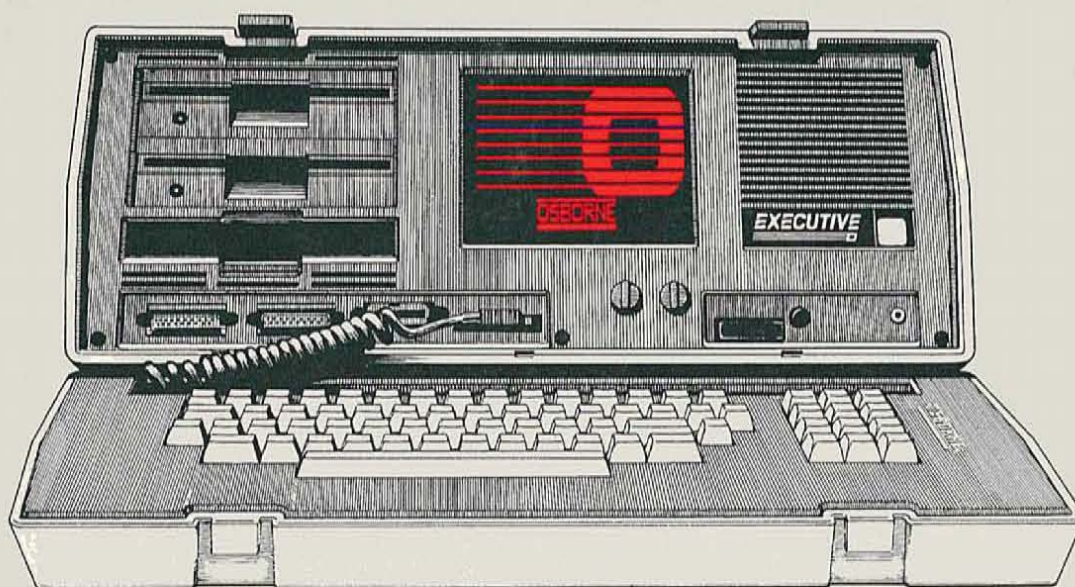


EXECUTIVE™

OSBORNE

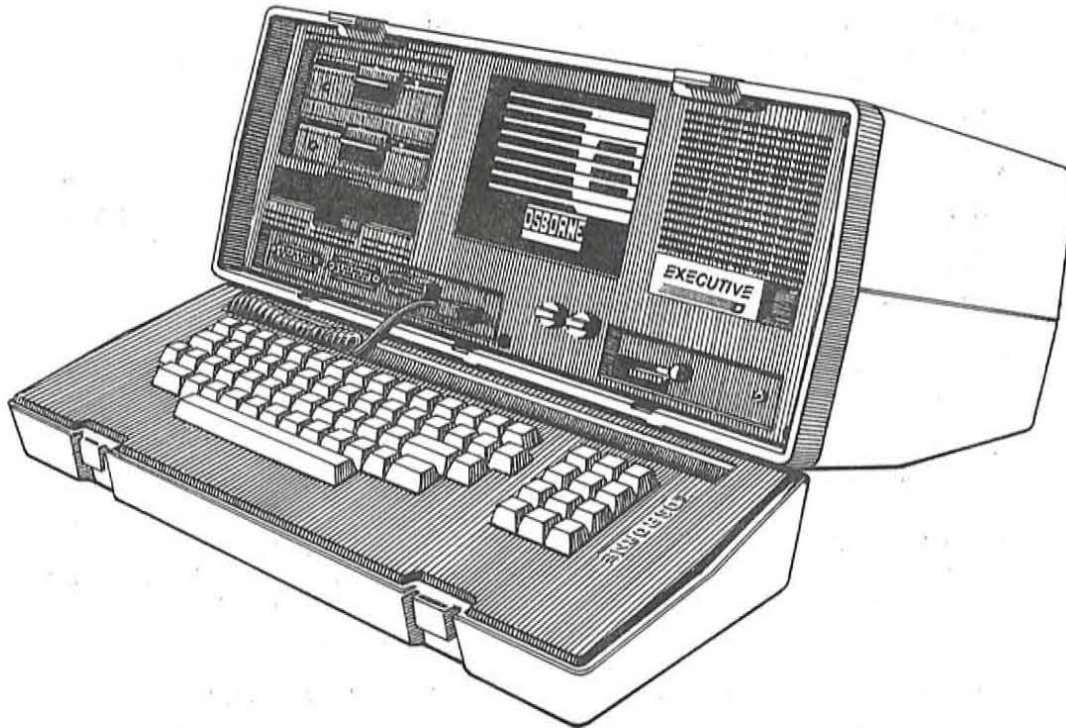
Technical Manual



Volume E: ROM 1.21/BIOS 1.1
Source Code Listing



**OSBORNE EXECUTIVE TECHNICAL MANUAL
VOLUME E: ROM 1.21/BIOS 1.1 SOURCE CODE LISTING**



ABSTRACT

This manual contains a listing of the source code for the Osborne Executive BIOS and ROM. Modules within the code are organized for easy reference. The code is accompanied with comments describing the program operations being performed.

COPYRIGHT © 1984 OSBORNE COMPUTER CORPORATION
26538 Danti Court, Hayward, CA 94545
(415) 887-8080

OSBORNE EXECUTIVE TECHNICAL MANUAL
Volume E: ROM 1.21/BIOS 1.1 Source Code Listing
Issue date: May 1984
Part Number: 2F00213-01

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopy, recording, or otherwise, without the prior written permission of OSBORNE COMPUTER CORPORATION.

The information in this document is subject to change without notice.

Neither OSBORNE COMPUTER CORPORATION nor this document makes any expressed or implied warranty, including, but not limited to the implied warranties of merchantability, quality, or fitness for a particular purpose. OSBORNE COMPUTER CORPORATION has no obligation to update or keep current the information contained in this document.

Under no circumstances will OSBORNE COMPUTER CORPORATION be liable for any loss or other damages arising out of the use of this manual

The following are trademarks of OSBORNE COMPUTER CORPORATION: OSBORNE, OSBORNE Executive.

Volume E was written by Roger Chapman, Wendy James, David Berg, and Chris Mayer. Technical consultation and production assistance was provided by Roy Robinson.



Production and design assistance provided by
Ulv och Lamm Media.

Volume E: ROM 1.21/BIOS 1.1 SOURCE CODE LISTING

EXECUTIVE ROM 1.21 SOURCE CODE LISTING

SYSS1.MAC.....	1-1
Executive System:	
Data definitions	
Macro definitions	
RAMS10.MAC.....	1-27
Common RAM storage declarations	
and allocations:	
RAMSMON.ASM.....	1-30
Monitor RAM storage definitions	
and allocations:	
ROM0.MAC.....	1-38
Entry Points to ROM operation:	
Start of ROM	
ROM jump table	
Boot routines	
Initialization routines	
Miscellaneous routines	
System PIA initialization equates	
Boot messages	
Log-on message	
Keyboard initialization routines	
Character font initialization routines	
ROM1.MAC.....	1-80
Power-On Diagnostics:	
Power on test	
Diagnostic initialization	
ROM2.MAC.....	1-91
Console Routines:	
Table handling routines	
Control character code processing	
-- cursor movements	
-- editing functions	
-- scrolling	
-- miscellaneous	
Set / Clear video display attributes	
Cursor coordinate processing	
Window processing	
Miscellaneous console routines	
ROM3.MAC.....	1-181
Keyboard Routines:	

TABLE OF CONTENTS

ROM4.MAC.....	1-201
I/O Drivers:	
BIOS calls - IEEE	
Parallel port drivers - IEEE	
Parallel port drivers - Centronics	
Initialize clock-timer chip	
Serial port drivers	
ROM5.MAC.....	1-247
Disk Drivers:	
ROM6.MAC.....	1-285
Interrupt Handling Routines:	
ROM7.MAC.....	1-320
Buffer Handling Routines:	
Character I/O Buffering routines	
ROM8.MAC.....	1-348
High Ram Routines:	
Interrupt handling routines	
Disk drive controller routines	
Bank switching routines	
MACRO AND SYMBOL TABLE.....	S

TABLE OF CONTENTS

EXECUTIVE BIOS 1.1 SOURCE CODE LISTING

BIOS0.MAC

Jump table and unchanging returns

BIOS1.MAC

Disk drive tables
DPH module
DPB module
SECTRAN module

BIOS2.MAC

Serial I/O tables
I/O redirection
XON/XOFF protocol routines
ETX/ACK protocol routines
IEEE routines

BIOS3.MAC

Miscellaneous routines
XMOVE routine
MOVE routine
SELMEM routine
TOROM access routine
TIME access routine

BIOS4.MAC

Disk I/O routines

BIOS5.MAC

Cold and Warm boot routines

BIOS6.MAC

Minimal BIOS to be used with CPMLDR

BOOT.MAC

Boot program found on track 1 for loading CCPLDR

WTSYS.MAC

A special program used for operating system development

```

      .280
      TITLE      'EXECUTIVE ROM MONITOR'

      SUBTTL     'DATA AND MACRO DEFINITIONS'
      INCLUDE    SYSS1.MAC      ;SYSTEST FILE

C
C
C      ;
C      ;
C      ;
C      ;
C      ;
C      ;
C      ; I/O PORTS
C
0000 C      SYS_DTA EQU    00H      ;SYSTEM PIA DATA A
0001 C      SYS_CNA EQU    01H      ;SYSTEM PIA CONTROL A
0002 C      SYS_DTB EQU    02H      ;SYSTEM PIA DATA B
0003 C      SYS_CNB EQU    03H      ;SYSTEM PIA CONTROL B
C
0004 C      TIM_CT0 EQU    04H      ;8253 TIMER COUNTER 1
0005 C      TIM_CT1 EQU    05H      ;8253 TIMER COUNTER 2
0006 C      TIM_CT2 EQU    06H      ;8253 TIMER COUNTER 3
0007 C      TIM_CNL EQU    07H      ;8253 TIMER CONTROL REGISTER
C
0008 C      DSK_CMD EQU    08H      ;FLOPPY DISK DISK COMMAND REG (WRITE)
0008 C      DSK_STS EQU    DSK_CMD  ;STATUS REG (READ)
0009 C      DSK_TRK EQU    DSK_CMD+1 ;TRACK REG
000A C      DSK_SEC EQU    DSK_CMD+2 ;SECTOR REG
000B C      DSK_DAT EQU    DSK_CMD+3 ;DATA REG (R/W)
C
000C C      SIO_DTA EQU    0CH      ;SIO DATA REG A
000D C      SIO_CNA EQU    0DH      ;SIO CONTROL REG A
000E C      SIO_DTB EQU    0EH      ;SIO DATA REG B
000F C      SIO_CNB EQU    0FH      ;SIO CONTROL REG B
C
C
0010 C      PAR_DTA EQU    10H      ;PARALLEL PORT PIA DATA A
0011 C      PAR_CNA EQU    11H      ;CONTROL REGISTER A
0012 C      PAR_DTB EQU    12H      ;PARALLEL PORT PIA DATA B
0013 C      PAR_CNB EQU    13H      ;CONTROL REGISTER B
C
0014 C      KEYBD EQU    14H      ;KEYBOARD
C
C      PAGE

```



```

C
C
C      ;SYSTEM PIA COMMANDS
C
C      ;DATA A:  +-----+
C      ;          1 7 1 6 1 5 1 4 1 3 1 2 1 1 0 1
C      ;          +-----+
C      ;          | | | | | | | |
C      ; ROM ENABLE (-+ | | | | | | |
C      ; VIDEO ENABLE (-+ | | | | | |
C      ; BANK 6 ENABLE (-+ | | | | |
C      ; BANK 5 ENABLE (-+ | | | | |
C      ; BANK 4 ENABLE (-+ | | | | |
C      ; BANK 3 ENABLE (-+ | | | | |
C      ; BANK 2 ENABLE (-+ | | | | |
C      ; BANK 1 ENABLE (-+ | | | | |
C
C
C      ;DATA B:  +-----+
C      ;          1 7 1 6 1 5 1 4 1 3 1 2 1 1 0 1
C      ;          +-----+
C      ;          | | | | | | | |
C      ; RI (ACT LOW) (+ | | | | | | |
C      ; DSR (ACT LOW) (-+ | | | | | |
C      ; TX CLOCK SELECT (-+ | | | | |
C      ; RX CLOCK SELECT (-+ | | | | |
C      ; SPEAKER (-+ | | | | |
C      ; DRIVE B (ACT LOW) (-+ | | | | |
C      ; DRIVE A (ACT LOW) (-+ | | | | |
C      ; DOUBLE DENSITY (ACT LOW) (-+ | | | | |
C
C
C      ;DATA
C
0080      DATA_INIT      EQU      10000000B      ;ENABLE ROM
0007      DATB_INIT      EQU      00000111B      ;DESELECT DRIVES A & B, SINGLE DENSITY
C      ;SPKR OFF
C
C      ;DIRECTION
C
00FF      DIRA_INIT      EQU      11111111B      ;SELECT ALL LINES AS OUTPUTS
003F      DIRB_INIT      EQU      00111111B      ;SELECT RI & DSR AS INPUTS, REST AS OUTPUTS
C
C      ;MASKS
C
0080      ROM_BANK      EQU      10000000B      ;ROM BANK
0040      VID_BANK      EQU      01000000B      ;VIDEO BANK
0008      SPEAKER EQU      00001000B      ;SPEAKER BIT
C
C      PAGE

```

```

C
C      ;SYSTEM PIA CONTROL REGISTERS
C
C      ;
C      ;
C      ;
C      ;
C      ; INTERRUPT FLAGS:
C      ;   A: DMA INTERRUPT
C      ;   B: REAL TIME CLOCK INTERRUPT  (<---+
C      ;
C      ; NOT USED
C      ; (<-----+
C      ;
C      ; CONTROL 2:
C      ;   A: 110 = 64 COLUMNS, 111 = 80 COLUMNS
C      ;   B: 110 = 60 HZ,    111 = 50 HZ    (<--+--+--+
C      ;
C      ; DDRA:
C      ;   A & B: 1 = DATA REGISTER, 0 = DIRECTION REGISTER (<--+
C      ;
C      ; CONTROL 1:
C      ;   A: DMA INTERRUPT REQUEST (HW)
C      ;   B: REAL TIME CLOCK INTERRUPT
C      ;
C      ;
C      ;   1 = ACTIVE HIGH (<-----+
C      ;   1 = ENABLE     (<-----+
C
C      ;CONTROL
C
0034 C      CA_DAT EQU    00110100B          ;64 COLUMNS
C                                     ;ENABLE DATA REG
C                                     ;DISABLE INTERRUPTS
0030 C      CA_DIR EQU    00110000B          ;64 COLUMNS
C                                     ;ENABLE DIRECTION REG
C                                     ;DISABLE INTERRUPTS
0035 C      CA_64C EQU    00110101B          ;64 COLUMNS
C                                     ;ENABLE DATA REG
C                                     ;ENABLE INTERRUPTS
C
0036 C      CB_DAT EQU    00110110B          ;60HZ
C                                     ;ENABLE DATA REG
C                                     ;DISABLE INTERRUPTS
0032 C      CB_DIR EQU    00110010B          ;60HZ
C                                     ;ENABLE DIRECTION REG
C                                     ;DISABLE INTERRUPTS
0037 C      CB_60H EQU    00110111B          ;60HZ
C                                     ;ENABLE DATA REG
C                                     ;ENABLE INTERRUPTS
C
C      ;CONTROL MASKS
C
0008 C      CDL_MSK EQU    00001000B          ;64/80 COLUMN MASK
0001 C      INT_MSK EQU    00000001B          ;INTERRUPT ENABLE MASK

```

'EXECUTIVE ROM MONITOR'
'DATA AND MACRO DEFINITIONS'

MACRO-80 3.44 09-Dec-81

PAGE 1-3

C

PAGE

```

C
C
C      ;SERIAL PORT -- SIO COMMANDS
C
C      ;SIO WRITE REGISTERS
C
C      ;WRITE REG 0:
C      ;
C      ;
C      ;CRC RESET CODES:
C      ;   NULL
C      ;   RESET RX CRC CHECKER
C      ;   RESET TX CRC GENERATOR
C      ;   RESET TX UNDERRUN/EOM LATCH
C      ;COMMAND BITS:
C      ;   NULL
C      ;   SEND ABORT
C      ;   RESET EXT/STATUS INTERRUPT
C      ;   CHANNEL RESET
C      ;   ENABLE INT ON NEXT RX CHARACTER
C      ;   RESET TX INT PENDING
C      ;   ERROR RESET
C      ;   RETURN FROM INT (CHAN A ONLY)
C      ;POINTER BITS:
C      ;   REGISTER 0
C      ;   REGISTER 1
C      ;   REGISTER 2
C      ;   REGISTER 3
C      ;   REGISTER 4
C      ;   REGISTER 5
C      ;   REGISTER 6
C      ;   REGISTER 7
C
C
C
C
C
C
C
C
C
C
C      ;WRITE REG 1:
C      ;
C      ;
C      ;
C      ;WAIT/READY:
C      ;   WAIT/READY ENABLE
C      ;   ~WAIT/READY FUNCTION
C      ;   WAIT/READY ON RX/TX
C      ;RX INTERRUPT MODE:
C      ;   RX INTERRUPTS DISABLED
C      ;   RX INT ON FIRST CHAR
C      ;   INT ON ALL RX CHARS
C      ;   INT ON ALL RX CHARS
C      ;STATUS AFFECTS VECTOR (CHAN B ONLY)
C      ;TX INT ENABLE
C      ;EXT INT ENABLE

```

	7	6	5	4	3	2	1	0
REGISTER 0	1	1	1	1	1	1	1	1
REGISTER 1	0	0	1	1	1	1	1	1
REGISTER 2	0	1	1	1	1	1	1	1
REGISTER 3	1	0	1	1	1	1	1	1
REGISTER 4	1	1	1	1	1	1	1	1
REGISTER 5	0	0	0	1	1	1	1	1
REGISTER 6	0	1	0	1	1	1	1	1
REGISTER 7	0	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
WAIT/READY ENABLE	1	1	1	1	1	1	1	1
~WAIT/READY FUNCTION	1	1	1	1	1	1	1	1
WAIT/READY ON RX/TX	1	1	1	1	1	1	1	1
RX INTERRUPTS DISABLED	0	0	1	1	1	1	1	1
RX INT ON FIRST CHAR	0	1	1	1	1	1	1	1
INT ON ALL RX CHARS	1	0	1	1	1	1	1	1
INT ON ALL RX CHARS	1	1	1	1	1	1	1	1
STATUS AFFECTS VECTOR (CHAN B ONLY)	1	1	1	1	1	1	1	1
TX INT ENABLE	1	1	1	1	1	1	1	1
EXT INT ENABLE	1	1	1	1	1	1	1	1


```

C
C ;WRITE REG 2:
C ;
C ;
C ;
C ;
C ;
C ;
C INTERRUPT VECTOR (<-----+-----+-----+-----+-----+-----+
C
C
C
C ;WRITE REG 3:
C ;
C ;
C ;
C ;RECEIVER BITS/CHARACTER:
C ;      5 BITS      0 0 | | | | | |
C ;      6 BITS      0 1 | | | | | |
C ;      7 BITS      1 0 | | | | | |
C ;      8 BITS      1 1 | | | | | |
C ,AUTO ENABLES (<-----+-----+-----+-----+-----+-----+
C ;ENTER HUNT PHASE (NOT USED) (<-----+-----+-----+-----+-----+
C ;RX CRC ENABLE (<-----+-----+-----+-----+-----+-----+
C ;ADDRESS SEARCH MODE (NOT USED) (<-----+-----+-----+-----+-----+
C ;SYNC CHAR LOAD INHIBIT (NOT USED) (<-----+-----+-----+-----+-----+
C ;RX ENABLE (<-----+-----+-----+-----+-----+-----+
C
C
C
C ;WRITE REG 4:
C ;
C ;
C ;
C ;CLOCK MODE:
C ; X1 CLOCK MODE    0 0 | | | | | |
C ; X16 CLOCK MODE   0 1 | | | | | |
C ; X32 CLOCK MODE   1 0 | | | | | |
C ; X64 CLOCK MODE   1 1 | | | | | |
C ;SYNC MODE: (NOT USED)
C ; 8 BIT SYNC CHAR     0 0 | | | | | |
C ; 16 BIT SYNC CHAR    0 1 | | | | | |
C ; SDLC MODE           1 0 | | | | | |
C ; EXT SYNC MODE       1 1 | | | | | |
C ;STOP BITS:
C ; SYNC MODE ENABLE    0 0 | | | | | |
C ; 1 STOP BIT/CHAR     0 1 | | | | | |
C ; 1.5 STOP BITS/CHAR  1 0 | | | | | |
C ; 2 STOP BITS/CHAR    1 1 | | | | | |
C ;PARITY:
C ; PARITY EVEN/ODD (<-----+-----+-----+-----+-----+
C ; PARITY ENABLE (<-----+-----+-----+-----+-----+
C
C
C

```

PAGE

```

C
C
C      ;SID READ REGISTERS
C
C      ;READ REG 0:
C      ;
C      ;
C      ;
C      ;BREAK/ABORT (-----+ | | | | | | |
C      ;TX UNDERRUN/EOM (-----+ | | | | | |
C      ;CTS (-----+ | | | | |
C      ;SYNC/HUNT (-----+ | | | |
C      ;DCD (-----+ | |
C      ;TX BUFFER EMPTY (-----+ |
C      ;INT PENDING (CHAN A ONLY) (-----+
C      ;RX CHARACTER AVAILABLE (-----+
C
C
C
C
C      ;READ REG 1:
C      ;
C      ;
C      ;
C      ;END OF FRAME (-----+ | | | | | |
C      ;CRC/FRAMING ERROR (-----+ | | | | |
C      ;RX OVERRUN ERROR (-----+ | | | |
C      ;PARITY ERROR (-----+ | | | |
C      ;RESIDUE CODES (NOT USED) (-----+
C      ;ALL SENT (-----+
C
C
C
C
C
C
C
C
C
C
C      ;READ REG 2:
C      ;
C      ;
C      ;
C      ;INTERRUPT VECTOR (-----+
C
C
C      PAGE

```

```

C
C
C      ;SIO CONTROL CONSTANTS
C
0000  C      REG0   EQU    00000000B      ;SELECT REGISTER 0
0001  C      REG1   EQU    00000001B      ;SELECT REGISTER 1
0002  C      REG2   EQU    00000010B      ;SELECT REGISTER 2
0003  C      REG3   EQU    00000011B      ;SELECT REGISTER 3
0004  C      REG4   EQU    00000100B      ;SELECT REGISTER 4
0005  C      REG5   EQU    00000101B      ;SELECT REGISTER 5
0006  C      REG6   EQU    00000110B      ;SELECT REGISTER 6
0007  C      REG7   EQU    00000111B      ;SELECT REGISTER 7
C
0000  C      INT_CNL EQU    00000000B      ;INTERRUPT CONTROL      (REG 1)
C      ;      NO INTERRUPTS
00E1  C      RCV_CNL EQU    11100001B      ;LOGIC CONTROL          (REG 3)
C      ;      8 BITS/CHAR, AUTO ENABLES ON, RX ENABLE,
C      ;      SYNC CHAR HUNT INHIBIT DISABLED
0044  C      BAS_CNL EQU    01000100B      ;BASIC CONTROL VALUES  (REG 4)
C      ;      X16 CLOCK, 1 STOP BIT, NO PARITY
00E8  C      TX_CNL_A  EQU    11101000B      ;TRANSMIT CONTROL FOR   (REG 5)
C      ;      CHANNEL A: DTR ON, 8 BITS/CHAR, RTS OFF
00EA  C      TX_CNL_B  EQU    11101010B      ;TRANSMIT CONTROL FOR   (REG 5)
C      ;      CHANNEL B: DTR ON, 8 BITS/CHAR, RTS ON
0017  C      SIO_I_IN  EQU    00010111B      ;COMMAND TO WRITE TO SIO WR1 TO SET:
C      ;      INTERRUPT ON ALL RX CHARS (PARITY
C      ;      AFFECTS VECTOR)
C      ;      STATUS AFFECTS VECTOR
C      ;      TX INT. ENABLE
C      ;      EXTERNAL INT. ENABLE
C
C      ;MASKS
C
0001  C      SI.RRDY EQU    00000001B      ;RECEIVE CHAR READY     (READ REG 0)
0004  C      SI.TRDY EQU    00000100B      ;TRANSMIT BUFFER EMPTY  (READ REG 0)
C
C
C      PAGE

```

```

C
C
C      ;8253 TIMER
C
C      ;CONTROL REG:
C      ;
C      ;
C      ;
C      ;SELECT COUNTER (-----+-----+
C      ;READ/LOAD:
C      ;   COUNTER LATCHING
C      ;   R/L MOST SIGNIFICANT BYTE ONLY
C      ;   R/L LEAST SIGNIFICANT BYTE ONLY
C      ;   R/L BOTH BYTES
C      ;MODE (-----+-----+
C      ;BCD (-----+-----+
C
C
C      ;CONTROL
C
0036  C      COUNTA EQU    00110110B      ;SELECT COUNTER 0, 2 BYTES, MODE 3
0076  C      COUNTB EQU    01110110B      ;SELECT COUNTER 1, 2 BYTES, MODE 3
C
C      ;SETBAUD CODES
C      ;   (CODE IS A TABLE OFFSET.
C      ;   TABLE ENTRIES ARE 8253 COUNT VALUES NECESSARY TO PRODUCE THE DESIRED BAUD RATE)
C
0008  C      BD1200 EQU      8              ;CODE FOR 1200 BAUD
C
C      PAGE

```



```

C
C
C      ;PARALLEL PORT CONSTANTS
C
C      ;PORT CTL REGISTER CONSTANTS.
C
002A   C      PA.CDR EQU      00101010B      ;TO ADDRESS PORT A DIRECTION
002E   C      PA.CDT EQU      00101110B      ;TO ADDRESS PORT A DATA AND SET
C      ;PORT A IN INPUT PROGRAM HANDSHAKE MODE.
0000   C      PB.CDR EQU      00000000B      ;TO ADDRESS PORT B DIRECTION
0004   C      PB.CDT EQU      00000100B      ;TO ADDRESS PORT B DATA
C
C      ;DIRECTION REGISTER CONSTANTS
C
00FF   C      PA.DRD EQU      0FFH           ;PORT A OUTPUT MODE
0000   C      PA.DRI EQU      00H           ;PORT A INPUT MODE
00BF   C      PB.DR  EQU      0BFH           ;PORT B DIRECTION
0002   C      PB.DTD EQU      00000010B      ;PORT B DATA FOR OUTPUT
0005   C      PB.DTI EQU      00001011B      ;PORT B DATA FOR INPUT
C
0040   C      PP.ORDY EQU      01000000B      ;OUTPUT RDY BIT IN PIB
0080   C      PP.IRDY EQU      10000000B      ;INPUT RDY BIT IN PIA CTL REG
0022   C      STRB  EQU      00100000B      ;STROBE BIT IN PORT B
C
C      ;PORT MODES
C
0001   C      PP.OUT EQU      1
0002   C      PP.IN  EQU      2
C
C      ;IEEE CONTROL CODES
C
0040   C      IE_TALK EQU      40H           ;MAKE TALKER
005F   C      IE_UTLK EQU      5FH           ;MAKE UNTALK
0020   C      IE_LSTN EQU      20H           ;MAKE LISTENER
003F   C      IE_ULST EQU      3FH           ;MAKE UNLISTEN
C
C
C
C      PAGE

```

```

C
C
C      ;VIDEO CONSTANTS
C
0000  C      FWAVM  EQU    0C000H
C      ;LVMEM      =      128*23 + 80      ;LENGTH OF VIDEO MEMORY
C      ;VIDEND      =      FWAVM+LVMEM-1    ;END OF VIDEO MEMORY
0018  C      VMROWS EQU    24                ;24 ROWS
0050  C      VMCOLS EQU    80                ;80 COLUMNS
C
0000  C      RESERVED EQU    0                ;WINDOW 0 IS RESERVED (ALWAYS SET TO FULL SCREEN)
C      ;NVDL        =      24
C      ;LVDIS        =      128*NVDL        ;LENGTH OF DISPLAY MEMORY
C      ;VFLO         =      -22             ;FIRST LINE VIDEO OFFSET
0080  C      VLL      EQU    128             ;LENGTH OF ONE VIDEO LINE
C      ;VLDL         =      64             ;LENGTH OF DISPLAYED CHAR/LINE
C      ;FWASH        =      FWAVM+VLDL     ;STATUS INFO
C      ;NLINE        =      24             ;NUMBER OF LINES
C
00F0  C      MODEMSK EQU    0F0H             ;MASK VIDEO MODE BITS
1000  C      MODEOFF EQU    1000H           ;OFFSET FROM VIDEO CHARACTER ADDRESS TO MODE ADDRESS
C
C      ;MASKS
C
0080  C      REVMODE EQU    10000000B       ;REVERSE BIT = HIGH BIT OF CHARACTER
0080  C      HAFMODE EQU    10000000B       ;HALF INTENSITY BIT = HIGH BIT OF MODE ADDRESS
0040  C      UNDMODE EQU    01000000B       ;UNDERLINE BIT
0020  C      BLMODE  EQU    00100000B       ;BLINK BIT
0010  C      ALTMODE EQU    00010000B       ;ALTERNATE CHARACTER SET BIT
C
0080  C      REV_CUR EQU    10000000B       ;REVERSE VIDEO (BLOCK) CURSOR
0040  C      UND_CUR EQU    01000000B       ;UNDERLINE CURSOR
0000  C      INV_CUR EQU    00000000B       ;INVISIBLE CURSOR
0020  C      BLK_CUR EQU    00100000B       ;BLINKING CURSOR
C
C      PAGE

```

```
C
C
0018 C      RTC_OFL EQU    18H          ;RTC OVERFLOW COUNTER PORT
C
003C C      SIXTY_HZ      EQU    60          ;DECIMAL 60 HZ (TICKS/SEC)
0032 C      FIFTY_HZ       EQU    50          ;50 HZ (TICKS/SEC)
C
0060 C      SEC_MIN EQU    60H          ;BCD SECONDS/MINUTE
0060 C      MIN_HR  EQU    60H          ;MINUTES/HR
0024 C      HR_DAY  EQU    24H          ;HOURS/DAY
C
C      PAGE
```

```

C
C
C      ;KEYBOARD CONSTANTS
C
0003   C      KL_LEN EQU      3           ;KEY LIST LENTH
0002   C      KLE_LEN EQU     2           ;KEY LIST ENTRY LENTH
C
0007   C      KL_USED EQU     7           ;KEYLIST ENTRY USED
0006   C      KY_SRVD EQU     6           ;KEY SERVICED ONCE
0038   C      KROW_M EQU     38H         ;ROW NUMBER MASK
0007   C      KCOL_M EQU     7H         ;COL NUMBER MASK
C
0001   C      RPTCTR EQU      1           ;REPEAT COUNT
0081   C      ROW0_M EQU     81H         ;MASK FOR ADDRESSING ROW 0
C
0001   C      DB_CT EQU       1           ;DEBOUNCE COUNT
0018   C      IRPTCT EQU     24           ;INITIAL REPEAT COUNT (400MS)
0003   C      SRPTCT EQU      3           ;SECOND REPEAT COUNT (100MS)   DAB9/17
C      ;REDUCED TO HALF ORIGINAL VALUE TO
C      ;SPEED REPEAT.
C
C
0008   C      TOT_ROW EQU     8           ;TOTAL ROWS
0008   C      TOT_COL EQU     8
C
0002   C      CTL_KY EQU      2           ;COLUMN NUMBER OF CTL, ALPHA AND SHIFT KEYS
0003   C      ALPH_KY EQU     3
0004   C      SHFT_KY EQU     4
C
C      PAGE

```

```
C
C
C      ;BUFFER HANDLING EQUATES
C
0010  C      MX_CH_BF      EQU      10H      ;MAX NUMBER OF CHARS IN A BUFFER
0004  C      BF_OV_HD      EQU      4        ;NO. BYTES OVERHEAD IN BUFFER
0006  C      D_RT_LEN      EQU      6        ;NO OF BYTES IN DEVICE LIST ROOT DATA STRUCTURE
0005  C      NO_DEVS EQU      5              ;NUMBER OF DEVICES
0028  C      NO_BUFS EQU      40            ;NUMBER OF BUFFERS
0004  C      THR_HOLD      EQU      4        ;NUMBER OF BUFFERS TO RESERVE FOR HIGH PRIORITY
C
C      ;DEVICES
C
0080  C      HI_PRIOR      EQU      80H      ;HIGH PRIORITY BIT TO INDICATE
C                                          ;HIGH PRIORITY BUFFERS CAN BE USED
C
C      PAGE
```

```

C
C
C      ;CONTROL CHARACTERS
C
000D  C      CR      EQU      0DH      ;^M, CR = CARRIAGE RETURN
000A  C      LF      EQU      0AH      ;^J, LF = LINE FEED
001F  C      CRLF    EQU      '^-40H  ;^_, CR AND LF
0008  C      BKS     EQU      08H      ;^H, BACKSPACE
0007  C      CBELL   EQU      'G'-40H  ;RING THE BELL
000B  C      MCUP    EQU      'K'-40H  ;MOVE CURSOR UP
000C  C      MCRIGH  EQU      'L'-40H  ;MOVE CURSOR RIGHT
0016  C      VDOWN   EQU      'V'-40H  ;MOVE CURSOR DOWN
001F  C      VLIN    EQU      '^-40H  ;NEW LINE (CR,LF)
001A  C      VCLRS   EQU      'Z'-40H  ;CLEAR AND HOME CURSOR
001E  C      VHOME   EQU      '^'-40H  ;HOME CURSOR
0003  C      ETX     EQU      'C' - 40H ;ASCII ETX CHAR
000E  C      ACK     EQU      'F' - 40H ;ASCII ACK CHAR
0011  C      XON     EQU      'Q' - 40H ;ASCII XON (DC1) CHAR
0013  C      XOFF    EQU      'S' - 40H ;ASCII XOFF (DC3) CHAR
C
C
C      ;ESCAPE SEQUENCE CHARACTERS
C
0018  C      ESC     EQU      1BH      ;^[, ESC = ESCAPE
C
0023  C      VLOCK   EQU      '#'      ;LOCK KEYBOARD
0022  C      VUNLK   EQU      '"'      ;UNLOCK KEYBOARD
003D  C      VCAD     EQU      '='      ;CURSOR ADDRESSING
0053  C      VSAD     EQU      'S'      ;SCREEN ADDRESSING
0051  C      VINC     EQU      'Q'      ;INSERT CHAR
0057  C      VDEL    EQU      'W'      ;DELETE CHAR
0045  C      VINL     EQU      'E'      ;INSERT LINE
0052  C      VDELL    EQU      'R'      ;DELETE LINE
0054  C      VCEOL    EQU      'T'      ;CLEAR TO END OF LINE
0059  C      VCEDS    EQU      'Y'      ;CLEAR TO END OF SCREEN (WINDOW)
C
0061  C      VSAL     EQU      'a'      ;START ALTERNATE CHARACTER SET
0041  C      VEAL     EQU      'A'      ;END
0029  C      VSHI     EQU      ')'      ;START HALF INTENSITY
0028  C      VEHI     EQU      '('      ;END
0067  C      VSGH     EQU      'g'      ;START GRAPHICS
0047  C      VEGH     EQU      'G'      ;END
0062  C      VRON     EQU      'b'      ;SET REVERSE VIDEO BACKGROUND (BLACK ON WHITE)
0064  C      VROFF    EQU      'd'      ;CLEAR REVERSE VIDEO BACKGROUND (SET TO WHITE ON BLACK)
002E  C      VCTYP    EQU      '.'      ;CURSOR TYPE DEFINE
006A  C      VSRV     EQU      'j'      ;START REVERSE VIDEO
006B  C      VERV     EQU      'k'      ;END REVERSE VIDEO
006C  C      VSUL     EQU      'l'      ;START UNDERLINE
006D  C      VEUL     EQU      'm'      ;END UNDERLINE
005E  C      VSBLE    EQU      '^'      ;START BLINK
0071  C      VEBL     EQU      'q'      ;END BLINK
003E  C      EKC      EQU      ')'      ;ENABLE KEY CLICK
003C  C      DKC      EQU      '('      ;DISABLE KEY CLICK
0065  C      EFKT     EQU      'e'      ;ENABLE FUNCTION KEY TRANSLATE
0066  C      DFXT     EQU      'f'      ;DISABLE FUNCTION KEY TRANSLATE

```


'DATA AND MACRO DEFINITIONS'

```
C
007A      C      VWDEF EQU      'z'      ;WINDOW DEFINE
0073      C      VWSET EQU      's'      ;WINDOW SET
006F      C      VOFF EQU      'o'      ;DEFINE X-Y OFFSET
007B      C      VBACK EQU      'x'      ;DEFINE BACKGROUND ATTRIBUTES
C
0008      C      UNDBIT EQU      00001000B ;UNDERLINE BIT
0010      C      ALTBIT EQU      00010000B ;
0004      C      REVBIT EQU      00000100B ;REVERSE VIDEO BIT
0002      C      BLKBIT EQU      00000010B ;BLINK BIT
0001      C      HAFBIT EQU      00000001B ;HALF INTENSITY BIT
C
C      PAGE
```

```
C
C
C      ;BOOT CONSTANTS
C
4000    C      BOOTADDR      EQU      4000H      ;BOOT LOAD ADDRESS FOR P-SYSTEM
C
0000    C      BIOSBANK      EQU      00000000B   ;BIOS BANK
0000    C      BOOTBANK      EQU      00000000B   ;BOOT STRAP LOADER BANK
C
CC00    C      TEMPBUF EQU    0CC00H              ;TEMPORARY BUFFER IN VIDED BANK
C
0002    C      PRDD_CODE     EQU      2            ; USED FOR DMA ADDRESS WHEN BOOTING SYSTEM
C
C
C
C      PAGE
```

```
C
C
C      ;INTERRUPT CONSTANTS
FFE0  C      SIO_I_TBL      EQU      0FFE0H      ;LOCATION IN RAM OF SIO INTERRUPT VECTOR
C                                              ;MUST BE IN HIGH RAM AND END IN 0
FFFE  C      INTVEC  EQU      0FFFEH      ;OTHER INTERRUPT'S VECTOR
C
C      ;INTERRUPT DEVICE NUMBERS
C
0000  C      KBD_DEV EQU      0      ;KEYBOARD DEVICE NUMBER
0001  C      S_CH_A_I      EQU      1      ;DEVICE NUMBER FOR SERIAL CHANNEL A INPUT
0002  C      S_CH_B_I      EQU      2      ;DEVICE NUMBER FOR SERIAL CHANNEL B INPUT
0003  C      S_CH_A_O      EQU      3      ;DEVICE NUMBER FOR SERIAL CHANNEL A OUTPUT
0004  C      S_CH_B_O      EQU      4      ;DEVICE NUMBER FOR SERIAL CHANNEL B OUTPUT
C
C      PAGE
```

```
C
C
C      ;ROM'S RAM VALUES
C
C      ;FLAGS FOR CONSOLE AND FUNCTION KEY TABLE FLAGS
C
0000  C      SYS_TBL EQU      0      ;USE SYSTEM TABLE
0001  C      APP_TBL EQU      1      ;USE APPLICATION TABLE
0002  C      NO_TBL  EQU      2      ;DON'T USE TRANSLATION TABLE (FUNCTION KEY ONLY)
C
C      PAGE
```

```
C
C
C      ;TABLE DEFINITION LOCATIONS
C
C      ;SYSTEM TRACK LOCATIONS
C
0002  C      KCTRK   EQU      2                ;TRACK FOR KEYBOARD AND CHARACTER FONT DEFINITIONS
0011  C      KEYSEC  EQU      17             ;SECTOR FOR KEYBOARD DEFINITION
0014  C      CHARSEC EQU      20             ;SECTOR FOR CHARACTER FONT DEFINITION
001E  C      ALTSEC  EQU      30             ;SECTOR FOR ALTERNATE CHARACTER FONT DEFINITION
C
C      ;RAM LOCATIONS
C
2000  C      RAMTBL  EQU      2000H          ;KEYBOARD AND CONSOLE TABLE POINTERS
0000  C      CHRFONT EQU      0000H          ;CHARACTER FONT RAM LOCATION
0000  C      ALTFONT EQU      800H           ;ALTERNATE CHARACTER FONT RAM LOCATION
C
C
C      PAGE
```

```

C
C
C      ;CP/M DEFINITIONS
C
000E   C      SVER   EQU    14
0040   C      ERGISE EQU    64      ; DEFAULT = 64K VERSION
C
0016   C      CVER   EQU    22      ; CPM VERSION NUMBER
000E   C      CBIOSV EQU    SVER    ; CBIOS VERSION
0002   C      NDSKS   EQU     2
0400   C      K       EQU   1024
0003   C      ABTC    EQU   'C'-40H ; ABORT CHARACTER
C
C      ;DRIVE DEFINITIONS
C
0100   C      LSECB   EQU    256    ; LENGTH OF A SECTOR
0028   C      MTRK    EQU     40    ; MAXIMUM TRACKS
000A   C      MSEC    EQU     10    ; MAXIMUM SECTORS
00E5   C      FMTCR   EQU   0E5H    ; DATA FORMAT CHAR
C
C      ;SET CP/M SYMBOLS.
C
0003   C      IOBYTE  EQU     3
0004   C      CDISK   EQU     4      ; CURRENT LOGGED-IN CPM DISK
0005   C      SYS     EQU     5      ; CPM MONITOR CALL ADDRESS
0006   C      SYSL    EQU   SYS+1    ; (LWA OF CP/M)
0040   C      TIMPTR  EQU    40H     ; POINTS TO WHERE TIME KEPT
0000   C      DBUF    EQU    80H     ; DEFAULT DISK BUFFER
0006   C      OM.LRAM EQU   SYSL
0000   C      SYSR    EQU     0      ; RETURN TO SYSTEM
0004   C      SYSDSK  EQU   CDISK    ; CONTAINS CURRENT SYSTEM DISK #
0010   C      SYSDAT  EQU    10H     ; DATE IN DDMMYY
0013   C      SYSTIM  EQU   SYSDAT+3 ; TIME IN HHMMSS
0100   C      ORGP    EQU    100H    ; TRANSIENT PROGRAM ORIGIN
C      ;EOL
C      ; DOES NOT EXIST IN CP/M
0024   C      EOM     EQU   '$'      ; END OF MESSAGE (FOR SYSTEM FUNCTION)
001A   C      EOF     EQU   'Z'-40H  ; END OF FILE
0000   C      L$PRU   EQU    128     ; BYTES PER FLOPPY SECTOR
C
005C   C      FCB     EQU     5CH    ; SYSTEM DEFAULT FILE CONTROL BLOCK
007C   C      FCB_NR  EQU   FCB+32   ; NEXT RECORD = LRN
000C   C      F_EX    EQU     12     ; EXTEND FIELD
000E   C      F_S2    EQU     14     ; CPM SYSTEM USAGE **USED WITH 2.0 SIZ FUNCT**
000F   C      F_RC    EQU     15     ; RECORD COUNT (0 TO 128)
0020   C      F_NR    EQU     32     ; NEXT RECORD ORDINAL IN FCB
0021   C      F_RRF   EQU     33     ; R0, R1, R2 (RANDOM FIELD) IN FCB
0008   C      L_FILEN EQU     8      ; LENGTH OF FILE NAME
0003   C      L_FILEX EQU     3      ; LENGTH OF FILENAME SUFFIX
0024   C      L_FCB   EQU     36     ; LENGTH OF ENTIRE FCB (COMP WITH 2.0)
0000   C      DMA     EQU    80H     ; WHERE SECTORS ARE READ
0000   C      TBUF    EQU    DMA     ; WHERE CCP PUTS COMMAND LINE
C
C      PAGE

```

```
C
C
C      ;      FIRMWARE RAM DEFINITIONS
C
C      ;TEMPBUF:      =      HSTBUF      ;* TEMPBUF WILL BE THE SAME AREA AS THE SPACE
C      ;      RESERVED FOR KEYBOARD AND CONSOLE TABLES --
C      ;      IT CAN ONLY BE USED BEFORE THESE AREAS ARE INITIALIZED
C
C      PAGE
```



```

C
C
C      ;      TRUE/FALSE DEFINITIONS
C
0000    C      FALSE EQU      0
FFFF    C      TRUE  EQU      NOT FALSE
C
C
C      ;DEFINED FUNCTION CODES
C
0100    C      ROMVEC EQU      100H      ;START OF ROM VECTOR
C
2000    C      MONRAM EQU      2000H
FD40    C      COMJMP EQU      0FD40H    ;COMMON MEMORY JUMP VECTORS
C      ;(BANKJMP,BANKMOVE,BANKCALL,DMA READ AND WRITE)
FD4A    C      MRAM   EQU      0FD4AH    ;HIGH MEMORY AREA FOR VARIABLES USED BY ROM AND BIGS
FEF0    C      HI_ROUT EQU      0FEF0H    ;HIGH MEMORY ROUTINES MOVED FROM ROM
C
0009    C      TAB    EQU      09H      ;TAB
007F    C      ERC    EQU      7FH      ;ILLEGAL KEY
C
0FA0    C      @FREQ  EQU      4000     ;FREQ OF PROCESSOR
00B5    C      SCLFRE EQU      @FREQ/22 ;FOR DELAY ROUTINE
C
00FF    C      BAD    EQU      0FFH     ;ENTRY IN KEYBD TABLE IF KEY IS INVALID
00FF    C      BADKEY EQU      BAD      ;
00FE    C      TOG_ALT_SET EQU      0FEH ;ENTRY IN KEYBD TABLE IF KEY IS USED TO
C      ;TOGGLE ALTERNATE CHAR SET
C
26C0    C      KEYTBL EQU      2800H-140H ;KEY CODE LOCATION
00FE    C      ROMVER EQU      0FEH      ;ROM VERSION NUMBER LOCATION
C
C      ;COLD BOOT MASKS
C
0004    C      SP_KEY  EQU      00000100B ;LOAD KEYBOARD TRANSLATION CODE FROM DISK
0002    C      SP_CHAR EQU      00000010B ;LOAD CHARACTER FONT FROM DISK
0001    C      SP_ALT  EQU      00000001B ;LOAD ALTERNATE CHARACTER FONT FROM DISK
C
C      PAGE

```

```
C
C
C      ;      *****
C      ;      *
C      ;      *      MACRO DEFINITIONS      *
C      ;      *
C      ;      *****
C
C      ;ALIGN  -- SET ORIGIN TO EVEN BOUNDARY.
C
C      ALIGN  MACRO      SIZ,0
C              DS          (%2+(%1)-1)/(%1)*(%1)-%2
C              ENDM
C
C      ;PUSHAL -- PUSH ALL REGISTER PAIRS ON THE STACK
C
C      PUSHAL MACRO
C              PUSH  BC
C              PUSH  DE
C              PUSH  HL
C              ENDM
C
C      ;POPALL -- POP ALL REGISTER PAIRS OFF THE STACK
C
C      POPALL MACRO
C              POP   HL
C              POP   DE
C              POP   BC
C              ENDM
C
C      ;      Endx      SYSTEXT
C
C      PAGE
```

'EXECUTIVE ROM MONITOR'
'DATA AND MACRO DEFINITIONS'

MACRO-80 3.44 09-Dec-81

PAGE 1-25

C

PAGE

'EXECUTIVE ROM MONITOR'
'DATA AND MACRO DEFINITIONS'

MACRO-80 3.44 09-Dec-81 PAGE 1-26

i

SUBTTL
PAGE

'COMMON RAM STORAGE DECLARATIONS'

'COMMON RAM STORAGE DECLARATIONS'

```

SUBTTL      'COMMON RAM STORAGE DECLARATIONS'
INCLUDE     RAMS10.MAC      ;RAM MEMORY LOCATIONS AND CONSTANTS

C
C
C      ;      *****
C      ;      *
C      ;      *      RAMS10.MAC -- STORAGE DECLARATIONS      *
C      ;      *
C      ;      *****
C
C      ;      RAMS10.MAC
C      ;      Used to assemble ROM resident and CBIOS
C
C      SUBTTL  'COMMON RAM STORAGE DECLARATIONS'
C
C      ;      Disk operation temps and control
C
FD4A      C      DMABANK      EQU      MRAM      ;Bank for read/write disk
FD4B      C      DMADR       EQU      DMABANK+1    ;Address for read/write disk
C
C      ;      Note order of xxxSEC,xxxTRK,xxxDSK must be maintained
C      ;      along with length (1,2,1).
C
FD4D      C      SAVSEC      EQU      DMADR+2      ;last sector requested
FD4E      C      SAVTRK      EQU      SAVSEC+1      ;last track requested
FD50      C      SDISK       EQU      SAVTRK+2      ;selected disk drive (0,1)
FD51      C      SAVTYP      EQU      SDISK+1      ;SELECTED TYPE (sector size)
C
C
FD52      C      IE_ADR      EQU      SAVTYP+1
C
FD53      C      CURPOS      EQU      IE_ADR+1      ;Current cursor position
C
C      ;      VARIABLES FOR HIGH RAM ROUTINES
C
FD55      C      TEMSAV      EQU      CURPOS+2      ;Used by jump bank routine
C
FD56      C      SRCBANK     EQU      TEMSAV+1      ;Source bank for block move
FD57      C      DESTBANK    EQU      SRCBANK+1     ;Destination bank for block move
C
C      ;      JMPSPOT and JMPADDR must stay next to each other
C      ;      To be filled in with a JMP instruction by the ROM
FD58      C      JMPSPOT     EQU      DESTBANK+1    ;Address to jump to for BANKJMP and BANKCALL
FD59      C      JMPADDR     EQU      JMPSPOT+1
FD5B      C      JMPBANK     EQU      JMPADDR+2     ;Bank to jump to for BANKJMP and BANKCALL
C
FD5C      C      EMRAM       EQU      JMPBANK+1
C
C      ;      INTERRUPT VARIABLES
C
FD5C      C      USR_BNK     EQU      JMPBANK+1     ;User's bank on interrupt bank switch
C
FD5F      C      I_PDL_BANK  EQU      USR_BNK+3
C
FD60      C      I_PDL_ADR   EQU      I_PDL_BANK+1
C

```

'COMMON RAM STORAGE DECLARATIONS'

```

FD62      C      USR_STK      EQU      I_POL_ADR+2      ;save current stk ptr
          C
          C      ;      Interrupt stack
          C
FDA0      C      ISTK          EQU      USR_STK+30*2+2
          C
          C      ;      ROM and BIOS stack
          C
FDDC      C      ROMSTK        EQU      ISTK+30*2
FDDC      C      BIOSK        EQU      ROMSTK
          C
FDDC      C      HIBUFF        EQU      ROMSTK
          C

```

PAGE

;

SUBTTL 'MONITOR RAM STORAGE'
PAGE

```

C          SUBTTL      'MONITOR RAM STORAGE'
C          INCLUDE     RAMSMON.MAC          ;MONITOR RAM MEMORY LOCATIONS
C          ;
C          ;*****
C          ;          *          MONITOR'S RAM STORAGE DEFINITION          *
C          ;          *
C          ;*****
C
C          ;RAMSMON.ASM
C          ;
C
C          ;      BIOS POINTERS
C          ;RTC POINTER
2000      CLK_PTR      EQU          MONRAM          ;POINTER TO CURRENT TIME
C          ;DISK CHANGE FLAG POINTER
2002      MEDIA      EQU          CLK_PTR + 2          ;ADDR FOR DISK CHANGE FLAG
2004      A_DRIVE    EQU          MEDIA + 2          ;ADDR FOR DRIVE A DISK CHANGE FLAG
2006      B_DRIVE    EQU          A_DRIVE + 2          ;ADDR FOR DRIVE B DISK CHANGE FLAG
C
C          ;PROTOCOL HANDLING VARIABLES:
C
C          ;PROTOCOL MODE FOR SIO CHANNEL A.
C          ;0 = NO PROTOCOL,
C          ;1 = XON/XOFF PROTOCOL AND
2008      A_PROTCL    EQU          B_DRIVE + 2          ;2 = ETX/ACK PROTOCOL.
C
C          ;PROTOCOL MODE FOR SIO CHANNEL B.
C          ;0 = NO PROTOCOL,
C          ;1 = XON/XOFF PROTOCOL AND
2009      B_PROTCL    EQU          A_PROTCL + 1          ;2 = ETX/ACK PROTOCOL.
C
C          ;      KEYBOARD MASKS:
200A      K_MSK_TBL    EQU          B_PROTCL + 1
C
C          ;      INITIAL VALUES --
C          ;
C          ;          11100011B          ;ROW 0 -- STRIP CTRL/ALPHA/SHIFT
C          ;          11111111B          ;ROWS 1-6 -- STRIP NOTHING
C          ;          11111111B
C          ;          11111111B
C          ;          11111111B
C          ;          11111111B
C          ;          11111111B
C          ;          11010111B          ;ROW 7 -- STRIP ALPHA/INVALID RET
C
C          ;
C          ;      SPECIAL KEY LOCATIONS:
C          ;
2012      CT_KEY      EQU          K_MSK_TBL + 8          ;000_101B          ;CTRL: ROW 0, COL 2
2013      AL_KEY      EQU          CT_KEY + 1          ;111_011B          ;ALPHA: ROW 7, COL 3

```


'EXECUTIVE ROM MONITOR'
'MONITOR RAM STORAGE'

MACRO-80 3.44 09-Dec-81 PAGE 1-31

2014	C	SH_KEY	EQU	AL_KEY + 1	;000_100B	;SHIFT: ROW 0, COL 4
	C					
2015	C	KEYS	EQU	SH_KEY + 1		;1ST 64BYTE-NORMAL, 2ND 64-SHIFT
	C					;3RD 64-CONTROL, 4TH 64-ALPHA
	C					;5TH 64-CONTROL SHIFT TABLE
	C	; FUNCTION KEY TABLES				
	C					
2155	C	FKEY_ADR	EQU	KEYS + 320		;SPACE RESVERED FOR FUNCTION KEYS
	C					
	C					
	C	;CONSOLE POINTERS				
	C					
2355	C	HERTZ	EQU	FKEY_ADR + 200H		;HERTZ RATE (50 OR 60)
2356	C	CURTYPE	EQU	HERTZ + 1		;CURSOR TYPE (3 MSB'S) --
	C					; (BLOCK, UNDERLINE, BLINK)
2357	C	BACKGND	EQU	CURTYPE + 1		;BACKGROUND ATTRIBUTE FLAG
2358	C	KCLICK	EQU	BACKGND + 1		;KEY CLICK FLAG
2359	C	ROMCON	EQU	KCLICK + 1		;POINTER TO ROM CONSOLE STRUCTURE
	C					
	C	PAGE				

```

C
C
C      ;NON-CONFIGURABLE PORTION
C
C      ;CONSOLE
C      ;ROM'S CONSOLE TABLE STRUCTURE
235B  C      CONTBLS      EQU      ROMCON + 2
C      ;CURRENT TABLE ADDRESS
235B  C      CTBLADDR     EQU      ROMCON + 2      ;CURRENT TABLE TO BE SEARCHED
C
C      ;POINTERS TO TABLES
C      ;
C      ;      CTBLADDR + 2      ; BASE TABLE POINTER
C      ;      CTBLADDR + 4      ; ESCAPE TABLE POINTER
C      ;      CTBLADDR + 6      ; CONTROL TABLE POINTER
C      ;      CTBLADDR + 8      ; Y COORDINATE TABLE POINTER
C      ;      CTBLADDR + 10     ; X COORDINATE TABLE POINTER
C      ;      CTBLADDR + 12     ; WINDOW DEFINE TABLE POINTER
C      ;      CTBLADDR + 14     ; FIRST ROW OF WINDOW TABLE POINTER
C      ;      CTBLADDR + 16     ; FIRST COLUMN OF WINDOW TABLE POINTER
C      ;      CTBLADDR + 18     ; LAST ROW OF WINDOW TABLE POINTER
C      ;      CTBLADDR + 20     ; LAST COLUMN OF WINDOW TABLE POINTER
C      ;      CTBLADDR + 22     ; WINDOW SET TABLE POINTER
C      ;      CTBLADDR + 24     ; CURSOR TYPE TABLE POINTER
C      ;      CTBLADDR + 26     ; X-Y OFFSET TABLE POINTER
C      ;      CTBLADDR + 28     ; BACKGROUND ATTRIBUTE POINTER
C      ;
C      ;CONSOLE VARIABLES
2379  C      OFFSET      EQU      CTBLADDR + 30      ;OFFSET FOR X-Y COORDINATES
237A  C      CFLAG       EQU      OFFSET + 1          ;CONSOLE FLAG
C
237B  C      CURWIND      EQU      CFLAG + 1          ;CURRENT WINDOW DEFINITION
C      ;(THESE BYTES MUST STAY TOGETHER)
237B  C      FRAWIND      EQU      CFLAG + 1          ;FIRST WORD ADDRESS OF WINDOW
237D  C      WINDROWS     EQU      FRAWIND + 2        ;NUMBER OF ROWS IN WINDOW
237E  C      WINDCOLS     EQU      WINDROWS + 1       ;NUMBER OF COLUMNS IN WINDOW
C
237F  C      TEMP         EQU      WINDCOLS + 1       ;TEMPORARY USED BY SCROLL ROUTINE
C
C      ;WHILE WINDOW IS BEING DEFINED --
2380  C      WINDNUM      EQU      TEMP + 1           ; NUMBER OF WINDOW
2381  C      TEMPFWA      EQU      WINDNUM + 1        ; NEW FWA
2383  C      TEMPROWS     EQU      TEMPFWA + 2        ; NEW NUMBER OF ROWS
2384  C      FSTROW       EQU      TEMPROWS + 1       ; FIRST ROW NUMBER
2385  C      FSTCOL        EQU      FSTROW + 1        ; FIRST COLUMN NUMBER
2386  C      SCRL_UPF      EQU      FSTCOL + 1        ;0= SCROL_DOWN, 0FFH= SCROL_UP
2387  C      WINDDEFS      EQU      SCRL_UPF + 1      ;ALL 16 WINDOW DEFINITIONS
C
C      PAGE

```

```

C
C
C      ;MONITOR VARIABLES
C
C      ;SINCE CP/M CANNOT BOOT OFF B:, THIS CELL IS USED
C      ;TO INVERT THE NAMES OF THE 2 DRIVES:
C      ;      =0, ALL NORMAL, A=A:, B=B:
C      ;      =1, ALL INVERTED, A=B:, B=A:
23C7  C      DSKSWP      EQU      WINDEFS + 64
C
C
C      ;DISK VARIABLES
23C8  C      RTRY      EQU      DSKSWP + 1      ;RETRY COUNTER
23C9  C      DSTSB     EQU      RTRY + 1      ;DISK STATUS BYTES
23CF  C      NUMSEC    EQU      DSTSB + 6      ;(1) # OF SECTORS TO R/W
23D0  C      R_WCOM    EQU      NUMSEC + 1      ;(1) # OF SECTORS TO R/W
23D1  C      NRETRY    EQU      R_WCOM + 1      ;INITIAL RETRY COUNTER
23D2  C      WP_STS     EQU      NRETRY + 1      ;WRITE PROTECT STATUS
23D3  C      DRV_MSK    EQU      WP_STS + 1
C
C      ;INTERRUPT VARIABLES
23D4  C      CUR_I_ST   EQU      DRV_MSK + 1      ;CELL TO SAVE CURRENT INTERRUPT
C      ;STATE
23D6  C      I_POL_BNK  EQU      CUR_I_ST + 2      ;CELL TO SAVE INTERRUPT POLLING
C      ;ROUTINE'S BANK MASK
C
23D7  C      S_A_FLG    EQU      I_POL_BNK + 1      ;FLAG SET IF --
C      ; CH. A TX BUFF EMPTY INT OCCURS
C      ;AND NO CHARS READY
C
23D8  C      S_B_FLG    EQU      S_A_FLG + 1      ;FLAG SET IF --
C      ; CH. B TX BUFF EMPTY INT OCCURS
C
C      ;FUNCTION KEY TABLE POINTERS
23D9  C      FKEYFLAG   EQU      S_B_FLG + 1      ;FLAG TO INDICATE --
C      ; WHICH TRANSLATION TABLE TO USE
C      ;AND NO CHARS READY
C
C      ;PARALLEL PORT VARIABLES
23DA  C      PIACTL     EQU      FKEYFLAG + 1
23DB  C      PP.MODE    EQU      PIACTL + 1
C
C      ;IEEE 488 VARIABLES:
23DC  C      IE_CHAR    EQU      PP.MODE + 1
C
C      ;KEYBOARD SCAN VARIABLES
23DD  C      KEYLCK     EQU      IE_CHAR + 1      ;ZERO IF LOCKED KEYBOARD
23DE  C      KEYLST     EQU      KEYLCK + 1
23DE  C      FIRSTKEY   EQU      KEYLST
23E0  C      SECONDKEY  EQU      FIRSTKEY + 2
23E2  C      LASTKEY    EQU      KEYLST + 4
C
C      ;FUNCTION KEY TRANSLATION VARIABLES

```

'MONITOR RAM STORAGE'

23E4	C	COUNT	EQU	KEYLST + 6	;NUMBER OF KEYS LEFT TO TRANSLATE
23E5	C	FPNTR	EQU	COUNT + 1	;PNTR TO CURRENT POS IN TRANX TABLE
	C				
	C				
	C				
	C				
23E7	C	LAST_TCK	EQU	FPNTR + 2	;LAST VALUE READ FROM --
	C				; TICK OVERFLOW COUNTER
23E8	C	DCHK_FLG	EQU	LAST_TCK + 1	;# OF TICKS LEFT BEFORE --
	C				; DISK CHANGE IS SERVICED
23E9	C	SYS_TICKS	EQU	DCHK_FLG + 1	;SYSTEM TICK COUNT
	C				
	C				
	C				
	C				
23EA	C	ERRFLG	EQU	SYS_TICKS + 1	;ERROR FLAG
	C				
	C				
	C	PAGE			

```

C
C
C      ;REAL TIME CLOCK VARIABLES
C
C      ;4 BYTES FOR --
C      ;DAYS (# OF DAYS SINCE JAN 1, 1979)
C
C      ;HOURS
C      ;MINUTES
23EF  C      ROMCLK      EQU      ERRFLG + 5      ;SECONDS
C
23F0  C      SEKDEL      EQU      ROMCLK + 1      ;SET FOR SEEK-RESTORE COMMAND IN ROM
C      ;LOWEST 2 BITS ARE SPEED. SIEMENS = 3H, MPI = 0H
C
23F1  C      RDFLAG      EQU      SEKDEL + 1      ;READ FLAG
C
C
C      ; USED TO DESELECT DRIVE WHEN THERE IS NO ACTIVITY
C      ; ON DRIVE FOR N SECONDS. SEE FDSK ROUTINE
23F2  C      DACTIVE      EQU      RDFLAG + 1      ;=0 BY FDSK, USED BY UPTIM
C
C      ;INTERRUPT TABLE
C
23F3  C      INT_TBL      EQU      DACTIVE + 1      ;INTERRUPT HANDLER TABLE
C      ;DMA
C      ;PARALLEL
C      ;SIO TX CHANNEL B
C      ;SIO EXTERNAL/STATUS CHANNEL B
C      ;SIO RX CHANNEL B
C      ;SIO RX CHANNEL B SPECIAL CONDITION
C      ;SIO TX CHANNEL A
C      ;SIO EXTERNAL/STATUS CHANNEL A
C      ;SIO RX CHANNEL A
C      ;SIO RX CHANNEL A SPECIAL CONDITION
C      ;INTELLIGENT KEYBOARD
C      ;REAL TIME CLOCK (TICK) INTERRUPT
C      ;FLOPPY DISK CONTROLLER
C      ;SYSTEM RESET
C
C
C      ;-----
C      ;      DATA AREAS
C      ;-----
C      ;RESERVE STORAGE FOR DEVICE BUFFER LIST ROOTS
C      ; ** REPT D_RT_LEN*NO_DEVS ;INITIALIZE ALL POINTERS TO 0
C
241D  C      D_RT_TBL      EQU      INT_TBL + 42
C
C      ; ** ENDM ;
C
243B  C      FRE_ADDR      EQU      D_RT_TBL + (D_RT_LEN*NO_DEVS) ;(2 BYTES) FREE LIST ROOT.
C      ; POINTER TO FIRST FREE BUFFER
C      ; IS INITIALIZED TO POINT TO FIRST BUFFER
C      ;(1 BYTE) FREE BUFFER COUNT.
C      ; IS INITIALIZED TO NUMBER OF BUFFERS.
C
C      ;-----

```

```

C      ;   BUFFERS FOLLOW. THE FIRST ENTRY IN EACH IS INITIALIZED TO POINT TO
C      ;   THE NEXT SO THAT ALL ARE IN THE FREE LIST.
C      ;-----
243E   C      BUFF_ST      EQU          FRE_ADDR + 3
C
C      ;PROTOCOL HANDLING VARIABLES:
C
C      ;TRANSMIT DISABLE FLAG FOR CHANNEL A.
C      ;000H = TRANSMIT ENABLED AND
C      ;0FFH = TRANSMIT DISABLED BECAUSE OF PROTOCOL
275E   C      A_XMT_OFF    EQU          BUFF_ST + ((MX_CH_BF + BF_OV_HD) * NO_BUFS)
C
C      ;TRANSMIT DISABLE FLAG FOR CHANNEL B.
C      ;000H = TRANSMIT ENABLED AND
C      ;0FFH = TRANSMIT DISABLED BECAUSE OF PROTOCOL
275F   C      B_XMT_OFF    EQU          A_XMT_OFF + 1
C
C      ;RESET VARIABLES
2760   C      RSTHLD        EQU          B_XMT_OFF + 1      ;SET BY INTERRUPT ROUTINE IF --
C                                          ; RESET BUTTON IS HELD DOWN.
2761   C      RSTMODE       EQU          RSTHLD + 1      ;RESET MODE --
C                                          ;      0 = BOOT AFTER RESET
C                                          ;     NOT 0 = SYSTEM AFTER RESET
C
C      PAGE

```

'EXECUTIVE ROM MONITOR'
'MONITOR RAM STORAGE'

MACRO-80 3.44 09-Dec-81 PAGE 1-37

i

SUBTTL 'BOOT AND INITIALIZATION ROUTINES'
PAGE

```

C          SUBTTL      'BOOT AND INITIALIZATION ROUTINES'
C          INCLUDE ROM0                      ;BOOT AND INITIALIZATION ROUTINES
C
C      ;.LAST MODIFIED: ROGER W. CHAPMAN REV 1.21 (4.030)
C      ;.TITLE: EXECUTIVE MONITOR ROM
C      ;.COMMENTS:
C      ;
C      ;      +-----+
C      ;      |
C      ;      | EXECUTIVE MONITOR |
C      ;      | COPYRIGHT 1984   |
C      ;      | BY
C      ;      | OSBORNE COMPUTER CORPORATION |
C      ;      | 26538 DANTI COURT |
C      ;      | HAYWARD, CA 94545 |
C      ;      |
C      ;      | CREATED BY
C      ;      | ROGER W. CHAPMAN |
C      ;      | WENDY JAMES
C      ;      | DAVID BERG
C      ;      | CHRIS MAYER
C      ;      |
C      ;      +-----+
C      ;
C      PAGE
  
```



```
C
C ;-----
C ; REVISION HISTORY
C ;
C ; 03/24/83:
C ;     A. MODIFIED RADR ROUTINE TO USE 100 MS TIME OUT INSTEAD OF 60 MS. THIS
C ;     CHANGE WAS NECESSARY BECAUSE OF THE DISK CHANGE DETECT ALGORITHM,
C ;     WHICH COULD CAUSE THE CONTROLLER TO LOSE SYNC AND POTENTIALLY MISS THE
C ;     NEXT ADDRESS MARK.
C ;
C ;     B. MODIFIED SIGN-ON TO REFLECT CURRENT RELEASE VERSION NUMBER V1.2
C ;
C ; 06/27/83:
C ;     A. ALL SOURCE CODE CONVERTED FROM ACT TO M80
C ;
C ;     B. ROM SOURCE SPLIT INTO EIGHT MODULES
C ;
C ; 07/11/83:
C ;     A. MODIFIED BUFFER ROUTINES (IN ROM7.MAC) TO FIX BUFFER ALLOCATION
C ;     PROBLEM.
C ;
C ; 07/25/83:
C ;     A. MODIFIED DISK ROUTINES (IN ROM5.MAC) TO SUPPORT DOUBLE SIDED DISK
C ;     DRIVES
C ;
C ; 08/01/83:
C ;     A. CORRECTED PARAMETER PASSING PROBLEM IN IEEE ROUTINES (ROM4.MAC)
C ;
C ;     B. MODIFIED SIGN-ON TO REFLECT CURRENT STATE OF ROM CODE X1.3
C ;
C ; 01/30/84:
C ;     A. Removed double-sided support from SENDEN (ROM5.MAC)
C ;
C ;     B. Changed interrupt handler to push all register on stack instead of
C ;     using alternate regs
C ;
C ;     C. Modified sign-on to reflect current version # V1.21
C ;     (double-sided support was removed, bug fix was implemented)
C ;-----
C
C PAGE
```

'BOOT AND INITIALIZATION ROUTINES'

```
C
C
C      ;      MODULE 0 --ROM0.MAC-- ENTRY POINTS
C      ;
C      ;      BOOT AND INITIALIZE SYSTEM
C      ;
C      ;      START:      START OF ROM
C      ;      AUTOBOOT:   BOOT THE SYSTEM
C      ;      INITIALIZE:  INITIALIZES (INIT.) MEMORY AND SYSTEM PIA
C      ;      CBOOT:      INIT. (STACK/MEMORY/INTERRUPTS) AND LOADS BOOT STRAP
C      ;      BOOT:       LOADS BOOT STRAP LOADER
C      ;      RES_SIO:    RESETS THE SIO
C      ;      INIT_INT:   INIT. THE SIO AND INTERRUPT STATE
C      ;      SYSINIT:    INIT. SYSTEM AND PARALLEL PIA
C      ;      INIT_POLL:  INIT. BANK AND ADDRESS OF INTERRUPT POLLING ROUTINE
C      ;      KB_INIT:    INIT. KEYBOARD TABLE POINTERS AND SCAN MASKS
C      ;      CONINIT:    INIT. ALL NON-ZERO VALUES FOR CONSOLE
C      ;      SETCHAR:    FILLS CHARACTER FONT RAM WITH CHARACTER SET
C      ;      SETJMP:     INIT. HIGH RAM JUMP VECTORS
C      ;      OSTR:       OUTPUTS STRING TO THE CONSOLE
C      ;
C      ;      NOTE.....THE ROM JUMP TABLE ALSO RESIDES IN THIS MODULE
C
C      PAGE
```

'BOOT AND INITIALIZATION ROUTINES'

```
C
C
C      .Z80
C      ;      .XLIST
C      ;      INCLUDE      SYSS1.MAC      ;SYSTEST FILE
C      ;      INCLUDE      RAMS10.MAC     ;RAM MEMORY LOCATIONS AND CONSTANTS
C      ;      INCLUDE      RAMSMON.MAC    ;MONITOR RAM MEMORY LOCATIONS
C      ;      .LIST
C      ;      ;SYSS1.MAC , RAMSMON.MAC AND RAMS10.MAC INCLUDED
C
C      ;      PAGE
C
C      ;      EXT      SET_INT, CON_IN, STINT, INT_HL, LENINT, R_INT_TBL, R_I_TBL_LEN
C      ;      EXT      IN_AL_BF, SKEY, COUT, SAI_STAT, SBI_STAT, SAD_STAT, SBD_STAT, SERAIN
C      ;      EXT      SERBIN, SERAOUT, SERBOUT, SETBAUD, IE.CO, IE.SI, IE.GTS, IE.TC, IE.OIM
C      ;      EXT      IE.ODM, IE.IDM, IE.PP, IE.INSTAT, IE.OSTAT, IE.INP, IE.OUT, PISTAT, POSTAT
C      ;      EXT      PARINP, PAROUT, RDRV, RSEC, WSEC, SENDEN, FORMAT, SELDRV, STEP, STEPIN
C      ;      EXT      STEPOUT, SEEK, HOME, FORINT, FMTTRK, DMAMRT, DMARD, READ, WRITE, RADR
C      ;      EXT      SETDMA, CTC_INIT, SIOINIT, INT_POLL, BANKJMP, BANKCALL, BANKMOVE
C      ;      EXT      CTBLPTRS, LENCTBL, PODIAG, L_PWR_UP, DIAG_INIT
C
C      ;      PUBLIC  AL_T_SZ, INIT_INT, OSTR, AUTOBOOT, INITIALIZE
C
C      PAGE
```

'BOOT AND INITIALIZATION ROUTINES'

```

C
C
0000'  C      ASEG
C      ORG  0      ;START THE LOC COUNTER AT LO MEM
C      .PHASE 0H
C
C      ; *****
C      ; *
C      ; *          START OF ROM
C      ; *
C      ; *****
C
0000  C      START:
C      ;INITIALIZE ALL DEPENDENT HARDWARE.
C
C
0000  31 FDDC  C      LD      SP,ROMSTK      ;SET STACK
0003  CD 003F  C      CALL   INITIALIZE      ;INITIALIZE MEMORY AND SYSTEM PIA
C
0006  AF      C      XOR      A
0007  32 2760  C      LD      (RSTHLD),A      ;CLEAR RSTHLD, THE RESET INTERRUPT ROUTINE
C      ; WILL SET RSTHLD IF RESET IS BEING PRESSED
C
000A  47      C      LD      B,A
000B  4F      C      LD      C,A      ;BC = 0
000C  CD 0E7A' C      CALL   SET_INT      ;INDIVIDUALLY DISABLE ALL INTERRUPTS EXCEPT RESET
C      ;(SIO HAS NOT BEEN INITIALIZED AT THIS POINT)
C
C      ;RESET SIO TO KEEP IT FROM INTERRUPTING AND CONFUSING THE RESET/POWER ON ISSUE
C
000F  CD 0218  C      CALL   RES_SIO
C
C      ;INITIALIZE SIO AND NORMAL INTERRUPT STATE
C
0012  CD 0223  C      CALL   INIT_INT      ;INITIALIZE SIO AND REENABLE ALL INTERRUPTS
C      ;(RESET SHOULD HAVE HAD TIME TO INTERRUPT AT THIS POINT)
C
C      ;CHECK FOR RESET
C
0015  3A 2760  C      LD      A,(RSTHLD)      ;CHECK FOR RESET INT
0018  B7      C      OR      A
0019  C2 001F  C      JP      NZ,AUTOBOOT      ;IF RESET PRESSED, BOOT
C
C      ;FALL THROUGH TO PWRUP IF RESET NOT PRESSED
C
001C  CD 08A4  C      CALL   PODIAG
C
C      PAGE

```

'BOOT AND INITIALIZATION ROUTINES'

```
001F      C      AUTOBOOT:
          C
          C
          C
          C      ;SIGN ON PROMPT
001F      11 03AE  C      LD      DE,IMSG
0022      CD 0323  C      CALL    OSTR      ;OUTPUT INITIAL MESSAGE
0025      FB      C      EI              ;ENABLE INTERRUPTS
          C
          C      ;BOOT
0026      CD 0018' C      XJ000: CALL    CON_IN      ;READ IN NEXT CHAR
          C
          C      CP      CR
0029      FE 0D    C
002B      06 00    C      LD      B,0      ;IF BOOT FROM DRIVE A
002D      28 0C    C      JR      Z,XJ001    ;IF COLD BOOT OFF OF A, TRY TO BOOT
          C
          C      INC     B      ;SWAP DRIVES: A=B, B=A
002F      04      C
0030      FE 09    C      CP      'I'-40H
0032      28 07    C      JR      Z,XJ001    ;IF NOT DRIVE B, TRY AGAIN
          C
          C      CP      'T'
0034      FE 54    C
0036      CA 00C8  C      JP      Z,L_PWR_UP  ;LOOP POWER UP TEST
          C
          C      JR      XJ000
0039      18 EB    C
          C
          C      XJ001: LD      A,B      ;A = 0 FOR DRIVE A, 1 FOR DRIVE B
003B      78      C      JP      BOOT
003C      C3 019F  C
          C
          C      PAGE
```

'BOOT AND INITIALIZATION ROUTINES'

```

003F      C      INITIALIZE:
          C      ;INITIALIZES MEMORY AND SYSTEM PIA
          C      ;ENTRY
          C      ;NONE
          C
          C      ;EXIT
          C      ;SYSTEM INITIALIZED
          C
          C
          C      ;INITIALIZE MEMORY
          C
          C      ;CLEAR ALL
003F      C      LD      HL,MONRAM      ;START
0042      C      LD      DE,MONRAM+1
0045      C      LD      BC,RSTMODE-MONRAM-1 ;LENGTH
0048      C      LD      (HL),0
004A      C      LDIR
          C
004C      C      LD      HL,MRAM
004F      C      LD      DE,MRAM+1
0052      C      LD      BC,EMRAM-MRAM
0055      C      LD      (HL),0
0057      C      LDIR
          C
          C      ;SET VALUES OF ONE
0059      C      LD      A,1      ;A = ONE
005B      C      LD      (KEYLCK),A ;INDICATE NOT LOCKED
          C
          C      ;SET VALUES OF TWO
005E      C      INC      A      ;A = TWO
005F      C      LD      (SEKDEL),A ;SET SEEK STEP RATE FOR SEMIENS
          C
          C      ;SET DISK RETRY NUMBER
0062      C      LD      A,10
0064      C      LD      (NRETRY),A
          C
          C      ; INITIALIZE DISK CHANGE DETECT VARIABLE
0067      C      LD      HL,WP_STS
006A      C      LD      (MEDIA),HL
006D      C      LD      (A_DRIVE),HL
0070      C      LD      (B_DRIVE),HL
          C
          C      ;INITIALIZE KEY CODE TABLE POINTERS
0073      C      CALL     KB_INIT
          C
          C
          C      ;INITIALIZE REAL TIME CLOCK POINTER
0076      C      LD      HL,ROMCLK
0079      C      LD      (CLK_PTR),HL ;POINT TO SPOT IN ROM
          C
          C      ;INITIALIZE CONSOLE VALUES
007C      C      CALL     CONINIT
          C

```

'BOOT AND INITIALIZATION ROUTINES'

```

                                C      ;INITIALIZE HIGH RAM JUMP VECTORS
007F  CD 030D                  C      CALL  SETJMP      ;INITIALIZE JUMP VECTORS FOR HIGH RAM ROUTINES
                                C                                ;(BANKMOVE, BANKJMP, BANKCALL, DMA READ AND WRITE)
                                C
                                C      ;INITIALIZE CHARACTER FONT RAM
0082  11 0000                  C      LD    DE,CHARFONT  ;INIT MAIN SET
0085  CD 02F8                  C      CALL  SETCHAR      ;
0088  11 0800                  C      LD    DE,CHARFONT+800H ;INIT ALTERNATE SET
008B  CD 02F8                  C      CALL  SETCHAR
                                C
                                C      PAGE
```

'BOOT AND INITIALIZATION ROUTINES'

```

C
C
C      ;SET INTERRUPTS
C
C      ;SET MODE 2 INTERRUPTS
C
008E  ED 5E      C      IM 2
C
C      ;SET INTERRUPT REGISTER
C
0090  3E FF      C      LD      A,HIGH SIO_I_TBL
0092  ED 47      C      LD      I,A
C
C      ;MOVE INTERRUPT ROUTINES TO HIGH RAM
C      ; (MUST DO THIS BEFORE INITIALIZING INTERRUPT JUMP TABLE)
C
0094  21 12FB'   C      LD      HL,STINT      ;START ADDRESS IN ROM
0097  11 FEF0    C      LD      DE,INT_HL      ;DESTINATION IN HIGH RAM
009A  01 00E6    C      LD      BC,LENINT      ;LENGTH
009D  ED B0      C      LDIR
C
C      ;INITIALIZE SIO INTERRUPT JUMP VECTORS
C
009F  06 08      C      LD      B,B          ;LOAD B WITH NUMBER OF ENTRIES TO INIT
00A1  21 FFE0    C      LD      HL,SIO_I_TBL    ;POINT HL TO LOCATION TO INIT
00A4  11 FEF0    C      LD      DE,INT_HL      ;LOAD DE WITH ADDRESS OF INTERRUPT HANDLER
00A7                                     C      XJ002:
00A7  73         C      LD      (HL),E        ;SAVE LOW BYTE OF INTERRUPT HANDLER ADDRESS
00A8  23         C      INC      HL           ;BUMP PTR TO NEXT CELL
00A9  72         C      LD      (HL),D        ;SAVE HI BYTE
00AA  23         C      INC      HL           ;BUMP PTR TO NEXT CELL
00AB  10 FA      C      DJNZ     XJ002        ;IF MORE ENTRIES
C
C      ;INITIALIZE OTHER INTERRUPT VECTOR
C
00AD  21 FFFE    C      LD      HL,INTVEC      ;LOCATION TO INITIALIZE
00B0  73         C      LD      (HL),E
00B1  23         C      INC      HL
00B2  72         C      LD      (HL),D
C
C      ;INITIALIZE INTERRUPT TABLE
C
00B3  21 1137'   C      LD      HL,R_INT_TBL
00B6  11 23F3    C      LD      DE,INT_TBL
00B9  01 002A    C      LD      BC,R_I_TBL_LEN
00BC  ED B0      C      LDIR
C
C      ;INITIALIZE I/O BUFFERS
00BE  CD 1161'   C      CALL     IN_AL_BF      ;INITIALIZE CHARACTER I/O BUFFERS
C
C      ;INITIALIZE POLLING ROUTINE ADDRESS AND BANK
00C1  CD 0294    C      CALL     INIT_POLL
C
C      ;INITIALIZE SYSTEM PIA
00C4  C3 0244    C      JP      SYSINIT      ;SYSINIT WILL RETURN
C      PAGE

```



```
C
C      ;ROM VERSION NUMBER
C
C      IF      $ 6T 0FCH
C      .PRINTX 'PAGE 0 OVERFLOW '
C      ENDIF
C
C      .DEPHASE
C
C      ORG      0FCH
C      .PHASE 0FCH
C
C      VERSION:
C      DB          2                ;PRODUCT CODE (EXECUTIVE = OCC 2)
C      DB          0                ;OPTIONS CODE (0 = BASE MACHINE)
C      DW      4030H                ;VERSION NUMBER
C
C      .DEPHASE
C
C
C
C      PAGE
```

00FC
00FC 02
00FD 00
00FE 4030

[illegible]

PAGE

'BOOT AND INITIALIZATION ROUTINES'

```

C
C
C      ;      DISK I/O
C
0157  C3 0B25'  C      JP      RDRV      ;CBIOS HOME
015A  C3 0B44'  C      JP      RSEC      ;CBIOS DISK SECTOR READ
015D  C3 0B4B'  C      JP      WSEC      ;CBIOS DISK SECTOR WRITE
0160  C3 0B97'  C      JP      SENDEN    ;CBIOS SENSE THE DENSITY OF DRIVE
0163  C3 0E1E'  C      JP      FORMAT    ;CBIOS FORMATING ROUTINE
0166  C3 0DBB'  C      JP      SELDRV    ;      SELECT DRIVE
0169  C3 0C2E'  C      JP      STEP      ;      STEP SAME DIRECTION
016C  C3 0C36'  C      JP      STEPOUT   ;      STEP OUT
016F  C3 0C32'  C      JP      STEPIN    ;      STEP IN
0172  C3 0C20'  C      JP      SEEK      ;      SEEK TO TRACK
0175  C3 0BF8'  C      JP      HOME      ;      HOME DISK DRIVE
0178  C3 0D7C'  C      JP      FORINT    ;      FORCE INTERRUPT
017B  C3 0D3B'  C      JP      FMTTRK    ;      FORMAT ONE TRACK
017E  C3 FF56    C      JP      DMAWRT    ;      DMA WRITE TO CONTROLER
0181  C3 FF39    C      JP      DMARD     ;      DMA READ FROM CONTROLER
C
C
C      ;      DIAGNOSTIC ENTRY POINTS
C
0184  C3 0323    C      JP      OSTR      ;      OUTPUT STRING TO CONSOLE
0187  C3 0AD6    C      JP      DIAG_INIT ;      INITIALIZE SYSTEM FOR DIAGNOSTIC
018A  C3 0C62'  C      JP      READ      ;      READ SECTOR (DIAG ONLY!!)
018D  C3 0C69'  C      JP      WRITE     ;      WRITE SECTOR (DIAG ONLY!!)
0190  C3 0CE9'  C      JP      RADR      ;      READ ADDRESS
C
C      PAGE

```

'BOOT AND INITIALIZATION ROUTINES'

```
C
C      ; *****
C      ; *
C      ; *          BOOT ROUTINES          *
C      ; *
C      ; *****
C
0193      C      CBOOT:
C      ;INITIALIZE STACK, MEMORY, I/O PORTS, INTERRUPTS
C      ;LOAD BOOT STRAP LOADER
C      ;ENTRY
C      ;A      =      DRIVE TO BOOT FROM
C
C      ;EXIT
C      ;NONE
C
0193      F3      C      DI          ;DISABLE INTERRUPTS WHILE WE
C      ;INIT EVERYTHING.
0194      31 FDDC  C      LD      SP,ROMSTK      ;SET STACK POINTER
0197      F5      C      PUSH     AF          ;SAVE DRIVE TO BOOT FROM
0198      CD 003F  C      CALL     INITIALIZE    ;INITIALIZE MEMORY AND PIA
019B      CD 0223  C      CALL     INIT_INT     ;INITIALIZE SIO AND INTERRUPT STATE
019E      F1      C      POP      AF          ;GET DRIVE TO BOOT FROM
C      ;      JMP      BOOT          ;FALL THROUGH TO BOOT
C      PAGE
```

'BOOT AND INITIALIZATION ROUTINES'

```

C
C
019F      C      BOOT:
           C      ;LOAD BOOTSTRAP
           C      ;ENTRY A = DRIVE TO BOOT FROM
C
019F 32 23C7 C      LD      (DSKSWP),A      ;SET DRIVE TO BOOT FROM
01A2 B7      C      OR      A              ;CHECK DRIVE TO SET DRIVE MASK
01A3 21 23D3 C      LD      HL,DRV_MSK
01A6 06 02   C      LD      B,2
01A8 20 02   C      JR      NZ,XJ012
C
01AA 06 04   C      LD      B,4
01AC 70      C      XJ012: LD      (HL),B      ;SAVE DRIVE MASK
C
           C      ;LOAD BOOT STRAP LOADER (ONE PHYSICAL SECTOR)
C
           C      ;READ SECTOR
C
01AD 11 034A C      XJ013: LD      DE,LD_MSG      ;POINT AT MESSAGE
01B0 CD 0323 C      CALL    OSTR          ;DISPLAY IT
C
01B3 CD 0B25' C      CALL    RDRV          ;HOME DRIVE
C
           C      ;SET "SAVTYP"
C
01B6 CD 0B97' C      CALL    SENDEN      ;DETERMINE DENSITY
01B9 20 3D   C      JR      NZ,XJ014      ;IF BAD...
C
01B8 21 4000 C      LD      HL,BOOTADDR
01BE 3E 00   C      LD      A,BOOTBANK
01C0 32 FD5B C      LD      (JMPBANK),A      ;SET BANK # TO JUMP TO
C
01C3 CD 0B1E' C      CALL    SETDMA
C
01C6 AF      C      XOR      A
01C7 32 FD4E C      LD      (SAVTRK),A      ;TRACK 0
01CA 3C      C      INC      A
01CB 32 FD4D C      LD      (SAVSEC),A      ;SECTOR 1
C
01CE 47      C      LD      B,A          ;READ 1 SECTOR
01CF 3A FD51 C      LD      A,(SAVTYP)      ;SAVE SAVTYP STATUS
01D2 F5      C      PUSH    AF
01D3 C8 8F   C      RES      1,A          ;SET SAVTYP TO READ SIDE 0
01D5 32 FD51 C      LD      (SAVTYP),A
01D8 CD 0B44' C      CALL    RSEC
01DB D1      C      POP      DE          ;RESTORE SAVTYP STATUS
01DC 7A      C      LD      A,D
01DD 32 FD51 C      LD      (SAVTYP),A
01E0 20 16   C      JR      NZ,XJ014      ;IF UNSUCCESSFUL SECTOR READ
C
01E2 3A 4000 C      LD      A,(BOOTADDR)      ;LOAD FIRST BYTE OF BOOT SECTOR
01E5 B7      C      OR      A              ;IS IT ZERO?
01E6 20 19   C      JR      NZ,XJ015      ;NO...
C      ;

```

'BOOT AND INITIALIZATION ROUTINES'

```

C          ;YES. IS PRODUCT CODE CORRECT?
01EB 3A 4004 C          LD      A,(BOOTADDR+4) ;LOAD PRODUCT CODE
01EB FE 02  C          CP      PROD_CODE      ;IS IT CORRECT?
01ED 20 12  C          JR      NZ,XJ015          ;NO...
C
01EF 21 4000 C          LD      HL,BOOTADDR      ;RETURN WITH HL = P-SYSTEM ENTRY ADDRESS
01F2 22 FD59 C          LD      (JMPADDR),HL      ;SET ADDRESS TO JUMP TO
01F5 CD FF88 C          CALL   BANKJMP      ;JUMP TO SYSTEM
C
01F8          C          XJ014: ;HARD READ ERROR
01F8 11 035E C          LD      DE,EB_MSG      ;DISPLAY MESSAGE
01FB CD 0323 C          CALL   OSTR
01FE C3 0207 C          JP      XJ016          ;AND WAIT FOR USER TO ENTER
C          ;CARRIAGE RETURN
C
0201          C          XJ015: ;INVALID SYSTEM DISK
0201 11 036E C          LD      DE,BD_MSG      ;DISPLAY ERROR
0204 CD 0323 C          CALL   OSTR
C
0207          C          XJ016: ;DISPLAY MESSAGE TRAILER AND
0207 11 0393 C          LD      DE,WT_CR_MSG
020A CD 0323 C          CALL   OSTR
C
020D CD 0018' C          XJ017: CALL   CON_IN      ;WAIT FOR USER TO ENTER CARRIAGE
0210 FE 0D  C          CP      CR          ;RETURN
0212 C2 020D C          JP      NZ,XJ017          ;NOT CR, GET ANOTHER CHAR
C          ;
0215 C3 01AD C          JP      XJ013          ;USER ENTERED CR, TRY TO BOOT AGAIN.
C
C          PAGE

```

'BOOT AND INITIALIZATION ROUTINES'

```
C
C ; *****
C ; *
C ; *          INITIALIZATION ROUTINES          *
C ; *
C ; *****
C
0218 C RES_SIO:
C ; RESET THE SIO BY SENDING CHANNEL RESETS TO BOTH CHANNELS.
C ; THIS KEEPS THE SIO FROM INTERRUPTING WHILE WE'RE CHECKING FOR
C ; SYSTEM RESET/POWER ON.
C ;
C ; ENTRY:
C ; NONE.
C ; EXIT:
C ; NONE.
C ;
0218 3E 18 C LD A,00011000B ;SIO CHANNEL RESET COMMAND
021A D3 0D C OUT (SIO_CNA),A ;SEND TO CHANNEL A
021C D3 0D C OUT (SIO_CNA),A ;TWICE
C
021E D3 0F C OUT (SIO_CNB),A ;SEND TO CHANNEL B
0220 D3 0F C OUT (SIO_CNB),A ;TWICE
0222 C9 C RET
C PAGE
```

'BOOT AND INITIALIZATION ROUTINES'

```
0223      C      INIT_INT:
          C      ;INITIALIZE SID AND INTERRUPT STATE.
          C      ;
          C      ;INIT_INT CALLS SET_INT, WHICH ENABLES INTERRUPTS.
          C      ;
          C      ;ENTRY
          C      ;NONE
          C
          C      ;EXIT
          C      ;NONE
          C
          C      ;INITIALIZE THE CLOCK-TIMER CHIP
          C
0223      CD 0A2C'  C      CALL    CTC_INIT
          C
          C      ;INITIALIZE SERIAL PORTS
0226      0E 0F      C      LD      C,SIO_CNB      ;PORT B
0228      CD 0A39'  C      CALL    SIOINIT
          C
022B      0E 0D      C      LD      C,SIO_CNA      ;PORT A
022D      CD 0A39'  C      CALL    SIOINIT
          C
          C
          C      ;INITIALIZE BAUD RATES TO 1200
0230      0E 08      C      LD      C,BD1200
0232      AF          C      XOR     A
          C      ;PORT A
0233      CD 0A7A'  C      CALL    SETBAUD
          C
0236      0E 08      C      LD      C,BD1200      ;PORT B
0238      3E 01      C      LD      A,1
023A      CD 0A7A'  C      CALL    SETBAUD
          C
          C      ;INITIALIZE INTERRUPT STATE
023D      01 F800     C      LD      BC,0F800H
0240      CD 0E7A'  C      CALL    SET_INT      ;INITIALIZE INTERRUPT STATES
0243      C9          C      RET
          C
          C      PAGE
```


'BOOT AND INITIALIZATION ROUTINES'

```

C
0244      C      SYSINIT:
C          C      ;INITIALIZE SYSTEM AND PARALLEL PIA
C          C      ;ENTRY
C          C      ;NONE
C
C          C      ;EXIT
C          C      ;NONE
C
C
C
C      ;INITIALIZE A CHANNEL OF SYSTEM PIA
C
0244      DB 01      C      IN      A, (SYS_CNA)      ;
0246      F6 04      C      OR      C_DAT      ;ENABLE DATA REG
0248      D3 01      C      OUT     (SYS_CNA),A      ;CONTROL A
C
024A      3E 80      C      LD      A, DATA_INIT      ;INITIALIZE -- ENABLE ROM
024C      D3 00      C      OUT     (SYS_DTA),A
C
024E      DB 01      C      IN      A, (SYS_CNA)      ;
0250      E6 FB      C      AND     C_DIR      ;ENABLE DIRECTION REG
0252      D3 01      C      OUT     (SYS_CNA),A      ;CONTROL A
C
0254      3E FF      C      LD      A, DIRA_INIT      ;INITIALIZE DIRECTION -- ALL OUTPUTS
0256      D3 00      C      OUT     (SYS_DTA),A
C
0258      DB 01      C      IN      A, (SYS_CNA)      ;
025A      F6 04      C      OR      C_DAT      ;ENABLE DATA REG
025C      E6 FD      C      AND     C_DMA      ;SET DMA INTERRUPT
025E      E6 CF      C      AND     C_I_KBD      ;SET INTELLIGENT KEYBOARD INTERRUPT
0260      D3 01      C      OUT     (SYS_CNA),A      ;CONTROL A
C
C      ;INITIALIZE B CHANNEL OF SYSTEM PIA
C
0262      DB 03      C      IN      A, (SYS_CNB)      ;
0264      F6 04      C      OR      C_DAT      ;ENABLE DATA REG
0266      D3 03      C      OUT     (SYS_CNB),A      ;CONTROL B
C
0268      3E 07      C      LD      A, DATB_INIT      ;INITIALIZE -- SINGLE DENSITY, DESELECT DRIVES
C          C          ;SPKR OFF
026A      D3 02      C      OUT     (SYS_DTB),A
C
026C      DB 03      C      IN      A, (SYS_CNB)      ;
026E      E6 FB      C      AND     C_DIR      ;ENABLE DIRECTION REG
0270      D3 03      C      OUT     (SYS_CNB),A      ;CONTROL B
C
0272      3E 3F      C      LD      A, DIRB_INIT      ;INITIALIZE DIRECTION -- RI AND DSR INPUTS, OTHERS OUTPUTS
0274      D3 02      C      OUT     (SYS_DTB),A
C
0276      DB 03      C      IN      A, (SYS_CNB)      ;
0278      F6 04      C      OR      C_DAT      ;ENABLE DATA REG
027A      F6 30      C      OR      C_60_HZ_0      ;SET 60 HZ
027C      E6 F7      C      AND     C_60_HZ_A      ;
027E      F6 02      C      OR      C_RTC      ;SET REAL TIME CLOCK INTERRUPT

