCLK</td>
<td>Read Clock</td>
<td>I</td>
<td>A data window signal which is generated in an external VFO circuit out of Read Data.</td>
</tr>
<tr>
<td>27</td>
<td>RAWREAD</td>
<td>Raw Read</td>
<td>I</td>
<td>A raw read data signal transferred from the FDD.</td>
</tr>
</tbody>
</table>

### Disk Write Operation Signal

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Symbol</th>
<th>Pin Name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>WG</td>
<td>Write Gate</td>
<td>O</td>
<td>Signal for indicating data is being written into a disk.</td>
</tr>
<tr>
<td>17</td>
<td>EARLY</td>
<td>Early Shift</td>
<td>O</td>
<td>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.</td>
</tr>
<tr>
<td>18</td>
<td>LATE</td>
<td>Late Shift</td>
<td>O</td>
<td>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.</td>
</tr>
<tr>
<td>31</td>
<td>WD</td>
<td>Write Data</td>
<td>O</td>
<td>A write data signal transferred to the FDD.</td>
</tr>
<tr>
<td>29</td>
<td>TG43</td>
<td>Track Greater Than 43</td>
<td>O</td>
<td>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.</td>
</tr>
<tr>
<td>33</td>
<td>WF/VFOE</td>
<td>Write Fault/ Variable Frequency Oscillator Enable</td>
<td>I/O</td>
<td>Input signal for informing a fault is detected during write operation for a disk (during WG = 1). Output signal for informing the FDC is reading a disk (during WG = 0).</td>
</tr>
<tr>
<td>34</td>
<td>WPRT</td>
<td>Write Protect</td>
<td>I</td>
<td>Signal for inhibiting write operation for disk.</td>
</tr>
</tbody>
</table>
## REGISTER SELECTION

<table>
<thead>
<tr>
<th>Chip Select</th>
<th>Address</th>
<th>Selected Register</th>
<th>Data Access Line Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS A₁ A₀</td>
<td></td>
<td>Read Mode (RE = 0)</td>
<td>DAL₀ ~ DAL₇ / DAL₀ ~ DAL₇</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Deselected</td>
<td>High Impedance</td>
</tr>
<tr>
<td>0 0 0</td>
<td></td>
<td>Status Register (STR)</td>
<td>Enabled</td>
</tr>
<tr>
<td>0 0 1</td>
<td></td>
<td>Track Register (TR)</td>
<td>Enabled</td>
</tr>
<tr>
<td>0 1 0</td>
<td></td>
<td>Sector Register (SCR)</td>
<td>Enabled</td>
</tr>
<tr>
<td>0 1 1</td>
<td></td>
<td>Data Register (DR)</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

*: Don't care

---

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

- **DATA I/O Buffer**
- **OUTPUT BUS LINE**
- **INPUT BUS LINE**
  - **DATA REG. (DR)**
  - **COMMAND REG. (CR)**
  - **SECTOR REG. (SCR)**
  - **TRACK REG. (TR)**
  - **STATUS REG. (STR)**
- **DATA SHIFT REG. (DSR)**
- **AM DET.**
- **CRC GEN.**
- **ALU**
- **WD**
- **RAWREAD**
- **<RCLK**
- **<RCBL**
- **MPU INTERFACE**
- **COMMAND PLA (232 x 19)**
- **DISK INTERFACE**

* 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 Command 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.)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output High Voltage (I_{OH} = -200\mu A)</td>
<td>V_{OH}</td>
<td>2.4</td>
<td>V</td>
</tr>
<tr>
<td>Output Low Voltage (I_{OL} = 1.8 mA)</td>
<td>V_{OL}</td>
<td>0.4</td>
<td>V</td>
</tr>
<tr>
<td>Three-State (Off-State) Input Current (V_{IN} = 0.4V to 2.4V)</td>
<td>I_{TSI}</td>
<td>10</td>
<td>\mu A</td>
</tr>
<tr>
<td>Input Leakage Current (See Note 1)</td>
<td>I_{IN1}</td>
<td>2.5</td>
<td>\mu A</td>
</tr>
<tr>
<td>Input Leakage Current (See Note 2)</td>
<td>I_{IN2}</td>
<td>100</td>
<td>\mu A</td>
</tr>
<tr>
<td>Output Leakage Current for Off-State (V_{OH} = 2.4V)</td>
<td>I_{LOH}</td>
<td>10</td>
<td>\mu A</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>P_c</td>
<td>350</td>
<td>mW</td>
</tr>
</tbody>
</table>

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

2): For HLT, TEST, WF, WPRT, and DDEN. (V_{IN} = 0V to 5.25V)

## AC CHARACTERISTICS

(Full Guaranteed Operating Conditions unless otherwise noted.)

MPU Read Timing (From FDC)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Setup Time</td>
<td>t_{SET}</td>
<td>50</td>
<td>ns</td>
</tr>
<tr>
<td>Address Hold Time</td>
<td>t_{HLD}</td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>RE Pulse Width</td>
<td>t_{RE}</td>
<td>280</td>
<td>ns</td>
</tr>
<tr>
<td>DRO Reset Time</td>
<td>t_{DRR}</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td>IRQ Reset Time</td>
<td>t_{IRR}</td>
<td>500</td>
<td>ns</td>
</tr>
<tr>
<td>Data Delay Time (C_L = 25pF)</td>
<td>t_{DACC}</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td>Data Hold Time (C_L = 25pF)</td>
<td>t_{DOH}</td>
<td>50</td>
<td>150</td>
</tr>
<tr>
<td>DRO Service Time (RCLK cycle = 2\mu s)</td>
<td>t_{SEVR}</td>
<td>13.5*</td>
<td>\mu s</td>
</tr>
</tbody>
</table>

*: These values are doubled when CLK = 1 MHz.
**MPU Write Timing (To FDC)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Setup Time</td>
<td>( t_{SET} )</td>
<td>50</td>
<td>ns</td>
</tr>
<tr>
<td>Address Hold Time</td>
<td>( t_{HLD} )</td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>WE Pulse Width</td>
<td>( t_{WE} )</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>DRQ Reset Time</td>
<td>( t_{DRR} )</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td>IRQ Reset Time</td>
<td>( t_{IRR} )</td>
<td>500</td>
<td>ns</td>
</tr>
<tr>
<td>Data Setup Time</td>
<td>( t_{DS} )</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td>Data Hold Time</td>
<td>( t_{DH} )</td>
<td>0</td>
<td>ns</td>
</tr>
<tr>
<td>DRQ Service Time (DDEN = &quot;L&quot;)</td>
<td>( t_{SEVW} )</td>
<td>11.5*</td>
<td>( \mu s )</td>
</tr>
</tbody>
</table>

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

**READ TIMING**

- DRQ
  - \( 0.4V \)
  - \( t_{DRR} \)
  - \( t_{SEVR} \)
- IRQ
  - \( 0.4V \)
- \( \overline{CS} \), \( A_0, A_1 \)
  - \( 2.0V \)
  - \( 0.8V \)
  - \( t_{SET} \)
  - \( t_{RE} \)
  - \( t_{HLD} \)
- \( \overline{RE} \)
  - \( 2.0V \)
  - \( 0.8V \)
  - \( t_{DAC} \)
  - \( t_{DCH} \)
- DAL/DAL
  - \( 0.4V \)

**WRITE TIMING**

- DRQ
  - \( 0.4V \)
  - \( t_{DRR} \)
  - \( t_{SEVW} \)
- IRQ
  - \( 0.4V \)
- \( \overline{CS} \), \( A_0, A_1 \)
  - \( 2.0V \)
  - \( 0.8V \)
  - \( t_{SET} \)
  - \( t_{WE} \)
  - \( t_{HLD} \)
- WE
  - \( 2.0V \)
  - \( 0.8V \)
  - \( t_{DS} \)
  - \( t_{DH} \)
- DAL/DAL
  - \( 0.4V \)

*: These values are doubled when CLK=1MHz.
# FDC Read Data Timing (From Disk)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAWREAD Pulse Width</td>
<td>( t_{PW} )</td>
<td>100*</td>
<td>250* ns</td>
</tr>
<tr>
<td>Clock Setup Time</td>
<td>( t_D )</td>
<td>40</td>
<td>ns</td>
</tr>
<tr>
<td>Clock Hold Time for MFM</td>
<td>( t_{CD} )</td>
<td>40</td>
<td>ns</td>
</tr>
<tr>
<td>Clock Hold Time for FM</td>
<td>( t_{CS} )</td>
<td>40</td>
<td>ns</td>
</tr>
<tr>
<td>RAWREAD Cycle Time</td>
<td>MFM</td>
<td>2* or 4*</td>
<td>( \mu s )</td>
</tr>
<tr>
<td></td>
<td>FM</td>
<td>2* or 4*</td>
<td>( \mu s )</td>
</tr>
<tr>
<td>RCLK High Pulse Width</td>
<td>MFM</td>
<td>0.8</td>
<td>1* 20 ( \mu s )</td>
</tr>
<tr>
<td></td>
<td>FM</td>
<td>0.8</td>
<td>2* 20 ( \mu s )</td>
</tr>
<tr>
<td>RCLK Low Pulse Width</td>
<td>MFM</td>
<td>0.8</td>
<td>1* 20 ( \mu s )</td>
</tr>
<tr>
<td></td>
<td>FM</td>
<td>0.8</td>
<td>2* 20 ( \mu s )</td>
</tr>
<tr>
<td>RCLK Cycle Time</td>
<td>MFM</td>
<td>2*</td>
<td>( \mu s )</td>
</tr>
<tr>
<td></td>
<td>FM</td>
<td>4*</td>
<td>( \mu s )</td>
</tr>
</tbody>
</table>

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

---

**READ DATA TIMING**

[Diagram showing RAW READ and RCLK timings with labels for \( t_{PW}, t_{BC}, t_D, t_{CD}, t_{CS}, t_A, t_B, t_C \) and voltage levels.]
## FDC Write Data Timing (To Disk)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Data Pulse Width</td>
<td>( t_{WD} ** )</td>
<td>( \text{CLK} = 2 , \text{MHz} )</td>
<td>FM: 450, 500, 550, 550; MFM: 150, 200, 250</td>
<td>ns</td>
</tr>
<tr>
<td>Write Gate to Write Data</td>
<td>( t_{WG} ** )</td>
<td>( \text{CLK} = 2 , \text{MHz} )</td>
<td>FM: 2; MFM: 1</td>
<td>( \mu \text{s} )</td>
</tr>
<tr>
<td>Write Gate off from WD</td>
<td>( t_{WF} ** )</td>
<td>( \text{CLK} = 2 , \text{MHz} )</td>
<td>FM: 2; MFM: 1, 2</td>
<td>( \mu \text{s} )</td>
</tr>
<tr>
<td>Early (Late) to Write Data</td>
<td>( t_{S} )</td>
<td>( \text{CLK} = 2 , \text{MHz} ) MFM</td>
<td>125</td>
<td>ns</td>
</tr>
<tr>
<td>Early (Late) from Write Data</td>
<td>( t_{H} )</td>
<td>( \text{CLK} = 2 , \text{MHz} ) MFM</td>
<td>-50 ns</td>
<td>ns</td>
</tr>
<tr>
<td>WD Valid after CLK</td>
<td>( t_{D1} )</td>
<td>( \text{CLK} = 2 , \text{MHz} ) MFM</td>
<td>200</td>
<td>ns</td>
</tr>
<tr>
<td>WD Valid to CLK</td>
<td>( t_{D2} )</td>
<td>( \text{CLK} = 2 , \text{MHz} ) MFM</td>
<td>50</td>
<td>ns</td>
</tr>
</tbody>
</table>

*: 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 \( \text{CLK} = 1 \, \text{MHz} \).

### WRITE DATA TIMING

- \( W_G \):
  - 2.4V
  - \( t_{WD} \):
    - 0.4V

- \( W_D \):
  - 2.4V
  - \( t_{WF} \):
    - 0.4V

- EARLY (LATE):
  - 2.4V
  - \( t_S \):
    - 0.4V
  - \( t_H \):
    - 0.4V

- \( C_L K \):
  - 2.4V
  - \( t_{D1} \):
    - 2.4V
  - \( t_{D2} \):
    - 2.4V

- \( W_D \):
  - 2.4V
## OTHER TIMINGS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min.</td>
<td>Typ.</td>
</tr>
<tr>
<td>CLK Low Pulse Width</td>
<td>$t_{CD1}$</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>CLK High Pulse Width</td>
<td>$t_{CD2}$</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>STEP Pulse Width</td>
<td></td>
<td>2*</td>
<td></td>
</tr>
<tr>
<td>MFM</td>
<td>$t_{STP}$</td>
<td>4*</td>
<td></td>
</tr>
<tr>
<td>FM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIRC Setup Time</td>
<td>$t_{DIR}$</td>
<td>12*</td>
<td></td>
</tr>
<tr>
<td>MR Pulse Width**</td>
<td>$t_{MR}$</td>
<td>50*</td>
<td></td>
</tr>
<tr>
<td>IP Pulse Width</td>
<td>$t_{IP}$</td>
<td>10*</td>
<td></td>
</tr>
<tr>
<td>WF Pulse Width</td>
<td>$t_{WF}$</td>
<td>10*</td>
<td></td>
</tr>
<tr>
<td>CLK Cycle Time</td>
<td>$t_{CYC}$</td>
<td>0.5*</td>
<td></td>
</tr>
</tbody>
</table>

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

**: During Master Reset, CLK of more than 10 cycles are required.
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.
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
SORCIM 808x Assembler ver 3.5E <t/55/7> 59:92 Page 1
SINGLE Density Monitor for Model 1 system.
C:SDROMA +ASM

*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 variables 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 length 0=128, 1=256, 2=512, 3=1024.
  *Added Dec. 2181 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 register 8.
  *Added Jan. 582 by DEB.*

There seems to be a problem in the timing of the RSEC AND MSEC routines. The problem shows up when CP/M reads the disk one sector after another. The fix is to use READ only when the drive is selected and not before every read or write.
  *Added Jan. 682 by DEB.*

  0000 = 0000
  CCP = 0
  ORG 0
  IFWA 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
FROM SECTIONS ARE:

: 0[1]  =  ROM STANDARDS
: 0[2]  =  CPM BOOT
: 0[3]  =  CONSOL ROUTINES
: 0[4]  =  PIA ROUTINES
: 0[5]  =  SIA ROUTINES
: 0[6]  =  DISK ROUTINES
: 0[7]  =  FORMAT ROUTINE
: 0[R]  =  RAM LOCATIONS
Monitor Main Loop

@1
0000 DISDIM ;disable DIM
0002 C33600 ;reset/restart
0005 USER:
0006 EXIT to user program from 'G' command
0007 EI
0008 C9
0009 RET ;Exit from G command
000A ILINT:
000B HERE on unknown interrupt
000C DI
000D DISDIM A
000E ED736FEF $ ;save interrupted proc stack
000F 3199EF LDK SP + IESTK ;insure stack in RAM
0010 3E49 ALPUSH ;save all registers
0011 3206EF STD A + 1
0012 EXIT:
0013笔者 interrupt code via exiting to RAM and
0014 then enable or disable ROM code depending on
0015 the value contained in ROMRAM cell.
0016 3AD02C LD A + H + V10 ;clear interrupt
0017 3A06EF LD A + ROMRAM
0018 C2DF00 OR A
0019 C2DF00 JNZ ROMJP1 ;if return to RAM, CPM
001A 3D766FEF $ ;get users stack back
001B ED766FEF $ ;load SP + IESTK
001C 3C
001D EXIT:
001E BMON:
001F 31C1EF LDK SP + ROMSTK
0020 AF XRA A
0021 3217EF STD A + SDISK ;disk to drive 0
0022 32EFEF STD A + VRTDPF ;OFFSET TO ZERO USED IN COUT
0023 3C INC A
0024 3C INR A
0025 3C INR A
0026 0A CALL HINT ;Initialize hardware
0027 D8000 CALL HINT
0028 210000 LDK HL + HIGH MRAM * 100h
0029 C2EAF STD HL + SSAVE ;Initialize User stack in ram
002A 210000 LDK HL + O
002B 2200EF STD HL + LDSEL ;clear select and last track
002C 218601 LDK DE + IMGS
002D C09701 CALL OSTR ;Output initial message
002E 180E ^006A$ JR BMON1 ;continue
002F 0066 ORG NMIA
0030 C33600 JMP BMON ;INMI
Monitor Main Loop

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

006A CD7E03 BMON1: CALL CI   ; Get next character
006B FE0D CMP CR
006C 21C7EF LK HL+DSKSWP
0072 0E00 LK C=0
0074 71 STO C,[hl]   ; set to A=A, B=B
0075 CA6802 JZ CB00T   ; if cold boot
0076 34 INC [hl]   ; swap drives: A=B, B=A
0078 FE22 CMP ***
0079 CA6802 JZ CB00T   ; if cold boot off of B
007E 1886 ^0036$ JR BMON
Monitor Main Loop

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

EXIT
   All hardware initialized.
   Set flag indicating in ROM

0080 PROC
0080 ENAROM
0089 DISDIM
0088 21F07 LDK HL,GKEY
008E 22F8EF STO HL,INTBL+(4*2) set keyboard interrupt
0091 ED5E $ IMZ
0093 3EEF LDK A;high INTBL
0095 ED47 $ MDV I;&A set interrupt page

   Initialize keyboard
0097 AF XRA A
0098 32F0EF STO A;EFFECT
0098 3268EF STO A;BELCNT iclear bell timer cell
009E 325EEF STO A;KEY clear last key cell
00A1 3207EF STO A;ECHOP iclear echo to list dev
00A4 3260EF STO A;ESC clear ESC hold flag
00A7 2104EF LDK HL,KEYLE
00AA 0A06 LDK B;KLLEN,KLELEN
00AC 1:1 STO A;HL
00AD 23 INC HL
00AE 10FC ^00AC5 DJNZ I;
0080 2F CMA
00B1 3263EF STO A;IDAY iset date invalid for SETUP.COM
00B4 3259EF STO A;KEYLLCK indicate NOT locked
00BB 3EE0LDK A;VLL
00BB 326CEF STO A;VLL LIMIT iset max line limit

   Now function PIA to set starting address
   and continue to move display window over
   the first display line.
   Set cursor as I=40 on 1st line.
00BC CD4F09 CALL SPAD iset up for output
00BF 0EEA LDK C;VFLO
00C1 CD6409 CALL OPAD iset for -10 char position AND DOUBLE DENSITY

   Set beginning line to 0
00C4 AF XRA A
00C5 4F MDV C;&A
00C6 CD7109 CALL DPBD iset line

   Initialize IEEE port
00C9 0EE1 LDK C;1
00CB CD7E09 CALL ie,co

   Reset-Master clear the SIO (ACIA)
00CE 0E55 LK C;SI,S16 iselect 16x clock for 1200 baud
```
Monitor Main Loop

00D0  CDF60A  CALL  SIRST  :reset

00D3  3E02  ; Set default seek to as defined by systext
00D5  3213EF  LDK  A:*SEEKTM  :defined from systext
            STD  A:*SEKDEL  :set seek step rate

00D8  3E3E  ; Set default prompt char
00DA  3206EF  LDK  A:*PMCHR
00DD  STD  A:*MPCHR  :set up default
00DE  C9  RET
Monitor Main Loop

<table>
<thead>
<tr>
<th>Address</th>
<th>Op Code</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00DF</td>
<td>113316</td>
<td>romjpl: 1d</td>
<td>$dk de+1633h: offset in bios jump table</td>
</tr>
<tr>
<td>00E2</td>
<td>1803</td>
<td>jr</td>
<td>biojpc</td>
</tr>
<tr>
<td>00E4</td>
<td>113616</td>
<td>romjpo2: 1d</td>
<td>$dk de+1636h: offset in bios jump table</td>
</tr>
<tr>
<td>00E7</td>
<td>2AD2EF</td>
<td>biojpo: 1d</td>
<td>hi+cpcadr</td>
</tr>
<tr>
<td>00EA</td>
<td>19</td>
<td>.add</td>
<td>hi+de</td>
</tr>
<tr>
<td>00EB</td>
<td>E9</td>
<td>.jmp</td>
<td>[hi]</td>
</tr>
</tbody>
</table>
Monitor Main Loop

IRDM JUMP TABLE

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0100</td>
<td>JMP C600D</td>
<td>C600D delay readout</td>
</tr>
<tr>
<td>$0103</td>
<td>JMP C6402</td>
<td>C6402 delay step out</td>
</tr>
<tr>
<td>$0106</td>
<td>JMP C37103</td>
<td>C37103 delay seek</td>
</tr>
<tr>
<td>$0109</td>
<td>JMP C37E03</td>
<td>C37E03 delay step in</td>
</tr>
<tr>
<td>$010C</td>
<td>JMP C31004</td>
<td>C31004 delay step out</td>
</tr>
<tr>
<td>$010F</td>
<td>JMP C31C08</td>
<td>C31C08 delay step in</td>
</tr>
<tr>
<td>$0112</td>
<td>JMP C31C08</td>
<td>C31C08 delay step in</td>
</tr>
<tr>
<td>$0115</td>
<td>JMP C30308</td>
<td>C30308 delay step out</td>
</tr>
</tbody>
</table>

; Disk I/O

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0118</td>
<td>JMP C34008</td>
<td>C34008 delay readout</td>
</tr>
<tr>
<td>$011B</td>
<td>JMP C9</td>
<td>C9 delay readout</td>
</tr>
<tr>
<td>$011E</td>
<td>JMP C3750C</td>
<td>C3750C delay readout</td>
</tr>
<tr>
<td>$0121</td>
<td>JMP C37D0C</td>
<td>C37D0C delay readout</td>
</tr>
<tr>
<td>$0124</td>
<td>JMP C36008</td>
<td>C36008 delay readout</td>
</tr>
<tr>
<td>$0127</td>
<td>JMP C35808</td>
<td>C35808 delay readout</td>
</tr>
<tr>
<td>$012A</td>
<td>JMP C36008</td>
<td>C36008 delay readout</td>
</tr>
<tr>
<td>$012D</td>
<td>JMP C31308</td>
<td>C31308 delay readout</td>
</tr>
<tr>
<td>$0130</td>
<td>JMP C38008</td>
<td>C38008 delay readout</td>
</tr>
<tr>
<td>$0133</td>
<td>JMP C30F00</td>
<td>C30F00 delay readout</td>
</tr>
<tr>
<td>$0136</td>
<td>JMP C36400</td>
<td>C36400 delay readout</td>
</tr>
<tr>
<td>$0139</td>
<td>JMP C33F0E</td>
<td>C33F0E delay readout</td>
</tr>
</tbody>
</table>

; IEEE

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$013C</td>
<td>JMP C3F60A</td>
<td>C3F60A delay readout</td>
</tr>
<tr>
<td>$013F</td>
<td>JMP C37E09</td>
<td>C37E09 delay readout</td>
</tr>
<tr>
<td>$0142</td>
<td>JMP C3C109</td>
<td>C3C109 delay readout</td>
</tr>
<tr>
<td>$0145</td>
<td>JMP C30409</td>
<td>C30409 delay readout</td>
</tr>
<tr>
<td>$0148</td>
<td>JMP C3009</td>
<td>C3009 delay readout</td>
</tr>
<tr>
<td>$014B</td>
<td>JMP C3200A</td>
<td>C3200A delay readout</td>
</tr>
<tr>
<td>$014E</td>
<td>JMP C3430A</td>
<td>C3430A delay readout</td>
</tr>
<tr>
<td>$0151</td>
<td>JMP C3850A</td>
<td>C3850A delay readout</td>
</tr>
<tr>
<td>$0154</td>
<td>JMP C3E10A</td>
<td>C3E10A delay readout</td>
</tr>
</tbody>
</table>

; SuperCalc

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0157</td>
<td>JMP C3F106</td>
<td>C3F106 delay readout</td>
</tr>
<tr>
<td>$015A</td>
<td>JMP C30507</td>
<td>C30507 delay readout</td>
</tr>
<tr>
<td>$015D</td>
<td>JMP C3E906</td>
<td>C3E906 delay readout</td>
</tr>
</tbody>
</table>

; Disk I/O

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0160</td>
<td>JMP C3200E</td>
<td>C3200E delay readout</td>
</tr>
<tr>
<td>$0163</td>
<td>JMP C3000E</td>
<td>C3000E delay readout</td>
</tr>
<tr>
<td>$0166</td>
<td>JMP C3F008</td>
<td>C3F008 delay readout</td>
</tr>
<tr>
<td>$0169</td>
<td>JMP C3100C</td>
<td>C3100C delay readout</td>
</tr>
<tr>
<td>$016C</td>
<td>JMP C3390C</td>
<td>C3390C delay readout</td>
</tr>
<tr>
<td>$016F</td>
<td>JMP C3400C</td>
<td>C3400C delay readout</td>
</tr>
<tr>
<td>$0172</td>
<td>JMP C3600C</td>
<td>C3600C delay readout</td>
</tr>
<tr>
<td>$0175</td>
<td>JMP C3640D</td>
<td>C3640D delay readout</td>
</tr>
<tr>
<td>$0178</td>
<td>JMP C3330D</td>
<td>C3330D delay readout</td>
</tr>
<tr>
<td>$017B</td>
<td>JMP C3490D</td>
<td>C3490D delay readout</td>
</tr>
</tbody>
</table>

; DMA

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0160</td>
<td>JMP DMAWRT</td>
<td>DMA WRITE TO CONTROLLER</td>
</tr>
<tr>
<td>$0163</td>
<td>JMP DMARO</td>
<td>DMA READ FROM CONTROLLER</td>
</tr>
<tr>
<td>$0166</td>
<td>JMP HOME</td>
<td>Home disk drive</td>
</tr>
<tr>
<td>$0169</td>
<td>JMP SEEK</td>
<td>Seek to track</td>
</tr>
<tr>
<td>$016C</td>
<td>JMP STEP</td>
<td>Step same direction</td>
</tr>
<tr>
<td>$016F</td>
<td>JMP STEPOUT</td>
<td>Step out</td>
</tr>
<tr>
<td>$0172</td>
<td>JMP FORINT</td>
<td>Force interrupt</td>
</tr>
<tr>
<td>$0175</td>
<td>JMP REACTR</td>
<td>Read track</td>
</tr>
<tr>
<td>$0178</td>
<td>JMP FWTHAR</td>
<td>Format one track</td>
</tr>
</tbody>
</table>
Monitor Main Loop

SCRCIM 808x Assembler ver 3.5E  c:/557=59:92 Page 9
C:SDROMA,ASM

017E C3A40D JMP SELDRV ; SELECT DRIVE
0181 C3280B JMP ACISTAT ; ICBIOS SERIAL PORT STATUS
0184 C3F00B JMP SCTRKR ; SET TRACK REGISTER IN CONTROLLER CHIP WITH VALUE IN SAVTRK
Monitor Main Loop:

0187  000A  EMBOTT: DB  CR#LF
0189  4244F5420  DB  "BOOT ERROR"*
,  0193  A0  DC  **
0194  EBOOT:
2$BOOT ERROR MESSAGE ROUTINE
2ENTRY
2NONE
0194  PROC
0194  118701  LDK  DE*EMBOOT  ;HERE ON BOOT ERROR
                     ;FALL THROUGH TO CSTR
Monitor Main Loop

0197 PROC
0197 1A LD A*IDE]
0198 87 OR A
0199 F5 PUSH AF
019A E67F AND 07FH
019C FE7F CMP 07FH
019E 4F MOV C+A
019F 200C ^01A$ JRNZ 14 ;IIF NOT REPEAT
01A1 13 INC DE
01A2 1A LD A*IDE]
01A3 30 DEC A
01A4 47 MOV B+A ;REPEAT COUNT
01A5 13 INC DE
01A6 1A LD A*IDE]
01A7 4F MOV C+A ;GET REPEAT CHAR
01A8 CD1004 ;2: CALL COUT ;OUTPUT CHAR
01A9 10F8 ^01A8$ DJNZ 12 ;IIF NOT DONE
01A0 CD1004 ;4: CALL COUT ;OUTPUT IT
01B0 13 INC DE
01B1 F1 PCP AF
01B2 F29701 JP OSTR ;IIF NOT DONE
01B5 C9 RET

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
:IDE = FWA OF SOURCE

MONITOR: OSM
0100 200C ^01A$ MOV C+A
0101 13 INC DE
0102 1A LD A*IDE]
0103 30 DEC A
0104 47 MOV B+A
0105 13 INC DE
0106 1A LD A*IDE]
0107 4F MOV C+A
0108 CD1004 ;2: CALL COUT
0109 10F8 ^01A$ DJNZ 12
010A CD1004 ;4: CALL COUT
010B 13 INC DE
010C F1 PCP AF
010D F29701 JP OSTR
010E C9 RET

SORMIM 808x Assembler ver 3.5E <1/55/7> 59:92 Page 11
C:SDROMA .ASM
<table>
<thead>
<tr>
<th>DEC</th>
<th>IMGS</th>
<th>DB</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0186</td>
<td>1A0A0A0ADA</td>
<td>DB</td>
<td>'Z'-40h, lf, lf, lf, lf</td>
</tr>
<tr>
<td>0188</td>
<td>7F0B20</td>
<td>DB</td>
<td>07H, 11, ' '</td>
</tr>
<tr>
<td>018E</td>
<td>1867</td>
<td>DB</td>
<td>ESC+VSGH+1, ESC+VEGH</td>
</tr>
<tr>
<td>01C0</td>
<td>11</td>
<td>DB</td>
<td>'A'-40h</td>
</tr>
<tr>
<td>01C1</td>
<td>7F1817</td>
<td>DB</td>
<td>07H, 24, 'W'-40h</td>
</tr>
<tr>
<td>01C4</td>
<td>05</td>
<td>DB</td>
<td>'E'-40h</td>
</tr>
<tr>
<td>01C5</td>
<td>1847</td>
<td>DB</td>
<td>ESC+VEGH</td>
</tr>
<tr>
<td>01C7</td>
<td>000A</td>
<td>DB</td>
<td>cr, lf</td>
</tr>
<tr>
<td>01C9</td>
<td>7F0B20</td>
<td>DB</td>
<td>07H, 11, ' '</td>
</tr>
<tr>
<td>01CC</td>
<td>1867011847</td>
<td>DB</td>
<td>ESC+VSGH+1, ESC+VEGH</td>
</tr>
<tr>
<td>01D1</td>
<td>2020</td>
<td>DB</td>
<td>'A'-40h</td>
</tr>
<tr>
<td>01D3</td>
<td>186C</td>
<td>DB</td>
<td>ESC+V1</td>
</tr>
<tr>
<td>01D5</td>
<td>4F53424F52</td>
<td>DB</td>
<td>'05BORNE System One'</td>
</tr>
<tr>
<td>01E8</td>
<td>186D</td>
<td>DB</td>
<td>ESC* 'm'</td>
</tr>
<tr>
<td>01EA</td>
<td>202020</td>
<td>DB</td>
<td>ESC+VSGH+4, ESC+VEGH</td>
</tr>
<tr>
<td>01ED</td>
<td>1867041847</td>
<td>DB</td>
<td></td>
</tr>
<tr>
<td>01F2</td>
<td>000A</td>
<td>DB</td>
<td>cr, lf</td>
</tr>
<tr>
<td>01F4</td>
<td>7F0B20</td>
<td>DB</td>
<td>07H, 11, ' '</td>
</tr>
<tr>
<td>01F7</td>
<td>1867</td>
<td>DB</td>
<td>ESC+VSGH</td>
</tr>
<tr>
<td>01F9</td>
<td>01</td>
<td>DB</td>
<td>'A'-40h</td>
</tr>
<tr>
<td>01FA</td>
<td>7F1820</td>
<td>DB</td>
<td>07H, 24, ' '</td>
</tr>
<tr>
<td>01FD</td>
<td>04</td>
<td>DB</td>
<td>'D'-40h</td>
</tr>
<tr>
<td>01FE</td>
<td>1847</td>
<td>DB</td>
<td>ESC+VEGH</td>
</tr>
<tr>
<td>0200</td>
<td>000A</td>
<td>DB</td>
<td>cr, lf</td>
</tr>
<tr>
<td>0202</td>
<td>7F0B20</td>
<td>DB</td>
<td>07H, 11, ' '</td>
</tr>
<tr>
<td>0205</td>
<td>1867</td>
<td>DB</td>
<td>ESC+VSGH</td>
</tr>
<tr>
<td>0207</td>
<td>01</td>
<td>DB</td>
<td>'A'-40h</td>
</tr>
<tr>
<td>0208</td>
<td>1829</td>
<td>DB</td>
<td>ESC+V*</td>
</tr>
<tr>
<td>020A</td>
<td>2052657620</td>
<td>DB</td>
<td>'Rev 1.3'</td>
</tr>
<tr>
<td>0222</td>
<td>1828</td>
<td>DB</td>
<td>ESC+'f'</td>
</tr>
<tr>
<td>0224</td>
<td>04</td>
<td>DB</td>
<td>'D'-40h</td>
</tr>
<tr>
<td>0225</td>
<td>1847</td>
<td>DB</td>
<td>ESC+VEGH</td>
</tr>
<tr>
<td>0227</td>
<td>000A</td>
<td>DB</td>
<td>cr, lf</td>
</tr>
<tr>
<td>0229</td>
<td>7F0B20</td>
<td>DB</td>
<td>07H, 11, ' '</td>
</tr>
<tr>
<td>022E</td>
<td>1A</td>
<td>DB</td>
<td>ESC+VSGH</td>
</tr>
<tr>
<td>022F</td>
<td>7F1818</td>
<td>DB</td>
<td>'Z'-40h</td>
</tr>
<tr>
<td>0232</td>
<td>03</td>
<td>DB</td>
<td>07H, 24, 'X'-40h</td>
</tr>
<tr>
<td>0233</td>
<td>1847</td>
<td>DB</td>
<td>'C'-40h</td>
</tr>
<tr>
<td>0235</td>
<td>000A0A0ADA</td>
<td>DB</td>
<td>cr, lf, lf, lf, lf</td>
</tr>
<tr>
<td>0239</td>
<td>7F0420</td>
<td>DB</td>
<td>07H, 4, ' '</td>
</tr>
<tr>
<td>023C</td>
<td>496E736572</td>
<td>DB</td>
<td>'Insert disk in Drive'</td>
</tr>
<tr>
<td>0251</td>
<td>186C</td>
<td>DB</td>
<td>ESC+'1'</td>
</tr>
<tr>
<td>0253</td>
<td>41</td>
<td>DB</td>
<td>'A'</td>
</tr>
<tr>
<td>0254</td>
<td>1890</td>
<td>DB</td>
<td>ESC+'m'</td>
</tr>
<tr>
<td>0256</td>
<td>20616E6420</td>
<td>DB</td>
<td>' and press RETURN'</td>
</tr>
<tr>
<td>0267</td>
<td>AE</td>
<td>DC</td>
<td></td>
</tr>
</tbody>
</table>
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 01), 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
0268 CB00T:
0268 ENTRY
0269 = drive to boot from
0269 EXIT
026A = DRIVE TO BOOT FROM
026B Proc
026C SET "SDISK" AND "SAVTYP"
026D 79 MOV A+C
026E 321EF STO A+SDISK : ISET DRIVE
026F CD00B CALL SENDEN : IDETERMINE DENSITY
0270 280S *0276s JRZ :2 iF GOOD
0271 CD9401 CALL EBCOT : IPRINT ERROR
0272 18F6 *026C$ JR :1
0273 READ AND SET FBA OF CCP
0274 210000 :2: LDK ML+0000H
0275 220FEF STD ML+DMADR : ISET DMA
0276 AF XRA A 13: :iset track
0277 321EF STO A+SAVTKRK : :iset track
0278 3C INC A
0279 3214EF STO A+SAVTSEC : :iset sector
027A 47 MOV B+A
027B CD5808 CALL RSEC : :READ SECTOR ONE
027C 2805 *028F5 JRZ :4 iF GOOD
027D CD9401 CALL EBCOT : IPRINT ERROR
027E 18ED *027C$ JR :3
027F 3A0200 :4: L0 A+00002H : :get ccp address/100h + 3
0280 0603 SUB 3
0281 2E0D LDK L0
0282 67 MOV H+A
0283 SET NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT
0284 063C LDK B+60 : :ICCP/BIOS/ICBIOS
0285 READ SYSTEM
0286 3A17EF LD A+SDISK : :IDRIVE TO BOOT FROM
0287 F5 PUSH AF
0288 E5 PUSH ML
0289 CD302 CALL BCPM : :boot system
028A F1 POP HL
028B F1 POP AF
028C JUMP SYSTEM
028D 2202EF STD ML+CCPADR
028E 110016 LDK DE+1600h : :offset for bios
028F 19 ADD HL+DE : :address of bios in hl
0290 E9 JMP (HL) : :enter com
Boot CP/M

SORMCIM 808x Assembler ver 3.5E C:\557= 59:92 Page 15
From disk
C:SDROMA:ASM

02A0

WBOOT:
ENTRY
NONE

EXIT

*!NOTE*

*!

THIS ROUTINE DOES NOT EXIT. IT ONLY SETS PARAMETERS FOR BCPM.*

*!A =

DRIVE TO BOOT FROM

*!B =

NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT

*!HL =

DMA ADDR FOR CCP

02A0 Proc

02A0 062C
LDK B+44
;ICCP/BIOS and don't read CBIOS

02AD 3A0400
LD A+COISK
;Current logged in drive

02B0 ZAD2E8
LD HL+CCPADR

Copyright © 1982 Osborne Computer Corporation
02B3 BCPM:
  !Boot CPM from disk
ENTRY
IA = DRIVE TO BOOT FROM
IB = NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT
IH = DMA ADDR FOR CCP

:EXIT
:None
:LOOPS ON ERR

02B3 PROC

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

02B3 3217EF STD A,*SDISK /*Set drive to boot from
02B6 220EFF STD HL,DMADR /*SET DMA
02B9 3601 LDX A,1
02BB 3214EF STD A,*SAVSEC /*Set sector
02BE 3D DEC A
02BF 3200EF STD A,*TEM /*MAKE TEM ZERO

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

02C2 C5 PUSH BC

02C3 CD80DB :RLOOP: CALL SENDEN /*DETERMINE DENSITY
02C6 2805 "02CD$ JRZ *GOOD /*IF GOOD
02C8 CD401 CALL EBOOT /*PRINT ERROR
02CB 18F6 "02C3$ JR *RLOOP
02CD D1 :GOOD: PDP DE /*=NUMBER OF 128 BYTE BLOCKS
02CE C5 PUSH BC /*SAVE NUMBER OF SECTORS IN ONE TRACK

*SET NUMBER OF SECTORS TO READ

02CF 3A00EF LD A,*SAVTYPE
02D2 C83F $ SRL A
02D4 C83F $ SRL A
02D6 E003 ANI 0000_0011B /*IA=NUMBER OF BYTES IN ONE SECTOR(0-3)
02DB FE00 CMP 0 /*IF 128 BYTES SECTORS
02DA 2818 "02F4$ JRZ 1 /*GET NUMBER TO DIVIDE BY

02DC 47 MOV B,A /*B=NUMBER OF BYTES IN ONE SECTOR(1-3)
02DD 3601 LDX A,1 /*TIMES TWO
02DF C827 $ :1LROOP: SLA A
02E1 10FC "02F4$ DJNZ :1LROOP

02E3 47 MOV B,A /*NUMBER TO DIVIDE BY
02E4 7A MOV A,0 /*A=NUMBER OF 128 BYTE BLOCKS
02E5 1600 LDK D,0 /*SUBTRACT WITH DIVISOR
02E7 90 :2LROOP: SUB B /*SAVE FLAGS
02EB 08 $ EX AF /*ICOUNT
02E9 14 $ EX AF /*RESTORE FLAGS
02EA 08 $ /*RESULT IS ZERO (NO PARTIAL SECTORS)
02EE 2807 "02F4$ JRZ 1 /*IF RESULT IS ZERO (NO PARTIAL SECTORS)
02ED 30F8 "02F7$ JR NC :2LROOP /*LOOP
02E5   ED44   $  NEE    A      1Z COMP
02F1   3200EF  STO    A+TEM  I SAVE REMAINDER AND INDICATE A PARTIAL SECTOR
02F4   C1   11    POP    BC  I B = NUMBER OF SECTORS IN ONE TRACK
02F5   4A    MOV    C,D  I C = NUMBER OF SECTORS TO READ

; READ SYSTEM
02F6   AF   XRA    A      IA=0
02F7   3215EF  ; LOP:  STO    A+SAVTRK  I SET TRACK

; CHECK FOR ALL SECTORS READ
02FA   79  MOV    A+C  I SECTORS TO READ
02FB   B7  ORA    A      I IF C IS NOT ZERO CONTINUE
02FC   2008  ~0306S  JNZ    15  I IF C IS NOT ZERO CONTINUE

; CHECK FOR NON PARCIAL SECTORS
02FE   3A00EF  LD     A+TEM
0301   B7  ORA    A      I YOUR DONE IF C=0 AND TEM=0
0302   2858  ~035C$  JNZ    14  I READ ONE MORE PARCIAL SECTOR

; UPDATE NUMBER OF SECTORS LEFT TO READ
0306   90   15    SUB    B      I SUBTRACT SECTORS IN ONE TRACK
0307   4F  MOV    C,A  I SAVE REMAINING SECTORS TO READ
0308   3005  ~030FS  JNC    12  I #B MORE THAN ONE TRACK LEFT TO READ

; IF THIS IS LAST TRACK ZERO NUMBER OF SECTORS LEFT TO READ
030A   ED44   $  NEE    A      1Z COMP
030C   47  MOV    B+A  I READ ALL THE REMAINING SECTORS
030D   0E00  LDK    C,D  I STOP AFTER THIS READ

; CHECK FOR NON ZERO VALUE IN TEM AND THE LAST SECTOR TO READ
030F   3A00EF  LD     A+TEM
0312   B7  ORA    A      I IF TEM IS ZERO SKIP THIS
0313   2808  ~0310S  JNZ    13  I IF TEM IS ZERO SKIP THIS
0315   AF   XRA    A      I IF REG C IS NOT ZERO SKIP THIS (NOT LAST TRACK)
0316   81  ORA    C
0317   2004  ~0310S  JNZ    13  I 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      IA=A-1
031B   2813  ~0330S  JNZ    13  I ONLY ONE SECTOR LEFT TO READ

; READ ONE TRACK
031D   CD5808  CALL    RSEC  I READ (BC IS SAVED)
0320   2805  ~0327$  JNZ    1600  I IF GOOD
0322   CD9401  CALL    EBOOT  I REPORT ERROR
0325   10F6  ~0310S  JR     13
UPDATE DMA

0327  220FEF  ;GO:
          STD  HL:DMADR
          ISET DMA

UPDATE TRACK

032A  3A15EF  ;LD
          A:SAVTRK
032D  3C    ;INC
          A
033E  18C7  ^02F7$  ;JR
          1 LOOP
          ;TRACK LOOP

READ A PARCIAL SECTOR

0330  E5    ;RLS:
          PUSH HL
          ;SAVE ADDRESS TO WRITE TO
0331  2180EA  ;LDK
          HL:DEAB0H
          ;ADDRESS OF HOST BUFFER IN BIOS
0334  220FEF  ;STO
          HL:DMADR
          ;SET DMA
0337  2115EF  ;LDK
          HL:SAVTRK
033A  35    ;DEC
          [HL]
033B  0601  ;LDK
          B:1
033D  CD750C  ;3 LOOP:
          CALL READ
          ;READ ONE SECTOR
0340  2B05  ^0347$  ;JRZ
          1 GO
          ;IF GOOD
0342  CD9401  ;CALL
          E800T
0345  18F6  ^033D$  ;JR
          13 LOOP
          ;REPORT ERROR

ISET NUMBER OF BYTES TO TRANSFER

0347  3A00EF  ;GO:
          LD  A:TEM
034A  47    ;MOV
          B:A
034B  210000  ;LDK
          HL:0
034E  118000  ;LDK
          DE:128
0351  19    ;4 LOOP:
          ADD HL:DE
0352  10F0  ^0351$  ;DJNZ
          1 LOOP
0354  E5    ;PUSH
          HL
0355  C1    ;POP
          BC
0356  2180EA  ;LDK
          HL:DEAB0H
          ;SOURCE
0359  01    ;POP
          DE
          ;DESTINATION
035A  E880  ;LDIR
          ;MOVE

CLEAR BUFFERS SET PARR. AND RETURN TO SYSTEM

035C  2150EF  ;LDK
          HL:HOSTACT
035F  110700  ;LDK
          DE:(LOGSEC-HSTACT)+1
0362  CD3009  ;CALL
          FILLZ
          ;Clear Host BIOS cells
0365  3EFE  ;LDK
          A:OFFH
0367  3255EF  ;STO
          A:UNASEC
036A  3E7F  ;LDK
          A:VLL-1
036C  326EEF  ;STO
          A:LOTRK
036E  AF    ;XRA
          A
0370  C9    ;RET
          ;Clear error indicator
;KEY = 1 ;production keyboard

; Control keys
= 0007 CBELL = "C"-40h ;Ring the Bell
= 0008 MCUP = "K"-40h ;Move cursor up
= 0009 MCDOWN = LF ;Move cursor down
= 000A MCLEFT = BKs ;Move cursor left
= 000C MCRIGHT = "L"-40h ;Move cursor right
= 001A VCLRS = "I"-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: = '"' ;Screen Addressing
= 0051 VINC: = '"' ;Insert Char
= 0057 VDEL: = '"' ;Delete char
= 0045 VINL: = '"' ;Insert line
= 0052 VDELL: = '"' ;Delete line
= 0054 VCEDL: = '"' ;Clear to end of line
= 0029 VSHI: = '"' ;Start half intensity
= 0028 VEHI: = '"' ;End
= 006C VSUL: = '"' ;Start underline
= 006D VEUL: = '"' ;End
= 0067 VSGH: = '"' ;Start graphics
= 0047 VEGH: = '"' ;End

;ARROW KEYS
= 008A UP = 8AH
= 008B RIGHT = 88H
= 008C DOWN = 86H
= 008D LEFT = 8DH
Keyboard and Console Routines

0371

SKEY:
Get status of keyboard

:EXIT
:IC bit set if no data ready

0371 3A59EF
0374 B7
0375 CB
0376 3A5EEF
0379 B7
037A CB
037B F6FF
037D C9

LD A,KEYLCK
OR A
LD A,LKEY
ORA A
RZ
ORI OFFH
RET

:if locked keyboard
:CHECK FOR ZERO
:IF NOT ZERO MAKE OFFH
:IF DATA
Keyboard and Console Routines.

037E C1:
037E RKEY:
;Read next key from keyboard

;EXIT
;[last key

037E PROC
037E CD7103 CALL SKEY
0381 28FB ^037E$ JRZ RKEY ;if NC 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
Keyboard and Console Routines

**NORM**: Normalize to UPPER case

```assembly
:ENTRY
IA = char

:EXIT
IA = CHAR
```

```assembly
038F FE61 CMP 'a'
0391 DB RC
0392 FE78 CMP 'z'+1 ; if upper
0394 DD RNC
0395 0620 SUI 'a'-'A'
0397 C9 RET

= 002D REP C = 45 ; initial rep delay
= 0035 REP K = 5 ; repeat constant
```
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 LDX HL+3ELCNT
0399 AF XOR A
039C B6 OR [hl] ; cell=0?
039D 280C *O3AB$ JRZ 1z : if bell now off
039F 35 DEC [hl] ; bell is on, decrement counter
03A0 2009 *O3AB$ JRNZ 1z : if bell should stay on, already
03A2 3A62EF LD A*PIABD
03A5 E6DF AND 1101_1111b ; clear bell bit
03A7 4F MDV C+A
03AB CDT109 CALL CP8D
03AB 1z:
03AB 28 DEC HL ; HL => DACTIVE
03AC 7E LD A*[hl]
03AD B7 OR A ; RETURN if inactive
03AE C8 RZ
03AF 35 DEC [hl] ; reset delay
03BD CC36DE CZ DDRV ; if deselect drive
03B3 C9 RET
Bit definitions for ESCM flag byte
Note Bit 7 is currently free.

= 0040  EF_X:  =  64  186= extenting X-coordinate
= 0020  EF_SCR:  =  32  185= Screen/Cursor Addressing
= 0010  EF_ADR:  =  16  184= exepcting address-chr
= 0008  EF_ESC:  =  8  183=last char was ESC
= 0004  EF_UN:  =  4  182= Underline mode
= 0002  EF_MA:  =  2  181= Half Intensity mode
= 0001  EF_GR:  =  1  180= Graphics mode
= 0007  EF_MSK:  =  EF_UN*EF_MA*EF_GR  IMask to get mode.
| Vector (branch) table for video output mode selection controlled by ESCH mode |
|-----------------------------|-------------------|------------------|
| 0384 6704                   | DW VNORM          | 10 Normal mode   |
| 0386 8105                   | DW VGRAPH         | 11 Graphics mode |
| 0388 9504                   | DW VHALF          | 12 Half intensity mode |
| 038A 9904                   | DW VHA_GR         | 13 Half and graphics |
| 038C 8704                   | DW VUNDER         | 14 Underline mode |
| 038E 8804                   | DW VUN_GR         | 15 Under and graphics |
| 03C0 8F04                   | DW VUN_MA         | 16 Under and half intensity |
| 03C2 9304                   | DW VUN_MA_GR      | 17 Under and half and graphics |
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 addr

03C4 3D
03C7 53
03CA 67
03CD 47
03D0 29
03D3 28
03D6 6C
03D9 6D

03D0 A1
03DF 51
03E2 57
03E5 45
03E8 52
03EA 54
03EE 23
03F1 22

:end:

03F4 4404

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)

VALETS: = (tend-VALIDE)/3 ; # of entries in table
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 addr

03F6 00  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  MCRIGHT ! 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
0408 1E  DB  VHOME ! DW VC_HOME  Cursor home

040E 9E05  DW  VOUT97  ! No match--ignore undef control char

= 0008  VALCTS: =  ((17-2)-VALIDC)/3  ! Number of valid entries
0410 COUT
  ; General output routine to Video Screen
  ENTRY
  IC=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
  : =3, 5, 6, 7 As above, but mode is combination
  : =9-15 Last chr Esc flag True otherwise like 1-7.

0410 PROC

  :RESET VERTICAL OFFSET WITH VRTOFF

  0410 F5 PUSH AF
  0411 C5 PUSH BC
  0412 3A62EF LD A, PIABD
  0413 E6E0 AND $11100000
  0414 47 MOV B, A
  0415 3AEEFF LD A, VRTOFF
  0416 E61F AND $0001111B
  0417 80 ORA B
  0418 4E MOV C, A
  0419 CD7109 CALL CPBD
  0420 C1 POP BC
  0421 F1 POP AF
  0422 E5 PUSH HL
  0423 D5 PUSH DE
  0424 C5 PUSH BC
  0425 2A5AEF LD HL, CURS
  0426 A500EF LD A, ESCH
  0427 47 MOV B, A
  0428 E608 AND EF, ESC
  0429 2023 $0455 JRNZ PTESC
  042A 79 MOV A, C
  042B FE1B CMP ESC
  042C 78 MOV A, B
  042D 2815 $0440 JRZ ESC

; Current chr is NOT ESCaped. Is this chr ESC?

  042E E608 AND EF, ESC
  042F 2023 $0455 JRNZ PTESC
  0430 79 MOV A, C
  0431 FE1B CMP ESC
  0432 78 MOV A, B
  0433 2815 $0440 JRZ ESC
0438 E5 push HL
0439 21B403 LDX HL+ESCH7B
043C 607 AND EF_MSK ;Mode bits only
043E B7 ADD A+A ;Times two
043F 5F MOV E,A
0440 1600 LDX D+0 ;DE = offset
0442 19 ADD HL+DE ;HL = tbl addr
0443 vector:
0444 7E LD A,[HL] ;entry point, note hl on stack
0445 66 LD H+[HL] ;1st byte (low order addr)
0446 6F MOV L,A ;2nd byte (hi order addr)
0447 E3 XTHL ;ML=addr from table
0448 79 MOV A+C ;Restore hl from stack ;stack=tbl addr
0449 C9 RET ;Enter routine per table addr:
044A 78 MOV A+8 ;recall ESC value
044B 18E8 ^0438$ JR tout ;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 STA A+ESCH
0452 C39E05 JMP VOUT97 ;Exit
0455  PSTESC:
| Last chr was ESC |
| Entry          |
| EA = EF_ESCAPE |
| EB = ESC     |
| EC = Char to output |
| HL = curs |

0455  Proc
0456  CB60  $   Bit 458
0457  2075 ^04CE$  JNZ  SETXY

; Is this chr really an address?
; **** If chr is part of an addr

; No cursor/screen addressing in effect:
0459  A8  XOR  B
045A  47  MOV  B,A  ; Set up B = ESC byte value.
045B  3260EF  STO  A,ESCH
045E  E5  PUSH  HL
045F  21C403  LDK  HL+VALIDE
0462  1610  LDK  E,VALETS
0464  79  MOV  A,C  ; Chr to A
0465  1815 ^04C6$  JR  LOOKUPB  ; Go to routine to branch per tbl
SORMAC 808x Assembler ver 3.5E <r>5/57= 59:92 Page 31
C:SDROMA *ASM

Keyboard and Console Routines.

0467
VNorm:
INORMAL mode character processing.
ENTRY
IA = char to output
IHL=curs
0467 FE20 CMP =
0469 380B *0476$ JRC :2 :IF control chr
VRIGHT:
0468 DI
046C ENADIM 19th bit memory
046E 3680 STD 8RBIT.[hl] :set this chr BRIGHT
0470 DISDIM
0472 EI
0473 C38105 JMP VOUT80

:2:
0476 E5 PUSH HL :Save Curs
0477 21F603 LDK HL+VALIDC :Branch table addrs
047A 1E08 LDK E+VALCTS :Table size
JMP LOOKUP8 :Scan table of valid control chrs
JMP :and branch to appropriate routine.
LOOKUPB:

$HL=1st byte of table (match code)
$E=table size (no. of entries)
$A=value to scan for possible match

047C

LOOKUPB:

逻辑 to scan 3 byte branch table

NOT a subroutine—do not CALL.

ENTRY

$HL=1st byte of table (match code)
$E=table size (no. of entries)

2 byte "No-Match" addr

Stack has HL saved as top entry

IC=char

COP $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 11st byte of next entry

0482 1D DEC E (Dec count of entries remaining

0483 20F7 *047C$ JNZ LOOKUPB Continue thru body of table

0485 18BC *0443$ JR VECTOR: No-Match, hl=points to vector
SORCIM 808x Assembler ver 3.5E "1/55/7= 5/192 Page 33

Keyboard and Console Routines.

; Processing for modes other than normal.
; --------------------------------------

; VGRAPH = VOUT80 ; Normal mode EXCEPT: cntl chr's are printed

0487 VUNDER:
; Underline only
0488 FE20 CMP 0
0489 380C 0467$ JRC VNORM ; if cntl-chr, process as normal
048A JR VUN_GR ; continue

048B VUN_GR:
; Underlined Graphics
048C F680 OR 80h ; underline bit
048D 180C 0468$ JR VBRIGH ; set this chr BRIGHT

048F VUN_HA:
; Underline and Half intensity
048F FE20 CMP 0
0490 3804 0467$ JRC VNORM ; if cntl-chr, process as normal
0491 JR VUN_HA_GR

0493 VUN_HA_GR:
; Underline, Half Intensity, Graphics
0493 F680 OR 80h ; set underline bit
0494 JR VHA_GR

0495 VHALF:
; Half Intensity
0495 FE20 CMP 0
0497 38CE 0467$ JRC VNORM ; if cntl-chr, process as normal
0498 JR VHA_GR

0499 VHA_GR:
; IC=Chr, HL=Curs
0499 DI
049A ENADIM
049C 3600 ; STO DIMBIT,[hl] ; set dim field bit
049D ; LD E,[hl] ; diagnostic
049E DISDIM
04A0 EI
04A1 C38105 ; JMP VOUT80 ; continue
SCREEN: SetXY for Screen movement
ENTRY
IB = ESC
IA = new coord val, NO OFFSET

04A4 Proc
04A4 C870 $ BIT 6+B
04A6 2012 ^04BA$ JRNZ :sx X-coordinate

:9Y:
04AB E61F AND 0001_1111b =mod 32
04AA 4F MOV C+A
04A8 32EFEF STO A+PIA8D SET VERTICAL OFFSET FOR COUT
04AE 3A62EF LD A+PIA8D
04B1 E6ED AND 1110_0000b
04B3 B1 OR C
04B5 4F MOV C+A
04B7 CD7109 CALL OP6D SET Y coordinate
04B8 182E ^04E8$ JR :exitY

:9X:
04BA 87 ADD A+A Idouble A
04BB C60A ADD A+VFLO IPA 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 JRA C FOR IN HORIZONTAL OFFSET
04C6 4F MOV C+A
04C7 CD6409 CALL OP6D FUNCTION PIA
04CA C8AB $ CBIT 5+B finished screen-addressing
04CC 1824 ^04F2$ JR :exitX
SOLCOM 808x Assembler ver 3.5E C:\S5\79\92 Page 33

Keyboard and Console Routines

04CE

SETXY:
: Set X-Y value for Cursor/Screen Addressing
ENTRY
HIL = cursor_addr
IB = ESCH
Ic = 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 icX :if X coordinate

04DD :cy:
04DE 67 MOV H=A :save
04EE 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 nybit
04E7 67 MOV H=A :HL= new cursor addr
: JR :exitY

04E8 :exitY:
04EB 3E40 LDK A=0100_0000b :next addr-chr will be X-coord
04EA 80 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 4D AND B :finished addressing: reset addr bits
04F5 :exit2:
04F5 3260EF STO A=ESCH
04F8 C39605 JMP VOUT90
Control Code character processing

04FB  VC_HOME: ;Home Cursor
04FB     CD1907    CALL      UN_CUR
04FE     3A62EF    LD        A,PIABD
0501     1F        RAR       :bit0 => CY
0502     2ED0      LDK       L+0     ;CY >= bit7, trash bit0
0504     CB1D      $         RR       L
0506     67        MOV       H+A
0507     182F ^0538$    JR        @fixhl    ;HL := HL or F000h
0509  CD1907 CALL  UN_CUR  ;old cursor must be on stack
050C  E5  PUSH  HL
050D  0180FF LDK  BC, (-VLL)
0510  1816  ~0528$ JR  fwa  ;...at this entry point

0512  VC_BKS:
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 L_LIMIT
051A  28  DEC  HL
0518  1879  ~0596$ JR  VCUT90  ;Exit

051D  :wrap:
051D  E5  PUSH  HL  ;save old cursor
051E  017FF LDK  BC, (-VLL+1)
0521  09  ADD  HL, BC
0522  3A6CEF LD  A, L_LIMIT  ;L_LIMIT = #columns on screen
0525  4F  MOV  C, A
0526  0600 LDK  B, 0
0528  :fwa:
0528  09  ADD  HL, BC
0529  E3  XTHL
052A  29  ADD  HL, HL
052B  3A62EF LD  A, PALBD
052E  F6E0 OR  1110_0000b
0530  BC  CMP  H
0531  E1  PDP  HL  ;get new cursor...  
0532  2004  ~0538$ JRNZ  :fixhl  ;if NOT @video home  
0534  0100CC LDK  BC, (24*VLL)  ;wrap constant
0537  09  ADD  HL, BC
0538  :fixhl:
0538  3EF0 LDK  A, OFOh
053A  84  DR  H  ;modulo result; keep cursor
053B  67  MOV  H, A  ;inside video memory
053C  1858  ~0596$ JR  VOUT90
VC_BEL:  Ring the bell via setting PIAB 2#5 bit

053E  3A62EF  LD  A,PIABD
0541  F620    OR  0010_0000b  ibell bit
0543  4F      MOV  C+A
0544  C07109  CALL  OPBD  ifunction PIAB
0547  3E1E    ADD  A+30  iring bell for 30 ticks
0549  326BEF  STU  A+BELCNT  1.5s = 1/2 second
054C  1850  *059E$  JR  VOUT97  iexit no change
<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Assembly</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0557</td>
<td>05</td>
<td>DS</td>
<td>PUSH DE</td>
</tr>
<tr>
<td>0558</td>
<td>DDEF</td>
<td></td>
<td>PDP IX</td>
</tr>
<tr>
<td>0559</td>
<td>CD0507</td>
<td>CALL</td>
<td>VLDIR</td>
</tr>
<tr>
<td>0560</td>
<td>E6E0</td>
<td>AND</td>
<td>not1_1111b</td>
</tr>
<tr>
<td>0562</td>
<td>4F</td>
<td>MOV</td>
<td>C+A</td>
</tr>
<tr>
<td>0563</td>
<td>CD7109</td>
<td>CALL</td>
<td>OPBD</td>
</tr>
<tr>
<td>0566</td>
<td>AF</td>
<td>XRA</td>
<td>A</td>
</tr>
<tr>
<td>0567</td>
<td>3200EF</td>
<td>STO</td>
<td>A+VRTOFF</td>
</tr>
<tr>
<td>056A</td>
<td>2100F0</td>
<td>LDK</td>
<td>HL+FMVAVM</td>
</tr>
<tr>
<td>056D</td>
<td>1827~0596$</td>
<td>JR</td>
<td>VOUT70</td>
</tr>
</tbody>
</table>

;clear 1st line
;clear remaining lines
;Reset for 1st line of display mem
;ZERO A
;SET VERTICAL OFFSET FOR COUT
;new cursor
;Exit
Keyboard and Console Routines

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  CD1706 CALL DO_LF ; Line Feed
0578  1819 0596$ JR  VOUT90
VC_MCRT: Move Cursor Right
CALL UN_CUR
LD A, (hl)
JR VOUT80 ; re-echo current chr
SORCIM 808x Assembler ver 3.5E <:/55/7= 59:92 Page 42

Keyboard and Console Routines

: 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  50   MOV   E+L ;E = col(cursor)
0583  C888   S   CBIT   T+X ;Y = last_legal_col
0584  3A6CEF   LD   A+LIMIT
0585  3D   DEC   A ;IA = last_legal_col - col(cursor)
0586  93   SUB   E
0587  2009 ^0595$   JRNZ   VOUT85 ;if not LIMIT
0588  3680   LDK   A+00h
0589  A5   AND   L
058A  6F   MOV   L+A
058B  CDAA06   CALL   DD_LF2 ;...and LF.
058C  1801 ^0596$   JR   VOUT90

0595    VOUT85:
0595  23   INC   HL ;move cursor

0596    VOUT90:
0596  7E   LD   A+[hl] ;This exit path turns on 00h 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  E1   POP   BC
059F  E1   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
Keyboard and Console Routines:

**ESC_LCK:** Lock Keyboard

