TEXAS INSTRUMENT INC.
PROFESSIONAL COMPUTER

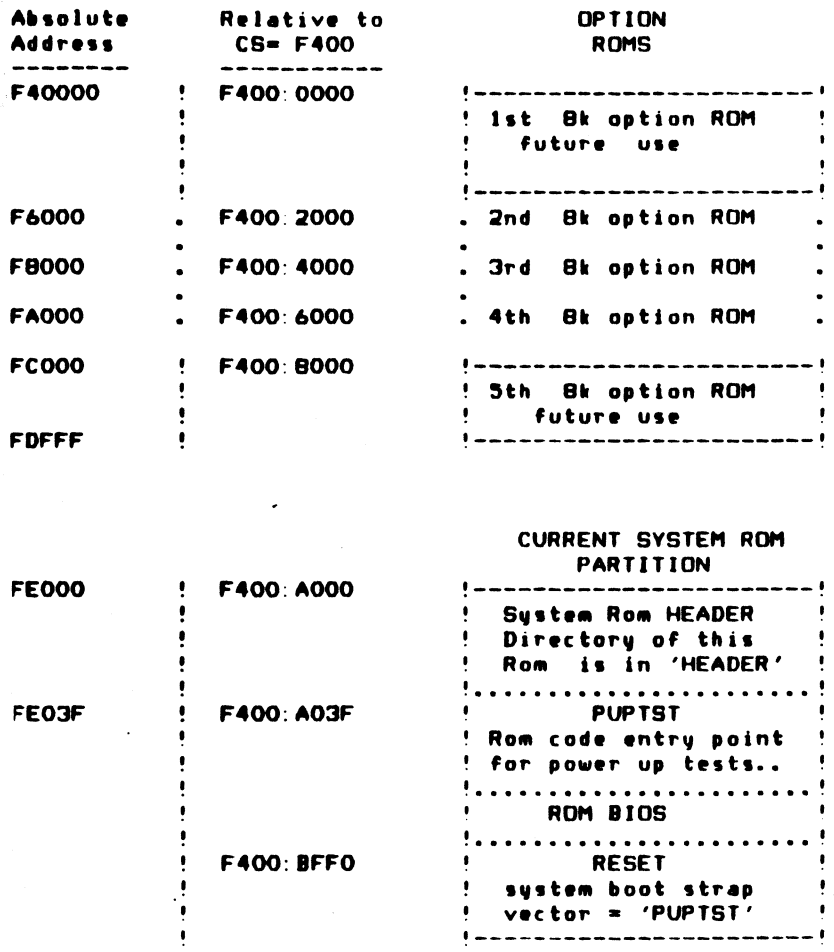
SYSTEM ROM LISTING

Version 1.23

CONTENTS

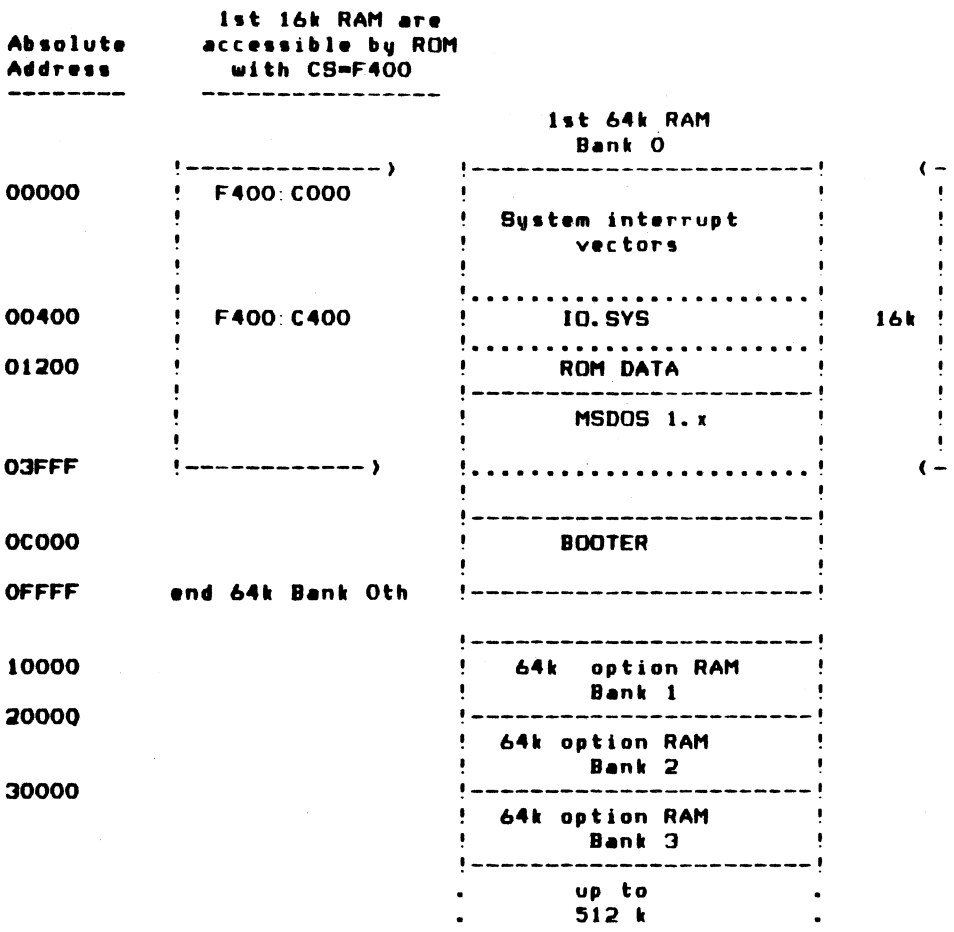
SECTION A.

SYSTEM ROM ORGANIZATION



The TIPC System memory Organization is defined in the 'SYSORG' module, wich must be the first module of the input link control file. This module does not allocate memory but assigns the Rom and Ram symbols.

SYSTEM RAM ORGANIZATION



Note that the location of ROMDAT is initially 40:0000. During the boot process ROMDAT is relocated in memory immediately after the location of RAM BIOS.

To determine the specific location of the ROMDAT(after boot), read the contents of the first word in software interrupt 60 (segment address) (absolute location 180H). The size of ROMDAT is indicated by the second word of software interrupt 60 (absolute location 182H). The contents of this software interrupt is set by the RAM BIOS loader (IO.SYS).

SECTION D.

MODULE LISTING INDEX (alphabetical order)

```
                                1   ;*************************************************************************
                                2   ;       The BIOS routines should be called through software interrupts.  *
                                3   ;       Each module includes a header describing the correct calling    *
                                4   ;       sequence.  Applications should not access the BIOS through      *
                                5   ;       absolute addresses.  Incompatibility with future TI releases can*
                                6   ;       result from incorrect use of the BIOS.                          *
                                7   ;*************************************************************************
                                8
                                9
Missing END statement
                               10

     No errors detected
```

```
                    ****************************
                    *       OUTPUT  MAP FILE     *
                    ****************************

Section    Type  Status     Base        Length      Range

.REL.       Rel  Float      0           0           0          FFFFF
ABS0        Rel  Fixed      0           C200        0          FFFFF
ROMDAT      Rel  Fixed      400         140         0          FFFFF
CRTDAT      Rel  Fixed      DE000       1820        0          FFFFF
ROMCOD      Rel  Fixed      F4000       BFDA        0          FFFFF
RESET       Rel  Fixed      FFFF0       10          0          FFFFF


Global        Value          Global        Value          Global        Value

ALTMON        FFFC3          ARGHHH        FFC9C          ATMERR        1
ATTERR        2              ATTSAV        439            BEEP          FEB84
BEEPIO        FEAFE          BEEPOF        FEB6E          BELEVT        420
BOOTLO        C000           BOOTMV        C000           BOOTSZ        200
CFG$IO        FFA6D          CINTER        20             CLK$IO        FEB9B
CLKSRV        FEBE5          CRAMER        8              CRT$IO        FEC1A
CRTCNT        430            CRTERR        437            CRTHLD        438
CRTINI        FE78F          CRTOUT        FEC52          CRTRET        FEC80
CRTTST        FE636          CURERR        4              CURPOS        42C
CURTYP        432            D$DBGN        462            D$DEND        4A5
D$DIT         462            D$EKNT        482            D$NCMD        483
D$NSTA        484            D$POS         485            D$REQ         48D
D$SOFT        49E            D$STAT        49F            D$WAIT        4A0
D$WKSP        4A1            DATE          53E            DBCERR        38
DCALL         FF286          DCRERR        32             DDMERR        39
DECLED        FFD71          DELAY         FFD66          DFINIS        FF293
DFMERR        37             DISBEQ        428            DISEND        42A
DKBOOT        FEA47          DKSPSV        4AC            DKSSSV        4AA
DNIERR        30             DNRERR        31             DNSERR        36
DRVTST        FE583          DSK$IO        FFOD9          DSKC$M        400
DSKERR        33             DSKEVT        408            DSKINI        FF299
DSKISP        516            DSKISR        FF325          DSKS$M        402
DSKTST        FE4A4          DSNERR        34             DSPDIE        FFB73
DSPERR        FFC04          DSPERZ        FFBF8          DSPKER        FFBFC
DUNERR        35             E$CMD         8              E$EVEN        1
E$ODD         2              E$REQ         4              ENDQ          462
EPROM         FC000          FATAL         FFC9A          FATERR        50
FILVEC        FE476          FL1EVT        410            FL2EVT        418
FLIERR        8              FLPDIR        FF475          FLUSH         FF77F
FRCINT        FF49F          QCONF1        FFA84          QCONFQ        FFA7E
HOURS         53D            HUNS          53A            ICNERR        0
INICRT        FE5ED          INILST        FE7B0          INIRET        FFD65
INITST        FE190          INIXIT        FFD42          INVALI        FFFF
IVIERR        1              IXSPSV        484            IXSSSV        482
KB$ALT        FFEC6          KBCTRL        FFE5E          KBFUNC        FFF2E
KBMODE        43F            KBNUMK        FFFA9          KBSHFT        FFDF6
```

```
;
;   Utilities
;
input   OUTPUT  ;  FF873         F400:B873       Screen display servive
input   INTXIT  ;  FFD42         F400:BD42       Common interrupts exit
input   ROMUTL  ;  FFD66         F400:BD66       System Rom Utilities
;
;   Tables
;
input   KBTABL  ;  FFD8E         F400:BD8E       Key board encoding tables
input   CRTTBL  ;  FFFB3         F440:BFB3       CRT initialization tables
;
input   ROMDAT  ;  00400         0000:0400       Rom data is Ram resident
input   REBET   ;  FFFF0         F400:BFF0       Boot strap vector


;
;                       Define the sections
;
BASE AB80=00000
BASE ROMDAT=400
BASE CRTDAT=DE000
BASE ROMCOD=F4000
BASE REBET=FFFF0
```

KBSPSV	4A8	KBSSSV	4A6	KBUNSH	FFD8E
KEY$IO	FF719	KEYERR	10	KEYIN	FF741
KEYINI	FF810	KEYISP	4E6	KEYISR	FF857
KEYOUT	FF79C	KEYRST	FF82B	KEYTST	FE8A1
KNIERR	10	KNRERR	11	KPAUSE	FF83A
KRAERR	12	KRCERR	15	KROERR	13
KUNERR	14	LBEEP	FEB7C	LED000	7
LED001	6	LED010	5	LED011	4
LED100	3	LED101	2	LED110	1
LED111	0	MINS	53C	MOTOFF	FF515
MSCTST	FE7DD	MSG	FFCD4	NMIERR	2
NMIGO	FFC45	NUMCNT	441	NUMDRV	8
NUMVAL	440	OPRERR	20	PRCO$M	401
PRINT	FFA34	PROMPT	FEA0E	PRT$IO	FF9F6
PRTINI	FFA13	PRTRST	FE28D	PUPNMI	FFC81
PUPTS1	FE915	PUPTST	FE03F	PURMPE	9
PUXNME	8	PWRUP	FFFF0	QDEPTH	43E
QFRONT	43A	QREAR	43C	QUEUE	442
RAMINI	FE143	RCONFQ	FFA8C	RETFLG	436
RMDTBQ	400	RMDTSZ	140	RMPERR	41
ROMERX	28	ROMEVT	406	ROMF4	F4000
ROMF6	F6000	ROMF8	F8000	ROMFA	FA000
ROMSIZ	2000	ROMTIM	FFA9D	ROMTST	FE0FD
ROMVER	FE007	SBEEP	FEB80	SECS	538
SETDAT	FEBB8	SETTIM	FEBBC	SOFINT	FF2EA
STAMON	FFFB3	STATLN	434	STATUS	FF75A
SYSCON	403	SYSROM	FE000	TOERR	20
T1ERR	40	T2ERR	80	THSLIN	42E
TIKCTR	405	TIMCAN	FFB18	TIMERR	4
TIMISP	53A	TIMISR	FFAC1	TIMOUT	FF2E5
TIMRST	FFB1D	TIMTST	FE333	TMSPSV	4B0
TMSSSV	4AE	TSTPAT	FFFD3	VBLERR	40
VECINI	FE454	VECTBO	FE3A0	VIDERR	10
VTOLEN	A	WILD$$	FFC96	WLDERR	42
XNMERR	40				

EQUATE.SRC

ACCBUF	0000	ACCFDC	0080	ACK	0006	ACTINT	0054	AH	Reserved	AL	Reserved
BELINT	0048	BH	Reserved	BL	Reserved	BOOCRC	01FE	BOOTID	01FC	BP	Reserved
BUFFDC	00C0	BX	Reserved	BYTE	Reserved	CDSPEN	0040	CH	Reserved	CINTEN	0080
CLKINT	004E	CL	Reserved	CINTIM	1500	CMPINT	0057	CONINT	004F	COMPLT	0055
CRCERR	0008	CR	000D	CRTACB	0010	CRTADR	1810	CRTCQR	0014	CRTCTX	0013
CRTGDB	0017	CRTINT	0049	CRTLPP	0004	CRTMOD	0000	CRTOFF	0020	CRTPTS	0018
CRTRAC	0008	CRTRCP	0003	CRTRPX	000D	CRTRVS	000F	CRTSAT	0016	CRTSCA	0012
CRTSCP	0002	CRTSDN	0007	CRTSDP	0005	CRTSGP	000B	CRTSTA	1811	CRTSUP	0006
CRTVBL	0020	CRTWAC	0009	CRTWCB	0011	CRTWCH	000A	CRTWPX	000C	CRTWTY	000E
CS	Reserved	CSWRAP	C000	CUREND	000B	CURPSH	000E	CURPSL	000F	CURSTR	000A
CX	Reserved	DATERR	0004	DC1	0011	DC2	0012	DC3	0013	DC4	0014
DH	Reserved	DI	Reserved	DISSTH	000C	DISSTL	000D	DITCYL	0007	DITDIR	0000
DITDSK	0008	DITERR	0009	DITINV	0000	DITLEN	000B	DITPRC	000A	DITSEC	0004
DITTRK	0006	DKABRT	000D	DKABTD	0011	DKBADC	000C	DKBADD	000F	DKBOUN	0009
DKCRCE	0010	DKDMAE	0008	DKFAIL	0020	DKFMTE	0002	DKFSET	0008	DKKMOT	000B
DKNDRM	0000	DKNRDY	0080	DKRDIT	000A	DKRNFE	0004	DKRSET	0000	DKSEEK	0040
DKSETC	000C	DKSTAT	0001	DKVERF	0004	DKVFYE	0005	DKVRFY	0006	DKWPRT	0003
DKWRIT	0003	DKXSET	0009	DL	Reserved	DLE	0010	DRSEL0	000E	DRSEL1	000D
DRSEL2	000B	DRSEL3	0007	DRVBYT	019B	DRVMSK	000F	DRVSSD	0001	DRV48T	0002
DR0MSK	0001	DR1MSK	0002	DR2MSK	0004	DR3MSK	0008	DS	Reserved	DSADDR	0180
DSADD0	0184	DSADD2	0188	DSADD4	018C	DSADD6	0190	DSADD8	0194	DSKC$P	0000
DSKINT	004D	DSKS$P	0004	DSSIZR	0182	DSSIZ0	0186	DSSIZ2	018A	DSSIZ4	018E
DSSIZ6	0192	DSSIZ8	0196	DWORD	Reserved	DX	Reserved	EM	0019	ENAB0	00FE
ENAB1	00FD	ENAB2	00FB	ENAB3	00F7	ENAB4	00EF	ENAB5	00DF	ENAB6	00BF
ENAB7	007F	ENQ	0005	EOI	0020	EOT	0004	ES	Reserved	ESC$	001B
ETB	0017	ETX	0003	EVTACK	0005	EVTEXP	0006	EVTSTA	0007	EVTTYP	0004
E$ABTD	0011	E$BADC	0001	E$BCYL	0004	E$BOUN	0009	E$BSEC	0004	E$BTRK	0004
E$CRC	0010	E$NORM	0000	E$NRDY	0080	E$RNF	0004	E$SEEK	0040	E$TIME	0080
E$VRFY	0005	E$WRPT	0003	E12MSK	0010	E34MSK	0020	E56MSK	0040	FAR	Reserved
FATINT	0050	FDC	0020	FDCBUF	0040	FDCCMD	0020	FDCDAT	0023	FDCSEC	0022
FDCSTA	0020	FDCTRK	0021	FF	000C	FMTERR	0020	FORCED	00D0	FS	001C
GAMINT	004C	GAMMSK	8000	GRAMSK	1000	GRASEQ	C000	GRBMSK	2000	GRBSEQ	C800
GRCMSK	4000	GRCSEQ	D000	GRPMSK	7000	GR$BLU	1010	GR$GRN	1020	GR$RED	1030
GS	001D	HIGHAT	000F	HT	0009	HZDCHR	0001	HZSPOS	0002	HZTCHR	0000
ICW1	0010	ICW4	0000	IC$EOI	0002	IC$IW4	0001	IC$LTM	0008	IC$M86	0001
IC$SNQ	0002	IEVCTR	0004	IEVFLQ	0003	IEVLNK	0000	IEVSUB	0006	IEVTID	0002
IMMINT	00DB	INTA00	0018	INTA01	0019	INTCTR	019A	IO8251	0010	IO8253	0014
IO8259	0018	IR0INT	0040	IR1INT	0041	IR2INT	0042	IR3INT	0043	IR4INT	0044
IR5INT	0045	IR6INT	0046	IR7INT	0047	I87MSK	0001	KEYINT	004A	LED1OF	0001
LED2OF	0002	LED3OF	0004	LF	000A	LINLEN	002D	LPENHI	0010	LPENLO	0011
MAPINT	005B	MEMBEG	0000	MEMEND	07FF	MEMSIZ	0198				

EQUATE.SRC

MMASK	FF3F	MODCTL	0008	MONTYP	0004	MOTR$0	0010	MOTR$1	0020
MOTR$2	0040	MOTR$3	0080	MSCINP	1000	MSCOUT	1820	M103P1	1000
M103P2	2000	M103P3	4000	M103P4	8000	M212P1	0100	M212P2	0200
M212P3	0400	M212P4	0800	NAK	0015	NEAR	Reserved	NMIINT	0002
NRYERR	0080	NUL	0000	OFFSET	Reserved	OP$ASP	0001	OP$BON	0009
OP$EP	0002	OP$FI	0000	OP$IR	0003	OP$IS	0004	OP$IW	0005
OP$NUL	00FF	OP$RIB	0008	OP$TER	000A	OP$VFC	0006	OP$VFD	0007
OVFINT	0004	PARIEN	0008	PARINT	0008	PAUINT	005C	PBKINT	005D
PHNBEG	0800	PHNEND	0FFF	PRAUTO	0010	PRBUSY	0010	PRCI$P	0001
PRCO$P	0003	PRDA$P	0002	PRECMP	0010	PREAUL	0080	PRINEN	0080
PRINIT	0040	PRPAPO	0020	PRSLCT	0040	PRSTRB	0020	PRTINT	0048
PSCINT	005E	PSPB00	0001	PSPB01	0002	PSPB02	0004	PSPB03	0008
PSPB04	0010	PSPB05	0020	PSPB06	0040	PSPB07	0080	PTR	Reserved
QUEINT	005F	RDCMD	0080	RD$REG	1813	READIR	000A	READIB	0008
RQ$MAX	0004	RQ$REA	0001	RQ$VFC	0003	RQ$VFD	0004	RQ$WRI	0002
RS	001E	RSTBUF	0022	RSTCMD	000C	RSTINT	0051	SCNLNB	0009
SCRLEN	4000	SC1MSK	0100	SC2MSK	0200	SC3MSK	0400	SC4MSK	0800
SECBUF	0020	SEQ	Reserved	SEKCMD	001C	SEKERR	0002	SHORT	Reserved
SI	Reserved	SIDE$0	0020	SINGLE	0008	SI$	000F	SLCINT	0058
SNFERR	0010	SOH	0001	SO$	000E	SP	Reserved	SPACE	0020
SPKREN	0001	SS	Reserved	STPCMD	005C	STPINT	0001	STPSPD	0000
STX	0002	SUB$	001A	SVCINT	0053	SYN	0016	SYNWID	0003
SYSCEL	01F0	SYSCO1	019C	SYSCO2	019E	S$DEP	00FF	S$DONE	0003
S$IDLE	0000	S$ISEE	0002	S$MAX	0003	S$WAIT	0001	TC$BCD	0001
TC$LAT	0000	TC$LSB	0010	TC$MD0	0000	TC$MD1	0002	TC$MD2	0004
TC$MD3	0006	TC$MD4	0008	TC$MD5	000A	TC$MSB	0020	TC$SCO	0000
TC$SC1	0040	TC$SC2	0080	TC$WRD	0030	TIMCMD	0017	TIMERO	0014
TIMER1	0015	TIMER2	0016	TIMINT	005A	TMR1EN	0002	TMR2EN	0004
UB	001F	VFDERR	0001	VRADJL	0005	VRDROW	0006	VRSPOS	0007
VRTROW	0004	VT	0008	WINMSK	0080	WINRST	0031	WORD	Reserved
WRPERR	0040	WRSTCM	0010	WRTCMD	00A0	WRTREG	1812	XITVEC	0059
ZDVINT	0000								

No errors detected

```
                        SECTION  B

                ************************************
                *       LINK LOAD THE MODULES      *
                ************************************

;
;  SYSORG must be loaded first
;
input   SYSORG                  !
;                               !
;  System equates               !   no memory allocation
;                               !   for these modules
input   ROMERR                  !
input   TSTERR                  !


;      Input                        Output
;  Powerup test           Modules entries reference.
;     Modules
                ;  absolute     relative to        module
                ;  address       CS= F400        description

input   HEADER  ;  FE000        F400:A000       Rom header information
input   PUPTST  ;  FE03F        F400:A03F       Power up test entry point
input   RAMINI  ;  FE143        F400:A143       Setup Rom's data in Ram
input   INTTST  ;  FE190        F400:A190       Interrupt controller tests
input   TIMTST  ;  FE333        F400:A333       Timers accuracy tests
input   VECINI  ;  FE454        F400:A454       Intrpt. vectors initialization
input   DSKTST  ;  FE4A4        F400:A4A4       Floppy disks controller tests
input   DRVTST  ;  FE583        F400:A583       Disk drivers tests
input   CRTTST  ;  FE636        F400:A636       CRT tests
input   MSCTST  ;  FE7DD        F400:A7DD       Miscelaneous initialization
input   KEYTST  ;  FE8A1        F400:A8A1       Key board initialization
input   PUPTST  ;  FE915        F400:A915       Option Rom & Ram tests
input   DKBOOT  ;  FEA47        F400:AA47       Disk booter loader

;
;    Device drivers
;
input   BELDSR  ;  FEAFE        F400:AAFE       Beeper dsr
input   CLKDSR  ;  FEB9B        F400:AB9B       Day time clock dsr
input   CRTDSR  ;  FEC1A        F400:AC1A       CRT dsr
input   DSK_IO  ;  FF0D9        F400:B0D9       Disk io dsr
input   DSKISR  ;  FF325        F400:B325       Disk dsr
input   FLPDIR  ;  FF475        F400:B475       Disk interrupt service
input   KEYDSR  ;  FF719        F400:B719       Key board dsr
input   PRTDSR  ;  FF9F6        F400:B9F6       printer dsr
input   CONFIG  ;  FFA84        F400:BA84       Return system configuration
input   ROMTIM  ;  FFA9D        F400:BA9D       Rom timing services
input   TIMISR  ;  FFAC1        F400:BAC1       System timer intr. serv.
input   TIMSUB  ;  FFB1D        F400:BB1D       Restart timer service
```

SECTION C.

EQUATE FILES

File name

PEQ:ASCII.EQU
PEQ:CRT.EQU
PEQ:CRTOP.EQU
PEQ:DSKDIT.EQU
PEQ:DSKERR.EQU
PEQ:DSKOP.EQU
PEQ:DSKOPS.EQU
PEQ:DSKREQ.EQU
PEQ:DSKSTA.EQU
PEQ:INTCTLR.EQU
PEQ:LATCHES.EQU
PEQ:PORTADDR.EQU
PEQ:FLOPPY.EQU
PEQ:SYSCELL.EQU
PEQ:SYSCON.EQU
PEQ:TIMERS.EQU
PEQ:TIMEVT.EQU
PEQ:VECTOR.EQU
PEQ:WINCHSTR.EQU

```
                          1          INCLUDE PEQ:ASCII.EQU
                          2  ;
                          3  ;---  ASCII CONTROL CHARACTER CODES ---
                          4  ;
=0000                     5  NUL     EQU     00H             ;
=0001                     6  SOH     EQU     01H             ; ^A
=0002                     7  STX     EQU     02H             ; ^B
=0003                     8  ETX     EQU     03H             ; ^C
=0004                     9  EOT     EQU     04H             ; ^D
=0005                    10  ENQ     EQU     05H             ; ^E
=0006                    11  ACK     EQU     06H             ; ^F
=0007                    12  BEL     EQU     07H             ; ^G
=0008                    13  BS      EQU     08H             ; ^H
=0009                    14  HT      EQU     09H             ; ^I
=000A                    15  LF      EQU     0AH             ; ^J
=000B                    16  VT      EQU     0BH             ; ^K
=000C                    17  FF      EQU     0CH             ; ^L
=000D                    18  CR      EQU     0DH             ; ^M
=000E                    19  SO_     EQU     0EH             ; ^N  Trailing '_' is to prevent keyword clash
=000F                    20  SI_     EQU     0FH             ; ^O  Trailing '_' is to prevent keyword clash
=0010                    21  DLE     EQU     10H             ; ^P
=0011                    22  DC1     EQU     11H             ; ^Q
=0012                    23  DC2     EQU     12H             ; ^R
=0013                    24  DC3     EQU     13H             ; ^S
=0014                    25  DC4     EQU     14H             ; ^T
=0015                    26  NAK     EQU     15H             ; ^U
=0016                    27  SYN     EQU     16H             ; ^V
=0017                    28  ETB     EQU     17H             ; ^W
=0018                    29  CAN     EQU     18H             ; ^X
=0019                    30  EM      EQU     19H             ; ^Y
=001A                    31  SUB_    EQU     1AH             ; ^Z Trailing '_' is to prevent keyword clash
=001B                    32  ESC_    EQU     1BH             ; Trailing '_' is to prevent keyword clash
=001C                    33  FS      EQU     1CH             ;
=001D                    34  GS      EQU     1DH             ;
=001E                    35  RS      EQU     1EH             ;
=001F                    36  US      EQU     1FH             ;
=0020                    37  SPACE   EQU     20H             ;
=007F                    38  DEL     EQU     7FH             ;
```

```
                        40          INCLUDE PEQ:CRT.EQU
                        41  ;
                        42  ;  CRT DEVICE EQUATES
                        43  ;
=0000                   44  MEMBEQ  EQU   0000H          ; BEQINNINQ OF CRT MEMORY
=07FF                   45  MEMEND  EQU   07FFH          ; END OF CRT MEMORY
=0800                   46  PHNBEQ  EQU   0800H          ; BEQINNINQ OF PHANTOM MEMORY
=0FFF                   47  PHNEND  EQU   0FFFH          ; END OF PHANTOM MEMORY
=1000                   48  MSCINP  EQU   1000H          ; MISC. INPUT BUFFER
=1010                   49  QR_BLU  EQU   1010H          ; QRAPHICS BLUE PALETTE
=1020                   50  QR_QRN  EQU   1020H          ; QRAPHICS QREEN PALETTE
=1030                   51  QR_RED  EQU   1030H          ; QRAPHICS RED PALETTE
=1800                   52  ATTLAT  EQU   1800H          ; ATTRIBUTE LATCH
=1810                   53  CRTADR  EQU   1810H          ; CRTC ADDRESS REQISTER
=1811                   54  CRTSTA  EQU   1811H          ; CRTC STATUS REQISTER
=1812                   55  WRTREQ  EQU   1812H          ; CRTC WRITE REQISTER
=1813                   56  RD_REQ  EQU   1813H          ; CRTC READ REQISTER
=1820                   57  MSCOUT  EQU   1820H          ; MISC. OUTPUT REQISTER
=C000                   58  QRASEQ  EQU   0C000H         ; QRAPHICS BANK A ADDRESS
=C800                   59  QRBSEQ  EQU   0C800H         ; QRAPHICS BANK B ADDRESS
=D000                   60  QRCSEQ  EQU   0D000H         ; QRAPHICS BANK C ADDRESS
=4000                   61  SCRLEN  EQU   04000H         ; LENQTH OF QRAPHICS MEMORY IN WORDS
=002D                   62  LINLEN  EQU     45           ; QRAPHICS LINE LENQTH IN WORDS
=005C                   63  BLNLEN  EQU     92           ; QRAPHICS LINE LENQTH IN BYTES
                        64  ;
                        65  ; CRTC REQISTER EQUATES
                        66  ;
=0000                   67  HZTCHR   EQU    00H          ; HORIZONTAL TOTAL CHARS.
=0001                   68  HZDCHR   EQU    01H          ; HORIZONTAL DISPLAYED CHARS.
=0002                   69  HZSPOS   EQU    02H          ; HORIZONTAL SYNC POSITION
=0003                   70  SYNWID   EQU    03H          ; VSYNC WIDTH, HSYNC WIDTH
=0004                   71  VRTROW   EQU    04H          ; VERTICAL TOTAL ROWS
=0005                   72  VRADJL   EQU    05H          ; VERTICAL ADJUST LINES
=0006                   73  VRDROW   EQU    06H          ; VERTICAL DISPLAYED ROWS
=0007                   74  VRSPOS   EQU    07H          ; VERTICAL SYNC POSITION
=0008                   75  MODCTL   EQU    08H          ; MODE CONTROL
=0009                   76  SCNLNS   EQU    09H          ; SCAN LINES PER ROW
=000A                   77  CURSTR   EQU    0AH          ; CURSOR START SCAN LINE
=000B                   78  CUREND   EQU    0BH          ; CURSOR END SCAN LINE
=000C                   79  DISSTH   EQU    0CH          ; DISPLAY START ADDRESS HIQH
=000D                   80  DISSTL   EQU    0DH          ; DISPLAY START ADDRESS LOW
=000E                   81  CURPSH   EQU    0EH          ; CURSOR POSITION ADDRESS HIQH
=000F                   82  CURPSL   EQU    0FH          ; CURSOR POSITION ADDRESS LOW
=0010                   83  LPENHI   EQU    10H          ; LIQHT PEN ADDRESS HIQH
=0011                   84  LPENLO   EQU    11H          ; LIQHT PEN ADDRESS LO
                        85  ;
                        86  ; CRT BIT EQUATES
                        87  ;
=0040                   88  CDSPEN   EQU    01000000B    ; CRT DISPLAY ENABLE (ALSO DISABLES INTERRUPT)
=0080                   89  CINTEN   EQU    10000000B    ; CRT INTERRUPT ENABLE
=1500                   90  CINTIM   EQU    1500H        ; CRT INTERRUPT WAIT LOOP COUNT
=0020                   91  CRTVBL   EQU    00100000B    ; CRT IN VERTICAL BLANK PERIOD
```