```

0280	D3 03	C	OUT	(SYS_CNB),A	;CONTROL B
		C	PAGE		

'BOOT AND INITIALIZATION ROUTINES'

```
C
C
C      ;INITIALIZE A CHANNEL OF PARALLEL PIA
C
0282  DB 11      C      IN      A, (PAR_CNA)      ;
0284  F6 02      C      OR      C_PAR      ;SET PARALLEL INTERRUPT
0286  D3 11      C      OUT      (PAR_CNA),A      ;
0288  0E 01      C      LD      C,1
028A  CD 07AF'   C      CALL     IE.CO
C
C      ;INITIALIZE B CHANNEL OF PARALLEL PIA
C
028D  DB 13      C      IN      A, (PAR_CNB)      ;
028F  F6 02      C      OR      C_FDC      ;SET FDC INTERRUPT
0291  D3 13      C      OUT      (PAR_CNB),A      ;
C
0293  C9         C      RET
C
C      PAGE
```

```

                                C
0294                          C   INIT_POLL:
                                C   ;   INITIALIZE BANK AND ADDRESS OF INTERRUPT POLLING ROUTINE
                                C   ;
                                C   ; ENTRY:
                                C   ;   NONE.
                                C   ; EXIT:
                                C   ;   NONE.
                                C   ;
0294   3E 80                  C       LD   A,ROM_BANK      ;INTERRUPT POLL RTN IS IN ROM
0296   32 FD5F               C       LD   (I_POL_BANK),A
                                C
0299   21 0EBD'              C       LD   HL,INT_POLL     ;SAVE ADDRESS
029C   22 FD60               C       LD   (I_POL_ADR),HL
029F   C9                   C       RET
                                C       PAGE

```

'BOOT AND INITIALIZATION ROUTINES'

```

C
02A0      C      KB_INIT:
C      ;      INITIALIZES ALL KEYBOARD TABLE POINTERS TO POINT TO THE ALPHA
C      ;      LOCK TABLE. ALL KEYSTROKES WILL BE TRANSLATED TO THEIR ALPHA LOCK
C      ;      EQUIVALENTS REGARDLESS OF THE STATE OF THE SHIFT, CTRL AND ALPHA LOCK
C      ;      KEYS.
C      ;
C      ;      ALSO INITIALIZES KEYBOARD SCAN MASKS AND LOCATIONS OF SHIFT-TYPE
C      ;      KEYS.
C      ;
C      ;      ALL OTHER KEY DECODE TABLES HAVE BEEN REMOVED FROM THE ROM FOR SPACE
C      ;      CONSIDERATIONS.
C      ;
C      ;      THE COLD BOOT ROUTINE OF ANY OS WILL HAVE TO INITIALIZE THE TABLES AND POINTERS
C      ;      TO THEM IN ROM'S RAM.
C      ;
C
02A0      21 04CA      C      LD      HL,R_K_IN_TB      ;COPY KEY MASK INTO RAM
02A3      11 200A      C      LD      DE,K_MSK_TBL
02A6      01 000C      C      LD      BC,K_TB_SZ
02A9      ED B0        C      LDIR
C
02AB      06 05        C      LD      B,5              ;COPY KEYBOARD TABLE
C
02AD      11 2015      C      LD      DE,KEYS          ;TO
02B0      C5          C      XJ003: PUSH BC            ;SAVE
02B1      01 0040      C      LD      BC,AL_T_SZ        ;ALPHA LOCK TABLE SIZE
02B4      21 04D6      C      LD      HL,AL_TB          ;(FROM) INITIALIZE KEYBOARD TABLE
02B7      ED B0        C      LDIR
C
02B9      C1          C      POP      BC
02BA      10 F4        C      DJNZ    XJ003
02BC      C9          C      RET
C
C      PAGE

```

```

02BD      C
          C      CONINIT:
          C      ;INITIALIZE ALL NON-0 VALUES FOR CONSOLE
          C      ; (THIS ROUTINE ASSUMES 0 VALUES HAVE ALREADY BEEN INITIALIZED)
          C      ;ENTRY
          C      ;NONE
          C
          C      ;EXIT
          C      ;CONSOLE VARIABLES INITIALIZED
          C
          C      ;INITIALIZE POINTERS TO CONSOLE TABLE
02BD      21 235B      C      LD      HL,CONTBLS      ;POINT TO CONSOLE TABLES
02C0      22 2359      C      LD      (ROMCON),HL      ; SYSTEM POINTER
          C
          C      ;INITIALIZE POINTERS IN THE SYSTEM CONSOLE TABLE
02C3      21 05DF      C      LD      HL,CTBLPTRS      ;MOVE TABLE POINTERS FROM ROM
02C6      11 235B      C      LD      DE,CONTBLS      ;TO RAM
02C9      01 001E      C      LD      BC,LENCTBL
02CC      ED B0        C      LDIR
          C
          C      ;INITIALIZE OTHER MEMORY LOCATIONS
02CE      21 C000      C      LD      HL,FWAVM
02D1      22 FD53      C      LD      (CURPOS),HL      ;POSITION CURSOR TO TOP OF SCREEN
          C
          C      LD      (FWAWIND),HL      ;INITIALIZE WINDOW TO FULL SCREEN
02D4      22 237B      C      LD      A,VMCOLS
02D7      3E 50        C      LD      (WINDCOLS),A      ;FULL 80 COLUMNS
02D9      32 237E      C      LD      A,VMROWS
02DC      3E 18        C      LD      (WINDROWS),A      ;FULL 24 ROWS
02DE      32 237D      C
          C
02E1      3E 20        C      LD      A,20H
02E3      32 2379      C      LD      (OFFSET),A      ;X-Y POSITIONING OFFSET = 20H
          C
          C      ;INITIALIZE ALL WINDOW DEFINITIONS TO FULL SCREEN
02E6      06 10        C      LD      B,16      ;NUMBER OF WINDOWS
02E8      11 2387      C      LD      DE,WINDDEFS      ;DESTINATION
          C
          C      XJ004: PUSH BC      ;INITIALIZE A WINDOW
02EB      C5          C      LD      HL,FWAWIND
02EC      21 237B      C      LD      BC,4
02EF      01 0004      C      LDIR
02F2      ED B0        C      POP BC
02F4      C1          C      DJNZ XJ004      ;IF MORE WINDOWS
02F5      10 F4        C
          C
02F7      C9          C      RET
          C      PAGE

```

C
C
C PAGE

```

C
C
02F8      C      SETCHAR:
C      ;FILL CHARACTER FONT RAM WITH CHARACTER SET
C      ;ENTRY
C      ;DE = FONT RAM ADDRESS
C
C
02F8      21 0516  C      LD      HL, FONT_TABLE  ;POINT AT FONT TABLE IN ROM
C
02F8      3E 5B    C      LD      A, FONT_ROWS   ;A = # ROWS TO TRANSFER
C
02FD      01 000A  C      XJ005: LD      BC, ROW_LEN   ;BC = # BYTES / ROW
0300      ED B0    C      LDIR                     ;TRANSFER A ROW
0302      01 0006  C      LD      BC, 6           ;BUMP PTR TO FONT RAM TO START OF NEXT
0305      EB       C      EX      DE, HL          ;RAM ROW
0306      09       C      ADD     HL, BC          ;
0307      EB       C      EX      DE, HL
0308      3D       C      DEC     A              ;IS THIS THE LAST ROW?
0309      C2 02FD  C      JP      NZ, XJ005       ;NO, CONTINUE...
C
C      ;YES...
030C      C9       C      RET                     ;RETURN
C
C      PAGE

```


'BOOT AND INITIALIZATION ROUTINES'

```
C
C
0300      C      SETJMP:
C          ;INITIALIZE HIGH RAM JUMP VECTORS
C          ; (VECTORS TO BANKJMP, BANKCALL, BANKMOVE, DMA READ, AND DMA WRITE)
C          ;ENTRY
C          ;NONE
C
C          ;EXIT
C          ;NONE
C
0300      21 0319      C          LD      HL,STJMP          ;START OF ROM COPY OF JUMP VECTORS
0310      11 FD40      C          LD      DE,COMJMP        ;DESTINATION IN HIGH RAM
0313      01 000A      C          LD      BC,LENJMP        ;LENGTH OF TABLE
0316      ED B0        C          LDIR
0318      C9          C          RET
C
C
0319      C      STJMP: ;HIGH RAM JUMP VECTORS
0319      FF80      C          DW          BANKJMP
031B      FF73      C          DW          BANKCALL
031D      FF9B      C          DW          BANKMOVE
031F      FF39      C          DW          DMARD
0321      FF56      C          DW          DMAWRT
C
000A      C      LENJMP EQU      $-STJMP
C
C          PAGE
```

'BOOT AND INITIALIZATION ROUTINES'

```

C
C
C      ; *****
C      ; *
C      ; *      MISCELLANEOUS ROUTINES
C      ; *
C      ; *****
C
0323  C      OSTR:
C      ;OUTPUT STRING TO CONSOLE
C      ;NOTE: OSTR RECOGNIZES 7F AS AN ESCAPE SEQUENCE TO REPEAT CHAR N TIMES.
C      ;      FORMAT IS: 7F, REPEAT COUNT, CHAR
C      ;ENTRY
C      ;DE      =      FWA OF SOURCE
C
C      ;EXIT
C      ;NONE
C
0323  1A      C      LD      A,(DE)
0324  B7      C      OR      A
0325  F5      C      PUSH   AF
C
0326  E6 7F   C      AND     07FH
0328  FE 7F   C      CP      07FH
032A  4F      C      LD      C,A
032B  20 10   C      JR      NZ,XJ192      ; IF NOT REPEAT
C
032D  13      C      INC     DE
032E  1A      C      LD      A,(DE)
032F  3D      C      DEC     A
0330  47      C      LD      B,A      ;REPEAT COUNT
0331  13      C      INC     DE
0332  1A      C      LD      A,(DE)   ;GET REPEAT CHAR
0333  4F      C      LD      C,A
0334  D5      C      PUSH   DE      ;SAVE DE
0335          C      XJ193:
0335  C5      C      PUSH   BC
0336  CD 0064' C      CALL   COUT      ;OUTPUT CHAR
0339  C1      C      POP     BC
033A  10 F9   C      DJNZ   XJ193     ;IF NOT DONE
033C  D1      C      POP     DE
C
033D          C      XJ192:
033D  D5      C      PUSH   DE
033E  CD 0064' C      CALL   COUT      ;OUTPUT IT
0341  D1      C      POP     DE
0342  13      C      INC     DE
C
0343  F1      C      POP     AF
0344  F2 0323 C      JP      P,OSTR      ;IF NOT DONE
C
0347  D3 1C   C      OUT     (1CH),A   ;ENABLE VIDEO OUTPUT
C
0349  C9      C      RET

```

'EXECUTIVE ROM MONITOR'

MACRO-80 3.44 09-Dec-81

PAGE 1-65

'BOOT AND INITIALIZATION ROUTINES'

C

C

PAGE

```

C
C
C      ;      *****
C      ;      *
C      ;      *          SYSTEM PIA INITIALIZATION EQUATES      *
C      ;      *
C      ;      *****
C
C      ;Masks:
C
0004   C      C_DAT   EQU      00000100B      ;ORED WITH SYS PIA CONTROL TO ENABLE DATA REGISTER
00FB   C      C_DIR   EQU      11111011B      ;ANDED WITH SYS PIA CONTROL TO ENABLE DIRECTION REG
C
0030   C      C_60_HZ_D   EQU      00110000B      ;ORED WITH SYS PIA CONTROL B TO SET 60 HZ
00F7   C      C_60_HZ_A   EQU      11110111B      ;ANDED WITH SYS PIA CONTROL B TO SET 60 HZ
C
00CF   C      C_I_KBD EQU      11001111B      ;ANDED WITH SYS PIA CONTROL A TO SET INTELLIGENT
C                                     ;KEYBOARD TO INTERRUPT ON A HIGH TO LOW
C                                     ;TRANSITION OF CA2
C
00FD   C      C_DMA   EQU      11111101B      ;ANDED WITH SYS PIA CONTROL A TO SET DMA TO
C                                     ;INTERRUPT ON A HIGH TO LOW TRANSITION
C                                     ;OF CA1
C
0002   C      C_RTC   EQU      00000010B      ;ORED WITH SYS PIA CONTROL B TO SET RTC TO
C                                     ;INTERRUPT ON A LOW TO HIGH TRANSITION
C                                     ;OF CB1
C
0002   C      C_PAR   EQU      00000010B      ;ORED WITH PARALLEL PIA CONTROL A TO SET
C                                     ;PARALLEL PORT TO INTERRUPT ON A LOW TO HIGH
C                                     ;TRANSITION OF CA1 (AS PER IE.CO ROUTINE).
C
0002   C      C_FDC   EQU      00000010B      ;ORED WITH PARALLEL PIA CONTROL B TO SET FDC
C                                     ;TO INTERRUPT ON A LOW TO HIGH TRANSITION OF
C                                     ;CB1
C
C                                     PAGE

```

'BOOT AND INITIALIZATION ROUTINES'

```

C
C
C      ; *****
C      ; *
C      ; *          BOOT MESSAGES          *
C      ; *
C      ; *****
C
034A      C      LD_MSG:
034A      1B 52      C      DB      ESC,VDELL      ;DELETE CURRENT LINE
034C      7F 20 20      C      DB      7FH, 32, 20H      ;21 SPACES
034F      4C 4F 41 44      C      DB      'LOADING SYSTEM'
0353      49 4E 47 20      C
0357      53 59 53 54      C
035B      45 4D      C
035D      AE      C      DC      ', '      ;SET HI BIT ON LAST CHAR
C
035E      C      EB_MSG:
035E      1B 52      C      DB      ESC,VDELL      ;DELETE CURRENT LINE
0360      7F 15 20      C      DB      7FH, 21, 20H      ;21 SPACES
0363      42 4F 4F 54      C      DB      'BOOT ERROR'
0367      20 45 52 52      C
036B      4F 52      C
036D      AE      C      DC      ', '      ;SET HI BIT ON LAST CHAR
C
036E      C      BD_MSG:
036E      1B 52      C      DB      ESC,VDELL      ;DELETE CURRENT LINE
0370      7F 0A 20      C      DB      7FH, 10, 20H      ;21 SPACES
0373      44 49 53 4B      C      DB      'DISK IS NOT A VALID SYSTEM DISK'
0377      20 49 53 20      C
037B      4E 4F 54 20      C
037F      41 20 56 41      C
0383      4C 49 44 20      C
0387      53 59 53 54      C
038B      45 4D 20 44      C
038F      49 53 4B      C
0392      AE      C      DC      ', '      ;SET HI BIT ON LAST CHAR
C
0393      C      WT_CR_MSG:
0393      20 50 52 45      C      DB      ' PRESS RETURN TO TRY AGAIN'
0397      53 53 20 52      C
039B      45 54 55 52      C
039F      4E 20 54 4F      C
03A3      20 54 52 59      C
03A7      20 41 47 41      C
03AB      49 4E      C
03AD      AE      C      DC      ', '
C
C      PAGE

```

'BOOT AND INITIALIZATION ROUTINES'

```

C
C ; *****
C ; *
C ; * LOG ON MESSAGE *
C ; *
C ; *****
C
03AE 1B 2E 30 C MSG: DB ESC, '0' ;DISABLE CURSOR
03B1 1A 0A 0A C DB 'Z'-40H, LF, LF
03B4 7F 15 20 C DB 07FH, 21, 20H
03B7 1B 67 C DB ESC, VSGH
03B9 11 C DB 'Q'-40H
03BA 7F 25 17 C DB 07FH, 37, 'W'-40H
03BD 05 C DB 'E'-40H
03BE 1B 47 C DB ESC, VEGH
C
03C0 1F C DB CRLF
03C1 7F 15 20 C DB 07FH, 21, 20H
03C4 1B 67 01 1B C DB ESC, VSGH, 1, ESC, VEGH
03C8 47 C
03C9 7F 0E 20 C DB 07FH, 14, 20H
03CC 45 58 45 43 C DB 'EXECUTIVE'
03D0 55 54 49 56 C
03D4 45 C
03D5 7F 0E 20 C DB 07FH, 14, 20H
03D8 1B 67 04 1B C DB ESC, VSGH, 4, ESC, VEGH
03DC 47 C
C
03DD 1F C DB CRLF
03DE 7F 15 20 C DB 07FH, 21, 20H
03E1 1B 67 C DB ESC, VSGH
03E3 01 C DB 'A'-40H
03E4 1B 47 C DB ESC, VEGH
03E6 7F 10 20 C DB 7FH, 16, 20H
03E9 1B 29 C DB ESC, VSHI
03EB 56 31 2E 32 C DB 'V1.21'
03EF 31 C
03F0 1B 28 C DB ESC, VEHI
03F2 7F 10 20 C DB 7FH, 16, 20H
03F5 1B 67 C DB ESC, VSGH
03F7 04 C DB 'D'-40H
03F8 1B 47 C DB ESC, VEGH
C
03FA 1F C DB CRLF
03FB 7F 15 20 C DB 07FH, 21, 20H
03FE 1B 67 C DB ESC, VSGH
0400 01 C DB 'A'-40H
0401 7F 25 20 C DB 07FH, 37, 20H
0404 04 C DB 'D'-40H
0405 1B 47 C DB ESC, VEGH
C
0407 1F C DB CRLF
0408 7F 15 20 C DB 07FH, 21, 20H
040B 1B 67 C DB ESC, VSGH
040D 01 C DB 'A'-40H

```

'BOOT AND INITIALIZATION ROUTINES'

040E	1B 47	C	DB	ESC, VEGH
0410	20	C	DB	20H
0411	1B 67	C	DB	ESC, VSGH
0413	13	C	DB	'S'-40H
0414	1B 47	C	DB	ESC, VEGH
0416	20 31 39 38	C	DB	' 1984 OSBORNE COMPUTER CORPORATION '
041A	34 20 4F 53	C		
041E	42 4F 52 4E	C		
0422	45 20 43 4F	C		
0426	4D 50 55 54	C		
042A	45 52 20 43	C		
042E	4F 52 50 4F	C		
0432	52 41 54 49	C		
0436	4F 4E 20	C		
0439	1B 67	C	DB	ESC, VSGH
043B	04	C	DB	'D'-40H
043C	1B 47	C	DB	ESC, VEGH
		C		
043E	1F	C	DB	CRLF
043F	7F 15 20	C	DB	07FH, 21, 20H
0442	1B 67	C	DB	ESC, VSGH
0444	01	C	DB	'A'-40H
0445	1B 47	C	DB	ESC, VEGH
0447	7F 0A 20	C	DB	7FH, 10, 20H
044A	32 36 35 33	C	DB	'26538 DANTI COURT'
044E	38 20 44 41	C		
0452	4E 54 49 20	C		
0456	43 4F 55 52	C		
045A	54	C		
045B	7F 0A 20	C	DB	7FH, 10, 20H
045E	1B 67	C	DB	ESC, VSGH
0460	04	C	DB	'D'-40H
0461	1B 47	C	DB	ESC, VEGH
		C		
0463	1F	C	DB	CRLF
0464	7F 15 20	C	DB	07FH, 21, 20H
0467	1B 67	C	DB	ESC, VSGH
0469	01	C	DB	'A'-40H
046A	1B 47	C	DB	ESC, VEGH
046C	7F 0A 20	C	DB	7FH, 10, 20H
046F	48 41 59 57	C	DB	'HAYWARD, CA 94545'
0473	41 52 44 2C	C		
0477	20 43 41 20	C		
047B	39 34 35 34	C		
047F	35	C		
0480	7F 0A 20	C	DB	7FH, 10, 20H
0483	1B 67	C	DB	ESC, VSGH
0485	04	C	DB	'D'-40H
0486	1B 47	C	DB	ESC, VEGH
		C		
0488	1F	C	DB	CRLF
0489	7F 15 20	C	DB	07FH, 21, 20H
048C	1B 67	C	DB	ESC, VSGH
048E	1A	C	DB	'Z'-40H
048F	7F 25 18	C	DB	07FH, 37, 'X'-40H
0492	03	C	DB	'C'-40H

'BOOT AND INITIALIZATION ROUTINES'

0493	1B 47	C	DB	ESC,VEGH	
		C			
0495	1F 0A 0A	C	DB	CRLF,LF,LF	
0498	7F 15 20	C	DB	07FH, 21, 20H	
049B	1B 2E 32	C	DB	ESC,'.2'	;ENABLE BLOCK CURSOR
049E	49 4E 53 45	C	DB	'INSERT DISK IN DRIVE '	
04A2	52 54 20 44	C			
04A6	49 53 4B 20	C			
04AA	49 4E 20 44	C			
04AE	52 49 56 45	C			
04B2	20	C			
04B3	1B 6C	C	DB	ESC,VSUL	
04B5	41	C	DB	'A'	
04B6	1B 6D	C	DB	ESC,VEUL	
04B8	20 41 4E 44	C	DB	' AND PRESS RETURN'	
04BC	20 50 52 45	C			
04C0	53 53 20 52	C			
04C4	45 54 55 52	C			
04C8	4E	C			
04C9	AE	C	DC	'.'	
		C			
		C	PAGE		

'BOOT AND INITIALIZATION ROUTINES'

```

C
C
C      ; *****
C      ; *
C      ; *          INITIALIZATION VALUES          *
C      ; *
C      ; *****
C
C      ; KEYBOARD TABLE COPIED INTO ROM'S RAM AT INIT TIME:
04CA  R_K_IN_TB:
C      ;
C      ; MASKS FOR REMOVING INVALID KEYS FROM SCAN:
C      ; (INVALID KEYS ARE SHIFT, CTRL, ALPHA LOCK AND THE DISABLED RETURN
C      ; KEY.)
C      ;
04CA  E3  C      DB      11100011B      ;MASK FOR ROW 0: STRIP OFF CTRL, ALPHA AND SHIFT
04CB  FF  C      DB      11111111B      ;FOR ROWS 1-6, STRIP OFF NOTHING
04CC  FF  C      DB      11111111B
04CD  FF  C      DB      11111111B
04CE  FF  C      DB      11111111B
04CF  FF  C      DB      11111111B
04D0  FF  C      DB      11111111B
04D1  00  C      DB      00000000B      ;FOR ROW 7, STRIP OFF ALL CHARS
C      ;
C      ; FOLLOWING ARE THE KEYBOARD MATRIX LOCATIONS FOR CTRL, ALPHA AND SHIFT:
C      ;
04D2  02  C      DB      0000100B      ;CTRL: ROW 0, COL 2
04D3  3B  C      DB      111011B      ;ALPHA: ROW 7, COL 3
04D4  04  C      DB      000100B      ;SHIFT: ROW 0, COL 4
C      ;
C      ; KEY CLICK INITIALIZATION VALUE:
C      ;
04D5  00  C      DB      0
000C  K_TB_SZ EQU  $ - R_K_IN_TB
C
C      PAGE

```

'BOOT AND INITIALIZATION ROUTINES'

```

C
C
C
C      ;      *****
C      ;      *
C      ;      *          ALPHA LOCK TABLE          *
C      ;      *
C      ;      *****
C
04D6      C      AL_TB: ;ALPHA LOCK TABLE
C
04D6      1B 09 FF FF      C      DB      ESC, TAB, BAD, BAD, BAD, CR, ' ', ' '
04DA      FF 0D 27 5B      C
C
04DE      31 32 33 34      C      DB      '1', '2', '3', '4', '5', '6', '7', '8'
04E2      35 36 37 38      C
C
04E6      51 57 45 52      C      DB      'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I'
04EA      54 59 55 49      C
C
04EE      41 53 44 46      C      DB      'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K'
04F2      47 48 4A 4B      C
C
04F6      5A 58 43 56      C      DB      'Z', 'X', 'C', 'V', 'B', 'N', 'M', ' ', ' '
04FA      42 4E 4D 2C      C
C
04FE      8A 8D 30 20      C      DB      BAH, BDH, '0', ' ', ' ', ' ', 'P', 'O', '9'
0502      2E 50 4F 39      C
C
0506      8B 8C 2D 2F      C      DB      BBH, BCH, ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
050A      3B 5C 4C 3D      C
C
050E      FF FF FF FF      C      DB      BAD, BAD, BAD, BAD, CR, BAD, BAD, BAD
0512      0D FF FF FF      C
C
0040      C      AL_T_SZ EQU      $-AL_TB
C
C      PAGE
C

```

'BOOT AND INITIALIZATION ROUTINES'

```

C
C
C      ;
C      ;
C      ;
C      ;
C      ;
C      ;
C      ;
C      ;CHARACTER FONT
C      FONT_TABLE:
0516      FF C3 C3 C3      C      DB      0FFH, 0C3H, 0C3H, 0C3H, 0C3H, 0C3H, 0C3H, 0C3H, 0FFH
051A      C3 C3 C3 C3      C
051E      C3 FF            C
0520      F0 F0 F0 F0      C      DB      0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 0F0H
0524      F0 F0 F0 F0      C
0528      F0 F0            C
052A      0F 0F 0F 0F      C      DB      00FH, 00FH, 00FH, 00FH, 00FH, 00FH, 00FH, 00FH, 00FH
052E      0F F0 F0 F0      C
0532      F0 F0            C
0534      0F 0F 0F 0F      C      DB      00FH, 00FH, 00FH, 00FH, 00FH, 0FFH, 0FFH, 0FFH, 0FFH
0538      0F FF FF FF      C
053C      FF FF            C
053E      0F 0F 0F 0F      C      DB      00FH, 00FH, 00FH, 00FH, 00FH, 00FH, 00FH, 00FH, 00FH
0542      0F 0F 0F 0F      C
0546      0F 0F            C
0548      FF FF FF FF      C      DB      0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 00FH, 00FH, 00FH, 00FH
054C      FF 0F 0F 0F      C
0550      0F 0F            C
0552      00 00 00 00      C      DB      000H, 000H, 000H, 000H, 000H, 0F0H, 0F0H, 0F0H, 0F0H
0556      00 F0 F0 F0      C
055A      F0 F0            C
055C      00 00 00 00      C      DB      000H, 000H, 000H, 000H, 000H, 00FH, 00FH, 00FH, 00FH
0560      00 0F 0F 0F      C
0564      0F 0F            C
0566      80 C0 E0 F0      C      DB      080H, 0C0H, 0E0H, 0F0H, 0F0H, 0F8H, 0F8H, 0FEH, 0FFH
056A      F0 F8 F8 FC      C
056E      FE FF            C
0570      00 00 00 3F      C      DB      000H, 000H, 000H, 03FH, 03FH, 03FH, 03FH, 03CH, 03CH, 03CH
0574      3F 3F 3F 3C      C
0578      3C 3C            C
057A      01 03 07 0F      C      DB      001H, 003H, 007H, 00FH, 00FH, 01FH, 01FH, 03FH, 07FH, 0FFH
057E      0F 1F 1F 3F      C
0582      7F FF            C
0584      3C 3C 3C 3C      C      DB      03CH, 03CH, 03CH, 03CH, 03CH, 03CH, 03CH, 03CH, 03CH, 03CH
0588      3C 3C 3C 3C      C
058C      3C 3C            C
058E      3C 3C 3C FF      C      DB      03CH, 03CH, 03CH, 0FFH, 0FFH, 0FFH, 0FFH, 03CH, 03CH, 03CH
0592      FF FF FF 3C      C
0596      3C 3C            C
0598      00 00 3C 7E      C      DB      000H, 000H, 03CH, 07EH, 07EH, 07EH, 07EH, 03CH, 000H, 000H
059C      7E 7E 7E 3C      C
05A0      00 00            C
05A2      F0 F0 F0 F0      C      DB      0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 00FH, 00FH, 00FH, 00FH, 00FH
05A6      F0 0F 0F 0F      C
05AA      0F 0F            C

```

'BOOT AND INITIALIZATION ROUTINES'

05AC	00 00 00 FF	C	DB	000H, 000H, 000H, 0FFH, 0FFH, 0FFH, 0FFH, 03CH, 03CH, 03CH
05B0	FF FF FF 3C	C		
05B4	3C 3C	C		
05B6	00 00 00 FC	C	DB	000H, 000H, 000H, 0FCH, 0FCH, 0FCH, 0FCH, 03CH, 03CH, 03CH
05BA	FC FC FC 3C	C		
05BE	3C 3C	C		
05C0	FF FF FF FF	C	DB	0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0F0H, 0F0H, 0F0H, 0F0H, 0F0H
05C4	FF F0 F0 F0	C		
05C8	F0 F0	C		
05CA	F0 F0 F0 F0	C	DB	0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 000H, 000H, 000H, 000H, 000H
05CE	F0 00 00 00	C		
05D2	00 00	C		
05D4	00 38 44 9A	C	DB	000H, 038H, 044H, 09AH, 0A2H, 0A2H, 09AH, 044H, 038H, 000H
05D8	A2 A2 9A 44	C		
05DC	38 00	C		
05DE	0F 0F 0F 0F	C	DB	00FH, 00FH, 00FH, 00FH, 00FH, 000H, 000H, 000H, 000H, 000H
05E2	0F 00 00 00	C		
05E6	00 00	C		
05E8	FF 7F 3F 1F	C	DB	0FFH, 07FH, 03FH, 01FH, 01FH, 00FH, 00FH, 007H, 003H, 001H
05EC	1F 0F 0F 07	C		
05F0	03 01	C		
05F2	FF FF FF FF	C	DB	0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH
05F6	FF FF FF FF	C		
05FA	FF FF	C		
05FC	FF FF FF FF	C	DB	0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 000H, 000H, 000H, 000H, 000H
0600	FF 00 00 00	C		
0604	00 00	C		
0606	00 00 00 00	C	DB	000H, 000H, 000H, 000H, 000H, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH
060A	00 FF FF FF	C		
060E	FF FF	C		
0610	FF FE FC F8	C	DB	0FFH, 0FEH, 0FCH, 0FBH, 0FBH, 0F0H, 0F0H, 0E0H, 0C0H, 0B0H
0614	F8 F0 F0 E0	C		
0618	C0 80	C		
061A	F0 F0 F0 F0	C	DB	0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH
061E	F0 FF FF FF	C		
0622	FF FF	C		
0624	00 00 00 FF	C	DB	000H, 000H, 000H, 0FFH, 0FFH, 0FFH, 0FFH, 000H, 000H, 000H
0628	FF FF FF 00	C		
062C	00 00	C		
062E	3C 3C 3C 3F	C	DB	03CH, 03CH, 03CH, 03FH, 03FH, 03FH, 03FH, 000H, 000H, 000H
0632	3F 3F 3F 00	C		
0636	00 00	C		
0638	C0 C6 CC D8	C	DB	0C0H, 0C6H, 0CCH, 0D8H, 030H, 06EH, 0C3H, 006H, 018H, 03FH
063C	30 6E C3 06	C		
0640	18 3F	C		
0642	3C 3C 3C FF	C	DB	03CH, 03CH, 03CH, 0FFH, 0FFH, 0FFH, 0FFH, 000H, 000H, 000H
0646	FF FF FF 00	C		
064A	00 00	C		
064C	3C 3C 3C FC	C	DB	03CH, 03CH, 03CH, 0FCH, 0FCH, 0FCH, 0FCH, 000H, 000H, 000H
0650	FC FC FC 00	C		
0654	00 00	C		
0656	00 00 00 00	C	DB	000H, 000H, 000H, 000H, 000H, 000H, 000H, 000H, 000H, 000H
065A	00 00 00 00	C		
065E	00 00	C		
0660	00 18 3C 3C	C	DB	000H, 018H, 03CH, 03CH, 018H, 018H, 000H, 018H, 000H, 000H
0664	18 18 00 18	C		

'BOOT AND INITIALIZATION ROUTINES'

0668	00 00	C		
066A	00 66 66 00	C	DB	000H, 066H, 066H, 000H, 000H, 000H, 000H, 000H, 000H
066E	00 00 00 00	C		
0672	00 00	C		
0674	00 24 24 7E	C	DB	000H, 024H, 024H, 07EH, 024H, 07EH, 024H, 024H, 000H, 000H
0678	24 7E 24 24	C		
067C	00 00	C		
067E	00 18 3E 58	C	DB	000H, 018H, 03EH, 058H, 038H, 01AH, 07CH, 018H, 000H, 000H
0682	38 1A 7C 18	C		
0686	00 00	C		
0688	00 26 76 2C	C	DB	000H, 026H, 076H, 02CH, 018H, 034H, 06EH, 064H, 000H, 000H
068C	18 34 6E 64	C		
0690	00 00	C		
0692	00 38 6C 6C	C	DB	000H, 038H, 06CH, 06CH, 038H, 06DH, 066H, 03FH, 003H, 000H
0696	38 6D 66 3F	C		
069A	03 00	C		
069C	00 06 06 0C	C	DB	000H, 006H, 006H, 00CH, 000H, 000H, 000H, 000H, 000H, 000H
06A0	00 00 00 00	C		
06A4	00 00	C		
06A6	00 0C 18 30	C	DB	000H, 00CH, 018H, 030H, 030H, 030H, 018H, 00CH, 000H, 000H
06AA	30 30 18 0C	C		
06AE	00 00	C		
06B0	00 30 18 0C	C	DB	000H, 030H, 018H, 00CH, 00CH, 00CH, 018H, 030H, 000H, 000H
06B4	0C 0C 18 30	C		
06B8	00 00	C		
06BA	00 10 54 38	C	DB	000H, 010H, 054H, 038H, 07CH, 038H, 054H, 010H, 000H, 000H
06BE	7C 38 54 10	C		
06C2	00 00	C		
06C4	00 00 00 18	C	DB	000H, 000H, 000H, 018H, 018H, 07EH, 018H, 018H, 000H, 000H
06C8	18 7E 18 18	C		
06CC	00 00	C		
06CE	00 00 00 00	C	DB	000H, 000H, 000H, 000H, 000H, 000H, 01CH, 01CH, 00CH, 018H
06D2	00 00 1C 1C	C		
06D6	0C 18	C		
06D8	00 00 00 00	C	DB	000H, 000H, 000H, 000H, 07EH, 07EH, 000H, 000H, 000H, 000H
06DC	7E 7E 00 00	C		
06E0	00 00	C		
06E2	00 00 00 00	C	DB	000H, 000H, 000H, 000H, 000H, 000H, 018H, 018H, 000H, 000H
06E6	00 00 18 18	C		
06EA	00 00	C		
06EC	00 02 06 0C	C	DB	000H, 002H, 006H, 00CH, 018H, 030H, 060H, 040H, 000H, 000H
06F0	18 30 60 40	C		
06F4	00 00	C		
06F6	00 3C 66 66	C	DB	000H, 03CH, 066H, 066H, 06EH, 076H, 066H, 03CH, 000H, 000H
06FA	6E 76 66 3C	C		
06FE	00 00	C		
0700	00 18 38 18	C	DB	000H, 018H, 038H, 018H, 018H, 018H, 03CH, 000H, 000H
0704	18 18 18 3C	C		
0708	00 00	C		
070A	00 3C 66 0C	C	DB	000H, 03CH, 066H, 00CH, 038H, 060H, 060H, 07EH, 000H, 000H
070E	38 60 60 7E	C		
0712	00 00	C		
0714	00 3C 66 06	C	DB	000H, 03CH, 066H, 006H, 01CH, 006H, 066H, 03CH, 000H, 000H
0718	1C 06 66 3C	C		
071C	00 00	C		
071E	00 0C 6C 6C	C	DB	000H, 00CH, 06CH, 06CH, 06CH, 07EH, 00CH, 00CH, 000H, 000H

'BOOT AND INITIALIZATION ROUTINES'

0722	6C 7E 0C 0C	C		
0726	00 00	C		
0728	00 7E 60 60	C	DB	000H, 07EH, 060H, 060H, 07CH, 006H, 066H, 03CH, 000H, 000H
072C	7C 06 66 3C	C		
0730	00 00	C		
0732	00 3C 66 60	C	DB	000H, 03CH, 066H, 060H, 07CH, 066H, 066H, 03CH, 000H, 000H
0736	7C 66 66 3C	C		
073A	00 00	C		
073C	00 7E 06 0C	C	DB	000H, 07EH, 006H, 00CH, 018H, 030H, 030H, 030H, 000H, 000H
0740	18 30 30 30	C		
0744	00 00	C		
0746	00 3C 66 66	C	DB	000H, 03CH, 066H, 066H, 03CH, 066H, 066H, 03CH, 000H, 000H
074A	3C 66 66 3C	C		
074E	00 00	C		
0750	00 3C 66 66	C	DB	000H, 03CH, 066H, 066H, 03EH, 006H, 066H, 03CH, 000H, 000H
0754	3E 06 66 3C	C		
0758	00 00	C		
075A	00 00 00 1C	C	DB	000H, 000H, 000H, 01CH, 01CH, 000H, 01CH, 01CH, 000H, 000H
075E	1C 00 1C 1C	C		
0762	00 00	C		
0764	00 00 00 38	C	DB	000H, 000H, 000H, 038H, 038H, 000H, 038H, 038H, 018H, 030H
0768	38 00 38 38	C		
076C	18 30	C		
076E	00 0C 18 30	C	DB	000H, 00CH, 018H, 030H, 060H, 030H, 018H, 00CH, 000H, 000H
0772	60 30 18 0C	C		
0776	00 00	C		
0778	00 00 00 00	C	DB	000H, 000H, 000H, 000H, 07EH, 000H, 07EH, 000H, 000H, 000H
077C	7E 00 7E 00	C		
0780	00 00	C		
0782	00 30 18 0C	C	DB	000H, 030H, 018H, 00CH, 006H, 00CH, 018H, 030H, 000H, 000H
0786	06 0C 18 30	C		
078A	00 00	C		
078C	00 3C 66 06	C	DB	000H, 03CH, 066H, 006H, 00CH, 018H, 000H, 018H, 000H, 000H
0790	0C 18 00 18	C		
0794	00 00	C		
0796	00 3C 42 4E	C	DB	000H, 03CH, 042H, 04EH, 056H, 05CH, 040H, 03CH, 000H, 000H
079A	56 5C 40 3C	C		
079E	00 00	C		
07A0	00 3C 66 66	C	DB	000H, 03CH, 066H, 066H, 07EH, 066H, 066H, 066H, 000H, 000H
07A4	7E 66 66 66	C		
07A8	00 00	C		
07AA	00 7C 66 66	C	DB	000H, 07CH, 066H, 066H, 07CH, 066H, 066H, 07CH, 000H, 000H
07AE	7C 66 66 7C	C		
07B2	00 00	C		
07B4	00 3C 66 60	C	DB	000H, 03CH, 066H, 060H, 060H, 060H, 066H, 03CH, 000H, 000H
07B8	60 60 66 3C	C		
07BC	00 00	C		
07BE	00 7C 66 66	C	DB	000H, 07CH, 066H, 066H, 066H, 066H, 066H, 07CH, 000H, 000H
07C2	66 66 66 7C	C		
07C6	00 00	C		
07C8	00 7E 60 60	C	DB	000H, 07EH, 060H, 060H, 078H, 060H, 060H, 07EH, 000H, 000H
07CC	78 60 60 7E	C		
07D0	00 00	C		
07D2	00 7E 60 60	C	DB	000H, 07EH, 060H, 060H, 078H, 060H, 060H, 060H, 000H, 000H
07D6	78 60 60 60	C		
07DA	00 00	C		

'BOOT AND INITIALIZATION ROUTINES'

07DC	00 3C 66 60	C	DB	000H, 03CH, 066H, 060H, 06EH, 066H, 066H, 03CH, 000H, 000H
07E0	6E 66 66 3C	C		
07E4	00 00	C		
07E6	00 66 66 66	C	DB	000H, 066H, 066H, 066H, 07EH, 066H, 066H, 066H, 000H, 000H
07EA	7E 66 66 66	C		
07EE	00 00	C		
07F0	00 3C 18 18	C	DB	000H, 03CH, 018H, 018H, 018H, 018H, 018H, 03CH, 000H, 000H
07F4	18 18 18 3C	C		
07F8	00 00	C		
07FA	00 06 06 06	C	DB	000H, 006H, 006H, 006H, 006H, 006H, 03CH, 000H, 000H
07FE	06 06 66 3C	C		
0802	00 00	C		
0804	00 66 6C 78	C	DB	000H, 066H, 06CH, 078H, 070H, 078H, 06CH, 066H, 000H, 000H
0808	70 78 6C 66	C		
080C	00 00	C		
080E	00 60 60 60	C	DB	000H, 060H, 060H, 060H, 060H, 060H, 07EH, 000H, 000H
0812	60 60 60 7E	C		
0816	00 00	C		
0818	00 63 77 7F	C	DB	000H, 063H, 077H, 07FH, 06BH, 06BH, 063H, 063H, 000H, 000H
081C	6B 6B 63 63	C		
0820	00 00	C		
0822	00 66 66 76	C	DB	000H, 066H, 066H, 076H, 07EH, 06EH, 066H, 066H, 000H, 000H
0826	7E 6E 66 66	C		
082A	00 00	C		
082C	00 3C 66 66	C	DB	000H, 03CH, 066H, 066H, 066H, 066H, 066H, 03CH, 000H, 000H
0830	66 66 66 3C	C		
0834	00 00	C		
0836	00 7C 66 66	C	DB	000H, 07CH, 066H, 066H, 07CH, 060H, 060H, 060H, 000H, 000H
083A	7C 60 60 60	C		
083E	00 00	C		
0840	00 3C 66 66	C	DB	000H, 03CH, 066H, 066H, 066H, 066H, 076H, 03CH, 00CH, 006H
0844	66 66 76 3C	C		
0848	0C 06	C		
084A	00 7C 66 66	C	DB	000H, 07CH, 066H, 066H, 07CH, 066H, 066H, 066H, 000H, 000H
084E	7C 66 66 66	C		
0852	00 00	C		
0854	00 3C 66 60	C	DB	000H, 03CH, 066H, 060H, 03CH, 006H, 066H, 03CH, 000H, 000H
0858	3C 06 66 3C	C		
085C	00 00	C		
085E	00 7E 18 18	C	DB	000H, 07EH, 018H, 018H, 018H, 018H, 018H, 018H, 000H, 000H
0862	18 18 18 18	C		
0866	00 00	C		
0868	00 66 66 66	C	DB	000H, 066H, 066H, 066H, 066H, 066H, 03CH, 000H, 000H
086C	66 66 66 3C	C		
0870	00 00	C		
0872	00 66 66 66	C	DB	000H, 066H, 066H, 066H, 066H, 03CH, 03CH, 018H, 000H, 000H
0876	66 3C 3C 18	C		
087A	00 00	C		
087C	00 63 63 6B	C	DB	000H, 063H, 063H, 06BH, 06BH, 07FH, 07FH, 036H, 000H, 000H
0880	6B 7F 7F 36	C		
0884	00 00	C		
0886	00 66 66 3C	C	DB	000H, 066H, 066H, 03CH, 018H, 03CH, 066H, 066H, 000H, 000H
088A	18 3C 66 66	C		
088E	00 00	C		
0890	00 66 66 66	C	DB	000H, 066H, 066H, 066H, 03CH, 018H, 018H, 018H, 000H, 000H
0894	3C 18 18 18	C		

'BOOT AND INITIALIZATION ROUTINES'

```

0898      00 00      C
089A      00 7E 46 0C  C          DB          000H,07EH,046H,00CH,018H,030H,062H,07EH,000H,000H
089E      18 30 62 7E  C
08A2      00 00      C
038E      C          FONTLEN EQU    $-FONT_TABLE
000A      C          ROW_LEN EQU    10
005B      C          FONT_ROWS EQU   FONTLEN / ROW_LEN
                                PAGE

```


;

SUBTTL
PAGE

'POWER ON DIAGNOSTICS'

```

C      SUBTTL      'POWER ON DIAGNOSTICS'
C      INCLUDE ROM1                      ;POWER ON DIAGNOSTICS
C      ;
C      ;      *****
C      ;      *
C      ;      *          POWER ON TEST          *
C      ;      *
C      ;      *****
C      ;
C      ;      PTEST.ASM
C      ;      FEBRUARY 22TH, 1983
C      ;      BY CHRIS MAYER
C      ;
C      ;      MODULE 1 --ROM1-- ENTRY POINTS
C      ;
C      ;      POWER ON DIAGNOSTICS
C      ;
C      ;      PODIAG:      START OF DIAGNOSTIC TEST
C      ;      MEMORY_TEST: START OF MEMORY TEST
C      ;      ROM_TEST:    START OF ROM TEST
C      ;      LSI_TEST:    START OF LSI REGISTERS TEST
C      ;      DIAG_INIT:   INITIALIZES SYSTEM FOR DIAGNOSTIC
C
C      PAGE
```

```
C
C
C      .Z80
C      ; .XLIST
C      ; INCLUDE      SYSS1.MAC      ;SYSTEST FILE
C      ; INCLUDE      RAMS10.MAC     ;RAM MEMORY LOCATIONS AND CONSTANTS
C      ; INCLUDE      RAMSMON.MAC    ;MONITOR RAM MEMORY LOCATIONS
C      ; .LIST
C      ; ;SYSS1.MAC , RAMSMON.MAC AND RAMS10.MAC INCLUDED
C
C      ; EXT      SET_INT, INITIALIZE, INIT_INT, COUT, AUTOBOOT, OSTR
C
C      ; PUBLIC  PODIAG, L_PWR_UP, DIAG_INIT
C
C      PAGE
```

```

C
C
08A4 C PODIAG:
C
08A4 3E 00 C LD A,0
08A6 F3 C PWR_UP: DI
08A7 08 C EX AF,AF' ;RESET REPEAT FLAG
C
08A8 CD 08CD C XJ007: CALL MEMORY_TEST
08A9 CD 0992 C MEM_DONE: CALL ROM_TEST
08AE CD 09CB C CALL LSI_TEST
08B1 3E FF C LD A,0FFH
08B3 08 C EX AF,AF'
08B4 FE FF C CP 0FFH ;LOOP FLAG SET ?
08B6 CA 08A8 C JP Z,XJ007
C
08B9 CD 0AD6 C CALL DIAG_INIT ;RESTORE INTERRUPTS & STUFF
08BC 0E 07 C LD C,'G' -040H
08BE CD 0064' C CALL COUT ;YEA! WE MADE IT
08C1 31 FDDC C LD SP,ROMSTK
08C4 FB C EI
08C5 C3 001F C JP AUTOBOOT ;ALL DONE
C
08C8 3E FF C L_PWR_UP: LD A,0FFH
08CA C3 08A6 C JP PWR_UP
C
C ;***** MEMORY TEST *****
C
08CD 31 8FFF C MEMORY_TEST: LD SP,0BFFFH ;STACK = TPA UNDER VIDEO MEMORY
08D0 3E C0 C LD A,0C0H ;TEST VIDEO MEMORY
08D2 D3 00 C OUT (SYS_DTA),A
08D4 21 C000 C LD HL,0C000H ;SET FIRST ADDRESS POINTER
08D7 01 1000 C LD BC,4*1024 ;SET BLOCK SIZE
08DA 3E AA C LD A,0AAH ;SET TEST PATTERN
08DC CD 0933 C CALL MEM_TEST
08DF 3E 55 C LD A,055H
08E1 CD 0933 C CALL MEM_TEST
C
08E4 08 C EX AF,AF'
08E5 F5 C PUSH AF ;SAVE LOOP FLAG
08E6 CD 0AD6 C CALL DIAG_INIT ;RESTORE CONSOLE STUFF
08E9 11 0ASC C LD DE,TEST_MESSAGE ;CLEAR JUNK FROM VIDEO TEST
08EC CD 0323 C CALL OSTR ; & SHOW TEST MESSAGE
08EF F3 C DI
08F0 F1 C POP AF
08F1 08 C EX AF,AF' ;RESTORE LOOP FLAG
C
08F2 21 2000 C LD HL,02000H ;TEST ROM'S RAM
08F5 01 0800 C LD BC,2*1024
08F8 3E AA C LD A,0AAH
08FA CD 0933 C CALL MEM_TEST
08FD 3E 55 C LD A,055H
08FF CD 0933 C CALL MEM_TEST
C
0902 31 27FF C LD SP,027FFH ;STACK = TOP OF ROM'S RAM

```

```

0905 3E 80      C      LD      A,080H      ;TEST BANK 0
0907 D3 00      C      OUT     (SYS_DTA),A
0909 21 4000     C      LD      HL,04000H
090C 01 C000     C      LD      BC,48*1024
090F 3E AA      C      LD      A,0AAH
0911 CD 0933     C      CALL    MEM_TEST
0914 3E 55      C      LD      A,055H
0916 CD 0933     C      CALL    MEM_TEST
                                C
0919 3E 81      C      LD      A,081H      ;TEST BANK 1
091B D3 00      C      OUT     (SYS_DTA),A
091D 21 4000     C      LD      HL,04000H
0920 01 B000     C      LD      BC,44*1024
0923 3E AA      C      LD      A,0AAH
0925 CD 0933     C      CALL    MEM_TEST
0928 3E 55      C      LD      A,55H
092A CD 0933     C      CALL    MEM_TEST
092D CD 094F     C      CALL    UNIQUE_ADDR
0930 C3 08AB     C      JP      MEM_DONE    ;RET IS FUNNY AFTER A MEM TEST
                                C
0933 E5          C      MEM_TEST:  PUSH   HL
0934 C5          C      PUSH   BC          ;SAVE REGISTERS FOR NEXT PATTERN
0935 E5          C      PUSH   HL
0936 C5          C      PUSH   BC          ;SAVE REGISTERS FOR CPI#
0937 5D          C      LD      E,L
0938 54          C      LD      D,H      ;DE (- HL
0939 13          C      INC      DE      ;SETTING DE UP FOR LDIR
093A 77          C      LD      (HL),A
093B 0B          C      DEC      BC      ;FIRST LOAD WAS DONE 'BY HAND'
093C ED B0       C      LDIR          ;FILL MEMORY
093E C1          C      POP      BC
093F E1          C      POP      HL      ;RESTORE REGISTERS FOR CPI#
0940 ED A1       C      MEM_CHECK: CPI
0942 FD 21 204D  C      LD      IY,' M'    ;M = MEMORY ERROR
0946 C2 0A46     C      JP      NZ,ERROR    ;HOPE IT'S THE SAME
0949 EA 0940     C      JP      PE, MEM_CHECK ;DONE YET ?
094C C1          C      POP      BC
094D E1          C      POP      HL      ;RESTORE REGISTERS FOR NEXT TEST
094E C9          C      RET
                                C
                                C
                                C
094F            C      UNIQUE_ADDR:
094F FD 21 2055  C      LD      IY,' U'    ;U = UNIQUE ADDRESS TEST ERROR
0953 3E 00       C      LD      A,0        ;DATA WRITTEN
0955 06 0F       C      LD      B,15
0957 11 0001     C      LD      DE,1
095A 21 8001     C      LD      HL,08001H    ;ADDRESS POINTER
                                C
095D 77          C      XJ008: LD      (HL),A    ;WRITE DATA
095E 3C          C      INC      A          ;CHANGE IT
095F 19          C      ADD      HL,DE      ;POINT TO NEXT ADDRESS
0960 EB          C      EX      DE,HL
0961 29          C      ADD      HL,HL
0962 EB          C      EX      DE,HL      ;DOUBLE DE
0963 10 F8       C      DJNZ    XJ008      ;15 TIMES

```

0991 C9

```

;10000000_00000001B      00
;10000000_00000010B      01
;10000000_00000100B      02
;10000000_00001000B      03
;10000000_00010000B      04
;10000000_00100000B      05
;10000000_01000000B      06
;10000000_10000000B      07
;10000001_00000000B      08
;10000010_00000000B      09
;10000100_00000000B      0A
;10001000_00000000B      0B
;10010000_00000000B      0C
;10100000_00000000B      0D
;11000000_00000000B      0E
;10000000_00000000B      0F

;01111111 11111111B      10

```

```

ROM_TEST:      LD      HL,0          ;POINTER TO BYTES TO ADD UP
               LD      IX,0          ;MID & LOW BYTES ACCUMULATOR
               LD      BC,8*(1024)-3 ;# OF BYTES TO ADD UP
               LD      DE,0

```

```

099F 3E 00 C LD A,0 ;HIGH BYTE ACCUMULATOR
09A1 F5 C PUSH AF
C
09A2 F1 C MORE: POP AF ;RESTORE HIGH BYTE OF SUM
09A3 5E C LD E,(HL) ;GET BYTE
09A4 23 C INC HL
09A5 0B C DEC BC
09A6 DD 19 C ADD IX,DE
09A8 CE 00 C ADC A,0 ;ADD CARRY TO HIGH BYTE
09AA F5 C PUSH AF ;SAVE HIGH BYTE OF CHECK-SUM
09AB 78 C LD A,B
09AC B1 C OR C
09AD 20 F3 C JR NZ,MORE ;FINISHED ADDING YET ?
C
09AF F1 C POP AF
09B0 21 1FFF C LD HL,8*(1024)-1
09B3 BE C CP (HL) ;COMPARE WITH HIGH BYTE
09B4 FD 21 2052 C LD IX,' R' ;R = ROM ERROR
09B8 C2 0A46 C JP NZ,ERROR
09BB DD E5 C PUSH IX
09BD D1 C POP DE
09BE 2B C DEC HL ;POINT TO MID BYTE
09BF 7A C LD A,D
09C0 BE C CP (HL) ;COMPARE WITH MIDDLE BYTE
09C1 C2 0A46 C JP NZ,ERROR
09C4 2B C DEC HL
09C5 7B C LD A,E
09C6 BE C CP (HL) ;COMPARE WITH LOW BYTE
09C7 C2 0A46 C JP NZ,ERROR
09CA C9 C RET ;YEA YEA, ROM'S O.K. !
C
;***** LSI REGISTERS TEST *****
C
C
C
C
09CB C LSI_TEST:
09CB FD 21 2050 C LD IX,' P' ;P = PARALELL CHIP ERROR
09CF CD 09DA C CALL PIA
09D2 FD 21 2053 C LD IX,' S' ;S = SIO ERROR
09D6 CD 0A27 C CALL SIO
C ; LK IX,' C' ;C = CTC ERROR
C ; CALL CTC8253
C
09D9 C9 C RET
C
09DA 3E 3F C PIA: LD A,03FH ;TOP 2 BITS ARE INTERRUPTS
09DC CD 09FF C CALL XJ010 ;TEST SYS CONTROL REGISTERS
09DF 3E 00 C LD A,0
09E1 CD 09FF C CALL XJ010
C
09E4 3E 80 C LD A,000H ;HIGH BIT KEEPS THE ROM ON
09E6 CD 0A15 C CALL XJ011 ;TEST SYS DIR REGISTERS
09E9 3E FF C LD A,0FFH
09EB CD 0A15 C CALL XJ011
C
09EE 3E 04 C LD A,04H
09F0 D3 01 C OUT (SYS_CNA),A

```

'POWER ON DIAGNOSTICS'

```

09F2 D3 03 C OUT (SYS_CNB),A ;ENABLE DATA REGISTERS
09F4 3E 88 C LD A,010001000B ;HIGH BIT KEEPS THE ROM ON
09F6 CD 0A15 C CALL XJ011 ;TEST SYS DATA REGISTERS
09F9 3E F7 C LD A,011110111B ;THIS PATTERN LEAVES DRIVES & SPEAKER OFF
09FB CD 0A15 C CALL XJ011
09FE C9 C RET
C
09FF 47 C XJ010: LD B,A
0A00 D3 01 C OUT (SYS_CNA),A
0A02 D3 03 C OUT (SYS_CNB),A
0A04 DB 01 C IN A,(SYS_CNA)
0A06 E6 3F C AND 00111111B
0A08 B8 C CP B
0A09 C2 0A46 C JP NZ,ERROR
0A0C DB 03 C IN A,(SYS_CNB)
0A0E E6 3F C AND 00111111B
0A10 B8 C CP B
0A11 C2 0A46 C JP NZ,ERROR
0A14 C9 C RET
C
0A15 47 C XJ011: LD B,A
0A16 D3 00 C OUT (SYS_DTA),A
0A18 D3 02 C OUT (SYS_DTB),A
0A1A DB 00 C IN A,(SYS_DTA)
0A1C B8 C CP B
0A1D C2 0A46 C JP NZ,ERROR
0A20 DB 02 C IN A,(SYS_DTB)
0A22 B8 C CP B
0A23 C2 0A46 C JP NZ,ERROR
0A26 C9 C RET
C
0A27 0E 00 C SIO: LD C,0
0A29 CD 0A32 C CALL SIO_TEST
0A2C 0E F1 C LD C,0F1H
0A2E CD 0A32 C CALL SIO_TEST
0A31 C9 C RET
C
0A32 3E 02 C SIO_TEST: LD A,2
0A34 D3 0F C OUT (SIO_CNB),A ;POINT TO WRITE REG 2
0A36 79 C LD A,C
0A37 D3 0F C OUT (SIO_CNB),A ;WRITE PATTERN
0A39 3E 02 C LD A,2
0A3B D3 0F C OUT (SIO_CNB),A ;POINT TO READ REGISTER 2
0A3D DB 0F C IN A,(SIO_CNB) ;READ IT BACK
0A3F E6 F1 C AND 011110001B ;REMOVE BITS WHICH MAY CHANGE
;BECAUSE OF STATUS AFFECTS VECTOR
;STATE OF SIO
C
0A41 B9 C CP C
0A42 C2 0A46 C JP NZ,ERROR ;THEY BETTER BE THE SAME !
0A45 C9 C RET
C
;CTC0253: LK C,TIM_CT0
; CALL CTC_TEST
; LK C,TIM_CT1
; CALL CTC_TEST

```



```

C      ;      LK      A,060H
C      ;      OUT     TIM_CT2
C      ;      LK      A,0
C      ;      OUT     TIM_CT2      ;CT2 NEVER GETS INITIALIZED
C      ;      LK      C,TIM_CT2
C      ;      CALL    CTC_TEST
C      ;      RET
C      ;
C      ;CTC_TEST:  IN,C      A
C      ;      MOV     E,A
C      ;      IN,C      A
C      ;      MOV     D,A      ;PUT VALUE IN DE
C      ;      IN,C      A
C      ;      MOV     L,A
C      ;      IN,C      A
C      ;      MOV     H,A      ;PUT 'LATER' VALUE IN HL
C      ;      OR      A      ;RESET CARRY FLAG FOR SBC
C      ;      SBC     HL,DE
C      ;      MOV     A,H
C      ;      OR      L
C      ;      JZ      ERROR      ;COUNTER NOT DECREMENTING !
C      ;      RET
C
C      ;*****      COMMON ROUTINES AND MESSAGES      *****
C
0A46    FD E5    C      ERROR:  PUSH    IY      ;SAVE ERROR CODE
0A48    CD 003F  C      CALL     INITIALIZE ;RESTORE INTERRUPTS & STUFF
0A4B    11 0A75  C      LD      DE,ERROR_MSG ;OH NOOOOOO !
0A4E    CD 0323  C      CALL    OSTR
0A51    3E C0    C      LD      A,0C0H
0A53    D3 00    C      OUT     (SYS_DTA),A
0A55    C1       C      POP     BC      ;RESTORE ERROR CODE
0A56    21 C000  C      LD      HL,0C000H
0A59    71       C      LD      (HL),C      ;SHOW WHICH ERROR
0A5A    1B FE    C      JR      $      ;SO RESET JUMPS TO BOOT ROUTINE
C
0A5C    1A 1B 3D 2A C      TEST_MESSAGE: DB      'Z'-040H,ESC,'=',32+10,32+30,'PERFORMING SELF-TEST'
0A60    3E 50 45 52 C
0A64    46 4F 52 4D C
0A68    49 4E 47 20 C
0A6C    53 45 4C 46 C
0A70    2D 54 45 53 C
0A74    D4       C      DC      'T'
C
0A75    1B 3D 2A 39 C      ERROR_MSG:  DB      ESC,'=',32+10,32+25,'COMPUTER NOT WORKING PROPERLY'
0A79    43 4F 4D 50 C
0A7D    55 54 45 52 C
0A81    20 4E 4F 54 C
0A85    20 57 4F 52 C
0A89    4B 49 4E 47 C
0A8D    20 50 52 4F C
0A91    50 45 52 4C C
0A95    59       C
0A96    1B 3D 2B 38 C      DB      ESC,'=',32+11,32+24,'PLEASE CONTACT AN OSBORNE DEALER'
0A9A    50 4C 45 41 C
0A9E    53 45 20 43 C

```

'POWER ON DIAGNOSTICS'

0AA2	4F 4E 54 41	C
0AA6	43 54 20 41	C
0AAA	4E 20 4F 53	C
0AAE	42 4F 52 4E	C
0AB2	45 20 44 45	C
0AB6	41 4C 45 52	C
0ABA	1B 3D 2C 3C	C
0ABE	50 52 4F 43	C
0AC2	45 45 44 20	C
0AC6	41 54 20 59	C
0ACA	4F 55 52 20	C
0ACE	4F 57 4E 20	C
0AD2	52 49 53	C
0AD5	CB	C
		C
		C

DB ESC, '=', 32+12, 32+28 , 'PROCEED AT YOUR OWN RIS'

DC 'K'

PAGE

```

C
C
C      ; *****
C      ; *
C      ; *          DIAGNOSTIC INITIALIZATION          *
C      ; *
C      ; *****
C
C      ;INITIALIZE SYSTEM FOR DIAGNOSTIC
C      DIAG_INIT:
0AD6      CALL    INITIALIZE      ;INITIALIZE MEMORY & SYSTEM PIA
0AD6      CD 003F
0AD9      CD 0E7A'
0ADC      CD 0223
0ADF      C9
C          RET
C          PAGE

```

'EXECUTIVE ROM MONITOR'
'POWER ON DIAGNOSTICS'

MACRO-80 3.44 09-Dec-81 PAGE 1-90

;

SUBTTL
PAGE

'CONSOLE ROUTINES'

```

C      SUBTTL      'CONSOLE ROUTINES'
C      INCLUDE ROM2      ;CONSOLE ROUTINES
C      ;
C      ; *****
C      ; *
C      ; *          CONSOLE ROUTINES          *
C      ; *
C      ; *****
C      ;
C      ; MODULE 2 --ROM2-- ENTRY POINTS
C      ;
C      ;          CONSOLE ROUTINES
C      ;
C      ; SKEY:      GET STATUS OF KEYBOARD
C      ; CON_IN:    GETS A KEYSTROKE
C      ; COUT:      GENERAL OUTPUT TO CONSOLE
C      ; LOOKUPB:   LOOK UP CHAR. IN TABLE AND BRANCH TO PROPER ROUTINE
C      ; SETTBL:    SET TABLE POINTER TO CURRENT TABLE
C      ; PROCNORM:  PROCESS 'NORMAL' CHAR. (I.E. NOT LEAD-IN CHAR.)
C      ;             CHECKS FOR GRAPHICS OR NORMAL CHAR. AND BRANCH TO
C      ;             PROPER ROUTINE
C      ; VGRAPH:    PROCESS GRAPHICS CHAR.
C      ; VNORM:     PROCESS NORMAL CHAR.
C      ; VOUT:      DISPLAY NEW CHAR. AND UPDATE CURSOR
C      ; VC_CR:     PERFORM CARRIAGE RETURN
C      ; VC_LF:     PERFORM LINE FEED
C      ; VC_NEWLINE: PERFORM CARRIAGE RETURN/LINE FEED
C      ; VC_BKS:    MOVE CURSOR LEFT (BACKSPACE)
C      ; VC_MCRT:   MOVE CURSOR RIGHT
C      ; VC_MCU:    MOVE CURSOR UP
C      ; VC_MCDOWN: MOVE CURSOR DOWN
C      ; VC_HOME:   'HOME' CURSOR (MOVE TO UPPER LEFT CORNER)
C      ; DO_CR:     PROCESS CARRIAGE RETURN
C      ; DO_LF:     PROCESS LINE FEED
C      ; EEOL:      ERASE TO END OF LINE
C      ; EEOS:      ERASE TO END OF SCREEN
C      ; EDEL:      DELETE CHAR.
C      ; EINSRT:    INSERT CHAR.
C      ; ESCRR:     DELETE LINE
C      ; ESCCE:     INSERT LINE
C      ; CLR_CHAR:  CLEAR CHAR. FOR GIVEN POSITION
C      ; CLR_EOS:   CLEAR TO END OF WINDOW
C      ; VC_CLRS:   CLEAR ENTIRE WINDOW
C      ; ESCZZ:     CLEAR ENTIRE WINDOW
C      ; SCROL_UP:  SCROLL UP
C      ; SCROL_DOWN: SCROLL DOWN
C      ; S_UP_DOWN: PERFORM SCROLLING
C      ; VC_BEL:    RING BELL
C      ; ESC_LCK:   LOCK KEYBOARD
C      ; ESC_ULK:   UNLOCK KEYBOARD
C      ; ESCSAL:    ENABLE ALTERNATE CHAR. SET
C      ; ESCSHI:    ENABLE HALF INTENSITY
C      ; ESCSGR:    ENABLE GRAPHICS
C      ; ESCSRV:    ENABLE REVERSE VIDEO

```

C	;	ESCSUN:	ENABLE UNDER LINE
C	;	ESCSBL:	ENABLE BLINK
C	;	ESCCAL:	DISABLE ALTERNATE CHAR. SET
C	;	ESCCHI:	DISABLE HALF INTENSITY
C	;	ESCCGR:	DISABLE GRAPHICS
C	;	ESCCRV:	DISABLE REVERSE VIDEO
C	;	ESCCUN:	DISABLE UNDERLINE
C	;	ESCCBL:	DISABLE BLINK
C	;	RECON:	SET REVERSE VIDEO BACKGROUND
C	;	REVOFF:	CLEAR REVERSE VIDEO BACKGROUND
C	;	NEW_BACK:	DEFINE NEW BACKGROUND
C	;	DEFCUR:	DEFINE CURSOR TYPE
C	;	SET_ATR:	SET SELECTED ATTRIBUTES
C	;	TOG_ATR:	TOGGLE SELECTED ATTRIBUTES
C	;	TOG_CUR:	TOGGLE CURSOR
C	;	SET_BACK:	SET BACKGROUND
C	;	GET__ATR:	CONVERT ATTRIBUTES (FROM TV950 TO INTERNAL)
C	;	YCOORD:	SET Y COORDINATE FOR CURSOR
C	;	XCOORD:	SET X COORDINATE FOR CURSOR
C	;	SETOFF:	SET X-Y COORDINATE OFFSET
C	;	ADDRROWS:	ADD ROWS TO GIVEN ADDRESS
C	;	ABSOLUTE:	GET ABSOLUTE VALUE FOR OFFSET
C	;	LASTROW:	CHECK IF ADDRESS ON LAST LINE
C	;	GETCOL:	GET COLUMN NUMBER OF ADDRESS
C	;	GETROW:	GET ROW NUMBER OF ADDRESS
C	;	SETCOL:	SET COLUMN NUMBER
C	;	SETROW:	SET ROW NUMBER
C	;	DEF_WINDOW:	DEFINE AND SET WINDOW
C	;	STARTROW:	DEFINE FIRST ROW OF WINDOW
C	;	STARTCOL:	DEFINE FIRST COLUMN OF WINDOW
C	;	ENDROW:	DEFINE LAST ROW OF WINDOW
C	;	ENDCOL:	DEFINE LAST COLUMN OF WINDOW
C	;	SETWIND:	SET WINDOW NUMBER
C	;	GETWIND:	GET ADDRESS OF WINDOW DEFINITION
C	;	SEQERR:	HANDLE UNDEFINED ESC SEQUENCE
C	;	RETPROC:	RETURN (USED WHEN NO PROCESSING REQUIRED)
C	;	DISPLAY:	DISPLAY CHAR. AT CURRENT POSITION
C	;	DIST2EOL:	CALCULATE DISTANCE TO END OF LINE
C	;	DIST2LASTROW:	CALCULATE DISTANCE TO END OF SCREEN
C	;	NOTE:	PLAY A MUSICAL NOTE
C			
C		PAGE	

0AE0

```
C
C
C      .Z80
C      ;      .XLIST
C      ;      INCLUDE      SYSS1.MAC      ;SYSTEST FILE
C      ;      INCLUDE      RAMS10.MAC      ;RAM MEMORY LOCATIONS AND CONSTANTS
C      ;      INCLUDE      RAMSMON.MAC      ;MONITOR RAM MEMORY LOCATIONS
C      ;      .LIST
C      ;      ;SYSS1.MAC , RAMSMON.MAC AND RAMS10.MAC INCLUDED
C
C      ;      EXT      RM_C_BF,GET_INT,SET_INT
C
C      ;      PUBLIC  SKEY,CON_IN,COUT,ALT_FLG,CTBLPTRS,LENCTBL
C
C      CSEG
C
C      PAGE
```

```

C
C
0000' C SKEY:
C ;GET STATUS OF KEYBOARD
C
C ;EXIT
C ;A = 0 IF NO CHARACTER READY
C ;A = 0FFH IF CHARACTER READY
C ;ZBIT SET IF NO DATA READY
C
C
C ;CHECK FOR LOCKED KEYBOARD
0000' C LD A, (KEYLCK)
0003' C OR A
0004' C RET Z ;IF LOCKED KEYBOARD
C
C ;CHECK FOR PROCESSING FUNCTION KEY
0005' C LD A, (COUNT)
0008' C OR A ;IF PROCESSING FUNCTION KEY --
0009' C JP NZ, XJ018 ; RETURN ZBIT CLEAR AND A = 0FFH (DATA READY)
C
C ;CHECK BUFFER STATUS
C ; LDK B, KBD_DEV ;KEYBOARD
C ; CALL RM_CH_ST ;CHECK KEYBOARD BUFFER STATUS
000C' C LD HL, (D_RT_TBL+D_RT_LEN*KBD_DEV+4) ; LOAD CHAR CNT FROM DEVICE ROOT
000F' C LD A, H ;IS CHAR COUNT ZERO?
0010' C OR L ;
0011' C JR Z, XJ019 ;ZBIT SET IF NO CHARACTER READY
C
0013' C XJ018: OR 0FFH ;SET A = 0FFH, CLEAR Z FLAG
0015' C RET
C
0016' C XJ019: XOR A ;A = 0 IF NO CHARACTER
0017' C RET
C PAGE

```



```

C
C
0018' C      CON_IN:
C      ;      GETS KEYSTROKE
C      ;      FUNCTION KEY TRANSLATION PERFORMED ON CODES 80H TO 8DH DEPENDING
C      ;      ON SYSTEM FLAG FKEYFLAG:
C      ;
C      ;      FKEYFLAG = 0 --- NO TRANSLATION
C      ;      FKEYFLAG = 1 --- TRANSLATION AS PER TABLE
C      ;
C      ;      EXIT  A = KEY CODE IN ASCII
C      ;
C
C      ;CHECK IF FUNCTION KEY IS BEING TRANSLATED
C
0018' 3A 23E4 C      XJ020: LD      A, (COUNT)
001B' B7      C      OR      A
001C' 28 0D   C      JR      Z, XJ021      ;NO FUNCT. TRANS. -- GET KEY FROM KEYBOARD BUFFER
C
C      ;FUNCTION KEY NEEDS TRANSLATION
C      ;      'PNTR' = ADDRESS OF NEXT BYTE OF TRANSLATION
C      ;      'COUNT' = # OF BYTES LEFT FOR TRANSLATION
C
001E' 2A 23E5 C      LD      HL, (FPNTR)
0021' 7E      C      LD      A, (HL)      ;GET NEXT BYTE OF TRANSLATION
0022' 23      C      INC      HL
0023' 22 23E5 C      LD      (FPNTR), HL      ;ADVANCE TRANSLATION POINTER
0026' 21 23E4 C      LD      HL, COUNT
0029' 35      C      DEC      (HL)      ;REDUCE COUNT OF BYTES LEFT
002A' C9      C      RET
C
C      ;NO TRANSLATION -- GET KEY FROM KEYBOARD BUFFER
C
C      ;WAIT FOR KEY READY
002B' CD 0000' C      XJ021: CALL    SKEY      ;GET STATUS
002E' 2B FB   C      JR      Z, XJ021      ;IF NO DATA, WAIT
C
C      ;GET KEY
0030' 06 00   C      LD      B, KBD_DEV      ;KEYBOARD
0032' F3      C      DI
0033' CD 1226' C      CALL    RM_C_BF ;GET CHARACTER FROM KEYBOARD BUFFER
0036' 2A 2421 C      LD      HL, (D_RT_TBL+D_RT_LEN*KBD_DEV+4) ; DECREMENT CHARACTER COUNT
0039' 2B      C      DEC      HL
003A' 22 2421 C      LD      (D_RT_TBL+D_RT_LEN*KBD_DEV+4), HL ; AND SAVE IT BACK
003D' FB      C      EI
C
C      ;CHECK IF KEY IS A FUNCTION KEY
003E' FE 80   C      CP      80H
0040' D8      C      RET      C      ;RETURN IF KEYSTROKE < 80H (NOT A FUNCTION KEY)
0041' FE 8E   C      CP      8EH
0043' D0      C      RET      NC     ;RETURN IF KEYSTROKE > 8FH (NOT A FUNCTION KEY)
C

```

```

C      ;IF FUNCTION KEY RETURNED FROM BUFFER
C      ;      DETERMINE WHICH TRANSLATION TABLE IS NEEDED
C
0044' 47      C      LD      B,A          ;SAVE KEY
0045' 3A 23D9 C      LD      A,(FKEYFLAG)
0048' B7      C      OR      A          ;SET FLAGS
0049' 78      C      LD      A,B        ;RESTORE CHARACTER
004A' C8      C      RET      Z          ;RETURN IF NO TRANSLATION
C
004B' 21 2155 C      LD      HL,FKEY_ADR    ;TRANSLATION TABLE ADDRESS
C
C      ;      INITIALIZE FPNTN AND COUNT
C      ;GET FUNCTION KEY NUMBER
004E' E6 0F   C      XJ022: AND    0FH          ;GET FUNCTION KEY NUMBER
0050' 47      C      LD      B,A          ;B = FUNCTION KEY NUMBER
0051' 2B 07   C      JR      Z,XJ023      ;IF FUNCTION KEY 0
C
C      ;GET FUNCTION KEY'S ADDRESS
0053' 16 00   C      LD      D,0
0055' 5E      C      XJ024: LD      E,(HL)    ;E = LENGTH OF THIS FUNCTION KEY
0056' 1C      C      INC      E          ;COUNT LENGTH BYTE
0057' 19      C      ADD     HL,DE        ;HL = ADDRESS OF NEXT FUNCTION KEY
0058' 10 FB   C      DJNZ    XJ024      ;IF NOT TO THE RIGHT KEY YET
C
C      ;INITIALIZE FPNTN AND COUNT
005A' 7E      C      XJ023: LD      A,(HL)    ;GET LENGTH OF THIS FUNCTION KEY
005B' 32 23E4 C      LD      (COUNT),A
005E' 23      C      INC      HL          ;POINT TO FIRST BYTE OF TRANSLATION
005F' 22 23E5 C      LD      (FPNTN),HL
0062' 1B B4   C      JR      XJ020      ;RETURN FIRST BYTE OF THIS FUNCTION KEY TRANSLATION
C
C      PAGE

```

```

C
C
0064' C      COUT:
C
C      ;GENERAL OUTPUT ROUTINE TO VIDEO SCREEN
C
C      ;ENTRY
C      ;C      =      CHARACTER
C      ;CURPOS =      CURSOR
C      ;CFLAG =      FLAG+MODE
C
C      ;CURPOS & CFLAG UPDATED, A=CHARACTER
C      ;(BC, DE, HL PRESERVED)
C
C
C      ;BRING IN VIDEO BANK
0064' DB 00 C      IN      A, (SYS_DTA)      ;GET CURRENT BANKS
0066' F5 C      PUSH     AF      ;SAVE CURRENT BANKS
0067' F6 40 C      OR      VID_BANK
0069' D3 00 C      OUT     (SYS_DTA), A      ;TURN ON VIDEO BANK
C
C      ;TURN OFF OLD CURSOR
006B' CD 0309' C      CALL    TOG_CUR      ;TOGGLE CURSOR
C
C      ;PROCESS CHARACTER
006E' CD 0078' C      CALL    LOOKUPB      ;RUN PROCEDURE ADDRESSED IN TABLE
C
C      ;TURN ON NEW CURSOR
0071' CD 0309' C      CALL    TOG_CUR
C
C      ;BRING BACK OLD BANKS
0074' F1 C      POP      AF      ;GET OLD BANKS
0075' D3 00 C      OUT     (SYS_DTA), A
C
C      RET      ;RETURN, END OF COUT SUBR.
C
C      PAGE

```

```

C
C ; *****
C ; *
C ; *          TABLE HANDLING ROUTINES
C ; *
C ; *****
C
0078' C LOOKUPB:
C ;LOOK UP CHARACTER IN THE SPECIFIED TABLE AND BRANCH TO PROCESSING PROCEDURE
C ;IF CHARACTER IS NOT IN THE TABLE, THEN BRANCH TO DEFAULT PROCEDURE.
C ;ENTRY
C ;C      =      CHAR
C ;*NOTE*
C ;AFTER GETTBL IS CALLED:
C ;HL      =      TABLE ADDRESS
C ;      1ST BYTE OF TABLE = NUMBER OF (NON-DEFAULT) TABLE ENTRIES
C ;      ENTRIES ARE 4 BYTES LONG:
C ;      1ST BYTE = MATCH CODE
C ;      2ND BYTE = NEXT TABLE NUMBER
C ;      4TH,5TH BYTES = BRANCH ADDRESS
C ;      THE LAST ENTRY IS A 3-BYTE "NO MATCH" DEFAULT NEXT TABLE NUMBER AND BRANCH ADDRESS
C
C ;EXIT
C ;JUMPS TO BRANCH ADDRESS IN TABLE
C ;C      =      CHARACTER
C
C
C ;GET CURRENT TABLE
0078' 2A 2359 C      LD      HL, (ROMCON)      ;GET DE = CURRENT TABLE POINTER
0078' 5E C      LD      E, (HL)
007C' 23 C      INC     HL
007D' 56 C      LD      D, (HL)
007E' EB C      EX      DE,HL      ;HL = CURRENT TABLE
C
C ;FIND CHARACTER IN TABLE
007F' 7E C      LD      A, (HL)      ;A = NUMBER OF ENTRIES
0080' 23 C      INC     HL      ;FIRST ENTRY
0081' B7 C      OR      A
0082' 28 0B C      JR      Z, XJ025      ;IF NO ENTRIES, DO DEFAULT PROCESSING
C
0084' 47 C      LD      B, A      ;B = NUMBER OF ENTRIES
0085' 79 C      LD      A, C      ;A = CHARACTER
C
XJ026: 0086' BE C      CP      (HL)      ;CHECK FOR MATCH
0087' 23 C      INC     HL      ; (2ND BYTE OF THIS 4 BYTE ENTRY)
0088' 28 05 C      JR      Z, XJ025      ;IF MATCH PROCESS
C
008A' 23 C      INC     HL      ; (3RD BYTE OF THIS ENTRY)
008B' 23 C      INC     HL      ; (4TH BYTE OF THIS ENTRY)
008C' 23 C      INC     HL      ;1ST BYTE OF NEXT ENTRY
008D' 10 F7 C      DJNZ    XJ026      ;CONTINUE THRU BODY OF TABLE
C
C ;SET UP NEXT TABLE ADDRESS
008F' E5 C      XJ025: PUSH    HL      ;SAVE CURRENT POSITION IN TABLE
C

```

0090'	7E	C	LD	A, (HL)	;A = NEXT TABLE NUMBER
0091'	CD 009C'	C	CALL	SETTBL	;SET NEXT TABLE
		C			
0094'	E1	C	POP	HL	;RESTORE
		C			
		C	;BRANCH TO PROCESSING PROCEDURE		
0095'	23	C	INC	HL	;GO TO BRANCH ADDRESS PORTION OF TABLE ENTRY
0096'	7E	C	LD	A, (HL)	;1ST BYTE (LOW ORDER ADRS)
0097'	23	C	INC	HL	
0098'	66	C	LD	H, (HL)	;2ND BYTE (HIGH ORDER ADRS)
0099'	6F	C	LD	L, A	;HL=ADRS FROM TABLE
009A'	79	C	LD	A, C	;A = CHAR
009B'	E9	C	JP	(HL)	;ENTER ROUTINE PER TABLE ADRS.
		C			
		C			
			PAGE		

```

C
009C' C      SETTBL:
C      ;SET CURRENT TABLE POINTER TO POINT TO SELECTED TABLE
C      ;ENTRY
C      ;A      =      TABLE NUMBER
C
C      ;EXIT
C      ;PRESERVES BC
C
009C' C5      C      PUSH    BC
C
009D' ED 5B 2359 C      LD      DE, (ROMCON) ;GET DE = CURRENT TABLE POINTER
C      ;      (BASE ADDRESS FOR TABLE OF TABLE POINTERS)
00A1' 87      C      ADD     A,A      ;A = TABLE NUMBER * 2
00A2' 6F      C      LD      L,A
00A3' 26 00    C      LD      H,0      ;HL = TABLE NUMBER * 2
00A5' 19      C      ADD     HL,DE     ;[HL] = NEXT TABLE ADDRESS
C      ;DE => TABLE ADDRESS POINTER
00A6' ED A0    C      LDI
00A8' ED A0    C      LDI          ;SET TABLE ADDRESS TO NEXT TABLE ADDRESS
00AA' C1      C      POP     BC
00AB' C9      C      RET
C
C      PAGE

```

```

C
C
C      ;BASTBL PROCEDURES
C
00AC'  C      PROCNORM:
C      ;NORMAL CHARACTER ENTERED (IE NOT A LEAD-IN CHARACTER)
C      ;CHECKS FOR GRAPHICS OR NORMAL CHARACTER AND PASSES CHARACTER ON TO APPROPRIATE
C      ;ROUTINE
C      ;ENTRY
C      ;C      =      CHARACTER
C
C      ;EXIT
C      ;JUMPS TO VGRAPH OR VNORM WHICH PROCESS CHARACTER AND UPDATE CURSOR
C
C
00AC'  3A 237A      C      LD      A, (CFLAG)      ;GET FLAGS
00AF'  E6 04      C      XJ027: AND     GR_FLG      ;CHECK FOR GRAPHICS MODE
00B1'  20 02      C      JR      NZ, VGRAPH      ;GRAPHICS MODE -- PROCESS CHARACTER
C
00B3'  18 10      C      JR      VNORM      ;NORMAL MODE -- PROCESS CHARACTER, SET HL = CURSOR POSITION
C
C      PAGE
C

```

```

C
00B5' C VGRAPH:
C ;GRAPHICS MODE CHARACTER PROCESSING
C ;CHARACTERS 60H-7FH ARE CONVERTED TO GRAPHICS CHARACTERS (00H-1FH)
C ;SO THAT BASIC CAN ACCESS ALL GRAPHICS CHARACTERS
C ;ENTRY
C ;C = CHAR TO PROCESS
C
00B5' 79 C LD A,C
00B6' CB 6F C BIT 5,A
00B8' CA 00D3' C JP Z,VOUT
00BB' CB 77 C BIT 6,A
00BD' CA 00D3' C JP Z,VOUT
00C0' E6 9F C AND 9FH ;CONVERT TO GRAPHICS CHARACTERS
00C2' C3 00D3' C JP VOUT
C
C
C
C
C
C
C
C
C
00C5' C V NORM:
C ;NORMAL MODE CHARACTER PROCESSING.
C ;ENTRY
C ;C = CHAR TO OUTPUT
C
00C5' 79 C LD A,C
00C6' FE 20 C CP 20H
00C8' D2 00D3' C JP NC,VOUT ;IF NOT CONTROL CHAR, OUTPUT IT
C
C ;IF CONTROL CHARACTER
00C8' 3E 03 C LD A,CTLNUM ;SET CURRENT TABLE TO CONTROL TABLE
00CD' CD 009C' C CALL SETTBL
C
00D0' C3 0078' C JP LOOKUPB ;SCAN TABLE OF VALID CONTROL CHRS AND BRANCH TO APPROPRIATE ROUTINE.
C
C PAGE

```



```

C
C
0003' C      VOUT:
C      ;DISPLAY NEW CHARACTER AND UPDATE CURSOR
C      ;ENTRY
C      ;A      =      CHAR
C
C
0003' C      CALL  DISPLAY      ;DISPLAY NEW CHARACTER
0006' C3 0113' C      JP      VC_MCRT  ;MOVE CURSOR RIGHT
C      PAGE
```

```

C
C
C      ;      *****
C      ;      *
C      ;      *          CONTROL CHARACTER CODE PROCESSING      *
C      ;      *          CURSOR MOVEMENTS                        *
C      ;      *
C      ;      *****
C
00D9'  C      VC_CR:
C      ;CARRIAGE RETURN
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;CURPOS =      NEW CURSOR POSITION
C
00D9'  2A FD53  C      LD      HL, (CURPOS)
00DC'  CD 0159' C      CALL   DD_CR
00DF'  22 FD53  C      LD      (CURPOS), HL
00E2'  C9      C      RET
C      PAGE

```

```

C
C
00E3' C      VC_LF:
C      ;LINE FEED (IF CURSOR IS ON LAST LINE, SCROLL SCREEN AND ADD
C      ; NEW LINE AT BOTTOM)
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;CURPOS =      NEW CURSOR POSITION
C
C
00E3' 2A FD53 C      LD      HL, (CURPOS)
00E6' CD 0167' C      CALL   DO_LF
00E9' 22 FD53 C      LD      (CURPOS), HL
00EC' C9      C      RET
C
C      PAGE

```

```

C
C
00ED' C VC_NEWLINE:
C ;DO CR AND LF
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;CURPOS UPDATED
C
C
00ED' CD 00D9' C CALL VC_CR
00F0' C3 00E3' C JP VC_LF
C
C PAGE
```

```

C
C
00F3'  VC_BKS:
C      ;MOVE CURSOR LEFT (BACKSPACE).
C      ;(IF IN 1ST COLUMN THEN WRAP AROUND TO LAST COLUMN OF PREVIOUS LINE)
C      ;(NO EFFECT IF IN 1ST POSITION OF WINDOW)
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;CURPOS =      NEW CURSOR POSITION
C
C      ;CHECK IF AT COLUMN 0
00F3'  2A FD53  C      LD      HL,(CURPOS)
00F6'  CD 03AC'  C      CALL   GETCOL      ;GET CURSOR COLUMN
00F9'  2B 06     C      JR      Z,XJ028      ;IF MUST WRAP FROM COL 0 TO LAST COL OF PREVIOUS LINE
C
C      ;IF NOT, BACKSPACE AND RETURN
00FB'  2A FD53  C      LD      HL,(CURPOS)
00FE'  2B      C      DEC     HL
00FF'  1B 0E     C      JR      XJ029
C
C      ;IF AT COLUMN 0, CHECK FOR ROW 0
0101'  CD 03B9'  C      XJ028: CALL   GETROW
0104'  CB      C      RET      Z      ;RETURN WITH NO EFFECT IF AT COL 0, ROW 0
C
C      ;IF NOT, WRAP AROUND TO PREVIOUS LINE
0105'  CD 012B'  C      CALL   VC_MCUP      ;MOVE UP A LINE (HL = NEW ADDRESS)
C
0108'  2A FD53  C      LD      HL,(CURPOS)      ;MOVE TO END OF LINE
010B'  CD 04B1'  C      CALL   DIST2EOL      ;BC = DISTANCE FROM CURSOR TO END OF LINE
010E'  09      C      ADD     HL,BC      ;HL = NEW CURSOR ADDRESS
C
010F'  22 FD53  C      XJ029: LD      (CURPOS),HL      ;SAVE NEW CURSOR POSITION
0112'  C9      C      RET
C      PAGE

```

```

C
C
0113' C VC_MCRT:
C ;MOVE CURSOR RIGHT (IF AT END OF LINE, WRAP AROUND TO NEXT LINE)
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;CURPOS = NEW CURSOR POSITION
C
0113' 2A FDS3 C LD HL, (CURPOS)
0116' CD 04B1' C CALL DIST2EOL ;A = # OF BYTES BETWEEN CURSOR AND END OF LINE
0119' 2B 06 C JR Z, XJ030 ;IF AT LAST COLUMN
C
011B' 2A FDS3 C LD HL, (CURPOS)
011E' 23 C INC HL ;MOVE CURSOR
011F' 1B 06 C JR XJ031
C ;IF AT LAST COLUMN, WRAP AROUND TO NEXT LINE
0121' CD 0159' C XJ030: CALL DO_CR ;DO CR...
0124' CD 0167' C CALL DO_LF ;...AND LF.
C
0127' 22 FDS3 C XJ031: LD (CURPOS), HL ;SAVE NEW CURSOR POSITION
012A' C9 C RET
C PAGE

```

```

C
C
012B' VC_MCU:
C      ;MOVE CURSOR UP. (IF CURSOR IS ON TOP LINE THEN NOTHING HAPPENS)
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;CURPOS =      NEW CURSOR POSITION
C
C
C      ;MOVE UP A LINE
012B' 2A FD53 C      LD      HL,(CURPOS)
012E' B7      C      OR      A      ;CLEAR CARRY
012F' 01 0080 C      LD      BC,VLL
0132' ED 42    C      SBC     HL,BC      ;MOVE UP A LINE
C
C      ;CHECK IF ABOVE SCREEN
0134' E5      C      PUSH    HL      ;SAVE CURSOR
0135' ED 5B 237B C      LD      DE,(FWIND)
0139' ED 52    C      SBC     HL,DE      ;CARRY FLAG SET IF ABOVE SCREEN
013B' E1      C      POP     HL
013C' 30 01    C      JR      NC,XJ032      ;DONE IF ON SCREEN
C
C      ;IF SO, GO BACK TO OLD POSITION
013E' 09      C      ADD     HL,BC
C
013F' 22 FD53 C      XJ032: LD      (CURPOS),HL
0142' C9      C      RET
C
C      PAGE

```

```

0143'      C      VC_MCDOWN:
C      ;MOVE CURSOR DOWN (NO EFFECT IF CURSOR IS ON LAST LINE)
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;CURPOS UPDATED
C
C      ;CHECK IF ON LAST LINE
0143'  2A FD53  C      LD      HL, (CURPOS)
0146'  CD 03A2'  C      CALL   LASTROW      ;CHECK IF ON LAST ROW
0149'  C8        C      RET      Z          ;IF ON LAST ROW, DO NOTHING
C
C      ;IF NOT, MOVE CURSOR DOWN
014A'  11 00B0  C      LD      DE, VLL
014D'  19        C      ADD     HL, DE
014E'  22 FD53  C      LD      (CURPOS), HL
0151'  C9        C      RET
C
C      PAGE

```



```

C
C
0152' C VC_HOME:
C ;HOME CURSOR
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;CURPOS = NEW CURSOR POSITION
C
C
0152' 2A 237B C LD HL,(FWAWIND) ;HOME POSITION
0155' 22 FD53 C LD (CURPOS),HL
0158' C9 C RET
C
C PAGE
```

```

C
C
0159' C DO_CR:
C      ;DO CARRIAGE RETURN PROCESSING
C      ;ENTRY
C      ;HL = ADDRESS TO PROCESS
C
C      ;EXIT
C      ;HL = ADDRESS AFTER CARRIAGE RETURN
C
C      ;CLEAR COLUMN BITS IN SELECTED ADDRESS
0159' 3E 80 C      LD A,80H
015B' A5 C      AND L
015C' 6F C      LD L,A
C
C      ;SET FIRST COLUMN OF WINDOW
015D' ED 5B 237B C      LD DE,(FWINDOW)
0161' 7B C      LD A,E
0162' E6 7F C      AND 7FH ;GET FIRST WINDOW COLUMN
C
C      ;SET ADDRESS COLUMN TO FIRST COLUMN OF WINDOW
0164' B5 C      OR L ;OR IN NEW COLUMN
0165' 6F C      LD L,A
0166' C9 C      RET
C
C      PAGE

```

```

C
0167' C DO_LF:
C ;DO LINE FEED PROCESSING (IF ADDRESS IS ON LAST LINE, SCROLL SCREEN AND ADD NEW
C ;LINE AT BOTTOM)
C ;ENTRY
C ;HL = ADDRESS TO PROCESS
C
C ;EXIT
C ;HL = ADDRESS AFTER LINE FEED
C
C
C ;CHECK IF ON LAST LINE
0167' CD 03A2' C CALL LASTROW
016A' 28 05 C JR Z,XJ053 ;SCROLL IF ON LAST LINE
C
C ;MOVE DOWN ONE LINE
016C' 11 0080 C LD DE,VLL ;LINE LENGTH
016F' 19 C ADD HL,DE ;HL = NEW ADDRESS POSITION
0170' C9 C RET
C
C ;IF ON LAST LINE, SCROLL WINDOW UP
0171' E5 C XJ053: PUSH HL ;SAVE ADDRESS
0172' 2A 237B C LD HL,(FWAWIND)
0175' CD 01F8' C CALL SCROL_UP ;SCROLL WHOLE WINDOW UP
0178' E1 C POP HL ;RESTORE ADDRESS
0179' C9 C RET
C
C PAGE

```

```

C
C
C      ;      *****
C      ;      *
C      ;      *      CONTROL CHARACTER CODE PROCESSING      *
C      ;      *      EDITING FUNCTIONS                      *
C      ;      *
C      ;      *****
C
C      EEOL:
C      ;ERASE TO END OF LINE
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;CURPOS UNCHANGED
C
C      017A'  2A FDS3      LD      HL, (CURPOS)
C      017D'  C3 01BB'    JP      CLRLN      ;CLEAR TO END OF LINE
C
C      PAGE

```

'EXECUTIVE ROM MONITOR'
'CONSOLE ROUTINES'

MACRO-80 3.44 09-Dec-81

PAGE 1-115

```
0180'      C      EEOS:
           C      ;ERASE TO END OF SCREEN
           C      ;ENTRY
           C      ;NONE
           C
           C      ;EXIT
           C      ;NONE
           C
0180'  2A FD53  C      LD      HL, (CURPOS)
0183'  C3 01D5' C      JP      CLR_EOS      ;CLEAR TO END OF SCREEN
           C
           C      PAGE
```

		C			
0186'		C	EDEL:		
		C	;DELETE CHARACTER		
		C	;ENTRY		
		C	;NONE		
		C			
		C	;EXIT		
		C	;NONE		
		C			
0186'	2A FD53	C	LD	HL, (CURPOS)	
0189'	CD 04B1'	C	CALL	DIST2EOL	;Z FLAG SET IF AT END OF LINE
018C'	28 05	C	JR	Z, XJ034	;IF AT END OF LINE
		C			
018E'	E5	C	PUSH	HL	
018F'	D1	C	POP	DE	;DE = CURSOR ADDRESS
0190'	23	C	INC	HL	;HL = CURSOR ADDRESS + 1
0191'	ED B0	C	LDIR		;MOVE CHARACTERS
		C			
0193'	C3 01C8'	C	XJ034: JP	CLR_CHAR	;LAST CHARACTER BECOMES BLANK
		C			
		C		PAGE	