```assembly
05A3  ESC_LCK:  :Lock Keyboard
05A3   Proc
05A3   AF     XDR     A
05A4   1802  ^05A8$  JR     :2

05A6  ESC_ULK:  :Unlock Keyboard
05A6   3EFF     LDX     A+OFFh
05A8   :2:
05A8   3259EF   STO     A+KEYLCK
05A9   1BF1  ^059E$  JR     VOUT97
```
Keyboard and Console Routines

$05AD$  EEOE: Erase to end of line
$05AD$  E5    PUSH  HL ; save cursor
$05AE$  C08406  CALL  CLRNL
$05B1$  E1    POP   HL
$05B2$  18E2 ^0596$  JR   VOUT90
SORCIM 808x Assembler ver 3.5E <1/55/7= 59:92 Page 45
Keyboard and Console Routines

ESC_CAD:
; Cursor Addressing
0584 3E07  ; LDK A*EF_MSK
0586 A0    ; AND B
0587 F618  ; OR EF_ESC or EF_ADR ; next chr will be Y-coord
0589 :exit3:
0589 3260EF ; STO A*ESCH
058C 18E0 ^059$  ; JR VOUT97

ESC_SAD:
; Screen Addressing
058E 3E07  ; LDK A*EF_MSK
05C0 A0    ; AND B
05C1 F638  ; OR EF_ESC or EF_ADR or EF_SCR
05C3 18F4 ^058$  ; JR :exit3
Subroutine for use with EDEL and EINSRT:
Calculate #chars to move; if move zero chars, never return.

Proc

05C5
05C6
05C7 4D
05C8 CBB9 $
05CA 91
05CB 2804 ^05D1$
05CD 4F
05CE 0600
05D0 C9

end:

05D1
05D2 E1
05D3 18C1 ^0596$
SORCIM 808x Assembler ver 3.5E <17/12/12> 59:12 Page 47

Keyboard and Console Routines.

05D5
05D6 E5
05D7 CDC505
05D8 $:
05D9 E5
05DA DDE1 $:
05DB 23
05DC CD0507
05DD 3620
05DE 3680
05DF 18A8 ^0596$
SIRCIM 808x Assembler ver 3.5E <057/7= 59:92 Page 49

Keyboard and Console Routines:

; ESC-Sequence processing.
0607 3E01 ESCRG: LDK A+EF_GR 125
0609 1806 ^061$ JR K

060B 3E02 ESCSHA: LDK A+EF_HA 125
060D 1802 ^061$ JR

060F 3E04 ESCSUN: LDK A+EF_UN 125
0611 80 OR B
0612 130 STO A+ESCH
0615 1887 ^059$ JR VOUT97

0617 3EFE ESCCGG: LDK A+NOT EF_GR 140
0619 1806 ^062$ JR

061B 3EF0 ESCCHA: LDK A+NOT EF_HA 140
061D 1802 ^062$ JR

061F 3EF8 ESCCUN: LDK A+NOT EF_UN 140
0621 A0 AND B
0622 18EE ^061$ JR 130

= 054E ESCZZ: = VC_CLRSH ESC-Z Clear screen -same as Control-Z routine.
SORCIM 808x Assembler ver 3.5E <1/55/7= 59:92 Page 50

Keyboard and Console Routines:

0624
ESCR:
:Delete Line

0624
ESCEE:
:Insert Line
:ENTRY
:HL = cursor
:IC = chr

:EXIT
:screen updated
:HL = new cursor ...to WOUT90

0624
Proc
0627
CD1907 CALL UN_CUR
0629
3E8D LDK A,1000_0000d
062A
A5 AND L
062B 6F MOV L+A
062C E5 PUSH HL
062D 29 ADD HL+HL
0630 3A62EF LD A,PIA8D
0633 C618 ADD A+24
0634 94 SUB H
0635 E61F AND 0001_1111b
0636 47 MOV B+A
0638 3E52 LDK A+VDELL
0639 89 CMP C
063A 78 MOV A+B
063B 2B23 ^065F$ JRZ :delit
063C
:insrt:
:Insert a line

063C
84 ADD H
063D 57 MOV D+A
063E 1E00 LDK E+O
0640 CB1A $ RR D
0642 CB18 $ RR E
0644 1B DEC DE
0645 78 MOV A+B
0646 2180FF LDK HL+VLL
0647 19 ADD HL+DE
0648 1806 ^0652$ JR :istrt
064A
:icont:
064C 05 PUSH DE
064D DDE1 $ POP IX
064F CDF106 CALL VLDDR
0652 CD7506 :istrt: CALL :ivmod
0655 20F5 ^064C$ JRNZ :icont
0657 23 INC HL

:do CR
:save new cursor
:IA = addr(25th line)
:IA = lines_to_move + 1
:mod 32
:recall #lines to move
:if deleting a line
:IA = addr(25th line)
:shift right DE
:IDE = addr(1st chr on 1st line)
:IA = #lines to move
:DE or F000h; HL := HL or F000h
:DE := DE or F000h; HL := HL or F000h
:move 1 line down
:if must move more lines
:HL => 1st chr of new line
SORCIM 8D8x Assembler ver 3.5E <1/557= 59:92 Page 51
Keyboard and Console Routines.

0658  :exit:
0658  CD8406  CALL  CLRLN  :recover cursor
0658  E1     POP  ML      :Main Exit
065C  C39605  JMP  VCUT90

065F  :delt:
065F  D1     POP  DE      :recover new cursor
0660  D5     PUSH DE
0661  218000  LDR  HL+VLL
0664  19     ADD  HL+DE    :HL = line_below_cursor
0665  1806  *066D$   JR  :dstrt

0667  :dcont:
0667  D5     PUSH DE
0668  0DE1   $     POP  IX
066A  CD0507  CALL  VLDIR  :move 1 line up

066D  :dstrt:
066D  CD7566  CALL  :vmod
0670  20F5  *0667$  JRNZ  :dcont
0672  EB     EX  HL+DE    :get addr of line to clear
0673  18E3  *065B$  JR  :exit
Keyboard and Console Routines

; HL = HL or F000h; DE = DE or F000h;
; Simple mod-4096 arithmetic to keep pointers INSIDE video memory

<table>
<thead>
<tr>
<th>Line No</th>
<th>Opcode</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0675</td>
<td>F5</td>
<td>PUSH AF</td>
<td>save A = #lines to move</td>
</tr>
<tr>
<td>0676</td>
<td>3EF0</td>
<td>LDK A+OF0h</td>
<td></td>
</tr>
<tr>
<td>0678</td>
<td>B4</td>
<td>OR H</td>
<td></td>
</tr>
<tr>
<td>0679</td>
<td>67</td>
<td>MOV H+A</td>
<td></td>
</tr>
<tr>
<td>067A</td>
<td>3EF0</td>
<td>LDK A+OF0h</td>
<td>set upper nybl of H</td>
</tr>
<tr>
<td>067C</td>
<td>B2</td>
<td>OR D</td>
<td></td>
</tr>
<tr>
<td>067D</td>
<td>57</td>
<td>MOV D+A</td>
<td></td>
</tr>
<tr>
<td>067E</td>
<td>010000</td>
<td>LDK BC+VLL</td>
<td>modulo 4096</td>
</tr>
<tr>
<td>0681</td>
<td>F1</td>
<td>POP AF</td>
<td></td>
</tr>
<tr>
<td>0682</td>
<td>3D</td>
<td>DEC A</td>
<td>decrement line_count</td>
</tr>
<tr>
<td>0683</td>
<td>C9</td>
<td>RET</td>
<td></td>
</tr>
</tbody>
</table>
CLRLN:
;iClear to end of line
;ENTRY
;IML = Cursor

;EXIT
;iclear to EOL
; Uses All

PROC

0684 3620 STO '*',[hl] ;iclear cursor***
0686 3680 STO BRTBIT,[hl] ;iset cursor 9RIGHT
068D 3A6CEF LD A+LIMIT
0691 3D DEC A ;max_cols => maximum_col_
0692 5D MOV E+L
0693 C888 $ CBIT 7+8
0695 93 SUB E ;IA = col(EDL) - col(cursor)
0696 C8 RZ ;if #EDL+ done
0697 3004 069D$ JRNC ;2 ;if inside logical_video_line
0699 3EB0 LDK A+VLL
069B 93 SUB E ;else clr to end of 128-char line
069C C8 RZ ;if cursor @ column #127

4F

069D 4F MOV C+4
069E 0600 LDK B+0 ;IBC = chr to move
069F 0E9F MOV E+L
06A0 0E5 MOV D+H
06A1 DDE1 $ ;PUSH HL
06A2 D023 $ ;INC DE
06A3 185E $ ;INC IX
06A5 0705$ JR VLDIR
Keyboard and Console Routines.

06A7

06A7 DD_LF1
; Do Line Feed processing
;ENTRY
;IHL = cursor_addr

;EXIT
;Cursor cleared
;IHL updated for current cursor pos
;window moved if necessary

06A7 PROC
06A7 CD1907 DD_LF2: CALL UN_CUR ;clear cursor
06AA E5
06AB 01B00D PUSH ML ;save original cursor
06AE 09 LDK BC+VLL ;line length
06AF 3004 +06B5$ ADD HL+BC ;not wrapping from LWAM to FWAM
06B1 0100F0 LDK BC*FWAM
06B4 09 ADD HL+BC ;IHL = new cursor at top of VM

:nowap
06B5 E3 XTHL ;save new cursor, get old
06B6 29 ADD HL+ML ;shift HL left
06B7 3A62EF LD A*PIABD
06BA C617 ADD A+23 ;start + 23 = last video line
06BC 94 SUB M ;IA = l_line - curr_line
06BD E61F AND 0001_1111b ;modulo 32
06BF 28D2 +06C3$ JRZ ;mov if cursor is on 24th line of screen
06C1 E1 POP HL ;get new cursor
06C2 C9 RET

:vmov:
06C3 3A6CEF LD A*LIMIT
06C6 CB3D S SRL L ;unshift L register
06C8 95 SUB L ;IA = LIMIT - 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.

06C8 E1 POP HL
06CC E5 PUSH HL
06CD 3880 LDK A+80h
06CF A5 AND L
06D0 6F MOV L:A ;IHL = beginning of line
06D1 CD940B CALL CLR1N ;erase to EOL
06D4 3A62EF LD A*PIABD
06D7 47 MOV B:A
06D8 E6E0 AND not 31 ;IA = line zero
06DA 4F MOV C:A ;IC = housekeeping bits 5..7
06DB 381F LDK A+31
06DE 04 INC B ;increment line#
06DF 0A AND B ;IA = line#
06E2 81 OR C ;IA = new line# OR housekeeping_bits
06E3 4F MOV C:A ;IC = new value for DPBD
06E4 CD7109 CALL DPBD ;move video screen down 1 line in memory
06E7 E1 POP HL
06E8 C9 RET
STODIM:
:STORE THE CONTENTS OF THE B REG IN THE ADDR POINTED TO BY THE HL PAIR

:ENTRY
:IB = VALUE
:IHL = ADDRESS

:EXIT
:EXIT

06E9 PROC
06E9 DI
06EA ENADIM
06EC 70 STO B,[HL] ;ENABLE DIG
06ED DISDIM ;STORE
06EF EI
06F0 C9 RET
Soccer 808x Assembler ver 3.5E 1/2/87 = 59:192 Page 56
Keyboard and Console Routines

06F1
:VLDR:
:Video Block Move
:ENTRY
:BC+ IX, HL set
:EXIT
:LD0R on main & 9th bit memory
:Uses BC, DE, HL, IX

06F1 DDE5 $ PUSH IX
06F3 D1 POP DE
06F4
06F7 EDB8 $ PUSHAL
06F9 POPALL :main memory
06FC D1 ENADIM
06FD EDB8 $ LD0R :9th bit memory
0701 DISDM
0703 EI
0704 C9 RET

0705
:VL0R:
:Video Block Move
:ENTRY
:BC+ IX, HL set
:EXIT
:LD0R on main & 9th bit memory
:Uses BC, DE, HL, IX

0705 DDE5 $ PUSH IX
0707 D1 POP DE
0708
0708 EDB0 $ LD0R :main memory
070D POPALL
0710 D1 ENADIM
0711 EDB0 $ LD0R :9th bit memory
0715 DISDM
0717 EI
0718 C9 RET
UN_CUR:
;Undo/Invert Cursor
;ENTRY
;HIL = 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
:KEYBOARD SCANNING & DECODE

071F

GKEY:
:KEYBOARD INTERRUPT PROCESSOR
:ENTRY
:NONE

:EXIT
:KEYBOARD PROCESSING DONE, RESULT IN LDKEY.
:SYSTEM CLOCK UPDATED BY UPTIM.

071F
PROC

0720 ED736FEF $
0724 3199EF
0727 ALPUSH

072F CD9803
CALL UPTIM

0732 3A59EF
0733 B7
0736 C43C07
0739 C31E00

0738 JMP EXITI

;EXIT INTERRUPT
This file contains the 2-key roll over keyboard driver for
the OSBorne I computer.

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 processed. If the key is still
on, it is placed in Ikey (for 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:

KBDVR - 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 KBESRV 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 IKEY for the BIOS 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.
S0RDIM 808X Assembler ver 3.5E  c:55/7= 59:92  Page 60
CISORDMA  .ASM

Keyboard and Console Routines.

073C KBDRVR:
   Routine name: KBDRVR - Detects and processes keystrokes.
   Input: none
   Output: LKEY contains keystroke.

073C  PROC
073C  CD7407
073C   call kbscan ; Scan keyboard and enter keys into keylist
073C   ; Examine keylist. If key found in keylist, call CHKKEY to see
073C   ; if key is still on. Remove from list when not on.
073C  21D4EF
073F  LDK h1=keyst1st ; Point to first entry of keylist
073G  0603
074H  LDK b=kl_len
074H  10: id a=1key
074H  or a
074H  RNZ ; Exit when a key is waiting in LKEY
074H  7E
074A  LDK h1= [hi] ; Get byte 0 of entry
074A  C87F
074E  bit kl_used+a
074E  JRA :40 ; If entry is in use then
074E  0407
074G  call CHKKEY ; Check if still on
074G  jrnz :20 ; If key is now off then
074G  2057
0751  2004 ^0757$ ; Remove key from list
075R  3600
075E  LDK 1818 ^076FS$ ; If else
075E  20: id
075E  ; Key is on. decrement its repeat count. If count goes to zero
075E  ; then it is time to service the key.
075E  95
075F  push h1 ; Save ptr to first byte of entry
075F  23
075G  inc h1 ; Point to repeat count
075G  35
075G  dec [hi] ; Exit when not time to service the key.
075G  5A 2012 ^076ES$
075H  Jr :30
075H  ; It is time to service the key. Set the next repeat count
075H  075C  E3
075C  ex [sp]=h1 ; Point back to the first byte of entry
075C  7E
075D  LDK a=1hi
075D  C877
075E  bit ky_srvd+1a
075E  SBIT ky_srvd+[hi]
075E  JRZ :22 ; If check if it is serviced before
075E  :22
0756  3618 ; Set the serviced once flag
0756  3606
0757  LDK 2802 ^0769$ ; Point back to the repeat count
0757  3602
0758  LDK 3606
0759  222
0759  E63F
075A  130
075B  call KBserv ; Call service the key
075C  E3F
075E  140
075E  pop h1 ; Get ptr to first byte of entry again
075E  ENDIF ; Send if
075F  C9
076G  RET ; Return to exit if code in ROM
**KBSCAN**

**0774 proc**

**0774 2EFF LDK 1:0ffh** : see if any key pressed

**0776 CDEFO7 call rdrow** : return when none

**0779 CB rz**

**077A 2E81 LDK 1:row0_m** : get row 0

**077C CDEFO7 call rdrow**

; LDK L+1
; CALL RDRDW
; PUSH psw
; LDK L+80h
; CALL rdrow
; ANI 08h
; MOV B+A
; PDP PSW
; OR B

**077F E6E3 and 11100011b** : remove ctrl/shift and alpha lock

**0781 0607 LDK b:tot_row**

; in this loop, reg b contains totrow- current row being scanned

**0783 2844 ^07C9$ jrz $50** : if any key is pressed then

**0785 C5 push bc** : save loop count

**0786 5F mov a=a** : e = columns

**0787 3E07 LDK a:tot_row**

**0789 90 sub b**

**078A 17 ral**

**0788 17 ral**

**078C 17 ral**

**078D 57 mov da** : Id = 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: jrnc $30** : 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 keylist 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** : kc = key number

**0798 D5 push de** : save de

**0799 E5 push hl** : save hl

**079A 0603 LDK b:kli_len** : length of keylist

**079C 2104EF LDK h:keystk**
079F 110000 LDK  dev=0
07A2 7E 118:  ld  a=\{hi\}
07A3 CB7F $  bit  kl_used+\{a\}
07A5 2807 *07AE$  jrz  t22
07A7 E63F and  krow_m+kcol_m
07A9 B9  cmp  c
07AA 2814 *07C0$  jrz  t27
07AC 1802 *0780$  jr  t25
07AE 50  mov  e=1
07AF 54  mov  d=\{hi\}
07B0 $ 25:  echo  kle_len

07B2 10EE *07A2$ ENDM djnz  :18

07B4 7A  check if an empty entry was found.
07B5 87  mov  a+d
07B6 CAC007 or  a
07B9 E8  js  t27
07BA 71  ex  de=\{hi\}
07BB CBFE $  sto  c=\{hi\}
07BD 23  sbit  kl_used+[\{hi\}]
07BE 3601  inc  hi
07C0  $ 27:  sto  DB_c,t+[\{hi\}]

07C0 E1  pop  hi
07C1 01  pop  de
07C2 C1  pop  bc
07C3  :30:  inc  e
07C4 AF  xor  a
07C5 BB  cmp  e
07C6 20C8 *0790$ jrnz  :15

07C8 C1  pop  bc
07C9 CB25 $  sla  l
07CB CDEF07 call  rdown
07CE 1083 *0783$ djnz  :10
07DD C9  ret

; if entry is used then
; check with current key
; exit if this key is in list
; else (an empty entry is found)
; save address of empty entry in de
; endif

; next entry
; still list scanned

; if empty entry was found then
; hi = empty entry
; store the key in the list
; set used flag
; store debounce delay
; endif
; restore all registers

; endif
; increment column number

; until all columns scanned
; restore bc
; move to next row
;.cbret: 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 h\l
07D2 F5 push af
07D3 1F rar
07D4 1F rar
07D5 1F rar
07D6 CDE507 call g\mask
07DA D5 push de
07DB CDE507 call g\mask
07DE E1 pop hl
07DF CDEF07 call rd\row
07E2 A3 and e
07E3 E1 pop hl
07E4 C9 ret
Keyboard and Console Routines.

```
; sbrt:  GTMASK - generates mask with one bit set.
; input: ia = bit number (0..7)
; output: ie = mask

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
```

# Keyboard and Console Routines

```assembly
;
; sbrt: RDROW - Reads a row of keys
; input:
; iREG L = lower 8 bits of adrs to read the row
; output:
; iREG A = row value

OTEF   RDROW: proc

OTEF  2622 LDK h*high(h.key) :hl = ptr adrs for given row
07F1  70 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*
**KB Serv**: services the key

```
:setup hl to point to keycode table entry for this key

07F8 5F MOV E,HL
07F9 1600 LDH D0
07FA 21C308 LDH hl+kycdtb
07FE 19 ADD HL,DE
07FF 7E LD A[HL]
0800 FE21 CEM H
0802 381F 08234 JNC L:12 ;ignore shift/ctl etc for chars less than 21h
0804 F5 PUSH AF
0805 2E01 LDS I,1
0807 CDE0F7 CALL RDRW ;get row containing ctrl,shift and alpha key
0808 2E80 LDS L,80H
080C CDE0F7 CALL RDRW
0810 E608 ANI 8
0812 5F MOV E,A
0813 F1 PDP AF
0814 B3 OR E
0815 5F MOV E,A
0816 F1 POP AF
0817 CB53 bit ctl_ky++
0819 202E 08494 JNZ L:50 ;igo process ctl key
0818 CB63 bit shft_ky++
081A 2014 08934 JNZ L:30 ;igo process shift key
081B CB85 bit alph_ky++
0821 2004 0827$ JNZ L:25 ;igo process alpha key

:store key code into lkey

0823 12:
0824 32EEFF STO A+LKEY
0826 C9 RET
0827 F1 CMP 'A'
0829 38FF 08234 JNC L:12 ;exit when less than 'A'. Alpha has no effect
082B 27:
082C FF1A CMP 80H
082D 30FF 08234 JNC L:12 ;ior when >= 80h
082F 28:
0830 EE20 XOR 20H
0831 18FF 08234 JR L:12
```
0833  FE61  cmp  9*
0835  30F4  0828$  jrcn  127  igt0 alpha when char > '9'
0837  FE5B  cmp  'F'
0839  3806  0841$  jrc  135  igt0 process shift numerics etc
083B  20F2  0825$  jrnz  128  invert shift bit for 'A'
083D  3E5D  LDK  a**7
083F  1BE2  0823$  jr  12  iconvert { to }
0841  35E  chars ' to > (ascii codes 27h to 3ah) are converted using
0843  1E   the shift_tb
0845  5F   mov  e+s
0847  21D408  LDK  hi+shift_tb = ****
0849  19    add  hi+de
084B  7E  id  a+[hi]
084D  180A  0823$  jr  12

0849  5F  if char is between a-zA then turn off the 3 high order
084B  1E   bits.
084D  7E  if char is between '!', '*','?' it is translated as per table ctl_tb
084F  1E   if char is the arrow keys or the ']'/'[' key the slide functions
0851  1E   are called.
0853  FE8D  cmp  lft_arw
0855  2833  0880$  jr  z slidel
0857  FE88  cmp  rt_arw
0859  2833  0884$  jr  z slider
085B  FC8A  cmp  up_arw
085D  2840  0895$  jr  z slideu
085F  2840  0899$  cmp  dn_arw
0861  7E  jr  z slidcd
0863  FE8A  cmp  hm_scrn
0865  2850  08AD$  jr  z dohome
0867  C663  $  BIT  shift_keye  ttest for cnt1 shift
0869  2808  0869$  jr  z 52  if not
086B  FE2F  cmp  */*  is it ?
086D  2004  0869$  jr  z 52  if not */*
086F  3E7F  LDK  a+07fh  Idelet key
0871  180A  0823$  jr  12
0873  FE40  52E  cmp  A*        igt0 translate chars from table
0875  3808  0875$  jrc  54  z*+1
0877  FE78  cmp  */*  igt0 translate chars from table
0879  382$  0823$  jrc  112  no translation if char below '*'
087B  FE1F  and  1fh
087D  18AE  0823$  jr  12

0875  5F  mov  e+s
0877  384A  0823$  jrc  112  id=0 from above
SOURCIM 808x Assembler ver 3.5E C:\55\7= 59:92 Page 68
Keyboard and Console Routines.

087E 18C6 ^08466  jr :37
   ; slide functions:
0880 0E02                     LDK c+2
0882 1802 ^0886$  jr :70
0884 0EFE                     LDK c+2
0886 3A61EF :70;  ld a,piaad  ;get horizontal coord.
0889 81                           add a+c
088A 4F                             MOV C,A
0888 CD6409 :72;  CALL CPAD  1FUNCTION PIA
088E :74;  ; set repeat count for these keys (override count set by the kbdrvrf)
088E D1                             pop de  ;get return adrs
088F E1                             pop hl  ;pointer to repeat entry
0890 3603                          sto sid Rica h; repeat count for slide keys
0892 E5                             push hl
0893 D5                             push de  ;irestore stack
0894 C9                             ret
0895 0E01                     LDK c+1
0897 1802 ^0898$  jr :76
0899 0EFF                     LDK c+1
089B :76;  ;merge new vertoffset to lower 5 bits of PIA
089B 2162EF                     LDK hi+piabd
089E 7E                             ld a,(hl)
089F 81                             add a+c
08A0 E61F                             and 1fh  ;modify current with +/or-1
08A2 4F                             mov c+a
08A3 7E                             ld a,(hl)
08A4 E60E                             and 0e0h
08A6 :78;  ;dohome:
08A6 81                             or c
08A7 4F                             mov c+a
08A8 CD7109                          call opbd
08A8 18E1 ^088E$  jr :74
08AD 3A61EF                     LD A,PIAAD  ;GET OLD VALUE
08BA E601                     ANI 0000_000018  ;SAVE DENSITY BIT
08BB F6EA                     ORIVFLC  ;OR IN HORIZONTAL OFFSET
08B4 4F                             MOV C,A
08B5 CD6409                          CALL CPAD  ;FUNCTION PIA
08B8 3A62EF                     ld a,piabd  ;MOUSE KEEPING BITS
08BB E60E                             AND 0e0h
08BD 4F                             MOV C,A
08BE 3AEF0F                     LD A,VRTOFF  ;GET LAST VERTICAL OFFSET
08C1 1BE3 ^08A6$  jr :78  ;and the vert to 0 also
key code translation tables

<table>
<thead>
<tr>
<th>Key Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>08C3</td>
<td>18097F7F</td>
</tr>
<tr>
<td>08C3</td>
<td>ESC, TAB, ERC, ERC</td>
</tr>
<tr>
<td>08C7</td>
<td>7F0D2758</td>
</tr>
<tr>
<td>08C8</td>
<td>ERR, CR, ****, [ ]</td>
</tr>
<tr>
<td>08C9</td>
<td>31323334</td>
</tr>
<tr>
<td>08CF</td>
<td>35363738</td>
</tr>
<tr>
<td>08D3</td>
<td>'1', '2', '3', '4'</td>
</tr>
<tr>
<td>08D7</td>
<td>71776572</td>
</tr>
<tr>
<td>08D8</td>
<td>'q', 'w', 'e', 'r'</td>
</tr>
<tr>
<td>08DF</td>
<td>61736466</td>
</tr>
<tr>
<td>08DB</td>
<td>'t', 'y', 'u', 'i'</td>
</tr>
<tr>
<td>08DB</td>
<td>67686A6B</td>
</tr>
<tr>
<td>08DF</td>
<td>'s', 'd', 'f'</td>
</tr>
<tr>
<td>08E3</td>
<td>7A78637B</td>
</tr>
<tr>
<td>08E7</td>
<td>'a', 's', 'z', 'x', 'c', 'v'</td>
</tr>
<tr>
<td>08EB</td>
<td>626E6D2C</td>
</tr>
<tr>
<td>08EB</td>
<td>'b', 'n', 'm', 'v'</td>
</tr>
<tr>
<td>08EF</td>
<td>8A8D3020</td>
</tr>
<tr>
<td>08FD</td>
<td>8Ah, 8Dh, '0', '+'</td>
</tr>
<tr>
<td>08E9</td>
<td>2E706F39</td>
</tr>
<tr>
<td>08F0</td>
<td>'&gt;', 'p', 'o', '9'</td>
</tr>
<tr>
<td>08F4</td>
<td>888C2D2F</td>
</tr>
<tr>
<td>08F6</td>
<td>8Bh, 8Ch, '4', '*'</td>
</tr>
<tr>
<td>08F7</td>
<td>3B5C6C3D</td>
</tr>
<tr>
<td>08F8</td>
<td>'.'</td>
</tr>
</tbody>
</table>

shift_tb:

<table>
<thead>
<tr>
<th>Key Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>08FB</td>
<td>2200000000</td>
</tr>
<tr>
<td>0900</td>
<td>3C5F3E3F29</td>
</tr>
<tr>
<td>0905</td>
<td>2140232425</td>
</tr>
<tr>
<td>090A</td>
<td>5E262A2800</td>
</tr>
<tr>
<td>090F</td>
<td>3A002800</td>
</tr>
<tr>
<td>0913</td>
<td>7B1F7D7E</td>
</tr>
<tr>
<td>0917</td>
<td>8081828384</td>
</tr>
<tr>
<td>091C</td>
<td>8586878889</td>
</tr>
<tr>
<td>0921</td>
<td>00000060</td>
</tr>
</tbody>
</table>

ctl_tb:

<table>
<thead>
<tr>
<th>Key Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0913</td>
<td>'(', ')', '!', '!', '00h', '00h', '00h', '00h'</td>
</tr>
<tr>
<td>0917</td>
<td>80h, 81h, 82h, 83h, 84h</td>
</tr>
<tr>
<td>091C</td>
<td>85h, 86h, 87h, 88h, 89h</td>
</tr>
<tr>
<td>0921</td>
<td>00h, 00h, 00h, 60h</td>
</tr>
</tbody>
</table>
*{4}*

0925

CODEH:

; Compare DE to HL as unsigned integers.
; ENTRY
; DE, HL set to values to compare

; EXIT
; NZ bit set if DE = HL
; NZ bit set if DE < HL.
; Uses AF.

0925 7A
0926 BC
0927 C0
0928 7B
0929 BD
092A C0
092B 37
092C 69

MOVD A+D
CMP H
RNZ :If D ≠ H
MOVD A+E
CMP L
RNZ :If DE ≠ HL
RET
DELAY:
ENTRY
IA = Number of Milliseconds to delay
SCLRE = (Freq/1000)/25

EXIT
END

092D Proc
092D C5
092E 4F
092F 3E85
0931 3D
0932 40
0933 49
0934 C23109
0937 00
0938 C22F09
0938 C1
093C C9

SCLRE = 2000/22 / 2Hz, 2mhz
***defined in OCCTX.ost
FILLZ:  
; Fill block of memory with byte value.  
; ENTRY  
; DE=LENGTH TO broadcast character  
; IC=character TO broadcast  
; HL=FWA TO START broadcast  

:EXIT  
:FILL DONE  

093D 0E00 093F 7B 0940 82 0941 C8 0942 71 0943 1B 0944 7B 0945 B2 0946 C8 0947 42 0948 4B 0949 54 094A 5D 094B 13 094C E8 094E C9

093D FILLZ:  
093F 7B FILLZ:  
0940 82 MDV A+D  
0941 C8 OR D  
0942 71 RZ ; return here if broadcast 0 bytes  
0943 1B STD C+(hl) ; last byte  
0944 7B DEC DE  
0945 B2 MDV A+D  
0946 C8 OR D  
0947 42 RZ ; return here if broadcast 1 byte  
0948 4B MDV B+D  
0949 54 MDV C+E ; BC := (count)  
094A 5D MDV D+H  
094B 13 MDV E+L  
094C E8 INC DE ; DE := HL + 1  
094E C9 LDIR ; overlapping move  

RET
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Type</th>
<th>Data</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>094F</td>
<td>3E03</td>
<td>LDK</td>
<td>A+03h</td>
<td>ICRA1=0, Interrupt inputs CA1</td>
</tr>
<tr>
<td>0951</td>
<td>32012C</td>
<td>STD</td>
<td>A+H.V10+1</td>
<td>set data direction</td>
</tr>
<tr>
<td>0954</td>
<td>3EFF</td>
<td>LDK</td>
<td>A+OFFh</td>
<td></td>
</tr>
<tr>
<td>0956</td>
<td>32002C</td>
<td>STD</td>
<td>A+H.V10</td>
<td>set all A lines as output</td>
</tr>
<tr>
<td>0959</td>
<td>3E00</td>
<td>LDK</td>
<td>A+0</td>
<td></td>
</tr>
<tr>
<td>095B</td>
<td>32032C</td>
<td>STD</td>
<td>A+H.V10+3</td>
<td></td>
</tr>
<tr>
<td>095E</td>
<td>3EFF</td>
<td>LDK</td>
<td>A+OFFh</td>
<td></td>
</tr>
<tr>
<td>0960</td>
<td>32022C</td>
<td>STD</td>
<td>A+H.V10+2</td>
<td>set all B lines as output</td>
</tr>
<tr>
<td>0963</td>
<td>C9</td>
<td>RET</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OPAD:
*:Output data to pia A register
  :  PIA definition
  :  7 6 5 4 3 2 1 0
  :  horizontal offset I0D1
  :  ------------------------
*:NOTE The DD(double density) bit is inverted and the jumper must be installed on the pc board.
*:NOTE 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

PROC
0964 3E07
0966 32012C
0969 79
096A 3261EF
096D 32002C
0970 C9

LDK A+3
STO A+H+VIO+1
MOV A+C
STO A+PIAAD
STO A+H+VIO
SEND data

RET
S0RCIM 808x Assembler ver 3.5E <i>/557= 59:92 Page 75
Hardware PIO function processors.

0971  OPBD:
  8Output data to pia 8 register
  7 PIA definition.
  6 ++++++-----+++++++-----++++
  5 ++++++-----+++++++-----++++
  4  ID1ID01*G1 vert offset 
  3 ++++++-----+++++++-----++++

ENTRY
C  =  data

(EXIT
INONE

0971  PROC
0971  3E04  LDK  A+4
0973  32032C  STD  A+H+VID+3
0976  79  MOV  A+C
0977  3262EF  STD  A+PIABD
097A  32022C  STD  A+H+VID+2  :send data
097D  C9  RET
LAST EDITED AT 09:29 ON 11 NOV 80

THERE ARE FOUR COMMANDS TO THE 6821

- 00 PERIPHERAL/DIRECTION REGISTER A
- 01 CONTROL REGISTER A
- 10 PERIPHERAL/DIRECTION REGISTER B
- 11 CONTROL REGISTER B

BIT 2 OF THE CONTROL REGISTER (A AND B) ALLOWS SELECTION OF EITHER
- A PERIPHERAL INTERFACE REGISTER OR A DATA DIRECTION REGISTER.
- "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 PD0-PA7, PB0-PB7, CA2, AND CB2 AS INPUTS.
AND ALL INTERRUPTS DISABLED.
SIGNS 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 (PBO)
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:

ICRA = 0011(1FC1)(DIR)10
ICRB = 0011(REN)(DIR)100
ICPRA = SOURCE DIRECTION 1111_1111
ICPRA = ACCEPTOR DIRECTION 1111_1111
ICPRB = SOURCE DIRECTION 0011_1111
ICPRB = ACCEPTOR DIRECTION 1101_0111
: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 EDI/DAY (ENABLED WHEN "0")
- PB3  EOI
- PB4  ATN
- PB5  DAY
- 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]

- IRA01 0  INTERRUPT FLAG SET BY FALL OF SRQ
- IRA02 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
- IRA01 0  NOT USED
- IRA02 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
SDCIM 808x Assembler ver 3.5E c:1/55/7= 59:92 Page 78

BMIEEE.ASM - IEEE-488 INTERFACE

: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 097E F5 IE.CO: PROC
097E F5 PUSH AF
097F E5 PUSH HL
0980 CB41 $ BIT DC 0C ;CHECK IFC SUB-COMMAND
0982 2828 ^09AF$ JRZ :B1C20

;INITIALIZE ALL IEEE-488 SIGNALS

0984 210129 LK HL + CCR A
0987 363A STO 0011.1101B,[HL] ;ENABLE SRQ AND SET IFC-OUT LOW
0989 3EFA LK A.1111.1111B ;DIRECT DATA OUT
098B 320229 STO A + CPDA
098D 363E STO 0011.1110B,[HL]
0990 AF XDR A ;SET DATA TO ZERO
0991 320229 STO A + CPDA
0994 210329 LK HL + CCR B
0997 363D STO 0011.0000B,[HL] ;SET REN-OUT HIGH
0999 3E3F LK A.1111.1111B ;DIRECT FOR SOURCE HANDSHAKE
099B 320229 STO A + CPDA
099D 3634 STO 0011.0000B,[HL]
09A0 3E02 LK A.0000.0010B ;VALUES FOR SOURCE HANDSHAKE
09A2 320229 STO A + CPDA

;LEAVE IFC LOW FOR 100 MICRO-SEC

09A5 3E19 LK A.25 ;DELAY 100 MICRO-SEC
09A7 30 ;B1C10: DEC A
09A8 2F0D ^09A7$ JRNZ :B1C10
09AA 3E36 LK A.0011.0110B ;SET IFC HIGH
09AC 320129 STO A + CCR A
09AF CB51 $ ;B1C20: BIT 2C ;CHECK IFC SUB-COMMAND
09B1 280B ^093E$ JRZ :B1C40

;SET/CLEAR REN

09B3 3E30 LK A.0011.0000B
09B5 CB49 $ BIT 1C
09B7 2B0E ^09B8$ JRZ :B1C30
09BB 3E38 LK A.0011.1000B
09BD 320329 ;B1C30: STO A + CCR B
09BE E1 ;B1C40: POP HL
09BF F1 POP AF
09C0 C9 RET
; 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 ML
09C2 3A0029 LD A+CPDRA ; CLEAR IRQA1
09C5 210229 LK HL+CPDRB ; PULSE ENABLE ndac/nrfd
09CB CB8E $ CBIT 1+(HL)
09CA CBCE $ SBIT 1+(HL)
09CC 3A0129 LD A+CCRA ; SET SRQ VALUE IN A
09CF E68D AND 1000_0000B
09D1 07 RLC A
09D2 E1 PDP HL
09D3 C9 RET
IBIOS CALL 3: GO TO STANDBY

; CAN BE CALLED ONLY WHILE IN SOURCE HANDSHAKE MODE

; NO PARAMETERS PASSED

IE::GTS: PROC

09D4 F5
09D5 3E02
09D7 320229
09DA AF
09DB 320029
09DE F1
09DF C9

PUSH AF
LK $0000_0010B ; SET ATN HIGH
STD $0CPDRB
XOR A
STD $0CPDRA
POP AF
RET

IE::GTS: END
SERCIM 808x Assembler ver 3.5E <::/55/7= 59:92 Page 81
BMTEEE.ASM - IEEE-488 INTERFACE
C:SRROMA .ASM

IBIOS 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
09E0 E5
09E1 21029
09E4 CB41
09E6 2022 *0A0AS$ JRNZ :84C30

IE TC: PROC
PUSH HL
LK HL*CPDRB
BIT 0=C
JRNZ :84C30

:TAKE CONTROL SYNCHRONOUSLY

09E8 3607
STO 0000_0111B* [HLL] ;DISABLE DRIVERS
09F0 3A0329
LD A,*CCRB
09F0 CB97
CBIT 2+8
09F2 320329
STO A,*CCRB
09F2 3607
STO 1101_0111B* [HLL];DIRECTION REGISTER
09F4 CB07
SBIT 2+8
09F6 320329
STO A,*CCRB
09F9 3685
STO 1000_0010B* [HLL];SET NRFD LOW
09FB 3E19
LK A=25
09F9 CB6E
SBIT 8+4[HLL]
09FF 2807 *0A08$ JRZ :84C20 ;DATA VALID HAS DROPPED
0A01 3D
DEC A
0A02 20F9 *0FD8$ JRNZ :84C10 ;WAIT 100 MICRO-SEC
0A04 3E81
LK A=1000_0001B
0A06 1816 *0A1$ JR :84C40

0A08 36C5
SBIT 4+4[HLL]

:SET-UP FOR SOURCE HANDSHAKE

0A0C 3A0329
LD A,*CCRB
0A0F CB97
SBIT 2+8
0A11 320329
STO A,*CCRB
0A14 363H
STO 0011_1111B* [HLL];DIRECTION REGISTER
0A16 CB07
SBIT 2+8
0A18 320329
STO A,*CCRB
0A18 3612
STO 0001_0010B* [HLL];CONTROL SIGNAL INITIAL VALUE
0A1D AF
XOR A
0A1E E1
PB 84C40 ;CLEAR ERROR CODE
0A1F C9
RET
SBCIM 808x Assembler ver 3.5E C:55/76 59:92 Page 82

BMIEEE.ASM - IEEE-488 INTERFACE

; BICS 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

IE DIO: PROC
    PUSH HL
    LK HL+CPDRB
    SBIT 4+ [HL] ; SET ATN LOW
    BIT 0+ [HL]
    JRZ IE SHK

; SET-UP FOR SOURCE HANDSHAKE
    STO 0001_0111B+ [HL] ; DISABLE DRIVERS
    LD A+CPRB
    CBIT 2+ A
    STO A+CPRB
    STO 0011_1111B+ [HL] ; DIRECTION REGISTER
    SBIT 2+ A
    STO A+CPRB
    STO 0001_0010B+ [HL] ; CONTROL SIGNAL INITIAL VALUE
    JR IE SHK

Copyright © 1982 Okkone Computer Corporation
SORCIM 808x Assembler ver 3.5E <t/55/7= 59:192 Page 83
BMIEEE.ASM - IEEE-488 INTERFACE
C:SDROMA *ASM

: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
;EDI REQUEST IN REGISTER B

:ERROR CODE RETURNED IN REGISTER A

IE.ODM: PROC

OA43 E5 PUSH HL
OA44 210229 LK HL+CPDRB
OA47 CBA6 $ CBIT 4+[HL] ; ISET ATN HIGH
OA49 CB40 $ BIT 0+B ; ICHECK IF EDI REQUESTED
OA48 2802 *OA4FS$ JRZ IE.SMK
OA4D CBDE $ SBIT 3+[HL]

:PERFORM SOURCE HANDSHAKE

OA4F C36E $ IE.SMK: BIT 5+[HL]
OA51 2018 *OA6E$ JRNZ :B6C50 ; DAC TIMEOUT RE-ENTRY
OA53 79 MOV A+C
OA54 320029 STO A+CPDRA
OA57 3EDA LK A+10
OA59 CB7E $ :B6C20: BIT 7+[HL]
OA5B 2807 *OA64S$ JRZ :B6C30 ; READY FOR DATA
OA5D 3D DEC A
OA55 20F9 *OA59S$ JRNZ :B6C20 ; IWAIT FOR 100 MICRO-SEC
OA60 3E82 LK A+1000_0010B ; ISET RFD TIMEOUT ERROR
OA62 181F *OA83S$ JR :B6C80

OA64 CB76 $ :B6C30: BIT 6+[HL]
OA66 2004 *OA6C$ JRNZ :B6C40 ; DATA ACCEPTED LOW
OA68 3E81 LK A+1000_00018 ; ISET DEVICE NOT PRESENT ERROR
OA6A 1817 *OA83S$ JR :B6C80

OA6C CBEE $ :B6C40: SBIT 5+[HL]
OA6E 3EFF :B6C50: LK A+255
OA70 CB76 $ :B6C60: BIT 6+[HL]
OA72 2807 *OA7BS$ JRZ :B6C70 ; DATA ACCEPTED
OA74 3D DEC A
OA75 20F9 *OA70S$ JRNZ :B6C60 ; IWAIT 1000 MICRO-SEC
OA77 3E84 LK A+1000_0100B ; ISET DAC TIMEOUT ERROR
OA79 1808 *OA83S$ JR :B6C80

OA7B CBAE $ :B6C70: CBIT 5+[HL]
OA7D CB9E $ CBIT 3+[HL]
OA7F AF XOR A ; IRESET DEVICE FROM BUS
OA80 320029 STD A+CPDRA
OA83 E1 :B6C80: POP HL
OA84 C9 RET
; IBIOS CALL 7:  INPUT DEVICE MESSAGE
;     CAN BE CALLED WHILE IN ANY MODE OR STATE
;     EXITS IN THE ACCEPTOR HANDSHAKE MODE WITH ATN HIGH.
; IDEVICE MESSAGE RETURNED IN BOTH REGISTERS A AND H
; IERROR CODE RETURNED IN REGISTER L

IE-IDM:  PROC
OA85  PUSH  DE
OA86  EX  HL,SP
OA87  210229
OA88  CB46  $  BIT  0,HL
OA89  2014  ^OA88  JNZ  \B7C10

:SET-UP FOR ACCEPTOR HANDSHAKE
OA8E  3617  STD  0001_0111B,hl  :IDISABLE DRIVERS
OA90  3A0329  LD  a,CCR8
OA93  CB97  $  CBIT  2,a
OA95  320329  STD  a,CCR8
OA98  3607  STD  1101_0111B,hl  :IDIIRECTION REGISTER
OA9A  CB7D  $  SBIT  2,a
OA9C  320329  STD  a,CCR8
OA9F  3E0F  LK  a,1111_1111B  :IFLOAT INTERNAL DATA BUS
OAAC  320029  STD  a,CPDR8
OAAD  3655  STD  0101_0101B,hl  :ICONTROL SIGNALS INITIAL VALUE
OAAE  3645  STD  0100_0101B,hl  :ISET ATN HIGH

:PERFORM ACCEPTOR HANDSHAKE
OA80  CB76  $  \B7C10:  BIT  6,hl
OA8A  2820  ^OA8A  $  JRZ  \B7C50  :IDATA INVALID TIMEOUT ERROR RE-ENTRY
OA8C  CB8E  $  CBIT  7,hl
OA8E  3E0A  LK  a,10
OA90  CB8E  $  \B7C20:  BIT  5,hl
OA92  200B  ^OA8C  $  JRNZ  \B7C30  :IDATA VALID
OA94  30  DEC  a
OA95  200F  ^OA94  $  JRNZ  \B7C20  :IWAIT 100 MICRO-SEC
OA97  118200  LK  d,1000_0010B  :ISET DATA VALID TIMEOUT ERROR
OA9A  1821  ^OA97  $  JR  \B7C80

OABC  C8FE  $  \B7C30:  SBIT  7,hl  :ISET NRF0 LOW
OA8E  3A0029  LD  a,CPDR8
OA91  57  MDV  a,0
OA92  1E00  LK  a,0  :IREAD E01
OA94  CB5E  $  BIT  3,hl
OA96  2002  ^OA94  $  JRZ  \B7C40
OA98  1E01  LK  a,1
OA9A  C8B6  $  \B7C40:  CBIT  6,hl
OACC  3EFF  STD  1110_1110B,hl
OACE  C86E  $  \B7C60:  BIT  5,hl
OADD  2009  ^OADC  $  JRZ  \B7C70  :DATA VALID DROPPED
OAD2  3D  DEC  a
OAD3  200F  ^OADD  $  JRNZ  \B7C70  :IWAIT 100 MICRO-SEC
OAD5  CB03  $  SBIT  2,E
OAD7  CB6F  $  SBIT  7,E
OAD9  1801  ^OAD7  $  JR  \B7C80
OADB  CB6F  $  \B7C70:  SBIT  6,hl  :ISET NDA0 LOW
OADD  E9  ::\B7C80:  EX  DE,HL  :MOVE RESULTS TO REGISTERS A AND HL
DADE TC
OADF D1
OAEO C9

MOV A+H
POP DE
RET
;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.

IE*PP: PROC

OAE1  E5            PUSH    HL
OAE2  210229        HK       HL×CPDRB
OAE3  3E1B           HK       A×0001_1011B   :FORM PARALLEL POLL
OAE7  320229        STO      A×CPDRB
OAE8  3E6F           STO      1111_1111B×[HL] :FLOAT INTERNAL DATA BUS
OAE9  7E             LD        A×[HL]       :READ PARALLEL POLL DATA
OAD0  3600           STO      C×[HL]       :RE-STORE SOURCE HANDSHAKE MODE
DAF2  210229        HK       HL×CPDRB
DAF4  3E12           STO      0001_0010B×[HL]
DAF4  E1             PDP      HL
DAF5  C9             RET

"
SIO - Serial I/O Processors.

*5*

DAF6

SIRST:

*Master reset SIO

ENTRY

IC = SI.S16 or SI.S64 for 1200/300 baud

EXIT

NONE

DAF6

PROC

OAF6 3E57

LDK A+SI.MRST

OAF8 32002A

STD A+H.SCTRL ; master reset

OAF8 79

MOV A+C

OAF8 32C1EF

STD A+ACIAD ; last-command cell

OAF8 32002A

STD A+H.SCTRL ; select SIO

O802 C9

RET
0803 PROC
0803 READ:
0803  "Input one byte from reader port"
0803  "None"
0803  EXIT
0803  IC = character read

0803  CALL ACISTAT
0806  ANI SI.RDY
0808  JNZ READER ;if not ready

080A  XRA A
0808  STD A,SEFLG ;SET FLAG

080E  3A012A
0811  4F
0812  C9
SLST:
!Get list device status
!ENTRY
!NONE

:EXIT
:IA = 0, IF NOT READY
:TA = OFFH IF READY
:ZBIT = SET IF NOT READY FOR OUTPUT

0B13 PROC
0B13 CD280B CALL ACISTAT
0B16 6602 ANI S1.TRDY
0B18 CB RZ
0B19 F6FF ORI TRUE
0B19 C9 RET
LIST:
;Output one byte to list port
ENTRY
IC    = character to output
EXIT

PROC

OB1C C5
PBBC

OB1D CD130B  ;1:
CALL SLST ;GET STATUS
JRZ :1

OB22 C1
POPB AC ;RESTORE CHARACTER
OB23 79
MOVA A+C
OB24 32012A
STOA A+H.5XMT ;send chr
OB27 C9
RET
ACISTAT: 2EXIT STATUS OF THE SERIAL PORT ENTRY INONE

IA = STATUS_REG

PROC
0828 3A012C
0828 0F
082C EE20
082E 4F
082F 3A002A
0832 EE0F
0834 B1
0835 4F
0836 3A00EF
0839 EE01
083B B1
083C 32DAEF
083F C9
LD A+H+VID+1
RRC A
ANI 20h
MDV C+A
LD A+H+SSTS
ANI 0DFh
ORA C
MDV C+A
LD A+SERFLG
ANI 01
ORA C
STO A+SERFLG
RET
Disk I/O routines

*[6]*

- **D00A**   NRETRY: = 10
- **EFDD**   SAVTYPE: = OEFDDH
- **EFD1**   RDWRTS: = OEFD1H
- **EFD2**   CCPADR: = OEFD2H
- **EFD4**   KEYLST: = OEFD4H

*NUMBER OF RETRYS MUST BE AT LEAST 2
IDISK TYPE
INumber of sectors to read or write
This location is assigned in BIOS
and filled in by loader in ROM
KEY LIST GOES HERE*
DISK EQUATES

= 0010  MSEC8: = 010H  IMULTI SECTOR BIT
= 0000  D^RES: = 000H  IRESTORE
= 0010  D^SEK: = 010H  ISEEK
= 0020  D^STP: = 020H  ISTEP
= 0040  D^STPI: = 040H  ISTEP IN
= 0060  D^STPD: = 060H  ISTEP OUT
= 0080  D^RDI: = 080H  IREAD SECTOR
= 00A0  D^WRT: = 0A0H  IWRITE SECTOR
= 00C0  D^RDA: = 0C0H  IREAD ADDRESS
= 00E0  D^RTI: = 0E0H  IREAD TRACK
= 00F0  D^WRTI: = 0F0H  IWRITE TRACK
= 0000  D^FINI: = 000H  IFORCE INTERRUPT

IDISK REGISTERS

= 2100  D^CMDR: = M^FDC  IDISK COMMAND REG  (WRITE)
= 2100  D^STSR: = M^FDC  ISTATUS REG  (READ)
= 2100  D^TRKR: = D^CMDR+1  ITRACK REG
= 2102  D^SECR: = D^CMDR+2  ISECTOR REG
= 2103  D^DATR: = D^CMDR+3  IData REG  (R/W)

ISTATUS DEFINITIONS

= 0000  BS^BSY: = 0  IBUSY
= 0001  DS^BSY: = 1  5SHL BS^BSY
= 0001  BS^DRQ: = 1  INDEX MARK DETECTED
= 0002  DS^INX: = 1  5SHL BS^DRQ
= 0002  DS^DRQ: = DS^INX  IDR IS FULL ON READ, EMPTY ON WRITE
= 0002  BS^TKO: = 2  ITRACK ZERO
= 0004  DS^TKO: = 1  5SHL BS^TKO
= 0004  DS^LDS: = DS^TKO  ILOST DATA
= 0008  DS^CRC: = 08H  ICRC ERROR IN ID FIELD
= 0004  BS^SEK: = 4  SSEEK
= 0010  DS^SEK: = 1  5SHL BS^SEK
= 0010  DS^RNF: = DS^SEK  ISEEK ERROR
= 0020  DS^HDL: = 20H  IHEAD LOADED
= 0020  DS^WTF: = DS^HDL  IWRITE FAULT
= 0040  DS^WTP: = 40H  IWRITE PROTECTED
= 0080  DS^NRY: = 80H  ICRAVE NOT READY

IDISK TIMING COUNTS

= 0014  D^CEL: = 20  IDELAY AFTER FUNCTION
MACRO DEFINITIONS

ENOAROM 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

POPALE MACRO
  POP HL
  POP DE
  POP BC
  ENDM
NEW DISK DRIVERS

0840

RDRV:
RESET DRIVE
ENTRY
NONE

EXIT
= SET IF ERROR

0840 PROC

0840 3EOA LDK A*RETRY
0842 3205EF :LOOP: STO A*TRY
0845 CD440D CALL SELDRV ISELECT DRIVE
0848 3804 #084E: JRC 1 IHOME DRIVE
084A CD70B CALL HOME IIF GOOD
084D 00 RNC

084E 3A05EF :1: LD A*TRY
0851 3D DEC A
0852 FE01 CMP 1
0854 20EC #0842: JRNZ :LOOP

0856 3E01 LDK A+1 IA = OFFH FOR CBios
0858 87 ORA A
0859 37 STC
085A C9 RET

INDICATE ERROR
NEW DISK DRIVERS

0858

RSEC:
READ SECTOR

*NOTE*
No retries are performed at this level

ENTRY
IB = NUMBER OF SECTORS

EXIT
IML = LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
IBIT = RESET IF ERRDR
IA = NONZERO IF ERROR
.ret = 1 IF ERROR (SO OLD BIOS DOESN'T DO RETRYS)

0858 PROC

0858 FD21750C $ 085F CDB40B 0862 3802 -0866$ 0864 AF 0865 C9 0866 3E01 0868 3209EF 086A B7 086C C9

LDK IY=READ
CALL R_WSEC
JRC :1
XRA A
:IF ERROR
RET

:INDICATE GOOD TO BIOS

LDK A=1
STO A\+\+\+\+\+
:SET RETRY TO 1 FOR OLD BIOS

ORA A
:RESET ZERO FLAG TO INDICATE ERROR

RET
NEW DISK DRIVERS

0860       WSEC:  
0861       WRITE A SECTOR  
0862       *NOTE*  
0863       :  
0864       _ENTRY  
0865       :B  
0866       NUMBER OF SECTORS  
0867       :EXIT  
0868       :HL  
0869       LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER  
086A       :IBIT  
086B       =  
086C       SET IF ERROR  
086D       :A  
086E       NONZERO IF ERROR  
086F       :ITRY  
0870       =  
0871       1 IF ERROR(SO OLD CB IOS DOESN'T DO RETRYS)

086D PROC

0870       FD2170DC $  
0871       CD840B  
0872       CALL @_WSEC  
0873       3802 *0878$  
0874       JRC :1L  
0875       ; IF ERROR
0876       AF       XRA A  
0877       C9       RET  
0878       3E01 :1L  
0879       LDK A+1  
087A       3205EF  
087B       STA A+ITRY  
087C       87  
087D       ORA A  
087E       C9       RET  
087F       C9       RET
NEW DISK DRIVERS

SENDEN:
1READ THE ADDRESS AND SET CHECK FOR SINGLE DENSITY
1ENTRY
1NONE

:EXIT
1A = NONZERO IF ERROR
1B = NUMBER OF SECTORS ON ONE TRACK
1ZBIT = RESET IF ERROR
1SAVTP IS SET WITH DENSITY AND SECTOR SIZE

PROC

:CHECK DENSITY
1DENSITY LOC (CHECK PRESENT DENSITY FIRST)

:0880 3EO2
0882 3205EF :RL1:
LDK A+2
1HOME DRIVE JUST ONCE
STD A+RTRY
IRETRY LOOP BEFORE AND AFTER HOME

:0885 0605
LDK B+RTRY/2
1HALF BEFORE HOME AND HALF AFTER

:0887 C5 :RL2:
PUSH BC
1SAVE RETRY COUNT

:0888 CDA40D
CALL SELDRV
1SELECT DRIVE
JRC ?
1ERROR CHECKING FOR SELDRV BECAUSE NO ERRORS ARE RETURNED AT THIS TIME

:0888 10EC
CALL RADR
1READ ADDRESS

:088E C1
PDC BC
1RESTORE RETRY

:0890 3004 *08956
JRCN 11
1IF GOOD

:0891 10F4 *08874
DJNZ 1RL2
1IRETRY IN THIS DENSITY

:0893 1812 *08A74
JR 1ERR1
1IF ERROR HOME DRIVE AND TRY AGAIN

:SET "SAVTP"

:0895 3A0CEF
1L:
LD A+DST5B+3
1SECTOR LENGTH STATUS BYE

:0899 E603
ANI 0000_0011B
10-3

:089A FE01
CP1 0000_0001B
1CHECK FOR 256

:089C 2009 *08A73
JNZ 1ERR1

:089E 3E05
LDK A+0000_01018
1SINGLE DENSITY 256

:08A0 3200EF
STD A+SAVTP

:08A3 060A
LDK B+10
1SET FOR SINGLE DENSITY

:08A5 A5
XRA A
1RESET FLAGS

:08A6 C9
RET
1GOOD RETURN

:IF DENSITY ERROR CHANGE DENSITY AND RETRY TO :RL1:

:08A7 2D709
1ERR1:
CALL HOME 1HOME DRIVE

:08AA 3A05EF
LD A+RTRY

:08AD 3D
DEC A

:08AE 20D2 *08823
JNZ 1RL1
1IRETRY LOOP

:08AF 3E01
LDK A+1

:08B2 B7
ORA A
1FLAGS TO NONZERO

:08B3 C9
RET
1ERROR RETURN
NEW DISK DRIVERS

R_WSEC:
IREAD OR WRITE SEGMENT
IENTRY
IB = NUMBER OF SECTORS TO READ OR WRITE
IIY = "READ" ADDR OR "WRITE" ADDR
IEXIT
IHL = LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
ICBIT = SET IF ERROR

08B4 PROC
08B4 3E0A LDK A+NRTRY
08B6 3205EF STD A+RTRY
08B9 C5 :RLOOP: PUSH BC
08BA C5 PUSH BC
08BB FDE5 $ PUSH IY
08BD CDA40D CALL SELDRV ITURN DRIVE ON
08C0 2805 *0BC7# JRZ 11 IF DRIVE WAS SELECTED DON'T READ ADDRESS
:SET "D.TRKR" TO HEAD POSITION
08C2 CD800B CALL SENDEN IREAD ADDRESS AND SET CONTROLLER
08C5 2022 *O8E9# JRNZ :ERR1 ISTOP
08C7 CD1ACC :I: CALL SEEK ISEEK TO TRACK
08C4 3B10 *O8E9# JRC :ERR1 ISTOP
08CC DDE1 $ POP IX
08CE C1 POP BC
08CF 21D50B LDK HL+IRTRY1
08D2 E5 PUSH HL IFOR RETURN
08D3 DDE9 $ JMP [IX] ICALL AND RETURN TO 12
08D5 C1 :RTRY1: POP BC IRESTORE NUMBER OF SECTORS TO READ
08D6 DD RNC
08D7 2105EF LDK HL+IRTRY
08DA 35 DEC [HL] IFIN INO MORE RETRYS
08DB 2811 *O3EE # JRZ :FIN
08DE 7E LD A+[HL] IGET NUMBER OF RETRY
08DF FE09 CMP NRETRY-1
08E0 2007 *O859# JRNZ :RLDOP ISTOP IF ERROR
08E0 CD800B CALL SENDEN ICHECK TRACK ON THE SECOND RETRY
08E4 2007 *O8EE # JRNZ :FIN
08E7 1800 *0899# JR :RLDOP ISTOP
08E9 FDE1 $ :ERR1: POP IY
08EA C1 POP BC IIF ERROR BEFORE READ OR WRITE
08EC C1 POP BC
08ED C9 RET
08EE 37 :FIN: STC
NEW DISK DRIVERS

SCTRKR:
SET CONTROLLER TRACK REGISTER
D.TRKR <= SAVTRK
USED IN FORMATING WHEN YOU DON'T KNOW WHERE THE HEAD IS
ENTRY
SAVTRK = TRACK
EXIT
D.TRKR = SAVTRK

08F0 PROC
08F0 3A15EF LD A+SAVTRK
08F3 320121 STO A+D.TRKR
08F6 C9 RET
NEW DISK DRIVERS

SOURCIM 808x Assembler ver 3.5E <:/55/7= 59:92 Page 102
C:SORDMA .ASM

08F7
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
:ISDISK = DRIVE
:EXIT
:CBIT = SET IF ERROR

08F7
:PROC
08F7 3A13EF
08FA E607
08FC 3213EF
08FF 3E00
OC01 CD760D
OC04 08
OC05 3A0021
OC08 CB57 1:
OC0A 280C 40C18E
OC0C 3A13EF
OC0F E604
OC11 C8
OC12 3A0021
OC15 E618
OC17 C8
OC18 37 1:
OC19 C9

LD A*SEKDEL
ANI 0000_0111B
STO A*SEKDEL
LDK A*D*RES
CALL PSERSE
RC
LD A*D*STR
BIT 2:A
JRZ 11
LD A*SEKDEL
ANI 0000_0100B
:VERIFY?
:VERIFY GOOD RETURN
LD A*D*STR
ANI 0001_0000B
:TEST SEEK AND CRC
RZ
IGOOD RETURN
STC
:RET

!IF ERROR
NEW DISK DRIVERS

OC1A

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 CONTROLLER TRACK = SAVTRK

OC1A

PROC
ML+0.TRKR
LDK
LD
CMP
RZ

OC10
3A15EF
8E
C8

OC25
3E10

OC27
CD760D

OC2A
DB

OC2B
3A13EF

OC2E
E604

OC30
C8

OC31
3A0021

OC34
E618

OC36
C8

OC37
37

OC38
C9

STC
RET

RETURN

STO
A+D.DATR

LDK
A+D.SEK

CALL
PSEKC
RC

IF ERROR

IF ERROR

VERIFY?