EQUATE.SRC

```
=0020                   92  CRTOFF   EQU   20H             ; CRT DISABLE
=000F                   93  HIGHAT   EQU   0FH             ; HIGH BRILLIANCE ATTRIBUTES
                        94  ;
                        95  ; CHARACTER EQUATES
                        96  ;
=00DB                   97  BLOCK    EQU   0DBH            ; BLOCK CHARACTER
```

```
                        99          INCLUDE PEG:CRTOP.EQU
                       100  ;
                       101  ; --- CRT DSR 'opcodes' (INT 49H)
                       102  ;
=0000                  103  CRTMOD  EQU     00H             ; Set CRT mode function
=0001                  104  CRTSCT  EQU     01H             ; Set cursor type function
=0002                  105  CRTSCP  EQU     02H             ; Set cursor position function
=0003                  106  CRTRCP  EQU     03H             ; Read cursor position function
=0004                  107  CRTLPP  EQU     04H             ; Read light pen position function
=0005                  108  CRTSDP  EQU     05H             ; Set active display page
=0006                  109  CRTSUP  EQU     06H             ; Scroll text up function
=0007                  110  CRTSDN  EQU     07H             ; Scroll text down function
=0008                  111  CRTRAC  EQU     08H             ; Read attribute and character function
=0009                  112  CRTWAC  EQU     09H             ; Write attribute and character function
=000A                  113  CRTWCH  EQU     0AH             ; Write character only function
=000B                  114  CRTSGP  EQU     0BH             ; Set graphics palette function
=000C                  115  CRTWPX  EQU     0CH             ; Write graphics pixel function
=000D                  116  CRTRPX  EQU     0DH             ; Read graphics pixel function
=000E                  117  CRTWTY  EQU     0EH             ; Write TTY screen output function
=000F                  118  CRTRVS  EQU     0FH             ; Read video state function
=0010                  119  CRTACB  EQU     10H             ; Write attribute and character block function
=0011                  120  CRTWCB  EQU     11H             ; Write character block function
=0012                  121  CRTSCA  EQU     12H             ; Change screen attributes function
=0013                  122  CRTCTX  EQU     13H             ; Clear text screen function
=0014                  123  CRTCGR  EQU     14H             ; Clear graphics screen function
=0015                  124  CRTSTY  EQU     15H             ; Set TTY status line function
=0016                  125  CRTSAT  EQU     16H             ; Set attribute latch function
=0017                  126  CRTGDB  EQU     17H             ; Get physical display begin pointer function
=0018                  127  CRTPTS  EQU     18H             ; Print TTY string function
```

```
                        129             INCLUDE PEQ:DSKDIT.EQU
                        130
                                                        ; Floppy disk DIT equates
=0000                   131     DITDIR  EQU     0       ; Disk Interface Routine vector (dword)
=0004                   132     DITSEC  EQU     4       ; Sector size in bytes (word)
=0006                   133     DITTRK  EQU     6       ; Track size in sectors (byte)
=0007                   134     DITCYL  EQU     7       ; Cylinder size in tracks (byte)
=0008                   135     DITDSK  EQU     8       ; Disk size in cylinders (word)
=0009                   136     DITERR  EQU     9       ; Maximum number of error retries
=000A                   137     DITPRC  EQU     10      ; Write pre-comp threshold cylinder
=000B                   138     DITLEN  EQU     11      ; Length of DIT
                        139                             ;
=0000                   140     DITINV  EQU     0000h   ; Used to signal that a DIT is Invalid
```

```
                        142             INCLUDE PEG:DSKERR.EQU
                        143
                                                        ; DISK DSR ERROR CODES(IBM COMPATIBLE)
=0000                   144     DKNORM  EQU     00000000B       ; No error
=0080                   145     DKNRDY  EQU     10000000B       ; Controller timeout - disk not ready
=0040                   146     DKSEEK  EQU     01000000B       ; Seek failed - track not found
=0020                   147     DKFAIL  EQU     00100000B       ; Controller hardware failure
=0010                   148     DKCRCE  EQU     00010000B       ; CRC error on read
=0008                   149     DKDMAE  EQU     00001000B       ; Data request error - controller failure
=0004                   150     DKRNFE  EQU     00000100B       ; Sector not found error
=0002                   151     DKFMTE  EQU     00000010B       ; Timeout - no data error - bad disk format
=0001                   152     DKCMDE  EQU     00000001B       ; Command error - bad command passed
=0003                   153     DKWPRT  EQU     00000011B       ; Write protect error
=0005                   154     DKVFYE  EQU     00000101B       ; Data verification error
=0009                   155     DKBOUN  EQU     00001001B       ; I/O transfer crosses 64k boundary
=0011                   156     DKABTD  EQU     00010001B       ; I/O operation aborted by request
                        157                                     ;
                        158                                     ; E_xxxx =) Operation status' (error codes)
                        159                                     ; These values are used in the disk driver and
                        160                                     ; are translated via equate to the above set of
                        161                                     ; error codes for compatibility with the values
                        162                                     ; used elsewhere in the system.
=0000                   163     E_NORM  EQU     DKNORM          ; No error
=0011                   164     E_ABTD  EQU     DKABTD          ; Disk I/O aborted by request
=0001                   165     E_BADC  EQU     DKCMDE          ; Bad command
=0004                   166     E_BCYL  EQU     DKRNFE          ; Non-existent cylinder requested
=0004                   167     E_BTRK  EQU     DKRNFE          ; Non-existent track requested
=0009                   168     E_BOUN  EQU     DKBOUN          ; I/O transfer crosses 64K boundry
=0004                   169     E_BSEC  EQU     DKRNFE          ; Non-existent sector requested
=0010                   170     E_CRC   EQU     DKCRCE          ; Bad CRC found
=0080                   171     E_NRDY  EQU     DKNRDY          ; Device is not ready
=0004                   172     E_RNF   EQU     DKRNFE          ; Could not find requested sector
=0040                   173     E_SEEK  EQU     DKSEEK          ; Could not find requested cylinder
=0080                   174     E_TIME  EQU     DKNRDY          ; Operation did not complete
=0005                   175     E_VRFY  EQU     DKVFYE          ; Write data did not verify
=0003                   176     E_WRPT  EQU     DKWPRT          ; Media is write protected
```

```
                        178          INCLUDE PEQ:DSKOP.EQU
                        179
                        180                                  ; OP_xxx =) Opcodes for generic disk operations
                        181                                  ; These values are displacements in the Device
                        182                                  ; Interface Table to the branch vector for each
                                                             ; corresponding routine.
=00FF                   183  OP_NUL  EQU     255            ; Null operation
=0000                   184  OP_FI   EQU     0              ; interrupt
=0001                   185  OP_ASP  EQU     1              ; Auxiliary State Processor
=0002                   186  OP_EP   EQU     2              ; Error Recovery Processor
=0003                   187  OP_IR   EQU     3              ; Initiate Read
=0004                   188  OP_IB   EQU     4              ; Initiate Seek
=0005                   189  OP_IW   EQU     5              ; Write
=0006                   190  OP_VFC  EQU     6              ; Verify CRC
=0007                   191  OP_VFD  EQU     7              ; Verify data
=0008                   192  OP_RIB  EQU     8              ; Read and Interpret Status
=0009                   193  OP_BGN  EQU     9              ; Begin request
=000A                   194  OP_TER  EQU     10             ; Process request termination
```

```
                        196             INCLUDE PEQ:DSKOPS.EQU
                        197
                        198 DKRSET  EQU  0          ; DISK DSR OPERATION CODES
=0000                   199 DKSTAT  EQU  1          ; Reset disk system, drive parms must be preset
=0001                   200 DKREAD  EQU  2          ; Get disk status in (al)
=0002                   201 DKWRIT  EQU  3          ; Read sectors into memory
=0003                   202 DKVERF  EQU  4          ; Write memory to disk sectors
=0004                   203 ;DKFORM EQU  5          ; Verify crc on disk sectors
                        204 DKVRFY  EQU  6          ; Format track (not implemented)
=0006                   205 DKSSTA  EQU  7          ; Verify memory against disk sectors
=0007                   206 DKFSET  EQU  8          ; Get disk status for preretry (if any)
=0008                   207 DKXSET  EQU  9          ; Set UNIT & standard DIT for a drive
=0009                   208 DKRDIT  EQU  10         ; Set UNIT & DIT address for a drive
=000A                   209 DKKMOT  EQU  11         ; Return DIT address for drive
=000B                   210 DKBADC  EQU  12         ; Turn off Floppy Disk Motors
=000C                   211 DKSETC  EQU  12         ; Old >= this is a bad command
=000C                   212 DKABRT  EQU  13         ; Set concurrent mode
=000D                   213 DKFSTA  EQU  14         ; Abort pending operation
=000E                   214 DKBADD  EQU  15         ; Return last full status
=000F                               ; New >= this is a bad command
```

```
                        216          INCLUDE PEQ:DSKREQ.EQU
                        217
                        218  ;RQ_SEEK EQU    0              ; Seek *************** OBSELETE ***************
=0001                   219  RQ_READ EQU    1              ; Read
=0002                   220  RQ_WRIT EQU    2              ; Write
=0003                   221  RQ_VFC  EQU    3              ; Verify CRC
=0004                   222  RQ_VFD  EQU    4              ; Verify data
=0004                   223  RQ_MAX  EQU    4              ; Maximum request number
```

```
                        225             INCLUDE PEQ:DSKSTA.EQU
                        226     ;
                        227     ; S_xxxx =) Driver states
=00FF                   228     S_DEP   EQU     255             ; Device-dependent code must set state
=0000                   229     S_IDLE  EQU     0               ; Waiting for a request
=0001                   230     S_WAIT  EQU     1               ; Waiting to retry a request
=0002                   231     S_ISEEK EQU     2               ; Implicit seek pending
=0003                   232     S_DONE  EQU     3               ; Operation completed
=0003                   233     S_MAX   EQU     3               ; The highest "standard" state number
                        234     ;                               ; Other values are device-dependent
```

```
                        235             INCLUDE PEQ:INTCTLR.EQU
                        236     ;
                        237     ;       INTERRUPT CONTROLLER EQUATES
                        238     ;                                       ; Interrupt controller ports
=0018                   239     INTA00  EQU     18H                     ; ICW1, OCW2, OCW3              ; Read IRR, ISR
=0019                   240     INTA01  EQU     19H                     ; OCW1, ICW2, ICW3, ICW4        ; Read IMR
                        241     ;
=0010                   242     ICW1    EQU     10H                     ; ICW1 base   (written to INTA00)
=0001                   243     IC_IW4  EQU     01H                     ; ICW4 needed
=0002                   244     IC_SNG  EQU     02H                     ; Single 8259 mode (not cascaded)
=0008                   245     IC_LTM  EQU     08H                     ; Level-triggered mode
                        246     ;
=0000                   247     ICW4    EQU     00H                     ; ICW4 base   (written to INTA01)
=0001                   248     IC_M86  EQU     01H                     ; Microprocessor mode = 8086/88
=0002                   249     IC_EOI  EQU     02H                     ; Auto End-Of-Interrupt
                        250     ;
=00FE                   251     ENAB0   EQU     11111110B               ; Interrupt Masks   (written to INTA01)
=00FD                   252     ENAB1   EQU     11111101B
=00FB                   253     ENAB2   EQU     11111011B
=00F7                   254     ENAB3   EQU     11110111B
=00EF                   255     ENAB4   EQU     11101111B
=00DF                   256     ENAB5   EQU     11011111B
=00BF                   257     ENAB6   EQU     10111111B
=007F                   258     ENAB7   EQU     01111111B
                        259     ;                                       ; Commands  (written to INTA00)
=0020                   260     EOI     EQU     20H                     ; Non-specific End-Of-Interrupt command
=000A                   261     READIR  EQU     0AH                     ; Read Interrupt Request Register next
=000B                   262     READIS  EQU     0BH                     ; Read In-Service Register next
```

```
                        264           INCLUDE PEQ:LATCHES.EQU
                        265   ;
                        266   ;   Floppy control and timer interrupt latch - 'DSKC_P'
                        267   ;
=0001                   268   SPKREN  EQU     01H             ; Speaker enable (timer0 gate input)
=0002                   269   TMR1EN  EQU     02H             ; Timer 1 interrupt enable
=0004                   270   TMR2EN  EQU     04H             ; Timer 2 interrupt enable
=0008                   271   SINGLE  EQU     08H             ; Single density (FM)  (vs. Double - MFM)
=0010                   272   PRECMP  EQU     10H             ; Write precomp enable (inner tracks only)
=0020                   273   SIDE_0  EQU     20H             ; Head 0 select
                        274   ;                                 Sector buffer control bits M1, M0 :
=0000                   275   ACCBUF  EQU     00H             ; Access RAM or reset RAM counter
=0080                   276   ACCFDC  EQU     80H             ; Access Floppy controller chip
=0040                   277   FDCBUF  EQU     40H             ; Select FDC --) BUFFER mode (disk read)
=00C0                   278   BUFFDC  EQU     0C0H            ; Select BUFFER --) FDC mode (disk write)
=FF3F                   279   MMASK   EQU     NOT 0C0H        ; Mask for setting sector buffer mode
                        280   ;
                        281   ;   Floppy select and motor control latch - 'DSKS_P'
                        282   ;
=000E                   283   DRSEL0  EQU     0EH             ; Floppy drive selects (Active LOW)
=000D                   284   DRSEL1  EQU     0DH
=000B                   285   DRSEL2  EQU     0BH
=0007                   286   DRSEL3  EQU     07H
=0010                   287   MOTR_0  EQU     10H             ; Motor On for drive 0  (Note: MOTR_0 & MOTR_1
=0020                   288   MOTR_1  EQU     20H             ; Motor On for drive 1   are OR'd in hardware)
=0040                   289   MOTR_2  EQU     40H             ; Motor On for drive 2
=0080                   290   MOTR_3  EQU     80H             ; Motor On for drive 3
                        291   ;
                        292   ;   Parallel port control input port - 'PRCI_P'
                        293   ;
=0001                   294   DRV8SD  EQU     01H             ; Drives are single-sided (vs. double-sided)
=0002                   295   DRV48T  EQU     02H             ; Drives are 48tpi (vs. 96tpi)
=0004                   296   MONTYP  EQU     04H             ; Monitor type ((60HZ system (vs. 50HZ)))
=0008                   297   PARINT  EQU     08H             ; Parity interrupt pending
=0010                   298   PRBUSY  EQU     10H             ; Parallel port BUSY signal
=0020                   299   PRPAPO  EQU     20H             ; Parallel port PAPER OUT signal
=0040                   300   PRSLCT  EQU     40H             ; Parallel port SELECTED signal
=0080                   301   PRFAUL  EQU     80H             ; Parallel port FAULT- signal
                        302   ;
                        303   ;   Parallel port control output latch - 'PRCO_P'
                        304   ;
=0001                   305   LED1OF  EQU     01H             ; LED 1 off \
=0002                   306   LED2OF  EQU     02H             ; LED 2 off  ) Active LOW
=0004                   307   LED3OF  EQU     04H             ; LED 3 off /
=0008                   308   PARIEN  EQU     08H             ; Parity interrupt enable
=0010                   309   PRAUTO  EQU     10H             ; Parallel port AUTO FEED- signal
=0020                   310   PRSTRB  EQU     20H             ; Parallel port STROBE- signal
=0040                   311   PRINIT  EQU     40H             ; Parallel port INIT- signal
=0080                   312   PRINEN  EQU     80H             ; Parallel port interrupt enable
```

```
                        314             INCLUDE PEQ:PORTADDR.EQU
                        315     ; --- I/O port addresses ---
=0010                   316     IO8251  EQU     10H             ; Base addr of USART                    (U44)
=0014                   317     IO8253  EQU     14H             ; Base addr of Timer Chip               (U45)
=0018                   318     IO8259  EQU     18H             ; Base addr of Interrupt Controller     (U46)
=0020                   319     FDC     EQU     20H             ; Base addr of Floppy Controller        (U13)
                        320     ;
=0000                   321     D8KC_P  EQU     00H             ; Floppy control and timer intr. latch  (U47)
=0001                   322     PRCI_P  EQU     01H             ; Parallel port control input port      (U48)
=0002                   323     PRDA_P  EQU     02H             ; Parallel port data output latch       (U49)
=0003                   324     PRCO_P  EQU     03H             ; Parallel port control output latch    (U50)
=0004                   325     D8KB_P  EQU     04H             ; Floppy select and motor control latch (U51)
```

```
                        327           INCLUDE PEQ:FLOPPY.EQU
                        328   ;
                        329   ;   DISK CONTROLLER EQUATES
                        330   ;
=0020                   331   FDCCMD  EQU     FDC+0           ; Command port
=0020                   332   FDCSTA  EQU     FDC+0           ; Status register
=0021                   333   FDCTRK  EQU     FDC+1           ; Track register
=0022                   334   FDCSEC  EQU     FDC+2           ; Sector register
=0023                   335   FDCDAT  EQU     FDC+3           ; Data register
=0020                   336   SECBUF  EQU     FDC+0           ; Sector buffer read/write port
=0022                   337   RSTBUF  EQU     FDC+2           ; Reset sector buffer port
                        338   ;
=0000                   339   STPSPD  EQU     00H             ; Track-to-track step speed:  00 --)  6 msec
                        340   ;                                                             01 --) 12 msec
                        341   ;                                                             02 --) 20 msec
                        342   ;   FDC Commands:                                            03 --) 30 msec
                        343   ;
=0080                   344   RDCMD   EQU     80H             ; Read sector
=00A0                   345   WRTCMD  EQU     0A0H            ; Write sector
=000C                   346   RSTCMD  EQU     0CH + STPSPD    ; Restore (home)
=001C                   347   SEKCMD  EQU     1CH + STPSPD    ; Seek w/verify
=005C                   348   STPCMD  EQU     5CH + STPSPD    ; Step in w/verify & update
=00D0                   349   FORCED  EQU     0D0H            ; Forced interrupt (gets type I status)
=00D8                   350   IMMINT  EQU     0D8H            ; Immediate interrupt
                        351   ;
                        352   ;   FDC and Disk DSR internal error codes:
                        353   ;
=0080                   354   NRYERR  EQU     80H             ; Not ready error
=0040                   355   WRPERR  EQU     40H             ; Write protect error
=0020                   356   FMTERR  EQU     20H             ; Disk format error (timeout w/disk in drive)
=0010                   357   SNFERR  EQU     10H             ; Sector not found error
=0008                   358   CRCERR  EQU     08H             ; CRC error
=0004                   359   DATERR  EQU     04H             ; Lost data error
=0002                   360   SEKERR  EQU     02H             ; Seek error (contrived - not an FDC error)
=0001                   361   VFDERR  EQU     01H             ; Data verification error
```

```
                        363     '       INCLUDE PEQ:SYSCELL.EQU
                        364     ; --- SYSTEM CELL OFFSETS ---
                        365     ;
                        366     ;   The 'System Cell' is an area of the boot sector reserved for system
                        367     ;   configuration and identification information.  These locations are
                        368     ;   defined as offsets from BOOTLO.
                        369     ;
=01F0                   370     SYSCEL  EQU     1F0H            ; Beginning of system cell area
=01FC                   371     BOOTID  EQU     1FCH            ; Offset of boot id word
=01FE                   372     BOOCRC  EQU     1FEH            ; Offset of boot sector CRC
```

```
                        374             INCLUDE PEQ:SYSCON.EQU
                        375     ; --- Primary System Configuration word (SYSCON, accessed with INT 4FH)
                        376     ;       NOTE: Several routines take advantage of the knowledge of which byte
                        377     ;         the bits are in and that the DRx, SCx, and QRx bits are adjacent.
=000F                   378     DRVMSK  EQU     0000000000001111B       ; Disk drive status bit mask
=0001                   379     DR0MSK  EQU     0000000000000001B       ; Floppy drive 0 installed (internal)
=0002                   380     DR1MSK  EQU     0000000000000010B       ; Floppy drive 1 installed (internal)
=0004                   381     DR2MSK  EQU     0000000000000100B       ; Floppy drive 2 installed (unused)
=0008                   382     DR3MSK  EQU     0000000000001000B       ; Floppy drive 3 installed (unused)
=0010                   383     E12MSK  EQU     0000000000010000B       ; 'E1-E2' jumper
=0020                   384     E34MSK  EQU     0000000000100000B       ; 'E3-E4' jumper
=0040                   385     E56MSK  EQU     0000000001000000B       ; 'E5-E6' jumper
=0080                   386     WINMSK  EQU     0000000010000000B       ; Winchester controller installed
                        387     ; (high byte):
=0100                   388     SC1MSK  EQU     0000000100000000B       ; Serial Channel 1 installed
=0200                   389     SC2MSK  EQU     0000001000000000B       ; Serial Channel 2 installed
=0400                   390     SC3MSK  EQU     0000010000000000B       ; Serial Channel 3 installed
=0800                   391     SC4MSK  EQU     0000100000000000B       ; Serial Channel 4 installed
=7000                   392     GRPMSK  EQU     0111000000000000B       ; 3-bits: graphics planes installed
=1000                   393     GRAMSK  EQU     0001000000000000B       ; Graphics RAM bank A installed
=2000                   394     GRBMSK  EQU     0010000000000000B       ; Graphics RAM bank B installed
=4000                   395     GRCMSK  EQU     0100000000000000B       ; Graphics RAM bank C installed
=8000                   396     GAMMSK  EQU     1000000000000000B       ; Game/clock board installed
                        397     ; (high byte of EXTWD1): comm boards
=1000                   398     M103P1  EQU     0001000000000000B       ; 300 bps Modem port 1 installed
=2000                   399     M103P2  EQU     0010000000000000B       ; 300 bps Modem port 2 installed
=4000                   400     M103P3  EQU     0100000000000000B       ; 300 bps Modem port 3 installed
=8000                   401     M103P4  EQU     1000000000000000B       ; 300 bps Modem port 4 installed
=0100                   402     M212P1  EQU     0000000100000000B       ; 1200bps Modem port 1 installed
=0200                   403     M212P2  EQU     0000001000000000B       ; 1200bps Modem port 2 installed
=0400                   404     M212P3  EQU     0000010000000000B       ; 1200bps Modem port 3 installed
=0800                   405     M212P4  EQU     0000100000000000B       ; 1200bps Modem port 4 installed
                        406     ; (low byte of EXTWD1): speech boards
=0001                   407     PSPB00  EQU     0000000000000001B       ; Speech board 0 installed
=0002                   408     PSPB01  EQU     0000000000000010B       ; Speech board 1 installed
=0004                   409     PSPB02  EQU     0000000000000100B       ; Speech board 2 installed
=0008                   410     PSPB03  EQU     0000000000001000B       ; Speech board 3 installed
=0010                   411     PSPB04  EQU     0000000000010000B       ; Speech board 4 installed
=0020                   412     PSPB05  EQU     0000000000100000B       ; Speech board 5 installed
=0040                   413     PSPB06  EQU     0000000001000000B       ; Speech board 6 installed
=0080                   414     PSPB07  EQU     0000000010000000B       ; Speech board 7 installed
                        415     ;
                        416     ;   Secondary System Configuration words (accessed with INT 48H)
                        417     ;
                        418     ;   SYSCO1 (in Interrupt Vector area):
=0001                   419     I87MSK  EQU     0000000000000001B       ; 8087 Numeric Coprocessor installed
                        420     ;
                        421     ;   SYSCO2 (in Interrupt Vector area):
                        422     ;   DRVBYT (in Interrupt Vector area):
                        423     ;
                        424
```

```
                        426             INCLUDE PEQ:TIMER8.EQU
                        427     ;
                        428     ;       TIMER CHIP EQUATE8
                        429     ;
=0014                   430     TIMER0  EQU     I08253          ; Timer 0 read/load addr  (8peaker beep)
=0015                   431     TIMER1  EQU     I08253+1        ; Timer 1 read/load addr  (8ystem timer)
=0016                   432     TIMER2  EQU     I08253+2        ; Timer 2 read/load addr
=0017                   433     TIMCMD  EQU     I08253+3        ; Command Register
                        434     ;
                        435     ;       Control word format
                        436     ;
=0000                   437     TC_8C0  EQU     00H             ; Select counter 0
=0040                   438     TC_8C1  EQU     40H             ; Select counter 1
=0080                   439     TC_8C2  EQU     80H             ; Select counter 2
                        440     ;
=0000                   441     TC_LAT  EQU     00H             ; Counter latching operation
=0020                   442     TC_M8B  EQU     20H             ; Read/load most significant byte only
=0010                   443     TC_L8B  EQU     10H             ; Read/load least significant byte only
=0030                   444     TC_WRD  EQU     30H             ; Read/load L8B first, then M8B
                        445     ;
=0000                   446     TC_MD0  EQU     00H             ; Select mode 0
=0002                   447     TC_MD1  EQU     02H             ; Select mode 1
=0004                   448     TC_MD2  EQU     04H             ; Select mode 2
=0006                   449     TC_MD3  EQU     06H             ; Select mode 3
=0008                   450     TC_MD4  EQU     08H             ; Select mode 4
=000A                   451     TC_MD5  EQU     0AH             ; Select mode 5
                        452     ;
=0001                   453     TC_BCD  EQU     01H             ; BCD counter mode
```

```
                        455             INCLUDE PEQ:TIMEVT.EQU
                        456     ;
                        457     ; TIMING EVENT OFFSETS
                        458     ;
=0000                   459     IEVLNK  EQU     00H     ;       Link pointer
=0002                   460     IEVTID  EQU     02H     ;       Task id
=0003                   461     IEVFLG  EQU     03H     ;       Flags
=0004                   462     IEVCTR  EQU     04H     ;       Counter
=0006                   463     IEVSUB  EQU     06H     ;       Subroutine address
                        464     ;
                        465     ; Event flags
                        466     ;
=0004                   467     EVTTYP  EQU     4       ;       Event type          0/1 = Interrupt/Task
=0005                   468     EVTACK  EQU     5       ;       Event acknowledged  0/1 = Yes/No
=0006                   469     EVTEXP  EQU     6       ;       Event expired       0/1 = No/Yes
=0007                   470     EVTSTA  EQU     7       ;       Event status        0/1 = Inactive/Active
                        471
```

```
                        473           INCLUDE PEQ:VECTOR.EQU
                        474     ;  ***************************************************************
                        475     ;  **** ANY CHANGES MADE TO THIS EQUATE FILE MUST ALSO BE MADE TO ****
                        476     ;  **** THE VECTOR INITIALIZATION TABLES IN THE MODULE 'VECINI'.  ****
                        477     ;  **** ALSO NOTE THAT CSWRAP CHANGES AFFECT THE MODULE 'BYBORG'. ****
                        478     ;  ***************************************************************
                        479     ; ---- Pegasus interrupt vector types ----
                        480     ;
                        481     ;   8088-specific (hardware) interrupts:
                        482     ;
=0000                   483     ZDVINT  EQU     00H             ; Divide-by-zero trap
=0001                   484     STPINT  EQU     01H             ; Single-step trap
=0002                   485     NMIINT  EQU     02H             ; Non-maskable interrupt
=0003                   486     BRKINT  EQU     03H             ; Break (single-byte) software interrupt
=0004                   487     OVFINT  EQU     04H             ; Overflow trap
                        488     ;
                        489     ;       (types 05H-1FH reserved by INTEL)
                        490     ;       (types 20H-3FH reserved for MSDOS)
                        491     ;
                        492     ;   Pegasus hardware interrupts:
                        493     ;
=0040                   494     IR0INT  EQU     40H             ; 8259 interrupt 0 (unused)
=0041                   495     IR1INT  EQU     41H             ; 8259 interrupt 1 (unused)
=0042                   496     IR2INT  EQU     42H             ; 8259 interrupt 2 (unused)
=0043                   497     IR3INT  EQU     43H             ; 8259 interrupt 3 (Timer 1)
=0044                   498     IR4INT  EQU     44H             ; 8259 interrupt 4 (unused)
=0045                   499     IR5INT  EQU     45H             ; 8259 interrupt 5 (Parallel port ACK)
=0046                   500     IR6INT  EQU     46H             ; 8259 interrupt 6 (Disk controller)
=0047                   501     IR7INT  EQU     47H             ; 8259 interrupt 7 (Keyboard USART)
                        502     ;
                        503     ;   ROM BIOS interface vectors:
                        504     ;
=0048                   505     BELINT  EQU     48H             ; System beeper I/O and general ROM interface
=0049                   506     CRTINT  EQU     49H             ; Screen I/O
=004A                   507     KEYINT  EQU     4AH             ; Keyboard I/O
=004B                   508     PRTINT  EQU     4BH             ; Parallel port I/O
=004C                   509     QAMINT  EQU     4CH             ; Analog Input/Clock I/O
=004D                   510     DSKINT  EQU     4DH             ; Floppy disk I/O
=004E                   511     CLKINT  EQU     4EH             ; Time-of-day clock I/O
=004F                   512     CONINT  EQU     4FH             ; System configuration
                        513     ;
=0050                   514     FATINT  EQU     50H             ; Fatal error trap
=0051                   515     RSTINT  EQU     51H             ; Restart timing event (Timing services)
=0052                   516     CANINT  EQU     52H             ; Cancel timing event (Timing services)
=0053                   517     SVCINT  EQU     53H             ; SVC Interface subroutine
=0054                   518     ACTINT  EQU     54H             ; Activate Task subroutine
=0055                   519     COMPLT  EQU     55H             ; Addr for disk I/O completion (concurrent env)
                        520     ;
                        521     ;   System interface vectors (soft interrupts):
                        522     ;
=0057                   523     CMPINT  EQU     57H             ; CRT character mapping vector
=0058                   524     SLCINT  EQU     58H             ; Time slicing, if needed (every 25 msec)
```

```
=0059                525   XITVEC  EQU     59H             ; Address of common interrupt exit routine
=005A                526   TIMINT  EQU     5AH             ; Dynamic timing services, if needed (100 msec)
=005B                527   MAPINT  EQU     5BH             ; Keyboard mapping vector
=005C                528   PAUINT  EQU     5CH             ; Keyboard program pause key vector
=005D                529   PBKINT  EQU     5DH             ; Keyboard program break key vector
=005E                530   PSCINT  EQU     5EH             ; Keyboard print screen vector
=005F                531   QUEINT  EQU     5FH             ; Keyboard queueing vector
                     532   ;
                     533   ;       (type OE0H reserved for CP/M)
                     534   ;-------------------------------------------------------------------------------
                     535   ;   FIXED ROM DATA AREA - (absolute offsets from absolute 0)
                     536   ;       These equates define the ROM communications area, containing data
                     537   ;   that must be accessed by both the ROM and user/application programs.
                     538   ;   This data is accessed from the 'user' program by setting DS = 0.  The
                     539   ;   ROM uses a 'trick' to get to this data by taking advantage of the fact
                     540   ;   that the hardware 'wraps' memory addresses (i.e. the address following
                     541   ;   OFFFFFH is 0).  By accessing relative to CS, and adding the magic number
                     542   ;   CSWRAP (defined below), the desired data (DSADDR, MEMSIZ, etc.) may
                     543   ;   be accessed without having to set up a separate DS first.
                     544   ;
=0180                545   DSADDR  EQU     4*60H           ; (WORD) pointer to DS for System ROM (ROMDAT)
=0182                546   DSSIZR  EQU     4*60H+2         ; (WORD) size of DS for System ROM (ROMDAT)
=0184                547   DSADD0  EQU     4*61H           ; (WORD) pointer to DS for ROM at ROMCOD:0000
=0186                548   DSSIZ0  EQU     4*61H+2         ; (WORD) size of DS for ROM at ROMCOD:0000
=0188                549   DSADD2  EQU     4*62H           ; (WORD) pointer to DS for ROM at ROMCOD:2000
=018A                550   DSSIZ2  EQU     4*62H+2         ; (WORD) size of DS for ROM at ROMCOD:2000
=018C                551   DSADD4  EQU     4*63H           ; (WORD) pointer to DS for ROM at ROMCOD:4000
=018E                552   DSSIZ4  EQU     4*63H+2         ; (WORD) size of DS for ROM at ROMCOD:4000
=0190                553   DSADD6  EQU     4*64H           ; (WORD) pointer to DS for ROM at ROMCOD:6000
=0192                554   DSSIZ6  EQU     4*64H+2         ; (WORD) size of DS for ROM at ROMCOD:6000
=0194                555   DSADD8  EQU     4*65H           ; (WORD) pointer to DS for ROM at ROMCOD:8000
=0196                556   DSSIZ8  EQU     4*65H+2         ; (WORD) size of DS for ROM at ROMCOD:8000
=0198                557   MEMSIZ  EQU     4*66H           ; (WORD) memory size (number of 16-byte blocks)
=019A                558   INTCTR  EQU     4*66H+2         ; (BYTE) outstanding-interrupt counter (for OS)
=019B                559   DRVBYT  EQU     4*66H+3         ; (BYTE) Drive configuration byte
=019C                560   SYSCO1  EQU     4*67H           ; (WORD) Additional system configuration info
=019E                561   SYSCO2  EQU     4*67H+2         ; (WORD) Additional system configuration info
                     562   ;
=C000                563   CSWRAP  EQU     0C000H          ; Magic number which causes CS to wrap to 00000
```

```
                        565             INCLUDE PEQ:WINCHSTR.EQU
                        566     ;       WINCHESTER PORT ADDRESSES AND EQUATES
=0031                   567     WINRST  EQU     31H             ;PORT FOR WINCH RESET
=0010                   568     WRSTCM  EQU     00010000B       ;RESET WINCHESTER
                        569             END
```

```
                        1
                        2   ;*****************************************************************************
                        3   ;  TITLE    - BELDSR - System beeper device service routine                  *
                        4   ;  ABSTRACT - This module contains the routines that handle the speaker I/O.  *
                        5   ;       It includes a user interface as well as routines used internally to   *
                        6   ;       the ROM.                                                              *
                        7   ;  COMPUTER - 8088 assembly language    (BSO)                               *
                        8   ; * The DSR entry was converted to be table driven from the opcodes in order  *
                        9   ;   to allow additional calls to be added.  The new opcodes are listed below. *
                       10   ;*****************************************************************************
                       11
                       12           NAME    BELDSR - System beeper device service routine
                       13   ;
                       14   ;    PUBLIC DEFINITIONS
                       15   ;
                       16           PUBLIC  BEEP
                       17           PUBLIC  BEEPIO
                       18           PUBLIC  BEEPOF
                       19           PUBLIC  LBEEP
                       20           PUBLIC  SBEEP
                       21   ;
                       22   ;    EXTERNAL REFERENCES
                       23   ;
=0000                  24           SECT    ROMDAT
                       25
                       26           EXTRN   BELEVT:WORD     ; System Bell timing event block      (ROMDAT)
                       27           EXTRN   DSKC_M:BYTE     ; RAM copy of disk control port       (ROMDAT)
=0000                  28           SECT    ROMCOD
                       29           EXTRN   ROMTST:NEAR     ; CRC calculatuon routine             (PUPTST)
                       30           EXTRN   DELAY:NEAR      ; 1 Millisecond delay subroutine      (ROMUTL)
                       31           EXTRN   DSPERZ:NEAR     ; Print error routine                 (OUTPUT)
                       32           EXTRN   MSG:NEAR        ; Print string routine                (OUTPUT)
                       33           EXTRN   GCONFG:NEAR     ; Get pointer to system configuration (CONFIG)
                       34           EXTRN   GCONF1:NEAR     ; Get pointer to system configuration (CONFIG)
                       35           EXTRN   RCONFG:NEAR     ; Return Extra Configuration status   (CONFIG)
                       36   ;
                       37   ;    LOCAL CONSTANTS
                       38   ;
=061B                  39   BELTMR  EQU     1563            ; 1563 ==) 800 Hz for system beeper
=000A                  40   BEPLEN  EQU     10              ; normal system beep length = 250 msec
=0028                  41   LBPLEN  EQU     40              ; Long beep = 1 sec
=0003                  42   SBPLEN  EQU     3               ; short beep = 75 msec
                       43   ;
                       44   ;    EXTERNAL EQU FILES
                       45   ;
                       46   ;       INCLUDE PEG:LATCHES.EQU
                       47   ;       INCLUDE PEG:PORTADDR.EQU
                       48   ;       INCLUDE PEG:TIMERS.EQU
                       49   ;       INCLUDE PEG:TIMEVT.EQU
                       50   ;       INCLUDE PEG:VECTOR.EQU
```

```
                                251
                                252   ;*****************************************************************************
                                253   ;                           CODE SEGMENT DEFINITION
                                254   ;*****************************************************************************
                                255   ;
          =0000                 256           SECT    ROMCOD
                                257           ASSUME  CS:ROMCOD
                                258   ;
                                259   ;*****************************************************************************
                                260   ;                           MODULE ENTRY POINT
                                261   ;*****************************************************************************
                                262   ;   BELL (or BEEPER) DSR - Entry point for ROM user calls - INT 48H
                                263   ;
                                264   ;   INPUT:  AH = Function:
                                265   ;                   0 - Sound the beeper for specified time (@ current frequency)
                                266   ;                         AL = Beep time in 25 millisecond increments
                                267   ;                   1 - Get beeper status, returned in Z-flag:
                                268   ;                         ZF = 0 if beeper is currently enabled
                                269   ;                         ZF = 1 if beeper is not enabled (no sound)
                                270   ;                   2 - Set beep frequency
                                271   ;                         CX = timer value (see routine header for explanation)
                                272   ;                   3 - Beeper ON
                                273   ;                   4 - Beeper OFF
                                274   ;                   5 - Wait subroutine
                                275   ;                   6 - CRC 16 subroutine ES:BX=data address, BP=block size
                                276   ;                         DX=returned CRC, ZF=1 if DX=0
                                277   ;                   7 - Print message @F400:BI terminated by zero.
                                278   ;                   8 - Display system error code in reg BX
                                279   ;                   9 - Get pointer to system configuration in ES:BX
                                280   ;                   A - Get pointer to extra system configuration in ES:BX
                                281   ;                   B - Return Extra sys config. info in AX,BX,CX
                                282   ;                  )B - Invalid opcode
                                283   ;   OUTPUT: as noted above
                                284   ;   USED:   AX, CX (see individual routine headers)
                                285   ;   STACK:  14 bytes  (BEEPER is worst-case)
                                286       ASSUME  DS:ROMDAT
                                287   ;-----------------------------------------------------------------------------
0000'                           288   BEEPIO  PROC    FAR
0000'  FB                       289           STI                     ;;; Interrupts back on
0001'  1E                       290           PUSH    DS              ;;; Save user's current DS
0002'  2E8E 1E C180            291           MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; Set up my DS
0007'  E8 001A                  292           CALL    DOBEEP          ; Do it
000A'  1F                       293           POP     DS
000B'  CA 0002                  294           RET     2               ; *** RETURN *** (Simulated IRET to pass flags)
          =000C                 295   JMPTBL  EQU     $-2
000E'  004C"                    296           DW      OFFSET BEEPST   ;1 Beep status
0010'  0052"                    297           DW      OFFSET BEEPFQ   ;2 Set beeper frequency
0012'  0062"                    298           DW      OFFSET BEEPON   ;3 Turn on beeper (does not go off)
0014'  0070"                    299           DW      OFFSET BEEPOF   ;4 Turn off beeper (used by timer dsr)
0016'  0004"                    300           DW      OFFSET DELAY    ;5 Delay subroutine
0018'  0003"                    301           DW      OFFSET ROMTST   ;6 CRC16 calculation
001A'  0006"                    302           DW      OFFSET MSG      ;7 Print message in ROMCOD CS
```

```
001C'  0005"                303           DW      OFFSET DSPERZ   ;8 Display system error code
001E'  0007"                304           DW      OFFSET QCONFQ   ;9 Get pointer to system configuration
0020'  0008"                305           DW      OFFSET QCONF1   ;A Get pointer to extra system configuration
0022'  0009"                306           DW      OFFSET RCONFG   ;B Return extra configuration info
       =0018                307   JMPTSZ  EQU     $-JMPTBL
                            308   ;
0024'                       309   DOBEEP  PROC    NEAR            ; Dispatch to function specified by AH
0024'  08 E4                310           OR      AH,AH           ; AH = 0
0026'  74 12                311           JZ      BEEPER          ;
0028'  8A C4                312           MOV     AL,AH           ; Convert opcode to jump table index
002A'  98                   313           CBW                     ; Make into a word
002B'  D1 E0                314           SHL     AX,1            ; Index into word table
002D'  3C 18                315           CMP     AL,OFFSET JMPTSZ; Q: Legal routine?
002F'  77 1A                316           JA      BPRET           ;   N: Just return
0031'  93                   317           XCHG    AX,BX           ;   Y: Get routine entry without screwing BX
0032'  2E8B 9F 000C"        318           MOV     BX,WORD PTR CS:JMPTBL[BX] ; Get routine address
0037'  93                   319           XCHG    AX,BX           ;
0038'  FF E0                320           JMP     AX              ; And go to it
                            321   ;*****************************************************************************
                            322   ;   BEEPER - Sound the beeper for the specified time (at current frequency)
                            323   ;
                            324   ;   INPUT:  DS points to ROMDAT
                            325   ;           AL = Beep time in 25 millisecond increments
                            326   ;   OUTPUT: ** beeeeeeep **
                            327   ;   USED:   AX
                            328   ;   STACK:  6 bytes
                            329       ASSUME  DS:ROMDAT
                            330   ;-----------------------------------------------------------------------------
003A'                       331   BEEPER  PROC    NEAR
003A'  3C 00                332           CMP     AL,00           ; Q: Did the user say 00 ?
003C'  74 0D                333           JZ      BPRET           ;   Y: Then exit without sound
003E'  30 E4                334           XOR     AH,AH           ;   N: Zero out the high byte
0040'  A3 0000"             335           MOV     BELEVT+IEVCTR,AX; Put the count into the event block
0043'  80 0E 0000" 80       336           OR      BYTE PTR BELEVT+IEVFLG,1 SHL EVTSTA     ; Set the active flag
0048'  E8 0017              337           CALL    BEEPON          ; Turn on the beeper
004B'                       338   BPRET:
004B'  C3                   339           RET                     ; *** RETURN ***
                            340   ;
                            341   ;*****************************************************************************
                            342   ;   BEEPST - Return beeper status
                            343   ;
                            344   ;   INPUT:  DS points to ROMDAT
                            345   ;   OUTPUT: ZF = 0 if speaker is enabled
                            346   ;           ZF = 1 if speaker is not enabled (no sound)
                            347   ;   USED:   (none)
                            348   ;   STACK:  2 bytes
                            349       ASSUME  DS:ROMDAT
                            350   ;-----------------------------------------------------------------------------
004C'                       351   BEEPST  PROC    NEAR
004C'  F6 06 0002" 01       352           TEST    DSKC_M,SPKREN   ; Look at the speaker bit in memory
0051'  C3                   353           RET                     ; *** RETURN ***
                            354   ;*****************************************************************************
```

```
                        355     ;   BEEPFQ - This subroutine is used to set the frequency of the timer
                        356     ;       that drives the beeper speaker.  The timer's input frequency is
                        357     ;       1.25 MHz and the value passed in CX is used as a divider for this
                        358     ;       frequency.  For example, the system timer (800 HZ) uses a value
                        359     ;       of (1,250,000 Hz / 800 Hz) = 1563.  Note that the caller's interrupt
                        360     ;       status is preserved.
                        361     ;
                        362     ;   INPUT:  CX = Frequency 'value'   ( 1.25 MHz / CX = true freq.)
                        363     ;   OUTPUT: (none)
                        364     ;   USED:   AL,CX
                        365     ;   STACK:  4 bytes
                        366         ASSUME  DS:ROMDAT
                        367     ;-----------------------------------------------------------------------------
0052'                   368     BEEPFQ  PROC    NEAR
0052'  9C               369             PUSHF
0053'  FA               370             CLI                                     ;;; Protect this
0054'  B0 36            371             MOV     AL,(TC_SCO + TC_WRD + TC_MD3) ;;; Set the timer
0056'  E6 17            372             OUT     TIMCMD,AL                       ;;;
0058'  8A C1            373             MOV     AL,CL                           ;;; 1.25 MHz / CX = frequency
005A'  E6 14            374             OUT     TIMER0,AL                       ;;;
005C'  8A C5            375             MOV     AL,CH                           ;;;
005E'  E6 14            376             OUT     TIMER0,AL                       ;;;
0060'  9D               377             POPF
0061'  C3               378             RET                                     ; *** RETURN ***
                        379     ;*****************************************************************************
                        380     ;   BEEPON - This subroutine is called to turn the system beeper on.
                        381     ;       Timing services typically is used to turn it back off.
                        382     ;       Note that the caller's interrupt status is preserved.
                        383     ;
                        384     ;   INPUT:  DS points to ROMDAT
                        385     ;   OUTPUT: (beeper turned on)
                        386     ;   USED:   AL
                        387     ;   STACK:  4 bytes
                        388         ASSUME  CS:ROMCOD, DS:ROMDAT
                        389     ;-----------------------------------------------------------------------------
0062'                   390     BEEPON  PROC    NEAR
0062'  9C               391             PUSHF                                   ; Save interrupt status
0063'  FA               392             CLI                                     ; Protect this
0064'  A0 0002"         393             MOV     AL,DSKC_M                       ;;; Get copy from RAM
0067'  0C 01            394             OR      AL,SPKREN                       ;;; Turn the speaker on
0069'  E6 00            395             OUT     DSKC_P,AL                       ;;;
006B'  A2 0002"         396             MOV     DSKC_M,AL                       ;;;
006E'  9D               397             POPF                                    ;;; Interrupts back on
006F'  C3               398             RET                                     ; *** RETURN ***
                        399     ;
                        400     ;*****************************************************************************
                        401     ;   BEEPOF - This subroutine is called by timing services to turn the
                        402     ;       system beeper off.  The event obviously has to be (re)started
                        403     ;       before this can happen.  Note that the caller's interrupt
                        404     ;       status is preserved.
                        405     ;
                        406     ;   INPUT:  DS points to ROMDAT (only because that's where event block is)
```

```
                                ;   OUTPUT: (beeper turned off)
                                ;   USED:   AL
                                ;   STACK:  4 bytes
                                    ASSUME  CS:ROMCOD, DS:ROMDAT
                                ;-----------------------------------------------------------------------------
0070'                           BEEPOF  PROC    NEAR
0070'  9C                               PUSHF                           ; Save interrupt status
0071'  FA                               CLI                             ; Protect again
0072'  A0 0002"                         MOV     AL,DSKC_M               ;;;
0075'  24 FE                            AND     AL,NOT SPKREN           ;;; Turn speaker off
0077'  E6 00                            OUT     DSKC_P,AL               ;;;
0079'  A2 0002"                         MOV     DSKC_M,AL               ;;;
007C'  9D                               POPF                            ;;; Interrupts back on
007D'  C3                               RET                             ; *** RETURN ***
                                ;-----------------------------------------------------------------------------
                                ;   BEEP - Beep the system bell.  Note the three entry points:
                                ;
                                ;       CALL LBEEP      - Approx. 1 sec    (used for error conditions)
                                ;       CALL SBEEP      - Approx. 75 msec  (used for prompting situations)
                                ;       CALL BEEP       - Approx. 250 msec (normal bell)
                                ;
                                ;   INPUT:  (none)
                                ;   OUTPUT: *** BEEP ***
                                ;   USED:   AX
                                ;   STACK:  14 bytes
                                    ASSUME  CS:ROMCOD, DS:ROMDAT
                                ;-----------------------------------------------------------------------------
007E'                           LBEEP   PROC    NEAR
007E'  B4 28                            MOV     AH,LBPLEN               ; Set up for long beep
0080'  EB 06                            JMP     SHORT BEEP1
                                ;
0082'                           SBEEP   PROC    NEAR
0082'  B4 03                            MOV     AH,SBPLEN               ; Set up for short beep
0084'  EB 02                            JMP     SHORT BEEP1
                                ;
0086'                           BEEP    PROC    NEAR
0086'  B4 0A                            MOV     AH,BEPLEN               ; Set up for normal beep
                                ;       JMP     SHORT BEEP1
0088'                           BEEP1:
0088'  1E                               PUSH    DS                      ; Save DS of caller
0089'  2E8E 1E C180                     MOV     DS,WORD PTR CS:DSADDR+CSWRAP  ; Set up my DS
008E'  51                               PUSH    CX
008F'  B9 061B                          MOV     CX,BELTMR               ; 1.25 MHz / BELTMR = frequency
0092'  E8 FFBD                          CALL    BEEPFQ                  ; Set the beep frequency
0095'  59                               POP     CX
0096'  8A C4                            MOV     AL,AH
0098'  E8 FF9F                          CALL    BEEPER                  ; Turn the beeper on
0098'  1F                               POP     DS
009C'  C3                               RET                             ; *** RETURN ***
                                ;
                                        END

No errors detected
```

```
                        1   ;*****************************************************************************
                        2   ;  TITLE    - CLKDSR - Time-of-day clock I/O and service routine
                        3   ;  ABSTRACT - This module contains the time-of-day clock routines: the
                        4   ;       setting and reading routines, as well as the service routine which
                        5   ;       is called every 100 milliseconds by the timer interrupt service
                        6   ;       routine to keep the T.O.D. clock ticking.
                        7   ;*****************************************************************************
                        8           NAME    CLKDSR - Time-of-day clock I/O and service routines
                        9   ;       SUBTTL
=FFFF                  11   DEBUG   EQU     0FFFFH
                       12   ;
                       13   ;                       PUBLIC DEFINITIONS
                       14   ;
                       15           PUBLIC  CLK_IO
                       16           PUBLIC  CLKSRV
                       17           PUBLIC  SETDAT
                       18           PUBLIC  SETTIM
                       19   ;
                       20   ;                       EXTERNAL REFERENCES
                       21   ;
=0000                  22           SECT    ROMDAT
                       23           EXTRN   DATE:WORD       ; Time-of-day clock - days since 1/1/80(ROMDAT)
                       24           EXTRN   HOURS:BYTE      ; Time-of-day clock - hours             (ROMDAT)
                       25           EXTRN   HUNS:BYTE       ; Time-of-day clock - hundredths        (ROMDAT)
                       26           EXTRN   MINS:BYTE       ; Time-of-day clock - minutes           (ROMDAT)
                       27           EXTRN   SECS:BYTE       ; Time-of-day clock - seconds           (ROMDAT)
                       28   ;
                       29   ;       INCLUDE PEQ:VECTOR.EQU
                       30   ;
                      124
                      125   ;*****************************************************************************
                      126   ;                       MODULE ENTRY POINT
                      127   ;*****************************************************************************
                      128   ;
=0000                 129           SECT    ROMCOD
                      130           ASSUME  CS:ROMCOD
                      131   ;
                      132   ;   TIME AND DATE ROUTINES - INT 4EH
                      133   ;
                      134   ;   INPUT:  AH = Function:
                      135   ;                   0 - Set the date
                      136   ;                   1 - Set the time
                      137   ;                   2 - Get the date & time
                      138   ;                  )2 - NULL call
                      139   ;           BX = Count of days since January 1, 1980
                      140   ;           CH = Hours (0-23)
                      141   ;           CL = Minutes (0-59)
                      142   ;           DH = Seconds (0-59)
                      143   ;           DL = Hundredths of seconds (0-99)
                      144   ;   OUTPUT: (depends on function, registers same as above)
                      145   ;   USED:   AX,BX,CX,DX
                      146   ;   STACK:
```

```
                                147         ASSUME  CS:ROMCOD, DS:NOTHING
                                148     ;
0000'                           149     CLK_IO  PROC    FAR
0000'  FB                       150             STI                     ;;; interrupts back on
0001'  1E                       151             PUSH    DS              ;;; save current DS
0002'  2E8E 1E C180            152             MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; point to our DS
0007'  53                       153             PUSH    BX
0008'  8A FC                    154             MOV     BH,AH           ; 'opcode' in BH
000A'  58                       155             POP     AX              ; date, if any, in AX
000B'  E8 0002                  156             CALL    DO_CLK          ; do the function
000E'  1F                       157             POP     DS
000F'  CF                       158             IRET                    ; *** RETURN ***
                                159     ;
0010'                           160     DO_CLK  PROC    NEAR            ; dispatch to function specified by BH
0010'  08 FF                    161             OR      BH,BH           ; BH = 0
0012'  74 09                    162             JZ      SETDAT
0014'  FE CF                    163             DEC     BH              ; BH = 1
0016'  74 09                    164             JZ      SETTIM
0018'  FE CF                    165             DEC     BH              ; BH = 2
001A'  74 18                    166             JZ      GETDAT
001C'  C3                       167             RET                     ; *** RETURN ***  (Invalid command)
                                168     ;-----------------------------------------------------------------------------
                                169     ;   SET THE DATE
                                170     ;
                                171     ;   INPUT:  AX = Count of days since January 1, 1980
                                172     ;   OUTPUT: (none)
                                173     ;   USED:   AX
                                174     ;   STACK:  2 bytes
                                175         ASSUME  CS:ROMCOD, DS:ROMDAT
                                176     ;
001D'                           177     SETDAT  PROC    NEAR
001D'  A3 0001"                 178             MOV     DATE,AX         ; Store the date
0020'  C3                       179             RET
                                180     ;
                                181     ;-----------------------------------------------------------------------------
                                182     ;   SET THE TIME
                                183     ;
                                184     ;   INPUT:  CH = hours (0-23)       (all numbers are binary)
                                185     ;           CL = minutes (0-59)
                                186     ;           DH = seconds (0-59)
                                187     ;           DL = hundredths of seconds (0-99)
                                188     ;   OUTPUT: (none)
                                189     ;   USED:   CX,DX
                                190     ;   STACK:  2 bytes
                                191         ASSUME  CS:ROMCOD, DS:ROMDAT
                                192     ;
0021'                           193     SETTIM  PROC    NEAR
0021'  FA                       194             CLI                     ;;; hide
0022'  88 2E 0002"              195             MOV     HOURS,CH        ;;; Store the time
0026'  88 0E 0004"              196             MOV     MINS,CL
002A'  88 36 0005"              197             MOV     SECS,DH
002E'  88 16 0003"              198             MOV     HUNS,DL
```

```
0032'  FB                      199          STI                         ;;; back on
0033'  C3                      200          RET                         ;;; *** RETURN ***
                               201  ;-----------------------------------------------------------------------
                               202  ;   GET THE DATE AND TIME
                               203  ;
                               204  ;   INPUT:   (none)
                               205  ;   OUTPUT:  AX = count of days since 01-01-80
                               206  ;            CH = hours
                               207  ;            CL = minutes
                               208  ;            DH = seconds
                               209  ;            DL = hundredths of seconds
                               210  ;   USED:    AX,CX,DX
                               211  ;   STACK:   2 bytes
                               212      ASSUME  CS:ROMCOD, DS:ROMDAT
                               213  ;
0034'                          214  GETDAT  PROC    NEAR
0034'  FA                      215          CLI                         ;;;
0035'  A1 0001"                216          MOV     AX,DATE             ;;; Get the date
0038'  8A 2E 0002"             217          MOV     CH,HOURS            ;;; Get the time
003C'  8A 0E 0004"             218          MOV     CL,MINS
0040'  8A 36 0005"             219          MOV     DH,SECS
0044'  8A 16 0003"             220          MOV     DL,HUNS
0048'  FB                      221          STI                         ;;; back on
0049'  C3                      222          RET                         ;;; *** RETURN ***
                               223  ;-----------------------------------------------------------------------
                               224  ;   CLKSRV - Time-of-day clock service routine.  This routine is called
                               225  ;       every 100 milliseconds by the system timer interrupt service
                               226  ;       routine.  It updates the time and date information kept in RAM.
                               227  ;
                               228  ;   INPUT:   DS points to ROMDAT
                               229  ;   OUTPUT:  T-o-D info in RAM is updated
                               230  ;   USED:    BX
                               231  ;   STACK:   2 bytes
                               232      ASSUME      CS:ROMCOD, DS:ROMDAT
                               233  ;
004A'                          234  CLKSRV  PROC    NEAR
004A'  FA                      235          CLI                             ; Protect this
004B'  BB 0003"                236          MOV     BX,OFFSET HUNS          ;;; point to time/date storage
004E'  80 07 0A                237          ADD     BYTE PTR [BX],10        ;;; bump hundredths
0051'  80 3F 64                238          CMP     BYTE PTR [BX],100       ;;; Q: is it >= 100 ? (= 1 second)
0054'  72 27                   239          JB      CLKXIT                  ;;;   N: that's all
0056'  C6 07 00                240          MOV     BYTE PTR [BX],0         ;;;   Y: reset hundredths
                               241  ;
0059'  43                      242          INC     BX                      ;;; point to seconds
005A'  FE 07                   243          INC     BYTE PTR [BX]
005C'  80 3F 3C                244          CMP     BYTE PTR [BX],60        ;;; Q: 60 ?
005F'  72 1C                   245          JB      CLKXIT                  ;;;   N: that's all
0061'  C6 07 00                246          MOV     BYTE PTR [BX],0         ;;;   Y: reset seconds
                               247  ;
0064'  43                      248          INC     BX                      ;;; point to minutes
0065'  FE 07                   249          INC     BYTE PTR [BX]
0067'  80 3F 3C                250          CMP     BYTE PTR [BX],60        ;;; Q: 60 ?
```

```
006A'  72 11                251             JB      CLKXIT                  ;;;  N: that's all
006C'  C6 07 00             252             MOV     BYTE PTR [BX],0         ;;;   Y: reset minutes
                            253     ;
006F'  43                   254             INC     BX                      ;;; point to hours
0070'  FE 07                255             INC     BYTE PTR [BX]
0072'  80 3F 18             256             CMP     BYTE PTR [BX],24        ;;; Q: 24 ?
0075'  72 06                257             JB      CLKXIT                  ;;;   N: that's all
0077'  C6 07 00             258             MOV     BYTE PTR [BX],0         ;;;   Y: reset hours
                            259     ;
007A'  43                   260             INC     BX                      ;;; point to days
007B'  FF 07                261             INC     WORD PTR [BX]           ;;; bump it
007D'                       262     CLKXIT:
007D'  FB                   263             STI
007E'  C3                   264             RET                             ;;; *** RETURN ***
                            265     ;
                            266             END
```