		C			
		C			
0196'		C	EINSRT:		
		C	;INSERT CHARACTER		
		C	;ENTRY		
		C	;NONE		
		C			
0196'	2A FD53	C	LD	HL, (CURPOS)	
0199'	CD 04B1'	C	CALL	DIST2EOL	;BC = DISTANCE TO END OF LINE
019C'	28 06	C	JR	Z, XJ035	;IF IN LAST COLUMN, JUST BLANK
		C		;CURRENT CHAR	
019E'	09	C	ADD	HL, BC	;HL = LAST POSITION IN CURSOR LINE
		C			
019F'	E5	C	PUSH	HL	
01A0'	D1	C	POP	DE	;DE = LAST POSITION IN LINE
01A1'	2B	C	DEC	HL	;HL = 2ND TO LAST POSITION
01A2'	ED B8	C	LDDR		;MOVE LINE OVER
		C			
01A4'	3E 20	C	XJ035: LD	A, 20H	
01A6'	C3 049F'	C	JP	DISPLAY	;PUT A ' ' IN CURSOR POSITION AND SET SELECTED ATTRIBUTES
		C			
		C	PAGE		

```

C
C
01A9' C ESCRR:
C ;DELETE LINE (SCROLLS LOWER LINES UP AND PUTS CURSOR AT BEGINNING OF LINE)
C ;ENTRY
C ;NONE
C
C
01A9' 2A FD53 C LD HL, (CURPOS)
01AC' CD 01F8' C CALL SCROL_UP ;DELETE LINE AND SCROLL LOWER LINES UP
C
C JP VC_CR ;PUT CURSOR AT BEGINNING OF LINE
C
C PAGE
```



```

C
C
01B2' C ESCEE:
C ;INSERT LINE (PUTS CURSOR AT BEGINNING OF INSERTED LINE)
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;NONE
C
C
01B2' 2A FDS3 C LD HL, (CURPOS)
01B5' CD 0200' C CALL SCROL_DOWN ;SCROLL LINES FROM CURSOR DOWN, CLEAR CURSOR LINE
C
01B8' C3 00D9' C JP VC_CR
01BB' C CLRNL:
C ;CLEAR TO END OF LINE
C ;ENTRY
C ;HL = START POSITION
C
C ;EXIT
C ;CLEAR TO EOL
C ; USES ALL.
C
C
C ;CLEAR START POSITION
01BB' CD 01C8' C CALL CLR_CHAR ;CLEAR CHARACTER
C
C ;FIND NUMBER OF BYTES TO CLEAR
01BE' CD 04B1' C CALL DIST2EOL ;BC = DISTANCE TO END OF LINE, Z FLAG SET
01C1' C8 C RET Z ;DONE IF AT END OF LINE
C
01C2' E5 C PUSH HL
01C3' D1 C POP DE
01C4' 13 C INC DE
01C5' ED B0 C LDIR ;CLEAR REST OF LINE
01C7' C9 C RET
C PAGE

```

		C	
		C	
01C8'		C	CLR_CHAR:
		C	;CLEAR CHARACTER FOR GIVEN POSITION
		C	;ENTRY
		C	;HL = ADDRESS
		C	
		C	;EXIT
		C	;HL PRESERVED
		C	
		C	
01C8'	E5	C	PUSH HL
		C	
01C9'	3E 20	C	LD A,20H
01CB'	77	C	LD (HL),A ;CLEAR CHARACTER
		C	
01CC'	3A 2357	C	LD A,(BACKGND)
01CF'	4F	C	LD C,A ;C = BACKGROUND ATTRIBUTES
01D0'	CD 02E8'	C	CALL SET_ATR ;SET BACKGROUND ATTRIBUTES
		C	
01D3'	E1	C	POP HL
01D4'	C9	C	RET
		C	
		C	PAGE

```

C
C
01D5' C CLR_EOS:
C ;CLEAR TO END OF WINDOW
C ;ENTRY
C ;HL = START ADDRESS
C
C ;EXIT
C ;NONE
C
C ;GET NUMBER OF ROWS TO CLEAR
01D5' CD 03B9' C CALL GETROW ;GET CURRENT ROW
01D8' 4F C LD C,A ;SAVE C = ROW
01D9' 3A 237D C LD A,(WINDROWS)
01DC' 91 C SUB C ;A = NUMBER OF ROWS TO CLEAR
C
C ;CLEAR ROWS
01DD' F5 C XJ054: PUSH AF
01DE' E5 C PUSH HL
C
01DF' CD 01BB' C CALL CLRNLN ;CLEAR TO END OF LINE
C
01E2' E1 C POP HL
01E3' CD 0159' C CALL DO_CR ;MAKE SURE ALL LINES AFTER THE 1ST ONE
C ;START AT THE FIRST COLUMN
01E6' 11 0080 C LD DE,VLL
01E9' 19 C ADD HL,DE ;HL = NEXT LINE'S ADDRESS
C
01EA' F1 C POP AF
01EB' 3D C DEC A
01EC' 20 EF C JR NZ,XJ054 ;IF MORE ROWS
C
01EE' C9 C RET
C
C PAGE

```

'EXECUTIVE ROM MONITOR'
'CONSOLE ROUTINES'

MACRO-80 3.44 09-Dec-81 PAGE 1-122

```

C
C
01EF'      C      VC_CLRS:
01EF'      C      ESCZZ:
C          C      ;CLEAR WINDOW
C          C      ;ENTRY
C          C      ;NONE
C
C          C      ;EXIT
C          C      ;HL      =      NEW CURSOR POSITION
C
01EF'  2A 237B      C          LD      HL, (FWAWIND)      ;BEGINNING OF WINDOW
01F2'  22 FD53      C          LD      (CURPOS),HL      ;POSITION CURSOR TO BEGINNING OF WINDOW
01F5'  C3 01D5'      C          JP      CLR_EOS      ;CLEAR WINDOW
C
C          C      PAGE
```

```

C
C
C ; *****
C ; *
C ; *          CONTROL CODE CHARACTER PROCESSING      *
C ; *          SCROLLING                                *
C ; *
C ; *****
C
01F8' C SCROL_UP:
C ;SCROLL UP THE LINES FROM SELECTED POSITION TO THE END OF THE WINDOW
C ;(CLEARS LAST LINE)
C ;ENTRY
C ;HL = SELECTED POSITION
C
C ;EXIT
C ;NONE
C
C ;POSITION AT BEGINNING OF LINE
01F8' 3E FF C LD A,0FFH
01FA' 32 2386 C LD (SCRL_UPF),A ;SET SCROL_UP FLAG
01FD' C3 0204' C JP S_UP_DOWN
C
C
C
C
0200' C SCROL_DOWN:
C ;SCROLL WINDOW DOWN FROM SELECTED START LINE TO END OF WINDOW
C
0200' AF C XOR A
0201' 32 2386 C LD (SCRL_UPF),A ;RESET SCROL_UP FLAG
C
C ;FALL THROUGH TO S_UP_DOWN ROUTINE
C
C PAGE

```

```

C
C
0204' C S_UP_DOWN:
C ;(CLEARS START LINE)
C ;ENTRY
C ;HL = START ADDRESS
C
C ;EXIT
C ;NONE
C
C ;POSITION AT BEGINNING OF LINE
0204' CD 0159' C CALL DO_CR ;MAKE SURE AT BEGINNING OF LINE
C
C ;SCROLL UP OR DOWN
0207' CD 04C0' C CALL DIST2LASTROW ;GET A = NUMBER OF ROWS TO SCROLL
020A' 28 33 C JR Z,XJ055 ;IF ON LAST LINE
C
020C' F5 C PUSH AF
C
020D' 3A 2386 C LD A,(SCRL_UPF) ;IS SCROLL UP ?
0210' B7 C OR A
0211' 20 07 C JR NZ,XJ056 ;YES-
0213' 3A 237D C LD A,(WINDROWS)
0216' 3D C DEC A
0217' CD 03E3' C CALL SETROW ;HL = LAST ROW'S ADDRESS
C
021A' 3A 237E C XJ056: LD A,(WINDCOLS)
021D' 4F C LD C,A
021E' 06 00 C LD B,0 ;BC = NUMBER OF COLUMNS
C
C
0220' F1 C POP AF ;RESTORE A = NUMBER OF ROWS
C
0221' C XJ057:
0221' 11 0080 C LD DE,VLL
0224' F5 C PUSH AF
0225' 3A 2386 C LD A,(SCRL_UPF) ;IS SCROLL UP ?
0228' B7 C OR A
0229' 28 05 C JR Z,XJ058 ;NO-
C
022B' F1 C POP AF
022C' EB C EX DE,HL
022D' 19 C ADD HL,DE
022E' 18 06 C JR XJ059
C
0230' F1 C XJ058: POP AF
0231' E5 C PUSH HL ;SAVE ROW ADDRESS
0232' B7 C OR A ;CLEAR CARRY
0233' ED 52 C SBC HL,DE
0235' D1 C POP DE ;HL = ROW ABOVE DE
C
0236' C5 C XJ059: PUSH BC ;SAVE # CHARS/ROW
0237' E5 C PUSH HL ;SAVE ROW ADDRESS
C
0238' ED B0 C LDIR

```

'EXECUTIVE ROM MONITOR'
'CONSOLE ROUTINES'

MACRO-80 3.44 09-Dec-81 PAGE 1-125

		C			
023A'	E1	C	POP	HL	;RESTORE
023B'	C1	C	POP	BC	
		C			
023C'	3D	C	DEC	A	
023D'	20 E2	C	JR	NZ, XJ057	;IF MORE ROWS
		C			
		C	;CLEAR START LINE		
		C			
023F'	CD 01BB'	C	XJ055:	CALL	CLRLN ;HL = START LINE'S ADDRESS
0242'	C9	C		RET	
		C			
		C		PAGE	

```
C
C
C      ; *****
C      ; *
C      ; *          CONTROL CHARACTER CODE PROCESSING      *
C      ; *          MISCELLANEOUS                          *
C      ; *
C      ; *****
C
0243'  C      VC_BEL:
C      ;RING THE BELL
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;NONE
C
0243'  21 0300  C      LD      HL,0300H          ;FREQUENCY (# MICROSECONDS IN 1/2 PERIOD)
0246'  11 0100  C      LD      DE,0100H          ;DURATION (# OF WAVES)
0249'  C3 04CA' C      JP      NOTE          ;PLAY ONE NOTE
C
C      PAGE
```



```

C
C
C      ;      ESC SEQUENCE CHARACTER PROCESSING
C
024C'      ESC_LCK:
C      ;LOCK KEYBOARD
C      ;ENTRY
C      ;NONE
C
C
024C'      AF      XOR      A      ;CLEAR A
024D'      18 02      JR      XJ033
C
024F'      ESC_ULK:
C      ;UNLOCK KEYBOARD
C      ;ENTRY
C      ;NONE
C
024F'      3E FF      LD      A,0FFH
C
0251'      32 23DD      XJ033: LD      (KEYLCK),A
0254'      C9      RET
C
C      PAGE

```

```

C
C
C      ; *****
C      ; *
C      ; *      SET / CLEAR 'CFLAG' ATTRIBUTES *
C      ; *
C      ; *****
C
C      ;SET ATTRIBUTES
C      ; CFLAG      1 1 1 1 1 1 1
C      ;             | | | | |----- GRAPHICS
C      ;             | | | | |----- ALTERNATE CHARACTER SET
C      ;             | | | |----- BLINK
C      ;             | | |----- UNDER LINE
C      ;             | |----- HALF INTENSITY
C      ;             |----- REVERSE VIDEO
C
C
C
C
C
C      0255' 06 08 C      ESCSAL: LD      B,ALT_FLG      ;START ALTERNATE CHARACTER SET ATTRIBUTE
C      0257' 18 12 C              JR      XJ036
C
C      0259' 06 40 C      ESCSHI: LD      B,HAF_FLG      ;START HALF INTENSITY ATTRIBUTE
C      025B' 18 0E C              JR      XJ036
C
C      025D' 06 04 C      ESCSGR: LD      B,GR_FLG      ;START GRAPHICS ATTRIBUTE
C      025F' 18 0A C              JR      XJ036
C
C      0261' 06 80 C      ESCSRV: LD      B,REV_FLG
C      0263' 18 06 C              JR      XJ036
C
C      0265' 06 20 C      ESCSUN: LD      B,UN_FLG
C      0267' 18 02 C              JR      XJ036
C
C      0269' 06 10 C      ESCSBL: LD      B,BNK_FLG
C
C      026B' 3A 237A C      XJ036: LD      A,(CFLAG)
C      026E' B0 C              OR      B      ;SET BIT
C      026F' 18 1A C              JR      XJ037
C
C      PAGE

```

```

C
C
C      ;CLEAR ATTRIBUTES
C
0271' 06 F7 C      ESCCAL: LD      B,NOT ALT_FLG      ;CLEAR ALTERNATE CHARACTER SET ATTRIBUTE
0273' 18 12 C          JR      XJ038
C
0275' 06 BF C      ESCEHI: LD      B,NOT HAF_FLG      ;CLEAR HALF INTENSITY ATTRIBUTE
0277' 18 0E C          JR      XJ038
C
0279' 06 FB C      ESCCBG: LD      B,NOT GR_FLG      ;CLEAR GRAPHICS ATTRIBUTE
027B' 18 0A C          JR      XJ038
C
027D' 06 7F C      ESCCRV: LD      B,NOT REV_FLG
027F' 18 06 C          JR      XJ038
C
0281' 06 DF C      ESCCUN: LD      B,NOT UN_FLG
0283' 18 02 C          JR      XJ038
C
0285' 06 EF C      ESCCBL: LD      B,NOT BNK_FLG
C
0287' 3A 237A C      XJ03B: LD      A,(CFLAG)
028A' A0 C          AND      B      ;CLEAR BIT
C
028B' 32 237A C      XJ037: LD      (CFLAG),A      ;STORE NEW ATTRIBUTE
028E' C9 C          RET
C
C      PAGE

```

```
C
C
C
028F'  C  ESCSKK:
028F'  3E 01  C      LD  A,1      ;ENABLE KEY CLICK
0291'  18 01  C      JR   XJ039
C
0293'  AF    C  ESCCKK: XOR  A      ;DISABLE KEY CLICK
0294'  32 2358 C  XJ039: LD  (KCLICK),A
0297'  C9    C      RET
C
0298'  C  ESCEFT:
0298'  3E 01  C      LD  A,1      ;ENABLE FUNCTION KEY TABLE
029A'  18 01  C      JR   XJ040
C
029C'  AF    C  ESCDFT: XOR  A      ;DISABLE FUNCTION KEY TABLE
029D'  32 23D9 C  XJ040: LD  (FKEYFLAG),A
02A0'  C9    C      RET
C
C      PAGE
```

```

C
C
02A1' C      REVON:
C      ;SET REVERSE VIDEO BACKGROUND
C      ;(LEAVES ALL OTHER BACKGROUND BITS ALONE)
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;SCREEN SET TO REVERSE VIDEO
C
02A1' 3A 2357 C      LD      A,(BACKGND)
02A4' F6 80   C      OR      REV_FLG      ;SET REVERSE VIDEO BIT
02A6' 47      C      LD      B,A          ;B = NEW BACKGROUND ATTRIBUTES
02A7' C3 0316' C      JP      SET_BACK      ;SET NEW BACKGROUND
C
C      PAGE

```

```

                                C
02AA'                          C      REVOFF:
                                C      ;CLEAR REVERSE VIDEO BACKGROUND ATTRIBUTE
                                C      ; (LEAVES ALL OTHER BACKGROUND BITS ALONE)
                                C      ;ENTRY
                                C      ;NONE
                                C
                                C      ;EXIT
                                C      ;SCREEN SET TO NORMAL VIDEO
                                C
02AA'  3A 2357                C      LD      A,(BACKGND)
02AD'  E6 7F                  C      AND     NOT REV_FLG      ;CLEAR REVERSE VIDEO BIT
02AF'  47                     C      LD      B,A              ;B = NEW BACKGROUND ATTRIBUTES
02B0'  C3 0316'               C      JP      SET_BACK        ;SET NEW BACKGROUND
                                C
                                C      PAGE
```

```
02B3'      C      NEW_BACK:
           C      ;DEFINE NEW BACKGROUND ATTRIBUTE BITS
           C
           C      ;ENTRY
           C      ;C      =      VIDEO ATTRIBUTE NUMBER (IN ASCII)
           C
           C      ;EXIT
           C      ;NONE
           C
02B3'      CD 033F' C      CALL      GET_ATR      ;GET ATTRIBUTES IN "CFLAG" FORM
           C      ;(INTERNAL NUMBERING SYSTEM INSTEAD OF TV950 SYSTEM)
02B6'      C3 0316' C      JP      SET_BACK      ;SET NEW BACKGROUND
           C      PAGE
```

```

C
02B9' C      DEFCUR:
C      ;DEFINE CURSOR TYPE
C      ;(IF SELECTION IS OUT OF RANGE, CURSOR TYPE IS UNCHANGED)
C      ;ENTRY
C      ;C      =      CURSOR ATTRIBUTE NUMBER (IN ASCII)
C      ;      0 = NO CURSOR DISPLAYED
C      ;      1 = BLINKING BLOCK
C      ;      2 = BLOCK
C      ;      3 = BLINKING UNDERLINE
C      ;      4 = UNDERLINE
C      ;      5 = BLINKING ONLY
C
C      ;EXIT
C      ;CURTYPE SET
C
02B9' 79 C      LD      A,C      ;A = CURSOR ATTRIBUTE NUMBER
02BA' D6 30 C      SUB     '0'
C
02BC' FE 01 C      CP      1
02BE' 30 04 C      JR      NC,XJ041      ;IF NOT 0
C
02C0' 06 00 C      LD      B,0      ;NO CURSOR DISPLAYED
02C2' 18 1F C      JR      XJ042
C
02C4' 20 04 C      XJ041: JR      NZ,XJ043      ;IF NOT 1
C
02C5' 06 90 C      LD      B,BNK_FLG + REV_FLG ;BLINKING BLOCK
02C8' 18 19 C      JR      XJ042
C
02CA' FE 03 C      XJ043: CP      3
02CC' 30 04 C      JR      NC,XJ044      ;IF NOT 2
C
02CE' 06 80 C      LD      B,REV_FLG ;BLOCK
02D0' 18 11 C      JR      XJ042
C
02D2' 20 04 C      XJ044: JR      NZ,XJ045      ;IF NOT 3
C
02D4' 06 30 C      LD      B,BNK_FLG+UN_FLG ;BLINKING UNDERLINE
02D6' 18 0B C      JR      XJ042
C
02D8' FE 05 C      XJ045: CP      5
02DA' 30 04 C      JR      NC,XJ046      ;IF NOT 4
C
02DC' 06 20 C      LD      B,UN_FLG ;UNDERLINE
02DE' 18 03 C      JR      XJ042
C
02E0' C0 C      XJ046: RET      NZ      ;IF UNDEFINED SELECTION, THEN DON'T CHANGE ANYTHING
C
02E1' 06 10 C      LD      B,BNK_FLG ;BLINKING ONLY
C
02E3' 78 C      XJ042: LD      A,B
02E4' 32 2356 C      LD      (CURTYPE),A ;SAVE NEW CURSOR TYPE
02E7' C9 C      RET
C

```



```

02E8'      C      SET_ATR:
           C      ;SET SELECTED ATTRIBUTES IN GIVEN MEMORY LOCATION
           C      ;ENTRY
           C      ;HL      =      LOCATION
           C      ;C      =      ATTRIBUTES (BIT 7 = REV, B6 = HF INT, B5 = UND, B4 = BLINK)
           C
           C      ;EXIT
           C      ;HL PRESERVED
           C
02E8'  E5      C      PUSH    HL
           C
02E9'  7E      C      LD      A, (HL)
02EA'  E6 7F   C      AND     7FH
02EC'  77      C      LD      (HL), A      ;CLEAR REVERSE VIDEO BIT
           C
02ED'  11 1000 C      LD      DE, MODEOFF
02F0'  19      C      ADD     HL, DE      ;HL = ATTRIBUTE LOCATION
02F1'  3E 80   C      LD      A, 80H
02F3'  77      C      LD      (HL), A      ;CLEAR ALL ATTRIBUTES
           C
02F4'  E1      C      POP     HL
02F5'  C3 02F8' C      JP      TOG_ATR      ;SET SELECTED ATTRIBUTES
           C      ; (TOGGLING 0'S IS THE SAME AS SETTING 1'S)
           C
           C      PAGE

```

```

C
02F8' C TOG_ATR:
C ;TOGGLE SELECTED ATTRIBUTES IN MEMORY LOCATION
C ;HL = LOCATION
C ;C = ATTRIBUTES TO TOGGLE (1 INDICATES TOGGLE, 0 FOR NO TOGGLE)
C ; (BIT 7 = REVERSE VIDEO, B6 = HF INT, B5 = UND, B4 = BLINK)
C
C ;EXIT
C ;HL PRESERVED
C
C
02F8' E5 C PUSH HL
C
02F9' 79 C LD A,C
02FA' E6 80 C AND REV_FLG ;GET REVERSE VIDEO BIT
02FC' AE C XOR (HL) ;IF REVERSE VIDEO IS SELECTED, TOGGLE THAT BIT
02FD' 77 C LD (HL),A
C
02FE' 11 1000 C LD DE,MODEOFF
0301' 19 C ADD HL,DE ;HL = ATTRIBUTE LOCATION
0302' 79 C LD A,C
0303' CB 27 C SLA A ;SHIFT TO PROPER POSITION
0305' AE C XOR (HL) ;TOGGLE SELECTED BITS
0306' 77 C LD (HL),A
C
0307' E1 C POP HL ;RESTORE ADDRESS
0308' C9 C RET
C
C PAGE

```

```

C
C
0309' C      TOG_CUR:
C      ;TOGGLE CURSOR
C      ;CURSOR MAY BE ANY COMBINATION OF BLOCK(REV VIDEO), UNDERLINE, AND BLINK
C      ;WHEN THE CURSOR TYPE IS CHANGED, THE FIRST TIME THROUGH THIS ROUTINE WILL ERASE
C      ;THE OLD CURSOR AND THEN SET CURTYP TO THE NEW CURSOR TYPE
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;CURSOR INVERTED
C      ;PRESERVES BC
C
0309' C5      PUSH    BC          ;SAVE
C
C      ;TOGGLE OLD CURSOR TYPE
030A' 3A 2356   LD      A, (CURTYPE) ;GET CURSOR TYPE
030D' 4F        LD      C, A
030E' 2A FD53   LD      HL, (CURPOS) ;GET CURSOR POSITION
0311' CD 02F8'  CALL    TOG_ATR      ;TOGGLE ATTRIBUTES
C
0314' C1        POP     BC          ;RESTORE
0315' C9        RET
C
C      PAGE

```

```

C
C
0316' C SET_BACK:
C ;CHANGE BACKGROUND BITS FOR WINDOW
C ;ENTRY
C ;B = NEW BACKGROUND DEFINITION
C ; (BIT 7 = REV VIDEO, B6 = HF INT, B5 = UND, B4 = BLINK, )
C ; B3 = ALTERNATE CHARACTER SET
C ;EXIT
C ;NONE
C
C ;DECIDE WHICH BITS TO TOGGLE
0316' 3A 2357 C LD A, (BACKGND)
0319' A8 C XOR B ;TOGGLE NEWLY SELECTED AND DESELECTED BITS
031A' 4F C LD C, A ;C = BITS TO TOGGLE
C
C ;SAVE NEW BACKGND FLAG
031B' 78 C LD A, B
031C' 32 2357 C LD (BACKGND), A
C
C ;TOGGLE APPROPRIATE BACKGROUND BITS FOR WHOLE WINDOW
031F' 3A 237E C LD A, (WINDCOLS)
0322' 47 C LD B, A ;B = # OF COLUMNS/ROW
0323' 2A 237B C LD HL, (FWAWIND) ;HL = FWA OF WINDOW
0326' 3A 237D C LD A, (WINDROWS) ;A = # OF ROWS
C
0329' F5 C XJ060: PUSH AF ;SAVE ROW NUMBER
032A' C5 C PUSH BC ;SAVE # OF COLUMNS/ROW
032B' E5 C PUSH HL ;SAVE ROW ADDRESS
C
032C' C5 C XJ061: PUSH BC ;SAVE ATTRIBUTE SELECTION
032D' CD 02F8' C CALL TOG_ATR
0330' C1 C POP BC
C
0331' 23 C INC HL
0332' 10 F8 C DJNZ XJ061 ;GO TO NEXT SPOT IN ROW
C
0334' E1 C POP HL ;RESTORE ROW ADDRESS
0335' 11 0000 C LD DE, VLL
0338' 19 C ADD HL, DE ;NEXT ROW'S ADDRESS
C
0339' C1 C POP BC
033A' F1 C POP AF
033B' 3D C DEC A
033C' 20 EB C JR NZ, XJ060 ;IF MORE ROWS
C
033E' C9 C RET
C
C PAGE

```

033F'

```

C
C      GET_ATR:
C      ;CONVERT SELECTED ATTRIBUTES FROM TV950 NUMBERING SYSTEM TO INTERNAL
C      ;SYSTEM
C      ;
C      ;TV950:
C      ;      (AFTER SUBTRACTING '0')
C      ;
C      ;
C      ;
C      ;
C      ;
C      ;ALTERNATE CHAR. <-----+-----+-----+-----+-----+-----+-----+-----+
C      ;UNDERLINE      <-----+-----+-----+-----+-----+-----+-----+-----+
C      ;REVERSE VIDEO  <-----+-----+-----+-----+-----+-----+-----+-----+
C      ;BLINKSED       <-----+-----+-----+-----+-----+-----+-----+-----+
C      ;HALF INTENSITY <-----+-----+-----+-----+-----+-----+-----+-----+
C
C      ;INTERNAL:
C      ;
C      ;
C      ;
C      ;
C      ;
C      ;REVERSE VIDEO <-----+-----+-----+-----+-----+-----+-----+-----+
C      ;HALF INTENSITY <-----+-----+-----+-----+-----+-----+-----+-----+
C      ;UNDERLINE <-----+-----+-----+-----+-----+-----+-----+-----+
C      ;ALTERNATE CHAR. <-----+-----+-----+-----+-----+-----+-----+-----+
C
C      ;ENTRY
C      ;C      =      VIDEO ATTRIBUTE NUMBER (TV950)
C
C      ;EXIT
C      ;B      =      INTERNAL CODE
C
C      LD      A,C
C      SUB     '0'      ;NORMALIZE ATTRIBUTE NUMBER
C      LD      C,A
C
C      LD      B,0      ;START WITH NO ATTRIBUTES SET
C
C      ;CHECK UNDERLINE BIT
C      LD      A,UNDBIT      ;UNDERLINE BIT IN VIDEO ATTRIBUTE NUMBER
C      AND     C      ;CHECK IF SET
C      JR      Z,XJ062      ;IF NOT UNDERLINE
C
C      LD      A,UN_FLG      ;IF UNDERLINE SELECTED
C      LD      B,A      ;SET UNDERLINE BIT IN B
C
C      ;CHECK REVERSE BIT
C      XJ062: LD      A,REVBIT      ;REVERSE BIT IN VIDEO ATTRIBUTE NUMBER
C      AND     C      ;CHECK IF SET
C      JR      Z,XJ063      ;IF NOT REVERSE VIDEO
C
C      LD      A,REV_FLG      ;IF REVERSE VIDEO SELECTED
C      OR      B

```

033F' 79
0340' D6 30
0342' 4F

0343' 06 00

0345' 3E 08
0347' A1
0348' 28 03

034A' 3E 20
034C' 47

034D' 3E 04
034F' A1
0350' 28 04

0352' 3E 80
0354' B0

```

0355' 47      C      LD      B,A      ;SET REVERSE VIDEO BIT IN B
C
C      ;CHECK BLINK BIT
0356' 3E 02   C      XJ063: LD      A,B_KBIT      ;BLINK BIT IN VIDEO ATTRIBUTE NUMBER
0358' A1      C      AND      C      ;CHECK IF SET
0359' 28 04   C      JR      Z,XJ064      ;IF NOT BLINK
C
035B' 3E 10   C      LD      A,B_K_FLG      ;IF BLINK SELECTED
035D' B0      C      OR      B
035E' 47      C      LD      B,A      ;SET BLINK BIT IN B
C
C      ;CHECK HALF INTENSITY BIT
035F' 3E 01   C      XJ064: LD      A,HAFBIT      ;HALF INTENSITY BIT IN VIDEO ATTRIBUTE NUMBER
0361' A1      C      AND      C      ;CHECK IF SET
0362' 28 05   C      JR      Z,XJ065      ;IF NOT HALF INTENSITY
C
0364' 3E 40   C      LD      A,HAF_FLG      ;IF HALF INTENSITY SELECTED
0366' B0      C      OR      B
0367' 47      C      LD      B,A      ;SET HALF INTENSITY BIT IN B
0368' C9      C      RET
C
0369' 3E 10   C      XJ065: LD      A,ALTBIT
036B' A1      C      AND      C
036C' C8      C      RET      Z
036D' 3E 08   C      LD      A,ALT_FLG
036F' B0      C      OR      B
0370' 47      C      LD      B,A
0371' C9      C      RET
C
C      PAGE

```

```

C
C
C      ;      *****
C      ;      *
C      ;      *      CURSOR COORDINATE PROCESSING      *
C      ;      *
C      ;      *****
C
C      YCOORD:
C      ;SET UP Y COORDINATE FOR CURSOR ADDRESSING
C      ;IF COORDINATE IS INVALID, ROW IS SET TO LAST ROW
C      ;ENTRY
C      ;C      =      CHARACTER (Y COORDINATE + OFFSET)
C
C      ;EXIT
C      ;CTBLADDR SET UP FOR X COORDINATE
C
C      0372'      CD 039B'      CALL      ABSOLUTE      ;GET ABSOLUTE COORDINATE
C
C      0375'      2A FD53      LD      HL, (CURPOS)
C      0378'      CD 03E3'      CALL      SETROW      ;SET HL TO ROW A
C      037B'      22 FD53      LD      (CURPOS),HL
C      037E'      C9      RET
C
C      PAGE

```

```

C
037F' C XCOORD:
C ;SET UP X COORDINATE. IF COORDINATE IS OUT OF RANGE, POSITION TO COLUMN 0
C ;ENTRY
C ;C = CHARACTER (COORDINATE + OFFSET)
C
C ;EXIT
C ;CTBLADDR RESET TO BASTBL, CURPOS SET TO NEW CURSOR POSITION
C
C
037F' CD 039B' C CALL ABSOLUTE ;GET ABSOLUTE COORDINATE VALUE
0382' 2A FD53 C LD HL, (CURPOS)
0385' CD 03CB' C CALL SETCOL ;SET COLUMN
0388' 22 FD53 C LD (CURPOS), HL
038B' C9 C RET
C
C PAGE
```


		C	
038C'		C	SETOFF:
		C	;SET X-Y COORDINATE OFFSET
		C	;ENTRY
		C	;C = OFFSET
		C	
		C	;EXIT
		C	;OFFSET UPDATED
		C	
		C	
038C'	79	C	LD A,C
038D'	32 2379	C	LD (OFFSET),A
0390'	C9	C	RET
		C	
		C	PAGE

```

C
C
0391' C ADDROWS:
C      ;ADD ROWS TO GIVEN ADDRESS
C      ;ENTRY
C      ;HL  =    START ADDRESS
C      ;A   =    NUMBER OF ROWS
C
C      ;*NOTE* THIS ROUTINE DOES NOT CHECK FOR ADDRESS OUT OF RANGE
C      ;        THE CALLING PROCEDURE MUST DO THAT
C
C      ;EXIT
C      ;HL  =    NEW ADDRESS
C
C      ;CHECK FOR ROW 0
0391' B7 C      OR    A          ;A = ROW NUMBER
0392' C8 C      RET    Z          ;DONE IF ROW 0
C
C      ;ADD NUMBER OF ROWS TO WINDOW BEGINNING ADDRESS
0393' 11 0080 C      LD    DE,VLL      ;DISTANCE TO NEXT ROW
0396' 19 C      XJ069: ADD    HL,DE      ;ADD A ROW
0397' 3D C      DEC    A
0398' 20 FC C      JR     NZ,XJ069      ;IF MORE ROWS
039A' C9 C      RET
C
C      PAGE

```

```
039B'      C      ABSOLUTE:
           C      ;GET ABSOLUTE VALUE FOR OFFSET NUMBER
           C      ;ENTRY
           C      ;C      =      NUMBER + OFFSET
           C
           C      ;EXIT
           C      ;A      =      ABSOLUTE NUMBER (WITHOUT OFFSET)
           C
039B' 3A 2379 C      LD      A, (OFFSET)      ;GET CURSOR POSITIONING OFFSET
039E' ED 44   C      NEG
03A0' 81      C      ADD      A, C      ;A = ABSOLUTE CURSOR POSITION
03A1' C9      C      RET
           C
           C      PAGE
```

```

C
C
03A2' C LASTROW:
C ;CHECK IF GIVEN ADDRESS IS ON LAST LINE
C ;ENTRY
C ;HL = ADDRESS
C
C ;EXIT
C ;Z BIT SET IF ON LAST LINE
C ;HL PRESERVED
C
03A2' CD 03B9' C CALL GETROW ;GET CURRENT ROW
03A5' 47 C LD B,A ;B = CURRENT ROW
03A6' 3A 237D C LD A,(WINDROWS)
03A9' 3D C DEC A ;A = LAST ROW
03AA' 90 C SUB B ;SET Z FLAG IF CURRENT ROW = LAST ROW
03AB' C9 C RET
C
C PAGE

```

```

C
03AC' C      GETCOL:
C      ;GET COLUMN NUMBER OF SELECTED ADDRESS
C      ;(IF ADDRESS IS OUT OF RANGE AN INVALID NUMBER IS RETURNED)
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;A      =      COLUMN NUMBER
C      ;PRESERVES HL
C
C      ;GET ACTUAL COLUMN NUMBER FOR FIRST COLUMN OF WINDOW
03AC' ED 5B 237B C      LD      DE, (FWAWIND)
03B0' 7B C      LD      A, E
03B1' E6 7F C      AND     7FH
03B3' 4F C      LD      C, A      ;C = FIRST COLUMN NUMBER
C
C      ;GET ACTUAL COLUMN NUMBER FOR SELECTED ADDRESS
03B4' 7D C      LD      A, L      ;LOW 7 BITS OF L = COLUMN
03B5' E6 7F C      AND     7FH      ;CLEAR HIGH BIT
C
C      ;GET WINDOW COLUMN NUMBER FOR SELECTED ADDRESS
03B7' 91 C      SUB     C
03B8' C9 C      RET
C
C      PAGE

```

```

C
03B9' C GETROW:
C ;GET CURRENT ROW OF SELECTED ADDRESS
C ; (INVALID NUMBER RETURNED IF ADDRESS IS OUT OF RANGE)
C ;ENTRY
C ;HL = ADDRESS
C
C ;EXIT
C ;A = ROW NUMBER
C ;PRESERVES HL
C
C ;GET ACTUAL ROW VALUE OF FIRST ROW IN WINDOW
03B9' E5 C PUSH HL
03BA' 2A 237B C LD HL, (FWAWIND)
03BD' 29 C ADD HL, HL ;SHIFT ROW BITS TO H
03BE' 7C C LD A, H
03BF' E6 1F C AND 1FH ;GET ROW BITS
03C1' 4F C LD C, A
03C2' E1 C POP HL
C
C ;GET ACTUAL ROW OF SELECTED ADDRESS
03C3' E5 C PUSH HL
03C4' 29 C ADD HL, HL ;SHIFT ROW BITS TO H
03C5' 7C C LD A, H
03C6' E6 1F C AND 1FH ;GET ROW BITS
03C8' E1 C POP HL
C
C ;GET WINDOW CURSOR ROW
03C9' 91 C SUB C
03CA' C9 C RET
C
C PAGE

```

```

03CB'      C      SETCOL:
C          ;SET HL TO GIVEN COLUMN, CURRENT ROW
C          ;(IF COLUMN IS OUT OF RANGE, SET TO COLUMN 0)
C          ;ENTRY
C          ;HL      =      SELECTED COLUMN
C          ;A        =      COLUMN NUMBER
C
C          ;EXIT
C          ;HL      =      SET TO SELECTED COLUMN
C
C
C          ;CHECK FOR VALID COLUMN
03CB'  4F      C          LD      C,A          ;SAVE COLUMN
03CC'  3A 237E C          LD      A,(WINDCOLS)
03CF'  3D      C          DEC      A          ;A = LAST COLUMN IN WINDOW
03D0'  B9      C          CP       C          ;SET CFLAG IF POSITION OUT OF RANGE
03D1'  79      C          LD      A,C
03D2'  30 01   C          JR       NC,XJ066          ;IF IN RANGE
C
C          XOR      A          ;IF OUT OF RANGE
03D4'  AF      C          ;SET COLUMN
C          ;GET ACTUAL COLUMN VALUE FOR WINDOW'S COLUMN 0
03D5'  ED 5B 237B C      XJ066: LD      DE,(FWAWIND) ;START OF WINDOW
03D9'  83      C          ADD      A,E
03DA'  E6 7F   C          AND      7FH          ;A = ACTUAL COLUMN VALUE
03DC'  4F      C          LD      C,A          ;SAVE COLUMN VALUE
C
C          ;SET COLUMN FOR SELECTED ADDRESS
03DD'  7D      C          LD      A,L
03DE'  E6 80   C          AND      80H          ;CLEAR COLUMN BITS
03E0'  B1      C          OR       C          ;PUT IN NEW COLUMN BITS
03E1'  6F      C          LD      L,A
03E2'  C9      C          RET
C
C          PAGE

```

```

C
C
03E3' C SETROW:
C ;SET ADDRESS TO GIVEN ROW, CURRENT COLUMN
C ;(IF ROW IS OUT OF RANGE, ROW SET TO LAST ROW)
C ;ENTRY
C ;A = ROW NUMBER
C
C ;EXIT
C ;HL = ADDRESS OF GIVEN ROW, COLUMN 0
C
C ;GET CURRENT COLUMN
03E3' 4F LD C, A
03E4' C5 PUSH BC ;SAVE ROW
C
03E5' CD 03AC' CALL GETCOL
03E8' C1 POP BC ;RESTORE ROW
03E9' F5 PUSH AF ;SAVE COLUMN
C
C ;CHECK FOR VALID ROW
03EA' 3A 237D LD A, (WINDROWS)
03ED' 3D DEC A ;A = LAST VALID ROW NUMBER
03EE' B9 CP C
03EF' 38 01 JR C, XJ067 ;IF INVALID
C
03F1' 79 LD A, C ;IF VALID
C
03F2' 2A 237B XJ067: LD HL, (FWAWIND) ;START AT TOP OF WINDOW
03F5' CD 0391' CALL ADDROWS ;AND ADD A ROWS
C
03F8' F1 XJ068: POP AF ;RESTORE COLUMN
03F9' C3 03CB' JP SETCOL
C
C PAGE

```



```

C
C
C      ; *****
C      ; *
C      ; *          WINDOW PROCESSING          *
C      ; *
C      ; *****
C
03FC'  C      DEF_WINDOW:
C      ;DEFINE AND SET WINDOW.
C      ;THE DEFINE WINDOW ESCAPE SEQUENCE SPECIFIES A WINDOW NUMBER AND DEFINES THE
C      ;UPPER LEFT AND LOWER RIGHT CORNERS. THEN THE CURSOR IS POSITIONED TO THE HOME
C      ;POSITION OF THE WINDOW.  ONCE A WINDOW HAS BEEN DEFINED, IT CAN BE SET AGAIN
C      ;BY JUST USING THE SET WINDOW COMMAND.
C      ;(WINDOW NUMBER CAN BE 1-15.  WINDOW 0 IS RESERVED AND IS ALWAYS DEFINED AS THE FULL SCREEN)
C      ;ENTRY
C      ;C      =      WINDOW NUMBER (IN ASCII)
C
C      ;EXIT
C      ;WINDNUM      =      WINDOW NUMBER
C
03FC'  79      C      LD      A,C
03FD'  D6 30    C      SUB     '0'      ;NORMALIZE
C
03FF'  E6 0F    C      AND     0FH      ;GET NUMBER MOD 16
0401'  32 2380  C      LD      (WINDNUM),A  ;SAVE WINDOW NUMBER
0404'  C9      C      RET
C
C      PAGE

```

```

0405'      C      STARTROW:
C          C      ;DEFINE FIRST ROW OF WINDOW
C          C      ;(IF ROW IS OUT OF RANGE, SET TO ROW 0)
C          C      ;ENTRY
C          C      ;C      =      ROW NUMBER + OFFSET
C          C
C          C      ;EXIT
C          C      ;TEMPFWA      =      ADDRESS SO FAR
C          C
C          C      ;CHECK FOR VALID ROW NUMBER
0405'  79      C          LD      A,C
0406'  CD 039B' C          CALL   ABSOLUTE      ;GET ABSOLUTE NUMBER (WITHOUT OFFSET)
0409'  06 18      C          LD      B,VMROWS
040B'  B8          C          CP      B
040C'  38 01      C          JR      C,XJ047      ;IF VALID ROW, A = SELECTED ROW
C          C
040E'  AF          C          XOR      A          ;IF INVALID, A = ROW 0
C          C
C          C      ;GET ROW ADDRESS
040F'  32 2384      C      XJ047: LD      (FSTROW),A      ;SAVE ROW NUMBER
0412'  21 C000      C          LD      HL,FHVM          ;START OF VIDEO MEMORY
0415'  CD 0391'      C          CALL   ADDRWS          ;GET NEW ADDRESS
C          C
0418'  22 2381      C          LD      (TEMPFWA),HL      ;SAVE IN TEMPORARY FWA FOR WINDOW
041B'  C9          C          RET
C          C
C          C      PAGE

```

```

041C'      C      STARTCOL:
           C      ;DEFINE FIRST COLUMN OF WINDOW
           C      ; (INVALID COLUMNS ARE SET TO COLUMN 0)
           C      ;ENTRY
           C      ;C      =      COLUMN NUMBER + OFFSET
           C      ;TEMPFWA SET TO SELECTED ROW, COLUMN 0
           C
           C      ;EXIT
           C      ;TEMPFWA      =      FWA FOR WINDOW
           C
           C
           C      ;CHECK FOR VALID COLUMN
041C'  79      C      LD      A,C      ;A = SELECTED COLUMN + OFFSET
041D'  CD 039B' C      CALL   ABSOLUTE ;A = SELECTED COLUMN
0420'  06 50      C      LD      B,VMCOLS ;B = NUMBER OF COLUMNS IN VIDEO MEMORY
0422'  B8          C      CP      B
0423'  38 01      C      JR      C,XJ040 ;IF VALID, A = SELECTED COLUMN NUMBER
           C
0425'  AF          C      XOR      A      ;IF INVALID, A = 0
           C
           C      ;SET COLUMN IN TEMPFWA
0426'  32 2385      C      XJ040: LD      (FSTCOL),A ;SAVE FIRST COLUMN NUMBER
0429'  2A 2381      C      LD      HL,(TEMPFWA) ;TEMPFWA = SELECTED ROW, COLUMN 0
042C'  5F          C      LD      E,A
042D'  16 00      C      LD      D,0
042F'  19          C      ADD     HL,DE
0430'  22 2381      C      LD      (TEMPFWA),HL
0433'  C9          C      RET
           C
           C      PAGE

```

```

0434' 79          C      ENDROW:
0435' CD 039B'   C      ;DEFINE LAST ROW OF WINDOW
0438' 06 18      C      ;(IF INVALID ROW SELECTED, ROW SET TO LAST ROW)
043A' B8         C      ;ENTRY
043B' 38 02      C      ;C      =      ROW NUMBER + OFFSET
043D' 05         C      ;EXIT
043E' 78         C      ;TEMPROWS      =      # OF ROWS IN NEW WINDOW
043F' 21 2384    C      ;CHECK FOR VALID ROW
0442' 46         C      LD      A,C          ;A = SELECTED ROW + OFFSET
0443' 90         C      CALL     ABSOLUTE    ;A = SELECTED ROW
0444' 30 01      C      LD      B,VMROWS      ;B = NUMBER OF ROWS
0446' AF         C      CP      B
0447' 3C         C      JR      C,XJ049      ;IF VALID, A = SELECTED ROW
0448' 32 2383    C      DEC     B
044B' C9         C      LD      A,B          ;IF INVALID, A = LAST ROW
044C'          C      ;CALCULATE NUMBER OF ROWS
044D'          C      XJ049: LD      HL,FSTROW
044E'          C      LD      B,(HL)        ;B = FIRST ROW NUMBER
044F'          C      ;A = LAST ROW NUMBER
0450'          C      SUB     B
0451'          C      JR      NC,XJ050      ;IF VALID...
0452'          C      ;
0453'          C      XOR      A          ;INVALID, FORCE IT TO 1...
0454'          C      XJ050: INC     A          ;A = NUMBER OF ROWS
0455'          C      LD      (TEMPROWS),A  ;SAVE NUMBER OF ROWS
0456'          C      RET
0457'          C      PAGE

```

```

044C'      C      ENDCOL:
C          C      ;DEFINE LAST COLUMN FOR WINDOW AND SAVE WINDOW DEFINITION
C          C      ;(IF COLUMN OUT OF RANGE, SET TO LAST COLUMN)
C          C      ;ENTRY
C          C      ;C      =      COLUMN NUMBER + OFFSET
C          C
C          C      ;EXIT
C          C      ;NEW WINDOW DEFINITIONS SET UP
C          C
C          C      ;CHECK FOR VALID COLUMN
044C'  79      C      LD      A,C      ;A = SELECTED COLUMN + OFFSET
044D'  CD 039B' C      CALL   ABSOLUTE ;A = SELECTED COLUMN
0450'  06 50    C      LD      B,VMCOLS ;B = NUMBER OF COLUMNS IN SCREEN
0452'  B8      C      CP      B
0453'  38 02    C      JR      C,XJ051 ;IF VALID, A = SELECTED COLUMN
C
0455'  05      C      DEC     B
0456'  78      C      LD      A,B      ;A = LAST COLUMN
C
C          C      ;GET NUMBER OF COLUMNS
0457'  21 2385 C      XJ051: LD      HL,FSTCOL
045A'  46      C      LD      B,(HL) ;B = FIRST COLUMN
C          C      ;A = LAST COLUMN
045B'  90      C      SUB     B
045C'  30 01    C      JR      NC,XJ052 ;IF LAST AFTER FIRST...
C          C      ;
045E'  AF      C      XOR      A ;LAST COL IS BEFORE FIRST,
C          C      ;FORCE TO 1.
C
045F'  3C      C      XJ052: INC     A ;A = NUMBER OF COLUMNS
C
C          C      ;SAVE NEW WINDOW DEFINITION
0460'  47      C      LD      B,A ;SAVE NUMBER OF COLUMNS
C
0461'  3A 2380 C      LD      A,(WINDNUM) ;A = WINDOW NUMBER THAT IS BEING DEFINED
0464'  FE 00    C      CP      RESERVED
0466'  C8      C      RET      Z ;DON'T DEFINE THE RESERVED WINDOW
C
0467'  CD 0491' C      CALL   GETWIND ;GET HL = ADDRESS OF WINDOW DEFINITION
046A'  ED 5B 2381 C      LD      DE,(TEMPFWA)
046E'  73      C      LD      (HL),E ;SAVE NEW FWA
046F'  23      C      INC     HL
0470'  72      C      LD      (HL),D
0471'  23      C      INC     HL
0472'  3A 2383 C      LD      A,(TEMPROWS)
0475'  77      C      LD      (HL),A ;SAVE NEW NUMBER OF ROWS
C
0476'  23      C      INC     HL
0477'  70      C      LD      (HL),B ;SAVE NEW NUMBER OF ROWS
C
C          C      ;SET WINDOW
0478'  3A 2380 C      LD      A,(WINDNUM)
047B'  C6 30    C      ADD     A,'0'
047D'  4F      C      LD      C,A ;C = ASCII VALUE FOR WINDOW

```

'EXECUTIVE ROM MONITOR'
'CONSOLE ROUTINES'

MACRO-80 3.44 09-Dec-81 PAGE 1-156

C
C ;FALL THROUGH TO SETWIND
C PAGE

```

C
C
047E' C      SETWIND:
C      ;SET WINDOW TO SELECTED WINDOW NUMBER
C      ;ENTRY
C      ;C      =      WINDOW NUMBER (IN ASCII)
C
C      ;EXIT
C      ;CURRENT WINDOW SET TO WINDOW NUMBER
C      ;CURSOR POSITIONED TO HOME POSITION OF NEW WINDOW
C
047E' 79 C      LD      A,C
047F' D6 30 C      SUB     '0'
0481' E6 0F C      AND     0FH      ;GET NUMBER MOD 16
0483' CD 0491' C      CALL    GETWIND  ;GET HL = ADDRESS OF WINDOW DEFINITION
C
0486' 11 237B C      LD      DE,CURWIND  ;CURRENT WINDOW DEFINITION
0489' 01 0004 C      LD      BC,4
048C' ED 80 C      LDIR     ;MOVE SELECTED WINDOW TO CURRENT WINDOW
C
048E' C3 0152' C      JP      VC_HOME  ;INITIALIZE CURSOR TO HOME POSITION OF WINDOW
C
C      PAGE

```

```

C
0491' C GETWIND:
C ;GET ADDRESS OF GIVEN WINDOW DEFINITION
C ;ENTRY
C ;A = WINDOW NUMBER
C
C ;EXIT
C ;HL = ADDRESS OF WINDOW DEFINITION
C ;BC PRESERVED
C
0491' 21 2387 C LD HL,WINDEFS ;START ADDRESS OF WINDOW DEFINITIONS
0494' 87 C ADD A,A ;A * 2
0495' 87 C ADD A,A ;A * 4
0496' 5F C LD E,A
0497' 16 00 C LD D,0 ;DE = WINDOW NUMBER * 4 (EACH DEFINITION IS 4 BYTES)
0499' 19 C ADD HL,DE ;HL = ADDRESS OF WINDOW DEFINITION
049A' C9 C RET
C
C PAGE
```



```

C
C
C      ; *****
C      ; *
C      ; * MISCELLANEOUS CONSOLE ROUTINES *
C      ; *
C      ; *****
C
049B' C SEQERR:
C      ;HANDLE ILLEGAL SEQUENCE (EG UNDEFINED ESC SEQUENCE)
C      ;IGNORE SEQUENCE, SET TABLE ADDRESS BACK TO BASTBL AND PROCESS CURRENT CHARACTER NORMALLY
C      ;ENTRY
C      ;C = CHARACTER
C
C      ;EXIT
C      ;NONE
C
049B' C3 007B' C JP LOOKUPB ;PROCESS CHARACTER NORMALLY
C
C
049E' C RETPROC:
C      ;PROCEDURE WHICH ONLY RETURNS
C      ;USED FOR BRANCH ADDRESSES FOR CONSOLE TABLE ENTRIES WHICH DON'T NEED TO
C      ;DO ANYTHING
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;NONE
C
049E' C9 C RET
C
C PAGE

```

```

049F'      C      DISPLAY:
          C      ;DISPLAY CHARACTER AT CURSOR POSITION
          C      ;ENTRY
          C      ;A      =      CHARACTER
          C
          C      ;EXIT
          C      ;NONE
          C
049F'  2A FD53      C      LD      HL, (CURPOS)      ;HL = CURRENT CURSOR POSITION
          C
          C      ;DISPLAY NEW CHARACTER
          C
04A2'  77          C      LD      (HL), A      ;STORE CHARACTER
          C
          C      ;GET SELECTED ATTRIBUTES
04A3'  3A 2357      C      LD      A, (BACKGND)      ;A = BACKGROUND ATTRIBUTES
04A6'  E6 F8          C      AND     0F8H      ;MAKE SURE ONLY BACKGROUND ATTRIBUTE BITS ARE SET
04A8'  4F          C      LD      C, A      ;C = BACKGROUND ATTRIBUTES
          C
04A9'  3A 237A      C      LD      A, (CFLAG)      ;A = CURRENT ESC SEQUENCE ATTRIBUTES
04AC'  A9          C      XOR     C      ;GET NET ATTRIBUTES
04AD'  4F          C      LD      C, A      ;C = SELECTED ATTRIBUTES
          C
04AE'  C3 02E8'      C      JP      SET_ATR      ;STORE ATTRIBUTES
          C
          C      PAGE

```

		C			
		C			
		C			
04B1'		C	DIST2EOL:		
		C	;CALCULATE #CHRS TO BETWEEN GIVEN ADDRESS AND LAST COLUMN IN LINE		
		C	;ENTRY		
		C	;HL = ADDRESS		
		C			
		C	;EXIT		
		C	;BC = # CHARACTERS BETWEEN ADDRESS AND LAST COLUMN		
		C	;A = # CHARACTERS BETWEEN ADDRESS AND LAST COLUMN		
		C	;Z BIT SET IF 0 CHARACTERS (IE ADDRESS IS AT END OF LINE)		
		C	;HL SAVED		
		C			
04B1'	E5	C	PUSH HL		
04B2'	CD 03AC'	C	CALL GETCOL	;GET COLUMN	
04B5'	4F	C	LD C,A	;C = COLUMN	
04B6'	3A 237E	C	LD A,(WINDCOLS)	;A = LINE LENGTH	
04B9'	3D	C	DEC A	;A = LAST COLUMN IN LINE	
04BA'	91	C	SUB C	;A = #CHRS BETWEEN ADDRESS AND LAST COLUMN	
04BB'	4F	C	LD C,A		
04BC'	06 00	C	LD B,0	;BC = #CHRS BETWEEN ADDRESS AND LAST COLUMN	
04BE'	E1	C	POP HL		
04BF'	C9	C	RET		
		C			
		C	PAGE		

```

C
C
04C0' C DIST2LASTROW:
C ;CALCULATE NUMBER OF ROWS BETWEEN GIVEN POSITION AND LAST LINE OF VIDEO
C
C ;ENTRY
C ;HL = GIVEN POSITION
C
C ;EXIT
C ;A = # OF ROWS FROM GIVEN POSITION TO BEGINNING OF LAST LINE
C ;FLAGS SET: Z IF GIVEN POSITION ON LAST LINE
C ; C IF GIVEN POSITION PAST LAST LINE
C
C
04C0' CD 0389' C CALL GETROW
04C3' 4F C LD C,A ;C = CURSOR ROW
C
04C4' 3A 237D C LD A,(WINDROWS)
04C7' 3D C DEC A ;A = LAST ROW
04C8' 91 C SUB C
04C9' C9 C RET ;A = # OF ROWS BETWEEN CURSOR AND LAST ROW
C
C PAGE

```

```

C
C
C      ; *****
C      ; *                                     *
C      ; *             MISCELLANEOUS ROUTINES             *
C      ; *                                     *
C      ; *****
C
04CA'  NOTE:
C      ;PLAY NOTE
C      ;ENTRY
C      ;HL  =   FREQUENCY (TIME FOR 1/2 PERIOD (IN MICROSEC))
C      ;DE  =   DURATION  (NUMBER OF WAVES)
C
C      ;EXIT
C      ;NOTE PLAYED
C
C      ;DISABLE 60HZ INTERRUPT
04CA'  CD 0E75'  C      CALL  GET_INT      ;GET INTERRUPT STATE
04CD'  C5      C      PUSH  BC          ;SAVE IT
04CE'  78      C      LD    A,B        ;MOVE CURRENT STATE TO A
04CF'  E6 EF   C      AND    0EFH      ;DISABLE 60HZ INTERRUPT
04D1'  47      C      LD    B,A
04D2'  CD 0E7A' C      CALL  SET_INT      ;DISABLE 60HZ INTERRUPT
C
C      ;PLAY NOTE
04D5'  01 FFFA C      LD    BC,-6          ;# OF MICROSECONDS IN HI/LOW LOOPS
C
04D8'  DB 02   C      LOOP: IN  A,(SYS_DTB) ;[3.75]
04DA'  F6 08   C      OR    SPEAKER      ;[1.75] SET OUTPUT TO +5V
04DC'  D3 02   C      OUT   (SYS_DTB),A  ;[3.50]
C
04DE'  E5      C      PUSH  HL          ;[2.75] SAVE DURATION (# OF MICROSECS IN 1/2 PERIOD)
C
04DF'  09      C      XJ070: ADD  HL,BC      ;[2.75] SUBTRACT # OF MICROSECS IN LOOP
04E0'  38 FD   C      JR    C,XJ070      ;[3.00] LOOP AGAIN IF MORE MICROSECS LEFT
C
04E2'  E1      C      POP   HL          ;[2.50] RESTORE DURATION (# OF MICROSECS IN 1/2 PERIOD)
C
04E3'  DB 02   C      IN    A,(SYS_DTB) ;[3.75]
04E5'  E6 F7   C      AND    NOT SPEAKER ;[1.75] SET OUTPUT TO 0V
04E7'  D3 02   C      OUT   (SYS_DTB),A  ;[3.50]
C
04E9'  E5      C      PUSH  HL          ;[2.75] SAVE DURATION
C
04EA'  09      C      XJ071: ADD  HL,BC      ;[2.75] SUBTRACT # OF MICROSECS IN LOOP
04EB'  38 FD   C      JR    C,XJ071      ;[3.00] LOOP AGAIN IF MORE MICROSECS LEFT
C
04ED'  E1      C      POP   HL          ;[2.50] RESTORE DURATION
C
04EE'  1B      C      DEC    DE          ;[1.50]
04EF'  7A      C      LD    A,D          ;[1.00]
04F0'  B3      C      OR    E            ;[1.00]
04F1'  20 E5   C      JR    NZ,LOOP      ;[3.00] LOOP IF MORE WAVES

```

```

C
C      ;RESTORE 60HZ INTERRUPT
04F3'  C1      POP      BC      ;RESTORE OLD INTERRUPT STATE
04F4'  C3 0E7A' JP      SET_INT
C
C      PAGE
C
```

```

C
C
C      ; *****
C      ; *
C      ; *          CONSOLE ROUTINES CONSTANTS          *
C      ; *
C      ; *****
C
C      ;BIT DEFINITIONS FOR CFLAG FLAG BYTE
C
C
C      ;
C      ;
C      ;
C      ;
C      ;REVERSE VIDEO (-----+ | | | | | | |
C      ;HALF INTENSITY (-----+ | | | | | |
C      ;UNDERLINE (-----+ | | | | |
C      ;BLINK VIDEO (-----+ | | | |
C      ;
C      ;ALTERNATE CHARACTER SET (-----+ | | |
C      ;GRAPHICS (-----+ | |
C      ;NOT USED (-----+ |
C      ;NOT USED (-----+
C
C
C      0080      REV_FLG EQU      80H          ;REVERSE VIDEO MODE
C      0040      HAF_FLG EQU      40H          ;HALF INTENSITY MODE
C      0020      UN_FLG EQU      20H          ;UNDERLINE MODE
C      0010      BNK_FLG EQU      10H          ;BLINK VIDEO MODE
C      0008      ALT_FLG EQU      08H          ;ALTERNATE CHARACTER SET
C      0004      GR_FLG EQU      04H          ;GRAPHICS MODE
C      00FC      ATR_MSK EQU      0FCH          ;MASK TO GET ATTRIBUTES
C      000F      CLATR_MSK EQU      0FH          ;MASK TO CLEAR TOP 4 ATTRIBUTES
C
C      PAGE

```

```

C
C
C      ; *****
C      ; *
C      ; *          CONSOLE ROUTINES VARIABLES          *
C      ; *
C      ; *****
C
C
C      ;CONSOLE TABLES
C
04F7'  C      BASET:
C      ;BASE TABLE -- USED TO LOOK UP THE FIRST CHARACTER IN A SEQUENCE
C      ;THE TABLE LISTS CHARACTERS AND PROCEDURES TO BRANCH TO TO PROCESS THE CHARACTER
C      ;1ST BYTE OF TABLE IS NUMBER OF ENTRIES
C      ;ENTRIES ARE 4 BYTES LONG:
C      ;      1ST BYTE = MATCH CODE
C      ;      2ND BYTE = NEXT TABLE NUMBER
C      ;      3RD, 4TH BYTES = BRANCH ADDRESS
C      ;THE LAST ENTRY IS A 3-BYTE "NO MATCH" DEFAULT NEXT TABLE AND BRANCH ADDRESS
C
C
C      04F7'  01      DB          BASSIZ                      ;NUMBER OF ENTRIES IN BASE TABLE
C      ;      +-----+
C      ;      | CHARACTER | NEXT TABLE | BRANCH ADDR |
C      ;      +-----+
C
C      04F8'  1B      DB          ESC
C      04F9'  02      DB ESCNUM
C      04FA'  049E'    DW RETPROC          ;ESC SEQUENCE
C      04FC'  01      JX01: DB BASNUM
C      04FD'  00AC'    DW PROCNORM          ;NO MATCH
C      ;-- PROCESS NORMAL 1 CHARACTER LONG "SEQUENCE"
C
C
C      0001      C      BASSIZ EQU      (JX01-BASET + 1)/4      ;NUMBER OF ENTRIES IN BAST
C
C      PAGE

```



```

C
C
04FF' C
C      ESCT:
C      ;VALID ESC-SEQUENCE TABLE
C      ;1ST BYTE IS NUMBER OF ENTRIES
C      ;4 BYTES PER ENTRY: ASCII CHAR, NEXT TABLE NUMBER, BRANCH ADDRESS
C      ;FOLLOWING BODY OF TABLE IS 3 BYTE NO-MATCH DEFAULT NEXT TABLE AND BRANCH ADDR
C
C
C
C
04FF' 21 C
C      DB          ESCSIZ                                ;NUMBER OF BYTES
C      ;
C      ; +-----+
C      ; | CHARACTER | NEXT TABLE | BRANCH ADDR |
C      ; +-----+
C      ;
0500' 61 C      DB          VSAL
0501' 01 C      DB BASNUM
0502' 0255' C      DW      ESCSAL      ;SET ALTERNATE CHARACTER SET
0504' 41 C      DB          VEAL
0505' 01 C      DB BASNUM
0506' 0271' C      DW      ESCCAL      ;CLEAR ALTERNATE CHARACTER SET
0508' 29 C      DB          VSHI
0509' 01 C      DB BASNUM
050A' 0259' C      DW      ESCSHI      ;SET HALF INTENSITY MODE
050C' 28 C      DB          VEHI
050D' 01 C      DB BASNUM
050E' 0275' C      DW      ESCCHI      ;CLEAR HALF INTENSITY MODE
0510' 67 C      DB          VSGH
0511' 01 C      DB BASNUM
0512' 025D' C      DW      ESCSGR      ;SET GRAPHICS MODE
0514' 47 C      DB          VEGH
0515' 01 C      DB BASNUM
0516' 0279' C      DW      ESCCGR      ;CLR GRAPHICS MODE
0518' 3D C      DB          VCAD
0519' 04 C      DB YNUM
051A' 049E' C      DW      RETPROC      ;CURSOR ADDRESSING
C
051C' 5A C      DB          'Z'
051D' 01 C      DB BASNUM
051E' 01EF' C      DW      ESCZZ      ;CLEAR SCREEN TO BLANKS
0520' 51 C      DB          VINC
0521' 01 C      DB BASNUM
0522' 0196' C      DW      EINSRT      ;INSERT CHAR
0524' 57 C      DB          VDEL
0525' 01 C      DB BASNUM
0526' 0186' C      DW      EDEL      ;DELETE CHAR
0528' 45 C      DB          VINL
0529' 01 C      DB BASNUM
052A' 01B2' C      DW      ESCCE      ;INSERT LINE
052C' 52 C      DB          VDELL
052D' 01 C      DB BASNUM
052E' 01A9' C      DW      ESCRR      ;DELETE LINE
0530' 54 C      DB          VCEOL
0531' 01 C      DB BASNUM
0532' 017A' C      DW      EEOL      ;CLEAR TO END OF LINE
0534' 23 C      DB          VLOCK

```

0535'	01	C	DB	BASNUM	
0536'	024C'	C	DW	ESC_LCK	;LOCK KEYBOARD
0538'	22	C	DB		VUNLK
0539'	01	C	DB	BASNUM	
053A'	024F'	C	DW	ESC_ULK	;UNLOCK KEYBOARD
053C'	59	C	DB		VCEOS
053D'	01	C	DB	BASNUM	
053E'	0180'	C	DW	EEOS	;CLEAR TO END OF SCREEN (WINDOW)
		C			
0540'	7A	C	DB		VWDEF
0541'	06	C	DB	WDEFNUM	
0542'	049E'	C	DW	RETPROC	;DEFINE A WINDOW
0544'	73	C	DB		VWSET
0545'	0B	C	DB	WSETNUM	
0546'	049E'	C	DW	RETPROC	;SET WINDOW
0548'	2E	C	DB		VCTYP
0549'	0C	C	DB	CTYPNUM	
054A'	049E'	C	DW	RETPROC	;DEFINE CURSOR TYPE
054C'	6F	C	DB		VOFF
054D'	0D	C	DB	OFFNUM	
054E'	049E'	C	DW	RETPROC	;DEFINE X-Y OFFSET
0550'	62	C	DB		VRON
0551'	01	C	DB	BASNUM	
0552'	02A1'	C	DW	REVON	;SET REVERSE VIDEO BACKGROUND
0554'	64	C	DB		VROFF
0555'	01	C	DB	BASNUM	
0556'	02AA'	C	DW	REVOFF	;SET NORMAL VIDEO BACKGROUND
0558'	7B	C	DB		VBACK
0559'	0E	C	DB	BACKNUM	
055A'	049E'	C	DW	RETPROC	;DEFINE BACKGROUND ATTRIBUTES
		C			
055C'	6A	C	DB		VSRV
055D'	01	C	DB	BASNUM	
055E'	0261'	C	DW	ESCSRV	;START REVERSE VIDEO
0560'	6B	C	DB		VERV
0561'	01	C	DB	BASNUM	
0562'	027D'	C	DW	ESCCRV	;END REVERSE VIDEO
0564'	6C	C	DB		VSUL
0565'	01	C	DB	BASNUM	
0566'	0265'	C	DW	ESCSUN	;START UNDERLINE
0568'	6D	C	DB		VEUL
0569'	01	C	DB	BASNUM	
056A'	0281'	C	DW	ESCCUN	;END UNDERLINE
056C'	5E	C	DB		VSBL
056D'	01	C	DB	BASNUM	
056E'	0269'	C	DW	ESCSBL	;START BLINK
0570'	71	C	DB		VEBL
0571'	01	C	DB	BASNUM	
0572'	0285'	C	DW	ESCCBL	;END BLINK
0574'	3E	C	DB		EKC
0575'	01	C	DB	BASNUM	
0576'	028F'	C	DW	ESCSKK	;ENABLE KEY CLICK
0578'	3C	C	DB		DKC
0579'	01	C	DB	BASNUM	
057A'	0293'	C	DW	ESCKKK	;DISABLE KEY CLICK
057C'	65	C	DB		EFKT

```
057D' 01      C          DB BASNUM
057E' 0298'   C          DW ESCEFT      ;ENABLE FUNCTION KEY TRANSLATION
0580' 66      C          DB          DFKT
0581' 01      C          DB BASNUM
0582' 029C'   C          DW ESCDFT      ;DISABLE FUNCTION KEY TRANSLATION
          C
0584' 01      C      JX02: DB BASNUM
0585' 049B'   C          DW SEQERR      ;NO MATCH EXIT
          C
0021      C      ESCSIZ EQU (JX02-ESCT+1)/4      ;# OF ENTRIES IN TABLE
          C
          C      PAGE
```

```

C
C
0587' C      CTLT:
C      ;VALID CONTROL CHARACTER TABLE
C      ;1ST BYTE IS NUMBER OF BYTES IN TABLE
C      ;5 BYTES PER ENTRY: ASCII CHAR, NEXT TABLE ADDRESS, BRANCH ADDRESS
C      ;FOLLOWING BODY OF TABLE IS 4 BYTE DEFAULT NEXT TABLE AND BRANCH ADRS
C
0587' 0A C      DB          CTLSIZ                      ;# ENTRIES IN TABLE
C      ;-----+-----+-----+-----+
C      ; | CHARACTER | NEXT TABLE | BRANCH ADDR |
C      ;-----+-----+-----+-----+
C
0588' 00 C      DB          CR
0589' 01 C      DB          BASNUM
058A' 00D9' C      DW          VC_CR                      ;CARRIAGE RETURN ROUTINE
058C' 0A C      DB          LF
058D' 01 C      DB          BASNUM
058E' 00E3' C      DW          VC_LF                      ;LINE FEED
0590' 08 C      DB          BKS
0591' 01 C      DB          BASNUM
0592' 00F3' C      DW          VC_BKS                      ;BACK SPACE
0594' 0C C      DB          MCRIGH
0595' 01 C      DB          BASNUM
0596' 0113' C      DW          VC_MCRT                      ;MOVE CURSOR RIGHT
0598' 0B C      DB          MCUP
0599' 01 C      DB          BASNUM
059A' 012B' C      DW          VC_MCUP                      ;MOVE CURSOR UP
059C' 16 C      DB          VDMN
059D' 01 C      DB          BASNUM
059E' 0143' C      DW          VC_MCDOWN                      ;MOVE CURSOR DOWN
05A0' 07 C      DB          CBELL
05A1' 01 C      DB          BASNUM
05A2' 0243' C      DW          VC_BEL                      ;RING BELL
05A4' 1F C      DB          VLIN
05A5' 01 C      DB          BASNUM
05A6' 00ED' C      DW          VC_NEWLINE                      ;NEW LINE (CR,LF)
05A8' 1A C      DB          VCLRS
05A9' 01 C      DB          BASNUM
05AA' 01EF' C      DW          VC_CLRS                      ;CLEAR SCREEN
05AC' 1E C      DB          VHOME
05AD' 01 C      DB          BASNUM
05AE' 0152' C      DW          VC_HOME                      ;CURSOR HOME
C
C
05B0' 01 C      JX03: DB          BASNUM
05B1' 049E' C      DW          RETPROC                      ;NO MATCH
C      ;--IGNORE UNDEF CONTROL CHAR AND GO
C      ; BACK TO BASE TABLE
C
000A C      CTLSIZ EQU      (JX03-CTLT+1)/4                      ;NUMBER OF VALID ENTRIES
C
C      PAGE

```

```

C
C
05B3'      C      YTAB:
C      ;Y COORDINATE TABLE. ONLY DEFAULT ENTRY SINCE ALL VALUES GO TO THE SAME PROCEDURE
C
05B3'  00    C      DB          0                      ;NUMBER OF ENTRIES
C      ;      +-----+
C      ;      | CHARACTER | NEXT TABLE | BRANCH ADDR |
C      ;      +-----+
C
C
05B4'  05    C      DB          XNUM
05B5'  0372' C      DW          YCOORD ;"NO MATCH"
C
C
05B7'      C      XTAB:
C      ;X COORDINATE TABLE. ONLY DEFAULT ENTRY SINCE ALL VALUES GO TO THE SAME PROCEDURE
C
05B7'  00    C      DB          0                      ;NUMBER OF ENTRIES
C      ;      +-----+
C      ;      | CHARACTER | NEXT TABLE | BRANCH ADDR |
C      ;      +-----+
C
C
05B8'  01    C      DB          BASNUM
05B9'  037F' C      DW          XCOORD ;"NO MATCH"
C
C
C      PAGE

```

```

C
C
05BB'      C      BACKT:
C           C      ;BACKGROUND ATTRIBUTE TABLE
C
05BB'  00    C      DB          BACKSIZ
C           C      ;-----+-----+
C           C      ; | CHARACTER | NEXT TABLE | BRANCH ADDR |
C           C      ; |-----+-----+
C
05BC'  01    C      JX04: DB          BASNUM
05BD'  02B3' C      DW          NEW_BACK      ;SET BACKGROUND ATTRIBUTES
C
0000      C      BACKSIZ EQU      (JX04-BACKT+1)/4
C
C      PAGE

```

```
C
C
C      WDEFT:
C      ;WINDOW DEFINE TABLE
C      ;(THIS LEVEL IN THE WINDOW DEFINITION SEQUENCE GETS THE WINDOW NUMBER TO DEFINE)
C
C      DB                      WDEFSIZ
C      ;      +-----+
C      ;      | CHARACTER | NEXT TABLE | BRANCH ADDR |
C      ;      +-----+
C
C      JX05: DB                FROMNUM
C              DW                DEF_WINDOW          ;GET WINDOW NUMBER
C
C      WDEFSIZ EQU        (JX05-WDEFT+1)/4
C
C      PAGE
```

```

C
C
05C3'      C      FROMT:
C           C      ;DEFINE FIRST ROW OF WINDOW
C
C
05C3'  00    C      DB          FROWSIZ
C           C      ;      +-----+
C           C      ;      | CHARACTER | NEXT TABLE | BRANCH ADDR |
C           C      ;      +-----+
C
05C4'  00    C      JX06: DB          FCOLNUM
05C5'  0405'  C      DW          STARTROW      ;DEFINE FIRST ROW OF WINDOW
C
0000        C      FROWSIZ EQU      (JX06-FROWT+1)/4
C
C
C
05C7'      C      FCOLT:
C           C      ;DEFINE FIRST COLUMN OF WINDOW
C
C
05C7'  00    C      DB          FCOLSIZ
C           C      ;      +-----+
C           C      ;      | CHARACTER | NEXT TABLE | BRANCH ADDR |
C           C      ;      +-----+
C
05C8'  09    C      JX07: DB          LROWNUM
05C9'  041C'  C      DW          STARTCOL      ;DEFINE FIRST COLUMN OF WINDOW
C
0000        C      FCOLSIZ EQU      (JX07-FCOLT+1)/4
C
C      PAGE
C

```



```

C
C
05CB'      C      LROWT:
C           C      ;DEFINE LAST ROW OF WINDOW
C
C
05CB'  00    C      DB          LROWSIZ
C           C      ;-----+-----+
C           C      ; | CHARACTER | NEXT TABLE | BRANCH ADDR |
C           C      ;-----+-----+
C
05CC'  0A    C      JX08: DB          LCOLNUM
05CD'  0434' C      DW          ENDROW          ;DEFINE LAST ROW OF WINDOW
C
0000        C      LROWSIZ EQU      (JX08-LROWT+1)/4
C
C
05CF'      C      LCOLT:
C           C      ;DEFINE LAST COLUMN OF WINDOW
C
C
05CF'  00    C      DB          LCOLSIZ
C           C      ;-----+-----+
C           C      ; | CHARACTER | NEXT TABLE | BRANCH ADDR |
C           C      ;-----+-----+
C
05D0'  01    C      JX09: DB          BASNUM
05D1'  044C' C      DW          ENDCOL          ;DEFINE LAST COLUMN OF WINDOW
C
0000        C      LCOLSIZ EQU      (JX09-LCOLT+1)/4
C
C      PAGE

```

```

C
C
05D3'      C      WSETT:
C           C      ;WINDOW SET
C
C
05D3'  00    C           DB          WSETSIZE
C           C      ;-----+-----+
C           C      ; | CHARACTER | NEXT TABLE | BRANCH ADDR |
C           C      ;-----+-----+
C
05D4'  01    C      JX10: DB          BASNUM
05D5'  047E' C           DW          SETWIND      ;SET CURRENT WINDOW
C           C      ;TO SELECTED NUMBER
C
0000        C      WSETSIZE EQU      (JX10-WSETT+1)/4
C
C           C      PAGE

```

```
C
C
05D7'      C      CTYPT:
C      ;CURSOR TYPE
C
05D7'  00    C      DB          CTYPSIZ
C      ;      +-----+
C      ;      | CHARACTER | NEXT TABLE | BRANCH ADDR |
C      ;      +-----+
C
05D8'  01    C      JX11: DB          BASNUM
05D9'  02B9' C      DW          DEFCUR      ;DEFINE CURSOR TYPE
C
0000      C      CTYPSIZ EQU      (JX11-CTYPT+1)/4
C
C      PAGE
```

```

C
C
05DB' 00      C      OFFT:
C              ;SET X OFFSET
C
C              DB          OFFSIZ
C              ;-----+-----+
C              ; | CHARACTER | NEXT TABLE | BRANCH ADDR |
C              ;-----+-----+
C
05DC' 01      C      JX12: DB          BASNUM
05DD' 038C'   C              DW          SETOFF          ;SET X-Y OFFSET
C
0000          C      OFFSIZ EQU      (JX12-OFFT+1)/4
C
C              ;CONSOLE TABLE NUMBERS
0001          C      BASNUM EQU      1          ;BASE TABLE NUMBER
0002          C      ESCNUM EQU      2          ;ESCAPE TABLE NUMBER
0003          C      CTLNUM EQU      3          ;CONTROL TABLE NUMBER
0004          C      YNUM EQU      4          ;Y COORDINATE TABLE NUMBER
0005          C      XNUM EQU      5          ;X COORDINATE TABLE NUMBER
0006          C      WDEFNUM EQU      6          ;WINDOW DEFINITION TABLE NUMBER
0007          C      FROWNUM EQU      7          ; FIRST ROW TABLE NUMBER
0008          C      FCOLNUM EQU      8          ; FIRST COLUMN TABLE NUMBER
0009          C      LROWNUM EQU      9          ; LAST ROW TABLE NUMBER
000A          C      LCOLNUM EQU     10          ; LAST COLUMN TABLE NUMBER
000B          C      WSETNUM EQU     11          ;WINDOW SET TABLE NUMBER
000C          C      CTYPNUM EQU     12          ;CURSOR TYPE TABLE NUMBER
000D          C      OFFNUM EQU     13          ;X OFFSET TABLE NUMBER
000E          C      BACKNUM EQU     14          ;Y OFFSET TABLE NUMBER
C
C              PAGE

```

```

C
C
C      ; *****
C      ; *                                     *
C      ; *               CONSOLE TABLE POINTERS               *
C      ; *                                     *
C      ; *****
C
C      ;CONSOLE TABLE POINTER STRUCTURE
05DF' CTBLPTRS:
05DF' 04F7' C      DW      BASET                      ;CURRENT TABLE TO BE SEARCHED
C      ;INITIALLY SEARCH THE BASE TABLE
C
C      ;TABLE POINTERS
05E1' 04F7' C      DW      BASET                      ; BASE TABLE
05E3' 04FF' C      DW      ESCT                       ; ESCAPE TABLE
05E5' 0587' C      DW      CTLT                      ; CONTROL TABLE
05E7' 05B3' C      DW      YTAB                      ; Y-COORDINATE TABLE
05E9' 05B7' C      DW      XTAB                      ; X-COORDINATE TABLE
05EB' 05BF' C      DW      WDEFT                    ; WINDOW DEFINE TABLE
05ED' 05C3' C      DW      FROMT                    ; FIRST ROW OF WINDOW (FOR WINDOW DEFINE)
05EF' 05C7' C      DW      FCOLT                    ; FIRST COLUMN OF WINDOW
05F1' 05CB' C      DW      LROWT                    ; LAST ROW OF WINDOW
05F3' 05CF' C      DW      LCOLT                    ; LAST COLUMN OF WINDOW
05F5' 05D3' C      DW      WSETT                    ; WINDOW SET TABLE
05F7' 05D7' C      DW      CTYPT                    ; CURSOR TYPE DEFINITION TABLE
05F9' 05DB' C      DW      OFFT                     ; X-Y OFFSET TABLE
05FB' 05BB' C      DW      BACKT                    ; BACKGROUND ATTRIBUTE DEFINITION TABLE
C
001E C      LENC_TBL EQU $-CTBLPTRS
C      PAGE

```

'EXECUTIVE ROM MONITOR'
'CONSOLE ROUTINES'

MACRO-80 3.44 09-Dec-81

PAGE 1-180

;

SUBTTL
PAGE

'KEYBOARD ROUTINES'

```

C      SUBTTL      'KEYBOARD ROUTINES'
C      INCLUDE ROM3      ;KEYBOARD ROUTINES
C      ;
C      ;
C      ;*****
C      ;*
C      ;*      KEYBOARD ROUTINES      *
C      ;*
C      ;*****
C      ;
C      ;      MODULE 3 --ROM3-- ENTRY POINTS
C      ;
C      ;      KEYBOARD ROUTINES
C      ;
C      ;      KBDVR:      DETECT AND PROCESS KEYSTROKES
C      ;      REMOVEKEY:  REMOVE ENTRY FROM KEYLIST
C      ;      CKSERV:     CHECK FOR TIME-OUT ON KEYLIST ENTRY
C      ;      KBSCAN:     SCAN THE KEYBOARD FOR DEPRESSED KEYS
C      ;      CHKEY:      CHECKS IF KEY NUMBER IS ON
C      ;      GTMASK:     GENERATES AN 8 BIT MASK WITH 1 BIT SET
C      ;      RDROW:      READS ROWS OF KEYS
C      ;      KBSERV:     SERVICE KEYLIST ENTRY
C      ;      SAVEKEY:    SEND CHAR. FROM KEYBOARD TO BIOS
C
C      PAGE
```

05FD'

```
C
C
C      .Z00
C      ;      .XLIST
C      ;      INCLUDE      SYSS1.MAC      ;SYSTEST FILE
C      ;      INCLUDE      RAMSMON.MAC    ;MONITOR RAM MEMORY LOCATIONS
C      ;      INCLUDE      RAMS10.MAC    ;RAM MEMORY LOCATIONS AND CONSTANTS
C      ;      .LIST
C      ;      ;SYSS1.MAC , RAMSMON.MAC AND RAMS10.MAC INCLUDED
C
C      ;      EXT      AL_T_SZ,FL_B_ST,FL_B_CH,ALT_FLG
C
C      ;      PUBLIC  KBDVR
C
C      CSEG
C
C      PAGE
```



```

C
C
05FD' C KBDVR:
C ;DETECTS AND PROCESSES KEYSTROKES.
C ; MODIFIED TO PROCESS KEY ROLLOVER LIKE THE TELEVIDEO 9XX SERIES.
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;LKEY = KEYSTROKE
C ; ANY VALID KEYSTROKES ARE TRANSLATED BY KBSERV AND SENT TO THE
C ; BIOS BY SAVEKEY.
C ;
C ;
C
05FD' CD 0692' C CALL KBSCAN ;SCAN KEYBOARD FOR LATCHED KEYS AND
C ;PLACE KEYS IN KEYLIST
C ;
C ; CHECK FOR ANY KEYS IN THE KEY LIST. IF THE FIRST KEYLIST ENTRY
C ; IS EMPTY, THEN THERE ARE NO KEYS IN LIST
C ;
C ;
0500' 21 23DE C LD HL,FIRSTKEY ;POINT AT FIRST KEYLIST ENTRY
0503' 7E C LD A,(HL) ;GET KEY LIST ENTRY
0504' CB 7F C BIT KL_USED,A ;SEE IF ENTRY USED BIT IS SET
0506' C8 C RET Z ;NO...KEYLIST ENTRY: RETURN
C ;
C ; NOW SEE IF KEYS IN LIST ARE STILL DOWN
C ;
0507' C CKEYSDN:
0507' 06 03 C LD B,KL_LEN ;SET LOOP COUNTER TO NUMBER OF KEYS IN
C ;LIST. HL POINTS TO FIRST ENTRY, A
C ;CONTAINS FIRST KEY ENTRY
0509' CB 7F C CKLP: BIT KL_USED,A ;IS THIS KEYLIST ENTRY IN USE?
050B' 20 1E C JR Z,ENDCK ;NO...NO MORE ENTRIES TO CHECK
C ;
C ; KEYLIST ENTRY NOT EMPTY: IS THE KEY STILL DOWN?
C ;
050D' C5 C PUSH BC ;SAVE LOOP COUNTER OVER CALL TO CHKEY
050E' E6 3F C AND KROW_M+KCOL_M ;MASK OFF ALL BITS EXCEPT ROW & COL NO.
0510' CD 06EF' C CALL CHKEY ;CALL CHKEY TO SEE IF KEY IN A IS STILL
C ;DOWN
0513' C1 C POP BC ;RESTORE LOOP COUNTER
0514' 20 0F C JR NZ,KEYON ;Z SET IF KEY OFF...
C ;
0516' C5 C PUSH BC ;SAVE LOOP COUNTER
0517' E5 C PUSH HL ;SAVE POINTER TO CURRENT LIST ENTRY
0518' CD 065C' C CALL REMOVEKEY ;REMOVE KEY FROM LIST. NEXT KEY (IF ANY)
C ;IS NOW IN CURRENT SLOT, SINCE REMOVEKEY
C ;MOVES EVERYTHING DOWN ONE SLOT.
051B' E1 C POP HL ;RESTORE PTR TO KEY LIST ENTRY
051C' C1 C POP BC ;RESTORE LOOP COUNTER
051D' 3E 01 C LD A,1 ;SET A TO VALUE B SHOULD BE IF THIS IS
C ;THE LAST ENTRY
051F' B8 C CP B ;COMPARE A WITH B. IS THIS THE LAST

```

```

0620' 28 09      C      JR      Z,ENDCK      ;ENTRY?
                                           ;YES...
0622' 7E         C      LD      A,(HL)      ;NO. MORE TO CHECK. LOAD NEXT KEY, NOW
                                           ;IN CURRENT SLOT
0623' 18 E4      C      JR      CKLP        ;AND CHECK THIS KEY.
                                           ;
0625'           C      KEYON:              ;KEY IS ON: GO TO NEXT KEY, IF ANY.
0625' 23         C      INC      HL          ;BUMP HL TWICE TO POINT TO NEXT ENTRY
0626' 23         C      INC      HL
0627' 7E         C      LD      A,(HL)      ;LOAD KEY POINTED TO BY HL
0628' 05         C      DEC      B          ;DECREMENT LOOP COUNTER. ARE WE ALL
                                           ;DONE?
0629' 20 DE      C      JR      NZ,CKLP      ;NO... CONTINUE
                                           ;
062B'           C      ENDCK:              ;ALL KEYS IN LIST HAVE BEEN CHECKED TO
                                           ;SEE IF THEY'RE STILL DOWN.
                                           ;
                                           ;
                                           ;
                                           ;
062B' 21 23DE    C      LD      HL,FIRSTKEY  ;POINT TO FIRST KEYLIST ENTRY
062E' CB 7E      C      BIT      KL_USED,(HL) ;IS IT IN USE?
0630' CB         C      RET      Z          ;NO. NO FIRST KEY IMPLIES NO KEYS AT
                                           ;ALL. RETURN.
                                           ;
                                           ;
                                           ;
                                           ;
                                           ;
0631' 21 23E2    C      LD      HL,SECONDKEY+2 ;POINT TO KEY AFTER SECOND
0634' 7E         C      LD      A,(HL)      ;GET IT.
0635' CB 7F      C      BIT      KL_USED,A    ;IS IT IN USE?
0637' C0         C      RET      NZ          ;YES. MORE THAN TWO KEYS DOWN. RETURN.
                                           ;
                                           ;
                                           ;
                                           ;
                                           ;
063B' 21 23E0    C      LD      HL,SECONDKEY  ;POINT AT SECOND KEYLIST ENTRY
063B' CB 7E      C      BIT      KL_USED,(HL) ;IS THERE A SECOND KEY IN THE LIST?
063D' 28 10      C      JR      Z,CHECK1ST   ;NO. CHECK FIRST KEY
                                           ;
                                           ;
                                           ;
                                           ;
                                           ;
                                           ;TWO KEYS DOWN. SEE IF IT'S TIME TO
                                           ;SERVICE THE SECOND KEY.
063F' E5         C      PUSH   HL          ;SAVE POINTER TO KEYLIST ENTRY AROUND
                                           ;CALL
0640' CD 0679'    C      CALL   CKSERV      ;CHECK KEYLIST ENTRY POINTED TO BY HL
                                           ;FOR SERVICE TIMEOUT. CARRY SET IF TIME
                                           ;TO SERVICE.
0643' E1         C      POP    HL          ;RESTORE PTR TO ENTRY
                                           ;
0644' D0         C      RET      NC          ;NOT TIME TO SERVICE KEY. RETURN.
                                           ;
0645' CD 0658'    C      CALL   SERVKEY     ;SERVICE KEY POINTED TO BY HL. IF KEY BAD, Z FLAG SET.
                                           ;
0648' C0         C      RET      NZ          ;RETURN IF KEY SENT TO BIOS
                                           ;

```

```

0649' 21 23DE C LD HL, FIRSTKEY ;SECOND KEY BAD, SEND FIRST KEY
064C' C3 0658' C JP SERVKEY
C ;
064F' C CHECK1ST: ;AS THERE IS NO SECOND KEY DOWN OR IT IS
C ;TIMED OUT OR BAD, CHECK FIRST KEY IN LIST.
064F' 21 23DE C LD HL, FIRSTKEY ;POINT AT FIRST KEYLIST ENTRY
0652' E5 C PUSH HL ;SAVE PTR
0653' CD 0679' C CALL CKSERV ;IS IT TIME TO SERVICE?
0656' E1 C POP HL ;RESTORE PTR
0657' D0 C RET NC ;NO: CARRY CLEAR, NOT TIME TO SERVICE.
C ;RETURN...
C ;
0658' C SERVKEY: ;SERVICE KEY POINTED TO BY HL
0658' CD 0713' C CALL KBSERV
065B' C9 C RET ;AND RETURN
C
C PAGE

```

	C			
	C			
065C'	C	REMOVEKEY:		
	C	; REMOVE KEYLIST ENTRY POINTED TO BY HL FROM KEYLIST		
	C	;ENTRY:		
	C	; HL ==> KEYLIST ENTRY TO DELETE.		
	C	;		
	C	;EXIT:		
	C	; ALL FOLLOWING ENTRIES IN KEYLIST (IF ANY) ARE SHIFTED DOWN		
	C	; ONE SLOT.		
	C	; IF THERE ARE NO FOLLOWING ENTRIES IN THE KEYLIST, THE ENTRY		
	C	; POINTED TO IS SIMPLY ZEROED.		
	C	;		
065C' E5	C	PUSH HL		;SAVE POINTER TO ENTRY TO DELETE
065D' 11 23E2	C	LD DE, LASTKEY		;LOAD ADDRESS OF LAST KEY IN LIST
0660' EB	C	EX DE, HL		;SWAP HL AND DE SO SUBSEQUENT SUBTRACT
	C			;OPERATION WILL RETURN THE NUMBER OF
	C			;CELLS TO MOVE OR ZERO IF CURRENT ENTRY
	C			;IS LAST.
0661' 37	C	SCF		;CLEAR CARRY SO DOUBLE PRECISION
0662' 3F	C	CCF		;SUBTRACT WITH CARRY CAN BE USED
0663' ED 52	C	SBC HL, DE		;SUBTRACT ADDRESS OF ENTRY TO DELETE
	C			;FROM ADDRESS OF LAST ENTRY IN LIST.
	C			;IF ZERO SET THE ENTRY TO DELETE IS THE
	C			;LAST IN THE LIST, ELSE HL CONTAINS THE
	C			;NUMBER OF BYTES TO MOVE.
0665' 44	C	LD B, H		;MOVE RESULT TO BC TO SET UP FOR LDIR
0666' 4D	C	LD C, L		;
0667' D1	C	POP DE		;RESTORE POINTER TO DE FOR LDIR
0668' 28 06	C	JR Z, DELLAST		;IF PREVIOUS SUBTRACT RESULTED IN ZERO,
	C			;ENTRY TO DELETE IS LAST IN LIST AND
	C			;THE ONLY ACTION TO TAKE IS TO DELETE
	C			;THE LAST LIST ENTRY.
066A' 62	C	LD H, D		;COPY POINTER TO ENTRY TO DELETE TO HL
066B' 6B	C	LD L, E		;
066C' 23	C	INC HL		;BUMP HL TO POINT TO NEXT ENTRY
066D' 23	C	INC HL		;
066E' ED B0	C	LDIR		;DO BLOCK MOVE: MOVE ENTRIES FOLLOWING
	C			;ENTRY TO DELETE DOWN INTO ENTRY TO
	C			;DELETE
0670'	C	DELLAST:		;FINALLY, CLEAR LAST LIST ENTRY.
0670' 21 23E2	C	LD HL, LASTKEY		;POINT TO LAST KEY LIST ENTRY
0673' 36 00	C	LD (HL), 0		;CLEAR FIRST BYTE OF ENTRY
0675' 23	C	INC HL		;POINT TO SECOND BYTE
0676' 36 00	C	LD (HL), 0		;CLEAR IT TO.
0678' C9	C	RET		;AND RETURN
	C			
	C	PAGE		

```

C
C
0679' C CKSERV:
C ; ROUTINE TO CHECK FOR TIME OUT ON KEY LIST ENTRY.
C ;ENTRY:
C ; HL ==> ENTRY TO CHECK FOR TIME OUT.
C ;
C ;EXIT:
C ; CARRY FLAG SET IF TIME TO SERVICE KEY LIST ENTRY.
C ; ADDITIONALLY, IF IT IS NOT TIME TO SERVICE THE KEY, THE TIME OUT
C ; COUNTER OF THE LIST ENTRY IS DECREMENTED. IF IT IS TIME TO
C ; SERVICE THE KEY, THE COUNTER IS RE-INITIALIZED. THE VALUE
C ; IT IS SET TO DEPENDS ON WHETHER THE KEY HAS SERVICED ALREADY.
C ; IF IT HASN'T, THE COUNTER IS SET TO A VALUE WHICH CAUSES A
C ; SLIGHT DELAY BEFORE THE KEY BEGINS TO REPEAT.
C ; OTHERWISE, THE KEY HAS ALREADY BEGUN TO REPEAT, SO THE COUNTER
C ; IS SET TO A SMALLER VALUE ASSOCIATED WITH THE RAPIDITY WITH
C ; WHICH THE KEY REPEATS.
C ; HL = VALUE ON ENTRY.
C ;
C
0679' 23 C INC HL ;INCREMENT HL TO POINT TO LIST ENTRY
C ;REPEAT COUNT
067A' 35 C DEC (HL) ;DECREMENT REPEAT COUNT
067B' 2B C DEC HL ;DECREMENT HL TO POINT BACK TO KEY.
067C' 20 11 C JR NZ,NOSERV ;REPEAT COUNT NOT ZERO; NOT TIME TO
C ;SERVICE KEY
C ;
067E' CB 76 C BIT KY_SRVD, (HL) ;KEY READY TO SERVICE. HAS IT BEEN
C ;SERVICED BEFORE?
0680' 20 07 C JR NZ,SRVDLRDY ;YES...
C ;
0682' CB F6 C SET KY_SRVD, (HL) ;NO. SET SERVICED ALREADY BIT FLAG IN
C ;KEYLIST ENTRY
0684' 23 C INC HL ;POINT TO KEYLIST ENTRY REPEAT COUNT.
0685' 36 18 C LD (HL),IRPTCT ;INITIALIZE REPEAT COUNT TO DELAY BEFORE
C ;KEY SHOULD START REPEATING
0687' 18 03 C JR XJ080 ;AND EXIT WITH CARRY SET AND HL POINTING
C ;TO LIST ENTRY
0689' C SRVDLRDY: ;NOT FIRST TIME KEY HAS BEEN SERVED
0689' 23 C INC HL ;POINT TO ENTRY REPEAT COUNT
068A' 36 03 C LD (HL),SRPTCT ;INITIALIZE REPEAT COUNTER TO DELAY
C ;BEFORE KEY SHOULD REPEAT AGAIN
C ;
068C' C XJ080: ;RESTORE HL TO INITIAL ENTRY VALUE AND
C ;SET CARRY FLAG DENOTING IT IS TIME TO
C ;SERVICE KEY.
068C' 2B C DEC HL ;DECREMENT HL TO POINT TO KEY LIST ENTRY
068D' 37 C SCF ;SET CARRY FLAG
068E' C9 C RET ;AND RETURN
C ;
068F' C NOSERV: ;NOT TIME TO SERVICE KEY. CLEAR CARRY
068F' 37 C SCF ;TO CLEAR, SET CARRY
0690' 3F C CCF ;AND COMPLEMENT
0691' C9 C RET ;AND RETURN