NO VERIFY GOOD RETURN

TEST SEEK AND CRC

GOOD RETURN

IF ERRDR
NEW DISK DRIVERS

STOP 808x Assembler ver 3.5E <5S7 59:92 Page 104

=SDROMA =ASM

OC39

STEPE
STEP ONE TRACK
;SAVRK IS NOT USED IN THIS PROC
;CONTROLLER TRK REG IS UPDATED
;VERIFY IS PERFORMED
;ENTRY
;NONE

;EXIT
;CBIT = SET IF ERROR
; IF NO ERROR CONTROLLER TRACK = TRACK +/- 1

OC39 3E2D
OC38 CD760D
OC3E 00
OC3F 3A13EF
OC42 E604
OC44 C8
OC45 3A0021
OC48 E618
OC4A C8
OC4B 37
OC4C C9

PROC

LDK A+D.STP
CALL PSERC
;PERFORM STEP COMMAND

RC 
;IF ERROR

LD A+SEKDEL
ANI 0000_2100B
;VERIFY?

RZ 
;NO VERIFY GOOD RETURN

LD A+D.STSR
ANI 0001_1000B
;TEST SEEK AND CRC

RZ 
;GOOD RETURN

STC 
;IF ERROR

RET
NEW DISK DRIVERS

;STEPIN:
;STEP IN ONE TRACK
;SAYTRK IS NOT USED IN THIS PROC
;CONTROLER URK REG IS UPDATED
;ENTRY
;NONE

;EXIT
;ICBIT = SET IF ERROR
;
; IF NO ERROR CONTROLLER TRACK = TRACK + 1

0C40 PROC
0C40 3E40 LDK A+D.STPI
0C4F CD760D CALL PSEKC ;PERFORM STEP-IN COMMAND
0CS2 D8 RC
0CS3 3A13EF LD A+SEKDEL
0CS6 E604 ANI 0000_01008 ;VERIFY?
0CS8 C8 RZ
0CS9 3A0021 LD A+D.STSR
0CSF E618 ANI 0001_10008 ;TEST SEEK AND CRC
0C9E C8 RZ ;GOOD RETURN
0C5F 37 STC
0C60 C9 RET ;IF ERROR
NEW DISK DRIVER'S

OC61
STEPOUT:
;STEP OUT ONE TRACK
;ISAVTRK IS NOT USED IN THIS PROC
;CONTROLLER TRK REG IS UPDATED
;VERIFY IS PERFORMED
;ENTRY
;NONE

;EXIT
;SCBIT = SET IF ERROR
;IF NO ERROR CONTROLLER TRACK = TRACK - 1

OC61 PROC
OC61 3E60
OC63 CD760D
OC66 D8
OC67 3A13EF
OC6A E604
OC6C C8
OC6D 3A0021
OC70 E618
OC72 C8
OC73 37
OC74 C9

LDK A+STPO
CALL PSEKC
PERFORM STEP-OUT COMMAND
RC
IFDEF ERROR
LD A+SEKDEL
ANI 0000_0100B
VERIFY?
RZ
IFDEF VERIFY GOOD RETURN
LD A+STSR
ANI 0001_1000B
TEST SEEK AND CRC
RZ
IFDEF GOOD RETURN
STC
IFDEF ERROR
RET

NEW DISK DRIVERS

OC75
READ:
ENTRY
IB = NUMB OF SECTORS TO READ

EXIT
$HL = LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
$CBIT = SET IF ERROR

OC75 PROC
OC75 3E80 LDK A+DDS
OC77 32D1EF STO A+ROT_WRTS
OC7A C3820C JMP RD_WRT :JMP AND RETURN TO CALLING PROC
NEW DISK DRIVERS

OC7D

WRITE:
 ENTRY
 IB = NUMB OF SECTORS TO WRITE

EXIT
 IML = LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
 ICBIT = SET IF ERROR

OC7D

PRDC

OC7D 3EA0
OC7F 3201EF

LDK A+D-WRTS
STO A+RDT_WRTS

JMP RD_WRT 1FALLS THROUGH TO RD_WRT
NEW DISK DRIVERS

OC82
RD_WRT:
READ OR WRITE
ENTRY
IB = NUM OF SECTORS TO READ OR WRITE
SROT_WRTS = DROS OR DWRTS

EXIT
INL = LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
IFBIT = SET IF ERROR

OC82
PROC

ISET SECTOR REG

OC82 3A146F
OC85 320221
LD A,SAVSEC
STO A,DSEC

ISET DE TO NUMBER OF BYTES IN ONE SECTOR

OC88 C5
PUSH BC
ISAVE NUMBER OF SECTORS TO READ OR WRITE

OC89 218000
OC8C 3A00FF
OC8F CBDF
OC91 CB3F
OC93 E603
OC95 B7
OC96 2804 *OC9C5
OC98 47
OC99 29
*IBLOOP: ADD HL,HL
OC9A 10FD *OC995
OC9C 8B
OC9D C1
OC9E C5
PUSH BC
ISAVE NUMBER OF SECTORS TO READ OR WRITE

ISAVE COMMAND AND CHECK FOR MULTI-SECTOR

OC9F 78
OC9A 0000
OC92 FE02
OC94 3802 *OCAB5
OC9A 6E10
OCAB 3A01EF
JL: LD A,SRODRITS
OCB 51
OR C
ISAVE MULTI-SECTOR OR NONMULTI-SECTOR

ISAVE HL TO NUMBER OF BYTES TO TRANSFER

OCAC 210000
OCAF 10
*ILoop: ADD HL,DE
OCB0 10FD *OCAB$ DJNZ 1Loop
OCB2 E9
PUSH HL
ISAVE LENGTH

IGIVE COMMAND

OC3 C8400
DI CALL FDSK
NEW DISK DRIVERS

OCB7 C1  POP BC  :IRESTORE LENGTH
OCB8 3003 *OCBD$  JRNZ 17  :IF GOOD
OCBAX D1  POP DE  :IRESTORE STACK
OCB8 EI
OCB8 C9  RET  :ERROR RETURN

IDDD DMA

OCBD 2A0EF  :71  LD HL,DMADR  :IHL = DMA ADDRESS
OCCE 11D10C  LDK DE+16  :IFOR RETURN
OCCE 05  PUSH DE
OCCE 3AD1EF  LD A,*RDWRTS  :IGET COMMAND
OCCE FE80  CMP 0*RDS
OCCE 203 *OCCE$  JRNZ 15  :IF WRITE
OCCE C30A0E  JMP DMARD  :IREAD DMA RETURNS TO 16
OCCE C3200E  :15  JMP DMANRT  :WRITE DMA RETURNS TO 16

:CHECK FOR BUSY AND RESET

OCDD 1A  :16  LD A,(DE)  :IGET STATUS
OCDE CB47  D,A  :IRESTORE NUMBER OF SECTORS
OCDF C1  POP BC  :IRESTORE NUMBER OF SECTORS
OCDE 280E *OCES$  JRZ 14  :IIF NOT BUSY
OCDE 05  DEC B  :ISUBTRACT ONE FROM THE NUMBER OF SECTORS AND SET THE ZERO FLAG
OCDE 2005 *OCDF$  JRNZ 1A1  :IIF MORE THAN ONE, FORCE INTERRUPT
OCCE CDE60D  CALL WBUSY  :IIF NON MULTI-SECTOR R/W WAIT FOR BUSY TO DROP
OCDE 1A03 *OCES$  JR 1A2
OCDE CDE40D  :1A1  CALL FORINT  :ICLEAR BUSY
OCCE 3A0021  :1A2  LD A+STS$  :IGET STATUS

:CHECK FOR ERRORS

OCDE 1E6C  :14  ANI 0101_1100B  :IEST write protect, rnf, crc, and lost data
OCDE 2801 *OCES$  JRZ 13  :IF GOOD

*NOTE*  
If this was a multi-sector function there 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.

OCDE 37  STC  :IF ERROR RECORD CONTROLLER REGESTERS
OCDE 13  EI
OCDE C9  RET  :RETURN
NEW DISK DRIVERS

OCEC
ADR:

READ ADDRESS INFO.

READS SIX BYTES INTO "DSTSB"

ENTRY

INONE

EXIT

IA = OFFH IF TIME OUT ERROR

ICBIT = SET IF ERROR

IDTRKR = HEAD POSITION

*NOTE*

SETS TRACK REG IN CONTROLLER IF GOOD

OCEC

PROC

LDX A+D.RDA

OCEG

DI

OCEF CD840D

CALL FDSK ;function disk

OCF2 383D *OD31s

JRC 11

IWAIT FOR FIRST DRQ OR TIME OUT

ISET REGESTERS FOR DMA TRANSFER

OCF4 010600

LDK BC+6

OCF7 2199EF

LDK HL+DSTSB

OCFA 110811

ILOOP;

LDK DE+4363

OCFD 3A0021

LD A+D.STSR

ODD0 1F

RAR

ODD1 1F

RAR

ODD2 DA120D

JC 13

ODD5 1B

DEC

O0D6 7A

MOV A+D

O0D7 83

CRA E

O0D8 C2FD0C

JNZ ILOOP

O0D8 CD840D

CALL PJRINT ;CLEAR BUSY

O0DE 3EFE

LDK A+OFFH

O0D10 181E *0030s

JR 12 ;INDICATE A TIME OUT ERROR

O0D12 3A0321

I3:

LD A+D.DATR

O0D15 77

STO A+HL

O0D16 23

INC HL

O0D17 08

DEC BC

O0D18 CD840E

CALL DMARD

O0D18 CD660D

CALL WBUSY

O0D1E 3B11 *0031s

JRC 11 ;IF TIME OUT ERROR

O0D20 3A0021

LD A+D.STSR ;GET STATUS
NEW DISK DRIVERS

; CHECK FOR ERRORS
    ANI 0001_10008  ; TEST RNF AND CRC
    ANI 0001_11008  ; TEST RNF, CRC AND DATA LOST
    JRNZ -2        ; IF ERROR

; SET TRACK REGISTER
    LD A+0$SEC  ; GET TRACK
    STO A+0$TRK  ; SET TRACK
    XRA A       ; RESET CARRY
    JR 11

    STC          ; SET CBIT
    EI
    RET
NEW DISK DRIVERS

0D33
READTRK:
IREAD ONE TRACK FROM THE DRIVE
\*ENTRY
\*DMADR = FWA OF BUFFER
\*EXIT
\*CBIT = SET IF ERROR

0D33 3EE0 PROC
0D35 DI
0D36 CD840D CALL FDSK
0D39 380C ^DD47$ JAC $11 ;IF ERROR

10D DMA

0D38 01FFFF LDK BC,FFFFH ;FOR ROM 1.2
0D3E 2A0FEF LD HL,DMADR ;IN ROM
0D41 CDDA0E CALL DMARD
0D44 EI
0D45 AF XRA A
0D46 C9 RET
0D47 $11 ;IF ERROR
0D48 C9 RET
NEW DISK DRIVERS

SROM 808x Assembler ver 3.5E <1/55/7> 59:92 Page 114
C:SORDMA .ASM

OD49  FMTTRK:
0D49  IFORMAT ONE TRACK
0D49  :ENTRY
0D49  IDMADR = FWA OF BUFFER
0D49  :EXIT
0D49  :CBIT = SET IF ERROR
0D49  OD49  3EFO
0D49  PROD
0D49  OD49  D1
0D49  LDK A+0.WRTT
0D49  ODAC CD840D
0D49  CALL FDSSK
0D49  OD4F  3810 *OD618
0D49  JRC 11 IIF ERROR

:DO DMA

OD51  01FFFF
OD51  LDK 8C+0FFFFH ;DEFEA T COUNTER
OD54  2AD0EF
OD54  LD  HL+DMADR
OD57  CD200E
OD57  CALL DMWRT

:CHECK FOR ERROR

OD5A  3A0021
OD5A  LD A+0.STSR ;GET STATUS
OD5D  E644
OD5D  ANI 0100_0100B ;TEST write protect* and data lost
OD5F  2801 *OD621
OD5F  JRZ 12 IIF GOOD

OD61  37 11 STC
OD62  12 EI
OD63  C9 RET
NEW DISK DRIVERS

FORINT:
; INTERRUPT DISK CONTROLLER
; ENTRY
; IDONE

; EXIT
; IDISKY CLEARED.

0D64 PROC
0D64 F5 PUSH AF
0D65 C5 PUSH BC
0D66 3ED0 LDK A,0,FIN
0D68 320021 STD A,0,CMDR

; WAIT FOR AT LEAST 28 MICROSECONDS

0D68 B7 ORA A I(4)
0D6C 0607 LDK B,7 I(7)
0D6E 10FE "0D6E" :WLOC: DJNZ :WLOC:
I(91) = (13*7) WAIT

; CHECK FOR BUSY DRCP

0D70 CDE60D CALL WBUSY

0D73 C1 POP BC
0D74 F1 POP AF
0D75 C9 RET
NEW DISK DRIVERS

0076

PSEKC:
IOR IN SEKDEL AND PERFORM SEEK TYPE COMMAND
ENTRY
IA = SEEK TYPE COMMAND
EXIT
CBIT = SET IF ERROR
COMMAND SENT TO CONTROLLER IF NO ERROR

0076 47
0077 3A13EF
007A E617
007C 80
007D CD340D
0080 08
0081 C3E60D

PROC
MOV B,A
LD A,SEKDEL
ANI 0001_0111B
ORA B
CALL FDSK
RC
JMP WBUSY

IB = A
IONLY UPDATE, VERIFY, & SPREAD
IOR IN COMMAND
IFUNCTION DISK
IIF ERROR
NEW DISK DRIVERS

SORCIM 808x Assembler ver 3.5E <55/7= 59:92 Page 117
C:SORDRA +ASM

OD84  FDSK:
     :FUNCTION DISK ROUTINE
     this is the only-route that writes to the command register of the controller chip.
     this routine has a built-in delay of at least 28 micro sec. before reading the status on the chip.
     i= entry
     ia = function code
     i= exit
     ia = offh is time cut error
     icbit = set if error

OD84  PROC
OD84  210021   LDH Martin: (HL)  ;status and command register
OD87  CB46     BIT 0x(HL)        ;if not busy
OD89  2B03 *OD8E$  JRZ 11            ;if not busy
OD8B  CD640D   CALL FONTINT        ;reset busy
OD8E  77     :11: STC A*(HL)        ;function drive(write command to controller)

:wait for at least 28 microseconds

OD8F  67     ORA A               ;(4)
OD90  0607   LDH B.T               ;(7)
OD92  10FE *OD92$  DJNZ 1WLOOPP  ;(91) = (13x7) wait

:wait for busy to be set

OD94  3EFF   LDH A,OFFH           ;(7)
OD96  47     MOV B,A             ;(4) 256 loops
OD97  CB46   $1WLOOPP: BIT BS$BY*(HL) ;b$by
OD99  2B04 *OD9F$  JRNZ 13          ;if chip went busy
OD9B  10FA *OD97$  DJNZ 1WLOOPP  ;if not time-out
OD9D  37     STC                  ;if error a=ff and cbit = set
OD9E  C9     RET
OD9F  326AEF  :3: STO A,DACTVE    ;set drive active counter
ODA2  AF     XRA A               ;reset carry flag
ODA3  C9     RET
NEW DISK DRIVERS

ODA4
SELDV:
:SELECT DRIVE
:ENTRY
:SDISK = DRIVE TO SELECT
:EXIT
:IBIT = SET IF PIABD WAS THE SAME AS SDISK
:IBIT = RESET IF PIABD WAS DIFFERENT THAN SDISK
:CBIT = SET IF THERE ARE NO INDEX PULSES

ODA4 PROC
CALL SELDEN :SELECT DENSITY

ODA7 3A17EF LD A,SDISK
ODAA 21C7EF LDK HL,DSKSWP :DISK DRIVE SWAP CELL
ODAD AE XOR (HL) :SWAP A FOR B IF DSKSWP=1
ODAE E001 AND 1 :CAN ONLY BE 0 OR 1
ODB0 FE01 CMP 1
ODB2 2002 ^ODB6$ JRNZ ;1 :IF NOT DRIVE 1
ODB4 3E40 LDK A+40H

ODB6 C640 ;2 ADI 40H
ODBB 4F MOV C,A
ODB9 3A62EF LD A,PIABD
ODBC 47 MOV B,A
ODBD E600 ANI 1100_0000B :GET DRIVE BITS ONLY
ODBF 89 CMP C
ODC0 2808 ^ODCD$x JRZ ;2 :IF DRIVE ALREADY SELECTED

:SELECT DRIVE

ODC2 CDFF0D CALL RDSKD :TURN DRIVE ON
ODC5 3EFA LDK A+250
ODC7 CD2D09 CALL DELAY :WAIT FOR MOTOR SPIN UP
; LDK A+250
; CALL DELAY :2ND DELAY
ODCA 3E01 LDK A+1 :INDICATED DRIVE WAS NOT SELECTED
ODCB B7 ORA A :SET FLAGS
ODCD 216AEE ;2 LDK HL+DACTVE
ODDD 36FF STD OFFH,[HL]
ODDE EI
ODDF C9 RET
NEW DISK DRIVERS

SELDEN:
ISELECT 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

ODD4 PROC
ODD4 3A61EF
ODD7 E6FE
ODD9 4F
LDA PIAAD
ANI 1111_1110B
MOVC A,A

;SET DENSITY BIT

ODDA 3ADD0EF
ODDD 0F
OEDE 3002 "ODE2"
ODE0 CBC1 $ SBIT 0+C
ODE2 C06409 ; CALL OPAD ;FUNCTION PIA
ODE5 C9
RET
NEW DISK DRIVERS

SORMIC 808x Assembler ver 3.5E  c://  55/7 =  59:92  Page 120
C:\SORMIC \ASM

ODE6

:WAIT FOR BUSY TO CLEAR
:This routine must wait for 2 seconds
: 12 seconds is the time it takes for the chip to seek 39 tracks and have five index holes go by.
:ENTRY
:INONE

:EXIT
:A    = OFFH IF TIME OUT OCCURRED
:CBIT = SET " " " "

ODE6  010000  PROC
ODE6  000000  LDK  BC,0
ODE9  3A0021  :LOOP:
ODEC  C847  $  BIT  D,A
ODEE  2B0E  *ODEF$  JRZ  -1
ODF0  E3  EX  [SP],HL
ODF1  E3  EX  [SP],HL
ODF2  23  DEC  HL
ODF3  23  INC  HL
ODF4  28  DEC  HL
ODF5  23  INC  HL
ODF6  0B  DEC  BC
ODF7  78  MOV  A,B
ODF8  B1  OR  C
ODF9  20EE  *ODE9$  JRNZ  +1
ODFB  3EFF  :LDK  A,OFFH
ODFD  37  CALL  FORINT
ODFE  C9  :1:  RET
NEW DISK DRIVERS

ODFF

RDSK:
:SELECT DRIVE BY SETING THE "PIA" WITH THE VALUE SPECIFIED BY C
:ENTRY
:IC = DRIVE

:EXIT
:NONE

ODFF

3A62EF
LD A,PIA8D

E63F
ANI 0011_11118

81
OR C

4F
MOV C,A

CD7109
CALL OPBD

C9
RET
NEW DISK DRIVERS

OEOA        DMARD:
           - TRANSFER DATA FROM CONTROLLER TO MEMORY
           - ENTRY
           $BC =  BYTES TO TRANSFER
           $HL =  FWA OF BUFFER
           - EXIT
           $HL =  NEXT ADDRESS
           $DE =  D*STSR

OEOA 110021 PROC
     LDK DE*D*STSR ;(10)
OEOO 1A   ;LOOP:  LD A*DE ;(7) GET STATUS
OEOE 1F    RAR ;(4)
OEOF 0D    RNC ;(5) RETURN IF NO BUSY
OE10 1F    RAR ;(4)
OE11 D2000E JNC ;LOCP ;(10) IF NO DRQ
OE14 3A0321 LD A*D*DATR ;(13) GET BYTE
OE17 77    STO A*HL ;(7) STORE BYTE
OE18 23    INC HL ;(6)
OE19 0B    DEC BC ;(6)
OE1A 78    MVP A*B ;(4)
OE1B 81    ORA C ;(4)
OE1C C2000E JNZ ;LOCP ;(10)
OE1F C9    RET
NEW DISK DRIVERS

; DMAWRT:
; ITransfer data from memory to disk
; ENTRY
; IBG = BYTES TO TRANSFER
; IHL = FWA OF BUFFER
; EXIT
; IHL = NEXT ADDRESS

DE20       PROC
DE20       110021       LDK    DE+D+STSR
DE23       1A            :LOOP: LD    A+(DE)   ; GET STATUS
DE24       1F            RAR
DE25       0D            RNC    ; RETURN IF NO BUSY
DE26       1F            RAR
DE27       D2230E        JNC    :LOOP     ; IF NO DRQ
DE2A       TE            LD    A+(HL)    ; GET BYTE
DE2B       320321        STO    A+D+DTR    ; STORE BYTE
DE2E       23            INC    HL
DE2F       08            DEC    BC
DE30       78            MOV    A+H
DE31       B1            ORA    C
DE32       C2230E        JNZ    :LOOP
DE35       C9            RET
NEW DISK DRIVERS

DDRV:
:Deselct drive
:Entry
:5DISK = current disk drive

0E36
0E36 3A62EF
0E39 E61F
0E38 4F
0E3C C37109

PROC
LD A\PIABD
AND L_1111b
MOV C\A
JMP DPRD :delect last drive
SORCIM 808x Assembler ver 3.5E 10/5/7 59:92 Page 125
C:SDROMA +ASM

*([7]

OE3F

FORMAT:

This proc will format the next track in IBM 3740 format consisting of 40 tracks, with each
track containing 10 sectors.

;Entry
;
;EXIT
;

OE3F PROC

*SET UP BUFFER AND "DMADR"

;SET DMADR AND GET LBA+1 OF PRESENT BUFFER

OE3F C5
OE40 E1
OE41 4E
OE42 23
OE43 46
OE44 23
OE45 220FEF
OE48 09

PUSH BC
POP HL
LD C +[HL]
INC HL
LD B + [HL]
INC HL
STO HL + DMADR
ADD HL + BC

;TEST DENSITY AND SET REG D TO 04EH OR OFFH

OE49 012003
OE4C 3ADD0F
OE4F 164E
OE51 0F
OE52 3002 ^OE56$
OE54 16FF

LDK BC + 800
LD A, SAVTYP
LDK D + 04EH
RRC A
JRC :PLOOP
LDK D + OFFH

;PAD REST OF BUFFER

OE56 72
OE57 23
OE58 0B
OE59 78
OE5A 81
OE5B 20F9 ^OE56$

: PLOOP: STO D + [HL]
INC HL
DEC BC
MOV A, B
ORA C
JNZ :PLOOP

*FUNCTION DRIVE

;SELECT DRIVE

OE5D CD40D
OE60 3827 ^OE89$

CALL SELDRV
JRC :ERROR

;TEST FOR STEP OR NO-STEP

OE62 2115EF
OE65 3A0121
OE68 8E
OE69 2817 ^OE82$

LDK HL + SAVTRK
LD A + D + TRKR
CMP (HL)
JRZ :1

;IF SAVTRK AND TRACK REG ARE THE SAME SKIP THE STEP
FORMAT

;STEP IN ONE TRACK

OE68 3A13EF
OE6E F610
OE70 3213EF
OE73 CD4D0C
OE76 F5
OE77 3A13EF
OE7A E603
OE7C 3213EF
OE7F F1
OE80 3807 ^OE89$ JRC :ERROR

;FORMAT TRACK

OE82 CD490D :1: CALL FMTTRK
OE85 3802 ^OE89$ JRC :ERROR
OE87 AF
OE88 C9
OE89 3EFF :ERROR: LDK A\,OFFM
OE8B B7
OE8C C9
OEBC RLWA = 0-1 ILWA OF ROM RESIDENT CODE
OE8D = OFFD org Offdh
OFFD = 0003 SERNO ds 3
SORCIM 808x Assembler ver 3.5E <<5/5/7= 59:92 Page 127
C:SOROMA .ASM

@R

1000 = ED80
    ORG MRAM
    SERFLG = OEFDAH

    ! Host disk xfer buffer and...
    ! Format track template holding buffer

    EDTBUF: DS 256+128

    ! Directory Buffer
    DIROBUF: = HSTBUF+256

EF00 = 0006
    TEM DS 6
    = EF01
    = EF02
    = EF04
    = EF05
    = EF06
    = EF07
    = EF08
    = EF09

EF10 = 0000
    MPCHRS DS 1
    = EF11
    = EF12

EF13 = 0001
    SEKDBL DS 1
    = EF14
    = EF15
    = EF16

EF18 = 0001
    SEKSEC DS 1
    = EF19
    = EF20

EF1C = 0001
    HSTSEC DS 1
    = EF1D
    = EF1F

EF20 = 0001
    TEMSEC DS 1
    = EF21
    = EF22
    = EF23

EF24 = 000C
    ALV$ DS ALVS
    CSV$ DS CSVS

EF50 = 0001
    HSTACT DS 1
    = EF51
    = EF52
    = EF53

EF55 = 0001
    HSTWRT$ DS 1
    = UF51
    = UF52
    = UF53

EF58 = 0001
    UNACT$ DS 1
    = UF54
    = UF55
    = UF56

EF5D = 0001
    UNATRK$ DS 2

! BIOS blocking-deblocking flags

: Debug Monitor RAM Storage.
: C:SOROMA .ASM
EF55 = 0001  UNASEC: DS  1  ;Sector
EF56 = 0001  LGSEC: DS  1  ;Logical sector
EF57 = 0002  LOADR DS  2  ;Zero if locked keyboard
EF59 = 0001  KEYLCK DS  1  ;Last valid keystroke
EF5A = 0002  CURS DS  2  ;Current cursor position
EF5C = 0001  TKEY DS  1  ;Temp holding key
EF5D = 0001  HKCNT DS  1  ;Debounce key
EF5E = 0001  LKEY DS  1  ;Last control key
EF5F = 0001  CKEY DS  1  ;ESC holding flag
EF60 = 0001  ESCH DS  1  ;ESC holding flag
EF61 = 0001  PIAAD: DS  1  ;Holds last PIA-A data
EF62 = 0001  PIABD: DS  1  ;Holds last PIA-B data
EF63 = 0003  IDAY DS  3  ;Calendar month, day year
EF64  =  EF64  IDMTN =  IDAY+1
EF65  =  EF65  IYR =  IDAY+2
EF66 = 0006  HOURS: DS  6  ;Wall clock time cells and disk active
EF67 =  EF67  MINS =  HOURS+1
EF68  =  EF68  SECS =  HOURS+2
EF69  =  EF69  SEC= =  HOURS+3
EF6A  =  EF6A  DACTVE =  HOURS+4 I=O by FDSK, Used by UPTIM
EF6B  =  EF6B  BELCNT =  HOURS+5 I=C bell timer cell
EF6C = 0001  LLIMIT DS  1  ;Max #columns in a logical line
EF6D = 0002  LOSEL DS  2  ;Last selected drive
EF6E  =  EF6E  LDTK =  LOSEL+1 ;Last track used for non-selected drive
EF6F = 0002  IESTK: DS  2  ;Save current stk ptr
EF71 = 0028  ISTK: DS  20#2
EF79 = 0000  ISTK: DS  0
EF99 = 0028  ISTACK: DS  20#2
EFC1 = 0000  ROSTK: DS  0
EFC1 = 0000  ACIA0: DS  1  ;Last command byte written to ACIA
SDROM 808x Assembler ver 3.5E <t/55/7= 59/92 Page 129

Debug Monitor RAM Storage.

EFC2 = 0004  R179x: DS 4 '179x register save area
EFC6 = 0001  KBDLY: DS 1 'keyboard debounce-delay cell

: since CP/M CANNOT boot off B1, this cell is used
: to invert the names of the 2 drives:
  : =0 all normal, A=A+, B=B:
  : =1 all inverted, A=B1, B=A:
EFC7 = 0001  DSKSWP: DS 1

: 80 Alternate Register Set
EFC8  ALIGN 10h
EF90  RAGS:
EF90 = 0002  DESAX: DS 2 'IDE'
EF92 = 0002  BCSAX: DS 2 'IBC'
EF94 = 0002  APSAX: DS 2 'IAF'
EF96 = 0002  HLSAX: DS 2 'IHL'
EF98 = 0002  IYSAX: DS 2 'IX'
EF9A = 0002  IVSAX: DS 2 'IXY'
EF9C = 0002  IVSAX: DS 2 'Interrupt page register

: 8000 Register Save Area
EFDE  ALIGN 10h
EFED = 0001  REGS:
EFEE = 0001  ESAVE: DS 1 'E Register save location
EFEO = 0001  OSAVE: DS 1 'O Register save location
EFEE = 0001  CSAVE: DS 1 'C Register save location
EFED = 0001  BSAVE: DS 1 'B Register save location
EFEC = 0001  PSAVE: DS 1 'FLAGS save location
EFED = 0001  ASAVE: DS 1 'A Register save location
EFEE = 0001  LSAVE: DS 1 'L Register save location
EFEO = 0001  HSAVE: DS 1 'H Register save location
EFEB = 0002  PSAVE: DS 2 'PM0 COUNTER save location
EFEC = 0002  SSAVE: DS 2 'USER STACK pointer save location
EFED = 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 EFFD, EFFF.
: EndX MRAM

no ERRORS. 687 Labels, 5EDFh bytes not used. Program LWA = EF00h.
<table>
<thead>
<tr>
<th>Memory Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>19#4</td>
</tr>
<tr>
<td>0003</td>
<td>66#44</td>
</tr>
<tr>
<td>007E</td>
<td>4/4</td>
</tr>
<tr>
<td>0074</td>
<td>128#13</td>
</tr>
<tr>
<td>0081</td>
<td>36#12</td>
</tr>
<tr>
<td>0083</td>
<td>36#10</td>
</tr>
<tr>
<td>0084</td>
<td>36#11</td>
</tr>
<tr>
<td>0086</td>
<td>36#12</td>
</tr>
<tr>
<td>0087</td>
<td>36#13</td>
</tr>
<tr>
<td>0088</td>
<td>36#14</td>
</tr>
<tr>
<td>0089</td>
<td>36#15</td>
</tr>
<tr>
<td>008A</td>
<td>36#16</td>
</tr>
</tbody>
</table>

This document appears to be a list of memory addresses and corresponding values, possibly related to a computer's memory management or similar technical context. The format suggests it might be part of a program or data structure in a computer's system.
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D=DEL 0014</td>
<td>93#52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=FIINT 0000</td>
<td>93#15 115/12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=RODA 0000</td>
<td>93#12 111/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=RES 0080</td>
<td>93#10 107/11 110/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=RES 0000</td>
<td>93#5 102/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=SECR 2102</td>
<td>93#22 109/16 112/10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=SEK 0010</td>
<td>93#6 103/18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=STF 0020</td>
<td>93#7 104/14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=STF 0040</td>
<td>93#8 105/13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=STF 0060</td>
<td>93#9 106/14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=STSR 2100</td>
<td>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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=TRKR 2101</td>
<td>93#21 101/13 103/12 112/11 125/58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=WSRT 00A0</td>
<td>93#11 108/11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=WSRT 00F0</td>
<td>93#14 114/9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DACTVE EF6A</td>
<td>117/39 118/41 128#35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s=DCBT 0000</td>
<td>62/27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDNY 0E36</td>
<td>23/22 123#27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELA 092D</td>
<td>70#19 118/34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DESAX E9D0</td>
<td>129#14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s=DI</td>
<td>3/12 3/33 3/49 5/11 21/11 31/10 33/37 47/13 53/12 56/11 56/16 56/38 58/15 109/60 111/18 113/12 114/11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DIRBUF E80</td>
<td>127#12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DISDAM</td>
<td>3/3 3/13 5/12 31/13 33/41 47/17 53/15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s=DMN 0080</td>
<td>55/14 56/10 56/41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DMADR EF11</td>
<td>127#26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DMDR EF0F</td>
<td>14/24 16/17 18/4 18/17 110/10 113/18 114/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=MDAR 0E0A</td>
<td>8/52 110/19 111/53 113/19 121#15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=MAWRTE 0E20</td>
<td>8/51 110/20 114/18 122#28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=NNRE 008C</td>
<td>66#11 67/35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=ODHOME 08AD</td>
<td>67/38 68#45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=OCAF 0647</td>
<td>40/9 53#39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=OCAF2 06AA</td>
<td>42/16 54#12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=ODMN 008C</td>
<td>19#38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.YS 0001</td>
<td>93#28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.CRC 0008</td>
<td>93#38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.DOQ 0002</td>
<td>93#32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.HOL 0020</td>
<td>93#44 93#48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.IOK 0002</td>
<td>93#31 93#32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.SDO 0004</td>
<td>93#36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.RKY 0080</td>
<td>93#48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.SDP 0010</td>
<td>93#62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.SEK 0010</td>
<td>93#41 93#42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.TKO 0004</td>
<td>93#55 93#36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.WTP 0020</td>
<td>93#48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DS.WTP 0040</td>
<td>93#47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DSEA EFE1</td>
<td>129#28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DSSW 0EFC</td>
<td>4/6 118/15 129#8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=DSSW 0EF0</td>
<td>98/37 111/76 127#22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBODT 0194</td>
<td>19#6 14/18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECHOR EF07</td>
<td>5/23 127#20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDDEL 050S</td>
<td>26/18 46#18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEED 0540</td>
<td>26/21 43#11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EFADR 0010</td>
<td>24#6 45/5 45/14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SORDIM 8908 Assembler ver 3.5E <:/55/7= 59:92 Page 132
C:SORDMA .ASM

SINGLE Density Monitor for Model 1 system

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
EFHMSK 0007 24/81 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
s ESI 3/8 3/30 5/11 6/13 21/17 31/14 33/42
sd EIM 47/18 53/16 55/15 56/20 56/42 110/6 113/21
EINSRT 05EE 26/17 47#20
ENMODT 0187 10#2 10/12
s ENADIM 31/11 33/38 47/14 53/13 55/12 56/17 56/39
s ENARDM 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 0584 26/7 44#7
ESCCDR 0617 26/10 49#18
ESCEHA 0618 26/12 49#21
ESCEHC 0618 26/14 49#24
ESCEH 0624 26/19 50#4
ESCH EF60 5/24 28/39 29/27 30/15 35/47 45/7 49/15
128#14
ESCHTB 0384 25#3 29/4
ESCLKC 05A3 26/22 42#46
ESCR 0624 26/20 49#31
ESCSSD 058E 26/8 49#10
ESCSSR 0607 26#9 49#3
ESCSHA 0608 26/11 49#7
ESCSUN 060F 26/13 49#11
ESCUK 05A6 26/23 43#6
ESCUZ 05BE 26/16 49#29
EXITI 001E 3/19 58/25
s FALSE 0000
s FCB 005C
n FDSK 0084 109/60 111/18 113/12 114/11 116/15 116#18
n FILLC 093F 72#11
FILLZ 093D 18/53 71#28
FMTRRX 0AD9 8/60 113#26 126/20
FDINT 0664 8/58 110/34 111/43 114#30 117/17
FORMAT 0E3F 8/29 125#3
n FSAYE EFE4 129#31
s FWAVM F000 38/10 39/13 54/17
s GREY 071F 5/12 58#3
n GREYX 0739 58#26
s GTMARK 0TE5 63/15 63/18 64#8
s H+FD 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+SRC 2A01 88/17
s H+SSTS 2A00 91/14
s H+SX 2A01 90/16
s H+VIO 2C00 23/7 73/4 73/8 73/10 74/21 74/25
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HINT</td>
<td>080D</td>
</tr>
<tr>
<td>HMCNT</td>
<td>EF5D</td>
</tr>
<tr>
<td>HLSAX</td>
<td>EF66</td>
</tr>
<tr>
<td>HMSMRR</td>
<td>055B</td>
</tr>
<tr>
<td>HMDE</td>
<td>08F7</td>
</tr>
<tr>
<td>HOURS</td>
<td>EF66</td>
</tr>
<tr>
<td>HSAVE</td>
<td>EF7</td>
</tr>
<tr>
<td>HSTACT</td>
<td>EF50</td>
</tr>
<tr>
<td>HSTBUFF</td>
<td>EDB0</td>
</tr>
<tr>
<td>HSTDSK</td>
<td>EF1F</td>
</tr>
<tr>
<td>HSTSEC</td>
<td>EF1C</td>
</tr>
<tr>
<td>HSTRX</td>
<td>EF1D</td>
</tr>
<tr>
<td>HSTWRT</td>
<td>EF51</td>
</tr>
<tr>
<td>IDAY</td>
<td>EF63</td>
</tr>
<tr>
<td>IE-CO</td>
<td>097E</td>
</tr>
<tr>
<td>IE-PTS</td>
<td>040D</td>
</tr>
<tr>
<td>IE-IDM</td>
<td>0A25</td>
</tr>
<tr>
<td>IE-ODM</td>
<td>0A43</td>
</tr>
<tr>
<td>IE-OIM</td>
<td>0A20</td>
</tr>
<tr>
<td>IE-PP</td>
<td>0AE1</td>
</tr>
<tr>
<td>IE-SHK</td>
<td>0A4F</td>
</tr>
<tr>
<td>IE-SI</td>
<td>09C1</td>
</tr>
<tr>
<td>IE-TC</td>
<td>09E0</td>
</tr>
<tr>
<td>IESTK</td>
<td>EF6F</td>
</tr>
<tr>
<td>ILINT</td>
<td>0007</td>
</tr>
<tr>
<td>IMONTH</td>
<td>EF64</td>
</tr>
<tr>
<td>IMSE</td>
<td>01B6</td>
</tr>
<tr>
<td>INTBL</td>
<td>EF00</td>
</tr>
<tr>
<td>IRPTCT</td>
<td>0018</td>
</tr>
<tr>
<td>ISTM</td>
<td>EF09</td>
</tr>
<tr>
<td>IVA5</td>
<td>EFDC</td>
</tr>
<tr>
<td>IVA5</td>
<td>EFDB</td>
</tr>
<tr>
<td>IYR</td>
<td>EF65</td>
</tr>
<tr>
<td>IYSA5</td>
<td>EFDA</td>
</tr>
<tr>
<td>KBDLY</td>
<td>EF66</td>
</tr>
<tr>
<td>KBRVR</td>
<td>O7SC</td>
</tr>
<tr>
<td>KBSCE</td>
<td>O774</td>
</tr>
<tr>
<td>KBSRV</td>
<td>07F8</td>
</tr>
<tr>
<td>KCOLM</td>
<td>0007</td>
</tr>
<tr>
<td>KEYLJK</td>
<td>EF59</td>
</tr>
<tr>
<td>KEYLST</td>
<td>EF64</td>
</tr>
<tr>
<td>KLELEN</td>
<td>0002</td>
</tr>
<tr>
<td>KLEN</td>
<td>0003</td>
</tr>
<tr>
<td>KLUSE</td>
<td>0007</td>
</tr>
<tr>
<td>KRMNW</td>
<td>0038</td>
</tr>
<tr>
<td>KYCDTB</td>
<td>08C3</td>
</tr>
<tr>
<td>KYSGD</td>
<td>0006</td>
</tr>
<tr>
<td>LDADR</td>
<td>EF57</td>
</tr>
<tr>
<td>LDSEL</td>
<td>EF6D</td>
</tr>
<tr>
<td>LDTRK</td>
<td>EF6E</td>
</tr>
<tr>
<td>LF</td>
<td>000A</td>
</tr>
<tr>
<td>LEFT</td>
<td>000D</td>
</tr>
<tr>
<td>LFTARM</td>
<td>0080</td>
</tr>
<tr>
<td>LIST</td>
<td>081C</td>
</tr>
<tr>
<td>LKEY</td>
<td>EF5E</td>
</tr>
<tr>
<td>LLLIMIT</td>
<td>EF6C</td>
</tr>
<tr>
<td>LOGSEG</td>
<td>EF56</td>
</tr>
<tr>
<td>LOOKUP</td>
<td>047C</td>
</tr>
</tbody>
</table>

Note: The values listed are hexadecimal addresses or values.

**SINGLE Density Monitor for Model 1 system**

*Cr:SOROMA .ASM*
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAVE</td>
<td>FF6</td>
<td>129#33</td>
</tr>
<tr>
<td>LVMEM</td>
<td>0000</td>
<td>39/3</td>
</tr>
<tr>
<td>MCDOWN</td>
<td>0000</td>
<td>19#9</td>
</tr>
<tr>
<td>MCLFET</td>
<td>0000</td>
<td>19#1C</td>
</tr>
<tr>
<td>MCRIGH</td>
<td>0000</td>
<td>19#11</td>
</tr>
<tr>
<td>MCRUP</td>
<td>0000</td>
<td>19#8</td>
</tr>
<tr>
<td>MINS</td>
<td>EF67</td>
<td>128#25</td>
</tr>
<tr>
<td>MPCHR</td>
<td>EF66</td>
<td>3/17</td>
</tr>
<tr>
<td>MRAM</td>
<td>ED80</td>
<td>3/4C</td>
</tr>
<tr>
<td>MSEC8</td>
<td>0010</td>
<td>93#3</td>
</tr>
<tr>
<td>MIA7</td>
<td>0066</td>
<td>3/4F</td>
</tr>
<tr>
<td>NORM</td>
<td>038F</td>
<td>21#1B</td>
</tr>
<tr>
<td>NRETRY</td>
<td>0000</td>
<td>92#3</td>
</tr>
<tr>
<td>NWOL</td>
<td>0018</td>
<td>95#46</td>
</tr>
<tr>
<td>OPAD</td>
<td>0964</td>
<td>5/45</td>
</tr>
<tr>
<td>OPAD</td>
<td>0971</td>
<td>5/51</td>
</tr>
<tr>
<td>OSRR</td>
<td>0972</td>
<td>3/45</td>
</tr>
<tr>
<td>PIAAD</td>
<td>EF61</td>
<td>34#79</td>
</tr>
<tr>
<td>PIAAD</td>
<td>EF62</td>
<td>39#7</td>
</tr>
<tr>
<td>PMCHR</td>
<td>003E</td>
<td>6/1C</td>
</tr>
<tr>
<td>POPALL</td>
<td>mac</td>
<td>56#15</td>
</tr>
<tr>
<td>PSAVE</td>
<td>DF68</td>
<td>129#33</td>
</tr>
<tr>
<td>PSEC</td>
<td>0076</td>
<td>124#18</td>
</tr>
<tr>
<td>PSETSC</td>
<td>0455</td>
<td>29#29</td>
</tr>
<tr>
<td>PUSHAL</td>
<td>mac</td>
<td>94#18</td>
</tr>
<tr>
<td>R17XR</td>
<td>EF61</td>
<td>128#61</td>
</tr>
<tr>
<td>RADR</td>
<td>OEC</td>
<td>280#19</td>
</tr>
<tr>
<td>RAGS</td>
<td>EF00</td>
<td>129#13</td>
</tr>
<tr>
<td>RDIFLAG</td>
<td>EF21</td>
<td>127#40</td>
</tr>
<tr>
<td>RDRKDN</td>
<td>D7EF</td>
<td>61#11</td>
</tr>
<tr>
<td>RDRV</td>
<td>DF40</td>
<td>8/16</td>
</tr>
<tr>
<td>ROSKO</td>
<td>DFF</td>
<td>118#32</td>
</tr>
<tr>
<td>RDTWRT</td>
<td>EF01</td>
<td>92#5</td>
</tr>
<tr>
<td>RDWR</td>
<td>DC82</td>
<td>107#14</td>
</tr>
<tr>
<td>READ</td>
<td>GC75</td>
<td>8/2C</td>
</tr>
<tr>
<td>READER</td>
<td>0803</td>
<td>8/14</td>
</tr>
<tr>
<td>READTR</td>
<td>D33</td>
<td>8/55</td>
</tr>
<tr>
<td>REGS</td>
<td>EF00</td>
<td>129#26</td>
</tr>
<tr>
<td>REPOD</td>
<td>0020</td>
<td>22#15</td>
</tr>
<tr>
<td>REPDK</td>
<td>0025</td>
<td>228#16</td>
</tr>
<tr>
<td>RIGHT</td>
<td>0088</td>
<td>19#37</td>
</tr>
<tr>
<td>RKEY</td>
<td>O37E</td>
<td>21#2</td>
</tr>
<tr>
<td>RLSA</td>
<td>D8C</td>
<td>122#53</td>
</tr>
<tr>
<td>RNDDV</td>
<td>EF01</td>
<td>127#15</td>
</tr>
<tr>
<td>RDMJP1</td>
<td>00DF</td>
<td>3/26</td>
</tr>
<tr>
<td>RDMJ2</td>
<td>00E4</td>
<td>7#3</td>
</tr>
<tr>
<td>RDMRAM</td>
<td>EF08</td>
<td>3/24</td>
</tr>
<tr>
<td>RMSTK</td>
<td>EF21</td>
<td>3/33</td>
</tr>
<tr>
<td>RDMWN</td>
<td>0081</td>
<td>61#14</td>
</tr>
<tr>
<td>RSEC</td>
<td>OS8B</td>
<td>8/23</td>
</tr>
<tr>
<td>RTDRW</td>
<td>0888</td>
<td>66#3</td>
</tr>
<tr>
<td>RTRC</td>
<td>EF04</td>
<td>127#17</td>
</tr>
<tr>
<td>RTRY</td>
<td>EF05</td>
<td>95/12</td>
</tr>
<tr>
<td>RMSCE</td>
<td>08B4</td>
<td>96#17</td>
</tr>
<tr>
<td>SAYSEC</td>
<td>EF14</td>
<td>14#29</td>
</tr>
<tr>
<td>SAYTRK</td>
<td>EF15</td>
<td>14#27</td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>SAVYTP EFDD</td>
<td>127#33 127/37</td>
<td></td>
</tr>
<tr>
<td>SCLFRD 0085</td>
<td>3/38 16/98 92# 4 98/43 109/23 119/20 125/33</td>
<td></td>
</tr>
<tr>
<td>SCREEN 0444</td>
<td>71/13 33#43 14/14</td>
<td></td>
</tr>
<tr>
<td>SCTRKR 0800</td>
<td>92 3 100# 2</td>
<td></td>
</tr>
<tr>
<td>SDISK EF17</td>
<td>3/35 14/13 14/14 16/16 118/14 127#34 127/38</td>
<td></td>
</tr>
<tr>
<td>SCA  EF69</td>
<td>128#31</td>
<td></td>
</tr>
<tr>
<td>SCS  EF68</td>
<td>128#30</td>
<td></td>
</tr>
<tr>
<td>SEER OC1A</td>
<td>8/54 99/29 102#34</td>
<td></td>
</tr>
<tr>
<td>SEEKTM 0002</td>
<td>67/5</td>
<td></td>
</tr>
<tr>
<td>SEDEL EF13</td>
<td>106/18 116/12 126/4 126/6 126/11 126/13 127#30</td>
<td></td>
</tr>
<tr>
<td>SEKOSK EF18</td>
<td>127#42</td>
<td></td>
</tr>
<tr>
<td>SEKSEC EF19</td>
<td>127#40</td>
<td></td>
</tr>
<tr>
<td>SEKTRAK EF19</td>
<td>127#41</td>
<td></td>
</tr>
<tr>
<td>SECTION 0044</td>
<td>118/12 118#45</td>
<td></td>
</tr>
<tr>
<td>SELDR 0044</td>
<td>8/61 95/13 98/26 99/21 117#42 125/52</td>
<td></td>
</tr>
<tr>
<td>SENDEL 0380</td>
<td>8/26 14/15 16/27 97#28 99/26 99/49</td>
<td></td>
</tr>
<tr>
<td>SEFLEGE EF60</td>
<td>118#15 91/18 91/21 127#4</td>
<td></td>
</tr>
<tr>
<td>SERNO 0004</td>
<td>126#41</td>
<td></td>
</tr>
<tr>
<td>SEXTY 0404</td>
<td>30/10 34#36</td>
<td></td>
</tr>
<tr>
<td>SHFRTY 0040</td>
<td>66/42 67/40</td>
<td></td>
</tr>
<tr>
<td>SHFRTY 0041</td>
<td>67/15 69#32</td>
<td></td>
</tr>
<tr>
<td>SIA#WRS 0057</td>
<td>87/10</td>
<td></td>
</tr>
<tr>
<td>SIARRD 0001</td>
<td>88/11</td>
<td></td>
</tr>
<tr>
<td>SIARSD 0055</td>
<td>5/60</td>
<td></td>
</tr>
<tr>
<td>SIALSD 0056</td>
<td>89/13</td>
<td></td>
</tr>
<tr>
<td>SIASTY 0002</td>
<td>5/61 8/33 87# 2</td>
<td></td>
</tr>
<tr>
<td>SKEOY 0371</td>
<td>194#20 21/ 3</td>
<td></td>
</tr>
<tr>
<td>SLEDDC 0003</td>
<td>68/20 68#28 68#4</td>
<td></td>
</tr>
<tr>
<td>SLIDE 0899</td>
<td>67/36 68#28</td>
<td></td>
</tr>
<tr>
<td>SLIDE 0899</td>
<td>67/36 68#28 68#4</td>
<td></td>
</tr>
<tr>
<td>SLIDEU 0899</td>
<td>67/34 68#28</td>
<td></td>
</tr>
<tr>
<td>SLT 0813</td>
<td>5/43 72#26</td>
<td></td>
</tr>
<tr>
<td>SPAD 0404</td>
<td>60/44 5/43 72#26</td>
<td></td>
</tr>
<tr>
<td>SSRCTC 0006</td>
<td>3/41 129#36</td>
<td></td>
</tr>
<tr>
<td>SSAAFE EF6A</td>
<td>9/55 103#32</td>
<td></td>
</tr>
<tr>
<td>STEPM 0C40</td>
<td>8/56 104#28 126/ 8</td>
<td></td>
</tr>
<tr>
<td>STEPDU 0C1A</td>
<td>8/57 105#27</td>
<td></td>
</tr>
<tr>
<td>STDIM 0699</td>
<td>9/67 54#58</td>
<td></td>
</tr>
<tr>
<td>SVRS 0001</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>SYS 0005</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>SYSDAT 0100</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>SYSL 0006</td>
<td>67/3</td>
<td></td>
</tr>
<tr>
<td>TAI 0099</td>
<td>16/21 17/3 17/21 17/40 18/33 127#14 127/15</td>
<td></td>
</tr>
<tr>
<td>TEM 0E00</td>
<td>127#48</td>
<td></td>
</tr>
<tr>
<td>TEMSEC EF20</td>
<td>128#10</td>
<td></td>
</tr>
<tr>
<td>TKEY EF5C</td>
<td>128#10</td>
<td></td>
</tr>
<tr>
<td>TOTDMD 0007</td>
<td>61/28 61/36</td>
<td></td>
</tr>
<tr>
<td>TRUE FFFF</td>
<td>89/16</td>
<td></td>
</tr>
<tr>
<td>UNACSEF EF52</td>
<td>127#59</td>
<td></td>
</tr>
<tr>
<td>UNASEC EF55</td>
<td>18/55 127#61</td>
<td></td>
</tr>
<tr>
<td>UNATRK EF53</td>
<td>127#60</td>
<td></td>
</tr>
<tr>
<td>UNCUR 0719</td>
<td>35/10 36/ 5 37/ 3 37/13 40/ 2 41/ 2 43/ 4</td>
<td></td>
</tr>
<tr>
<td>UP 008A</td>
<td>54/11 56#43</td>
<td></td>
</tr>
<tr>
<td>UPAWR 008A</td>
<td>19#36</td>
<td></td>
</tr>
<tr>
<td>UPAWR 008A</td>
<td>66#10 67/33</td>
<td></td>
</tr>
<tr>
<td>GRCSIM 808x Assembler ver 3.5E &lt;t/55&gt; 59:92 Page 136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SINGLE Density Monitor for Model 1 System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C:SDROMA +ASM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| UPTIM | 0398 | 22/17 | 58/19 |
| USER | 0005 | 385 |  |
| VALCTS | 0088 | 27/17 | 31/19 |
| VALETS | 0010 | 26/28 | 30/18 |
| VALDIE | 0356 | 26/17 | 31/18 |
| VALIDE | 0364 | 25/12 | 26/28 | 30/17 |
| VBRIGH | 0468 | 31/8 | 33/15 |
| VCAD | 0030 | 19/19 | 26/6 |
| VCECL | 053E | 27/12 | 37/63 |
| VCEBS | 0512 | 27/9 | 37/11 |
| VCECLS | 054E | 27/13 | 38/10 | 49/29 |
| VCCR | 056F | 27/7 | 39/15 |
| VCEDL | 0044 | 19/25 | 26/20 |
| VCHOME | 04FB | 27/14 | 36/3 |
| VCLK | 0578 | 27/8 | 40/8 |
| VCLRS | 001A | 19/22 | 26/15 | 27/12 |
| VCMCHR | 0570 | 27/10 | 40/11 |
| VCMCUP | 0509 | 27/11 | 36/12 |
| VDEL | 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 |
| VEH | 0028 | 19/27 | 26/11 |
| VEUL | 006D | 19/29 | 26/13 |
| VFAD | FFA | 5/44 | 34/23 | 68/51 |
| VGRAP | 0581 | 25/5 | 42#4 |
| VHAGR | 0499 | 25/7 | 33#34 |
| VHALF | 0495 | 25/6 | 33#28 |
| VHOMA | 001E | 19/13 | 27/13 |
| VINC | 0051 | 19/21 | 26/16 |
| VINT | 0065 | 19/23 | 26/18 |
| VLDDR | 06F1 | 8/45 | 48/13 |
| VLDIR | 0705 | 8/46 | 39/6 | 47/10 | 51/16 | 55/16 | 55/23 |
| VLDL | 0034 |  |

| YLL | 0080 | 5/35 | 18/56 | 37/5 | 37/22 | 37/36 | 39/3 | 46/5 |
| YLOCK | 0023 | 19/17 | 26/21 |
| YNMR | 0467 | 25/4 | 30#21 | 33/9 | 33/20 | 33/31 |
| YOUT | 0581 | 31/14 | 33/42 | 42#5 | 42/45 |
| YOUTS | 0595 | 42/12 | 42#19 |
| YOUTU | 0596 | 35/48 | 37/18 | 37/42 | 39/14 | 40/6 | 40/10 | 42/17 |
| YOUTV | 0597 | 42/23 | 44/6 | 46/17 | 47/19 | 51/4 |

| YOUTW | 0598 | 42/29 | 48/18 |
| YOUTX | 059E | 27/15 | 29/28 | 38/9 | 42#36 | 42/43 | 43/10 | 45/8 |
| YRTOFF | EFEF | 3/36 | 28/27 | 34/13 | 39/12 | 54/52 | 68/58 | 129#41 |
| YSAD | 0053 | 19#20 | 26/7 |
| YSGH | 0067 | 12/4 | 12/12 | 12/18 | 12/22 | 12/30 | 12/40 | 19#31 |
| YSH | 0029 | 19#26 | 26/10 |
| YSUL | 006C | 19/28 | 26/12 |
| YVNDER | 0487 | 25/8 | 33#6 |
| YUNGR | 0488 | 25/9 | 33#12 |
| YUNH | 048F | 25/10 | 33#17 |
| YUNHAG | 0493 | 25/11 | 33#23 |
| YUNLIK | 0022 | 19#10 | 26/22 |
| YWOD | 02A8 | 8/8 | 14#61 |
| YWBUSY | 00E6 | 110/31 | 111/57 | 115/23 | 116/17 | 119#28 |
WRITE OC70 8/21 97/16 107#15
n WRTYPE EF23 127#51
WSEC 0860 8/24 96#27
NOTE: FOR USE WITH QGCXT6.AST ONLY

- 4D20C7-00 MASTER *ASM
- 2D20C7-00 ASSY  *ASM
- 1D20C7-00 LISTING *PRN
- 4D1007-00 MASTER *COM
- 2D1007-00 ASSY  *COM

Copyright 1982, Osborne.

This product is a copyright program product of Osborne and is supplied for use with the Osborne.