No errors detected

```
                                1  ;*****************************************************************************
                                2  ;  TITLE    - CONFIG - Return system configuration
                                3  ;  ABSTRACT - This routine is a ROM function (software interrupt) that
                                4  ;       returns the current system configuration, as determined during
                                5  ;       powerup initialization.
                                6  ;*****************************************************************************
                                7          NAME    CONFIG - Return System Configuration
                                8  ;       SUBTTL
                                9  ;
                               10  ;                           PUBLIC DEFINITIONS
                               11  ;
                               12          PUBLIC  CFG_IO
                               13          PUBLIC  QCONFG
                               14          PUBLIC  QCONF1
                               15          PUBLIC  RCONFG
                               16  ;
                               17  ;
                               18  ;                           EXTERNAL REFERENCES
                               19  ;
          =0000                20          SECT    ROMDAT
                               21          EXTRN   SYSCON:WORD     ; System configuration storage
                               22  ;*****************************************************************************
                               23  ;                           LOCAL CONSTANTS
                               24  ;*****************************************************************************
                               25  ;       INCLUDE PEG:VECTOR.EGGU
                               26  ;
                              120  ;*****************************************************************************
                              121  ;                           CODE SEGMENT DEFINITION
                              122  ;*****************************************************************************
                              123  ;
          =0000               124          SECT    ROMCOD
                              125          ASSUME  CS:ROMCOD, DS:ROMDAT
                              126  ;
                              127  ;*****************************************************************************
                              128  ;                           MODULE ENTRY POINT
                              129  ;*****************************************************************************
                              130  ;
                              131  ;   CFG_IO - Returns the system configuration word and contiguous memory
                              132  ;       size in 16-byte blocks.
                              133  ;
                              134  ;   INPUT:  (none)
                              135  ;   OUTPUT: AX = (SYSCON) - System configuration word
                              136  ;           BX = (MEMSIZ) - Contiguous RAM in 16-byte blocks
                              137  ;   USED:   AX, BX
                              138  ;   STACK:
                              139  ;-----------------------------------------------------------------------------
0000'                         140  CFG_IO  PROC    FAR
0000'  FB                     141          STI                                     ; Interrupts back on
0001'  1E                     142          PUSH    DS
0002'  2E8E 1E C180           143          MOV     DS,word ptr CS:DSADDR+CSWRAP    ; Point to ROMDAT
0007'  A1 0001"               144          MOV     AX,SYSCON                       ; Pick up config word
000A'  2E8B 1E C198           145          MOV     BX,word ptr CS:MEMSIZ+CSWRAP    ; Pick up memory size
```

```
000F'  1F                      146          POP     DS
0010'  CF                      147          IRET
                               148  ;
                               149  ;*****************************************************************************
                               150  ;   QCONFQ - Returns a pointer to system configuration word and other useful
                               151  ;       information.  Called NEAR from general ROM I/O handler in BELDSR.
                               152  ;
                               153  ;   INPUT:   (none)
                               154  ;   OUTPUT:  ES:BX = pointer to system configuration info
                               155  ;            ES:[BX-3] = DSKC_M copy of DSKC_P latch
                               156  ;            ES:[BX-2] = PRCO_M copy of PRCO_P latch
                               157  ;            ES:[BX-1] = DSKS_M copy of DSKS_P latch
                               158  ;
                               159  ;   USED:    BX,ES
                               160  ;   STACK:
                               161  ;-----------------------------------------------------------------------------
0011'                          162  QCONFQ  PROC    NEAR
0011'  BB 0001"                163          MOV     BX,offset SYSCON                ; point to config area (OFFSET)
0014'  1E                      164          PUSH    DS                              ; and set up segment
0015'  07                      165          POP     ES
0016'  C3                      166          RET
                               167  ;*****************************************************************************
                               168  ;   QCONF1 - Returns a pointer to other system information.
                               169  ;       Called NEAR from general ROM I/O handler in BELDSR.
                               170  ;
                               171  ;   INPUT:   (none)
                               172  ;   OUTPUT:  ES:BX = pointer to additional system configuration info
                               173  ;            ES:[BX-3] = (MEMSIZ) - Size of memory in 16 byte words (word)
                               174  ;            ES:[BX-1] = (INTCTR) - Pending interrupt count (byte)
                               175  ;            ES:[BX+0] = (DRVBYT) - Drive type byte
                               176  ;            ES:[BX+1] = (SYSCO1) - Additional system configuration byte
                               177  ;            ES:[BX+2] = (SYSCO2) - Additional system configuration word
                               178  ;   USED:    BX,ES
                               179  ;   STACK:
                               180  ;-----------------------------------------------------------------------------
0017'                          181  QCONF1  PROC    NEAR
0017'  31 DB                   182          XOR     BX,BX                   ; Set up segment of config area
0019'  8E C3                   183          MOV     ES,BX
001B'  BB 019B                 184          MOV     BX,offset DRVBYT        ; Offset of config area
001E'  C3                      185          RET
                               186  ;*****************************************************************************
                               187  ;   RCONFQ - Return extra system configuration information in registers.
                               188  ;       Called NEAR from general ROM I/O handler in BELDSR.
                               189  ;
                               190  ;   INPUT:   (none)
                               191  ;   OUTPUT:  AL = DRVBYT - Drive type byte
                               192  ;            AH = INTCTR - Pending interrupt count (not very useful)
                               193  ;            BX = SYSCO1 - Extra system configuration word #1
                               194  ;            CX = SYSCO2 - Extra system configuration word #2
                               195  ;   USED:    AX,BX,CX
                               196  ;   STACK:
                               197  ;-----------------------------------------------------------------------------
```

```
001F'                          198  RCONFG  PROC    NEAR
001F'  2EA1 C19A               199          MOV     AX,word ptr CS:INTCTR+CSWRAP    ; Pick up INTCTR and DRVBYT
0023'  86 C4                   200          XCHG    AL,AH                           ; Put in correct registers
0025'  2E8B 1E C19C            201          MOV     BX,word ptr CS:SYSCO1+CSWRAP    ; Pick up SYSCO1
002A'  2E8B 0E C19E            202          MOV     CX,word ptr CS:SYSCO2+CSWRAP    ; Pick up SYSCO2
002F'  C3                      203          RET
                               204          END
```