```

C

C

PAGE


```

C
C      ;KEY DOWN. B = ROW, C = COLUMN
C
06B2' D5      C      PUSH    DE      ;SAVE ROW BIT MASK
06B3' F5      C      PUSH    AF      ;AND COLUMN BIT MASK AROUND KEY SAVE
C
06B4' 3E 08    C      LD      A,TOT_ROW ;TRANSFORM ROW & COL # INTO BITS 0-5 OF A
06B5' 90      C      SUB     B      ;GET RELATIVE-0 ROW # TO A
06B7' 17      C      RLA          ;SHIFT ROW # TO BITS 3, 4 & 5
06B8' 17      C      RLA
06B9' 17      C      RLA
06BA' B1      C      OR      C      ;OR COL # INTO BITS 0, 1 & 2
C
06BB' 57      C      LD      D,A      ;PUT ROW & COL IN D
C
06BC' 1E 03    C      LD      E,KL_LEN ;INIT KEY LIST ENTRY COUNTER
06BE' 21 23DE  C      LD      HL,FIRSTKEY ;POINT HL AT FIRST KEY LIST ENTRY
C
06C1'          C      XJ085: ;IS CURRENT KEY LIST ENTRY FREE?
06C1' CB 7E    C      BIT     7,(HL) ;CHECK ENTRY IN USE BIT.
06C3' CA 06D6' C      JP      Z,XJ086 ;
C
C      ;
C      ;THIS ENTRY NOT FREE.
C      ;IS THIS ENTRY SAME AS KEY WE'RE TRYING TO
C      ;SAVE?
06C6' 7E      C      LD      A,(HL) ;GET KEY FROM LIST
06C7' E6 3F    C      AND     KROW_M+KCOL_M ;STRIP OFF FLAG BITS
06C9' BA      C      CP      D      ;SAME AS ROW/COL # IN D?
06CA' CA 06DC' C      JP      Z,XJ087 ; YES: KEY ALREADY IN LIST
C
C      ;
C      ;KEY NOT IN THIS LIST ENTRY. CHECK NEXT ENTRY.
06CD' 1D      C      DEC     E      ;DECREMENT KEY LIST ENTRY COUNTER
06CE' CA 06EB' C      JP      Z,XJ088 ; IF ZERO, NO FREE LIST ENTRIES
C
C      ;
C      ;NOT LAST LIST ENTRY.
06D1' 23      C      INC     HL      ;POINT TO NEXT ENTRY
06D2' 23      C      INC     HL
06D3' C3 06C1' C      JP      XJ085 ;CHECK NEXT ENTRY
C
C      ;
06D6'          C      XJ086: ;FREE LIST ENTRY FOUND. SAVE KEY IN ENTRY.
C      ;HL ==> FREE ENTRY.
06D6' CB FA    C      SET     KL_USED,D ;SET ENTRY USED FLAG IN ENTRY IN D
06D8' 72      C      LD      (HL),D ;SAVE ENTRY IN FREE SLOT IN LIST.
06D9' 23      C      INC     HL      ;POINT AT COUNT FIELD OF ENTRY
06DA' 36 01    C      LD      (HL),DB_CT ;SAVE DEBOUNCE COUNT.
C
C      PAGE

```



```

C
06DC'      C      XJ087: ;KEY ALREADY IN LIST OR JUST SAVED IN LIST.
06DC'  F1      C      POP      AF      ;RESTORE COLUMN BIT MASK FOR CURRENT ROW
06DD'  D1      C      POP      DE      ;AND ROW BIT MASK IN D
C
06DE'      C      XJ084: ;CURRENT KEY NOT DOWN.
C      ;IS THIS THE LAST KEY IN CURRENT ROW?
06DE'  0D      C      DEC      C      ;DECREMENT COLUMN COUNTER
06DF'  F2 06AD' C      JP      P,XJ083      ;IF NO BORROW (C POSITIVE), NOT DONE WITH ROW.
C      ;
C      ;DONE WITH CURRENT ROW.
06E2'      C      XJ082: ;CHECK NEXT ROW
06E2'  CB 22      C      SLA      D      ;SHIFT ROW MASK LEFT TO GET MASK FOR NEXT ROW
06E4'  05      C      DEC      B      ;IS THIS THE LAST ROW?
06E5'  C2 06A0' C      JP      NZ,XJ081      ;NO...
C      ;
06E8'  C3 06ED' C      JP      XJ092      ;YES, END SCAN
C      ;
C
06EB'      C      XJ088: ;NO FREE ENTRIES IN LIST.
06EB'  F1      C      POP      AF      ;RESTORE COLUMN BIT MASK FOR CURRENT ROW
06EC'  D1      C      POP      DE      ;AND ROW MASK...
C
06ED'      C      XJ092: ;LAST ROW CHECKED OR NO MORE ENTRIES IN KEY LIST.
06ED'  E1      C      POP      HL      ;RESTORE PTR TO MASK TABLE TO BALANCE STACK
06EE'  C9      C      RET      ;AND RETURN.
C
C      PAGE

```

```

C
C
06EF' C CHKEY:
C ;CHECKS IF KEY NUMBER IS ON.
C ;ENTRY
C ;A = KEYNUMBER
C ;EXIT
C ;Z CLR = KEY IS ON.
C ;Z SET = KEY IS OFF.
C
C
C
06EF' E5 C PUSH HL ;SAVE CALLERS HL
06F0' F5 C PUSH AF ;SAVE KEYNUMBER
06F1' 1F C RRA
06F2' 1F C RRA
06F3' 1F C RRA ;RIGHT JUSTIFY ROW NUMBER
06F4' CD 0704' C CALL GTMASK
06F7' F1 C POP AF ;GET KEY NUMBER
C
06F8' D5 C PUSH DE ;SAVE ROW MASK
06F9' CD 0704' C CALL GTMASK ;GET COL MASK (COL NUM IS IN BITS 0..2)
06FC' E1 C POP HL ;MOVE ROW MASK TO L
06FD' 7D C LD A,L ;RDROW EXPECTS MASK IN A
C
06FE' CD 070E' C CALL RDROW ;GET ROW OF KEYS ADRESSED BY L
0701' A3 C AND E ;Z IND = VALUE OF KEY
0702' E1 C POP HL
C
0703' C9 C RET
C
C PAGE

```

```
C
C
0704' C      BTMASK:
C      ;GENERATES MASK WITH ONE BIT SET.
C      ;ENTRY
C      ;A      =      BIT NUMBER (0..7)
C
C      ;EXIT
C      ;E      =      MASK
C
C
0704' 1E 01 C      LD      E,1
0705' E6 07 C      AND      7
C
0708' C8 C      XJ094: RET      Z
0709' CB 23 C      SLA      E
070B' 3D C      DEC      A
070C' 18 FA C      JR      XJ094
C
C      PAGE
```

```

C
C
070E' C RDRW:
C ;READS ROWS OF KEYS
C ;ENTRY
C ;A = ROWS TO READ (EACH BIT INDICATES A ROW)
C
C ;EXIT
C ;A = ROW VALUE
C ; (A BIT IS SET IF THAT BIT (KEY) IS SELECTED IN ANY OF THE CHOSEN ROWS)
C
C
070E' DB 14 C IN A, (KEYBD) ;READ SELECTED ROW
0710' EE FF C XOR 0FFH ;INVERT VALUES
0712' C9 C RET
C
C PAGE
```

```

C
C
0713' C KBSERV:
C ; ROUTINE TO SERVICE KEY LIST ENTRY.
C ;ENTRY:
C ; HL ==> KEY LIST ENTRY TO SERVICE.
C ;EXIT:
C ; NONE.
C ;
C ;
0000 C CT_BIT EQU 0 ;TEST
0001 C SH_BIT EQU 1 ;TEST
0002 C AL_BIT EQU 2 ;TEST
C ;
C
0713' 7E C LD A,(HL) ;GET KEYLIST ENTRY POINTED TO BY HL.
0714' E6 3F C AND KROW_M+KCOL_M ;MASK TO LEAVE ONLY COLUMN AND ROW
C ;NUMBER BITS.
0716' F5 C PUSH AF ;SAVE KEY LIST ENTRY
C ;
C ; GET STATUS OF CTRL, SHIFT AND ALPHA LOCK KEYS.
C ;
0717' 2E 00 C LD L,0 ;CLEAR L. BITS CORRESPONDING TO SHIFT-TYPE KEYS
C ;WILL BE ACCUMULATED THERE.
C
0719' 3A 2012 C LD A,(CT_KEY) ;GET LOC OF CTRL KEY
071C' CD 06EF' C CALL CHKEY ;IS IT DOWN?
071F' CA 0724' C JP Z,XJ095 ;NO...
C ;
0722' CB C5 C SET CT_BIT,L ;YES, SET BIT IN L
C
0724' 3A 2014 C XJ095: LD A,(SH_KEY) ;GET LOC OF SHIFT KEY.
0727' CD 06EF' C CALL CHKEY ;IS IT DOWN?
072A' CA 072F' C JP Z,XJ096 ;NO...
C ;
072D' CB CD C SET SH_BIT,L ;YES, SET BIT IN L
C
072F' 3A 2013 C XJ096: LD A,(AL_KEY) ;GET LOC OF ALPHA LOCK.
0732' CD 06EF' C CALL CHKEY ;IS IT DOWN?
0735' CA 073A' C JP Z,XJ097 ;NO...
C ;
0738' CB D5 C SET AL_BIT,L ;YES, SET BIT IN L
C
073A' C XJ097:
C ;
C ; L NOW CONTAINS SHIFT, CTRL AND ALPHA LOCK.
C ; THUS WE CAN NOW CHECK FOR THESE KEYS BEING DOWN:
C ;
073A' CB 45 C BIT CT_BIT,L ;IS CTRL DOWN?
073C' 28 0E C JR Z,NOCTRL ;NO...
C ;
073E' CB 4D C BIT SH_BIT,L ;YES; IS SHIFT DOWN TOO?
0740' 28 05 C JR Z,NOSHCTRL ;NO...ONLY CTRL IS DOWN
C ;

```

```

0742' 11 0100 C          LD    DE,4*AL_T_SZ    ;CTRL AND SHIFT DOWN. INDICATE THAT
C          ;CURRENT CTRL/SHIFT TABLE IS TO BE USED
0745' 18 1A C          JR     XLATEKEY    ;TRANSLATE KEY USING TABLE INDICATED BY
C          ;C
C          ;
0747' C          NOSHCTRL: ;ONLY CTRL DOWN.
0747' 11 0080 C          LD    DE,2*AL_T_SZ    ;USE CTRL TABLE...
074A' 18 15 C          JR     XLATEKEY    ;TO TRANSLATE KEY.
C          ;
074C' C          NOCTRL: ;CTRL NOT DOWN...
074C' CB 4D C          BIT    SH_BIT,L      ;IS SHIFT DOWN?
074E' 28 05 C          JR     Z,NOSHIFT    ;NO...
C          ;
0750' 11 0040 C          LD    DE,1*AL_T_SZ    ;YES...USE SHIFT TABLE TO
0753' 18 0C C          JR     XLATEKEY    ;TRANSLATE KEY
C          ;
0755' C          NOSHIFT: ;SHIFT NOT DOWN.
0755' CB 55 C          BIT    AL_BIT,L      ;IS ALPHA LOCK DOWN?
0757' 28 05 C          JR     Z,NOALPHA    ;NO...
C          ;
0759' 11 00C0 C          LD    DE,3*AL_T_SZ    ;YES...USE ALPHA LOCK TABLE TO
075C' 18 03 C          JR     XLATEKEY    ;TRANSLATE KEY
C          ;
075E' C          NOALPHA: ;ALPHA LOCK NOT DOWN. NO 'TRANSLATE'
C          ;KEYS DOWN, SO USE
075E' 11 0000 C          LD    DE,0*AL_T_SZ    ;'NORMAL' TABLE TO TRANSLATE KEYS
C          ;
0761' C          XLATEKEY: ;TRANSLATE KEYSTROKE ON STACK ACCORDING
C          ;TO THE TABLE WHOSE NUMBER IS IN A
C          ;
0761' 21 2015 C          LD    HL,KEYS      ;LOAD POINTER TABLE BASE
0764' 19 C          ADD    HL,DE      ;INDEX TO CORRECT TABLE ENTRY
C          ;
0765' F1 C          POP    AF        ;RESTORE KEY NUMBER TO A
0766' 16 00 C          LD    D,0      ;ZERO HI BYTE OF HL TO INDEX INTO
C          ;TRANSLATE TABLE
0768' 5F C          LD    E,A        ;MOVE KEY NUMBER TO L
0769' 19 C          ADD    HL,DE      ;ADD BASE ADDRESS OF TABLE TO INDEX
076A' 7E C          LD    A,(HL)      ;GET TRANSLATED KEYSTROKE FROM TABLE.
076B' FE FF C          CP    BADKEY    ;IS IT AN INVALID KEYSTROKE? (MOST
C          ;LIKELY A BAD CTRL/SHIFT COMBINATION)
076D' C8 C          RET    Z        ;YES...RETURN WITHOUT SENDING CHAR TO
C          ;BIOS
C          ;
076E' FE FE C          CP    TOG_ALT_SET ;IS CHAR TOGGLE ALTERNATE CHAR SET COMMAND?
0770' CA 0779' C          JP    Z,XJ098 ;YES...
C          ;
C          ;NO, PROCESS AS NORMAL CHAR
0773' F5 C          PUSH  AF        ;SAVE Z FLAG
0774' CD 0783' C          CALL SAVEKEY ;SEND CHAR TO BIOS
0777' F1 C          POP    AF        ;RESTORE Z FLAG
0778' C9 C          RET             ;AND RETURN
C          PAGE

```

```

0779'      C
          C      XJ090:
          C      ;      CHAR RETRIEVED FROM TABLE IS TOGGLE ALTERNATE CHAR SET COMMAND.
          C      ;
0779'  3A 237A  C      LD      A,(CFLAG)      ;RETRIEVE CURRENT ATTRIBUTE FLAG BYTE
077C'  EE 08    C      XOR      ALT_FLG      ;TOGGLE THE ALTERNATE CHAR SET BIT
077E'  32 237A  C      LD      (CFLAG),A      ;SAVE NEW ATTR BYTE
0781'  AF       C      XOR      A              ;CLEAR Z FLAG TO INDICATE INVALID CHAR
0782'  C9       C      RET                    ;RETURN TO CALLER
          C
          C      PAGE
```

```

C
C
0783' C SAVEKEY: ;
C ; ROUTINE TO SEND CHARACTER FROM KEYBOARD TO BIOS.
C ;ENTRY:
C ; A = CHAR TO SEND TO BIOS
C ;EXIT:
C ; NONE.
C ;
C
C ;TRY TO BUFFER THE CHAR.
0783' 4F C LD C,A ;PUT CHAR IN C
0784' 06 00 C LD B,KBD_DEV ;SPECIFY KEYBOARD DEVICE
0785' C5 C PUSH BC ;SAVE
C
0787' CD 11B6' C CALL FL_B_ST ;GET STATUS
C
078A' C1 C POP BC
078B' C8 C RET Z ;IF FULL
C
078C' 3A 2358 C LD A,(KCLICK) ;IS KEY CLICK FLAG SET?
078F' B7 C OR A
0790' CA 079A' C JP Z,XJ099 ;NO...
C ;
C ;YES, TOGGLE SPEAKER
0793' DB 02 C IN A,(SYS_DTB) ;GET PIA REG WITH SPEAKER TOGGLE
0795' F6 08 C OR SPEAKER ;TURN SPEAKER ON
0797' D3 02 C OUT (SYS_DTB),A
0799' F5 C PUSH AF ;SAVE PIA STATE
C
079A' XJ099:
079A' CD 11C5' C CALL FL_B_CH ;PUT CHAR IN
079D' 2A 2421 C LD HL,(D_RT_TBL+D_RT_LEN*KBD_DEV+4) ; BUMP CHAR COUNT
07A0' 23 C INC HL
07A1' 22 2421 C LD (D_RT_TBL+D_RT_LEN*KBD_DEV+4),HL ; AND SAVE IT BACK
C
07A4' 3A 2358 C LD A,(KCLICK) ;IS KEY CLICK FLAG SET?
07A7' B7 C OR A
07A8' C8 C RET Z ;NO...RETURN
C ;
C ;YES, TURN SPKR OFF
07A9' F1 C POP AF ;RESTORE PIA STATE
07AA' E6 F7 C AND NOT SPEAKER ;TURN SPEAKER OFF
07AC' D3 02 C OUT (SYS_DTB),A
07AE' C9 C RET
C
C PAGE

```



```

C
C
C      ; *****
C      ; *
C      ; *          KEYBOARD TABLE EQUATES
C      ; *
C      ; *****
C
C      ;KEYBOARD TABLE NUMBERS
0000  C      NM_TB_NO      EQU      0          ;NORMAL TABLE NUMBER
0001  C      SH_TB_NO      EQU      NM_TB_NO + 1      ;SHIFT TABLE NUMBER
0002  C      CT_TB_NO      EQU      SH_TB_NO + 1      ;CONTROL TABLE NUMBER
0003  C      AL_TB_NO      EQU      CT_TB_NO + 1      ;ALPHA LOCK TABLE NUMBER
0004  C      CS_TB_NO      EQU      AL_TB_NO + 1      ;CONTROL SHIFT TABLE NUMBER
C
0040  C      KTBL_LEN      EQU      64          ;LENGTH OF KEYBD XLATE TABLES

```

PAGE

'EXECUTIVE ROM MONITOR'
'KEYBOARD ROUTINES'

MACRO-80 3.44 09-Dec-81 PAGE 1-200

SUBTTL 'I/O DRIVERS'
PAGE

```

C      SUBTTL      'I/O DRIVERS'
C      INCLUDE ROM4      ;I/O ROUTINES
C      ;
C      ;
C      ; 83-08-01      Fixed parameter passing bug in IE.IDM
C      ;
C      ; *****
C      ; *
C      ; *          I / O DRIVERS          *
C      ; *
C      ; *****
C      ;
C      ; MODULE 4 --ROM4-- ENTRY POINTS
C      ;
C      ;          I / O ROUTINES
C      ;
C      ; IE.CO:      CONTROL OUT (BIOS CALL 1)
C      ; IE.SI:      STATUS IN (BIOS CALL 2)
C      ; IE.GTS:     GO TO STANDBY (BIOS CALL 3)
C      ; IE.TC:      TAKE CONTROL (BIOS CALL 4)
C      ; IE.OIM:     OUTPUT INTERFACE MESSAGE (BIOS CALL 5)
C      ; IE.ODM:     OUTPUT DEVICE MESSAGE (BIOS CALL 6)
C      ; IE.IDM:     INPUT DEVICE MESSAGE (BIOS CALL 7)
C      ; IE.PP:      PARALLEL POLL (BIOS CALL 8)
C      ; IEINSTAT:   GET STATUS OF INPUT DEVICE AT IEEE PORT
C      ; IEINP:      INPUT A CHAR. FROM IEEE PORT
C      ; IEOUT:      OUTPUT A CHAR. TO IEEE PORT
C      ; CV2OP:      INITIALIZE TO PARALLEL OUTPUT PORT
C      ; CV2IP:      INITIALIZE TO PARALLEL INPUT PORT
C      ; POSTAT:     GET STATUS OF OUTPUT DEVICE AT PARALLEL PORT
C      ; PISTAT:     GET STATUS OF INPUT DEVICE AT PARALLEL PORT
C      ; PARINP:     INPUT A CHAR. FROM PARALLEL PORT
C      ; PAROUT:     OUTPUT A CHAR TO PARALLEL PORT
C      ; CTC_INIT:   INITIALIZE THE CLOCK-TIMER CHIP
C      ; SIOINIT:    INITIALIZE THE SERIAL PORT
C      ; SETBAUD:    SET BAUD RATE FOR SERIAL PORT
C      ; SERAIN:     INPUT A CHAR. FROM SERIAL PORT A
C      ; SERBIN:     INPUT A CHAR. FROM SERIAL PORT B
C      ; SERAOUT:    OUTPUT A CHAR. FROM SERIAL PORT A
C      ; SERBOUT:    OUTPUT A CHAR. FROM SERIAL PORT B
C      ; SAO_STAT:   OUTPUT STATUS FROM SERIAL PORT A
C      ; SAI_STAT:   INPUT STATUS FROM SERIAL PORT A
C      ; GET_O_STAT: GET OUTPUT STATUS FROM SERIAL PORT
C      ; GET_I_STAT: GET INPUT STATUS FROM SERIAL PORT
C      ; SETDMA:     SET DMA ADDRESS AND BANK
C
C      PAGE

```

```
C
C
C      .Z80
C      ;
C      ; .XLIST
C      ; INCLUDE      SYSS1.MAC      ;SYSTEST FILE
C      ; INCLUDE      RAMSMON.MAC    ;MONITOR RAM MEMORY LOCATIONS
C      ; INCLUDE      RAMS10.MAC     ;RAM MEMORY LOCATIONS AND CONSTANTS
C      ; .LIST
C      ; ;SYSS1.MAC , RAMSMON.MAC AND RAMS10.MAC INCLUDED
C
C      ; EXT      RM_C_BF,FL_B_CH,FL_B_ST,RM_C_ST
C
C      ; PUBLIC IE.CO,IE.SI,IE.6TS,IE.TC,IE.OIM,IE.ODM,IE.IDM,IE.PP,IE.INSTAT
C      ; PUBLIC IE.OSTAT,IE.INP,IE.OUT,POSTAT,PISTAT,PARINP,PAROUT,CTC_INIT
C      ; PUBLIC SIOINIT,SETBAUD,SERAIN,SERBIN,SERAOUT,SERBOUT,SAO_STAT,SBO_STAT
C      ; PUBLIC SETDMA,SAI_STAT,SBI_STAT
C
C      CSEG
C
C      PAGE
```

07AF'

```

C
C
C      ;      +-----+
C      ;      | ENTERED 05/01/81 FROM TNW XEROX, SEH. |
C      ;      +-----+
C
C
C      ;LAST EDITED AT 09:29 ON 11 NOV 80
C
C      ;THERE ARE FOUR COMMANDS TO THE 6821
C
C      ;      00      PERIPHERAL/DIRECTION REGISTER A      PAR_DTA
C      ;      01      CONTROL REGISTER A                  PAR_CNA
C      ;      10      PERIPHERAL/DIRECTION REGISTER B      PAR_DTB
C      ;      11      CONTROL REGISTER B                  PAR_CNB
C
C      ;BIT 2 OF THE CONTROL REGISTER (A AND B) ALLOWS SELECTION OF EITHER
C      ;A PERIPHERAL INTERFACE REGISTER OR A DATA DIRECTION REGISTER.
C      ;A "1" IN BIT 2 SELECTS THE PERIPHERAL REGISTER.
C
C      ;THE TWO DATA DIRECTION REGISTERS ALLOW CONTROL OF THE DIRECTION
C      ;OF DATA THROUGH EACH CORRESPONDING PERIPHERAL DATA LINE.
C      ;A DATA DIRECTION REGISTER BIT SET AT "0" CONFIGURES
C      ;THE CORRESPONDING PERIPHERAL DATA LINE AS AN INPUT.
C
C      ;A RESET AT POWER UP HAS THE EFFECT OF ZEROING ALL PIA REGISTERS.
C      ;THIS WILL SET PA0-PA7, PB0-PB7, CA2, AND CB2 AS INPUTS,
C      ;AND ALL INTERRUPTS DISABLED.
C      ;SIGNALS ATN, REN, AND IFC WILL BE DRIVEN LOW
C      ;UNTIL INITIALIZED BY SOFTWARE.
C
C      ;DATA DIRECTION IS ALWAYS SET FOR OUTPUT FOR THE DATA REGISTER.
C      ;DATA MUST BE SET TO ALL ONES WHEN INPUTTING.
C      ;THE INTERFACE IS IN SOURCE HANDSHAKE MODE IF DATA ENABLE (PB0)
C      ;IS SET TO "0", AND IN ACCEPTOR HANDSHAKE MODE IF SET TO "1".
C      ;WHEN SWITCHING FROM SOURCE TO ACCEPTOR HANDSHAKE,
C      ;ATN WILL ALWAYS BE LOW.
C      ;TAKE CONTROL CAN ONLY BE CALLED FOLLOWING A GO TO STANDBY.
C      ;AFTER A FATAL ERROR, PERFORM AN IFC RESET.
C
C      ;STANDARD VALUES USED:
C
C      ;PAR_CNA      0011(IFC)(DIR)10
C
C      ;PAR_CNB      0011(REN)(DIR)00
C
C      ;PAR_DTA      SOURCE      DIRECTION      1111_1111
C      ;              DATA      DATA
C
C      ;PAR_DTB      ACCEPTOR    DIRECTION      1111_1111
C      ;              DATA      1111_1111
C
C      ;PAR_DTB      SOURCE      DIRECTION      0011_1111
C      ;              DATA      000A_0010      ;A = ATN
C

```

'EXECUTIVE ROM MONITOR'
'I/O DRIVERS'

MACRO-80 3.44 09-Dec-81

PAGE 1-204

C	;PAR_DTB	ACCEPTOR	DIRECTION	1101_0111
C	i		DATA	0100_0101
C				
C		PAGE		

```

C
C
C      ;PIA SIGNAL DEFINITIONS:
C      ;ALL SIGNALS ARE LOW ON THE IEEE BUS WHEN PIA REGISTER CONTAINS "1".
C
C      ;      PA0      DIO 1
C      ;      PA1      DIO 2
C      ;      PA2      DIO 3
C      ;      PA3      DIO 4
C      ;      PA4      DIO 5
C      ;      PA5      DIO 6
C      ;      PA6      DIO 7
C      ;      PA7      DIO 8
C
C      ;      CA1      SRQ
C      ;      CA2      IFC
C
C      ;      PB0      ENABLE DATA OUT      (ENABLED WHEN "0")
C      ;      PB1      ENABLE NDAC/NRFD      (ENABLED WHEN "0")
C      ;      PB2      ENABLE EDI/DAV        (ENABLED WHEN "0")
C      ;      PB3      EDI
C      ;      PB4      ATN
C      ;      PB5      DAV
C      ;      PB6      NDAC
C      ;      PB7      NRFD
C
C      ;      CB1      NOT USED
C      ;      CB2      REN
C
C      ;CONTROL WORD FORMAT
C
C      ;[ 7 ][ 6 ][ 5 ][ 4 ][ 3 ][ 2 ][ 1 ][ 0 ]
C
C      ;[IRQA1][IRQA2][  CA2 CONTROL  ][ DDRA ][ CA1 CONTROL]
C      ;[IRQB1][IRQB2][  CB2 CONTROL  ][ DDRB ][ CB1 CONTROL]
C
C      ;      IRQA1  0      INTERRUPT FLAG SET BY FALL OF SRQ
C      ;      IRQA2  0      NOT USED
C      ;      CA2    110     SET IFC HIGH
C      ;      CA2    111     SET IFC LOW
C      ;      DDRA   0      R/W DATA DIRECTION REGISTER A
C      ;      DDRA   1      R/W PERIPHERAL REGISTER A
C      ;      CA1    10     SET IRQA1 HIGH ON RISE OF SRQ
C
C      ;      IRQB1  0      NOT USED
C      ;      IRQB2  0      NOT USED
C      ;      CB2    110     SET REN HIGH
C      ;      CB2    111     SET REN LOW
C      ;      DDRB   0      R/W DATA DIRECTION REGISTER B
C      ;      DDRB   1      R/W PERIPHERAL REGISTER B
C      ;      CB1    00     NOT USED
C
C      PAGE

```

```
C
C
C      ; *****
C      ; *
C      ; *          BIOS CALLS          *
C      ; *
C      ; *****
C
C      ;BIOS CALL 1:  CONTROL OUT
C
C      ;      CAN BE CALLED WHILE IN ANY STATE.
C
C      ;      EXITS IN THE CONTROLLER STANDBY STATE (ATN HIGH),
C      ;      SOURCE HANDSHAKE MODE
C
C      ;PARAMETER PASSED IN REGISTER C:
C
C      ;      BIT 0  IF "1", THE IFC SIGNAL IS SET LOW FOR 100 MICRO-SEC
C      ;      AND ALL PIA SIGNALS ARE INITIALIZED
C
C      ;      BIT 2 1
C      ;      0 X    NO ACTION
C      ;      1 0 SETS REN HIGH
C      ;      1 1 SETS REN LOW
C
C      PAGE
```



```

C
C
07AF' C IE.CO:
C ; MODIFIED SO AS NOT TO DESTROY THE STATE OF THE PARALLEL PIA
C ; DAB 11/15/82
C ;
C
C
07AF' F5 C PUSH AF
C
07B0' CB 41 C BIT 0,C ;CHECK IFC SUB-COMMAND
07B2' 28 34 C JR Z,XJ100
C
C ;INITIALIZE ALL IEEE-488 SIGNALS
C
07B4' DB 11 C IN A,(PAR_CNA)
07B6' F6 38 C OR 00111000B
07B8' D3 11 C OUT (PAR_CNA),A ;SET IFC-OUT LOW AND ENABLE DIRECTION REG
C
07BA' 3E FF C LD A,11111111B ;DIRECT DATA OUT
07BC' D3 10 C OUT (PAR_DTA),A
C
07BE' DB 11 C IN A,(PAR_CNA)
07C0' F6 04 C OR 00000100B ;ENABLE DATA REG
07C2' D3 11 C OUT (PAR_CNA),A
C
07C4' AF C XOR A
07C5' D3 10 C OUT (PAR_DTA),A
C
07C7' DB 13 C IN A,(PAR_CNB)
07C9' F6 30 C OR 00110000B ;SET REN-OUT HIGH
07CB' E6 F3 C AND 11110011B ;AND DISABLE DATA DIRECTION REG
07CD' D3 13 C OUT (PAR_CNB),A
C
07CF' 3E 3F C LD A,00111111B ;DIRECTION FOR SOURCE HANDSHAKE
07D1' D3 12 C OUT (PAR_DTB),A
07D3' DB 13 C IN A,(PAR_CNB) ;
07D5' F6 04 C OR 00000100B ;ENABLE DATA REG
07D7' D3 13 C OUT (PAR_CNB),A
C
07D9' 3E 02 C LD A,00000010B ;VALUES FOR SOURCE HANDSHAKE
07DB' D3 12 C OUT (PAR_DTB),A
C
C ;LEAVE IFC LOW FOR 100 MICRO-SEC
C
07DD' 3E 19 C LD A,25 ;DELAY 100 MICRO-SEC
C
07DF' 3D C XJ101: DEC A
07E0' 20 FD C JR NZ,XJ101
C
07E2' DB 11 C IN A,(PAR_CNA) ;
07E4' E6 F7 C AND 11110111B ;SET IFC HIGH
07E6' D3 11 C OUT (PAR_CNA),A
C
07E8' CB 51 C XJ100: BIT 2,C ;CHECK REN SUB-COMMAND

```

```
07EA' 28 0E      C      JR      Z,XJ102
                  C
                  C      ;SET/CLEAR REN
                  C
07EC' DB 13      C      IN      A,(PAR_CNB)
07EE' F6 30      C      OR      00110000B
07F0' E6 F7      C      AND     11110111B ;
07F2' CB 49      C      BIT     1,C
07F4' 28 02      C      JR      Z,XJ103
                  C
07F6' F6 08      C      OR      00001000B
                  C
07F8' D3 13      C      XJ103: OUT     (PAR_CNB),A
                  C
07FA' F1         C      XJ102: POP     AF
07FB' C9         C      RET
                  C
                  C      PAGE
```

```

C
C
07FC' C IE.SI:
C ;BIOS CALL 2. STATUS IN
C ;CAN BE CALLED ONLY WHILE IN SOURCE HANDSHAKE MODE.
C ;BIT 0 OF REGISTER A SET IF SRQ IS LOW
C
C
07FC' DB 10 C IN A, (PAR_DTA) ;CLEAR IRDA1
07FE' DB 12 C IN A, (PAR_DTB) ;PULSE ENABLE NDAC/NRFD
0800' CB 8F C RES 1, A
0802' D3 12 C OUT (PAR_DTB), A
0804' CB CF C SET 1, A
0806' D3 12 C OUT (PAR_DTB), A
0808' DB 11 C IN A, (PAR_CNA) ;SET SRQ VALUE IN A
080A' E6 80 C AND 10000000B
080C' 07 C RLCA
C
080D' C9 C RET
C
C PAGE

```

```

C
C
000E' C IE.GTS:
C ;BIOS CALL 3 GO TO STANDBY
C ;CAN BE CALLED ONLY WHILE IN SOURCE HANDSHAKE MODE
C ;ENTRY
C ;NONE
C
C
C
000E' F5 C PUSH AF
C
000F' 3E 02 C LD A,00000010B ;SET ATN HIGH
0011' D3 12 C OUT (PAR_DTB),A
0013' AF C XOR A ;FLOAT DATA BUS
0014' D3 10 C OUT (PAR_DTA),A
C
0016' F1 C POP AF
0017' C9 C RET
C
C PAGE

```

```

C
C
C      0010'      C      IE.TC:
C                  ;BIOS CALL 4 TAKE CONTROL
C                  ;CAN BE CALLED ONLY WHILE IN THE CONTROLLER STANDBY STATE (ATN HIGH).
C                  ;EXITS IN THE CONTROLLER ACTIVE STATE (ATN LOW), SOURCE HANDSHAKE MODE.
C                  ;BIT 0 OF REGISTER C SET TO TAKE CONTROL ASYNCHRONOUS
C
C                  ;EXIT
C                  ;A      =      ;ERROR CODE
C
C
C      0010'      C5      C      PUSH      BC
C
C      0019'      CB 41      C      BIT      0,C
C      001B'      0E 12      C      LD      C,PAR_DTB
C      001D'      20 29      C      JR      NZ,XJ104      ;IF ASYNCHRONOUS
C
C                  ;TAKE CONTROL SYNCHRONOUSLY
C
C      001F'      06 07      C      LD      B,00000111B
C      0021'      ED 41      C      OUT      (C),B      ;DISABLE DRIVERS
C      0023'      DB 13      C      IN      A,(PAR_CNB)
C      0025'      CB 97      C      RES      2,A
C      0027'      D3 13      C      OUT      (PAR_CNB),A
C      0029'      06 D7      C      LD      B,11010111B
C      002B'      ED 41      C      OUT      (C),B      ;DIRECTION REGISTER
C
C      002D'      CB D7      C      SET      2,A
C      002F'      D3 13      C      OUT      (PAR_CNB),A
C
C      0031'      06 85      C      LD      B,10000101B
C      0033'      ED 41      C      OUT      (C),B      ;SET NRFD LOW
C      0035'      3E 0C      C      LD      A,12
C
C      0037'      ED 40      C      XJ105: IN      B,(C)
C      0039'      CB 68      C      BIT      5,B
C      003B'      20 07      C      JR      Z,XJ106      ;DATA VALID HAS DROPPED
C
C      003D'      3D      C      DEC      A
C      003E'      20 F7      C      JR      NZ,XJ105      ;WAIT 100 MICRO-SEC
C
C      0040'      3E 81      C      LD      A,10000001B      ;SET DATA VALID TIMEOUT ERROR
C      0042'      18 1D      C      JR      XJ107
C
C      0044'      06 C5      C      XJ106: LD      B,11000101B
C      0046'      ED 41      C      OUT      (C),B      ;SET NDAC LOW
C
C      0048'      ED 40      C      XJ104: IN      B,(C)
C      004A'      CB E0      C      SET      4,B      ;SET ATN LOW
C      004C'      ED 41      C      OUT      (C),B
C
C                  ;SET-UP FOR SOURCE HANDSHAKE
C

```

'I/O DRIVERS'

```
004E' DB 13      C      IN      A, (PAR_CNB)
0050' CB 97      C      RES      2, A
0052' D3 13      C      OUT      (PAR_CNB), A
0054' 06 3F      C      LD       B, 00111111B
0056' ED 41      C      OUT      (C), B      ;DIRECTION REGISTER
C
C
C
0058' CB D7      C      SET      2, A
005A' D3 13      C      OUT      (PAR_CNB), A
005C' 06 12      C      LD       B, 00010010B
005E' ED 41      C      OUT      (C), B      ;CONTROL SIGNAL INITIAL VALUE
C
C
0060' AF          C      XOR      A      ;CLEAR ERROR CODE
C
C
0061' C1          C      XJ107: POP      BC
0062' C9          C      RET
C
C      PAGE
```

```

C
C
0863' C      IE.OIM:
C      ;BIOS CALL 5 OUTPUT INTERFACE MESSAGE
C      ;CAN BE CALLED WHILE IN ANY MODE OR STATE
C      ;EXITS IN THE SOURCE HANDSHAKE MODE WITH ATN LOW.
C
C      ;ENTRY
C      ;C      =      MULTI-LINE MESSAGE
C
C      ;EXIT
C      ;A      =      ERROR CODE
C
C
0863' C5      C      PUSH      BC
C
0864' DB 12    C      IN        A,(PAR_DTB)
0866' CB E7    C      SET      4,A      ;SET ATN LOW
0868' D3 12    C      OUT      (PAR_DTB),A
086A' CB 47    C      BIT      0,A
086C' 28 2C    C      JR        Z,IE.SHK
C
C      ;SET-UP FOR SOURCE HANDSHAKE
C
086E' 3E 17    C      LD        A,00010111B
0870' D3 12    C      OUT      (PAR_DTB),A      ;DISABLE DRIVERS
C
0872' DB 13    C      IN        A,(PAR_CNB)
0874' CB 97    C      RES      2,A
0876' D3 13    C      OUT      (PAR_CNB),A
0878' 47      C      LD        B,A
C
0879' 3E 3F    C      LD        A,00111111B
087B' D3 12    C      OUT      (PAR_DTB),A      ;DIRECTION REGISTER
C
087D' 78      C      LD        A,B
087E' CB D7    C      SET      2,A
0880' D3 13    C      OUT      (PAR_CNB),A
C
C      ;FLOAT EXTERNAL DATA BUS
C
0882' AF      C      XOR        A
0883' D3 10    C      OUT      (PAR_DTB),A
0885' 3E 12    C      LD        A,00010010B
0887' D3 12    C      OUT      (PAR_DTB),A      ;CONTROL SIGNAL INITIAL VALUE
C
0889' 18 0F    C      JR        IE.SHK
C
C      PAGE
C

```

```

C
C
088B' C
C IE.DDM:
C ;BIOS CALL 6 OUTPUT DEVICE MESSAGE
C ;CAN BE CALLED ONLY WHILE IN THE SOURCE HANDSHAKE MODE WITH ATN HIGH OR LOW.
C ;EXITS IN THE SOURCE HANDSHAKE MODE WITH ATN HIGH.
C
C ;ENTRY
C ;C = MULTI-LINE MESSAGE
C ;B = EOI REQUEST
C ;EXIT
C ;A = ERROR CODE
C
C
088B' C5 C PUSH BC
C
088C' DB 12 C IN A, (PAR_DTB)
088E' CB A7 C RES 4, A ;SET ATN HIGH
0890' D3 12 C OUT (PAR_DTB), A
C
0892' CB 40 C BIT 0, B ;CHECK IF EOI REQUESTED
0894' 20 04 C JR Z, IE.SHK
C
0896' CB DF C SET 3, A
0898' D3 12 C OUT (PAR_DTB), A
C
C ;PERFORM SOURCE HANDSHAKE
C
089A' DB 12 C IE.SHK: IN A, (PAR_DTB)
089C' CB 6F C BIT 5, A ;
089E' 20 20 C JR NZ, XJ100 ;DAC TIMEOUT RE-ENTRY
C
08AA' 79 C LD A, C
08A1' D3 10 C OUT (PAR_DTB), A ;PLACE DATA ON BUS
C
C ; WAIT FOR READY
C
08A3' 0E 0A C LD C, 10
08A5' DB 12 C XJ109: IN A, (PAR_DTB)
08A7' CB 7F C BIT 7, A
08A9' 20 07 C JR Z, XJ110 ;READY FOR DATA
C
08AB' 0D C DEC C
08AC' 20 F7 C JR NZ, XJ109 ;WAIT FOR 100 MICRO-SEC
C
08AE' 3E 02 C LD A, 10000010B ;SET RFD TIMEOUT ERROR
08B0' 10 20 C JR XJ111
C
08B2' DB 12 C XJ110: IN A, (PAR_DTB)
08B4' CB 77 C BIT 6, A
08B6' 20 04 C JR NZ, XJ112 ;DATA ACCEPTED LOW
C
08B8' 3E 01 C LD A, 10000001B ;SET DEVICE NOT PRESENT ERROR
08BA' 10 1E C JR XJ111

```


'I/O DRIVERS'

		C			
08BC'	CB EF	C	XJ112:	SET 5,A	;SET DAV LOW
08BE'	D3 12	C		OUT (PAR_DTB),A	
		C			
		C	;	WAIT FOR DATA TO BE ACCEPTED	
08C0'	0E 64	C	XJ108:	LD C,100	
		C		PAGE	

```

C
C
08C2' DB 12 C XJ113: IN A, (PAR_DTB)
08C4' CB 77 C BIT 6, A
08C6' 28 07 C JR Z, XJ114 ;DATA ACCEPTED
C
08C8' 0D C DEC C
08C9' 20 F7 C JR NZ, XJ113 ;WAIT 1000 MICRO-SEC
C
08CB' 3E 84 C LD A, 10000100B ;SET DAC TIMEOUT ERROR
08CD' 18 0B C JR XJ111
C
08CF' DB 12 C XJ114: IN A, (PAR_DTB)
08D1' CB AF C RES 5, A ;SET DAV HIGH
08D3' CB 9F C RES 3, A ;SET EDI HIGH
08D5' D3 12 C OUT (PAR_DTB), A
C
08D7' AF C XOR A ;REMOVE DATA FROM BUS
08D8' D3 10 C OUT (PAR_DTB), A
C
08DA' C1 C XJ111: POP BC
08DB' C9 C RET
C
C PAGE

```

```

C
C
08DC' C IE.IDM:
C ;BIOS CALL 7 INPUT DEVICE MESSAGE
C ;CAN BE CALLED WHILE IN ANY MODE OR STATE
C ;EXITS IN THE ACCEPTOR HANDSHAKE MODE WITH ATN HIGH.
C ;EXIT
C ;L = ERROR CODE
C ;A = DEVICE MESSAGE
C ;H = DEVICE MESSAGE
C
C
08DC' C5 C PUSH BC
C
08DD' DB 12 C IN A,(PAR_DTB)
08DF' CB 47 C BIT 0,A
08E1' 20 20 C JR NZ,XJ115
C
C ;SET-UP FOR ACCEPTOR HANDSHAKE
C
08E3' 06 17 C LD B,00010111B
08E5' 0E 12 C LD C,PAR_DTB
08E7' ED 41 C OUT (C),B ;DISABLE DRIVERS
C
08E9' DB 13 C IN A,(PAR_CNB)
08EB' CB 97 C RES 2,A
08ED' D3 13 C OUT (PAR_CNB),A
C
08EF' 06 D7 C LD B,11010111B
08F1' ED 41 C OUT (C),B ;DIRECTION REGISTER
C
08F3' CB D7 C SET 2,A
08F5' D3 13 C OUT (PAR_CNB),A
C
08F7' 3E FF C LD A,11111111B ;FLOAT INTERNAL DATA BUS
08F9' D3 10 C OUT (PAR_DTB),A
08FB' 3E 55 C LD A,01010101B
08FD' D3 12 C OUT (PAR_DTB),A ;CONTROL SIGNALS INITIAL VALUE
C
08FF' 3E 45 C LD A,01000101B ;SET ATN HIGH
0901' D3 12 C OUT (PAR_DTB),A
C
C ;PERFORM ACCEPTOR HANDSHAKE
C
0903' DB 12 C XJ115: IN A,(PAR_DTB)
0905' CB 77 C BIT 6,A
0907' 20 20 C JR Z,XJ116 ;DATA INVALID TIMEOUT ERROR RE-ENTRY
C
0909' CB BF C RES 7,A ;SET NRFD HIGH
090B' D3 12 C OUT (PAR_DTB),A
C
090D' 06 0C C LD B,12
090F' DB 12 C XJ117: IN A,(PAR_DTB)
0911' CB 6F C BIT 5,A

```

'I/O DRIVERS'

0913'	20 07	C	JR	NZ, XJ118	;DATA VALID
		C			
0915'	10 F8	C	DJNZ	XJ117	;WAIT 100 MICRO-SEC
		C			
0917'	21 0082	C	LD	HL, 10000010B	;SET DATA VALID TIMEOUT ERROR
091A'	18 29	C	JR	XJ119	
		C	PAGE		

```

C
C
091C' CB FF C XJ118: SET 7,A ;SET NRFD LOW
091E' D3 12 C OUT (PAR_DTB),A
0920' DB 10 C IN A,(PAR_DTB) ;READ DATA
0922' 67 C LD H,A
0923' 2E 00 C LD L,0
0925' DB 12 C IN A,(PAR_DTB) ;READ EDI
0927' CB 5F C BIT 3,A
0929' 20 02 C JR Z,XJ120
C
092B' CB C5 C SET 0,L ;L=1 8/1/83 AOS
C
092D' CB B7 C XJ120: RES 6,A ;SET NDAC HIGH
092F' D3 12 C OUT (PAR_DTB),A
C
0931' 06 78 C XJ116: LD B,120
C
0933' DB 12 C XJ121: IN A,(PAR_DTB)
0935' CB 6F C BIT 5,A
0937' 20 08 C JR Z,XJ122 ;DATA VALID DROPPED
C
0939' 10 F8 C DJNZ XJ121 ;WAIT 1000 MICRO-SEC
C
093B' CB D5 C SET 2,L ;SET DATA INVALID TIMEOUT ERROR
093D' CB FD C SET 7,L
093F' 10 04 C JR XJ119
C
0941' CB F7 C XJ122: SET 6,A ;SET NDAC LOW
0943' D3 12 C OUT (PAR_DTB),A
C
0945' 7C C XJ119: LD A,H
C
0946' C1 C POP BC
0947' C9 C RET
C
C PAGE

```

```

C
C
0948' C IE.PP:
C ;BIDS CALL 8 PARALLEL POLL
C ;CAN BE CALLED ONLY WHILE IN THE SOURCE HANDSHAKE MODE WITH ATN HIGH OR LOW.
C ;EXITS IN THE SOURCE HANDSHAKE MODE WITH ATN LOW.
C
C ;EXIT
C ;A = PARALLEL POLL VALUE
C
C
0948' 3E 1B C LD A,00011011B ;FORM PARALLEL POLL
094A' D3 12 C OUT (PAR_DTB),A
094C' 3E FF C LD A,11111111B ;FLOAT INTERNAL DATA BUS
094E' D3 10 C OUT (PAR_DTA),A
C
0950' DB 10 C IN A,(PAR_DTA) ;READ PARALLEL POLL DATA
0952' F5 C PUSH AF ;SAVE
C
0953' AF C XOR A
0954' D3 10 C OUT (PAR_DTA),A ;RE-STORE SOURCE HANDSHAKE MODE
C
0956' 3E 12 C LD A,00010010B
0958' D3 12 C OUT (PAR_DTB),A
C
095A' F1 C POP AF ;RESTORE PARALLEL POLL VALUE
095B' C9 C RET
C
C PAGE

```

```
C
C
C      ; *****
C      ; *
C      ; *          IEEE DRIVERS          *
C      ; *
C      ; *****
C      ;[ ]
C
C      ;IEEE DRIVERS:
C
C      ;THE ROUTINES IEINSTAT, IEINP AND IEOUT ARE USED TO
C      ;TRANSFER CHARACTERS TO AND FROM AN IEEE DEVICE ATTACHED TO THE
C      ;OSBORNE IEEE PORT. THE ADDRESS OF THE DEVICE IS SPECIFIED IN
C      ;THE CELL IE_ADRS.
C      ;THE FUNCTION IEINSTAT RETURNS THE STATUS OF THE INPUT DEVICE.
C      ;UNFORTUNATELY THERE IS NO STANDARD WAY BY WHICH AN IEEE DEVICE
C      ;INDICATES THAT IT HAS A CHARACTER. IN ORDER TO DETERMINE THIS, ONE
C      ;HAS TO READ THE CHARACTER DEVICE. AS A CP/M TRANSIENT CAN CALL
C      ;IEINSTAT MANY TIMES BEFORE CALLING IEINP TO READ A CHAR, AND IEINSTAT
C      ;HAS TO READ THE CHAR TO DETERMINE THE STATUS, THE CHARACTER READ HAS TO
C      ;BE BUFFERED UNTIL CALL TO IEINP IS MADE. IEINSTAT READS THE DEVICE
C      ;ONLY WHEN THE BUFFER IS EMPTY. AS ZEROS ARE USED TO INDICATE
C      ;THAT THE BFR IS EMPTY, A NULL CHARACTER CAN NOT BE READ FROM THE
C      ;IEEE DEVICE.
C
C      PAGE
```

```

C
C
095C' C      IEINSTAT:
C      ;GETS STATUS OF THE INPUT DEVICE ATTACHED TO IEEE PORT
C      ;IF A CHAR IS PRESENT IN IE_CHAR THEN RETURN WITH 0FFH STATUS
C      ;ELSE
C      ;MAKE DEVICE TALKER
C      ;READ THE DEVICE
C      ;IF CHAR READ THEN
C      ; STORE IN BFR
C      ;MAKE UNTALK
C      ;RETURN WITH STATUS OF BUFFER
C
C
C
095C' 3A 23DC C      LD      A, (IE_CHAR)
095F' B7      C      OR      A
0960' 20 03   C      JR      Z, IEI10      ;IF CHAR PRESENT THEN
C
0962' C      IEOSTAT:
C      ;RETURNS STATUS OF IEEE
C      ;IEEE ALWAYS APPEARS TO BE READY
0962' F6 FF   C      OR      0FFH      ;RETURN WITH 0FFH STATUS
0964' C9      C      RET
C
C      PAGE

```



```

C
C
0965' C IEI10:
C ;MAKE TALKER
C
0965' 3A FD52 C LD A, (IE_ADR)
0968' C6 40 C ADD A, IE_TALK ;GET PRIMARY ADDRESS
096A' 4F C LD C, A
096B' CD 0863' C CALL IE.OIM ;OUTPUT INTERFACE MESSAGE
096E' B7 C OR A
096F' 20 F4 C JR NZ, IEI10 ;TRY AGAIN IF ERROR
C
0971' C IEI20:
C ;READ A CHAR.
C
0971' CD 08DC' C CALL IE.IDM
0974' CB 7D C BIT 7, L
0976' 28 01 C JR Z, IEI30 ;IF ERROR THEN
C
0978' AF C XOR A ;INICATE NO CHAR RECVD
C
0979' 32 23DC C IEI30: LD (IE_CHAR), A ;STOR THE CHAR
C
097C' C IEI40:
C ;MAKE UNTALK
C
097C' 0E 5F C LD C, IE_UTLK
097E' CD 0863' C CALL IE.OIM
0981' B7 C OR A
0982' 20 F8 C JR NZ, IEI40
C
C ;RETURN WITH STATUS OF THE CHAR
C
0984' 3A 23DC C LD A, (IE_CHAR)
0987' B7 C OR A
0988' C8 C RET Z
C
0989' F6 FF C OR 0FFH
098B' C9 C RET
C
C PAGE

```

```

C
C
098C'      C      IEINP:
C          C      ;READS A CHARACTER FROM IEEE PORT
C
C
C
098C'      CD 095C'      C      CALL    IEINSTAT
098F'      28 FB        C      JR      Z, IEINP      ;WAIT TILL CHAR AVAIL
C
C          C      LD      HL, IE_CHAR
0991'      21 23DC      C          LD      A, (HL)
0994'      7E            C          LD      (HL), 0      ;CLEAR THE BUFFER
0995'      36 00        C          RET
0997'      C9            C
C          C      PAGE
C
```

```

C
C
0998' C IEOUT:
C ;OUTPUTS THE CHARACTER IN REG C TO IEEE PORT
C ;USES ROM RESIDENT PRIMITIVES.
C
C
C
C
0998' C5 C PUSH BC ;SAVE THE CHAR
C ;MAKE LISTENER
0999' 3A FD52 C IE005: LD A, (IE_ADR)
099C' C6 20 C ADD A, IE_LSTN ;GET PRIMARY ADDRESS
099E' 4F C LD C, A
099F' CD 0863' C CALL IE.OIM ;OUTPUT INTERFACE MESSAGE
09A2' B7 C OR A
09A3' 20 F4 C JR NZ, IE005 ;TRY AGAIN IF ERROR
C
09A5' C1 C POP BC
09A6' 06 00 C LD B, 0 ;DO NOT SEND EOI
C
09A8' C5 C IE022: PUSH BC ;SAVE CHAR AGAIN IN CASE OF RETRY
09A9' CD 088B' C CALL IE.ODM
09AC' C1 C POP BC
09AD' B7 C OR A
09AE' 20 F8 C JR NZ, IE022 ;TRY AGAIN IF ERROR
C
09B0' C IE040:
C ;MAKE UNLISTEN
C
C
09B0' 0E 3F C LD C, IE_ULST
09B2' CD 0863' C CALL IE.OIM
09B5' B7 C OR A
09B6' 20 F8 C JR NZ, IE040
C
09B8' C9 C RET
C
C PAGE

```

```
C
C
C      ; *****
C      ; *
C      ; *          PARALLEL PORT          *
C      ; *
C      ; *****
C
C      ;THE PARALLEL PORT IS ACTUALLY THE IEEE PORT DRIVEN WITH THE CENTRONIX
C      ;PROTOCOL.  THE BIT ASSIGNMENTS OF THE PIA AND PIB ARE AS FOLLOWS:
C      ;PIA0-7 = DATA BUS
C      ;PIB0 = 0, DATA BUS IS OUTPUT. 1, DATA BUS IS INPUT
C      ;PIB1 = SET TO 1.
C      ;PIB2 = SET TO 0.
C      ;PIB3 = 0 OUTPUT, 1 INPUT
C      ;PIB4 = NOT USED
C      ;PIB5 = OUTPUT STROBE. ACTIVE = 1.
C      ;PIB6 = 0, PRINTER BUSY. 1, PRINTER IS READY.
C      ;PIB7 = NOT USED.
C
C      ;CA2 = GOING LOW INDICATES TO DEVICE THAT WE ARE BUSY.
C      ;CA1 = LOW TO HIGH TRANSITION GATES INPUT DATA TO PORT A.
C
C      ;THE PORT IS BIDIRECTIONAL BUT ONLY ONE DIRECTION
C      ;CAN BE ACTIVE AT ANY TIME. THE DIRECTION OF PORT IS DETERMINED
C      ;BY WHICH ROUTINES ARE CALLED.  IF POSTAT OR PAROUT ARE
C      ;CALLED, IT IS MADE AN OUTPUT PORT AND AN INPUT PORT IF
C      ;PISTAT OR PARINP ARE CALLED.
C
C      PAGE
```

LD B, PP. OUT
LD E, PA. DRD
LD D, PB. DTO
JP CV2IOP
PAGE

```

C
C
09C2'      C      CV2IP:
C          ;INITIALIZES THE PORT TO A PARALLEL INPUT PORT.
C
C
C
C
09C2'      06 02      C          LD      B,PP.IN
09C4'      1E 00      C          LD      E,PA.DRI
09C6'      16 0B      C          LD      D,PB.DTI
C          ;      JMP      CV2IOP      ; FALL THROUGH CV2IOP:
C
C
09C8'      C      CV2IOP:
09C8'      3A 23DB      C          LD      A,(PP.MODE)
09CB'      B8          C          CP      B
09CC'      C8          C          RET      Z      ;RETURN WHEN IN CORRECT MODE
C
09CD'      3E 2A      C          LD      A,PA.CDR
09CF'      D3 11      C          OUT     (PAR_CNA),A      ;SELECT DIRECTION REG
09D1'      7B          C          LD      A,E
09D2'      D3 10      C          OUT     (PAR_DTA),A      ;OUTPUT CONSTANT TO DIR. REG TO PUT A PORT IN INPUT MODE
C
09D4'      3E 2E      C          LD      A,PA.CDT
09D6'      D3 11      C          OUT     (PAR_CNA),A      ;SELECT PORT A DATA REG.
C
09D8'      3E 00      C          LD      A,PB.CDR
09DA'      D3 13      C          OUT     (PAR_CNB),A      ;SELECT PORT B DIRECTION
09DC'      3E BF      C          LD      A,PB.DR
09DE'      D3 12      C          OUT     (PAR_DTB),A      ;ALL LINES ARE OUTPUT EXCEPT THE OUTPUT BUSY SIGNAL ON BIT 6
C
09E0'      3E 04      C          LD      A,PB.CDT
09E2'      D3 13      C          OUT     (PAR_CNB),A      ;SELECT DATA REGISTER
09E4'      7A          C          LD      A,D
09E5'      D3 12      C          OUT     (PAR_DTB),A      ;INITIALIZE PORT B DATA
C
09E7'      78          C          LD      A,B
09E8'      32 23DB      C          LD      (PP.MODE),A
09EB'      C9          C          RET
C
C          PAGE

```

```

C
C
09EC' C POSTAT:
C ;GETS STATUS OF THE PARALLEL (CENTRONIX) PRINTER ATTACHED TO THE IEEE PORT
C
C
C
C
09EC' CD 09B9' C CALL CV2OP ;CONVERT TO OUTPUT
09EF' DB 12 C IN A, (PAR_DTB) ;GET PORT B DATA
09F1' E6 40 C AND PP.ORDY
09F3' C8 C RET Z
C
09F4' F6 FF C OR 0FFH
C
09F6' C9 C POS10: RET
C
C PAGE
C
```

```

C
C
09F7' C PISTAT:
C ;GETS STATUS OF THE INPUT DEVICE ATTACHED TO THE PARALLEL PORT
C
C
C
C
09F7' CD 09C2' C CALL CV2IP
09FA' 3A 23DA C LD A, (PIACTL)
09FD' E6 80 C AND PP.IRDY
09FF' 20 08 C JR NZ, PIS20 ;IF SAVED STATUS INDICATES THERE IS A CHAR IN THE PIA
C
0A01' DB 11 C IN A, (PAR_CNA)
0A03' 32 23DA C LD (PIACTL), A ;THIS IS SAVED AS READING THE
C ;PIA CLEARS THE STATUS
0A06' E6 80 C AND PP.IRDY
0A08' C8 C RET Z
C
0A09' F6 FF C PIS20: OR 0FFH
0A0B' C9 C RET
C
C PAGE

```



```

C
C
0A0C' C PARINP:
C ;INPUTS A CHARACTER FROM PARALLEL PORT.
C
C
C
0A0C' CD 09F7' C CALL PISTAT
0A0F' 28 FB C JR Z,PARINP ;WAIT TILL CHAR IN PIA
C
C
0A11' AF C XOR A
0A12' 32 23DA C LD (PIACTL),A ;CLEAR SAVED STATUS
0A15' DB 10 C IN A,(PAR_DTA)
0A17' 2F C CPL ;INVERT DATA
0A18' 4F C LD C,A ;ALSO IN C
0A19' C9 C RET
C
C PAGE
```

		C	
		C	
0A1A'		C	PAROUT:
		C	;OUTPUTS THE CHARACTER IN C TO THE IEEE PORT TREATING THE PORT AS A PARALLEL PORT.
		C	
		C	
		C	
0A1A'	CD 09EC'	C	CALL POSTAT
0A1D'	2B FB	C	JR Z, PAROUT
		C	
0A1F'	79	C	LD A, C
0A20'	2F	C	CPL ; INVERT DATA
0A21'	D3 10	C	OUT (PAR_DTA), A
0A23'	3E 22	C	LD A, PB.DTO+STRB
0A25'	D3 12	C	OUT (PAR_DTB), A ; SET STROBE
0A27'	3E 02	C	LD A, PB.DTO
0A29'	D3 12	C	OUT (PAR_DTB), A ; CLEAR STROBE
0A2B'	C9	C	RET
		C	
		C	PAGE

```

C
C
0A2C' C CTC_INIT:
C ; INITIALIZE THE CLOCK-TIMER CHIP.
C ; ENTRY:
C ; NONE
C ;
C ; EXIT:
C ; NONE
C ;
0A2C' 3E 34 C LD A,00110100B ;SET COUNTER 0 TO RATE GENERATOR MODE
C ;AND TWO BYTE BINARY PROGRAMMING
0A2E' D3 07 C OUT (TIM_CNL),A
0A30' 3E 74 C LD A,01110100B ;SET COUNTER 1 TO RATE GENERATOR MODE
C ;AND TWO BYTE BINARY PROGRAMMING
0A32' D3 07 C OUT (TIM_CNL),A
0A34' 3E B6 C LD A,10110110B ;SET COUNTER 2 TO SQUARE WAVE MODE
C ;AND TWO BYTE BINARY PROGRAMMING
0A36' D3 07 C OUT (TIM_CNL),A
0A38' C9 C RET
C
C PAGE

```

```

C
C
C      ; *****
C      ; *
C      ; *          SERIAL PORT
C      ; *
C      ; *****
C
0A39'  C      SIOINIT:
C      ;INITIALIZE SERIAL PORT
C      ;ENTRY
C      ;C      =      CONTROL PORT (SIO_CNA OR SIO_CNB)
C
C      ;EXIT
C      ;NONE
C
0A39'  F3      C      DI          ;DISABLE INTERRUPTS UNTIL EVERYTHING IS SET UP
C      ;INITIALIZE WRITE REG 4 -- BASIC CONTROL
0A3A'  3E 04      C      LD      A,REG4          ;SELECT WRITE REG 4 FIRST
0A3C'  ED 79      C      OUT     (C),A
0A3E'  3E 44      C      LD      A,BAS_CNL        ;BASIC CONTROL -- X16 CLOCK, 1 STOP BIT, NO PARITY
0A40'  ED 79      C      OUT     (C),A
C
C      ;INITIALIZE WRITE REG 3 -- RECEIVE CONTROL
0A42'  3E 03      C      LD      A,REG3          ;SELECT REG 3
0A44'  ED 79      C      OUT     (C),A
0A46'  3E E1      C      LD      A,RCV_CNL        ;RECEIVE CONTROL -- 8 BITS/CHAR, AUTO ENABLES DN, RX ENABLE
0A48'  ED 79      C      OUT     (C),A
C
C      ;INITIALIZE WRITE REG 5 -- TRANSMIT CONTROL
0A4A'  3E 05      C      LD      A,REG5          ;SELECT REG 5
0A4C'  ED 79      C      OUT     (C),A
C
C      ;
C      ; USE MASK CORRESPONDING TO CHANNEL: A HAS RTS LOW, B HAS RTS HIGH
C      ;
C
0A4E'  3E 0D      C      LD      A,SIO_CNA        ;ARE WE SETTING CHANNEL A?
0A50'  B9          C      CP      C              ;
C
0A51'  3E E8      C      LD      A,TX_CNL_A        ;LET'S ASSUME WE ARE.
C
0A53'  CA 0A58'   C      JP      Z,XJ123          ;AH HA! WE ARE...
C      ;
0A56'  3E EA      C      LD      A,TX_CNL_B        ;NO, NOT A, B.
C
0A58'  ED 79      C      XJ123: OUT     (C),A        ;OUTPUT CONTROL BYTE.
C
C      ;INITIALIZE WRITE REG 2 -- INTERRUPT VECTOR
0A5A'  3E 02      C      LD      A,REG2          ;
0A5C'  ED 79      C      OUT     (C),A          ;SELECT WR2
0A5E'  3E E0      C      LD      A,LOW_SIO_I_TBL  ;SET VECTOR TO LOW BYTE OF SIO INTERRUPT VECTOR TABLE
0A60'  ED 79      C      OUT     (C),A          ;

```

```

C
C ;INITIALIZE WRITE REG 1 -- INTERRUPT CONTROL
0A62' 3E 01 C LD A,REG1 ;SELECT REG 1
0A64' ED 79 C OUT (C),A
0A66' 3E 17 C LD A,SIO_I_IN ;INTERRUPT CONTROL
0A68' ED 79 C OUT (C),A
C
0A6A' FB C EI ;REENABLE INTERRUPTS
C
C ;SET FLAG
0A6B' 3E 0D C LD A,SIO_CNA ;CHECK IF PORT A
0A6D' B9 C CP C ;SET Z FLAG IF PORT A
0A6E' 3E FF C LD A,0FFH
0A70' 20 04 C JR NZ,XJ124
C
0A72' 32 23D7 C LD (S_A_FL6),A ;IF PORT A
0A75' C9 C RET
C
0A76' 32 23D8 C XJ124: LD (S_B_FL6),A ;IF PORT B
0A79' C9 C RET
C
C PAGE

```

```

C
C
0A7A' C SETBAUD:
C ;SET BAUD RATE FOR SERIAL PORT
C ;ENTRY
C ;C = BAUD RATE CODE (0-15)
C ;A = PORT (0 = PORT A, 1 = PORT B)
C
C ;EXIT
C ;NONE
C
0A7A' E5 C PUSH HL ;SAVE
0A7B' C5 C PUSH BC
0A7C' F5 C PUSH AF
C
C ; CHECK FOR VALID CODE ENTRY
C
0A7D' 79 C XJ125: LD A,C
0A7E' FE 10 C CP BTBLEN ;MAXIMUM CODE VALUE
0A80' 38 03 C JR C,XJ126 ;IF CODE IS VALID, LOOK UP VALUE
C
0A82' F1 C POP AF ;RESTORE STACK
0A83' 18 14 C JR XJ127 ;IF CODE IS INVALID, QUIT
C
C ; FIND TABLE ENTRY FOR THE BAUD RATE CODE
C
0A85' 21 0A9C' C XJ126: LD HL,BAUDTBL ;HL = BAUD RATE TABLE
0A88' C8 21 C SLA C ;C = TABLE OFFSET IN BYTES
0A8A' 06 00 C LD B,0 ;BC = TABLE OFFSET IN BYTES
0A8C' 09 C ADD HL,BC ;HL = TABLE ENTRY'S ADDRESS
C
C ; SET 8253 TIMER TO THE VALUE IN THE TABLE ENTRY
C
0A8D' F1 C POP AF ;DETERMINE WHICH COUNTER TO
0A8E' B7 C OR A ;PROGRAM
0A8F' 0E 04 C LD C,TIM_CT0 ;ASSUME COUNTER 1
0A91' 28 02 C JR Z,XJ128 ;IT IS COUNTER 1 (CHANNEL A)
C
C ;
C ;IT'S NOT COUNTER 1,
0A93' 0E 05 C LD C,TIM_CT1 ;IT'S COUNTER 2 (CHANNEL B)
C
0A95' 06 02 C XJ128: LD B,2 ;OUTPUT THE TWO BYTE STRING IN BAUDTBL
0A97' ED B3 C OTIR ;POINTED TO BY HL
C
0A99' C1 C XJ127: POP BC ;RESTORE
0A9A' E1 C POP HL
0A9B' C9 C RET
C
C PAGE

```

```

C
C
0A9C' C BAUDTBL:
C ;TABLE OF COUNT VALUES WHICH MUST BE INPUT TO THE 8253 COUNTER TO GIVE THE SELECTED BAUD RATE
C ; 23.9619/13 X 10**6= 1.8432MHZ CLK FOR 8253
C ; TIME CONSTANT = CLK / (16 * BAUD RATE )
C ;EX: 12 = 1,843,200 / (16 * 9600)
C
0A9C' 0003 C DW 03 ;38.4
0A9E' 0900 C DW 2304 ;50 BAUD
0AA0' 0600 C DW 1536 ;75 BAUD
0AA2' 0417 C DW 1047 ;110 BAUD
0AA4' 0359 C DW 857 ;134.5 BAUD
0AA6' 0300 C DW 768 ;150 BAUD
0AA8' 0180 C DW 384 ;300 BAUD
0AAA' 00C0 C DW 192 ;600 BAUD
0AAC' 0060 C DW 96 ;1200 BAUD
0AAE' 0040 C DW 64 ;1800 BAUD
0AB0' 0030 C DW 48 ;2400 BAUD
0AB2' 0020 C DW 32 ;3600 BAUD
0AB4' 0018 C DW 24 ;4800 BAUD
0AB6' 0010 C DW 16 ;7200 BAUD
0AB8' 000C C DW 12 ;9600 BAUD
0ABA' 0006 C DW 6 ;19,200 BAUD
C
0010 C BTBLEN EQU ($ - BAUDTBL) / 2
C
C PAGE
C

```

```

C
C
0ABC' C SERAIN:
C ;INPUT ONE BYTE FROM SERIAL PORT A ( CP/M PTR: PORT)
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;C = CHARACTER READ
C
C
C
C ;WAIT FOR CHARACTER
0ABC' CD 0B04' C CALL SAI_STAT
0ABF' 2B FB C JR Z,SERAIN ;IF NO CHARACTER READY
C
C ;GET CHARACTER
0AC1' 06 01 C LD B,S_CH_A_I ;SID CHANNEL A INPUT
0AC3' F3 C DI
0AC4' CD 1226' C CALL RM_C_BF ;REMOVE CHARACTER FROM BUFFER
0AC7' FB C EI
0AC8' 4F C LD C,A ;C = CHARACTER
0AC9' C9 C RET
C
C PAGE

```



```

C
C
0ACA' C SERBIN:
C ;INPUT ONE BYTE FROM SERIAL PORT B ( CP/M CRT: PORT)
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;C = CHARACTER READ
C
C
C ;WAIT FOR CHARACTER
0ACA' CD 0B09' C CALL SBI_STAT ;GET STATUS
0ACD' 2B FB C JR Z,SERBIN ;IF NO CHARACTER READY
C
C ;GET CHARACTER
0ACF' 06 02 C LD B,S_CH_B_I ;SIO CHANNEL B INPUT
0AD1' F3 C DI
0AD2' CD 1226' C CALL RM_C_BF ;REMOVE CHARACTER FROM BUFFER
0AD5' FB C EI
0AD6' 4F C LD C,A ;C = CHARACTER
0AD7' C9 C RET
C
C PAGE

```

```

C
C
0AD8' C SERAOUT:
C ;OUTPUT ONE BYTE TO SERIAL PORT A (CP/M PTP: PORT)
C ;ENTRY
C ;C = CHARACTER TO OUTPUT
C
C ;EXIT
C ;NONE
C
C
0AD8' C5 C PUSH BC ;SAVE CHAR
C
C ;WAIT TILL READY
0AD9' F3 C XJ129: DI ;DISABLE INTERRUPT SO STATUS
C ;IS VALID
0ADA' CD 0AFA' C CALL SBO_STAT ;GET SERIAL PORT A OUTPUT STATUS
0ADD' 20 03 C JR NZ,XJ130 ;READY...
C ;
0ADF' FB C EI ;NOT READY. RE-ENABLE AND
0AE0' 18 F7 C JR XJ129 ;TO BECOME FREE AND LOOP
C
C XJ130: ;SEND CHARACTER
0AE2' C1 C POP BC
0AE3' 06 43 C LD B,S_CH_A_0 + 40H ;SID CHANNEL A OUTPUT
0AE5' CD 11C5' C SERA01: CALL FL_B_CH ;PUT CHARACTER IN BUFFER
0AE8' FB C EI ;RE-ENABLE INTERRUPTS
0AE9' C9 C RET
C
C
C
0AEA' C SERBOUT:
C ;OUTPUT ONE BYTE TO SERIAL PORT B (CP/M CRT: PORT)
C ;ENTRY
C ;C = CHARACTER TO OUTPUT
C
C ;EXIT
C ;NONE
C
C
0AEA' C5 C PUSH BC ;SAVE CHAR
C
C ;WAIT TILL READY
0AEB' F3 C XJ131: DI ;DISABLE INTERRUPT SO STATUS
C ;IS VALID
0AEC' CD 0AFF' C CALL SBO_STAT ;GET SERIAL PORT A OUTPUT STATUS
0AEF' 20 03 C JR NZ,XJ132 ;READY...
C
C EI ;NOT READY. RE-ENABLE
0AF1' FB C JR XJ131 ;TO BECOME FREE AND LOOP
0AF2' 18 F7 C
C
C XJ132: ;SEND CHARACTER
0AF4' C1 C POP BC
0AF5' 06 44 C LD B,S_CH_B_0 + 40H ;SID CHANNEL A OUTPUT
0AF7' C3 0AE5' C JP SERA01 ;PUT CHARACTER IN BUFFER

```

C

C

PAGE

```

C
C
0AFA' C      SAO_STAT:
C      ;OUTPUT STATUS FOR SERIAL PORT A
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;A      =      0, IF NOT READY
C      ;A      =      0FFH IF READY
C      ;ZBIT   =      SET IF NOT READY FOR OUTPUT
C
C
0AFA' 06 03 C      LD      B,S_CH_A_D      ;SIO CHANNEL A OUTPUT
0AFC' C3 0B0E' C      SAO_S1: JP      GET_O_STAT      ;GET STATUS
C
C
C
C
0AFF' C      SBO_STAT:
C      ;OUTPUT STATUS FOR SERIAL PORT B
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;A      =      0, IF NOT READY
C      ;A      =      0FFH IF READY
C      ;ZBIT   =      SET IF NOT READY FOR OUTPUT
C
C
0AFF' 06 04 C      LD      B,S_CH_B_D      ;SIO CHANNEL B OUTPUT
0B01' C3 0B0E' C      JP      GET_O_STAT      ;GET STATUS
C
C      PAGE

```

```

C
C
0B04' C SAI_STAT:
C ;SERIAL PORT A INPUT STATUS
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;A = 0 IF NOT READY, 0FFH IF READY
C ;Z BIT = SET IF NOT READY
C
C
0B04' 06 01 C LD B,S_CH_A_I ;SID CHANNEL A INPUT
0B06' C3 0B18' C JP GET_I_STAT ;GET STATUS
C
C
0B09' C SBI_STAT:
C ;SERIAL PORT B INPUT STATUS
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;A = 0 IF NOT READY, 0FFH IF READY
C ;Z BIT = SET IF NOT READY
C
C
0B09' 06 02 C LD B,S_CH_B_I ;SID CHANNEL B INPUT
0B0B' C3 0B18' C JP GET_I_STAT ;GET STATUS
C
C PAGE

```

```

C
C
0B0E' C GET_O_STAT:
C ;SERIAL PORT OUTPUT STATUS
C ;ENTRY
C ;B = DEVICE NUMBER
C
C ;EXIT
C ;A = 0 IF NOT READY, 0FFH IF READY
C ;ZBIT = SET IF NOT READY
C
0B0E' CD 11B6' C CALL FL_B_ST ;GET STATUS
0B11' 2B 03 C GET_O_S1: JR Z,XJ133 ;IF NOT READY
C
C DR 0FFH ;IF READY
0B13' F6 FF C RET ;RETURN A = 0FFH
0B15' C9 C
C
0B16' AF C XJ133: XOR A ;IF NOT READY
0B17' C9 C RET ;RETURN A = 0
C
C
0B18' C GET_I_STAT:
C ;SERIAL PORT INPUT STATUS
C ;ENTRY
C ;B = DEVICE NUMBER
C
C ;EXIT
C ;A = 0 IF NOT READY, 0FFH IF READY
C ;ZBIT = SET IF NOT READY
C
0B18' CD 1273' C CALL RM_C_ST ;GET STATUS
0B1B' C3 0B11' C JP GET_O_S1 ;IF NOT READY
C
C PAGE

```

0B1E'

```
C
C
C      SETDMA:
C      ;SET DMA ADDRESS AND BANK
C      ;ENTRY
C      ;HL      =      ADDRESS
C      ;A       =      BANK
C
C      ;EXIT
C      ;DMADR SET
C      ;DMABANK SET
C
C      LD      (DMADR),HL
C      LD      (DMABANK),A
C      RET
C      PAGE
```

0B1E' 22 FD4B

0B21' 32 FD4A

0B24' C9

'EXECUTIVE ROM MONITOR'
'I/O DRIVERS'

MACRO-80 3.44 09-Dec-81

PAGE 1-246

i

SUBTTL
PAGE

'DISK DRIVERS'


```

C      SUBTTL      'DISK DRIVERS'
C      INCLUDE ROM5                                ;DISK ROUTINES
C      ;
C      ;
C      ;*****
C      ;*                                     *
C      ;*          DISK DRIVERS              *
C      ;*                                     *
C      ;*****
C      ;
C      ;MODULE 5 --ROM5-- ENTRY POINTS
C      ;
C      ;          DISK ROUTINES
C      ;
C      ;RDRV:      RESET DRIVE
C      ;RSEC:      READ SECTOR
C      ;WSEC:      WRITE SECTOR
C      ;R_WSEC:    PERFORM READ OR WRITE OF SECTOR
C      ;SENDEN:    DETERMINE DENSITY AND NUMBER OF SECTORS/TRACK
C      ;HOME:      HOME DRIVE (TRACK 0)
C      ;SEEK:      SEEK TO GIVEN TRACK
C      ;STEP:      STEP ONE TRACK
C      ;STEPIN:    STEP IN ONE TRACK (TRACK = TRACK + 1)
C      ;STEPOUT:   STEP OUT ONE TRACK (TRACK = TRACK - 1)
C      ;PSECC:     PERFORM SEEK-TYPE COMMAND
C      ;READ:      READ GIVEN NUMBER OF SECTORS
C      ;WRITE:     WRITE GIVEN NUMBER OF SECTORS
C      ;RD_WRT:    PERFORM READ OR WRITE OF SECTOR(S)
C      ;RADR:      READ ADDRESS INFORMATION
C      ;ROMREAD:   FAST DMA READ (USED BY RADR)
C      ;FMTTRK:    FORMAT ONE TRACK
C      ;FORINT:    INTERRUPT DISK CONTROLLER
C      ;FDSK:      FUNCTION DISK
C      ;SELDV:     SELECT DRIVE
C      ;WBUSY:     WAIT FOR BUSY TO CLEAR
C      ;FORMAT:    FORMAT TRACK IN IBM 3740 FORMAT
C      ;DELAY:     DELAY 'N' MILLISECONDS
C
C      PAGE
```

0B25'

```
C
C
C      .Z80
C      .LIST
C      ; SYSS1.MAC , RAMSMON.MAC AND RAMS10.MAC INCLUDED
C
C      ; EXT      DMARD,DMAWRT,BANKMOVE
C
C      ; PUBLIC  RDRV,RSEC,WSEC,SENDEN,HOME,SEEK,STEP,STEPIN,STEPOUT,READ,WRITE
C      ; PUBLIC  RADR,FMTRK,FORINT,SELDRV,FORMAT,D.FINT
C
C      CSEG
C
C      PAGE
```

	C				
	C				
	C	:			
	C	:		+-----+-----+	
	C	:		DISK ROUTINES	
	C	:		+-----+-----+	
	C				
0B25'	C	RDRV:			
	C	;RESET DRIVE			
	C	;THIS ROUTINE WILL HOME THE DRIVE, IF THERE IS AN ERROR, A RECALIBRATION IS PERFORMED			
	C	:			
	C	;ENTRY			
	C	;NONE			
	C				
	C	;EXIT			
	C	;ZBIT = RESET IF ERROR			
	C				
	C				
	C				
0B25' 3A 23D1	C		LD	A, (NRETRY)	
0B28' 32 23C8	C		LD	(RTRY), A	
	C				
0B2B' CD 0DBB'	C	XJ134:	CALL	SELDRV	;SELECT DRIVE
0B2E' CD 0C32'	C		CALL	STEPIN	;MAKE SURE HEAD IS NOT PAST TRACK 0
0B31' CD 0BF8'	C		CALL	HOMI	;HOME DRIVE
0B34' 30 0C	C		JR	NC, XJ135	;IF GOOD
	C				
0B36' 3A 23C8	C	XJ136:	LD	A, (RTRY)	
0B39' 3D	C		DEC	A	
0B3A' 32 23C8	C		LD	(RTRY), A	
0B3D' 20 EC	C		JR	NZ, XJ134	& TRY AGAIN
	C				
0B3F' 3C	C		INC	A	;MAKE NOT ZERO
0B40' 37	C		SCF		;INDICATE ERROR
0B41' C9	C		RET		
	C				
0B42' AF	C	XJ135:	XOR	A	
0B43' C9	C		RET		
	C		PAGE		

```

C
C
0B44' C      RSEC:
C      ;READ SECTOR
C      ;#NOTE#
C      ;      NO RETRIES ARE PERFORMED AT THIS LEVEL
C      ;ENTRY
C      ;B      =      NUMBER OF SECTORS
C
C      ;EXIT
C      ;HL      =      LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
C      ;ZBIT      =      RESET IF ERROR
C      ;A      =      0 IF NO ERROR, 1 IF HARD ERROR
C      ;B      =      FDC STATUS IF ERROR
C
C
0B44' 3E 88 C      LD      A,D.RDS
0B46' 32 23D0 C      LD      (R_WCOM),A
C
0B49' 18 05 C      JR      R_WSEC
C      PAGE

```

0B4B'

```
C
C
C      WSEC:
C      ;WRITE A SECTOR
C      ;*NOTE*
C      ;      NO RETRIES ARE PERFORMED AT THIS LEVEL
C      ;ENTRY
C      ;B      =      NUMBER OF SECTORS
C
C      ;EXIT
C      ;HL      =      LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
C      ;ZBIT    =      RESET IF ERROR
C      ;A      =      0 IF NO ERROR, 1 IF HARD ERROR, 2 IF WRITE PROTECTED
C      ;B      =      FDC STATUS IF ERROR
C
C
C      0B4B'  3E AB      LD      A,D.WRTS
C      0B4D'  32 23D0    LD      (R_WCOM),A
C
C      ;FALLS THROUGH TO "R_WSEC"
C
C      PAGE
```

```

C
C
0B50' C R_WSEC:
C ;READ OR WRITE SEGMENT
C ;ENTRY
C ;B = NUMBER OF SECTORS TO READ OR WRITE
C ;R_WCOM = D.RDS OR D.WRTS
C ;RDFLAG = 0 FOR READ, NON-ZERO FOR WRITE (FOR HYSTERESIS)
C
C ;EXIT
C ;HL = LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
C ;ZBIT = RESET IF ERROR
C ;A = 0 IF GOOD, 1 IF HARD ERROR, 2 IF WRITE PROTECTED
C ;B = FDC STATUS IF ERROR
C
C
0B50' 78 C LD A,B ;PUT SECTOR COUNT IN A
0B51' 32 23CF C LD (NUMSEC),A ;AND SAVE IT
C
0B54' 3A 23D1 C LD A,(NRETRY)
0B57' 32 23C8 C LD (RTRY),A ;SET RETRY NUMBER
0B5A' CB 3F C SRL A ;A=A/2
0B5C' 32 23F1 C LD (RDFLAG),A ;SAVE VALUE FOR RECALIBRATION
C
C ;SELECT DRIVE
C
0B5F' CD 0DBB' C CALL SELDRV ;TURN DRIVE ON
0B62' 28 06 C JR Z,XJ13B ;IF DRIVE ON DON'T READ ADDRESS
C
C ;SET "DSK_TRK" TO HEAD POSITION
C
0B64' CD 0CE9' C XJ139: CALL RADR ;READ ADDRESS AND SET CONTROLLER
0B67' 47 C LD B,A ;SAVE FDC STATUS
0B68' 38 13 C JR C,XJ140 ;ERROR ?- YES, RETRY
C
C ;SEEK
C
0B6A' CD 0C20' C XJ138: CALL SEEK ;SEEK TO TRACK
0B6D' 47 C LD B,A ;SAVE FDC STATUS
0B6E' 38 0D C JR C,XJ140 ;ERROR ?- YES, RETRY
C
C ;READ OR WRITE
C
0B70' 3A 23CF C LD A,(NUMSEC) ;RESTORE # SECTORS TO R/W
0B73' 47 C LD B,A ;TO B
0B74' CD 0C6E' C CALL RD_WRT ;READ/WRITE PER R_WCOM
0B77' 38 1C C JR NC,XJ141 ;IF GOOD, RETURN
C
0B79' FE 02 C CP 02H ;IS WRITE PROTECTED ?
0B7B' 28 18 C JR Z,XJ141 ;YES-
C
0B7D' 21 23C8 C XJ140: LD HL,RTRY ;GET RETRY COUNT
0B80' 3A 23F1 C LD A,(RDFLAG) ;GET RECALIBRATION ERROR NUMBER
0B83' BE C CP (HL)
0B84' 20 09 C JR NZ,XJ142 ;NOT TIME TO RECALIBRATE

```

		C			
0B86'	7E	C		LD A, (HL)	;SAVE RETRY COUNT
0B87'	F5	C		PUSH AF	;&
0B88'	E5	C		PUSH HL	;RTRY ADDRESS
0B89'	CD 0B25'	C		CALL RDRV	;RECALIBRATE TO TRACK 0
0B8C'	E1	C		POP HL	;RESTORE VALUES
0B8D'	F1	C		POP AF	
0B8E'	77	C		LD (HL), A	
		C			
0B8F'	35	C	XJ142:	DEC (HL)	
0B90'	C2 0B64'	C		JP NZ, XJ139	;YES-RETRY
		C			
0B93'	3E 01	C	XJ143:	LD A, 1	;INDICATE HARD ERROR
0B95'	B7	C	XJ141:	OR A	
0B96'	C9	C		RET	
		C		PAGE	

```

0097'
SENDEN:
; DETERMINE THE DENSITY, # AND SIZE OF SECTORS PER TRACK & # OF SIDES ON THIS DISKETTE
;
; ENTRY NONE

; EXIT
; B = NUMBER OF SECTORS ON ONE TRACK
; ZBIT = RESET IF ERROR
; SAVTYP IS SET WITH DENSITY, SECTOR SIZE & # OF SIDES
;
; SAVTYPE
;
;
;
; Unchanged (undefined bits)
; Bytes/sector
; 0 = 128, 1 = 256
; 2 = 512, 3 = 1024
; Side ( 0 = SS, 1 = DS )
; Density ( 0 = DD, 1 = SD )

; Test side 0 first
LD HL, SAVTYP
LD A, (HL)
AND 11110001b ; Reset side bit & clear sector length
LD (HL), A

LD A, (NRETRY) ; Get retry count into C-reg
LD C, A

; DENSITY LOOP (CHECK PRESENT DENSITY FIRST)
XJ144: LD B, 2 ; CHECK BOTH DENSITIES
XJ145: PUSH BC ; Save retry counts
PUSH HL ; Save SAVTYP address

; CHECK THIS DENSITY
CALL SELDRV ; SELECT DRIVE
CALL RADR ; READ ADDRESS
POP HL
POP BC ; Restore retry counts
JR NC, XJ146 ; IF GOOD

; IF DENSITY ERROR CHANGE DENSITY AND LOOP TO :RL2:
LD A, (HL) ; PRESENT DENSITY

```



```

0BB1' EE 01      C      XOR    1      ;CHANGE DENSITY BIT
0BB3' 77         C      LD      (HL),A  ;NEW DENSITY
                                C
0BB4' 10 EE      C      DJNZ   XJ145   ;DENSITY LOOP
                                C
                                C      ;IF BOTH DENSITYS FAIL HOME DRIVE AND TRY AGAIN
                                C
0BB6' 0D         C      DEC     C
0BB7' 28 3B      C      JR      Z,XJ147 ;END IF SECOND TIME THROUGH
                                C
0BB9' C5         C      PUSH   BC
0BBA' E5         C      PUSH   HL
0BBB' CD 0BF8'   C      CALL   HOME    ;HOME DRIVE
0BBE' E1         C      POP     HL
0BBF' C1         C      POP     BC
                                C
0BC0' 18 E0      C      JR      XJ144   ;TRY AGAIN
                                C
                                C      ;SET "SAVTYP"
                                C
0BC2' 3A 23CC    C      XJ146: LD     A,(DSTSB+3) ;SECTOR LENTGH STATUS BYTE
0BC5' E6 03      C      AND     0000011B ;0-3
0BC7' C9 27      C      SLA     A
0BC9' CB 27      C      SLA     A      ;NOW IS 0000_XX00B WAS 0000_00XXB
                                C
0BCB' B6         C      OR      (HL)    ;Or sector length with SAVTYP
0BCC' 77         C      LD      (HL),A  ;And save it
                                C
                                C      ;READ ADDRESS AND SET NUMBER OF SECTORS AND RETURN
                                C
0BCD' 3A 23CB    C      LD      A,(DSTSB+2) ;SECTOR JUST READ
0BD0' 57         C      LD      D,A
                                C
                                C      ;RETRY LOOP
                                C
0BD1' 3A 23D1    C      LD      A,(NRETRY) ;SET RETRYS
0BD4' 5F         C      LD      E,A
                                C
                                C      ;READ HEADER LOOP
                                C
0BD5' D5         C      XJ149: PUSH   DE      ;SAVE LAST SECTOR ADDR & retry count
0BD6' E5         C      PUSH   HL
0BD7' CD 0CE9'   C      CALL   RADR    ;READ ADDRESS
0BDA' E1         C      POP     HL
0BDB' D1         C      POP     DE
                                C
0BDC' 38 11      C      JR      C,XJ150   ;IF ERROR IN RADR
                                C
                                C      ;CHECK FOR LAST HEADER
                                C
0BDE' 3A 23CB    C      LD      A,(DSTSB+2)
0BE1' 47         C      LD      B,A      ;B=PRESENT SECTOR
0BE2' 7A         C      LD      A,D      ;A=LAST SECTOR
0BE3' 90         C      SUB     B      ;LAST SECTOR - PRESENT SECTOR
0BE4' 28 EF      C      JR      Z,XJ149   ;IF THE LAST SECTOR = THE PRESENT SECTOR
                                C      ;(THIS SHOULD HAPPEN ONLY ON ERROR RETRY)

```

```

C
0BE6' 30 03 C JR NC,XJ151 ;IF THE VALUE IN A WAS THE LAST SECTOR OF THE TRACK
C
0BE8' 50 C LD D,B ;D=LAST SECTOR READ
0BE9' 18 EA C JR XJ149 ;LOOP TILL YOU FIND THE LAST TRACK
C
;SET NUMBER OF SECTORS PER TRACK
C
0BEB' 3C C XJ151: INC A ;A=NUMBER OF SECTORS PER TRACK
0BEC' 47 C LD B,A
C
;REMOVE THESE TWO STATEMENTS TO IMPLEMENT DOUBLE-SIDED
C xor a
C ret ;clear z-flag and ret
C ;and remove the ";" before each of the following executable statements
C
C ;Determine # of sides
C
C ; PUSH BC
C
C ;Try side 1
C
C ; SET 1,(HL) ;Select side 1 in SAVTYP
C
C ;Now try to read the sector header
C
C ; CALL RADR
C ; JR NC,CONT ;Continue if no error
C
C ; CALL RADR ;Try again if error
C ; JR C,SSDD ;Error again, must be single sided diskette
C
C ;CONT: LD A,(DSTSB+1) ;Get side from header
C ; OR A ;A=0 on single sided drive, 1 on double sided drive
C ; JR NZ,DSDD ;If A=1 must be double sided
C
C ;SSDD: LD HL,SAVTYP
C ; RES 1,(HL) ;Clear side bit
C
C ;DSDD: POP BC ;Restore # of sectors
C ; XOR A ;RESET CFLAG
C ; JR XJ152 ;GOOD RETURN
C
C ;IF NUMB. SEC. ERROR MAKE LAST SECTOR READ ZERO AND RETRY TO :LOOP1
C
0BEF' 16 00 C XJ150: LD D,0
0BF1' 1D C DEC E
0BF2' 20 E1 C JR NZ,XJ149 ;RETRY
C
0BF4' 3E 01 C XJ147: LD A,1
0BF6' B7 C OR A ;CFLAG TO NONZERO
C
0BF7' C9 C XJ152: RET
C PAGE

```

```

C
C
0BF8' C HOME:
C ;HOME DISK DRIVE
C ;DRIVE IS ALREADY SELECTED AND READY
C ;IF "SEKDEL" HAS THE VERIFY BIT SET THIS PROC WILL CHECK FOR SEEK AND CRC ERRORS
C ;ENTRY
C ;SDISK = DRIVE
C
C ;EXIT
C ;CBIT = SET IF ERROR
C ;A = FDC STATUS IF ERROR
C
C
0BF8' 3A 23F0 C LD A,(SEKDEL) ;GET SEEK DELAY
0BF8' E6 07 C AND 00000111B ;SPEAD & VERIFY BITS ONLY
C
0BFD' F3 C DI
0BFE' CD 0D90' C CALL FDSK ;FUNCTION DISK
0C01' 38 1B C JR C,XJ153 ;IF ERROR
C
0C03' CD 0E07' C CALL WBUSY ;WAIT FOR BUSY TO DROP
0C05' 38 16 C JR C,XJ153
C
0C08' DB 08 C IN A,(DSK_STS)
0C0A' CB 57 C BIT 2,A
0C0C' 28 0F C JR Z,XJ154 ;IF NOT ON TRACK ZERO
C
0C0E' 3A 23F0 C LD A,(SEKDEL)
0C11' E6 04 C AND 00000100B ;VERIFY?
0C13' 28 09 C JR Z,XJ153 ;NO VERIFY GOOD RETURN
C
0C15' DB 08 C IN A,(DSK_STS)
0C17' 47 C LD B,A ;SAVE FDC STATUS
0C18' E6 18 C AND 00011000B ;TEST SEEK AND CRC
0C1A' 78 C LD A,B ;RESTORE FDC STATUS
0C1B' 28 01 C JR Z,XJ153 ;GOOD RETURN
C
0C1D' 37 C XJ154: SCF ;IF ERROR
0C1E' FB C XJ153: EI
0C1F' C9 C RET
C PAGE

```

```

C
C
0C20' C      SEEK:
C      ;SEEK TO TRACK DEFINED BY SAVTRK
C      ;TRACK REG UPDATED AND VERIFIED
C      ;ENTRY
C      ;SAVTRK SET TO DESIRED TRACK
C
C      ;EXIT
C      ;CBIT =      SET IF ERROR
C      ;A   =      FDC STATUS IF ERROR
C      ;      IF NO ERROR CONTROLLER TRACK = SAVTRK
C
C
0C20' 21 FD4E C      LD      HL, SAVTRK
0C23' DB 09   C      IN      A, (DSK_TRK)
0C25' BE      C      CP      (HL)
0C26' C8      C      RET     Z      ;RETURN
C
0C27' 7E      C      LD      A, (HL)
0C28' D3 0B   C      OUT     (DSK_DAT), A      ;SET TRACK WANTED
C
0C2A' 06 10   C      LD      B, D. SEK
0C2C' 18 0C   C      JR      PSEKC      ;PERFORM SEEK COMMAND
C      PAGE

```

```

C
C
0C2E' C STEP:
C ;STEP ONE TRACK
C ;SAVTRK IS NOT USED IN THIS PROC
C ;CONTROLLER TRK REG IS UPDATED
C ;VERIFY IS PERFORMED
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;CBIT = SET IF ERROR
C ;A = FDC STATUS IF ERROR
C ; IF NO ERROR CONTROLLER TRACK = TRACK +/- 1
C
C
0C2E' 06 20 C LD B,D.STP
0C30' 18 08 C JR PSEKC ;PERFORM STEP COMMAND
C PAGE
```

```
C
C
0C32' C STEPIN:
C ;STEP IN ONE TRACK
C ;SAVTRK IS NOT USED IN THIS PROC
C ;CONTROLLER TRK REG IS UPDATED
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;CBIT = SET IF ERROR
C ;A = FDC STATUS IF ERROR
C ; IF NO ERROR CONTROLLER TRACK = TRACK + 1
C
0C32' 06 40 C LD B,D.STPI
0C34' 18 04 C JR PSEKC ;PERFORM STEP-IN COMMAND
C PAGE
```

```

C
C
0C36' C STEPOUT:
C      ;STEP OUT ONE TRACK
C      ;SAVTRK IS NOT USED IN THIS PROC
C      ;CONTROLLER TRK REG IS UPDATED
C      ;VERIFY IS PERFORMED
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;CBIT = SET IF ERROR
C      ;A = FDC STATUS IF ERROR
C      ; IF NO ERROR CONTROLLER TRACK = TRACK - 1
C
0C36' 06 60 C LD B,D.STPO
0C38' 18 00 C JR PSEKC ;PERFORM STEP-OUT COMMAND
C PAGE
```

```

C
C
0C3A' C PSEKC:
C ;OR IN SEKDEL AND PERFORM SEEK TYPE COMMAND
C ;ENTRY
C ;B = SEEK TYPE COMMAND
C
C ;EXIT
C ;CBIT = SET IF ERROR
C ;A = FDC STATUS IF ERROR
C
C
0C3A' 3A 23F0 C LD A, (SEKDEL)
0C3D' E6 17 C AND 00010111B ;UPDATE, VERIFY, & STEP RATE
0C3F' B0 C OR B ;OR IN ACTUAL COMMAND
C
0C40' F3 C DI
0C41' CD 0D90' C CALL FDSK ;FUNCTION DISK
0C44' 3B 1A C JR C, XJ155 ;IF ERROR
C
0C46' CD 0E07' C CALL WBUSY ;WAIT FOR BUSY TO DROP
0C49' 3B 15 C JR C, XJ155 ;RETURN IF TIME OUT ERROR
C
C ;HEAD SETTLE TIME
0C4B' 3E 14 C LD A, 20 ;Head settle time of 20ms
0C4D' CD 0E69' C CALL DELAY
C
C ;CHECK FOR ERRORS
0C50' 3A 23F0 C LD A, (SEKDEL)
0C53' E6 04 C AND 00000100B ;VERIFY?
0C55' 2B 09 C JR Z, XJ155 ;NO VERIFY GOOD RETURN
C
0C57' DB 08 C IN A, (DSK_STS)
0C59' 47 C LD B, A ;SAVE FDC STATUS
0C5A' E6 18 C AND 00011000B ;TEST SEEK AND CRC
0C5C' 2B 02 C JR Z, XJ155 ;GOOD RETURN
C
0C5E' 78 C LD A, B ;RESTORE FDC STATUS
0C5F' 37 C SCF ;IF ERROR
0C60' FB C XJ155: EI
0C61' C9 C RET
C PAGE