REV  = 1.3
IDATE = FEB 14 1982
IRWC

Revisions:
1. Extensions to CB10S added by:
   Microcode Corporation, Fremont, CA
   Y. M. Sahne
   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
LINK OCC81D13.ASM :Jump Table
LINK OCC81D23.ASM :CP/M disk definitions
LINK OCC81D33.ASM :Unit record I/O
LINK OCC81D43.ASM :Non data transfer disk
LINK OCC81D53.ASM :Cold and warm boot
LINK OCC81D63.ASM :Disk data transfer I/O
LINK OCC81D73.ASM :Utility routines
LINK OCC81D83.ASM :Utility routines
LINK OCC81A2.ASM :Common ram definitions

: END OCC81D03.ASM
SORCIM 808x Assembler ver 3.5E <1/557= 59:92 Page 3
E:0CC81013.ASM

Revisions:
- YNS: 18 AUG 81
- RWC: 05 OCT 81
- RWC: 16 OCT 81

; change bios jump vector to call new routines
; standardize cpm jump table (change calls to jumps)
; added function key table and user defined switches

= 0005 MRTRY: = 5 ;Maximum number of retries.
= 003C msize: = 60
= C000 CCP: = (msize-8)+1024 ;location of CCP
= E500 bios: = CCP+1600h
= DT06 bjos: = CCP+806h

MSG *Assembling BIOS for LWA of \, LWAMEM,\n* *Assembling BIOS for LWA of FFFFh.*

; CP/M to host disk constants
= 0100 MTSIZE: = 256 ;Blocking/Deblocking buffer size
= 0010 FPYS1B: = 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)/FPYS1B

; BDOS constants on entry to write
= 0000 WRALL: = 0 ;write to allocated
= 0001 WRCIR: = 1 ;write to directory
= 0002 WRAUL: = 2 ;write to unallocated

; ROM equates.
= 0000 ENROM: = 0 ;Port to enable ROM
= 0001 DIROM: = 1 ;Port to disable ROM
Macro for generating Control Blocks for disk drives
The format of these disk control blocks are as follows:
16 bits = \rightarrow translation table,
48 bits = Work area for CP/M,
16 bits = \rightarrow DSKBUF,
16 bits = \rightarrow parameter block,
16 bits = \rightarrow check vector,
16 bits = \rightarrow allocation vector.

= 0000  NOSK:  SET  0  ; Number of disk drives
= 0000  NOFOD: SET  0  ; Number of floppy disk drives
= 0000  ALVSZ: SET  0  ; Allocation vector size
= 0000  CSVSZ: SET  0  ; Check vector size

LIST 9,CP/M

DPGEN MACRO TYPE,XLATE,DIRBUF,CPBADR
NOSK: SET NOSK+1
DW  22
DW  0+0
DW  23
DW  24
DW  CSV+CSVZ
DW  ALV+ALVSZ
NOFOD: SET NOFOD+1
CSVZ: SET CSVZ+(64/4)
ALVSZ: SET ALVSZ+((SIOM+7)/8)
ENDM

Make sure Systext agrees with assembled size

Macro for generating the Disk Parameter Blocks.
Disk type definition blocks for particular mode.
The format of these areas are as follows:
8 bit = disk type code
15 bit = sectors per track
8 bit = block shift
8 bit = S5 mask
3 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,55H,55H,EXM,DSM,DIRSZ,ALVMSK,OFFSET
DW  31
DW  22
DS  23,24,25
DW  26-1,27-1,REV (28)
DW  (27+3)/4
DW  29
ENDM
0000 = E500

E500 C30CE7 jmp CBIDT :Cold boot
E503 C334E7 jmp W800T :Warp boot
E506 C30BE9 jmp cnsa :Console status (input)
E509 C31DE9 CONIN1 jmp cnin :Console input
E50C C31BE9 CONOUT: jmp cnout :Console output
E50F C31DE9 LIST: jmp lst :List output
E512 C325E9 PUNCH: jmp punch :Punch output
E515 C320E9 READER: jmp rdr :Reader input
E518 C39FE6 JMP HOME :Set track to zero
E51B C38BE6 RSELX: JMP SEDSK :Select disk unit
E51E C30DE6 JMP SETTRK :Set track
E521 C3F0E6 JMP SETSEC :Set sector
E524 C3F5E6 JMP SETDMA :Set Disk Memory Address
E527 C3D0E7 RRDK: JMP RRD :Read from disk
E52A C3F0E7 RWDK: JMP WRD :Write onto disk
E52D C335E9 LISTST: jmp lstst :Return LST device status
E530 C3F5E6 JMP SECTRN :Sector translation routine

Extensions

E533 C34FE6 RRI: JMP ROMRI
E536 C36AE6 JMP ROMJMP
E539 C663E6 PMTJ: CALL ROMODE :From resident call
E53C C663E6 SRAUD: CALL ROMODE

IEEE-488 vectors

E53F C663E6 ieb1c: CALL ROMODE :IControl Out
E542 C663E6 ieb2c: CALL ROMODE :ISubstrate
E545 C663E6 ieb3c: CALL ROMODE :ISubstrate
E548 C663E6 ieb4c: CALL ROMODE :ITake Control
E54B C663E6 ieb5c: CALL ROMODE :IOutput Interface Message
E54E C663E6 ieb6c: CALL ROMODE :IOutput Device Message
E551 C663E6 ieb7c: CALL ROMODE :IInput Device Message
E554 C663E6 ieb8c: CALL ROMODE :IParallel Poll
E557 C663E6 call romode :Extensions
E55A C663E6 call romode :for
E55D C663E6 call romode :memory-mapped video
E560 C3C5E4 jmp acint :Hook for serial command port write
E563 C39FEA 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, iobase value,
- and auto horizontal scroll flag
- RWC

---------------------------

E566 40 Iobase db 40h :default to serial printer=40h
E571 parallel printer=80h
E574 IEEE printer=60h
SORCIM 808x Assembler version 3.5E  <1/55/7>  59:92  Page 7

Osborne CP/M 2.2 CBIOS

: 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, 5*9, 16*17
: DB 2*3, 10*11, 18*19
: DB 4*5, 12*13
: DB 6*7, 14*15

= 0000

IF (9-XLTS) <> (MSEC@2)
MSG 'Translation table error',ERROR
ENDIF

: Control Blocks for disk drives

E614   DPBASE:
E614   DPGEN DSK51 XLTS DIRBUF CP851+1 ;Drive A:
       00001 NDSK: SET NDSK+1
E614   +00E6 DW XLTS
E615   +000000000 DW 0+0+0
00
E61C   +0D0E DW DIRBUF
E61D   +35E6 DW DP851+1
E620   +30EF DW CSV+CSV\[Z
E622   +24EF DW ALV+ALVSZ
       00001 NOFD\[C: SET NOFD\[C+1
       0010 CSV\[Z: SET CSV\[Z+(64/4)
       0006 ALVSZ: SET ALVSZ+(S1DSM+1/8)

E624   +0002 NDSK: SET NDSK+1 ;Drive B:
E624   +00E6 DW XLTS
E625   +000000000 DW 0+0+0
00
E62C   +0D0E DW DIRBUF
E62D   +35E6 DW DP851+1
E630   +40EF DW CSV+CSV\[Z
E632   +2A0F DW ALV+ALVSZ
       00002 NOFD\[C: SET NOFD\[C+1
       0020 CSV\[Z: SET CSV\[Z+(64/4)
       000C ALVSZ: SET ALVSZ+(S1DSM+1/8)

: Disk type definition blocks for each particular mode.

E634   DP851: ;Single density, single sided.
E634   DPGEN DSK51,2*MSEC+,4,15+1,S1DSM,64,1000000000000000000000003
E634   +00 DB DSK51
E635   +1400 DW 2*MSEC
E637   +000F01 CB 4,15+1
E638   +2D03F00000 DW S1DSM-1,64-1,REV (1000000000000000005
E640   +1000 DW (64/3)/4
E642   +0300 DW 3
= 0000
  AL IF ALVSZ <> ALVS
  MSG "Allocation problems ALVS< > ALVSZ, ALVSZ= " ALVSZ
  AL ENDIF

  endx GS801352
Revisions:
- YNS 18AUG81 added iobyte function to listst
- YNS 30AUG81 modified romode routine
- YNS 23SEP81 moved const and listst to osbios9.asm
- YNS 30AUG81 saved hl in ROMOE routine when returning
to RAM as IEEE routines return status in
- RML CONV91 HL
- Preserved value of HL when calling ROM
- routines, HL used in VLDRX, VLDIR, & STDDIK
- Preserved HL & OE on exit

E644
SETRAM: Set ROM-RAM flag
Entry A= port to output (setting ROM or RAM enable)
Exit none.
Uses A

E644
E645 +F3
E646 4F
E647 ED79 $ GT+C A
E649 3208EF STC A,ROMRAM
E64C C1
E64D +FB
E64E +C9 RET

E64F
ROMRI: Exit ROM resident interrupt routine.
E652 4F
E653 ED79 $ GT+C A
E655 DDE1 $ POP Y
E657 DDE1 $ POP IX
E659 E1
E660 D1
E661 C1
E66C F1
E66D ED76BEF $ LD SP, IESTK
E66E +FB
E66F +C9 RET

E670 0F
E663 ROMCODE: iCall 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 CBICS vector in reg E
E663 D1 POP DE  ;Get calling address
E664 7B MOV A,E
E665 603 SUI 5
E666 5F MOV E,A
E667 ROMCD1: LDX 0,high (ROMVEC)
E667 1601
E668 ROMJMP: iEntry here to jump to ROM function code directly
   ; Entry DE = ROM jump address
   ; BC, HL, IX, any parameters
E66A 29DE6 sta hl,stohl
E66B D1 DI
E66C +F3 DB 0F3h
E66D 210000 LDX HL,SP  ;Load stack to HL
E671 39 ADD HL,SP
E672 31C1EF LDX SP,BXSTK  ;Save old stack pointer
E675 E5 PUSH HL
E676 2A9DE6 ld hl,stoih  ;Get user hl
E679 28AD0 CMROM
E679 + DI
E679 +F3 DS 0F3h
E67A +D300 OUT 0
E67C +5800 LDX A+0
E67E +3208EF STG A,RROMRAM
E681 EI
E681 +F3 DB 0F8h
E682 DF push DF  ;ROM jump address to IY
E683 EDE1 $ pop IY
E685 CD9AE6 CALL GORJM
E688 D1 DI
E688 +F3 DB 0F3h
E689 FE PUSH AF  ;save status returned
E68A D1 SROM
E68A + DI
E68A +F3 DB 0F3h
E68B +D301 OUT 1
E68D +3E1L LDX A+1
E68F +3208EF STG A,RROMRAM
E692 EI
E692 +F3 DB 0F9h
E693 EI POP AF  ;Restore old stack pointer
E694 FDE1 $ pop IY
E696 FD9 $ MOV SP,iy
E698 RET EI ! RET
E698 +F6 DB 0F8h
E699 +C9 RET
E69A G3ROM: EI
E69A +F8 DB 0F8h
E69B FDE9 $ JMP [IY]
E69D = 0002 STOHL DS 2
Return disk to home. This routine sets the track number to zero. The current host disk buffer is flushed to the disk.

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 of drive if drive is selected.

ENTRY C = disk selection value (0..15).
DE and 1 = 0: first call for this disk.
EXIT HL = DPH address of drive if drive is selected.
SET TRACK.

Set track number. The track number is saved for later use during a disk transfer operation.

ENTRY BC = track number.

E600  E64310EF $  STC BC,SEETRK  :Set track
E610  2A33EF    LHLD UNATRK
E660  7D       MOV A+L
E661  4A       XRA C
E662  4F       MOV C+A
E663  7C       MOV A+H
E664  4B       XRA B
E665  51       ORA C
E666  C8       JMP CUNACT  :If same track

Clear unallocated block count (force pre-reads).

E667  AF       CUNACT: XRA A  ; A = 0
E668  3232EF   STO A+UNACNT  :Clear unallocated block count
E669  C9       RET

Set the sector for later use in the disk transfer. No actual disk operations are performed.

ENTRY BC = sector number.

E6F0  79       SETSEC: MOV A+C
E6F1  3220EF   STO A+TEMSEC  :sector to seek
E6F2  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 SC = Disk memory address.

EXIT DMAADR = SC.

E6F5  SETDMA: E64311EF $  STO BC,DMAADR
E6F6  C9       RET
Translate sector number from logical to physical.

ENTRY DE = 3, no translation required.

DE = translation table address.

SC = sector number to translate.

EXIT HL = translated sector.

E6FA   3A55EF          LDA  UNASSE
E6FD   B9              CMP  C
E6FE   C42BE6          CMZ  CUNACT
E701   79              MOV  A+C
E702   3256EF          STO  A+LOGSEC
E705   69              MOV  L+C
E706   60              MOV  H+B
E707   19              ADD  H+DE
E708   6E              MOV  L+W
E709   2500            MVI  H+C
E70B   C9              RET
Revisions:
- RWC 200CCT91: warm boot makes dir 2 files across
- RWC 190CCT91: included baud rate and screen size initialization
- RWC 140CCT91: change lbyte to variable
- YNS 200ALG91: modified auto start feature
- YNS 200ALG91: change lbyte defaults

Booting CP/M from disk.

The LH9D5 entry point gets control from the colt ext 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 warm start jump vector.
- 3: Set to the initial I63YTE 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 0 must contain the selected drive, which is zero to select the A drive. The exit address is to the CGP routine.

The LH9D5 entry point gets control when a warm start occurs, a "C" from the console, a jump to BDOS (function 01), or a jump to location zero. The LH9D5 routine reads the CGP and BDOS from the appropriate disk sectors.

LH9D5 must also re-initialize locations C:1;2 and 5:6,7.

The LH9D5 routine exits with the 0 register set to the appropriate drive selection value. The exit address is to the CGP routine.

E70C E70C 47
E70D E70D 8A
E70E E70E DISCM
E70F E70F CI
E710 E710 OF3h
E711 E711 D9h
E712 E712 01
E713 E713 LDK A:1
E714 E714 STO A:ROMRDM
E715 E715 +300h
E716 E716 +300h
E717 E717 MOV A:0
E718 E718 MOV A:COISK
E719 E719 MOV A:CCP
E71A E71A S1000h

Entry A: drive to boot off of.

MOV B: A

Save requested drive

E710 3A66E5
E712 320300
E713 3A69E5
E715 4C
E716 CD3CE5
E71A 3A6AE5

Id a:lobyte
STC A:163YTE
Id a:rate
mov c,a
call 3baud
Ida scrsze

Get lbyte value
Set I/C byte to default
Set baud rate
E720 326CEF sta limit
E730 3EO1 ldk a+1
E732 1814 E740$j jr BCCP
E734 E800 stora
E737 3100CF LDK SP+CCP
E73A 110301 LDK DE+ROMVEC+391
E73D 0100CF LDK BC+CCP
E740 CD6AE6 CALL RDMP
E743 87 ORA A
E744 2DF4 E73A$j jrz BCPM
E746 3E02 mvi a+2
E748 BCCP: isEntry A = 01, if cold boot
E749 F5 push AF
E74A 018000 LDK BC+BUF
E74C CDF5E6 CALL SETDMA
E74F 2100ED LDK HL+HSTBUF
E752 2200EF STC HL+DMADR
E755 AF XRA A
E756 3260EF STG A\ESC
E759 3C inc a
E75A 3232D3 sto a+ccp+04b2h
E75D 3EC3 LDK A+GC37h
E75F 320000 STO A+0
E762 320500 STG A+5
E765 2103E5 LDK HL+3105+3
E768 220100 STO HL+1
E769 2106D7 LDK HL+300D
E76E 220600 STO HL+6
E771 2163EF LDK HL+1DAY
E774 224000 STO HL+TIMEPR
E777 218AE5 LDK HL+CAUTO

; Clear console control ESC cell
; Set-up low core pointer cells
; Store jumps in low memory
; Digital Research has informed SORCIM that CP/M 2.2
; CANNOT be booted off 81.
; Manipulation of CDISK is kept here to avoid the bugs
; which would appear with its disappearance.
; Booting off 81 is accomplished with the RAM cell DSKSWP.

E77A E800 pop bc
E77B 3489E5 ld a+acmd
E77C AO
E77D 2838 E7BC$s jrz done
E77E 7E ld a+11h
E77F 87 ORA A
E780 3837 E7BC$s jrz DNE
E781 1107CF LDK DE+CCP+7
E782 0600 LDK B+0
E783 4F MOV C+A
E784 E800 ld b
E785 110000 LDK de+0
E786 1838 E7CA$s jr done1
E792 1A DB 'Z'='40h
E793 4F73626F72 DB 'Osborne Computer System'
           6E520436F 6D70757465
           7220537973 74656D
ETAA 00043630 D3 CR LF MSIZE/10"C" MSIZE mod 10 + "0"
ETAE 4B2043502F D3 "K CP/M *VER$/10+"0"*""*VER$ mod 10 + "0"
       4020322E32
ET88 41000A00 D3 C810SV "S" CR LF +0

E78C 3E02 mvi a+2
E78E 88 cmp b
E78F 2806 ^E7C?= jrz done0
ETC1 2192E7 idk hi,signon
ETC4 C0FDE8 call print
ETC7 110300 idk de+3
done0:
ETC8 2100CF idk hi,ccc
ETC9 19 add hi+de
ETCE 3A0400 id a+disk
ETD0 4F mov c+a
ETD2 E9 jmp [hl]

; Endx CSB105S.asm
E7D3  READ:  in CP/M 128 byte sector.  
         :  
         :  EXIT  A = 0, successful read operation.  
         :  A = 1, unsuccessful read operation.  
         :  Z bit = 1, successful read operation.  
         :  Z bit = 0, unsuccessful read operation.  

E7D6  CDF2E8  CALL  MVINFO  : Move information for transfer  
E7D7  AF  XRA  A  : Set flag to force a read  
E7D8  3252EF  STO  A=UNACNT  : Clear sector counter  
E7D9  CD48E8  CALL  FILL  : Fill buffer with data  
E7DA  E1  POP  HL  
E7DB  D5  POP  DE  
E7DC  018000  PUS  BC,123  : Move 128 bytes  
E7DE  E8DB  5  LDIR  
E7DF  3A22EF  LD  A,ERFLAG  
E7E0  37  CALL  OVRA  A  
E7E1  C8  RET  : If no error  
E7E2  AF  XRA  A  
E7E3  3252EF  STO  A=HSTACT  : Clear host active  
E7E4  F601  ORI  001H  : Set error flag  
E7E5  C9  

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 an unallocated  
         :  data block.  
         :  
         :  EXIT  A = 0, write was successful.  
         :  A = 1, write was unsuccessful.  
         :  Z bit = 1, write was successful.  
         :  Z bit = 0, write was unsuccessful.  

E7F3  CDF2E8  CALL  MVINFO  : Move information for transfer  
E7F4  79  MOV  A,C  : Write type in c  
E7F5  3223EF  STO  A=WRTYPE  
E7F6  FE02  CPI  WR4LE  
E7F7  2011 *E80C  JNZ  WRIT2  : If write to allocated  
E7F8  3E10  LDX  A=2048/128  
E7F9  E80C  3252EF  STO  A=UNACNT  
E7FA  2A19EF  LD  HL=SEKRX  
E7FB  2253EF  STO  HL=UNATRK  : JNATRK = SEKRX  
E7FC  3A36EF  LD  A=LOGSEC  
E7FD  3C  INC  A  
E7FE  1819 *E825  JR  WRIT3  
E7FF  2152EF  WRIT2:  LD  HL=UNACNT  
E800  7E  LD  A=INHI  
E801  E7  CPI  A  
E802  C2AE8  JZ  WRIT4  : If no unallocated records  
E803  35  DEC  INHI  : Dec unalloc record count  
E804  3455EF  LD  A=UNASEC  
E805  3C  INC  A  
E806  FE14  CPI  2910  : If not end of track  
E807  2008 *E825  JNZ  WRIT3
FILL - fill host buffer with appropriate 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
- POP y

x = host record address.
y = caller's buffer address.

```
E848 3221EF FILL: STC A,RDFLAG ;Save read flag
E84E 1180ED LDX DE MHTBUF ;Initial offset
E851 2100DD LDX HL,128 ;128 byte records
E854 3A18EF LD A,SEKSEC ;Get logical sector
E857 EB EX DE,HL
E858 OF 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
E85F 67F ANI 3FH ;Mask sector
E860 3218EF STO A,SEKSEC
E863 2A11EF LD HL,OMAADR
E866 E3 XTHL
E867 D5 PUSH DE
E868 E5 PUSH HL ;Set return parameters
E869 2150EF LDX HL MSTACT ;Host active flag
E86C 7E LD A,(HL) ;Always becomes 1
E86D 3601 STC I,(HL)
E86F 87 ORA A
E870 2814 ^E865$ JRZ FILL6 ;If host buffer inactive
E872 211CEF LDK HL MHTSEC
E875 1118EF LDX DE SEKSEC
E878 0604 LDK B SEKSEC SEKSEC+1
E87A 1A FILL4: LD A,E63
E87B 0E CMP [HL]
E87C 2005 ^E883$ JRNZ FILL5 ;If mis-match
E87F 23 INC HL
E880 13 INC DE
E880 10F8 ^E87A$ DJNZ FILL4 ;If all bytes not checked
E882 C9 RET
E883 CDAA8 FILL5: CALL FLUSH ;Flush host buffer
E886 3A18EF FILL6: LD A,SEKOSK ;Move disk and type
E889 321EFF STO A,HSTOSK
E88C 3217EF STG A,ACTOSK
E88F 2A19EF LD HL,SEKTRK
E892 2210EF STO HL,SEKTRK
E895 2215EF STG HL,ACTTRK
E898 3A18EF LD A,SEKSEC
E89B 321CEF STO A,HSTSEC
E89E 3214EF STG A,ACTSEC
E8A1 3A21EF LD A,RODFLAG
E8A4 B7 ERA A
E8A5 C0 RNZ ;If no read required
E8A6 3E00 LDX A,0 ;Read
E8A8 191C ^E866$ JR FINAL
```
: FLUSH - Write out active host buffer onto disk.

: 3151EF LDX HL+HSTWR
: 3151EF LD A, (HL)
: 3151EF JMP FINAL

: If host buffer already on disk

: Move disk and type

: Write flag

: FINAL: Preform final transfer processing.

: ENTRY A = 0 ** read disk.**
: = 3 ** write disk.**
: Calls: Rom resident routine to read/write ONE sector only.

: 3205EF JSR #0

: update sector+1

: Set retry count

: Clear retry count

: indicate one sector xfer

: process read or write

: Set possible error flag

: If no errors

: Get retry counter

: If not permanent error

: Set error flag

: endx 033135a.asm
: MVINFO: Move information necessary for transfer:

EBF2  AF
EBF3  3222EF
EBF6  3A20EF
EBF9  3218EF
EBFC  C9

XRA A
STD A*ERFLAG
LD A*TEMSEC
STD A*SEXSEC
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.

EBFD  7E
EBFE  B7
EBFF  23
E900  C6
E901  E5
E902  4F
E903  CD0C05
E906  E1
E907  18F4 ^EBFD

PRINT: LO A*{hi}
DR A
INC HL
RZ HL
PUSH HL
MOV C,A
CALL CONOUT
POP HL
JR PRINT

E909 = 0002
SAVAR: CS 2

:Disk transfer routine vector
Revisions:
- YNS 20AUG81 Initial release
- YNS 20AUG81 Expanded translation table to include ctrl/numeric
- YNS 01SEP81 Added parallel port support
- YNS 03SEP81 Invert data to/from parallel port
- YNS 03SEP81 Return character in C as well as A from all input routines
- YNS 05SEP81 Fixed "some" iee bug
- YNS 05SEP81 Reorder iee drivers to send until all are unlisted commands after each char xfr.
- YNS 05SEP81 Add auto horiz. scroll
- YNS 20SEP81 Reassignment in dispatch table
- RWC 14OCT81 Redefinition of ioboe assignments
- RWC 29SEP81 Printer protocols added to serial list device
- RWC 20OCT81 Function key decoding added
- RWC 24OCT81 Horizontal scroll toggle added
- RWC 24OCT81 Horizontal scroll modified

; the following routines will use the I/O BYTE to transfer
; control to the appropriate device driver
; return console status

;---------------------
CNSTA: proc
E909 8179E9
E908 1808 ^E918$ ldk hi+ptr+cstat ; status table
E90E jr godisoch ; call appropriate rtn
;---------------------
; read input character from device

;---------------------
CNIN: proc
E910 8179E9
E910 1803 ^E918$ ldk hi+ptr+cinp ; table of input rts
E913 jr godisoch
;---------------------
; put output character to device
; c contains output character

;---------------------
CNOUT: proc
E915 8179E9
E915 1801 mvi b,1 ; number of shifts required to align
E918 mvi b,1
E91A C33DE9
E91A disp ch
;---------------------
; list device character output

;---------------------
LIST: proc
E91D 0604
E91F 8179E9
E922 C33DE9
E922 disp ch
;---------------------
; output to punch

PNCH: proc
LIST\$1: proc

b+4
m
h+4t
L1: align LIST fields of output

\textbf{LIST status.**}

Return the ready status for the list device.

EXIT if the list device is not ready.

Accept another list device if ready to accept
another device.

The list device returns 0 upon a list device ready
status of CRT. ALWAYS ready

\textbf{LIST status.**}

The output list device status is determined
depending upon the output fields

\textbf{List field.**}

Status of serial printer

\textbf{List status.**}

Status of the output port (lines ready)
E93D  dispn: proc
  ; on entry here reg 3 contains the left shift count
  ; required to align the iobyte field to bit 1 position
  ; reg HL contains address of select table
  ; A special stack is used to avoid overflows (e.g. when
  ; called by WordStar)
E93D  2290E8  stoh hl,hl+1 ;Save hl
E940  210000  ldk hl,hl+1 ;Old stack to hl
E943  39      add hl,sp ;Old stack to hl
E944  379E9   ldk spr,hl+2
E947  55      push hl ;Save old stack
E943  2A9DE6  ldl hl,hl+1 ;Restore hl
E948  3A0300  ldl iobyte
E94E  17      ral
E94F  10FD ^E94E$  djnz dshch
E951  E065  anl b
E953  5F      mov e,a
E954  1600  ldk d,r
E956  19      dac de
E957  3E      mov e,m
E958  23      ldx hl
E959  66      mov h,m
E95A  68      mov l,e
E95B  1160E9  ldk dec,sp
E95E  D5      push de
E95F  E9      pchl
E960  FDE1  pop iy ;push return address
E962  FDF9  mov sp,sp+1
E964  C9      ret
E965  =0014  ds 10*2
E979  =0000  disstk: ds 0

; addresses of routines
E979  A1E9  dw const ;keyboard status
E975  6EA  dw sistat ;serial port input status
E97D  DAEB  dw pintat ;parallel input status
E97F  FBEA  dw leinstat ;status of input device on IEEE port
E981  ptr+rdr: ;
E981  DBF9  dw ptr+cin
E981  32BE  dw keyinp
E983  74EA  dw spinp
E985  F0E8  dw parinp
E987  2FE8  dw lainp
E989  ptr+conch: ;
E989  F5E9  dw crtout ;output character to CRT
E98B  87EA  dw spout ;serial port output
E98D  04EC  dw parout ;parallel output
E98F  33EB  dw leout ;IEEE port output
E991  ptr+list:
E991  F3E9   dw  crstat  ; serial output status
E993  62EA   dw  sopstat ; parallel output status
E995  C1E9   dw  iopstat ; lromo output status
E997  F7E9   dw  ioeostat
E999  ptr+list:
E999  55E9   dw  crotout
E99B  98EA   dw  prout
E99D  D4E9   dw  parout
E99F  33E9   dw  leout
**CONSOL STATUS**

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**

```asm
E9A1 const: proc
  check if any translated keys are pending
  E9A1 3A2E9 if a$count
  E9A4 67 ora a
  E9A5 2C06 ^E9AD$ jrz 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
E9AD const5:
E9AF C9 RET
```
```assembly
; 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 basv10 equ 80h ;lowest value of translatable keys
E9A0 = 0002 xitkey ds 2

E9B2 KEYINP:
    proc
    ; if there are no xlated keys waiting then
    ; call the keyboard driver in rom
    E9B2 C00F9A callinha
    E9B5 21C2E9 ldk h1, count
    E9B8 7E ld a,h1
    E9B9 B7 ora a
    E9BA 2B89 *E9C5S jrz kin10
    E9BC DD2AB0E9 $ ld lx, xitkey
    E9C0 DD7E db 0D0h, 7Eh ;simulate LD A,(IX+COUNT)
    E9C2 0D count: db 00h
    E9C3 34 inc [h1]
    E9C4 C9 ret

E9C5 kin10:
    ldk e, low(conin)
    E9C7 CD68E6 call romcd1

    ; When console has returned this code will check
    ; for function key and perform some translation
    E9CA F800 cpi 80h ; function keys have value
    E9CC 08 rc 18ch-50h
    E9CD F80E cpi 50h
    E9CF 00 rnc indo a to make pointer
    E9D0 CB27 $ sla a
    E9D2 5F mov e,a
    E9D3 1500 ldk d,e
    E9D5 DD2168E5 $ ldk lx, xltbl
    E9D9 DD19 $ add lx, +de
    E9DB DD56E00 $ ld lx, [lx+0]
    E9DE DD6601 $ ld h, [lx+1]
    E9E1 DD5E02 $ ld e, [lx+2]
    E9E4 DD5603 $ ld d, [lx+3]
    E9E7 DD4B1E9 $ stx d, xitkey
    E9EB ED52 $ sbc h, +de
    E9ED 7D mov a,l
    E9EE 32C2E9 stx a, count
    E9F1 18BF ^E9B2$ jr keyino

    ;
```
;oe
;routine: CRSTAT
;returns status of crt.
;crt is always ready
;EOSTAT returns status of iee
;iee always appears to be ready

E9F3 ilostat:
E9F3 CRSTAT: proc
E9F5 F6FF or i OFFh
E9F5 C9 ret

;oe
;routinenm: CRTOUT
;inputs: C: output character
= 0008 EF+ESC1 = 8 ;escape flag bit definitions
= 0001 EF+GR = 1

E9F6 CRTOUT: proc
E9F6 3A60EF ldp esch
E9F9 ES09 and ef+esc+ef+gr
; push af
E9F8 200D *EAOA$ jrnz crt10
E9FD 79 mov a+c
E9FE FE14 cpi 14h
EA00 2008 *EAOA$ jrnz crt10
EA02 3A68E5 ld a+ahsenb
EA05 2F cmn
EA06 3268E5 sto a+ahsenb
; pop af
EA09 C9 ret
EA0A CRT10:
;call outch
;pop af
;rnz
;mvi a+c
;cmp c
;cz ahscrl
;ret

EA0A outch:
EA0A lEOC
EA0C C368E6
;sbtr: outch - calls rom cout routine
;jmp romed1

;sbtr: ahscrl - does auto horizontal scroll if required.
EA0F ahscrl:
;proc
EA0F 3A68E5 ld a+ahsenb
EA12 27 or a
EA13 C8 rz
EA14 2A5AEF ld hl+curs ;get cursor
EA17 29 add hl+hl
EA18 3E64 ;check for cursor in home window
ldk a*100
EA1A  8D  cmp  l
EA1B  3809  ^EA24  jrc  rhi
EA1C  3A61EF  ld a,plaad
EA1D  D6EA  sub vflc
EA1E  C8  rz
EA1F  AF  xra a
EA20  1818  ^EA3E  jr  scr1

EA22  rhc:
EA23  3A61EF  ld a,plaad
EA24  D6EA  sub vflc
EA25  C6A4  add a,103
EA26  8D  cmp  l
EA27  DA3BEA  jc  :30

EA28  :30:
EA29  D65A  sub 90
EA2A  B0  cmp  l
EA2B  D8  rc
EA2C  7D  mov a,1
EA2D  D60A  sub 10
EA2E  C8  rz
EA2F  1803  ^EA3E  jr  scr1

EA30  :30:
EA31  70  mov a,1
EA32  D664  sub 100
EA33  1F  scrl:
EA34  IF  rar
EA35  C620  add a,1
EA36  2161EA  lk hl_ascsq+3
EA37  77  sto a[hl]
EA38  3A62EF  ld a,plaad
EA39  E61F  anp 1fh
EA3A  C620  add a,1
EA3B  28  dec hl
EA3C  77  sto a[hl]
EA3D  23  dec hl
EA3E  23  dec hl
EA3F  0604  lk bx
EA40  150:
EA41  C5  push bc
EA42  E5  push hl
EA43  4E  li c[hl]
EA44  C20AFA  call outch
EA45  E1  pop hl
EA46  C1  pop bc
EA47  23  inc hl
EA48  10F5  ^EA52  djnz :50
EA49  C9  ret

EA50  escsq  db  esc
EA51  53  db "*"  
EA52  00  db 0
EA53  00  db 0

:escsq = vert. coords
:point to start of esc seq

Copyright © 1982 Osborne Computer Corporation
; routine: SOSTAT
; gets status of output device attached to serial port
EA62 SOSTAT: proc
EA62 C3D9EA call acistat ; get 6850 status
EA65 E602 ani si-trdy
EA67 C5 rz true
EA68 F6FF ori ret

; routine: SISTAT
; gets status of input device attached to serial port
EA68 SISTAT: proc
EA68 C3D9EA call acistat
EA6C E601 ani si-rrdy rz
EA70 C3 ret true
EA71 F6FF ori 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 C3D6EA call sistat
EA77 28FB *EA74s jr2 SPINP ; if not ready
EA79 C1D1EC call sw2rom
EA7C A4 xra a
EA7D 32D8EA stor a,serflg ; reset RDRF bit
EA80 3A012A id 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 C3D2EA call sOSTat
EA8B 28FB *EA88s jr2 11 ; if not ready
EA8D C1 pop bc
EA8E C1D1EC call sw2rom
EA91 79 mov a,c
EA92 32D12A stor a,h,sxmt ; send chr
EA95 C32CEC jmp sw2ram

; routine does serial i/o and does printer protocols
```
; routine: ACICTL
; outputs character in c to the AC1A CTL port.

ACICTL: proc

sw2rom
mov a+c
sta h+scrl
jmp sw2ram
```

; routine: acisr
; returns usart status in A
; The ring in signal is read from the oia 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 (serfig) keeps
track of the correct status. Every time the status register
is read, serfig is updated. If the RDRF bit is ever set in the
status register, that bit remains high in serfig until the
Receive Data Register is read. The RDRF bit from serfig is
ORed in with the status register to give the correct status.

+ 2C01 riport equ hvio+1
  = 0040 rbit equ 040h
  = 0020 ormask equ 020h
  = 0001 RDRF= equ 001h
EA09 00 serfig db 00 ; masks out all bits except RDRF
                                 ; flag to keep track of RDRF bit
                                 ; (to correct for hardware timing problem)

acistat: proc
EA09 CD1EEC call sw2rom
EA0C 3A012C lda riport
EA0F 0F rrc
EA12 6E20 ani ormask
EA15 4F mov c+a
EA18 3A002A lda h.ssts
EA1B 6E0F ani Offh-ormask
EA1E 61 ora c
EA21 4F mov c+a
EA24 3A08EA lda serfig
EA27 6E01 ani RDRF=
EA2A 61 ora c
EA2E 3208EA sta serfig
EA33 632EEC jmp sw2ram

(t)
IEEE drivers:

The routines IEIINST, 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 call IE=ADDRS. No facility is provided at present to allow a transient program to specify the device address. Thus the device attached must be addressable as 0 (zero).

The function IEIINST 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 IEIINST many times before calling IEINP to read a character, and IEIINST has to read the char to determine the status, the character read has to be buffered till call to IEIINST is made. IEIINST reads the device only when the buffer is empty. As zeros are used to indicate that the blf is empty, a null character cannot be read from the IEEE device.

IEEE control codes

```
= 0040 iet=talk equ 40h
= 005F iet=utilk equ 5Fh
= 0020 iet=lstn equ 20h
= 003F iet=unlstn equ 3Fh
```

```
EAF6 00 iet=chars db 0
EAF7 00 iet=chars db 0
```

; routine; IEIINST
; gets status of the input device attached to ieee port
; if a char is present in iet=chars 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

```
EAF8 3AF7EA lda iet=char
EAF8 B7 ora a
EAF8 CA02E8 jsr iet=10
EAF8 F6FF ori 0Fh
EAF8 C9 ret

EAF8 iet=10: if char present then
EAF8 0Fh ret
EAF8 :endif
EAF8 iet=make talker
EAF8 iet=make untalk
```

```
EB02 3AF6EA lda iet=chars
EB02 C640 adi iet=talk
EB02 4F mov a
EB08 1E48 ldk e=low(ieb5c)
```
EB0A CD68E6     call romcd1 ; output interface message
EB0D B7         ora a
EB0E 20F2 ^EB02$ jrz 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 romcd1
EB15 CB7D       bit 7:1
EB17 2801 ^EBA1$ jrz iei30 ; if error then
EB1A 4F         xra a
EB1A iei30:     stx i+char
EB1A 32F7EA     stor the char

; make untalk
EB1D iei40:     mvi c rie=utilk
EB1E 0E5F       ldk e+low(ieb5c)
EB21 CD68E6     call romcd1
EB24 B7         ora a
EB25 20F6 ^EB1D$ jrz iei40

; return with status of the char
EB27 3AF7EA     lda i+char
EB2A B7         ora a
EB2B C8         rz
EB2C F6FF       ori 0fh
EB2E C9         ret

; routine: IEINP
; Reads a character from IEEE port

EB2F IEINP: proc
EB2F C3F8EA     call ieiinst
EB32 28F6 ^EB2F$ jrz iei
EB34 21F7EA     ldk h.i+char
EB37 7E         id a.[hl]
EB38 3600       sto 0.[hl]
EB3A C9         ret

; routine: IEDOUT
; Outputs the character in reg C to IEEE port
; Uses ROM resident primitives.

EB3B IEDOUT: proc
EB3B C5         push bc
EB3C ieo05:    ; save the char
EB3C 3AF6EA     lda i=aadr
EB3F C620       adi i+1stn
EB41 4F         mov c=a
EB42 1E45       ldk e+low(ieb5c)
EB44 CD68E6     call romcd1
EB47 B7         ora a
EB48 20F2 ^EB3C$ jrz ieo05 ; try again if error
EB4A 0600
EB4C C1
EB4D ie022:
EB4D C5
EB4E 1E4E
EB50 CD68E6
EB53 C1
EB54 B7
EB55 20F6 "EB4D$
EB57
EB57 0E3F
EB59 1E43
EB5B CD68E6
EB5E B7
EB5F 23F6 "EB57$
EB61 C9

: secondary address output may be added here

: i do not send eoi
: i save char again in case of retry
: i try again if error
: i make unlisten

mvi b+0
pop b
push b
ldk e+low("ieb6c")
call romcd1
pop b
ora a
jnz ie022
mvi c+ie+ulstn
ldk e+low("ieb5c")
call romcd1
ora a
jnz ie040
ret
The Parallel port is actually the IEEE port driven with the centronix protocol. The bit assignments of the PIA and P13 are as follows:

- P1A0-7 = data bus
- P1B0 = 0, data bus is output, 1, data bus is input
- P1A1 = set to 1
- P1A2 = set to 0
- P1B3 = 0, output = 1 input
- P1B4 = not used
- P1B5 = output strobe, Active = 1
- P1B6 = 0, printer busy, 1, printer is ready
- P1B7 = 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 data equ h.1ee+0
- 2900: PA dir equ pa.data
- 2901: PA ctl equ h.1ee+1
- 2902: PB data equ h.1ee+2
- 2902: PB dir equ pb.data
- 2903: PB ctl equ h.1ee+3

**Port Ctrl Register Constants**

- 002A: PA cd = 00101010b: to address port a direction
- 002B: PA cd = 00111110b: to address port a data and set port a in input program handshake mode.
- 0000: PB cd = 00000000b: to address port b direction
- 0004: PB cd = 00001000b: to address port b data

**Direction Register Constants**

- 00FF: PA d = 000h: port a output mode
- 0000: PA d = 000h: port a input mode
- 00BF: PB d = 00fh: port b direction
- 0002: PB d = 00001010b: port b data for output
- 0000: PB d = 00001011b: port b data for input
- 0040: PA rdy = 01000000b: output rdy bit in pib
- 0080: PB rdy = 10000000b: input rdy bit in pia ctl reg
- 0020: Strb = 00100000b: strobe bit in port b

**Port Modes**

- 0000: PA undef equ 0
- 0001: PA out equ 1
- 0002: PA in equ 2

EE62 00: PA mode db PA undef

**Initialization**

- CB20P: initializes the port to a parallel output

**Port**

EE63 3A62E3 I7A PA mode
EB66 FE01  cpi  pp.out  ;return when in output mode
EB68 C8   ;rz
EB69 3E2A  ;set port a to output on all lines
EB6B 320129  ;sta pa.ctl
EB6D 3EFF  ;lk ap.a cdr
EB70 320029  ;sta pa.dir
EB73 3E2E  ;lk ap.a cdt
EB75 320129  ;sta pa.ctl
EB77 3E00  ;lk ap.b cdr
EB79 320329  ;sta pb.ctl
EB7D 3E0F  ;lk ap.b dr
EB7F 320229  ;sta pb.dir
EB82 3E04  ;lk ap.b cdt
EB84 320329  ;sta pb.dir
EB88 3E02  ;lk ap.b dto
EB8A 320229  ;sta pb.data
EB8C 3E01  ;lk app.out
EB8E 3262EB  ;sta pp.mode
EB91 C9  ;ret

;==
; sbri:  
; cv2ip:  
EB92 3A62ES  ;cv2ip:  proc  pp.mode
EB95 FE02  ;cpi  pp.in
EB97 C8   ;rz
EB98 3E2A  ;set port a to input on all lines
EB9A 320129  ;sta pa.ctl
EB9C 3E00  ;lk ap.a dir
EB9E 320029  ;sta pa.dir
EBA2 3E2E  ;lk ap.a cdt
EBA4 320129  ;sta pa.ctl
EBA7 3E00  ;lk ap.b cdr
EBAC 3E8F  ;lk ap.b dr
EBAE 320229  ;sta pb.dir
EBB1 3E04  ;lk ap.b cdt
EBB3 320329  ;sta pb.ctl
EBB7 3E0B  ;lk ap.b dto
EBB9 320229  ;sta pb.data
EBB8 3E02  ;lk app.in
EBBD 3262EB  ;sta pp.mode
EBE0 C9  ;ret

;==
; routine:  POSTAT
; gets status of the parallel (centronix) printer
; attached to the ieee port
SORCIM 808x Assembler ver 3.5E <i>/557= 59:92 Page 38
Osborne CP/M 2.2 EBOS
Copyright 1982 Osborne Computer Corporation

EBC1    POSTAT: proc
        EBC1 CD1E7C  call    sw2rom
        EBC4 CD63EB  call    cv2op
        EBC7 3A029A  lda    p0+dta
        EBCA E640  anl    po+ordy
        EBCB 2802 ^EBD0$  jr3z  pos10
        EBCC F6FF  ori    true
        EBD0  pos10:
        EBD0 C32ECC  jmp    sw2ram

;;
;; routine:    PISTAT
;; gets status of the input device attached to the
;; parallel port
;;
EBD3    00  piactl db    0

EBD5    PISTAT: proc
        EBD5 CD1E7C  call    sw2rom
        EBD7 CD92EB  call    cv2ip
        EBD9 3A03EB  lda    piactl
        EBD9 E680  anl    po+irdy
        EBD9 280A ^EBE9$  jr3nz  pis20
        EAE1 3A0129  lda    pa+actl
        EBE4 3203EB  sta    piactl
        EBE4 E680  anl    po+irdy
        EBE7 2A02 ^EBE9$  jr3z  pis30
        EBE8 pis20:
        EBE8 F6FF  ori    true
        EBE9 C32ECC  jmp    sw2ram

;;
;; routine:    PARINP
;; inputs a character from parallel port.

EBF0    PARINP: proc
        EBF0 CD94EB  call    pisstat
        EBF3 28F8 ^EBF0$  jr3z  parin0
        EBF5 CD1E7C  call    sw2rom
        EBF8 AF  xra    a
        EBF9 3203EB  sta    piactl
        EBF9 3A0029  lda    pa+dta
        EBFF 2F  cma
        ECO0 4F  mov    c+a
        ECO1 C32ECC  jmp    sw2ram

;;
;; routine:    PAROUT
;; outputs the character in c to the IEEE port treating the
;; port as a parallel port.

EC04    PAROUT: proc
        EC04 CD1E7C  call    postat
        EC07 28F8 ^EC04$  jr3z  parout
EC09 CD1EEC    CALL swzrom
EC0C 79        MOV a+c
EC0E 2F        CTA
EC10 320029    STA p3+dta
EC11 3E22      LK a+p0+sto+strb
EC13 320229    STA p5+dta
EC16 3502      LK a+p5+sto
EC18 320229    STA p5+dta
EC1A C32CEC    JMP swzram

:invert data
:set strobe
:clear strobe
```assembly
; ***
; sbtr: SW2RAM
; switches to ram
; saves all registers
EC15
SW2RAM: proc
EC15
di
EC15 +F3
DB
OF3h
EC1F
F5
push
af
EC20
enarom
EC20 +
di
EC20 +F3
DB
OF3h
EC21 +0300
OUT
C
EC23 +3E00
LDK
A+D
EC25 +3208EF
STG
A+RMRAW
EC28 +
ej
EC28 +F8
dB
OF8h
EC29
j
EC29 C9
ret
; ***
; sbtr: SW2RAM
; switches to ram
; preserves all registers
EC2C
SW2RAM: proc
EC2C
di
EC2C +F3
DB
OF3h
EC20
F5
push
af
EC2E
lisrom
EC2E +
di
EC2E +F3
DB
OF3h
EC2F +0D01
OUT
1
EC31 +3E01
LDK
A+1
EC33 +3208EF
STG
A+RMRAW
EC36 +
ej
EC36 +F3
DB
OF8h
EC37
j
EC38
ec
EC38 +F8
dB
OF3h
EC39
C9
ret

= EC3A
xxx:
*
- = 0000
if
xxx>MRAMEX
CCDE TCO LARGE.
endif

EC3A = EC70
EC70 00CF
DRG
EDOS+770H
DW
CCP
; This is location ED070H in 60K system
; This location needed by RCM for
; last sector read bombs out part
tof BMRAM in 60K system

DECRC8\2\54
```
: Used to assemble RCM resident and BIOS

: This location is assigned in BIOS
: and filled in by loader in ROM

: Host disk xfer buffer and...
: Format track template holding buffer

: Directory Buffer
: DISTRUF = HSTBUF+256

: Disk operation temps and control
: DMAADR = Address for read/write Disk
: IEBIOS, users DMA

: Note order of xxxSEC,xxxTRK,xxx3SK must be maintained
: along with length (1x2x1).

: Set for seek-restore command in RCM
: Depends on disk type: Siemens = 3h, MPI = 0h

: Last sector requested

: Selected disk drive (0,1)

: Used in bios only

: Read flag

: Error reporting

: Write operation type

: ALVIS = CSVIS
31CS blocking/detooling flags

EF53 = 0001
MSTACT: DS 1 nHost active flag
EF51 = 0001
MSTWR: DS 1 nHost written flag
EF52 = 0001
UNACNT: DS 1 nUnallocated record count
EF53 = 0002
UNATRK: DS 2 nTrack
EF54 = 0001
UNASEG: DS 1 nSector
EF56 = 0001
LOSGSEC: DS 1 nLogical sector

EF57 = 0002
LOADDR: DS 2
EF59 = 0001
KEYLOC: DS 1 nZero if locked keyboard
EF5A = 0002
CURRS: DS 2 nCurrent cursor position

EF5C = 0001
TKEY: DS 1 nKeyPressing key
EF5D = 0001
MKCHT: DS 1 nDebounce key
EF5E = 0001
LKEY: DS 1 nLast valid keystroke
EF5F = 0001
CKEY: DS 1 nLast control key
EF60 = 0001
ESCM: DS 1 nESC holding flag

PIAAD and PIACO must be kept sequential, PIAAD first.

Dependancy in VC+HOME of T3KEY.asm

EF61 = 0001
PIAAD: DS 1 nHolds last PIA-A data
EF62 = 0001
PIACO: DS 1 nHolds last PIA-C data

EF63 = 0003
DAYS: DS 2
= EF64
MONTH = 0001
EF65
YR = 0002

Calendar month, day, year

EF63 = 0006
HOURS: DS 6
= EF67
Min. = HOURS+1
= EF68
Sec. = HOURS+2
= EF69
SUS: = HOURS+3

Wall clock time cells and disk active

See UPTIM in MMKEY.asm

EF6A = 0006
ACTIVE: = HOURS+4 iICF by FDISK, used by UPTIM
EF6B
ULCUM: = HOURS+5 iICF Slack timer cell

EF6C = 0001
LLIMIT: DS 1 nMax columns in a logical line

MSG = *LLIMIT*H.

Disk drive current positions

EF6D = 0002
LOSSL: DS 7 nLast selected drive
= EF6E
LOSTK = LOSSL+1 nLast track used for non-selected drive

EF6F = 0002
IESTK: DS 2 nSave current stk ptr

Interrupt stack

EF71 = 0028
DS 2092
EF99 = 0000
IESTK: DS 6

Stack entry
SDRICM 803x Assembler ver 3.5E c:\5972 59-92 Page 43
Debug Monitor RAM Storage.

EF99 = 0028  DS  2002
EF01 = 0009  DS  0
EF01 = 0001  DS  1  Last command byte written to ACIA
EF01 = 0004  DS  4  1794 register save area
EF06 = 0001  DS  1  Keyboard debounce-delay cell

Since CP/M CANNOT boot off 91, this cell is used
to invert the names of the 2 drives:
* #1 all normal, A=31, B=31
* #1 all inverted, A=31, B=31
EF07 = 0001  DS  1

; 780 Alternate Register Set
EF08 .. EF0C:
ALIGN 10h
EF08 += 0008  DS  (9*(10h)-1)/10h=(10h)=-

EF0C = 0002  DS  2  ID*
EF0A = 0002  DS  2  IC*
EF0E = 0002  DS  2  IF*
EF10 = 0002  DS  2  IH*
EF14 = 0002  DS  2  IX*
EF16 = 0002  DS  2  IY*
EF18 = 0002  DS  2  Interrupt page register

; 8080 Register Save Area.
EF1C .. EF2E:
ALIGN 10h
EF1C += 0002  DS  (9*(10h)-1)/10h=(10h)=-

EF1C = 0001  DS  1  IE Register save location
EF1E = 0001  DS  1  ID Register save location
EF20 = 0001  DS  1  IC Register save location
EF24 = 0001  DS  1  IA Register save location
EF28 = 0001  DS  1  IA Register save location
EF2C = 0001  DS  1  I flags save location
EF30 = 0001  DS  1  IL Register save location
EF34 = 0001  DS  1  IH Register save location
EF38 = 0001  DS  1  SPOCM COUNTER save location
EF3C = 0002  DS  2  USER STACK pointer save location
EF40 = 0002  DS  2  Last breakpoint address
EF42 = 0001  DS  1  Contents of bkp
EF44 = 0001  DS  1  Last vertical offset taken from CCUT

; Interrupt Jump Vector is between EFF8, EFFF.
; @0x80 RAM

no ERRORS. 517 labels, 646th byte not used.  Program LWA = EFF0n.
<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMTJ</td>
<td>0539</td>
</tr>
<tr>
<td>FN1L</td>
<td>0105</td>
</tr>
<tr>
<td>PPS13</td>
<td>0010</td>
</tr>
<tr>
<td>KSAVE</td>
<td>0439</td>
</tr>
<tr>
<td>WAVR</td>
<td>0300</td>
</tr>
<tr>
<td>GDISP</td>
<td>0918</td>
</tr>
<tr>
<td>GCEM</td>
<td>0694</td>
</tr>
<tr>
<td>ADDX</td>
<td>2900</td>
</tr>
<tr>
<td>ASCTR</td>
<td>2400</td>
</tr>
<tr>
<td>ASIO</td>
<td>0200</td>
</tr>
<tr>
<td>ASRC</td>
<td>2040</td>
</tr>
<tr>
<td>ASSTS</td>
<td>2400</td>
</tr>
<tr>
<td>ASAMT</td>
<td>2400</td>
</tr>
<tr>
<td>ASVR</td>
<td>2200</td>
</tr>
<tr>
<td>HKCR</td>
<td>0516</td>
</tr>
<tr>
<td>HSLAX</td>
<td>0466</td>
</tr>
<tr>
<td>HME</td>
<td>069F</td>
</tr>
<tr>
<td>HOURS</td>
<td>0667</td>
</tr>
<tr>
<td>SAVE</td>
<td>04E7</td>
</tr>
<tr>
<td>STACT</td>
<td>0520</td>
</tr>
<tr>
<td>STOUF</td>
<td>0630</td>
</tr>
<tr>
<td>HDSK</td>
<td>011F</td>
</tr>
<tr>
<td>HTSNC</td>
<td>011F</td>
</tr>
<tr>
<td>HSTSLT</td>
<td>0102</td>
</tr>
<tr>
<td>HSTTRK</td>
<td>010D</td>
</tr>
<tr>
<td>HSTWRT</td>
<td>0511</td>
</tr>
<tr>
<td>JDAY</td>
<td>0637</td>
</tr>
<tr>
<td>JREADS</td>
<td>010F</td>
</tr>
<tr>
<td>IEBC</td>
<td>0537</td>
</tr>
<tr>
<td>IEBC</td>
<td>0542</td>
</tr>
<tr>
<td>IEBC</td>
<td>0547</td>
</tr>
<tr>
<td>IEBC</td>
<td>054B</td>
</tr>
<tr>
<td>IEBC</td>
<td>054C</td>
</tr>
<tr>
<td>IEBC</td>
<td>054D</td>
</tr>
<tr>
<td>IEBC</td>
<td>0551</td>
</tr>
<tr>
<td>IEBC</td>
<td>0554</td>
</tr>
<tr>
<td>IEBC</td>
<td>0562</td>
</tr>
<tr>
<td>IECMAR</td>
<td>010F</td>
</tr>
<tr>
<td>EIN10</td>
<td>0302</td>
</tr>
<tr>
<td>IEI20</td>
<td>0310</td>
</tr>
<tr>
<td>IEL30</td>
<td>011B</td>
</tr>
<tr>
<td>IEI40</td>
<td>0310</td>
</tr>
<tr>
<td>IEINP</td>
<td>021F</td>
</tr>
<tr>
<td>IEINST</td>
<td>0108</td>
</tr>
<tr>
<td>IELSTN</td>
<td>0200</td>
</tr>
<tr>
<td>IEI5S</td>
<td>0358</td>
</tr>
<tr>
<td>IEI20</td>
<td>0444</td>
</tr>
<tr>
<td>IEI22</td>
<td>0340</td>
</tr>
<tr>
<td>IEI40</td>
<td>0357</td>
</tr>
<tr>
<td>IEOSTA</td>
<td>0935</td>
</tr>
<tr>
<td>IEIUT</td>
<td>0330</td>
</tr>
<tr>
<td>IEIK</td>
<td>06F6</td>
</tr>
<tr>
<td>IETALK</td>
<td>0040</td>
</tr>
<tr>
<td>IEUST</td>
<td>033F</td>
</tr>
<tr>
<td>IENGRL</td>
<td>035F</td>
</tr>
<tr>
<td>IMONTH</td>
<td>0664</td>
</tr>
<tr>
<td>ISTRT</td>
<td>0576</td>
</tr>
<tr>
<td>IIVSAX</td>
<td>0FDC</td>
</tr>
<tr>
<td>I1XSAX</td>
<td>0423</td>
</tr>
<tr>
<td>I1XSAX</td>
<td>0425</td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>PISTAT</td>
<td>E8D4</td>
</tr>
<tr>
<td>POS10</td>
<td>EBD0</td>
</tr>
<tr>
<td>POSTAT</td>
<td>E8C1</td>
</tr>
<tr>
<td>PPJRD</td>
<td>0080</td>
</tr>
<tr>
<td>PPQRD</td>
<td>0040</td>
</tr>
<tr>
<td>PPJUND</td>
<td>0000</td>
</tr>
<tr>
<td>PRINTER</td>
<td>E567</td>
</tr>
<tr>
<td>PRT20</td>
<td>EAC4</td>
</tr>
<tr>
<td>PSAVE</td>
<td>EFE8</td>
</tr>
<tr>
<td>PTOCIN</td>
<td>E981</td>
</tr>
<tr>
<td>PTOCST</td>
<td>E979</td>
</tr>
<tr>
<td>PTLST</td>
<td>E991</td>
</tr>
<tr>
<td>PTPRDA</td>
<td>E981</td>
</tr>
<tr>
<td>R179X</td>
<td>EFC2</td>
</tr>
<tr>
<td>RAGS</td>
<td>EFD0</td>
</tr>
<tr>
<td>RDI</td>
<td>E920</td>
</tr>
<tr>
<td>REF</td>
<td>EE00</td>
</tr>
<tr>
<td>READER</td>
<td>E515</td>
</tr>
<tr>
<td>RMC</td>
<td>EA26</td>
</tr>
<tr>
<td>RBIT04</td>
<td>0040</td>
</tr>
<tr>
<td>RPORT</td>
<td>0001</td>
</tr>
<tr>
<td>ROMC10</td>
<td>E688</td>
</tr>
<tr>
<td>ROMJMP</td>
<td>E66A</td>
</tr>
<tr>
<td>ROMRAM</td>
<td>EFO8</td>
</tr>
<tr>
<td>ROMRAF</td>
<td>E64F</td>
</tr>
<tr>
<td>ROMSTK</td>
<td>EFC1</td>
</tr>
<tr>
<td>ROMSEC</td>
<td>0100</td>
</tr>
<tr>
<td>RDX</td>
<td>ES27</td>
</tr>
<tr>
<td>RRI</td>
<td>ES33</td>
</tr>
<tr>
<td>RSELK</td>
<td>ES18</td>
</tr>
<tr>
<td>TRC</td>
<td>EFC4</td>
</tr>
<tr>
<td>TRY</td>
<td>EFO5</td>
</tr>
<tr>
<td>RDK</td>
<td>ES2A</td>
</tr>
<tr>
<td>SIW</td>
<td>002E</td>
</tr>
<tr>
<td>SAVADR</td>
<td>E9D9</td>
</tr>
<tr>
<td>SAVSEC</td>
<td>EF14</td>
</tr>
<tr>
<td>SAVTRK</td>
<td>EF15</td>
</tr>
<tr>
<td>SSAUD</td>
<td>ES3C</td>
</tr>
<tr>
<td>SCRL</td>
<td>EA3E</td>
</tr>
<tr>
<td>XLTS</td>
<td>E600</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>XOFF</td>
<td>0013</td>
</tr>
<tr>
<td>XON</td>
<td>0011</td>
</tr>
<tr>
<td>XUNKDF</td>
<td>EAB7</td>
</tr>
<tr>
<td>XXX</td>
<td>EC3A</td>
</tr>
</tbody>
</table>
ROM/BIOS 1.4
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>DDDD</td>
<td>DRG 0</td>
</tr>
<tr>
<td>0007</td>
<td>SCROLL</td>
<td>'5Y-40h  Ring the bell</td>
</tr>
<tr>
<td>0008</td>
<td>MCRS</td>
<td>'5Y-40h  Move cursor up</td>
</tr>
<tr>
<td>0009</td>
<td>MRG</td>
<td>'LT-40h  Move cursor right</td>
</tr>
<tr>
<td>001A</td>
<td>VCRS</td>
<td>'ZY-40h  IClear and home cursor</td>
</tr>
<tr>
<td>001E</td>
<td>VHOME</td>
<td>'AY-40h  IHome Cursor</td>
</tr>
<tr>
<td>0023</td>
<td>VLOCK</td>
<td>'8k     ILock Keyboard</td>
</tr>
<tr>
<td>0022</td>
<td>VUNLK</td>
<td>'7k     IUnlock Keyboard</td>
</tr>
<tr>
<td>0030</td>
<td>VCAD</td>
<td>'k     ICursor Addressing</td>
</tr>
<tr>
<td>0053</td>
<td>VSAD</td>
<td>'Zk     IScreen Addressing</td>
</tr>
<tr>
<td>0051</td>
<td>VINC</td>
<td>'Qk     IInsert Char</td>
</tr>
<tr>
<td>0057</td>
<td>VDEL</td>
<td>'Wk     IDelete char</td>
</tr>
<tr>
<td>0045</td>
<td>VINL</td>
<td>'Ek     IInsert line</td>
</tr>
<tr>
<td>0052</td>
<td>VDELL</td>
<td>'Rk     IDelete line</td>
</tr>
<tr>
<td>0054</td>
<td>VCELD</td>
<td>'Tk     IClear to end of line</td>
</tr>
<tr>
<td>0029</td>
<td>VSTH</td>
<td>'Yk     IStart half intensity</td>
</tr>
<tr>
<td>002B</td>
<td>VSTL</td>
<td>'yk     ITend</td>
</tr>
<tr>
<td>006C</td>
<td>VSUL</td>
<td>'Gk     IStart underline</td>
</tr>
<tr>
<td>0067</td>
<td>VSSH</td>
<td>'Gk     IStart graphics</td>
</tr>
<tr>
<td>0047</td>
<td>VESG</td>
<td>'Gk     IEnd</td>
</tr>
<tr>
<td>0081</td>
<td>ROWOM</td>
<td>'081H   IStart</td>
</tr>
<tr>
<td>0057</td>
<td>SLMST</td>
<td>'0.10_101_11b Master reset</td>
</tr>
<tr>
<td>0056</td>
<td>SLS16</td>
<td>'0.10_101_01b Select 16 clock, xmit/rec</td>
</tr>
<tr>
<td>0001</td>
<td>SLRDY</td>
<td>'01     IReceiver ready</td>
</tr>
<tr>
<td>0002</td>
<td>SLCRDY</td>
<td>'02     ITransmit ready</td>
</tr>
<tr>
<td>0066</td>
<td>WMN1</td>
<td>'6h     WMN address</td>
</tr>
<tr>
<td>1000</td>
<td>LWMEM</td>
<td>128*32 Length of video memory</td>
</tr>
<tr>
<td>000A</td>
<td>WPL</td>
<td>-22    IFfirst line video offset</td>
</tr>
<tr>
<td>0000</td>
<td>VLL</td>
<td>128    LLength of one video line</td>
</tr>
<tr>
<td>0085</td>
<td>SCLR</td>
<td>085H   IFor DELAY routine</td>
</tr>
<tr>
<td>000A</td>
<td>LF</td>
<td>'0Ah   IFJ, LF = Line Feed</td>
</tr>
<tr>
<td>0000</td>
<td>CR</td>
<td>'0DH   IM, CR = Carriage Return</td>
</tr>
<tr>
<td>0003</td>
<td>ESC</td>
<td>'13H   IT, ESC = Escape</td>
</tr>
<tr>
<td>007F</td>
<td>ECR</td>
<td>'7Fh   ILLEGAL KEY</td>
</tr>
<tr>
<td>0008</td>
<td>SBS</td>
<td>'08h   IBACKSPACE</td>
</tr>
<tr>
<td>0009</td>
<td>TAB</td>
<td>'09h   IBACKSPACE</td>
</tr>
<tr>
<td>0033</td>
<td>KLLEN4</td>
<td>3      Idt KEY LIST LENT4</td>
</tr>
<tr>
<td>0002</td>
<td>KLELEN</td>
<td>2      Idt KEY LIST ENTRY LENT4</td>
</tr>
<tr>
<td>0007</td>
<td>KLUSED</td>
<td>7      Idt KEY LIST ENTRY USED</td>
</tr>
</tbody>
</table>
RMM STORAGE LOCATIONS