No errors detected

```
                        ;*****************************************************************************
                        ; TITLE    - CRT DSR
                        ;
                        ; COMPUTER - 8088 ASSEMBLY LANGUAGE
                        ;
                        ; ABSTRACT - This module contains the necessary set of routines needed
                        ;           for interacting with the CRT.  The interface to these routines
                        ;           is via a software interrupt (INT  49H) with the CRT function
                        ;           number in register AH.  Additional parameters required by each
                        ;           function and their register designations are described below
                        ;           as INPUT under the given function.
                        ;*****************************************************************************
                        ;
                        ;                               SET CRT MODE
                        ;                                 AH = 00H
                        ;
                        ;       THIS ROUTINE IS USED TO SET THE CRT MODE (NOTE:  this is not
                        ;       applicable to PEGASUS but is provided for compatibility with
                        ;       the IBM CRT calling conventions), IT DOES NOTHING AND SIMPLY
                        ;       RETURNS.
                        ;
                        ;       INPUT - AH = 0
                        ;               AL = 1-7 (ignored)
                        ;
                        ;       OUTPUT - None
                        ;
                        ;       USED - None
                        ;
                        ;-----------------------------------------------------------------------------
                        ;
                        ;                     SET CURSOR TYPE BASED ON REGISTER CX
                        ;                               AH = 01H
                        ;
                        ;       INPUT - CH = Bits 4-0 cursor start line (0-B hex)
                        ;               Bits 6-5 cursor type  i.e. 00 = no blink
                        ;                                          01 = no cursor
                        ;                                          10 = 1/16 blink
                        ;                                          11 = 1/32 blink
                        ;
                        ;               CL = Bits 4-0 cursor end line (0-B hex)
                        ;               Bits 7-5 not used
                        ;
                        ;       OUTPUT - none
                        ;
                        ;       USED - AX,CX,DL
                        ;-----------------------------------------------------------------------------
                        ;
                        ;               SET CURSOR AT COLUMN(DH),ROW(DL) POSITION
                        ;                               AH = 02H
                        ;
                        ;       INPUT - DH = X coordinate (column)
                        ;               DL = Y coordinate (row)
```

```
                ;
                ;       OUTPUT - CURSOR WRITTEN AT (X,Y) COORDINATES
                ;
                ;-----------------------------------------------------------------------
                ;
                ;                       READ CURRENT CURSOR POSITION
                ;                               AH = 03H
                ;
                ;       THIS ROUTINES READS THE CURRENT CURSOR POSITION AND RETURNS
                ;       THE POSITION IN DX AS (COLUMN,ROW), (X,Y).
                ;
                ;       INPUT - NONE
                ;
                ;       OUTPUT - DH = (column),DL = (row)
                ;                CH,CL = Cursor type
                ;
                ;       USED - DX,CX,AX,BX,
                ;
                ;-----------------------------------------------------------------------
                ;
                ;                       READ LIGHT PEN POSITION
                ;                               AH = 04H
                ;
                ;       THIS ROUTINE READS THE LIGHT PEN AND RETURNS IT'S POSITION
                ;       TO THE CALLER.
                ;
                ;       INPUT - NONE
                ;
                ;       OUTPUT - RETURN NO ACTION FOR NOW
                ;
                ;       USED -
                ;
                ;-----------------------------------------------------------------------
                ;
                ;                       SET ACTIVE DISPLAY PAGE
                ;                               AH = 05H
                ;
                ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE (IT DOES NOTHING)
                ;       AND IS PROVIDED FOR IBM COMPATIBILITY ONLY.
                ;
                ;-----------------------------------------------------------------------
                ;
                ;                       SCROLL TEXT BLOCK UP
                ;                               AH = 06H
                ;
                ;                       SCROLL TEXT BLOCK DOWN
                ;                               AH = 07H
                ;
                ;
                ;       THIS ROUTINE "SCROLLS" THE SCREEN IN EITHER DIRECTION BY
                ;       SIMPLY PERFORMING A MOVS FROM THE SOURCE TO THE DESTINATION
                ;       LOCATION OF THE SCREEN.  THIS METHOD PROVIDES FOR DEFINING
```

```
105   ;       A WINDOW AND MOVING IT TO ANOTHER POSITION ON THE SCREEN.
106   ;       THE SOURCE WINDOW BOUNDARY COORDINATES ARE AS FOLLOWS AND
107   ;       THE SAME APPLIES FOR THE DESTINATION COORDINATES:
108   ;
109   ;       Upper left = (src. col.,src row)
110   ;       Upper right = (src. col. + col. cnt.,src. row)
111   ;       Lower left = (src. col.,src. row + line cnt.)
112   ;       Lower right = ( src. col. + col. cnt.,src. row + line cnt.)
113   ;
114   ;       INPUT - AL = 0 blank out source text
115   ;             * AL ) 0 Don't blank source
116   ;            ** (DH,DL) = (Source column,Source row) of scroll
117   ;               (BH,BL) = (Destination column,Destination row) of scroll
118   ;               CH = Column count(1-80)     CL = Line count(1-25)
119   ;
120   ;       * NOTE:  If the source and destination windows
121   ;                overlap then some of the source will
122   ;                be overwritten.
123   ;
124   ;      ** NOTE:  If destination is less than source
125   ;                then the scroll is a scroll up and
126   ;                to the left, if source is less than
127   ;                destination then the scroll is a scroll
128   ;                down and to the right.
129   ;
130   ;
131   ;                        Example:  You want to scroll the screen up "n"
132   ;                                  lines.
133   ;
134   ;                                  input - source col,row (DH,DL) = (0,n)
135   ;                                          dest. col,row  (BH,BL) = (0,0)
136   ;                                          column count (CH) = 80
137   ;                                          line count (CL) = 25-n
138   ;
139   ;       OUTPUT - NONE
140   ;
141   ;       USED -
142   ;
143   ;_______________________________________________________________________
144   ;
145   ;               READ CHARACTER AND ATTRIBUTE AT CURRENT CURSOR POSITION
146   ;                                  AH = 08H
147   ;
148   ;       THIS ROUTINE READS AND RETURNS THE CHARACTER AND ATTRIBUTE AT
149   ;       THE CURRENT CURSOR POSITION.
150   ;
151   ;       INPUT - NONE
152   ;
153   ;       OUTPUT - AH = Attribute of character read
154   ;                AL = chracter read
155   ;
156   ;       USED - AX,
```

```
157   ;
158   ;-----------------------------------------------------------------------
159   ;
160   ;           WRITE CHARACTER AND ATTRIBUTE AT CURRENT CURSOR POSITION
161   ;                               AH = 09H
162   ;
163   ;       THIS ROUTINE IS USED TO WRITE A CHARACTER AND ASSOCIATED
164   ;       ATTRIBUTE AT THE CURRENT CURSOR POSITION.  THE USER SHOULD
165   ;       NOTE THAT THE CURSOR IS NOT AUTOMATICALLY ADVANCED WITH THIS
166   ;       ROUTINE.
167   ;
168   ;       INPUT - AL = Character to write
169   ;               BL = Atribute of character
170   ;               CX = Count of chars. to write
171   ;
172   ;       OUTPUT - None
173   ;
174   ;       USED - AX,BL,CX
175   ;
176   ;-----------------------------------------------------------------------
177   ;
178   ;               WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
179   ;                               AH = 0AH
180   ;
181   ;       THIS ROUTINE IS USED FOR WRITING A CHARACTER ONLY AT
182   ;       THE CURRENT CURSOR POSITION.  IT IS ALSO USED BY THE
183   ;       ABOVE PROCEDURE BUT IGNORES THE ATTRIBUTE PARAMETER.
184   ;       AGAIN THE USER SHOULD NOTE THAT THE CURSOR IS NOT AUTO-
185   ;       MATICALLY ADVANCED AND IS LEFT AT IT'S ORIGINAL POSITION.
186   ;
187   ;       INPUT - AL = Character to write
188   ;               CX = Count of chars. to write
189   ;
190   ;       OUTPUT - None
191   ;
192   ;       USED - AX,CX
193   ;
194   ;-----------------------------------------------------------------------
195   ;
196   ;                       SET GRAPHICS COLOR PALETTE
197   ;                               AH = 0BH
198   ;
199   ;       THIS ROUTINE IS USED TO SET THE COLOR PALETTE FOR THE GRAPHICS
200   ;       BANKS A,B,C.
201   ;
202   ;       INPUT -
203   ;
204   ;       OUTPUT - Palette set accordingly
205   ;
206   ;       USED -
207   ;
208   ;-----------------------------------------------------------------------
```

```
209   ;
210   ;               WRITE GRAPHICS PIXEL AT (X,Y) LOCATION
211   ;                       AH = 0CH
212   ;
213   ;       THIS ROUTINE IS USED TO WRITE A GRAPHICS PIXEL AT THE
214   ;       SPECIFIED LOCATION.
215   ;
216   ;       INPUT - (To be determined)
217   ;
218   ;       OUTPUT -
219   ;
220   ;       USED -
221   ;
222   ;_______________________________________________________________________
223   ;
224   ;               READ GRAPHICS PIXEL AT (X,Y) LOCATION
225   ;                       AH = 0DH
226   ;
227   ;       THIS ROUTINE IS USED TO READ A GRAPHICS PIXEL AT THE
228   ;       SPECIFIED LOCATION.
229   ;
230   ;       INPUT - (To be determined)
231   ;
232   ;       OUTPUT -
233   ;
234   ;       USED -
235   ;
236   ;_______________________________________________________________________
237   ;
238   ;               ASCII TELETYPE WRITE ROUTINE
239   ;                       AH = 0EH
240   ;
241   ;       THIS ROUTINE IS USED TO WRITE CHARACTERS TO THE SCREEN IN AN
242   ;       ASCII TELETYPE MANNER.  WRITING STARTS AT THE CURRENT CURSOR
243   ;       POSITION AND THE CURSOR IS ADVANCED AUTOMATICALLY TO ITS NEXT
244   ;       LOCATION.  THE SCREEN IS SCROLLED AUTOMATICALLY IF NEED BE
245   ;       (i.e. writing past the end of screen)*  AND CONTROL CHARACTERS
246   ;       ARE EXECUTED (CR,LF,BS and BEL) INSTEAD OF WRITTEN.
247   ;
248   ;               * NOTE:  If a status line is currently being implemented
249   ;                        a scroll will occur on the line previous to the
250   ;                        start of the status region as if that line were
251   ;                        the end of the screen.
252   ;
253   ;       INPUT - AL = Character to write
254   ;
255   ;       OUTPUT - None
256   ;
257   ;       USED -
258   ;
259   ;_______________________________________________________________________
260   ;
```

```
                        ;                       CURRENT VIDEO STATE
                        ;                            AH = 0FH
                        ;
                        ;       THIS ROUTINE IS USED TO RETURN THE STATE (or MODE) OF THE
                        ;       CRT TO THE CALLER  (NOTE: this is not applicable to PEGASUS
                        ;       but is provided for compatibility with the IBM CRT calling
                        ;       conventions). IT DOES NOTHING AND SIMPLY RETURNS.
                        ;
                        ;       INPUT - None
                        ;
                        ;       OUTPUT - (To be determined)
                        ;
                        ;       USED -
                        ;
                        ;-------------------------------------------------------------------
                        ;
                        ;               WRITE BLOCK OF ATTRIBUTES & CHARS @ CURSOR
                        ;                            AH = 10H
                        ;
                        ;       THIS ROUTINE IS USED TO WRITE A GIVEN BLOCK OF CHARACTERS TO
                        ;       THE SCREEN STARTING AT THE CURSOR USING THE SPECIFIED ATTRIBUTE.
                        ;       ALL CONTROL CHARACTERS (i.e. CR,LF,BEL etc.) ARE IGNORED AS SUCH
                        ;       AND PRINTED ON THE SCREEN AS ASCII CHARACTERS.
                        ;
                        ;
                        ;       INPUT - AL = Attribute of characters to write
                        ;               DX = Segment location of character block
                        ;               BX = Offset location of character block
                        ;               CX = Length of character block (not to exceed 2000)
                        ;
                        ;               NOTE:  This routine does not increment the cursor
                        ;                      location. The caller must be aware of the
                        ;                      cursor location before using this routine
                        ;                      and make sure that the block will "fit" on
                        ;                      the screen or he will lose what won't "fit". *
                        ;
                        ;       OUTPUT - None
                        ;
                        ;       USED - None
                        ;
                        ;       * CX must be ( or = the number of char. positions left on the screen.
                        ;         CX (block length) must be ( or = 2000-((cur. row - 1 * 80)+cur. col.)
                        ;         Attempting to write characters beyond the end of screen will cause
                        ;         them to be be lost (not written).
                        ;
                        ;-------------------------------------------------------------------
                        ;
                        ;                       WRITE BLOCK OF CHARS. ONLY @ CURSOR
                        ;                            AH = 11H
                        ;
                        ;       THIS ROUTINE IS USED TO WRITE A GIVEN BLOCK OF CHARACTERS TO
                        ;       THE SCREEN STARTING AT THE CURSOR ALL CONTROL CHARACTERS
```

```
                ;       (i.e. CR,LF,BEL etc.) ARE IGNORED AS SUCH AND PRINTED ON THE
                ;       SCREEN AS ASCII CHARACTERS.
                ;
                ;
                ;       INPUT - DX = Segment location of character block
                ;               BX = Offset location of character block
                ;               CX = Length of character block (not to exceed 2000)
                ;
                ;               NOTE:  This routine does not increment the cursor
                ;                      location. The caller must be aware of the
                ;                      cursor location before using this routine
                ;                      and make sure that the block will "fit" on
                ;                      the screen or he will lose what won't "fit". *
                ;                      With this type of write the characters will
                ;                      take on the attributes of the last attribute
                ;                      change.
                ;
                ;       OUTPUT - None
                ;
                ;       USED - None
                ;
                ;       * CX must be ( or = the number of char. positions left on the screen.
                ;         CX (block length) must be ( or = 2000-((cur. row - 1 * 80)+cur. col.)
                ;         Attempting to write characters beyond the end of screen will cause
                ;         them to be be lost (not written).
                ;
                ;_______________________________________________________________________________
                ;
                ;                       CHANGE ENTIRE SCREEN ATTRIBUTES
                ;                               AH = 12H
                ;
                ;               THIS ROUTINE CHANGES THE ENTIRE SCREEN ATTRIBUTES
                ;                       RIGHT BEFORE YOUR VERY EYES
                ;
                ;       INPUT - AL = ATRIBUTE TO USE
                ;
                ;       OUTPUT - None
                ;
                ;       USED - None
                ;
                ;_______________________________________________________________________________
                ;
                ;               CLEAR THE CRT SCREEN MEMORY AND HOME CURSOR
                ;                               AH = 13H
                ;
                ;       INPUT - None
                ;
                ;       OUTPUT - None
                ;
                ;       USED - AX,CX,DL,DI
                ;
                ;_______________________________________________________________________________
```

```
                        ;
                        ;               CLEAR GRAPHICS MEMORY AND HOME THE PIXEL CURSOR
                        ;                               AH = 14H
                        ;
                        ;       THIS ROUTINE IS USED TO CLEAR ALL THREE GRAPHICS BANKS AND HOME
                        ;       THE PIXEL CURSOR (i.e. place cursor at (0,0) ).
                        ;
                        ;       INPUT - None
                        ;
                        ;       OUTPUT - None
                        ;
                        ;       USED -
                        ;
                        ;_______________________________________________________________________
                        ;
                        ;                       SET TTY STATUS LINE.
                        ;                            AH = 15H
                        ;
                        ;       THIS ROUTINE IS USED TO DEFINE THE STARTING LINE OF THE STATUS
                        ;       REGION.  IT IS ASSUMED THAT THIS LINE THROUGH THE END OF THE
                        ;       SCREEN IS RESERVED FOR STATUS IMFORMATION.  WHEN USING THIS IN
                        ;       CONJUNCTION WITH A TTY WRITE THE END OF SCREEN FOR THE TTY ROUTINE
                        ;       IS SET AT THE STATUS LINE MINUS ONE.
                        ;
                        ;       INPUT - CH = 0 (always)
                        ;               CL = Start line number of status line (0-24 decimal)
                        ;
                        ;                       NOTE:  CH must always be zero. This represents
                        ;                              the first column of the status line.
                        ;                              If an attempt is made to set a status
                        ;                              line which falls prior to the current
                        ;                              line of the cursor then no status line
                        ;                              is implemented.
                        ;
                        ;               CX = 0 = Full screen TTY.
                        ;
                        ;       OUTPUT - None
                        ;
                        ;       USED - None
                        ;
                        ;_______________________________________________________________________
                        ;
                        ;                       SET ATTRIBUTE LATCH
                        ;                            AH = 16H
                        ;
                        ;       THIS ROUTINE IS USED TO WRITE THE ATTRIBUTE LATCH WITHOUT
                        ;       HAVING TO WRITE A CHARACTER.
                        ;
                        ;       INPUT - BL = Attribute to use
                        ;
                        ;       OUTPUT - Latch set accordingly
                        ;
```

```
                        417   ;-------------------------------------------------------------------------------
                        418   ;       ********************************************************************
                        419   ;       *** ITEMS BELOW THIS LINE WERE ADDED OPCODES IN THE V1.20 ROM ***
                        420   ;       ********************************************************************
                        421   ;_______________________________________________________________________________
                        422   ;
                        423   ;                       GET PHYSICAL DISPLAY BEGIN OFFSET
                        424   ;                               AH = 17H
                        425   ;
                        426   ;       THIS ROUTINE IS USED TO RETURN THE OFFSET (FROM DE00H) OF
                        427   ;       THE PHYSICAL BEGINNING OF DISPLAY.
                        428   ;
                        429   ;       INPUT - NONE
                        430   ;
                        431   ;       OUTPUT - DX = Display begin offset
                        432   ;
                        433   ;
                        434   ;-------------------------------------------------------------------------------
                        435   ;
                        436   ;                               PRINT TTY STRING
                        437   ;                                   AH = 18H
                        438   ;
                        439   ;       THIS ROUTINE IS USED TO PRINT A STRING OF CHARACTERS IN THE
                        440   ;       USERS CB: (GOTTEN FROM THE STACK) WITH A TTY WRITE.
                        441   ;
                        442   ;       INPUT - BX = Address (offset) of the string
                        443   ;                    Where: [BX] byte 0 = length of string
                        444   ;                           [BX] byte 1 = first char. of string
                        445   ;
                        446   ;       OUTPUT - NONE
                        447   ;
                        448   ;
                        449   ;-------------------------------------------------------------------------------
                        450   ;
                        451           NAME    CRTDSR (CRT DSR CODE)
                        452           SUBTTL
=FFFF                   453   DEBUG   EQU     0FFFFH
```

```
                        ;*****************************************************************************
                        ;                       PUBLIC DEFINITIONS
                        ;*****************************************************************************
                        ;
                                PUBLIC  CRTOUT
                                PUBLIC  CRT_IO
                                PUBLIC  CRTRET
                        ;
                        ;*****************************************************************************
                        ;                       EXTERNAL REFERENCES
                        ;*****************************************************************************
                        ;
=0000                           SECT    ROMCOD
                        ;
                                EXTRN   BEEP:NEAR               ; BEEP BELL ROUTINE             (BELDSR)
                                EXTRN   INILST:NEAR             ; INIT LOCAL STORAGE            (CRTTST)
                        ;
=0000                           SECT    ROMDAT
                        ;
                                EXTRN   CURPOS:WORD             ; CURSOR POSITION POINTER       (ROMDAT)
                                EXTRN   CURTYP:WORD             ; CURRENT CURSOR TYPE           (ROMDAT)
                                EXTRN   THSLIN:WORD             ; CURRENT LINE POINTER          (ROMDAT)
                                EXTRN   DISBEG:WORD             ; CURRENT DISPLAY BEGIN POINTER(ROMDAT)
                                EXTRN   DISEND:WORD             ; CURRENT DISPLAY END POINTER   (ROMDAT)
                                EXTRN   CRTCNT:WORD             ; CURRENT CHARS ON LINE COUNT   (ROMDAT)
                                EXTRN   STATLN:WORD             ; STATUS LINE BEGINNING         (ROMDAT)
                                EXTRN   RETFLG:BYTE             ; LOCAL RETURN FLAG             (ROMDAT)
                                EXTRN   CRTHLD:BYTE             ; PAUSE STATUS BYTE             (ROMDAT)
                                EXTRN   ATTSAV:BYTE             ; ATTRIBUTE LATCH SAVE          (ROMDAT)
                        ;
                        ;*****************************************************************************
                        ;                       LOCAL CONSTANTS
                        ;*****************************************************************************
                        ;
                        ;       INCLUDE PEQ:ASCII.EQU
                        ;       INCLUDE PEQ:CRT.EQU
                        ;       INCLUDE PEQ:VECTOR.EQU
                        ;
                        ;
                        ;*****************************************************************************
                        ;                       LOCAL MACROS
                        ;*****************************************************************************
                        ;
                        ;
                                SUBTTL  MAIN
                        ;*****************************************************************************
                        ;                       MODULE ENTRY POINT
                        ;*****************************************************************************
                        ;
                        ;   CRT_IO - INT 49H
                        ;
=0000                           SECT    ROMCOD
```

```
                                696             ASSUME CS:ROMCOD, DS:ROMDAT, ES:CRTDAT
                                697     ;
0000'                           698     CRT_IO  PROC    FAR
0000'  FB                       699             STI                             ;;; INTERRUPTS BACK ON
0001'  FC                       700             CLD                             ; CLEAR DIRECTION FLAG
0002'  E8 0033                  701             CALL    CRTOUT                  ; DO IT
0005'  CF                       702             IRET                            ; *** RETURN ***
                                703     ;
0006'                           704     CRTABL  LABEL   WORD                    ; JUMP TABLE OF CRT FUNCTIONS
0006'  0066"                    705             DW      CRTRET                  ; SET SCREEN MODE (DOES A RETURN)
0008'  0073"                    706             DW      STCTYP                  ; SET CURSOR TYPE
000A'  007F"                    707             DW      STCPOS                  ; SET CURSOR POSITION
000C'  00B0"                    708             DW      RDCPOS                  ; READ CURSOR POSITION
000E'  0066"                    709             DW      CRTRET                  ; READ LIGHT PEN POS. (DOES A RETURN)
0010'  0066"                    710             DW      CRTRET                  ; SET ACTIVE DISP. PAGE (DOES A RETURN)
0012'  00C3"                    711             DW      SCROLL                  ; SCROLL UP
0014'  00C3"                    712             DW      SCROLL                  ; SCROLL DOWN
0016'  01B9"                    713             DW      RATCHR                  ; READ ATTRIBUTE & CHARACTER @ CURSOR
0018'  01C8"                    714             DW      WATCHR                  ; WRITE ATTRIBUTE & CHARACTER @ CURSOR
001A'  01CD"                    715             DW      WRTCHR                  ; WRITE CHARACTER ONLY @ CURSOR
001C'  0066"                    716             DW      CRTRET                  ; SET COLOR PALETTE (DOES A RETURN)
001E'  0066"                    717             DW      CRTRET                  ; WRITE DOT (DOES A RETURN)
0020'  0066"                    718             DW      CRTRET                  ; READ DOT (DOES A RETURN)
0022'  0218"                    719             DW      WRTTTY                  ; ASCII TELETYPE WRITE
0024'  0066"                    720             DW      CRTRET                  ; CURRENT VIDEO STATE
0026'  039C"                    721             DW      WATBLK                  ; WRITE BLOCK OF CHARS & ATTRIBUTES
0028'  03A0"                    722             DW      WBLOCK                  ; WRITE BLOCK OF CHARS ONLY
002A'  03C4"                    723             DW      CHSCAT                  ; CHANGE SCREEN ATTRIBUTES
002C'  03DB"                    724             DW      CLRTXT                  ; CLEAR ALPHA SCREEN
002E'  0405"                    725             DW      CLRGRP                  ; CLEAR GRAPHICS SCREEN
0030'  01EF"                    726             DW      SETTTY                  ; SET TTY STATUS LINE
0032'  042A"                    727             DW      SETATT                  ; SET ATTRIBUTE LATCH
0034'  0432"                    728             DW      GTDSBG                  ; GET DISPLAY BEGIN
0036'  0439"                    729             DW      PRTTTY                  ; PRINT TTY STRING
       =0032                    730     ENDTAB  EQU     $-CRTABL                ; END OF JUMP TABLE
                                731     ;
0038'                           732     CRTOUT  PROC    NEAR
0038'  57                       733             PUSH    DI                      ; SAVE DI
0039'  56                       734             PUSH    SI                      ; SAVE SI
003A'  1E                       735             PUSH    DS                      ; SAVE OLD DS
003B'  06                       736             PUSH    ES                      ; SAVE OLD ES
003C'  53                       737             PUSH    BX                      ; SAVE BX
003D'  50                       738             PUSH    AX                      ; SAVE AX
003E'  53                       739             PUSH    BX                      ; SAVE BX (TEMPORARILY)
003F'  2E8E 1E C180             740             MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; SET UP DS AS ROMDAT
0044'  BB 0000'                 741             MOV     BX,CRTDAT               ; SET UP ES
0047'  8E C3                    742             MOV     ES,BX                   ; AS CRTDAT
0049'  5B                       743             POP     BX                      ; RESTORE BX
004A'                           744     HOLDUP:
004A'  80 3E 000B" 00           745             CMP     CRTHLD,0                ; Q: IS SCREEN I/O IN PAUSE STATE
004F'  75 F9                    746             JNE     HOLDUP                  ;    Y: THEN WAIT TIL IT AIN'T
0051'  CD 57                    747             INT     CMPINT                  ;    N: GO MAP ANY CHARACTERS
```

```
0053'  8A C4                748             MOV     AL,AH                   ; GET INTO LOW BYTE
0055'  30 E4                749             XOR     AH,AH                   ; ZERO TO HIGH BYTE
0057'  D1 E0                750             SAL     AX,1                    ; WORD OFFSET FOR TABLE LOOKUP
0059'  8B F0                751             MOV     SI,AX                   ; SI = OFFSET FOR JUMP
005B'  3D 0032              752             CMP     AX,OFFSET ENDTAB        ; SEE IF VALID FUNCTION
005E'  58                   753             POP     AX                      ; GET  PARAMETER BACK
005F'  73 05                754             JAE     CRTRET                  ; NO, DO NOTHING IF NOT IN RANGE
                            755     ;
0061'                       756     CRTQQ:
0061'  2EFF A4 0006"        757             JMP     CRTABL[SI]              ; TAKE JUMP BASED ON FUNCTION
                            758     ;
                            759     ;*****************************************************************************
                            760     ;               THIS IS THE COMMON RETURN POINT FOR ALL CRT CALLS
                            761     ;           IF THE RETFLQ IS SET THEN THE RETURN IS A "LOCAL" RETURN
                            762     ;       (i.e. DO NOT POP THE STACK AND RETURN OUT OF CRT CODE AREA BUT
                            763     ;               INSTEAD RETURN WITHIN CRT CODE FROM LOCAL CALLER)
                            764     ;*****************************************************************************
                            765     ;
0066'                       766     CRTRET  PROC    NEAR
0066'  80 3E 000A" 00       767             CMP     RETFLQ,00H              ; Q: CONDITIONAL RETURN FLAG SET ?
006B'  75 05                768             JNE     LOCRET                  ;   Y: DO A "LOCAL" RETURN
006D'  5B                   769             POP     BX                      ;
006E'  07                   770             POP     ES                      ;
006F'  1F                   771             POP     DS                      ;
0070'  5E                   772             POP     SI                      ; ALL CRT CODE JUMPS TO CRTRET AND
0071'  5F                   773             POP     DI                      ;
0072'  C3                   774     LOCRET: RET                             ; RETURNS THRU HERE TO CLEAR THE STACK
                            775     ;
                            776     ;
                            777     ;*****************************************************************************
                            778     ;       THIS ROUTINE IS USED TO SET THE CRT MODE (NOTE: this is not
                            779     ;       applicable to PEGASUS but is provided for compatibility with
                            780     ;       the IBM CRT calling conventions), IT DOES NOTHING AND SIMPLY
                            781     ;       RETURNS.
                            782     ;
                            783     ;       INPUT - AH = 0
                            784     ;               AL = 1-7 (ignored)
                            785     ;
                            786     ;       OUTPUT - None
                            787     ;
                            788     ;       USED - None
                            789     ;
                            790     ;*****************************************************************************
                            791     ;
                            792     ;SETMOD PROC    NEAR
                            793     ;       ASSUME  ES:CRTDAT
                            794     ;       JMP     CRTRET                  ; DO NOTHING
                            795     ;
                            796     ;*****************************************************************************
                            797     ;               SET CURSOR TYPE BASED ON REGISTER CX
                            798     ;
                            799     ;       INPUT - CH = Bits 4-0 cursor start line (0 - 0BH)
```

```
                        800   ;                   Bits 6-5 cursor type  i.e. 00 = no blink
                        801   ;                                              01 = no cursor
                        802   ;                                              10 = 1/16 blink
                        803   ;                                              11 = 1/32 blink
                        804   ;
                        805   ;               CL = Bits 4-0 cursor end line (0 - 0BH)
                        806   ;                    Bits 7-5 not used
                        807   ;
                        808   ;       OUTPUT - none
                        809   ;
                        810   ;       USED - AX,CX,DL
                        811   ;*****************************************************************************
                        812   ;
0073'                   813   STCTYP  PROC    NEAR                    ; SET CURSOR TYPE
                        814           ASSUME  ES:CRTDAT
0073'  89 0E 0004"      815           MOV     CURTYP,CX               ; SAVE CURSOR TYPE LOCALLY
0077'  91               816           XCHG    AX,CX                   ; AX=CURSOR TYPE SPECIFIED BY CX
0078'  B2 0A            817           MOV     DL,CURSTR               ; GET CURSOR SCAN LINE REG HIGH
007A'  E8 040C          818           CALL    PTRUP                   ; GO UPDATE CRTC
007D'  EB E7            819           JMP     CRTRET                  ; ALL DONE
                        820   ;
                        821   ;*****************************************************************************
                        822   ;               SET CURSOR AT COLUMN(DH),ROW(DL) POSITION
                        823   ;
                        824   ;       INPUT - DH = X (column) coordinate (0-79)
                        825   ;               DL = Y (row) coordinate (0-24)
                        826   ;
                        827   ;       OUTPUT - CURSOR WRITTEN AT (X,Y) COORDINATES
                        828   ;*****************************************************************************
                        829   ;
007F'                   830   STCPOS  PROC    NEAR                    ; SET CURSOR POSITION
                        831           ASSUME  ES:CRTDAT               ;
007F'  80 FE 50         832           CMP     DH,80                   ; Q: IS DH ( 80 COLUMNS ?
0082'  73 2A            833           JAE     NOCSET                  ;   N: THEN DON'T SET THE CURSOR
0084'  80 FA 18         834           CMP     DL,24                   ; Q: IS DL ( = 24 LINES ?
0087'  77 25            835           JA      NOCSET                  ;   N: THEN DON'T SET THE CURSOR
0089'  B0 50            836           MOV     AL,80                   ;   Y: AL = CHARACTERS ON THE LINE
008B'  8B DA            837           MOV     BX,DX                   ; GET COL/ROW IN BX
008D'  F6 E3            838           MUL     BL                      ; MULTIPLY Y COORDINATE BY 80
008F'  03 06 0006"      839           ADD     AX,DISBEG               ; ADD Y TO DISBEG
0093'  25 07FF          840           AND     AX,7FFH                 ; MAGIC NUMBER
0096'  A3 0005"         841           MOV     THSLIN,AX               ; UPDATE THIS LINE
0099'  31 D2            842           XOR     DX,DX                   ; CLEAR DX REGISTER
009B'  8A D7            843           MOV     DL,BH                   ; PUT X coordinate IN REG. DX
009D'  89 16 0008"      844           MOV     CRTCNT,DX               ; PUT X AS CHAR COUNT ON LINE
00A1'  01 D0            845           ADD     AX,DX                   ; THIS IS CURSOR POSITION
00A3'  A3 0003"         846           MOV     CURPOS,AX               ; SAVE NEW CURSOR POSITION
00A6'  B2 0E            847           MOV     DL,CURPSH               ; DL = CURSOR POS. ADD. REG. HIGH
00A8'  E8 03C6          848           CALL    PTRADJ                  ; ADJUST CURSOR TO LOGICAL POSITION
00AB'  E8 03DB          849           CALL    PTRUP                   ; GO WRITE CURSOR
                        850   ;
00AE'                   851   NOCSET:
```

```
00AE' EB B6                852          JMP     CRTRET                  ; ALL DONE
                           853   ;
                           854   ;*****************************************************************************
                           855   ;       THIS ROUTINES READS THE CURRENT CURSOR POSITION AND RETURNS
                           856   ;       THE POSITION IN DX AS (COLUMN,ROW), (X,Y).
                           857   ;
                           858   ;       INPUT - NONE
                           859   ;
                           860   ;       OUTPUT - DH = (column),DL = (row)
                           861   ;                CH,CL = Cursor type
                           862   ;
                           863   ;       USED - DX,CX,AX,BX,
                           864   ;
                           865   ;*****************************************************************************
                           866   ;
00B0'                      867   RDCPOS  PROC    NEAR                    ; READ CURSOR POSITION
                           868           ASSUME  ES:CRTDAT
00B0' A1 0003"             869           MOV     AX,CURPOS               ; AX = ABSOLUTE CURSOR POSITION
00B3' E8 0388              870           CALL    PTRADJ                  ; ADJUST TO LOGICAL POSITION
00B6' 29 D8                871           SUB     AX,BX                   ; AX = OFFSET FROM DISBEQ
                           872   ;                                       ;  (BX = DISBEQ RETURNED FROM CALL)
00B8' B3 50                873           MOV     BL,80                   ;
00BA' F6 F3                874           DIV     BL                      ; DIVIDE AX BY 80
00BC' 92                   875           XCHG    DX,AX                   ; PUT COL,ROW IN DH,DL
00BD' 8B 0E 0004"          876           MOV     CX,CURTYP               ; GET CURSOR TYPE IN CX
00C1' EB A3                877           JMP     CRTRET                  ; RETURN
                           878   ;
                           879   ;*****************************************************************************
                           880   ;       THIS ROUTINE READS THE LIGHT PEN AND RETURNS IT'S POSITION
                           881   ;       TO THE CALLER.
                           882   ;
                           883   ;       INPUT - NONE
                           884   ;
                           885   ;       OUTPUT - RETURN NO ACTION FOR NOW
                           886   ;
                           887   ;       USED -
                           888   ;
                           889   ;*****************************************************************************
                           890   ;
                           891   ;RLPPOS PROC    NEAR
                           892   ;       ASSUME  ES:CRTDAT
                           893   ;       JMP     CRTRET
                           894   ;
                           895   ;*****************************************************************************
                           896   ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE (IT DOES NOTHING)
                           897   ;       AND IS PROVIDED FOR IBM COMPATIBILITY ONLY.
                           898   ;*****************************************************************************
                           899   ;
                           900   ;STDSPG PROC    NEAR
                           901   ;       ASSUME  ES:CRTDAT
                           902   ;       JMP     CRTRET
                           903   ;
```

```
                                904   ;*****************************************************************************
                                905   ;       THIS ROUTINE "SCROLLS" THE SCREEN IN EITHER DIRECTION BY
                                906   ;       SIMPLY PERFORMING A MOVS FROM THE SOURCE TO THE DESTINATION
                                907   ;       LOCATION OF THE SCREEN.  THIS METHOD PROVIDES FOR DEFINING
                                908   ;       A WINDOW AND MOVING IT TO ANOTHER POSITION ON TEH SCREEN.
                                909   ;       THE SOURCE WINDOW BOUNDARY COORDINATES ARE AS FOLLOWS AND
                                910   ;       THE SAME APPLIES FO RTHE DESTINATION COORDINATES:
                                911   ;
                                912   ;       Upper left = (src. col.,src row)
                                913   ;       Upper right = (src. col. + col. cnt.,src. row)
                                914   ;       Lower left = (src. col.,src. row + line cnt.)
                                915   ;       Lower right = ( src. col. + col. cnt.,src. row + line cnt.)
                                916   ;
                                917   ;       INPUT - AL = 0 blank out source text
                                918   ;             * AL ) 0 Don't blank source
                                919   ;            ** (DH,DL) = (Source column,Source row) of scroll
                                920   ;               (BH,BL) = (Destination column,Destination row) of scroll
                                921   ;               CH = Column count(1-80)    CL = Line count(1-25)
                                922   ;
                                923   ;       * NOTE: If the source and destination windows
                                924   ;               overlap then some of the source will
                                925   ;               be overwritten.
                                926   ;
                                927   ;      ** NOTE: If destination is less than source
                                928   ;               then the scroll is a scroll up and
                                929   ;               to the left, if source is less than
                                930   ;               destination then the scroll is a scroll
                                931   ;               down and to the right.
                                932   ;
                                933   ;
                                934   ;                       Example: You want to scroll the screen up "n"
                                935   ;                                lines.
                                936   ;
                                937   ;                                input - source col,row (DH,DL) = (0,n)
                                938   ;                                        dest. col,row  (BH,BL) = (0,0)
                                939   ;                                        column count (CH) = 80
                                940   ;                                        line count (CL) = 25-n
                                941   ;
                                942   ;       OUTPUT - NONE
                                943   ;
                                944   ;       USED -
                                945   ;
                                946   ;*****************************************************************************
                                947   ;
00C3'                           948   SCROLL  PROC    NEAR                    ; SCROLL THE SCREEN UP OR DOWN
                                949           ASSUME  ES:CRTDAT
00C3'  80 FF 4F                 950           CMP     BH,79                   ; Q: DEST. COL ) 79 ?
00C6'  77 0F                    951           JA      LEAVE                   ;    Y: THEN DO NOTHING
00C8'  80 FE 4F                 952           CMP     DH,79                   ; Q: SRC. COL ) 79 ?
00CB'  77 0A                    953           JA      LEAVE                   ;    Y: THEN DO NOTHING
00CD'  80 FB 18                 954           CMP     BL,24                   ; Q: DEST. ROW ) 24 ?
00D0'  77 05                    955           JA      LEAVE                   ;    Y: THEN DO NOTHING
```

```
00D2' 80 FA 18          956             CMP     DL,24                   ; Q: SRC. ROW ) 24 ?
00D5' 76 02             957             JBE     GO_ON                   ;   N: THEN CONTINUE
00D7'                   958     LEAVE:
00D7' EB 8D             959             JMP     CRTRET                  ; QUIT IF ANY VALUES OUT OF RANGE
00D9'                   960     GO_ON:
00D9' 50                961             PUSH    AX                      ; SAVE AX
00DA' 53                962             PUSH    BX                      ; SAVE DESTINATION COL/ROW
00DB' 52                963             PUSH    DX                      ; SAVE SOURCE COL/ROW
00DC' 31 C0             964             XOR     AX,AX                   ; CLEAR AX
00DE' B0 50             965             MOV     AL,80                   ;
00E0' F6 E2             966             MUL     DL                      ; AX = SRC. ROW OFFSET FROM 0
00E2' 8A D6             967             MOV     DL,DH                   ; DX = SRC. COL.
00E4' 30 F6             968             XOR     DH,DH                   ; CLEAR DH
00E6' 01 D0             969             ADD     AX,DX                   ; AX = SRC COL/ROW ABS LOCATION
00E8' 96                970             XCHG    SI,AX                   ; SI = SRC COL/ROW ABS LOCATION
00E9' 03 36 0006"       971             ADD     SI,DISBEG               ; SI = SRC ABS LOCATION FROM DISBEG
00ED' 5A                972             POP     DX                      ; RESTORE DX
00EE' 31 C0             973             XOR     AX,AX                   ; CLEAR AX
00F0' B0 50             974             MOV     AL,80                   ;
00F2' F6 E3             975             MUL     BL                      ; AX = DEST. ROW OFFSET FROM 0
00F4' 8A DF             976             MOV     BL,BH                   ; BX = DEST. COL.
00F6' 30 FF             977             XOR     BH,BH                   ; CLEAR BH
00F8' 01 D8             978             ADD     AX,BX                   ; AX = DEST. COL/ROW ABS LOCATION
00FA' 97                979             XCHG    DI,AX                   ; DI = DEST. COL/ROW ABS LOCATION
00FB' 03 3E 0006"       980             ADD     DI,DISBEG               ; DI = DEST. ABS LOCATION FROM DISBEG
00FF' 5B                981             POP     BX                      ; RESTORE BX
0100' 58                982             POP     AX                      ; RESTORE AX
0101' 39 FE             983             CMP     SI,DI                   ; Q: SRC. BEG. IN FRONT OF DEST. BEG. ?
0103' 7C 3D             984             JL      SCRLDN                  ;   Y: THEN SCROLL DOWN
                        985     ;
0105'                   986     SCRLUP:                                 ;   N: THEN SCROLL UP
0105' 8B D9             987             MOV     BX,CX                   ; BX = COL. AND LINE COUNTS
0107' 31 C9             988             XOR     CX,CX                   ; CLEAR CX FOR REP AND LOOP
0109' 8A CB             989             MOV     CL,BL                   ; SET UP CX FOR LOOP
                        990     ;
010B'                   991     NORMUP:                                 ;   N: THEN NORMAL SCROLL UP
010B' 8B 16 0006"       992             MOV     DX,DISBEG               ; DX = BEGINNING OF DISPLAY ADDRESS
010F' 81 C2 07CF        993             ADD     DX,7CFH                 ; DX = END OF DISPLAY ADDRESS
0113'                   994     NRMUP0:
0113' 51                995             PUSH    CX                      ; SAVE IT
0114' 56                996             PUSH    SI                      ; SAVE SRC. LOCATION
0115' 57                997             PUSH    DI                      ; SAVE DEST. LOCATION
0116' 31 C9             998             XOR     CX,CX                   ; CLEAR CX FOR REP
0118' 8A CF             999             MOV     CL,BH                   ; CX = COL. COUNT
011A'                  1000     NRMUP1:
011A' 39 D6            1001             CMP     SI,DX                   ; Q: SOURCE OFF THE END OF SCREEN ?
011C' 7F 1E            1002             JG      QUIT1                   ;   Y: THEN QUIT IT
011E' 3C 00            1003             CMP     AL,0                    ; Q: BLANKING SOURCE TEXT ?
0120' 74 06            1004             JE      SPECUP                  ;   Y: THEN SPECIAL UP CONDITION
0122' 26A4             1005             MOVS    BYTE PTR [DI],ES:BYTE PTR[SI] ; MOVS OF COL. COUNT LENGTH
0124' E2 F4            1006             LOOP    NRMUP1                  ; NEXT CHARACTER
0126' EB 07            1007             JMP     SHORT NXTUP             ; NEXT LINE
```

```
0128'                          1008  SPECUP:
0128'  E8 006B                 1009        CALL    MOVEIT                  ; GO DO THE MOVING AND LATCH SETTING
0128'  46                      1010        INC     SI                      ; NEXT SOURCE CHARACTER
012C'  47                      1011        INC     DI                      ; NEXT DESTINATION LOCATION
012D'  E2 EB                   1012        LOOP    NRMUP1                  ; DO IT FOR COL. COUNT TIMES
012F'                          1013  NXTUP:
012F'  5F                      1014        POP     DI                      ; RESTORE DEST. LOC.
0130'  83 C7 50                1015        ADD     DI,80                   ; NEXT DESTINATION LINE
0133'  5E                      1016        POP     SI                      ; RESTORE SRC. LOC.
0134'  83 C6 50                1017        ADD     SI,80                   ; NEXT SOURCE LINE
0137'  59                      1018        POP     CX                      ; RESTORE LINE COUNT
0138'  E2 D9                   1019        LOOP    NRMUPO                  ; DO IT LINE COUNT TIMES
013A'  EB 03                   1020        JMP     SHORT   SCRLRT          ; ...AND RETURN
                               1021  ;
013C'                          1022  QUIT1:
013C'  5F                      1023        POP     DI                      ; CLEAR THE STACK
013D'  5E                      1024        POP     SI                      ;
013E'  59                      1025        POP     CX                      ;
                               1026  ;
013F'                          1027  SCRLRT:
013F'  E9 FF24                 1028        JMP     CRTRET                  ; ...AND LEAVE
                               1029  ;
                               1030  ;
                               1031  ;*****************************************************************************
                               1032  ;       THIS IS THE SCROLL DOWN PORTION OF THE SCROLL.  IT DOES THE
                               1033  ;       THE MOVS SEQUENCE STARTING FROM THE END OF THE SOURCE
                               1034  ;       WINDOW TO THE END OF THE DESTINATION WINDOW BACKWARDS.
                               1035  ;*****************************************************************************
                               1036  ;
0142'                          1037  SCRLDN:                                 ;
0142'  51                      1038        PUSH    CX                      ; SAVE CX
0143'  50                      1039        PUSH    AX                      ; SAVE AX
0144'  30 E4                   1040        XOR     AH,AH                   ; CLEAR AH
0146'  B0 50                   1041        MOV     AL,80                   ;
0148'  FE C9                   1042        DEC     CL                      ; MAKE LINE COUNT 0 RELATIVE
014A'  FE CD                   1043        DEC     CH                      ; MAKE COL. COUNT 0 RELATIVE
014C'  F6 E1                   1044        MUL     CL                      ;
014E'  8A CD                   1045        MOV     CL,CH                   ;
0150'  30 ED                   1046        XOR     CH,CH                   ; CLEAR CH
0152'  01 C8                   1047        ADD     AX,CX                   ; AX = SRC. END OFFSET FROM SRC. BEGIN
0154'  01 C6                   1048        ADD     SI,AX                   ; SI = SRC. END COL/ROW ABS. LOCATION
0156'  01 C7                   1049        ADD     DI,AX                   ; DI = DEST. END COL/ROW ABS. LOCATION
0158'  58                      1050        POP     AX                      ; RESTORE AX
0159'  59                      1051        POP     CX                      ; RESTORE COL. AND LINE COUNTS
015A'  8B D9                   1052        MOV     BX,CX                   ; BX = COL. AND LINE COUNTS
015C'  30 ED                   1053        XOR     CH,CH                   ; CLEAR CX FOR REP AND LOOP
015E'  8A CB                   1054        MOV     CL,BL                   ; SET UP CX FOR LOOP
                               1055  ;
0160'                          1056  NORMDN:
0160'  8B 16 0006"             1057        MOV     DX,DISBEG               ; AX = BEGINNING OF DISPLAY ADDRESS
0164'  81 C2 07CF              1058        ADD     DX,7CFH                 ; AX = END OF DISPLAY ADDRESS
0168'  FD                      1059        STD                             ; SET DIRECTION FLAG FOR MOVS
```

```
0169'                           1060    NRMDNO:
0169'  51                       1061            PUSH    CX                      ; SAVE IT
016A'  56                       1062            PUSH    SI                      ; SAVE SRC. LOCATION
016B'  57                       1063            PUSH    DI                      ; SAVE DEST. LOCATION
016C'  30 ED                    1064            XOR     CH,CH                   ; CLEAR CX FOR REP
016E'  8A CF                    1065            MOV     CL,BH                   ; CX = COL. COUNT
0170'                           1066    NRMDN1:
0170'  39 D7                    1067            CMP     DI,DX                   ; Q: DEST. OFF THE END OF SCREEN ?
0172'  7F 1F                    1068            JQ      QUIT2                   ;   Y: THEN QUIT IT
0174'  3C 00                    1069            CMP     AL,0                    ; Q: BLANK SOURCE TEXT ?
0176'  74 06                    1070            JE      SPECDN                  ;   Y: THEN SCROLL DOWN WITH BLANKING
0178'  26A4                     1071            MOVS    BYTE PTR [DI],ES:BYTE PTR[SI] ; MOVS OF COL. COUNT LENGTH
017A'  E2 F4                    1072            LOOP    NRMDN1                  ;   N: THEN KEEP GOING
017C'  EB 07                    1073            JMP     SHORT NXTDN             ; NEXT LINE
017E'                           1074    SPECDN:
017E'  E8 0015                  1075            CALL    MOVEIT                  ; GO DO THE MOVING AND LATCH SETTING
0181'  4E                       1076            DEC     SI                      ; NEXT SOURCE CHARACTER
0182'  4F                       1077            DEC     DI                      ; NEXT DESTINATION POSITION
0183'  E2 EB                    1078            LOOP    NRMDN1                  ;   N: DO IT FOR COL. COUNT TIMES
0185'                           1079    NXTDN:
0185'  5F                       1080            POP     DI                      ; RESTORE DEST. LOC.
0186'  83 EF 50                 1081            SUB     DI,80                   ; NEXT DESTINATION LINE
0189'  5E                       1082            POP     SI                      ; RESTORE SRC. LOC.
018A'  83 EE 50                 1083            SUB     SI,80                   ; NEXT SOURCE LINE
018D'  59                       1084            POP     CX                      ; RESTORE LINE COUNT
018E'  E2 D9                    1085            LOOP    NRMDNO                  ; DO IT LINE COUNT TIMES
0190'  FC                       1086            CLD                             ; CLEAR DIRECTION FLAG
0191'  EB AC                    1087            JMP     SHORT   SCRLRT          ; ...AND RETURN
                                1088    ;
0193'                           1089    QUIT2:
0193'  FC                       1090            CLD                             ; CLEAR DIRECTION FLAG
0194'  EB A6                    1091            JMP     SHORT   QUIT1           ; ...AND LEAVE
                                1092    ;
                                1093    ;*******************************************************************************
                                1094    ;       THIS ROUTINE IS USED TO DO THE BLANKING MANEUVER REQUIRED BY
                                1095    ;       THE SCROLLING ROUTINE WHEN BLANKING OF SOURCE TEXT IS REQUIRED.
                                1096    ;       IT SAVES THE ATTRIBUTE LATCH, SAVE THE CHAR'S ATTRIBUTE, SET THE
                                1097    ;       LATCH TO THE DEFAULT ATTRIBUTE (HIGH INTENSITY), BLANKS THE SRC.
                                1098    ;       CHAR POSITION, RESTORE THE ATTRIBUTE OF THE CHAR., WRITES THE
                                1099    ;       CHAR. @ THE DESTINATION AND THEN RESTORES THE LATCH TO IT'S
                                1100    ;       ORIGINAL SETTING.
                                1101    ;
                                1102    ;       INPUT - ES = CRTDAT
                                1103    ;               DS = ROMDAT
                                1104    ;
                                1105    ;       OUTPUT - IT DID IT
                                1106    ;
                                1107    ;       USED - NONE
                                1108    ;*******************************************************************************
                                1109    ;
0196'                           1110    MOVEIT  PROC    NEAR
0196'  50                       1111            PUSH    AX
```

```
0197'  E8 0307              1112          CALL    SAVATT                  ; SAVE THE LATCH
019A'  268A 04              1113          MOV     AL,ES:BYTE PTR [SI]     ; GET FIRST SRC. CHAR.
019D'  268A 26 1800         1114          MOV     AH,ES:BYTE PTR ATTLAT   ; SAVE THE SOURCE CHAR'S ATTRIBUTE
01A2'  26C6 06 1800 0F      1115          MOV     ES:BYTE PTR ATTLAT,HIGHAT ; SET THE LATCH TO THE DEFAULT
01A8'  26C6 04 20           1116          MOV     ES:BYTE PTR [SI],SPACE  ; CLEAR SOURCE TEXT
01AC'  2688 26 1800         1117          MOV     ES:BYTE PTR ATTLAT,AH   ; RESTORE THE CHAR'S ATTRIBUTE
01B1'  2688 05              1118          MOV     ES:BYTE PTR [DI],AL     ; WRITE IT AT DEST. LOCATION
01B4'  E8 02FC              1119          CALL    RSTATT                  ; RESET THE ATTRIBUTE LATCH
01B7'. 58                   1120          POP     AX
01B8'  C3                   1121          RET
                            1122  ;
                            1123  ;*****************************************************************************
                            1124  ;       THIS ROUTINE READS AND RETURNS THE CHARACTER AND ATTRIBUTE AT
                            1125  ;       THE CURRENT CURSOR POSITION.
                            1126  ;
                            1127  ;       INPUT - NONE
                            1128  ;
                            1129  ;       OUTPUT - AH = Attribute of character read
                            1130  ;                AL = character read
                            1131  ;
                            1132  ;       USED - AX,DI
                            1133  ;
                            1134  ;*****************************************************************************
                            1135  ;
01B9'                       1136  RATCHR  PROC    NEAR                    ; READ CHAR. & ATTRIBUTE @ CURSOR
                            1137          ASSUME  ES:CRTDAT
01B9'  8B 3E 0003"          1138          MOV     DI,CURPOS               ; DI = CURRENT CURSOR POSITION
01BD'  268A 05              1139          MOV     AL,ES:BYTE PTR [DI]     ; AL = CHARACTER @ CURSOR
01C0'  268A 26 1800         1140          MOV     AH,ES:BYTE PTR ATTLAT   ; AH = ATTRIBUTE
01C5'  E9 FE9E              1141          JMP     CRTRET                  ;
                            1142  ;
                            1143  ;*****************************************************************************
                            1144  ;       THIS ROUTINE IS USED TO WRITE A CHARACTER AND ASSOCIATED
                            1145  ;       ATTRIBUTE AT THE CURRENT CURSOR POSITION.  THE USER SHOULD
                            1146  ;       NOTE THAT THE CURSOR IS NOT AUTOMATICALLY ADVANCED WITH THIS
                            1147  ;       ROUTINE.
                            1148  ;
                            1149  ;       INPUT - AL = Character to write
                            1150  ;               BL = Atribute of character
                            1151  ;               CX = Count of chars. to write
                            1152  ;
                            1153  ;       OUTPUT - None
                            1154  ;
                            1155  ;       USED - AX,BL,CX
                            1156  ;
                            1157  ;*****************************************************************************
                            1158  ;
01C8'                       1159  WATCHR  PROC    NEAR                    ; WRITE ATT. & CHAR. @ CURSOR
                            1160          ASSUME  ES:CRTDAT
01C8'  2688 1E 1800         1161          MOV     ES:BYTE PTR ATTLAT,BL   ; WRITE THE ATTRIBUTE LATCH
                            1162  ;
                            1163  ;*****************************************************************************
```

```
                        1164    ;       THIS IS THE ENTRY POINT FOR WRITING A CHARACTER ONLY
                        1165    ;       @ THE CURRENT CURSOR POSITION.  IT IS ALSO USED BY THE
                        1166    ;       ABOVE PROCEDURE BUT IGNORES THE ATTRIBUTE PARAMETER.
                        1167    ;       AGAIN THE USER SHOULD NOTE THAT THE CURSOR IS NOT AUTO-
                        1168    ;       MATICALLY ADVANCED AND IS LEFT AT IT'S ORIGINAL POSITION.
                        1169    ;
                        1170    ;       INPUT - AL = Character to write
                        1171    ;               CX = Count of chars. to write
                        1172    ;
                        1173    ;       OUTPUT - None
                        1174    ;
                        1175    ;       USED - AX,CX
                        1176    ;
                        1177    ;*******************************************************************************
                        1178    ;
01CD'                   1179    WRTCHR  PROC    NEAR                    ; WRITE CHAR. ONLY @ CURSOR
                        1180            ASSUME  ES:CRTDAT
01CD'  83 F9 00         1181            CMP     CX,0                    ; Q: ANY CHARACTERS TO WRITE ?
01D0'  76 1A            1182            JBE     NOWRIT                  ;   N: THEN DON'T WRITE ANY
01D2'  50               1183            PUSH    AX                      ; SAVE AX
01D3'  A1 0003"         1184            MOV     AX,CURPOS               ; AX = CURPOS PHYSICAL LOCATION
01D6'  E8 0298          1185            CALL    PTRADJ                  ; GO GET IT'S LOGICAL POSITION
01D9'  97               1186            XCHG    DI,AX                   ; PUT IT IN DI
01DA'  A1 0007"         1187            MOV     AX,DISEND               ; AX = DISPLAY END PHYSICAL LOCATION
01DD'  E8 0291          1188            CALL    PTRADJ                  ; GET IT'S LOGICAL LOCATION
01E0'  93               1189            XCHG    BX,AX                   ; PUT IT IN BX
01E1'  58               1190            POP     AX                      ; RESTORE AX
01E2'                   1191    WRTLOP:
01E2'  2688 05          1192            MOV     ES:MEMBEG[DI],AL        ; WRITE CHAR. STARTING AT CURSOR
01E5'  47               1193            INC     DI                      ; NEXT LOCATION
01E6'  39 DF            1194            CMP     DI,BX                   ; Q: NEXT LOCATION OFF END OF SCREEN ?
01E8'  77 02            1195            JA      NOWRIT                  ;   Y: THEN DON'T WRITE IT THERE
01EA'  E2 F6            1196            LOOP    WRTLOP                  ; DO IT CX TIMES
01EC'                   1197    NOWRIT:
01EC'  E9 FE77          1198            JMP     CRTRET                  ; ...AND RETURN
                        1199    ;
                        1200    ;*******************************************************************************
                        1201    ;       THIS ROUTINE IS USED TO SET THE COLOR PALETTE FOR THE GRAPHICS
                        1202    ;       BANKS A,B,C.
                        1203    ;
                        1204    ;       INPUT -
                        1205    ;
                        1206    ;       OUTPUT - Palette set accordingly
                        1207    ;
                        1208    ;       USED -
                        1209    ;
                        1210    ;*******************************************************************************
                        1211    ;
                        1212    ;SETPAL PROC     NEAR
                        1213    ;       ASSUME  ES:CRTDAT
                        1214    ;       JMP     CRTRET
                        1215    ;
```

```
1216  ;*************************************************************************
1217  ;       THIS ROUTINE IS USED TO WRITE A GRAPHICS PIXEL AT THE
1218  ;       SPECIFIED LOCATION.
1219  ;
1220  ;       INPUT -
1221  ;
1222  ;       OUTPUT -
1223  ;
1224  ;       USED -
1225  ;
1226  ;*************************************************************************
1227  ;
1228  ;WRPIXL PROC    NEAR
1229  ;       ASSUME  ES:CRTDAT
1230  ;       JMP     CRTRET
1231  ;
1232  ;*************************************************************************
1233  ;       THIS ROUTINE IS USED TO READ A GRAPHICS PIXEL AT THE
1234  ;       SPECIFIED LOCATION.
1235  ;
1236  ;       INPUT -
1237  ;
1238  ;       OUTPUT -
1239  ;
1240  ;       USED -
1241  ;
1242  ;*************************************************************************
1243  ;
1244  ;RDPIXL PROC    NEAR
1245  ;       ASSUME  ES:CRTDAT
1246  ;       JMP     CRTRET
1247  ;
1248  ;*************************************************************************
1249  ;       THIS ROUTINE IS USED TO SET THE COLOR PALETTE FOR THE GRAPHICS
1250  ;       BANKS A,B,C.
1251  ;
1252  ;       INPUT - BH = COLOR ID
1253  ;               BL = COLOR VALUE (OOOOORGB)
1254  ;
1255  ;       OUTPUT - Palette set accordingly
1256  ;
1257  ;       USED - CL, BL IS DESTROYED
1258  ;
1259  ;*************************************************************************
1260  ;
1261  ;SETPAL PROC    NEAR
1262  ;       ASSUME  ES:CRTDAT,DS:ROMDAT
1263  ;       MOV     CL,BH                   ;COPY BIT ADDRESS TO SHIFT COUNT
1264  ;       MOV     DI,GR_RED               ;DI = POINTS TO RED PALETTE LATCH
1265  ;       MOV     SI,OFFSET REDPAL        ;SI = POINTS TO LOCAL COPY OF PALETTES
1266  ;       MOV     AL,03H                  ;AL = LOOP COUNT
1267  ;PALOOP:
```

```
1268    ;       MOV     AH,[SI]                 ;AH = CURRENT RED PALETTE SETTING
1269    ;       CALL    DOPAL                   ;
1270    ;       MOV     ES:BYTE PTR [DI],AH     ;SET THE LATCH
1271    ;       INC     SI                      ;
1272    ;       INC     DI                      ;
1273    ;       DEC     AL                      ;
1274    ;       JNZ     PALOOP                  ;DO ALL THREE PALETTES
1275    ;       JMP     CRTRET                  ;...AND LEAVE
1276    ;DOPAL:
1277    ;       ROR     AH,CL                   ;MOVE BIT TO CHANGE TO LSB
1278    ;       RCR     BL,1                    ;GET NEW BIT INTO CARRY
1279    ;       RCR     AH,1                    ;MERGE BIT INTO MSB,DROP OLD BIT
1280    ;       ROL     AH,1                    ;BACK IN SHIFTED POSITION
1281    ;       ROL     AH,CL                   ;PUT BACK INTO PROPER POSITION
1282    ;       RET
1283    ;
1284    ;*****************************************************************************
1285    ;       THIS ROUTINE IS USED TO WRITE A GRAPHICS PIXEL AT THE
1286    ;       SPECIFIED LOCATION.
1287    ;
1288    ;       INPUT - CX = COLUMN ADDRESS
1289    ;               DX = ROW ADDRESS
1290    ;               AL = COLOR ID   , THAT IS VALUE TO PUT INTO PLANES
1291    ;                                 NOT COLOR THAT WILL COME FROM PALETTE
1292    ;
1293    ;       OUTPUT - TO BITS IN FRAME BUFFER MEMORY (GRAPHICS BIT MAP)
1294    ;
1295    ;       USED -
1296    ;
1297    ;       FORMULAE - BYTE ADDRESS=(ROW*LINELEN+(COLUMN/8)) XOR 1
1298    ;                  BIT ADDRESS = COLUMN MOD 8
1299    ;*****************************************************************************
1300    ;
1301    ;WRPIXL PROC    NEAR
1302    ;       ASSUME  DS:ROMDAT
1303    ;       CALL    MKCURS                  ;A LOCAL ROUTINE FOR CONVERSION OF
1304    ;                                       ;ROW/COLUMN TO BYTE/ BIT
1305    ;       MOV     DI,DX                   ;
1306    ;       MOV     BX,GRASEG
1307    ;       MOV     ES,BX                   ;POINT ES TO PLANE 1
1308    ;       CALL    WRTPXL                  ;DO IT FOR BANK A
1309    ;       MOV     BX,GRBSEG
1310    ;       MOV     ES,BX                   ;
1311    ;       CALL    WRTPXL                  ; DO IT FOR BANK B
1312    ;       MOV     BX,GRCSEG
1313    ;       MOV     ES,BX
1314    ;       CALL    WRTPXL                  ; DO IT FOR BANK C
1315    ;       JMP     CRTRET                  ; ...AND LEAVE
1316    ;WRTPXL:
1317    ;       RCR     AL,1                    ;BIT TO ENTER TO CARRY FLAG
1318    ;       OR      ES:BYTE PTR [DI],CL     ;SET THE BIT THEN
1319    ;       JB      WREND                   ;
```

```
1320    ;       AND     ES:BYTE PTR [DI].CH     ;
1321    ;WREND:
1322    ;       RET                             ;
1323    ;
1324    ;*******************************************************************************
1325    ;       THIS SUBROUTINE CONVERTS ROW COLUMN ADDRESS INTO BYTE ADDRESS AND
1326    ;               HIGH AND LOW ACTIVE BIT MASKS
1327    ;       INPUT - CX = COLUMN ADDRESS (<= 736)
1328    ;               DX = ROW ADDRESS (<= 300)
1329    ;       OUTPUT - CH = ONE BIT LOW
1330    ;                CL = ONE BIT HIGH
1331    ;                DX = BYTE ADDRESS
1332    ;
1333    ;MKCURS PROC NEAR
1334    ;       MOV     BX,CX           ;MOVE COLUMN OUT OF CX
1335    ;       PUSH    AX              ;NEED A FOR MULTIPLY
1336    ;       MOV     CX,BLNLEN       ;CX = BYTES PER ROW
1337    ;       MOV     AX,DX           ;MOVE ROW TO MULTIPLIER
1338    ;       MUL     CX              ;TIMES NUMBER OF BYTES PER ROW
1339    ;       MOV     CL,3            ;DIVIDE COLUMNS BY 8 TO GET BYTE COLUMN
1340    ;       MOV     DX,BX           ;COPY COLUMN VALUE
1341    ;       SHR     DX,CL           ;BYTE PART OF COLUMN
1342    ;       ADD     DX,AX           ;FINAL BYTE ADDRESS RETURNED IN DX
1343    ;       MOV     CL,BL           ;LS BYTE OF COLUMN ADDRESS
1344    ;       AND     CL,07H          ;SNATCH BIT ADDRESS
1345    ;       MOV     BL,01H          ;CREATE A BIT
1346    ;       SHL     BL,CL           ;SHIFT IT INTO BIT POSITION
1347    ;       MOV     BH,BL           ;COPY TO MS BYTE TO
1348    ;       NOT     BH              ;CREATE COMPLEMENT FOR ANDING
1349    ;       MOV     CX,BX           ;RETURNED IN CX
1350    ;       POP     AX              ;RECOVER IN CASE NEEDED
1351    ;       RET
1352    ;
1353    ;*******************************************************************************
1354    ;       THIS ROUTINE IS USED TO READ A GRAPHICS PIXEL AT THE
1355    ;       SPECIFIED LOCATION.
1356    ;
1357    ;       INPUT - CX = COLUMN
1358    ;               DX = ROW
1359    ;       OUTPUT - AL = COLOR NUMBER FROM BIT MAP, NOT RGB VALUE
1360    ;
1361    ;       USED - A,B,C,D 3 BYTES OF STACK(IN MKCURS)
1362    ;
1363    ;*******************************************************************************
1364    ;
1365    ;RDPIXL PROC     NEAR
1366    ;       CALL    MKCURS          ;
1367    ;       MOV     DI,DX           ;
1368    ;       XOR     AL,AL           ;PREPARE AL TO RECEIVE DATA
1369    ;       MOV     BX,GRCSEG
1370    ;       MOV     ES,BX           ;POINT TO FIRST PLANE
1371    ;       CALL    READPXL         ;DO IT FOR BANK C
```

```
                        1372    ;       MOV     BX,CRBSEG
                        1373    ;       MOV     ES,BX           ;POINT TO NEXT PLANE
                        1374    ;       CALL    REAPXL          ;DO IT FOR BANK B
                        1375    ;       MOV     BX,GRASEG       ;
                        1376    ;       MOV     ES,BX           ;
                        1377    ;       CALL    REAPXL          ;DO IT FOR BANK A
                        1378    ;       JMP     CRTRET          ;...AND LEAVE
                        1379    ;REAPXL:
                        1380    ;       MOV     AH,ES:BYTE PTR [DI] ;GET THE BYTE
                        1381    ;       AND     AH,CL           ;MASK THE BIT
                        1382    ;       JZ      REAEND          ; TEST IT
                        1383    ;       OR      AL,01H
                        1384    ;       ROL     AL,1            ;INTO POSTION,NEED TO FILL LSB
                        1385    ;REAEND:
                        1386    ;       RET                     ;
                        1387    ;
                        1388    ;*****************************************************************************
                        1389    ;               SET TTY STATUS LINE.
                        1390    ;
                        1391    ;       INPUT - CH = 0 (always)
                        1392    ;               CL = Start line number of status line (0-24 decimal)
                        1393    ;
                        1394    ;                       NOTE:  CH must always be zero. This represents
                        1395    ;                              the first column of the status line.
                        1396    ;
                        1397    ;               CX = 0 = Full screen TTY.
                        1398    ;
                        1399    ;                       NOTE:  If an attempt is made to cause a status
                        1400    ;                              line which doesn't fall after the current
                        1401    ;                              line of the cursor the then no status line
                        1402    ;                              is implemented.
                        1403    ;
                        1404    ;       OUTPUT - None
                        1405    ;
                        1406    ;       USED - None
                        1407    ;
                        1408    ;*****************************************************************************
                        1409    ;
01EF'                   1410    SETTTY  PROC    NEAR                    ; SET TTY STATUS LINE BEGINNING
01EF'  A1 0005"         1411            MOV     AX,THSLIN               ; AX = THSLIN PHYSICAL LOCATION
01F2'  E8 027C          1412            CALL    PTRADJ                  ; GET IT'S LOGICAL LOCATION
01F5'  97               1413            XCHG    DI,AX                   ; PUT IT IN DI
01F6'  8B D1            1414            MOV     DX,CX                   ; DX = STATUS LINE COL,ROW
01F8'  30 F6            1415            XOR     DH,DH                   ; MAKE SURE COL = 0
01FA'  30 E4            1416            XOR     AH,AH                   ; CLEAR AH
01FC'  B0 50            1417            MOV     AL,80                   ;
01FE'  F6 E2            1418            MUL     DL                      ; AX = STAT. ROW OFFSET FROM 0
0200'  03 06 0006" R    1419            ADD     AX,DISBEG               ; SI = STAT ABS LOCATION FROM DISBEG
0204'  39 F8            1420            CMP     AX,DI                   ; Q: STATUS LINE (= CURRENT LINE ?
0206'  76 07            1421            JBE     STTTY1                  ;    Y: THEN SET NO STATUS LINE.
0208'  89 0E 0009"      1422            MOV     STATLN,CX               ; SAVE STATUS LINE BEGINNING
020C'  E9 FE57          1423            JMP     CRTRET                  ; ...AND RETURN
```

```
020F'                          1424    STTTY1:
020F'  C7 06 0009" 0000        1425            MOV     STATLN,0                ; SET NO STATUS LINE
0215'  E9 FE4E                 1426            JMP     CRTRET                  ; ...LEAVE
                               1427    ;
                               1428    ;*****************************************************************************
                               1429    ;               ASCII TELETYPE WRITE ROUTINE
                               1430    ;
                               1431    ;       INPUT - AL = Character to write
                               1432    ;
                               1433    ;       OUTPUT - None
                               1434    ;
                               1435    ;       USED -
                               1436    ;*****************************************************************************
                               1437    ;
0218'                          1438    WRTTTY  PROC    NEAR                    ; ASCII TELETYPE WRITE
                               1439            ASSUME  ES:CRTDAT
0218'  3C 07                   1440            CMP     AL,BEL                  ; IS CHAR A BELL CHARACTER ?
021A'  74 0F                   1441            JE      BEPBEL                  ; YES, Then go beep the bell
021C'  3C 08                   1442            CMP     AL,BS                   ; IS CHAR A BACKSPACE
021E'  74 11                   1443            JE      BAKSP                   ; YES, DO SHORT JUMP
0220'  3C 0A                   1444            CMP     AL,LF                   ; IS IT A LINE FEED
0222'  74 65                   1445            JE      LINFED                  ; YES, DO SHORT JUMP
0224'  3C 0D                   1446            CMP     AL,CR                   ; IS CHAR A CARRIAGE RETURN
0226'  74 4A                   1447            JE      CARET                   ; YES, DO SHORT JUMP
0228'  E9 0129                 1448            JMP     CHRWRT                  ; NO, THEN WRITE CHAR @ CURSOR POS.
022B'                          1449    BEPBEL:
022B'  E8 0001"                1450            CALL    BEEP                    ; GO BEEP THE BELL
022E'  E9 FE35                 1451            JMP     CRTRET                  ; ...AND LEAVE
                               1452    ;
                               1453    ;*****************************************************************************
                               1454    ;                       PERFORM A BACKSPACE
                               1455    ;*****************************************************************************
                               1456    ;
0231'                          1457    BAKSP   PROC    NEAR
                               1458            ASSUME  ES:CRTDAT
0231'  A1 0003"                1459            MOV     AX,CURPOS               ; GET CURRENT CURSOR POSITION
0234'  8B 1E 0006"             1460            MOV     BX,DISBEG               ; GET BEGINNING OF DISPLAY POINTER
0238'  39 D8                   1461            CMP     AX,BX                   ; SEE IF THE SAME
023A'  74 20                   1462            JE      NOBKSP                  ; DO NOTHING (RETURN)
023C'  8B 1E 0005"             1463            MOV     BX,THSLIN               ; GET THISLIN POINTER
0240'  39 D8                   1464            CMP     AX,BX                   ; SEE IF AT BEGINNING OF LINE
0242'  74 1B                   1465            JE      BAKLIN                  ; WRAP BACK TO PREVIOUS LINE
0244'                          1466    BKSP1:
0244'  48                      1467            DEC     AX                      ; BACK-UP ONE CHAR POSITION
0245'  25 07FF                 1468            AND     AX,7FFH                 ; MAGIC NUMBER !!
0248'  A3 0003"                1469            MOV     CURPOS,AX               ; SAVE NEW CURSOR POSITION
024B'  B2 0E                   1470            MOV     DL,CURPSH               ; DL = CURS. POS. REQ. HIGH
                               1471    ;                                       ; AH = CURS. POS. ADD. HIGH
024D'  E8 0221                 1472            CALL    PTRADJ                  ; GET CURSOR LOGICAL POSITION
0250'  E8 0236                 1473            CALL    PTRUP                   ; DO DOUBLE WRITE TO CRTC
0253'  8B 1E 0008"             1474            MOV     BX,CRTCNT               ; GET NUMBER OF CHRS ON LINE
0257'  4B                      1475            DEC     BX                      ; NOW THERE IS ONE LESS
```

```
0258'  89 1E 0008"          1476            MOV     CRTCNT,BX               ; SAVE NEW COUNT
                            1477    ;
025C'  E9 FE07              1478    NOBKSP: JMP     CRTRET                  ; ALL DONE
                            1479    ;
025F'                       1480    BAKLIN:
025F'  83 EB 50             1481            SUB     BX,80                   ; MOVE THSLIN BACK ONE LINE
0262'  81 E3 07FF           1482            AND     BX,7FFH                 ; MAGIC NUMBER !!
0266'  89 1E 0005"          1483            MOV     THSLIN,BX               ; SAVE NEW THSLIN POINTER
026A'  C7 06 0008" 0050     1484            MOV     CRTCNT,80               ; SET CHAR COUNT = TO A FULL LINE
0270'  EB D2                1485            JMP     SHORT BKSP1             ; GO BACK AND FINISH
                            1486    ;
                            1487    ;*****************************************************************************
                            1488    ;                       PERFORM A CARRIAGE RETURN
                            1489    ;*****************************************************************************
                            1490    ;
0272'                       1491    CARET   PROC    NEAR
                            1492            ASSUME  ES:CRTDAT
0272'  B2 0E                1493            MOV     DL,CURPSH               ; GET CURSOR POS. ADD. HIGH
0274'  A1 0005"             1494            MOV     AX,THSLIN               ; GET CURRENT LINE ADDRESS
0277'  A3 0003"             1495            MOV     CURPOS,AX               ; SAVE NEW CUR. POS. = THSLIN
027A'  E8 01F4              1496            CALL    PTRADJ                  ; GET CURSOR LOGICAL POSITION
027D'  E8 0209              1497            CALL    PTRUP                   ; DO DOUBLE WRITE TO CRTC
0280'  C7 06 0008" 0000     1498            MOV     CRTCNT,0                ; SET CHARACTER COUNT TO ZERO
                            1499    ;                                       ; IMPLIES CURSOR AT POS. 0 ON LINE
0286'  E9 FDDD              1500            JMP     CRTRET                  ; ALL DONE
                            1501    ;
                            1502    ;*****************************************************************************
                            1503    ;                       PERFORM A LINE FEED
                            1504    ;*****************************************************************************
                            1505    ;
0289'                       1506    LINFED  PROC    NEAR
                            1507            ASSUME  ES:CRTDAT
0289'  A1 0003"             1508            MOV     AX,CURPOS               ; GET CURRENT CURPOS
028C'  05 0050              1509            ADD     AX,80                   ; ADJUST CURSUR FORWARD 80 CHARS.
028F'  25 07FF              1510            AND     AX,7FFH                 ; MAGIC NUMBER !!
0292'  A3 0003"             1511            MOV     CURPOS,AX               ; SAVE NEW CURSOR
0295'  A1 0005"             1512            MOV     AX,THSLIN               ; GET CURRENT LINE ADDRESS
0298'  05 0050              1513            ADD     AX,80                   ; ADJUST THSLINE FORWARD ONE LINE
029B'  25 07FF              1514            AND     AX,7FFH                 ; MAGIC NUMBER
029E'  A3 0005"             1515            MOV     THSLIN,AX               ; SAVE NEW LINE ADDRESS
02A1'  E8 01CD              1516            CALL    PTRADJ                  ; GET LINE LOGICAL LOCATION
02A4'  97                   1517            XCHG    DI,AX                   ; PUT IT IN DI
02A5'  83 3E 0009" 00       1518            CMP     STATLN,0                ; Q: FULL SCREEN TTY ?
02AA'  75 5B                1519            JNE     SPECLF                  ;   N: THEN DO SPECIAL LINFED
02AC'  A1 0007"             1520            MOV     AX,DISEND               ; GET END OF DISPLAY POINTER
02AF'  E8 018F              1521            CALL    PTRADJ                  ; GET DISPLAY END LOGICAL LOCATION
02B2'  39 C7                1522            CMP     DI,AX                   ; SEE IF WENT PAST END OF DISPLAY
02B4'  77 0E                1523            JA      ADJBEG                  ; GO CAUSE A SCROLL OF ONE LINE
02B6'  A1 0003"             1524    LF1:    MOV     AX,CURPOS               ; GET NEW CURSOR PHYSICAL POSITION
02B9'  B2 0E                1525            MOV     DL,CURPSH               ; DL = CUR. POS. ADDRESS HIGH REGISTER
02BB'  E8 0183              1526            CALL    PTRADJ                  ; GET CURSOR LOGICAL POSITION
02BE'  E8 01C8              1527            CALL    PTRUP                   ; GO UPDATE SCREEN POINTERS
```

```
                        1528    ;
02C1' E9 FDA2           1529            JMP     CRTRET                  ; ALL DONE
                        1530    ;
02C4'                   1531    ADJBEQ:
02C4' B9 0050           1532            MOV     CX,80                   ; SET UP CX FOR REP
02C7' 81 E7 07FF        1533            AND     DI,7FFH                 ; MAGIC NUMBER !!
02CB' E8 01C8           1534            CALL    REPIT                   ; CLEAR ONE LINE W/ DEFAULT ATTRIBUTES
02CE' A1 0006"          1535            MOV     AX,DISBEQ               ; GET BEGINNING OF DISPLAY
02D1' 05 0050           1536            ADD     AX,80                   ; MOVE FORWARD 80 CHARS
02D4' 25 07FF           1537            AND     AX,7FFH                 ; MAGIC NUMBER !!
02D7' A3 0006"          1538            MOV     DISBEG,AX               ; SAVE NEW DISPLAY BEGINNING
02DA' A1 0007"          1539            MOV     AX,DISEND               ; GET END OF DISPLAY POINTER
02DD' 05 0050           1540            ADD     AX,80                   ; MOVE FORWARD 80 CHARS
02E0' 25 07FF           1541            AND     AX,7FFH                 ; MAGIC NUMBER !!
02E3' A3 0007"          1542            MOV     DISEND,AX               ; SAVE NEW END OF DISPLAY
02E6' B2 0C             1543            MOV     DL,DISSTH               ; DL=DISPLAY START ADDRESS REQ. HIGH
02E8' A1 0006"          1544            MOV     AX,DISBEQ               ; AX=DISPLAY BEGIN ADDRESS
02EB'                   1545    BLKTS1:
02EB' 26F6 06 1811 20   1546            TEST    ES:BYTE PTR CRTSTA,CRTVBL ; Q: VERTICAL NON-BLANK ??
02F1' 75 F8             1547            JNZ     BLKTS1                  ;   N: WAIT FOR NON-BLANK
                        1548                                            ;
                        1549                                            ; NOW WAIT ONE FULL NON-BLANK PERIOD
                        1550                                            ; INORDER TO CATCH LEADING EDGE OF
                        1551                                            ; BLANKING PERIOD
02F3'                   1552    BLKTS2:
02F3' 26F6 06 1811 20   1553            TEST    ES:BYTE PTR CRTSTA,CRTVBL ; Q: VERTICAL BLANK ??
02F9' 74 F8             1554            JZ      BLKTS2                  ;   N: WAIT UNTIL BLANK
                        1555                                            ;   Y: THEN QUICKLY GO AND
02FB' E8 018B           1556            CALL    PTRUP                   ; ...UPDATE BEGIN OF DISPLAY
02FE' F6 06 000A" 02    1557            TEST    RETFLG,02H              ; Q: DOING SPECIAL SCROLL ?
0303' 75 22             1558            JNZ     SPCLF1                  ;   Y: THEN MOVE STATUS LINE DOWN
0305' EB AF             1559            JMP     SHORT LF1               ;   N: THEN GO BACK
                        1560    ;
0307'                   1561    SPECLF:
0307' 8B 16 0009"       1562            MOV     DX,STATLN               ; DX = STATUS LINE COL,ROW
030B' 30 F6             1563            XOR     DH,DH                   ; MAKE SURE COL = 0
030D' 30 E4             1564            XOR     AH,AH                   ; CLEAR AH
030F' B0 50             1565            MOV     AL,80                   ;
0311' F6 E2             1566            MUL     DL                      ; AX = STAT. ROW OFFSET FROM 0
0313' 03 06 0006"       1567            ADD     AX,DISBEQ               ; SI = STAT ABS LOCATION FROM DISBEQ
0317' 39 C7             1568            CMP     DI,AX                   ; Q: CURRENT LINE ( STATUS LINE ?
0319' 72 9B             1569            JB      LF1                     ;   Y: THEN UPDATE CURSOR POINTER
031B'                   1570    SCRLLF:                                 ;   N: THEN SCROLL TTY REGION
031B' 80 0E 000A" 02    1571            OR      RETFLG,02H              ; SET CONDITIONAL RETURN FLAG
0320' 8B 3E 0007"       1572            MOV     DI,DISEND               ; GET END OF DISPLAY
0324' 47                1573            INC     DI                      ; START WRITING BLANKS @ DISEND + 1
0325' EB 9D             1574            JMP     SHORT ADJBEG            ; GO SCROLL THE SCREEN AND COME BACK
0327'                   1575    SPCLF1:
0327' 8B 16 0009"       1576            MOV     DX,STATLN               ;
032B' 4A                1577            DEC     DX                      ; DX = SOURCE COL,ROW = STATLN - 1
032C' 8B 1E 0009"       1578            MOV     BX,STATLN               ; BX = DEST. COL,ROW = STATLN
0330' A1 0009"          1579            MOV     AX,STATLN               ;
```