```



```

C
C
0C62'      C      READ:
C          ;ENTRY
C          ;B      =      NUMB OF SECTORS TO READ
C
C          ;EXIT
C          ;HL      =      LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
C          ;CBIT    =      SET IF ERROR
C          ;B      =      FDC STATUS IF ERROR
C
C
0C62'      3E 88      C      LD      A,D.RDS
0C64'      32 23D0     C      LD      (R_WCOM),A
C
0C67'      18 05      C      JR      RD_WRT      ;JMP AND RETURN TO CALLING PROC
C                  PAGE

```

```
C
C
C      WRITE:
C      ;ENTRY
C      ;B      =          NUMB OF SECTORS TO WRITE
C
C      ;EXIT
C      ;HL     =          LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
C      ;CBIT   =          SET IF ERROR
C      ;B      =          FDC STATUS IF ERROR
C
C
C
C      LD      A,D.WRTS
C      LD      (R_WCDM),A
C
C      ;FALLS THROUGH TO RD_WRT
C      PAGE
```

```

C
C
C
0C6E'      C      RD_WRT:
C      ;READ OR WRITE A SECTOR
C      ;      THIS ROUTINE ASSUMES THAT IX IS NOT CHANGED BY ANY CALLED
C      ;      ROUTINE (FDSK, ETC).
C
C      ;ENTRY
C      ;B      =      NUMB OF SECTORS TO READ OR WRITE
C      ;R_WCOM =      D.RDS OR D.WRTS
C
C      ;EXIT
C      ;HL      =      LAST DMA ADDRESS PLUS ONE IF GOOD TRANSFER
C      ;CBIT    =      SET IF ERROR
C      ;A      =      0 IF NO ERROR, 1 IF HARD ERROR, 2 IF WRITE PROTECTED
C      ;B      =      FDC STATUS IF ERROR
C
C
C      ;SET SECTOR REG
C
0C6E'      3A FD4D      C      LD      A,(SAVSEC)
0C71'      D3 0A        C      OUT     (DSK_SEC),A
C
C      PUSH     BC      ;SAVE NUMBER OF SECTORS TO R/W
C
C      ;LOCATE ADDRESS OF DMA TRANSFER ROUTINE
C
0C74'      3A 23D0      C      LD      A,(R_WCOM)      ;GET D.RDS OR D.WRTS
0C77'      FE 88        C      CP      D.RDS      ;IS COMMAND IN A A READ?
0C79'      DD 21 FF39    C      LD      IX,DMARD
0C7D'      28 04        C      JR      Z,XJ156      ;IF READ
C
0C7F'      DD 21 FF56    C      LD      IX,DMANRT      ;IF WRITE
C
C      ;SET DE TO NUMBER OF BYTES IN ONE SECTOR
0C83'      21 0080      C      XJ156: LD      HL,128
0C86'      3A FD51      C      LD      A,(SAVTYP)      ;DISK TYPE
0C89'      CB 3F        C      SRL     A      ;DUMP TWO BITS
0C8B'      CB 3F        C      SRL     A
0C8D'      E6 03        C      AND     00000011B      ;SIZE ONLY
0C8F'      B7           C      OR      A      ;SET CFLAG
0C90'      28 04        C      JR      Z,XJ157      ;IF 128
C
C      LD      B,A
0C92'      47           C      XJ158: ADD     HL,HL      ;SHIFT LEFT ONE BIT
0C93'      29           C      DJNZ    XJ158
0C94'      10 FD        C
C
C      XJ157: EX      DE,HL      ;DE=NUMBER OF BYTES IN ONE SECTOR
C
C      POP     BC      ;RESTORE NUMBER OF SECTORS TO R/W
0C97'      C1           C      PUSH    BC      ;SAVE
0C98'      C5           C
C
C      ;GET COMMAND AND CHECK FOR MULTI-SECTOR
0C99'      78           C      LD      A,B      ;GET NUMBER OF SECTORS
0C9A'      0E 00        C      LD      C,0      ;MAKE NONMULTI-SECTOR

```

```

0C9C' FE 02      C                CP      2
0C9E' 38 02      C                JR      C,XJ159      ;IF LESS THAN TWO SECTORS
C
0CA0' 0E 10      C                LD      C,10H        ;MAKE MULTI-SECTOR
C
0CA2' 3A 23D0     C      XJ159:    LD      A,(R_WCOM)    ;GET D.RDS OR D.WRTS
0CA5' B1          C                OR      C            ;MAKE MULTI-SECTOR OR NONMULTI-SECTOR
C
C      ;A = COMMAND FOR FDSK
C
C      ;SET HL TO NUMBER OF BYTES TO TRANSFER
C
0CA6' 21 0000     C                LD      HL,0
0CA9' 19          C      XJ160:    ADD     HL,DE
0CAA' 10 FD       C                DJNZ   XJ160
C
0CAC' E5          C                PUSH   HL            ;SAVE LENTH
C
C      ;GIVE COMMAND
C
0CAD' F3          C                DI
0CAE' CD 0D74'    C                CALL   SSEL          ;Form command with correct side
0CB1' CD 0D90'    C                CALL   FDSK
0CB4' 30 06       C                JR      NC,XJ161      ;IF GOOD
C
C      ;IF ABORT BEFORE DMA
C
0CB6' FB          C                EI
0CB7' 47          C                LD      B,A
0CB8' D1          C                POP     DE            ;RESTORE STACK
0CB9' D1          C                POP     DE
0CBA' 18 29       C                JR      XJ162          ;RETURN IF ERROR IN FDSK
C
C      ;SET RETURN FROM DMA, DMA ADDR AND NUMBER OF BYTES TO TRANSFER
C
0CBC' C1          C      XJ161:    POP     BC            ;RESTORE LENGTH
0CBD' 2A FD4B     C                LD      HL,(DMADR)    ;HL = DMA ADDRESS
C
C                LD      DE,XJ163
0CC0' 11 0CC6'    C                PUSH   DE            ;FOR RETURN
0CC3' D5          C
C
0CC4' DD E9       C                JP      (IX)          ;CALL DMA TRANSFER RTN
C
C      ;RETURN FROM DMA AND CHECK FOR BUSY AND RESET
C
0CC6' C1          C      XJ163:    POP     BC            ;RESTORE NUMBER OF SECTORS
C
C                IN      A,(DSK_STS)    ;GET STATUS
0CC7' DB 08       C                BIT     0,A
0CC9' CB 47       C                JR      Z,XJ164        ;IF NOT BUSY
0CCB' 28 0D       C
C
0CCD' 11 0CD8'    C                LD      DE,XJ165
0CD0' D5          C                PUSH   DE            ;FOR RETURN
C
C                DEC     B              ;SUBTRACT ONE FROM THE NUMBER OF SECTORS AND SET THE ZERO FLAG
0CD1' 05          C
C
0CD2' CA 0E07'    C                JP      Z,WBUSY        ;IF NON MULTI-SECTOR R/W WAIT FOR BUSY TO DROP
C
0CD5' C3 0D7C'    C                JP      FORINT        ;CLEAR BUSY
C

```

```

0CD8' DB 00      C    XJ165:      IN      A, (DSK_STS)    ;RETURN AND GET STATUS
                  C
                  C    ;CHECK FOR ERRORS
0CDA' FB         C    XJ164:      EI
0CDB' 47         C          LD      B,A
0CDC' E6 5C      C          AND     01011100B    ;TEST WRITE PROTECT, RNF, CRC, AND LOST DATA
0CDE' C8         C          RET      Z          ;IF GOOD (CBIT RESET WITH ANI INSTRUCTION)
                  C    ;FDC ERROR
0CDF' E6 40      C          AND     01000000B    ;IS WRITE PROTECTED ?
0CE1' 3E 02      C          LD      A,2
0CE3' 20 02      C          JR      NZ,XJ166    ;YES
0CE5' 3E 01      C    XJ162:      LD      A,1    ;HARD ERROR
0CE7' 37         C    XJ166:      SCF
0CE8' C9         C          RET              ;RETURN
                  C          PAGE

```

```

C
C
0CE9' C RADR:
C ;READ ADDRESS INFO.
C ;READS SIX BYTES INTO "DSTSB"
C
C ;ENTRY
C ;NONE
C
C ;EXIT
C ;A = FDC STATUS IF ERROR
C ;CBIT = SET IF ERROR
C ;DSK_TRK = HEAD POSITION
C ;NOTE*
C ; SETS TRACK REG IN CONTROLLER IF GOOD
C
C
0CE9' 3E C0 C LD A,D,RDA
0CEB' F3 C DI
0CEC' CD 0D74' C CALL SSEL
0CEF' CD 0D90' C CALL FDSK ;FUNCTION DISK
0CF2' 3B 36 C JR C,XJ160
C
C ;WAIT FOR FIRST DRQ OR TIME OUT
C
C ;SET REGISTERS FOR DMA TRANSFER
C
0CF4' 06 05 C LD B,5 ;SIX BYTES TO READ
0CF6' 21 23C9 C LD HL,DSTSB ;FBA FOR DMA
C
C ;WAIT FOR 1/2 OF A TRACK (100MS) OR DRQ
C
0CF9' 11 1C69 C LD DE,7273
0CFC' DB 08 C XJ169: IN A,(DSK_STS) ;(11+2) GET STATUS
0CFE' 1F C RRA ;(4)
0CFF' 1F C RRA ;(4)
0D00' DA 0D0E' C JP C,XJ170 ;(10) GOT DRQ
C
0D03' 1B C DEC DE ;(6)
0D04' 7A C LD A,D ;(4)
0D05' B3 C OR E ;(4)
0D06' C2 0CFC' C JP NZ,XJ169 ;(10)
C
C ;INDICATE TIME OUT ERROR
C
0D09' CD 0D7C' C CALL FORINT ;CLEAR BUSY
0D0C' 1B 1B C JR XJ171 ;INDICATE A TIME OUT ERROR
C
C ;TRANSFER FIRST BYTE AND CALL DMARD FOR LAST FIVE BYTES
C
0D0E' DB 0B C XJ170: IN A,(DSK_DAT) ;(11+2) GET BYTE
0D10' 77 C LD (HL),A ;(7) STORE BYTE
0D11' 23 C INC HL ;(6)
C ;BC = 5
0D12' CD 0D2C' C CALL ROMREAD ;(17) CALL ROM RESIDENT DMA READ ROUTINE (NO BANK SWITCHING)

```

```

C
C      ;RETURN FROM DMARD AND WAIT FOR BUSY TO BE RESET
C
0D15'  CD 0E07'  C      CALL  WBUSY
0D18'  38 10      C      JR     C,XJ168      ;IF TIME OUT ERROR
C
0D1A'  D3 08      C      IN     A,(DSK_STS)  ;GET STATUS
0D1C'  47          C      LD     B,A
C
C      ;CHECK FOR ERRORS
C
0D1D'  E6 1C      C      AND    00011100B    ;TEST RNF, CRC AND DATA LOST
0D1F'  78          C      LD     A,B
0D20'  20 07      C      JR     NZ,XJ171      ;IF ERROR
C
C      ;SET TRACK REGISTER
C
0D22'  DB 0A      C      IN     A,(DSK_SEC)  ;GET TRACK
0D24'  D3 09      C      OUT    (DSK_TRK),A  ;SET TRACK
0D26'  AF          C      XOR     A          ;RESET CARRY
0D27'  18 01      C      JR     XJ168
C
0D29'  37          C      XJ171:    SCF          ;SET CBIT
C
0D2A'  FB          C      XJ168:    EI
0D2B'  C9          C      RET
C      PAGE

```

```

C
C
0D2C' C ROMREAD:
C ;DMA READ ROUTINE USED BY RADR. RADR NEEDS A VERY QUICK
C ;DMA ROUTINE SINCE IT HAS ALREADY READ ONE BYTE AND MUST GET
C ;TO THE NEXT BYTE IN TIME. THE NORMAL DMA ROUTINE SPENDS TOO
C ;MUCH TIME DOING BANK SWITCHING
C ;ENTRY
C ;B = NUMBER OF BYTES TO TRANSFER
C ;HL = FWA OF BUFFER
C
C ;EXIT
C ;HL = NEXT ADDRESS
C
C ;DO DMA
C
0D2C' DB 08 C XJ172: IN A, (DSK_STS) ;(13) GET STATUS
0D2E' 1F C RRA ;(4)
0D2F' D0 C RET NC ;(5) RETURN IF NO BUSY
0D30' 1F C RRA ;(4)
0D31' D2 0D2C' C JP NC, XJ172 ;(10) IF NO DRQ
C
0D34' DB 09 C IN A, (DSK_DAT) ;(13) GET BYTE
0D36' 77 C LD (HL), A ;(7) STORE BYTE
0D37' 23 C INC HL ;(6)
0D38' 10 F2 C DJNZ XJ172 ;(13) ON JUMP
C
0D3A' C9 C RET
C PAGE

```



```

C
C
0D3B' C FMTTRK:
C ;FORMAT ONE TRACK
C ;ENTRY
C ;BC = LENTH
C ;DMADR = FWA OF BUFFER
C ;DMABANK = BUFFER'S BANK
C
C ;EXIT
C ;CBIT = SET IF ERROR
C ;A = FDC STATUS IF ERROR
C
C ;TEST DENSITY AND SET REG D TO 04EH OR 0FFH
C
0D3B' 3A FD51 C LD A, (SAVTYP)
0D3E' 16 4E C LD D, 04EH ;DOUBLE
0D40' 0F C RRC A
0D41' 30 02 C JR NC, XJ173 ;IF DOUBLE
C
0D43' 16 FF C LD D, 0FFH ;SINGLE
C
C ;GIVE COMMAND
C
0D45' 3E F0 C XJ173: LD A, D.WRTT
0D47' D5 C PUSH DE ;FILL BYTE
0D48' C5 C PUSH BC ;LENTH
C
0D49' F3 C DI
0D4A' CD 0D74' C CALL SSEL
0D4D' CD 0D90' C CALL FDSK
0D50' C1 C POP BC ;LENTH
0D51' D1 C POP DE ;FILL BYTE
0D52' 38 1D C JR C, XJ174 ;IF ERROR
C
C ;DO DMA
C
0D54' D5 C PUSH DE ;FILL BYTE
C
0D55' 2A FD4B C LD HL, (DMADR)
0D58' CD FF56 C CALL DMAWRT
C
C ;PAD REST OF TRACK
C
0D5B' C1 C POP BC ;B = FILL BYTE
C
0D5C' DB 0B C XJ175: IN A, (DSK_STS) ;GET STATUS
0D5E' 1F C RRA
0D5F' 30 0B C JR NC, XJ176 ;FINISHED IF NO BUSY
0D61' 1F C RRA
0D62' 30 F8 C JR NC, XJ175 ;IF NO DRQ
C
0D64' 78 C LD A, B
0D65' D3 0B C OUT (DSK_DAT), A ;STORE BYTE
0D67' 18 F3 C JR XJ175

```

'EXECUTIVE ROM MONITOR'
'DISK DRIVERS'

MACRO-00 3.44 09-Dec-81

PAGE 1-272

C

PAGE

```

C
C
C      ;CHECK FOR ERROR
C
0D69'  DB 08      C      XJ176:      IN      A, (DSK_STS)      ;GET STATUS
0D6B'  47          C          LD      B,A          ;SAVE
0D6C'  E6 44      C          AND      01000100B      ;TEST WRITE PROTECT, AND DATA LOST
0D6E'  78          C          LD      A,B
0D6F'  28 01      C          JR      Z,XJ177          ;IF GOOD
C
0D71'  37          C      XJ174:      SCF
C
0D72'  FB          C      XJ177:      EI
0D73'  C9          C          RET
C          PAGE
```

```

C
C
0D74' C SSEL:
C ;SIDE SELECT, selects appropriate side using SAVTYP bit 1
C ;
C ;ENTRY
C ; A = FDC Command
C ; SAVTYP = Side bit set to appropriate value
C ;
C ;EXIT
C ; A = FDC Command with side bit modified
C ;
C
C
0D74' 47 C LD B,A ;Save command
0D75' 3A FD51 C LD A,(SAVTYP)
0D78' E6 02 C AND 00000010B ;Just get side bit
0D7A' B0 C OR B ;And form new command
C
0D7B' C9 C RET
C PAGE

```

```

C
C
0D7C' C FORINT:
C ; INTERRUPT DISK CONTROLLER
C ; IF ANY APPLICATION PROGRAM CALLS THIS ROUTINE, THAT PROGRAM SHOULD
C ; DISABLE AND ENABLE THE INTERRUPT FOR FORINT ROUTINE.
C ; ENTRY
C ; NONE
C
C ; EXIT
C ; BUSY CLEARED.
C ; CBIT = SET IF ERROR
C ; A = FDC STATUS IF ERROR
C
C
0D7C' 3E D0 C LD A,D.FINT
0D7E' D3 0B C OUT (DSK_CMD),A
C
C ; WAIT FOR AT LEAST 24 MICROSECONDS FOR SINGLE DENSITY
C ; 12 MICROSECONDS FOR DOUBLE DENSITY TO READ BUSY STATUS
C ; FOR 1 MHZ CLOCK
C
0D80' 3A FDS1 C LD A,(SAVTYP) ; (13) DISK TYPE
0D83' 0F C RRCA ; (4)
0D84' 06 01 C LD B,1 ; (7)
0D86' 30 02 C JR NC,XJ178 ; (12) OR (7) - DOUBLE DENSITY
0D88' 06 04 C LD B,4 ; (7)
0D8A' 10 FE C XJ178: DJNZ XJ178 ; (13) OR (8)
C
C ; CHECK FOR BUSY DROP
C
0D8C' CD 0E07' C CALL WBUSY ; (17)
C
0D8F' C9 C RET
C PAGE

```

```

C
C
0D90' C FDSK:
C ;FUNCTION DISK ROUTINE
C ;THIS IS THE ONLY ROUTINE THAT WRITES TO THE COMMAND REGISTER OF THE CONTROLLER CHIP
C ;THIS ROUTINE HAS A BUILT IN DELAY OF AT LEAST 28 MICRO SEC. BEFORE READING THE STATUS ON THE CHIP
C ;ENTRY
C ;A = FUNCTION CODE
C
C ;EXIT
C ;A = 0 IF NO ERROR, FDC STATUS IF ERROR
C ;CBIT = SET IF ERROR
C
0D90' 47 C LD B,A ;SAVE FUNCTION CODE IN B
0D91' DB 08 C IN A,(DSK_STS) ;STATUS REGISTER
0D93' 0F C RACA
0D94' 78 C LD A,B
0D95' 30 03 C JR NC,XJ179 ;IF NOT BUSY
C
0D97' CD 0D7C' C CALL FORINT ;RESET BUSY
C
0D9A' D3 08 C XJ179: OUT (DSK_CMD),A ;FUNCTION DRIVE(WRITE COMMAND TO CONTROLLER)
C
C ;WAIT FOR 56 SINGLE AND 28 DOUBLE(MICRO SEC.)
C
0D9C' 3A FD51 C LD A,(SAVTYP) ;(13) DISK TYPE
0D9F' 06 05 C LD B,5 ;(7)
0DA1' 0F C RACA ;(4)
0DA2' 30 02 C JR NC,XJ180 ;(10) IF DOUBLE DENSITY
C
0DA4' 06 0D C LD B,13 ;(7)
C
0DA6' 10 FE C XJ180: DJNZ XJ180 ;(13) WAIT
C
C ;WAIT FOR BUSY TO BE SET
C
0DAB' 06 FF C LD B,0FFH ;(7) 256 LOOPS
C
0DAA' DB 08 C XJ181: IN A,(DSK_STS) ;TEST BUSY BIT
0DAC' CB 47 C BIT 0,A
0DAE' 20 04 C JR NZ,XJ182 ;IF CHIP WENT BUSY
C
0DB0' 10 F8 C DJNZ XJ181 ;IF NOT TIME-OUT
C
0DB2' 37 C SCF ;IF ERROR A=FDC STATUS AND CBIT = SET
0DB3' C9 C RET
C
0DB4' F6 FF C XJ182: OR 0FFH
0DB6' 32 23F2 C LD (DACTIVE),A ;SET DRIVE ACTIVE COUNTER
0DB9' AF C XOR A ;NO ERROR A=0 AND CBIT=RESET
0DBA' C9 C RET
C PAGE

```

```

C
C
0DBB' C SELDRV:
C ;SELECT DRIVE
C ;ENTRY
C ;SDISK = DRIVE TO SELECT
C
C ;EXIT
C ;ZBIT = SET IF DRIVE ALREADY SELECTED
C ;ZBIT = RESET IF DRIVE NOT ALREADY SELECTED
C
0DBB' F3 C DI ;DISABLE INTERRUPTS WHILE USING SYSTEM PIA
C ;(TO PREVENT INTERRUPT FROM TURNING DRIVE OFF)
C
C ;SELECT DENSITY
0DBC' DB 02 C IN A, (SYS_DTB) ;GET CURRENT REG VALUE
0DBE' E6 FE C AND 1111110B ;CLEAR BIT 0
0DC0' 4F C LD C, A
C
C ;SET DENSITY BIT
0DC1' 3A FD51 C LD A, (SAVTYP) ;GET DISK TYPE INFO
0DC4' 0F C RRCA ;CBIT (= BIT 0
0DC5' 30 02 C JR NC, XJ183 ;IF "SAVTYP" BIT0 IS 0
C
0DC7' CB C1 C SET 0, C ;SET BIT 0 OF REG C
C
0DC9' 79 C XJ183: LD A, C
0DCA' D3 02 C OUT (SYS_DTB), A ;FUNCTION PIA
C
C ;SET DRIVE ACTIVE COUNTER
0DCC' 21 23F2 C LD HL, DACTIVE
0DCF' 36 FF C LD (HL), 0FFH ;SET DRIVE ACTIVE COUNTER
C
C ;GET DRIVE TO SELECT
0DD1' 3A FD50 C LD A, (SDISK)
0DD4' 21 23C7 C LD HL, DSKSWP ;DISK DRIVE SWAP CELL
0DD7' AE C XOR (HL) ;SWAP A FOR B IF DSKSWP=1
0DD8' E6 01 C AND 1 ;CAN ONLY BE 0 OR 1
0DDA' FE 01 C CP 1
0DDC' 20 02 C JR Z, XJ184 ;IF DRIVE 1
C
0DDE' 3E 02 C LD A, 2 ;IF DRIVE 0
C
0DE0' CB 27 C XJ184: SLA A ;POSITION BITS (DR 0 = 0000_0100, DR 1 = 0000_0010)
0DE2' 4F C LD C, A
0DE3' DB 02 C IN A, (SYS_DTB)
0DE5' 47 C LD B, A
0DE6' E6 06 C AND 00000110B ;GET DRIVE BITS ONLY
0DE8' B9 C CP C
0DE9' 20 02 C JR NZ, XJ185 ;IF DRIVE ISN'T ALREADY SELECTED
C
0DEB' FB C EI
0DEC' C9 C RET ;RETURN IF DRIVE ALREADY SELECTED
C PAGE

```

```

C
C
C      ;SELECT DRIVE
C
C      ;TURN DRIVE ON
0DED' DB 02 C      XJ185:      IN      A, (SYS_DTB)      ;GET SETTINGS
0DEF' E6 F9 C              AND      11111001B      ;SET DRIVE BITS TO 0
0DF1' B1   C              OR       C              ;ADD NEW DRIVE BITS
0DF2' D3 02 C              OUT      (SYS_DTB), A      ;SAVE NEW SETTING
C
0DF4' 3E 19 C              LD       A, 25
0DF6' 32 23E8 C           LD       (DCHK_FLG), A      ;DON'T CHECK FOR DISK CHANGE
0DF9' FB   C              EI              ;& DON'T LEAVE INTERRUPTS DISABLED WHILE
C                                     ;MOTOR SPINS UP.
C
C      ;SPIN UP
0DFA' 3E FA C              LD       A, 250
0DFC' CD 0E69' C           CALL      DELAY      ;WAIT FOR MOTOR SPIN UP
0DFF' 3E FA C              LD       A, 250
0E01' CD 0E69' C           CALL      DELAY      ;2ND DELAY
C
0E04' F6 FF C              OR       0FFH      ;INDICATED DRIVE WAS NOT ALREADY SELECTED
C                                     ;CLEAR ZBIT
0E06' C9   C              RET
C              PAGE

```

DAB12/13


```

0E07'      01 0000      LD      BC,0
0E0A'      DB 0B      XJ186:    IN      A,(DSK_STS)      ;(11)
0E0C'      57          LD      D,A          ;SAVE STATUS
0E0D'      0F          RACA          ;(4) DS.BSY (CBIT RESET WITH ANI INSTRUCTION)
0E0E'      30 0D      JR      NC,XJ187      ;(7) GOOD RETURN
0E10'      E3          EX      (SP),HL      ;(19) DELAY
0E11'      E3          EX      (SP),HL      ;(19) DELAY
0E12'      E3          EX      (SP),HL      ;(19) DELAY
0E13'      E3          EX      (SP),HL      ;(19) DELAY
0E14'      0B          DEC     BC          ;(6)
0E15'      78          LD      A,B          ;(4)
0E16'      B1          OR      C          ;(4)
0E17'      20 F1      JR      NZ,XJ186      ;(12) IF NOT TIME-OUT
0E19'      CB FA      SET     7,D          ;TIME OUT ERROR (NOT READY BIT)
0E1B'      7A          LD      A,D          ;FDC STATUS IS IN A
0E1C'      37          SCF          ;SET ERROR
0E1D'      C9      XJ187:    RET

```

```

C
C
0E1E' C      FORMAT:
C      ;      THIS PROC WILL FORMAT THE NEXT TRACK IN IBM 3740 FORMAT CONSISTING OF 40 TRACKS, WITH EACH
C      ;TRACK CONTAINING 10 SECTORS.
C
C      ;ENTRY
C      ;SAVTRK =      THE TRACK TO BE FORMATED
C
C      ;EXIT
C      ;Z      =      0 IF ERROR, 1 IF NO ERROR
C      ;A      =      0 IF NO ERROR, FDC STATUS IF ERROR
C
C
C      ;SELECT DRIVE
0E1E' CD 0DBB' C      CALL      SELDRV
C
C      ;TEST FOR STEP OR NO-STEP
0E21' 21 FD4E C      LD      HL, SAVTRK
0E24' DB 09   C      IN      A, (DSK_TRK)      ;TRACK REG
0E26' BE      C      CP      (HL)
0E27' 2B 17   C      JR      Z, XJ188      ;IF SAVTRK AND TRACK REG ARE THE SAME SKIP THE STEP
C
C      ;STEP IN ONE TRACK
0E29' 3A 23F0 C      LD      A, (SEKDEL)
0E2C' F6 10   C      OR      00010000B      ;UPDATE
0E2E' 32 23F0 C      LD      (SEKDEL), A      ;SET UP SEKDEL
C
0E31' CD 0C32' C      CALL      STEPIN
C
0E34' F5      C      PUSH     AF      ;SAVE CFLAG
0E35' 3A 23F0 C      LD      A, (SEKDEL)
0E38' EE 10   C      XOR      00010000B      ;TOGGLE UPDATE BIT
0E3A' 32 23F0 C      LD      (SEKDEL), A      ;RESET SEKDEL
0E3D' F1      C      POP      AF      ;RESTORE
C
0E3E' 3B 27   C      JR      C, XJ189
C      PAGE

```

```

C
C
C      ;SET "DMADR"
C      ;MOVE THE LENGTH BYTES TO A SPOT WHERE THEY CAN BE READ
C
0E40'  2A FD4B  C      XJ188:      LD      HL, (DMADR)      ;HL = SOURCE ADDRESS
0E43'  11 FDDC  C              LD      DE, HIBUFF      ;DE = DESTINATION
0E46'  3A FD4A  C              LD      A, (DMABANK)
0E49'  32 FD56  C              LD      (SRCBANK), A      ;SOURCE BANK
0E4C'  3E 80    C              LD      A, ROM_BANK      ;DESTINATION BANK
0E4E'  32 FD57  C              LD      (DESBANK), A
0E51'  01 0002  C              LD      BC, 2            ;MOVE 2 BYTES
0E54'  CD FF9B  C              CALL     BANKMOVE
C
C      ;SAVE DMADR
0E57'  22 FD4B  C              LD      (DMADR), HL      ;BANKMOVE RETURNS WITH
C      ;HL = NEXT ADDRESS IN SOURCE STRING
C      ;GET LENGTH
0E5A'  21 FDDC  C              LD      HL, HIBUFF
0E5D'  4E      C              LD      C, (HL)
0E5E'  23      C              INC     HL
0E5F'  46      C              LD      B, (HL)            ;BC = LENGTH OF FORMAT DATA
C
C      ;FORMAT TRACK
0E60'  CD 0D3B' C              CALL     FMTTRK
0E63'  3B 02    C              JR      C, XJ189
C
0E65'  AF      C              XOR     A
0E66'  C9      C              RET
C
0E67'          C      XJ189:          ;A = FDC STATUS
0E67'  B7      C              OR      A
0E68'  C9      C              RET
C              PAGE

```

```

C
C
0E69' C      DELAY:
C      ;'N' MILLISECONDS
C      ;ENTRY
C      ;A      =      NUMBER OF MILLISECONDS TO DELAY
C      ;SCLFRE =      (FREQ/1000)/(# OF TICS IN LOOP)
C
C      ;EXIT
C      ;NONE
C
C
0E69' 4F      C      LD      C,A
0E6A' 3E B5    C      XJ190:  LD      A,SCLFRE
C
0E6C' 3D      C      XJ191:  DEC     A           ;(4 TICS)
0E6D' 40      C      LD      B,B           ;(4 TICS)
0E6E' 49      C      LD      C,C           ;(4 TICS)
0E6F' 20 FB    C      JR      NZ,XJ191      ;(10 TICS) IF 1 MS NOT ELAPSED
C
0E71' 0D      C      DEC     C
0E72' 20 F6    C      JR      NZ,XJ190      ; IF REQUESTED MSEC NOT DONE
C
0E74' C9      C      RET
C      PAGE

```

```

C
C
C      ;      *****
C      ;      *
C      ;      *          DISK EQUATES          *
C      ;      *
C      ;      *****
C
0010  C      D.SEK      EQU      010H      ;SEEK
0020  C      D.STP      EQU      020H      ;STEP
0040  C      D.STPI     EQU      040H      ;STEP IN
0060  C      D.STPD     EQU      060H      ;STEP OUT
0080  C      D.RDS      EQU      080H      ;READ SECTOR
00A0  C      D.WRTS     EQU      0A0H      ;WRITE SECTOR
00C0  C      D.RDA      EQU      0C0H      ;READ ADDRESS
00E0  C      D.RDT      EQU      0E0H      ;READ TRACK
00F0  C      D.WRTT     EQU      0F0H      ;WRITE TRACK
00D0  C      D.FINT     EQU      0D0H      ;FORCE INTERRUPT
C
0004  C      DRV_A      EQU      04H      ;SELECT DRIVE A
0002  C      DRV_B      EQU      02H      ;SELECT DRIVE B
C

```

PAGE

'EXECUTIVE ROM MONITOR'
'DISK DRIVERS'

MACRO-80 3.44 09-Dec-81 PAGE 1-284

;

SUBTTL
PAGE

'INTERRUPT HANDLING ROUTINES'

```

C      SUBTTL      'INTERRUPT HANDLING ROUTINES'
C      INCLUDE ROM6                                ;INTERRUPT HANDLING ROUTINES
C      ;
C      ;
C      ; *****
C      ; *                                           *
C      ; *           INTERRUPT HANDLERS           *
C      ; *                                           *
C      ; *****
C      ;
C      ; MODULE 6 --ROM6-- ENTRY POINTS
C      ;
C      ; INTERRUPT HANDLING ROUTINES
C      ;
C      ; GET_INT:      GET INTERRUPT STATE
C      ; SET_INT:      SET INTERRUPT STATE
C      ; INT_POLL:     INTERRUPT POLLING
C      ; DMA_INT:      DMA INTERRUPT HANDLER
C      ; PAR_INT:      PARRALLEL INTERRUPT HANDLER
C      ; INT_KBD:      INTELLIGENT KEYBOARD INTERRUPT HANDLER
C      ; FDC_INT:      FLOPPY DISK CONTROLLER INTERRUPT HANDLER
C      ; RESET_INT:    RESET INTERRUPT HANDLER
C      ; S_R_I_CH_A:    SIO PORT (CHANNEL) A RECIEVE INTERRUPT HANDLER
C      ; S_R_I_CH_B:    SIO PORT B RECIEVE INTERRUPT HANDLER
C      ; S_T_I_CH_A:    SIO PORT A TRANSMIT INTERRUPT HANDLER
C      ; S_T_I_CH_B:    SIO PORT B TRANSMIT INTERRUPT HANDLER
C      ; EXT_CH_A:      SIO PORT A EXTERNAL/STATUS INTERRUPT HANDLER
C      ; EXT_CH_B:      SIO PORT B EXTERNAL/STATUS INTERRUPT HANDLER
C      ; CH_A_SPEC:     SIO PORT A SPECIAL RX CONDITION
C      ; CH_B_SPEC:     SIO PORT B SBECIAL RX CONDITION
C      ; SPEC_HD:       SPECIAL HANDELING ROUTINE FOR BUFFERING
C      ; RTC_INT:       REAL TIME CLOCK INTERRUPT HANDLER
C      ; RTC_DRV:       REAL TIME CLOCK DRIVER -- ALSO DISK CHANGE DETECT
C
C      PAGE
```

'INTERRUPT HANDLING ROUTINES'

0E75'

```
C
C
C      .Z80
C      ;
C      ; .XLIST
C      ; INCLUDE      SYSS1.MAC          ;SYSTEST FILE
C      ; INCLUDE      RAMSMON.MAC        ;MONITOR RAM MEMORY LOCATIONS
C      ; INCLUDE      RAMS10.MAC         ;RAM MEMORY LOCATIONS AND CONSTANTS
C      ;
C      ; .LIST
C      ; ;SYSS1.MAC , RAMSMON.MAC AND RAMS10.MAC INCLUDED
C
C      ; EXT      INT_HD,FL_B_CH, RM_C_ST, RM_C_BF, D.FINT, KBDVR
C
C      ; PUBLIC  GET_INT, SET_INT, SPEC_HD, R_INT_TBL, R_I_TBL_LEN, INT_POLL
C
C      CSEG
C
C      PAGE
```



```
C
C
C
C ;-----
C ; SHELLY INTERRUPT STRUCTURE
C ; BY
C ; DAVID ANDREW BERG
C ; 29 OCTOBER 1982
C ;-----
C ; THE FOLLOWING INTERRUPTS WILL BE SUPPORTED:
C ;
C ; PIA INTERRUPTS:
C ;     DMA (SYSTEM PIA CA1)
C ;     INTELLIGENT KEYBOARD (SYSTEM PIA CA2)
C ;     REAL TIME CLOCK (SYSTEM PIA CB1)
C ;     IEEE 488 (PARALLEL PIA CA1)
C ;     FLOPPY DISK CONTROLLER (PARALLEL PIA CB1)
C ;     RESET INTERRUPT
C ;
C ; SIO INTERRUPTS (ON EITHER CHANNEL A OR B):
C ;     RX CHARACTER
C ;     RX CHAR W/PARITY ERR
C ;     TX BUFFER EMPTY
C ;     EXTERNAL/STATUS TRANSITION (DCD, CTS, SYNC, BREAK/ABORT,
C ;     OR TRANSMIT UNDERRUN/EOM)
C ;
C ;
C ; THEY WILL BE POLLED IN THE FOLLOWING ORDER TO MINIMIZE INFORMATION
C ; LOSS AND DUE TO HARDWARE CONSIDERATIONS:
C ;
C ;     DMA
C ;     IEEE-488
C ;     SIO INTERRUPTS
C ;     INTELLIGENT KEYBOARD
C ;     REAL TIME CLOCK (KEYBOARD SCAN)
C ;     FDC
C ;
C ; SIO INTERRUPTS ARE HANDLED BY READING THE INTERRUPT VECTOR FROM
C ; RR2 ON CHANNEL B AND USING BITS 1-3 AS AN INDEX INTO THE
C ; INTERRUPT TABLE. BITS 1-3 ARE SET ACCORDING TO THE
C ; INTERRUPTING CONDITION OF THE SIO, SINCE THE SIO IS SET
C ; TO STATUS AFFECTS VECTOR IN WR1.
C ;
C ; IF NO INTERRUPT HAS OCCURRED, A SYSTEM RESET IS PERFORMED.
C ;
C ; DETECTION OF ANY INTERRUPT CAUSES THE SYSTEM TO SWITCH IN THE BANK
C ; AND JUMP TO THE ADDRESS SPECIFIED BY THE INTERRUPT'S TABLE ENTRY.
C ; CONTROL IS RETURNED TO THE USER'S PROGRAM BY THE INTERRUPT HANDLER.
C ; 14 3-BYTE TABLE ENTRIES ARE REQUIRED.
C ;
C ; ONCE AN INTERRUPT IS DETECTED IT IS SERVICED AND CONTROL IS RETURNED
C ; TO THE USER'S PROGRAM. THIS ALLOWS A HIGH PRIORITY INTERRUPT TO BE
C ; SERVICED BEFORE LOWER PRIORITY INTERRUPTS.
C ;
C ; PAGE
```

```
C
C
C ;-----
C ; INTERRUPT DETECTION STRATEGY
C ;-----
C ; THE FOLLOWING INTERRUPT DETECTION STRATEGY IS IMPLEMENTED IN THE
C ; INTERRUPT POLLING ROUTINE:
C ;
C ; DMA:
C ;     EXAMINE CA1 LINE OF SYS PIA (SYS_CNA BIT 7)
C ;     IF SET,
C ;         SERVICE DMA INT
C ;
C ; IEEE 488 RECEIVE INTERRUPT (WHEN IMPLEMENTED) IS DETECTED BY
C ; EXAMINE THE SRQ LINE ON THE PARALLEL PIA (CA1: BIT 7 ON
C ; PAR_CNA).
C ;
C ; SIO INTERRUPTS ARE DETECTED BY:
C ; EXAMINING THE INT PENDING BIT (BIT 1) OF RR0 ON CH A (SIO_CNA)
C ; IF SET,
C ;     LOAD THE SIO INTERRUPT VECTOR (RR2)
C ;     USE BITS 1-3 AS AN INDEX INTO THE INTERRUPT TABLE
C ;     (SHIFT RIGHT ONCE)
C ;
C ; INTELLIGENT KEYBD INTERRUPT IS DETECTED BY
C ; EXAMINING CA2 LINE OF THE SYSTEM PIA (SYS_CNA BIT 6)
C ; IF ACTIVE,
C ;     SERVICE INTELLIGENT KEYBD INTERRUPT
C ;
C ; RTC:
C ;     EXAMINE CB1 LINE OF THE SYSTEM PIA (PORT SYS_CNB BIT 7)
C ;     IF ACTIVE,
C ;         SERVICE RTC INTERRUPT
C ;
C ; FDC:
C ;     EXAMINE THE CB1 LINE ON THE PARALLEL PIA.
C ;     IF ACTIVE,
C ;         SERVICE FDC INT
C ;
C ; IF NO INTERRUPTS DETECTED,
C ;     SERVICE RESET INTERRUPT.
C ;-----
C
C
C PAGE
```

'INTERRUPT HANDLING ROUTINES'

0E75'

```

C
C
C      GET_INT:
C      ;      ROUTINE TO GET INTERRUPT STATE.
C      ;
C      ;      CUR_I_ST MUST BE ZEROED AT SYSTEM INITIALIZATION.
C      ; ENTRY:
C      ; NONE
C
C      ; EXIT:
C      ;      INTERRUPT STATE IS RETURNED IN BC.
C      ;      THE BIT POSITIONS FOR EACH INTERRUPT ARE:
C      ;
C      ;
C      ; REGISTER B:
C      ;      +---+---+---+---+---+---+---+---+
C      ;      | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
C      ;      +---+---+---+---+---+---+---+---+
C      ;      |   |   |   |   |   |   |   |   |
C      ; DMA      (-+ |   |   |   |   |   |   |
C      ; PARALLEL PORT (-+ |   |   |   |   |   |   |
C      ; INTELLIGENT KB (-+ |   |   |   |   |   |   |
C      ; REAL TIME CLOCK (-+ |   |   |   |   |   |   |
C      ; FLOPPY DISK CON (-+ |   |   |   |   |   |   |
C      ; UNUSED      (-+ |   |   |   |   |   |   |
C      ; UNUSED      (-+ |   |   |   |   |   |   |
C      ; UNUSED      (-+ |   |   |   |   |   |   |
C      ;
C      ; REGISTER C:
C      ;      +---+---+---+---+---+---+---+---+
C      ;      | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
C      ;      +---+---+---+---+---+---+---+---+
C      ;      |   |   |   |   |   |   |   |   |
C      ; UNUSED    (-+ |   |   |   |   |   |   |
C      ; UNUSED    (-+ |   |   |   |   |   |   |
C      ; UNUSED    (-+ |   |   |   |   |   |   |
C      ; UNUSED    (-+ |   |   |   |   |   |   |
C      ; UNUSED    (-+ |   |   |   |   |   |   |
C      ; UNUSED    (-+ |   |   |   |   |   |   |
C      ; UNUSED    (-+ |   |   |   |   |   |   |
C      ; UNUSED    (-+ |   |   |   |   |   |   |
C      ;
C
C      LD      BC, (CUR_I_ST) ;LOAD CURRENT INTERRUPT STATE TO BC
C      RET                      ;AND RETURN
C
C      PAGE

```

0E75' ED 4B 23D4

0E79' C9

```

C
C
0E7A' C SET_INT:
C ; ROUTINE TO SET INTERRUPT STATE.
C ; ANY PIA INTERRUPTS CAN BE SELECTIVELY ENABLED OR DISABLED.
C ; ENTRY:
C ; BC = VECTOR OF INTERRUPTS STATES TO SET.
C ; A 1 BIT MEANS TO ENABLE AN INTERRUPT,
C ; 0 MEANS TO DISABLE AN INTERRUPT.
C ; (SEE GET_INT FOR A DESCRIPTION OF EACH BIT POSITION)
C ;
C ; EXIT:
C ; INTERRUPTS ARE APPROPRIATELY ENABLED OR
C ; DISABLED. INTERRUPTS ARE ENABLED UPON EXIT.
C
C
0E7A' F3 C DI ;DISABLE INTERRUPTS TO PREVENT DISRUPTION
0E7B' ED 43 23D4 C LD (CUR_I_ST),BC ;SAVE STATE TO SET AS CURRENT
C
C ;SET DMA
0E7F' DB 01 C IN A, (SYS_CNA) ;READ SYSTEM PIA CH. A
0E81' F6 01 C OR 00000001B ;ASSUME DMA ENABLE
C
0E83' CB 20 C SLA B ;CHECK DMA BIT.
0E85' 38 02 C JR C,XJ194 ;IF ENABLE
C
0E87' E6 FE C AND 1111110B ;IF DISABLE
C
0E89' D3 01 C XJ194: OUT (SYS_CNA),A ;WRITE A TO PIA
C
C
C ;SET PARALLEL PORT
0E8B' DB 11 C IN A, (PAR_CNA) ;READ CHANNEL A OF PARALLEL PIA
0E8D' F6 01 C OR 00000001B ;ASSUME ENABLE INTS
C
0E8F' CB 20 C SLA B ;IS PARALLEL BIT SET?
0E91' 38 02 C JR C,XJ196 ;YES, ENABLE PARALLEL INT...
C
0E93' E6 FE C AND 1111110B ;NO, DISABLE PARALLEL INT
C
0E95' D3 11 C XJ196: OUT (PAR_CNA),A
C
C
C ;SET INTELLIGENT KEYBOARD
0E97' DB 01 C IN A, (SYS_CNA) ;READ SYSTEM PIA
0E99' F6 08 C OR 00001000B ;ASSUME ENABLE
C
0E9B' CB 20 C SLA B ;IS INTELLIGENT KEYBD BIT SET?
0E9D' 38 02 C JR C,XJ197 ;YES, ENABLE IT
C
0E9F' E6 F7 C AND 11110111B ;NO, DISABLE INT
C
0EA1' D3 01 C XJ197: OUT (SYS_CNA),A
C
C

```

'INTERRUPT HANDLING ROUTINES'

```
C      ;SET REAL TIME CLOCK (60HZ) INTERRUPT STATE
0EA3'  DB 03      C      IN      A, (SYS_CNB)      ;READ CHANNEL B OF SYSTEM PIA
0EA5'  F6 01      C      OR      00000001B      ;ASSUME ENABLE RTC INT. SET BIT 0.
C
0EA7'  CB 20      C      SLA      B      ;IS RTC BIT SET?
0EA9'  38 02      C      JR      C, XJ198      ;YES, ENABLE RTC
C
0EAB'  E6 FE      C      AND      11111110B      ;NO, DISABLE
C
0EAD'  D3 03      C      XJ198: OUT      (SYS_CNB), A
C
C
C      ;SET FLOPPY DISK CONTROLLER INTERRUPT STATE
0EAF'  DB 13      C      IN      A, (PAR_CNB)      ;READ CHANNEL B OF PARALLEL PIA
0EB1'  F6 01      C      OR      00000001B      ;ASSUME ENABLE
C
0EB3'  CB 20      C      SLA      B      ;IS FDC BIT SET?
0EB5'  38 02      C      JR      C, XJ199      ;YES, ENABLE IT
C
0EB7'  E6 FE      C      AND      11111110B      ;NO, DISABLE IT
C
0EB9'  D3 13      C      XJ199: OUT      (PAR_CNB), A      ;WRITE FDC STATE
C
0EBB'  FB      C      EI      ;RE-ENABLE INTERRUPTS
0EBC'  C9      C      RET      ;AND RETURN
C
C      PAGE
```

```

C
C
C
C      ;-----
C      ;      INTERRUPT POLLING ROUTINE
C      ;-----
0002    C      S_INT_PDG      EQU      00000010B      ;INTERRUPT PENDING MASK FOR SIO RR0
C
C      ;
0080    C      IRQ1      EQU      10000000B      ;INTERRUPT PENDING MASK FOR PIA IRQ1
0040    C      IRQ2      EQU      01000000B      ;INTERRUPT PENDING MASK FOR PIA IRQ2
C
C      ;
0EBD'   C      INT_POLL:
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;HL      ==>      ENTRY IN INTERRUPT TABLE
C
C
0EBD'   C      XJ200: ;CHECK FOR DMA INTERRUPT
0EBD'   C      LD      E,0      ;FIRST ENTRY IN TABLE (E = ENTRY NUMBER)
0EBF'   C      IN      A,(SIO_CNA) ;CHECK FOR SIO INT
0EC1'   C      AND      S_INT_PDG ;TEST INTERRUPT PDG BIT (1)
0EC3'   C      JP      Z,XJ201 ;CK_I_KB ;NOT SET...NO SIO INTERRUPT. CHECK INT. KBD
C
C      ;SIO INTERRUPT PDG:
C      ;READ INTERRUPT VECTOR
0EC6'   C      LD      A,2      ;POINT TO RR2 ON CHANNEL B
0EC8'   C      OUT     (SIO_CNB),A ;
C
C      IN      A,(SIO_CNB) ;READ RR2: INTERRUPT VECTOR
C
0ECC'   C      AND      00001110B ;STRIP OFF ALL BITS BUT 1-3
0ECE'   C      SRA      A ;SHIFT RIGHT SO A CONTAINS INDEX INTO TABLE
0ED0'   C      ADD      A,E ;A = ENTRY NUMBER
0ED1'   C      LD      E,A ;E = ENTRY NUMBER
C
0ED2'   C      JP      XJ202 ;HANDLE INTERRUPT PER TABLE ENTRY
C
0ED5'   C      XJ201:
C      ;CHECK FOR IEEE-488 (PARALLEL) INT
0ED5'   C      LD      A,8 ;BUMP ENTRY NUMBER COUNT PAST SIO VECTORS
0ED7'   C      ADD      A,E
0ED8'   C      LD      E,A
C
0ED9'   C      IN      A,(SYS_CNB) ;
0EDB'   C      AND      IRQ1 ;
0EDD'   C      JP      NZ,XJ202 ;HANDLE RTC INTERRUPT
C      PAGE

```

'INTERRUPT HANDLING ROUTINES'

```

C
C
0EE0'      C      XJ203: ;CHECK SID
0EE0'      1C      C      INC      E      ;NEXT ENTRY IN TBL
0EE1'      DB 11    C      IN       A, (PAR_CNA) ;READ PARALLEL PIA CONTROL PORT A
0EE3'      E6 80    C      AND      IRQ1      ;CHECK CA1 (IRQ1) BIT (7): SRQ LINE
0EE5'      C2 0F01' C      JP       NZ, XJ202      ;NO INTERRUPT PDG...CHECK SID
C
0EE8'      C      XJ204: ;CHECK INTELLIGENT KEYBD
C
0EE8'      1C      C      INC      E      ;NEXT ENTRY IN TBL
0EE9'      DB 01    C      IN       A, (SYS_CNA) ;READ SYSTEM PIA CONTROL REG
0EEB'      E6 40    C      AND      IRQ2      ;CHECK CA2 (IRQ2) BIT (6)
0EED'      C2 0F01' C      JP       NZ, XJ202      ;IT'S SET...
C
0EF0'      C      XJ205: ;NOT INT. KBD, CHECK RTC
0EF0'      1C      C      INC      E      ;NEXT ENTRY IN TBL: TICK INTERRUPT
0EF1'      DB 01    C      IN       A, (SYS_CNA) ;CHECK CA1 (IRQ1) BIT (7) ON SYSTEM PIA CH A
0EF3'      E6 80    C      AND      IRQ1      ;
0EF5'      C2 0F01' C      JP       NZ, XJ202      ;IT'S ACTIVE...SERVICE INT
C
0EF8'      C      XJ206: ;NOT RTC, CHECK FOR FDC INTERRUPT
0EF8'      1C      C      INC      E      ;NEXT ENTRY IN TBL
0EF9'      DB 13    C      IN       A, (PAR_CNB) ;CHECK CB1 (IRQ1) LINE OF PARALLEL PIA CH. B
0EFB'      E6 80    C      AND      IRQ1      ;
0EFD'      C2 0F01' C      JP       NZ, XJ202      ;IT'S ACTIVE...SERVICE INTERRUPT
C
0F00'      C      XJ207: ;THIS MUST BE A INTERRUPT GENERATED
C      ;BY PRESSING THE SYSTEM RESET BUTTON
0F00'      1C      C      INC      E      ;NEXT ENTRY IN TBL
C      ;NONE...HANDLE SYSTEM RESET INT
C
0F01'      C      XJ202: ;POINT TO TABLE ENTRY
0F01'      21 23F3 C      LD       HL, INT_TBL ;POINT HL TO INTERRUPT TBL
0F04'      7B      C      LD       A, E      ;E = ENTRY NUMBER
0F05'      87      C      ADD      A, A      ;*2
0F06'      83      C      ADD      A, E      ;*3 (ENTRY LENGTH = 3)
0F07'      5F      C      LD       E, A
0F08'      16 00    C      LD       D, 0      ;DE = INDEX INTO TABLE
0F0A'      19      C      ADD      HL, DE      ;HL ==> TABLE ENTRY
0F0B'      C3 FF25 C      JP       INT_HD ;HANDLE INTERRUPT
C
C      PAGE

```

'INTERRUPT HANDLING ROUTINES'

```

C
C
C
C ;-----
C ; PIA INTERRUPT HANDLERS
C ;-----
C ;
C ; THESE HANDLERS ARE FOR INTERRUPTS GENERATED BY THE PIAS. NO REAL
C ; HANDLING IS DONE. THE INTERRUPT CONDITION IS SIMPLY CLEARED.
C ; THE REAL TIME CLOCK INTERRUPT IS NOT INCLUDED SINCE IT ALREADY
C ; EXISTS IN THE ROM.
C ;
C
0F0E' C DMA_INT:
C ; DMA INTERRUPT HANDLER
0F0E' DB 00 C IN A, (SYS_DTA) ;
0F10' C9 C RET ;
C
C
C
0F11' C PAR_INT:
C ; PARALLEL INTERRUPT HANDLER
0F11' DB 10 C IN A, (PAR_DTA) ;
0F13' C9 C RET ;
C
C
C
0F14' C INT_KBD:
C ; INTELLIGENT KEYBOARD INTERRUPT HANDLER
0F14' DB 00 C IN A, (SYS_DTA) ;
0F16' C9 C RET ;
C
C
C
0F17' C FDC_INT:
C ; FLOPPY DISK CONTROLLER INTERRUPT HANDLER
0F17' DB 12 C IN A, (PAR_DTB) ;
0F19' C9 C RET ;
C
C PAGE

```



```

C
C
C ;-----
C ;      RESET INTERRUPT HANDLER
C ;-----
C
C RESET_INT:
C     LD      A,0FFH
C     LD      (RSTHLD),A      ;INDICATE RESET BUTTON PUSHED
C     RET
C
C
C     PAGE
```

0F1A' 3E FF
0F1C' 32 2760
0F1F' C9

[illegible]

```

C      ;      WAS DISABLED.
C      ;
C      ;      IF A PROTOCOL IS ENABLED, ALL PROTOCOL CHARS ARE DISCARDED.
C      ;
C      ;      IF NO PROTOCOL IS ENABLED, ALL CHARS ARE BUFFERED.
C      ;
C      ;
C      ;
0F20'   DB 0C      C      IN      A, (SID_DTA)      ;GET CHARACTER FROM SID
0F22'   4F          C      LD      C,A          ;SAVE IT IN C
0F23'   3A 2008     C      LD      A, (A_PROTCL)      ;IS THERE A PROTOCOL ENABLED FOR THIS CHANNEL?
0F26'   B7          C      OR      A          ;
0F27'   CA 0F5C'    C      JP      Z, XJ208          ;NO...BUFFER CHAR
C      ;
C      ;
0F2A'   C          C      XJ209: ;A PROTOCOL IS ENABLED.
0F2A'   3D          C      DEC      A          ;IS IT XON/XOFF?
0F2B'   C2 0F4B'    C      JP      NZ, XJ210          ;NO, IT MUST BE ETX/ACK.
C      ;
C      ;      ;PROTOCOL IS XON/XOFF.
0F2E'   79          C      LD      A,C          ;RESTORE CHAR TO A
0F2F'   FE 13       C      CP      XOFF          ;IS IT XOFF?
0F31'   C2 0F3B'    C      JP      NZ, XJ211          ;NO...
C      ;
C      ;      ;YES. DISABLE TRANSMISSION.
0F34'   3E FF       C      LD      A, 0FFH          ;
0F36'   32 275E     C      LD      (A_XMT_OFF), A      ;SET XMT DISABLE FLAG
0F39'   ED 4D       C      RETI          ;AND RETURN FROM INTERRUPT.
C      ;
C      ;
0F3B'   C          C      XJ211: ;CHAR REC'D IS NOT XOFF.
0F3B'   FE 11       C      CP      XON          ;IS IT XON?
0F3D'   C2 0F5C'    C      JP      NZ, XJ208          ;NO, BUFFER CHAR AND RETURN FROM INTERRUPT
C      ;
C      ;      ;YES. SEE IF XMT DISABLED. IF IT IS, ENABLE IT
C      ;      ;AND CALL XMT INTERRUPT HANDLER TO RESUME XMT.
0F40'   21 275E     C      LD      HL, A_XMT_OFF      ;POINT AT XMT DISABLE FLAG.
0F43'   AF          C      XOR      A          ;CLEAR A
0F44'   BE          C      CP      (HL)          ;IS FLAG NOT ZERO?
0F45'   77          C      LD      (HL), A          ;RESET IT REGARDLESS.
0F46'   C4 0F46'    C      CALL     NZ, S_T_I_CH_A      ;IF XMT WAS DISABLED, CALL INTERRUPT HANDLER
C      ;      ;TO RESUME XMT
0F49'   ED 4D       C      RETI          ;AND RETURN FROM INTERRUPT.
C
C      PAGE

```

'INTERRUPT HANDLING ROUTINES'

```

C
C
0F4B'      C      XJ210: ;PROTOCOL IS ETX/ACK.
0F4B'  79      C      LD      A,C          ;IS CHAR ACK?
0F4C'  FE 06    C      CP      ACK          ;
0F4E'  C2 0F5C' C      JP      NZ,XJ208      ;NO. RETURN FROM INTERRUPT AND DISCARD CHAR.
C      ;
C      ;YES. SEE IF XMT DISABLED. IF IT IS, ENABLE IT
C      ;AND CALL XMT INTERRUPT HANDLER TO RESUME XMT.
0F51'  21 275E C      LD      HL,A_XMT_OFF ;POINT AT XMT DISABLE FLAG.
0F54'  AF      C      XOR      A          ;CLEAR A
0F55'  BE      C      CP      (HL)       ;IS FLAG NOT ZERO?
0F56'  77      C      LD      (HL),A     ;RESET IT REGARDLESS.
0F57'  C4 0FA6' C      CALL     NZ,S_T_I_CH_A ;IF XMT WAS DISABLED, CALL INTERRUPT HANDLER
C      ;TO RESUME XMT
0F5A'  ED 4D    C      RETI          ;AND RETURN FROM INTERRUPT.
C
0F5C'      C      XJ208: ;SAVE CHAR IN C IN BUFFER
0F5C'  06 81    C      LD      B,S_CH_A_I  + HI_PRIOR ; SPECIFY DEVICE AND HIGH PRIORITY
0F5E'  CD 11C5' C      CALL     FL_B_CH ;BUFFER THE CHAR
0F61'  ED 4D    C      RETI          ;AND RETURN FROM INTERRUPT.
C
C      PAGE

```

```

C
C
0F63' C S_R_I_CH_B:
C ; SIO CHANNEL B RECEIVE INTERRUPT HANDLER
C ;
C ; XON/XOFF AND ETX/ACK PROTOCOLS ARE HANDLED WITHIN.
C ; B_PROTCL INDICATES WHETHER A PROTOCOL IS ENABLED OR NOT:
C ;
C ; B_PROTCL = 0 IF NO PROTOCOL IS ENABLED,
C ; 1 IF XON/XOFF IS ENABLED OR
C ; 2 IF ETX/ACK IS ENABLED.
C ;
C ; IF XON/XOFF IS ENABLED AND XOFF IS RECEIVED, A FLAG IS SET INDICATING TO
C ; THE TRANSMIT INTERRUPT HANDLER THAT TRANSMITTING IS TO BE STOPPED UNTIL
C ; XON IS RECEIVED.
C ;
C ; WHEN XON IS RECEIVED, THE FLAG IS CLEARED AND IF THE FLAG WAS SET, THE
C ; TRANSMIT INTERRUPT HANDLER IS CALLED TO RESUME TRANSMISSION.
C ;
C ; IF ETX/ACK IS ENABLED AND ACK IS RECEIVED, THE TRANSMIT DISABLE FLAG
C ; IS RESET AND THE TRANSMIT INTERRUPT HANDLER IS CALLED IF TRANSMISSION
C ; WAS DISABLED.
C ;
C ; IF A PROTOCOL IS ENABLED, ALL PROTOCOL CHARS ARE DISCARDED.
C ;
C ; IF NO PROTOCOL IS ENABLED, ALL CHARS ARE BUFFERED.
C ;
C ;
C
0F63' DB 0E C IN A, (SIO_DTB) ;GET CHARACTER FROM SIO
0F65' 4F C LD C, A ;SAVE IT IN C
0F66' 3A 2009 C LD A, (B_PROTCL) ;IS THERE A PROTOCOL ENABLED FOR THIS CHANNEL?
0F69' B7 C OR A ;
0F6A' CA 0F9F' C JP Z, XJ212 ;NO...BUFFER CHAR
C ;
C
0F6D' C XJ213: ;A PROTOCOL IS ENABLED.
0F6D' 3D C DEC A ;IS IT XON/XOFF?
0F6E' C2 0F8E' C JP NZ, XJ214 ;NO, IT MUST BE ETX/ACK.
C ;
C ;PROTOCOL IS XON/XOFF.
0F71' 79 C LD A, C ;RESTORE CHAR TO A
0F72' FE 13 C CP XOFF ;IS IT XOFF?
0F74' C2 0F7E' C JP NZ, XJ215 ;NO...
C ;
C ;YES. DISABLE TRANSMISSION.
0F77' 3E FF C LD A, 0FFH ;
0F79' 32 275F C LD (B_XMT_OFF), A ;SET XMT DISABLE FLAG
0F7C' ED 4D C RETI ;AND RETURN FROM INTERRUPT.
C ;
C
0F7E' C XJ215: ;CHAR REC'D IS NOT XOFF.
0F7E' FE 11 C CP XON ;IS IT XON?
0F80' C2 0F9F' C JP NZ, XJ212 ;NO, BUFFER CHAR AND RETURN FROM INTERRUPT
C ;

```

'INTERRUPT HANDLING ROUTINES'

```

                                C      ;YES. SEE IF XMT DISABLED. IF IT IS, ENABLE IT
                                C      ;AND CALL XMT INTERRUPT HANDLER TO RESUME XMT.
0F83' 21 275F      C      LD      HL,B_XMT_OFF      ;POINT AT XMT DISABLE FLAG.
0F86' AF          C      XOR      A                ;CLEAR A
0F87' BE          C      CP      (HL)              ;IS FLAG NOT ZERO?
0F88' 77          C      LD      (HL),A            ;RESET IT REGARDLESS.
0F89' C4 0FE0'    C      CALL     NZ,S_T_I_CH_B     ;IF XMT WAS DISABLED, CALL INTERRUPT HANDLER
                                C      ;TO RESUME XMT
0F8C' ED 4D      C      RETI                      ;AND RETURN FROM INTERRUPT.
                                C
                                C      PAGE
```

'INTERRUPT HANDLING ROUTINES'

```

C
C
0F8E' C XJ214: ;PROTOCOL IS ETX/ACK.
0F8E' 79 C LD A,C ;IS CHAR ACK?
0F8F' FE 06 C CP ACK ;
0F91' C2 0F9F' C JP NZ,XJ212 ;NO. RETURN FROM INTERRUPT AND DISCARD CHAR.
C ;
C ;YES. SEE IF XMT DISABLED. IF IT IS, ENABLE IT
C ;AND CALL XMT INTERRUPT HANDLER TO RESUME XMT.
0F94' 21 275F C LD HL,B_XMT_OFF ;POINT AT XMT DISABLE FLAG.
0F97' AF C XOR A ;CLEAR A
0F98' BE C CP (HL) ;IS FLAG NOT ZERO?
0F99' 77 C LD (HL),A ;RESET IT REGARDLESS.
0F9A' C4 0FE0' C CALL NZ,S_T_I_CH_B ;IF XMT WAS DISABLED, CALL INTERRUPT HANDLER
C ;TO RESUME XMT
0F9D' ED 4D C RETI ;AND RETURN FROM INTERRUPT.
C
0F9F' C XJ212: ;SAVE CHAR IN C IN BUFFER
0F9F' 06 82 C LD B,S_CH_B_I + HI_PRIOR ; SPECIFY DEVICE AND HIGH PRIORITY
0FA1' CD 11C5' C CALL FL_B_CH ;BUFFER THE CHAR
0FA4' ED 4D C RETI ;AND RETURN FROM INTERRUPT.
C
C PAGE

```



```

C
C
0FA6' C S_T_I_CH_A:
C ; SIO CHANNEL A TRANSMIT INTERRUPT HANDLER
C ;
C ; XON/XOFF AND ETX/ACK PROTOCOLS ARE HANDLED WITHIN.
C ; A_PROTCL INDICATES WHETHER A PROTOCOL IS ENABLED OR NOT:
C ;
C ; A_PROTCL = 0 IF NO PROTOCOL IS ENABLED,
C ; 1 IF XON/XOFF IS ENABLED OR
C ; 2 IF ETX/ACK IS ENABLED.
C ;
C ; IF A_XMT_OFF IS SET (0FFH), TRANSMISSION IS DISABLED UNTIL PERMISSION TO
C ; PROCEED IS RECEIVED. TO DISABLE TRANSMISSION, A RESET TRANSMIT INTERRUPT
C ; PENDING COMMAND IS GIVEN TO THE SIO.
C ;
C ; WHEN THE RECEIVE INTERRUPT HANDLER RECEIVES PERMISSION FROM THE DEVICE TO
C ; PROCEED (IN THE FORM OF XON OR ACK), IT RESETS THE TRANSMIT DISABLE
C ; FLAG AND CALLS THIS ROUTINE.
C ;
C ; IF ETX/ACK IS ENABLED AND ETX IS TO BE TRANSMITTED, THE TRANSMIT DISABLE FLAG
C ; IS SET TO PREVENT TRANSMISSION UNTIL ACK IS RECEIVED.
C ;
C
0028 C XJ216 EQU 00101000B ;SIO RESET TRANSMIT INTERRUPT PENDING COMMAND.
C
0FA6' 3A 275E C LD A, (A_XMT_OFF) ;IS XMT DISABLED?
0FA9' B7 C OR A ;
0FAA' C2 0FD8' C JP NZ, XJ217 ;YES...RESET TX INTERRUPT
C ;
C ;NO.
0FAD' 06 03 C LD B, S_CH_A_D ;ARE THERE CHARS TO SEND?
0FAF' CD 1273' C CALL RM_C_ST ;
0FB2' CA 0FD3' C JP Z, XJ218 ;NO...RESET TX INTERRUPT
C ;
C ;YES. GET CHAR.
0FB5' 06 03 C LD B, S_CH_A_D ;
0FB7' CD 1226' C CALL RM_C_BF ;
0FBA' 47 C LD B, A ;SAVE IT IN B
0FBB' 3A 2008 C LD A, (A_PROTCL) ;IS ETX/ACK PROTOCOL ENABLED?
0FBE' 3D C DEC A ;(I.E., A = 2)?
0FBF' 3D C DEC A ;
0FC0' 78 C LD A, B ;RESTORE CHAR TO A REGARDLESS.
0FC1' C2 0FCF' C JP NZ, XJ219 ;NO ETX...
C ;
C ;YES. ETX ENABLED.
0FC4' FE 03 C CP ETX ;IS CHAR ETX?
0FC6' C2 0FCF' C JP NZ, XJ220 ;NO...
C ;
0FC9' 3E FF C LD A, 0FFH ;YES. DISABLE XMT.
0FCB' 32 275E C LD (A_XMT_OFF), A ;
0FCE' 78 C LD A, B ;RESTORE CHAR TO REG A FROM B AND SEND IT
C
C
0FCF' C XJ219: ;ETX/ACK NOT ENABLED

```



```

0FCF'      C      XJ220: ;OR ETX NOT SENT
                        C      ;OR ETX HANDLED, SO SEND CHAR TO SIO
0FCF' D3 0C      C      OUT      (SIO_DTA),A
0FD1' ED 4D      C      RETI      ;AND RETURN FROM INTERRUPT.
                        C      ;
                        C
0FD3' 3E FF      C      XJ218: LD      A,0FFH      ;NO CHARS TO SEND. SET FLAG CHECKED BY
                        C      ;BUFFERING ROUTINE.
0FD5' 32 23D7    C      LD      (S_A_FLG),A      ;IF FLAG SET, NEXT CHAR BUFFERED IS SENT TO
                        C      ;SIO IMMEDIATELY TO RESUME TRANSMISSION.
                        C      ;TRANSMISSION INTERRUPTS ARE NOT REENABLED
                        C      ;AFTER THE XMT INTERRUPT IS CLEARED UNTIL
                        C      ;ANOTHER CHARACTER IS SENT TO THE SIO, SO THE
                        C      ;NEXT CHAR BUFFERED MUST BE SENT TO THE SIO,
                        C      ;OTHERWISE IT WILL NEVER BE SENT.
                        C
                        C      ;XMT DISABLED BECAUSE OF PROTOCOL.
0FD8'      C      XJ217:
0FD8' 3E 28      C      LD      A,XJ216      ;SEND RESET XMT INT PDG COMMAND
0FDA' D3 0D      C      OUT      (SIO_CNA),A      ;
0FDC' D3 0D      C      OUT      (SIO_CNA),A      ;
0FDE' ED 4D      C      RETI      ;AND RETURN FROM INTERRUPT.
                        C
                        C      PAGE

```

```

C
C
0FE0' C S_T_I_CH_B:
C ; SIO CHANNEL B TRANSMIT INTERRUPT HANDLER
C ;
C ; XON/XOFF AND ETX/ACK PROTOCOLS ARE HANDLED WITHIN.
C ; B_PROTCL INDICATES WHETHER A PROTOCOL IS ENABLED OR NOT:
C ;
C ; A_PROTCL = 0 IF NO PROTOCOL IS ENABLED,
C ; 1 IF XON/XOFF IS ENABLED OR
C ; 2 IF ETX/ACK IS ENABLED.
C ;
C ; IF B_XMT_OFF IS SET (0FFH), TRANSMISSION IS DISABLED UNTIL PERMISSION TO
C ; PROCEED IS RECEIVED. TO DISABLE TRANSMISSION, A RESET TRANSMIT INTERRUPT
C ; PENDING COMMAND IS GIVEN TO THE SIO.
C ;
C ; WHEN THE RECEIVE INTERRUPT HANDLER RECEIVES PERMISSION FROM THE DEVICE TO
C ; PROCEED (IN THE FORM OF XON OR ACK), IT RESETS THE TRANSMIT DISABLE
C ; FLAG AND CALLS THIS ROUTINE.
C ;
C ; IF ETX/ACK IS ENABLED AND ETX IS TO BE TRANSMITTED, THE TRANSMIT DISABLE FLAG
C ; IS SET TO PREVENT TRANSMISSION UNTIL ACK IS RECEIVED.
C ;
C
0028 C XJ221 EQU 00101000B ;SIO RESET TRANSMIT INTERRUPT PENDING COMMAND.
C
0FE0' 3A 275F C LD A,(B_XMT_OFF) ;IS XMT DISABLED?
0FE3' B7 C OR A ;
0FE4' C2 1012' C JP NZ,XJ222 ;YES...RESET TX INTERRUPT
C ;
C ;NO.
0FE7' 06 04 C LD B,S_CH_B_D ;ARE THERE CHARS TO SEND?
0FE9' CD 1273' C CALL RM_C_ST ;
0FEC' CA 100D' C JP Z,XJ223 ;NO...RESET TX INTERRUPT
C ;
C ;YES. GET CHAR.
0FEF' 06 04 C LD B,S_CH_B_D ;
0FF1' CD 1226' C CALL RM_C_BF ;
0FF4' 47 C LD B,A ;SAVE IT IN B
0FF5' 3A 2009 C LD A,(B_PROTCL) ;IS ETX/ACK PROTOCOL ENABLED?
0FF8' 3D C DEC A ;(I.E., A = 2)?
0FF9' 3D C DEC A ;
0FFA' 78 C LD A,B ;RESTORE CHAR TO A REGARDLESS.
0FFB' C2 1009' C JP NZ,XJ224 ;NO ETX...
C ;
C ;YES. ETX ENABLED.
0FFE' FE 03 C CP ETX ;IS CHAR ETX?
1000' C2 1009' C JP NZ,XJ225 ;NO...
C ;
1003' 3E FF C LD A,0FFH ;YES. DISABLE XMT.
1005' 32 275F C LD (B_XMT_OFF),A ;AND SEND CHAR...
1008' 78 C LD A,B ;RESTORE CHAR TO REG A FROM B AND SEND IT
C
C
1009' C XJ224: ;ETX/ACK NOT ENABLED

```

```

1009'      C      XJ225: ;OR ETX NOT SENT
              C      ;OR ETX HANDLED, SO SEND CHAR TO SIO
1009'  D3 0E      C      OUT      (SIO_DTB),A
100B'  ED 4D      C      RETI              ;AND RETURN FROM INTERRUPT.
              C      ;
              C
100D'  3E FF      C      XJ223: LD      A,0FFH      ;NO CHARS TO SEND. SET FLAG CHECKED BY
              C      ;BUFFERING ROUTINE.
100F'  32 23D8    C      LD      (S_B_FLG),A      ;IF FLAG SET, NEXT CHAR BUFFERED IS SENT TO
              C      ;SIO IMMEDIATELY TO RESUME TRANSMISSION.
              C      ;TRANSMISSION INTERRUPTS ARE NOT REENABLED
              C      ;AFTER THE XMT INTERRUPT IS CLEARED UNTIL
              C      ;ANOTHER CHARACTER IS SENT TO THE SIO, SO THE
              C      ;NEXT CHAR BUFFERED MUST BE SENT TO THE SIO,
              C      ;OTHERWISE IT WILL NEVER BE SENT.
              C
              C      ;XMT DISABLED BECAUSE OF PROTOCOL.
1012'      C      XJ222:
1012'  3E 28      C      LD      A,XJ221      ;SEND RESET XMT INT PDG COMMAND
1014'  D3 0F      C      OUT      (SIO_CNB),A      ;
1016'  D3 0F      C      OUT      (SIO_CNB),A      ;
1018'  ED 4D      C      RETI              ;AND RETURN FROM INTERRUPT.
              C
              C      PAGE

```

```

C
C
C
C ;-----
C ;   OTHER SIO INTERRUPT HANDLERS
C ;-----
C ;   EXTERNAL/STATUS INTERRUPT HANDLER FOR CHANNEL A
C ;
C ;   RESET EXT/STATUS INTERRUPT AND RETURN
C ;
101A' EXT_CH_A:
C
C
C
0010 XJ226 EQU 010H ;RESET EXT/STATUS INTERRUPT COMMAND
C
C LD A,XJ226 ;SEND RESET COMMAND
101A' 3E 10
C OUT (SIO_CNA),A ;SEND TO SIO
101C' D3 0D
C OUT (SIO_CNA),A ;TWICE TO ACCOUNT FOR BUG IN CHIP
101E' D3 0D
1020' ED 4D
C RETI
C
C PAGE

```

```

C
C
C
C ;-----
C ;      EXTERNAL/STATUS INTERRUPT HANDLER FOR CHANNEL B
C ;
C ;      RESET EXT/STATUS INTERRUPT AND RETURN
C ;
1022' EXT_CH_B:
C
C
C
0010 XJ227 EQU 010H ;RESET EXT/STATUS INTERRUPT COMMAND
C
C      LD A,XJ227 ;SEND RESET COMMAND
1024' D3 0F C      OUT (SIO_CNB),A ;SEND TO SIO
1025' D3 0F C      OUT (SIO_CNB),A ;TWICE TO ACCOUNT FOR BUG IN CHIP
1028' ED 4D C      RETI
C
C      PAGE

```

'INTERRUPT HANDLING ROUTINES'

```
C
C
102A' C CH_B_SPEC:
C ; SPECIAL RX CONDITION ON CHANNEL B
C
C
0030 C XJ228 EQU 30H ;ERROR RESET COMMAND
C
102A' 3E 30 C LD A,XJ228 ;SEND ERROR RESET
102C' D3 0F C OUT (SIO_CNB),A ;
102E' D3 0F C OUT (SIO_CNB),A ;TWICE TO ACCOUNT FOR BUG IN CHIP
C
1030' DB 0E C IN A,(SIO_DTB) ;READ CHAR AND DISCARD
1032' ED 4D C RETI
C
C PAGE
```

```

C
C
1034' C CH_A_SPEC:
C ; SPECIAL RX CONDITION ON CHANNEL A
C
C
0030 C XJ229 EQU 30H ;ERROR RESET COMMAND
C
1034' 3E 30 C LD A,XJ229 ;SEND ERROR RESET
1035' D3 0D C OUT (SIO_CNA),A ;
1038' D3 0D C OUT (SIO_CNA),A ;TWICE TO ACCOUNT FOR BUG IN CHIP
C
103A' DB 0C C IN A,(SIO_DTA) ;READ CHAR AND DISCARD
103C' ED 4D C RETI
C
C PAGE
```

```

C
C
103E' C SPEC_HD:
C ; SPECIAL HANDLING ROUTINE CALLED BY FILL BUFFER WITH CHAR ROUTINE
C ; IF SPECIAL HANDLING BIT (6) OF REGISTER B IS SET.
C ;
C ; CURRENTLY THIS IS USED TO RE-ENABLE SIO TX INTERRUPTS AFTER THEY
C ; HAVE BEEN DISABLED BECAUSE THERE WAS NO CHARACTER TO TRANSMIT.
C ;
C ; THIS ROUTINE IS CALLED AFTER THE CHARACTER HAS BEEN BUFFERED
C ; IMMEDIATELY BEFORE RETURNING FROM THE BUFFER FILL ROUTINE.
C ; ENTRY:
C ; B = NUMBER OF DEVICE THAT WAS FILLED WITH CHAR.
C ; C = CHARACTER THAT WAS PUT INTO BUFFER.
C ;
C ; EXIT:
C ; SIO INTERRUPTS RE-ENABLED BY CALLING SIO TX INTERRUPT HANDLER
C ; IF NECESSARY.
C ;
C
103E' 7B C LD A,B ;PUT DEVICE NO. IN A
103F' E6 3F C AND 00111111B ;STRIP OFF PRIORITY AND SPECIAL HANDLING BITS
C
1041' FE 03 C CP S_CH_A_0 ;IS DEVICE SIO CHANNEL A OUTPUT?
1043' 20 0D C JR NZ,XJ230 ;NO, SEE IF IT IS CHANNEL B DEVICE
C ;
C ;YES. SEE IF INTERRUPTS NEED TO BE RE-ENABLED
1045' 3A 23D7 C LD A,(S_A_FLG) ;LOAD CHANNEL A FLAG
1048' B7 C OR A ;IS IT SET?
1049' C8 C RET Z ;NO, RETURN
C ;
C ;YES.
104A' AF C XOR A ;RESET FLAG
104B' 32 23D7 C LD (S_A_FLG),A ;
C
104E' CD 0FA6' C CALL S_T_I_CH_A ;AND CALL INTERRUPT HANDLER, WHICH WILL
C ;TRANSMIT CHAR IN BUFFER, THEREBY RE-ENABLING
C ;INTERRUPTS.
1051' C9 C RET ;RETURN TO CALLER
C ;
C
1052' C XJ230: ;DEVICE IS NOT SIO TX CHANNEL A.
1052' FE 04 C CP S_CH_B_0 ;IS IT CHANNEL B?
1054' C0 C RET NZ ;NO, RETURN
C ;
1055' 3A 23D8 C LD A,(S_B_FLG) ;YES. IS FLAG SET?
1058' B7 C OR A ;
1059' C8 C RET Z ;NO, RETURN.
C ;
105A' AF C XOR A ;YES. CLEAR FLAG
105B' 32 23D8 C LD (S_B_FLG),A ;
C
105E' CD 0FE0' C CALL S_T_I_CH_B ;CALL CH. B INTERRUPT HANDLER.
1061' C9 C RET ;AND RETURN
C

```


'INTERRUPT HANDLING ROUTINES'

1062'

```
C
C
C      ;      KEYBOARD HANDLER FOR SHELLY
C      ;      INITIAL IMPLEMENTATION FOR USE IN TRESCA
C      ;      DAB 9/9/82 V
C      ;
C
C
C      RTC_INT:
C      ;
C      ;REAL TIME CLOCK INTERRUPT HANDLER
C      ;ENTRY
C      ;NONE
C
C      ;EXIT
C      ;KEYBOARD SCAN, RTC MAINTENANCE AND DISK CHANGE DETECT PROCESSING DONE
C      PAGE
```

'INTERRUPT HANDLING ROUTINES'

```

C
C
C ;-----
C ; REAL TIME CLOCK DRIVER BY DAVID ANDREW BERG
C ;-----
C
C
1062' RTC_DRV:
1062' 21 23E7 C LD HL, LAST_TCK ; PT TO LAST, DCHK, SYS
1065' 46 C LD B, (HL) ; LOAD LAST TICK OVERFLOW VALUE
C
1066' DB 18 C IN A, (RTC_OFL) ; GET CURRENT TICK OVERFLOW VALUE
1068' 77 C LD (HL), A ; SAVE IT AS LAST
1069' 90 C SUB B ; COMPUTE DIFFERENCE BETWEEN CURRENT TICK AND:
C ; LAST: TICKS OCCURRED.
C
106A' 4F C LD C, A ; SAVE TICKS OCCURRED
C ; IS THE TICK RATE 50 OR 60 HZ?
106B' 23 C INC HL
106C' 7E C LD A, (HL)
106D' 91 C SUB C
106E' 30 01 C JR NC, NOTNOW
C
1070' AF C XOR A
C
1071' 77 C NOTNOW: LD (HL), A
C
1072' 06 3C C LD B, SIXTY_HZ ; ASSUME IT'S SIXTY.
C ; CHECK PIA
1074' DB 03 C IN A, (SYS_CNB) ;
1076' CB 5F C BIT 3, A ; BIT 3 IS 0 IF 60 HZ
1078' CA 107D' C JP Z, XJ072 ; IT'S 60...
C ;
107B' 06 32 C LD B, FIFTY_HZ ; NO, IT'S 50.
C
107D' 79 C XJ072: LD A, C ; RESTORE TICKS OCCURRED
C
C ; C = SECONDS TO ADD
107E' 0E 00 C LD C, 0 ; ZERO SECONDS TO ADD
C
1080' C XJ073: ; SEE IF WE LOST ANY SECONDS: IS TICKS OCCURRED
C ; => TICK RATE?
1080' B8 C CP B ;
1081' DA 1089' C JP C, XJ074 ; NO...
C ;
1084' 90 C SUB B ; YES. REDUCE TICKS OCCURRED BY TICK RATE
1085' 0C C INC C ; AND BUMP SECONDS TO ADD
1086' C3 1080' C JP XJ073 ; LOOP AGAIN TO SEE IF MORE SECONDS LOST
C ;
1089' C XJ074: ; A = TICKS OCCURRED MOD TICK RATE (TICKS TO ADD)
C ; C = SECONDS LOST,
1089' 57 C LD D, A ; PUT TICKS TO ADD IN D
108A' 3A 23E9 C LD A, (SYS_TICKS) ; LOAD LAST SYSTEM TICK COUNT.
108D' 02 C ADD A, D ; ADD TICKS TO ADD
108E' B8 C CP B ; IS SYS TICKS => TICK RATE? (I.E., SECOND OVERFLOW?)
108F' DA 1094' C JP C, XJ075 ; NO...
C ;

```

'INTERRUPT HANDLING ROUTINES'

```

1092' 90      C      SUB    B      ;YES. REDUCE SYSTICKS BY TICK RATE
1093' 0C      C      INC    C      ;AND BUMP SECONDS TO ADD
      C
1094'      C      XJ075:      ;A = SYSTICKS, C = SECONDS TO ADD
1094' 32 23E9 C      LD      (SYS_TICKS),A ;SAVE SYSTEM TICK COUNT
1097' AF      C      XOR      A      ;CLEAR A TO CHECK FOR 0 SECONDS TO ADD
1098' B1      C      OR      C      ;IS C (SECONDS TO ADD) 0?
1099' 28 29   C      JR      Z,XJ076 ;NO. RETURN...
      C      ;
      C      ;YES. GET POINTER TO CLOCK DATA STRUCTURE
      C      ;POINTER POINTS TO SECONDS CELL, THE LAST
      C      ;ENTRY IN THE STRUCTURE.
109B' 2A 2000 C      LD      HL,(CLK_PTR) ;LOAD POINTER TO CLOCK TO HL.
      C      ;HL ==> SECONDS CELL
109E' 7E      C      LD      A,(HL) ;GET SYSTEM SECOND COUNT
109F' 81      C      ADD      A,C ;ADD SECONDS TO SYS SECONDS
10A0' 27      C      DAA      ;BCD ADJUST IT.
10A1' 77      C      LD      (HL),A ;SAVE SYS SECONDS
10A2' D6 60   C      SUB      SEC_MIN ;CHECK FOR SECOND OVERFLOW.
10A4' 38 1E   C      JR      C,XJ076 ;NO OVERFLOW, RETURN
      C      ;
10A6' 77      C      LD      (HL),A ;OVERFLOW. REDUCE SECONDS BY SECONDS/MINUTE
      C      ;AND SAVE
10A7' 2B      C      DEC      HL ;POINT TO MINUTES CELL
10A8' 7E      C      LD      A,(HL) ;LOAD SYSTEM MINUTE COUNT
10A9' 3C      C      INC      A ;BUMP IT.
10AA' 27      C      DAA      ;BCD ADJUST IT
10AB' 77      C      LD      (HL),A ;SAVE IT
10AC' FE 60   C      CP      MIN_HR ;IS IT = MINUTES/HR?
10AE' 20 14   C      JR      NZ,XJ076 ;NO, RETURN
      C      ;
10B0' 36 00   C      LD      (HL),0 ;YES. CLEAR MINUTE
      C
10B2' 2B      C      DEC      HL ;POINT HL TO HOUR CELL
10B3' 7E      C      LD      A,(HL) ;BUMP HOUR
10B4' 3C      C      INC      A ;
10B5' 27      C      DAA      ;
10B6' 77      C      LD      (HL),A ;SAVE IT
10B7' FE 24   C      CP      HR_DAY ;IS IT = HOURS/DAY?
10B9' 20 09   C      JR      NZ,XJ076 ;NO. RETURN
      C      ;
10BB' 36 00   C      LD      (HL),0 ;YES. CLEAR HOUR
      C
10BD' 2B      C      DEC      HL ;POINT TO LOW BYTE OF DAY
10BE' 2B      C      DEC      HL ;
10BF' 34      C      INC      (HL) ;INCREMENT LOW BYTE.
      C      ;IS THERE AN OVERFLOW TO HIGH BYTE?
10C0' 20 02   C      JR      NZ,XJ076 ;NO, RETURN...
      C      ;
10C2' 23      C      INC      HL ;YES. POINT TO HI BYTE
10C3' 34      C      INC      (HL) ;BUMP IT
      C
      C      PAGE

```

```

C
C
C
C      ;-----
C      ; DISK CHANGE DETECT ROUTINE
C      ; BY ROGER W. CHAPMAN & YOUNG MOO KIM
C      ;
C      ; THIS ROUTINE CALLS A ROUTINE IN COMMON MEMORY
C      ; IF A DISK CHANGE IS DETECTED WHICH MODIFIES THE
C      ; APPROPRIATE BIOS & BDOS VARIABLES.
C      ;
C      ; VARIABLES USED :
C      ;     WP_STS BIT 7 = WRITE PROTECT STATUS A DRIVE
C      ;             BIT 6 = WRITE PROTECT STATUS B DRIVE
C      ;
C      ; EXIT PARAMETERS :
C      ;     MEDIA FLAG(SCB) = 0FFH IF CHANGE IN STATUS DETECTED
C      ;     DRIVE A FLAG    = 0FFH IF CHANGE IN A STATUS
C      ;     DRIVE B FLAG    = 0FFH IF CHANGE IN B STATUS
C      ;-----
C
10C4' XJ076:
C
C
C
10C4' DB 08          C           IN   A,(DSK_STS)       ;GET FDC STATUS
10C6' 1F              C           RRA                     ;BUSY ?
10C7' 38 54           C           JR    C,XJ077         ;YES- DO NOT ISSUE FORCE INTERRUPT
C
C
10C9' 3E D0           C           LD    A,D.FINT                ;ISSUE FORCE INTERRUPT COMMAND FOR DISK CHANGE DETECT
10CB' D3 08           C           OUT   (DSK_CMD),A            ;SHOULD WAIT 112T BEFORE READ FDC STATUS
C
10CD' 3A 23E8         C           LD    A,(DCHK_FLG)          ;(13) IS IT TIME TO CHECK FOR CHANGE?
10DE' B7              C           OR    A                    ;(4)
10DF' 20 4A           C           JR    NZ,XJ077              ;(7) NO, NOT YET
C
10E1' 3E 14           C           LD    A,20                  ;(7) WAIT 1/3 SEC FOR NEXT CHECK
10E3' 32 23E8         C           LD    (DCHK_FLG),A          ;(13) SAVE IT
C
10E5' 21 23D2         C           LD    HL,WP_STS             ;(10) TO GET OLD & SAVE NEW STATUS INFO
10E7' 46              C           LD    B,(HL)                 ;(7) GET OLD WRITE PRTECT STATUS
C
10E9' 3A 23D3         C           LD    A,(DRV_MSK)           ;(13) GET DRIVE MASK (LOGICAL TO PHYSICAL)
10EB' 5F              C           LD    E,A                    ;(4)
C
10ED' DB 02           C           IN    A,(SYS_DTB)           ;(11) GET CURRENT SYSTEM PIA STATUS
10EF' 4F              C           LD    C,A                    ;(4) & SAVE IT FOR LATER
C
10F1' E6 F8           C           AND   0F8H                 ;(7) SAVE NON-DRIVE BITS
10F3' B3              C           OR    E                      ;(4) FIRST EXAMINE DRIVE A STATUS
10F5' 5F              C           LD    E,A                    ;(4) SAVE FOR LATER
C
10F7' D3 02           C           OUT   (SYS_DTB),A           ;(11) SELECT DRIVE & MFM
10F9' DB 08           C           IN    A,(DSK_STS)           ;READ DRIVE A STATUS
10FB' 57              C           LD    D,A                    ;& SAVE IT

```

```

C
10EC' 7B      C      LD      A,E      ;NOW EXAMINE DRIVE B STATUS
10ED' EE 06   C      XOR      6
C
10EF' D3 02   C      OUT     (SYS_DTB),A ;SELECT DRIVE & MFM
10F1' DB 08   C      IN      A,(DSK_STS) ;READ DRIVE B STATUS
10F3' 5F      C      LD      E,A      ;& SAVE IT
C
10F4' 79      C      LD      A,C      ;RESTORE SYSTEM PIA
10F5' D3 02   C      OUT     (SYS_DTB),A
C
10F7' 7A      C      LD      A,D      ;GET A STATUS
10F8' 17      C      RLA          ;MOVE BIT 6 TO BIT 7
10F9' E6 80   C      AND     10000000B ;JUST WANT BIT 7
10FB' 57      C      LD      D,A      ;SAVE IT
C
10FC' 7B      C      LD      A,E      ;GET B STATUS
10FD' E6 40   C      AND     01000000B ;JUST WANT BIT 6
10FF' B2      C      OR      D        ;FORM NEW STATUS BYTE
1100' 77      C      LD      (HL),A    ;& SAVE IT
C
1101' A8      C      XOR      B        ;COMPARE WITH OLD STATUS
1102' CA 111D' C      JP      Z,XJ077      ;NO CHANGE DETECTED GO TO RTC_DRV
C
1105' 2A 2002  C      LD      HL,(MEDIA) ;GET ADDRESS OF MEDIA FLAG
1108' 36 FF   C      LD      (HL),0FFH ;& SET IT
C
110A' CB 7F   C      BIT     7,A      ;CHECK DRIVE A STATUS CHANGE
110C' 28 0A   C      JR      Z,B_CHNG ;DRIVE A NOT CHANGED
C
110E' 2A 2004  C      LD      HL,(A_DRIVE) ;GET ADDRESS OF DRIVE A FLAG
1111' 36 FF   C      LD      (HL),0FFH ;& SET IT
C
1113' CB 77   C      BIT     6,A      ;CHECK DRIVE B STATUS CHANGE
1115' CA 111D' C      JP      Z,XJ077      ;NO CHANGE GO TO DONE
C
1118' 2A 2006  C      B_CHNG: LD     HL,(B_DRIVE) ;GET ADDRESS OF DRIVE B FLAG
111B' 36 FF   C      LD      (HL),0FFH ;& SET IT
C
PAGE

```

```
C
C
C
C      ;ROUTINE CHECKS TO SEE IF THE DISK DRIVE MOTOR SHOULD BE TURNED OFF BY UPDATING DACTIVE...ROUTINE ALSO
C
111D' 21 23F2 C XJ077: LD HL, DACTIVE
1120' 7E C LD A, (HL)
1121' B7 C OR A
1122' 28 09 C JR Z, XJ078 ;IF INACTIVE
C
C      ;TURN DRIVE OFF IF DACTIVE = 1
C
1124' 35 C DEC (HL) ;RESET DELAY
1125' 20 06 C JR NZ, XJ078 ;IF DESELECT DRIVE
C
C      ;DESELECT DRIVE
C
1127' DB 02 C IN A, (SYS_DTB)
1129' F6 06 C OR 00000110B ;DESELECT DRIVES
112B' D3 02 C OUT (SYS_DTB), A
C
C      ;READ KEYBOARD
C
112D' 3A 23DD C XJ078: LD A, (KEYLCK)
1130' B7 C OR A
1131' C4 05FD' C CALL NZ, KBDVR ;READ KEYBOARD IF KEYLOCK NOT ACTIVE
C
1134' DB 02 C IN A, (SYS_DTB) ;CLEAR INTERRUPT
1136' C9 C RET
C
C      PAGE
```

'INTERRUPT HANDLING ROUTINES'

```

C
C
C      ; *****
C      ; *                                     *
C      ; *               INTERRUPT TABLE               *
C      ; *                                     *
C      ; *****
C
C      ;-----
C      ; ROM COPY OF INTERRUPT TABLE
C      ;-----
C      ;
C      ; EACH TABLE ENTRY CONSISTS OF A ONE BYTE BANK SWITCH MASK (IN THIS
C      ; CASE, ALL TO SWITCH IN THE ROM) FOLLOWED BY A TWO BYTE INTERRUPT
C      ; HANDLING ROUTINE ADDRESS.
C      ;
C      ; THE SIO ENTRIES ARE IN THE ORDER CORRESPONDING TO THE BIT PATTERN SET
C      ; IN RR2 ON AN INTERRUPT.
C      ;
C      R_INT_TBL:
1137'      C      DB      ROM_BANK      ;SIO TX CHANNEL B
1137'      C      DW      S_T_I_CH_B    ;
1138'      C      DB      ROM_BANK      ;SIO EXTERNAL/STATUS CHANNEL B
1138'      C      DW      EXT_CH_B      ;
1139'      C      DB      ROM_BANK      ;SIO RX CHANNEL B
1139'      C      DW      S_R_I_CH_B    ;
1140'      C      DB      ROM_BANK      ;SIO RX CHANNEL B SPECIAL CONDITION
1140'      C      DW      CH_B_SPEC      ;
1141'      C      DB      ROM_BANK      ;SIO TX CHANNEL A
1141'      C      DW      S_T_I_CH_A    ;
1142'      C      DB      ROM_BANK      ;SIO EXTERNAL/STATUS CHANNEL A
1142'      C      DW      EXT_CH_A      ;
1143'      C      DB      ROM_BANK      ;SIO RX CHANNEL A
1143'      C      DW      S_R_I_CH_A    ;
1144'      C      DB      ROM_BANK      ;SIO RX CHANNEL A SPECIAL CONDITION
1144'      C      DW      CH_A_SPEC      ;
1145'      C      DB      ROM_BANK      ;REAL TIME CLOCK (TICK) INTERRUPT
1145'      C      DW      RTC_INT        ;
1146'      C      DB      ROM_BANK      ;PARALLEL ENTRY
1146'      C      DW      PAR_INT        ;
1147'      C      DB      ROM_BANK      ;INTELLIGENT KEYBOARD
1147'      C      DW      INT_KBD        ;
1148'      C      DB      ROM_BANK      ;DMA ENTRY
1148'      C      DW      DMA_INT        ;
1149'      C      DB      ROM_BANK      ;FLOPPY DISK CONTROLLER
1149'      C      DW      FDC_INT        ;
1150'      C      DB      ROM_BANK      ;SYSTEM RESET
1150'      C      DW      RESET_INT      ;
1151'      C
1151'      C      R_I_TBL_END EQU $
002A      C      R_I_TBL_LEN EQU R_I_TBL_END - R_INT_TBL

```

PAGE

'EXECUTIVE ROM MONITOR'

MACRO-80 3.44 09-Dec-81

PAGE 1-319

'INTERRUPT HANDLING ROUTINES'

;

SUBTTL
PAGE

'BUFFER HANDLING ROUTINES'

```

C          SUBTTL      'BUFFER HANDLING ROUTINES'
C          INCLUDE ROM7                      ;BUFFERING ROUTINES
C          ;
C          ;
C          ; 83-07-18      Fixed the buffer allocation problem in IN_AL_BF and RM_C_BF
C          ;
C          ; *****
C          ; *                                     *
C          ; *               BUFFERING ROUTINES               *
C          ; *                                     *
C          ; *****
C          ;
C          ; MODULE 7 --ROM7-- ENTRY POINTS
C          ;
C          ;          BUFFERING ROUTINES
C          ;
C          ; IN_AL_BF:      INITIALIZE ALL BUFFERS TO NULLS AND PUT ON FREE LIST
C          ; FL_B_ST:       RETURNS BUFFER STATUS (BUFFER FULL OR NOT)
C          ; FL_B_CH:       INSERT CHAR. IN BUFFER
C          ; RM_C_BF:       REMOVE CHAR. FROM BUFFER
C          ; RM_C_ST:       RETURNS BUFFER STATUS (BUFFER EMPTY OR NOT)
C          ; FIL_BUF:       RETURNS ADDRESS OF CURRENT FILL BUFFER
C          ; RMV_BUF:       RETURNS ADDRESS OF CURRENT REMOVE BUFFER
C          ; DEV_ROOT:      RETURNS ADDRESS OF DEVICE LIST ROOT
C          ; BUF_INIT:      INITIALIZES BUFFER HEADER
C          ; ALC_BUF:       ALLOCATE BUFFER IF AVAILABLE
C          ; ALC_ST:        CHECKS BUFFER ALLOCATION STATUS
C          ; D_ALC_BF:      DEALLOCATE BUFFER--RETURN TO FREE LIST
C          ;
C          PAGE

```

1161'

```
C
C
C      .Z80
C      .LIST
C      ;SYSS1.MAC , RAMSHON.MAC AND RAMS10.MAC INCLUDED
C
C      ;      EXT      SPEC_HD
C
C      ;      PUBLIC  IN_AL_BF,FL_B_ST,FL_B_CH,RM_C_BF,RM_C_ST
C
C      CSEG
C
C      PAGE
```

CHARACTER I/O BUFFERING ROUTINES
BY
DAVID ANDREW BERG
10/5/82

THESE ROUTINES IMPLEMENT A SERIAL I/O BUFFERING SCHEME USING DYNAMIC BUFFER ALLOCATION. IN PRACTICE, EACH DEVICE WILL HAVE A LIST OF BUFFERS (WHICH MAY BE EMPTY) ASSOCIATED WITH IT. CHARACTERS MAY BE ADDED TO THE END OF THE BUFFER LIST AND REMOVED FROM THE BEGINNING. MORE BUFFERS ARE DYNAMICALLY ALLOCATED UPON DEMAND. BUFFERS ARE REMOVED FROM THE LIST WHEN ALL CHARACTERS HAVE BEEN REMOVED FROM THEM.