= 0008  KEY_SVRD = 6  ; KEY SERVICED ONCE
= 0038  KROW_MSK = 38M  ; KEY NUMBER MASK
= 0037  KCOL_MSK = 7H  ; KCOL NUMBER MASK
= 0031  DB.CS = 1  ; DBOUNCE COUNT
= 0038  IPTCCT = 24  ; INITIAL REPEAT COUNT (400MS)
= 0006  SRTCCT = 6  ; SECOND REPEAT COUNT (100MS)
= 0007  TOT_RDM = 7  ; TOTAL ROWS
= 0002  CLR.LKY = 2  ; COLUMN NUMBER OF CLR+ALPHA AND SHIFT KEYS
= 0033  ALPH.KY = 3  ;
= 0034  SHIFT.KY = 4  ; REPEAT COUNT FOR SLIDE KEYS (50MS APPROX)
= 0033  SLU.CT = 3  ;
= 0080  BRTBIT = 80h  ; IS set bfr/dim memory BRIGHT
= 0000  DMBIT = 00h  ; IS set bfr/dim memory DIM
= 003A  WRTRY = 10  ; NUMBER OF RETRYS

MEMORY MAPPED I/O

= 2100  D.CMDR = 0200h  ; Floppy disk disk command reg (WRITE)
= 2100  D.CMDR = 0200h  ; Floppy disk disk command reg (READ)
= 2101  D.TRKR = D.CMDR+1  ; TRACK REG
= 2102  D.SECR = D.CMDR+2  ; SECTOR REG
= 2103  D.CATR = D.CMDR+3  ; DATA REG (R/W)
= 2200  H.KEYS = 0220h  ; Keyboard
= 2900  CPDA4 = 02900h  ; Peripheral/directory register 4
= 2901  CPRA1 = CPDA4+1  ; Control register 4
= 2902  CPDA5 = CPRA1+1  ; Peripheral/directory register 5
= 2903  CPRA2 = CPDA5+1  ; Control register 3
= 2900  PA.DTA = CPRA2+2  ;
= 2901  PA.CTL = CPRA2+3  ;
= 2902  PA.DTA = CPRA2+2  ;
= 2903  PA.CTL = CPRA2+3  ;
= 2A00  A.SCTRL = 02A00h  ; Set control reg (WRITE)
= 2A00  A.SCTRL = 02A00h  ; Status reg (READ)
= 2A01  A.SMXMT = 02A00h+1  ; Transmit address
= 2A01  A.SMXMT = 02A00h+1  ; Receive (read from address)
= 2900  D.VID = 02500h  ; Video memory controls

RAM MEMORY LOCATIONS

= EF00  TVM = OEF00h  ; (11) USED IN 300T ROUTINES
= EF05  TRY = OEF05h  ; (11) RETRY COUNTER
= EF38  ROM_RAM = OEF38h  ; (11) ROM/RAM FLAG
= EF39  DISK = OEF39h  ; (11) DISK INFO
= EF3F  DMA = OEF3Fh  ; (11) DMA ADDRESS
= EF13  SESSION = OEF13h  ; (11) DISK STEP DELAY
= EF14  SAVSEC = OEF14h  ; (11) SECTOR
= EF15  SAVTRK = OEF15h  ; (11) TRACK
= EF17  SISK = OEF17h  ; (11) DISK
= EF50  MSTART = OEF50h  ; (11) DISK
= EF55  UMARK = OEF55h  ; (11) DISK
= EF56  LOGSEC = OEF56h  ; (11) DISK
= EF59  KEYLOC = OEF59h  ; (11) KEYBOARD LOCKED CELL
= EF5A  CURS = OEF5Ah  ; (11)
= EF5E  LKEYS = OEF5Eh  ; (11)
= EF60  ESCH = OEF60h  ; (11)
= EF61  PIAAD = OEF61h  ; (11)
= EF62  PIAAD = OEF62h  ; (11)
= EF6A  DATA = OEF6Ah  ; (11) DISK ACTIVE FLAG
= EF58  RELCNT: = DEF58H  \(1\) IS BELL RINGING
= EF6C  LIMIT: = DEF6CH
= EF6E  LDTK: = DEF6EH
= EF6F  IESTK: = DEF6FH  \(1\)(2) SAVE STACK POINTER HERE
= EF99  ISTK: = DEF99H  \(1\)(−40) INTERRUPT STACK
= EFCA  ACIAO: = DEFCAH  \(1\)(−40) ROM STACK
= EFCA  ACIAH: = DEFCAH
= EFCC  SEQ: = DEFCH  \(1\)(2) COUNTER
= EFDD  SAVTPY: = DEFDH  \(1\)(1) DISK TYPE
= EFDC  L4C3M: = DEFCH  \(1\)(1) NUMBER OF SECTORS TO READ OR WRITE
= EFDD  CDADR: = DEFDH  \(1\)(2) This location is assigned in BIOS and filled in by loader in ROM
= EFDA  KEYS: = DEF4H  \(1\)(6) KEY LIST GOES HERE
= EFDA  SETFLG: = DEF4H
= EFDB  IE_ADRS: = DEF0H  \(1\)(1) device address
= EFDC  IE_CHAN: = DEF0H  \(1\)(1) IE in char buffer
= EFDD  PIACTL: = DEF2H  \(1\)(1)
= EFDE  PPMODE: = DEF2H  \(1\)(1) Parallel port input, output, undefined
= EFEF  VTORPF: = DEF7H  \(1\)(1) LAST VERTICAL OFFSET
= EFFF  INTRBL: = DEF0H  \(1\)(16) interrupt vector table
= F000  FWAIV: = OF00H  \(1\) FIRST ADDRESS OF MEMORY MAP
IEEE EQUATES

PORT CTI REGISTER CONSTANTS

= 002A
PA.COR = 00101010b (to address port a direction)
= 002E
PA.CST = 00101110b (to address port a data and set)
= 0030
PB.COR = 00000000b (to address port b direction)
= 0034
PB.CST = 00000100b (to address port b data)

PORT DIRECTION REGISTER CONSTANTS

= 00FF
PA.OUT = OFFh (port a output mode)
= 0030
PA.IN = 00h (port a input mode)
= 008F
PB.OUT = 00h (port b direction)
= 0038
PB.IN = 00000010b (port b data for output)
= 0039
PB.STB = 00001011b (port b data for input)
= 0040
PP.OUT = 01000000b (output rdy bit in pb)
= 0080
PP.IN = 10000000b (input rdy bit in olb ct1 reg)
= 0020
STB = 00100000b (istrobe bit in port b)

PORT MODES

= 0001
PP.OUT = 1
= 0002
PP.IN = 2

IEEE CONTROL CODES

= 0040
IE.TALK = 60h (make talker)
= 005F
IE.UTLK = 5Fh (make untalk)
= 0020
IE.LSTN = 20h (make listener)
= 003F
IE_ULST = 3Fh (make unlisten)
DISK EQUATES

= 0010  D.SET:    = 010H    ;SEEK
= 0020  D.STP:    = 020H    ;STEP
= 0040  D.STP1:   = 040H    ;STEP IN
= 0060  D.STP0:   = 060H    ;STEP OUT
= 0080  D.RDS:    = 080H    ;READ SECTOR
= 00A0  D.WRS:    = 0AH    ;WRITE SECTOR
= 00C0  D.RBA:    = 0CH    ;READ ADDRESS
= 00E0  D.RDT:    = 0EH    ;READ TRACK
= 0FF0  D.WRT:    = 0FH    ;WRITE TRACK
= 0000  D.FINT:   = 00H    ;FORCE INTERRUPT

;MACRO DEFINITION

ENADIM: MACRO
    OUT 2
ENDM

DISDIM: MACRO
    OUT 3
ENDM
Monitor Main Loop

$11

0000  START:
  ;initialize all dependent hardware.

0000  PROC
0000  31C1EF  LDX  SP  ;ROMSTK  iset stack
0003  CD300E  CALL  FORINT  ;disable current DISK command, if any

  ;disable dim bit

0006  DISDIM

  ;set interrupts
  ;set mode 2 interrupts

0008  EDE5  $  IM2
  ;set interrupt register

000A  3EEF  $  LDX  A+high INTBL
000C  ED47  $  MOV  I=A
  ;set keyboard vector

000E  21FC06  LDX  ML+GKEY
0011  22F8EF  STD  ML+INTBL+(4*2)  ;set keyboard interrupt
  ;set serial vector
  ;set IEEE vector

  ;initialize memory

  ;clear all but interrupt vectors

0014  2100EF  LDX  4L+OFF004  ;start
0017  06FD  LDX  B+OFF0H  ;length
0019  AF  XRA  A
001A  77  ;CLOOP: STD  A+IHL
001B  23  INC  HL
001C  10FC  *01A5  DJNZ  ;CLOOP  ;clear LDJP

  ;set values of one

001E  3C  INC  A  ;A = one
001F  3239EF  STD  A+KEYLCK  ;indicate not locked
0022  4F  MOV  C,A  ;FOR "IE+CM"

  ;set values of two

0023  3C  INC  A  ;A = two
0024  3233EF  STD  A+SEKDEL  ;set seek step rate for SEMIENS
0027  C06800  CALL  SPAD  ;set up for output (Saves REG :3)
MONITOR MAIN LOOP

0020 CD3529 CALL IE+CD  ;REG C=1 ($AVES REG C)

0022 DD DEC C  ;IC=0
0026 CD8600 CALL OFAD

;Set for -10 char position AND DOUBLE DENSITY
0032 DEEA LDK C+FLD
0036 CD7990 CALL OFAD
0038 3E8D LDK A+VLL
0038 326E5F STO A+LIMIT  ;set max line limit

;Reset-Master clear the SI3 (ACIA)
0038 DE55 LK C+SI,S16  ;select 16x clock for 1200 baud
003D CD8209 CALL SIRST  ;Reset

*SIGN ON PROMPT

004B 113F01 LDK 0E,EMSG  ;Output initial message
004C CD9000 CALL OSTR
004E FB EI  ;ENABLE INTERRUPTS

0047 CD7303 CALL EI  ;Get character
004A FE18 CMP ESC
004C CA4D2F JZ MCBOOT  ;IF COLD BOOT CF CF HARD DISK

004F 21C7EF LK =L+DSKSW?  ;disk swap call
0052 FE0D CMP CR
0054 CA5522 JZ CB0DT  ;If cold boot off of 4

0057 34 INC [HI]  ;SWAP drives: 4=3, 3=A
0058 FE22 CMP ***
005A CA5522 JZ CB0DT  ;If cold boot off of 3

005D 18A1 *C000$ JR START  ;LOOP
Monitor Main Loop.

*Interrupt vector for reset button (non-maskable interrupt)*

```
006F = 0066          JRG NMIA
0066 1898 800004     JR START ;START
```
SOURCE: BD8X Assembler ver 3.5E

Monitor Main Loop.

0068

SPAD:

;Set PIA for output
;ENTRY
;INC

;EXIT

HL = 4*VID+2

;CHANGE

HL

0068 PROC

0068 21012C LDK HL*VID+1 ;HL*VID*

0069 3603 STD 3+[HL] ;set data direction

0069 28 DEC HL ;HL

006E 36FF STD OFFH*[HL] ;set all A lines as output

0070 23 INC HL ;HL*VID+1

0071 23 INC HL ;HL*VID+2

0072 23 INC HL ;HL*VID+3

0073 3600 STD 0+[HL]

0075 23 DEC HL ;HL

0076 36FF STD OFFH*[HL] ;set all 8 lines as output

0078 C9 RET
Monitor Main Loop

0079

; PAD:
; Output data to PIA A register
; PIA definition:
; 7 6 5 4 3 2 1 0
; +-----+-----+-
; | horizontal offset 1001 |
; +-----+-----+-

; NOTE: The D0 (double density) bit is inverted and the jumper must be installed on the 00 board.
; If the D0 bit is LOW double density is set if it is HIGH single density is set.

; NOTE: bit D0 of "PIA A":
; set = single density
; reset = double density

; NOTE:
; SAVE ML

; ENTRY = data

; EXIT =

0079 PROC

0079 3E07
0078 32012C

007E 79
007F 3261EF
0082 32002C
0085 C9

LDR A+43
STD A+4, V1+1

MOV A+1
STD A+PIA40
STD A+4, V1C
SEND data

RET
MONITOR Main Loop

0086

DB88:
1) Output data to pia b register
   - PIA definition:
     7 6 5 4 3 2 1 0
     ++++++++-----++-
     1D1001^01 vert offset 1
     ++++++++-----++-

get
IC = data

; ENTRY
; IC = data
; EXIT
; IDONE

0086 PROC

0086 3E04   LDX A+4
0088 32032C  STO A+4, V10+3
008B 7F     MOV A+C
008C 3262EF  STO A+PIA3D
008F 32022C  STO A+4, V10+2
0092 C9     RET : send data
Monitor Main Loop,

<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0093</td>
<td>B016</td>
<td>LDX DE+1533h</td>
<td>Offset in BIOS jump table</td>
</tr>
<tr>
<td>0096</td>
<td>1803</td>
<td>JR B1JP</td>
<td></td>
</tr>
<tr>
<td>0098</td>
<td>B016</td>
<td>LDX DE+1635h</td>
<td>Offset in BIOS jump table</td>
</tr>
<tr>
<td>009E</td>
<td>19</td>
<td>ADD HL,CCPA00</td>
<td></td>
</tr>
<tr>
<td>009F</td>
<td>E9</td>
<td>JMP [HL]</td>
<td></td>
</tr>
</tbody>
</table>
Monitor Main Loop.

EMBOOT: DB CR,LF
EMBOOT ERROR: DB '*BOOT ERROR*', DC 1H
EMBOOT: IADOT ERROR MESSAGE ROUTINE
ENTRY
ENTRY
PROC
LDK DE,EMBOOT WHERE ON BOOT ERROR
FALL THROUGH TO DSTR
Monitor Main Loop

0080

OSTR:
OUTPUT STRING TO CONSOL
NOTE: OSTR RECOGNIZES FF AS AN ESCAPE SEQUENCE TO REPEAT CHAR N TIMES. FORMAT IS: FF, REPEAT COUNT, CHAR
ENTRY
IDC = FWA OF SOURCE
EXIT
DONE

0080

PRDC

0080 1A
0081 07
0082 F5
0083 E67F
0085 FE7F
0087 4F
0088 200C ^0065
0089 13
008A 1A
008B 3D
008C 47
008D 13
008E 1A
008F 4F
0090 CD0003
0091 12
0092 D698 ^0015
0093 12
0094 CD0003
0095 14
0096 CD0003
0097 13
0098 F1
0099 F28000
009A 59
009B 59
009C 59
009D 59
009E 59
009F 59
00A0 59
00A1 59
00A2 59
00A3 59
00A4 59
00A5 59
00A6 59
00A7 59
00A8 59
00A9 59
00AA 59
00AB 59
00AC 59
00AD 59
00AE 59
00AF 59
00B0 59
00B1 59
00B2 59
00B3 59
00B4 59
00B5 59
00B6 59
00B7 59
00B8 59
00B9 59
00BA 59
00BB 59
00BC 59
00BD 59
00BE 59
00BF 59
00C0 59
00C1 59
00C2 59
00C3 59
00C4 59
00C5 59
00C6 59
00C7 59
00C8 59
00C9 59
00CA 59
00CB 59
00CC 59
00CD 59
00CE 59
00CF 59
00D0 59
00D1 59
00D2 59
00D3 59
00D4 59
00D5 59
00D6 59
00D7 59
00D8 59
00D9 59
00DA 59
00DB 59
00DC 59
00DD 59
00DE 59
00DF 59
00E0 59
00E1 59
00E2 59
00E3 59
00E4 59
00E5 59
00E6 59
00E7 59
00E8 59
00E9 59
00EA 59
00EB 59
00EC 59
00ED 59
00EE 59
00EF 59
00F0 59
00F1 59
00F2 59
00F3 59
00F4 59
00F5 59
00F6 59
00F7 59
00F8 59
00F9 59
00FA 59
00FB 59
00FC 59
00FD 59
00FE 59
00FF 59

Copyright ©1983 Osborne Computer Corporation
DELAY:
: N = Milliseconds
: ENTRY
: IA = Number of Milliseconds to delay
: SCLFRE = (Freq/1000)/25

: EXIT
: NONE

: PROC

: 4F
: 3E35
: 01
: LDK A+SCLFRE

: 30
: MLODP: DEC A
: (4 tics)

: 40
: MOV B=B
: (4 tics)

: 49
: MOV C=C
: (4 tics)

: 20F8 ^0024
: JRNZ :MLODP
: (10 tics) If l as not elapsed

: 00
: DEC C
: If requested usec not done

: 20F8 ^0008
: JRNZ :1

: 09
: RET

: SCLFRE = 2000/22 =280, 2mhz
: *defined in OCCTXT.ast
Monitor Main Loop.

SERCIM 808x Assembler ver 3.xE  <i>/55/7</i> =8:42 Page 16
8:43M41 .4SM

0000 = 0110 ORG 100h

FROM JUMP TABLE
: CBiOS = Jumps used mainly by CBiOS
: SC  = Jumps used mainly by SuperCalc

0100 C35502 JMP CBiOS ICBiOS cold boot
0103 C39C02 JMP WBOOT ICBiOS warm boot
0106 C36503 JMP SKEY ICBiOS keyboard status
0109 C37303 JMP CI ICBiOS keyboard input
010C C3E003 JMP COUT ICBiOS console output
010F C3D908 JMP LIST ICBiOS list output
0112 C3D908 JMP LIST ICBiOS punch output
0115 C3E908 JMP READER ICBiOS reader input

: Disk I/O
0118 C3020C JMP RDRV ICBiOS HWE
011B C9 RET ICBiOS SELECT DISK
011C 0O NOP
011D 00 NOP
011E C33000 JMP READ ICBiOS READ SECTOR
0121 C33700 JMP WRITE ICBiOS WRITE SECTOR
0124 C3A50D JMP RDOR ICBiOS READ SECTOR ANY SECTOR HEADER
0127 C3100C JMP RSEC ICBiOS DISK SECTOR READ
012A C3230C JMP WSEC ICBiOS DISK SECTOR WRITE
012D C30008 JMP SLST ICBiOS LIST DEVICE STATUS
0130 C3690C JMP SENDEN ICBiOS SENSE THE SECTOR OF DRIVE
0133 C3930D JMP ROMPJ ICBiOS
0136 C39900 JMP ROMPJ ICBiOS
0139 C30F0F JMP FORMAT ICBiOS FORMATTING ROUTINE

: IEEE
013C C38208 JMP SIRST ICBiOS SIO reset
013F C33609 JMP IE.CO ICBiOS IEEE Control Out
0142 C37909 JMP IE.SI ICBiOS Status In
0145 C38C09 JMP IE.GTS ICBiOS Go To Standby
0148 C39509 JMP IE.TC ICBiOS Take Control
014B C38B09 JMP IE.DIM ICBiOS Output Interface Message
014E C3F309 JMP IE.DOM ICBiOS Output Device Message
0151 C33304 JMP IE.IO ICBiOS Input Device Message
0154 C39904 JMP IE.DO ICBiOS Parallel Poll

: SuperCalc
0157 C3CF06 JMP VLDR ICSC VIDEO BLOCK MOVE DEC
015A C3F206 JMP VLDIR ICSC VIDEO BLOCK MOVE INC
015D C3C006 JMP STDIM ICSC STD reg B IN [HL]

: DISK I/O
0160 C3F00E JMP DMWRT ICSC DMA WRITE TO CONTROLLER
0163 C3D40E JMP DMRD ICSC DMA READ FROM CONTROLLER
0166 C33D0C JMP HDDM ICSC HOME DISK DRIVE
0169 C3F40C JMP SEEK ICSC SEEK TO TRACK
016C C3990D JMP STEP ICSC STEP SAME DIRECTION
016F C3000D JMP STEP IN ICSC STEP IN
0172 C3110D JMP STEP OUT ICSC STEP OUT
Monitor Main Loop

SOLCIM 808x Assembler ver 3.5E C:\/557= -9:92 Page 17

0175 C3363E JMP FORINT  FOICE INTERRUPT
0176 C3EE0D JMP READTRK  READ TRACK
0177 C3013E JMP FMTTRK  FORMAT ONE TRACK
0178 C3713E JMP SELDRV  SELECT DRIVE
0181 C3E80B JMP ACISTAT  ICBIOS SERIAL PORT STATUS
0194 C3013E JMP SCTRKR  SET TRACK REGISTER IN CONTROLLER CHIP WITH VALUE IN SAVTRK
0187 C3810A JMP IEINSTAT  ICBIOS IEEE
0188 C3A50A JMP IEOUT  ICBIOS IEEE
0190 C3E00A JMP IEINIT  ICBIOS IEEE
0193 C3780B JMP PSTAT  ICBIOS IEEE
0196 C36C0B JMP POSTAT  ICBIOS IEEE
0199 C38E0B JMP PAINP  ICBIOS IEEE
019C C3900B JMP PAINOUT  ICBIOS IEEE
019F 1A0A0A0A IMG: DB 'Z'-40h,lf,lf,lf,lf
01A4 7FDB20 DB 07Fn, 11, **
01A7 1867 DB ESC+VSGH
01A9 11 DB 'Q'-40h
01AA 7F1817 DB 07Fn, 24, 'W'-40h
01AD 05 DB 'E'-40h
01AE 1847 DB ESC+VEGH
01B0 00DA DB cr,lf
01B2 7F0B20 DB 07Fn, 11, **
01B5 1B57111847 DB ESC+VSGH, 1, ESC+VEGH
01BA 2020202020 DB ESC, **
01C1 186C DB 'DISKONE 1'
01C3 4F53424F52 DB 'DISKONE 1'
01C4 186D DB ESC+*m
01CE 2020202020 DB *
01D6 1B57941847 DB ESC+VSGH, 4, ESC+VEGH
01DB 00DA DB cr,lf
01D0 7F0B20 DB 07Fn, 11, **
01D4 1867 DB ESC+VSGH
01D6 01 DB 'A'-40h
01D8 7F1820 DB 07Fn, 24, **
01DE 04 DB 'D'-40h
01E7 1847 DB ESC+VEGH
01E9 00DA DB cr,lf
01EB 7F0B20 DB 07Fn, 11, **
01EE 1867 DB ESC+VSGH
01F0 01 DB 'A'-40h
01F1 1847 DB ESC+VEGH
01F3 1529 DB ESC+*m
01F5 2020520576 DB Ray* 1, 41 (c) 1982 OCC *,
0200 1828 DB ESC+*m
020F 1867 DB ESC+VSGH
0211 04 DB 'D'-40h
0212 1847 DB ESC+VEGH
0214 00DA DB cr,lf
0216 7F0B20 DB 07Fn, 11, **
0219 1867 DB ESC+VSGH
021B 1A DB 'Z'-40h
021C 7F1818 DB 07Fn, 24, 'X'-40h
021F 03 DB 'C'-40h
0220 1847 DB ESC+VEGH
0222 000A0A0A DB cr,lf,lf,lf
0226 7F3420 DB 07Fn, 4, **
0229 495E35572 DB 'Insert Disk in Drive '
022E 186C DB ESC+*m
0230 41 DB 'A'
0234 1853 DB ESC+*m
0243 20615E5420 DB 'and press RETURN',
0254 AE DC **
0254 C025C: "LOAD ALL THE OPERATING SYSTEM INCLUDING THE CMS ENTRY"
0255 "HOME"
0256 "EXIT"
0257 "A = DRIVE TO BOOT FROM"

0258 PROC
0259 "SET "SAVITY""

0259 E025C "1: CALL RORV HOME DRIVE"
0259 E025D CALL SENDEN "DETERMINE DENSITY"
0259 E0269 JRZ 12 "IF GOOD"
0259 E0269 CALL EB00 "PRINT ERROR"
0259 E026F JR 11 "READ AND SET FBA OF CCP"

0260 18F3 "10000 CALL LDL HL+0000H"
0261 220FE CALL STD HL+0000H "SET DMA"
0262 55 PUSH HL
0263 E026B CALL LDL A+1 "SET SECTOR"
0264 E026C MOV B+A "READ SECTOR ONE"
0265 E026D CALL RSEC "READ SECTOR ONE"
0266 E026E JC 11 "IF GOOD"
0267 E026E CALL EB00 "PRINT ERROR"
0268 18F0 "0269 CALL LDL HL+0000H"
0269 E026F JR 13 "CHECK FIRST TWO BYTES OF THE CCP"

0269 E027A LD A,[HL] "IF FIRST BYTE"
0269 E027B CMP D0H "IF NOT THE SAME"
0270 20DE "025D CALL JRNZ FERR"
0271 E027D INC HL "IF NOT THE SAME"
0272 E027E LD A,[HL] "IF SECOND BYTE"
0273 E027F CMP 05H "IF NOT THE SAME"
0274 20DE "025D CALL JRNZ FERR"
0275 E027F INC HL "IF NOT THE SAME"
0276 E0280 "07E SET LOAD ADDRESS"

0276 E0280 LD A,[HL] "GET CCP ADDRESS/1004 + 3"
0277 E0287 D03 SUB 3
0278 E0287 2000 CALL LDL L+0 "SET LOAD ADDRESS"
0279 E0288 57 MOV M+A
SET NUMBER OF 128 BYTE BLOCKS TO READ FOR SIO TO

028C 063C   LDK   B+60   ICCP/BDOS/CDOS

READ SYSTEM

028E E5   PUSH   HL   ;SAVE FWA FOR "CCPADR"
028F CD102  CALL   BCPM   ;boot system
0292 E1   PDP   HL

JJMP SYSTEM

0293 4F   XRA   A   ;DRIVE BOTTED FROM
0294 22D2EF  STD   HL+CCPADR   ;SET "CCPADR"
0297 110016  LDK   D0160Dh   ;offset for FWC
029A 10   ADD   HL+DE   ;address of BIOS in
029B E9   JMP   [HL]   ;enter com
D29C

4B00:
LOAD ONLY THE CCP AND THE BDOS FROM DRIVE A
ENTRY

INDEX

NOTE

$13 = NUMBER OF 128 BYTE BLOCKS TO READ FOR BDOS
$18L = DMA ADDR FOR CCP

D29C

PROC

D29C 362E
D29E 2932EF

LDX 5+44        CCP/BDOS and don't read CAI0S
LD  HL*CCPADR
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. TCM 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
TB = NUMBER OF 128 BYTE BLOCKS TO READ FOR BOOT
HLM = DMA ADDR FOR CPM
EXIT
IN:NE

D2A1

PRDC

D2A1 220FEF  STD ML+MACR ;SET DMA
D2A4 6F     XRA A                  ;SET DMA
D2A5 3217EF  STD A+SDISK              ;BOOT ONLY FROM DRIVE A
D2A8 3200EF  STD A+TEM              ;MAKE TEM ZERO
D2AB 3C     INC A
D2AC 3214EF  STD A+SAVESec            ;Get sector

D2AF C5     PUSH BC                  ;SAVE NUMBER OF 128 BLOCKS
D2B0 CD02DC  RR11: CALL RDRV         ;SHOW DRIVE
D2B3 CD690C  CALL SENDEN              ;DETERMINE DENSITY
D2B6 2805 *O28D$ JRZ 11               ;IF GOOD
D2B8 CDADD0  CALL EBOOT                ;PRINT ERROR
D2BD 18F3 *O230$ JR RR11             ;
D2BE 31     PDP DE                   ;ID=NUMBER OF 128 BYTE BLOCKS
D2BF C5     PUSH BC                  ;SAVE NUMBER OF SECTORS IN ONE TRACK

D2BF 3A00EF  LD A+SAVTYPE
D2C2 CB3F $  SRL A
D2C4 CB3F $  SRL A
D2CA 9603  AND 0000_0013             ;IA=NUMBER OF BYTES IN ONE SECTOR(0-3)
D2CB 2318 *O252$ JRZ 12               ;IF 128 BYTES SECTORS
D2C8 47     MOV B,A                  ;GET NUMBER TO DIVIDE BY
D2CA 47     MOV B,A                  ;IB=NUMBER OF BYTES IN ONE SECTOR(1-3)
D2CB 3E01     LDX A+1
D2CD CB27 $  ILOCP:SLA A                ;TIMES TWO
D2CF 10FC *O250$ DJNZ ILOCP
D2D1 47     MOV B,A                  ;NUMBER TO DIVIDE BY
Boot / from disk

S07CIM 808x Assembler ver 3.5E 1/15/70 #9192 Page 23

2002 7A
2003 1600
2005 90   #2005: SUB D
2006 08   $ EX AF
2007 14   INC D
2008 08   $ EX AF
2009 2007 #2007: JRA 12
200A 30F8  #200A: JRNC #2005
200B E046  $ NEG A
200C 3200EF STD A,TE2
200D 41    #21: PDP BC
200E 4A    MOV C,D

;A=NUMBER OF 128 BYTE BLOCKS
;SUBTRACT WITH DIVISOR
;SAVE FLAGS
;RESTORE FLAGS
;IF RESULT IS ZERO (4) PARTIAL SECTORS
;OOP
;12 COMP
;SAVE REMAINDER AND INDICATE A PARTIAL SECTOR
;NUMBER OF SECTORS IN ONE TRACK
;NUMBER OF SECTORS TO READ
READ SYSTEM

D2E4 AF
D2E5 3215EF
TL0OP:
KRA A
STD A
SAVTRK IS
SET TRACK

ICHECK FOR ALL SECTORS READ

D2EA 2008 D2F4-
D2EB 79
D2EC 3210EF
D2ED 97
D2EF 2858 D34D-
D2F2 1826 D31A-
MDV A+3
DRA C
JRZ: 13
IIF C IS NOT ZERO CONTINUE

ICHECK FOR NO PARCIAL SECTOR

D2F4 99 10 D2F5 3002 D2F6-
D2F7 41
D2F8 AF
LD A+3
DRA A
JRZ 17
READ PARCIAL SECTOR

UPDATE NUMBER OF SECTORS LEFT TO READ

D2F9 4F 14
D2FA 3A0DEF
D2FD 57
D2FE 2807 D307-
MDV C+A
DRA A
JRZ 15
IIF TEM IS ZERO SKIP THIS

ICHECK FOR NONZERO VALUE IN TEM AND THE LAST SECTOR TO READ

D300 AF
D301 81
D302 2003 D307-
D304 05
D305 2B1C D31A-
XRA A
DRA C
JRIZ 15
IIF REG C IS NOT ZERO SKIP THIS(NOT LAST TRACK)

READ ONE LESS THAN THE LAST SECTOR

D307 CD1C0C 15
D308 2B05 D311-
D30C CD4D00
D30F 18F6 D307-
DEC 8
JRZ 17
IIF ONLY ONE SECTOR LEFT TO READ

READ ONE TRACK

D307 CD1C0C 15
D308 2B05 D311-
CALL RSEC
JRZ 86
IIF READ (BC IS SAVED)

D30C CD4D00
D30F 18F6 D307-
CALL EB0DT
JZ 15
IIF REPORT ERROR

UPDATE DMA

D311 2206EF 26
STO HL+DMADR
SET DMA

UPDATE TRACK
3314 BA15EF
3317 3C
3318 18CB *02E5$ JR :TRACK :LC1P
      :LC1P

LD A+,SAVTRK
INC A

Copyright ©1982 Osborne Computer Corporation
READ A PARTIAL SECTOR

D318 2190EA
D319 2200EF

; Save address to write to
D320 10  ; push hl
D321 2804  ; load hl, SP
D322 2A  ; addr of host buffer in bios
D323 2A  ; set dma

; If 9=0 then SAVTRK was not incremented
D324 2804  ; load hl, SP
D325 2A  ; addr of host buffer in bios
D326 2A  ; set dma

; Set the sector to 3 + 1
D327 24  ; inc 3
D328 7B  ; mov a+b
D329 3216EF  ; mov a+SAVSEC
D330 10  ; inc 10

; Read sector into host buff
D331 2805  ; load hl, SP
D332 2A  ; addr of host buffer in bios
D333 2A  ; set dma

; Read one sector
D334 2805  ; load hl, SP
D335 2A  ; addr of host buffer in bios
D336 2A  ; set dma

; Increment error
D337 2805  ; load hl, SP
D338 2A  ; addr of host buffer in bios
D339 2A  ; set dma

; Report error
D340 2805  ; load hl, SP
D341 2A  ; addr of host buffer in bios
D342 2A  ; set dma

; Set number of bytes to transfer
D343 2805  ; load hl, SP
D344 2A  ; addr of host buffer in bios
D345 2A  ; set dma

; Number of bytes to transfer
D346 2805  ; load hl, SP
D347 2A  ; addr of host buffer in bios
D348 2A  ; set dma

; Source
D349 2805  ; load hl, SP
D34A 2A  ; addr of host buffer in bios
D34B 2A  ; set dma

; Destination
D34C 2805  ; load hl, SP
D34D 2A  ; addr of host buffer in bios
D34E 2A  ; set dma

; Move
D34F 2805  ; load hl, SP
D350 2A  ; addr of host buffer in bios
D351 2A  ; set dma

; Transfer bytes
D352 2805  ; load hl, SP
D353 2A  ; addr of host buffer in bios
D354 2A  ; set dma

; Source
D355 2805  ; load hl, SP
D356 2A  ; addr of host buffer in bios
D357 2A  ; set dma

; Destination
D358 2805  ; load hl, SP
D359 2A  ; addr of host buffer in bios
D35A 2A  ; set dma

; Move
D35B 2805  ; load hl, SP
D35C 2A  ; addr of host buffer in bios
D35D 2A  ; set dma

; Transfer bytes
D35E 2805  ; load hl, SP
D35F 2A  ; addr of host buffer in bios
D360 2A  ; set dma

; Source
D361 2805  ; load hl, SP
D362 2A  ; addr of host buffer in bios
D363 2A  ; set dma

; Destination
D364 2805  ; load hl, SP
D365 2A  ; addr of host buffer in bios
D366 2A  ; set dma

; Move
D367 2805  ; load hl, SP
D368 2A  ; addr of host buffer in bios
D369 2A  ; set dma

; Transfer bytes
CLEAR BUFFERS SET PARA AND RETURN TO SYSTEM

0340 2150EF     LDK HL, MSTATC
0350 3600       STD $[HL]     ; 1st byte
0352 01050D     LDK BC, (LISRSEC-MSTATC)
0355 1151EF     LDK DE, (MSTATC+1) ; DE = HL + 1
0358 E030      LDIR         ; overlapping move
035A 3EFF       LDK A, EFFh
035C 3275EF     STD A, UNASEC
035F 3EFF       LDK A, VLL-1
0361 324EEF     STD A, LDRX    ; set other drive NCT int
0364 AF         XRA A        ; Clear error indicator
0365 39         RET
```assembly
0366      3A59EF     LD    A,KEY1K
0369      B7        OR    A
036A      C8        RZ

036B      3A5EEF     LD    A,KEY
036E      B7        ORA   A
036F      C8        RZ

0370      F6FF      QRI   0FFH
0372      C9        RET
```

$KEY:
:Get status of keyboard

:EXIT
:Exit set if no data ready

:EXIT
:Exit set if no data ready

if locked keyboard

:CHECK FOR ZERO

if not zero make 0FFH

if data
`0373 CI:
0373 `RKEY:
  `Read next key from keyboard
0373 `EXIT
0373 PA = `last key

0373 PROC
0373 006033 CALL SKEY
0376 28FB `3373$ JRZ RKEY `if NO data
0378 F3 DI
0379 3A5EEF LD A+KEY `GET CHARACTER
037C 6F MOV C+A
037D AF XRA A
037E 325EEF STO A+KEY `clear key from hold
0381 79 MOV A+C
0382 F8 EI
0383 C9 RET`
8 Bit definitions for ESC4 flag byte
*Note Bit 7 is currently free.*

= 0020 EF_SCR: = 32 185= Screen/ Cursor Addressing
= 0010 EF_ADR: = 16 184= expecting address-adr
= 0008 EF_ESC: = 8 183= last char was ESC
= 0004 EF_U:
= 0002 EF_HA: = 2 181= Half intensity mode
= 0001 EF_GL: = 1 180= Graphics mode
= 0000 EF_MSK: = EF_UN + EF_HA + EF_SR 1Mask to get mode.*
Vector (branch) table for video output mode selection
controlled by ESCM mode

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0384</td>
<td>3704</td>
<td>Normal mode</td>
</tr>
<tr>
<td>0386</td>
<td>5405</td>
<td>Graphics mode</td>
</tr>
<tr>
<td>0388</td>
<td>6504</td>
<td>Half intensity mode</td>
</tr>
<tr>
<td>038A</td>
<td>5904</td>
<td>Half and graphics</td>
</tr>
<tr>
<td>038C</td>
<td>5704</td>
<td>Underline mode</td>
</tr>
<tr>
<td>038E</td>
<td>5B04</td>
<td>Under and graphics</td>
</tr>
<tr>
<td>0390</td>
<td>5F04</td>
<td>Under and half intensity</td>
</tr>
<tr>
<td>0392</td>
<td>5304</td>
<td>Under and half and graphics</td>
</tr>
</tbody>
</table>
Succinct 808x Assembler ver 3.5E c: 557 = 9392 Page 32

Keyboard and Console Routines.

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 addr.

0394 30
DB VCAD "DW" ESCCAD "Cursor Addressing"
0397 33
DB VSAD "DW" ESCSD "Screen Addressing"
039A 57
DB VSGH "DW" ESCSSR "Set graphics mode"
039D 47
DB VEGH "DW" ESCSGR "Clr graphics mode"
03A0 29
DB VSH1 "DW" ESCSHA "Set half int. mode"
03A3 28
DB VEHL "DW" ESCSCH "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" ESCLE "Insert line"
03B8 52
DB VDELL "DW" ESCRL "Delete line"
03BA 54
DB VCEOL "DW" EEOL "Clear to end of line"
03BE 23
DB VLOCX "DW" ELOCX "Lock keyboard"
03C1 22
DB VULCX "DW" EULCX "Unlock keyboard"

03C4 1A04 0:
\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
VALICT: Valid control character table
13 bytes per entry: ASCII char, "DW" - Vector no., no. of entries is VALCTS
Following body of table is 2 byte No-Match adrs

DB CR <DW VC_CR Carriage return routine
DB LF <DW VC_LF Line feed
DB BK <DW VC_BK Back space
DB MC <DW VC_MC Move cursor right
DB MCUP <DW VC_MCUP Move cursor up
DB CBELL <DW VC_BEL Ring bell
DB VCRS <DW VC_CLRS Clear screen
DB VHOME <DW VC_HOME Cursor home

DW VOUT97 <NO match--ignore unless control char

= DD308 VALCTS = (#-2)-VALICT//3 <Number of valid entries
03E0
CDUT:
; General output routine to Video Screen
ENTRY
IC = Character
ICURS = Cursor
IESCH = Flag.Mode
; CURS & ESCH updated, A=Character
; (IC, de, hi preserved)
; ESCH is flag + mode byte as follows
; 00 = Normal mode & last chr Esc flag false
; 01 = Normal mode & last chr Esc flag true
; 03, 07: Mode is Graphics, Half, or Under, respectively and last chr Esc flag is false.
; 03, 07: As above, but mode is combination
; 09-15: Last chr Esc flag true otherwise like 1-7.

03E0
PROC

03E0 F5
03E1 C5
03E2 3A62EF
03E3 EE60
03E4 47
03E5 3A62EF
03E6 EE60
03E7 50
03E8 3A62EF
03E9 EE60
03E0 4F
03E1 C08B00
03E2 C1
03E3 F1
03E4 E6
03E5 D5
03E6 C5
03E7 2A5AEF
03E8 3A5AEF
03E9 47
03E0 E608
03E1 2023 ^04256
03E2 79
03E3 FE1A
03E4 78
03E5 2915 ^041D6

03E6 RES VERTICAL OFFSET WITH VRTOFF

RESET VERTICAL OFFSET WITH VRTOFF

; Current chr is NOT ESCbased. Is this chr ESC?

0402 MOV A,C
0403 CMP ESC
0404 JRZ ESC

; if this chr = ESC
HERE with A=3

0408 55   OUT:  PUSH  HL
0409 213403  LDX  HL, ESCH?
040C E607  AND  EF_MSK
040E 37  ADD  A\#A
040F 5F  MOV  E\#A
0410 1603  LDX  D\#D
0412 19  ADD  -HL, D

VECTOR:

0413 7E  LD  A\#(HL)
0414 23  INC  HL
0415 66  LD  H\#(HL)
0416 6F  MOV  L\#A
0417 E3  XTHL
0418 79  ADD  A\#C
0419 C9  RET

ENTRY point, note HL on stack.

First byte (low order addr).  
12nd byte (hi order addr).  
HL=addr from table.  
Restore HL from stack.  
Stack to HL.

Entry routine per table adrs.
$ORACLE 808x Assembler ver 3.5E 3/15/72 Page 36
Keyboard and Console Routines

D41A CDJT2
D41A 78 MOV A, B ; recall ESC value
D41B 18E3 ^040B$ JR ; output chr per current settings

D41D ESC: ; Current chr is ESC. Set flag and exit
D41D D60B OR EF, ESC
D41F 3260FF STD A, ESC
D422 C37B95 JMP VDJT97 ; Exit
SOLCIM 808x Assembler ver 3.5E <>55/7-9092 Page 37
Keyboard and Console Routines.
31R3M141 ASM

0425 PSTESC:
      last chr was ESC
      Entry
      $A  =  $EF_ESC
      $B  =  $ESCH
      $C  =  char to output
      $HL  =  $CURS

0425  PROC

0425  C353  $  BIT  $4B  ; Is this chr really an address?
0425  2075  *349E$  JRNZ  $SRTXY  ; If not, is part of an addr

in cursor/screen addressing in effect:

0429  A8  XOR  $B  ; Clear $EF_ESC bit (for next time)
042A  47  MOV  $B,A  ; Set up A = ESCH byte value
042B  3250EF  STO  $A,ESCH
042E  E5  PUSH  $ML  ; Save $CURS
042F  219403  LDX  $ML,VALIDE  ; Branch table addrs
0432  1E10  LDX  $E,VALETS  ; Table size
0434  79  MOV  $A,$C  ; Chr to $A
0435  1815  *044C$  JR  $LOOKJP3  ; Go to routine to see if cursor is after
NORMAL mode character processing

IA = char to output
IML = curs

FE20 CMP * *
3E68 JRC $211 IF control chr

F3 V97124: DI
3E80 ENA1M : 9th bit memory
3ED0 STO LR8BIT,[HL] Isat this chr LR8IF?
3EE0 DISC1M
3EE2 EI
35E0 JMP VDUT39

ES $21: PUSH HL : Save Curs
215603 LDX HL + VALID : Branch table adrs
1E08 LDX E + VALCTS : Table size

JMP LDDK,P93 : Scan table of valid control cns and branch to appropriate routine.
LOCUTPB:
Logic to scan 3 byte branch table:
ENTRY:
IHNL 1st byte of table (match code)
IE (2nd+3rd bytes + branch addr)
ITE = is table size (no. of entries)
IT (table body is followed with
12 byte "No-Match" addr)
ISE = stack has HL saved as top entry.
ITC = char
ITA = value to scan for possible match

DF4C BE
DF4D 23
DF4E 2853 *DF4F
DF50 23
DF51 23
DF52 10
DF53 20FF *DF54
DF55 1BEC *DF56
JMP [HL] (2nd byte of this 3 byte entry)
INC HL (3rd byte of this entry)
INC HL (1st byte of next entry)
DEC E (No. count of entries remaining)
JRZ LOCKUPB (Continue thru body of table)
JRB VECTOR "No-Match". HL points to vector
PROCESSING for modes other than normal.

IVGRAPH

Normal mode EXCEPT: ctsi chars are printed

0457
WUNDER:

0457 FE20
CMP * *
3459 380C *04374
JRC VNORM
; JR VUN_G2
; continue

045B

1Underlined Graphics

045B FE20
JR 80h

045D

1Underline bit

045D LE30
JR VARIGHT

ISet this chr BRIGHT

045F

1Underline and Half intensity

045F FE20
CMP * *
3451 3804 *04376
JRC VNORM
; JR VUN_MH_G2
; if ctsi-chor, process as normal

0463

1Underline, Half Intensity, Graphics

3463 F6B0
JR 80h

ISet underline bit

; JR VHA_G2

0465

1Half Intensity

0465 FE20
CMP * *
3467 380C *04375
JRC VNORM
; JR VHA_G2
; if ctsi-chor, process as normal

0469

1IC=Ctrl, RL=Cursor

0469 F3
SI

046A
ENANDIM

046C 3600
STD DIMBIT,41h

ISet dim field bit

046E
JISDIM

0470 F8
SI

0471 C35EC5
JMP WOUTSO
; continue
SCREEN:

:SETXY for Screen movement

ENTRY

IB = ESC4

IA = new co-ord val, NO OFFSET

SCREEN

PROC

C370 B 370 $ BIT 6,5
C200 200 200 JNZ :sxk ;if x-coordinate

E61F 661 F ;sxk AND 0001_1111b ÷mod 32

47A 6F 6F MOV CX,A

E2EF 32E EF STD AsVAIYFF ;SET VERTICAL OFFSET

E64F 664 F LO A, AsPIAD

E5EF 5E5 F AND 1110_0000b

91 39 91 JR C

48A 4F 4F MOV C,A

C960 C960 CALL OPAD ;set Y coordinate

182E 182 E JR :exitY

sxk ADD A, A ;double A

E6EA E66 A ADD AsVFLD ;PIA A-reg magic offset constant

1111_1111b AND

48F 4F 4F MOV C,A

;SET DENSITY BIT

3A1EF 3A1 EF LD A, AsPIAD ;GET OLD VALUE

E601 E6A 61 ANI 0000_0000b ;SAVE DENSITY BIT

91 39 91 DRA C ;FOR IN HORIZONTAL OFFSET

48A 4F 4F MOV C,A

CD90 CD7 90 CALL OPAD ;FUNCTION PIAD

C98 $ CRIT S+3 ;finished screen-addressing

1824 182 4 8 JR :exitX
Set X-Y value for Cursor/Screen Addressing

ENTRY

HL = cursor_addr
IY = ESCH
IC = chr

EXIT

Ifs VDOT90: ESC4 updated.

CDEF506 CALL UN_CJR
3EEE LDY A, (IY')
81 ADD C
CB68 BIT 5,3
E466 201C *E4745 JANZ SCREEN

Cursor addressing:

29 ADD HL,HL
C80D BIT 6,0
201D JR *E43D$ Iff X coordinate

67 MOV H,A
3A62EF LD A,(IY + 5)
3481 34 ADD H
1F RAR L
CB1D RR L
FF30 DR DF3h
67 MOV H,A

3E40 LDY A, 0150_00000b
1808 JR exitz2

17 RAL
C92C SRA H
1F RAR H
6F MOV L,A

2E7 LDY A, 0EF_MSK
40 ANB B
3250EF STC A,ESCH
C37305 JMP VDOT90
Control Code character processing

VC_HOME:

Home Cursor

CALL UN_CJR
LD A,PIA0
LD R,L
LD H,L
INC A
JR fijn

bit0 => CY

CY => bit7, trash bit0

HL := HL or F000h
S02C1M B03x Assembler ver 3.5E <1/55/7> Page 44

Keyboard and Console Routines.

3409 VC_MCU:
IA=CHR=MCUP, HL=CURs

3409 CDF606 CALL UN_CR
340C 05 PUSH HL
34DD 0180FF LDX BC, (VLL)
34DE 1817 0049 JR ifwa

....st this entry point

34E2 VC_B(S):
IL=CURS=current fold) char

34E2 CDF626 CALL UN_CR
34E5 3E7F LDX 4,7Fh
34E7 A5 AND L
34E8 2804 04EE5 JRZ wrap

....st this entry point

34EA 20 DEC HL
34EB C37305 JMP VOUT70

34EE 05 wrap PUSH HL
34EF D17FF LDL BC, (VLL+1)
34F2 09 ADD HL, h
34F3 34CEEF LDI AsLLLIMi
34F6 45 MOV C,A
34F9 00 LDL 3+0

34FA 03 ADD HL, BC
34FB 0A ADD HL, HL
34FC 342E LDh AsPF6O
34FF F6E0 JR 1110_0000b
3500 3C CMP H
3502 41 DOP HL
3503 2004 05095 JRNZ fixhl

3505 D1000C LDL BC, (24+VLL)
3508 09 ADD HL, BC

3509 3E00 fixhl: LDL AsPF60
350B 84 JR 4
350C 57 MOV H,A
350D 1854 05735 JR VOUT70
Keyboard and Console Routines.

**VC_BELL**: ring the bell via setting PIA3 20h bit

```
050F 3A62EF   LD  A,PIA3
0512 F620   OR  0010_0000b  : bell bit
0514 4F   MOV  C,A
0515 CD9600   CALL DP3D  : function PIA3
0518 3E1E   LDK  A+30  : ring bell for 30 ticks
051A 3258EF   STD  A+BELCNT  : **** = 1/2 second
051D 135C ^3578$   JR  VOUT97  : exit no change
```
Keyboard and Console Routines.

051F VC_CLR3:
0521 2100FD LD X HL,FVAM
0522 CD610D CALL CLRIV
0525 0100DF LDX BCVMEM-VLL
0529 05 PUSH DE
052A 00E1 $ POP IX
052B CE206 CALL VLD1R
052E 3A62EF LD A,PL435
0531 E65D AND n0H1111b
0533 4F MOV C,A
0534 CD8530 CALL DP85
0537 4F XRA A
0538 32E5EF STD A
053B 2100FD LD X HL,FVAM
053E 1833 °3573$ JR VOUT90

:clear 1st line
:clear remaining lines
:Reset for 1st line of display mem
:ZERO A
:SET VERTICAL OFFSET FOR COUT
:Exit
VC_CRI: 0540 CDF606   CALL UN_CUR  ; Erase cursor
        0543 3E3D   LDX A+80h  ; Carriage return
        0545 A5     AND L
        0546 5F     MOV L+A
        0547 182A  *0573#  JR VOUT90

VC_LF: 0549 CD8406   CALL OG_LF  ; Line Feed
        054C 1825  *0573#  JR VOUT90
Keyboard and Console Routines

Move Cursor Right

354E  WC_MCRT

354E  CDF06    CALL  UN_CUR
D551  7E        LD    A+[hl]
3552  130A 355E$  JR    VOUT80  ;re-echo current cur
SIRCIM 808x Assembler ver 3.5E C1/5/57= 21:52 Page 49
Keyboard and Console Routines

5554  VGLAPM:
5554 C95F  $ BIT  5:A
5556 2806 0556: JRZ  VOUT80
5558 C877  $ BIT  6:A
555A 2802 0556: JRZ  VOUT80
555C E69F  ANI  9FH

555E  VOUT80:
;Exit points for COUT
;Here to store new data and to update cursor
555E 77  STD  A:[hl] ;This exit path stores A (new char)
555F 5D  MOV  E+L
5560 C933  $ CBIT  7,E ;IE = old(cursor)
5562 35CEF  LD  A:LLIMIT
5565 3D  DEC  A ;IA = last_legal_char
5566 93  SUB  E ;IA = last_legal_char - old(cursor)
5567 2009 0572: JRNZ  VOUT85 ;if not ALLIMIT
5569 3E80  LDK  A:901
556B 45  AND  L
556C 6F  MOV  L:A ;Do CCR...
556D CD736  CALL  DD_LF2 ;...and LF.
5570 1801 0573: JR  VOUT90

5572 23  VOUT85: INC  HL ;move cursor
;Here if no cursor update
5573 7E  VOUT90: LD  A:[hl] ;This exit path turns on 80h bit
;Here if new data already in A
5574 17  RAL ;Make this char cursor
5575 3F  VOUT96: CMC ;Invert cursor bit
5576 1F  RAR
5577 77  STD  A:[hl]
5578 225AEF  STD  HL+CUR3 ;update cursor
;Here if no change to cursor, restore reg and exit
5578 C1  VOUT97: POP  BC
5579 C1  POP  DE
557A E1  POP  HL
557B E9  MOV  A:C ;Exit with A=chr
557C C9  RET ;return; end of cout subr.

= 04FA ;First = VOUT97 - (127 + 2) ;earliest possible JR
= 050C ;Last = VOUT80 + (128 - 2) ;latest possible JR
580  ESC_LCK:  Block Keyboard

      PROC
580  AF  XOR  A
581  1002  ^5858  JR  :2
583  3EFF  ESC_ULK  LDK  A,OFFn  Unlock Keyboard
585  3259EF  :2;  STO  A,KEYLC
588  19F1  ^5573;  JR  WOUT97
58A E5
58B C05106
58E E1
58F 1852 ^C5735
ESC_CAD: Cursor Addressing

3591  3E07    LDK    A+EF_MSK
3593  AD    AND    B
3594  F018    OR    EF_ESC or EF_ADR    :next chr will be Y-coord

3596  3250EF  :exit3: STD    A+ESCH
3599  18ED ^55784    JR    VNDT97

ESC_SAD: Screen Addressing

3598  3E07    LDK    A+EF_MSK
359D  4D    AND    B
359E  F638    OR    EF_ESC or EF_ADR or EF_SCR
35A0  18F4 ^55964    JR    :exit3
EDELCE
; Delete Character

05A2 E5 PUSH HL ; save cursor_addr
05A3 C0305 CALL CALC ; calculate BC
05A6 E5 PUSH HL
05A7 DE1 $ POP IX
05A8 23 INC HL ; HL = cursor_addr + 1
05A9 C0396 CALL VD1R ; move characters
05A0 3620 STO *(HL) ; last chr becomes blank
05A0 F3 DI
0583 23 ENADIM ; 16th bit memory
0582 3680 REC HL ; HL = last chr on this line
0581 DISDIM ; last char @CHR
058F F3 EI ; main memory
0588 E1 POP HL ; restore cursor_addr
0589 1889 *0573* JR VOUT90 ; next
CALL _UN_CUR
CALL _CALC
LDK _A+27Fh
DR _L
MOV _L+1
PUSH _HL
PDP _IX
DEC _HL
CALL _VLDD1
POP _HL
LD _A+[hl]
RAL
LDK _A++ shr 1
JR VOUT96
0504 "CALC:
/Subroutine for use with EDELC and EINSRT:
//Calculate #chars to move: if move zero chars, never return.
0504 3E7F LDK A0.LL-1 IA = max #chars to be moved
0505 4D MCV C+L
0506 C889 $ CBIT 7+C IC = col cursor
0507 91 SUB C IA = #chars to move
0508 2B04 ^05E0$ JRZ :end if move zero characters
0509 4F MCV C+A
050A 0600 LDX B+O IRC = #chars to move
050B C9 RET
050C E1 :end: PDP HL ;trash return_addr
050D E1 PDP HL ;cursor_addr
050E 1BBF ^0573$ JR VOUT90
05E4       ESCSEQ:
             \ESC-Sequence processing\n
05E4       PROC
05E4       3E01       LDK       A*EF_G2
05E6       1806 ^05EE#       JR       ;125       \ESC-g
             \set graphics mode\n
05E8       3E02       ESCSEcA:       LDK       A*EF_MA
05E9       1802 ^05EE#       JR       ;125       \ESC-l set half intensity
             \igo set flag bit\n
05EC       3E04       ESCSUN:       LDK       A*EF_UN
05ED       3E08       ;125:       JR       B
             \Reg 8 is ESC byte value

05EF       3250EF       ;130:       ;STO       A*ESCH
25F2       1887 ^057B#       JR       VOUT97
             \store desired value\n
25F4       3EFE       ESCCJ:       LDK       A*EF_G2
25F6       1806 ^05FE#       JR       ;140       \ESC-G Clear graphics mode
             \igo clear ESCH bit

25F8       3EFD       ESCCHcA:       LDK       A*EF_MA
25FA       1802 ^05FE#       JR       ;140       \ESC-l Clear half intensity

25FB       3EFA       ESCSUN:       LDK       A*EF_UN

25FE       40         ;140:       AND       B
25FF       18EE ^05EF#       JR       ;130       \igo store ESCH byte

= 051F       ESCZZ:       =       VC_CLRS
             \ESC-Z Clear screen - same as \Control-Z routine\n
0601 ESCR: 
1Delete Line

0601 ESCCE: 
1Insert Line
ENTRY
ITHL = cursor
IC = chr

EXIT
1screen updated
ITHL = new cursor ...to WOUT90

0601 PROC

0601 CDF6D6 CALL UN_CUR
0606 3E80 LODK A+1000_0300b
0606 A5 AND L
0607 5F MOV L+A 1do CR
0608 E5 PUSH HL 1save new cursor
0609 29 ADD HL+HL
060A 3462EF LD A+PIABD
060D 3C18 ADD A+24 IA = addr(25th line)
060F 94 SUB H IA = lines_to_move + 1
0610 261F ANH 0001_1111b :mod 32
0612 47 MOV B+A
0613 3E52 LODK A+VDELL
0615 79 CMP C 1recall #lines to move
0616 79 MOV A+B
0617 2823 *063C% JIRZ :indel 1if deleting a line

1Insert a line

0619 44 :insrt: ADD H IA = addr(25th line)
061A 57 MOV D+A
061B 1ED0 LODK E+D
061C CB1A $ RR D
061F CB18 $ RR E 1shift right DE
0621 1B DEC DE IA = addr(last_char_of_last_line)
0622 7A MOV A+3 IA = #lines to move
0623 2190FF LODK HL+VLL 1HL = addr line above DE
0626 19 ADD HL+DE IA = D or F000h HL := HL or F000h
0627 18D6 *362F% JR tictst

0629 35 :tictst: PUSH DE
062A 00E1 $ POP IX
062C 00ED6 CALL VLD9R 1move 1 line down
062F CD5266 tictst: CALL tvmod
0632 20F5 *0529% JNZ :tictst 1if must move more lines
0634 23 INC HL 1HL => 1st char of new line
Keyboard and Console Routines.