MAIN CRTDSR.SRC

```
0333'  B1 19                1580            MOV     CL,25                   ;
0335'  28 C1                1581            SUB     CL,AL                   ; CL = 25 - STATLN = STAT. BLOCK SIZE
0337'  B5 50                1582            MOV     CH,80                   ; CH = COLUMN COUNT FOR SCROLL
0339'  E8 0165              1583            CALL    SAVATT                  ; (*** V1.22 ***) Save attribute latch
                            1584                                            ; AX = non-zero at this point always
033C'  E8 FD84              1585          ;CALL     SCROLL                  ; GO DO THE SCROLL OF THE TTY REGION
                            1586                                            ; without blanking
033F'  E8 0171              1587            CALL    RSTATT                  ; (*** V1.22 ***) Reset attribute latch
                            1588                                            ; (*** V1.22 ***)
0342'  8B 3E 0005"          1589            MOV     DI,THSLIN               ; DI = points to this line for blanking
0346'  B9 0050              1590            MOV     CX,80                   ; CX = length of one line
0349'  E8 014A              1591            CALL    REPIT                   ; Blank the line above the status line
                            1592                                            ;
034C'  80 26 000A" FD       1593            AND     RETFLG,NOT 02H          ; RESET THE RETURN FLAG
0351'  E9 FF62              1594            JMP     SHORT LF1               ; ...AND GO UPDATE CURSOR
                            1595    ;
                            1596    ;*****************************************************************************
                            1597    ;               WRITE SPECIFIED CHAR AT CURSOR AND ADVANCE CURSOR
                            1598    ;*****************************************************************************
                            1599    ;
0354'                       1600    CHRWRT  PROC    NEAR
                            1601            ASSUME  ES:CRTDAT
0354'  8B 36 0003"         1602            MOV     SI,CURPOS               ; GET CURRENT CURSOR POSITION
0358'  8B 1E 0008"         1603            MOV     BX,CRTCNT               ; GET CURRENT CHARS ON LINE
035C'  83 FB 50             1604            CMP     BX,80                   ; SEE IF LINE IS FULL
035F'  74 1F                1605            JE      NEWLIN                  ; YES, THEN ADVANCE FORWARD ONE LINE
0361'                       1606    CWRT1:
0361'  2688 04              1607            MOV     ES:MEMBEG[SI],AL        ; WRITE CHAR. AT CUR. POS.
0364'  46                   1608            INC     SI                      ; MOVE CURSOR OVER ONE
0365'  43                   1609            INC     BX                      ; ONE MORE CHAR. ON LINE
0366'  81 E6 07FF           1610            AND     SI,7FFH                 ; MAGIC NUMBER !!
036A'  89 36 0003"          1611            MOV     CURPOS,SI               ; SAVE NEW CURSOR POSITION
036E'  89 1E 0008"          1612            MOV     CRTCNT,BX               ; SAVE NEW CHAR. COUNT
0372'  50                   1613            PUSH    AX                      ; SAVE CHAR AND ATTRIBUTE
0373'  96                   1614            XCHG    AX,SI                   ; AX = CURS. POS. ADD.
0374'  B2 0E                1615            MOV     DL,CURPSH               ; DL = CURS. POS. REG. HIGH
0376'  E8 00F8              1616            CALL    PTRADJ                  ; GET CURSOR LOGICAL POSITION
0379'  E8 010D              1617            CALL    PTRUP                   ; DO DOUBLE WRITE TO CRTC
037C'  58                   1618            POP     AX                      ; RESTORE CHAR
037D'  E9 FCE6              1619            JMP     CRTRET                  ; ALL DONE
                            1620    ;
0380'  50                   1621    NEWLIN: PUSH    AX                      ; SAVE CHAR. AND ATTRIBUTE
0381'  80 0E 000A" 01       1622            OR      RETFLG,01H              ; SET CONDITIONAL RETURN FLAG
0386'  E8 FEE9              1623            CALL    CARET                   ; DO A CARRIAGE RETURN AND A
0389'  E8 FEFD              1624            CALL    LINFED                  ; LINE FEED WITH SCROLL IF NECCESARY
038C'  80 26 000A" FE       1625            AND     RETFLG,NOT 01H          ; RESET RETURN FLAG
0391'  58                   1626            POP     AX                      ; RESTORE CHAR.
0392'  8B 36 0003"          1627            MOV     SI,CURPOS               ; GET NEW CURSOR POSITION
0396'  8B 1E 0008"          1628            MOV     BX,CRTCNT               ; GET NEW CHAR. COUNT
039A'  EB C5                1629            JMP     SHORT CWRT1             ; CONTINUE
                            1630    ;*****************************************************************************
                            1631    ;       THIS ROUTINE IS USED TO RETURN THE STATE (or MODE) OF THE
```

```
                        1632    ;       CRT TO THE CALLER  (NOTE: this is not applicable to PEGASUS
                        1633    ;       but is provided for compatibility with the IBM CRT calling
                        1634    ;       conventions), IT DOES NOTHING AND SIMPLY RETURNS.
                        1635    ;
                        1636    ;       INPUT - AH = 15
                        1637    ;
                        1638    ;       OUTPUT - (To be determined)
                        1639    ;
                        1640    ;       USED -
                        1641    ;
                        1642    ;*******************************************************************************
                        1643    ;
                        1644    ;VIDSTA PROC    NEAR
                        1645    ;       JMP     CRTRET
                        1646    ;*******************************************************************************
                        1647    ;       THIS ROUTINE IS USED TO WRITE A GIVEN BLOCK OF CHARACTERS TO
                        1648    ;       THE SCREEN STARTING AT THE CURSOR USING THE SPECIFIED ATTRIBUTE.
                        1649    ;       ALL CONTROL CHARACTERS (i.e. CR,LF,BEL etc.) ARE IGNORED AS SUCH
                        1650    ;       AND PRINTED ON THE SCREEN AS ASCII CHARACTERS.
                        1651    ;
                        1652    ;
                        1653    ;       INPUT - AL = Attribute of characters to write
                        1654    ;               DX = Segment location of character block
                        1655    ;               BX = Starting location of character block
                        1656    ;               CX = Length of character block (not to exceed 2000)
                        1657    ;
                        1658    ;               NOTE:  This routine does not increment the cursor
                        1659    ;                      location. The caller must be aware of the
                        1660    ;                      cursor location before using this routine
                        1661    ;                      and make sure that the block will "fit" on
                        1662    ;                      the screen or he will lose what want "fit". *
                        1663    ;
                        1664    ;       OUTPUT - None
                        1665    ;
                        1666    ;       USED - None
                        1667    ;
                        1668    ;       * CX (block length) must be ( or = 2000-((cur. row - 1 * 80)+cur. col.)
                        1669    ;         Attempting to write characters beyond the end of screen will cause
                        1670    ;         them to be be lost (not written).
                        1671    ;
                        1672    ;*******************************************************************************
                        1673    ;
039C'                   1674    WATBLK  PROC    NEAR
                        1675            ASSUME  ES:CRTDAT
039C'  26A2 1800        1676            MOV     ES:BYTE PTR ATTLAT,AL   ; WRITE THE ATTRIBUTE LATCH
                        1677                                            ; ... AND FALL THROUGH TO DO
                        1678                                            ; ... THE BLOCK WRITE
                        1679    ;*******************************************************************************
                        1680    ;       THIS ROUTINE IS USED TO WRITE A GIVEN BLOCK OF CHARACTERS TO
                        1681    ;       THE SCREEN STARTING AT THE CURSOR ALL CONTROL CHARACTERS
                        1682    ;       (i.e. CR,LF,BEL etc.) ARE IGNORED AS SUCH AND PRINTED ON THE
                        1683    ;       SCREEN AS ASCII CHARACTERS.
```

```
                        1684    ;
                        1685    ;
                        1686    ;       INPUT - DX = Segment location of character block
                        1687    ;               BX = Starting location of character block
                        1688    ;               CX = Length of character block (not to exceed 2000)
                        1689    ;
                        1690    ;               NOTE:  This routine does not increment the cursor
                        1691    ;                      location. The caller must be aware of the
                        1692    ;                      cursor location before using this routine
                        1693    ;                      and make sure that the block will "fit" on
                        1694    ;                      the screen or he will lose what want "fit". *
                        1695    ;                      With this type of write the characters will
                        1696    ;                      take on the attributes of the last attribute
                        1697    ;                      change.
                        1698    ;
                        1699    ;       OUTPUT - None
                        1700    ;
                        1701    ;       USED - None
                        1702    ;
                        1703    ;       * CX (block length) must be ( or = 2000-((cur. row - 1 * 80)+cur. col.)
                        1704    ;         Attempting to write characters beyond the end of screen will cause
                        1705    ;         them to be be lost (not written).
                        1706    ;
                        1707    ;*******************************************************************************
                        1708    ;
03A0'                   1709    WBLOCK  PROC    NEAR
                        1710            ASSUME  ES:CRTDAT
03A0'  83 F9 00         1711            CMP     CX,0                    ; Q: ANY CHARACTERS TO WRITE ?
03A3'  76 1B            1712            JBE     TOOFAR                  ;   N: THEN DON'T WRITE ANY
03A5'  53               1713            PUSH    BX                      ; SAVE BX
03A6'  A1 0003"         1714            MOV     AX,CURPOS               ; GET CURSOR PHYSICAL POSITION
03A9'  E8 00C5          1715            CALL    PTRADJ                  ; GET IT'S LOGICAL POSITION
03AC'  97               1716            XCHG    DI,AX                   ; PUT IT IN DI
03AD'  A1 0007"         1717            MOV     AX,DISEND               ; GET DISPLAY END PHYSICAL LOCATION
03B0'  E8 008E          1718            CALL    PTRADJ                  ; GET IT'S LOGICAL LOCATION
03B3'  5B               1719            POP     BX                      ; RESTORE BX
03B4'  8B F3            1720            MOV     SI,BX                   ; SI = BLOCK STARTING ADDRESS
03B6'  1E               1721            PUSH    DS                      ; SAVE MY DS
03B7'  8E DA            1722            MOV     DS,DX                   ; DS = BLOCK SEGMENT
03B9'                   1723    BLKLOP:
03B9'  A4               1724            MOVS    BYTE PTR [DI],BYTE PTR[SI] ;...BLOCK LENGTH TIMES
03BA'  39 C7            1725            CMP     DI,AX                   ; Q: OFF THE END OF SCREEN ?
03BC'  77 02            1726            JA      TOOFAR                  ;   Y: THEN QUIT WRITING
03BE'  E2 F9            1727            LOOP    BLKLOP                  ;   N: KEEP GOING
03C0'                   1728    TOOFAR:
03C0'  1F               1729            POP     DS                      ; RESTORE MY DS
03C1'  E9 FCA2          1730            JMP     CRTRET                  ; ...AND RETURN
                        1731    ;*******************************************************************************
                        1732    ;               THIS ROUTINE CHANGES THE ENTIRE SCREEN ATTRIBUTES
                        1733    ;                       RIGHT BEFORE YOUR VERY EYES
                        1734    ;
                        1735    ;       INPUT - AL = ATRIBUTE TO USE
```

```
                                1736  ;
                                1737  ;       OUTPUT - None
                                1738  ;
                                1739  ;       USED - None
                                1740  ;
                                1741  ;*****************************************************************************
                                1742  ;
03C4'                           1743  CHSCAT  PROC    NEAR
                                1744          ASSUME  ES:CRTDAT,DS:ROMDAT
03C4'  8B 36 0006"              1745          MOV     SI,DISBEG               ; GET DISBEG IN SI
03C8'  B9 07D0                  1746          MOV     CX,2000                 ; SET UP CX FOR LOOP
03CB'                           1747  AGAIN:
03CB'  268A 24                  1748          MOV     AH,ES:MEMBEG[SI]        ; READ A CHARACTER
03CE'  26A2 1800                1749          MOV     BYTE PTR ES:ATTLAT,AL   ; WRITE THE ATTRIBUTE LATCH
03D2'  2688 24                  1750          MOV     ES:MEMBEG[SI],AH        ; WRITE THE CHARACTER BACK TO MEMORY
03D5'  46                       1751          INC     SI                      ; NEXT CHARACTER IN MEMORY
03D6'  E2 F3                    1752          LOOP    AGAIN                   ; LOOP THROUGH ENTIRE SCREEN
03D8'  E9 FC8B                  1753          JMP     CRTRET                  ; ALL DONE
                                1754  ;*****************************************************************************
                                1755  ;               CLEAR THE CRT SCREEN MEMORY AND HOME CURSOR
                                1756  ;
                                1757  ;       INPUT - None
                                1758  ;
                                1759  ;               NOTE:  This routine will also clear the status line
                                1760  ;                      but will not clear the status line setting.
                                1761  ;                      If the user wishes to keep what is on the
                                1762  ;                      status line he must rewrite it after clearing
                                1763  ;                      the screen.
                                1764  ;
                                1765  ;       OUTPUT - None
                                1766  ;
                                1767  ;       USED - AX,CX,DL,DI
                                1768  ;
                                1769  ;*****************************************************************************
                                1770  ;
03D8'                           1771  CLRTXT  PROC    NEAR                    ; CLEAR SCREEN PROCEDURE
                                1772          ASSUME  ES:CRTDAT               ;
03D8'  80 0E 000A" 02           1773          OR      RETFLG,02H              ; Set conditional return flag
03E0'  31 C0                    1774          XOR     AX,AX                   ; Clear AX
03E2'  E8 0002"                 1775          CALL    INILST                  ; Init local storage
03E5'  C6 06 000A" 00           1776          MOV     RETFLG,00H              ;
03EA'  B9 0800                  1777          MOV     CX,2048                 ; SET UP CX FOR REP
03ED'  31 FF                    1778          XOR     DI,DI                   ; CLEAR DI FOR REPIT
03EF'  E8 00A4                  1779          CALL    REPIT                   ; Go clear the screen using HIGHAT
03F2'  A1 0006"                 1780          MOV     AX,DISBEG               ; GET BEGIN OF DISPLAY POINTER
03F5'  B2 0C                    1781          MOV     DL,DISSTH               ; GET DISPLAY START ADD. REG HIGH
03F7'  E8 008F                  1782          CALL    PTRUP                   ; GO UPDATE DISPLAY BEGIN
03FA'  B2 0E                    1783          MOV     DL,CURPSH               ; GET CUR. POS. ADDR. REG. HIGH
03FC'  A1 0003"                 1784          MOV     AX,CURPOS               ; GET CURSOR POSITION
03FF'  E8 0087                  1785          CALL    PTRUP                   ; GO HOME THE CURSOR
0402'  E9 FC61                  1786          JMP     CRTRET                  ; ALL DONE
                                1787  ;*****************************************************************************
```

```
                        1788  ;       THIS ROUTINE IS USED TO CLEAR ALL THREE GRAPHICS BANKS.
                        1789  ;
                        1790  ;       INPUT - None
                        1791  ;
                        1792  ;       OUTPUT - None
                        1793  ;
                        1794  ;       USED -
                        1795  ;
                        1796  ;*******************************************************************************
                        1797  ;
0405'                   1798  CLRGRP  PROC    NEAR                    ; CLEAR GRAPHICS MEMORY
                        1799          ASSUME  ES:NOTHING
0405'  B8 C000          1800          MOV     AX,GRASEG               ; SET UP ES
0408'  8E C0            1801          MOV     ES,AX                   ; AS BANK A
040A'  E8 0013          1802          CALL    CLRIT                   ; CLEAR THIS BANK
040D'  B8 C800          1803          MOV     AX,GRBSEG               ; SET UP ES
0410'  8E C0            1804          MOV     ES,AX                   ; AS BANK B
0412'  E8 000B          1805          CALL    CLRIT                   ; CLEAR THIS BANK
0415'  B8 D000          1806          MOV     AX,GRCSEG               ; SET UP ES
0418'  8E C0            1807          MOV     ES,AX                   ; AS BANK C
041A'  E8 0003          1808          CALL    CLRIT                   ; CLEAR THIS BANK
041D'  E9 FC46          1809          JMP     CRTRET                  ; ...AND LEAVE
0420'                   1810  CLRIT:
0420'  31 C0            1811          XOR     AX,AX                   ; CLEAR AX
0422'  31 FF            1812          XOR     DI,DI                   ; CLEAR DI
0424'  B9 4000          1813          MOV     CX,SCRLEN               ; CX = BANK LENGTH FOR REP
0427'  F2               1814          REP
0428'  AB               1815          STOS    WORD PTR [DI]           ; DO IT
0429'  C3               1816          RET                             ; ...AND RETURN
                        1817  ;*******************************************************************************
                        1818  ;
                        1819  ;                       SET ATTRIBUTE LATCH
                        1820  ;                            AH = 16H
                        1821  ;
                        1822  ;       THIS ROUTINE IS USED TO WRITE THE ATTRIBUTE LATCH WITHOUT
                        1823  ;       HAVING TO WRITE A CHARACTER.
                        1824  ;
                        1825  ;       INPUT - BL = Attribute to use
                        1826  ;
                        1827  ;       OUTPUT - Latch set accordingly
                        1828  ;
                        1829  ;*******************************************************************************
                        1830  ;
042A'                   1831  SETATT  PROC    NEAR                    ; SET ATTRIBUTE LATCH
                        1832          ASSUME  ES:CRTDAT
042A'  2688 1E 1800     1833          MOV     ES:BYTE PTR ATTLAT,BL   ; WRITE THE ATTRIBUTE LATCH
042F'  E9 FC34          1834          JMP     CRTRET                  ; ...AND RETURN
                        1835  ;*******************************************************************************
                        1836  ;
                        1837  ;                 GET PHYSICAL DISPLAY BEGIN OFFSET
                        1838  ;                          AH = 17H
                        1839  ;
```

```
                        1840    ;       THIS ROUTINE IS USED TO RETURN THE OFFSET (FROM DE00H) OF
                        1841    ;       THE PHYSICAL BEGINNING OF DISPLAY.
                        1842    ;
                        1843    ;       INPUT - NONE
                        1844    ;
                        1845    ;       OUTPUT - DX = Display begin offset
                        1846    ;
                        1847    ;
                        1848    ;*******************************************************************************
                        1849    ;
0432'                   1850    GTDSBG  PROC    NEAR
0432'  8B 16 0006"      1851            MOV     DX,DISBEG               ; DX = display begin
0436'  E9 FC2D          1852            JMP     CRTRET                  ; ...adn return
                        1853    ;
                        1854    ;*******************************************************************************
                        1855    ;
                        1856    ;                               PRINT TTY STRING
                        1857    ;                                   AH = 18H
                        1858    ;
                        1859    ;       THIS ROUTINE IS USED TO PRINT A STRING OF CHARACTERS IN THE
                        1860    ;       USERS CS: (GOTTEN FROM THE STACK) WITH A TTY WRITE.
                        1861    ;
                        1862    ;       INPUT - BX = Address (offset) of the string
                        1863    ;                       Where:  [BX] byte 0 = length of string
                        1864    ;                               [BX] byte 1 = first char. of string
                        1865    ;
                        1866    ;       OUTPUT - NONE
                        1867    ;
                        1868    ;
                        1869    ;*******************************************************************************
                        1870    ;
0439'                   1871    PRTTTY  PROC    NEAR
0439'  1E               1872            PUSH    DS                      ; Save DS (ROMDAT pointer)
043A'  55               1873            PUSH    BP                      ; Save BP
043B'  8B EC            1874            MOV     BP,SP                   ; BP = SP
043D'  83 C5 12         1875            ADD     BP,18                   ; Point to user's CS on the stack
0440'  8E 5E 00         1876            MOV     DS,SS:WORD PTR [BP]     ; DS = user's CS
0443'  8B EB            1877            MOV     BP,BX                   ; BP = offset of string
0445'  30 ED            1878            XOR     CH,CH                   ; Clear CH
0447'  3E8A 4E 00       1879            MOV     CL,DS:BYTE PTR [BP]     ; CX = string length count
044B'  21 C9            1880            AND     CX,CX                   ; Q: Zero length string ?
044D'  74 1D            1881            JZ      PTYEND                  ;    Y: Then quit
044F'                   1882    PTTYLP:
044F'  45               1883            INC     BP                      ; Point to character in string
0450'  3E8A 46 00       1884            MOV     AL,DS:BYTE PTR [BP]     ; Get character
0454'  51               1885            PUSH    CX                      ; Save loop count
0455'  1E               1886            PUSH    DS                      ; Save this DS
0456'  2E8E 1E C180r    1887            MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; SET UP DS AS ROMDAT
045B'  80 0E 000A" 04   1888            OR      RETFLG,04H              ; Set conditional return flag
0460'  E8 FDB5          1889            CALL    WRTTTY                  ; Do TTY character write
0463'  80 26 000A" FB   1890            AND     RETFLG,NOT 04H          ; Reset flag in case of last char.
0468'  1F               1891            POP     DS                      ; Restore DS
```

```
0469'  59                1892            POP     CX                      ; Restore CX
046A'  E2 E3             1893            LOOP    PTTYLP                  ; Do entire string
                         1894    ;
046C'                    1895    PTYEND:
046C'  5D                1896            POP     BP                      ; Restore BP
046D'  1F                1897            POP     DS                      ; Restore DS (ROMDAT pointer)
046E'  E9 FBF5           1898            JMP     CRTRET                  ; ...and return
                         1899    ;
                         1900    ;*******************************************************************************
                         1901    ;               FOLLOWING ARE VARIOUS SUBROUTINES THAT THE CRT
                         1902    ;               FUNCTIONS USE AT DIFFERENT TIMES.   I CALL THIS
                         1903    ;               SET OF ROUTINES THE "CRT UTILITIES".  NOT TO BE
                         1904    ;               CONFUSED WITH "WATER WORKS" AND "ELECTRIC COMPANY".
                         1905    ;*******************************************************************************
                         1906    ;
                         1907    ;*******************************************************************************
                         1908    ;       THIS SUBROUTINE FIGURES OUT ANY POINTER POSITION IN FRONT OF THE
                         1909    ;       BEGINNING OF THE DISPLAY.  THE POINTERS ARE STORED LOCALLY IN AN
                         1910    ;       ABSOLUTE ADDRESS FORM AND ARE SOMETIMES BEHIND THE DISBEG POINTER.
                         1911    ;       FOR THAT CASE, THIS CODE FIGURES OUT THE ABSOLUTE ADDRESS OF THE
                         1912    ;       POINTER IN FRONT OF DISBEG AND RETURNS IT IN REGISTER AX.
                         1913    ;
                         1914    ;               NOTE:  IF THE POINTER IS ALREADY IN FRONT OF DISBEG
                         1915    ;                      THEN NOTHING HAPPENS.
                         1916    ;
                         1917    ;       INPUT - AX = POINTER ABSOLUTE ADDRESS
                         1918    ;
                         1919    ;       OUTPUT - AX = POINTER ADDRESS IN FRONT OF DISBEG
                         1920    ;                BX = DISBEG POINTER
                         1921    ;
                         1922    ;       REGISTERS USED - AX,BX
                         1923    ;*******************************************************************************
                         1924    ;
0471'                    1925    PTRADJ  PROC    NEAR
                         1926            ASSUME  ES:CRTDAT
0471'  8B 1E 0006"      1927            MOV     BX,DISBEG               ; GET BEGINNING OF DISPLAY POINTER
0475'  39 D8             1928            CMP     AX,BX                   ; Q:  IS POINTER BEHIND DISPLAY BEGIN ?
0477'  7C 01             1929            JL      ADJPTR                  ;   Y:  GO FIGURE OUT OFFSET
0479'  C3                1930            RET                             ;   N:  DO NOTHING
                         1931    ;
047A'  05 0800           1932    ADJPTR: ADD     AX,2048                 ; PUT POINTER IN FRONT OF DISPLAY BEGIN
047D'  C3                1933            RET                             ; ALL DONE
                         1934    ;
                         1935    ;*******************************************************************************
                         1936    ;       THIS SUBROUTINE PERFORMS THE LOOPING MECHANISM REQUIRED TO WRITE
                         1937    ;       DATA TO THE CRT CONTROLLER.  IT WRITES THE REGISTER NUMBER PASSED
                         1938    ;       TO IT INTO THE CRT ADDRESS REGISTER, INCREMENTS TO THE CRT WRITE
                         1939    ;       REGISTER AND WRITES THE VALUE PASSED INTO IT AND DECREMENTS BACK
                         1940    ;       TO THE CRT ADDRESS REGISTER.
                         1941    ;
                         1942    ;       INPUT - DL = THE CRT REGISTER NUMBER TO PUT IN THE CRTADR
                         1943    ;               AH = THE VALUE TO PUT IN THE CRT WRITE REG.
```

```
                        1944    ;
                        1945    ;       OUTPUT - DL = UNCHANGED
                        1946    ;                AH = UNCHANGED
                        1947    ;
                        1948    ;       REGISTERS USED - DL,AH
                        1949    ;
                        1950    ;*****************************************************************************
                        1951    ;
047E'                   1952    CTRLWR  PROC    NEAR
                        1953            ASSUME  ES:CRTDAT
047E'  2688 16 1810     1954            MOV     BYTE PTR ES:CRTADR,DL   ;;; PUT CRT REG. # INTO CRTADR
0483'  2688 26 1812     1955            MOV     BYTE PTR ES:WRTREG,AH   ;;; PUT VALUE IN CRT WRITE REG.
0488'  C3               1956            RET                             ;;; RETURN
                        1957    ;
                        1958    ;*****************************************************************************
                        1959    ;       THIS SUBROUTINE PERFORMS THE DOUBLE CALL TO THE CTRLWR SUBROUTINE.
                        1960    ;       THIS DOUBLE CALL IS REQUIRED TO UPDATE CRT SCREEN POINTERS THAT
                        1961    ;       ARE ACCESSED VIA TWO CRT ADDRESS REGISTERS (HIGH AND LOW).
                        1962    ;
                        1963    ;       INPUT - DL = THE CRT REGISTER (HIGH) TO WRITE INTO THE CRTADR.
                        1964    ;               * NOTE: LOW REGISTER IS ASSUMED TO BE NEXT BY INCREMENTING DL
                        1965    ;                 AX = ADDRESS OF SCREEN POINTER TO BE CHANGED
                        1966    ;
                        1967    ;       OUTPUT - DL = POINTS TO CRT LOW REGISTER
                        1968    ;                AH = AL = LOW ORDER ADDRESS OF SCREEN POINTER IN QUESTION
                        1969    ;
                        1970    ;       REGISTERS USED - DL,AX
                        1971    ;
                        1972    ;       SUBROUTINES CALLED : CTRLWR
                        1973    ;*****************************************************************************
                        1974    ;
0489'                   1975    PTRUP   PROC    NEAR
                        1976            ASSUME  ES:CRTDAT
0489'  FA               1977            CLI                             ;   Shields up
048A'  E8 FFF1          1978            CALL    CTRLWR                  ;;; WRITE TO CRTC
048D'  8A E0            1979            MOV     AH,AL                   ;;; AH = START ADD. OF DISPLAY LOW
048F'  FE C2            1980            INC     DL                      ;;; DL = DISPLAY STRT ADD. REG LOW
0491'  E8 FFEA          1981            CALL    CTRLWR                  ;;; WRITE TO CRTC
0494'  FB               1982            STI                             ;;; Shields down
0495'  C3               1983            RET                             ;;; ALL DONE
                        1984    ;
                        1985    ;*****************************************************************************
                        1986    ;       THIS ROUTINE IS USED TO CLEAR "CX" AMOUNT OF SCREEN MEMORY
                        1987    ;       USING THE DEFAULT (i.e. HIGHAT) ATTRIBUTES ON THE BLANKS
                        1988    ;       THAT IT WRITES.
                        1989    ;
                        1990    ;       INPUT - DI = POINTS TO THE STARTING MEMORY ADDRESS OF THE REP
                        1991    ;               CX = NUMBER OF BYTES TO CLEAR
                        1992    ;               ES = CRTDAT
                        1993    ;               DS = ROMDAT
                        1994    ;
                        1995    ;       OUTPUT - DI = DI+CX
```

```
                        ;               CX = 0
                        ;               AH = SAVED ATTRIBUTE
                        ;               AL = SPACE
                        ;
                        ;       USED - AX,DI,CX
                        ;*******************************************************************************
                        ;
0496'                   REPIT   PROC    NEAR
0496'  E8 0008          ;               CALL    SAVATT                  ; SAVE THE ATTRIBUTE LATCH
0499'  B0 20            ;               MOV     AL,SPACE                ; AL = SPACE FOR CLEAR
049B'  F2               ;               REP
049C'  AA               ;               STOS    BYTE PTR [DI]           ; DO IT CX TIMES
049D'  E8 0013          ;               CALL    RSTATT                  ; RESTORE THE LATCH
04A0'  C3               ;               RET                             ; ...AND LEAVE
                        ;
                        ;*******************************************************************************
                        ;       THIS ROUTINE IS USED TO SAVE THE CURRENT ATTRIBUTE LATCH
                        ;       AND RESET IT TO THE DEFAULT SETTING FOR SUBSEQUENT WRITES.
                        ;
                        ;       INPUT - ES = CRTDAT
                        ;               DS = ROMDAT
                        ;
                        ;       OUTPUT - LATCH SET TO HIGHAT (0FH)
                        ;
                        ;       USED - NONE
                        ;*******************************************************************************
                        ;
04A1'                   SAVATT  PROC    NEAR
04A1'  50               ;               PUSH    AX
04A2'  268A 26 1800     ;               MOV     AH,ES:BYTE PTR ATTLAT           ; SAVE ATTRIBUTE IN AH
04A7'  88 26 000C"      ;               MOV     ATTSAV,AH               ;
04AB'  26C6 06 1800 0F  ;               MOV     ES:BYTE PTR ATTLAT,HIGHAT       ; SET LATCH TO DEFAULT
04B1'  58               ;               POP     AX                      ;
04B2'  C3               ;               RET                             ; ...AND LEAVE
                        ;
                        ;*******************************************************************************
                        ;       THIS ROUTINE IS USED TO RESTORE THE ATTRIBUTE LATCH TO IT'S
                        ;       LAST SETTING.
                        ;
                        ;       INPUT - ES = CRTDAT
                        ;
                        ;       OUTPUT - LATCH SET TO LAST SAVED SETTING
                        ;
                        ;       USED - NONE
                        ;*******************************************************************************
                        ;
04B3'                   RSTATT  PROC    NEAR
04B3'  50               ;               PUSH    AX
04B4'  8A 26 000C"      ;               MOV     AH,ATTSAV                       ; GET THE OLD LATCH SETTING
04B8'  2688 26 1800     ;               MOV     ES:BYTE PTR ATTLAT,AH           ; RESTORE THE LATCH
04BD'  58               ;               POP     AX                      ;
04BE'  C3               ;               RET                             ; ...AND LEAVE
```

```
                        2048    ;
                        2049            SUBTTL
                        2050            END
```