ROUTINES ARE ALSO IMPLEMENTED TO OBTAIN THE STATUS OF A DEVICE: WHETHER THERE ARE CHARACTERS WAITING TO BE REMOVED AND IF THERE IS ROOM IN THE BUFFER FOR MORE CHARACTERS.

IN PRACTICE IF INTERRUPT HANDLERS ALLOCATE BUFFERS, THEN STATUS CHECKS WILL BE VALID ONLY IF INTERRUPTS ARE DISABLED.

11/5/82: DAB:

A. CHANGED FL_B_CH TO INTERPRET BIT 6 OF THE DEVICE NUMBER AS A FLAG INDICATING THAT THE BUFFER FILL NEEDS SPECIAL HANDLING. THIS IS IMPLEMENTED PRIMARILY FOR HANDLING SIO TRANSMISSION. IF THE SIO INTERRUPTS WITH A TX BUFFER EMPTY CONDITION AND THERE ARE NO CHARS READY TO TRANSMIT, THE TX BUFFER EMPTY INTERRUPT MUST BE DISABLED, OTHERWISE IT WILL CONTINUOUSLY INTERRUPT AND LOCK THE PROCESSOR INTO THE INTERRUPT HANDLERS.

ON THE OTHER HAND, IF THIS INTERRUPT IS DISABLED, IT WILL NEVER INTERRUPT TO REMOVE CHARS FROM THE BUFFER WHEN THERE ARE SOME READY. THUS, THE INTERRUPT HAS TO BE RE-ENABLED WHEN A CHARACTER IS PUT INTO THE BUFFER.

THIS FLAG BIT IMPLEMENTATION ALLOWS THIS: WHEN BIOS CALLS THE BUFFERING ROUTINE TO BUFFER A CHAR, IT WILL SET THE FLAG. THE SPECIAL HANDLING ROUTINE WILL THEN CHECK TO SEE IF INTERRUPTS HAVE BEEN DISABLED AND IF SO WILL RE-ENABLE THEM.

B. FIX BUG IN IN_AL_BF. IT CLEARED ONE TOO MANY BYTES OF BUFFER SPACE.

11/4/82: DAB:

A. CHANGED RM_C_BF TO ALWAYS RETURN A CHARACTER. IT MUST NOT BE CALLED UNLESS THERE IS A CHARACTER READY. IN OTHER WORDS, RM_C_BF MUST NOT BE CALLED UNLESS RM_C_ST RETURNS POSITIVE STATUS.

B. FL_B_ST AND FL_B_CH ARE CHANGED SO THAT THE HIGH BIT OF THE DEVICE NUMBER PASSED IN B INDICATES THE PRIORITY OF THE

```
C ;      DEVICE. IF THE BIT IS SET THE DEVICE HAS HIGH PRIORITY AND
C ;      CAN ALLOCATE BUFFERS BELOW THE SPECIFIED THRESHOLD (SEE BELOW).
C ;      IF THE BIT IS CLEAR, THE DEVICE HAS LOW PRIORITY AND CAN NOT
C ;      ALLOCATE BUFFERS BELOW THE THRESHOLD. THIS IS DONE TO REDUCE
C ;      THE PROBABILITY OF AN INFORMATION LOSS CAUSED BY AN
C ;      INTERRUPTING INPUT DEVICE NOT HAVING A BUFFER IN WHICH TO
C ;      STORE A CHARACTER. INTERRUPTING INPUT DEVICES ARE ASSIGNED
C ;      HIGH PRIORITY AND ALL OTHERS ARE ASSIGNED LOW PRIORITY.
C ;
C ;      THE THRESHOLD LEVEL IS THE MINIMUM NUMBER OF BUFFERS THAT
C ;      SHOULD BE AVAILABLE FOR HIGH PRIORITY DEVICES ONLY. IT IS AN
C ;      ASSEMBLY TIME PARAMETER CHECKED BY THE BUFFER ALLOCATION AND
C ;      BUFFER ALLOCATION STATUS ROUTINES.
C ;
C ;-----
C PAGE
```

```
C
C ;-----
C ;
C ;   FUNCTIONS:
C ;
C ;   FILL DEVICE BUFFER WITH CHARACTER
C ;   REMOVE CHARACTER FROM DEVICE BUFFER
C ;   RETURN FILL STATUS OF DEVICE BUFFER (ROOM FOR MORE CHARS?)
C ;   RETURN REMOVE STATUS OF DEVICE BUFFER (CHARS TO REMOVE?)
C ;   RETURN NUMBER OF BYTES AVAILABLE IN DEVICE BUFFER
C ;   RETURN BUFFER ALLOCATION STATUS (ARE THERE BUFFERS AVAILABLE TO
C ;       ALLOCATE)
C ;   ALLOCATE BUFFER
C ;   DE-ALLOCATE BUFFER
C ;   INITIALIZE BUFFERS
C ;-----
C PAGE
```

[illegible]

```
C
C
C      ;-----+
C      ; BUFFER:
C      ;
C      ; A BUFFER HAS ONE OF TWO FORMATS DEPENDING ON WHETHER IT IS ALLOCATED
C      ; OR NOT.
C      ;
C      ; AN ALLOCATED BUFFER HAS THE FOLLOWING FORMAT:
C      ;
C      ; +-----+
C      ; | POINTER TO |
C      ; | NEXT REMOVE |
C      ; | BUFFER      |
C      ; +-----+
C      ; | BUFFER FILL |
C      ; | INDEX      |
C      ; +-----+
C      ; | BUFFER REMOVE|
C      ; | INDEX      |
C      ; +-----+
C      ; | BUFFER DATA |
C      ; | AREA        |
C      ; +-----+
C      ;
C      ; THE FIRST ENTRY POINTS TO THE NEXT REMOVE BUFFER (FROM WHICH TO
C      ; REMOVE CHARACTERS) IN THE LIST. IT IS NIL IF THERE IS NO NEXT.
C      ;
C      ; THE SECOND ENTRY IS AN INDEX TO THE NEXT LOCATION IN THE BUFFER IN
C      ; WHICH TO FILL IN NEW CHARS (I.E., WHERE TO PUT CHARS IN THE BUFFER)
C      ;
C      ; THE THIRD ENTRY IS AN INDEX TO THE NEXT LOCATION FROM WHICH TO REMOVE
C      ; CHARS (I.E., WHERE TO TAKE CHARS FROM BUFFER).
C      ;
C      ; AN UNALLOCATED BUFFER (I.E., ONE IN THE FREE LIST) HAS THE FOLLOWING
C      ; FORMAT:
C      ;
C      ; +-----+
C      ; | POINTER TO |
C      ; | NEXT FREE  |
C      ; | BUFFER     |
C      ; +-----+
C      ; | UNDEFINED |
C      ; |           |
C      ; +-----+
C      ; | UNDEFINED |
C      ; |           |
C      ; +-----+
C      ; | UNDEFINED |
C      ; |           |
C      ; +-----+
C      ;
C      ; THE FIRST ENTRY IS A POINTER TO THE NEXT FREE BUFFER, IF ANY. IT
C      ; IS UNDEFINED IF THERE IS NO NEXT FREE BUFFER. THE NUMBER OF FREE
C      ; BUFFERS IS TRACKED BY THE COUNT IN THE FREE LIST ROOT.
```



```
C ;  
C ; ALL OTHER ENTRIES IN THE BUFFER ARE UNDEFINED IN A FREE BUFFER.  
C ;-----  
C PAGE
```

```
C
C      ;-----
C      ;
C      ;   FREE LIST ROOT:
C      ;
C      ;   THE FREE LIST ROOT IS A DATA STRUCTURE CONTAINING TWO ELEMENTS:
C      ;
C      ;   +-----+
C      ;   | POINTER TO FIRST |
C      ;   |   FREE BUFFER   |
C      ;   +-----+
C      ;   | NUMBER OF FREE |
C      ;   |   BUFFERS      |
C      ;   +-----+
C      ;
C      ;   THE FIRST ENTRY IS A POINTER TO THE FIRST FREE BUFFER. IF THERE ARE
C      ;   MORE THAN ONE BUFFER, EACH FREE BUFFER POINTS TO NEXT. SEE BUFFER
C      ;   DESCRIPTION ABOVE.
C      ;
C      ;   THE SECOND ENTRY IS A COUNT OF THE NUMBER OF BUFFERS IN THE LIST.
C      ;-----
C
C
C      PAGE
```

```

C
C
1161' C      IN_AL_BF:
C      ;      INITIALIZES ALL BUFFERS TO NULLS AND PUTS ALL ON FREE LIST.
C      ;
C      ; ENTRY:
C      ;      NONE.
C      ;
C      ; EXIT:
C      ;      ALL BUFFERS ARE INITIALIZED AND PUT ON FREE LIST
C      ;
C      ; REGISTERS AFFECTED:
C      ;
C      ;
0014' C      GROSS_BF      EQU      MX_CH_BF + BF_OV_HD      ;GROSS BUFFER SIZE (DATA AREA + OVERHEAD)
241D  C      KEY_RT      EQU      D_RT_TBL+KBD_DEV*D_RT_LEN  ;START OF KEYBOARD ROOT TABLE
243E  C      KEY_BF      EQU      BUFF_ST+KBD_DEV*GROSS_BF    ;INITIAL DEDICATED KEYBOARD BUF
C
C      ;      7/22/AOS
C      ;      ASSIGN THE FIRST BUFFER TO THE KEYBOARD
C      ;      (I.E. INIT THE KEYBOARD DEVICE ROOT TO POINT TO A BUFFER)
C      ;
1161' 21 241D C      LD      HL,KEY_RT      ;GET PNTR TO TOP OF KEYBOARD ROOT TABLE TO HL
1164' 11 243E C      LD      DE,KEY_BF      ; AND PNTR TO DEDICATED KEYBOARD BUF IN DE
1167' 73 C      LD      (HL),E      ;INIT PNTR TO FILL BUF
1168' 23 C      INC      HL          ; IN KEYBOARD ROOT TABLE
1169' 72 C      LD      (HL),D
116A' 23 C      INC      HL
116B' 73 C      LD      (HL),E      ;INIT PNTR TO REMOVE BUF
116C' 23 C      INC      HL          ; IN KEYBOARD ROOT TABLE
116D' 72 C      LD      (HL),D
116E' 23 C      INC      HL          ;BUMP PNTR TO INIT THE # OF CHARS.
C      ;      IN KEYBOARD BUFS AND THE REST
C      ;      OF THE ROOT TABLE
C
C      ;
C      ;      INITIALIZE THE REST OF THE DEVICE ROOTS
C      ;
116F' 54 C      LD      D,H          ;POINT DE TO 1 BEFORE HL
1170' 5D C      LD      E,L          ; AND INIT THE COUNT FOR LDIR
1171' 13 C      INC      DE
1172' 01 0019 C      LD      BC,D_RT_LEN*(NO_DEVS-1)+1
1175' 36 00 C      LD      (HL),0
1177' ED B0 C      LDIR
C
C      ;COMPUTE SIZE OF AREA TO CLEAR:
1179' 21 0000 C      LD      HL,0          ;ZERO HL
117C' 11 0014 C      LD      DE,GROSS_BF    ;LOAD DE WITH GROSS BUFFER SIZE
117F' 06 28 C      LD      B,NO_BUFS ;LOAD B WITH NO. OF BUFFERS
1181' 19 C      XJ231: ADD      HL,DE      ;MULTIPLY GROSS BUFFER SIZE TIMES NO_BUFS
C      ;BY ADDING
1182' 10 FD C      DJNZ      XJ231      ;TILL B = 0
C
1184' 44 C      LD      B,H          ;MOVE TOTAL TO BC FOR LDIR
1185' 4D C      LD      C,L
1186' 0B C      DEC      BC          ;REDUCE COUNT BY ONE SINCE FIRST CELL GETS

```

'BUFFER HANDLING ROUTINES'

```

C          ;CLEARED DIRECTLY                                11/5B
1187' 21 243E C LD HL,BUFF_ST ;POINT HL TO FIRST BUFFER START
118A' 11 243F C LD DE,BUFF_ST+1 ;AND HL ONE PAST
118D' 36 00 C LD (HL),0 ;CLEAR FIRST CELL OF FIRST BUFFER
118F' ED B0 C LDIR ;AND ALL THE REST
C
C          ;POINT EACH BUFFER TO THE NEXT:
1191' 11 2452 C LD DE,BUFF_ST+GROSS_BF ;POINT DE AT FIRST BUFFER 7/22/AOS
1194' 21 0014 C LD HL,GROSS_BF ;LOAD HL WITH GROSS BUFFER SIZE TO
1197' 19 C ADD HL,DE ;COMPUTE ADDRESS OF NEXT BUFFER
1198' 06 26 C LD B,NO_BUFS-2 ;LOAD B WITH NO. BUFFERS - 2 7/22/AOS
C
119A' C XJ232:
119A' EB C EX DE,HL ;NOW HL == 'CURRENT' BUFFER, DE == NEXT
C ;BUFFER
119B' 73 C LD (HL),E ;STORE PTR TO NEXT IN CURRENT
119C' 23 C INC HL ;
119D' 72 C LD (HL),D ;
C
119E' 21 0014 C LD HL,GROSS_BF ;LOAD HL WITH INCREMENT TO NEXT BUFFER
11A1' 19 C ADD HL,DE ;COMPUTE ADDR OF NEXT, DE NOW = CURRENT BUFFER
C ;HL = NEXT
11A2' 10 F6 C DJNZ XJ232 ;LOOP TILL B = 0: ALL BUFFERS POINT TO NEXT
C ;EXCEPT THE LAST.
C ;
11A4' EB C EX DE,HL ;NOW HL == CURRENT
11A5' 36 00 C LD (HL),0 ;CLEAR PTR (SET TO NIL)
11A7' 23 C INC HL ;
11A8' 36 00 C LD (HL),0 ;
C
11AA' 21 2452 C LD HL,BUFF_ST+GROSS_BF ;POINT FREE LIST ROOT TO 7/22/AOS
11AD' 22 243B C LD (FRE_ADDR),HL ; FIRST FIRST BUFFER
11B0' 3E 27 C LD A,NO_BUFS-1 ;INIT FREE BUFFER COUNT CELL
11B2' 32 243D C LD (FRE_ADDR+2),A ; TO # OF BUFS - 1 (DEDICATED KEYBOARD BUF)
C
11B5' C9 C RET ;AND RETURN TO CALLER
C
C PAGE

```

```

C
C
11B6' C FL_B_ST:
C
C ; RETURNS FILL STATUS FOR DEVICE INDICATED BY B. Z IS SET IF THERE
C ; IS NO ROOM FOR ANOTHER CHARACTER.
C ;
C ; THE HIGH BIT OF B IS SET IF THE DEVICE HAS HIGH PRIORITY.
C ;
C ; IF THERE IS NO FILL BUFFER FOR DEVICE B, BUFFER ALLOCATION STATUS
C ; IS RETURNED.
C ;
C ; OTHERWISE IF THE FILL INDEX FOR THE DEVICE'S FILL BUFFER IS NOT
C ; EQUAL TO THE MAXIMUM CHARS IN A BUFFER, 'GOOD' STATUS IS RETURNED.
C ;
C ; OTHERWISE BUFFER ALLOCATION STATUS IS RETURNED (THERE IS NO ROOM IN
C ; THE CURRENT BUFFER).
C ;
C ; ENTRY:
C ; B = NUMBER OF DEVICE FOR WHICH STATUS IS DESIRED.
C ; THE HIGH BIT OF B IS SET IF THE DEVICE HAS HIGH PRIORITY.
C ; HI BIT IS SET IF THE DEVICE IS HIGH PRIORITY.
C ; HI BIT IS SET IF THE DEVICE IS HIGH PRIORITY.
C ;
C ; EXIT:
C ; Z BIT IS SET IF THERE NO ROOM FOR ANOTHER CHARACTER ('BAD')
C ; Z BIT IS CLEAR IF THERE IS ROOM.
C ;
C ; REGISTERS AFFECTED:
C ;
C
11B6' CD 1288' C CALL FIL_BUF ;GET ADDRESS OF DEVICE'S FILL BUFFER TO HL
C ;RETURNS Z SET IF THERE IS NO BUFFER
C ;(BC) SHOULD BE SAVED
11B9' 28 06 C JR Z,XJ233 ;NO FILL BUFFER...
C ;
C ; DEVICE HAS A FILL BUFFER. IS THERE ROOM IN IT FOR ANOTHER CHARACTER?
C ;
11B8' 23 C INC HL ;BUMP PTR TO FILL INDEX ENTRY IN FILL BUFFER
11BC' 23 C INC HL ;
11BD' 7E C LD A,(HL) ;GET FILL INDEX TO A
11BE' FE 10 C CP MX_CH_BF ;COMPARE IT WITH MAX CHARS IN BUFFER
11C0' C0 C RET NZ ;RETURN IF NOT ZERO WITH Z CLEAR TO INDICATE
C ;THAT THERE IS ROOM IN BUFFER.
C ;
11C1' C XJ233: ;EITHER THIS DEVICE HAS NO FILL BUFFER OR
C ;THERE IS NO ROOM IN THE BUFFER: RETURN
C ;ALLOCATION STATUS.
11C1' CD 12D1' C CALL ALC_ST ;GET ALLOCATION STATUS: Z IS SET IFF THERE ARE
C ;NO BUFFERS.
11C4' C9 C RET ;RETURN...
C
C PAGE

```

```

C
C
11C5' C FL_B_CH:
C ; ROUTINE TO FILL_BUFFERWITH_CHAR
C ; ENTRY:
C ; B = DEVICE NUMBER
C ; HI BIT IS SET IF THE DEVICE IS HIGH PRIORITY.
C ; BIT 6 IS SET IF SPECIAL HANDLING IS REQUIRED.
C ;
C ; C = CHARACTER WITH WHICH TO FILL DEVICE
C ; EXIT:
C ; DEVICE BUFFER IS FILLED WITH CHAR
C ; Z BIT SET IF FILL SUCCESSFUL
C ; Z CLEAR IF UNSUCCESSFUL
C ;
C ; REGISTERS AFFECTED:
C ; A
C ; BC
C ; DE
C ; HL
C ;
C
11C5' CD 1288' C CALL FIL_BUF ;GET ADDRESS OF FILL BUFFER TO HL
C ;RETURNS Z SET IF NO BUFFER
C ;(BC) SHOULD BE SAVED
11C8' 20 19 C JR NZ,XJ234 ;NO...
C ;
11CA' C5 C PUSH BC
11CB' CD 1287' C CALL ALC_BUF ;YES. THERE IS NO FILL BUFFER, SO ALLOCATE
C ;BUFFER. ADDRESS OF BUFFER IN HL. Z BIT SET
C ;ON FAILURE
11CE' C1 C POP BC
11CF' 28 52 C JR Z,XJ235 ;JUMP TO EXIT PT.
C ;
11D1' E5 C PUSH HL
11D2' CD 129E' C CALL DEV_ROOT ;GET ADDRESS OF DEVICE LIST ROOT TO HL
C ;(BC) SAVED
11D5' D1 C POP DE ;RESTORE NEW BUFFER ADDR TO HL
11D6' 73 C LD (HL),E ;SAVE NEW BUFFER ADDR IN FILL BUFFER ADDR ENTRY
11D7' 23 C INC HL ;OF DEVICE LIST ROOT
11D8' 72 C LD (HL),D ;
11D9' 23 C INC HL ;AND REMOVE BUFFER ENTRY
11DA' 73 C LD (HL),E ;
11DB' 23 C INC HL ;
11DC' 72 C LD (HL),D ;
C ;
11DD' EB C EX DE,HL
11DE' E5 C PUSH HL
11DF' CD 12AD' C CALL BUF_INIT ;INITIALIZE BUFFER
11E2' E1 C POP HL ;MOVE NEW BUFFER ADDRESS TO HL
C ;
11E3' C XJ234: ;HL NOW POINTS TO A BUFFER TO FILL WITH CHAR
11E3' 23 C INC HL ;BUMP HL TO POINT TO FILL INDEX ENTRY
11E4' 23 C INC HL
11E5' 7E C LD A,(HL) ;GET FILL INDEX ENTRY

```

```

11E6' 2B      C      DEC    HL
11E7' 2B      C      DEC    HL
11E8' FE 10   C      CP     MX_CH_BF      ;IS IT EQUAL TO MAX CHARS IN BUFFER?
11EA' 20 26   C      JR     NZ,XJ236      ;NO...
                                   ;
11EC' C5      C      PUSH   BC      ;YES...NO ROOM IN THIS BUFFER, ALLOCATE NEW
11ED' E5      C      PUSH   HL      ;ONE. SAVE DEVICE NO, CHAR TO SAVE AND ADDR
                                   ;OF CURRENT BUFFER
11EE' CD 12B7' C      CALL   ALC_BUF      ;ALLOCATE A BUFFER. ADDR IN HL, Z SET IF FAIL
11F1' D1      C      POP     DE      ;RESTORE ADDR OF CURRENT BUFFER TO DE
11F2' C1      C      POP     BC      ;RESTORE DEV NO AND CHAR
11F3' 20 2E   C      JR     Z,XJ235      ;RETURN IF NO BUFFER AVAILABLE TO ALLOCATE
                                   ;
11F5' EB      C      EX      DE,HL      ;SWAP CURRENT BUFFER AND NEW BUFFER ADDR
                                   ;SO CURR BUFF ADDR IS IN HL FOR INDEXING
11F6' 73      C      LD      (HL),E      ;SET POINTER TO NEXT REMOVE BUFFER IN
                                   ;CURRENT BUFFER HEADER TO POINT TO NEWLY
                                   ;ALLOCATED BUFFER.
11F7' 23      C      INC     HL      ;DITTO
11F8' 72      C      LD      (HL),D      ;DITTO
                                   ;
11F9' D5      C      PUSH   DE      ;SAVE NEW BUFF ADDR
11FA' CD 129E' C      CALL   DEV_ROOT      ;GET LIST ROOT FOR CURRENT DEVICE TO HL
11FD' D1      C      POP     DE      ;RESTORE NEW BUFF ADDR
11FE' 73      C      LD      (HL),E      ;SET POINTER TO FILL BUFFER IN LIST ROOT TO
                                   ;POINT TO NEWLY ALLOCATED BUFFER
11FF' 23      C      INC     HL      ;
1200' 72      C      LD      (HL),D      ;...HI BYTE...
1201' 23      C      INC     HL      ;
1202' 7E      C      LD      A,(HL)      ;IS CURRENT REMOVE BUFFER NIL?
1203' 23      C      INC     HL      ;
1204' B6      C      OR      (HL)      ;
1205' 20 03   C      JR     NZ,XJ237      ;NO...
                                   ;
1207' 72      C      LD      (HL),D      ;SAVE NEXT BUFFER ADDR AS REMOVE BUFFER
1208' 2B      C      DEC     HL      ;
1209' 73      C      LD      (HL),E      ;
                                   ;
120A' EB      C      XJ237: EX     DE,HL      ;PUT ADDR OF NEW BUFFER IN HL
                                   ;
120B' E5      C      PUSH   HL      ;SAVE ADDR OF NEW BUFFER
120C' CD 12AD' C      CALL   BUF_INIT      ;INITIALIZE NEW BUFFER, (BC,DE) SAVED
120F' E1      C      POP     HL      ;RESTORE ADDR OF NEW BUFFER
1210' 3E 00   C      LD      A,0      ;LOAD A WITH VALUE OF FILL INDEX
                                   ;
1212'         C      XJ236: ;HL NOW POINTS TO FILL BUFFER
                                   ;A = FILL INDEX
1212' 23      C      INC     HL      ;BUMP HL TO POINT TO BEGINNING OF DATA AREA
1213' 23      C      INC     HL      ;PAST POINTER TO NEXT REMOVE BUFFER
1214' 34      C      INC     (HL)      ;BUMP FILL INDEX
1215' 23      C      INC     HL      ;PAST FILL INDEX
1216' 23      C      INC     HL      ;PAST REMOVE INDEX
                                   ;
1217' 5F      C      LD      E,A      ;MOVE FILL INDEX TO DE TO INDEX INTO BUFFER
1218' 16 00   C      LD      D,0      ;ZERO D FOR DOUBLE ADD
121A' 19      C      ADD     HL,DE      ;INDEX INTO BUFFER DATA AREA BY FILL INDEX

```

'EXECUTIVE ROM MONITOR'
'BUFFER HANDLING ROUTINES'

MACRO-80 3.44 09-Dec-81

PAGE 1-334

```

121B' 71      C      LD      (HL),C      ;SAVE CHAR IN BUFFER AREA AT FILL INDEXTH BYTE
C
C      ;
121C' CB 70    C      BIT      6,B
121E' C4 103E' C      CALL    NZ,SPEC_HD      ;IF CARRY SET (I.E., BIT 6 OF DEVICE #), CALL 11/5A
C      ;SPECIAL HANDLING ROUTINE.      11/5A
C      ;
1221' AF      C      XOR      A      ;SET Z BIT TO INDICATE SUCCESS.
1222' C9      C      RET      ;AND RETURN
C      ;
1223'         C      XJ235: ;A NEW BUFFER COULD NOT BE ALLOCATED
1223' AF      C      XOR      A      ;CLEAR A
1224' 3D      C      DEC      A      ;AND DECREMENT TO FORCE Z CLEAR
1225' C9      C      RET      ;AND RETURN.
C
C      PAGE

```



```

C
C
1226' C RM_C_BF:
C ; ROUTINE TO REMOVE_CHARACTERFROM_BUFFER
C ;
C ; THERE MUST BE A CHARACTER READY TO RETURN, OTHERWISE THE RESULTS 11/4A
C ; ARE UNPREDICTABLE. 11/4A
C ;
C ; ENTRY:
C ; B = NUMBER OF DEVICE FROM WHICH CHARACTER IS TO BE TAKEN
C ; EXIT:
C ; A = CHARACTER FROM DEVICE IN REGISTER B
C ;
C ; REGISTERS AFFECTED:
C ;
C ; A
C ; BC
C ; DE
C ; HL
C ;
C
1226' CD 1292' C CALL RMV_BUF ;GET ADDR OF REMOVE BUFFER TO
C ;HL
1229' 23 C INC HL ;BUMP HL TO POINT TO REMOVE INDEX OF BUFFER
122A' 23 C INC HL ;PAST PTR TO NEXT REMOVE BUFFER...
122B' 23 C INC HL ;PAST FILL INDEX
122C' 5E C LD E, (HL) ;GET REMOVE INDEX
122D' 23 C INC HL ;BUMP PTR TO DATA AREA
122E' E5 C PUSH HL ;SAVE POINTER TO DATA AREA
122F' 16 00 C LD D, 0 ;ZERO HI BYTE OF DE. DE CONTAINS INDEX TO
C ;CHAR TO REMOVE IN BUFFER
1231' 19 C ADD HL, DE ;ADD INDEX TO BASE. HL POINTS TO CHAR TO
C ;REMOVE
1232' 7E C LD A, (HL) ;GET CHAR TO A
1233' E1 C POP HL ;RESTORE PTR TO DATA AREA
1234' F5 C PUSH AF ;SAVE CHAR
C ;
1235' 78 C LD A, E ;MOVE REMOVE INDEX TO A
1236' FE 0F C CP MX_CH_BF-1 ;IS LAST CHAR IN BUFFER?
1238' 28 04 C JR Z, XJ238 ;YES, DE-ALLOCATE THIS BUFFER AND SET POINTER
C ;TO POINT NEXT BUFFER
C ;
123A' F1 C POP AF ;RESTORE CHARACTER TO RETURN TO A
123B' 2B C DEC HL ;POINT HL TO REMOVE INDEX ENTRY
123C' 34 C INC (HL) ;INCREMENT REMOVE INDEX ENTRY
123D' C9 C RET ;AND RETURN TO CALLER
C
123E' C XJ238:
C ; 7/22/AOS
C ; THE LAST CHAR WAS REMOVED FROM THE BUFFER SO--
C ; DEALLOCATE THE BUFFER UNLESS IT IS THE DEDICATED
C ; KEYBOARD BUFFER
C
C ;THE CHAR TO RETURN IS ON THE STACK, B= DEVICE #, HL= BEGINNING OF DATA AREA
123E' 2B C DEC HL ;RESTORE HL TO POINT TO BEGINNING

```

'BUFFER HANDLING ROUTINES'

```

123F' 2B      C      DEC  HL      ; OF BUF TO DEALLOCATE
1240' 2B      C      DEC  HL
1241' 2B      C      DEC  HL
1242' E5      C      PUSH HL      ;SAVE PNTR TO BUF TO DEALLOCATE
                                C
1243' CD 129E' C      CALL DEV_ROOT ;GET ADDR OF DEVICE LIST ROOT TO HL
1246' D1      C      POP  DE      ;DE=PNTR TO BUF TO DEALLOCATE
                                C      ;HL=PNTR TO DEVICE LIST ROOT
1247' E5      C      PUSH HL      ;SAVE PNTR TO DEVICE LIST ROOT
1248' 7B      C      LD    A,E
1249' BE      C      CP    (HL)    ;COMPARE PNTR TO BUF TO DEALLOCATE
124A' 20 17   C      JR    NZ,GT_1_BUF ; WITH PNTR TO DEVICE FILL BUF
124C' 7A      C      LD    A,D      ; TO SEE IF ONLY 1 BUFFER IS
124D' 23      C      INC  HL      ; ALLOCATED TO THIS DEVICE
124E' BE      C      CP    (HL)
124F' 20 12   C      JR    NZ,GT_1_BUF
                                C
1251' 3E 00   C      LD    A,KBD_DEV ;ONLY 1 BUFFER FOR THIS DEVICE--
1253' B8      C      CP    B        ; IS THE DEVICE THE KEYBOARD?
1254' C2 125E' C      JP    NZ,NOT_KEYBRD ; NO..SO JUMP
                                C
1257' EB      C      EX    DE,HL    ;KEYBOARD ONLY HAS ONE BUFFER LEFT--
1258' CD 12AD' C      CALL BUF_INIT ; DO NOT DEALLOCATE! -- INIT BUF
125B' E1      C      POP  HL      ;REMOVE PNTR TO DEVICE LIST ROOT FROM STACK
125C' F1      C      POP  AF      ;RESTORE CHAR TO RETURN
125D' C9      C      RET          ;RETURN TO CALLER
                                C
125E'          C      NOT_KEYBRD: ;DE=PNTR TO BUF TO DEALLOCATE
                                C      ;HL=2ND BYTE OF DEVICE LIST ROOT
125E' 36 00   C      LD    (HL),0    ;CLEAR PNTR TO FILL BUF IN DEVICE LIST
1260' 2B      C      DEC  HL      ; ROOT -- LAST BUF IS BEING DEALLOCATED
1261' 36 00   C      LD    (HL),0
                                C
1263'          C      GT_1_BUF: ;DE=PNTR TO BUF TO DEALLOCATE
1263' E1      C      POP  HL      ;HL=PNTR TO DEVICE LIST ROOT
1264' 23      C      INC  HL      ;BUMP SO HL POINTS TO LIST ROOT
1265' 23      C      INC  HL      ; REMOVE PNTR
1266' EB      C      EX    DE,HL
1267' 01 0002 C      LD    BC,2      ;UPDATE REMOVE PNTR IN DEVICE LIST ROOT
126A' ED B0   C      LDIR          ; TO POINT TO NEXT REMOVE BUFFER
                                C      ;HL=PNTR TO FILL INDEX OF BUF TO DEALLOCATE
126C' 2B      C      DEC  HL      ;RETURN HL TO TOP OF BUF
126D' 2B      C      DEC  HL      ; TO DEALLOCATE
126E' CD 12E5' C      CALL D_ALC_BUF ;DEALLOCATE THE BUFFER
1271' F1      C      POP  AF      ;RESTORE CHAR TO RETURN
1272' C9      C      RET          ;RETURN TO CALLER
                                C
                                C      PAGE

```

```

C
C
1273' C RM_C_ST:
C ; RETURNS REMOVE CHARACTER STATUS (ARE THERE CHARS TO REMOVE?) FOR
C ; DEVICE IN B.
C ;
C ; IF THERE IS NO REMOVE BUFFER FOR DEVICE, RETURNS FALSE
C ;
C ; IF THERE IS MORE THAN ONE REMOVE BUFFER (I.E., IF THE FIRST POINTS
C ; TO ANOTHER), THEN THERE ARE CHARS TO REMOVE.
C ;
C ; OTHERWISE IF THE REMOVE INDEX OF THE BUFFER IS GREATER THAN OR
C ; EQUAL TO THE FILL INDEX, THERE ARE NO CHARS IN BUFFER, SO RETURN FALSE,
C ; ELSE RETURN TRUE.
C ;
C ; ENTRY:
C ; B = NUMBER OF DEVICE FOR WHICH REMOVE STATUS IS DESIRED
C ;
C ; EXIT:
C ; Z BIT IS CLEAR IF THERE ARE CHARS TO REMOVE
C ;
C ; Z BIT IS SET IF THERE ARE NO CHARS TO REMOVE
C ;
C ; REGISTERS AFFECTED:
C ;
C ;
1273' CD 1292' C CALL RMV_BUF ;GET ADDRESS OF REMOVE BUFFER FOR DEVICE IN B
C ;TO HL. Z IS SET IF ADDRESS IS NIL (0).
1276' C8 C RET Z ;RETURN IF Z SET: NO FILL BUFFER. Z SET
C ;MEANS NO CHARS TO REMOVE.
C ;
1277' 5E C LD E, (HL) ;LOAD LO BYTE OF PTR TO NEXT REMOVE BUFFER TO E
1278' 23 C INC HL ;
1279' 7E C LD A, (HL) ;LOAD HI BYTE TO A.
127A' B3 C OR E ;OR HI & LO BYTES TO CHECK FOR ZERO
127B' C0 C RET NZ ;RETURN IF NOT ZERO WITH Z BIT CLEAR
C ;TO INDICATE CHARS WAITING
C ;
127C' 23 C INC HL ;BUMP PTR TO POINT TO FILL INDEX
127D' 5E C LD E, (HL) ;GET FILL INDEX TO E
127E' 23 C INC HL ;BUMP PTR TO REMOVE INDEX
127F' 7E C LD A, (HL) ;GET REMOVE INDEX TO A
1280' BB C CP E ;TEST FOR REMOVE INDEX < FILL INDEX:
1281' 38 02 C JR C, XJ241 ;YES... CHARS WTG
C ;
1283' AF C XOR A ;REMOVE INDEX => FILL INDEX: NO CHARS WAITING.
C ;SET Z
1284' C9 C RET ;AND RETURN
C ;
1285' C XJ241: ;CHARS WAITING...
1285' AF C XOR A ;SET Z BIT
1286' 3D C DEC A ;FORCE Z BIT CLEAR
1287' C9 C RET ;AND RETURN
C

```

'EXECUTIVE ROM MONITOR'

MACRO-80 3.44 09-Dec-81

PAGE 1-338

'BUFFER HANDLING ROUTINES'

C

PAGE


```

C
C
1292' C RMV_BUF:
C ; RETURNS ADDRESS OF CURRENT REMOVE BUFFER FOR DEVICE SELECTED BY
C ; REGISTER B. Z BIT IS SET IF THERE IS NONE.
C ;
C ; RETURNS VALUE LOCATED IN SECOND WORD OF DEVICE LIST ROOT. THIS VALUE
C ; IS ZERO IF THERE IS NO FILL BUFFER.
C ;
C ; ENTRY:
C ; B = DEVICE NUMBER FOR WHICH CURRENT REMOVE BUFFER ADDRESS IS DESIRED
C ; EXIT:
C ; Z BIT SET IF ADDRESS IS ZERO.
C ; HL = ADDRESS OF CURRENT REMOVE BUFFER, IF ANY, ELSE 0.
C ;
C ; REGISTERS AFFECTED:
C ; A
C ; E
C ; HL
C ; Z BIT
C
1292' CD 129E' C CALL DEV_ROOT ;GET ADDRESS OF DEVICE LIST ROOT TO HL
1295' 23 C INC HL ;BUMP TO POINT TO REMOVE BUFFER ADDR
1296' 23 C INC HL ;ENTRY
1297' 5E C LD E, (HL) ;GET LOW BYTE OF CURRENT REMOVE BUFFER
C ;ADDRESS TO E
1298' 23 C INC HL ;BUMP PTR TO HI BYTE
1299' 56 C LD D, (HL) ;GET HI BYTE OF SAME TO
129A' 7A C LD A, D ;PUT HI BYTE IN A TO OR WITH LO BYTE
129B' B3 C OR E ;OR HI AND LO BYTES TO CHECK FOR ZERO
129C' EB C EX DE, HL ;PUT ADDR IN HL
129D' C9 C RET ;AND RETURN TO CALLER.
C
C PAGE

```

```

C
C
129E' C      DEV_ROOT:
C      ;      RETURNS ADDRESS OF DEVICE LIST ROOT IN HL FOR DEVICE SPECIFIED IN B.
C      ;
C      ;      INDEXES INTO LIST ROOT DATA STRUCTURE BY VALUE IN B
C      ;
C      ;      THE TWO HIGH ORDER BITS ARE ASSUMED TO BE FLAG BITS AND ARE
C      ;      THUS IGNORED.
C      ;
C      ; ENTRY:
C      ;      B = DEVICE NUMBER FOR WHICH LIST ROOT IS DESIRED.
C      ;
C      ; EXIT:
C      ;      HL = ADDRESS OF LIST ROOT FOR DEVICE.
C      ;
C      ; REGISTERS AFFECTED:
C      ;      A
C      ;      DE
C      ;      HL
C      ;
C
003F' C      XJ242 EQU 00111111B ;MASK TO STRIP OFF BITS 6 & 7
C
129E' 78 C      LD A,B ;MOVE DEVICE NUMBER TO A TO REMOVE HIGH BITS
129F' E6 3F C      AND XJ242 ;MASK OFF HI BITS
12A1' 17 C      RLA ;A = DEV # * 2 (CBIT IS RESET WITH ANI)
12A2' 5F C      LD E,A ;E = DEV # * 2
12A3' 17 C      RLA ;A = DEV # * 4
12A4' 83 C      ADD A,E ;A = DEV # * 6
12A5' 5F C      LD E,A ;MOVE A TO DE TO DO DOUBLE ADD
12A6' 16 00 C      LD D,0 ;ZERO HI BYTE OF D
12A8' 21 241D C      LD HL,D_RT_TBL ;POINT HL TO BASE OF DEVICE LIST ROOT TABLE
12AB' 19 C      ADD HL,DE ;ADD DEVICE OFFSET TO LIST ROOT
12AC' C9 C      RET ;AND RETURN...
C
C      PAGE

```

```

C
C
12AD' C   BUF_INIT:
C       ;   INITIALIZES BUFFER POINTED TO BY HL
C       ;
C       ;   CLEARS THE FOLLOWING ENTRIES IN THE BUFFER HEADER:
C       ;       POINTER TO NEXT REMOVE BUFFER
C       ;       BUFFER FILL INDEX
C       ;       BUFFER REMOVE INDEX
C       ;
C       ; ENTRY:
C       ;   HL = ADDRESS OF BUFFER TO INITIALIZE
C       ; EXIT:
C       ;   BUFFER IS INITIALIZED
C       ;
C       ; REGISTERS AFFECTED:
C       ;   A
C
12AD' 3E 00 C       LD    A,0           ;SET A TO VALUE TO SAVE IN BUFFER HEADER
12AF' 77 C       LD    (HL),A       ;CLEAR LOW BYTE OF NEXT REMOVE BUFFER ADDR
12B0' 23 C       INC    HL          ;POINT TO HI BYTE OF SAME
12B1' 77 C       LD    (HL),A       ;ZERO HI BYTE
12B2' 23 C       INC    HL          ;POINT TO FILL INDEX ENTRY
12B3' 77 C       LD    (HL),A       ;ZERO IT
12B4' 23 C       INC    HL          ;POINT TO REMOVE INDEX ENTRY
12B5' 77 C       LD    (HL),A       ;ZERO IT
12B6' C9 C       RET              ;AND RETURN
C
C       PAGE

```



```

C
C
12B7' C ALC_BUF:
C ; ALLOCATES A BUFFER IF ONE IS AVAILABLE. IF NONE AVAILABLE Z IS SET.
C ;
C ; IF THERE ARE ANY BUFFERS IN THE FREE LIST, THE FIRST IS ALLOCATED.
C ; THE BUFFER COUNT IN THE FREE LIST IS DECREMENTED.
C ;
C ; ENTRY:
C ; HIGH BIT OF B SET IF DEVICE HAS HIGH PRIORITY.
C ;
C ; EXIT:
C ; HL = ADDRESS OF ALLOCATED BUFFER, IF ANY, ELSE UNDEFINED.
C ; Z BIT IS SET IF NO BUFFERS ARE AVAILABLE.
C ;
C ; REGISTERS AFFECTED:
C ;
C ; HL
C ;
C
12B7' CD 12D1' C CALL ALC_ST ;SEE IF THERE IS A BUFFER TO ALLOCATE.
C ;HL ==> BUFFER COUNT CELL IN FREE LIST ROOT
C ;Z SET IF NO BUFFERS AVAILABLE.
12BA' C8 C RET Z ;IF NO BUFFERS AVAILABLE, RETURN.
C ;
12BB' 35 C DEC (HL) ;DECREMENT BUFFER COUNT
C ;Z BIT IS SET IF COUNT IS NOW ZERO
12BC' 2B C DEC HL ;DECREMENT HL TO POINT BACK TO FIRST FREE
12BD' 2B C DEC HL ;BUFFER ADDR ENTRY
12BE' 5E C LD E, (HL) ;GET LO BYTE OF FREE BUFFER ADDRESS
12BF' 23 C INC HL ;BUMP HL TO POINT TO HI BYTE
12C0' 56 C LD D, (HL) ;GET HI BYTE
12C1' 2B 0A C JR Z, XJ243 ;IF Z BIT IS SET, THE LAST FREE BUFFER HAS
C ;BEEN ALLOCATED (I.E., THE COUNT IS NOW ZERO),
C ;SO THERE IS NO NEED TO UPDATE THE POINTER IN
C ;THE FREE LIST ROOT TO THE NEXT FREE BUFFER.
C ;DE CONTAINS ADDRESS OF ALLOCATED BUFFER.
C ;
12C3' 2B C DEC HL ;DEC HL TO POINT BACK TO NEXT FREE BUFFER PTR ENTRY
12C4' EB C EX DE, HL ;SWAP ADDRESS OF FREE BUFFER AND FREE BUFFER PTR
C ;HL NOW POINTS TO CELLS CONTAINING ADDRESS
C ;OF NEXT FREE BUFFER.
12C5' 4E C LD C, (HL) ;GET ADDRESS OF NEXT FREE BUFFER TO BC
12C6' 23 C INC HL ;
12C7' 46 C LD B, (HL) ;HI BYTE...
12C8' 2B C DEC HL ;DEC HL TO POINT BACK TO NEWLY ALLOCATED BUFFER
C ;
12C9' EB C EX DE, HL ;SWAP ADDRESS OF FREE BUFFER AND FREE BUFFER
C ;PTR AGAIN.
12CA' 71 C LD (HL), C ;CHANGE PTR TO FIRST FREE BUFFER IN FREE LIST
12CB' 23 C INC HL ;ROOT TO POINT TO NEXT BUFFER
12CC' 70 C LD (HL), B ;HI BYTE...
C ;DE NOW CONTAINS ADDRESS OF ALLOCATED BUFFER
C ;AND THE FREE LIST HAS BEEN UPDATED
C ;

```

'BUFFER HANDLING ROUTINES'

```
12CD'      C      XJ243: ;DE CONTAINS THE ADDRESS OF ALLOCATED BUFFER
            C      ;AND THE FREE LIST HAS BEEN UPDATED IF
            C      ;NECESSARY.
12CD'  EB    C      EX      DE,HL      ;PUT BUFFER ADDRESS IN HL
12CE'  AF    C      XOR      A      ;CLEAR Z BIT TO INDICATE SUCCESSFUL ALLOCATION
12CF'  3D    C      DEC      A      ;
12D0'  C9    C      RET      ;AND RETURN.
            C
            C      PAGE
```

```

C
C
12D1' C      ALC_ST:
C      ;      CHECKS BUFFER ALLOCATION STATUS: ARE THERE ANY BUFFERS TO ALLOCATE.
C      ;
C      ; ENTRY:
C      ;      HIGH BIT OF B SET IF ALLOCATING DEVICE HAS HIGH PRIORITY.
C      ;
C      ; EXIT:
C      ;      Z IS SET IF THERE ARE NO BUFFERS AVAILABLE.
C      ;      HL ==> BUFFER COUNT CELL IN FREE LIST ROOT.
C      ;      (THIS IS PASSED TO AVOID ALC_BUF HAVING TO COMPUTE THAT ADDRESS.)
C      ;
C      ; REGISTERS AFFECTED:
C      ;
C      ;      A
C      ;      B
C      ;      HL
C      ;
C
12D1' 21 243B C      LD      HL,FRE_ADDR ;GET ADDRESS OF FREE LIST ROOT TO HL
C      ;DEVIDE NUMBER IN B IS PRESERVED
12D4' 23 C      INC      HL      ;BUMP TO POINT TO NUMBER OF BUFFERS ENTRY
12D5' 23 C      INC      HL      ;
12D6' 7E C      LD      A,(HL) ;GET NUMBER OF BUFFERS TO A
12D7' B7 C      OR      A      ;SET Z FLAG IF ZERO
12D8' C8 C      RET      Z      ;AND RETURN IF NO BUFFERS AVAILABLE
C      ;
12D9' C8 00 C      RLC      B      ;ROTATE HI BIT OF B INTO CY FLAG
12DB' 30 02 C      JR      NC,XJ244 ;IT'S NOT SET: PRIORITY IS LOW...
C      ;
C      ;PRIORITY IS HIGH. OK TO ALLOCATE LAST BUFFER
12DD' B7 C      OR      A      ;SET Z FLAG
12DE' C9 C      RET      ;AND RETURN
C      ;
12DF' XJ244: ;DEVICE'S PRIORITY IS NOT HIGH. CHECK TO SEE
12DF' FE 04 C      CP      THR_HOLD ;IF NUMBER OF BUFFERS AVAILABLE IS GREATER THAN
C      ;THE PRIORITY THRESHOLD
12E1' C8 C      RET      Z      ;IT'S EQUAL: RETURN WITH Z SET INDICATING
C      ;NO BUFFERS AVAILABLE
C      ;
12E2' D0 C      RET      NC      ;IT'S GREATER: RETURN WITH Z CLEAR INDICATING
C      ;BUFFERS AVAILABLE.
C      ;
12E3' AF C      XOR      A      ;IT'S LESS THAN: RETURN WITH Z SET INDICATING
12E4' C9 C      RET      ;NO BUFFERS AVAILABLE
C
C      PAGE

```

```

C
C
12E5' C D_ALC_BF:
C ; DE-ALLOCATES (I.E., PUTS IN THE FREE LIST) THE BUFFER POINTED TO
C ; BY HL.
C ; ENTRY:
C ; HL = ADDRESS OF BUFFER TO PUT IN FREE LIST
C ;
C ; EXIT:
C ; BUFFER POINTED TO BY HL IS PUT IN FREE LIST
C ;
C ; REGISTERS AFFECTED:
C ;
C ; DE
C ; HL
C
C
12E5' E5 C PUSH HL ;SAVE ADDRESS OF BUFFER TO DE-ALLOCATE
12E6' 21 243B C LD HL,FRE_ADDR ;GET ADDRESS OF FREE LIST ROOT TO HL
12E9' 4E C LD C,(HL) ;GET POINTER TO FIRST FREE BUFFER
12EA' 23 C INC HL ;TO BC
12EB' 46 C LD B,(HL) ;
12EC' 23 C INC HL ;GET FREE BUFFER COUNT TO A
12ED' 7E C LD A,(HL) ;
12EE' 34 C INC (HL) ;BUMP FREE BUFFER COUNT
12EF' 2B C DEC HL ;POINT HL BACK TO FREE LIST ROOT + 1, HI BYTE
C ; OF PTR TO FIRST FREE BUFFER ADDR
12F0' D1 C POP DE ;RESTORE ADDRESS OF BUFFER TO DE-ALLOCATE TO DE
12F1' 72 C LD (HL),D ;SET PTR TO FIRST FREE BUFFER TO POINT TO
12F2' 2B C DEC HL ;THIS BUFFER
12F3' 73 C LD (HL),E ;
C ;
12F4' B7 C OR A ;IS OLD BUFFER COUNT ZERO? (I.E., IS THE
C ; BUFFER JUST ADDED TO THE LIST THE ONLY ONE IN
C ; IT?
12F5' C8 C RET Z ;YES, RETURN: THERE IS NO POINT UPDATING PTR
C ; IN BUFFER TO POINT TO NEXT FREE BUFFER SINCE
C ; THERE IS NO NEXT FREE BUFFER
C ;
12F6' EB C EX DE,HL ;PUT POINTER TO BUFFER ADDRESS IN HL
12F7' 71 C LD (HL),C ;SET POINTER IN BUFFER TO POINT TO NEXT
12F8' 23 C INC HL ;FREE BUFFER
12F9' 70 C LD (HL),B ;
12FA' C9 C RET ;AND RETURN
C

```

PAGE

'EXECUTIVE ROM MONITOR'
'BUFFER HANDLING ROUTINES'

MACRO-80 3.44 09-Dec-81

PAGE 1-347

SUBTTL
PAGE

'HIGH RAM ROUTINES (BANK SWITCHING)'

```
C          SUBTTL      'HIGH RAM ROUTINES (BANK SWITCHING)'  
C          INCLUDE ROM8                                ;HIGH RAM ROUTINES (BANK SWITCHING)  
C          ;  
C          ;  
C          ;          *****  
C          ;          *                                *  
C          ;          *          HIGH RAM ROUTINES          *  
C          ;          *                                *  
C          ;          *****  
C          ;  
C          ;          MODULE 8 --ROM8-- ENTRY POINTS  
C          ;  
C          ;          HIGH RAM ROUTINES  
C          ;  
C          ;          INT_HL:      INTERRUPT HANDELING (CHECK FOR INTERRUPTS)  
C          ;          INT_HD:      INTERRUPT HANDELING (SERVICE INTERRUPTS)  
C          ;          DMARD:       TRANSFER DATA FROM CONTROLLER TO MEMORY  
C          ;          DMAWRT:      TRANSFER DATA FROM MEMORY TO CONTROLLER  
C          ;          BANKCALL:    CALLS AN ADDRESS IN ANY BANK  
C          ;          BANKJMP:     JUMP TO AN ADDRESS IN ANY BANK  
C          ;          BANKMOVE:    MOVE A BLOCK FROM ONE BANK TO ANOTHER  
C          ;          MOVE:        PERFORM A BLOCK MOVE  
C          ;  
C          PAGE
```

'HIGH RAM ROUTINES (BANK SWITCHING)'

```
C
C
C      ;      .Z80
C      ;      .XLIST
C      ;      INCLUDE      SYSS1.MAC      ;SYSTEST FILE
C      ;      INCLUDE      RAMSMON.MAC    ;MONITOR RAM MEMORY LOCATIONS
C      ;      INCLUDE      RAMS10.MAC     ;RAM MEMORY LOCATIONS AND CONSTANTS
C      ;      .LIST
C      ;      ;SYSS1.MAC , RAMSMON.MAC AND RAMS10.MAC INCLUDED
C
C      PUBLIC  STINT,INT_HL,INT_HD,DWARD,DMAWRT,BANKCALL,BANKIMP,BANKMOVE
C      PUBLIC  LENINT
C
C
C      PAGE
```


[illegible]

'HIGH RAM ROUTINES (BANK SWITCHING)'

```

C
C
FF25      C      INT_HD:
C          ;      THE INTERRUPT HANDLER ENTRY POINT.
C          ;      THE INPUT TO THIS ROUTINE IS A PTR TO A TABLE ENTRY
C          ;      INDICATING THE PIA MASK TO SWITCH IN THE BANK IN WHICH THE
C          ;      INTERRUPT HANDLER RESIDES AND THE ADDRESS OF THE HANDLER.
C          ;
C          ;      THE CURRENT BANK STATE IS SAVED, THE INTERRUPT HANDLER'S
C          ;      BANK IS FLAGGED IN AND THE ROUTINE IS CALLED.
C          ;
C          ;      SINCE THIS ROUTINE SWITCHES BANKS, IT MUST RESIDE IN HIGH MEMORY.
C          ;
C          ; ENTRY:
C          ;      HL ==> TABLE ENTRY
C          ;
C          ; EXIT:
C          ;      INTERRUPT HAS BEEN SERVICED.
C          ;
C
C
FF25      DB 00      C          IN      A, (SYS_DTA)
FF27      F5          C          PUSH   AF          ;SAVE CURRENT BANKS
C
FF28      7E          C          LD      A, (HL)          ;LOAD INTERRUPT HANDLER BANK STATE FROM TABLE
C
FF29      23          C          INC     HL
FF2A      5E          C          LD      E, (HL)          ;LOAD INT. HDLR. ADDR FROM TABLE
FF2B      23          C          INC     HL
FF2C      66          C          LD      H, (HL)
FF2D      6B          C          LD      L, E          ;HL = INTERRUPT HANDLER ADDRESS
C
FF2E      D3 00      C          OUT     (SYS_DTA), A      ;SET INTERRUPT HANDLER BANK
C
FF30      11 FF35     C          LD      DE, XJ246
FF33      D5          C          PUSH   DE          ;SET RETURN ADDRESS
C
FF34      E9          C          JP      (HL)          ;"CALL" ROUTINE
C
FF35      F1          C      XJ246: POP     AF          ;RESTORE CURRENT BANKS
FF36      D3 00      C          OUT     (SYS_DTA), A
FF38      C9          C          RET
C
C          PAGE

```

'HIGH RAM ROUTINES (BANK SWITCHING)'

```

C
C
FF39      C      DMARD:
C      ;TRANSFER DATA FROM CONTROLLER TO MEMORY
C      ;ENTRY
C      ;BC          =      BYTES TO TRANSFER
C      ;HL          =      FWA OF BUFFER
C      ;DMABANK     =      BANK
C
C      ;EXIT
C      ;HL          =      NEXT ADDRESS
C
C
C      ;SAVE CURRENT BANKS AND SWITCH TO DMA BANK
FF39      DB 00      C      IN      A,(SYS_DTA)      ;(13) GET CURRENT BANKS
FF3B      F5          C      PUSH   AF              ;(11) SAVE
FF3C      3A FD4A     C      LD      A,(DMABANK)      ;(13)
FF3F      D3 00      C      OUT     (SYS_DTA),A      ;(13) SET DMA BANK
C
C      ;DO DMA
C
FF41      DB 00      C      XJ247: IN      A,(DSK_STS)      ;(13) GET STATUS
FF43      1F          C      RRA                      ;(4)
FF44      30 0C      C      JR      NC,XJ248          ;(7) RETURN IF NO BUSY
FF46      1F          C      RRA                      ;(4)
FF47      30 F8      C      JR      NC,XJ247          ;(10) IF NO DRQ
C
FF49      DB 0B      C      IN      A,(DSK_DAT)      ;(13) GET BYTE
FF4B      77          C      LD      (HL),A          ;(7) STORE BYTE
FF4C      23          C      INC     HL              ;(6)
FF4D      0B          C      DEC     BC              ;(6)
FF4E      78          C      LD      A,B              ;(4)
FF4F      B1          C      OR      C                ;(4)
FF50      20 EF      C      JR      NZ,XJ247          ;(10)
C
C      ;RESTORE BANKS
FF52      F1          C      XJ248: POP   AF
FF53      D3 00      C      OUT     (SYS_DTA),A
FF55      C9          C      RET
C
C      PAGE

```

'HIGH RAM ROUTINES (BANK SWITCHING)'

```

C
C
FF56      C      DMAWRT:
C      ;XFER DATA FROM MEMORY TO DISK
C      ;ENTRY
C      ;BC          =      BYTES TO TRANSFER
C      ;HL          =      FWA OF BUFFER
C      ;DMABANK     =      BANK
C
C      ;EXIT
C      ;HL          =      NEXT ADDRESS
C
C
C      ;SAVE CURRENT BANKS AND SWITCH TO DMA BANK
FF56      DB 00      C      IN      A, (SYS_DTA)      ; (13) GET CURRENT BANKS
FF58      F5          C      PUSH   AF              ; (11) SAVE
FF59      3A FD4A     C      LD      A, (DMABANK)      ; (13)
FF5C      D3 00      C      OUT     (SYS_DTA), A      ; (13) SET DMA BANK
C
C      ;DO DMA
C
FF5E      DB 0B      C      XJ249: IN      A, (DSK_STS)      ; GET STATUS
FF60      1F          C      RRA
FF61      30 0C      C      JR      NC, XJ250      ; RETURN IF NO BUSY
FF63      1F          C      RRA
FF64      30 FB      C      JR      NC, XJ249      ; IF NO DRQ
C
FF66      7E          C      LD      A, (HL)      ; GET BYTE
FF67      D3 0B      C      OUT     (DSK_DAT), A      ; STORE BYTE
FF69      23          C      INC     HL
FF6A      0B          C      DEC     BC
FF6B      78          C      LD      A, B
FF6C      B1          C      OR      C
FF6D      20 EF      C      JR      NZ, XJ249
C
C      ;RESTORE BANKS
FF6F      F1          C      XJ250: POP   AF
FF70      D3 00      C      OUT     (SYS_DTA), A
FF72      C9          C      RET
C
C      PAGE

```

'HIGH RAM ROUTINES (BANK SWITCHING)'

```

C
C
FF73      C      BANKCALL:
C      ;CALLS AN ADDRESS IN ANY BANK
C      ; (SAVES ALL REGISTERS)
C      ;ENTRY
C      ;JMPBANK      =      BANK
C      ;JMPADDR      =      ADDRESS
C
C      ;EXIT
C      ;NONE
C
C      ;SAVE CURRENT BANKS
FF73      32 237F      C      LD      (TEMP),A      ;SAVE A
FF76      DB 00      C      IN      A,(SYS_DTA)
FF78      F5      C      PUSH     AF      ;SAVE BANKS
FF79      3A 237F      C      LD      A,(TEMP)      ;RESTORE A
C
C      ;CALL THE ADDRESS
FF7C      CD FF88      C      CALL     BANKJMP
C
C      ;RESTORE BANKS
FF7F      E3      C      EX      (SP),HL      ;SAVE HL ON STACK, GET BANKS IN HL
FF80      F5      C      PUSH     AF      ;SAVE AF
C
FF81      E5      C      PUSH     HL      ;PUT BANKS ON STACK
FF82      F1      C      POP      AF      ;GET A = BANKS
C
FF83      D3 00      C      OUT      (SYS_DTA),A      ;RESTORE BANKS
C
FF85      F1      C      POP      AF      ;RESTORE AF
FF86      E1      C      POP      HL      ;RESTORE HL
FF87      C9      C      RET
C
C      PAGE

```

'HIGH RAM ROUTINES (BANK SWITCHING)'

```
C
C
FF88      C      BANKJMP:
C      ;JUMP TO AN ADDRESS IN ANY BANK
C      ; *NOTE -- USES A AND HL*
C      ;ENTRY
C      ;JMPBANK      =      BANK
C      ;JMPADDR      =      ADDRESS
C
C      ;EXIT
C      ;ALL REGISTERS SAVED
C
C
C      ;SWITCH BANKS
FF88      32 FD55   C      LD      (TEMSAV),A      ;SAVE A
FF88      3A FD5B   C      LD      A,(JMPBANK)
FF8E      D3 00     C      OUT     (SYS_DTA),A      ;SWITCH BANKS
C
C      ;JUMP TO ADDRESS
FF90      3E C3     C      LD      A,0C3H
FF92      32 FD58   C      LD      (JMPSPOT),A      ;SET UP TO JUMP
FF95      3A FD55   C      LD      A,(TEMSAV)      ;RESTORE A
C
FF98      C3 FD58   C      JP      JMPSPOT
C
C      PAGE
```

```

C
C
FF9B      C      BANKMOVE:
C      ;MOVE A BLOCK (1-10000H BYTES) FROM ONE BANK TO ANOTHER
C      ; (IF BC = 0, 10000H BYTES ARE MOVED)
C      ; THIS ROUTINE USE "5" LEVEL OWN STACK
C      ;ENTRY
C      ;BC          =      NUMBER OF BYTES TO TRANSFER (0 INDICATES 10000H)
C      ;HL          =      SOURCE ADDRESS
C      ;DE          =      DESTINATION ADDRESS
C      ;SRCBANK      =      SOURCE BANK
C      ;DESBANK      =      DESTINATION BANK
C
C      ;EXIT
C      ;HL          =      NEXT SOURCE ADDRESS
C      ;DE          =      NEXT DESTINATION ADDRESS
C
C
C      ;SAVE CURRENT BANKS
C
FF9B      DB 00      C      IN      A, (SYS_DTA)
FF9D      FS          C      PUSH   AF
C
C      ;MOVE BLOCK
C      ;MOVE C BYTES
FF9E      79          C      LD      A,C
FF9F      B7          C      OR      A
FFA0      2B 0B       C      JR      Z,XJ251      ;IF C = 0, ONLY MOVE B * 100H
C
FFA2      C5          C      PUSH   BC          ;SAVE B
FFA3      06 00       C      LD      B,0
FFA5      CD FFBB     C      CALL   MOVE      ;MOVE C BYTES
FFA8      C1          C      POP     BC
C
C      ;MOVE B * 100H BYTES
FFA9      78          C      LD      A,B
FFAA      B7          C      OR      A
FFAB      2B 0A       C      JR      Z,XJ252      ;IF B = 0
C
FFAD      C5          C      XJ251: PUSH   BC
FFAE      01 0100     C      LD      BC,100H
FFB1      CD FFBB     C      CALL   MOVE      ;MOVE 100H BYTES
FFB4      C1          C      POP     BC
FFB5      10 F6       C      DJNZ   XJ251      ;IF MORE
C
FFB7      F1          C      XJ252: POP     AF          ;RESTORE ORIGINAL BANK
FFB8      D3 00       C      OUT     (SYS_DTA),A
C
FFBA      C9          C      RET
C
C      PAGE

```

'HIGH RAM ROUTINES (BANK SWITCHING)'

```

C
C
FFB8      C      MOVE:
C      ;MOVE A BLOCK FROM ONE BANK TO ANOTHER (1-256 BYTES)
C      ;*NOTE -- NO ERROR CHECKING DONE AT THIS LEVEL, BC MUST BE BETWEEN 1 AND 256
C      ;ENTRY
C      ;HL      =      SOURCE ADDRESS
C      ;DE      =      DESTINATION ADDRESS
C      ;BC      =      NUMBER OF BYTES
C
FFB8      C5      C      PUSH      BC      ;SAVE COUNT
FFB8      D5      C      PUSH      DE      ;SAVE DESTINATION ADDRESS
C
C      ;MOVE SOURCE TO HIGH MEMORY
FFB8      3A FD56  C      LD      A, (SRCBANK)
FFC0      D3 00   C      OUT     (SYS_DTA), A      ;SET SOURCE BANK
C
FFC2      11 FDDC  C      LD      DE, HIBUFF      ;HIGH RAM BUFFER
FFC5      ED B0   C      LDIR
C
C      ;MOVE BLOCK FROM HIGH MEMORY TO DESTINATION
FFC7      D1      C      POP      DE      ;RESTORE DESTINATION
FFC8      C1      C      POP      BC      ;RESTORE COUNT
C
FFC9      E5      C      PUSH     HL      ;SAVE NEXT SOURCE ADDRESS
C
FFCA      3A FD57  C      LD      A, (DESBANK)
FFCD      D3 00   C      OUT     (SYS_DTA), A      ;SET DESTINATION
C
FFCF      21 FDDC  C      LD      HL, HIBUFF
FFD2      ED B0   C      LDIR
C
FFD4      E1      C      POP      HL      ;RESTORE NEXT SOURCE ADDRESS
FFD5      C9      C      RET
C
C      IF      $ GT 0FFDFH
C      .PRINTX      /SIO VECTOR INTERRUPT TABLE CLASHED/
C      ENDIF
C
00E6      C      LENINT EQU      $ - INT_HL      ;LENGTH OF INTERRUPT ROUTINES TO BE MOVED TO HIGH RAM
C
C      .DEPHASE
C      PAGE

```


'EXECUTIVE ROM MONITOR' MACRO-80 3.44 09-Dec-81 PAGE 1-359
'HIGH RAM ROUTINES (BANK SWITCHING)'

;
END

Macros:

ALIGN POPALL PUSHAL

Symbols:

0FA0	@FREQ	039B'	ABSOLUTE	0003	ABTC
0006	ACK	0391'	ADDROWS	12B7'	ALC_BUF
12D1'	ALC_ST	0003	ALPH_KY	0010	ALTBIT
0000	ALTFONT	0010	ALTMODE	001E	ALTSEC
0008	ALT_FLG	0002	AL_BIT	2013	AL_KEY
04D6	AL_TB	0003	AL_TB_NO	0040	AL_T_SZ
0001	APP_TBL	00FC	ATR_MSK	001F	AUTOBOOT
2004	A_DRIVE	2008	A_PROTCL	275E	A_XMT_OFF
2357	BACKGND	000E	BACKNUM	0000	BACKSIZ
05BB'	BACKT	00FF	BAD	00FF	BADKEY
FF73I	BANKCALL	FF8BI	BANKJMP	FF9BI	BANKMOVE
04F7'	BASET	0001	BASNUM	0001	BASSIZ
0044	BAS_CNL	0A9C'	BAUDTBL	0008	BD1200
036E	BD_MSG	0004	BF_OV_HD	0000	BIOSBANK
FDDC	BIOSTK	0008	BKS	0002	BLKBIT
0020	BLK_CUR	0020	BLMODE	0010	BNK_FLG
019F	BOOT	4000	BOOTADDR	0000	BOOTBANK
0010	BTBLEN	243E	BUFF_ST	12AD'	BUF_INIT
1110'	B_CHNG	2006	B_DRIVE	2009	B_PROTCL
275F	B_XMT_OFF	0035	CA_64C	0034	CA_DAT
0030	CA_DIR	0007	CBELL	000E	CBIOSV
0193	CBOOT	0037	CB_60H	0036	CB_DAT
0032	CB_DIR	0004	CDISK	237A	CFLAG
0014	CHARGESEC	064F'	CHECK1ST	06EF'	CHKEY
0000	CHARFONT	1034'	CH_A_SPEC	102A'	CH_B_SPEC
0607'	CKEYSDN	0609'	CKLP	0679'	CKSERV
000F	CLATR_MSK	2000	CLK_PTR	01BB'	CLRLN
01C8'	CLR_CHAR	01D5'	CLR_EOS	0008	COL_MSK
FD40	COMJMP	02BD	CONINIT	235B	CONTBLS
0018'	CON_IN	23E4	COUNT	0036	COUNTA
0076	COUNTB	0064'	COUT	000D	CR
001F	CRLF	0004	CS_TB_NO	235B	CTBLADDR
05DF'	CTBLPTRS	0A2C'	CTC_INIT	0003	CTLNUM
000A	CTLSIZ	0587'	CTLT	0002	CTL_KY
000C	CTYPNUM	0000	CTYPSIZ	05D7'	CTYPT
0000	CT_BIT	2012	CT_KEY	0002	CT_TB_NO
FD53	CURPOS	2356	CURTYPE	237B	CURWIND
23D4	CUR_I_ST	09C8'	CV2IOP	09C2'	CV2IP
09B9'	CV2OP	0016	CVER	00F7	C_60_HZ_A
0030	C_60_HZ_D	0004	C_DAT	00FB	C_DIR
00FD	C_DMA	0002	C_FDC	00CF	C_I_KBD
0002	C_PAR	0002	C_RTC	00D0	D.FINT
00C0	D.RDA	0088	D.RDS	00E0	D.RDT
0010	D.SEK	0020	D.STP	0040	D.STPI
0060	D.STPD	00A8	D.WRTS	00F0	D.WRTT
23F2	DACTVE	0080	DATA_INIT	0007	DATB_INIT
0080	DBUF	0001	DB_CT	23E8	DCHK_FLG
02B9'	DEFCUR	03FC'	DEF_WINDOW	0E69'	DELAY
0670'	DELLAST	FD57	DESBANK	129E'	DEV_ROOT
0066	DFKT	0AD6	DIAG_INIT	00FF	DIRA_INIT
003F	DIRB_INIT	049F'	DISPLAY	04B1'	DIST2EOL
04C0'	DIST2LASTROW	003C	DKC	0080	DMA

FD4A	DMA8ANK	FD4B	DMADR	FF39I	DMARD
FF56I	DMAWRT	0F0E'	DMA_INT	0159'	DO_CR
0167'	DO_LF	0004	DRV_A	0002	DRV_B
23D3	DRV_MSK	23C7	DSKSWP	0008	DSK_CMD
000B	DSK_DAT	000A	DSK_SEC	0008	DSK_STS
0009	DSK_TRK	23C9	DSTSB	12E5'	D_ALC_BF
0006	D_RT_LEN	241D	D_RT_TBL	035E	EB_MSG
0186'	EDELIC	017A'	EEOL	0180'	EEOS
0065	EFKT	0196'	EINSRT	003E	EKC
FD5C	EMRAM	062B'	ENDCK	044C'	ENDCOL
0434'	ENDROW	001A	EOF	0024	EDM
007F	ERC	23EA	ERRFLG	0A46	ERROR
0A75	ERROR_MSG	0040	ERSIZE	001B	ESC
0271'	ESCCAL	0285'	ESCCBL	0279'	ESCCGR
0293'	ESCCCK	027D'	ESCCRV	0281'	ESCCUN
029C'	ESCDFT	01B2'	ESCEE	0298'	ESCEFT
0275'	ESCEHI	0002	ESCNUM	01A9'	ESCR
0255'	ESCSAL	0269'	ESCSBL	025D'	ESCSGR
0259'	ESCSHI	0021	ESCSIZ	028F'	ESCSKK
0261'	ESCSRV	0265'	ESCSUN	04FF'	ESCT
01EF'	ESCZZ	024C'	ESC_LCK	024F'	ESC_ULK
0003	ETX	101A'	EXT_CH_A	1022'	EXT_CH_B
0000	FALSE	005C	FCB	007C	FCB_NR
0008	FCOLNUM	0000	FCOLSIZ	05C7'	FCOLT
0F17'	FDC_INT	0D90'	FDSK	0032	FIFTY_HZ
1288'	FIL_BUF	23DE	FIRSTKEY	23D9	FKEYFLAG
2155	FKEY_ADR	11C5'	FL_B_CH	11B6'	FL_B_ST
00E5	FMTCHR	0D3B'	FMTTRK	038E	FONTLEN
005B	FONT_ROWS	0516	FONT_TABLE	0D7C'	FORINT
0E1E'	FORMAT	23E5	FPNTR	243B	FRE_ADDR
0007	FROWNUM	0000	FROWSIZ	05C3'	FRONT
2385	FSTCOL	2384	FSTROW	C000	FWAVM
237B	FWAWIND	000C	F_EX	0020	F_NR
000F	F_RC	0021	F_RRF	000E	F_S2
03AC'	GETCOL	03B9'	GETROW	0491'	GETWIND
033F'	GET_ATR	0E75'	GET_INT	0B18'	GET_I_STAT
0B11'	GET_Q_S1	0B0E'	GET_Q_STAT	0014	GROSS_BF
0004	GR_FLG	0704'	GTMASK	1263'	GT_I_BUF
0001	HAFBIT	0000	HAFMODE	0040	HAF_FLG
2355	HERTZ	FD0C	HIBUFF	0080	HI_PRIOR
FEF0	HI_ROUT	0BF8'	HOME	0024	HR_DAY
07AF'	IE.CO	080E'	IE.GTS	08DC'	IE.IDM
088B'	IE.DDM	0863'	IE.OIM	0948'	IE.OP
089A'	IE.SHK	07FC'	IE.SI	0818'	IE.TC
0965'	IEI10	0971'	IEI20	0979'	IEI30
097C'	IEI40	098C'	IEINP	095C'	IEINSTAT
0999'	IED05	09A8'	IED22	09B0'	IED40
0962'	IEDSTAT	0998'	IEDUT	FD52	IE_ADR
23DC	IE_CHAR	0020	IE_LSTN	0040	IE_TALK
003F	IE_ULST	005F	IE_UTLK	03AE	IMSG
003F	INITIALIZE	0223	INIT_INT	0294	INIT_POLL
FFFE	INTVEC	0000	INT_CNL	FF25I	INT_HD
FEF0I	INT_HL	0F14'	INT_KBD	0001	INT_MSK
0EBD'	INT_POLL	23F3	INT_TBL	0000	INV_CUR
1161'	IN_AL_BF	0003	IOBYTE	0018	IRPTCT
0080	IRQ1	0040	IRQ2	FDA0	ISTK

'HIGH RAM ROUTINES (BANK SWITCHING)'

FD60	I_POL_ADR	FD5F	I_POL_BANK	23D6	I_POL_BNK
FD59	JMPADDR	FD5B	JMPBANK	FD58	JMPSPOT
04FC'	JX01	05B4'	JX02	05B0'	JX03
05BC'	JX04	05C0'	JX05	05C4'	JX06
05C8'	JX07	05CC'	JX08	05D0'	JX09
05D4'	JX10	05D8'	JX11	05DC'	JX12
0400	K	05FD'	KBDVR	0000	KBD_DEV
0692'	KBSCAN	0713'	KBSERV	02A0	KB_INIT
2358	KCLICK	0007	KCOL_M	0002	KCTRK
0014	KEYBD	23DD	KEYLCK	23DE	KEYLST
0625'	KEYDN	2015	KEYS	0011	KEYSEC
26C0	KEYTBL	243E	KEY_BF	241D	KEY_RT
0002	KLE_LEN	0003	KL_LEN	0007	KL_USED
0038	KROW_M	0040	KTBL_LEN	0006	KY_SRVD
200A	K_MSK_TBL	000C	K_TB_SZ	0000	L\$PRU
23E2	LASTKEY	03A2'	LASTROW	23E7	LAST_TCK
000A	LCOLNUM	0000	LCOLSIZ	05CF'	LCOLT
034A	LD_MSG	001E	LENCTBL	00E6I	LENINT
000A	LENJMP	000A	LF	0078'	LOOKUPB
04D8'	LOOP	0009	LROWNUM	0000	LROWSIZ
05CB'	LROWT	0100	LSECB	09CB	LSI_TEST
0024	L_FCB	0000	L_FILEN	0003	L_FILEX
08C8	L_PWR_UP	000C	MCRIGH	000B	MCUP
2002	MEDIA	08CD	MEMORY_TEST	0940	MEM_CHECK
08AB	MEM_DONE	0933	MEM_TEST	0060	MIN_HR
00F0	MODEMSK	1000	MODEOFF	2000	MONRAM
09A2	MORE	FFBB	MOVE	FD4A	MRAM
000A	MSEC	0028	MTRK	0010	MX_CH_BF
0002	NDSKS	02B3'	NEW_BACK	0000	NM_TB_NO
075E'	NOALPHA	074C'	NOCTRL	068F'	NOSERV
0747'	NOSHCTRL	0755'	NOSHIFT	04CA'	NOTE
1071'	NOTNOW	125E'	NOT_KEYBRD	0028	NO_BUFS
0005	NO_DEVS	0002	NO_TBL	23D1	NRETRY
23CF	NUMSEC	000D	OFFNUM	2379	OFFSET
0000	OFFSIZ	05DB'	OFFT	0006	OM.LRAM
0100	ORGP	0323	OSTR	002A	PA.CDR
002E	PA.CDT	0000	PA.DRI	00FF	PA.DRD
0A0C'	PARINP	0A1A'	PAROUT	0011	PAR_CNA
0013	PAR_CNB	0010	PAR_DTA	0012	PAR_DTB
0F11'	PAR_INT	0000	PB.CDR	0004	PB.CDT
00BF	PB.DR	000B	PB.DTI	0002	PB.DTO
09DA	PIA	23DA	PIACTL	0A09'	PIS20
09F7'	PISTAT	08A4	PODIAG	09F6'	POS10
09EC'	POSTAT	0002	PP.IN	0000	PP.IRDY
23DB	PP.MODE	0040	PP.ORDY	0001	PP.OUT
00AC'	PROCNDRM	0002	PROD_CODE	0C3A'	PSEKC
08A6	PWR_UP	0CE9'	RADR	2000	RAMTBL
00E1	RCV_CNL	23F1	RDFLAG	070E'	RDROW
0B25'	RDRV	0C6E'	RD_WRT	0C62'	READ
0000	REG0	0001	REG1	0002	REG2
0003	REG3	0004	REG4	0005	REG5
0006	REG6	0007	REG7	065C'	REMOVEKEY
0000	RESERVED	0F1A'	RESET_INT	0218	RES_SIO
049E'	RETPROC	0004	REVBIT	0000	REVMODE
02AA'	REVOFF	02A1'	REVON	0000	REV_CUR
0000	REV_FLG	1292'	RMV_BUF	1226'	RM_C_BF

1273'	RM_C_ST	23EF	ROMCLK	2359	ROMCON
0D2C'	ROMREAD	FDDC	ROMSTK	0100	ROMVEC
00FE	ROMVER	0080	ROM_BANK	0992	ROM_TEST
0081	ROW0_M	000A	ROW_LEN	0001	RPTCTR
0B44'	RSEC	2760	RSTHLD	2761	RSTMODE
1062'	RTC_DRV	1062'	RTC_INT	0018	RTC_OFL
23C8	STRY	1137'	R_INT_TBL	1161'	R_I_TBL_END
002A	R_I_TBL_LEN	04CA	R_K_IN_TB	23D0	R_WCOM
0B50'	R_WSEC	0B04'	SAI_STAT	0AFC'	SAD_S1
0AFA'	SAD_STAT	0783'	SAVEKEY	FD4D	SAVSEC
FD4E	SAVTRK	FD51	SAVTYP	0B09'	SBI_STAT
0AFF'	SBD_STAT	00B5	SCLFRE	2386	SCRL_UPF
0200'	SCROL_DOWN	01F8'	SCROL_UP	FD50	SDISK
23E0	SECONDKEY	0060	SEC_MIN	0C20'	SEEK
23F0	SEKDEL	0DB8'	SELDIV	0B97'	SENDEN
049B'	SEDEER	0ABC'	SERAIN	0AE5'	SERAO1
0A08'	SERADUT	0ACA'	SERBIN	0AE4'	SERBOUT
0658'	SERVKEY	0A7A'	SETBAUD	02F8	SETCHAR
03CB'	SETCOL	0B1E'	SETDMA	030D	SETJMP
038C'	SETOFF	03E3'	SETROW	009C'	SETTBL
047E'	SETWIND	02E8'	SET_ATR	0315'	SET_BACK
0E7A'	SET_INT	0004	SHFT_KY	0001	SH_BIT
2014	SH_KEY	0001	SH_TB_NO	0001	SI_ARDY
0004	SI_TRDY	0A27	SIO	0A39'	SIOINIT
000D	SIO_CNA	000F	SIO_CNB	000C	SIO_DTA
000E	SIO_DTB	0017	SIO_I_IN	FFE0	SIO_I_TBL
0A32	SIO_TEST	003C	SIXTY_HZ	0000'	SKEY
0008	SPEAKER	103E'	SPEC_HD	0001	SP_ALT
0002	SP_CHAR	0004	SP_KEY	FD56	SRCBANK
0003	SRPTCT	0689'	SRVDLDRY	0D74'	SSEL
0000	START	041C'	STARTCOL	0405'	STARTROW
0C2E'	STEP	0C32'	STEPIN	0C36'	STEPOUT
12FB1'	STINT	0319	STJMP	0020	STRB
000E	SVER	0005	SYS	0010	SYSDAT
0004	SYSDSK	0244	SYSINIT	0006	SYSL
0000	SYSR	0013	SYSTEM	0001	SYS_CNA
0003	SYS_CNB	0000	SYS_DTA	0002	SYS_DTB
0000	SYS_TBL	23E9	SYS_TICKS	23D7	S_A_FLS
23D8	S_B_FLS	0001	S_CH_A_I	0003	S_CH_A_O
0002	S_CH_B_I	0004	S_CH_B_O	0002	S_INT_PDG
0F20'	S_R_I_CH_A	0F63'	S_R_I_CH_B	0FA6'	S_T_I_CH_A
0FE0'	S_T_I_CH_B	0204'	S_UP_DOWN	0009	TAB
0000	TBUFF	237F	TEMP	0D00	TEMPBUF
2381	TEMPFWA	2383	TEMPROWS	FD5E	TENSAV
0A5C	TEST_MESSAGE	0004	THR_HOLD	0042	TIMPTR
0007	TIM_CNL	0004	TIM_CT0	0005	TIM_CT1
0006	TIM_CT2	00FE	TOG_ALT_SET	02F8'	TOG_ATR
0309'	TOG_CUR	0008	TOT_COL	0008	TCT_ROW
FFFF	TRUE	00E8	TX_CNL_A	00EA	TX_CNL_B
0008	UNDBIT	0040	UNDMODE	0040	UND_CUR
094F	UNIQUE_ADDR	0020	UN_FLS	FD5C	USR_BNK
FD62	USR_STK	0078	VBACK	003D	VCAD
0054	VCEOL	0059	VCEOS	001A	VCARS
002E	VCTYP	0243'	VC_BEL	00F3'	VC_BKS
01EF'	VC_CLRS	00D9'	VC_CR	0152'	VC_HOME
00E3'	VC_LF	0143'	VC_MCDOWN	0113'	VC_MCRT

012B'	VC_MCPU	00ED'	VC_NEWLINE	0057	VDEL
0052	VDELL	0016	VDWN	0041	VEAL
0071	VEBL	0047	VEGH	0028	VEHI
00FC	VERSION	006B	VERV	006D	VEUL
00B5'	VGRAPH	001E	VHOME	0040	VID_BANK
0051	VINC	0045	VINL	001F	VLIN
0080	VLL	0023	VLOCK	0050	VMCOLS
0018	VMROWS	00C5'	VNORM	006F	VCFF
00D3'	VOUT	0064	VROFF	0062	VRON
0053	VSAD	0061	VSAL	005E	VSBL
0067	VSGH	0029	VSHI	006A	VSRV
006C	VSUL	0022	VUNLK	007A	VWDEF
0073	VWSET	0E07'	WBUSY	0006	WDEFNUM
0000	WDEFSIZ	05BF'	WDEFT	237E	WINDCOLS
2387	WINDDEFS	2380	WINDNUM	237D	WINDROWS
23D2	WP_STS	0C69'	WRITE	0B4B'	WSEC
000B	WSETNUM	0000	WSETGIZ	05D3'	WSETT
0393	WT_CR_MSG	037F'	XCOORD	0026	XJ000
003B	XJ001	00A7	XJ002	02B0	XJ003
02EB	XJ004	02FD	XJ005	08A8	XJ007
095D	XJ008	0976	XJ009	09FF	XJ010
0A15	XJ011	01AC	XJ012	01AD	XJ013
01F8	XJ014	0201	XJ015	0207	XJ016
020D	XJ017	0013'	XJ018	0016'	XJ019
0018'	XJ020	0020'	XJ021	004E'	XJ022
005A'	XJ023	0055'	XJ024	008F'	XJ025
0086'	XJ026	00AF'	XJ027	0101'	XJ028
010F'	XJ029	0121'	XJ030	0127'	XJ031
013F'	XJ032	0251'	XJ033	0193'	XJ034
01A4'	XJ035	026B'	XJ036	020B'	XJ037
0287'	XJ038	0294'	XJ039	029D'	XJ040
02C4'	XJ041	02E3'	XJ042	02CA'	XJ043
02D2'	XJ044	02D8'	XJ045	02E0'	XJ046
040F'	XJ047	0426'	XJ048	043F'	XJ049
0447'	XJ050	0457'	XJ051	045F'	XJ052
0171'	XJ053	01DD'	XJ054	023F'	XJ055
021A'	XJ056	0221'	XJ057	0230'	XJ058
0236'	XJ059	0329'	XJ060	032C'	XJ061
034D'	XJ062	0356'	XJ063	035F'	XJ064
0369'	XJ065	03D5'	XJ066	03F2'	XJ067
03FB'	XJ068	0396'	XJ069	04D0'	XJ070
04EA'	XJ071	107D'	XJ072	1080'	XJ073
1089'	XJ074	1094'	XJ075	10C4'	XJ076
111D'	XJ077	112D'	XJ078	068C'	XJ080
06A0'	XJ081	06E2'	XJ082	06AD'	XJ083
06DE'	XJ084	06C1'	XJ085	06D6'	XJ086
06DC'	XJ087	06EB'	XJ088	06ED'	XJ092
0708'	XJ094	0724'	XJ095	072F'	XJ096
073A'	XJ097	0779'	XJ098	079A'	XJ099
07EB'	XJ100	07DF'	XJ101	07FA'	XJ102
07FB'	XJ103	0848'	XJ104	0837'	XJ105
0844'	XJ106	0861'	XJ107	08C0'	XJ108
08A5'	XJ109	08B2'	XJ110	08DA'	XJ111
08BC'	XJ112	08C2'	XJ113	08CF'	XJ114
0903'	XJ115	0931'	XJ116	090F'	XJ117
091C'	XJ118	0945'	XJ119	092D'	XJ120

'HIGH RAM ROUTINES (BANK SWITCHING)'

0933'	XJ121	0941'	XJ122	0A58'	XJ123
0A76'	XJ124	0A7D'	XJ125	0A85'	XJ126
0A99'	XJ127	0A95'	XJ128	0AD9'	XJ129
0AE2'	XJ130	0AEB'	XJ131	0AF4'	XJ132
0B16'	XJ133	0B2B'	XJ134	0B42'	XJ135
0B36'	XJ136	0B6A'	XJ138	0B64'	XJ139
0B7D'	XJ140	0B95'	XJ141	0B8F'	XJ142
0B93'	XJ143	0BA2'	XJ144	0BA4'	XJ145
0BC2'	XJ146	0BF4'	XJ147	0BD5'	XJ149
0BEF'	XJ150	0BEB'	XJ151	0BF7'	XJ152
0C1E'	XJ153	0C1D'	XJ154	0C60'	XJ155
0C83'	XJ156	0C96'	XJ157	0C93'	XJ158
0CA2'	XJ159	0CA9'	XJ160	0CBC'	XJ161
0CE5'	XJ162	0CC6'	XJ163	0CDA'	XJ164
0CD8'	XJ165	0CE7'	XJ166	0D2A'	XJ168
0CFC'	XJ169	0D0E'	XJ170	0D29'	XJ171
0D2C'	XJ172	0D45'	XJ173	0D71'	XJ174
0D5C'	XJ175	0D69'	XJ176	0D72'	XJ177
0D8A'	XJ178	0D9A'	XJ179	0DA6'	XJ180
0DAA'	XJ181	0DB4'	XJ182	0DC9'	XJ183
0DE0'	XJ184	0DED'	XJ185	0E0A'	XJ186
0E1D'	XJ187	0E40'	XJ188	0E67'	XJ189
0E6A'	XJ190	0E6C'	XJ191	033D	XJ192
0335	XJ193	0E89'	XJ194	0E95'	XJ196
0EA1'	XJ197	0EAD'	XJ198	0EB9'	XJ199
0EBD'	XJ200	0ED5'	XJ201	0F01'	XJ202
0EE0'	XJ203	0EE8'	XJ204	0EF0'	XJ205
0EF8'	XJ206	0F00'	XJ207	0F5C'	XJ208
0F2A'	XJ209	0F4B'	XJ210	0F3B'	XJ211
0F9F'	XJ212	0F6D'	XJ213	0F6E'	XJ214
0F7E'	XJ215	0028	XJ216	0FD8'	XJ217
0FD3'	XJ218	0FCF'	XJ219	0FCF'	XJ220
0028	XJ221	1012'	XJ222	100D'	XJ223
1009'	XJ224	1009'	XJ225	0010	XJ226
0010	XJ227	0030	XJ228	0030	XJ229
1052'	XJ230	1181'	XJ231	119A'	XJ232
11C1'	XJ233	11E3'	XJ234	1223'	XJ235
1212'	XJ236	120A'	XJ237	123E'	XJ238
1285'	XJ241	003F	XJ242	12CD'	XJ243
12DF'	XJ244	FF12	XJ245	FF35	XJ246
FF41	XJ247	FF52	XJ248	FF5E	XJ249
FF6F	XJ250	FFAD	XJ251	FFB7	XJ252
0761'	XLATEKEY	0005	XNUM	0013	XOFF
0011	XON	05B7'	XTAB	0372'	YCOORD
0004	YNUM	05B3'	YTAB		

No Fatal error(s)

.280

TITLE bios0 (excbios0) jmp table & unchanging rtns

; *****

; external version #

; 1.1

1100

\$version equ 1100h ; Must be in hex form
; v m m m
; v=version #
; mmm = mod level
; (leading zeroes not shown)

; internal version #

; date last changed

3191

version equ 3191h ; MUST be in hex form
; y d d d
; y=last digit of year
; ddd = days since 1 January

; *****

; 1983-06-23 changed version # etc. for 2nd release

;*****

```

;*          *
;*      EXECUTIVE BIOS      *
;*          for              *
;*      CP/M 3.XX            *
;*          by                *
;*          *                *
;*      Glenn S. Tenney      *
;*          and               *
;*      Roger W. Chapman     *
;*          *                *
;*      Copyright (c) 1983   *
;*      Osborne Computer Corporation *
;*          *                *
;*****

```

; Table of contents of this module:

- ; (1) the signon message, with copyright
- ; (2) The bios jump table
- ; (3) A few unchanging routines of the bios.

page


```

0000'                                dseg                ; signon message only needed at cboot

0000"                                signon:              ; signon message at cold boot
0000" 1A                                db 1ah            ; clear screen

0001" 4F 73 62 6F                                db 'Osborne Executive CP/M Plus BIOS'
0005" 72 6E 65 20
0009" 45 78 65 63
000D" 75 74 69 76
0011" 65 20 43 50
0015" 2F 4D 20 50
0019" 6C 75 73 20
001D" 42 49 4F 53

0021" 20 56                                db ' V'          ; external version follows

0023" 31                                db ($version/1000h and 0fh)+'0' ; v
0024" 2E                                db ', '          ; .
0025" 31                                db ($version/100h and 0fh)+'0'  ; m

                                if ($version/10h and 0fh) OR ($VERSION AND 0Fh)
                                db ($version/10h and 0fh)+'0'      ; m
                                endif

                                IF $VERSION AND 0Fh
                                db ($version and 0fh)+'0'          ; m
                                ENDIF

0026" 0D 0A                                db 0dh,0ah

                                ; official copyright warning message

0028" 43 6F 70 79                                db 'Copyright ',1bh,'g',13h,1bh,'G ' ; "c" within a circle
002C" 72 69 67 68
0030" 74 20 1B 67
0034" 13 1B 47 20
0038" 31 39 38 33                                db '1983 Osborne Computer Corporation',0dh,0ah
003C" 20 4F 73 62
0040" 6F 72 6E 65
0044" 20 43 6F 6D
0048" 70 75 74 65
004C" 72 20 43 6F
0050" 72 70 6F 72
0054" 61 74 69 6F
0058" 6E 0D 0A
005B" 0D 0A 0A                                db 0dh,0ah,0ah ; cr lf lf
005E" 00                                db 0 ; ends the msg

                                ; Note: the very first data in common (cseg) MUST be
                                ; the jump table. The above messages are in bank 0
                                ; (dseg) and therefore can appear first in the source.