3635  EXIT:  CALL  CLRLE
3638  E1  POP  HL  irecover cursor
3639  C37305  JMP  VOUT90  the Main Exit
363C  01  idact: POP  DE  irecover new cursor
363D  05  PUSH  DE
363E  218000  LDK  HL+VLL
3641  19  ADD  HL+DE  :HL = line_below_cursor
3642  1806  ^364A$  JR  idstrt
3644  25  idcont: PUSH  DE
3645  0DE1  S  POP  IX  move 1 line up
3647  CDE206  CALL  VLDIT
3644  0D9206  idstrt: CALL  tvmod
364D  20F5  ^3644$  JNZ  idcont
364F  E3  EX  HL,DE  :get addr of line to clear
3650  18E3  ^3635$  JR  EXIT
;mod:
;HIL = 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  LDR A, OFFh
0655 34    DR A
0656 67    MOV A, H
0657 3EF0  LDR A, OFFh
0659 82    DR D
065A 57    MOV D, A
065B D18000 LDR BC+VLL
065E F1    POP AF
065F 3D    DEC A    ;decrement line_count
0660 C9    RET
ORG 8038

; Clear to end of line
ENTRY

; ENTRY = Cursor

; EXIT = Clear to EOL

; Uses = A, All

0661 PROC

0662 3620 STD **(hl) ; Clear cursor***

0663 F3 DI

0664 3680 STD @RTBI,(hl) ; Set cursor RIGHT

0665 FB EI

0666 3A5CEF LD A+LLIMIT

0667 30 DEC A ; max_cols => maximum_col_

0668 5D MOV E+L

0670 C888 $ CBIT 7+L

0672 93 SUB E ; IA = col(IA) - col(cursor)

0673 C8 RZ

0674 3D04 ^0674$ JRNC 12 ; if inside logical_video_line

0676 3E80 LDK A+VLL

0678 93 SUB E ; else clr to end of 128-col line

0679 C8 RZ ; if cursor @ column %127

067A 4F :21 MOV C+A

067B 0600 LDK B+O

067D EE PUSH HL

067E D31 $ POP IX

0680 DD23 $ INC IX

0682 185E ^0682$ JR VLDIR
0684 DD_LF:
100 line feed processing
ENTRY

IHL = cursor addr
EXIT

ICursor cleared
IHL updated for current cursor pos
Iwindow moved if necessary

0684 PROC
0684 CDP636 CALL UN_CUR: iclear cursor
0687 E5 DD_LF2: PUSH HL: isave original cursor
0688 018000 LDX BC+VLL: iline length
0688 09 ADD HL, BC: i
068C 3034 *36925 JRC: inwarp: if not wrapping from LAVM to FAVM
068E 0100FO LDX BC+FAVM
0690 09 ADD HL, BC: IHL = new cursor & top of VM
0692 E3 inwarp: XTHL: isave new cursor, get old
0693 29 ADD HL, HL: ishift HL left
0694 3A62EF LD A, PI8D
0697 CS17 ADD A, 23: istart + 23 = last_video_line
0699 94 SUB H: IA = l_line - curr_line
069A E61F AND 0001_1111b: imodulo 32
069C 2B02 *36405 JRZ: tvmov: if cursor is on 24th line of screen
069E E1 tend: POP HL: iget new cursor
069F C9 RET

06A0 3A6CEF tvmov: LD A, LLIMIT
06A3 CB30 SRL L: ilunshift L register
06A5 95 SUB L: IA = LLIMIT - col (cursor)
06A6 38F6 *369E6 JRC: tend: 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 3EB0 LDX A, 30h
06AC 45 AND L
06AD 5F MOV L, A: IHL = beginning of line
06AE CDB136 CALL CLRNL: ierase to EDL
06B1 3A62EF LD A, PI8D
06B4 47 MOV B, A
06B5 E6D AND not 31: IA = line zero
06B7 4F MOV C, A: IC = houseskeeping bits $7 +
06B8 3E1F LDX A, 31
06BA 04 INC B: increment line#
06BB 40 AND B: IA = line#
06BCC 32EFF STS A, WRTOFF: ISET VERTICAL OFFSET FOR OUT
06BF B1 OR C: IA = new line# OR houseskeeping_bits
06C0 4F MOV C, A: IC = new value for D90D
06C1 CDB60D CALL D98D: tmove video screen down 1 line in memory
06C4 E1 POP HL
D6C5 C9  RET
STDDIM:  
ISDOR THE CONTENTS OF THE B REG IN THE ADDR POINTED TO BY THE HL PAIR
ENTRY
IB = VALUE
IHL = ADDRESS
:EXIT
:INVALID

PROC
06C6 F3   DI
06C7 EIADIM  ;ENABLE DIM
06C9 7D   STO B+IHL  ;STORE
06CA F8   DISDIM  ;DISABLE DIM
06CC F9   EI
06CD 29   RET
06CE  WLD01
   :Video Block Move
   :ENTRY
   :BC, IX, HL set
   :
   :

06CE  DDE5  $   PUSH  IX
06D0  01    PDP  DE
06D1  C5    PUSH  BC
06D2  D5    PUSH  DE
06D3  E5    PUSH  HL
06D4  EDB3  $   LDDE  ; main memory
06D6  E1    PDP  HL
06D7  D1    PDP  DE
06D8  C1    PDP  BC
06D9  F3    DI
06DA  ENadin
06DC  EDB8  $   LDDE  ; 9th bit memory
06DE  F8    DISdin
06E0  F8    EI
06E1  C9    RET
Keyboard and Console Routines

 mlx
 :Video Block Move
 :ENTRY
 :IX, IX, HL set

 :EXIT
 :LDIR on main & 9th bit memory
 :Uses BC, DE, HL, IX

 36E2  36E4  36E5  36E6  36E7  36E8  36EA  36EB  36EC  36ED  36EE  36F0  36F2  36F4  36F5
O0E5 $  O1    POP  DE
        POP  DE
  C5    PUSH  BC
  D5    PUSH  DE
  E5    PUSH  HL
  ED80 $  LDIR
        :main memory
  E1    POP  HL
  D1    POP  DE
  C1    POP  BC
  F3    DI
  ENADIM
  ED80 $  LDIR
        :9th bit memory
  DISDIM
  F8    EI
  C9    RET
D6F6
UN_CURR
:Undo/Invert Cursor
:ENTRY
:HL = cursor_addr
:EXIT
:cursor inverted
:
Uses A, CY.

D6F6 7E
LD A, [h1] :get the chr
D6F7 17
RAL :cursor_bit => CY
D6F8 3F
CMC :invert it
D6F9 1F
RAR
D6FA 77
ST0 A, [h1] :...
D6FB C9
RET
KEYBOARD SCANNING & DECODE

;[ ]

06FC
SKEY:
:KEYBOARD INTERRUPT PROCESSOR
:ENTRY
:IN

:EXIT
:KEYBOARD PROCESSING DONE, RESULT IN LOKEY.

06FC
PROC

06FC F3
06FD ED736EFE $ STO SP*3STK ;SAVE INTERRUPTED PROCESS STK
0701 3199EF LOK SP*1STK ;SET TO RAM INT STK
0704 F5
0705 C5
0706 D5
0707 E5
0708 005S S PUSH AF ;SAVE ALL REGESTERS
0709 F3
070A FDE5 S 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, turn off bell.

;CHECK BELL

070C 216BEF LOK HL+3CNT
070F AF
0710 86
0711 280C *071FS OR [hl] ;icell=zero ?
0713 35 JNZ [hl] ;if bell was on
0714 2009 *071FS JRNZ [hl] ;if bell still rings 0x40 instructions yet

;TURN BELL OFF

0716 3A62EF LD A+PIABS
0719 ESDF AND 1101,1111
071B 4F
071C CD660D CALL ODBD
071F 28 11 DEC HL ;HML = HL - 1 DACTIVE
0720 7E
0721 37
0722 2804 *728S OR A
0724 35 JRZ [hl] ;RETURN if inactive

;TURN DRIVE OFF IF DACTIVE = 1

0725 CC060F DEC [hl] ;reset delay
0728 2ACCEF CZ ODRV ;if deselect drive

;UPDATE COUNTER

0728 2ACCEF 12: LD HL+SEQ ;GET LOW TWO BYTES
0728 23 INC HL ;H+1
072C 22CCEF STO HL+SEQ ;STORE
SDICIM 8086 Assembler ver 3.5E c1/55/7= 9:92 Page 68
KEYBOARD SCANNING & DECODE

; READ KEYBOARD

072F 3A59EF LD A, KEYLOCK
0732 87 DR 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, +VID ; clear interrupt
0739 3A08EF LD A, ROMRAM ; IS ROM IN OR OUT?
073C 87 DR A
073D C29300 JNZ ROMJP1 ; if return to RAM

0740 F0E1 $ POP IY ; RESTORE REGISTERS
0742 00E1 $ POP IX
0744 C1 POP HL
0745 01 POP DE
0746 C1 POP BC
0747 F1 POP AF

0748 ED785FEF $ LD SP+IESTK ; get user's stack back
074C FB EI
074D C9 RET
This file contains the 2-key roll over keyboard driver for the JS9010E computer.

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 50Hz 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 processed. If the key is still on it is placed in the list for 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:

KBDRV - Examines the keylist, calls CKEY 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 KBESC to service the key. Calls GSCAN 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.

KBESC - It calls the routine CHKEY to check if shift/ctrl or alphlock keys are on. It then translates the key number into the ASCII code and places it in the LKEY for the C930S 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 7-5: contain the repeat count for the key.
D74E  KBORVR:  
IDENTIFIES AND PROCESSES KEYSTROKES.  
ENTRY  
NONE  
EXIT  
ILKEY = KEYSFOKE

D74E  PROC  

D74E  CD85D7  CALL  KBSCAN  
ISCAN KEYBOARD AND ENTER KEYS INTO KEVLIST  

D751  21D4EF  LDK  *KLKEYLST  
:POINT TO FIRST ENTRY OF KEVLIST  

D754  2603  LDK  3*KLLEN  

D756  3A65EF  ;1:  
LD  A,ILKEY  

D759  37  DR  0  

D75A  E0  RNZ  
:RETURN WHEN A KEY IS WAITING IN ILKEY  

D75B  7E  
LD  A,(HL)  
:GET BYTE OF ENTRY  

D75C  287F  $  
BIT  KL_FORE+4  

D75E  2821  *Q79F$  
JNZ  15  
:IF ENTRY IS IN USE THEN  

D760  C9ED27  CALL  CHKKEY  
:CHECK IF STILL ON  

D763  3004  *Q759S  
JNZ  12  
:IF KEY IS NOW OFF THEN  

D765  3600  
STO  0+(HL)  
:REMOVE KEY FROM LIST  

D767  1B18  *Q731S  
JR  13  

;KEY IS ON, DECREMENT ITS REPEAT COUNT. IF COUNT GOES TO ZERO THEN IT IS TIME TO SERVICE THE KEY.  

D769  EF  ;2:  
PUR  HL  
:SAVE PTR TO FIRST BYTE OF ENTRY  

D76A  23  
INC  HL  
:POINT TO REPEAT COUNT  

D76B  35  
DEC  (HL)  

D76C  2012  *Q730S  
JNZ  14  
:EXIT WHEN NOT TIME TO SERVICE THE KEY.  

;IT IS TIME TO SERVICE THE KEY. SET THE NEXT REPEAT COUNT  

D76E  63  
EX  [SP]+HL  
:POINT BACK TO THE FIRST BYTE OF ENTRY  

D76F  7E  
LD  A,HL  

D770  C9F7  $  
BIT  KY_SVX3+4  
:CHECK IF IT IS SERVICED BEFORE  

D772  C9F6  $  
BIT  KY_SVX+H[L]  
:SET THE SERVICED DATA FLAG  

D774  63  
EX  [SP]+HL  
:POINT BACK TO THE REPEAT COUNT  

D775  3618  
STO  *QPTCT+[HL]  
:AND STORE PTR COUNT AS PER SERVICED FLAG  

D777  2802  *Q773S  
JRZ  13  

D779  3606  
STC  *QPTCT+[HL]  

D77A  63F  ;3:  
AND  KROW_M+KISDL+4  

D77D  C90008  CALL  KBSEV  
:CALL TO SERVICE THE KEY  

D77E  EI  

D780  EF  
POP  HL  
:GET PTR TO FIRST BYTE OF ENTRY AGAIN  

D781  :5:  
ECCH  KLEKEY  
INC  HL  
ENDM
DJNZ 1001 ^07564
DJNZ 1
;UNTIL COMPLETE LIST SCANNED
RET
;RETURN
KBSCAN:
ISCAN KEYBOARD AND ENTER DETECTED KEYS IN THE KEYLIST.
ENTRY
ENTRY
EXIT

KEYLIST = CONTAINS ANY KEYS DETECTED.

PROC

LDK L,OFFH
CALL RDROW
RZ
RETURN WHEN NONE

LDK L,ROW0,M
GET ROW 0

LDK RDROW

AND 111000113
REMOVE CT.,SHIFT AND ALPHA LOCK

LDK B,TOT_ROW

IN THIS LOOP, REG B CONTAINS TOTP, CURRENT ROW BEING SCANNED

JNZ 1B
IIF ANY KEY IS PRESSED THEN

PUSH BC
SAVE LOOP COUNT

MDV E,A
I = COLUMNS

LDK A,TOT_ROW

SUB D
I = ROW NUMBER + 8

RAL
I = 1/2 ROW NUMBER + 8

RAL
I = 1/2 ROW NUMBER + 8

MDV D,A
I = ROW NUMBER + 8

LDK C,0
I = COLUMN COUNTER

ISCAN THIS ROW FROM RIGHT TO LEFT TO GET THE COLUMN NUMBER

SRL E
SHIFT COLUMN BIT INTO CARRY

JNC 17
IIF A KEY IS FOUND THEN

ENTER THE KEY WHOSE COLUMN NUMBER IS IN C AND ROW# IS IN D INTO THE KEYLIST PROVIDED THE KEY IS NOT
ALREADY IN LIST AND THERE IS AN EMPTY SLOT IN THE LIST.

MDV A,D
ADD A,C

PUSH BC
IC = KEY NUMBER

MDV C,A
SAVE DE

PUSH HL
SAVE HL

LDK B,KLEN
LENGTH OF KEYLIST

LDK HL,KEYLIST

LDK DE,0

LD A,[HL]

LD KL_USED+A

JNZ 16
IIF ENTRY IS USED THEN

AND KROW_M*KCOL_M
CHECK WITH CURRENT KEY

JNZ 16
EXIT IF THIS KEY IS IN LIST
S0CIM 808x Assembler ver 3.5E 4/15/7= 9:192 Page 73
S1ROM141 4.5W

J7BE 1802 ^J7C2$  JR $5
J7C0 50  :I: MOV E+L
J7C1 54  :I: MOV D+H
J7C2  :I: ECMD KLE_LEN
         INC  HL
         ENDM
J7C4 10EE ^J79A$  DJNZ $3
         I:STILL LIST SCANNED
         ;CHECK IF AN EMPTY ENTRY WAS FOUND.
J7C6 7A  MOV A+D
J7C7 87 JR A
J7C8 2807 ^J7D1$  JRZ $6
J7CA EB  EX DE,HL
J7CB 71  STD C*(HL)
J7CC CBFE $  SBIT KL_USED*(HL)
J7CE 23  INC HL
J7CF 3601  STO DB,CT*(HL)
J7D1 E1  :I: PDP HL
J7D2 D1  PDP DE
J7D3 C1  PDP BC
J7D4 DC  :I: INC C
J7D5 AF  XOR A
J7D6 B8  CMP E
J7D7 209 ^07A2$  JRNZ $2
J7D9 C1  PDP BC
J7DA CB25 $  :I: SLA L
J7DC C00008  CALL RDROW
J7DF 1084 ^0795$  DJNZ $1
J7E1 C9  RET
DTE2

CHKEY:

;CHECKS IF KEY NUMBER IS ON.
ENTRY
IA = KEYNJMBER
EXIT
IZ CLR = KEY IS OFF.
IZ SET = KEY IS ON.

DTE2

PROC

DTE2 E5
DTE3 F5
DTE4 IF
DTE5 IF
DTE6 IF
DTE7 CDF607
DTE8 F1
DTE9 D5
DTEC CDF607
DTEF S1
DTE0 CDD008
DTE3 A3
DTE4 E1
DTE5 C9

PUSH HL
PUSH AF
RAR
RAR
RAR
CALL GTMASK
CALL GTMASK
POP AF
PUSH DE
CALL GTMASK
POP HL
CALL RDRW
AND E
POP HL
RET

;SAVE CALLERS HL
;SAVE KEYNJMBER
;RIGHT JUSTIFY ROW NJMBER
;GET KEY NJMBER
;SAVE ROW MASK
;GET COL MASK (COL NJM IS IN BITS 0..2)
;SAVE ROW MASK TO L
;GET ROW OF KEYS ADRESSED BY L
;IZ IND = VALUE OF KEY
;
8000       PROC
8000  2622  LDK  M*HIGH(M*KEY)  ; HL = PRT ADRS FOR GIVEN ROW
8002  70    MOV  A+L
8003  5D4F  $    MOV  R+A
8005  7E    LD   A*(HL)
8006  EEFF  XOR  DFFH  ; INVERT VALUES
8008  C9    RET
KEYBOARD SCANNING & DECODE

;SDCIM 808x Assembler ver 3.5E <1/55/79> Page 77

= 008D  LFT_ARW = 8DH
= 008E  RT_ARW = 9BH
= 008A  JP_ARW = 8AH
= 008C  DN_ARW = 8CH
= 0053  HM_SCRN = 'F'

0809  KBSEXV:
      ;SERVICES THE KEY
      ;ENTRY
      ;IA = KEYNJM9ER
      ;[SP]-4 = POINTER TO KEYLIST ENTRY (USE FOR SLIDE FNC ONLY)
      ;EXIT
      ;INDNE
      ;PRESERVES REG B

0809  PROC

;SETUP HL TO POINT TO KEYPAD TABLE ENTRY FOR THIS KEY

0809  5F  MOV  E,A
080A  1600  LDK  D+O
080C  210408  LDK  HL,KYCDTB
080F  19  ADD  HL,DE
0810  7E  LD  A,[HL]
0811  FE21  CMP  *+1
0813  3834  JRC  KEYE  ;IGNORE SHIFT/CTL ETC FOR CHAR LESS THAN 31

0815  F5  PUSH AF
0816  2E01  LDK  L+1
0818  CDO008  CALL  RDR0W  ;GET ROW CONTAINING CTL,SHIFT AND ALPHA KEY

0818  F5  PUSH AF
081C  2E80  LDK  L,80H
081E  CDO008  CALL  RDR0W
0821  6008  ANI  B
0823  5F  MOV  E,A
0824  F1  POP AF
0825  83  OR  E
0826  5F  MOV  E,A
0827  F1  POP AF  ;RESTORE KEYCODE

0828  CB53  $  BIT  CTL_KY+\E
082A  202E  *095A  JNZ  KEY4  ;GO PROCESS CTL KEY
082C  CB63  $  BIT  SHFT_KY+\E
082E  2014  *0844  JNZ  KEY5  ;GO PROCESS SHIFT KEY
0830  CB58  $  BIT  ALPH_KY+\E
0832  2004  *0838  JNZ  KEY1  ;GO PROCESS ALPHA KEY

;FALL THROUGH TO "KEYE"
KEYE:
:STORE KEY CODE INTO "LKEY" AND RETURN

DB34  325EEF STO A\LKEY
DB37  C9   RET
KEYBOARD SCANNING & DECODE

0830  0E1:
0831  FE51    CMP  'a'    I PROCESS ALPHA KEY
0833  38F8  ^0834$  JRC  KEYE    EXIT WHEN LESS THAN 'a'. ALPHA HAS NO EFFECT
083C  FE0D    ;27:    CMP  80H
083E  30F4  ^0834$  JRNc  KEYE    FOR WHEN >= 80H
0840  EE20    ;28:    XOR  20H    ;FOLD CHAR TO UPPER CASE
0842  18F0  ^0834$  JR  KEYE
<table>
<thead>
<tr>
<th>Address</th>
<th>Key</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0844</td>
<td>FFS1</td>
<td>CMP 'a'</td>
<td>:PROCESS SHIFT KEY</td>
</tr>
<tr>
<td>0844</td>
<td>30F4</td>
<td>JRC :27</td>
<td>:GOTO ALPHA WHEN CHAR &gt; 'a'</td>
</tr>
<tr>
<td>0844</td>
<td>FFS8</td>
<td>JMP FFS8</td>
<td></td>
</tr>
<tr>
<td>0844</td>
<td>3806</td>
<td>JRC KEY3</td>
<td>:GOTO PROCESS SHIFT NUMERICS ETC</td>
</tr>
<tr>
<td>084C</td>
<td>20F2</td>
<td>JRNZ :28</td>
<td>:INVERT SHIFT BIT FOR ''</td>
</tr>
<tr>
<td>084E</td>
<td>3E50</td>
<td>LDK A0''</td>
<td></td>
</tr>
<tr>
<td>0850</td>
<td>18E2</td>
<td>JR KEY5</td>
<td>:CONVERT ( T3 )</td>
</tr>
</tbody>
</table>

Note: The key sequence 'A0' and '?' are specific to the input and output in this context.
KEYBOARD SCANNING & DECODE

DB52  KEY3:
    1CHARS * T> > (ASCII CODES 27H TO 3EH) ARE CONVERTED USING
    THE SHFT_TB. D=0 FROM BEFORE

DB52  5F
DB53  21E508
DB56  19

DB57  7E  KEY3A:  LD  A,[HL]
DB58  180A *DB34$  JR  KEYE
DB5A

HIGH: PROCESS CONTROL KEY
IF CHAR IS BETWEEN A-Z THEN TURN OFF THE 3 HIGH ORDER
BITs.
IF CHAR IS BETWEEN '0'..'9' IT IS TRANSLATED AS PER TABLE CTL_T09
IF CHAR IS THE ARROW KEYS OR THE 'Tab' KEY THE SLIDE FUNCTIONS
ARE CALLED.

DB5A  FEB3  CMP  LEFT_ARROW
DB5C  2833  JRZ  SLIDE_LEFT
DB5E  FEB8  CMP  RIGHT_ARROW
DB60  2833  JRZ  SLIDE_RIGHT
DB62  FEBA  CMP  UP_ARROW
DB64  2840  JRZ  SLIDE_UP
DB66  FEBC  CMP  DOWN_ARROW
DB68  2840  JRZ  SLIDE_DOWN
DB6A  FE53  CMP  HOME
DB6C  2850  JRZ  DOHOME

DB6E  CB63  $  BIT  SHIFT_KEY  ITEST FOR CNTRL SHIFT
DB70  2808  JRZ  KEYS  IIF NOT

DB72  FE26  CMP  '*'  IIS IT ?
DB74  2004  JRNZ  KEYS  IIF NOT '?'

DB76  3E7F  LDX  A+DFTMH  IDELETE KEY
DB78  188A  JR  KEYE
KEYBOARD SCANNING & DECODE

087A    087A    FE40
      CMP      'G'

087E    0880    0886$ JRC      KEYE
       JMP      'Z'+1
       GOTO    TRANSLATE CHARs FROM TABLE

0882    0884    0834$ AND      IFH

0884    0884    0834$ JR       KEYE
KEYBOARD SCANNING & DECODE

3886       KEY6:
3886 FE2C  CMP    "*"
3888 38AA  *3834$  JRC    KEYE
             INQ TRANSLATION IF CHAR BELOW "*"
388A 21F808  LDK    HL,CTL_T0-"*"
388D 5F    MOV    E,A
388E 19    ADD    HL,DE
388F 18C6  *3857$  JR     KEY3A
SLIDE FUNCTIONS:

SLIDEL:
0891  SLIDEL:   LDK   E+2
0891  0ED2
0893  1902 ^0897$  JR   SL41

SLIDER:
0895  SLIDER:   LDK   E-2
0895  0EFE
0897  3461EF  SL21:  LD   A+PIAAD  ;GET HORIZONTAL COORD.
0899  81    ADD   A+C
089B  4F    MOV   C+A
089C  C07900  CALL  DPAD  ;FUNCTION PIA

;SET REPEAT COUNT FOR THESE KEYS (OVERRIDE COUNT SET BY THE KBDRVR)

089F  D1    SL22:  PDP   DE  ;GET RETURN ADDR
08A0  E1    PDP   HL  ;POINTER TO REPEAT ENTRY
08A1  3603  STD   SLD,REG*(HL)  ;REPEAT COUNT FOR SLIDE KEYS
08A3  E5    PUSH  HL
08A4  D5    PUSH   DE  ;RESTORE STACK
08A5  C9    RET
DBA6 DE01 SLIDE: LDK C=1
DBA8 1802 ^DBAC$ JR SLD1
DBAA DEFF SLIDE: LDK C=-1
DBAC 2152EF SLD1: LDK HL PIABD \MERGE NEW VERT OFFSET TO LOWER 5 BITS OF PIAB
DBAF 7E LD A+[HL]
DBB0 81 ADD A+C
DBB1 E61F AND 1FH \MODIFY CURRENT WITH +1/J2-1
DBB3 4F MOV C+A
DBB4 7E LD A+[HL]
DBB5 E650 AND DE0H
DBB7 01 SLD2: OR C
DBB8 4F MCV C+A
DBB9 CDBE60 CALL OPBD
DBBC 1801 ^DBEF$ JR SLR2
;SET DENSITY BIT

DBE 3451EF  LD A,PIAAD ;GET OLD VALUE
DBE 3451EF  ANI 0000_0001B ;SAVE DENSITY BIT
DBE 3451EF  DRI VFLO ;FOR IN HORIZONTAL OFFSET
DBE 3451EF  MDV C,A
DBE 3451EF  CALL PAD ;FUNCTION PIA
DBE 3451EF  LD A,PIAAD
DBE 3451EF  AND ODOM ;HOUSE KEEPING BITS
DBE 3451EF  MDV C,A
DBE 3451EF  LD A,VATOFF ;GET LAST VERTICAL OFFSET
DBE 3451EF  JR SL02 ;AND THE VERT TO 0 ALSO
### KEY CODE TRANSLATION TABLES

<table>
<thead>
<tr>
<th>KEY</th>
<th>CODE</th>
<th>DB</th>
<th>ASC</th>
<th>TAB</th>
<th>RCR</th>
<th>ERC</th>
<th>ELC</th>
<th>ELE</th>
<th>CR</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>38D4</td>
<td>18397FF</td>
<td>38D4</td>
<td>esc, tab, rcr, erc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38DA</td>
<td>7F0D2753</td>
<td>DB</td>
<td>esc, cr, ******, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38DB</td>
<td>31323334</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38D8</td>
<td>5363738</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38DC</td>
<td>71775772</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38E0</td>
<td>54797569</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38E8</td>
<td>61755576</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38F0</td>
<td>7585558</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38F4</td>
<td>7478576</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38F8</td>
<td>5266602C</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38FC</td>
<td>3A00302D</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3900</td>
<td>2E706F39</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3904</td>
<td>493C252F</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3908</td>
<td>395C253D</td>
<td>DB</td>
<td>******, FF, FF, FF, FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SHIFT**

<table>
<thead>
<tr>
<th>CODE</th>
<th>DB</th>
<th>FF, FF, FF, FF, FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>390C</td>
<td></td>
<td>FF, FF, FF, FF, FF</td>
</tr>
<tr>
<td>3911</td>
<td>325F3F29</td>
<td>FF, FF, FF, FF, FF</td>
</tr>
<tr>
<td>3916</td>
<td>2140232425</td>
<td>FF, FF, FF, FF, FF</td>
</tr>
<tr>
<td>3918</td>
<td>52622A2800</td>
<td>FF, FF, FF, FF, FF</td>
</tr>
<tr>
<td>3920</td>
<td>3A002500</td>
<td>FF, FF, FF, FF, FF</td>
</tr>
<tr>
<td>3924</td>
<td>791F707E</td>
<td>FF, FF, FF, FF, FF</td>
</tr>
<tr>
<td>3928</td>
<td>8081828334</td>
<td>FF, FF, FF, FF, FF</td>
</tr>
<tr>
<td>392D</td>
<td>8586978889</td>
<td>FF, FF, FF, FF, FF</td>
</tr>
<tr>
<td>3932</td>
<td>00000050</td>
<td>FF, FF, FF, FF, FF</td>
</tr>
</tbody>
</table>
SRCM 808x Assembler ver 3.5E <1:55/7> 9:92 Page 89
IEEE-488 INTERFACE.

# [ ]

I ENERED 05/01/81 FROM TYN SHERON. SEM. I

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 CCA
# 10 PERIPHERAL/DIRECTION REGISTER B CPDRB
# 11 CONTROL REGISTER B CCB

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.
IT WILL SET P0-P17, PD0-PD7, C24, AND C32 AS INPUTS,
AND ALL INTERRUPTS DISABLED.
SIGNS 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 (PD0) 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.
STATE CONTROL CAN ONLY BE CALLED FOLLOWING A GO TO STANDBY.
AFTER A FATAL ERROR, PERFORM AN IFC RESET.

STANDARD VALUES USED:
CRA 0D11(IFC)0D11010
CCRA 0D11(REN)0D1100
CPDRA SOURCE DIRECTION 1111_1111
CPDRA DIRECTION DATA 1111_1111
CPDRA ACCEPTOR DIRECTION 1111_1111
CPDRA DATA 1111_1111
CPDRB SOURCE DIRECTION 0011_1111
CPDRB DATA 0000_0110 1A = ATN
CPDRB ACCEPTOR DIRECTION 1110_0111
CPDRB DATA 0110_0110
IOIA SIGNAL DEFINITIONS:
ALL SIGNALS ARE LOW ON THE IEEE BUS WHEN VIA 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")
P11  ENABLE NDAC/NRFD  (ENABLED WHEN "0")
P32  ENABLE EDI/DAV  (ENABLED WHEN "0")
P54  ATN
P55  DAV
P56  NDAC
P57  NRFD

: CB1  NOT USED
: CB2  REN

; CONTROL WORD FORMAT
[(7) (6) (5) (4) (3) (2) (1) (0)]
[(IRQA1) (IRQA2) (CA2 CONTROL) (DDRA) (CA1 CONTROL)]
[(IRQA1) (IRQA2) (CA2 CONTROL) (DDRA) (CA1 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  \# DATA DIRECTION REGISTER A
: 1  \# 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
: DDRA 0  \# DATA DIRECTION REGISTER B
: 1  \# PERIPHERAL REGISTER B
: CB1 00  NOT USED
BIOS CALL: CONTROL 3UT

: 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 "I", THE IFG 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
SERCIM 8086 Assembler ver 3.5E <5/57> 89:92 Page 22

IEEE-488 INTERFACE.

IECO:
PRO
PUSH AF
PUSH HL
BIT 0x8C CHECK IFC SUB-COMMAND
JRZ 191C20

INITIALIZE ALL IEEE-488 SIGNALS

LJ ML+CCRA
STO 0011.1010F,HL ENABLE SRQ AND SET IFJ-JIT LOW

LJ A.1111.11113 DIRECT DATA OUT
STO A+CCDR

LJ A.1111.11113 SET REW-JIT HIGH
STO 0011.01009,HL

LJ A.1111.11113 DIRECTION FOR SOURCE HANDSHAKE
STO A+CCDR

LJ A.1111.01109 VALUES FOR SOURCE HANDSHAKE
STO A+CCDR

IDFAV IFC LOW FOR 100 MICRO-SEC

LJ A.25 IDELAY 100 MICRO-SEC

B1C10: DEC A
JRZ# B1C10

LJ A.0011.01109 SET IFC HIGH
STO A+CCDR

LJ A.0011.01009 SET/CLEAR REW

LJ A.0011.10009
STO A+CCDR

LJ A.0011.11009
STO A+CCDR

P09 HL
P09 AF
RET
IE Thing

;IO105 CALL 2, STATUS IN
;CAN BE CALLED ONLY WHILE IN SOURCE HANDSHAKE MODE.
;BIT 0 OF REGISTER A SET IF SRQ IS LOW

PROC

E5          PUSH HL
3A0029      LD A,$0029  ;CLEAR IO101
210229      LK HL,$0229  ;PULSE ENABLE ndsc/n+f4
CB0E        $          CBIT 1$HL
CACE        $          SBIT 1$HL
3A0129      LD A,$0129  ;SET SRQ VALUE IN A
683D        AND 1000_0000B
07           RLC A
81          POP HL
09           RET
IE,STS:
READY CALL 3 GO TO STANDBY
CAN BE CALLED ONLY WHILE IN SOURCE HANDSHAKE MODE
ENTRY
NONE

098C       PROC
098C  F5      PUSH  AF
098D 3EO2      LK  A,0000_0010B  ISET ATN HIGH
098F 32D219     STO  A,CPRDB
0992 AF      XDR  A  IFLCAT DATA BUS
0993 32D229     STO  A,CPRDA
0996 F1      POP  AF
0997 C9      RET
SOL C808 Interface

IE.TC1
IBIOS CALL 4 TAKE CONTROL
CAN BE CALLED ONLY WHILE IN THE CONTROLLER STANDBY STATE (ATN HIGH).
EXIT IN THE CONTROLLER ACTIVE STATE (ATN LOW), SOURCE HANDSHAKE MODE.
BIT 0 OF REGISTER C SET TO TAKE CONTROL ASYNCHRONOUS

EXIT = ERROR CODE

IE.TC1

IE.TC1

IE.TC1

IE.TC1

IE.TC1

IE.TC1

IE.TC1
IE.OIM:
:BIOS CALL 5 OUTPUT INTERFACE MESSAGE
:CAN BE CALLED WHILE IN ANY MODE OR STATE.
:EXIT IN THE SOURCE HANDSHAKE MODE WITH ATN LOW.

:EXIT
IA  =  ERROR CODE
IC  =  MULTI-LINE MESSAGE

D9D8

PROC

D9D8  E5
PUSH  HL
D9D9  210229
LD  HL, CPDRB
D9DC  C8E6  $SBIT  A[R][HL] ;SET ATN LOW
D9DE  C346  $BIT  0[R][HL]
D9E0  2825  "DA07"
JRZ  IE*SHK

:SET-UP FOR SOURCE HANDSHAKE
D9E2  3617
STD  0001_0011B[R][HL] ;DISABLE DRIVERS
D9E4  3A0329
LD  A,CCR8
D9E7  CB97  $CBIT  2[A]
D9E9  320329
STD  A,CCR8
D9EC  363F
STD  0011_1111B[R][HL] ;DIRECTION REGISTER
D9EE  CB07  $SBIT  2[A]
D9F0  320329
STD  A,CCR8

:FLOAT EXTERNAL DATA BUS
D9F3  AF
XOR  A
D9F4  320029
STD  A,CPDR
D9F7  3612
STD  0001_0010B[R][HL] ;CONTROL SIGNAL INITIAL VALUE
D9F9  180C  "DA07"
JR  IE*SHK
IE.ODM:
FRSIS 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
IC = MULTI-LINE MESSAGE
IB = EOI REQUEST
IA = ERROR CODE

D9FB
PROC
D9FB 85
PUSH HL
D9FC 210229
LK HL,CPDRB
D9FF C3A6 $ SBIT 4,[HL] ;SET ATN HIGH
DA01 CB40 $ SBIT 0,[HL] ;CHECK IF EOI REQUESTED
DA03 2802 ^DA078
JRZ IE,SHK
DA05 CRDE $ SBIT 3,[HL]

:PERFORM SOURCE HANDSHAKE
DA07 CB6E $ IE,SHK: SBIT 5,[HL] ;
DA09 2018 ^DA268
JNZ 1B6C50 ;DTC TIMEDJT RE-ENTRY
DA0B 79
MOV A,C
DA0C 320029
STD A,CPDRB
DA0F 3ED8
LK A+10
DA11 CB7E $ 1B6C30: SBIT 7,[HL]
DA13 2807 ^DA1CS
JRZ 1B6C30 ;READY FOR DATA
DA15 3D
DEC A
DA16 20F9 ^DA115
JNZ 1B6C20 ;WAIT FOR 100 MICRO-SEC
DA18 3E82
LK A+1000_02108 ;SET RFD TIMEDJT ERROR
DA1A 1818 ^DA388
JR 1B6C80
DA1C CB76 $ 1B6C30: SBIT 6,[HL]
DA1E 2004 ^DA248
JNZ 1B6C40 ;DATA ACCEPTED LOW
DA20 3E81
LK A+1000_03018 ;SET DEVICE NOT PRESENT ERROR
DA22 1817 ^DA398
JR 1B6C90
DA24 CB8E $ 1B6C40: SBIT 5,[HL] ;SET DAV LOW
DA26 3E6E $ 1B6C50: LK A+255
DA28 CB7E $ 1B6C40: SBIT 6,[HL]
DA2A 2807 ^DA338
JRZ 1B6C70 ;DATA ACCEPTED
DA2C 3D
DEC A
DA2D 20F9 ^DA288
JNZ 1B6C50 ;WAIT 1000 MICRO-SEC
DA2F 3E84
LK A+1000_01029 ;SET DAC TIMEDJT ERROR
DA31 1808 ^DA388
JR 1B6C80
DA33 CB8E $ 1B6C70: SBIT 5,[HL] ;SET DAV HIGH
IEEE-488 INTERFACE

DA35 CB9E $ CBIT 3,[HL] SET EOI HIGH
DA37 AF XOR A IREAD DATA FROM BUS
DA38 320029 STD A,CPDR
DA3B E1 #B6C8D0# POP HL
DA3C 9 RET
DA30
IE+SMH:
IBIOS CALL 7 INPUT DEVICE MESSAGE
CAN BE CALLED WHILE IN ANY MODE OR STATE
EXITS IN THE ACCEPTOR HANDSHAKE MODE WITH ATN HIGH.
EXIT
IL = IR30R CODE
IA = DEVICE MESSAGE
IH = DEVICE MESSAGE

DA30 PROC
DA30 D5
PUSH DE
DA3B EB
EX DE:HL  ;SAVE RE-ENTRY DATA
DA3D 210229
LK HL:CPDR8
DA3F CB46
$ BIT 0,[HL]
DA41 291A *DA40S
JNZN *B7C10

;SET-UP FOR ACCEPTOR HANDSHAKE
DA46 3617
STD 0001_0111B,[HL] IDISABLE DRIVERS
DA48 3A0329
LD A,[CC8B]
DA4A CB97
$ CBIT 2,A
DA4D 320329
STD A,[CC8B]
DA50 3607
STD 1110_0111B,[HL] IDIRECTION REGISTER
DA52 CB07
$ SBIT 2,A
DA54 320329
STD A,[CC8B]
DA55 3EEF
LK A,1111_1111B ;FLOAT INTERNAL DATA BUS
DA59 320029
STD A,CPRA
DA5C 3655
STD 0101_0101B,[HL] ICONTROL SIGNALS INITIAL VALVE
DA5E 3645
STD 0100_0101B,[HL] ISET ATN HIGH

;PERFORM ACCEPTOR HANDSHAKE
DA60 CB76
$ BIT B7C10: 6,[HL]
DA62 2820 *DA84S
JRZ *B7C50  ;DATA INVALID TIMEDOUT ERROR RE-ENTRY
DA64 CB9E
$ CBIT 7,[HL]
DA66 3E0A
LK A,10
DA68 CB9E
$ BIT B7C20: 5,[HL]
DA6A 2008 *DA74S
JNZN *B7C30  ;DATA VALID
DA6C 30
DEC A
DA6D 20F9 *DA68S
JNZN *B7C20  ;WAIT 100 MICRO-SEC
DA6F 118230
LK DE+1000_0010B  ;ISET DATA VALID TIMEDOUT ERROR
DA72 1821 *DA95S
jr *B7C40
DA74 CB9E
$ BIT B7C30: SBIT 7,[HL]
DA76 3A0029
LO A,[CPRA]
DA79 57
MOV D,A
DA7A 1E00
LK E,0  ;READ EDI
DA7C CB5E
$ BIT 3,[HL]
DA7E 2802 *DA82S
JRZ *B7C40
SBCIM 808x Assembler ver 3.5E c1/55/7* ©9192 Page 100
81ROM141-ASM

IEEE-488 INTERFACE.

DA80 1E01 LK E+1
DA82 CBB6 $:87C40: SBIT 6+HL ISET NDAC HIGH
DA84 3EFF :87C50: LK A+255
DA86 3B5E $:87C50: SBIT 5+HL IDATA VALID DROPPED
DA88 2809 ^DA93% JRZ :87C70 IDATA Valid DROPPED
DA8A 3D DEC A
DA88 20F9 ^DA86% JRNZ :87C50 IWAIT 1000 MICRO-SEC
DA8D CBB3 $ SBIT 2+6 ISET DATA INVALID TIMEOUT ERROR
DA8F C3F8 $ SBIT 7+6 ISET DATA INVALID TIMEOUT ERROR
DA91 1802 ^DA95% JR :87C80
DA93 :3F6 $:87C70: SBIT 6+HL ISET NDAC LOW
DA95 EB :87C80: EX DE,HL IMOVE RESULTS TO REGISTERS A AND HL
DA96 7C MOV A,H
DA97 D1 PDP DE
DA98 C9 RET
DA99
IE_PP:
IBIOS CALL 8 PARALLEL POLL
SCAN BE CALLED ONLY WHILE IN THE SOURCE HANDSHAKE MODE WITH ATN HIGH OR LOW.
EXIT IN THE SOURCE HANDSHAKE MODE WITH ATN LOW.

:EXIT
IA = PARALLEL POLL VALUE

DA99
PROC

DA99 E5
PUSH HL

DA9A 210029
LK HL,CPORA

DA9D 3E18
LK A+0001_1011B
IFORM PARALLEL POLL

DA9F 320229
STD A+CPDRB

DAA2 36FF
STD 1111_1111B+{HL} #FLOAT INTERNAL DATA BJS

DAA4 7E
LD A+[HL]
#READ PARALLEL POLL DATA

DAA5 3600
STD D+[HL]
#RE-STORE SOURCE HANDSHAKE MODE

DAA7 210229
LK HL,CPODB

DAAA 3612
STD 0001_0010B+{HL}

DAAC E1
POP HL

DAAD C9
RET
IEEE drivers:

The routines IIEINSTAT, 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 call IIEADRS.

The function IIEINSTAT 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 IIEINSTAT many times before calling IEINP to read a char, and IIEINSTAT has to read the char to determine the status, the character read has to be buffered until call to IEINP is made. IIEINSTAT 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.
DAAE

IEESTAT:
returns status of IEEE
IEEE always appears to be ready

DAAE
PROC

DAAE F6FF
       CPI OFFh

DA8D C9
       RET
DA81
IEINSTAT:
gets status of the input device attached to IEEE port
if a char is present in IE_char then return with OFFH status
else
make device talker
Read the device
if char read then
store in bfr
make untalk
return with status of buffer

DA81 PROC
DA81 BDACEF  LDA IE_CHAR
DA84 87  ORA A
DA85 2803  *DA9A$  JRZ IEI10  ;if char present then
DA87 F6FF  DAI OFFH  ;return with OFFH status
DA89 C9  RET
S0C3IM 808x Assembler ver 3.5E 3:15 =9:32 Page 105

IEEE-488 INTERFACE.

DABA
DABD
DABF
DAC0
DAC3
DAC4
DAC6
DAC9
DACB
DACD
DACE
DAD1
DAD1
DAD3
DAD6
DAD7
DAD9
DADC
DADD
DADE
DAEO
IEI10:
: Make Talker
LDA IE_ADDR
ADI IE_TALK
: Get Primary Address
MOV C+IA
: Output Interface Message
CALL IE_OIM
ORA A
: Try again if error
JNZ IEI10

IEI20:
: Read a char
CALL IE_IDM
BIT T+L
JRZ IEI30
: If error then
XRA A
: Indicate no char recvd
STA IE_CHAR
: Store the char

IEI42:
: Make UnTalk
LDK C+IE_UTLC
CALL IE_OIM
ORA A
JRZ IEI42
: Return with status of the char
LDA IE_CHAR
DRA A
RZ
: DRI OFFh
RET
DAE1  IEINP:
;Reads a character from IEEE port

DAE1  PROC

DAE1  CD310A
DAE4  28FB  ^DAE1#

CALL  IEINSTAT
JRZ   IEINP

LDK   HL,IE_CHAR
LD    A,IE_CHAR
STO   D,[HL]
RET   D,[HL]

;wait till char avail

DAE6  21DCEF
DAE9  7E
DAEF  3600
DAEC  59

;clear the buffer
DAED

IEOUT:
:Outputs the character in reg C to IEEE port
:Uses ROM resident primitives.

DAED

PROC

DAED C5

PUSH BC

; save the char
; make listener

DAEE 3A08EF IE005:
LDA IE_ADRS

; get primary address

DAF1 0620
ADI IE_LSTN

; output interface message

DAF4 8309
CALL IE_JIM

; try again if error

DAF8 20F4 "DAEE"
JRNZ IE005

; do not send echo

DAFD C5

PUSH B

; save char again in case of retry

DAFE 00809
CALL IE_JIM

DB01 C1

PDP B

DB02 87

ORA A

DB03 20F8 "DADF"
JRNZ IE022

; try again if error

DB05 IE40:

; make unlisten

DB05 DE3F

LDK IE_ULST

DB07 8309
CALL IE_JIM

DB0A 87

ORA A

DB0B 20F8 "DB05"
JRNZ IE040

DB0D C9

RET
The parallel port is actually the IEEE port driven with the centronix protocol. The bit assignments of the PIA and PIB are as follows:

- SPIAO-7 = data bus
- SPIB0 = 0, data bus is output. 1, data bus is input
- SPIB1 = set to 1.
- SPIB2 = set to 0.
- SPIB3 = 0 output. 1 input
- SPIB4 = not used
- SPIB5 = output strobe. Active = 1.
- SPIB6 = 0, printer busy. 1, printer is ready.
- SPIB7 = not used.

IC42 = going low indicates to device that we are busy.
IC41 = 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 postat or parin are called.
CV2OP:
Initialize the port to a Parallel output port.

OB0E PROC
OB0E 3A5EEF LOA PP_MODE
OB01 FED1 CPI PP_OUT
OB13 C8 RZ ; Return when in output mode

; Set port a to output on all lines
OB14 3F2A LDK A+PA+DIR
OB16 320129 STA PA+CTL ; Select direction reg
OB19 3EFF LDK A+PA+DIR
OB18 320029 STA PA+DIR ; Output constant to dir reg to put a port in output mode

OB1E 3E2E LK A+PA+CDT
OB20 320129 STA PA+CTL ; Select port a data reg

OB23 3E00 LK A+PB+DIR
OB25 320329 STA PB+CTL ; Select port b direction
OB28 3E3F LK A+PB+DIR
OB2A 320229 STA PB+DIR ; All lines are output except the output busy signal on bit 6

OB2D 3E04 LK A+PB+CDT
OB2F 320329 STA PB+CTL ; Select data register
OB32 3E02 LK A+PB+DTO
OB34 320229 STA PB+DATA ; Initialize port b data

OB37 3E01 LK A+PP+OUT
OB39 320EEF STA PP_MODE
OB3C C9 RET
Sorbim 808x Assembler ver 3.1F <05/5/73 Page 113
IEEE-488 INTERFACE.

0330 PROC
0340 3A3EFF LDA PP\$MODE
0340 FEO2 CPI PP\$IN
0342 C9 RZ

; Initialize the port to a parallel input port

0353 LEAD

; Set port a to input on all lines

0354 3E2A LK A\$PA\$DIR
0354 3E29 STA PA\$CTL

; Select direction reg

0355 3E2D LK A\$PA\$DIR
0355 3E29 STA PA\$DIR

; Output constant to dir. reg to put a port in input mode

0356 3F4E LK A\$PA\$DIR
0356 3F49 STA PA\$CTL

; Select port a data reg.

0357 3E30 LK A\$PB\$DIR
0357 3E39 STA PB\$CTL

; Select port b direction

0358 3E3F LK A\$PB\$DIR
0358 3E39 STA PB\$DIR

; All lines are output except the output busy signal on bit 5

0359 3E44 LK A\$PB\$DIR
0359 3E49 STA PB\$CTL

; Select data register

035A 3E50 LK A\$PB\$DIR
035A 3E59 STA PB\$DATA

; Initialize port b data

035B 3E62 LK A\$PP\$IN
035B 3E69 STA PP\$MODE

RET
DB6C
POS11:
: tests status of the parallel (centronics) printer attached to the IEEE port

DB6C
PROC

DB6C C0E03
CALL CW20P
: convert to output

DB6F 3A0229
LDA PB.OTA
: test port a data

DB72 6640
AND PP.OTA
: test port A data

DB74 59
RZ

DB75 F6FE
DRI OFFH

DB77 59
POS12: PET
PISTAT:

Igats status of the input device attached to the parallel port

PROC

CALL C8ZIP
LDA PIACRL
ANI PP122Y
JRNZ PIS20

LDA PAACRL
STA PIACRL

ANI PP122Y
RZ

PIS20: BRI OFFH

RET
PARINP: Inputs a character from Parallel port.

PARINP PROC

CD780B
28FB "338E"

CALL PISTAT
JRZ PARINP

XRA A
STA PIACTL
LDA PIA::TA
CMA WDV C::A
SET C::A

IEEE-488 INTERFACE.
PAROUT: Outputs the character in c to the IEEE port treating the port as a parallel port.

```assembly
DB9D PROC
    CALL POSTAT
    JRZ PAROUT
    MOV A,C
    CMA
    STA PAORTA
    LK A+PAORTA STRB
    STA PBORTA
    LK A+PBORTA
    STA PBORTA
    SET PBORTA
```

invert data

set strobe

clear strobe
@5

OBA2 8157  MASTER reset S10
OBA4 32002A  A×S1×MST
OBA7 79  MOV  A×C
OBA8 32C1EF  STD  A×ACIAC
OBA8 32002A  STD  A×MSCTRL
OBAE C9  RET
DBBF  LEADER:
   ;Input one byte from reader port
   ;ENTRY
   ;None
   ;EXIT
   IC  = character read

DBBF       PROC

DBBF      CDE8DB     CALL      ACISTAT
DBC2      E601       ANI      SI, RDY
DBC4      28F9  DB3F$    JRZ      READER       ; if not ready
DBC6      21D8EF     LDK      HL, SERFLS
DBC9      C886       C81T     O+HL
DBCB      3A012A     LD       A+1, SREC       ; get data
DBCE      4F         MOV      C+1       IC=A
DBCF      C9         RET
SLST:
:GET list device status
:ENTRY
:INONE

:EXIT
:IA = 0, IF NOT READY
:IA = OFFH IF READY
:Z3IT = SET IF NOT READY FOR OUTPUT

PROC
C800 C0E808 CALL ACISTAT
C803 E902 ANI SI Trey
C805 28 RZ ;RETURN
C806 F6FF CRI OFFH
C808 C9 RET
LIST:
:Output one byte to list port
ENTRY
IC = character to output
:EXIT
:NONE

PROC
CALL SLIST, GET STATUS
JRZ LIST, LOOP
MOV A,C
STX A++, SKM
:send chr
LDX HL, $FF
SBIT 1, HL
RET
ACISTAT:
RETN STATUS OF THE SERIAL PORT
ENTRY

:EXIT
: = STATUS REG

PROC
PUSH 6C

LD A+H.WID+1
RRC A
ANL 2Dh
MOV C,A
LD A+H.SSTS
ANL 0DFh
ORA C
MOV C,A
LD A+H.SERFLG
ANI 03
ORA C
STD A+H.SERFLG
POP 9C
RET
NEW DISK DRIVERS

; [5]

OC02
RDVR:
; RESET DRIVE
; ENTRY
; NJNE

; EXIT
; ZRIT = RESET IF ERROR

OC02
PROC

OC02 3E0A
OC04 3205EF
OC07 CD71DE
OC0A 3E09 ^OC114
OC0C CD080C
OC0F 309 ^OC1A4
OC11 3A05EF
OC14 3D
OC15 2D6 ^OC341
OC17 3C
OC18 37
OC19 C9
OC1A 6F
OC1B C9

JRC:1
JRC:1
CALL SEDRV
; SELECT DRIVE
CALL HOME
; HOME DRIVE
LD A,TRY
DEC A
JNZ: LOOP
INC A
; MAKE NOT ZERO
STC
; INDICATE ERROR
XRA A
; ENDC
; RET
DCIC

RSEC:

;READ SECTOR

;NOTE:

; No retries are performed at this level

;ENTRY

;IB = NUMBER OF SECTORS

;EXIT

;HML = LAST DMA ADDRESS PLUS ONE IF S2D TRANSFER

;Z91T = RESET IF ERROR

;TA = NONZERO IF ERROR

;ENTRY = 1 IF ERROR(SO OLD CHIPS DOESN'T DO RETRYS)

DCIC PROC

DCIC 3E80 LDK A+DARDS

DC1E 3201EF STD A+R_WCM

DC21 1805 ^DG28$ JR R:\_WSEC
DC23

; WRITE A SECTOR
; NOTE:
; 1. No retries are performed at this level
; 2. ENTRY
; IA = NUMBER OF SECTORS

; EXIT
; HL = LAST DMA ADDRESS PLUS ONE IF GOOD Transfer
; ZCBIT = RESET IF Error
; IA = NONZERO IF Error
; ENTRY = 1 IF ERROR (SO DLD CHKS DOESN'T GO RETRY)

DC23

PROC

DC23 3E40
DC23 32D1EF

;FALLS THROUGH TO "R\$SEC"

NEW DISK DRIVERS

DC28

lis
READ OR WRITE SEGMENT
ENTRY
SR = NUMBER OF SECTORS TO READ OR WRITE
SR.WDC = 3.IDS OR D.WRTS

EXIT
HL = LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
ZRT = RESET IF ERROR
TRY = 0 IF GOOD, 1 IF ERROR

DC28
PROC
DC28 ED43C4EF $ STD BC+NUMSEC ISAVE BC
DC2C 3EOA LDK A+RTRY
DC2E 3205EF STD A+TRY ISET RETRY NUMBER

ISELECT DRIVE

DC31 CDHDE JRLOP: CALL SELDRV RETURN DRIVE ON
DC34 2800 *DC38$ JRE 11 IIF DRIVE ON DON'T READ ADDRESS

ISET "DTRKR" TO HEAD POSITION

DC36 CD455C CALL RDRR READ ADDRESS AND SET CONTROLLER
DC39 3820 *DC35$ JRC ERR ISTOP

ISEEK

DC3B CDFA6C F11 CALL SEEK ISEEK TO TRACK
DC3E 3818 *DC59$ JRC ERR ISTOP

READ OR WRITE

DC40 ED44C4EF $ LD BC+NUMSEC IA = NUMBER OF SECTORS TO R/W
DC44 D3C00 CALL RDR.WRT IREAD/WRITE PER R.WDC
DC47 3017 *DC50$ JRNCE IEND IIF GOOD

IRETRY?