No errors detected

```
                                1  ;*****************************************************************************
                                2  ; TITLE    - CRTTBL (CRT INITIALIZATION TABLES)
                                3  ; ABSTRACT -    THIS MODULE CONTAINS THE INITIALIZATION TABLES FOR EACH OF THE
                                4  ;               VARIOUS MONITORS THAT MIGHT BE PRESENT WITH PEGASUS.  THE TYPE
                                5  ;               OF MONITOR IS DETERMINED BY READING A JUMPER AND THE CONTROLLER
                                6  ;               WILL BE SET UP WITH THE APPROPRIATE VALUES FROM THE APPROPRIATE
                                7  ;               TABLE CONTAINED HEREIN.
                                8  ;*****************************************************************************
                                9          NAME    CRTTBL (CRT INITIALIZATION TABLES)
                               10          SUBTTL
          =FFFF                12  DEBUG   EQU     0FFFFH
                               13  ;*****************************************************************************
                               14  ;                           PUBLIC DEFINITIONS
                               15  ;*****************************************************************************
                               16  ;
                               17          PUBLIC  STAMON
                               18          PUBLIC  ALTMON
                               19          PUBLIC  TSTPAT
                               20  ;*****************************************************************************
                               21  ;                           MODULE ENTRY POINT
                               22  ;*****************************************************************************
                               23  ;
                               24  ;       THIS IS THE 60 HZ CRT PROGRAMMING TABLE
                               25  ;
          =0000                26          SECT    ROMCOD
                               27          ASSUME  CS:ROMCOD
                               28  ;
0000'                          29  STAMON  LABEL   BYTE
0000'  67                      30          DB      104-1     ;; INITIALIZATION       H TOTAL CHAR
0001'  50                      31          DB      80        ;;                      H DISP  CHAR
0002'  54                      32          DB      84        ;;     VALUES           H SYNC POSITION
0003'  39                      33          DB      3*16+9    ;;                      VSYNC, HSYNC WIDTHS
0004'  18                      34          DB      25-1      ;;       FOR            V TOTAL ROWS
0005'  14                      35          DB      20        ;;                      V TOTAL LINE ADJUST
0006'  19                      36          DB      25        ;;       THE            V DISP ROWS
0007'  19                      37          DB      25        ;;                      V SYNC POSITION
0008'  00                      38          DB      00H       ;;      6545            MODE CONTROL
0009'  08                      39          DB      12-1      ;;                      SCAN LINES PER ROW
000A'  40                      40          DB      40H+0     ;;       CRT            CURSOR START
000B'  08                      41          DB      11        ;;                      CURSOR END
000C'  00                      42          DB      00H       ;;   CONTROLLER        DISPLAY START HIGH
000D'  00                      43          DB      00H       ;;                      DISPLAY START LOW
000E'  00                      44          DB      00H       ;;   REGISTERS         CURSOR HIGH
000F'  00                      45          DB      00H       ;;     0-15             CURSOR LOW
                               46  ;*****************************************************************************
                               47  ;
                               48  ;       THIS IS THE TABLE FOR 50HZ EUROPEAN OPERATION
                               49  ;
0010'                          50  ALTMON  LABEL   BYTE
0010'  67                      51          DB      104-1     ;; INITIALIZATION       H TOTAL CHAR
0011'  50                      52          DB      80        ;;                      H DISP  CHAR
0012'  54                      53          DB      84        ;;     VALUES           H SYNC POSITION
```

CRTTBL.SRC

```
0013' 39                54          DB      3*16+9  ;;                      VSYNC, HSYNC WIDTHS
0014' 19                55          DB      26-1    ;;      FOR             V TOTAL ROWS
0015' 14                56          DB      20      ;;                      V TOTAL LINE ADJUST
0016' 19                57          DB      25      ;;      THE             V DISP ROWS
0017' 19                58          DB      25      ;;                      V SYNC POSITION
0018' 00                59          DB      00H     ;;      6545            MODE CONTROL
0019' 0D                60          DB      14-1    ;;                      SCAN LINES PER ROW
001A' 40                61          DB      40H+0   ;;      CRT             CURSOR START
001B' 0B                62          DB      11      ;;                      CURSOR END
001C' 00                63          DB      00H     ;;  CONTROLLER         DISPLAY START HIGH
001D' 00                64          DB      00H     ;;                      DISPLAY START LOW
001E' 00                65          DB      00H     ;;  REGISTERS           CURSOR HIGH
001F' 00                66          DB      00H     ;;    0-15              CURSOR LOW
                        67  ;
                        68  ;*************************************************************************
                        69  ;       THIS IS A TABLE OF TEST PATTERNS FOR THE VARIOS CRTC REGISTERS
                        70  ;       WHICH IS USED DURING THE POWERUP DIAGNOSTICS.
                        71  ;
0020'                   72  TSTPAT  LABEL   BYTE
0020' F0                73          DB      0F0H
0021' CC                74          DB      0CCH
0022' AA                75          DB      0AAH
0023' 55                76          DB      055H
0024' F0                77          DB      0F0H
0025' CC                78          DB      0CCH
0026' AA                79          DB      0AAH
                        80  ;
                        81          END
```