```

page

.list

```

public wboot@      ; to reference jmp tbl
public signon      ; signon message for cboot
public vers        ; and version number
public rvers       ; and place to put rom's vers

EXTRN CBOOT, WBOOT      ;SYSTEM INITIALIZATION
EXTRN DEVINI

EXTRN CONIST, CONIN, CONOUT ;CHARACTER I/O
EXTRN LIST, AUXOUT, AUXIN
EXTRN LISTST, CONOST
EXTRN AUXIST, AUXOST

EXTRN HOME, SELDSK, SETTRK ;DISK I/O
EXTRN SETSEC, SETDMA, SECTR
EXTRN READ, WRITE
EXTRN MULTIO, FLUSH

EXTRN MOVE, XMOVE, SELMEM, SETBNK ;MEMORY MANAGEMENT

EXTRN TIME

extrn drvtbl,devtbl      ; various tables
extrn torom              ; to give access to rom

```

005F" cseg ; common memory

```

0000' C3 0000* JP CBOOT
0003' C3 0000* wboot@: JP WBOOT
0006' C3 0000* JP CONIST
0009' C3 0000* JP CONIN
000C' C3 0000* JP CONOUT
000F' C3 0000* JP LIST
0012' C3 0000* JP AUXOUT
0015' C3 0000* JP AUXIN
0018' C3 0000* JP HOME
001B' C3 0000* JP SELDSK
001E' C3 0000* JP SETTRK
0021' C3 0000* JP SETSEC
0024' C3 0000* JP SETDMA
0027' C3 0000* JP READ
002A' C3 0000* JP WRITE
002D' C3 0000* JP LISTST
0030' C3 0000* JP SECTR
0033' C3 0000* JP CONOST
0036' C3 0000* JP AUXIST
0039' C3 0000* JP AUXOST
003C' C3 0064' JP ptdev ; DEVTBL
003F' C3 0000* JP DEVINI
0042' C3 0068' JP ptdrv ; DRVTL

```

0045'	C3 0000*	JP	MULTIO	
0048'	C3 0000*	JP	FLUSH	
004B'	C3 0000*	JP	MOVE	
004E'	C3 0000*	JP	TIME	
0051'	C3 0000*	JP	SELMEM	
0054'	C3 0000*	JP	SETBNK	
0057'	C3 0000*	JP	XMOVE	
005A'	C3 0000*	JP	torom ;	USERF
005D'	C3 0063'	JP	RESERV1	
0060'	C3 0063'	JP	RESERV2	

page

```

;      the following are (or should be) routines that will
;      not change

0053'      reserv1:
0053'      reserv2:
0053'  C9          ret      ;      not currently implemented

          extrn  devtbl,drvtbl  ; tables

;      ptdev  point to device table
;      returns address of serial device table

0064'  21 0000*      ptdev: ld      hl,devtbl      ; just pt to the table
0067'  C9          ret

;      ptdrv  point to drive table
;      returns address of dph (disk) table

0068'  21 0000*      ptdrv: ld      hl,drvtbl      ; just pt to the table
006B'  C9          ret

          page

```

```

;      the following represents the special area in page 0
;      of bank 1 (the tpa) that allows programs to reference
;      information about the rom and bios

;      This information is initialized during cboot and then
;      moved to 40h bank 1 during every wboot.

006C'      dseg      ; ok to keep it over there

;      (1)      for the bios

005F"  3191      vers:  dw      version      ; version number of this bios
0061"  02        db      2          ; product code
0062"  00        db      0          ; feature code

;      (2)      same for the rom

0063"      rvers:  ds       4          ; version, product and feature
0067"      ds      8          ; 16 bytes currently available

END

```

Macros:

Symbols:

1100	\$VERSION	FD50	ACTDSK	FD4D	ACTSEC
FD4E	ACTTRK	FD51	ACTTYP	2004	ADRIVE
2008	APROTO	0016*	AUXIN	0037*	AUXIST
003A*	AUXOST	0013*	AUXOUT	2006	BDRIVE
2009	BPROTO	0001*	CBOOT	0006	CI
2000	CLKPTR	FDDA	COMNSP	000A*	CONIN
0007*	CONIST	0034*	CONOST	000D*	CONOUT
0009	COUT	0008	DEC	0004	DEVBAUD
0001	DEVIN	0040*	DEVINI	0002	DEVOUT
0008	DEVSER	0065*	DEVTBL	0010	DEVXACT
FD4A	DMABANK	FD4B	DMADR	0069*	DRVTL
000C	DSKD1	0005	DSKS1	FE1C	DSKSAVE
FE1C	DSKSP	0049*	FLUSH	2355	HERTZ
0019*	HOME	0008	IBM	01EC	IDELAY
0002	IDELLEN	0080	IDEVLEN	00A2	IDEVSHD
0022	IDEVTBL	0020	IDRVLEN	0002	IDRVTL
000A	IDVILEN	00E2	IDVINI	0100	IDVSLN
00EC	IDVSTR	FD52	IEADR	0045	IEINP
003F	IEISTAT	0042	IEDSTAT	0048	IEDUT
01EE	IXED	0040	ISHLEN	0000	IVERS
200A	KBDTBL	0010*	LIST	002E*	L1STST
2002	MEDIA	004C*	MOVE	0046*	MULTIO
0051	PARINP	0054	PAROUT	0048	PISTAT
004E	POSTAT	0064'	PTDEV	0068'	PTDRV
FD57	RDBANK	0028*	READ	0063'	RESERV1
0063'	RESERV2	0002	ROMETX	FD44	ROMMOVE
00FC	ROMPROD	00FE	ROMVER	0001	ROMXON
FD56	RSBANK	005A	RSEC	0063I"	RVERS
000C	SAISTAT	0012	SAOSTAT	FDDA	SAVESP
000F	SBISTAT	0015	SBOSTAT	0031*	SECTRN
001C*	SELSK	0052*	SELMEM	0060	SENDEN
0018	SERAIN	001E	SERAOUT	001B	SERBIN
0021	SERBOUT	0024	SETBAUD	0055*	SETBNK
0025*	SETDMA	0022*	SETSEC	001F*	SETTRK
0000I"	SIGNON	0003	SKEY	0000	SYS
0003	SYSB	FDDC	TBUF	004F*	TIME
005B*	TOROM	005FI"	VERS	3191	VERSION
0004*	WBOOT	0003I'	WBOOT@	002B*	WRITE
005D	WSEC	0001	XEROX	0058*	XMOVE

No Fatal error(s)

title bios1 (excbios1) disk drive tables

```

;*****
;*
;*          EXECUTIVE BIOS          *
;*          for                      *
;*          CP/M 3.XX                *
;*          by                      *
;*
;*          Glenn S. Tenney          *
;*          and                      *
;*          Roger W. Chapman         *
;*
;*          Copyright (c) 1983       *
;*          Osborne Computer Corporation *
;*
;*****

```

```

;      This module is only the various disk parameter
;      tables.

```

```

;      Built-in drives are generated as permanent with
;      directory checksums.

```

```

;      Written by: Glenn S. Tenney, November 1982

```

```

;      Various macros used to generate disk tables for
;      CP/M Plus.

```

```

;      define the 16-drive drive table

```

```

dtbl macro ?list
      local ?n
?n    set     0
      irp     ?drv, (?list)
?n    set     ?n+1
      dw      ?drv
      endm
      if      ?n gt 16
      . ' Too many drives. Max 16 allowed'
      exitm
      endif
      if      ?n lt 16
      rept    (16-?n)
      dw      0
      endm
      endif
      endm

```



```
; defines the disk parameter header
```

```
dph macro ?trans,?dph,?csize,?asize
  local ?csv,?alv
  dw ?trans ; translate (skew) table
  db 0,0,0,0,0,0,0,0,0 ; BDOS scratch area
  db 0 ; media flag
  dw ?dph ; disk parameter block
  if not nul ?csize
  dw ?csv ; checksum vector
  else
  dw 0ffffh ; GENCPM sets checksum vector
  endif
  if not nul ?asize
  dw ?alv ; allocation vector
  else
  dw 0ffffh ; GENCPM sets allocation vector
  endif
  dw 0ffffh,0ffffh,0ffffh ; GENCPM sets: dirbcb, dtabcb, hash alloc'd
  db 0 ; hash bank
  if not nul ?csize
?csv: ds ?csize ; checksum vector
  endif
  if not nul ?asize
?alv: ds ?asize ; allocation vector
  endif
endm
```

```
; defines disk parameter blocks
```

```
dph macro ?psize,?pspt,?trks,?bls,?ndirs,?off,?ncks
  local ?spt,?bsh,?blm,?exm,?dsm,?drm,?al0,?al1,?cks,?psh,?psm
  local ?n
;; physical sector mask and physical sector shift
;;
?psh set 0
?n set ?psize/128
?psm set ?n-1
      rept 8
?n set ?n/2
      if ?n eq 0
        exitm
      endif
?psh set ?psh+1
      endm
?spt set ?pspt*(?psize/128)
?bsh set 3
?n set ?bls/1024
      rept 8
?n set ?n/2
      if ?n eq 0
        exitm
      endif
```

```

?bsh    set    ?bsh + 1
        endm
?blm    set    ?bls/128-1
?size   set    (?trks-?off)*?spt
?dsm    set    ?size/(?bls/128)-1
?exm    set    ?bls/1024
        if    ?dsm gt 255
        if    ?bls eq 1024
        .' Invalid disk size with 1k block size'
        exitm
        endif
?exm    set    ?exm/2
        endm
?exm    set    ?exm-1
?all    set    0
?n      set    ((?ndirs and 7fffh)*32+?bls-1)/?bls
        rept  ?n
?all    set    (?all shr 1) or 8000h
        endm
?al0    set    high ?all
?all    set    low ?all
?drm    set    (?ndirs and 7fffh)-1
        if    not nul ?ncks
?cks    set    ?ncks
        else
?cks    set    (?ndirs and 7fffh)/4+(?ndirs and 8000h)
        endif
;;      all computed finally
dw      ?spt          ; 128 byte sectors per track
db      ?bsh,?blm     ; block shift and mask
db      ?exm          ; extent mask
dw      ?dsm          ; maximum block number
dw      ?drm          ; maximum directory number
db      ?al0,?all     ; alloc vector for directory
dw      ?cks          ; checksum size
dw      ?off          ; number of reserved tracks
db      ?psh,?psm     ; physical sector size and mask
        endm

;
;
;      include EXCBIOS.LIB
.list

page

```

```

; declare the drive table

public drvtbl      ; table of drives
public dpbtbl      ; our dpb table w/ prefix

public dph0,dph1   ; Built-in drive's dph's

extrn stdini,stdsel,stdrd,stdwt ; built-in std routines

0000'              cseg          ; must be in common

0000'              drvtbl: dtbl   (dph0,dph1)
0000' 0029'        +            dw    dph0
0002' 004B'        +            dw    dph1
0004' 0000'        +            dw    0
0005' 0000'        +            dw    0
0008' 0000'        +            dw    0
000A' 0000'        +            dw    0
000C' 0000'        +            dw    0
000E' 0000'        +            dw    0
0010' 0000'        +            dw    0
0012' 0000'        +            dw    0
0014' 0000'        +            dw    0
0016' 0000'        +            dw    0
0018' 0000'        +            dw    0
001A' 0000'        +            dw    0
001C' 0000'        +            dw    0
001E' 0000'        +            dw    0

; Note: With disk change in the interrupt routine, the dph must be
; in common memory.

; every dph MUST have a prefix as follows:
; -9    dw    init-routine (at cboot once only)
; -7    dw    read-routine
; -5    dw    write-routine
; -3    dw    seldisk-routine
; -1    db    controller-relative drive number
; +0    dph    --- the dph itself ---

; see the stdsel, stdrd and stdwt routines
; for documentation

; drive 0

0020' 0000*        dw    stdini    ; standard init routine
0022' 0000*        dw    stdrd     ; standard read routine

```

```

0024' 0000*          dw stdwt          ; standard write routine
0025' 0000*          dw stdsel         ; standard seldisk routine
0028' 00              db 0              ; controller drive # 0

; dph0: dph oddxlt, dpbd1, 16, 48 ; use Osborne dd for now (as max)
; but, let genccpm alloc space
0029'          dph0: dph oddxlt, dpbd1 ; use Osborne dd for now (as max)
0029' 0000          dw oddxlt          ; translate (skew) table
002B' 00 00 00 00    + db 0,0,0,0,0,0,0,0,0 ; BDOS scratch area
002F' 00 00 00 00    +
0033' 00            +
0034' 00            + db 0              ; media flag
0035' 0068'          + dw dpbd1         ; disk parameter block
0037' FFFE          + dw 0ffffh        ; GENCCPM sets checksum vector
0039' FFFE          + dw 0ffffh        ; GENCCPM sets allocation vector
003B' FFFE FFFE      + dw 0ffffh,0ffffh,0ffffh ; GENCCPM sets: dirbcb, dtabcb, hash alloc'd
003F' FFFE          +
0041' 00            + db 0              ; hash bank

```

; drive 1

```

0042' 0000*          dw stdini         ; standard init routine
0044' 0000*          dw stdrd          ; standard read routine
0046' 0000*          dw stdwt          ; standard write routine
0048' 0000*          dw stdsel         ; standard seldisk routine
004A' 01              db 1              ; controller drive # 1

; dph1: dph oddxlt, dpbd1, 16, 48
004B'          dph1: dph oddxlt, dpbd1
004B' 0000          dw oddxlt          ; translate (skew) table
004D' 00 00 00 00    + db 0,0,0,0,0,0,0,0,0 ; BDOS scratch area
0051' 00 00 00 00    +
0055' 00            +
0056' 00            + db 0              ; media flag
0057' 0068'          + dw dpbd1         ; disk parameter block
0059' FFFE          + dw 0ffffh        ; GENCCPM sets checksum vector
005B' FFFE          + dw 0ffffh        ; GENCCPM sets allocation vector
005D' FFFE FFFE      + dw 0ffffh,0ffffh,0ffffh ; GENCCPM sets: dirbcb, dtabcb, hash alloc'd
0061' FFFE          +
0063' 00            + db 0              ; hash bank

```

```

0064'          cseg          ; must be in common

```

```

0064'          dpbtbl:      ; each dpb is preceeded by
; associated device type code
; then sectors per track (actual)
; then addr of skew table (or 0)

```

```

; the table is terminated by an entry
; with a sectors/trk of 0ffh

```

```

;      db      disk-type-code
;      db      sectors-per-track
;      dw      skew-table
; lbl: dpb      psize , pspt , trks , bls , ndirs , off , ncks

0064' 0C          db      dskd1          ; Osborne ssdd
0065' 05          db      5
0066' 0000        dw      oddxlt
0068'          dpbd1: dpb      1024,5,40,1024,64+8000h,3
0068' 0028        +      dw      ..0005      ; 128 byte sectors per track
006A' 03 07        +      db      ..0006,..0007 ; block shift and mask
006C' 00          +      db      ..0008      ; extent mask
006D' 00B8        +      dw      ..0009      ; maximum block number
006F' 003F        +      dw      ..000A      ; maximum directory number
0071' C0 00        +      db      ..000B,..000C ; alloc vector for directory
0073' 0010        +      dw      ..000D      ; checksum size
0075' 0003        +      dw      3           ; number of reserved tracks
0077' 03 07        +      db      ..000E,..000F ; physical sector size and mask

0079' 05          db      dskd1          ; Osborne ssdd
007A' 0A          db      10
007B' 0000"        dw      osdxlt
007D'          dpbs1: dpb      256,10,40,2048,64+8000h,3
007D' 0014        +      dw      ..0011      ; 128 byte sectors per track
007F' 04 0F        +      db      ..0012,..0013 ; block shift and mask
0081' 01          +      db      ..0014      ; extent mask
0082' 002D        +      dw      ..0015      ; maximum block number
0084' 003F        +      dw      ..0016      ; maximum directory number
0086' 80 00        +      db      ..0017,..0018 ; alloc vector for directory
0088' 8010        +      dw      ..0019      ; checksum size
008A' 0003        +      dw      3           ; number of reserved tracks
008C' 01 01        +      db      ..001A,..001B ; physical sector size and mask

008E' 01          db      xerox          ; Xerox
008F' 12          db      18
0090' 000A"        dw      xrxxtlt
0092'          dpbx0: dpb      128,18,40,1024,32+8000h,3
0092' 0012        +      dw      ..001D      ; 128 byte sectors per track
0094' 03 07        +      db      ..001E,..001F ; block shift and mask
0096' 00          +      db      ..0020      ; extent mask
0097' 0052        +      dw      ..0021      ; maximum block number
0099' 001F        +      dw      ..0022      ; maximum directory number
009B' 80 00        +      db      ..0023,..0024 ; alloc vector for directory
009D' 8008        +      dw      ..0025      ; checksum size
009F' 0003        +      dw      3           ; number of reserved tracks
00A1' 00 00        +      db      ..0026,..0027 ; physical sector size and mask

00A3' 08          db      ibm            ; IBM
00A4' 08          db      8
00A5' 0000        dw      ibmxtlt
00A7'          dpbibm: dpb      512,8,40,1024,64+8000h,1
00A7' 0020        +      dw      ..0029      ; 128 byte sectors per track
00A9' 03 07        +      db      ..002A,..002B ; block shift and mask
00AB' 00          +      db      ..002C      ; extent mask
00AC' 009B        +      dw      ..002D      ; maximum block number
00AE' 003F        +      dw      ..002E      ; maximum directory number

```



```

00B0' C0 00      +      db      ..002F,..0030      ; alloc vector for directory
00B2' 8010      +      dw      ..0031              ; checksum size
00B4' 0001      +      dw      1                  ; number of reserved tracks
00B6' 02 03      +      db      ..0032,..0033      ; physical sector size and mask

00B8' 08              db      dec                  ; DEC
00B9' 09              db      9
00BA' 001C"        dw      decxlt
00BC'              dpbdec: dpb      512,9,40,1024,64+8000h,2
00BC' 0024      +      dw      ..0035              ; 128 byte sectors per track
00BE' 03 07      +      db      ..0036,..0037      ; block shift and mask
00C0' 00          +      db      ..0038              ; extent mask
00C1' 00AA      +      dw      ..0039              ; maximum block number
00C3' 003F      +      dw      ..003A              ; maximum directory number
00C5' C0 00      +      db      ..003B,..003C      ; alloc vector for directory
00C7' 8010      +      dw      ..003D              ; checksum size
00C9' 0002      +      dw      2                  ; number of reserved tracks
00CB' 02 03      +      db      ..003E,..003F      ; physical sector size and mask

00CD' 00              db      0                  ; 8" ssdd
00CE' FF          db      0ffh      ; end of table 26
00CF' 0025"        dw      stdxlt
00D1'              ibm0: dpb      128,26,77,1024,64+8000h,2
00D1' 001A      +      dw      ..0041              ; 128 byte sectors per track
00D3' 03 07      +      db      ..0042,..0043      ; block shift and mask
00D5' 00          +      db      ..0044              ; extent mask
00D6' 00F2      +      dw      ..0045              ; maximum block number
00D8' 003F      +      dw      ..0046              ; maximum directory number
00DA' C0 00      +      db      ..0047,..0048      ; alloc vector for directory
00DC' 8010      +      dw      ..0049              ; checksum size
00DE' 0002      +      dw      2                  ; number of reserved tracks
00E0' 00 00      +      db      ..004A,..004B      ; physical sector size and mask

00E2'              dseg              ; ok in bank 0

;      these are the actual skew tables

0000              oddxlt equ      0      ; no skewing needed

0000" 00 02 04 06      osdxlt: db      0,2,4,6,8,1,3,5,7,9
0004" 08 01 03 05
0008" 07 09

000A" 00 05 0A 0F      xrxxt: db      0,5,10,15      ; physical equals logical
000E" 02 07 0C 11      db      2,7,12,17
0012" 04 09 0E          db      4,9,14
0015" 01 06 0B 10      db      1,6,11,16
0019" 03 08 0D          db      3,8,13

0000              ibmxt equ      0      ; no skewing needed

001C" 00 02 04 06      decxlt: db      0,2,4,6,8,1,3,5,7
0020" 08 01 03 05
0024" 07

```

```
0025" 00 06 0C 12      stdxlt: db    0, 6, 12, 18, 24, 4, 10, 16, 22, 2, 8, 14, 20
0029" 18 04 0A 10
002D" 16 02 08 0E
0031" 14
0032" 01 07 0D 13      db    1, 7, 13, 19, 25, 5, 11, 17, 23, 3, 9, 15, 21
0036" 19 05 0B 11
003A" 17 03 09 0F
003E" 15
```

end

Macros:

DPB	DPH	DTBL
-----	-----	------

Symbols:

0002	..0000	0028	..0005	0003	..0006
0007	..0007	0000	..0008	0008	..0009
003F	..000A	00C0	..000B	0000	..000C
0010	..000D	0003	..000E	0007	..000F
0002	..0010	0014	..0011	0004	..0012
000F	..0013	0001	..0014	002D	..0015
003F	..0016	0080	..0017	0000	..0018
0010	..0019	0001	..001A	0001	..001B
0001	..001C	0012	..001D	0003	..001E
0007	..001F	0000	..0020	0052	..0021
001F	..0022	0080	..0023	0000	..0024
0008	..0025	0000	..0026	0000	..0027
0001	..0028	0020	..0029	0003	..002A
0007	..002B	0000	..002C	009B	..002D
003F	..002E	00C0	..002F	0000	..0030
0010	..0031	0002	..0032	0003	..0033
0002	..0034	0024	..0035	0003	..0036
0007	..0037	0000	..0038	00AA	..0039
003F	..003A	00C0	..003B	0000	..003C
0010	..003D	0002	..003E	0003	..003F
0002	..0040	001A	..0041	0003	..0042
0007	..0043	0000	..0044	00F2	..0045
003F	..0046	00C0	..0047	0000	..0048
0010	..0049	0000	..004A	0000	..004B
0002	..004C	079E	?SIZE	FD50	ACTDSK
FD4D	ACTSEC	FD4E	ACTTRK	FD51	ACTTYP
2004	ADRIVE	2008	APROTO	2006	BDRIVE
2009	BPROTO	0006	CI	2000	CLKPTR
FDDA	COMNSP	0009	COUT	0008	DEC
001C"	DECTL	0004	DEVBAUD	0001	DEVIN
0002	DEVOUT	0008	DEVSER	0010	DEVXACT
FD4A	DMABANK	FD4B	DMADR	0068'	DPBD1
00BC'	DPBDEC	00A7'	DPBIBM	007D'	DPBS1
0064I'	DPBTBL	0092'	DPBX0	0029I'	DPH0
004BI'	DPH1	0000I'	DRVTBL	000C	DSKD1
0005	DSKS1	FE1C	DSKSAVE	FE1C	DSKSP
2355	HERTZ	0008	IBM	00D1'	IBM0
0000	IBMXLT	01EC	IDELAY	0002	IDELLEN
0080	IDEVLEN	00A2	IDEVSHD	0022	IDEVTBL
0020	IDRVLEN	0002	IDRVTBL	000A	IDVILEN
00E2	IDVINI	0100	IDVSLEN	00EC	IDVSTR
FD52	IEADR	0045	IEINP	003F	IEISTAT
0042	IEQSTAT	0048	IEOUT	01EE	IEXD
0040	ISHDLEN	0000	IVERS	200A	KBDTBL
2002	MEDIA	0000	ODDXLT	0000"	OSDXLT
0051	PARINP	0054	PARDUT	004B	PISTAT
004E	POSTAT	FD57	RDBANK	0002	ROMETX
FD44	ROMMOVE	00FC	ROMPROD	00FE	ROMVER
0001	ROMXON	FD56	RSBANK	005A	RSEC
000C	SAISTAT	0012	SAOSTAT	FDDA	SAVESP
000F	SBISTAT	0015	SBOSTAT	0060	SENDEN
0018	SERAIN	001E	SERRAOUT	001B	SERBIN

0021	SERBOUT	0024	SETBAUD	0003	SKEY
0042*	STDINI	0044*	STDRD	0048*	STDSEL
0046*	STDWT	0025"	STDHLT	0000	SYS
0003	SYSB	FDDC	TBUF	005D	WSEC
0001	XEROX	000A"	XRXHLT		

No Fatal error(s)

.z80

title bios2 (excbios2) serial i/o tables & routines

```
; 1983-11-07   Changed initial configuration to no printer assigned
;             Change made in initvec
;
; 1983-23-06   Made device time-out delay a SETUP definable value,
;             can disable timeout feature, user prompt will go away and
;             restore original screen/cursor, message will continue to
;             check for device ready
;
```

```
*****
;*             *
;*   EXECUTIVE BIOS   *
;*       for         *
;*   CP/M 3.XX       *
;*             *
;*       by          *
;*             *
;*   Glenn S. Tenney  *
;*       and         *
;*   Roger W. Chapman *
;*             *
;*   Copyright (c) 1983 *
;*   Osborne Computer Corporation *
;*             *
;*****
```

```
;   This module handles all of the non-disk i/o.
;   This supports full re-direction, xon/xoff, etx/ack
;   and IEEE devices.
```

.list

```
public devtbl      ; the device table

public devini      ; device init
public conout,auxout,list ; output routines
public conost,auxost,listst ; output status
public conin,auxin  ; input routines
public conist,auxist ; input status
public initvec     ; initial i/o redirection vector
public vecptr      ; ptr to crnt vector wd
PUBLIC DELAY       ;Device delay time-out value
PUBLIC DELAY0
PUBLIC OUTST       ;Output device ready status

extrn @covec,@aovec,@lovec ; re-direction
extrn @civec,@aivec        ; of serial i/o
EXTRN SERNR                ;Not ready routine
```

```

;      special equates for this routine only

0003      etx      equ      03h          ; the etx char sent for etx/ack
0006      ack      equ      06h          ; the ack to our etx
0011      xon      equ      11h          ; xon lets output continue
0013      xoff     equ      13h          ; xoff pauses output 'til an xon

0004      xonact   equ      4            ; (devxact) 1=xon (or etx) active
;      (bit number)

;      we have a serial shadow table
;      these are the meanings of the bits in the flag byte
;      (these are all bit numbers)

0007      xonetx   equ      7            ; (80h) 1=etx/ack 0=xon/xoff
0006      waiting  equ      6            ; (40h) 1=waiting for ack/xon 0=not waiting
0005      isieee   equ      5            ; (20h) 1=is an ieee device

;      serial device vectoring request codes

0000      vecist   equ      0            ; input status
0001      vecin    equ      1            ; input a char
0002      vecost   equ      2            ; output status
0003      vecout   equ      3            ; output a char

;      serial device output delay counter before
;      issuing device not ready message

B000      FIVESEC  EQU      0B000h      ; about 5 seconds

```

```

0000'                                cseg                                ; common

;
;   todseg
;
;   call a routine in bank 80h (0 + rom)
;   hl= routine to jp to in bank 80h
;   a= destroyed
;
;   all of the bios routines gain control in any bank
;   and to conserve bank 1 space have most of the actual
;   routine in bank 0. The sp is switched too.

0000' ED 73 FDDA      todseg: ld      (save sp),sp      ; (banks changing, sp has to)
0004' 31 FDDA          ld      sp,comnsp      ; (stack into common area)
0007' DB 00           in       a,(sys)        ; (save current bank)
0009' F5             push     af
000A' 3E 80           ld      a,80h          ; (and set bank 0 + rom)
000C' D3 00           out      (sys),a

000E' CD 0000"        call     setvec        ; off to it after getting ready

;   return here when done

0011' E1             pop      hl            ; h=saved bank #
0012' 6F             ld      l,a            ; save result to pass back
0013' 7C             ld      a,h            ; get saved bank
0014' D3 00           out      (sys),a       ; reset bank # (flags still ok)
0016' 7D             ld      a,l            ; restore a (result)
0017' ED 7B FDDA      ld      sp,(save sp)   ; and restore sp too
001B' C9             ret                    ; now all done

001C'                                dseg

;   finally go off to routine pointed to in hl
;   setting up for re-direction vectors

0000" E5             setvec: push hl          ; hl=routine address
0001" EB             ex       de,hl          ; get ptr to vector
0002" 22 000C"        ld      (vecptr),hl    ; save in case we have to unassign
0005" 7E             ld      a,(hl)         ; then get the vector itself
0006" 23             inc      hl
0007" 66             ld      h,(hl)
0008" 6F             ld      l,a            ; hl=i/o redirection vector
0009" AF             xor      a             ; clear carry
000A" 47             ld      b,a            ; and device # for redirecting
000B" C9             ret                    ; off to routine in hl originally

000C"                vecptr: ds      2      ; ptr to crnt vector

```

000E"

cseg

page

```

; devini
;
; perform serial device initialization
; there are 2 kinds of init: (1) viavec which
; can init onboard hardware and (2) sndstr which
; sends the init string of characters for this
; device. This routine is coded the way it is
; for the sake of cboot (see there for details).
;

001C' devini: ; perform serial device init
001C' 21 000E' ld hl,init ; pt to init routine in bank 80h
001F' C3 0000' jp todseg ; call that routine in bank 0

0022' dseg ; following routine in bank 0

; init
;
; set this device to 'requires init'
; then call its init routine
; when calling the real init routine, regs must be the
; same as when calling the vecio routine (b=device,
; de->flag byte)
;

000E" init:
000E" C5 push bc ; save incoming device # (in c)

000F" 79 ld a,c ; get device #
0010" E6 0F and 0fh ; force to 0 - 15

0012" 87 add a,a ; #4
0013" 87 add a,a

0014" 5F ld e,a ; and into 16 bits
0015" 16 00 ld d,0

0017" 21 020F" ld hl,viareal+2 ; pt to real table
001A" 19 add hl,de
001B" EB ex de,hl

001C" 01 0359" ld bc,viadev ; and to active table
001F" 09 add hl,bc ; hl->active de->real

0020" 01 003F" ld bc,vec1st ; pt to our 1st time code
0023" 71 ld (hl),c ; active table pts to 1st time
0024" 23 inc hl
0025" 70 ld (hl),b
0026" 23 inc hl

0027" 1A ld a,(de) ; then copy flag bytes fm real
0028" 77 ld (hl),a ; to active

0029" E5 push hl ; save ptr to active flag byte

```

```

002A" 13      inc    de
002B" 23      inc    hl
002C" 1A      ld     a,(de)
002D" 77      ld     (hl),a

002E" 1B      dec    de      ; back up to rtn addr
002F" 1B      dec    de

0030" 1A      ld     a,(de)      ; hi of rtn ptr
0031" 67      ld     h,a
0032" 1B      dec    de
0033" 1A      ld     a,(de)      ; lo
0034" 6F      ld     l,a      ; hl=real rtn addr

0035" B4      or     h      ; is addr = 0

0036" 11 FFD  ld     de,-3      ; but 1st, back up to init rtn jmp
0039" 19      add    hl,de      ; hl pts to jmp to init rtn

003A" D1      pop    de      ; restore ptr to active table flag byte
003B" C1      pop    bc      ; restore device # too
003C" 41      ld     b,c      ; must pass it in b

003D" C8      ret     z      ; finally, addr=0 -- init all done

003E" E9      jp     (hl)      ; off to rtn pt'd to in hl
                                ; (it will ret)

```

```

;      each active vector initially points to the following
;      1st time code. When the first activity occurs for that
;      device, this code is executed. It is at this time that
;      the initial string is sent out to this device.

```

```

003F"          vecist:      ;      de->flag byte of active table

003F" F5      push    af      ; save fcn type (may be needed)
0040" D5      push    de      ; save ptr to flags for real rtn
0041" C5      push    bc      ; save device # and char

0042" 21 FFE  ld     hl,-2
0045" 19      add    hl,de      ; hl=start of entry in viadev

0046" EB      ex     de,hl      ; hl=viadev+2+(dev#*4)
0047" 01 035B ld     bc,viadev+2 ; bc=viadev+2
004A" AF      xor     a
004B" ED 42   sbc     hl,bc      ; hl-bc = offset fm viadev
004D" 01 020D ld     bc,viareal
0050" 09      add    hl,bc      ; hl->real de->active

0051" 7E      ld     a,(hl)      ; copy real address in
0052" 12      ld     (de),a
0053" 4F      ld     c,a      ; and save the addr too
0054" 13      inc    de

```

```

0055" 23          inc    hl
0056" 7E          ld     a,(hl)
0057" 12          ld     (de),a
0058" 47          ld     b,a          ; bc=copy of addr (now in viadev)

0059" D1          pop     de          ; restore incoming bc
005A" E1          pop     hl          ; and incoming de
005B" F1          pop     af          ; and fcn code

005C" C5          push    bc          ; real rtn addr (ret to there)

005D" F5          push    af          ; incoming fcn code
005E" E5          push    hl          ; real de (as on entry)
005F" D5          push    de          ; re-save real bc

0060" 21 0257"    ld     hl,initstr    ; pt to table of strings

0063" AF          xor     a          ; clear it
0064" 42          ld     b,d          ; get device # for loop counter
0065" 57          ld     d,a          ; to use de as 16 bit value
0066" B0          or     b          ; is device # 0 ?
0067" 20 05       jr     z,sndfnd      ; yes, then hl pts to string

0069"            sndflp:              ; loop here til at start of string
0069" 5E          ld     e,(hl)        ; get length byte crnt string
006A" 23          inc     hl          ; bump over length byte
006B" 19          add     hl,de        ; then bump over string itself
006C" 10 FB       djnz    sndflp      ; continue thru strings

006E"            sndfnd:              ; hl->length byte of string
006E" C1          pop     bc          ; restore b=device #
006F" C5          push    bc          ; re-save it for when done

0070" 7E          ld     a,(hl)        ; get starting length
0071" 3C          inc     a          ; dummy up for the dec following

0072" 3D          sndlp: dec     a          ; drop count until 0
0073" 20 04       jr     nz,sndmor     ; not done yet, send more

0075" C1          snddn: pop     bc          ; restore incoming dev # + char
0076" D1          pop     de          ; and de
0077" F1          pop     af          ; restore, may be fcn type
0078" C9          ret                ; off to real device driver now

0079" F5          sndmor: push    af          ; save crnt length
007A" 23          inc     hl          ; pt to next char
007B" E5          push    hl          ; save string ptr too

007C" CD 00CA"    call    outwait      ; wait for device to get ready
007F" E1          pop     hl          ; restore string ptr
0080" 20 0D       jr     z,snduna      ; it got unassigned — oops

0082" E5          push    hl          ; and save it back
0083" 4E          ld     c,(hl)        ; get next char to send it out
0084" 3E 03       ld     a,vecout      ; to output a char device in b

```



```

;          note that viavec is called by us and we were
;          called by viavec. This is ok, since we reset
;          viadev to pt to the real routine address

0086" C5          push    bc          ; save device #
0087" CD 0180"    call    viavec
008A" C1          pop     bc          ; restore device #
008B" E1          pop     hl
008C" F1          pop     af

008D" 18 E3          jr     sndlp      ; keep going, 'til 0 that is

008F" E1          snduna: pop    hl      ; dummy to dump af saved on stack
0090" 18 E3          jr     snddn      ; and all done sending string

0092"          cseg          ;          back to bank 1
          page

```

```

;      conout, auxout and list
;      output the character to all such devices

0022' 11 0000# list: ld de,@lovec ; list re-direction
0025' C3 0031' jp doout ; output to those devices

0028' 11 0000# auxout: ld de,@aovec ; auxout re-direction
002B' C3 0031' jp doout

002E' 11 0000# conout: ld de,@covec ; conout re-direction word

0031' 21 0092" doout: ld hl,output ; pt to common output routine
0034' C3 0000' jp todseg ; and off to it in bank 80h

0037' dseg

0092" output: ; common output routine
0092" outnext:
0092" B4 or h ; leftmost bit = 1 ?
0093" F2 00A9" jp p,outnxt2 ; no, try next device
0096" E5 push hl ; yes, send char to that device

0097" CD 00CA" call outwait ; wait until device ready
009A" 28 0C jr z,outnxt1 ; told to ignore this char

009C" C5 push bc ; back to send it now
009D" 3E 03 ld a,vecout ; say we want to vector output
009F" CD 0100" call viavec ; do that output

;      if we sent a cr and etx/ack is enabled, then send the etx

;      is this the cr (this is faster, so is 1st)
00A2" C1 pop bc ; restore character
00A3" 79 ld a,c ; get char sent
00A4" FE 0D cp 0dh ; is it cr?
00A6" 28 08 jr z,gotcr ; yes--check for etx this char

00A8" outnxt1: ; restore hl before advancing to next
00A8" E1 pop hl
00A9" outnxt2: ; actually advance to next device
00A9" 04 inc b ; next device #
00AA" AF xor a ; clear a and carry
00AB" ED 6A adc hl,hl ; shift left and see if 0
00AD" 20 E3 jr nz,outnext ; some other device wanted
00AF" C9 ret ; all done

;      on the cr, see if xon/etx (either) is active

00B0" CD 0134" gotcr: call chkact ; is xon active (could be xon or etx)
00B3" 28 F3 jr z,outnxt1 ; no, done this device

```

```
00B5"  CB 7E          bit    xonetc, (hl)    ; ok, now is it etc that is active
00B7"  2B EF          jr     z, outnxt1      ; no (after all that) done this device
```

```
;      we finally know that we just sent a cr to a device with
;      etc/ack protocol active, so send the etc
```

```
00B9"          outetc:
```

```
00B9"  CD 00CA"       call   outwait        ; wait til ready for etc
00BC"  2B EA          jr     z, outnxt1      ; ignore this character

00BE"  C5            push   bc              ; save original char (and dev #)
00BF"  0E 03         ld     c, etc          ; ask for an ack
00C1"  3E 03         ld     a, vecout       ; say we want to vector output
00C3"  CD 0180"       call   viavec         ; do that output
00C6"  C1            pop     bc             ; restore char and dev #

00C7"  1B DF          jr     outnxt1        ; proceed to next device
```

page

```

00C9" 01          DELAY0:      DB      1          ;For 5 sec delay time

;      outwait
;      ;
;      ;      common routine to wait for output device
;      ;      to become ready to send a char
;      ;      this routine will go off to the error routine
;      ;      after a delay
;      ;      returns z if output to be ignored

00CA"          outwait:
00CA" 3A 0358"      ld      a,(delay0)          ;use delay loop and message?
00CD" B7          or      a
00CE" 20 08          jr      nz,outloop          ;yes, go to delay loop

00D0" C5          wait:  push  bc              ;no, try forever
00D1" CD 0147"      call  outst
00D4" C1          pop   bc
00D5" 2B F9          jr   z,wait              ;if device not ready, try again
00D7" C9          ret                        ;else return

00D8"          outloop:
00D8" 3A 00C9"      ld      a,(delay0)          ;get number of times to loop
00DB" 5F          ld      e,a

00DC"          deloop:
00DC" 21 B000      ld      hl,fivsec

00DF" E5          outwlp: push  hl
00E0" D5          push  de
00E1" C5          push  bc              ; save char while checking

00E2" CD 0147"      call  outst              ; is that device ready?

00E5" C1          pop   bc              ; restore each time
00E6" D1          pop   de
00E7" E1          pop   hl

00E8" C0          ret      nz              ; all done, device ready

00E9" 2B          dec     hl              ; hmm not ready yet
00EA" 7C          ld      a,h
00EB" B5          or      l              ; did we count down to zero
00EC" 20 F1          jr      nz,outwlp      ; no, keep waiting

00EE" 1D          dec     e
00EF" 20 EB          jr      nz,deloop

00F1" CD 0000"      call  sernr          ; yes, tell operator and ask
00F4" 20 E2          jr      nz,outloop      ; retry again and again

;      we have been told to unassign this device

00F6" C5          push   bc              ; save device #
00F7" 2A 000C"      ld     hl,(vecptr)      ; pt to redirection vector

```

```

00FA" 78          ld    a,b          ; get device #
00FB" D6 08       sub    8           ; convert to 0-7
00FD" 30 02       jr     nc,unalo    ; use lo byte
00FF" 23          inc    hl          ; pt to hi byte
0100" 78          ld     a,b         ; get device back (it is 0-7)
0101" 47          unalo: ld    b,a    ; b=loop count
0102" 04          inc    b           ; adjust for 1 indexed
0103" AF          xor    a           ; clear a
0104" 37          scf              ; and set carry

0105" 1F          unalp: rra          ; dev # 0-7 into mask 80h-01h
0106" 10 FD       djnz   unalp
0108" 2F          cpl              ; make a bit into a mask
0109" A6          and    (hl)        ; turns that assignment off
010A" 77          ld     (hl),a      ; device now unassigned

010B" 2A 000C"    ld     hl,(vecptr) ; point to it again
010E" 7E          ld     a,(hl)      ; lo
010F" 23          inc    hl
0110" B6          or     (hl)        ; see if it went to 0000
0111" 20 0A       jr     nz,unardy   ; no, then it is unassigned ok

0113" 01 0000*    ld     bc,@covec   ; see if it is conout
0116" 03          inc    bc          ; pt to high byte of vector word
0117" 79          ld     a,c         ; check only low byte of address
0118" BD          cp     1           ; is this the one being unassigned?
0119" 20 02       jr     nz,unardy   ; no, ok to be set to 0000h

; conout cannot be unassigned to 0000h, force to something
011B" 36 00       ld     (hl),00h    ; force to device 0 (crt)

011D" AF          unardy: xor    a    ; show unassigned and to be ignored
011E" C1          pop    bc          ; restore device #
011F" C9          ret              ; otherwise ignore char (z set)

0120"            cseg

page

```

```

;      conost, auxost, listst
;      same as output, just check status

0037' 11 0000*      listst: ld      de,@lovec
003A' C3 0046'      JP      doostat      ; common rtn to chk status

003D' 11 0000*      auxost: ld      de,@aovec
0040' C3 0046'      JP      doostat

0043' 11 0000*      conost: ld      de,@covec      ; re-direction word

0046' doostat:      ; here to call common output status rtn
0046' 21 0120"      ld      hl,outstat      ; pt to common routine
0049' C3 0000'      JP      todseg      ; call it in bank 80h

004C'      dseg      ; into bank 0

;      outstat
;      common output status routine

0120" outstat:      ; de=redirection vector

0120" ostnext:
0120" B4      or      h      ; leftmost bit = 1 ?
0121" F2 012C" JP      p,ostnxt1      ; no, try next device
0124" E5      push    hl      ; yes, send char to that device
0125" C5      push    bc      ; save regs 1st
0126" CD 0147" call    outst      ; is that device ready?
0129" C1      pop     bc      ; but first restore regs
012A" E1      pop     hl
012B" C8      ret      z      ; exit NOW if any not ready

012C" ostnxt1:
012C" 04      inc     b      ; next device #
012D" AF      xor     a      ; clear a and carry
012E" ED 6A      adc     hl,hl      ; shift left and see if 0
0130" 20 EE      jr     nz,ostnext      ; some other device wanted
0132" 3D      dec     a      ; make a=0ffh to show ready
0133" C9      ret      ; and all done

```

page

```

;          a couple of common outst routines

;          chkact
;          sees if xon (or etx) active this device
;          z=no  hl=ptr to flag byte in shadow table

0134" 68      chkact: ld    l,b          ; get device #
0135" 26 00      ld    h,0          ; into 16 bits
0137" 29      add    hl,hl          ; * 8 (offset devtbl)
0138" 29      add    hl,hl
0139" E5      push   hl            ; save #4 for shadow tbl access
013A" 29      add    hl,hl
013B" 11 0072'   ld    de,devtbl+6   ; the mode byte
013E" 19      add    hl,de          ; this device's mode byte
013F" CB 66      bit    xonact,(hl)   ; is xon (or etx) active
0141" E1      pop    hl            ; and restore so hl=device#
0142" 11 035B"   ld    de,viadev+2   ; pt to shadow table flag byte
0145" 19      add    hl,de          ; hl pts to flag byte
0146" C9      ret                ; flag and hl set

;          outst
;          common output status routine
;          b=device #, return a=z if not ready, ff if ready

0147"      outst:                ; bios no longer does xon/etx, rom does

0147"      outst1:               ; get real output device status
0147" 3E 02      ld    a,vecost       ; as for output status
0149" C3 0180"   jp    viavec        ; for this device
014C"      cseg                ; back to bank 1

page

```

```

;      conist, auxist
;      same as output, just check status

004C' 11 0000#      auxist: ld      de,@aivec
004F' C3 0055'      jp      doist      ; common rtn to chk status

0052' 11 0000#      conist: ld      de,@civec      ; re-direction word

0055' 21 014C"      doist: ld      hl,instat      ; pt to common input status rtn
0058' C3 0000'      jp      todseg      ; call it in bank 0

005B'              dseg

014C"              instat:              ; common input status routine

014C"              istnext:
014C" B4              or      h              ; leftmost bit = 1 ?
014D" F2 0154"      jp      p,istnxt1      ; no--check next device
0150" CD 015D"      call     ist              ; yes, is that device ready
0153" C8              ret      z              ; exit NOW if any not ready

0154"              istnxt1:
0154" 04              inc      b              ; next device #
0155" AF              xor      a              ; clear a and carry
0156" ED 6A          adc      hl,hl          ; shift left and see if 0
0158" C2 014C"      jp      nz,istnext      ; some other device wanted
015B" 3D              dec      a              ; a=ff, all were ready
015C" C9              ret                  ; and all done

;      ist
;      input status      b=device #
;;      save bc,hl because used during xon/off checking

015D"              ist:
015D" E5              push     hl              ; save regs
015E" C5              push     bc
015F" 3E 00          ld      a,vecist      ; ask for input status
0161" CD 0180"      call     viavec      ; vector it now
0164" C1              pop      bc
0165" E1              pop      hl
0166" C9              ret                  ; and done

0167"              cseg              ; back to bank 1

page

```



```

;      conin, auxin
;      like in status, but get first avail character

005B'  11 0000#      auxin: ld      de,@aivec
005E'  C3 0064'      jp      doin          ; common rtn to chk status

0061'  11 0000#      conin: ld      de,@civec      ; re-direction word

0064'  21 0167"      doin:  ld      hl,input      ; pt to common input routine
0067'  C3 0000'      jp      todseg      ; call it in bank 0

006A'                      dseg          ; following in bank 0

0167"                      input:          ; common input routine

0167"  E5            inwait: push    hl          ; save re-direction vector
0168"  AF                      xor      a          ; a=0 for inner loop
0169"  47                      ld      b,a          ; b=device # (0-15)
016A"                      innext:
016A"  B4                      or      h          ; leftmost bit = 1 ?
016B"  F2 0173"          jp      p,innxt1      ; no, try next device
016E"  CD 015D"          call    ist          ; yes, is that device ready
0171"  20 0A            jr      nz,inready      ; yes, go get that character
0173"  04            innxt1: inc      b          ; next device #
0174"  AF                      xor      a          ; clear a and carry
0175"  ED 6A            adc      hl,hl          ; shift left and see if 0
0177"  C2 016A"          jp      nz,innext      ; some other device wanted
017A"  E1            pop      hl          ; restore re-direction vector
017B"  18 EA            jr      inwait      ; loop til someone ready !!

017D"                      inready:
017D"  E1            pop      hl          ; drop off vector

;                      and proceed to get the character

017E"                      getin:          ; common routine to get a character
017E"  3E 01            ld      a,vecin      ; ask for input

```

[illegible]

```

; fall through

```

jp viavec ; off to do it

=====

page

```

; viavec
; off to device routine via a vector
; a=type of request (see vecxxx equates)
; b=device #
; goes off to proper routine based on viadev table
;
; viavia as viavec, but de->table to vector thru

0180" viavec: ; off to routine handling this device

0180" 11 0359" ld de,viadev ; addr is 1st 2 bytes of the 4

0183" viavia: ; here when de->table to use to via

0183" 26 00 ld h,0
0185" 68 ld l,b ; hl=device # 16 bits
0185" 29 add hl,hl ; *2
0187" 29 add hl,hl ; *4 (each entry is 4 bytes wide)
0188" 19 add hl,de ; hl points to entry this device

0189" 5E ld e,(hl) ; get routine address
018A" 23 inc hl
018B" 56 ld d,(hl) ; de -> routine for this device
018C" 23 inc hl ; hl -> flags for this device
018D" EB ex de,hl ; de->flags hl->routine
018E" E9 jp (hl) ; off to routine (like a call)

; called routine will do the RET

; the following is what an unused device will vector to
; it is preceeded by the specified prefix which is 'jp'ed to
; to do an init. Please note that it is essentially a 'nop'
; but MUST be there and this way.

018F" C3 0192" jp unused ; 3 byte prefix to a vector routine
0192" AF unused: xor a ; just set a=0 and ret
0193" C9 ret ; used to be part of outst

```

```

;      stdrom
;      standard rom serial i/o driver (no init)

;      because bank 80h is in effect, we can call rom
;      directly

;      a=type of i/o (see vecxxx equates)
;      b=device #
;      de=pointer to flag byte, then rom table offset

;      NOTE:
;      every routine called for i/o via the viadev
;      table must have a 3 byte prefix. This prefix
;      is jp'ed to to do the init, if any, for this
;      device.

```

```

0194"  C3 01AD"          jp  romini          ;      stdrom init

0197"  EB                stdrom: ex  de,hl      ; make ptr to flags be in hl
0198"  23                inc  hl              ; bump over flag byte to table offset

0199"  5E                ld    e,(hl)         ; table offset
019A"  16 00             ld    d,0           ; 16 bits worth
019C"  21 0399"          ld    hl,romtbl      ; pt to std rom table
019F"  19                add    hl,de        ; hl=start of proper rom table
01A0"  5F                ld    e,a           ; get type of request
01A1"  19                add    hl,de        ; hl points to proper rom offset
01A2"  7E                ld    a,(hl)        ; get low byte of rtn addr
01A3"  B7                or     a            ; 00=do nothing (special case)
01A4"  CA 01AB"          jp    z,romnop      ; a nop that always gives a=ff

01A7"  26 01             ld    h,01h        ; h=hi addr (always 01)
01A9"  6F                ld    l,a           ; hl=true rom address
01AA"  E9                jp    (hl)         ; off to rom directly

01AB"  3D                romnop: dec  a       ; show ready (a=ff) always
01AC"  C9                stdret: ret        ; and exit

```

```

;      romini
;      stdrom comes here to init a rom driven
;      device. If it is serial A or B it's baud
;      rate is set. If not, nothing is done.
;      registers same as for stdrom call.

```

```

01AD"          romini:          ; process device init for a rom serial driver

01AD"  13                inc  de            ; pt beyond flag byte
01AE"  1A                ld    a,(de)       ; get rom table offset byte

```



```

01E1" F1          pop    af          ; restore port a or b code
01E2" 5F          ld      e,a        ; cnvrt to 16 bits
01E3" 16 00       ld      d,0
01E5" 21 2000     ld      hl,aproto
01E8" 19          add     hl,de      ; pt to protocol this port

01E9" 36 00       ld      (hl),0    ; assume no protocol at all
01EB" E5          push   hl         ; save proto ptr

01EC" CD 0134"    call    chkact     ; see what protocol if any
01EF" D1          pop     de         ; restore ptr to proto
01F0" C8          ret     z         ; no protocol, all done

01F1" 3E 01       ld      a,romxon   ; assume it is xon
01F3" CB 7E       bit     xonetx,(hl) ; ahhh is it xon or etx
01F5" 20 01       jr      z,erset    ; it is xon, a=proper value
01F7" 3C          inc     a          ; make it show etx proto
01F8" 12          serset: ld      (de),a ; post protocol flag to rom
01F9" C9          ret              ; and all done

```

```

;      stdiee
;
;      standard ieee serial device type i/o
;      this uses the rom routines for simple ieee i/o
;      and may get confused if more than one
;      bios supported ieee device is active.
;      a      type of request (viaxxx)
;      b      device #
;      de     ptr to flag byte (next byte is ieee device addr)

```

```

01FA" C3 01AC"    jp      stdret     ; stdiee init (none)

01FD"            stdiee:
01FD" 6F          ld      l,a        ; convert to 16 bit value
01FE" 26 00       ld      h,0
0200" 13          inc     de         ; byte after flags is ieee addr
0201" 1A          ld      a,(de)     ; get ieee device address
0202" 32 FD52     ld      (ieadr),a  ; let rom know what device
0205" 11 03A9"    ld      de,ieetbl  ; pt to table of routine adrs
0208" 19          add     hl,de      ; hl pts to proper addr

0209" 6E          ld      l,(hl)     ; lo of rom addr
020A" 26 01       ld      h,01h     ; always 100h+

020C" E9          jphl:  jp      (hl) ; let rom do it directly

020D"            cseg

page

```

```

;      devtbl
;      this table defines the valid serial devices
;      by name as well as by functionality.
;      This is the standard DRI table. All 16 entries
;      MUST be here because setup uses this data.

;      this table has a special prefix:
;      dw our-tables-in-bank-0

006A' 0200"      dw      viareal      ; pt to start of ours in bank 0

006C'      devtbl:

006C' 43 52 54 20      db      'CRT '      ; device 0 crt
0070' 20 20
0072' 03 00      db      devin+devout,0 ; i/o device no baud

0074' 43 45 4E 20      db      'CEN '      ; device 1 std printer
0078' 20 20
007A' 03 00      db      devin+devout,0 ; may be i/o no baud

007C' 4D 4F 44 45      db      'MODEM '      ; device 2 is modem port (A)
0080' 4D 20
0082' 0F 00      db      devin+devout+devser+devbaud,0 ; baud rate 0=no set needed

0084' 50 52 4E 54      db      'PRNTR '      ; device 3 is serial printer port (B)
0088' 52 20
008A' 0F 00      db      devin+devout+devser+devbaud,0 ; again, 0=no setbaud needed

008C' 49 45 45 45      db      'IEEE '      ; the ieee-488 port
0090' 20 20
0092' 03 00      db      devin+devout,0

0094' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 05 unused slot fill to 16
0098' 00 00 00 00
009C' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 06 unused slot fill to 16
00A0' 00 00 00 00
00A4' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 07 unused slot fill to 16
00A8' 00 00 00 00
00AC' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 08 unused slot fill to 16
00B0' 00 00 00 00
00B4' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 09 unused slot fill to 16
00B8' 00 00 00 00
00BC' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 10 unused slot fill to 16
00C0' 00 00 00 00
00C4' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 11 unused slot fill to 16
00C8' 00 00 00 00
00CC' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 12 unused slot fill to 16
00D0' 00 00 00 00
00D4' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 13 unused slot fill to 16
00D8' 00 00 00 00
00DC' 00 00 00 00      db      0,0,0,0,0,0,0,0 ; 14 unused slot fill to 16

```

```
00E0' 00 00 00 00
00E4' 00 00 00 00      db  0,0,0,0,0,0,0,0      ; 15  unused slot  fill to 16
00E8' 00 00 00 00
```

```
00EC' 00      db  0      ; end of table
```

```
00ED'      dseg      ; following tables appear in bank 0
                        ; and MUST be in this order for setup
```

```

;      general device shadow table
;      this is the real table, but not the one
;      viavec uses.
;      this is a table with one entry for each of the 16
;      devices. Each entry is 4 bytes wide as follows:
;      dw      routine      call this routine
;      Note:   this routine has a 3 byte
;              prefix that is called to do
;              an init. (see stdrom)
;      db      flags      see flag equates
;              see equ's at beginning
;      db      anything    up to you
;      stdrom:  offset to rowtbl
;      stdiee:  ieee device address

```

```

020D"      viareal:      ;      vector each device
;      crt
020D" 0197"      dw      stdrom      ; a rom routine
020F" 00      db      0      ; flags
0210" 08      db      rowcrt-rowtbl  ; offset from table start

;      cent
0211" 0197"      dw      stdrom      ; uses std rom routines
0213" 00      db      0      ; flags
0214" 0C      db      rowcen-rowtbl  ; offset from table start

;      modem
0215" 0197"      dw      stdrom      ; again
0217" 00      db      0
0218" 00      db      rowmod-rowtbl

;      printer
0219" 0197"      dw      stdrom
021B" 00      db      0
021C" 04      db      rowprt-rowtbl

;      ieee
021D" 01FD"      dw      stdiee      ; routine address for std ieee
021F" 20      db      20h      ; flags (show this is an ieee)
0220" 04      db      04      ; ieee device address (default)

;      unused slots (up to 16 devices)
0221" 0192" 0000      dw      unused,0      ; 5

```



```

0225" 0192" 0000      dw  unused,0      ; 6
0229" 0192" 0000      dw  unused,0      ; 7
022D" 0192" 0000      dw  unused,0      ; 8
0231" 0192" 0000      dw  unused,0      ; 9
0235" 0192" 0000      dw  unused,0      ; 10
0239" 0192" 0000      dw  unused,0      ; 11
023D" 0192" 0000      dw  unused,0      ; 12
0241" 0192" 0000      dw  unused,0      ; 13
0245" 0192" 0000      dw  unused,0      ; 14
0249" 0192" 0000      dw  unused,0      ; 15

```

```

;      initvec
;      initializes the re-direction vectors
;      All 5 entries MUST be here.

```

```

024D"      initvec:
024D" 8000      dw  8000h      ; conin = crt
024F" 8000      dw  8000h      ; conout = crt
0251" 0000      dw  0000h      ; auxin = nothing
0253" 0000      dw  0000h      ; auxout = nothing
0255" 0000      dw  0000h      ; LIST = Nothing

```

```

;      initstr
;      strings of characters sent to a device
;      during its initialization
;
;      this table contains 1 string for each device
;      (0-15) in the following format:
;          db      length-of-string (may be 0)
;          db      'string-itself' (if any)

```

```

0257"      initstr:      ; strings (1 each device)
;      each string (= 255
;      entire table only 256 bytes

```

```

0257" 00 00 00 00      db  0,0,0,0,0,0,0,0 ; initially no strings each device
025B" 00 00 00 00
025F" 00 00 00 00      db  0,0,0,0,0,0,0,0
0263" 00 00 00 00

```

```

0267"      ds  15*16      ; remainder of table space

```

```

;      DELAY  is used by the output routines for a delay in printing
;      device not ready message

```

```

0357" 18      DELAY:      DB  24      ;For 2 minute delay
0358" 01      DELAYON:    DB  1      ;Use delay time (0 means ignore)

```



```

;      additional things that can be in bank 0

;      viadev
;      this is the table that viavec really uses
;      this allows us to take care of 1st time
;      init and strings with no overhead on each
;      character.

0359"      viadev:
0359"      003F" 0000      dw      vecist,0      ; 0 room for all 16 devices
035D"      003F" 0000      dw      vecist,0      ; 1
0361"      003F" 0000      dw      vecist,0      ; 2
0365"      003F" 0000      dw      vecist,0      ; 3
0369"      003F" 0000      dw      vecist,0      ; 4
036D"      003F" 0000      dw      vecist,0      ; 5
0371"      003F" 0000      dw      vecist,0      ; 6
0375"      003F" 0000      dw      vecist,0      ; 7
0379"      003F" 0000      dw      vecist,0      ; 8
037D"      003F" 0000      dw      vecist,0      ; 9
0381"      003F" 0000      dw      vecist,0      ; 10
0385"      003F" 0000      dw      vecist,0      ; 11
0389"      003F" 0000      dw      vecist,0      ; 12
038D"      003F" 0000      dw      vecist,0      ; 13
0391"      003F" 0000      dw      vecist,0      ; 14
0395"      003F" 0000      dw      vecist,0      ; 15

;      romtbl
;      one entry each device accessed directly via rom
;      routines. Each device is ist, in, ost, out
;      rom offsets. 00=do nothing.
;      note: modem must be first, followed by printer.
;      That is how we know whether the device is
;      serial port A or B for devini.

0399"      romtbl:      ;      4 bytes each device

;      modem      must be 1st
0399"      0C 18 12 1E      rommod: db      saistat,serain,saostat,seraout
;      printer      must be 2nd
039D"      0F 1B 15 21      romprt: db      sbistat,serbin,sbostat,serbout
;      crt
03A1"      03 06 00 09      romcrt: db      skey,ci,0,cout
;      centronics
03A5"      4B 51 4E 54      romcen: db      pistat,parinp,postat,parout

;      ieeeetbl
;      this is the table of rom routine offsets for
;      the std ieee interface

```

```
03A9"          ieeeTbl:          ; std viaxxx order excluding init which is ignored
03A9" 3F 45 42 48          db      ieistat,ieinp,ieostat,ieout
```

```
end
```

Macros:

Symbols:

005C*	@AIVEC	003E*	@ADVEC	0062*	@CIVEC
0044*	@COVEC	0038*	@LOVEC	0006	ACK
FD50	ACTDSK	FD4D	ACTSEC	FD4E	ACTTRK
FD51	ACTTYP	2004	ADRIE	2008	APROTO
005B1'	AUXIN	004C1'	AUXIST	003D1'	AUXOST
00281'	AUXOUT	2006	BDRIVE	2009	BPROTO
0134"	CHKACT	0006	CI	2000	CLKPTR
01CB"	CLRLP	FDDA	COMNSP	0061I'	CONIN
0052I'	CONIST	0043I'	CONOST	002E1'	CONOUT
0009	COUT	0008	DEC	0357I"	DELAY
00C9I"	DELAY0	0358"	DELAYON	00DC"	DELOOP
0004	DEVBAUD	0001	DEVIN	001C1'	DEVINI
0002	DEVOUT	0008	DEVSER	006C1'	DEVTBL
0010	DEVXACT	FD4A	DMABANK	FD4B	DMADR
0064'	DOIN	0055'	DOIST	0046'	DOOSTAT
0031'	DOOUT	000C	DSKD1	0005	DSKS1
FE1C	DSKSAVE	FE1C	DSKSP	0003	ETX
0000	FIVESEC	017E"	GETIN	00B0"	GOTCR
2355	HERTZ	0008	IBM	01EC	IDELAY
0002	IDELLEN	0080	IDEVLEN	00A2	IDEVSHD
0022	IDVTBL	0020	IDRVLEN	0002	IDRVTBL
000A	IDVILEN	00E2	IDVINI	0100	IDVSLEN
00EC	IDVSTR	FD52	IEADR	03A9"	IEETBL
0045	IEINP	003F	IEISTAT	0042	IEOSTAT
0048	IEOUT	01EE	IEXD	000E"	INIT
0257"	INITSTR	024D1"	INITVEC	016A"	INNEXT
0173"	INNXT1	0167"	INPUT	017D"	INREADY
014C"	INSTAT	0167"	INWAIT	0040	ISHDLEN
0005	ISIEEE	015D"	IST	014C"	ISTNEXT
0154"	ISTNXT1	0000	IVERS	020C"	JPHL
200A	KBDTBL	00221'	LIST	0037I'	LISTST
2002	MEDIA	0120"	OSTNEXT	012C"	OSTNXT1
00B9"	OUTETX	00D8"	OUTLOOP	0092"	OUTNEXT
00A8"	OUTNXT1	00A9"	OUTNXT2	0092"	OUTPUT
0147I"	OUTST	0147"	OUTST1	0120"	OUTSTAT
00CA"	OUTWAIT	00DF"	OUTWLP	0051	PARINP
0054	PAROUT	004B	PISTAT	004E	POSTAT
FD57	RDBANK	03A5"	ROMCEN	03A1"	ROMCRT
0002	ROMETX	01AD"	ROMINI	0399"	ROMMOD
FD44	ROMMOVE	01AB"	ROMNOP	00FC	ROMPROD
039D"	ROMPRT	0399"	ROMTBL	00FE	ROMVER
0001	ROMXON	FD56	RSBANK	005A	RSEC
000C	SAISTAT	0012	SAOSTAT	FDDA	SAVESP
000F	SBISTAT	0015	SBOSTAT	0060	SENDEN
0018	SERAIN	001E	SERADOUT	001B	SERBIN
0021	SERBOUT	01CA"	SERCLR	00F2*	SERNR
01E1"	SERPROTO	01F8"	SERSET	0024	SETBAUD
0000"	SETVEC	0003	SKEY	0075"	SNDDN
0069"	SNDFLP	006E"	SNDFND	0072"	SNDLP
0079"	SNDMOR	008F"	SNDUNA	01FD"	STDIEEE
01AC"	STDRET	0197"	STDROM	0000	SYS
0003	SYSB	FDDC	TBUF	0000'	TODSEG

0101"	UNALO	0105"	UNALP	011D"	UNARDY
0192"	UNUSED	003F"	VECIST	0001	VEGIN
0000	VECIST	0002	VECDST	0003	VEGOUT
000CI"	VECPTR	0359"	VIADEV	020D"	VIAREAL
0180"	VIAVEC	0183"	VIAVIA	00D0"	WAIT
0006	WAITING	005D	WSEC	0001	XEROX
0013	XOFF	0011	XON	0004	XONACT
0007	XONETX				

No Fatal error(s)

.z80

title bios3 (excbios3) misc routines (memory, time...)

```

;*****
;*
;*      EXECUTIVE BIOS
;*      for
;*      CP/M 3.XX
;*      by
;*
;*      Glenn S. Tenney
;*      and
;*      Roger W. Chapman
;*
;*      Copyright (c) 1983
;*      Osborne Computer Corporation
;*
;*****

```

.list

```

PUBLIC MOVE, XMOVE
public sbank,dbank,kludge      ; allow xmove's
PUBLIC SELMEM
public time

public torom                   ; to rom and comm stack
PUBLIC ROMMV

extrn dmabk      ; selmem must set this too

cseg      ; common

```

0000'

```

;*****      warning      !!!!!!!!!!!
;
;      Read the DRI documentation carefully regarding
;      move and xmove. They may not work as you might be
;      led to believe from some of the wording. It is
;      like this: the xmove most recently before the move
;      determines the source/destination banks, any move
;      resets an xmove setting. This version works with
;      the stack anywhere (it is saved).
;

```

0000'

```

MOVE:
;
;      ENTRY  HL = Destination address
;             DE = Source address
;             BC = Count
;
;      EXIT   HL = Next destination address
;             DE = Next source address
;

```

; re-done by Glenn Tenney 1982-12-17

```

0000' 3A 0084'      ld    a,(kludge)      ; should we kludge s/d banks
0003' B7           or     a              ; 0=no previous xmove
0004' 20 05        jr     nz,maybed      ; previous xmove, may be difficult

0006' EB          smove: ex    de,hl      ; simple move
0007' ED B0        ldir                ; just move
0009' EB          ex     de,hl      ; leaving regs ask
000A' C9          ret                  ; and get out quickly

000B'             maybed:                ; see if a difficult xbank move

000B' ED 73 FDDA    ld     (saveesp),sp   ; save crnt stack
000F' 31 FDDA      ld     sp,commsp      ; and use one in common

0012' DB 00        in     a,(sys)        ; current bank
0014' 32 0081'     ld     (ubank),a      ; save current bank

0017' E5          push   hl            ; save
0018' 2A 0082'     ld     hl,(sbank)     ; get s and d bank
001B' 7C          ld     a,h
001C' BD          cp     l              ; both in same bank
001D' 20 0C        jr     nz,dmove      ; no--difficult move

; same bank move (but may not be current bank)

001F' E1          pop     hl            ; restore hl 1st
0020' 3A 0082'     ld     a,(sbank)     ; desired bank for move
0023' D3 00        out    (sys),a
0025' CD 0006'     call   smove          ; do a simple move
0028' C3 0041'     jp     done           ; move done, reset s and d bank

002B'             dmove:                ; move between different banks

002B' 22 FD56      ld     (rsbank),hl    ; copy s and d banks for rom
002E' E1          pop     hl            ; restore
002F' DD E5        push   ix            ; save
0031' EB          ex     de,hl      ; make regs look like ldir
0032' CD 003B'     call   rommv         ; off to rom's move routine
0035' EB          ex     de,hl      ; and back to d.r.'s way
0036' DD E1        pop     ix            ; (used to get to rom)
0038' C3 0041'     jp     done           ; and all done

003B' DD 2A FD44    rommv: ld     ix,(rommove) ; get address of rom's move routine
003F' DD E9        jp     (ix)          ; off to rom's move routine (ret back)

0041'             DONE:
0041' 3A 0081'     ld     a,(ubank)     ; restore current bank

```

```

0044'  D3 00          out    (sys),a      ; all back to what it shd be
0046'  AF            xor     a            ; clear kludge flag for dri
0047'  32 0084'       ld      (kludge),a   ; no more xmoves in sight
004A'  ED 7B FDDA     ld      sp,(savesp)  ; restore stack too
004E'  C9            RET

```

```

004F'  XMOVE:
;      set for cross bank moves
;
;      ENTRY  B = Destination bank
;             C = Source bank
;
;      EXIT   DBANK = Destination bank
;             SBANK = Source bank
;
;      an alternate method....
004F'  21 0082'       ld      hl,sbank     ; pt to 3 byte area
0052'  71            ld      (hl),c       ; source bank
0053'  23            inc     hl
0054'  70            ld      (hl),b       ; destination bank
0055'  23            inc     hl
0056'  74            ld      (hl),h       ; kludge to a known non-zero (h)
0057'  C9            RET

```

PAGE

```
0050'                                cseg    ; must be common
0050'                                SELMEM:
                                ;
                                ; ENTRY  A = Memory bank to select
                                ;
                                ; EXIT   None
                                ;
                                ; USES A-reg only!
0050'    D3 00                                OUT    (SYS),A
0050'    C9                                RET
                                PAGE
```



```

;      torom
;      calls rom with a common stack not user's stack
;      this is for the tpa user to allow calls to rom
;
;      This routine jumps to ROM address and switches stacks
;
;      ENTRY   E = ROM address (offset from 0100h)
;              A, BC, HL, IX = Any necessary parameters
;
;      EXIT    All ROM exit parameters remain in tact
;

```

```

005B'  ED 73 FDDA      torom: ld      (savesp),sp      ; save user's stack
005F'  31 FDDA        ld      sp,comnsp      ; pt to one in common

0062'  57             ld      d,a            ; save a
0063'  DB 00          in      a,(sys)        ; get crnt bank
0065'  32 0081'       ld      (ubank),a      ; save for later exit
0068'  F6 80          or      80h           ; force into rom
006A'  D3 00          out     (sys),a        ; now
006C'  7A            ld      a,d            ; restore a

006D'  16 01          ld      d,01h         ; de=01xxh
006F'  CD 007E'       call    romjp         ; off to the rom (via our code)

0072'  F5             push    af            ; save a and flags
0073'  3A 0081'       ld      a,(ubank)     ; incoming bank info
0076'  D3 00          out     (sys),a        ; restore bank now
0078'  F1             pop     af            ; all regs now restored

0079'  ED 7B FDDA      ld      sp,(savesp)   ; and return
007D'  C9             ret                  ; with user's stack

007E'  D5             romjp: push    de      ; dummy up like a call to rom
007F'  C9             ret                  ; off to rom (addr was in de)

```

page

```

;      time
;      ;      this routine is used to get/set the date
;      ;      and time
;
;      the time is being updated by the rom interrupt routine
0000' C9      time: ret      ;      do nothing

;      these are areas used only within this module
0001'      ubank: ds      1      ; save crnt bank here going to rom
;      ; or when flipping banks

;      these are bank numbers for transfers
;      they MUST be in this order ! ! ! ! !
0002'      sbank: ds      1      ; current source bank
0003'      dbank: ds      1      ; current destination bank

;      this is a kludge because dri calls xmove then selmem and then move
0004' 00      kludge: db      0      ; nonzero = use s/d bank

end
```

Macros:

Symbols:

FD50	ACTDSK	FD4D	ACTSEC	FD4E	ACTTRK
FD51	ACTTYP	2004	ADRIIVE	2008	APROTO
2006	BDRIVE	2009	BPROTO	0006	CI
2000	CLKPTR	FD0A	COMNSP	0009	COUT
0003I'	DBANK	0008	DEC	0004	DEVBAUD
0001	DEVIN	0002	DEVOUT	0008	DEVSER
0010	DEVXACT	FD4A	DWABANK	0000*	DWABK
FD4B	DWADR	002B'	DMOVE	0041'	DONE
000C	DSKD1	0005	DSKS1	FE1C	DSKSAVE
FE1C	DSKSP	2355	HERTZ	0008	IBM
01EC	IDELAY	0002	IDELLEN	0000	IDEVLEN
00A2	IDEVSHD	0022	IDEVTBL	0020	IDRVLEN
0002	IDRVTL	000A	IDVILEN	00E2	IDVINI
0100	IDVSLEN	00EC	IDVSTR	FD52	IEADR
0045	IEINP	003F	IEISTAT	0042	IEOSTAT
0048	IEDUT	01EE	IEDX	0040	ISHDLEN
0000	IVERS	200A	KBDTBL	0084I'	KLUDGE
000B'	MAYBED	2002	MEDIA	0000I'	MOVE
0051	PARINP	0054	PAROUT	004B	PISTAT
004E	POSTAT	FD57	RDBANK	0002	ROMETX
007E'	ROMJP	FD44	ROMMOVE	003BI'	ROMMV
00FC	ROMPROD	00FE	ROMVER	0001	ROMXON
FD56	RSBANK	005A	RSEC	000C	SAISTAT
0012	SAOSTAT	FD0A	SAVESP	0082I'	SBANK
000F	SBISTAT	0015	SBOSTAT	0058I'	SELMEM
0060	SENDEN	0018	SERAIN	001E	SERADUT
001B	SERBIN	0021	SERBOUT	0024	SETBAUD
0003	SKEY	0006'	SMOVE	0000	SYS
0003	SYSB	FD0C	TBUF	0000I'	TIME
005BI'	TOROM	0001'	UBANK	005D	WSEC
0001	XEROX	004FI'	XMOVE		

No Fatal error(s)

.280

TITLE bios4 (excbios4) disk i/o

```
; 1983-03-20    set for 1st release
; 1983-03-13    fix in BIOS for DRI bug in BDOS
; 1983-02-21    support extrn rtn for i/o err msgs + retry
; 1983-02-17    re-did stacks for going to bank 0
; 1983-02-15    slight restructure for code in bank 80h
```

```
*****
;*              *
;*      EXECUTIVE BIOS      *
;*              for        *
;*      CP/M 3.XX          *
;*              by         *
;*              *
;*      Glenn S. Tenney    *
;*      COMPRO             *
;*      P.O. Box 5260      *
;*      San Mateo, CA 94402 *
;*              *
;*      Copyright (c) 1983 *
;*      Osborne Computer Corporation *
;*              *
;*****
```

```
;      Some original code still remains which was written
;      by Roger Chapman.  This code appears in upper case.
```

```
;      include EXCBIOS.LIB
.list
```

```
PUBLIC HOME, SELDSK, SETTRK, SETSEC, SETDMA
PUBLIC READ, WRITE, SECTRN, MULTIO, FLUSH
public setbnk
public dwabk ; selmem must set this
```

```
public stdini, stdsel, stdrd, stdwt ; std routines 5.25 built-in
public drive, actcnt ; so can be in err msgs
```

```
extrn drvtbl ; table of drives
extrn dpbtbl ; table of dpb info
extrn stdsen, stderr ; std i/o err rtns
```

0015

```
dpblen equ 17+4 ; length of dpb + prefix
```

0000'

```
cseg ; initial entry shd be common bank always
```

```

0000'          SETDMA:
;
;          Set source/destination address for later disk write/read
;
;          ENTRY  BC = Direct memory access address
;
0000'  ED 43 0075'      LD      (DMAADR),BC
0004'  DB 00           in      a,(sys)      ; get current bank

```

```

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

```

```

;          fall through to set this bank for future i/o's too

```

```

0005'          SETBNK:
;
;          ENTRY  A = Memory bank for next disk transfer
;
0005'  32 0074'      LD      (DMABK),A
0009'  C9           RET

```

```

000A'          HOME:
;
;          Sets track value to 0
;
;          ENTRY  None
;
000A'  01 0000      ld      bc,0           ; track zero

```

```

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;          fall thru to settrk

```

```

000D'          SETTRK:
;
;          Sets track number. The track number is saved for
;          later use during a disk transfer operation.
;
;          ENTRY  BC = Track number
;
000D'  ED 43 0078'      LD      (SETRK),BC
0011'  C9           RET

```

```

0012'          SETSEC:

```

```

;
; Set the sector number for later disk read/write operation.
;
; ENTRY BC = Sector number
;

0012' 79          ld    a,c    ; get sector number
0013' 3C          inc    a      ; adjust for 1 origin
0014' 32 0077'    ld    (seksec),a    ; save for later use
0017' C9          RET

0018'            FLUSH:
;
; Force physical buffer flushing for user supported deblocking
;
; ENTRY None
;
; EXIT None
;

0018' AF          xor    a      ; do nothing, but did it ok
0019' C9          RET

PAGE
```

```
001A'          MULTIO:
;
;      Set count of consecutive sectors for READ or WRITE
;
;      ENTRY  C = Mutisector count
;
;      EXIT   None
;

001A'  79          LD      A,C
001B'  32 006F'    LD      (SECCNT),A

001E'  AF          xor     a          ; clear actual mult count
001F'  32 006E'    ld      (actcnt),a ; to start 'er off

0022'  C9          RET
```

PAGE

0023'

SECTRN:

```

;
; Translate sector number given translate table
;
; ENTRY BC = Logical sector number
; DE = Translate table address
; DE = 0, No translation required
;
; EXIT HL = Physical sector number
;

```

0023' 69

LD l,c ; get logical sector #

0024' 60

LD h,b

0025' 7B

LD A,E

0026' B2

OR D

0027' C8

RET Z

0028' 19

ADD HL,DE

```

; the tables MUST be in bank 0 or common

```

0029' DB 00

in a,(sys) ; get crnt bank

002B' 5F

ld e,a ; save bank

002C' AF

xor a ; force to bank 0

002D' D3 00

out (sys),a ; to locate sectran table

002F' 6E

LD L,(HL) ; pick up sector number from table

0030' 67

ld h,a ; (a was 0) hl=sector number

0031' 7B

ld a,e ; restore bank number

0032' D3 00

out (sys),a

0034' C9

RET

page


```

;      this starts the routines that are further vectored
;      by disk device via the dph prefix. We may be
;      in other than bank 0 and the dph header is in bank 0,
;      so we switch to bank 0 first.

```

```

;      todseg
;      transfer control to a bank 0 routine
;      af      destroyed on entry, preserved on exit
;      bc      preserved
;      de      preserved
;      hl      entry: addr of routine, preserved on exit

```

```

0035'      todseg:
0035'      ED 73 FE1C      ld      (dsksave),sp      ; save sp
0039'      31 FE1C      ld      sp,dsksp      ; make sure it is in common
003C'      DB 00      in      a,(sys)      ; save current bank
003E'      32 0055'      ld      (curbnk),a
0041'      3E 80      ld      a,80h      ; select bank 0 + rom
0043'      D3 00      out     (sys),a      ; and select rom + bank 0

0045'      CD 0054'      call     jphl      ; call rtn in hl

0048'      F5      push     af      ; save result if any
0049'      3A 0055'      ld      a,(curbnk)      ; get saved bank
004C'      D3 00      out     (sys),a      ; bank is now back
004E'      F1      pop      af      ; restore result
004F'      ED 7B FE1C      ld      sp,(dsksave)      ; restore sp
0053'      C9      ret      ; and all done

```

```

0054'      E9      jphl:  jp      (hl)      ; simulate 'call (hl)'

```

```

0055'      curbnk: ds      1      ; a place to save bank #

```

page

```

;
;      Select the specified Disk Drive
;
;      ENTRY  C = Disk Drive (0 - 15)
;              BIT 0 E = 0, first call for this disk
;
;      EXIT   HL = DPH address for specified drive
;              HL = 0000 if drive does not exist
;
;

0056' 21 0000'      seldsk: ld    hl,dsel      ; pt to bank 0 routine
0059' C3 0035'      jp      todseg          ; change to other bank

005C'              dseg

0060" 79           dsel: ld      a,c          ; save drive #
0061" 32 006A'     ld      (drive),a        ; change to that drive now
0064" 69           ld      l,c              ; get drive #
0065" 26 00       ld      h,0              ; into 16 bits
0067" 01 0000*    ld      bc,drvtbl        ; start of table
006A" 29           add     hl,hl            ; drive # 2
006B" 09           add     hl,bc           ; hl=ptr to proper entry
006C" 7E           ld      a,(hl)          ; lo byte
006D" 23           inc     hl
006E" 66           ld      h,(hl)          ; hi byte
006F" 6F           ld      l,a             ; hl=that entry from table
0070" 22 006B'    ld      (dphnow),hl      ; post current dph addr
0073" B4           or      h               ; is there any such drive??
0074" C8           ret      z              ; no, get out NOW

;      hl is dph for this drive
0075" E5           push    hl              ; save for when we exit
0076" 2B           dec     hl              ; back up to drive #
0077" 7E           ld      a,(hl)          ; drive # controller relative
0078" 32 007A'    ld      (seldsk),a       ; post controller's drive #

;      let seldsk routine for this dph handle it now

007B" 2B           dec     hl              ; back up to hi of rtn addr
007C" 7E           ld      a,(hl)
007D" 2B           dec     hl              ; to lo of seldsk addr
007E" 6E           ld      l,(hl)
007F" 67           ld      h,a             ; hl= seldsk for this dph

0080" 7B           ld      a,e             ; show if sense density needed
0081" 1F           rra                     ; carry=no      no carry=yes

0082" E3           ek      (sp),hl         ; hl=dph  tos=seldsk rtn addr

0083" C9           ret                     ; off to seldsk for this dph

0084"              cseg

page

```

005C'

READ:

```

;
;   Read a sector from the specified drive
;
;   ENTRY   None
;
;   EXIT    A = 00h, if no errors
;           A = 01h, if non-recoverable error occurred
;

```

005C' 11 FFF9

ld de,-7 ; offset of read addr in dph prefix

005F' 18 03

jr towr ; off to other bank (common rtn)

0061'

WRITE:

```

;
;   Write a sector to the specified drive
;
;   ENTRY   None
;
;   EXIT    A = 00h, if no errors
;           A = 01h, if physical error
;           A = 02h, if disk is Read Only
;

```

0061' 11 FFFB

ld de,-5 ; offset of write addr in dph prefix

0064' 21 0024"

torw: ld hl,drw ; pt to bank 0 routine

0067' C3 0035'

jp todseg ; and off to other bank

006A'

dseg

```

;   common routine to vector read or write
;   de=offset from start of this dph for prefix
;   word having address of routine.

```

0024" 2A 006B'

drw: ld hl,(dphnow) ; pt to current dph

0027" 19

add hl,de ; hl=addr of write routine addr

0028" 7E

ld a,(hl) ; lo

0029" 23

inc hl

002A" 66

ld h,(hl) ; hi

002B" 6F

ld l,a ; hl=write routine addr

002C" E9

jp (hl) ; off to write for this dph

002D"

cseg

page

```

;       these are the various routines used by the standard
;       built-in 5.25" floppies (or flosies for comic strip aficionados)
;       These must be in either common or bank 0. They will
;       be called with bank 0 active (see above parent routines
;       which call these).

006A'          dseg

;       stdini
;       one time initialization (at reboot)

002D'  C9      stdini: ret      ;      nothing

;       stdsel
;       standard built-in 5.25" floppy select disk
;       with sense density
;       e       bit 0 =1 if no sense density needed
;               (ld a,e ... rra ; to chk lo bit of e)
;       note:   carry already set on entry
;       hl      addr of dph for that drive
;               in may not = out of hl if error on sense density

002E'  DA 007B' stdsel: jp      c,seltyp      ; done, set type

0031'  7C          ld      a,h              ; 1st time, is drive valid
0032'  B5          or      l                ; hl=0 means no real drive
0033'  C8          ret      z              ; not valid, no senden and no seltyp!

0034'  E5          push    hl              ; save ptr to dph

0035'
0035'  3A 007A'    selrtry:                ; here to retry
0038'  32 FD50      ld      a,(sekdisk)      ; desired disk number
;               ld      (actdisk),a        ; so senden can see it

003B'  CD 0160      call    100h+senden      ; call rom directly

003E'  2B 11        jr      z,selok          ; some diskette there

;       no diskette or sense density didn't work
0040'  CD 0000*     call    stdsen          ; see if msg needed
0043'  20 F0        jr      nz,selrtry      ; yes, and user said RETRY

0045'  E1          pop     hl              ; just to clear the air
0046'  21 0000      ld      hl,0            ; show no drive
0049'  AF          xor     a              ; and show things rather screwed up
004A'  32 007B'     ld      (sektyp),a      ; some weird type (NOT Osborne dd)
004D'  32 0070'     ld      (spt),a        ; dummy up to spt=0
0050'  C9          ret                    ; and exit quickly!