DC49 2105EF LDK H+RTRY IGET RETRYS
DC4C 35 DEC (HL) IMORE RETRYS
DC4D 200C *DC58$ JRE ERR INORE RETRYS
DC4F E7 LD A+HL IGET NUMBER OF RETRY
DC50 FE09 CMP RTRY-1
DC52 2000 *DC31$ JNZ FLROOP ILOOP IF NOT FIRST RETRY
DC54 CD455C CALL RDRR ICHECK TRACK ON THE FIRST RETRY
DC57 3002 *DC58$ JRC ERR ISTOP IF ERROR
DC59 1806 *DC31$ JR FLROOP ILOOP
DC5B 3ED1 IERR: LDK A+1 IINDICATE ERROR
DC5D 37 JRA A IEXIT
DC5E 1801 *DC61$ JR 32 IEXIT
DC60 AF IEND: XRA A IINDICATE 1DDD
NEW DISK DRIVERS

SET "TRY" AND REGISTER B

DC61 3205EF 12: STO A*TRY
DC64 000000FF & LD BC*NUMSEC
DC68 09 RET

SET RETRY TO 1 FOR ERROR AND 0 FOR GOOD

SRCIM 808X Assembler ver 3.55 c: /5577 = 9:92 Page 124
3:23W141 +ASM
NEW DISK DRIVERS

0C69
SEVEN:
IDENTIFY THE DENSITY AND NUMBER OF SECTORS PER TRACK OF THIS DISK DRIVE
ENTRY
EXIT

EXIT
IB = NUMBER OF SECTORS ON ONE TRACK
DSIT = RESET IF ERROR
ISAVTYP IS SET WITH DENSITY AND SECTOR SIZE

0C69
PROC

0C69
%CHECK DENSITY

0C69
:IHOME LOOP

0C69
3EO2
0C69
STO A,TRY

0C71
CD71DE:
CALL SDRY
ISELECT DRIVE
JRC ?
NO ERROR CHECKING FOR SDRY BECAUSE NO ERRORS ARE RETURNED AT THIS TIME

0C74
CD856D:
CALL RDR
IREAD ADDRESS

0C77
HD:
PUSH BC
ISAVE COUNT

0C81
CD91:
CALL SAVTY:
IDENTIFY THIS DENSITY

0C84
CD93:
CALL DR
ICHANGE DENSITY BIT

0C86
CD95:
LD A, SAVTY:
PREVIOUS DENSITY

0C89
CD97:
LD X, 1
CHANGE DENSITY BIT

0C8B
CD99:
LD X, 0
NEW DENSITY

0C9B
CD05:
DJNZ 0RL2:
IDENTITY LOOP

0C9D
CD07:
DI30TH DENSITY FAIL HOME DRIVE AND TRY AGAIN

0C9F
CD09:
LDK HL, RTRY

0C9F
CD0B:
DEC HL:
SEND IF SECOND TIME THROUGH

0C9F
CD0D:
CALL HOME
IHOME DRIVE

0C9F
CD0F:
JRC :FRET
SEND IF ERROR IN HOME

0C9F
CD11:
SET "SAVTY:

0C9F
:1:
LD A, DSTS+3
ISECTOR LENGTH STATUS BYTE

0C9F
CD13:
LD A, 0300_0310:
ID-3

0C9F
CD15:
SLA A
NOW IS 0000_XX008 WAS 0000_00XX8
```assembly
; CD1
SCTRKR:
; SET CONTROLLER TRACK REGISTER
LD TRKR <= SAVTRK
; USED IN FORMATTING WHEN YOU DON'T KNOW WHERE THE HEAD IS
ENTRY
; SAVTRK = TRACK
EXIT
; ID TRKR = SAVTRK

CD1 PROC
CD1 3A15EF
CD4 320121
CD7 C9
RET
```
NEW DISK DRIVERS

;DRIVE IS ALREADY SELECTED AND READY
;IF *SEKDEL* has the verify bit set, this proc will check for seek and CRC errors.
;ENTRY
;DISK = DRIVE
;EXIT
;CBIT = SET IF ERROR

;PROC

;SECTION 3A13EF
;LD A, S.SEKDEL
;CALL DFSK
;RC
;CE = SEEK DELAY

;SECTION 5D57
;ANI 0000_001B
;ISREAD & VERIFY BITS
;FUNCTION DISK
;IF ERROR

;SECTION 5D580E
;CALL WBUSY
;IWAIT FOR BUSY TO DROP

;SECTION 38021
;LD A, 3*STSR
;BIT 2, A
;JNZ 11
;IF NOT ON TRACK ZERO

;SECTION 3A13EF
;LD A, +SEKDEL
;ANI 0000_010B
;IF VERIFY?
;NO VERIFY GOOD RETURN

;SECTION 3A021
;LD A, +STSR
;ANI 0001_00FD
;TEST SEEK AND CRC
;GOOD RETURN

;SECTION 37
;STC
;RC
;IF ERROR
DCFA
SEEK
; SEEK TO TRACK DEFINED BY SAVTRK
; TRACK REG UPDATED AND VERIFIED
ENTRY
; SAVTRK SET TO DESIRED TRACK

DCFA
;EXIT
CSIT = SET IF ERROR
; IF NO ERROR CONTROLLER TRACK = SAVTRK

---

DCFA
PROC
DCFA 210121
DCFD 3A15EF
D000 0E
D001 C9
+2
D002 320321
D003 0E10
D007 1B0C ^0D156
LDK ML+DTRK2
LD A+SAVTRK
CMP [ML]
STD A+D+JATR
LDK B+D+SEK
JR PSEKC
; RETURN
; SET TRACK WANTED
; PERFORM SEEK COMMAND
NEW DISK DRIVERS

0009

STEP:
STEP ONE TRACK
ISAVTRK IS NOT USED IN THIS PROC
HCUTROLLER TRK RES IS UPDATED
IVSIFY IS PERFORMED
JUMP
INONE

EXIT
ICBIT = SET IF ERROR
: IF NO ERROR CONTROLLER TRACK = TRACK +/- 1

0009
PROC

0009 0620
LDC 8+STP
0009 1908 ^C015% JR PSEKC ;PERFORM STEP COMMAND
NEW DISK DRIVERS

STEPIN:
STEP IN ONE TRACK
IAVRK IS NOT USED IN THIS PROC
CONTROLLER JRM REG IS UPDATED
ENTRY
FINISH

EXIT

SET IF ERROR
IF NO ERROR CONTROLLER TRACK = TRACK + 1

PROC

640

LDX B+D*STPI

1804 *DD15$ JR PSEKC  PERFORM STEP-IN COMMAND
NEW DISK DRIVERS

STEPJUT:
*STEP OUT ONE TRACK
*SAVTRK IS NOT USED IN THIS PROC
*CONTROLLER TRK REG IS UPDATED
*VERIFY IS PERFORMED
*ENTRY
*NONE

:EXIT
ICBIT = SET IF ERROR
; IF NO ERROR CONTROLLER TRACK = TRACK - 1

PROC

0D11 0660 LDK B+D.STPD
0D13 1800 ^2015$ JR PSEEKC ;PERFORM STEP-OUT COMMAND
NEW DISK DRIVERS

0015 PROC

0015 3A13EF LD A,SEKDEL
0017 E617 ANI 0001_0111B :ONLY UPDATE,VERIFY, & SPREAD
001A 90 ORA 9 :FOR IN COMMAND

0012 CD480E CALL FOSK :FUNCTION DISK
0015 98 RC :IF ERROR

001E CD880E CALL HBUSY :WATCH FOR BUSY TO DROP

0021 3A13EF LD A,SEKDEL
0024 E604 ANI 0000_01009 :VERIFY?
0027 C9 RZ :INFO VERIFY GOOD RETURN

0028 3A0021 LD A,STSR
002B E61B ANI 0001_10009 :TEST SEEK AND CRC
002E C9 RZ :GOOD RETURN

002F 37 STC
0030 C9 RET :IF ERROR
NEW DISK DRIVERS

0037    WRITE:
0038    ENTRY
0039    IS = NUM OF SECTORS TO WRITE
0040    EXIT
0041    PML = LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
0042    PSIT = SET IF ERROR

0037    PROC
0037    READ
0039    STD A+0xFFFF
0039    STD A+2_WCOM

: Falls through to RD_WAT
NEW DISK DRIVERS

303C 4D_WRIT
READ OR WRITE A SECTOR
ENTRY
IB = NUM OF SECTORS TO READ OR WRITE
WRKCOM = D16S OR D16RTS
EXIT
HLX = LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
S16 = SET IF ERR

303C PASC

303C 3A14EF
303F 320211
3042 C5  PUSH BC
ISAVE NUMBER OF SECTORS TO R/W

3043 2100ED
3044 3A0DEF
3049 C3F 5
3043 C3F 5
3040 5303
304F 47
3050 2B04 +D0588
3052 47
3053 79
3054 10FD +D0538
3056 E8
3057 C1
3059 C5
ISET B
ISAVE NUMBER OF BYTES IN ONE SECTOR

305A 78
305C 0ED0
305E 3502 +D0523
3060 0F10
3062 3A11EF
3065 81
ISAVE NUMBER OF SECTORS TO R/W

3066 210000
3069 19
306A 10FD +D064B
306C E5  PUSH HL
ISAVE LENGTH

SOURC BDX Assembler ver 3.5E cl/657/7+ e/9/92  Page 134
315H14A  134
ISAVE COMMAND

DD60 F3 DI
DD64 CD000E CALL FSYS
DD71 3D04 A0771 JNC 13 IIF GOOD

IIF ABORT BEFORE DMA

DD73 D1 POP DE IRESTORE STACK
DD74 D1 POP DE
DD75 F3 EI
DD76 C9 RET IRETURN IF ERROR IN FSYS

ISET RETURN FROM DMA. DMA ADDR AND NUMBER OF BYTES TO TRANSFER

DD77 C1 13: POP BC IRESTORE LVI+1
DD78 2A0EF LD HL,DMADR SML = DMA ADDRESS
DD7B 119D00 LDK DE,RET
DD7E D5 PUSH DE IFOR RETURN

I00 DMA

DD7F 3A0EF LD A,2-WCRM ((13) GET COMMAND
DD82 FE80 CMP D,00S ((7)
DD84 2A250E JZ DMA300D ((10) READ DMA RETURNS TO IRET
DD87 23F00E JMP DMAMRT ((10) WRITE DMA RETURNS TO IRET

IRETURN FROM DMA AND CHECK FOR BUSY AND RESET

DD8A C1 IRETI POP BC IRESTORE NUMBER OF SECTORS
DD8B 1A LD A,[DE] IGET STATUS
DD8C C847 4 BIT 0,A
DD8E 230E A00E JPC 14 IIF NOT BUSY
DD90 119D00 LDK DE,IRETI
DD93 D5 PUSH DE IFOR RETURN
DD94 05 DEC 3 ISUBTRACT ONE FROM THE NUMBER OF SECTORS AND SET THE ZROD FLAG
DD95 2A200E JZ WBUSY IIF VON MULTI-SECTOR 3/4 WAIT FOR BUSY TO DROP
DD98 C3300E JMP FMINT ICLEAR BUSY
DD9B 3A0031 IRETI: LD A,2-STR IRETURN AND SET STATUS

ICHECK FOR ERRORS

DD9E 285C 141 ANI 0101_11000 RST 0400, E000, 000, AND LOW DATA
DDA0 2801 A0DA35 JZ 15 IIF GOOD
DDA2 37 STC IIF ERROR RECORD CONTROLLER REGISTERS
DDA3 F0 15 EI
DDA4 C9 RET IRETURN
DATA
READ ADDRESS INFO
READ SIX BYTES INTO 'DISK'

ENTRY

EXIT
IA = OFF IF TIMEOUT ERROR
ICENT = SET IF ERROR
IN, TNR = READ POSITION
IN77, IN78 = SETS TRACK RES IN CONTROLLER IF 5773

PROC

C045 3EOC  LDX A, 3EOC
C046  63   DJ
C047 CD40E  CALL DOSX (FUNCTION DISK
C048 383E ADDSPS  JRC 11

WAIT FOR FIRST PRO OR TIME OUT

SET REGISTERS FOR DMA TRANSFER

C049 010E0D  LD X B, 010E0D
C050 210F6  LD X H, 210F6

WAIT FOR 1/4 OF A TRACK (512552) OR PRO

C051 110F11  LOOP
C052 1A0211  LD X D, 1A0211
C053 1F   LD H A, 1F
C054 71 A8020C  JC 13

C055 13 OF9A 1F
C056 7A MOV A, 7A
C057 83  JR 7A
C058 23D60D  JNZ 1FL00

INDICATE TIME OUT ERROR

C059 CD3B6E  CALL PRIORITY (CLEAR BUSY
C060 3EFF  LD X H, 3EFF
C061 111D ADDR  JR 12

INDICATE A TIME OUT ERROR

TRANSFER FIRST BYE AND CALL DMA FOR LAST FIVE BYTES

C062 3A0321 13
C063 77  LD A, 77
C064 3A030C  INC H
C065 00  MOV A, 00
C066 CD64DE  CALL DMA (117) CALL DMA

RETURN FROM DMA AND WAIT FOR BUSY TO BE RESET

C067 CD3B6E  CALL BUSY
C068 3E11 ADDR  JR 11

IF TIME OUT ERROR
NEW DISK DRIVERS

DDB
READTRK:
READ ONE TRACK FROM THE DRIVE
ENTRY
IDMADR = FWA OR JFPER
EXIT
ICBIT = SET IF ERROR

DDEB
PRCC
DDEB 3E6D
LDX A+DRT
DDEB F3
EI
DDEE CD48DE
CALL FDSK
DFDF 380C +DDEFF
JR C 11
:
DDEE
LDK BC+OFFFF
DDE9 CD3ADE
CALL DMADR
DDEF F8
EI
DDEF AF
XRA A
DDEF C9
RET
DDEFF FB
:
DDE0 C9
RET
NEW DISK DRIVERS

DE01
EWRK:
IPMAT ONE TRACK
ENTRY
IBC = LENTH
IDMAD = FWA OF BUFFER
EXIT
ICMAT = SET IF ERROR

DE01
PROC

TEST DENSITY AND SET REG C TO DASH OR OFF4

DE01 3A30EF
DE04 164E
DE06 0F
DE07 3002 ^DE05
DE09 16FF
LD A+SAYYP
LDK D+4EH
RA
JLNK I1
LDK D+OFFH
ISINGLE

IGIVE COMMAND

DE0B 3F01
DE0D F3
DE0E D5
DE0F CS
DE10 CD480E
DE13 C1
DE14 01
DE15 381C ^DE33
PUSH BC
CALL FDSK
PUSH BC
POP DE
JRC 13
ISINGLE

ISMA

DE17 D5
DE18 2A00EF
DE1B 200D0E
PUSH DE
CALL DMAWT
IFILL BYTE
LD ML+DMADR
CALL DMAWT

IPAD REST OF TRACK

DE1E C1
DE1F 210321
DE22 1A
DE23 IF
DE24 3006 ^DE2C
DE26 1F
DE27 30F9 ^DE29
DE29 70
DE2A 18F6 ^DE22
POP BC
LD ML+21034
POP RC
JRC
12
JRC
HLOP
STO B+([HL])
HLOP
IB = FILL BYTE
DATA REGISTOR
A+([DE])
GET STATUS
FINISHED IF NO BUSY
IF NO DRQ
ISTORE BYTE

ICHECK FOR ERROR

DE2C 3A021
DE2F E644
DE31 2B01 ^DE34
I2
ANI 0100_D100B
STO B+([HL])
GET STATUS
ITEST write protect, and data lost
IF GOOD
NEW DISK DRIVERS

DE33 37  131  STC
DE34 FB  141  EI
DE35 29  RET
NEW DISK DRIVERS

DEFB
PRINT
INTERRUPT DISK CONTROLLER
ENTRY
NONE
EXIT
BUSY CLEARED.

DEFB PROC
0336 F5     PUSH AF
0337 D5     PUSH BC
0338 3E00   LDX A.+FIN
033A 320021  STO A.+CHRD

WAIT FOR AT LEAST 20 MICROSECONDS

033D 97     LDA A (4)
033E 9607    LDY B+7 (7)
0340 10FE ^03408  WLDOP DJNZ .WLDOP (91) = (1307) WAIT

CHECK FOR BUSY DROP

0342 C0B9OE  CALL WBUSY
0345 F1      POP BC
0346 F1      POP AF
0347 29      RET
DE48

FDSK:

FUNCTION DISK ROUTINE

THIS IS THE ONLY ROUTINE THAT WRITES TO THE COMMAND REGISTER OF THE CONTROLLER CHIP.

THIS ROUTINE HAS A DELAY OF AT LEAST 28 MICRO SEC. BEFORE READING THE STATUS BY THE CHIP.

ENTRY

IA = FUNCTION CODE

IA = OFFH IS TIME OUT ERROR

ICBIT = SET IF ERROR

DE48

PROC

DE48 210021 LDK HL,0:STS R STATUS AND COMMAND REGISTER
DE48 C346 :DE524 JRT 11 IF NOT 3JSY

DE6F CD3D FF CALL PRINT :RESET 8JSY

DE52 77 :1: STO A,HL :FUNCTION DRIVE(WRITE COMMAND TO CONTROLLER)

WAIT FOR 56 SINGLE AND 28 DOUBLE

DE53 3A0EFF LD A,SAVTYPE (13) DISK TYPE
DE55 0605 LDK B,5 (7)
DE58 DF RRC A (14)
DE59 CE5D8E JNC :WLOOP (10) IF DOUBLE DENSITY

DE5C AF XRA A (4) RESET CARRY FLAG

DE5D 0600 LDK B,4:13 (7)

DE5F 100E ^DE5F1 :WLOOP: DJNZ :WLOOP (13) WAIT

WAIT FOR BUSY TO BE SET

DE61 3EFF LDK A,OFFH (7)
DE63 47 MOV B,A (4) 255 LOOPS

DE64 C846 $ :LOOP: BIT 0,HL :TEST BUSY BIT
DE66 2004 ^DE5C5 JRMNT 13 IF CHIP WENT BUSY

DE68 10FA ^DE644 :JNZ :LOOP (1) IF NOT TIME-OUT

DE6A 37 STC

DE69 90 RET

DE6C 3264FF :3: STO A,ACTIVE (3) SET DRIVE ACTIVE COUNTER

DE6F AF XRA A (4) RESET CARRY FLAG

DE70 90 RET
NEW DISK DRIVERS

DE71
SELDRV:
ISELECT DRIVE
IENTRY
ISDISK = DRIVE TO SELECT

IEXIT
IZIT = SET IF PIABD WAS THE SAME AS SDISK
IZIT = RESET IF PIABD WAS DIFFERENT THAN SDISK
IZIT = SET IF THERE ARE NO INDEX PULSES

DE71
PROC
DE71 CDA6DE CALL SELDEN ISELECT DENSITY
DE74 3A17EF LD A+SDISK
DE77 21C7EF LDX HL+DISKSWP :DISK DRIVE SWAP CELL
DE7A AE XDE [HL] :ISWAP A FJ=3 IF DISKSWP=1
DE7B E561 ANL 1 :CAN ONLY BE 0 OR 1
DE7D FE01 CMP 1
DE7F 2092 +DE83E JNZ #1 :IF WDT DRIVE 1
DE81 3E40 LDY A+40H
DE83 C640 :1: ADI 40H
DE85 4F MOV C+A
DE86 3AB2EF LD A+PIABD
DE89 47 MOV B+A
DE8A 5EC0 ANI 1100_0000B :SET DRIVE BITS ONLY
DE8C 39 CMP C
DE8D 2810 +DE9F5 JNZ #2 :IF DRIVE ALREADY SELECTED

ISELECT DRIVE
DE8F CDCCDE CALL DISKJ :TURN DRIVE ON
DE92 3EFA LDY A+250
DE94 CD5D CALL DELAY :WAIT FOR MOTOR SPIN UP
DE97 CD44 LDY A+2D
DE99 CD5D CALL DELAY :2ND DELAY
DE9C 3F01 LDY A+1 :INDICATED DRIVE WAS NOT SELECTED
DE9E 37 ORA A :SET FLAGS
DE9F 2164EF :2: LDY HL+ACTIVE
DEA2 3F6F STY OFFH+4HL
DEA4 FB EI
DEA5 C9 RET
; SELDEN: SELECT SINGLE OR DOUBLE DENSITY
; ENTRY
; $SAVTYP = BIT 0:
; 1 = SINGLE, 0 = DOUBLE
; ; NEXT:
; $NOTE: Bit 0 of "PIAAD":
; ; set = single density
; ; reset = double density

DEA6 PROC
DEA6 3A61EF LD A, PIADAD
DEA9 E6FE ANI 1111_11DB
DEAB 4F MOV C,A

; SET DENSITY BIT
DEAC 3AD0EF LD A, $SAVTYP
DEAF DF RRC
DEB0 3002 $DE945 JRC 11
DEB2 C8C1 % SBI 0,C
DEB4 CD700D 11 CALL DRAD
DEB7 C9 RET ;FUNCTION PIAD
DEC8	DEEB

DEC9

RET
NEW DISK DRIVERS

;DISK:
;SELECT DRIVE BY SETTING THE "PIA" WITH THE VALUE SPECIFIED BY C
;ENTRY
;IC = DRIVE

;EXIT
;IN3NE

;PROC

DECF 3A52EF
DE02 E63F
DE04 81
DE05 4F
DE06 CD8600
DE09 C9

LD A,PIA8D
ANI 0011_1111B
IGET VIO OFFSET AND BELL

OR C
MOV C,A
CALL DP80
;FUNCTION PID-3

RET
DED A
DMARK:
TRANSFER DATA FROM CONTROLLER TO MEMORY
ENTRY
IBC = BYTES TO TRANSFER
IHL = FW4 OF BJFPER
:NEXIT
IHL = NEXT ADDRESS
IDE = D*STSR

DED A PROC
DED 110021
LDK DE*+STSR i(10)
DED 1A
JLOP: LD A+[IDE] i(7) GET STATUS
DED 1F
RAR i(4)
DED 0D
RNC i(5) RETJR IF NO BUSY
DED 1F
RAR i(4)
DED 0D00DE
JNC :LLOOP i(10) IF NO DRQ
DED 340321
LD A+[D4ATR] i(13) GET BYTE
DED 77
STD A+[HL] i(7) STORE BYTE
DED 23
INC HL i(6)
DED 0B
DEC BC i(6)
DED 78
MOV A,B i(4)
DED 31
ORA C i(4)
DED 2D00DE
JNZ :LLOOP i(10)
DED C9
RET
DEF0
    DMAWRITE:
    ;Xfer data from memory to disk
    ;ENTRY
    ;BC = BYTES TO TRANSFER
    ;HL = FWA of BYTE
    ;EXIT
    ;HL = NEXT ADDRESS

DEF0
PROC

DEF0 110021
    LOD  DE.*STSR

DEF3 1A
    :LOOP: LD  A.*DE)  ;GET STATUS
DEF4 1F
    RAR
DEF5 0A
    INC
    ;RETJRAn IF NO BUSY
DEF6 1F
    RAR
DEF7 02F3DE
    JNC  :LOOP  ;IF NO C39

DEF9 7E
    LD  A.*HL)  ;GET BYTE
DEFB 320321
    STD  A.*D.*ATP  ;STORE BYTE
DEFE 23
    INC HL
DEFF 0B
    DEC BC
DF00 78
    MOV A.*B
DF01 81
    ORA C
DF02 02F3DE
    JNZ  :LOOP
DF05 C9
    RET
; Select disk drivers
; Entry = current disk drive

DF06  PROC
DF06  3A42EF  LD  A,P1450
DF09  261F    AND  1,11110
DF08  4F      MOV  E,A
DF0C  C3B603  JMP  DPJS  ; deselect last drive
FORMAT

DFOF

FORMAT:

Entry:
$BC = FWA of buffer
$BUF+0 = SW length
$BUF+2 = beginning of data
$SAVTRK = THE TRACK TO BE FORMATTED

EXIT

INJNE

DFOF

PROC

DFOF E0430F0E $ STD BC,DMADR ;SAVE BC

SELECT DRIVE

DF13 0D71DE CALL SELDRV
DF16 3831 4DF49$ JRC 1ERROR

TEST FOR STEP OR NO-STEP

DF18 2115FF LDK HL,SAVTRK
DF19 3A0121 LD A,AC,TRK
DF1E 9E CMP [HL]
DF1F 2B17 4DF38$ JRE 1 ;IF SAVTRK AND TRACK REG ARE THE SAME SKIP THE STEP

STEP IN ONE TRACK

DF21 3A13EF LD A,SEKDEL
DF24 F610 JRI 0001_0003 ;UPDATE
DF26 3213EF STD A,SEKDEL ;SET UP SEKDEL
DF29 4D0D00 CALL STEPIN

DF2C 55 PUSH AF
DF2D 3A13EF LD A,SEKDEL
DF30 5603 ANI 0000_0013 ;ONLY SPREAD LEFT
DF32 3213EF STD A,SEKDEL ;RESET SEKDEL
DF35 F1 POP AF ;RESTORE

DF36 3811 4DF49$ JRC 1ERROR

SET "DMADR"

DF38 2A0F0E $11 LD HL,DMADR ;GET ADDRESS
DF38 4E LD C,[HL]
DF3C 23 INC HL
DF3D 46 LD 3,[HL] ;IBC = LENGTH OF FORMAT DATA
DF3E 23 INC HL
DF3F 220F0E STD HL,DMADR ;SET DMAD
DF42  C00DE  CALL  FMTTRK
DF45  3D02  *DF49*  JRC  ERROR
DF47  AF  XRA  A
DF48  C9  RET
DF49  3EFF  ERROR:  LDL  A,OFFH
DF4B  87  CDR  A
DF4C  C9  RET

DF4D  
HEAD:
; If "ESC" is pressed in response to the first prompt this is the place to put a cold boot loader other
; than the floppy one supplied.
; ENTRY
; NONE

; EXIT
; NONE

DF4D  PROG

DF4D  C30000  JMP  START

    = 0F50  RLWA  =  *  ILWA OF ROM RESIDENT CODE

    = OF4F  MSG  "LENGTH OF THIS ROM IS = ",RLWA-1
    -  0000  IF  RLWA < OFFH
    ;9  ERROR CODE TOO LARGE.
    ENDIF
    ECHO  OFFFH-RLWA 1  DB  OFFH
    ENDM

    END

no ERRORS, 472 Locals, 5770h bytes not used. Program LWA = OFFFA.
| ACIAO | EFF0 | 3# 6 | 115/15 |
| ACIST | 08E0 | 77/ | 114/12 117/14 119# 2 |
| ALP | 0D03 | 2# 9 | 77/55 |
| BCPM | 0241 | 20/ 9 | 22# 2 |
| BLCNT | EF68 | 2#01 | 45/10 67/30 |
| BIJYP | 0033 | 12# 3 | 12# 8 |
| BKS | 0008 | 1#55 | 33/ 9 |
| BRTBIT | 05D0 | 2#12 | 38/13 53/17 60/16 |
| CALC | 0534 | 53# 7 | 54# 7 55# 2 |
| CDE | 0037 | 1#19 | 33/12 |
| CDDOT | 0555 | 7/36 | 7/40 16/ 3 13# 3 |
| CDPAD | EF02 | 2#13 | 12/ 8 20/14 21/7 |
| CCRA | 2901 | 2#25 | 92/ 3 92/40 93/15 |
| CCRB | 2903 | 2#27 | 92/22 92/53 95/22 95/24 95/23 95/23 95/15 95/15 95/15 |
| CCR | 0000 | 1#52 | 7/35 13# 2 19/10 18/20 18/20 18/40 |
| CEE | 0T72 | 70/27 | 74# 2 |
| C1 | 0373 | 7/30 | 1b/11 29# 2 |
| CRLN | 0651 | 45# 5 | 51# 5 58# 3 60# 2 61/48 |
| COUT | 0030 | 14/29 | 14/32 16/12 34# 2 |
| CPU2 | 041A | 32/25 | 36# 2 |
| CPUT | 2900 | 2#24 | 2/25 2/26 2/27 2/28 2/30 2/31 |
| CR | 0030 | 1#52 | 7/35 13# 2 19/10 18/20 18/20 18/40 |
| CTLSY | 0002 | 2# 8 | 77/49 |
| CTLIB | 0924 | 94# 6 | 88#4 |
| CURS | EF5A | 2#55 | 34/40 49/40 |
| CV2IP | 0530 | 1#02 | 112# 7 |
| CV2CP | 08E0 | 1#9# 2 | 111# 7 |
| D.CMIX | 2100 | 2#13 | 2/19 2/20 2/21 2/22 143/15 |
| D.SQTR | 2103 | 2#22 | 129/19 138/51 149/22 150/22 |
| D.FINT | 0000 | 5#11 | 143/14 |
| D.BSA | 0050 | 5# 8 | 138/18 |
| D.BSC | 0050 | 5# 5 | 121/17 134/12 137/25 |
| D.BST | 0050 | 5# 9 | 140/12 |
| D.SCMB | 2120 | 2821 | 136/18 139/11 |
| D.SDC | 0010 | 5# 2 | 129/21 |
| D.SPD | 0020 | 54# 3 | 130/16 |
| D.STP | 0040 | 5# 4 | 131/15 |
| D.STP | 0050 | 5# 5 | 132/16 |
| D.STR | 2100 | 2#19 | 129/23 128/31 133/27 137/48 139/33 139# 2 141/57 146/15 147/17 147/14 150/13 |
| D.TK | 2101 | 2#20 | 127/15 129/14 139/12 152/28 |
| D.WRTS | 0040 | 5# 7 | 127/17 135/12 |
| D.WRT | 00F0 | 5#10 | 141/24 |
| DCT | EF64 | 2#60 | 144/48 145/44 |
| DB | 0001 | 2# 4 | 73/24 |
| DTR | EF26 | 57/53 | 151# 2 |
| DEC | 014F | 152 2 | 145/37 145/39 |
| DIM | 2030 | 2#13 | 40/43 |
| DISIM | ac | 5#21 | 6/13 38/15 42/45 53/13 52/18 63/17 |
| OMAD | EF2F | 2#46 | 19/26 22/21 24/57 26# 7 137/17 142/22 |
| OMAD | EF2A | 2#45 | 19/26 22/21 24/57 26# 7 137/17 142/22 |
| OMART | 05F0 | 16#55 | 137/27 138/55 143/21 149# 2 |
| OMART | 05F0 | 16#55 | 137/27 138/55 143/21 149# 2 |
### RAM STORAGE LOCATIONS

<table>
<thead>
<tr>
<th>Location</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00C</td>
<td>0000</td>
<td>02/19</td>
</tr>
<tr>
<td>D05E</td>
<td>0823</td>
<td>87/2</td>
</tr>
<tr>
<td>O684</td>
<td>47/10</td>
<td>61/2</td>
</tr>
<tr>
<td>0577</td>
<td>24/45</td>
<td>125/57</td>
</tr>
<tr>
<td>00AD</td>
<td>13/6</td>
<td>19/20</td>
</tr>
<tr>
<td>03A2</td>
<td>32/19</td>
<td>53/2</td>
</tr>
<tr>
<td>0959</td>
<td>32/22</td>
<td>51/2</td>
</tr>
<tr>
<td>0010</td>
<td>32/6</td>
<td>52/7</td>
</tr>
<tr>
<td>0937</td>
<td>30/7</td>
<td>34/43</td>
</tr>
<tr>
<td>0021</td>
<td>30/11</td>
<td>57/2</td>
</tr>
<tr>
<td>0002</td>
<td>30/9</td>
<td>56/10</td>
</tr>
<tr>
<td>00D7</td>
<td>30/11</td>
<td>35/6</td>
</tr>
<tr>
<td>0020</td>
<td>30/5</td>
<td>52/17</td>
</tr>
<tr>
<td>0004</td>
<td>30/5</td>
<td>56/13</td>
</tr>
<tr>
<td>3538</td>
<td>32/13</td>
<td>54/2</td>
</tr>
<tr>
<td>0040</td>
<td>13/2</td>
<td>13/13</td>
</tr>
<tr>
<td>007F</td>
<td>51/6</td>
<td>38/13</td>
</tr>
<tr>
<td>0010</td>
<td>18/4</td>
<td>93/5</td>
</tr>
<tr>
<td>0537</td>
<td>34/31</td>
<td>18/4</td>
</tr>
<tr>
<td>0002</td>
<td>18/13</td>
<td>19/22</td>
</tr>
<tr>
<td>0039</td>
<td>18/35</td>
<td>18/36</td>
</tr>
<tr>
<td>0580</td>
<td>32/23</td>
<td>50/2</td>
</tr>
<tr>
<td>0601</td>
<td>32/20</td>
<td>57/5</td>
</tr>
<tr>
<td>0550</td>
<td>30/10</td>
<td>35/5</td>
</tr>
<tr>
<td>0394</td>
<td>30/17</td>
<td>55/4</td>
</tr>
<tr>
<td>0356</td>
<td>32/23</td>
<td>50/2</td>
</tr>
<tr>
<td>052B</td>
<td>32/12</td>
<td>56/10</td>
</tr>
<tr>
<td>053C</td>
<td>32/24</td>
<td>50/4</td>
</tr>
<tr>
<td>051F</td>
<td>32/16</td>
<td>56/31</td>
</tr>
<tr>
<td>0602</td>
<td>123/17</td>
<td>133/16</td>
</tr>
<tr>
<td>05F1</td>
<td>17/3</td>
<td>14/1</td>
</tr>
<tr>
<td>0636</td>
<td>5/8</td>
<td>16/51</td>
</tr>
<tr>
<td>0736</td>
<td>15/32</td>
<td>152/3</td>
</tr>
<tr>
<td>0300</td>
<td>30/2</td>
<td>45/4</td>
</tr>
<tr>
<td>004F</td>
<td>6/27</td>
<td>67/3</td>
</tr>
<tr>
<td>07F6</td>
<td>74/17</td>
<td>74/21</td>
</tr>
<tr>
<td>2100</td>
<td>24/3</td>
<td>76/12</td>
</tr>
<tr>
<td>240D</td>
<td>28/3</td>
<td>115/12</td>
</tr>
<tr>
<td>2601</td>
<td>28/7</td>
<td>116/19</td>
</tr>
<tr>
<td>2400</td>
<td>28/5</td>
<td>119/19</td>
</tr>
<tr>
<td>2801</td>
<td>28/3</td>
<td>118/15</td>
</tr>
<tr>
<td>2400</td>
<td>28/5</td>
<td>118/15</td>
</tr>
<tr>
<td>0540</td>
<td>19/5</td>
<td>10/28</td>
</tr>
<tr>
<td>0602</td>
<td>15/56</td>
<td>120/19</td>
</tr>
<tr>
<td>0550</td>
<td>125/53</td>
<td>129/2</td>
</tr>
<tr>
<td>007C</td>
<td>7/3</td>
<td>16/37</td>
</tr>
<tr>
<td>0936</td>
<td>15/33</td>
<td>94/2</td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>M5I80H</td>
<td>0030</td>
<td></td>
</tr>
<tr>
<td>MCUP</td>
<td>0030</td>
<td></td>
</tr>
<tr>
<td>NA4I6</td>
<td>33/10</td>
<td></td>
</tr>
<tr>
<td>NPE5Y</td>
<td>28/14</td>
<td></td>
</tr>
<tr>
<td>NUMEC</td>
<td>28/14</td>
<td></td>
</tr>
<tr>
<td>NUMSC</td>
<td>24/3</td>
<td></td>
</tr>
<tr>
<td>OPAV</td>
<td>7/13</td>
<td></td>
</tr>
<tr>
<td>OPGD</td>
<td>7/3</td>
<td></td>
</tr>
<tr>
<td>OSF</td>
<td>7/26</td>
<td></td>
</tr>
<tr>
<td>PA.CON1</td>
<td>28/2</td>
<td></td>
</tr>
<tr>
<td>PA.COT</td>
<td>28/2</td>
<td></td>
</tr>
<tr>
<td>PA.CTL</td>
<td>28/2</td>
<td></td>
</tr>
<tr>
<td>PA.DIR</td>
<td>28/2</td>
<td></td>
</tr>
<tr>
<td>PA.DRI</td>
<td>28/2</td>
<td></td>
</tr>
<tr>
<td>PA.DRT</td>
<td>28/2</td>
<td></td>
</tr>
<tr>
<td>PARIP</td>
<td>17/15</td>
<td></td>
</tr>
<tr>
<td>PAR1O</td>
<td>17/16</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>PSC+OR</td>
<td>4/7</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>$0000</td>
<td>125/66</td>
<td></td>
</tr>
<tr>
<td>$0001</td>
<td>126/16</td>
<td></td>
</tr>
<tr>
<td>$0002</td>
<td>126/49</td>
<td></td>
</tr>
<tr>
<td>$0003</td>
<td>3#12</td>
<td></td>
</tr>
<tr>
<td>$0004</td>
<td>121/18</td>
<td></td>
</tr>
<tr>
<td>$0005</td>
<td>122/18</td>
<td></td>
</tr>
<tr>
<td>$0006</td>
<td>134/13</td>
<td></td>
</tr>
<tr>
<td>$0007</td>
<td>135/13</td>
<td></td>
</tr>
<tr>
<td>$0008</td>
<td>136/50</td>
<td></td>
</tr>
<tr>
<td>$0009</td>
<td>137/24</td>
<td></td>
</tr>
<tr>
<td>$000A</td>
<td>121/20</td>
<td></td>
</tr>
<tr>
<td>$000B</td>
<td>123#</td>
<td></td>
</tr>
<tr>
<td>$000C</td>
<td>2#48</td>
<td></td>
</tr>
<tr>
<td>$000D</td>
<td>19/31</td>
<td></td>
</tr>
<tr>
<td>$000E</td>
<td>22/20</td>
<td></td>
</tr>
<tr>
<td>$000F</td>
<td>25/20</td>
<td></td>
</tr>
<tr>
<td>$0010</td>
<td>136/17</td>
<td></td>
</tr>
<tr>
<td>$0011</td>
<td>2#449</td>
<td></td>
</tr>
<tr>
<td>$0012</td>
<td>24/5</td>
<td></td>
</tr>
<tr>
<td>$0013</td>
<td>24/61</td>
<td></td>
</tr>
<tr>
<td>$0014</td>
<td>26/13</td>
<td></td>
</tr>
<tr>
<td>$0015</td>
<td>127/14</td>
<td></td>
</tr>
<tr>
<td>$0016</td>
<td>129/15</td>
<td></td>
</tr>
<tr>
<td>$0017</td>
<td>152/27</td>
<td></td>
</tr>
<tr>
<td>$0018</td>
<td>141/15</td>
<td></td>
</tr>
<tr>
<td>$0019</td>
<td>144/25</td>
<td></td>
</tr>
<tr>
<td>$001A</td>
<td>146/22</td>
<td></td>
</tr>
<tr>
<td>$001B</td>
<td>150/13</td>
<td></td>
</tr>
<tr>
<td>$001C</td>
<td>41#</td>
<td></td>
</tr>
<tr>
<td>$001D</td>
<td>2#10</td>
<td></td>
</tr>
<tr>
<td>$001E</td>
<td>17/6</td>
<td></td>
</tr>
<tr>
<td>$001F</td>
<td>127#</td>
<td></td>
</tr>
<tr>
<td>$0020</td>
<td>2#50</td>
<td></td>
</tr>
<tr>
<td>$0021</td>
<td>22/24</td>
<td></td>
</tr>
<tr>
<td>$0022</td>
<td>145/16</td>
<td></td>
</tr>
<tr>
<td>$0023</td>
<td>10/57</td>
<td></td>
</tr>
<tr>
<td>$0024</td>
<td>123/32</td>
<td></td>
</tr>
<tr>
<td>$0025</td>
<td>126#</td>
<td></td>
</tr>
<tr>
<td>$0026</td>
<td>128/14</td>
<td></td>
</tr>
<tr>
<td>$0027</td>
<td>128/27</td>
<td></td>
</tr>
<tr>
<td>$0028</td>
<td>133/12</td>
<td></td>
</tr>
<tr>
<td>$0029</td>
<td>133/23</td>
<td></td>
</tr>
<tr>
<td>$002A</td>
<td>152/34</td>
<td></td>
</tr>
<tr>
<td>$002B</td>
<td>152/36</td>
<td></td>
</tr>
<tr>
<td>$002C</td>
<td>152/41</td>
<td></td>
</tr>
<tr>
<td>$002D</td>
<td>152/43</td>
<td></td>
</tr>
<tr>
<td>$002E</td>
<td>145/14</td>
<td></td>
</tr>
<tr>
<td>$002F</td>
<td>146#</td>
<td></td>
</tr>
<tr>
<td>$0030</td>
<td>17/4</td>
<td></td>
</tr>
<tr>
<td>$0031</td>
<td>120/16</td>
<td></td>
</tr>
<tr>
<td>$0032</td>
<td>123/22</td>
<td></td>
</tr>
<tr>
<td>$0033</td>
<td>125/29</td>
<td></td>
</tr>
<tr>
<td>$0034</td>
<td>145#</td>
<td></td>
</tr>
<tr>
<td>$0035</td>
<td>2#15</td>
<td></td>
</tr>
<tr>
<td>$0036</td>
<td>116/16</td>
<td></td>
</tr>
<tr>
<td>$0037</td>
<td>118/17</td>
<td></td>
</tr>
<tr>
<td>$0038</td>
<td>119/24</td>
<td></td>
</tr>
<tr>
<td>$0039</td>
<td>119/27</td>
<td></td>
</tr>
<tr>
<td>$003A</td>
<td>37/13</td>
<td></td>
</tr>
<tr>
<td>$003B</td>
<td>42#</td>
<td></td>
</tr>
<tr>
<td>$003C</td>
<td>2#10</td>
<td></td>
</tr>
<tr>
<td>$003D</td>
<td>77/52</td>
<td></td>
</tr>
<tr>
<td>$003E</td>
<td>92/25</td>
<td></td>
</tr>
<tr>
<td>$003F</td>
<td>81#7</td>
<td></td>
</tr>
<tr>
<td>$0040</td>
<td>88#34</td>
<td></td>
</tr>
<tr>
<td>$0041</td>
<td>1#42</td>
<td></td>
</tr>
<tr>
<td>$0042</td>
<td>115/11</td>
<td></td>
</tr>
<tr>
<td>$0043</td>
<td>1#44</td>
<td></td>
</tr>
<tr>
<td>$0044</td>
<td>115/13</td>
<td></td>
</tr>
<tr>
<td>$0045</td>
<td>164#</td>
<td></td>
</tr>
<tr>
<td>$0046</td>
<td>7/20</td>
<td></td>
</tr>
<tr>
<td>$0047</td>
<td>1#45</td>
<td></td>
</tr>
<tr>
<td>$0048</td>
<td>117/15</td>
<td></td>
</tr>
<tr>
<td>$0049</td>
<td>15/36</td>
<td></td>
</tr>
<tr>
<td>$004A</td>
<td>115#</td>
<td></td>
</tr>
<tr>
<td>$004B</td>
<td>2#10</td>
<td></td>
</tr>
<tr>
<td>$004C</td>
<td>28#</td>
<td></td>
</tr>
<tr>
<td>$004D</td>
<td>29/11</td>
<td></td>
</tr>
<tr>
<td>$004E</td>
<td>16/10</td>
<td></td>
</tr>
<tr>
<td>$004F</td>
<td>28#</td>
<td></td>
</tr>
<tr>
<td>$0050</td>
<td>3#10</td>
<td></td>
</tr>
<tr>
<td>$0051</td>
<td>67/57</td>
<td></td>
</tr>
<tr>
<td>$0052</td>
<td>67/59</td>
<td></td>
</tr>
<tr>
<td>$0053</td>
<td>3#15</td>
<td></td>
</tr>
<tr>
<td>$0054</td>
<td>116/16</td>
<td></td>
</tr>
<tr>
<td>$0055</td>
<td>118/17</td>
<td></td>
</tr>
<tr>
<td>$0056</td>
<td>119/24</td>
<td></td>
</tr>
<tr>
<td>$0057</td>
<td>119/27</td>
<td></td>
</tr>
<tr>
<td>$0058</td>
<td>42#</td>
<td></td>
</tr>
<tr>
<td>$0059</td>
<td>2#48</td>
<td></td>
</tr>
<tr>
<td>$005A</td>
<td>70/30</td>
<td></td>
</tr>
<tr>
<td>$005B</td>
<td>9/6</td>
<td></td>
</tr>
<tr>
<td>$005C</td>
<td>153/24</td>
<td></td>
</tr>
<tr>
<td>$005D</td>
<td>16/58</td>
<td></td>
</tr>
<tr>
<td>$005E</td>
<td>130#</td>
<td></td>
</tr>
<tr>
<td>$005F</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$0060</td>
<td>16/59</td>
<td></td>
</tr>
<tr>
<td>$0061</td>
<td>131#</td>
<td></td>
</tr>
<tr>
<td>$0062</td>
<td>152/38</td>
<td></td>
</tr>
<tr>
<td>$0063</td>
<td>15/50</td>
<td></td>
</tr>
<tr>
<td>$0064</td>
<td>132#</td>
<td></td>
</tr>
<tr>
<td>$0065</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$0066</td>
<td>16/50</td>
<td></td>
</tr>
<tr>
<td>$0067</td>
<td>63#</td>
<td></td>
</tr>
<tr>
<td>$0068</td>
<td>4#10</td>
<td></td>
</tr>
<tr>
<td>$0069</td>
<td>114/13</td>
<td></td>
</tr>
<tr>
<td>$006A</td>
<td>18/56</td>
<td></td>
</tr>
<tr>
<td>$006B</td>
<td>89/5</td>
<td></td>
</tr>
<tr>
<td>$006C</td>
<td>2#42</td>
<td></td>
</tr>
<tr>
<td>$006D</td>
<td>22/25</td>
<td></td>
</tr>
<tr>
<td>$006E</td>
<td>23/13</td>
<td></td>
</tr>
<tr>
<td>$006F</td>
<td>24/15</td>
<td></td>
</tr>
<tr>
<td>$0070</td>
<td>24/34</td>
<td></td>
</tr>
<tr>
<td>$0071</td>
<td>25/33</td>
<td></td>
</tr>
<tr>
<td>$0072</td>
<td>2#7</td>
<td></td>
</tr>
<tr>
<td>$0073</td>
<td>72/23</td>
<td></td>
</tr>
<tr>
<td>$0074</td>
<td>72/25</td>
<td></td>
</tr>
<tr>
<td>$0075</td>
<td>2#5</td>
<td></td>
</tr>
<tr>
<td>$0076</td>
<td>28/2</td>
<td></td>
</tr>
<tr>
<td>$0077</td>
<td>27/14</td>
<td></td>
</tr>
<tr>
<td>$0078</td>
<td>42/12</td>
<td></td>
</tr>
<tr>
<td>$0079</td>
<td>43/3</td>
<td></td>
</tr>
<tr>
<td>$007A</td>
<td>44/5</td>
<td></td>
</tr>
<tr>
<td>$007B</td>
<td>44/14</td>
<td></td>
</tr>
<tr>
<td>$007C</td>
<td>47/3</td>
<td></td>
</tr>
<tr>
<td>$007D</td>
<td>48/5</td>
<td></td>
</tr>
<tr>
<td>$007E</td>
<td>54/5</td>
<td></td>
</tr>
<tr>
<td>$007F</td>
<td>57/17</td>
<td></td>
</tr>
<tr>
<td>$0080</td>
<td>61/4</td>
<td></td>
</tr>
<tr>
<td>$0081</td>
<td>66#</td>
<td></td>
</tr>
<tr>
<td>$0082</td>
<td>77#</td>
<td></td>
</tr>
<tr>
<td>$0083</td>
<td>82/16</td>
<td></td>
</tr>
<tr>
<td>$0084</td>
<td>33#</td>
<td></td>
</tr>
<tr>
<td>$0085</td>
<td>38/71</td>
<td></td>
</tr>
<tr>
<td>$0086</td>
<td>32#30</td>
<td></td>
</tr>
<tr>
<td>$0087</td>
<td>37/22</td>
<td></td>
</tr>
<tr>
<td>$0088</td>
<td>33#</td>
<td></td>
</tr>
<tr>
<td>$0089</td>
<td>33/18</td>
<td></td>
</tr>
<tr>
<td>$008A</td>
<td>38/20</td>
<td></td>
</tr>
<tr>
<td>$008B</td>
<td>32#</td>
<td></td>
</tr>
<tr>
<td>$008C</td>
<td>32/30</td>
<td></td>
</tr>
<tr>
<td>$008D</td>
<td>37/21</td>
<td></td>
</tr>
<tr>
<td>$008E</td>
<td>38#11</td>
<td></td>
</tr>
<tr>
<td>$008F</td>
<td>40/16</td>
<td></td>
</tr>
<tr>
<td>$0090</td>
<td>1#27</td>
<td></td>
</tr>
<tr>
<td>$0091</td>
<td>32/7</td>
<td></td>
</tr>
<tr>
<td>$0092</td>
<td>33/13</td>
<td></td>
</tr>
<tr>
<td>$0093</td>
<td>45#</td>
<td></td>
</tr>
<tr>
<td>$0094</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$0095</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$0096</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$0097</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$0098</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$0099</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$009A</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$009B</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$009C</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$009D</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$009E</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>$009F</td>
<td>2#</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Location</td>
<td>Size</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>VCRS</td>
<td>04:2</td>
<td>33/10</td>
</tr>
<tr>
<td>VCLRS</td>
<td>051F</td>
<td>33/14</td>
</tr>
<tr>
<td>VCLR</td>
<td>0540</td>
<td>33/8</td>
</tr>
<tr>
<td>VCLL</td>
<td>0545</td>
<td>1#33</td>
</tr>
<tr>
<td>VCHOME</td>
<td>04:8</td>
<td>33/15</td>
</tr>
<tr>
<td>VLCF</td>
<td>0549</td>
<td>33/9</td>
</tr>
<tr>
<td>VCLRS</td>
<td>00:1A</td>
<td>1#22</td>
</tr>
<tr>
<td>VCMCR</td>
<td>354C</td>
<td>33/11</td>
</tr>
<tr>
<td>VCMCP</td>
<td>3639</td>
<td>33/12</td>
</tr>
<tr>
<td>VDELC</td>
<td>00:75</td>
<td>1#30</td>
</tr>
<tr>
<td>VDELL</td>
<td>0552</td>
<td>1#32</td>
</tr>
<tr>
<td>VDUNL</td>
<td>04:13</td>
<td>35#12</td>
</tr>
<tr>
<td>VEGAS</td>
<td>00:47</td>
<td>1#39</td>
</tr>
<tr>
<td>VEHIC</td>
<td>02:8</td>
<td>1#35</td>
</tr>
<tr>
<td>VEUH</td>
<td>00:50</td>
<td>1#37</td>
</tr>
<tr>
<td>VFLD</td>
<td>FFEA</td>
<td>1#68</td>
</tr>
<tr>
<td>VGRAPH</td>
<td>3554</td>
<td>31/7</td>
</tr>
<tr>
<td>VHASK</td>
<td>04:59</td>
<td>31/9</td>
</tr>
<tr>
<td>VHADF</td>
<td>34:55</td>
<td>31/9</td>
</tr>
<tr>
<td>VHME</td>
<td>00:1E</td>
<td>1#23</td>
</tr>
<tr>
<td>VINC</td>
<td>00:51</td>
<td>1#29</td>
</tr>
<tr>
<td>VINL</td>
<td>00:45</td>
<td>1#31</td>
</tr>
<tr>
<td>VLODS</td>
<td>06:0E</td>
<td>16/48</td>
</tr>
<tr>
<td>VLOIR</td>
<td>06:12</td>
<td>16/49</td>
</tr>
<tr>
<td>VLL</td>
<td>00:50</td>
<td>1#49</td>
</tr>
<tr>
<td>VLL</td>
<td>00:50</td>
<td>1#49</td>
</tr>
<tr>
<td>VLOCK</td>
<td>00:23</td>
<td>1#25</td>
</tr>
<tr>
<td>VNDRM</td>
<td>04:37</td>
<td>31/5</td>
</tr>
<tr>
<td>VDUT80</td>
<td>05:5E</td>
<td>38/17</td>
</tr>
<tr>
<td>VDUTBS</td>
<td>35:72</td>
<td>49/19</td>
</tr>
<tr>
<td>VDUT90</td>
<td>35:73</td>
<td>42/45</td>
</tr>
<tr>
<td>VDUT96</td>
<td>35:75</td>
<td>49#37</td>
</tr>
<tr>
<td>VDUT97</td>
<td>05:78</td>
<td>33/15</td>
</tr>
<tr>
<td>VRTOFF</td>
<td>EE:2F</td>
<td>38/20</td>
</tr>
<tr>
<td>VS40</td>
<td>00:53</td>
<td>1#28</td>
</tr>
<tr>
<td>VSHG</td>
<td>00:57</td>
<td>1#38</td>
</tr>
<tr>
<td>VSHM</td>
<td>00:29</td>
<td>1#34</td>
</tr>
<tr>
<td>VSHUL</td>
<td>00:56</td>
<td>1#36</td>
</tr>
<tr>
<td>VJODER</td>
<td>04:57</td>
<td>31/10</td>
</tr>
<tr>
<td>VJNRK</td>
<td>04:58</td>
<td>31/11</td>
</tr>
<tr>
<td>VJNRA</td>
<td>04:5F</td>
<td>31/12</td>
</tr>
<tr>
<td>VJNHAG</td>
<td>04:53</td>
<td>31/13</td>
</tr>
<tr>
<td>VJNLK</td>
<td>00:22</td>
<td>1#26</td>
</tr>
<tr>
<td>#8007</td>
<td>02:9C</td>
<td>15/9</td>
</tr>
<tr>
<td>WRUSF</td>
<td>0E:38</td>
<td>12/20</td>
</tr>
<tr>
<td>WRITR</td>
<td>0D:37</td>
<td>15/24</td>
</tr>
<tr>
<td>WSEC</td>
<td>0C:23</td>
<td>16/27</td>
</tr>
</tbody>
</table>
ASSE 0000 FF00
C002 EF00 0000
DATA EF00 0000

Date: 6/4/82
Author: Roger W. Chapman
Title: Osborne CP/M 2.2 CA10S Rev 1.4

NOTE: FOR USE WITH OCC10S.AST ONLY

MASTER *ASM
ASSY *ASM
LISTING *PRN
MASTER *COM
ASSY *COM

by Roger W. Chapman

Copyright 1982, Osborne.
This product is a copyright program product of Osborne and is supplied for use with the Osborne.

VERSI: 22

LINK OCC10S5.ASM : Jump Table
LINK OCC1025.ASM : Key translation & initialization values
LINK OCC1095.ASM : CP/M disk tables
LINK OCC1335.ASM : ROM call interface
LINK OCC1345.ASM : Non data transfer disk
LINK OCC1555.ASM : Cold and warm boot
LINK OCC1555.ASM : Disk data transfer I/O
LINK OCC1375.ASM : Utility routines
LINK OCC1755.ASM : Iobyte dispatch table
LINK OCC1815.ASM : Ibis ram definitions
LINK OCC1925.ASM : Common ram definitions
SOCIM For Assembler ver 3.3E <1/55/7=9:92 Page 2
Osborne CP/M 2.2 CBMOS Rev 1.4 C:OCCAM1315.ASM

= 0033
= 0300
= 0010
= 0008
= 0005
= 0002
= 0039
= 0000
= 0000
= 0001

MSG "Disk constants which define the
: disk types and maximum storage capacity of each
: disk type."

= 0034
= 0003
= 0001
= 0001
= 0000

= 002E
= 003A
= 0033
= 002C
= 0000

= 0001
= 0000

= 0035
= 0002
= 0010
I = 256 I
I = 512 I
I = 1024 I
I = 2048 I
I = 4096 I
I = 8192 I
I = 16384 I
I = 32768 I
I = 65536 I

; CP/M to host disk constants

= 0000
= 0010
= 0008

= 1024
= 2048/128
= 1024/128

= 0124
= 00Ch
= 01Ch
= 02h
= 03h

; Blocking/Declocking buffer size
; Sectors in floppy disk block (Osborne single density block size = 2KB)
; Sectors in floppy disk block (Osborne double density block size = 1KB)

; Assemblying BIOS for LWA of "'", LWAEM, etc.

: ROM equates.

= 0000
= 0001

= 0
= 1

: Port to enable ROM
: Port to disable ROM

; Assemblying BIOS for LWA of FFFFh;
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 = Drive parameter block.
- 16 bits = Check vector.
- 16 bits = Allocation vector.

= 0000
NOSK: 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

DPGEN MACRO TYPE,XKATE,DIRSIZ,ALVSZ,OFFSET
NOSK: SET NOSK+1
DW R2
DW O+O+3
DW %3
DW %4
DW CSV+CSV
DW ALV+ALVSZ
NOFDD: SET NOFDD+1
CSVSZ: SET CSVSZ+(32/4)
ALVSZ: SET ALVSZ+((32*7+7)/8)
ENDM

Macro for generating the Disk Parameter blocks:

Disk type definition blocks for each particular node:

The format of these areas are as follows:

- 8 bit = disk type code
- 16 bit = Sectors per track
- 8 bit = Block shift
- 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