No errors detected

```
                        1   ;*******************************************************************************
                        2   ; TITLE    - CRTTST  (CRT DIAGNOSTICS)
                        3   ; ABSTRACT - THIS MODULE IS RESPONSIBLE FOR THE CRT POWERUP DIAGNOSTIC
                        4   ;            TESTS.  IF THE TEST PASSES THEN THE LED'S ARE DECREMENTED
                        5   ;            FROM THEIR CURRENT VALUE BY ONE AND POWERUP CONTINUES.  IF
                        6   ;            THE TEST FAILS AN ERROR CODE IS OUTPUT TO THE PRINTER PORT
                        7   ;            AND THE POWERUP SEQUENCE AND TESTING STOPS.
                        8   ;*******************************************************************************
                        9           NAME    CRTTST (CRT DIAGNOSTICS)
                       10           SUBTTL
=FFFF                  12   DEBUG   EQU     0FFFFH
                       13   ;*******************************************************************************
                       14   ;                           PUBLIC DEFINITIONS
                       15   ;*******************************************************************************
                       16           PUBLIC  INICRT
                       17           PUBLIC  INILST
                       18           PUBLIC  CRTTST
                       19           PUBLIC  CRTINI
                       20   ;*******************************************************************************
                       21   ;                           EXTERNAL REFERENCES
                       22   ;*******************************************************************************
                       23   ;       EXTERNAL JUMPS AND CALLS
                       24   ;
                       25           EXTRN   CRTRET:NEAR                     ; (CRTDSR)
                       26           EXTRN   DSPDIE:NEAR                     ; (OUTPUT)
                       27           EXTRN   DECLED:NEAR                     ; (ROMUTL)
                       28           EXTRN   MSCTST:NEAR                     ; (MSCTST)
                       29           EXTRN   PRTRST:NEAR                     ; (TINTTS)
                       30   ;
                       31   ;*******************************************************************************
                       32   ;       EXTERNAL EQUATES
                       33   ;
                       34           EXTRN   ATMERR:ABS                      ; (TSTERR)
                       35           EXTRN   ATTERR:ABS                      ; (TSTERR)
                       36           EXTRN   CINTER:ABS                      ; (TSTERR)
                       37           EXTRN   CURERR:ABS                      ; (TSTERR)
                       38           EXTRN   CRAMER:ABS                      ; (TSTERR)
                       39           EXTRN   VIDERR:ABS                      ; (TSTERR)
                       40           EXTRN   VBLERR:ABS                      ; (TSTERR)
                       41           EXTRN   LED010:ABS                      ; (ROMERR)
                       42   ;
                       43   ;*******************************************************************************
                       44   ;       EXTERNAL TABLES
                       45   ;
=0000                  46           SECT    ROMCOD
                       47           ASSUME  CS:ROMCOD
                       48   ;
                       49           EXTRN   STAMON:BYTE
                       50           EXTRN   ALTMON:BYTE                     ; (CRTTBL)
                       51           EXTRN   TSTPAT:BYTE                     ; (CRTTBL)
                       52   ;
                       53   ;*******************************************************************************
```

```
                        ;       EXTERNAL STORAGE
                        ;
=0000                           SECT    ROMDAT
                        ;
                                EXTRN   CRTCNT:WORD                     ; (ROMDAT)
                                EXTRN   CRTERR:BYTE                     ; (ROMDAT)
                                EXTRN   CRTHLD:BYTE                     ; (ROMDAT)
                                EXTRN   CURPOS:WORD                     ; (ROMDAT)
                                EXTRN   CURTYP:WORD                     ; (ROMDAT)
                                EXTRN   DISBEG:WORD                     ; (ROMDAT)
                                EXTRN   DISEND:WORD                     ; (ROMDAT)
                                EXTRN   PRCO_M:BYTE                     ; (ROMDAT)
                                EXTRN   RETFLG:BYTE                     ; (ROMDAT)
                                EXTRN   THSLIN:WORD                     ; (ROMDAT)
                                EXTRN   STATLN:WORD                     ; (ROMDAT)
                        ;
```

```
                        71    ;*****************************************************************************
                        72    ;                       LOCAL CONSTANTS
                        73    ;*****************************************************************************
                        74    ;       INCLUDE         PEQ:ASCII.EQU
                        75    ;       INCLUDE         PEQ:PORTADDR.EQU
                        76    ;       INCLUDE         PEQ:LATCHES.EQU
                        77    ;       INCLUDE         PEQ:CRT.EQU
                        78    ;       INCLUDE         PEQ:VECTOR.EQU
                        79    ;
                       330    ;*****************************************************************************
                       331    ;                       LOCAL MACROS
                       332    ;*****************************************************************************
                       333
                       334
                       335            SUBTTL  MAIN
                       336    ;*****************************************************************************
                       337    ;                       MODULE ENTRY POINT
                       338    ;*****************************************************************************
                       339    ;       THIS IS THE TEMPORARY INITIALIZATION DONE AT FIRST SIGN OF
                       340    ;       POWER ON TO THE UNIT.  NO STACK OR RAM IS AVAILABLE AT THIS
                       341    ;       TIME FOR USE SO THIS IS QUICK AND DIRTY !!!
                       342    ;*****************************************************************************
                       343    ;
=0000                  344            SECT    ROMCOD
                       345            ASSUME  CS:ROMCOD, DS:ROMCOD, ES:CRTDAT
                       346    ;
0000'                  347    INICRT  PROC    NEAR
0000'  BB 0000'        348            MOV     BX,CRTDAT               ; SET UP ES
0003'  8E C3           349            MOV     ES,BX                   ; AS CRTDAT
0005'  31 FF           350            XOR     DI,DI                   ; DI = BEGINNING OF SCREEN MEMORY (0)
0007'  26C6 06 1800 OF 351            MOV     BYTE PTR ES:ATTLAT,HIGHAT ; SET ATTRIBUTES TO HIGH BRILLIANCE
000D'  B9 0800         352            MOV     CX,2048                 ; SET UP CX FOR REP STOS
0010'  B0 20           353            MOV     AL,SPACE                ; PUT A BLANK IN AL
0012'  F2              354            REP                             ; DI=MEMBEG
0013'  AA              355            STOS    BYTE PTR [DI]           ; CLEAR SCREEN MEMORY
0014'  8C CB           356            MOV     BX,CS                   ; SET UP DS
0016'  8E DB           357            MOV     DS,BX                   ; TO POINT TO ROMCOD
0018'  B1 10           358            MOV     CL,16                   ; 16 REGISTERS TO INIT.
001A'  B2 00           359            MOV     DL,HZTCHR               ; START AT CRTC REGISTER ZERO
001C'  BE 000F"        360            MOV     SI,OFFSET ALTMON        ; ALTMON = ALTERNATE MONITOR PROG.
001F'  E4 01           361            IN      AL,PRCI_P               ; READ LATCH FOR MONITOR TYPE
0021'  A8 04           362            TEST    AL,MONTYP               ; Q: LATCH = ALTERNATE MONITOR ?
0023'  74 03           363            JZ      CRT2                    ;    Y: THEN TABLE ALREADY SET UP
0025'  BE 000E"        364            MOV     SI,OFFSET STAMON        ;    N: THEN SI = STANDARD MONITOR
0028'                  365    CRT2:
0028'  2688 16 1810    366            MOV     BYTE PTR ES:CRTADR,DL   ; OUT REG. NUMBER TO CRTADR REG.
002D'  AC              367            LODS    BYTE PTR[SI]            ; LOAD INIT TABLE VALUE IN REG. AL
002E'  26A2 1812       368            MOV     BYTE PTR ES:WRTREG,AL   ; OUT INIT VALUE TO WRITE REG.
0032'  FE C2           369            INC     DL                      ; NEXT REGISTER TO INIT.
0034'  E2 F2           370            LOOP    CRT2                    ; LOOP UNTIL ALL INIT.
                       371    ;       MOV     AL,00                   ;
0036'  26A2 1030       372            MOV     BYTE PTR ES:GR_RED,AL   ; SET GRAPHICS PALETTES
```

```
003A'  26A2 1020              373          MOV     BYTE PTR ES:GR_GRN,AL   ; ...ALL TO ZERO
003E'  26A2 1010              374          MOV     BYTE PTR ES:GR_BLU,AL   ; ...INITIALLY
0042'  26C6 06 1820 40        375          MOV     BYTE PTR ES:MSCOUT,CDSPEN ; TURN ON SCREEN
0048'  C3                     376          RET                             ; RETURN TO CALLER
                              377  ;
                              378  ;*****************************************************************************
                              379  ;          THIS IS THE INITIALIZATION LOGIC FOR THE CRT CONTROLLER.
                              380  ;        IT PERFORMS THE CRT PUP TEST AND LEAVES THE CRT INITIALIZED
                              381  ;                      IF NO ERRORS OCCURRED.
                              382  ;
                              383  ;       NOTE:  THE PROCEDURE CRTINI IS THE FULL INITIALIZATION LOGIC FOR
                              384  ;              THE CRTC CONTROLLER AND IT MAY BE CALLED AT ANY TIME FOR
                              385  ;              THIS PURPOSE.
                              386  ;*****************************************************************************
                              387  ;
        =0049                 388          SECT    ROMCOD
                              389          ASSUME  CS:ROMCOD, DS:CRTDAT
                              390          ASSUME  ES:NOTHING
                              391  ;
0049'                         392  CRTTST  PROC    NEAR
                              393  ;
0049'  B8 0000'               394          MOV     AX,CRTDAT               ; SET UP
004C'  8E D8                  395          MOV     DS,AX                   ; DS AS CRTDAT
                              396  ;
                              397  ;*****************************************************************************
                              398  ;       BEGIN TESTING OF THE CRTC READABLE REGISTERS
                              399  ;
004E'  BE 0010"               400          MOV     SI,OFFSET TSTPAT        ; SI = TABLE OF TEST PATTERNS
0051'  B9 0004                401          MOV     CX,4                    ; NUMBER OF TESTS
0054'  30 D2                  402          XOR     DL,DL                   ; CLEAR ERROR ACCUMULATOR
0056'  C6 06 1820 20          403          MOV     BYTE PTR MSCOUT,CRTOFF  ; SCREEN OFF SO NO TRASH WILL APPEAR
005B'                         404  CRGTST:
005B'  2E8A 04                405          MOV     AL,CS:[SI]              ; AL=TEST PATTERN
005E'  A2 1800                406          MOV     BYTE PTR ATTLAT,AL      ; WRITE ATT. LATCH
0061'  C6 06 1810 0E          407          MOV     BYTE PTR CRTADR,CURPSH  ;
0066'  2E8A 44 01             408          MOV     AL,CS:BYTE PTR 1[SI]    ; AL=TEST PATTERN
006A'  A2 1812                409          MOV     BYTE PTR WRTREG,AL      ; WRITE CURSOR POSITION HIGH
006D'  C6 06 1810 0F          410          MOV     BYTE PTR CRTADR,CURPSL  ;
0072'  2E8A 44 02             411          MOV     AL,CS:BYTE PTR 2[SI]    ; AL=TEST PATTERN
0076'  A2 1812                412          MOV     BYTE PTR WRTREG,AL      ; WRITE CURSOR POSITION LOW
                              413  ;
                              414  ;       CHECK THE REGISTERS
                              415  ;
0079'  2E8A 04                416          MOV     AL,CS:[SI]              ; AL=TEST PATTERN
007C'  38 06 1800             417          CMP     BYTE PTR ATTLAT,AL      ; Q: IS ATT. LATCH WHAT I WROTE
0080'  74 03                  418          JE      CRGTSO                  ;   Y: JUMP AROUND EXIT
0082'                         419  ATTBAD:                                 ;   N: THEN ATTRIBUTE LATCH FAILURE
0082'  80 CA 07*   \          420          OR      DL,ATTERR               ; ATTRIBUTE LATCH FAILURE
0085'                         421  CRGTSO:                                 ; CONTINUE TESTING
0085'  C6 06 1810 0E          422          MOV     BYTE PTR CRTADR,CURPSH  ;
008A'  2E8A 44 01             423          MOV     AL,CS:BYTE PTR 1[SI]            ; AL=TEST PATTERN
008E'  32 06 1813             424          XOR     AL,BYTE PTR RD_REG      ; Q: IS CUSOR HIGH WHAT I WROTE
```

```
0092'  24 3F                425             AND     AL,00111111B            ;       (CURSOR HIGH IS ONLY 6 BITS)
0094'  74 03                426             JE      CRQTS1                  ;   Y: JUMP AROUND EXIT
0096'                       427   CRHBAD:                                   ;   N: THEN CURSOR HIGH FAILURE
0096'  80 CA 09*            428             OR      DL,CURERR               ; CURSOR ADDRESS REQ. FAILURE
0099'                       429   CRQTS1:                                   ; CONTINUE TESTING
0099'  C6 06 1810 OF       430             MOV     BYTE PTR CRTADR,CURPSL  ;
009E'  2E8A 44 02           431             MOV     AL,CS:BYTE PTR 2[SI]    ; AL=TEST PATTERN
00A2'  38 06 1813           432             CMP     BYTE PTR RD_REG,AL      ; Q: IS CURSOR LOW WHAT I WROTE
00A6'  74 03                433             JE      CRQTS2                  ;   Y: JUMP AROUND EXIT
00A8'                       434   CRLBAD:                                   ;   N: THEN CURSOR LOW FAILURE
00A8'  80 CA 09*            435             OR      DL,CURERR               ; AL=CURSOR ADDRESS REQ. FAILURE
00AB'                       436   CRQTS2:                                   ; CONTINUE TESTING
00AB'  46                   437             INC     SI                      ; POINT TO THE NEXT PATTERN TO CHECK
00AC'  E2 AD                438             LOOP    CRQTST                  ; AND TRY AGAIN
                            439   ;
                            440   ;*************************************************************************
                            441   ;       BEGIN TESTING OF THE CRT SCREEN RAM AND SIMULTANEOUSLY
                            442   ;       TESTING THE ATTRIBUTE MEMORY.
                            443   ;
                            444             ASSUME  ES:CRTDAT
                            445   ;
00AE'  B8 0000'             446   CRMTST: MOV     AX,CRTDAT               ; SET UP ES ALSO
00B1'  8E C0                447             MOV     ES,AX                   ; AS CRTDAT
00B3'  BD 0100              448             MOV     BP,100000000B           ; WALKING ONE
00B6'                       449   CRMTS0:
00B6'  B9 07FF              450             MOV     CX,MEMEND               ; TEST SCREEN RAM
00B9'  31 FF                451             XOR     DI,DI                   ; SET UP START OF CRT RAM.
00BB'  8B F7                452             MOV     SI,DI                   ; COPY TO SI FOR LATER.
00BD'  8B C5                453             MOV     AX,BP                   ; GET WALKING PATTERN TO AX.
00BF'  D1 E8                454             SHR     AX,1                    ; SETUP THE CARRY BIT
00C1'                       455   CRMTS1:
00C1'  A2 1800              456             MOV     BYTE PTR ATTLAT,AL      ; STORE WALKING ONE IN ATT. MEMORY
00C4'  AA                   457             STOS    BYTE PTR [DI]           ; ...AND IN SCREEN MEMORY
00C5'  D0 D0                458             RCL     AL,1                    ; WALK IT
00C7'  E2 F8                459             LOOP    CRMTS1                  ;
00C9'  8B DD                460             MOV     BX,BP                   ; GET INITIAL PATTERN
00CB'  B9 07FF              461             MOV     CX,MEMEND               ; SIZE OF SCREEN RAM
00CE'  D1 EB                462             SHR     BX,1                    ; SETUP THE CARRY BIT
00D0'                       463   CRMTS2:
00D0'  9F                   464             LAHF                            ; SAVE FLAGS
00D1'  8A 04                465             MOV     AL,[SI]                 ; GET PATTERN IN AL
00D3'  38 C3                466             CMP     BL,AL                   ; Q: IS THIS OK ?
00D5'  74 03                467             JE      CRMTS3                  ;   Y: JUMP AROUND JUMP
00D7'                       468   RAMBAD:                                   ;   N: THEN RAM FAILURE
00D7'  80 CA 0A*            469             OR      DL,CRAMER               ; CRT RAM FAILURE
00DA'                       470   CRMTS3:                                   ; CONTINUE TESTING
00DA'  3A 1E 1800           471             CMP     BL,BYTE PTR ATTLAT      ; Q: IS ATTRIBUTE OK ?
00DE'  74 03                472             JE      CRMTS4                  ;   Y: JUMP AROUND JUMP
00E0'                       473   ATMBAD:                                   ;   N: THEN ATTRIBUTE MEMORY FAILURE
00E0'  80 CA 06*            474             OR      DL,ATMERR               ; ATTRIBUTE MEMORY FAILURE
00E3'                       475   CRMTS4:                                   ; CONTINUE TESTING
00E3'  46                   476             INC     SI                      ; NEXT LOCATION
```

```
00E4'  9E                    477             SAHF                            ; GET FLAGS
00E5'  D0 D3                 478             RCL     BL,1                    ; NEXT PATTERN TO CHECK
00E7'  E2 E7                 479             LOOP    CRMTS2                  ; CONTINUE TILL FINISHED
00E9'  81 F5 FFFF            480             XOR     BP,0FFFFH               ; SET UP FOR WALKING ZERO
                             481     ;                                       ; Q: COMPLEMENTED?
00ED'  78 C7                 482             JS      CRMTS0                  ;   N: JUMP AND TEST AGAIN
00EF'  D1 ED                 483             SHR     BP,1                    ; Q: DONE NINE LOOPS YET?
00F1'  73 C3                 484             JNB     CRMTS0                  ;   N: JUMP AND CONTINUE TESTING
                             485     ;
                             486     ;*************************************************************************
                             487     ;       BEGIN TESTING OF THE VIDEO OUTPUT
                             488     ;
                             489             ASSUME  ES:CRTDAT
                             490     ;
00F3'  31 FF                 491     VIDTST: XOR     DI,DI                   ; START OF SCREEN MEMORY
00F5'  B9 0800               492             MOV     CX,2048                 ; SET UP CX FOR REP
00F8'  C6 06 1800 0F         493             MOV     BYTE PTR ATTLAT,HIGHAT  ; ATTLAT = HIGH INT.
00FD'  B0 20                 494             MOV     AL,SPACE                ; PUT A BLANK IN AL
00FF'  F2                    495             REP                             ; DI=MEMBEG
0100'  AA                    496             STOS    BYTE PTR [DI]           ; CLEAR SCREEN MEMORY
0101'  C6 06 1820 40         497             MOV     BYTE PTR MSCOUT,CDSPEN  ; ENABLE CRT FOR VIDEO TEST
0106'  B6 02                 498             MOV     DH,2                    ; DH = TEST COUNT
0108'  B0 DB                 499             MOV     AL,BLOCK                ; AL=BLOCK CHARACTER
010A'  B3 F7                 500             MOV     BL,0F7H                 ; TEST PATTERN FOR A BLOCK CHAR.
010C'                        501     VDTST0:
010C'  FE CE                 502             DEC     DH                      ; DH = NUMBER OF CURRENT TEST
010E'  31 FF                 503             XOR     DI,DI                   ; GET BEGIN OF MEMORY
0110'  B9 0050               504             MOV     CX,80                   ; CX=80 CHARACTERS TO WRITE
0113'  F2                    505             REP
0114'  AA                    506             STOS    BYTE PTR [DI]           ; WRITE 80 BLOCK CHARS. TO SCREEN
0115'  B9 0002               507             MOV     CX,2                    ; CX=25us LOOP COUNT
0118'  BD FFFF               508             MOV     BP,0FFFFH               ;
011B'                        509     VDTST1:
011B'  4D                    510             DEC     BP                      ;
011C'  74 1E                 511             JZ      VBLBAD                  ; TRAP IN CASE CRTVBL IS NOT WORKING
011E'  F6 06 1811 20         512             TEST    BYTE PTR CRTSTA,CRTVBL  ; Q: VERTICAL BLANK ??
0123'  74 F6                 513             JZ      VDTST1                  ;   N: WAIT UNTIL BLANK
0125'                        514     VDTST2:
0125'  4D                    515             DEC     BP                      ;
0126'  74 14                 516             JZ      VBLBAD                  ; TRAP IN CASE CRTVBL IS NOT WORKING
0128'  F6 06 1811 20         517             TEST    BYTE PTR CRTSTA,CRTVBL  ; Q: VERTICAL NON-BLANK ??
012D'  75 F6                 518             JNZ     VDTST2                  ;   N: WAIT UNTIL NON-BLANK
012F'  E2 FE                 519             LOOP    $                       ;   Y: START 25us TIMEOUT
0131'  38 1E 1000            520             CMP     BYTE PTR MSCINP,BL      ; Q: VIDEO LOOPBACK = TEST PATTERN ??
0135'  74 08                 521             JE      VDTST3                  ;   Y: JUMP AROUND BRANCH
0137'                        522     VIDBAD:                                 ;   N: THEN VIDEO FAILURE
0137'  80 CA 0B*             523             OR      DL,VIDERR               ; VIDEO FAILURE
013A'  EB 03                 524             JMP     SHORT VDTST3            ;
013C'                        525     VBLBAD:
013C'  80 CA 0C*             526             OR      DL,VBLERR               ; VERTICAL BLANKING INTERRUPT BAD
013F'                        527     VDTST3:                                 ; CONTINUE TESTING
013F'  08 F6                 528             OR      DH,DH                   ; Q: LAST TEST ??
```

```
0141'  74 06                529           JE      CINTST                  ;   Y: THEN GO TEST THE CRT INTERRUPT.
0143'  B0 20                530           MOV     AL,SPACE                ;   N: THEN TEST A BLANK CHAR.
0145'  B3 F0                531           MOV     BL,0F0H                 ; TEST PATTERN FOR A BLANK
0147'  EB C3                532           JMP     SHORT VDTST0            ; GO TEST WITH DIFFERENT CHAR.
                            533   ;
                            534   ;*****************************************************************************
                            535   ;       THIS PERFORMS A TEST OF THE CRT INTTERUPT
                            536   ;
                            537           ASSUME  DS:CRTDAT, ES:ABS0
                            538   ;
0149'                       539   CINTST:
0149'  06                  540           PUSH    ES                      ; SAVE ES
014A'  31 DB                541           XOR     BX,BX                   ; SET UP ES
014C'  8E C3                542           MOV     ES,BX                   ; AS ABS0
014E'  B9 1500              543           MOV     CX,CINTIM               ; WAIT COUNT FOR CRT INTERRUPT
0151'  FA                   544           CLI                             ; DISBALE INTERRUPTS
                            545   ;
0152'  1E                  546           PUSH    DS                      ;;; SAVE DS
0153'  2E8E 1E C180         547           MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; SET UP DS AS ROMDAT
0158'  E8 0005"             548           CALL    PRTRST                  ;;; RESET PRINTER NMI
015B'  1F                   549           POP     DS                      ;;; RESTORE DS
015C'  2688 1E 0008         550           MOV     BX,ES:WORD PTR (NMIINT*4)       ;;; SAVE CURRENT NMI VECTOR
0161'  26C7 06 0008 01D8"  551           MOV     ES:WORD PTR (NMIINT*4),OFFSET CRTVCT    ;;; CRT INT VECTOR
0168'  B4 FF                552           MOV     AH,0FFH                 ;;; FLAG TO CHECK FOR INTERRUPT
016A'  C6 06 182v 80        553           MOV     BYTE PTR MSCOUT,CINTEN  ;;; ENABLE CRT INTERRUPTS
016F'  E2 FE                554           LOOP    $                       ;;; WAIT FOR INTERRUPT
0171'  C6 06 1820 40        555           MOV     BYTE PTR MSCOUT,CDSPEN  ;;; DISABLE CRT INTERRUPTS JUST IN CASE
0176'  80 FC 00             556           CMP     AH,00H                  ;;; Q: DID IT OCCUR ?
0179'  74 03                557           JE      CINTS0                  ;;;   N: THEN CONTINUE
017B'  80 CA 08*            558           OR      DL,CINTER               ;;; CRT INTERRUPT ERROR ?
017E'  2689 1E 0008         559   CINTS0: MOV     ES:WORD PTR (NMIINT*4),BX       ;;; RESTORE NMI VECTOR
0183'  07                   560           POP     ES                      ;;; RESTORE ES
                            561           ASSUME  DS:ROMDAT               ;;;
0184'  1E                  562           PUSH    DS                      ;;; SAVE DS
0185'  2E8E 1E C180         563           MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; SET UP DS AS ROMDAT
018A'  80 0E 0018" 08       564           OR      PRCO_M,PARIEN           ;;; RE-ENABLE PARITY NMI INTERRUPT
018F'  A0 0018"             565           MOV     AL,PRCO_M               ;;;
0192'  E6 03                566           OUT     PRCO_P,AL               ;;;
0194'  1F                   567           POP     DS                      ;;; RESTORE DS
0195'  FB                   568           STI                             ;;; INTERRUPTS BACK ON
                            569   ;
                            570   ;*****************************************************************************
                            571   ;       TESTING COMPLETE!!!  ALL TESTS PASSED WITH HONORS.
                            572   ;
0196'                       573   TSTEND:
0196'  E8 0047              574           CALL    CRTHLT                  ; GO SEE IF ANY ERRORS OCCURRED
0199'  E8 0003"             575           CALL    DECLED                  ; TESTS PASSED YEA!!!!!
019C'  E8 0003              576           CALL    CRTINI                  ; LEAVE CRT FULLY INITIALIZED
019F'  E9 0004"             577           JMP     MSCTST                  ; GO TO MISCELLANEOUS TEST ROUTINE
                            578   ;
                            579   ;*****************************************************************************
                            580   ;
```

```
                                581   ; CRTINI - THIS IS THE CRT FULL INITIALIZATION SUBROUTINE.
                                582   ;
                                583   ;       INPUT - NONE
                                584   ;
                                585   ;       OUTPUT - CRT INITIALIZED
                                586   ;
                                587   ;       REGISTERS USED - ALL REGISTERS PRESERVED
                                588   ;
                                589   ;       STACK USED -
                                590   ;
                                591   ;*****************************************************************************
                                592   ;
          =01A2                 593           SECT    ROMCOD
                                594           ASSUME  CS:ROMCOD, DS:ROMDAT
                                595           ASSUME  ES:CRTDAT               ;
                                596   ;
     01A2'                      597   CRTINI  PROC    NEAR
     01A2'  FC                  598           CLD                             ; CLEAR DIRECTION FLAG
     01A3'  57                  599           PUSH    DI                      ; SAVE DI
     01A4'  56                  600           PUSH    SI                      ; SAVE SI
     01A5'  1E                  601           PUSH    DS                      ; SAVE OLD DS
     01A6'  06                  602           PUSH    ES                      ; SAVE OLD ES
     01A7'  53                  603           PUSH    BX                      ; SAVE BX
     01A8'  E8 FE55             604           CALL    INICRT                  ; GO AND INIT THE CRT
     01AB'  2E8E 1E C180        605           MOV     DS,WORD PTR CS:DSADDR+CSWRAP ; SET UP DS AS ROMDAT
     01B0'  31 C0               606           XOR     AX,AX                   ; Clear AX
     01B2'  88 26 0019"         607           MOV     RETFLG,AH               ;
     01B6'  88 26 0013"         608           MOV     CRTHLD,AH               ; INITIALIZE
     01BA'  C7 06 0015" 400B    609           MOV     CURTYP,400BH            ;
     01C0'  A3 001B"            610           MOV     STATLN,AX               ; LOCAL -
     01C3'                      611   INILST:                                 ;
     01C3'  A3 0014"            612           MOV     CURPOS,AX               ;
     01C6'  A3 001A"            613           MOV     THSLIN,AX               ;
     01C9'  A3 0016"            614           MOV     DISBEG,AX               ; STORAGE -
     01CC'  C7 06 0017" 07CF    615           MOV     DISEND,07CFH            ;
     01D2'  A3 0011"            616           MOV     CRTCNT,AX               ;
     01D5'  E9 0001"            617           JMP     CRTRET                  ; RETURN TO CALLER
                                618   ;
                                619   ;*****************************************************************************
                                620   ;       THIS IS THE CRT INTERRUPT LOGIC
                                621   ;*****************************************************************************
                                622   ;
     01D8'                      623   CRTVCT:
     01D8'  C6 06 1820 40       624           MOV     BYTE PTR MSCOUT,CDSPEN  ; DISABLE CRT INTERRUPTS
     01DD'  B4 00               625           MOV     AH,00H                  ; CLEAR AH
     01DF'  CF                  626           IRET                            ; RETURN FROM INTERRUPT
                                627   ;
                                628   ;*****************************************************************************
                                629   ;       THIS ROUTINE IS CALLED UPON COMPLETION OF ALL CRT TESTS.
                                630   ;       IT CHECKS TO SEE IF ANY ERRORS HAVE OCCURRED UP TILL NOW.
                                631   ;       IF NO ERRORS HAVE OCCURRED THEN IT RETURNS, HOWEVER IF
                                632   ;       THERE WERE ANY ERRORS THEN THEY ARE OUTPUT TO THE PARALLEL
```

```
                        633   ;       PRINTER PORT AND A JUMP TO DSPDIE IS PERFORMED WHERE THE
                        634   ;       THE ERROR IS PUT TO THE SCREEN (ACADEMIC AT THIS POINT)
                        635   ;       AND ALL TESTING HALTS!!!
                        636   ;
                        637   ;       NOTE: SEE MODULE TSTERR.SRC FOR POSSIBLE ERROR CODES.
                        638   ;*******************************************************************************
                        639   ;
01E0'                   640   CRTHLT:
01E0'  08 D2            641           OR      DL,DL                   ; Q: ANY ERRORS TO REPORT ??
01E2'  74 05            642           JE      HLTRET                  ;   N: THEN RETURN
01E4'  B0 0D*           643           MOV     AL,LED010               ; AL = CRT ERROR CODE = LED'S
01E6'  E9 0002"         644           JMP     DSPDIE                  ; GO DISPLAY ERROR AND DIE
                        645   ;
01E9'  C3               646   HLTRET: RET                             ; GO BACK - NO ERRORS
                        647   ;
                        648           END

No errors detected
```

```
                        1   ;*******************************************************************************
                        2   ;  TITLE    - DKBOOT - Disk boot routine
                        3   ;  ABSTRACT - This logic is responsible for reading the first sector
                        4   ;       (The BOOT sector) from the disk, checking its validity and then
                        5   ;       transferring control to BOOTLO (the first location of the data
                        6   ;       just read).
                        7   ;
                        8   ;*******************************************************************************
                        9           NAME    DKBOOT - Disk boot routine                              -
                       10   ;*******************************************************************************
                       11   ;                           PUBLIC DEFINITIONS
                       12   ;*******************************************************************************
                       13   ;
                       14           PUBLIC  DKBOOT
                       15           PUBLIC  PROMPT
                       16   ;
                       17   ;*******************************************************************************
                       18   ;                           EXTERNAL REFERENCES
                       19   ;*******************************************************************************
                       20   ;
                       21           EXTRN   BOOTSZ:ABS      ; Boot sector size                      (SYSORG)
                       22           EXTRN   DBCERR:ABS      ; Bad CRC on boot sector error code     (ROMERR)
                       23           EXTRN   DCRERR:ABS      ; CRC error code                        (ROMERR)
                       24           EXTRN   DFMERR:ABS      ; Disk format error code                (ROMERR)
                       25           EXTRN   DNIERR:ABS      ; 'No drives installed' error code      (ROMERR)
                       26           EXTRN   DNRERR:ABS      ; 'Disk not ready' error code           (ROMERR)
                       27           EXTRN   DNSERR:ABS      ; 'Not a TI system disk' error code     (ROMERR)
                       28           EXTRN   DSKERR:ABS      ; Seek error code                       (ROMERR)
                       29           EXTRN   DSNERR:ABS      ; Sector-not-found error code           (ROMERR)
                       30           EXTRN   DUNERR:ABS      ; 'Disk error' (unknown) error code     (ROMERR)
                       31           EXTRN   DDMERR:ABS      ; DRQ error                             (ROMERR)
                       32   ;
                       33           EXTRN   DSPERR:NEAR     ; System error display routine          (OUTPUT)
                       34           EXTRN   CRTOUT:NEAR     ; System error display routine          (CRTDSR)
                       35           EXTRN   BOOTLO:FAR      ; First instr of boot sector            (SYSORG)
                       36           EXTRN   FLUSH:NEAR      ; Keyboard buffer flush routine         (KEYDSR)
                       37           EXTRN   KEYIN:NEAR      ; Keyboard character input routine      (KEYDSR)
                       38           EXTRN   LBEEP:NEAR      ; Long (error) beep routine             (BELDSR)
                       39           EXTRN   MSG:NEAR        ; CRT string output routine             (OUTPUT)
                       40           EXTRN   ROMTST:NEAR     ; CRC checking routine                  (PUPTST)
                       41           EXTRN   SBEEP:NEAR      ; Short beep routine                    (BELDSR)
                       42   ;
=0000                  43           SECT    ABS0
                       44           ASSUME  CS:ABS0
                       45           EXTRN   BOOTMV:BYTE     ; Location of boot sector in memory     (SYSORG)
                       46   ;
=0000                  47           SECT    ROMDAT
                       48           EXTRN   SYSCON:WORD     ; System Configuration word             (ROMDAT)
                       49   ;
                       50   ;*******************************************************************************
                       51   ;                           LOCAL CONSTANTS
                       52   ;*******************************************************************************
```

```
                                53  ;
          =000D                 54  CR      EQU     0DH
          =000A                 55  LF      EQU     0AH
                                56  ;
                                57  ;-----------------------------------------------------------------------------
                                58  ;       INCLUDE PEQ:DSKERR.EQU
                                59  ;       INCLUDE PEQ:DSKOPS.EQU
                                60  ;       INCLUDE PEQ:SYSCELL.EQU
                                61  ;       INCLUDE PEQ:SYSCON.EQU
                                62  ;       INCLUDE PEQ:CRTOP.EQU
                                63  ;       INCLUDE PEQ:VECTOR.EQU
                                64  ;-----------------------------------------------------------------------------
                               302  ;*****************************************************************************
                               303  ;                           CODE SEGMENT DEFINITION
                               304  ;*****************************************************************************
                               305  ;
          =0000                306          SECT    ROMCOD
                               307          ASSUME  CS:ROMCOD
                               308  ;
                               309  ;*****************************************************************************
                               310  ;                           LOCAL DATA AREA
                               311  ;*****************************************************************************
                               312  ;
0000' 0D 0A 0A 50 6C 65        313  INSMSG  DB      CR,LF,LF,'Please insert system disk and'
0006' 61 73 65 20 69 6E
000C' 73 65 72 74 20 73
0012' 79 73 74 65 6D 20
0018' 64 69 73 6B 20 61
001E' 6E 64
0020' 0D 0A 53 74 72 69        314  PROMPT  DB      CR,LF,'Strike any key when ready ...',0
0026' 6B 65 20 61 6E 79
002C' 20 6B 65 79 20 77
0032' 68 65 6E 20 72 65
0038' 61 64 79 20 2E 2E
003E' 2E 00
0040' 20 6F 6E 20 44 72        315  DRVMSG  DB      ' on Drive ',0
0046' 69 76 65 20 00
                               316  ;
                               317  ;   Error code mapping table to take Disk DSR errors to System error codes.
                               318  ;   Each entry consists of 2 bytes - first byte is error code from disk
                               319  ;       DSR, second byte is the boot error code to be displayed on the
                               320  ;       CRT as '** System Error **  - xxxx on Drive x'.
                               321  ;       Entries marked with '*' are either hardware failures or errors
                               322  ;       which will never happen since I control the interface to the DSR.
                               323  ;
004B' 40 08*                   324  ERRTAB  DB      DKSEEK,DSKERR   ; Seek failed - track not found
004D' 10 03*                   325          DB      DKCRCE,DCRERR   ; CRC error on read
004F' 08 0B*                   326          DB      DKDMAE,DDMERR   ; Data request error - controller failure
0051' 04 09*                   327          DB      DKRNFE,DSNERR   ; Sector not found error
0053' 02 04*                   328          DB      DKFMTE,DFMERR   ; Timeout - no data error - bad disk format
0055' 07* 07*                  329          DB      DNSERR,DNSERR   ; Not a TI disk error.
0057' 20 0A*                   330          DB      DKFAIL,DUNERR   ; * Controller hardware failure
```

```
                        ;       DB      DKNRDY,DNRERR   ; Controller timeout - disk not ready
                        ;       DB      DKCMDE,DUNERR   ; * Command error - bad command passed
                        ;       DB      DKWPRT,DUNERR   ; * Write protect error
                        ;       DB      DKVFYE,DUNERR   ; * Data verification error
                        ;       DB      DKBOUN,DUNERR   ; * I/O transfer crosses 64k boundary
                        ;
=0007                   ERRTSZ  EQU     ($-ERRTAB)/2    ; Table size
                        ;*******************************************************************************
                        ;                               MODULE ENTRY POINT
                        ;*******************************************************************************
                        ;
                        ;   DKBOOT - This routine gets control after powerup initialization in order
                        ;       to 'boot' the disk system.  It reads the first sector on the default
                        ;       disk into memory, and transfers control it.
                        ;
                        ;   INPUT:  (none)
                        ;   OUTPUT: BL = Drive to boot from
                        ;           (transfers control to the code loaded from the boot sector)
                        ;   USED:
                        ;   STACK:
                            ASSUME      CS:ROMCOD, DS:ROMDAT, ES:ABS0
                        ;
                        ;       Check if any drives are installed.  If not, report the system error
                        ;       and wait forever.
                        ;
0059'                   DKBOOT  PROC    NEAR
0059'  31 DB                    XOR     BX,BX
005B'  8E C3                    MOV     ES,BX                   ; Set up ES to ABS0
005D'  2E8E 1E C180             MOV DS,WORD PTR CS:DSADDR+CSWRAP ; Set up my DS
0062'  A0 0016"                 MOV     AL,BYTE PTR SYSCON      ; Look at floppy drive status
0065'  24 0F                    AND AL,DROMSK+DR1MSK+DR2MSK+DR3MSK
                                                                ; Q: Are any drives installed?
0067'  75 0A                    JNZ     DKB0                    ;  Y: Continue boot process.
0069'  B0 05*                   MOV     AL,DNIERR               ;  N: 'No drives' error code
006B'  E8 000C"                 CALL    DSPERR                  ; Display the error
006E'                   WAITLP:
006E'  E8 0010"                 CALL    KEYIN                   ; Wait for any key
0071'  EB FB                    JMP     WAITLP                  ; Loop forever (til they hit reset)
                        ;
                        ;       Try to boot from an installed drive.  If successful, do not return.
                        ;       If NOT successful, try the next installed drive.
                        ;       If no drive is successful, Prompt user for a disk installation.
                        ;
0073'  30 DB            DKB0:   XOR     BL,BL                   ; Set initial boot drive to 00
0075'  A0 0016"                 MOV     AL,BYTE PTR SYSCON      ; Get configuration byte.
0078'  D0 E8            DKB1:   SHR     AL,1                    ; Q: Is drive installed ?
007A'  73 07                    JNB     DKB2                    ;  N: Try another
007C'  50                       PUSH    AX                      ;  Y: Save state
007D'  53                       PUSH    BX
007E'  E8 001C                  CALL    BOOTDR                  ; Boot this drive
0081'  5B                       POP     BX                      ; restore state
0082'  58                       POP     AX
```

```
0083' FE C3                    383   DKB2:   INC     BL                      ; Try next drive
0085' 80 FB 04                 384           CMP     BL,4                    ;Q: Tryed all drives?
0088' 75 EE                    385           JNZ     DKB1                    ; N: Jump and continue
008A' B0 06*                   386           MOV     AL,DNRERR               ; Not able to boot error.
008C' E8 000C"                 387           CALL    DSPERR                  ; Beep the error bell
008F' BE 0000"                 388           MOV     SI,OFFSET INSMSG        ; Point to 'Insert disk' message
0092' E8 0012"                 389           CALL    MSG                     ; and print it (with 'any key' prompt)
0095' E8 000F"                 390           CALL    FLUSH                   ; Empty the keyboard buffer
0098' E8 0010"                 391           CALL    KEYIN                   ; Wait for any key
009B' EB D6                    392           JMP     SHORT DKB0              ; Try boot again
                               393   ;
                               394   ;       Do a complete boot action from this drive.
                               395   ;
009D'                          396   BOOTDR:
009D' 26C6 06 0000" 00         397           MOV     ES:BOOTMV+BOOTID,00     ; Insure ID isn't there from last time
00A3' 06                       398           PUSH    ES                      ; Hang on to transfer segment
00A4' 53                       399           PUSH    BX                      ; Save boot drive
00A5' B0 01                    400           MOV     AL,1                    ; One sector to load
00A7' B5 00                    401           MOV     CH,0                    ; Starting at track 0
00A9' B1 01                    402           MOV     CL,1                    ;  ...sector 1
00AB' 8A D3                    403           MOV     DL,BL                   ; Boot from available drive
00AD' B6 00                    404           MOV     DH,00                   ; Head 0
00AF' B4 02                    405           MOV     AH,DKREAD               ; Sector read function
00B1' BB 0015"                 406           MOV     BX,OFFSET BOOTMV        ; Transfer offset (segment already set)
00B4' CD 4D                    407           INT     DSKINT                  ; Get the sector
00B6' 5B                       408           POP     BX                      ; Restore boot drive
00B7' 07                       409           POP     ES
00B8' 72 2C                    410           JB      CHKERR                  ; Quit if error
                               411   ;
                               412   ;   Check CRC of boot sector:
                               413   ;                                       ; (ES already set to ABS0)
00BA' 53                       414           PUSH    BX                      ; Save the boot drive
00BB' BB 0015"                 415           MOV     BX,OFFSET BOOTMV        ; Offset of boot sector
00BE' BD 0001*                 416           MOV     BP,BOOTSZ               ; Size of sector
00C1' E8 0013"                 417           CALL    ROMTST                  ; Q: Does the CRC calculate OK ?
00C4' 5B                       418           POP     BX                      ; Save the boot drive
00C5' 74 0A                    419           JZ      CHKID                   ;   Y: Go on and check the ID
00C7' 2683 3E 0000" 00         420           CMP     WORD PTR ES:BOOTMV+BOOCRC,0000  ; Q: Well, is the 'CRC' = 0000?
                               421                                           ;        (this allows pre-CRC disks)
00CD' B0 02*                   422           MOV     AL,DBCERR               ; ('Bad CRC on boot sector' error code)
00CF' 75 2B                    423           JNE     CHKERX                  ;   N: Blow it off with the error
00D1'                          424   CHKID:                                  ;   Y: Continue
00D1' 2681 3E 0000" 6974       425           CMP WORD PTR ES:BOOTMV+BOOTID,'it'
                               426                                           ; Q:   Is boot sector ID = 'ti' ?
00D8' B0 07*                   427           MOV     AL,DNSERR               ;      'Not a TI system disk' error code
00DA' 75 20                    428           JNE     CHKERX                  ;   N: error
00DC' E8 0014"                 429           CALL    SBEEP                   ; Beep the bell before going to boot
00DF' 53                       430           PUSH    BX                      ; Save boot drive
00E0' 9A 000E" 0000"           431           CALL    BOOTLO                  ; *** EXIT *** to boot routine
00E5' 5B                       432           POP     BX                      ; Should return to here only if ERROR
                               433   ;       JMP     SHORT CHKERR            ;
                               434.  ;
```

```
                                435   ;       This routine prints the error message for a system error on boot
                                436   ;
00E6'                           437   CHKERR:
00E6'  80 FC 80                 438           CMP     AH,DKNRDY               ; Q: Was the error 'Disk not ready' ?
00E9'  74 24                    439           JZ      CE9                     ;   Y: Don't report as an error
                                440                                           ;      and let them try again.
00EB'  BE 0049"                 441           MOV     SI,OFFSET ERRTAB-2      ; Point to error translation table
00EE'  B9 0007                  442           MOV     CX,OFFSET ERRTSZ        ; Size of table.
00F1'                           443   CE1:
00F1'  46                       444           INC     SI                      ; Point to next entry
00F2'  46                       445           INC     SI
00F3'  2E3A 24                  446           CMP     AH,BYTE PTR CS:[SI]     ; Q: Is this the error ?
00F6'  E0 F9                    447           LOOPNZ  CE1                     ;   N: Point to next entry
00F8'  46                       448           INC     SI                      ;   Y: Point to translation
00F9'  2E8A 04                  449           MOV     AL,CS:[SI]              ; Get new error code
00FC'  53                       450   CHKERX: PUSH    BX                      ; Save drive number
00FD'  E8 000C"                 451           CALL    DSPERR                  ; Display the error
0100'  BE 0040"                 452           MOV     SI,OFFSET DRVMSG        ; Drive message
0103'  E8 0012"                 453           CALL    MSG                     ; Print it
0106'  5B                       454           POP     BX                      ; Get drive number
0107'  B8 0E41                  455           MOV     AX,CRTWTY*100H+'A'      ; Format number
010A'  00 D8                    456           ADD     AL,BL                   ;
010C'  E8 000D"                 457           CALL    CRTOUT                  ; Print it.
010F'  C3                       458   CE9:    RET                             ;
                                459   ;
                                460           END

No errors detected
```

```
                        1   ;*******************************************************************************
                        2   ;  TITLE    - DRVTST - Disk drive test routine
                        3   ;  ABSTRACT - This module is responsible for determining the number of
                        4   ;         disk drives in the system.
                        5   ;*******************************************************************************
                        6           NAME    DRVTST
                        7   ;       SUBTTL
                        8   ;*******************************************************************************
                        9   ;                           PUBLIC DEFINITIONS
                       10   ;*******************************************************************************
                       11   ;
                       12           PUBLIC  DRVTST
                       13   ;
                       14   ;*******************************************************************************
                       15   ;                           EXTERNAL REFERENCES
                       16   ;*******************************************************************************
                       17   ;
                       18           EXTRN   DELAY:NEAR      ; 1 millisecond delay routine          (ROMUTL)
                       19   ;
=0000                  20           SECT    ROMDAT
                       21           EXTRN   DSKS_M:BYTE     ; RAM copy of disk select port         (ROMDAT)
                       22           EXTRN   SYSCON:WORD     ; System configuration word            (ROMDAT)
                       23   ;*******************************************************************************
                       24   ;                           LOCAL CONSTANTS
                       25   ;*******************************************************************************
                       26   ;
=0001                  27   BSYMSK  EQU     00000001B       ; Busy bit mask in FDC status reg
=0004                  28   TROMSK  EQU     00000100B       ; Track 0 bit mask in FDC status reg
=0056                  29   NUMSTP  EQU     86              ; Number of steps to be performed during recal
                       30   ;
=0061                  31   STOCMD  EQU     61H             ; FDC STEP OUT command (12 msec, no verify)
=0010                  32   SKNCMD  EQU     10H             ; FDC seek command (6 msec, no verify)
                       33   ;
                       34   ;       INCLUDE PEQ:PORTADDR.EQU
                       35   ;       INCLUDE PEQ:LATCHES.EQU
                       36   ;       INCLUDE PEQ:FLOPPY.EQU
                       37   ;       INCLUDE PEQ:SYSCON.EQU
                       38
                      188   ;*******************************************************************************
                      189   ;                           CODE SEGMENT DEFINITION
                      190   ;*******************************************************************************
                      191   ;
=0000                 192           SECT    ROMCOD
                      193           ASSUME  CS:ROMCOD
                      194   ;
                      195   ;*******************************************************************************
                      196   ;                           MODULE ENTRY POINT
                      197   ;*******************************************************************************
                      198   ;
                      199   ;   DRVTST - This routine determines the number of operative floppy drives
                      200   ;       in the system.  This is done by attempting a restore operation on
                      201   ;       the drives and watching for TRACK 00 indication.
```

```
                        202     ;
                        203     ;       At the point this routine is normally called during system powerup
                        204     ;       initialization, the floppy controller test has been performed, and
                        205     ;       the controller should be 'settled-down'.
                        206     ;
                        207     ;       Note that this routine handles all four drives.  Also,
                        208     ;       the motors are not turned on during these operations.
                        209     ;
                        210     ;   INPUT:  Sector buffer control bits set so that the CPU can talk to the
                        211     ;               floppy disk controller
                        212     ;           DS pointing to ROMDAT
                        213     ;   OUTPUT: All drives restored and SYSCON updated to indicate installed
                        214     ;           drives
                        215     ;   USED:   ax,bx,cx,dx
                        216     ;   STACK:
                        217         ASSUME  CS:ROMCOD, DS:ROMDAT
                        218     ;-----------------------------------------------------------------------------
0000'                   219     DRVTST  PROC    NEAR
0000'  B0 F0            220             MOV     AL,0F0H                 ; All motors on and all drives selected
0002'  E6 04            221             OUT     DSKS_P,AL               ; Select all drives
0004'  E8 0049          222             CALL    CLRFDC                  ; Clear the controller
                        223     ;
                        224     ;   Need to step in at least 4 steps in case the head is outside track 00
                        225     ;
0007'  30 C0            226             XOR     AL,AL                   ; AL = 0
0009'  E6 21            227             OUT     FDCTRK,AL               ; Fake current track = 00
000B'  B0 04            228             MOV     AL,4
000D'  E6 23            229             OUT     FDCDAT,AL               ; Fake destination track = 04
000F'  B0 10            230             MOV     AL,SKNCMD               ; Set up Seek with no verify command
0011'  E8 0048          231             CALL    DO_CMD                  ; Go do it.
                        232     ;
                        233     ;   Recalibrate the drive:
                        234     ;
0014'  B6 F0            235             MOV     DH,0F0H                 ; All motors on and all drives selected
                        236                                             ; Select all four drives
0016'  B4 56            237             MOV     AH,86                   ; Restore for 86 tracks
0018'  8A C6            238     CD2:    MOV     AL,DH                   ; Select the drive
001A'  E6 04            239             OUT     DSKS_P,AL               ;
001C'  B0 61            240             MOV     AL,STOCMD               ; Restore the drive at 6 milliseconds
001E'  E8 003B          241             CALL    DO_CMD                  ; go do it.
                        242     ;
                        243     ;       CHECK ALL FOUR DRIVES FOR TR00
                        244     ;
0021'  B2 F7            245             MOV     DL,DRSEL3 OR 0F0H       ; Start with drive 3
                        246     ;
0023'  8A C2            247     CD4:    MOV     AL,DL                   ; Select the drive
0025'  E6 04            248             OUT     DSKS_P,AL               ;
0027'  E8 0026          249             CALL    CLRFDC                  ; Go get the drive status
002A'  A8 04            250             TEST    AL,TROMSK               ; Q:  are we at track 00?
002C'  74 06            251             JZ      CD5                     ;  N: Continue.
002E'  F6 D2            252             NOT     DL                      ;  Y: Disable stepping on this drive
0030'  08 D6            253             OR      DH,DL                   ;     By killing the select for restore
```

```
0032'  F6 D2                    254             NOT     DL                              ;       restore sense of select
0034'  D0 CA                    255     CD5:    ROR     DL,1                            ; Q;    All drives checked?
0036'  72 E8                    256             JB      CD4                             ;  N;   Try next drive
                                257     ;
0038'  FE CC                    258             DEC     AH                              ; Q: Finished seeking?
003A'  75 DC                    259             JNZ     CD2                             ;    N: Continue
                                260     ;
                                261     ;       FINISHED RESTORING.
                                262     ;
003C'  80 E6 0F                 263             AND     DH,DRVMSK                       ;
003F'  80 26 0003" F0           264             AND BYTE PTR SYSCON,NOT DRVMSK  ; Mask out the diskdrive bits
0044'  08 36 0003"              265             OR      BYTE PTR SYSCON,DH      ; OR in the installed drive bits
                                266     ;
0048'  B0 0F                    267             MOV     AL,0FH                          ; Turn off everything
004A'  A2 0002"                 268             MOV     DSKB_M,AL                       ; After all this messin' around...
004D'  E6 04                    269             OUT     DSKB_P,AL                       ; Make sure to restore the select latch
004F'  C3                       270             RET                                     ; *** RETURN ***
                                271
                                272     ;
                                273     ;       Clear the disk controller and set status into AL.
                                274     ;
0050'                           275     CLRFDC  PROC    NEAR
0050'  B0 D0                    276             MOV     AL,FORCED                       ; Clear the controller & return status
0052'  E6 20                    277             OUT     FDCCMD,AL                       ; ... and force Type I status
0054'  B9 000A                  278             MOV     CX,10                           ; 10 * 3.4 usec/loop = 34 usec
0057'  E2 FE                    279             LOOP    $                               ; Delay for FDC status to become valid
0059'  E4 20                    280             IN      AL,FDCSTA                       ; Look at status reg
005B'  C3                       281             RET                                     ; *** RETURN ***
                                282     ;
                                283     ;       EXECUTE A FLOPPY DISK COMMAND
                                284     ;
005C'                           285     DO_CMD  PROC    NEAR
005C'  E6 20                    286             OUT     FDCCMD,AL                       ; Do it
005E'                           287     CD1:
005E'  B9 000A                  288             MOV     CX,10                           ; 10 * 3.4 usec/loop = 34 usec
0061'  E2 FE                    289             LOOP    $                               ; Delay for FDC status to become valid
0063'  E4 20                    290             IN      AL,FDCSTA                       ; Look at floppy status
0065'  A8 01                    291             TEST    AL,BSYMSK                       ; Q: Is controller still busy ?
0067'  75 F5                    292             JNZ     CD1                             ;    Y: Spin till not busy
0069'  C3                       293             RET                                     ;    N: Return to user
                                294     ;
                                295             END
```