```

```

; sense returned something, see what it was
0051" 21 0000* selok: ld hl,dpbtbl ; pt to head of table
0054" 11 0014 ld de,dpblen-1 ; length each entry
0057" sellp: ; figure out which dpb to use
0057" 3A FD51 ld a,(acttyp) ; actual device
005A" BE cp (hl) ; same as this tbl entry?
005B" 23 inc hl ; 1st bump to sec/trk
005C" 7E ld a,(hl) ; get spt from table
005D" 20 03 jr nz,selnxt ; no, try next tbl entry
005F" B0 cp b ; same as this drive
0060" 28 06 jr z,selfnd ; yes--found it
0062" 3C selnxt: inc a ; see if end of table
0063" 28 03 jr z,selfnd ; yes (0ffh sec/trk)
0065" 19 add hl,de ; bump to next entry
0066" 18 EF jr sellp ; and try next entry

; this is the proper dpb
0068" 23 selfnd: inc hl ; pt to skew ptr
0069" EB ex de,hl ; let de pt to this dpb entry
006A" E1 pop hl ; restore ptr to dph
006B" E5 push hl

006C" 1A ld a,(de) ; copy skew ptr
006D" 77 ld (hl),a ; into dph
006E" 23 inc hl
006F" 13 inc de
0070" 1A ld a,(de)
0071" 77 ld (hl),a

0072" 01 000B ld bc,11 ; offset into dph for dpb
0075" 09 add hl,bc ; hl=dph+12
0076" 13 inc de ; de=addr of dpb
0077" 73 ld (hl),e ; plunk dpb addr in
0078" 23 inc hl
0079" 72 ld (hl),d ; dph now all set
007A" E1 pop hl

; seldsk done, just set type for possible i/o
; and multok = or of skew tbl addr
; hl=dph ptr, get type from dpb prefix

007B" 7E seltyp: ld a,(hl) ; pick up skew tbl addr
007C" 23 inc hl
007D" B6 or (hl) ; a=0 if no skewing
007E" 2B dec hl ; restore dph address
007F" 32 006D' ld (multok),a ; so multio will work

0082" EB ex de,hl ; de=dph
0083" 21 000C ld hl,12
0086" 19 add hl,de ; hl=dph+12 (ptr to dpb)
0087" 7E ld a,(hl) ; lo of dpb addr
0088" 23 inc hl
0089" 66 ld h,(hl) ; hi of dpb addr
008A" 6F ld l,a ; hl=this dpb

```



```

008B" 2B          dec hl          ; back up to sectors per track
008C" 2B          dec hl
008D" 2B          dec hl
008E" 7E          ld a,(hl)        ; get sectors per track
008F" 32 0070"    ld (spt),a      ; for multio usage

0092" 2B          dec hl          ; back up to disk type value
0093" 7E          ld a,(hl)        ; get crnt disk type info
0094" 32 007B"    ld (sektyp),a    ; and that is to be used
0097" EB          ex de,hl         ; restore regs

0098" C9          ret             ; all done with seldsk

;          stdrd
;          stdwt
;          standard 5.25" built-in floppy read and
;          write routines (see read/write above)

0099" 1E 5A       stdrd: ld e,rsec   ; to read a sector
009B" C3 00A0"    jp stdrw          ; common for read or write

009E" 1E 5D       stdwt: ld e,wsec   ; to write a sector

;          ;
;          jp stdrw          ; fall thru to common routine
;          ;
;          ;

;          Performs a read or a write
;          ;
;          entry e=offset to proper rom routine (rsec or wsec)

00A0"            stdrw:             ; i/o may be deferred based on multio

00A0" 3A 006E"    ld a,(actcnt)     ; is this 1st rw?
00A3" B7          or a              ; to init dma and sector
00A4" C2 00B3"    jp nz,rwmayb      ; no, may be a real i/o though
00A7" 3A 0077"    ld a,(seksec)     ; crnt sector
00AA" 32 0071"    ld (sec1st),a     ; is 1st sector of multiple
00AD" 2A 0075"    ld hl,(dmaadr)    ; same for dma addr
00B0" 22 0072"    ld (dma1st),hl

00B3" 21 006E"    rwmayb: ld hl,actcnt ; adj cnts
00B6" 34          inc (hl)          ; actual count +1
00B7" 23          inc hl
00B8" 35          dec (hl)          ; multio count -1
00B9" CA 00CD"    jp z,rwrdy        ; 0=this must be last rw of mult

00BC" 3A 006D"    ld a,(multok)     ; 00=multio allowed

```

```

00BF" B7          or    a          ; xx=no multio (skew active)
00C0" C2 00CD"    JP     nz,rwrdy   ; not supported for multio

00C3" 23          inc    hl          ; pt to spt
00C4" 3A 0077"    ld     a,(seksec)  ; crnt sector for r/w
00C7" BE          cp     (hl)        ; is this last sector of track?
00C8" CA 00CF"    JP     z,rwnow    ; yes, read full or partial track

; no i/o done this time, just say it went ok
00CB" AF          xor     a          ; then just clear it for no err
00CC" C9          ret              ; and exit as if it was ok

00CD"             rwrdy:           ; rw ready (count reached 0)
00CD" 36 01       ld     (hl),1      ; set secCnt to 1

00CF"             rwnow:           ; i/o required this time
00CF" D5          push    de         ; save rom jmp offset

; Transfer BIOS variables to ROM variables

00D0" 21 0074"    LD     HL,dmabk    ; copy our disk parms to rom's
00D3" 11 FD4A     LD     DE,dmabank  ; this is start of rom's ram
00D5" 01 0008     LD     BC,SEKTP-dmabk+1 ; length to copy
00D9" ED B0      LDIR

00DB" 3A 0071"    ld     a,(sec1st)  ; 1st sector of multiple
00DE" 32 FD4D     ld     (actsec),a
00E1" 2A 0072"    ld     hl,(dma1st) ; and starting addr
00E4" 22 FD4B     ld     (dmadr),hl

00E7" 3A 006E"    ld     a,(actcnt)  ; get number of sectors
00EA" 47          ld     b,a         ; for rom to read/write

; now ready to do the actual i/o then check for errors

00EB" E1          pop     hl         ; restore rom rtn addr (l)
00EC" E5          push    hl         ; save rom rtn addr offset

00ED" 26 01       ld     h,01h      ; make it true addr
00EF" CD 0054"    call    jphl       ; call (hl)

00F2" D1          pop     de         ; restore rom rtn offset
00F3" B7          or     a          ; were there any errors
00F4" 2B 1F       JR     z,rwok      ; no, ok to exit rw then

```

```

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

```

```

; NOTICE:

```

```

; The following is a bit of a kludge inserted because
; the BDOS does not appear to handle both media change

```

```

; and density change on the same drive. The problem is
; that when the media changes (which is posted by an
; interrupt routine) the BDOS notices the flag and tries
; to see if the media really changed. Although the
; documentation states that the BDOS will then issue
; an initial SELDISK and then read the directory, the
; BDOS just goes ahead and blindly reads the new disk.
; Well, if the density, sector size or sectors per track
; just happen to be different on the new disk, the
; results are UNPREDICTABLE!!!!!!
; If the density changes, then an error will be posted.
; If the density remains the same, it is likely that the
; read will, if it is for a sector valid on the new media,
; detect that the directory is different.

;=====

; we had some error, see if media changed

00F6" F5          push    af          ; save error condition
00F7" D5          push    de          ; save rom rtn addr
00F8" C5          push    bc          ; and fdc status byte

00F9" CD 0160     call    100h+senden ; see if same density
00FC" 20 1D      jr      nz,dskchg   ; error on sense, media changed

00FE" 3A 0070'    ld      a,(spt)    ; sense ok, see if same as old
0101" B8          cp      b          ; same sectors per track?
0102" 20 17      jr      nz,dskchg   ; no -- must have changed
0104" 21 007B'    ld      hl,sektyp  ; hmm then check disk type
0107" 3A FD51     ld      a,(acttyp) ; new type
010A" BE          cp      (hl)       ; vs. old type
010B" 20 0E      jr      nz,dskchg   ; different! a changed disk

; at least it is the same type of media, as far as
; we can tell, so process the error

010D" C1          pop     bc          ; b=fdc status byte
010E" D1          pop     de          ; e=rom rtn addr
010F" F1          pop     af          ; seems the same, process error

; let our error routine process the error.
; That routine may issue an error message or ask
; us to retry.
; We pass to it:
;     a      error flag
;     b      fdc status byte
;     e      rom rtn addr
; The error routine must restore all regs and:
;     Z      no retry, return error
;     NZ     retry again

0110" CD 0000*    call    stderr      ; issue std err msg (maybe)
0113" 20 BA      jr      nz,rtnow    ; asked to retry

```



```
0115" 21 006E"      rwok: ld    hl,actent    ; reset sector count
0118" 36 00          ld    (hl),0          ; without changing af
011A" C9             ret                    ; and all done

011B" C1             dskchg: pop    bc        ; media HAS changed, show it
011C" D1             pop    de
011D" F1             pop    af            ; clear the stack
011E" AF             xor     a
011F" 3D             dec     a            ; a=ffh to show media changed
0120" 18 F3          jr      rwok         ; and exit w/ i/o completed

0122"               cseg

               page
```

```

006A'          cseg      ; common cause of dmabk

;      the following are a few areas used only w/in this
;      module

;      parameters local to the bios re: disk i/o

006A' 00          drive: db      0      ; current drive # 0 relative
006B' 0000        dphnow: dw      0      ; addr of crnt dph (set by seldsk)

006D' 00          multok: db      0      ; the or of skew table addr
                                ; 0=multio ok this drive
                                ; x=no multio this drive

;      these 3 items must be in this sequence
006E' 00          actcnt: db      0      ; actual number of sectors this i/o
006F' 01          secct: db      1      ; multio number of sectors
0070'            spt:   ds      1      ; sectors per track crnt drive

0071'            sec1st: ds      1      ; starting sector when multiples
0072'            dma1st: ds      2      ; starting dma addr when multiples

;      the following MUST be the SAME as the rom

0074' 00          dmabk: db      00h     ; desired bank for xfr ( default = 00 )
0075'            dmaadr: ds      2      ; desired address for xfr
0077'            seksec: ds      1      ; desired sector
0078'            sektrk: ds      2      ;          track
007A'            sekdsd: ds      1      ;          disk unit
007B' 0C          sektyp: db      0ch     ; sector size -----))) forced to Osborne dd
                                ;          sector size

                                end

```

Macros:

Symbols:

006E1'	ACTCNT	FD50	ACTDSK	FD4D	ACTSEC
FD4E	ACTTRK	FD51	ACTTYP	2004	ADRIIVE
2008	APROTO	2005	BDRIVE	2009	BPROTO
0006	CI	2000	CLKPTR	FDDA	COMNSP
0009	COUT	0055'	CURBNK	0008	DEC
0004	DEVBAUD	0001	DEVIN	0002	DEVOUT
0008	DEVSER	0010	DEVXACT	0072'	DMA1ST
0075'	DMAADR	FD4A	DMA BANK	0074I'	DMA BK
FD4B	DMA DR	0015	DPBLEN	0052*	DPBTBL
006B'	DPHNDW	006AI'	DRIVE	0008*	DRV TBL
0024"	DRW	0000"	DSEL	011B"	DSKCHG
000C	DSKD1	0005	DSKS1	FE1C	DSKSAVE
FE1C	DSKSP	0018I'	FLUSH	2355	HERTZ
000AI'	HOME	0008	IBM	01EC	IDELAY
0002	IDELLEN	0000	IDEVLEN	00A2	IDEVSHD
0022	IDEV TBL	0020	IDRVLEN	0002	IDRV TBL
000A	IDVILEN	00E2	IDVINI	0100	IDVSLEN
00EC	IDVSTR	FD52	IEADR	0045	IEINP
003F	IEISTAT	0042	IEQSTAT	0048	IEOUT
01EE	IEXD	0040	ISHDLEN	0000	IVERS
0054'	JPHL	200A	KBD TBL	2002	MEDIA
001AI'	MULTIO	006D'	MULTOK	0051	PARINP
0054	PAROUT	004B	PISTAT	004E	POSTAT
FD57	RDBANK	005CI'	READ	0002	ROMETX
FD44	ROMMOVE	00FC	ROMPROD	00FE	ROMVER
0001	ROMXON	FD56	RSBANK	005A	RSEC
00B3"	RW MAYB	00CF"	RW NOW	0115"	RWOK
00CD"	RWRDY	000C	SAISTAT	0012	SAOSTAT
FDDA	SAVESP	000F	SBISTAT	0015	SBOSTAT
0071'	SEC1ST	006F'	SECCNT	0023I'	SECTRN
007A'	SEKDSK	0077'	SEKSEC	0078'	SEKTRK
007B'	SEKTYP	0056I'	SELDISK	0068"	SELFND
0057"	SELLP	0062"	SELNXT	0051"	SELOK
0035"	SELRTY	007B"	SELTYP	0060	SENDEN
0018	SERAIN	001E	SERAOUT	001B	SERBIN
0021	SERBOUT	0024	SETBAUD	0006I'	SETBNK
0000I'	SETDMA	0012I'	SETSEC	000DI'	SETTRK
0003	SKEY	0070'	SPT	0111*	STDERR
002DI"	STDINI	0099I"	STDRD	00A0"	STDRW
002EI"	STDSEL	0041*	STDSEN	009EI"	STDWT
0000	SYS	0003	SYSB	FDDC	TBUF
0035'	TODSEG	0064'	TORW	0061I'	WRITE
005D	WSEC	0001	XEROX		

No Fatal error(s)

.z80

title bios5 (excbios5) cold / warm boots

```
; 06-23-1983 added initialization of device timeout values,
; changed serno to check for status, display message and restore screen
; and to change delay value after first time
```

```
*****
;*                               *
;*      EXECUTIVE BIOS          *
;*      for                     *
;*      CP/M 3.XX               *
;*                               *
;*      by                      *
;*                               *
;*      Glenn S. Tenney        *
;*      and                    *
;*      Roger W. Chapman       *
;*                               *
;*      Copyright (c) 1983     *
;*      Osborne Computer Corporation *
;*                               *
*****
```

```
; this module contains the code for the cold and warm
; boots for the actual bios. The cboot will always
; load the ccp from the system tracks into low mem of
; bank 0, but will attempt to load the file 'EXECST.COM'
; into the tpa (only on cboot). This allows some user
; program to gain control instead of the ccp and without
; using the profile.sub feature which requires submit
; and a writeable disk.
```

.list

```
public cboot,wboot
public stdsen,stderr ; std disk i/o err rtns
public sernr         ; serial not ready err rtn

extrn wboot@,@mxtpa
extrn conout,conin,conist
extrn @civec,selmem ; setbnk
extrn @sec          ; seconds field of tod
extrn move,xmove    ; for ccp.com in bank 0

extrn drvtbl,devtbl ; cboot init from sys tracks

extrn @media        ; media change flag in bios
extrn dph0,dph1     ; our own built-in dph's
extrn dpbtbl        ; table of dph's

extrn signon        ; signon message at cboot
```

```

extrn  initvec      ; initial i/o redirection vectors
extrn  devini       ; serial device init
extrn  @bnkbf       ; 128 byte area for use during boots
extrn  drive,actcnt,dmabk ; for err msgs
extrn  @errmode     ; flag if errs passed to bdos

extrn  vers         ; the version number this bios
extrn  rvers        ; place to put rom's version

EXTRN  DELAY,DELAY0
EXTRN  OUTST
EXTRN  ROMMV

; equates used by the boot

0100      tpa      equ    0100h      ; start of the tpa
0005      bdos     equ    5          ; where to call the bdos

000F      open     equ    15         ; bdos open call
001A      setdma   equ    26         ; bdos set dma addr call
002C      setmult  equ    44         ; bdos set multi sector count
0014      read     equ    20         ; bdos read that many sectors
0030      flush    equ    48         ; bdos purge buffers

000B      dphmf    equ    0bh        ; offset from dph to its media flag byte
000B      dpbcks   equ    0bh        ; offset fm dpb to its cks (for perm disk)
0011      dpblen   equ    17         ; length of one dpb (not our header)

0CFF      ccplen   equ    0cffh      ; length of ccp.com

; cboot
; perform cold (onetime) initialization
; This sets up various areas as needed, including
; reading the ccp.com from track 1 into bank 0 location 0.

; Special note: @bnkbf is a ptr to a 128 byte buffer
; within resident bdos for use during cboot or wboot.
; Well, this is being used as the boot time stack to
; help conserve space.

0000'      cseg      ; bios jmp table pts to a common entry

; cboot
; the cboot entry is in common so in case some
; weirdo jp's to it the system won't go south.

0000'      ED 7B 0000*      cboot: ld    sp, (@bnkbf)      ; put a stack in common
0004'      3E 80              ld    a, 80h          ; force rom + bank 0
0005'      D3 00              out    (sys), a
0008'      C3 0000"          jp     cboot0          ; go do real cboot work

```

```

000B'          dseg      ; cold boot ok in bank 0 (it really is)

0000'          cboot0:   ; actual cold boot work here

;              set rom's ram to keep time in scb
0000' 21 0000*      ld     hl,@sec      ; pt to seconds field
0003' 22 2000      ld     (clkptr),hl   ; now rom knows where to keep time

;              point rom's ram to media change flags so that interrupt
;              routine can post a disk changing
0006' F3          di          ; data changed during interrupts
0007' 21 0000*      ld     hl,@media    ; pt to scb's flag
000A' 22 2002      ld     (media),hl    ; so rom know's what to post
000D' 21 000B*      ld     hl,dph0+dphmf ; and built-in drive a
0010' 22 2004      ld     (adrive),hl
0013' 21 000B*      ld     hl,dph1+dphmf ; and b drive
0016' 22 2006      ld     (bdrive),hl
0019' FB          ei          ; rom can handle interrupts now

001A' 2A 00FE      ld     hl,(romver)   ; copy rom's version 0
001D' 22 0000*      ld     (rvers),hl
0020' 2A 00FC      ld     hl,(romver-2) ; and product+feature
0023' 22 0002*      ld     (rvers+2),hl

;              read ccp.com into bank 0 location 0

0026' 21 0000      ccplp: ld     hl,0
0029' 22 FD4A      ld     (dmabank),hl   ; set bank 0
002C' 22 FD4B      ld     (dmadr),hl    ; set dma addr to zero
002F' 21 0101      ld     hl,0101h     ; track 1 sector 1
0032' 22 FD4D      ld     (actsec),hl
0035' 06 05      ld     b,5            ; read 5 sectors (full track)
;              ; 1-4 ccp.com 5 our tables

1000          table equ 1000h          ; this is where sector 5
;              ; appears in memory

0037' CD 015A      call    0100h+rsec   ; ask rom to read it in

003A' B7          or      a            ; see if any error
003B' C2 00CD*     jp      nz,booterr   ; yes, issue a msg and hang

;              ; (Note: still bank 0)
003E' AF          xor      a            ; turn rom off for a while now
003F' D3 00      out     (sys),a        ; while we look at what we just read

;              copy our tables from system track (last sector) just
;              after the ccp.com

;              the problem here is making sure that when the bios is
;              re-assembled the old data on disk doesn't get used
;              this is taken care of by verifying that the data on disk

```



```

0080" 01 1000          ld      bc,16*256      ; # of times to loop + dev = 0
0083"                initdip:                ; loop here to init each device
0083" C5              push     bc              ; save crnt value
0084" C0 0000:        call     devini          ; init device # in c
0087" C1              pop      bc
0088" 0C              inc      c              ; advance device #
0089" 10 F8          djnz     initdip         ; do all devices (0-15)

;      init the disk controllers

;      each dph prefix has a pointer to an init routine
;      this routine is called now to allow any one time
;      initialization.

008B" 21 0000:        ld      hl,drvtbl        ; pt to table of dph's
008E" 06 10          ld      b,16             ; do 16 disks
0090" 5E            inidisk: ld      e,(hl)      ; pick up ptr to dph
0091" 23            inc      hl
0092" 56            ld      d,(hl)
0093" 23            inc      hl              ; hl ready for next in tbl

0094" C5            push     bc              ; save count
0095" E5            push     hl              ; save drvtbl pointer
0096" 7A            ld      a,d              ; first, check to see if any dph
0097" B3            or       e              ; 0=no dph
0098" C4 00A2:        call     nz,inidisk2      ; there is a dph, init it
009B" E1            pop      hl              ; restore drvtbl pointer
009C" C1            pop      bc              ; restore count
009D" 10 F1          djnz     inidisk         ; and init all disk drives

009F" C3 00AB:        jp      ready           ; all init work taken care of

00A2"                inidisk2:                ; de=this drive's dph, jp to init

00A2" 21 FFF7        ld      hl,-9           ; offset for init addr
00A5" 19            add      hl,de           ; hl=addr of addr of init rtn
00A6" 7E            ld      a,(hl)          ; pick up init routine addr
00A7" 23            inc      hl
00A8" 66            ld      h,(hl)
00A9" 6F            ld      l,a              ; hl=init routine address

00AA" E9            jphl:   jp      (hl)      ; off to init routine

;      all done with init's, tell the world
00AB" ready:        ;      all ready to rip now!

00AB" 11 0000:        ld      de,signon       ; pt to the message
00AE" 21 0000:        ld      hl,conout       ; use this routine
00B1" 3E 80          ld      a,80h           ; force back to bank 0
00B3" D3 00          out     (sys),a

```



```
00B5"  CD 03A8"          call  outmsg          ; issue signon msg
                           ;
                           ;      copy 1st time fcb to 5ch to try to open
                           ;      execst.com
00B8"  01 0100          ld    bc,0100h          ; source 0, destination bank 1
00BB"  CD 0000*         call  xmove             ; set to just move in
00BE"  21 005C          ld    hl,fcbl           ; destination in bank 1
00C1"  11 0112"         ld    de,theftcb        ; source in bank 0
00C4"  01 0023          ld    bc,fcblen         ; length of an fcb
00C7"  CD 0000*         call  move              ; move it in

00CA"  C3 000B'         jp    wboot            ; blend to warm boot
```

page

```

;      subroutines and constants used during cboot only

;      booterr
;      come here when we got a disk error of any kind
;      reading our special sectors from the system
;      tracks (including ccp.com). An error msg is
;      issued and the system hangs.

00CD"      booterr:      ; issue a msg saying disk error during boot
00CD"      ld      hl,100h+cout      ; pt to crt output rtn
00D0"      ld      de,bootmsg      ; and special error msg
00D3"      cd 03A8"      call      outmsg      ; issue this message
00D6"      hang:      jr      hang      ; loop forever

00D8"      bootmsg:      ; the message on disk error w/in boot
00D8"      db      0dh,0ah,07      ; beeeeeep
00DB"      db      'BIOS boot error reading disk. '

00DF"      db
00E3"      db
00E7"      db
00EB"      db
00EF"      db
00F3"      db
00F7"      db
00F9"      db      'System reset required.'
00FD"      db
0101"      db
0105"      db
0109"      db
010D"      db
010F"      db      0dh,0ah,0      ; 00 ends the msg

;      thefcb
;      this is the fcb to load a special file
;      it is defined here and copied into bank 1
;      location 5c just before 1st wboot

005C      fcb      equ      5ch      ; where we will put it later

0112"      thefcb: db      0,'EXECST ','COM',0,0,0,0
0116"      db
011A"      db
011E"      db
0122"      ds      16
007C      fcbnr      equ      $-thefcb+fcbl      ; where the nr field will be
0132"      db      0,0,0
0023      fcblen      equ      $-thefcb      ; length of an fcb

```

page

```

0135"                                cseg      ; common

000B'                                wboot:
000B'  ED 7B 0000*                   ld        sp, (@bnkbf)      ; set up a stack

;                                     setup jumps in page 0 bank 1
000F'  3E 01                         ld        a, 1           ; force into bank 1
0011'  CD 0000*                       call       selmem        ; for execution and i/o

;                                     copy special page 0 common area (at 40h)
0014'  01 0100                       ld        bc, 0100h       ; source 0, destination bank 1
0017'  CD 0000*                       call       xmove        ; set to just move in
001A'  21 0040                       ld        hl, 40h        ; destination in bank 1
001D'  11 0000*                       ld        de, vers       ; source in bank 0
0020'  01 0010                       ld        bc, 16         ; maximum length of common area
0023'  CD 0000*                       call       move        ; move it in

;                                     set up jumps in page zero too
0026'  3E C3                         ld        a, 0c3h       ; a JMP opcode
0028'  32 0000                       ld        (0), a        ; at loc 0000h
002B'  32 0005                       ld        (5), a        ; and 0005h
002E'  21 0000*                       ld        hl, wboot@     ; pt to that entry of jmp tbl
0031'  22 0001                       ld        (1), hl       ; at loc 0001h
0034'  2A 0000*                       ld        hl, (@xtpa)    ; bdos entry pt from scb
0037'  22 0006                       ld        (6), hl       ; at loc 0006h

;                                     at this point, a is known to be non-zero

;                                     modified code
;                                     this xor is changed into an or
;                                     therefore, if the z flag is set, this is
;                                     the 1st time through wboot after a cboot
;                                     so we can try to load a file instead of ccp.com
;
003A'  AF                             once:  xor        a          ; modified to OR A after 1st

003B'  CC 0053'                       call       z, tryfile      ; one time ONLY

;                                     now, move the ccp into bank 1 from bank 0
;                                     and execute it
003E'  01 0100                       ld        bc, 0100h       ; source 0, destination bank 1
0041'  CD 0000*                       call       xmove        ; set to just move in

0044'  21 0100                       ld        hl, 100h       ; destination in bank 1
0047'  11 0000                       ld        de, 0          ; source in bank 0
004A'  01 0CFF                       ld        bc, ccplen     ; for length of ccp.com
004D'  CD 0000*                       call       move        ; move ccp in

0050'  C3 0100                       jp         tpa          ; and off to it

```

```

;      tryfile
;      one time only (at 1st wboot after cboot)
;      if the special file exists, use it.

0053'      tryfile:
0053'      3E B7      ld      a,0b7h      ; (OR A) disables this ever again
0055'      32 003A'   ld      (once),a    ; overlaying the XOR A does it

0058'      AF        xor      a          ; clear

0059'      32 0068      ld      (fcb+12),a    ; current extent
005C'      32 006B      ld      (fcb+15),a    ; record count
005F'      21 0000      ld      hl,0
0062'      22 007C      ld      (fcbnr),hl
0065'      11 005C      ld      de,fcb      ; pt to the fcb
0068'      0E 0F        ld      c,open      ; open the file
006A'      CD 0005      call     bdos
006D'      B7          or      a          ; was there an error?
006E'      C0          ret      nz        ; yes --- no special file

;      read the file into the tpa one logical sector at a time
;      this way it can be as long or short as need be

006F'      11 0100      ld      de,tpa      ; init where to read it
0072'      D5          filelp: push de      ; save across bdos call

0073'      0E 1A        ld      c,setdma    ; read into there
0075'      CD 0005      call     bdos

0078'      11 005C      ld      de,fcb      ; pt to fcb
007B'      0E 14        ld      c,read
007D'      CD 0005      call     bdos      ; read one more logical sector

0080'      D1          pop      de          ; restore dma address
0081'      21 0080      ld      hl,128
0084'      19          add     hl,de        ; bump dma address
0085'      EB          ex      de,hl       ; de=next dma address

0086'      B7          or      a          ; any errors??
0087'      28 E9        jr      z,filelp    ; no, keep reading
0089'      3D          dec     a          ; see if eof or error
008A'      C0          ret      nz        ; error, forget using the file

;      we now have read in the file with no errors

008B'      21 0000      ld      hl,0        ; dummy up
008E'      22 0080      ld      (80h),hl    ; to look like no cmd line

0091'      C3 0100      jp      tpa        ; execute it (ret will do ccp)

```

page

0094'

dseg

```

; the following are the std disk i/o err routines
; they are in this source module so that for the
; ldrbios they need not do anything

; These routines issue some message of the general form:
;     BIOS error xxxxxxing drive x: (text)
;     Accept this condition (y=yes)?
; The xxx above is either 'select', 'read' or 'writ'
; and the x is the alpha of the drive number. The
; (text) depends on the type of error.

; For a select error, all we can say is either the drive
; is not ready or the disk is not properly formatted.

; For an i/o error the (text) is the breakdown of the
; fdc status bits (in english) followed by: xx/yy-zz.zz
; Where:
;     xx      the fdc code in hex
;     yy      the number of sectors this i/o
;     zz.zz   the 8 bytes beginning at dmabank
;              bank(1),address(2),sector(1),
;              track(2),unit(1),type(1)

```

```

; stdsen
; err on std seldisk, say not ready
; return 'z' if no retry, else 'nz' to retry

```

```

0135"
0135" 3A 0000*
0138" 3C
0139" C8

```

```

stdsen:
    ld    a,(@errde)    ; 0ffh=no err msgs pls
    inc   a              ; is bdos passing errs back
    ret   z              ; yes (z=no retry)

    call  beep           ; honk horn and start a line

    ld    de,selfcn     ; select function
    call  saydrive       ; tell what drive w/msg

    ld    de,selmsg     ; text of select msg
    call  conmsg         ; issue text to console

    jp    askrc          ; ask if we should retry
                        ; z=no retry  nz=retry

```

```

013A" CD 0396"
013D" 11 03CB"
0140" CD 037A"
0143" 11 03EA"
0146" CD 0388"
0149" C3 032D"

```

```

; stderr
; std disk i/o read or write err
; return 'z' = no retry 'nz' = retry

```



```

;      a      error code to bdos (1 or 2)
;      b      fdc status from rom
;      e      rom rtn offset (rsec or wsec)

;      must return a and e as on input

014C"  4F      stderr: ld      c,a          ; save error code for bdos
014D"  3A 0000* ld      a,(@errde)      ; passing errors back to user
0150"  3C      inc      a              ;
0151"  79      ld      a,c          ; (error code passed back)
0152"  C8      ret      z            ; yes, no msgs (z=no retry)

0153"  C5      push     bc           ; save error codes
0154"  D5      push     de           ; save fcn code

0155"  CD 0396" call     beep        ; honk horn and start a line

0158"  D1      pop      de           ; get rd or wt rtn address
0159"  D5      push     de           ; must return de valid

015A"  3E 5A      ld      a,rsec      ; see if read or write
015C"  BB      cp      e            ; equal if read
015D"  11 03D2"   ld      de,rdfcn    ; assume it is
0160"  28 03      jr      z,isrd      ; yup
0162"  11 03D7"   ld      de,wtfcn    ; nope (must be a write)
0165"  CD 037A"   isrd: call     saydrive ; say function and drive

0168"  D1      pop      de           ; rom rtn addr
0169"  C1      pop      bc           ; b=fdc err code c=bdos err code
016A"  D5      push     de           ; save rom rtn for exit
016B"  C5      push     bc           ; and bdos err flag too

;      tell what fdc errors we got + a hex sequence

016C"  78      ld      a,b          ; fdc status
016D"  21 0553"   ld      hl,stattbl-1 ; pt to start of msg tbl

0170"  23      statlp: inc     hl      ; next tbl entry
0171"  CB 27      sla      a          ; check next bit left to right
0173"  30 15      jr      nc,statno   ; that one not on

0175"  F5      push     af           ; save fdc status (z=done)
0176"  16 00      ld      d,0        ;
0178"  5E      ld      e,(hl)        ; get this tbl entry
0179"  E5      push     hl          ; save crnt tbl addr
017A"  21 050D"   ld      hl,statmsg  ; pt to start of msgs proper
017D"  19      add      hl,de        ; pt to this message itself
017E"  EB      ex      de,hl        ; whew, de->msg for this bit
017F"  CD 0388"   call     conmsg     ; tell opr this one

0182"  11 055C"   ld      de,comma    ; separate status messages
0185"  CD 0388"   call     conmsg     ;
0188"  E1      pop      hl          ; tbl ptr
0189"  F1      pop      af          ; crnt fdc status

```

```

018A" 20 E4          statno: jr    nz,statlp    ; if more status bits

018C" 11 055F"       ld    de,codemsg    ; status decoded, show in hex
018F" CD 0388"       call   conmsg

0192" C1            pop    bc            ; restore fdc code (in c)
0193" C5            push   bc            ; and keep saving it back

;                  code=xx/yy
0194" 78            ld     a,b           ; fdc code
0195" CD 0363"       call   hexout        ; xx is fdc code
0198" 0E 2F         ld     c,'/'        ; weird separator
019A" CD 0000"       call   conout
019D" 3A 0000"       ld     a,(actcnt)    ; yy is #sectors this i/o
01A0" CD 0363"       call   hexout

;                  code=xx/yy-zz..zz
01A3" 0E 2D         ld     c,'-'        ; another separator
01A5" CD 0000"       call   conout
01A8" 21 FD4A       ld     hl,dmabank    ; start here to display zz..zz
01AB" 06 08         ld     b,8           ; for 8 bytes
01AD"               ; loop showing all of code
01AD" 7E           ld     a,(hl)         ; next byte to display
01AE" C5           push   bc
01AF" E5           push   hl
01B0" CD 0363"       call   hexout        ; displayed in hex
01B3" E1           pop    hl
01B4" C1           pop    bc
01B5" 23           inc    hl
01B6" 10 F5        djnz   codelp        ; display all bytes

01B8" 11 0568"       ld     de,codeend    ; issue trailer of code msg
01BB" CD 0388"       call   conmsg

01BE" CD 032D"       call   askrc         ; see whether to retry (nz=retry)
01C1" 20 20        jr     z,stddn        ; all done, no retry desired

;                  retry was wanted, see if media did change
01C3" 2A 2004       ld     hl,(adrive)    ; assume it was unit 0
01C6" 3A FD50       ld     a,(actdsk)     ; get our internal #
01C9" B7           or     a             ; was it a (unit 0)
01CA" 28 03        jr     z,was0         ; yes, hl proper ptr
01CC" 2A 2006       ld     hl,(bdrive)    ; no, pt to proper flag byte
01CF" 7E           ld     a,(hl)         ; see if that drive did change
01D0" B7           or     a
01D1" 28 0F        jr     z,retryok      ; it did not change, retry allowed

01D3" E5           push   hl            ; save ptr to media chg flag byte

01D4" 11 04C7"       ld     de,mediams    ; it did change, verify fm opr
01D7" CD 0388"       call   conmsg
01DA" CD 0336"       call   askyn        ; then ask for yes or no reply
;                  ; nz=yes (continue?)
01DD" E1           pop    hl            ; restore ptr to media chg byte

01DE" 28 03        jr     z,stddn        ; no means no retry attempt

```



```
01E0" AF          xor    a          ; force a=0
01E1" 77          ld     (hl),a      ; and clear media chg byte
                                   ; fall thru to allow retry

01E2"             retryok:          ; retry allowed, force nz
01E2" 3C          inc     a          ; nz=retry    z=no retry

                                   ; here to return to caller. z=no retry nz=retry
01E3" stddn:

                                   ; bc, de were saved earlier
01E3" C1          pop     bc
01E4" 79          ld     a,c        ; restore bdos err flag

01E5" D1          pop     de        ; restore rom rtn addr too

01E6" C9          ret              ; z/nz set by askrc

page
```

```

;      sernr
;      device not ready
;      issue the std bios error message, but
;      only to crt. This is important, because
;      conout may be the serial device.
;      device #, must be retained
;
;      ENTRY  B = Device #
;
;      EXIT   Z-flag = Set   Unassign this device
;             Z-flag = Clear Leave device assigned
;

01E7"  E5      SERNR:  PUSH  HL          ;Save caller's regs
01E8"  D5      PUSH  DE
01E9"  C5      PUSH  BC          ;Save device #

01EA"
01EA"  CD 0103  SERCLR:          ;Clear kbd 1st
01ED"  28 05      CALL  100h+SKEY  ;Is there a key
01EF"  CD 0106      JR      Z,SEROK  ;No, ready to issue msg
01F2"  18 F6      CALL  100h+CI     ;Yes, get the key
                                ;Then throw it away

01F4"  0E 07      SEROK:  LD      C,07H  ;Beep to notify user of error condition
01F6"  CD 0109      CALL  COUT+100H ;Use ROM routine

01F9"  C1      POP   BC          ;Device #
01FA"  C5      PUSH  BC

01FB"  78      LD     A,B        ;Get device # in A-reg
01FC"  87      ADD    A,A        ;A * 8 to get appropriate entry
01FD"  87      ADD    A,A
01FE"  87      ADD    A,A
01FF"  06 00      LD     B,0
0201"  4F      LD     C,A
0202"  21 0000*   LD     HL,DEVTLB ;Start of device table
0205"  09      ADD    HL,BC       ;HL = this device entry

                                ;HL is address of device name
0206"  11 0413"   LD     DE,DEVMSG ;Point to message position for device name
0209"  01 0006     LD     BC,6     ;Length of device name
020C"  ED B0      LDIR          ;Move it

020E"  CD 02EB"   CALL  SAVMSG    ;Save last video line in temp locations

0211"  21 0406"   LD     HL,DNRMSG ;Address of device not ready message
0214"  11 0456"   LD     DE,DNRATR ;Address of attributes for message
0217"  CD 030A"   CALL  RESTOR    ;Move 'em to last video line

021A"  C1      ISITAGAIN: POP   BC  ;Retrieve device #
021B"  C5      PUSH  BC          ;and save it again
021C"  CD 0000*   CALL  OUTST    ;Check to see if the device has become ready
021F"  20 15      JR     NZ,DONE  ;Ready, restore screen and leave device assigned

```

```

0221"  CD 0103          CALL 100H+SKEY      ;Check for response
0224"  28 F4            JR   Z,ISITAGAIN    ;Is there a key
                                           ;No, check again

0226"  CD 0106          CALL 100H+CI        ;Get key
0229"  E6 1F            AND   1Fh           ;Accept upper or lower case

022B"  FE 19            CP   'Y' and 1Fh    ;Is the response "Yes"?
022D"  28 07            JR   Z,DONE          ;Yes, unassign device and restore screen

022F"  FE 0E            CP   'N' and 1Fh    ;Is the response "No"?
0231"  20 E7            JR   NZ,ISITAGAIN   ;No, try again

0233"  3E 01            LD   A,1             ;Clear Z-flag to indicate leave device assigned
0235"  B7              DR   A

0236"  F5              DONE:  PUSH AF        ;Save Z-flag (exit state)

0237"  21 024B"         LD   HL,TEMPVID     ;Point to user's last screen line
023A"  11 029B"         LD   DE,TEMPATR     ;Point to last line's attributes
023D"  CD 030A"         CALL RESTOR         ;and move them back to screen

0240"  3A 0000*         LD   A,(DELAY)      ;Get user defined delay for next time
0243"  32 0000*         LD   (DELAY0),A    ;Save it

0245"  F1              POP   AF            ;Restore exit parameter
0247"  C1              POP   BC            ;Restore all regs
0248"  D1              POP   DE
0249"  E1              POP   HL
024A"  C9              RET
page

```

024B" TEMPVID: DS 80 ;Buffer to save last screen line
029B" TEMPATR: DS 80 ;Buffer to save attributes for last screen line

02EB" SAVEMSG:
; This routine saves the last line of the screen into temporary
; storage and its corresponding attributes
;
; ENTRY None
;
; EXIT TEMPVID contains last screen line
; TEMPATR contains last screen line attributes
;

02EB" DB 00 IN A, (SYS) ;Get current bank
02ED" 67 LD H,A ;For destination
02EE" 2E 40 LD L,40H ;Video bank for source
02F0" 22 FD56 LD (RSBANK),HL ;Set parameters

02F3" 21 CB80 LD HL,0CB80h ;Last line of video
02F6" 11 024B" LD DE,TEMPVID ;Destination of move
02F9" 01 0050 LD BC,80 ;Move complete line
02FC" CD 0325" CALL MOVEMSG ;Move it

02FF" 21 DB80 LD HL,0DB80h ;Last attribute line
0302" 11 029B" LD DE,TEMPATR
0305" 01 0050 LD BC,80
; CALL MOVEMSG
0308" 18 1B JR MOVEMSG ;Movemsg will return correctly
; RET
; PAGE

030A"

RESTOR:

```

;      This routine moves 80 bytes from HL pointer to the last video line
;      and 80 bytes from DE pointer to last line video attribute
;

```

```

;      ENTRY  HL = address of message
;              DE = address of attributes
;

```

```

;      EXIT   None
;

```

030A" D5

PUSH DE ;Save attribute pointer

030B" E5

PUSH HL ;Save message pointer

030C" 26 40

LD H,40H ;Video for destination bank

030E" DB 00

IN A,(SYS) ;Get current bank

0310" 6F

LD L,A ;For source

0311" 22 FD56

LD (RSBANK),HL ;Set parameters

0314" E1

POP HL ;Status message (source)

0315" 11 CB80

LD DE,0CB80H ;Last video line (destination)

0318" 01 0050

LD BC,80

031B" CD 0325"

CALL MOVMSG

031E" E1

POP HL ;Attribute pointer

031F" 11 DB80

LD DE,0DB80H

0322" 01 0050

LD BC,80

```

;      CALL MOVMSG ;Fall through
;

```

RET

PAGE

0325"

MOVMSG:

```
; Uses high ram routine to transfer data between banks
; must save IX-reg, as ROMMV uses IX to preform jump to
; High RAM routine (see EXCBIOS3)
;
; ENTRY HL = Source start address
;        DE = Destination start address
;        BC = # bytes to transfer
;
; EXIT  None
;
```

```
0325" DD E5
0327" CD 0000*
032A" DD E1
032C" C9
```

```
PUSH IX
CALL ROMMV
POP IX
RET
PAGE
```

```

;      askrc
;      ask opr to accept this error (only y = yes)
;      to conout, from conin

032D"  CD 038E"      askrc: call  crlf          ; start a new line 1st
0330"  11 04B2"      ld      de,dskmsg      ; pt to message
0333"  CD 0388"      call   conmsg          ; to console

0336"  11 04A6"      askyn: ld      de,yorn      ; append the y or n message
0339"  CD 0388"      call   conmsg

033C"                                     ; loop here to read from conin
033C"  E5            push  hl              ; save output rtn addr

033D"  CD 0000"      call   conin           ; get next keystroke
0340"  E1            pop    hl

0341"  CD 0347"      call   isity           ; see if answer is valid
0344"  D0            ret    nc              ; valid (Y or N)
0345"  18 F5            jr     ynagain       ; keep trying

0347"  11 0509"      isity: ld      de,ymsg      ; assume only yes one
034A"  E6 1F          and     1fh           ; make upper/lower/ctl same
034C"  FE 19          cp     'Y' and 1fh    ; is it Yes
034E"  28 09          jr     z,isyes        ; yup

0350"  11 0506"      ld      de,nmsg        ; may be no, pt to proper msg
0353"  FE 0E          cp     'N' and 1fh    ; not yes, was it No
0355"  28 03          jr     z,isno         ; it was no
0357"  37            scf                    ; show not a valid response
0358"  C9            ret                    ; and exit as is

0359"  B7            isyes: or     a          ; a=Y, set NZ to show yes
035A"              isno:          ; if no, a=N and Z is set
035A"  F5            push  af              ; save retry/continue flag

035B"  CD 03A8"      call   outmsg          ; tell what we saw opr enter
035E"  CD 0391"      call   newline        ; and complete the line

0361"  F1            pop    af              ; restore 'a'
0362"  C9            ret                    ; and all done

```

page

```

;      hexout
;      displays 'a' as two ascii characters
;      to conout

0363"  F5      hexout: push    af          ; save full value
0364"  0F              rrca              ; get left nibble 1st
0365"  0F              rrca
0366"  0F              rrca
0367"  0F              rrca
0368"  CD 036C"    call    hexify          ; display it
0368"  F1              pop     af          ; restore original

036C"                      hexify:          ; display the lower nibble in 'a'
036C"  E6 0F              and     0fh          ; make it a nibble
036E"  FE 0A              cp      0ah          ; is it a-f
0370"  38 02              jr      c,hex2        ; no, ok as is
0372"  C6 07              add     a,7          ; adjust for a-f
0374"  C6 30      hex2:  add     a,'0'          ; now valid hex character
0376"  4F              ld      c,a          ; prep to output it
0377"  C3 0000*         jp      conout        ; send it to console device
```

page


```

;      saydrive
;      display msg pointed to be de to conout,
;      then say 'ing drive x:'
;      to conout

037A"      saydrive:
037A"      CD 0388"      call    conmsg      ; 1st issue fcn name

037D"      11 03DC"      ld      de,drmsg      ; pt to drive message
0380"      3A 0000"      ld      a,(drive)      ; get bdos drive
0383"      C6 41          add     a,'A'          ; adjust to an alpha char
0385"      32 03E6"      ld      (drmsgdr),a      ; save in msg

0388"      21 0000"      conmsg: ld    hl,conout      ; pt to console output rtn
038B"      C3 03A8"      jp      outmsg      ; and output that message
;      msg text will follow

;      crlf
;      output crlf to conout

;      newline
;      output crlf to routine in hl

038E"      21 0000"      crlf:  ld    hl,conout      ; newline to conout

0391"      newline:
0391"      11 038B"      ld      de,acrlf      ; pt to msg
0394"      18 12          jr      outmsg      ; and do it

;      beep
;      crlf, honk the horn and say who we are
;      BIOS error .....
;      to conout
;      Note: also clear any waiting keys from conin 1st

0396"      CD 038E"      beep:  call    crlf      ; new line 1st

0399"      beepclr:      ; clear any waiting keys
0399"      CD 0000"      call    conist      ; is there anything waiting
039C"      28 05          jr      z,beepok      ; nothing waiting
039E"      CD 0000"      call    conin      ; something waiting, get it
03A1"      18 F6          jr      beepclr      ; and ignore it

03A3"      11 03BE"      beepok: ld    de,abeep      ; pt to bell + crlf
03A6"      18 E0          jr      conmsg

```

```

;      outmsg
;
;      send msg to some device up to 255 chars
;      in length and terminated by a 00h character.
;      this is intended for the initial logon msg,
;      disk error messages or device timeout msgs
;
;      note:  this resides in bank 0
;
;      bc      destroyed
;      de      ptr to start of msg (0 terminated)
;      hl      ptr to routine to output a char (rom or bios)
;              typically conout or cout (rom)
;              preserved on output
;
;      outlen
;
;      an alternate entry to output a message, but the
;      length is specified in 'b'.  The message may
;      not contain a 00h character.
;
;      b      # characters to send

```

```

03A8" 06 00      outmsg: ld      b,00h    ; maximum # of characters in msg

03AA"
03AA" 1A          outlen:                ; alternate entry: 'b' = msg length
03AB" B7          ld      a,(de)    ; next char
03AC" C8          or      a          ; is this a terminator?
03AD" 4F          ret      z        ; yes -- all done then
03AE" C5          ld      c,a        ; char to be output
03AF" D5          push    bc        ; save char count
03B0" E5          push    de        ; and ptr to char
03B1" CD 00AA"    push    hl        ; and routine addr
03B4" E1          call    jphl      ; call routine pt'd to by hl
03B5" D1          pop     hl
03B6" C1          pop     bc        ; all restored
03B7" 13          inc     de        ; next char
03B8" 10 F0      djnz    outlen    ; loop 'b' number of characters
03BA" C9          ret              ; done

```

page

```

; messages

; a crlf
03BB" 0D 0A 00 acrlf: db 0dh,0ah,0

; a beep (bell) followed by who we are
03BE" 07 abeep: db 07h
03BF" 42 49 4F 53 db 'BIOS error ',0
03C3" 20 65 72 72
03C7" 6F 72 20 00

; disk function names (followed by 'ing')

03CB" 73 65 6C 65 selfcn: db 'select',0
03CF" 63 74 00
03D2" 72 65 61 64 rdfcn: db 'read',0
03D6" 00
03D7" 77 72 69 74 wtfcn: db 'writ',0
03DB" 00

; generic drive name message (after function,
; before error text)

03DC" 69 6E 67 20 drmsg: db 'ing drive '
03E0" 64 72 69 76
03E4" 65 20
03E6" drmsgdr: ; drive char inserted here
03E6" 50 3A 20 00 db 'X: ',0

; select error text

03EA" 6E 6F 74 20 selmsg: db 'not ready or not formatted.',0
03EE" 72 65 61 64
03F2" 79 20 6F 72
03F6" 20 6E 6F 74
03FA" 20 66 6F 72
03FE" 6D 61 74 74
0402" 65 64 2E 00

; unassign msg for device not ready msg
0406" 42 49 4F 53 DNRMSG: DB 'BIOS error. '
040A" 20 65 72 72
040E" 6F 72 2E 20
0412" 20
0413" DEVMSG: DS 6 ;Space for device name
0419" 20 6E 6F 74 DB ' not ready or buffer full, unassign this device (Y or N)? '
041D" 20 72 65 61
0421" 64 79 20 6F
0425" 72 20 62 75
```

```

0429" 66 66 65 72
042D" 20 66 75 6C
0431" 6C 2C 20 75
0435" 6E 61 73 73
0439" 69 67 6E 20
043D" 74 68 69 73
0441" 20 64 65 76
0445" 69 63 65 20
0449" 28 59 20 6F
044D" 72 20 4E 29
0451" 3F 20
0453" A0
0454" 20 20

                                DB      0A0h          ;Simulate cursor
                                DB      ' '          ;Pad with spaces

0456" 0000 0000          DNRATR:    DW      0,0,0,0,0,0,0,0,0,0 ;No attributes set
045A" 0000 0000
045E" 0000 0000
0462" 0000 0000
0466" 0000 0000
046A" 0000 0000          DW      0,0,0,0,0,0,0,0,0,0
046E" 0000 0000
0472" 0000 0000
0476" 0000 0000
047A" 0000 0000
047E" 0000 0000          DW      0,0,0,0,0,0,0,0,0,0
0482" 0000 0000
0486" 0000 0000
048A" 0000 0000
048E" 0000 0000
0492" 0000 0000          DW      0,0,0,0,0,0,0,0,0,0
0496" 0000 0000
049A" 0000 0000
049E" 0000 0000
04A2" 0000 0000

;      part of the yes or no question
yorn:  db      ' (Y or N)? ',0

;      retry msg for a disk error
dskmsg: db      'Retry this operation',0

04B2" 52 65 74 72
04B6" 79 20 74 68
04BA" 69 73 20 6F
04BE" 70 65 72 61
04C2" 74 69 6F 6E
04C6" 00

;      retry desired, but media changed
mediamsg:
04C7"          db      1bh,'^WARNING:',1bh,'q '          ; blinking
04C7" 1B 5E 57 41
04CB" 52 4E 49 4E
04CF" 47 3A 1B 71
04D3" 20
04D4" 64 61 74 61          db      'data on disk inserted may be destroyed. Continue',0
04D8" 20 6F 6E 20

```

```

04DC" 64 69 73 6B
04E0" 20 69 6E 73
04E4" 65 72 74 65
04E8" 64 20 6D 61
04EC" 79 20 62 65
04F0" 20 64 65 73
04F4" 74 72 6F 79
04F8" 65 64 2E 20
04FC" 20 43 6F 6E
0500" 74 69 6E 75
0504" 65 00

```

```

; tell what opr said

```

```

0506" 4E 6F 00
0509" 59 65 73 00

```

```

nmsg: db 'No',0
ymsg: db 'Yes',0

```

```

; the following are the various fdc status messages

```

```

050D"
050D" 6E 6F 74 2D
0511" 72 65 61 64
0515" 79 00
0517" 77 72 69 74
051B" 65 2D 70 72
051F" 6F 74 65 63
0523" 74 65 64 00
0527" 66 61 75 6C
052B" 74 00
052D" 6E 6F 74 2D
0531" 66 6F 75 6E
0535" 64 00
0537" 63 72 63 2D
053B" 65 72 72 6F
053F" 72 00
0541" 6C 6F 73 74
0545" 2D 64 61 74
0549" 61 00
054B" 64 72 71 00
054F" 62 75 73 79
0553" 00

```

```

statmsg: ; start of table
fdcnr: db 'not-ready',0

fdcmp: db 'write-protected',0

fdcmf: db 'fault',0
fdcnf: db 'not-found',0

fdccrc: db 'crc-error',0

fdclst: db 'lost-data',0

fdcdrq: db 'drq',0
fdcbusy: db 'busy',0

```

```

0554"
0554" 00
0555" 0A
0556" 1A
0557" 20
0558" 2A
0559" 34
055A" 3E
055B" 42

```

```

stattbl: ; table of 1 byte offsets
db fdcnr-statmsg
db fdcmp-statmsg
db fdcmf-statmsg
db fdcnf-statmsg
db fdccrc-statmsg
db fdclst-statmsg
db fdcdrq-statmsg
db fdcbusy-statmsg

```

```

; a separator between status texts

```

```

055C" 2C 20 00

```

```

comma: db ', ',0

```

```
                                ; the following is the lead-in for the hex code of error
                                codemsg:
055F"                            db      1bh,')(code=',0          ; enter dim mode
055F"    1B 29 28 63
0563"    6F 64 65 3D
0567"    00
0568"                                codeend:
0568"    29 1B 28 00          db      ')',1bh,'(',0          ; end dim mode

                                end
```

Macros:

Symbols:

000D*	@BNKBF	0079*	@CIVEC	014E*	@ERMDE
0008*	@MEDIA	0035*	@MXTPA	0001*	@SEC
03BE"	ABEEP	03BB"	ACRLF	019E*	ACTCNT
FD50	ACTDSK	FD4D	ACTSEC	FD4E	ACTTRK
FD51	ACTTYP	2004	ADRIVE	2008	APROTO
032D"	ASKRC	0336"	ASKYN	0005	BDS
2006	BDRIVE	0396"	BEEP	0399"	BEEPCLR
03A3"	BEEPCK	00CD"	BOOTERR	00D0"	BOOTMSG
2009	BPROTO	00001'	CBOOT	0000"	CBOOT0
0CFF	CCPLEN	0026"	CCPLP	0006	CI
2000	CLKPTR	0568"	CODEEND	01AD"	CODELP
055F"	CODEMSG	055C"	COMMA	FDDA	COMMSP
039F*	CONIN	039A*	CONIST	0388"	CONMSG
038F*	CONOUT	0009	COUT	038E"	CRLF
0008	DEC	0241*	DELAY	0244*	DELAY0
0004	DEVBAUD	0001	DEVIN	0085*	DEVINI
0413"	DEVMSG	0002	DEVOUT	0008	DEVSER
0203*	DEVTBL	0010	DEVXACT	FD4A	DMABANK
0000*	DMABK	FD4B	DMADR	0456"	DNRATR
0406"	DNRMSG	0236"	DONE	000B	DPBCKS
0011	DPBLEN	0000*	DPBTBL	000E*	DPH0
0014*	DPH1	000B	DPHMF	0381*	DRIVE
03DC"	DRMSG	03E6"	DRMSGDR	008C*	DRVTL
000C	DSKD1	04B2"	DSKMSG	0005	DSKS1
FE1C	DSKSAVE	FE1C	DSKSP	005C	FCB
0023	FCBLEN	007C	FCBNR	054F"	FDCBSY
0537"	FDCRC	054B"	FDCDRQ	0541"	FDCLST
052D"	FDCNF	050D"	FDCNR	0527"	FDCWF
0517"	FDCWP	0072'	FILELP	0030	FLUSH
00D6"	HANG	2355	HERTZ	0374"	HEX2
036C"	HEXIFY	0363"	HEXOUT	0008	IBM
01EC	IDELAY	0002	IDELLEN	0080	IDEVLEN
00A2	IDEVSHD	0022	IDEVTBL	0020	IDRVLEN
0002	IDRVTL	000A	IDVILEN	00E2	IDVINI
0100	IDVLEN	00EC	IDVSTR	FD52	IEADR
0045	IEINP	003F	IEISTAT	0042	IEOSTAT
0048	IEOUT	01EE	IEXD	0090"	INIDSK
00A2"	INIDSK2	0083"	INITDLP	0076*	INITVEC
0040	ISHDLEN	021A"	ISITAGAIN	0347"	ISITY
035A"	ISNO	0165"	ISRD	0359"	ISYES
0000	IVERS	00AA"	JPHL	200A	KBDTBL
2002	MEDIA	04C7"	MEDIAMSG	004E*	MOVE
0325"	MOVMSG	0391"	NEWLINE	0506"	NMSG
0075"	NOTBLS	003A'	ONCE	000F	OPEN
03AA"	OUTLEN	03AB"	OUTMSG	021D*	OUTST
0051	PARINP	0054	PAROUT	004B	PISTAT
004E	POSTAT	FD57	RDBANK	03D2"	RDFCN
0014	READ	00AB"	READY	030A"	RESTOR
01E2"	RETRYOK	0002	ROMETX	FD44	ROMMOVE
0328*	ROMMV	00FC	ROMPROD	00FE	ROMVER
0001	ROMXON	FD56	RSBANK	005A	RSEC
0024*	RVERS	000C	SAISTAT	0012	SAOSTAT

02EB"	SAVEMSG	FDDA	SAVESP	037A"	SAYDRIVE
000F	SBISTAT	0015	SBOSTAT	03CB"	SELFEN
0012*	SELMEM	03EA"	SELMMSG	0060	SENDEN
0018	SERAIN	001E	SERADUT	001B	SERBIN
0021	SERBOUT	01EA"	SERCLR	01E7I"	SERNR
01F4"	SEROK	0024	SETBAUD	001A	SETDMA
002C	SETMULT	00AC*	SIGNON	0003	SKEY
0170"	STATLP	0500"	STATMSG	018A"	STATND
0554"	STATTBL	01E3"	STDDN	014CI"	STDERR
0135I"	STDSEN	0000	SYS	0003	SYSB
1000	TABLE	FDDC	TBUF	029B"	TEMPATR
024B"	TEMPVID	0112"	THEFCB	0100	TPA
0053*	TRYFILE	001E*	VERS	01CF"	WAS0
000BI'	WBOOT	002F*	WBOOT0	005D	WSEC
03D7"	WTFCN	0001	XEROX	0042*	XMOVE
0509"	YMSG	033C"	YNAGAIN	04A6"	YORN

No Fatal error(s)

.z80

title bios6 (excbios6) LDRBIOS cold / warm boots

```
; 1983-03-20    set for 1st release
; 1983-02-23    nop support of err msgs (disk, dev not rdy)
; 1983-02-22    change name romjmp to torom
```

```
;*****
;*              *
;*      EXECUTIVE BIOS      *
;*      for                *
;*      CP/M 3.XX          *
;*      by                  *
;*                          *
;*      Glenn S. Tenney     *
;*      COMPRO              *
;*      P.O. Box 5260        *
;*      San Mateo, CA 94402  *
;*                          *
;*      Copyright (c) 1983   *
;*      Osborne Computer Corporation *
;*                          *
;*****
```

```
;    this is a fairly stripped down bios allowing the
;    std bios to be used as much as possible.  This bios
;    is used ONLY for cpmldr and could be a single source
;    module, but uses the std bios disk drivers, jump table
;    and misc routines (move etc.).  Obviously, the boot
;    functions are different for this ldrbios.
```

```
;    This module written by Glenn S. Tenney
```

```
;    include EXCBIOSE.LIB
```

```
.list
```

```
public cboot,wboot
```

```
public area0      ; so we can know where i/o area is
extrn drvtbl      ; we have to change the drive 0 dph
extrn setbnk      ; init the bank for disk i/o too
public devtbl     ; the device table
```

```
public devini     ; device init
public conout,auxout,list ; output routines
public conost,auxost,listst ; output status
public conin,auxin ; input routines
public conist,auxist ; input status
public stdsen,stderr,stdnr ; err msg routines
```

```
extrn torom      ; off to rom
```

```

0000'                                cseg      ;      to be like rest of bios

0000'                                cboot:
0000'                                wboot:
0000'  DB 00                          in      a,(sys)      ; we have to dummy up the bcb's
0002'  CD 0000#                       call    setbnk      ; current bank
                                           ; this is bank for doing all disk i/o

0005'  2A 0000#                       ld      hl,(drvtbl)  ; pt to list of drives
0008'  11 0012                       ld      de,18
000B'  19                             add     hl,de        ; hl=dph0+18 ( dirbcb field )

000C'  01 0037'                       ld      bc,ldrbc0    ; now store in addr of 1st bcb
000F'  71                             ld      (hl),c
0010'  23                             inc     hl
0011'  70                             ld      (hl),b

0012'  23                             inc     hl          ; hl=dph0+ 20 ( dtabcb field )
0013'  01 0046'                       ld      bc,ldrbc1    ; now store in addr of 2nd bcb
0016'  71                             ld      (hl),c
0017'  23                             inc     hl
0018'  70                             ld      (hl),b

0019'  C9                             ret                ;      these don't need to do anything

```

page

```

;      just a very stripped down bios

;      the following are dummy error routines so that the same
;      disk drivers can be used by ldrbios

001A'      stdsen:
001A'      stderr:
001A'      stdnr:  cp      a      ; force zero flag (=no retry)
001B'      ret

;      devini
;      dummy serial device init

001C'      devini:
001C'      C9      ret      ;      nothing for now !!

;      conout, auxout and list
;      output the character to console crt

001D'      conout:
001D'      auxout:
001D'      list:

001D'      1E 09      ld      e,cout      ; rom to type a character
001F'      C3 0000*   jp      torom      ; output it and exit

;      conost, auxost, listst
;      all return aok now

0022'      conost:
0022'      auxost:
0022'      listst:

0022'      AF      xor      a      ; all ready
0023'      C9      ret      ; and done

;      conist, auxist
;      just check status

0024'      conist:
0024'      auxist:

0024'      1E 03      ld      e,skey      ; check keyboard

```

```

0026' C3 0000#           jp    torom           ; let rom do it

;      conin, auxin
;      get console character

0029'      conin:
0029'      auxin:

0029' 1E 06             ld    e,ci           ; get character
002B' C3 0000#           jp    torom           ; also let rom do it

;      devtbl
;      only a dummy table for cpmldr

002E'      devtbl:

002E' 43 52 54 20       db    'CRT '         ; device 0 crt
0032' 20 20
0034' 03 00             db    devin+devout,0 ; i/o device no baud

0036' 00               db    0               ; end of table

;      the following are the bcb's to be used during cpmldr ONLY
;      they are dummied up and used for a non-banked system for
;      booting from drive A: ONLY! ! ! ! !

0037'      ldrbcb0:           ; for dirbcb
0037' FF 00 00 00       db    0ffh,0,0,0,0,0
003B' 00 00
003D' 0000 0000         dw    0,0
0041' 0000"            dw    area0           ; i/o area
0043' 00               db    0
0044' 0000             dw    0               ; (link and bank for longest bcb)

0046'      ldrbcb1:           ; for dtabcb
0046' FF 00 00 00       db    0ffh,0,0,0,0,0
004A' 00 00
004C' 0000 0000         dw    0,0
0050' 0400"            dw    areal          ; and it's i/o area
0052' 00               db    0
0053' 0000             dw    0

0055'      dseg             ; so areas appear last (I hope)

;
;      WARNING
;
;      the i/o areas are equated so as to not take up

```

```

;          space on the system tracks. Therefore, this module
;          must be linked LAST ! ! ! ! !

0000"      area0: ds      1          ; the end of the ldrbios

0400"      area1 equ     area0+1024  ; the 2nd i/o area

                        end
```

Macros:

Symbols:

FD50	ACTDSK	FD4D	ACTSEC	FD4E	ACTTRK
FD51	ACTTYP	2004	ADRIE	2008	APROTO
0000I'	AREA0	0400"	AREA1	0029I'	AUXIN
0024I'	AUXIST	0022I'	AUXOST	001DI'	AUXOUT
2006	BDRIVE	2009	BPROTO	0000I'	CBOOT
0006	CI	2000	CLKPTR	FDDA	COMNSP
0029I'	CONIN	0024I'	CONIST	0022I'	CONOST
001DI'	CONOUT	0009	COUT	0008	DEC
0004	DEVBAUD	0001	DEVIN	001CI'	DEVINI
0002	DEVOUT	0008	DEVSER	002EI'	DEVTBL
0010	DEVXACT	FD4A	DMABANK	FD4B	DMADR
0006*	DRVTL	000C	DSKD1	0005	DSKS1
FE1C	DSKSAVE	FE1C	DSKSP	2355	HERTZ
0008	IBM	01EC	IDELAY	0002	IDELLEN
0080	IDEVLEN	00A2	IDEVSHD	0022	IDEVTBL
0020	IDRVLEN	0002	IDRVTL	000A	IDVILEN
00E2	IDVINI	0100	IDVSLEN	00EC	IDVSTR
FD52	IEADR	0045	IEINP	003F	IEISTAT
0042	IEDSTAT	0048	IEDUT	01EE	IEXD
0040	ISHDLEN	0000	IVERS	200A	KBDTBL
0037'	LDRBCB0	0046'	LDRBCB1	001DI'	LIST
0022I'	LISTST	2002	MEDIA	0051	PARINP
0054	PAROUT	004B	PISTAT	004E	POSTAT
FD57	RDBANK	0002	ROMETX	FD44	ROMMOVE
00FC	ROMPROD	00FE	ROMVER	0001	ROMXON
FD56	RSBANK	005A	RSEC	000C	SAISTAT
0012	SACSTAT	FDDA	SAVESP	000F	SBISTAT
0015	SBOSTAT	0060	SENDEN	0018	SERAIN
001E	SERROUT	001B	SERBIN	0021	SERBOUT
0024	SETBAUD	0003*	SETBNK	0003	SKEY
001AI'	STDERR	001AI'	STDNR	001AI'	STDSEN
0000	SYS	0003	SYSB	FDDC	TBUF
002C*	TOROM	0000I'	WBOOT	005D	WSEC
0001	XEROX				

No Fatal error(s)

```
title 'System Control Block Definition for CP/M3 BIOS'
```

```
public @civec, @covec, @aivec, @aovec, @lovec, @bnkbf
public @crdma, @crdsk, @vinfo, @resel, @fx, @usrcd
public @mltio, @ermde, @erdsk, @media, @bflgs
public @date, @hour, @min, @sec, ?erjmp, @mxtpa
```

```
FE00 =      scb$base equ    0FE00H      ; Base of the SCB

FE22 =      @CIVEC equ      scb$base+22h ; Console Input Redirection
                                           ; Vector (word, r/w)
FE24 =      @COVEC equ      scb$base+24h ; Console Output Redirection
                                           ; Vector (word, r/w)
FE26 =      @AIVEC equ      scb$base+26h ; Auxiliary Input Redirection
                                           ; Vector (word, r/w)
FE28 =      @AOVEC equ      scb$base+28h ; Auxiliary Output Redirection
                                           ; Vector (word, r/w)
FE2A =      @LOVEC equ      scb$base+2Ah ; List Output Redirection
                                           ; Vector (word, r/w)
FE35 =      @BNKBF equ      scb$base+35h ; Address of 128 Byte Buffer
                                           ; for Banked BIOS (word, r/o)
FE3C =      @CRDMA equ      scb$base+3Ch ; Current DMA Address
                                           ; (word, r/o)
FE3E =      @CRDSK equ      scb$base+3Eh ; Current Disk (byte, r/o)
FE3F =      @VINFO equ      scb$base+3Fh ; BDOS Variable "INFO"
                                           ; (word, r/o)
FE41 =      @RESEL equ      scb$base+41h ; FCB Flag (byte, r/o)
FE43 =      @FX equ         scb$base+43h ; BDOS Function for Error
                                           ; Messages (byte, r/o)
FE44 =      @USRCD equ      scb$base+44h ; Current User Code (byte, r/o)
FE4A =      @MLTIO equ      scb$base+4Ah ; Current Multi-Sector Count
                                           ; (byte, r/w)
FE4B =      @ERMDE equ      scb$base+4Bh ; BDOS Error Mode (byte, r/o)
FE51 =      @ERDSK equ      scb$base+51h ; BDOS Error Disk (byte, r/o)
FE54 =      @MEDIA equ      scb$base+54h ; Set by BIOS to indicate
                                           ; open door (byte, r/w)
FE57 =      @BFLGS equ      scb$base+57h ; BDOS Message Size Flag (byte, r/o)
FE58 =      @DATE equ       scb$base+58h ; Date in Days Since 1 Jan 78
                                           ; (word, r/w)
FE5A =      @HOUR equ       scb$base+5Ah ; Hour in BCD (byte, r/w)
FE5B =      @MIN equ        scb$base+5Bh ; Minute in BCD (byte, r/w)
FE5C =      @SEC equ        scb$base+5Ch ; Second in BCD (byte, r/w)
FE5F =      ?ERJMP equ      scb$base+5Fh ; BDOS Error Message Jump
                                           ; (word, r/w)
FE62 =      @MXTPA equ      scb$base+62h ; Top of User TPA
                                           ; (address at 6,7) (word, r/o)

0000      end
```

.z80

title boot (excboot) 1st sector rom bootstrap

```
; 1983-03-21 issue bios boot err msg on read errors
; 1983-03-20 set for 1st release
; 1983-02-21 boot hdr, enable key table
; 1983-02-17 clear screen before font load, len err msg
```

```
;*****
;*                                     *
;*          EXECUTIVE BIOS           *
;*          for                       *
;*          CP/M 3.XX                 *
;*          by                         *
;*                                     *
;*          Glenn S. Tenney           *
;*          COMPRO                     *
;*          P.O. Box 5260             *
;*          San Mateo, CA 94402       *
;*                                     *
;*          Copyright (c) 1983        *
;*          Osborne Computer Corporation *
;*                                     *
;******
```

```
; By: Glenn S. Tenney 1982-12-27
; this code is limited to 1 physical sector
; (or 1k) and must read in the ccpldr from the
; remainder of the 1st track
```

```
; system track layout
; track 0 this program sector 1 (max of 1k)
;          cpmlldr sector 2-5 (max of 4k)
; track 1 sector 1-4 ccp.com (actually only 3+k)
;          sector 5 bios overrides
; track 2 sectors 1-4 are character generation tables
;          sector 5 rom's ram overrides
```

```
; this bootstrap is read in by the rom into bank 0
; location 4000h and then reads the ccpldr into 4400h
```

```
; include EXCBIOSE.LIB
```

.list

0000'

aseq

```
; the following is the layout of the rom's ram
; override sector (track 2 sector 5)
```

F000

table equ 0f000h ; place in common to load a table

```
; 1st is rom's version number (not used)
```



```

F002          ikbdtbl equ    table+2          ; keyboard masks and decode tables
                                           ; function keys
034B          ikbdlen equ    11+320+512       ; length of all of that
F34D          ihz   equ      ikbdtbl+ikbdlen   ; frequency (50 or 60)
F34E          icur   equ      ihz+1           ; escape sequence char for cursor type
F34F          ibg    equ      icur+1          ; esc seq char for background type
F350          iclick equ      ibg+1           ; esc seq char for kbd click

```

```

4400          cpmldr equ     4400h           ; place cpmldr is linked to execute

```

```

org 4000h          ; must be here !!!

```

```

; the following is the header that MUST appear
; at the beginning of the 1st sector of the system
; disk. This header is used by the rom to verify
; that this is a valid system diskette and may be
; used in the future by this and other bios'es to
; determine diskette formats.

```

```

4000  00          nop          ; x'00' = system bootable disk
4001  C3 4007      jp    start  ; followed by jmp to real code

4004  02          db    2      ; product code (must match rom)
4005  00          db    0      ; feature code

```

```

; the next byte is a length byte.
; this tells how many table bytes follow.
; it is here for future expansion, since it is
; expected that the media type byte and the dpb for
; this diskette will be here. That will allow diskettes
; to be other than 01 format double density.
; It is zero now so that when the format is resolved
; future bios'es can tell the difference.

```

```

4006  00          db    0      ; number of table bytes following

```

```

4007  31 FD00      start: ld    sp,0fd00h      ; stack up in common

400A  DB 00          in    a,(sys)           ; current bank
400C  F6 80          or     80h              ; turn on rom
400E  D3 00          out   (sys),a

```

```

; reads in cpmldr from system track 0

4010 21 40FB      ld    hl,raminit    ; init parms in ram from here
4013 3E 04      ld    a,04          ; read 4 sectors (2-5)
4015 CD 40A5     call   read          ; read disk
4018 B7         or     a             ; any errors?
4019 C2 40B1     jp     nz,booterr    ; yes--issue a msg and hang

; reads in 4k of character sets from track 2

401C 21 4102     ld    hl,chrinit    ; init parms to read in char sets
401F 3E 04      ld    a,4           ; read both character sets
4021 CD 40A5     call   read          ; read disk
4024 C2 40B1     jp     nz,booterr    ; yes--issue a msg and hang
;                               font ram will be loaded a little later

; read in table of rom's ram to init

4027 21 4109     ld    hl,tblinit    ; disk parm table
4029 3E 01      ld    a,1           ; read 1 sector
402C CD 40A5     call   read          ; read disk
402F C2 40B1     jp     nz,booterr    ; yes--issue a msg and hang

; table is as follows
; rom version (2)          unused
; kbd decode (11+320)
; fcn keys (512)
; hertz (1)
; crt esc chars (3)       curtype,bg,click

4032 F3         di                ; just in case kbd is touched
4033 01 034B     ld     bc,11+320+512 ; copy keyboard masks, decode and function keys
4036 11 200A     ld     de,kbdtbl    ; where it is to be in rom's ram
4039 21 F002     ld     hl,ikbdtbl   ; from disk
403C ED B0      ldir              ; move it in

403E 06 08      ld     b,08h        ; bit on if 50hz
4040 3A F34D     ld     a,(ihz)      ; get hertz value
4043 32 2355     ld     (hertz),a    ; set into rom's ram
4046 FE 3C      cp     60           ; see how to init the port
4048 20 02      jr     nz,not50     ; 50hz means b is right
404A 06 00      ld     b,0          ; 60hz, so no bit gets set
404C DB 03      not50: in    a,(sysb) ; get crnt status
404E E6 F7      and    255-08h      ; clear 50/60 bit
4050 B0         or     b            ; and set to desired rate
4051 D3 03      out    (sysb),a     ; so it is all set

4053 FB         ei

; now ready to alter the screen etc.

4054 0E 1A      ld     c,lah        ; ctl-z = clear screen
4056 CD 0109     call   100h+cout    ; 1st clear the screen

```

; then copy font data to font memory

```

4059 21 5400      ld    hl,chrfont    ; from what we just read
405C 11 0000      ld    de,0        ; into font memory
405F 01 1000      ld    bc,4*1024    ; both fonts
4062 ED B0        ldir             ; just move it in
                                   ; (rom is already enabled)

```

; send escape sequences to set cursor type, bg type and click

```

4064 0E 1B        ld    c,1bh        ; escape
4066 CD 0109      call   cout+100h
4069 0E 2E        ld    c,'.'        ; period
406B CD 0109      call   cout+100h
406E 3A F34E      ld    a,(icur)     ; and initial cursor type
4071 4F           ld    c,a
4072 CD 0109      call   cout+100h

4075 0E 1B        ld    c,1bh
4077 CD 0109      call   cout+100h
407A 0E 78        ld    c,'x'
407C CD 0109      call   cout+100h
407F 3A F34F      ld    a,(ibg)      ; initial background
4082 4F           ld    c,a
4083 CD 0109      call   cout+100h

4086 0E 1B        ld    c,1bh
4088 CD 0109      call   cout+100h
408B 3A F350      ld    a,(iclick)   ; and initial keyboard click
408E 4F           ld    c,a
408F CD 0109      call   cout+100h

4092 0E 1B        ld    c,1bh
4094 CD 0109      call   cout+100h
4097 0E 65        ld    c,'e'        ; enable kbd table decode
4099 CD 0109      call   cout+100h

```

; all done, turn rom off

```

409C DB 00        in     a,(sys)      ; current bank
409E E6 7F        and    07fh        ; turn off rom
40A0 D3 00        out    (sys),a     ; all ready for cpmldr
40A2 C3 4400      jp     cpmldr      ; off to it now

```

page

```

;      subroutines etc....

;      read
;      hl=ptr to parms in our ram area here
;      a=number of sectors to read
;      setup disk parms and read disk

40A5  11 FD4A      read:  ld    de,dmabank    ; to common ram area
40A8  01 0007      ld    bc,ramilen    ; this many bytes
40AB  ED B0      ldir    ; copy
40AD  47      ld    b,a    ; copy number of sectors to read
40AE  C3 015A      jp     rsec+100h    ; let rom read now!

;      booterr
;      when we detect any disk error we come
;      here to issue a message and then just hang

40B1      booterr:      ; issue an error message
40B1  21 40C1      ld    hl,errmsg    ; pt to start of message
40B4  4E      errlp:  ld    c,(hl)    ; get next character of msg
40B5  79      ld    a,c    ; see if this is the end of the msg
40B6  B7      or     a    ; if so, forcibly loop forever
40B7  28 FB      jr     z,errlp    ; end ----> loop forever

40B9  23      inc    hl    ; advance to next character
40BA  E5      push   hl
40BB  CD 0109      call  cout+100h    ; send that character
40BE  E1      pop    hl
40BF  18 F3      jr     errlp    ; keep sending the msg

```

page

```

; equates and constants

; the error message we issue in case of any disk read error
; (a 00h byte ends the msg)

40C1 0D 0A 07      errmsg: db 0dh,0ah,07
40C4 42 49 4F 53      db 'BIOS boot error reading disk. '
40C8 20 62 6F 6F
40CC 74 20 65 72
40D0 72 6F 72 20
40D4 72 65 61 64
40D8 69 6E 67 20
40DC 64 69 73 6B
40E0 2E 20
40E2 53 79 73 74      db 'System reset required.'
40E6 65 6D 20 72
40EA 65 73 65 74
40EE 20 72 65 71
40F2 75 69 72 65
40F6 64 2E
40F8 0D 0A 00      db 0dh,0ah,00

40FB              raminit: ; table to init ram parms for disk read
40FB 00              db 00h ; bank
40FC 4400           dw cpmldr ; dma address
40FE 02             db 2 ; start at sector 2
40FF 0000           dw 0 ; of track 0
4101 00             db 0 ; from drive 0
0007              ramilen equ $-raminit ; length to init

5400              chrfont equ cpmldr+4096 ; after cpmldr comes fonts

4102              chrinit: ; table for reading character sets
4102 80             db 80h ; rom bank
4103 5400           dw chrfont ; where character sets are
4105 01             db 1 ; start sector 1
4106 0002           dw 2 ; track 2
4108 00             db 0 ; from drive 0

4109              tblinit: ; to init rom's other ram areas from disk
4109 80             db 80h ; any bank (really in common)
410A F000           dw table ; up in hi memory
410C 05             db 5 ; sector 5
410D 0002           dw 2 ; track 2
410F 00             db 0 ; drive 0

```

```
;      at the very end of this boot sector will be the
;      serialization info.  Therefore, show it takes up some
;      space in case the boot is too large.
```

```
0109      bootlen equ    $-start      ; length of this program

          if    bootlen and 0fc00h    ; if > 1k (1 sector)

          .'    error, boot is too large  '

          endif

          end
```

Macros:

Symbols:

FD50	ACTDSK	FD4D	ACTSEC	FD4E	ACTTRK
FD51	ACTTYP	2004	ADRIVE	2008	APROTO
2006	BDRIVE	40B1	BOOTERR	0109	BOOTLEN
2009	BPROTO	5400	CHRFONT	4102	CHRINIT
0006	CI	2000	CLKPTR	FDDA	COMNSP
0009	COUT	4400	CPMLDR	0008	DEC
0004	DEVBAUD	0001	DEVIN	0002	DEVOUT
0008	DEVSER	0010	DEVXACT	FD4A	DMABANK
FD4B	DMADR	000C	DSKD1	0005	DSKS1
FE1C	DSKSAVE	FE1C	DSKSP	40B4	ERRLP
40C1	ERRMSG	2355	HERTZ	F34F	IBB
0008	IBM	F350	ICLICK	F34E	ICUR
01EC	IDELAY	0002	IDELLEN	0080	IDEVLEN
00A2	IDEVSHD	0022	IDEVTBL	0020	IDRVLEN
0002	IDRVTBL	000A	IDVILEN	00E2	IDVINI
0100	IDVSLEN	00EC	IDVSTR	FD52	IEADR
0045	IEINP	003F	IEISTAT	0042	IEOSTAT
0048	IEOUT	01EE	IEXD	F34D	IHZ
034B	IKBDLEN	F002	IKBDTBL	0040	ISHDLEN
0000	IVERS	200A	KBDTBL	2002	MEDIA
404C	NOT50	0051	PARINP	0054	PAROUT
004B	PISTAT	004E	POSTAT	0007	RAMILEN
40FB	RAMINIT	FD57	RDBANK	40A5	READ
0002	ROMETX	FD44	ROMMOVE	00FC	ROMPROD
00FE	ROMVER	0001	ROMXON	FD56	RSBANK
005A	RSEC	000C	SAISTAT	0012	SAOSTAT
FDDA	SAVESP	000F	SBISTAT	0015	SBOSTAT
0060	SENDEN	0018	SERAIN	001E	SERADUT
001B	SERBIN	0021	SERBOUT	0024	SETBAUD
0003	SKEY	4007	START	0000	SYS
0003	SYSB	F000	TABLE	4109	TBLINIT
FD0C	TBUF	005D	WSEC	0001	XEROX

No Fatal error(s)

.z80

title wtssys (excwtsys) writes cpmldr.com to system tracks

;1983-06-23 Set for second release
; added initialization of track 1 sector 5 for device time-out

```
*****
;*                                     *
;*          EXECUTIVE BIOS           *
;*          for                       *
;*          CP/M 3.XX                *
;*          by                       *
;*                                     *
;*          Glenn S. Tenney          *
;*          COMPRO                   *
;*          P.O. Box 5250            *
;*          San Mateo, CA 94402     *
;*                                     *
;*          Copyright (c) 1983       *
;*          Osborne Computer Corporation *
;*                                     *
;*****
```

; this is really a very crude program designed for
; the infrequent need to create a diskette with
; new system tracks (0, 1 and sometimes 2) composed of:
; excboot, cpmldr, ccp, the special bios overlays,
; the fonts and the rom's ram overrides.

; Written by: Glenn S. Tenney

; include EXCBIOSE.LIB
.list

```
000F      open    equ    15
0014      read    equ    20
001A      setdma  equ    26
0009      pmsg    equ    9

0005      bdos    equ    5

003C      getdev  equ    3ch      ; bios fcn: hl=devtbl
0042      getdrv  equ    42h      ; bios fcn: hl=drvtbl
004B      move    equ    4bh      ; bios fcn: move
0057      xmove   equ    57h      ; bios fcn: set banks for move
```

```
0000'    31 5400      start: ld    sp,stack      ; pt to a stack

0003'    01 0015              ld    bc,codelen   ; len of our code
0005'    11 5000              ld    de,himem     ; destination
0009'    21 0192'         ld    hl,ourcode      ; from here
```



```

000C' ED B0          ldir          ; move a hunk to non-rom area

000E' 01 3C00        ld      bc,second-secl ; init entire area 1st
0011' 11 03EE'       ld      de,secl       ; i/o area
0014' 21 03DE'       ld      hl,initial    ; what to init it with
0017' ED B0          ldir          ; all cleared out

0019' 11 01AD'       ld      de,startmsg   ; tell what we're going to do
001C' CD 00FC'       call     msg

; the following are NOT optional files

001F' AF            xor      a
0020' 32 01A7'       ld      (optional),a  ; indicate not optional

0023' 11 0276'       ld      de,excboot    ; 1st fcb to open
0026' 21 03EE'       ld      hl,bootsec    ; and where to read it
0029' CD 0129'       call     dofile       ; open and read it in

002C' 11 02B2'       ld      de,cpmlldr    ; 2nd file
002F' 21 07EE'       ld      hl,ldrsec
0032' CD 0129'       call     dofile

; note: cpmlldr.com is limited in size

0035' 11 024B'       ld      de,ldrmsg     ; in case msg needed
0038' FE 21          cp      4096/128+1    ; see if cpmlldr too long
003A' D2 0172'       jp      nc,abort      ; yes, abort

003D' 11 02EE'       ld      de,ccp        ; 3rd file to read in
0040' 21 17EE'       ld      hl,ccpsec
0043' CD 0129'       call     dofile

; the following files are optional

0046' 3E FF          ld      a,-1
0048' 32 01A7'       ld      (optional),a  ; indicate now optional

004B' 11 032A'       ld      de,fontm     ; main font file
004E' 21 2BEE'       ld      hl,fmsec
0051' CD 0129'       call     dofile
0054' 32 01A8'       ld      (wtfm),a     ; 00=not there   xx=is there

0057' 11 0366'       ld      de,fonta     ; alt font file
005A' 21 33EE'       ld      hl,fasec
005D' CD 0129'       call     dofile
0060' 32 01A9'       ld      (wtfa),a     ; 00=not there   xx=is there

0063' 11 03A2'       ld      de,romram    ; final file is rom's ram override
0066' 21 3BEE'       ld      hl,romsec
0069' CD 0129'       call     dofile
006C' 32 01AA'       ld      (wtrom),a    ; 00=not there   xx=is there

; all files have been read in

; from the running system, build up the special bios

```

```

;      override sector. This is needed so that setup will
;      function. Of course setup could do this itself
;      when taking the current values from memory (the
;      running system), but.....

006F'  2A 0040      ld    hl,(40h)          ; bios version number
0072'  22 27EE'    ld    (ivers+oursec),hl      ; into our sector

0075'  3E 42      ld    a,getdrv          ; ask bios to get drvtbl addr
0077'  CD 00F0'    call   bios

007A'  11 27F0'    ld    de,drvtbl+oursec      ; where to put it
007D'  01 0020      ld    bc,drvlen
0080'  ED B0      ldir

0082'  3E 3C      ld    a,getdev          ; ask bios to get devtbl addr
0084'  CD 00F0'    call   bios

0087'  E5      push   hl          ; save to get prefix later
0088'  11 2810'    ld    de,idevtbl+oursec      ; where to put it
008B'  01 0080      ld    bc,idevlen          ; primary only
008E'  ED B0      ldir

0090'  01 0100      ld    bc,0100h          ; set banks src=00 dest=01
0093'  3E 57      ld    a,xmove
0095'  CD 00F0'    call   bios          ; set next move across banks

0098'  E1      pop    hl          ; restore devtbl addr
0099'  2B      dec    hl          ; back up to use prefix
009A'  56      ld    d,(hl)        ; get addr of initvector
009B'  2B      dec    hl
009C'  5E      ld    e,(hl)        ; shadow ptr is at devtbl-2
                                ; de is source for 'move'
009D'  21 2890'    ld    hl,idevshd+oursec      ; copy it to our sector
00A0'  01 014C      ld    bc,ishdlen+idvilen+idvslent+idellen
                                ; shadow+initvec+strings+time-out

00A3'  3E 4B      ld    a,move
00A5'  CD 00F0'    call   bios          ; ask bios to move x banks

;      entire area now all set up, just write it out

;      set up the ram for doing the i/o

;      track 0
00AB'  21 03EE'    ld    hl,track0          ; pt to data area
00AB'  11 0000      ld    de,0          ; and track number
00AE'  CD 0101'    call   write5          ; write it all (5 sectors)

;      track 1
00B1'  21 17EE'    ld    hl,track1
00B4'  11 0001      ld    de,1

```

```

00B7'  CD 0101'          call  write5

;      track 2

00BA'  3A 01A8'          ld     a,(wtfm)      ; see if writing main font
00BD'  21 2BEE'          ld     hl,fmsec      ; starting data area
00C0'  11 0002          ld     de,0002h      ; sector 1 (1-1=0) track 2
00C3'  B7              or     a              ; shd we write?
00C4'  3E 02           ld     a,2            ; if we do, write 2 sectors
00C6'  C4 0103'          call  nz,write      ; write only if something to write


00C9'  3A 01A9'          ld     a,(wtfa)      ; see if writing alt font
00CC'  21 3BEE'          ld     hl,fasec      ; starting data area
00CF'  11 0202          ld     de,0202h      ; sector 3 (3-1=2) track 2
00D2'  B7              or     a              ; shd we write?
00D3'  3E 02           ld     a,2            ; if we do, write 2 sectors
00D5'  C4 0103'          call  nz,write      ; write only if something to write


00D8'  3A 01AA'          ld     a,(wtrom)     ; see if writing rom's ram
00DB'  21 3BEE'          ld     hl,romsec     ; starting data area
00DE'  11 0402          ld     de,0402h      ; sector 5 (5-1=4) track 2
00E1'  B7              or     a              ; shd we write?
00E2'  3E 01           ld     a,1            ; if we do, write only 1 sector
00E4'  C4 0103'          call  nz,write      ; write only if something to write


00E7'  11 01E2'          ld     de,donemsg    ; that is it!
00EA'  CD 00FC'          call  msg

;      all done

00ED'  exit:
00ED'  C3 0000          jp     0              ; warm boot


;      call the bios with rtn addr (lo) in a

00F0'  32 00FA'          bios: ld     (biosl),a      ; set lo of bios addr
00F3'  3A 0002          ld     a,(2)            ; get hi of addr
00F6'  32 00FB'          ld     (biosh),a

;~~~~~
;      modified code follows
;~~~~~

00F9'  C3 0000          biosjp: jp     0              ; really jp to bios entry
00FA'  biosl equ biosjp+1 ; lo of bios fcn addr
00FB'  biosh equ biosl+1  ; and hi of addr


00FC'  0E 09          msg:  ld     c,msg          ; let bdos issue a msg

```

00FE' C3 0005

jp bdos

; pt'ed to by de

page

```

;      write
;      write all or part of a sector to disk
;      a      # sectors to write
;      e      track #
;      d      sector number-1 (0=1)
;      hl     ptr to dma address

;      write5
;      same, but sets a to 5 (full track)

```

```

0101' 3E 05      write5: ld      a,5          ; set to full track

0103' B7
0104' C8      write: or      a          ; see if anything to write
               ret      z          ; no, all done

0105' 47          ld      b,a          ; save # sectors to write
0106' 22 026F'    ld      (wtaddr),hl    ; set dma addr
0109' 7A          ld      a,d          ; get sector number
010A' 3C          inc      a          ; adj properly
010B' 32 0271'    ld      (wtsec),a
010E' 7B          ld      a,e          ; get track number
010F' 32 0272'    ld      (wttrk),a
0112' 78          ld      a,b          ; restore # sectors to write

0113' 01 0008    ld      bc,iolen      ; length of i/o init area
0116' 11 FD4A    ld      de,dmabank    ; in hi common ram
0119' 21 026E'    ld      hl,wtparms   ; to do some track
011C' ED B0      ldir                ; this is it

011E' 47          ld      b,a          ; get # sectors to write
011F' 3E 5D      ld      a,wsec        ; and the offset to write it
0121' CD 5000    call     torom
0124' B7          or      a          ; any errors writing?
0125' C8          ret      z          ; no, all done writing

0126' C3 016F'    jp      wterr        ; yes, then abort

```

page

```

;      dofile
;
;      open then read in a file
;      if "optional"() then file is optional
;      a      returned=#sectors actually read in
;      de     ptr to fcb to use
;      hl     dma address starting

0129'      dofile:      ; open then read this file
0129'      E5          push    hl      ; save dma addr
012A'      ED 53 01AB' ld      (thefcb),de ; and ptr to fcb

012E'      21 0001      ld      hl,1      ; copy fname and ftype
0131'      19          add     hl,de      ; from=fcb+1
0132'      11 01F8'      ld      de,fname ; copy in this file name
0135'      01 0008      ld      bc,8
0138'      ED B0        ldir
013A'      13          inc     de      ; bump over the '.'
013B'      01 0003      ld      bc,3      ; for file type
013E'      ED B0        ldir

0140'      ED 5B 01AB' ld      de,(thefcb) ; pt to file
0144'      0E 0F        ld      c,open    ; open the file
0146'      CD 0005      call    bdos
0149'      3C          inc     a      ; ff=not found
014A'      D1          pop     de      ; restore starting dma addr
014B'      28 2B        jr      z,badopen ; open error

014D'      06 FF        ld      b,-1      ; init # sectors read count

014F'      04          rdloop: inc     b      ; bump # sectors read
0150'      D5          push    de
0151'      C5          push    bc      ; save sector counter

0152'      0E 1A        ld      c,setdma ; set dma addr
0154'      CD 0005      call    bdos

0157'      ED 5B 01AB' ld      de,(thefcb) ; crnt fcb
015B'      0E 14        ld      c,read
015D'      CD 0005      call    bdos      ; read one 128 byte sector

0160'      C1          pop     bc      ; restore sector count
0161'      D1          pop     de
0162'      21 0080      ld      hl,128    ; advance dma addr too
0165'      19          add     hl,de
0166'      EB          ex      de,hl

0167'      B7          or      a      ; see if done or error
0168'      28 E5        jr      z,rdloop ; not done yet

016A'      3D          dec     a      ; 1=done
016B'      20 10        jr      nz,badread ; no, error

016D'      B0          or      b      ; pass back # sectors read in

```

```

016E'  C9                                ret                                ; all done reading, no errors

016F'  11 022B'                          wterr: ld      de,wtmmsg      ; write error (no fname)
0172'  CD 00FC'                          abort: call   msg
0175'  C3 00ED'                          jp      exit      ; and all done

0178'                                     badopen:
0178'  11 0219'                          ld      de,openmsg    ; issue msg
017B'  18 03                             jr      fabort

017D'                                     badread:
017D'  11 0208'                          ld      de,readmsg    ; issue msg

;~~~~~
;      fall through
;~~~~~

;      common open/read error handler
;      if "optional" then return a=0 and done, else error msg.

0180'                                     fabort:                                ; error opening then reading a file
0180'  CD 00FC'                          call   msg      ; isse msg
0183'  11 01F8'                          ld      de,fname    ; then say file name
0186'  CD 00FC'                          call   msg

0189'  3A 01A7'                          ld      a,(optional) ; is this file optional
018C'  B7                             or      a
018D'  CA 00ED'                          jp      z,exit      ; no, abort now
0190'  AF                             xor     a      ; yes, pass back a=0
0191'  C9                             ret      ; and done with file

```

page

```

;      torom
;      a=rom routine address
;      aka ourcode (before it is moved)
0192'      ourcode:      ; code which is moved above 4000h
0192'      ld      (thecall+1),a      ; alter the call
0195'      in      a,(sys)
0197'      or      80h      ; turn on rom
0199'      out     (sys),a
019B'      xcall: call 100h      ; call the rom
019E'      push    af      ; save
019F'      in      a,(sys)
01A1'      and     07fh      ; turn rom off
01A3'      out     (sys),a
01A5'      pop     af
01A6'      ret      ; and all done
0015      codelen equ $-ourcode      ; length to move up

5000      himem equ 5000h      ; move the code here

5000      torom equ himem      ; routine to call to rom
5009      thecall equ torom+(xcall-ourcode)      ; the call instruction

5400      stack equ himem+400h      ; someplace also in non-rom

```

page


```

;      misc variables etc....

01A7'  00      optional: db  0      ; non-zero if file is optional

01A8'  00      wtfm:  db  0      ; ? write fontm (0=no)
01A9'  00      wtfa:  db  0      ; ? write fonta (0=no)
01AA'  00      wtrom: db  0      ; ? write rom's ram (0=no)


01AB'  0276'    thefcb: dw      excboot      ; ptr to current fcb


01AD'          startmsg:
01AD'  0D 0A          db      0dh,0ah
01AF'  45 58 43 57          db      'EXCWTSYS will write 3.0 system tracks to drive B'
01B3'  54 53 59 53
01B7'  20 77 69 6C
01BB'  6C 20 77 72
01BF'  69 74 65 20
01C3'  33 2E 30 20
01C7'  73 79 73 74
01CB'  65 6D 20 74
01CF'  72 61 63 6B
01D3'  73 20 74 6F
01D7'  20 64 72 69
01DB'  76 65 20 42
01DF'  0D 0A 24          db      0dh,0ah,'$'


01E2'          donemsg:
01E2'  0D 0A          db      0dh,0ah
01E4'  53 79 73 74          db      'System tracks aok',0dh,0ah,'$'
01E8'  65 6D 20 74
01EC'  72 61 63 6B
01F0'  73 20 61 6F
01F4'  6B 0D 0A 24


01F8'          fname: ds      8      ; common name for messages
0200'  2E          db      '._'
0201'          ftype: ds      3
0204'  0D 0A 07 24          db      0dh,0ah,07h,'$'


0208'          readmsg:
0208'  0D 0A 45 72          db      0dh,0ah,'Error reading $'
020C'  72 6F 72 20
0210'  72 65 61 64
0214'  69 6E 67 20
0218'  24


0219'          openmsg:
0219'  0D 0A 55 6E          db      0dh,0ah,'Unable to open $'
021D'  61 62 6C 65
0221'  20 74 6F 20
0225'  6F 70 65 6E
0229'  20 24

```

```
0022B' 0D 0A 45 72          wtmsg: db      0dh,0ah,'Error writing system tracks',0dh,0ah,'$'
0022F' 72 6F 72 20
00233' 77 72 69 74
00237' 69 6E 67 20
0023B' 73 79 73 74
0023F' 65 6D 20 74
00243' 72 61 63 6B
00247' 73 0D 0A 24
```

024B'	0D 0A 45 72	ldrmq: db	0dh,0ah,'Error: cpmlr.com is too long'
024F'	72 6F 72 3A		
0253'	20 63 70 6D		
0257'	6C 64 72 2E		
025B'	63 6F 6D 20		
025F'	69 73 20 74		
0263'	6F 6F 20 6C		
0267'	6F 6E 67		
026A'	07 0D 0A 24	db	7,0dh,0ah,'\$'

```
;      this is set of parms passed to the rom
;      to write 1-5 sectors
```

```

026E'          wtparms:
026E'  01              db      1          ; bank
026F'  0000          wtaddr: dw      0          ; addr
0271'  01              wtsec: db      1          ; sector
0272'  0000          wtrrk:  dw      0          ; track
0274'  01              db      1          ; unit 'B'
0275'  0C              db      0ch         ; type
0008          iolen  equ      $-wtparms      ; length to move up there

```

```
00276'          excboot:
00276'    00 45 58 43             db      0,'EXCBOOT COM'
0027A'    42 4F 4F 54
0027E'    20 43 4F 4D
00282'    0000 0000             dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

0286'	0000	0000
028A'	0000	0000
028E'	0000	0000
0292'	0000	0000
0296'	0000	0000
029A'	0000	0000
029E'	0000	0000
02A2'	0000	0000
02A6'	0000	0000
02AA'	0000	0000
02AE'	0000	0000

```
02B2'          cpmldr:
02B2' 00 43 50 4D          db      0,'CPMLDR COM'
02B6' 4C 44 52 20
02BA' 20 43 4F 4D
```

```

02BE' 0000 0000          dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
02C2' 0000 0000
02C6' 0000 0000
02CA' 0000 0000
02CE' 0000 0000
02D2' 0000 0000
02D6' 0000 0000
02DA' 0000 0000
02DE' 0000 0000
02E2' 0000 0000
02E6' 0000 0000
02EA' 0000 0000

```

```

02EE'                                ccp:
02EE' 00 43 43 50          db      0,'CCP      COM'
02F2' 20 20 20 20
02F6' 20 43 4F 4D
02FA'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
02FE'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0302' 0000 0000
0306' 0000 0000
030A' 0000 0000
030E' 0000 0000
0312' 0000 0000
0316' 0000 0000
031A' 0000 0000
031E' 0000 0000
0322' 0000 0000
0326' 0000 0000

```

```

032A'                                fontm:
032A' 00 46 4F 4E          db      0,'FONTM   COM'
032E' 54 4D 20 20
0332' 20 43 4F 4D
0336'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
033A'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
033E'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0342'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0346'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
034A'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
034E'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0352'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0356'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
035A'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
035E'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0362'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

```

```

0366'                                fonta:
0366' 00 46 4F 4E          db      0,'FONTA   COM'
036A' 54 41 20 20
036E' 20 43 4F 4D
0372'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0376'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
037A'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
037E'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0382'                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

```

0386'	0000	0000
038A'	0000	0000
038E'	0000	0000
0392'	0000	0000
0396'	0000	0000
039A'	0000	0000
039E'	0000	0000

03A2'	
03A2'	00 52 4F 4D
03A6'	52 41 4D 20
03AA'	20 43 4F 4D
03AE'	0000 0000
03B2'	0000 0000
03B6'	0000 0000
03BA'	0000 0000
03BE'	0000 0000
03C2'	0000 0000
03C6'	0000 0000
03CA'	0000 0000
03CE'	0000 0000
03D2'	0000 0000
03D6'	0000 0000
03DA'	0000 0000

```
romram:
    db      0,'ROMRAM COM'

    dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

03DE'				
03DE'	52	65	73	65
03E2'	72	76	65	64
03E6'	52	65	73	65
03EA'	72	76	65	64

```
initial:                                ; do not change this
; db 'Glenn S. Tenney '
db 'ReservedReserved'                  ; oh well, fill the unused
```

```
; areas on the system
; tracks with something
```

03EE'
07EE'
08EE'
0FEE'
13EE'
03EE'

```
sec1    equ    $          ; start of i/o areas  ( 3 tracks long)
sec2    equ    sec1+1024
sec3    equ    sec2+1024
sec4    equ    sec3+1024
sec5    equ    sec4+1024
track0  equ    sec1
```

17EE'
18EE'
1FEE'
23EE'
27EE'
17EE'

```
sec6    equ    sec5+1024        ; sector 1 track 1
sec7    equ    sec6+1024
sec8    equ    sec7+1024
sec9    equ    sec8+1024
sec10   equ    sec9+1024
track1  equ    sec6
```

2BEE'
2FEE'
33EE'
37EE'
3BEE'
2BEE'

```
sec11 equ sec10+1024 ; sector 1 track 2
sec12 equ sec11+1024
sec13 equ sec12+1024
sec14 equ sec13+1024
sec15 equ sec14+1024
track2 equ sec11
```

```

3FEE'          secend equ    sec15+1024

03EE'          bootsec equ   sec1          ; excboot sec 1 trk 0
07EE'          ldrsec equ    sec2          ; cpmlr sec 2 trk 0
17EE'          ccpsec equ    sec6          ; ccp sec 1 tk 1
27EE'          oursec equ    sec10         ; where special stuff goes
2BEE'          fmsec equ     sec11         ; where main font begins
33EE'          fasec equ     sec13         ; where alt font begins
3BEE'          romsec equ    sec15         ; rom's ram override

                end

```

Macros:

Symbols:

0172'	ABORT	FD50	ACTDSK	FD4D	ACTSEC
FD4E	ACTTRK	FD51	ACTTYP	2004	ADRIVE
2008	APROTO	0178'	BADOPEN	017D'	BADREAD
0005	BDOS	2006	BDRIVE	00F0'	BIOS
00FB'	BIOSH	00F9'	BIOSJP	00FA'	BIOSL
03EE'	BOOTSEC	2009	BPROTO	02EE'	CCP
17EE'	CCPSEC	0006	CI	2000	CLKPTR
0015	CODELEN	FDDA	COMNSP	0009	COUT
02B2'	CPMLDR	0008	DEC	0004	DEVBAUD
0001	DEVIN	0002	DEVOUT	0008	DEVSER
0010	DEVXACT	FD4A	DMABANK	FD4B	DMADR
0129'	DOFILE	01E2'	DONEMSG	000C	DSKD1
0005	DSKS1	FE1C	DSKSAVE	FE1C	DSKSP
0276'	EXCBOOT	00ED'	EXIT	0180'	FABORT
33EE'	FASEC	2BEE'	FMSEC	01F8'	FNAME
0366'	FONTA	032A'	FONTM	0201'	FTYPE
003C	GETDEV	0042	GETDRV	2355	HERTZ
5000	HIMEM	0008	IBM	01EC	IDELAY
0002	IDELLEN	0080	IDEVLEN	00A2	IDEVSHD
0022	IDEVTBL	0020	IDRVLEN	0002	IDRVTBL
000A	IDVILEN	00E2	IDVINI	0100	IDVSLEN
00EC	IDVSTR	FD52	IEADR	0045	IEINP
003F	IEISTAT	0042	IEDSTAT	0048	IEDUT
01EE	IEDX	03DE'	INITIAL	0008	IOLN
0040	ISHDLEN	0000	IVERS	200A	KBDTBL
024B'	LDRMSG	07EE'	LDRSEC	2002	MEDIA
004B	MOVE	00FC'	MSG	000F	OPEN
0219'	OPENMSG	01A7'	OPTIONAL	0192'	OURCODE
27EE'	OURSEC	0051	PARINP	0054	PAROUT
004B	PISTAT	0009	PMSG	004E	POSTAT
FD57	RDBANK	014F'	RDLOOP	0014	READ
0208'	READMSG	0002	ROMETX	FD44	ROMMOVE
00FC	ROMPROD	03A2'	ROMRAM	3BEE'	ROMSEC
00FE	ROMVER	0001	ROMXON	FD56	RSBANK
005A	RSEC	000C	SAISTAT	0012	SADSTAT
FDDA	SAVESP	000F	SBISTAT	0015	SBOSTAT
03EE'	SEC1	27EE'	SEC10	2BEE'	SEC11
2FEE'	SEC12	33EE'	SEC13	37EE'	SEC14
3BEE'	SEC15	07EE'	SEC2	0BEE'	SEC3
0FEE'	SEC4	13EE'	SEC5	17EE'	SEC6
1BEE'	SEC7	1FEE'	SEC8	23EE'	SEC9
3FEE'	SECEND	0060	SENDEN	0018	SERAIN
001E	SERADUT	001B	SERBIN	0021	SERBOUT
0024	SETBAUD	001A	SETDMA	0003	SKEY
5400	STACK	0000'	START	01AD'	STARTMSG
0000	SYS	0003	SYSB	FDDC	TBUF
5009	THECALL	01AB'	THEFCB	5000	TOROM
03EE'	TRACK0	17EE'	TRACK1	2BEE'	TRACK2
0103'	WRITE	0101'	WRITES	005D	WSEC
026F'	WTADDR	016F'	WTERR	01A9'	WTFA
01A8'	WTFM	022B'	WTMSG	026E'	WTPARMS
01AA'	WTROM	0271'	WTSEC	0272'	WTRK

019B* XCALL 0001 XEROX 0057 XMOVE

No Fatal error(s)

5078 AUXIN	5073 AUXIST	5071 AUXOST	506C AUXOUT
504F CBOOT	5078 CONIN	5073 CONIST	5071 CONOST
506C CONOUT	506B DEVINI	507D DEVTBL	4E6C DRVTLBL
4FEB FLUSH	4FDD HOME	506C LIST	5071 LISTST
4F4E MOVE	4FED MULTIO	502F READ	5107 RVERS
4FF6 SECTRN	5029 SELDSK	4FA6 SELMEM	4FD9 SETBNK
4FD3 SETDMA	4FE5 SETSEC	4FE0 SETTRK	50A4 SIGNON
4FCE TIME	4FA9 TOROM	5103 VERS	504F WBOOT
4E03 WBOOT0	5034 WRITE	4F9D XMOVE	4ED0 DPBTBL
4E95 DPH0	4EB7 DPH1	517F STDINI	51EB STDRD
5180 STDSEL	51F0 STDWT	4FD1 DBANK	5047 DMABK
4FD2 KLUDGE	4F89 ROMMV	4FD0 SBANK	5041 ACTCNT
503D DRIVE	5069 STDERR	5069 STDSEN	5274 AREA0
5069 STDNR			
01A9 AUXIN	019A AUXIST	018B AUXOST	0176 AUXOUT
033C CBOOT	01AF CONIN	01A0 CONIST	0191 CONOST
017C CONOUT	016A DEVINI	01BA DEVTBL	006C DRVTLBL
02D8 FLUSH	02CA HOME	0170 LIST	0185 LISTST
023B MOVE	02DA MULTIO	031C READ	0463 RVERS
02E3 SECTRN	0316 SELDSK	0293 SELMEM	02C6 SETBNK
02C0 SETDMA	02D2 SETSEC	02CD SETTRK	0400 SIGNON
02BB TIME	0296 TOROM	045F VERS	0347 WBOOT
0003 WBOOT0	0321 WRITE	028A XMOVE	00D0 DPBTBL
0095 DPH0	00B7 DPH1	0888 STDINI	08F4 STDRD
0889 STDSEL	08F9 STDWT	FE26 @AIVEC	FE28 @AOVEC
FE22 @CIVEC	FE24 @COVEC	FE2A @LOVEC	0805 DELAY
0577 DELAY0	06FB INITVE	05F5 OUTST	0864 SERNR
04BA VECPTR	02BE DBANK	0334 DMABK	02BF KLUDGE
0276 ROMMV	02BD SBANK	032E ACTCNT	032A DRIVE
0AC9 STDERR	0AB2 STDSEN	FE35 @BNKBF	FE4B @ERMDE
FE54 @MEDIA	FE62 @MXTPA	FE5C @SEC	FE3C @CRDMA
FE3E @CRDSK	FE3F @VINFO	FE41 @RESEL	FE43 @FX
FE44 @USRCD	FE4A @MLTIO	FE51 @ERDSK	FE57 @BFLGS
FE58 @DATE	FE5A @HOUR	FE5B @MIN	

B2)