DPGEN MACRO TYPE,SPT,BSH,BSM,EXM,DSM,DIRSIZ,ALVMSK,OFFSET
SB %1
DW R2
SB %3, %4, R5
DW %6-1, %7-1, %8 (R8)
DW (S7+3)/4
DW R9
ENDM
The following jump table defines the entry points into the BIOS for use by CP/M and other external routines; therefore the order of these jumps cannot be modified. The location of these jumps can only be modified by 4096 locations, which is a restriction of MOSCPM.

```
   DDDD = $100D

ORG $A000

E100 C3ED0E4 JMP CSO0T ;Cold boot
E103 C31355 JMP $8007 ;Warm boot
E106 C32E77 JMP CSTRT ;Console status input
E109 C33387 JMP CNTN ;Console input
E10C C3338E7 JMP CONOUT ;Console output
E10F C33FE7 JMP LIST ;List output
E112 C35577 JMP PUNCH ;Punch output
E115 C35677 JMP READER ;Reader input
E118 C356D7 JMP HME ;Set track to zero
E11B C350E3 JMP SELDISK ;Select disk unit
E11E C3B8E4 JMP SETTRK ;Set track
E121 C3C2E4 JMP SETSEC ;Set sector
E124 C3D3E4 JMP SETMA ;Set Disk Memory Address
E127 C3E1E5 JMP READ ;Read from disk
E12A C3EE65 JMP WRITE ;Write to disk
E12D C3F3E7 JMP LISTST ;Return LIST device status
E130 C3FD9E4 JMP SECTR ;Sector translation routine

Extensions

E133 C3FFF2 JMP RMI ;Rom resident call
E136 C3192F JMP RAMIP ;Rom resident call
E139 CD123F CALL ROMCDE ;Rom resident call
E13C CD1243 JMP RAMIP ;Rom resident call

IEEE-488 vectors

E13F CD1243 CALL IEEE2 ;Control Out
E142 CD1243 CALL IEEE2 ;Status In
E145 CD1243 CALL IEEE2 ;Go To Start
E148 CD1243 CALL IEEE2 ;Take Control
E151 CD1243 CALL IEEE2 ;Output Interface Message
E154 CD1243 CALL IEEE2 ;Input Device Message
E157 CD1243 CALL IEEE2 ;Parallel I/O
E15A CD1243 CALL IEEE2 ;Extensions
E15D CD1243 CALL IEEE2 ;Memory-mapped video
E160 C330C1 JMP ACICL ;Hook for serial command port write
E163 C330D8 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, I/O byte value
; and auto horizontal scroll flag

E166 40 IDBYTE DB 40h ; Default to serial printer = 40h
E167 00 PRINTER DB 00h ; Parallel printer = 0h
E168 FF AHSEN DB TRUE ; Auto horizontal scroll enable
E169 55 SERIAL DB S1.120 ; Default baud rate = 1200
E16A 80 SCRSIZE DB 128 ; Default screen size = 128

E16B A5E1 XLTBL DW CNTRL0 ; Fixed length tables
E16C 46E1 DW CNTRL1 ; Contains pointers
E16E 47E1 DW CNTRL2 ; In string
E171 A8E1 DW CNTRL3 ; In decode
E173 49E1 DW CNTRL4 ; Function keys
E175 AE1 DW CNTRL5
E177 AE1 DW CNTRL6
E179 A1E1 DW CNTRL7
E17A AD1 DW CNTRL8
E17D AE1 DW CNTRL9
E17F AE1 DW UP
E181 30E1 DW RIGHT
E183 51E1 DW DOWN
E185 32E1 DW LEFT
E187 33E1 DW EOTBL ; End of table address

E189 C4 IEEEAD DB 04h ; IEEE device address
E18A 00 PRINTGB DB 0 ; Flag indicates if printer has been initialized
E18B 00 INIT DB 00h ; Length of string
E18C 00 001D DS 01 ; Printer initialization string
E18D 01 ACMD DB 01 ; Auto command string
E190 00 CAUTL DB CAUTL ; Length of auto command string
E192 41555453 DB *AUTOT ; Auto command goes here
E194 0037 CAUTL = 0-CAUTL-1
E1A5 30  CNTRL0:  DB  * *
E1A6 31  CNTRL1:  DB  * *
E1A7 32  CNTRL2:  DB  * *
E1A8 33  CNTRL3:  DB  * *
E1A9 34  CNTRL4:  DB  * *
E1AA 35  CNTRL5:  DB  * *
E1AB 36  CNTRL6:  DB  * *
E1AC 37  CNTRL7:  DB  * *
E1AD 38  CNTRL8:  DB  * *
E1AE 39  CNTRL9:  DB  * *
E1AF 03  JP:  DB  K*40h  :Default values
E1B0 0C  XSCFT:  DB  L*40h  :for the cursor
E1B1 0A  DOWNS:  DB  J*40h  :Keys are standard
E1B2 0B  LEFTS:  DB  W*40h  :Values for CP/M

E183  = 0200  DRG  8135+256  :Space reserved for full function
                       :Key decoding and 16 byte auto
                       :boot command
Control Blocks for disk drives

E200 DBHSEN CSK+1, DDXLTS, DIR+1, CPB+1
Drive A:

** 0001 W5SK: SET NOSK+1
E200 +0000 DW DDXLTS
E202 +0000000000 DW 0+3
E204 +8000 DW DIR+1
E206 +9FEB DW DIR+1
E208 +CE9 DW CSV+5V5
E20A +CE9 DW ALV+ALV5Z
** 0001 W5FD0: SET NDFD0+1
** 0010 CSV5Z: SET CSV5Z+(4/4)
** 0010 ALV5Z: SET ALV5Z+(4/4)+7/8

Drive B:

E210 DBHSEN CSK+1, DDXLTS, DIR+1, CPB+1

** 0002 W5SK: SET NOSK+1
E210 +0000 DW DDXLTS
E212 +0000000000 DW 0+3
E214 +9FE1 DW DIR+1
E216 +CE9 DW CSV+5V5
E218 +CE9 DW ALV+ALV5Z
** 0002 W5FD0: SET NDFD0+1
** 0020 CSV5Z: SET CSV5Z+(4/4)
** 0030 ALV5Z: SET ALV5Z+(4/4)+7/8
E220 XTAB: Translation table addresses
E220 000D DW DDXLTS: Double density Osborne translation table address
E222 2CE2 DW XLTS: Single density Osborne
E224 40E2 DW XXXLTS: Xerox translation table address
E226 000D DW IBMXLTS: IBM translation table address
E228 89E2 DW DECKLT: DEC translation table address
E22A 76E2 DW XTRXLTS: 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 0001940A09 DB 0* 1* 4* 5* 9* 12* 13* 16* 17
E236 023306070A DB 2* 3* 6* 7* 10* 11* 14* 15* 18* 19

E240 XXX: Translation table 5 to 1
E240 00050A9F DB 0* 5* 10* 15
E244 0078C11 DB 2* 7* 12* 17
E248 040901 DB 4* 9* 14
E24C 0109010 DB 1* 5* 11* 16
E250 030900 DB 3* 8* 13

= 0000 IBMXLTS: = 0 IBM translation table, 32 translation 1 to 1

E252 DECKLT: DEC TRANSLATION TABLE 2 to 1
E252 0001222509 DB 0* 1* 2* 3* 9* 10* 11* 16* 17* 18* 19* 24* 25* 26* 27* 32* 33* 34* 35
E256 043508270C DB 4* 5* 6* 7* 12* 13* 14* 15* 20* 21* 22* 23* 29* 30* 31

E276 = 0028 XTRXLTS: DS 40: Space for user defined translation
: Disk type definition blocks for each particular node.

E29E  DPBSTART: : Start of disk parameter blocks, used by select disk routine

E29E  DPB01: : Osborne Double density, single sided
E29E  +3C  DB  0SK01
E29E  +2800  DW  8+5
E2A1  +O30700  DB  3+7+3
E2A4  +8003F3000  DW  DIOSM-1,54-1,REV (110000000000000003)
E2AA  +1000  DW  (64+3)/4
E2AC  +0300  DW  3

E2AE  DPB51: : Osborne Single density, single sided.
E2AE  DPB51: : Osborne Single density, single sided.
E2AE  +05  DB  0SKS1
E2AE  +1400  DW  2+10
E2BF  +040F31  DB  4+15+1
E2BA  +20003F0300  DW  S1DSM-1,54-1,REV (1000000000000003)
E2BA  +1000  DW  (64+3)/4
E2BC  +0300  DW  3

E3BE  DPBX2: : Osborne Single density, single sided.
E3BE  DPBX2: : Osborne Single density, single sided.
E3BE  +01  DB  XER0X
E3BE  +1200  DW  1+10
E3C1  +030700  DB  3+7+3
E3C4  +52001F0000  DW  XXDSM-1,32-1,REV (1000000000000003)
E3CA  +0800  DW  (32+3)/4
E3CC  +0300  DW  3

E3CE  DPB13M: : Osborne Single density, single sided.
E3CE  DPB13M: : Osborne Single density, single sided.
E3CE  +4B  DB  18M
E3CE  +2000  DW  48
E3E1  +030700  DB  3+7+3
E3E4  +90003F0300  DW  18MDSM-1,64-1,REV (1100000000000003)
E3EA  +1000  DW  (64+3)/4
E3EC  +0100  DW  1

E3DE  DPBDEC: : Osborne Single density, single sided.
E3DE  DPBDEC: : Osborne Single density, single sided.
E3DE  +38  DB  DEC
E3DF  +2400  DW  48
E3E1  +030700  DB  3+7+3
E3E4  +AA003F3000  DW  DECDSM-1,64-1,REV (1100000000000003)
E3EA  +1000  DW  (64+3)/4
E3EC  +0200  DW  2

E3EE  DPBXTR: : Osborne Single density, single sided.
E3EE  +D10  DW  16

= 0006  NUMDPB: = (E-DPBSTART)/10H
<table>
<thead>
<tr>
<th>E2FE</th>
<th>BA7BEF</th>
<th>LD</th>
<th>A*ROM&amp;R4M</th>
</tr>
</thead>
<tbody>
<tr>
<td>E301</td>
<td>4F</td>
<td>MOV</td>
<td>C1A</td>
</tr>
<tr>
<td>E302</td>
<td>8D79</td>
<td>JTAC</td>
<td>A</td>
</tr>
<tr>
<td>E304</td>
<td>9E1</td>
<td>PDP</td>
<td>IY</td>
</tr>
<tr>
<td>E306</td>
<td>0E1</td>
<td>PDP</td>
<td>IX</td>
</tr>
<tr>
<td>E308</td>
<td>E1</td>
<td>PDP</td>
<td>HL</td>
</tr>
<tr>
<td>E309</td>
<td>01</td>
<td>PDP</td>
<td>DE</td>
</tr>
<tr>
<td>E30A</td>
<td>1</td>
<td>PDP</td>
<td>EC</td>
</tr>
<tr>
<td>E30B</td>
<td>F1</td>
<td>PDP</td>
<td>AF</td>
</tr>
<tr>
<td>E30C</td>
<td>ED7B5FEF</td>
<td>LD</td>
<td>SP,IESTK</td>
</tr>
<tr>
<td>E310</td>
<td>FS</td>
<td>EI</td>
<td></td>
</tr>
<tr>
<td>E311</td>
<td>09</td>
<td>RET</td>
<td></td>
</tr>
</tbody>
</table>

ROMR1: Exits ROM resident interrupt routine.

* A*ROM&R4M is printed in red.

Ipart
Set ROM or RAM Usage

Reset to interrupt entry stack.
E312       ROMCD1:
             Calls ROM resident processor
             Entry DE = resident processor to call biased
             by C810S jump vector.
             NOTE: ROM jump vector must match C810S vector
             Entry at ROMCD1 with low digit of C810S vector in reg E
E312       D1
E313       7B
E314       D603
E316       5F
E317       1601
E318       LDH Dhigh (ROMVEC)
E319
E319
E319       ROMJMP:
             Entry here to jump to ROM function code directly
             Entry DE = ROM jump address
             Index, HL, IX = any parameters
E319       FDE5 $      PUSH IY     ;Save user Index registers
E319       DDE5 $      PUSH IX
E319       F3 $       DI       ;Set up local stack for ROM
E319       08 $       EXX
E31F       21000D
E322       39
E323       31C1EF
E326       E5
E327       09 $       EXX
E328       CDD7E8
E329       CDDA63
E32E       CDE2E8
E331       FDE1 $      POP IY     ;Restore old stack pointer
E333       F0F9 $      MOV SP, IY
E335       DDE1 $      POP IX
E337       FDE1 $      POP IY
E339       C9       RET

E33A       GDROM:
             This routine is used to simulate a CALL (DE) instruction
E33A       D5
E33B       FDE1 $      PUSH DE     ;ROM jump address to IY
E33C       FDE1 $      POP IY
E33D       FDE9 $      JMP [IY]     ;Go to ROM, ROM will RET to next instruction
Returns disk to zero. This routine sets the track number to zero. The current host disk buffer is flushed to the disk.

CALL FLUSH
XRA A
STO A+HSTACT
STO A+UNACT
STO A+SEKTRK
STO A+SEKTRK+1
RET
SELDSK:
; Selects disk drive for next transfer.
ENTRY C = disk selection value (0..15).
; DE and I = 0, first call for this disk.
EXIT HL = 0: if drive not selectable.
; HL = DPH address if drive is selected.

E350 0F2E5 $   PROC   IX     ;Save user IX
E352 79       MOV   A\#C
E353 F002     CPI   NODSK
E355 02A6E3    JNC    SELD
E358 3213EF    STD   \#SELDSK
E35B 59       MOV   L\#C
E35C 2600     WVI   H\#O
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 70       MOV   A\#E  \get initial bit
E363 110DE2    LDK   DE\#DP4SE
E366 19       ADD   HL\#DE  \#HL = DP4 address
E367 2206E9    STD   HL\#SAVDPH
E36A 5F       MOV   E\#A  \restore initial bit
E36B 2DFAE3    CALL   CHKSEL
E36E 02A6E3    JNZ    SELECT
E371 110A3D    LDK   DE\#10  \form dob address
E374 19       ADD   HL\#DE  \to get type
E375 5E       LD    E\#HLJ
E376 23       INC   HL
E377 56       LD    D\#(HL)
E378 18       DEC   DE
E379 1A       LD    A\#[DE]  \get disk type
E37A 325CE9    STD   A\#SEKTYP
E37B 2ACCEF    LD    HL\#SEQ  \set current sequence count
E37D 007500    $   STD   L\#[IX+0]
E37F 037401    $   STD   H\#[IX+1]  \store seq # in LASTA or LASTB
E382 2A06E9    LD    HL\#SAVDPH
E383 0DE1      $   POP   IX
E385 C9       RET
: Select disk error handling

E30C 11D8E3  SELECT: LDK DE+ERROR
E39F 6601  ANI 1 YES diskette unformatted?
E391 2003 +E396%  JRNZ SELECT: YES
E393 1132E3  LDK DE+ERROR NO, get density error message

E396 3A18E3  SELECT: LD A+SECK0SK
E399 CA41  ADD A+*A'
E39B 3207E3  STD A+DRV
E39E 32F9E3  STD A+DRV1
E3A1 C011E7  CALL PRINT 1Print appropriate message on console

E3A4 0DE1  $ SELD: PDV IX 1Restore user IX
E3A6 21000D  LDK HL:0
E3A9 3A0400  LDA CDISK
E3AC 91  SUB C
E3AD 00  RNZ
E3AE 320400  STD A+CDISK
E3B1 C9  RET

E3B2 25  DE+ERROR: DB DENL
E3B3 0DA4351E  DB CR+LF'+Can't recognize diskette on drive'
E3D7 = 0001  DRV: DS 1
   = 0025  DENL: = 0-DE+ERROR-1

E3D8 21  FORERR: DB FORL
E3D9 0DA4555E66  DB CR+LF'+Unformatted diskette on drive'
E3F9 = 0001  DRV1: DS 1
   = 2021  FORL: = 0-FORERR-1
E3FA

CHKSCL:

: Determines if new CB8 should be initialized
:
: ENTRY C = disk selection value ($C-15)
: E & D1 = 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 E5
E3FB E5
E3FC D0210099 $
LDK IX+LASTA ; Get last count for selected drive
E400 79
MOV A,C
; Current drive to A - reg
E401 87
ORA A
; Is this drive A?
E402 2804 ^E403$ JAN Z ; CHKSEQ
; Yes, check sequence number
E404 DD23 $ INC IX
; No, form address for 3
E406 DD23 $ INC IX

E408 DD400 $ CHKSEQ: 
LD C*(IX+D) ; Get last sequence number for this drive
E408 DD400 $ LD B*(IX+1)
E40E 2AC4EF $ 
LD $L+SQ ; Set current sequence number
E411 DD42 $ 
SBC HL+3C
; Compare seq #'s
E413 C1 
PDP BC
E414 DD45 $ 
PUSH IX

E416 CB43 $ 
RIT 0xE ; First call for this disk?
E418 2800 ^E427$ 
Jrz 110 ; Yes, fill in CB8

E41A 7C 
MOV A+H ; No, check elapsed time
E41B 97 
ORA A
; Elapsed time < 4 sec
E41C 280E ^E42C$ 
Jrz GRET ; Yes, good return
E41E FE02 
CP1 DZ
E420 3005 ^E427$ 
JNC 110
E422 7D 
MOV A+L ; Elapsed time > 5 sec
E423 FE40 
CP1 0ADH
E425 3005 ^E42C$ 
JRC GRET ; No, indicate good return

E427 CD31E4 
110: CALL GDEN
E42A 1BD1 ^E42D$ 
JR RETURN
E42C AF 
GRET: XRA A
E42D DDEI $ RET: POP IX
E42F C1 
POP ML
E430 89 
RET
E47F D0FE30  $    LD    A,[IX+2]  
E482 77            STD   A,[HL]    
E483 23            INX   HL      
E484 D0FE31  $    LD    A,[IX+1]  
E487 77            STD   A,[HL]    
E488 AF            XOR   A      ; Indicate good return  
E489 C9            RET

; Error return section
E48A 3E01            SEI01:  MVP   A+1      ; Indicate unformatted disk Z-flag = clear, A = 1  
E48C C9            RET
E48D F632            SELD2:  ORI   02     ; Indicate unrecognizable disk Z-flag = clear, A = 2  
E48F C9            RET
E490 SETDEN:
  Sets density of selected disk
  ENTRY SEKDSK = Current drive
  EXIT C = TYPE

  Undef = 0
  Bytes/sector <=------------>
  Sides <=------------------>
  Density <=----------------->

  B = # of physical sectors per track
  A = 3, good return
  A <> 0, error return

= 0130  SENDEN: = 130H

E490 PROC
E490 3A17EF  LD  A,SOISK  ;Save current value
E493 3208EF  STD A,TEMDSK  ;of SDISK
E496 3A03EF  LD  A,SAVTYP  ;and SAVTYP
E499 3239EF  STD A,TEMTP  ;(SOISK & SAVTYP are used by SENDEN)
E49C 3A18EF  LD  A,SEKDSK  ;Disk to be select
E49F 3217EF  STD A,SOISK  ;in SDISK (parameter to SENDEN)
E4A2 113001  LDX DE,SENDEN  ;Call SENDEN
E4A5 0D19EF  CALL ROMJMP
E4AB 3A08EF  LD  A,TEMDSK  ;Restore caller's value of SDISK
E448 3217EF  STD A,SOISK
E44E 3A00EF  LD  A,SAVTYP  ;Exit TYPE parameter
E451 4F  MOV C,A  ;into C - Reg
E482 3A09EF  LD  A,TEMTP  ;Restore caller's SAVTYP
E485 3230EF  STD A,SAVTYP
E488 CO  RNZ ERRET  ;Error return, flag set by SENDEN
E489 4F  XRA A  ;Indicate good return
E48A C9  RET
**SETTRK:**
- Sets track number. The track number is saved for later use during a disk transfer operation.
- ENTRY BC = track number.

```
E48B  SETTRK:
\nE48B  ED4319EF $  STD  RC,SETRK  ;Set track
E48B  ED4335EF $  LHL  UTRK
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  JMP  SUVACT  ;If same track
```

**SUVAFT:**
- Clear unallocated block count (force pre-reads).

```
E4C9  SUVACT:
\nE4C9  AF  XRA  A  ;IA = 0
E4CA  3252EF $  STD  A+UNAVT  ;Clear unallocated block count
E4CD  C9  RET
```

**SETSEC:**
- Set the sector for later use in the disk transfer. No actual disk operations are performed.
- Entry BC = sector number.

```
E4CE  SETSEC:
\nE4CE  79  MOV  A+C
E4CE  3220EF $  STD  A+TSMSEC  ;sector to seek
E4D2  C9  RET
```

**SETDMA:**
- Sets disk memory address for subsequent disk read or write routines. This address is saved in DAMAD until the disk transfer is performed.
- ENTRY BC = disk memory address.
- EXIT DAMAD = BC.

```
E4D3  SETDMA:
\nE4D3  ED4311EF $  STD  BC,DMAAD1
E4DD  C9  RET
```
E4DB SECTRN:
:: Translates sector number from logical to physical
::
:: ENTRY DE = 0, no translation required.
:: DE = translation table address.
:: BC = sector number to translate.
::
:: EXIT ML = translated sector.

E4DB 3A55EF LDA UNASEC
E4DB 39 CMP C
E4DC C4C9E4 CNZ GUNGAT ;If sectors do not match
E4DF 79 MOV A+C
E4E0 3256EF STD A+LOGSEC
E4E3 69 MOV L+C
E4E4 60 MOV H+B
E4E5 78 MOV A+E ;Check if translation is required
E4E6 32 DRA D ;None required, return
E4E7 C8 RZ
E4E8 19 ADD ML,DE ;Translation required
E4E9 6E MOV L+M
E4EA 2600 MOV H+D
E4EC C9 RET
BOOT C/P/M from disk:

The #BOOT 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:

1. 2: Set to the warmstart jmp vector.
2. 3: Set to the initial I/O byte value.
3. 4: Default and logged on drive.
4. 5: Set to a jump to DDS.

The #BOOT entry point gets control when a warm start occurs, a C/P/M from the console, a jump to DDS (function 01), or a jump to location zero. The #BOOT routine reads the C/P/M and DDS from the appropriate disk sectors.

#BOOT must also re-initialize locations 2, 3, and 5, 6, 7.

The #BOOT routines exit with the C register set to the appropriate drive selection value. The exit address is to the C/P/M routine.

E4ED
E4ED 3100C8
E4F0 3100C8
E4F8 3100E8
E4FA 310130
E4FC 310160
E4FD C13E11
E500 3A6A11
E503 326CEF
E50A 3A39E1
E50C 3E0D30
E510 3C
E511 180E ^E511
E513 3100CB
E516 303FE3
E519 110301
E51C 3019E3
E51F 3E02
E521 9CCP;
E521 F5
E522 918000
E524 201000
E52C CD28
E55A C2M0;
E525  C03E4 CALL SETOWA
E528  2190EA LDS ML+MSTBF
E52B  220E6F STS ML+MADR :set ROM DMA address

; Clear console control ESC cell
E52E  4F XRA A
E52F  32B9EF STD A+ESCH :clear ESC

; Set up low core pointer cells
E532  0E2C3 LDS A+OC3h :Store jumps in low memory
E534  320000 STD A+0
E537  320300 STD A+5
E53A  2103E1 LDS ML+BIOS+3
E53D  220100 STD ML+1
E540  210633 LDS ML+BIOS
E543  220500 STD ML+6

E546  2190E1 LDS ML+CAUTO
E549  01 PDP RC :cool/warm indicator in c
E54A  30E11 LDS A+ACWO
E54D  40 AND B
E54E  2844 A+59A$ JAZ DONE

E550  7E LO A+CHL
E551  87 DRA A
E552  2846 A+59A$ JRZ DONE

E554  107C8 LDS DE+CCP+7
E557  0600 LDS S+0
E559  4F MOV C+A
E55A  ED9D $ LDIR:"Move command line buffer
E55C  110000 LDS DE+0
E55F  1047 A+54A$ JR DDNE1

E561  38 SIGNN DB SIGNL :Length of signon message
E562  1A DB *ZL-40n
E563  4735Z6F7 DB *Osborne Computer System'
E57A  000A DB CR+LF
E57C  5339 DB MSIZE/10+*D*+MSIZE mod 10 + '0'
E57E  600B43502F DB *CR+MOV*VERS/10+"*",*VERS mod 10 + "G"
E580  000A34249 DB CR+LF,*C1015 14+*CR+LF
   = 038 SIGNL = 0-SIGNN-1
E59A  3E02 DOWEI LDS A+2
E59C  88 CMP B
E59D  2806 A+54A$ JRZ DDNE0
E59F  1161E5 LDS DE+SIGNV
E5A2  CD11E7 CALL PRINT
E5A5  110330 DOWEI LDS DE+3

E5A8  210DCB DOWEI LDS ML+GSP
E5AB  19 ADD ML+CD
E5AC  3A3400 LDS A+CDISK
E5AF  4F MOV C+A
E5B0  99 JMP EHL
READ:
1: A = 0, successful read operation
2: A = 1, unsuccessful read operation
3: Z bit = 1, successful read operation
4: Z bit = 0, unsuccessful read operation

E581 PROC
E582 CD06E7 CALL MVINFO
E584 AF CALL XRA A
E586 32EE6 CALL STD A+UNACVT
E588 CD44E6 CALL FILL
E58A E1 POP HL
E58C 01 POP DE
E590 310000 LDK BC,128
E592 ED80 LD A,ERFLAG
E594 97 GRA A
E596 C8 RZ
E598 AF XRA A
E59A 3250EF STO A,AHSTACT
E59C F601 DR1 1
E5C0 C9 RET

; Move information for transfer
; Set flag to force a read
; Clear sector counter
; Fill buffer with data
; Move 128 bytes
; If no error
; Clear host active (A = 0)
; Indicate error
ESCE

WRITE:

the selected 128 byte CP/M sector.

ENTRY C = 3, 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 = 3, write was successful.
A = 1, write was unsuccessful.
Z bit = 1, write was successful.
Z bit = 3, write was unsuccessful.

ESCE  CD5E7  CALL  MVINFO  "Move information for transfer"
          STD  WRTYPE  "Write type in c"
ESDE  29EF  STD  A+1
ESDF  FEO2  CPI  WRTUL  "If write to allocated"
ESDG  21A4  JRZ  WRT2  "Check for 2K block size"
ESDE  20E5  LD  A+SEKTP  "Type = Osborne single density (2K block size)"
          CPI  S  "Otherwise 1K block size"
ESDC  13D0  LDK  A+2048/128  "If write to allocated"
ESDE  28E2  JRZ  WRT1  "Check for 2K block size"
ESDE  02D0  LDK  A+1024/128  " Otherwise 1K block size"
ESDF  233EF  WRIT1:  STD  A+JNACNT  "JNACNT = SEKTRK"
ESDE  2A13EF  LD  HL+SEKTRK  "SEKTRK"
ESDF  2233EF  STD  HL+JNACNT  "JNACNT = SEKTRK"
ESDF  3456EF  LD  A+LO5SEC  "SEKTRK"
ESDF  3C  INC  A
ESDF  1E61  JR  WRIT3
ESDF  2152EF  WRIT2:  LDK  HL+JNACNT  "If no unallocated records"
ESDF  7E  LD  A+[HI]
ESDF  37  JP  A+HI  "Load unallocated record count"
ESDF  2A37  JR  WRIT4  "If no unallocated records"
ESDF  35  DEC  [HI]  "Load unallocated record count"
ESDF  21FAEF  LDK  HL+D034EF-16+10  "Get number of sectors per track"
ESDF  1110D0  LDK  DE+16  "If not end of track"
ESDF  3A13EF  LD  A+SEKSK  "SEKSK"
ESDF  05  MOV  B+A  "SEKSK"
ESDF  04  INC  B
ESDC  19  WRIT25:  ADD  HL+DF  "HL+DF"
ESDE  10D0  JRNZ  WRIT25  "JNZ WRIT25"
ESDE  5E  LD  $+MLHI  "Number of sectors per track in A reg"
ESDE  23  INC  HL  "Number of sectors per track in A reg"
ESDF  56  LD  A+[DE]  "Increment logical sector"
ESDE  47  MOV  B+A  "Number of sectors per track in A reg"
ESDF  3A53EF  LD  A+UNACNT  "Set logical sector"
ESDF  3C  INC  A  "Set logical sector"
ESDF  9B  CMP  3  "If not end of track"
ESDF  20D8  JRNZ  WRIT3  "JNZ WRIT3"
ESDF  2433EF  LD  HL+JNACNT  "Set logical sector"
ESDF  23  INC  HL  "Set logical sector"
ESDF  2233EF  STD  HL+JNACNT
FILLS host buffer with appropriate 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 ix = host record address.
POP y iy = caller's buffer address.

```
E644  DPROC
E644  3221EF    STD  A, #DFLAG  ;Save read flag
E647  345CE9    LD    A, #SEKTyp
E64A  0F        RR  
E64B  0F        RR  
E64C  E03       ANI  D3
E64E  47        MOV  B,A
E64F  1180EA    LD    DE, #STBF  ;Initial offset
E652  3418EF    LD    A, #SEKSEC  ;Get logical sector
E655  7180DC    LD    HL, #128  ;128 byte records
E658  2308 *E655s JRZ  FILL33 ;Jump when sector size <> 128, no deblocking necessary
E65A  E0        FILL2: EX  DE,HL
E65B  0F        RR  
E65C  3001 *E65Fs JRNC FILL3 ;If low bit not set
E65E  19        ADD  HL,DE  ;Add bias to offset
E65F  E9        FILL3: EX  DE,HL
E660  29        ADD  HL,HL  
E661  E67F       ANI  #FFh  
E663  10F5 *E65As DNZ FILL2
E665  3218EF    FILL35: STD  A, #SEKSEC  ;SEKSEC = physical sector - 1
E668  2411EF    LD    HL, #160DD  ;Set return parameters
E66B  E3        XTHL  
E66C  05        PUSH  DE  
E66D  E5        PUSH  HL  ;Set return address
E66E  7150EF    LD    HL, #HSTACT  ;Host active flag
E671  7E        LD    A, (HL)
E672  3601      STO  (HL)  
E674  37        JR A  
E675  2810 *E674s JRZ  FILL6  ;If host buffer inactive
E677  2152EF    LD    HL, #HSTTY  
E67A  345CE9    LD    A, #SEKTyp
E67C  3604      CMP  (HL)  
E67E  2D11 *E671s JRNZ FILL5  ;If mismatch
E680  311CEF    LD    HL, #SEKSEC  
E683  1118EF    LD    DE, #SEKSEC
E686  0604      LD    B, #SEKSK-SEKSEC+1
E688  1A        FILL4: LD    A, #OF1
E689  3E        CMP  (HL)  
E69A  2005 *E691s JRNZ FILL5  ;If mismatch
E69C  23        INC  HL  
E69D  13        INC  DE  
E69E  10F8 *E683s DNZ FILL4  ;If all bytes not matched
```
E692 29

E691 C01E FILL5: CALL FLUSH ;Flush host buffer

E694 3A1BEF FILL5: LD A, SEKOK
E695 321FEF STD A, HSTOSK
E696 3217EF STD A, ACTOSK
E697 3219EF LD HL, SCCTRK
E698 221DEF STD HL, HSTTRK
E699 2215EF STD HL, ACTTRK
E6A0 3A18EF LD A, SEKSEC
E6A1 321CEF STD A, HSTSEC
E6A2 3214EF STD A, ACTSEC
E6A3 3A0CF9 LD A, SETYP
E6A4 320DE9 STD A, HSTYP
E6A5 3200EF STD A, ACTYP
E6A6 3A21EF LD A, ROFLAGS
E6A7 37 ORA A
E6A8 C0 NZ

E6A9 3E00 LDX A, O
E6AA 1822 Jr FINAL ;Read
E6C1
E6C1 2151EF PROC
E6C4 7E LDX MLHSTART
E6C5 97 LD A*HII
E6C6 CB JRA A
E6C7 360D STO A*HII  ; If host buffer already on disk
E6C9 3A1FEF LD A*HSTDISK
E6CC 3217EF STO A*ACTDISK
E6CF 2A10EF LD MLHSTTRK
E6D0 2215EF STO MLACTTRK  ; Move disk
E6D5 3A1CEF LD A*STSEC
E6D8 3214EF STC A*ACTSEC
E6D3 3A02E9 LD A*HSTTyp
E6DE 1202EF STC A*ACTTyp
E6E1 3E03 LDX A*E  ; Write file
E6E1 FINAL

**FLUSH:**

*Writes out active host buffer onto disk.*
E6E3 5F
E6E4 1600
E6E5 212701
E6E6 19
E6E7 220AE9
E6ED 2114EF
E6F0 34
E6F1 2A0AE9
E6F2 08
E6F3 0001
E6F4 CD19E3
E6F5 3222EF
E6F6 2B
E6F7 2122EF
E701 7E
E702 F001
E704 77
E705 C9

Copyright © 1982 Osborn Computer Corporation

SOACIM 808x Assembler ver 3.5E C/M/5/7 - 9193 Page 29
Osborn CP/M 2.2 C815S Rev 1.4
C102C31085.1SM
E706  MINFO: Move information necessary for transfer
E707  XRA  A
E70A  STA AF
E70D  STA SEC
E710  RET

; Clear error flag
E711 PRINT:
    : Print message string to CONSOLE device
E711 213BE7
    LDX HL*CONOUT
E714 1803 $E719$  
    JR STROUT

E716 PRINT:
    : Print message string to LIST device
E716 213BE7
    LDX HL*LIST
    JR STROUT

E719 STROUT:
    : Print message terminated by zero byte.
    ENTRY DE -> message buffer, first byte = length
    EXIT DE -> DE + length
    A = D0
    Z bit set.
    JSES A, BC, DE, HL

E719 PROC
E71A 1A
    LD A, [DE] ; Get a length of print string
E71A B7
    DRA A
    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 4A
    MOV E, 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 112BE7
    LDX DE*NEXTC
E726 D5
    PUSH DE ; Return address to stack
E727 E9
    JMP [HL] ; Output
E728 E1
    NEXTC: PDP HL
E729 C1
    PDP BC
E72A D1
    PDP DE
E728 10F3 $E719$
    DJNZ NEXTC ; Print next character if not done
E72D C9
    RET
The following routines will use the I03YTS to transfer
control to the appropriate device driver.

**E72E**

```
   DEF Const:
   I: Returns console status
```

```
E72E PROC
E72E 21F9E7  LDX  #L+PTR_CSTAT ;Status table
E731 1808 ^E733$ JR  GODISPCH ;Call appropriate rtn
```

**E733**

```
   PROCT: Reads input character from device
```

```
E733 PROC
E733 2131E7  LDX  #L+PTR_CINP ;Table of input rtt's
E736 1808 ^E738$ JR  GODISPCH
```

**E738**

```
   PROC: Puts output character to device
   c contains output character
```

```
E738 PROC
E738 2139E7  LDX  #L+PTR_COUT ;Table of output rtt's
```

**E738**

```
   GODISPCH:
E738 0601  LDX  8+1 ;Number of shifts required to align
E738 1829 ^E738$ JR  DISPC4
```

**DISPC4**
E73F

LIST:
    List device character output
    proc
E73F
    E73F  213AE1
    LDK  HL,PRINTFS
    ; Get printer initialization flag
E742  7E
    LD  A,(HL)
E743  G7
    JRA  A
E744  2203 H74ES
    JNZ  LIST1
    ; Printer previously initialized
E746  35
    DEC  (HL)
    ; Non-zero value to PRINTFS
E747  C5
    PUSH BC
    ; Save character
E748  23
    INC  HL
E749  23
    EX  HL,DE
    ; Get initialization string
E74A  CD16E7
    CALL PSTR
    ; Send it
E74D  C1
    POP  BC
    ; Restore character
E74E  0604
    LIST1:
    LDK  HL,PR_LIST
    ; Table of list routines
E750  2199E7
    LDK  HL,PR_PUNCH
    ; Punch routines
E753  1813 H753S
    JR  DISPCH

E755

PUNCH:
    Output to punch
    proc
E755
E755  0606
    LDK  HL,PUNCH
E757  2199E7
    LDK  HL,PR_PUNCH
    ; Punch routines
E75A  180C H75AS
    JR  DISPCH

E75C

READER:
    Reader input
    proc
E75C
E75C  0608
    LDK  HL,PR_READER
E75E  2181E7
    LDK  HL,PR_READER
    ; Reader routines
E761  1805 H761S
    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.
    ;
    proc
E763
E763  0604
    LDK  HL,PR_LIST
    ; List status routines
E765  2191E7
    ; Align LIST field of ID3YFE
    JR  DISPCH
E768  
DISPCH:
; on entry here reg B contains the left shift count
; required to align the byte field to bit 1 position.
; and reg HL contains address of select table
E768  proc
E768  3A03CO
E768  LD A,10BYTE
E768  DSPCH1:
E768  17  RAL
E768  10FD ^E763$ DJNZ DSPCH1
E768  E06 ANI 6 ; get select field#2
E770  9F  MDV E+A
E771  1630  LDR D+0 ; DE = ioByte field #?
E773  19  DAD DE
E774  5E  MDV E,[HL]
E775  23  INC HL
E776  56  MDV H,[HL] ; get the routine address
E777  68  MDV L,[HL] ; into hi and exchange with pc
E778  E9  JMP [HL]
Dispatch Table

E779  A1E7  DW  CUSTAT  ; keyboard status
E779  69E8  DW  SISTAT  ; serial port input status
E77D  CBE8  DW  PSSTAT  ; parallel input status
E77F  9BE8  DW  IEINSTAT  ; status of input device on IEEE port

E781  PTR_INP:
E781  AFE7  DW  KEYINP  ; get input from keyboard
E783  72E8  DW  SPINP  ; parallel input
E785  CFE8  DW  PARINP  ; parallel input
E787  9FE8  DW  IEINP  ; IEEE port input

E789  PTR_PCH:
E789  4CE8  DW  CROUT  ; output character to CRT
E78D  62E8  DW  SOUT  ; serial port output
E78D  D3E8  DW  PAROUT  ; parallel output
E78F  C3E8  DW  IEOUT  ; IEEE port output

E791  PTR_LST:
E791  49E8  DW  CRSTAT  ; serial port status
E793  5FE8  DW  SSSTAT  ; serial port status
E795  C7E8  DW  POSSTAT  ; parallel output status
E797  BOE8  DW  IEOSTAT  ; IEEE output status

E799  PTR_LST:
E799  4CE8  DW  CROUT
E799  7AE8  DW  PRTOUT
E79D  D3E8  DW  PAROUT
E79F  C3E8  DW  IEOUT
E7A1       CNST:
           CONSOL STATUS
           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 87    DPA       A
E7A5 2005   *E7AC    JRNZ      CNST:
            if no translated keys pending, check <yrit> flag
E7A7 3A5EEF  LD       A,KEY    ;Set Key hit flag
E7AA 97    DPA       A
E7AB C8    RZ       ;If data not available
E7AC       CNST:
E7AC F6FF   ORI       DFFh
E7AE C9    RET
ETAF

    KEYINP:
    1    Gets keystroke from ROM keyboard driver.
    1    Translates the codes 80h to 9fh as per table.
    1    EXIT  A = translated code in ASCII

    = 0980   BASVLD = 80h : lowest value of translatable keys

ETAF

    proc
    ETAF  DOES  $  PUSH IX  : Save user IX

ET81 DOF6E7    KINO:  CALL AHSLCL
ET84 211E7    LD  A 4L+COUNT
ET87 7E       LD  A 4L
ET88 B7       ORA A
ET89 2808  ^ETC6$  JNZ KIN03
ET8B 0D28E6E9 $  LD  IX+XLKEY
                   : get base address in IX reg
ET8F 0D7E00 $   LD  A 4L  : IX+COUNT
                   = ETC1 COUNT = = = 9-1
ETC2 34       INC (HL) : Increment count
ETC3 DOE1 $   RET; PDP IX  : Restore user IX
ETC5 C9       RET

ETC6

    KIN03:

ETC6 1E09    LD  $,09
ETC8 C017E3  CALL ROMC01

    When console has returned this code will check
    for function key and perform some translation

ETC8 FEB0    CPI 80h  : Function keys have value
ETC9 38F6  ^ETC3$  JRC KRET  : 80h-8Fh
ETCF FE8E    CPI 8Bh  : Do a shift to make pointer
ETD1 30F5  ^ETC3$  JRNC KRET  : Into table and return if not function key
ETD3 CB27 $   SLA A
ETD5 5A       MDV E,A
ETD8 1603 $   LD  0,3
ETD9 002158E1 $  LD  IX,XLTBL
ETD9 3019 $   ADD IX+01
ETD8 1D9000 $  LD  L,(IX+0)
ETD9 005601 $  LD  H,(IX+1)
ETD8 1D5602 $  LD  E,(IX+2)
ETD8 1D5603 $  LD  D,(IX+3)
ETD8 ED35E9 $  STO DE,XLTKEY
ETD8 ED52 $   SBC 4L,0E
ETF0 7D       MDV A=L
ETF1 32C1E7  STO A,COUNT
ETF4 1895  ^ETF11$  JR KINO
AHSCLL:  
: ahscrl - does auto horizontal scroll if required.

: proc
E7F6 3A6AE1  
E7F9 37  
E7FA 28  
E7FB 2A5AEF  
E7FE 29  
E7FF 3E64  
E801 8D  
E802 3A09 +E80D$  
E804 3A61EF  
E807 D6EA  
E809 C8  
E80A 4F  
E808 1818 ^E825$  
E80D 5C6A  
E810 76EA  
E812 C664  
E814 50  
E815 D222E9  
E818 D654  
E81A 30  
E81C 7D  
E81D 96D4  
E81F C8  
E820 1803 ^E825$  
E822 7D  
E823 0D64  
E825 1F  
E826 6024  
E828 2148E8  
E82B 77  
E830 3A52EF  
E832 E61F  
E833 25  
E834 77  
E835 28  
E836 29  
E837 0604  
E839 C5  
E83A E5  

: check for cursor in home window

: check right-hand margin

: check left hand margin

: return if cursor at column 2

: SCRL:  
RAR
ADD A,A  
LK HL,ESCSQ+3    
LD A,H[HL]  
AND 1Fh  
ADD A,A  
STD HL  
STD A[HL]  
ADD HL,0  
ADD HL,0  
LDA BC  
PUSH BC  
PUSH HL
E838 4E          LD C [HL]
E83C 0D60E8      CALL CRT10
E83F E1          POP HL
E840 C1          POP BC
E841 23          INC HL
E842 10F5 ^E8395 DJNZ 130
E844 C9          RET

E845 18          ESC52: DB ESC
E846 53          DB "*"
E847 DD          DB 0
E848 DD          DB 0

* y coord
* x coord

escape sequence
EB49  CRSTAT:
    \* Returns status of crt.
    \* crt is always ready

EB49  PROC
EB49  F6FF
EB48  C9
EB47  RET

EB4C  CRTOJT:
    ENTRY  C = output character
= 0008  EF_ESC:  = 8
= 0031  EF_GR:   = 1

EB4C  PROC
EB4C  3A60EF
EB4F  E609
EB51  2000  ^E850$  JNZ  CR10
EB53  79
EB54  FF14
EB5A  2008  ^E850$  JNZ  CR10
EB5B  3A68E1
EB5C  3268E1
EB5F  C9
EB60  RET

EB60  CRT10:
EB60  160C
EB62  0317E3  JMP  ROMCD1

5escape flag bit definitions
input to crt
E865
SOSTAT:

Sets status of output device attached to serial port

E865
E865 1E20
PROC
E865 1847 ^E830$  
LDK  E*20h
JR  JMPROM  /*Call rom driver*/

E869
SISTAT:

Sets status of input device attached to serial port

E869
E869 CD838
PROC
E869 5601
CALL  ACISTAT
E86E C8
ANI  SI+RDY
E86F F6FF
RZ
E871 59
ORI  TRUE
RET

E872
SPINP:

Inputs a character from the serial port

E872
PROC
E872 1E15
LDK  E*15h
E874 193A ^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
E87A       PRTGJT:
          routine does LIST output and printer protocols

= 0011     XDN: = 11h
= 0013     XOFF: = 13h
= 0003     ETX: = 3
= 0006     ACK: = 5

E87A       PROC
E87A       CALL SPDUT
E870       3467E1
E880       B7
E881       C8
E882       E62
E884       2013
E886       3E00
E888       39
E889       C0
E88A       DE03
E88C       CD76E8

E88F       CALL SPINP
E892       E67F
E894       FED6
E896       20FF
E898       C9

E899       CALL SISTAT
E89C       B7
E89D       C8
E89E       CD72E8
E8A1       E67F
E8A3       F61B
E8A5       C0

E8A6       CALL SPINP
E8A9       E67F
E8AB       F61B
E8AD       20FF
E8AF       C9

E88F       CALL SPDUT

E894       CALL SPDOUT

Imask out parity bit

Imask out parity bit

Imask out parity bit
E890  C317E3  JMPROM: JMP    ROMCD1
    = E13C  ACICTL: =  $9A0D
          : Outputs character in a to the ACIA CTL port.

E993  ACISTAT: =
      : Returns usart status in A

E883  PROC
E883  1E81  LDX  F81H
E885  18F9  $E830S
          JR    JMPROM
          : Low order byte of ACISTAT routine in ROM.
E887  IEINSTAT:
;  Returns status of IEEE input port
;  
;  EXIT  A = 0  character not available
;  A = FF4 character available
E887  proc
E887  IE87
E889  18F5 ^E8305
E890  JR  JMPROM

E883  IE2STAT:
;  Returns status of IEEE output port
;  
;  EXIT  A = 0  transmitter not ready
;  A = FF4 transmitter ready
E884  proc
E884  IE84
E886  18F1 ^E8925
E888  JR  JMPROM

E88F  IEIN:
;  Reads a character from IEEE port
E89F  IE9D
E8C1  18ED ^E8305
E8C3  JR  JMPROM

E8C3  IEDUT:
;  Outputs the character in reg C to IEEE port
E8C3  proc
E8C3  IE9D
E8C5  18E9 ^E8925
E8C7  JR  JMPROM
?POSTAT:
: Gets status of the parallel (centronics) printer
: attached to the ieee port

EBC7
EBC7 E996
EBC9 E8E5 E8501
      JR      JMPROM

EBCB
?ISTRAT:
: Gets status of the input device attached to the
: parallel port

EBCB
EBCB E993
EBCD E8E1 E8301
      JR      JMPROM

EBCF
?PARINP:
: Inputs a character from parallel port.

EBCF
EBCF E999
EBO1 E803 E8801
      JR      JMPROM

EBD3
?PAROUT:
: Outputs the character in c to the IEEE port treating the
: port as a parallel port.

EBD3
EBD3 E9C6
EBD5 E859 E8805
      JR      JMPROM
SWIRJM:
; Switches to RAM
; saves all registers

EB07  PROC
EB07  D1
EB08  PUSH AF
EB09  XRA A
EB0A  OUT D
EB0C  STO A,ROWRAM
EB0D  PDP AF
EB0E  EI
EB0F  RET

EB2E  SWIRAM:
; Switches to RAM
; preserves all registers

EB2E  PROC
EB2E  D1
EB2F  PUSH AF
EB30  LDX A,X
EB31  OUT 1
EB33  STO A,ROWRAM
EB34  PDP AF
EB35  EI
EB36  RET
= E8EE
    XXX:  = 
= 0000
    IF  XXX > CCP + 1600H
-    .9
    ERROR  SYSTEM SIZE TOO LARGE
    ENDF

; SYSTEM SPACE AVAILABLE = +1E00H - (XXX - CCP)
*SYSTEM SPACE AVAILABLE = 0012*

EBEE = E900
    ORG  CCP + 1600H

E900 = 0004
    LASTA: DS  4
E904 = 0002
    DPB1: DS  2
E906 = 0002
    SAVDPL: DS  2
E908 = 0001
    TEMOSK: DS  1
E909 = 0001
    TENTYP: DS  1
E90A = 0002
    SAVADR: DS  2
E90C = 0030
    ALV: DS  ALVSZ
E93C = 0020
    CSV: DS  CSVSZ
E95C = 0001
    SEXTYP: DS  1
E95D = 0001
    HSTYP: DS  1
E95E = 0002
    XLTKEY: DS  2

= E960
    YYY:  = 
= 0000
    IF  YYY > MRAM
-    .9
    ERROR  CODE HIT MRAM
    ENDF
10CCRAM2.ASM
E960 = E800
  ORG MRAM

; Used to assembly ROM resident and BIOS

E800 = 0480
  HSTBUF:
  DS 1024+128

; Directory Buffer
  = E880
  DEIRBUF:
  = HSTBUF+1024

EF00 = 0006
  TEM
  DS 6

= EF01
  RNDV = TEM+1
  random number seed

= EF02
  ERCNT = RNDV+1
  IDW ERCNT

= EF04
  TRC = ERCNT+2
  retry count

= EF05
  TRY = TRC+1
  ; prompt character

EF07 = 0001
  ECHOP
  DS 1
  =0, list echo off

EF08 = 0001
  ORMAM
  DS 1
  =0= RAM, 1= ROM

EF09 = 0006
  DSTS
  DS 6
  ; Disk status bytes

EF10 = 0002
  DMAD1
  DS 2
  ; Address for read/write Disk

EF11 = 0002
  DMADM
  DS 2
  ; E910s, users DMA

; Disk operation temp and control

EF13 = 0001
  SEKDEL:
  DS 1
  ; Set for seek/restore command in IAM
  ; 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)

; SYSTYP = DEFIND
; SELECTED TYPE (sector size)

= EF14
  ACTSEC = SAVSEC

= EF15
  ACTTRK = SAVTRK

= EF17
  ACTDISK = SDISK

EF18 = 0001
  SEKSEC
  DS 1

EF19 = 0002
  SEKTRK
  DS 2

EF1A = 0001
  SEKDISK
  DS 1

EF1C = 0001
  HTSEC
  DS 1

EF1D = 0002
  HTTRK
  DS 2

EF1F = 0001
  HTDISK
  DS 1

EF20 = 0001
  TEMSEC
  DS 1
  ; Used in bios only

EF21 = 0001
  RDFLAG
  DS 1
  ; Read flag

EF22 = 0001
  WRFLAG
  DS 1
  ; Error reporting

EF23 = 0001
  WRTYPE
  DS 1
  ; Write operation type

: ALVS
  DS ALVS

: CSVS
  DS CSVS

EF24 = 000C
  DS ALVS

EF30 = 0020
  DS CSVS

; BIOS blocking-deblocking flags
<table>
<thead>
<tr>
<th>Register</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF50</td>
<td>HSTAT: DS 1 Host active flag</td>
</tr>
<tr>
<td>EF51</td>
<td>HSTWR: DS 1 Host written flag</td>
</tr>
<tr>
<td>EF52</td>
<td>UNACG: DS 1 Unallocated rec count</td>
</tr>
<tr>
<td>EF53</td>
<td>JNTK: DS 2 JTrack</td>
</tr>
<tr>
<td>EF55</td>
<td>UNASEG: DS 1 ISector</td>
</tr>
<tr>
<td>EF56</td>
<td>LOSSEG: DS 1 ILosector</td>
</tr>
<tr>
<td>EF57</td>
<td>LDAD: DS 2 Zero if locked keyboard</td>
</tr>
<tr>
<td>EF59</td>
<td>KEYP C: DS 1 Current cursor position</td>
</tr>
<tr>
<td>EF60</td>
<td>TKEY: DS 1 Holding key</td>
</tr>
<tr>
<td>EF61</td>
<td>WKEY: DS 1 Debounce key</td>
</tr>
<tr>
<td>EF62</td>
<td>LKEY: DS 1 Last valid keystroke</td>
</tr>
<tr>
<td>EF63</td>
<td>CKEY: DS 1 Last control key</td>
</tr>
<tr>
<td>EF64</td>
<td>ESCH: DS 1 ESC holding flag</td>
</tr>
</tbody>
</table>

; SPIAAD and PIADD must be kept sequential. PIAAD first dependency in VC_HOME of BMKEY.asm

<table>
<thead>
<tr>
<th>Register</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF65</td>
<td>PIAAD: DS 1 Holds last PIA-A data</td>
</tr>
<tr>
<td>EF66</td>
<td>PIADD: DS 1 Holds last PIA-B data</td>
</tr>
<tr>
<td>EF67</td>
<td>IDAY: DS 3 Calendar month, day year</td>
</tr>
<tr>
<td>EF68</td>
<td>IMNTH = IDAY+1 Wall clock time cells and disk active</td>
</tr>
<tr>
<td>EF69</td>
<td>Y2 = IDAY+2 see UPTIM: in BMKEY.asm</td>
</tr>
<tr>
<td>EF70</td>
<td>HOURS: DS 8 Used to deselect drive when there is no activity</td>
</tr>
<tr>
<td>EF71</td>
<td>MINS: = HOURS+1 on drive for n seconds. See FOSK routine</td>
</tr>
<tr>
<td>EF72</td>
<td>SECS: = HOURS+2</td>
</tr>
<tr>
<td>EF73</td>
<td>SEC6: = HOURS+3</td>
</tr>
<tr>
<td>EF74</td>
<td>DACTV: = HOURS+4 =0 by FOSK, used by UPTIM</td>
</tr>
<tr>
<td>EF75</td>
<td>BELCT: = HOURS+5 =5 bell timer cell</td>
</tr>
<tr>
<td>EF76</td>
<td>LLIMIT: DS 1 IMAX #columns in a line</td>
</tr>
<tr>
<td>EF77</td>
<td>MSG = LLIMIT + MSG</td>
</tr>
<tr>
<td>EF78</td>
<td>LDSEL: DS 2 Disk drive current positions</td>
</tr>
<tr>
<td>EF79</td>
<td>LDTRK = LDSEL+1 ILast track used for non-selected drive</td>
</tr>
<tr>
<td>EF80</td>
<td>IESTK: DS 2 ISave current stk str</td>
</tr>
<tr>
<td>EF81</td>
<td>ISTK: DS 3 Interrupt stack</td>
</tr>
<tr>
<td>EF82</td>
<td>ISTK: DS 3 Stack entry</td>
</tr>
<tr>
<td>EF83</td>
<td>BIOSTK: DS 3 BIOS stack</td>
</tr>
</tbody>
</table>
Monitor RAM Storage.

SDR CIM 808x Assembler ver 305 </1987> Page 33

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFC1 = 0000</td>
<td>ROMSTK: DS 0</td>
</tr>
<tr>
<td>EFC1 = 0001</td>
<td>ACIA: DS 1 (Last command byte written to ACIA)</td>
</tr>
<tr>
<td>EFC2 = 0004</td>
<td>R179x: DS 4 (R179x register save area)</td>
</tr>
<tr>
<td>EFC6 = 0001</td>
<td>BDLY: DS 1 (Keyboard debounce delay cell)</td>
</tr>
</tbody>
</table>

; since CP/M CANNOT boot off 3", this cell is used
; to invert the names of the 2 drives:
; =0, all normal, A=B=0, B=B=
; =1, all inverted, A=B=0, B=A

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFC7 = 0001</td>
<td>OS&lt;5WP DS 1</td>
</tr>
<tr>
<td>EFC8 ++ 0008</td>
<td>ALIGN 10h</td>
</tr>
<tr>
<td>EFC8 = EFCC</td>
<td>DS (.asList()-1)/10h#10h-2</td>
</tr>
<tr>
<td>EFC0 = 0000</td>
<td>ACCTYP:</td>
</tr>
<tr>
<td>EFDD = 0001</td>
<td>SAVTY: DS 1</td>
</tr>
<tr>
<td>EFDD = 0001</td>
<td>RDT_WRTS: DS 1</td>
</tr>
<tr>
<td>EFDD = 0002</td>
<td>CCPADR: DS 2</td>
</tr>
<tr>
<td>EFDD = 0006</td>
<td>KEYLST: DS 6</td>
</tr>
<tr>
<td>EFDA = 0001</td>
<td>SERFLG: DS 1</td>
</tr>
<tr>
<td>EFDB = 0001</td>
<td>IE_ADDR: DS 1</td>
</tr>
<tr>
<td>EFDC = 0001</td>
<td>IE_CHARSET: DS 1</td>
</tr>
<tr>
<td>EFDD = 0001</td>
<td>PIACTL: DS 1</td>
</tr>
<tr>
<td>EFDE = 0001</td>
<td>PP_MODE: DS 1</td>
</tr>
</tbody>
</table>

; 8080 Register Save Area.

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFDF ++ 0001</td>
<td>ALIGN 10h</td>
</tr>
<tr>
<td>EFDD = 0001</td>
<td>REGS:</td>
</tr>
<tr>
<td>EFE1 = 0001</td>
<td>ESAVE: DS 1 (E Register save location)</td>
</tr>
<tr>
<td>EFE2 = 0001</td>
<td>DSAVE: DS 1 (D Register save location)</td>
</tr>
<tr>
<td>EFE3 = 0001</td>
<td>CSAVE: DS 1 (C Register save location)</td>
</tr>
<tr>
<td>EFE4 = 0001</td>
<td>BSAVE: DS 1 (B Register save location)</td>
</tr>
<tr>
<td>EFE5 = 0001</td>
<td>ASAVE: DS 1 (A Register save location)</td>
</tr>
<tr>
<td>EFE6 = 0001</td>
<td>LSAVE: DS 1 (L Register save location)</td>
</tr>
<tr>
<td>EFE7 = 0001</td>
<td>RSSAVE: DS 1 (R Register save location)</td>
</tr>
<tr>
<td>EFE8 = 0002</td>
<td>PSAVE: DS 2 (PSW Counter save location)</td>
</tr>
<tr>
<td>EFEA = 0002</td>
<td>SSAVE: DS 2 (USER STACK pointer save location)</td>
</tr>
<tr>
<td>EEEC = 0002</td>
<td>BKPA: DS 2 (Last breakpoint address)</td>
</tr>
<tr>
<td>EEEF = 0001</td>
<td>BKP: DS 1 (Contents of bkp)</td>
</tr>
<tr>
<td>EFFE = 0001</td>
<td>VATOFF: DS 1 (LAST VERTICAL OFFSET TAKEN FROM CNT)</td>
</tr>
</tbody>
</table>

Interrupt Jump Vector is between EFF9, EFFF.

RAM 4K

no ERRORS, 523 labels, 5723 bytes not used. Program LWA = EFFF.
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILL3 E65F</td>
<td>25/28</td>
<td>25/41</td>
</tr>
<tr>
<td>FILL35 E665</td>
<td>25/24</td>
<td>25/39</td>
</tr>
<tr>
<td>FILL4 E689</td>
<td>26/89</td>
<td>26/69</td>
</tr>
<tr>
<td>FILL5 E691</td>
<td>26/91</td>
<td>26/57 27# 3</td>
</tr>
<tr>
<td>FILL6 E694</td>
<td>25/46</td>
<td>27# 5</td>
</tr>
<tr>
<td>FILL7 E6E1</td>
<td>27/22</td>
<td>29# 2</td>
</tr>
<tr>
<td>FLUSH E6E2</td>
<td>12# 8</td>
<td>25/20 27# 3 23# 2</td>
</tr>
<tr>
<td>n FMTJ E139</td>
<td>4#32</td>
<td></td>
</tr>
<tr>
<td>n FRRER E328</td>
<td>14# 4</td>
<td>14#28 14#31</td>
</tr>
<tr>
<td>n FRRH E329</td>
<td>14# 5</td>
<td>14#31</td>
</tr>
<tr>
<td>n FPYSH D038</td>
<td>2#15</td>
<td>2#23 2#29 2#30 2#31</td>
</tr>
<tr>
<td>n FPYSH P039</td>
<td>2#14</td>
<td>2#27</td>
</tr>
<tr>
<td>n FSAVE EF34</td>
<td>50#41</td>
<td></td>
</tr>
<tr>
<td>s FSWAV E020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDEN E431</td>
<td>15#41</td>
<td>15#2</td>
</tr>
<tr>
<td>GETDEN E490</td>
<td>15#13</td>
<td>13#2</td>
</tr>
<tr>
<td>GDDS E733</td>
<td>12#11</td>
<td>12#21 32#35</td>
</tr>
<tr>
<td>GRROM E335</td>
<td>11#35</td>
<td>11#50</td>
</tr>
<tr>
<td>GRET E42C</td>
<td>15#34</td>
<td>15#39 15#44</td>
</tr>
<tr>
<td>s H.IEEE 2900</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s H.IOO 2A00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n HCNT EF5D</td>
<td>49#15</td>
<td></td>
</tr>
<tr>
<td>HOMIE E33F</td>
<td>4#19</td>
<td>12# 2 21/49</td>
</tr>
<tr>
<td>HOUAS EF56</td>
<td>48#32</td>
<td>49/33 49/34 49/35 49/39 49/41</td>
</tr>
<tr>
<td>n HSAVE EF77</td>
<td>50#44</td>
<td></td>
</tr>
<tr>
<td>n HSTAC EF50</td>
<td>12#10</td>
<td>23/23 26/42 49#61</td>
</tr>
<tr>
<td>n HSTBUF E450</td>
<td>22# 2</td>
<td>26/21 48# 7 49#11</td>
</tr>
<tr>
<td>n HSTOK EF11</td>
<td>27# 6</td>
<td>28/11 48#67</td>
</tr>
<tr>
<td>n HSTSS EF1C</td>
<td>25#11</td>
<td>28/12 28/15 48#45</td>
</tr>
<tr>
<td>n HSTSSL E430</td>
<td>2#13</td>
<td></td>
</tr>
<tr>
<td>n HSTRK EF1D</td>
<td>27# 9</td>
<td>28/13 48#46</td>
</tr>
<tr>
<td>n HSTRYP E950</td>
<td>28#47</td>
<td>27#15 28#17 47#27</td>
</tr>
<tr>
<td>n HSTRWT EF51</td>
<td>25#13</td>
<td>25/6 49# 2</td>
</tr>
<tr>
<td>n IBM E008</td>
<td>2#24</td>
<td>9/35</td>
</tr>
<tr>
<td>n IBMOS E09C</td>
<td>2#30</td>
<td>9/35</td>
</tr>
<tr>
<td>n IMXLT D000</td>
<td>8# 7</td>
<td>88#26</td>
</tr>
<tr>
<td>n IDAY EF53</td>
<td>49#26</td>
<td>49/27 49/28</td>
</tr>
<tr>
<td>n IEADR EF3B</td>
<td>21#39</td>
<td>50#25</td>
</tr>
<tr>
<td>n IEBIC E13F</td>
<td>4#35</td>
<td></td>
</tr>
<tr>
<td>n IEBIC E142</td>
<td>4#37</td>
<td></td>
</tr>
<tr>
<td>n IEBIC E145</td>
<td>4#38</td>
<td></td>
</tr>
<tr>
<td>n IEBIC E148</td>
<td>4#39</td>
<td></td>
</tr>
<tr>
<td>n IEBIC E149</td>
<td>4#40</td>
<td></td>
</tr>
<tr>
<td>n IEBIC E14E</td>
<td>4#41</td>
<td></td>
</tr>
<tr>
<td>n IEBIC E151</td>
<td>4#42</td>
<td></td>
</tr>
<tr>
<td>n IEBIC E154</td>
<td>4#43</td>
<td></td>
</tr>
<tr>
<td>n IEC ARR EF3C</td>
<td>50#26</td>
<td></td>
</tr>
<tr>
<td>n IEEAO E19</td>
<td>5#42</td>
<td>21/33</td>
</tr>
<tr>
<td>n IEINP E93F</td>
<td>35#15</td>
<td>44#23</td>
</tr>
<tr>
<td>n IEINST E857</td>
<td>35# 8</td>
<td>44# 2</td>
</tr>
<tr>
<td>n IESTA E986</td>
<td>35#28</td>
<td>44#15</td>
</tr>
<tr>
<td>n IESTA E989</td>
<td>35#22</td>
<td>35#34 44#37</td>
</tr>
<tr>
<td>n IESTK EF5F</td>
<td>10#14</td>
<td>48#51</td>
</tr>
<tr>
<td>n IMVTH EF64</td>
<td>49#27</td>
<td></td>
</tr>
<tr>
<td>n ICI-BTE E136</td>
<td>59#11</td>
<td>21/23</td>
</tr>
<tr>
<td>s ICYSTE D003</td>
<td>21#29</td>
<td>34# 8</td>
</tr>
<tr>
<td>n ISC EF99</td>
<td>49#55</td>
<td></td>
</tr>
<tr>
<td>n ILY E555</td>
<td>49#23</td>
<td></td>
</tr>
<tr>
<td>n JMPQW E930</td>
<td>41# 7</td>
<td>41/30 41/40 43# 2 43/18 44/10 44/23 44/32 44/42 45# 8 45/19 45/29 45/40</td>
</tr>
<tr>
<td>Symbol</td>
<td>Value</td>
<td>Offset</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>STRLOUT</td>
<td>E719</td>
<td>31/5</td>
</tr>
<tr>
<td>SVEX</td>
<td>00DE</td>
<td>11/37</td>
</tr>
<tr>
<td>SW2M</td>
<td>E8E2</td>
<td>11/35</td>
</tr>
<tr>
<td>SW2D</td>
<td>E8D7</td>
<td>11/35</td>
</tr>
<tr>
<td>SYS</td>
<td>0025</td>
<td>14/</td>
</tr>
<tr>
<td>SYSDAT</td>
<td>0010</td>
<td>14/</td>
</tr>
<tr>
<td>SYSL</td>
<td>0006</td>
<td>14/</td>
</tr>
<tr>
<td>TEM</td>
<td>EF30</td>
<td>18/4</td>
</tr>
<tr>
<td>TEMDCS</td>
<td>E908</td>
<td>18/4</td>
</tr>
<tr>
<td>TEMSEC</td>
<td>EF20</td>
<td>18/7</td>
</tr>
<tr>
<td>TEMTP</td>
<td>9099</td>
<td>18/40</td>
</tr>
<tr>
<td>NKLE</td>
<td>EF3C</td>
<td>19#14</td>
</tr>
<tr>
<td>NTRAN</td>
<td>E638</td>
<td>20#22</td>
</tr>
<tr>
<td>TRUE</td>
<td>FFFF</td>
<td>5/18</td>
</tr>
<tr>
<td>UNACNT</td>
<td>EFS2</td>
<td>12/11</td>
</tr>
<tr>
<td>UNASEC</td>
<td>EF55</td>
<td>20/10</td>
</tr>
<tr>
<td>UNATR</td>
<td>EFS3</td>
<td>19/9</td>
</tr>
<tr>
<td>UP</td>
<td>E1AF</td>
<td>5/36</td>
</tr>
<tr>
<td>VERS</td>
<td>0016</td>
<td>1#33</td>
</tr>
<tr>
<td>VFL3</td>
<td>FFF4</td>
<td>38/19</td>
</tr>
<tr>
<td>VLDL</td>
<td>0034</td>
<td>50#51</td>
</tr>
<tr>
<td>VRTPF</td>
<td>EEF</td>
<td>4/12</td>
</tr>
<tr>
<td>WBO</td>
<td>0054</td>
<td>2#35</td>
</tr>
<tr>
<td>WONGL</td>
<td>000D</td>
<td>2#36</td>
</tr>
<tr>
<td>WR11</td>
<td>E564</td>
<td>24/24</td>
</tr>
<tr>
<td>WR12</td>
<td>E53F</td>
<td>24/20</td>
</tr>
<tr>
<td>WRITZ</td>
<td>E5D7</td>
<td>24#6</td>
</tr>
<tr>
<td>WRIT3</td>
<td>E61E</td>
<td>24/32</td>
</tr>
<tr>
<td>WRIT4</td>
<td>E263</td>
<td>24/37</td>
</tr>
<tr>
<td>WRITE</td>
<td>E5CE</td>
<td>4/25</td>
</tr>
<tr>
<td>WWRTPE</td>
<td>EF31</td>
<td>24/18</td>
</tr>
<tr>
<td>WRAUTL</td>
<td>D0D2</td>
<td>1#37</td>
</tr>
<tr>
<td>XER3X</td>
<td>D001</td>
<td>2#13</td>
</tr>
<tr>
<td>XLTBA</td>
<td>E150</td>
<td>5#15</td>
</tr>
<tr>
<td>XLTBP</td>
<td>E158</td>
<td>5#25</td>
</tr>
<tr>
<td>XLTCE</td>
<td>E95E</td>
<td>37/7</td>
</tr>
<tr>
<td>XLTS</td>
<td>E22C</td>
<td>8/5</td>
</tr>
<tr>
<td>XOFF</td>
<td>0013</td>
<td>42#6</td>
</tr>
<tr>
<td>XON</td>
<td>0011</td>
<td>42#5</td>
</tr>
<tr>
<td>XONKO</td>
<td>E899</td>
<td>42/#6</td>
</tr>
<tr>
<td>XTA3</td>
<td>E220</td>
<td>16/29</td>
</tr>
<tr>
<td>XTRXLT</td>
<td>EE76</td>
<td>8#9</td>
</tr>
<tr>
<td>XXDSM</td>
<td>0053</td>
<td>2#29</td>
</tr>
<tr>
<td>XXX</td>
<td>E8EE</td>
<td>47#2</td>
</tr>
<tr>
<td>XXXLTS</td>
<td>E240</td>
<td>8/#6</td>
</tr>
<tr>
<td>YYY</td>
<td>E950</td>
<td>47#36</td>
</tr>
</tbody>
</table>