No errors detected

```
                                ;*****************************************************************************
                                ; TITLE     - DSK_IO
                                ; COMPUTER  - 8088 ASSEMBLY LANGUAGE
                                ; ABSTRACT  - Interface to the disk driver in ROM. Includes the routine DFINISH
                                ;             called by the interrupt-driven, machine independent code to indi-
                                ;             cate that interrupt level processing has completed.
                                ;             Uses the IBM PC conventions.
                                ; INPUTS    - AH = Function code
                                ;
                                ;                IBM compatible function codes:
                                ;                00 = Reset diskette system
                                ;                01 = Return status code for last operation in AL
                                ;                02 = Read the specified sectors into memory
                                ;                03 = Write the specified sectors from memory
                                ;                04 = Verify the specified sectors CRC's
                                ;                05 = Null operation (format track on IBM PC)
                                ;
                                ;                TI private function codes:
                                ;                06 = Verify data in specified sectors
                                ;                07 = Return retry status for last operation in AL
                                ;                08 = Set standard DIT for unit
                                ;                09 = Set DIT address for unit
                                ;                10 = Return DIT address for unit
                                ;                11 = Turn off all Floppy drive motors
                                ;
                                ;                NOTE: DIT = Disk Interface Table
                                ;
                                ;             Register assignments for Read/Write/Verify (crc & data):
                                ;             AL = Number of sectors
                                ;             CH = Cylinder ("track") number
                                ;             CL = Sector number
                                ;             DL = Drive number
                                ;             DH = Track ("head") number
                                ;             ES:BX = buffer address (except for Verify crc)
                                ;
                                ;             Note the two equivalent nomenclatures for disk position:
                                ;             Cylinder, track, & sector (==) "track", "head", & "sector"
                                ;             This code uses cylinder, track, & sector
                                ;
                                ;             Register assignments for Set standard DIT for unit:
                                ;             AL = Disk DIT type:
                                ;                  0 = Single sided, single track density
                                ;                  1 = Double sided, single track density
                                ;                  2 = Single sided, double track density
                                ;                  3 = Double sided, double track density
                                ;             DL = PEGASUS unit (drive) #
                                ;
                                ;             Register assignments for Set DIT address for unit:
                                ;             DL = PEGASUS unit (drive) #
                                ;             ES:BX = address of DIT
                                ;
                                ;             OUTPUTS:
```

```
                        ;
                        ;           AL = status return for functions 01 & 07
                        ;           AH = status return for functions 00, 02 - 06
                        ;           AL = number of unprocessed sectors for any I/O function
                        ;           BX = address of WORD with bad data for function 06
                        ;           ES = same as input ES (except for function 10)
                        ;           Carry flag is set (=1) to indicate an error condition
                        ;
                        ; REGISTERS USED -
                        ;
                        ; STACK USED -
                        ;
                        ;*****************************************************************************
                                NAME    DSK_IO
                                SUBTTL  ROM disk driver front end (uses IBM PC interface)
=FFFF                   DEBUG   EQU     0FFFFH
                        ;*****************************************************************************
                                SUBTTL  ROM disk driver front end: STRUCTURE OF THE DISK DRIVER
                        ;*****************************************************************************
                        ; Structure of the disk driver:
                        ;
                        ;               ----------------
                        ;               |    DSK_IO    |       Implements the standard (IBM PC) inter-
                        ;               ----------------       face to all disk-type devices supported
                        ;                       |              under this driver.
                        ;                       V
                        ;               ----------------
                        ;               |    DSKISR    |       Implements a high-level generic disk
                        ;               ----------------       interface which all disk-type devices
                        ;                       |              are made to look like using software.
                        ;                       |              Includes the basic disk functions and
                        ;                       |              the basic interrupt structure for pro-
                        ;                       |              cessing disk I/O.
                        ;                       V
                        ;               ----------------
                        ;               | Generic disk |       Implements the generic disk operations
                        ;               | operations   |       for a specific disk-type device. Per-
                        ;               ----------------       forms the device-specific operations
                        ;                                      necessary to emulate the high-level
                        ;                                      generic disk operations. This includes
                        ;                                      executing device-specific functions,
                        ;                                      programming of hardware interfaces and
                        ;                                      processing of interrupts not included
                        ;                                      in the basic interrupt structure.
                        ;
                        ;*****************************************************************************
                                SUBTTL  ROM disk driver front end: THE DIT DATA STRUCTURE
                        ;*****************************************************************************
                        ; The Disk Interface Table structure is used to interface device-specific code
                        ; to the generalized disk driver code.
                        ;
                        ; DIT's contain read-only data exclusively and may be ROMmed. If drive speci-
```

```
                        ; fic code requires RAM space, it may use the small workspace provided in the
                        ; ROM BIOS' RAM data area or have its own RAM space allocated somewhere.
                        ;
                        ; The structure of a DIT is as follows:
                        ;
                        ;        (----16 bits---)
                        ;       ------------------
                        ;   00H |                |      Long vector to Disk Interface Routine
                        ;       ---  DITDIR    ---
                        ;   02H |                |
                        ;       ------------------
                        ;   04H |    DITSEC      |      Sector size in bytes
                        ;       ------------------
                        ;   06H |DITTRK |DITCYL  |      Track size in sectors; Cylinder size in tracks
                        ;       ------------------
                        ;   08H |DITDSK |DITERR  |      Disk size in cylinders; Error retry limit
                        ;       ------------------
                        ;   0AH          .
                        ;                .
                        ;                .
                        ;        all other fields
                        ;        are dependent on
                        ;       the requirements of
                        ;        the code for the
                        ;        specific device
                        ;
                        ; The DIT structure for all of the floppy drives is:
                        ;
                        ;        (----16 bits---)
                        ;       ------------------
                        ;       |                |
                        ;       ----  FLPDIR  ----      Floppy Disk Interface Routine
                        ;       |                |
                        ;       ------------------
                        ;       |     DITSEC     |      Sector size in bytes
                        ;       ------------------
                        ;       |DITTRK |DITCYL  |      Track size in sectors; Cylinder size in tracks
                        ;       ------------------
                        ;       |DITDSK |DITERR  |      Disk size in cylinders; Error retry limit
                        ;       ------------------
                        ;       | xPRCMP |             Threshold track # for changing write pre-comp
                        ;       ----------
                        ;*****************************************************************************
                        ;                          PUBLIC DEFINITIONS
                        ;*****************************************************************************
=0000                   SECT    ROMCOD
                        PUBLIC  DCALL                   ; Call Generic Disk Operation
                        PUBLIC  DFINISH                 ; Called at end of interrupt processing
                        PUBLIC  DSK_IO                  ; Main entry point for general disk I/O
                        PUBLIC  DSKINI                  ; Initialization entry pt. (if needed)
                        PUBLIC  INVALID                 ; Invalid position value
                        PUBLIC  SOFINT                  ; Software interruptinto interrupt code
```

```
                        158           PUBLIC  TIMOUT                  ; Entry point for disk timeout event
                        159   ;*****************************************************************************
                        160   ;                               EXTERNAL REFERENCES
                        161   ;*****************************************************************************
=0000                   162           SECT    ROMCOD
                        163           EXTRN   FLPDIR:FAR              ; Disk Interface Routine for floppy dsk
                        164           EXTRN   MOTOFF:NEAR             ; Kill motor and timeout
                        165   ;*****************************************************************************
=0000                   166           SECT    ROMDAT
                        167           EXTRN   D_DBGN:BYTE             ; Beginning of data area
                        168           EXTRN   D_DIT:WORD              ; Table of drive DIT's
                        169           EXTRN   D_DEND:BYTE             ; End of data area
                        170           EXTRN   D_POS:BYTE              ; Table of drive current positions
                        171           EXTRN   D_REQ:BYTE              ; Disk driver Request Information Block
                        172           EXTRN   D_SOFT:BYTE             ; Software interrupt flag for DSKISR
                        173           EXTRN   D_STAT:BYTE             ; Disk driver current state
                        174           EXTRN   D_WAIT:BYTE             ; Waiting for interrupt processing flag
                        175           EXTRN   NUMDRV:ABS              ; Max number of drives in the system
                        176           EXTRN   SYSCON:WORD             ; System configuration word
                        177   ;*****************************************************************************
                        178   ;                               LOCAL CONSTANTS
                        179   ;*****************************************************************************
                        180   ; NOTE:  These values have been assigned equates for mnemonic purposes.  THIS
                        181   ;        ABSOLUTELY, POSITIVELY *DOES NOT* mean that the values can be changed!!
                        182   ;        DON'T CHANGE THE VALUES, the code WILL NOT work if you do!!!!!!!!!!!!!!
                        183   ;*****************************************************************************
=0001                   184   BUSY    EQU     1                       ; Busy  value for wait flag
=FFFF                   185   INVALID EQU     -1                      ; "Invalid" pointer value
=0004                   186   NUMDIT  EQU     4                       ; Number of standard DIT's
=0004                   187   NUMFLP  EQU     4                       ; Number of floppy drives
=0000                   188   READY   EQU     0                       ; Ready value for wait flag
                        189   ;*****************************************************************************
                        190   ;                               DISK DRIVER STATE DEFINITIONS
                        191   ;*****************************************************************************
                        192   ;       INCLUDE PEO:DSKSTA.EQU
                        193
                        205   ;*****************************************************************************
                        206   ;                               DISK INTERFACE TABLE DEFINITIONS
                        207   ;*****************************************************************************
                        208   ;       INCLUDE PEO:DSKDIT.EQU
                        209
                        224   ;*****************************************************************************
                        225   ;                               DISK OPCODES (IBM COMPATIBLE ROM BIOS)
                        226   ;*****************************************************************************
                        227   ;       INCLUDE PEO:DSKOPS.EQU
                        228
                        250   ;*****************************************************************************
                        251   ;                               DIT DEFINITIONS
                        252   ;*****************************************************************************
=0002                   253   DSCYL   EQU     2                       ; Double sided cylinder size (tracks)
=0001                   254   SSCYL   EQU     1                       ; Single sided cylinder size (tracks)
=0050                   255   DTDSK   EQU     80                      ; Double track density disk size (cyls)
```

```
                =0028                   256   STDSK   EQU     40                      ; Single track density disk size (cyls)
                =0028                   257   DTPRCMP EQU     DTDSK/2                 ; Double track density pre-comp s-over
                =0002                   258   FLPERR  EQU     2                       ; Maximum number of times to retry
                =0014                   259   STPRCMP EQU     STDSK/2                 ; Single track density pre-comp s-over
                =0200                   260   XXSEC   EQU     512                     ; Sector size in bytes for all cases
                =0008                   261   XXTRK   EQU     8                       ; Track size in sectors for all cases
                                        262   ;*****************************************************************************
                                        263   ;                               ERROR CODE DEFINITIONS
                                        264   ;*****************************************************************************
                                        265   ;       INCLUDE PEQ:DSKERR.EQU
                                        266
                                        304   ;*****************************************************************************
                                        305   ;                               GENERIC DISK FUNCTION CODES (INTERNAL USE ONLY)
                                        306   ;*****************************************************************************
                                        307   ;       INCLUDE PEQ:DSKREQ.EQU
                                        308
                                        319   ;*****************************************************************************
                                        320   ;                               PEGASUS INTERRUPT VECTORS
                                        321   ;*****************************************************************************
                                        322   ;       INCLUDE PEQ:VECTOR.EQU
                                        323
                                        417   ;*****************************************************************************
                                        418   ;                               REQUEST INFORMATION BLOCK STRUCTURE DEFINITION
                                        419   ;*****************************************************************************
                                        420   ;       INCLUDE PEQ:DSKRIB.EQU
                                        421
                                        437
                                        438           SUBTTL  ROM disk driver front end:  DSK_IO (ENTRY POINT)
                                        439   ;*****************************************************************************
                                        440   ;                               MODULE ENTRY POINT
                                        441   ;*****************************************************************************
                =0000                   442           SECT    ROMCOD
                                        443           ASSUME  CS:ROMCOD,DS:ROMDAT,ES:NOTHING
                                        444   ;*****************************************************************************
0000'                                   445   DSK_IO  PROC    FAR
0000'  FB                               446           STI                             ; We entered via a software interrupt,
                                        447                                           ;   so enable the interrupts
0001'  1E                               448           PUSH    DS                      ; Save registers used by DSKIO
0002'  51                               449           PUSH    CX
0003'  56                               450           PUSH    SI
0004'  57                               451           PUSH    DI
0005'  55                               452           PUSH    BP
0006'  52                               453           PUSH    DX
0007'  2E8E 1E C180                     454           MOV     DS,word ptr CS:DSADDR+CSWRAP ; Set local DS
                                        455                                           ; Store request info (defined or not)
000C'  8C 06 0000"                      456           MOV     word ptr D_REQ+BUF+2,ES ; Buffer's segment base
0010'  89 1E 0000"                      457           MOV     word ptr D_REQ+BUF+0,BX ; Buffer's segment offset
0014'  A2 0000"                         458           MOV     D_REQ+SECKNT,AL         ; Sector count
0017'  88 2E 0000"                      459           MOV     D_REQ+CYL,CH            ; Cylinder # for request
001B'  88 0E 0000"                      460           MOV     D_REQ+SEC,CL            ; Sector # for request
001F'  88 36 0000"                      461           MOV     D_REQ+TRK,DH            ; Track # for request
                                        462                                           ;
```

```
0023'  8A C4                463           MOV     AL,AH                   ; AL := Command code
0025'  3C 0C                464           CMP     AL,DKBADC               ; Q: Is this a good command?
0027'  73 25                465           JAE     CMDERR                  ;   N: Return bad command error
0029'  98                   466           CBW                             ;   Y: Convert to word value
002A'  D1 E0                467           SHL     AX,1                    ;      Convert to word index
002C'  8B F0                468           MOV     SI,AX                   ;      Put in index register for jump
002E'  E8 012C              469           CALL    CHGDRV                  ;      Call CHGDRV to change drive
                            470                                           ;      Also sets error flags for use by
                            471                                           ;        the function routines
0031'  2EFF A4 0036"        472           JMP     CS:CMDTBL[SI]           ;      Jump to return for command
                            473   ;*****************************************************************************
0036'  00A3"                474   CMDTBL  DW      OPRSET                  ; Vector for reset
0038'  00AB"                475           DW      OPSTAT                  ; Vector for return status
003A'  00C9"                476           DW      OPREAD                  ; Vector for read sectors
003C'  00CD"                477           DW      OPWRIT                  ; Vector for write
003E'  00D1"                478           DW      OPVERF                  ; Vector for verify CRC
0040'  006D"                479           DW      OKEXIT                  ; Vector for format (null operation)
0042'  00D5"                480           DW      OPVRFY                  ; Vector for verify data
0044'  00B0"                481           DW      OPSSTA                  ; Vector for return retry status
0046'  0071"                482           DW      OPFSET                  ; Vector for set floppy DIT for drive
0048'  0088"                483           DW      OPXSET                  ; Vector for set DIT address for drive
004A'  00B5"                484           DW      OPRDIT                  ; Vector for return DIT addr for unit
004C'  00A8"                485           DW      OPKILM                  ; Vector for turning off Floppy motors
                            486   ;*****************************************************************************
                            487           SUBTTL  ROM disk driver front end: EXIT (EXIT FROM BIOS CALL)
                            488   ;*****************************************************************************
004E'  B0 01                489   CMDERR: MOV     AL,DKCMDE               ; Return with command error
                            490   ;       JMP     short ptr EXITE         ; EXITE immediately follows
                            491   ;*****************************************************************************
0050'  A2 0000"             492   EXITE:  MOV     D_REQ+STAT,AL           ; Slow set status & return to caller
0053'  8A 26 0000"          493   EXIT:   MOV     AH,D_REQ+STAT           ; Set status & return to caller
0057'  80 FC 00             494           CMP     AH,DKNORM               ; Fast set status & return to caller
005A'  74 01                495           JE      RESRET                  ; If no error, go restore & return
005C'  F9                   496           STC                             ; Otherwise, set carry 1st
                            497                                           ; Set appropriate return values
005D'  A0 0000"             498   RESRET: MOV     AL,D_REQ+SECKNT         ; CL := count of unprocessed sectors
0060'  C4 1E 0000"          499           LES     BX,dword ptr D_REQ+BUF  ; ES:BX := Updated pointer in buffer
                            500                                           ;
0064'  5A                   501   POPRET: POP     DX                      ; Restore the saved registers
0065'  5D                   502           POP     BP
0066'  5F                   503           POP     DI
0067'  5E                   504           POP     SI
0068'  59                   505           POP     CX
0069'  1F                   506           POP     DS                      ; Finally, restore calling DS
006A'  CA 0002              507           RET     2                       ; And return (from software interrupt)
                            508   ;*****************************************************************************
006D'  B0 00                509   OKEXIT: MOV     AL,DKNORM               ; Normal exit
006F'  EB DF                510           JMP     short ptr EXITE
                            511   ;*****************************************************************************
                            512           SUBTTL  ROM disk driver front end: NON-I/O FUNCTIONS
                            513   ;*****************************************************************************
                            514   ; OPFSET - SET STANDARD (FLOPPY) DIT FOR DRIVE
```

```
                                515   ;           INPUT:  * DL = Drive number
                                516   ;                   * [D_REQ+SECKNT] (byte) = DIT index
                                517   ;                   * ZF = NZ if drive number is bad
                                518   ;
        =0071                   519   OPFSET  EQU     $                       ; Set DIT addr to standard floppy DIT
0071'   75 DB                  520   JCMDER: JNZ     CMDERR                  ; Return command error if drive # bad
0073'   A0 0000"                521           MOV     AL,D_REQ+SECKNT         ; AL := DIT number
0076'   3C 03                  522           CMP     AL,NUMDIT-1             ; Q: Is it a valid standard DIT index?
0078'   77 D4                  523           JA      CMDERR                  ;    N: Return command error
007A'   98                     524           CBW                             ;    Y: Convert index to word value
007B'   D1 E0                  525           SHL     AX,1                    ;       Convert to word index
007D'   8B D8                  526           MOV     BX,AX                   ;       BX := index into DIT table
007F'   2E8B 87 0318"          527           MOV     AX,CS:DITTBL[BX]        ;       ES:AX := @ desired DIT
0084'   0E                     528           PUSH    CS
0085'   07                     529           POP     ES
0086'   EB 06                  530           JMP     short ptr SETDIT        ;       Go set DIT address
                               531   ;*****************************************************************************
                               532   ; OPXSET - SET DIT ADDRESS FOR DRIVE
                               533   ;           INPUT:  * DL = Drive number
                               534   ;                   * ZF = NZ if drive number is bad
                               535   ;
        =0088                  536   OPXSET  EQU     $                       ; Set DIT addr to supplied DIT address
0088'   75 C4                  537           JNZ     CMDERR                  ; Return command error if drive # bad
008A'   C4 06 0000"            538           LES     AX,dword ptr D_REQ+BUF  ; ES:AX := @ supplied DIT address
                               539   ;       JMP     short ptr SETDIT        ; And go set DIT address
                               540   ;*****************************************************************************
                               541   ; SETDIT - COMMON CODE FOR OPFSET & OPXSET
                               542   ;           INPUT:  * DL = Drive number
                               543   ;
008E'   30 F6                  544   SETDIT: XOR     DH,DH                   ; Convert drive number to word value
0090'   8B DA                  545           MOV     BX,DX                   ; BX := Word value drive number
0092'   D1 E3                  546           SHL     BX,1                    ; Convert to double word index
0094'   D1 E3                  547           SHL     BX,1                    ;
0096'   89 87 0000"            548           MOV     D_DIT+0[BX],AX          ; Store DIT address for drive
009A'   8C 87 0000"            549           MOV     D_DIT+2[BX],ES          ;
009E'   E8 00C7                550           CALL    CHGD10                  ; Now, go reset the data structures
00A1'   EB CA                  551           JMP     short ptr OKEXIT        ;
                               552   ;*****************************************************************************
                               553   ; OPRSET - RESET DISK SYSTEM
                               554   ;
00A3'   E8 013E                555   OPRSET: CALL    INVPOS                  ; Invalidate all disk positions
00A6'   EB C5                  556           JMP     short ptr OKEXIT
                               557   ;*****************************************************************************
                               558   ; OPKILM - Kill the motor timeout for the floppies
                               559   ;
00A8'   E8 0002"               560   OPKILM: CALL    MOTOFF
                               561   ;*****************************************************************************
                               562                   SUBTTL  ROM disk driver front end: INFORMATION RETURNS
                               563   ;*****************************************************************************
                               564   ; OPSTAT - RETURN STATUS CODE
                               565   ;
00AB'   A0 0000"               566   OPSTAT: MOV     AL,D_REQ+STAT           ; Return last status code
```

```
00AE'  EB 12                    567             JMP     short ptr QEXIT
                                568     ;*******************************************************************************
                                569     ; OPSSTA - RETURN RETRY STATUS
                                570     ;
00B0'  A0 0000"                 571     OPSSTA: MOV     AL,D_REQ+RSTAT          ; Get retry status if any
00B3'  EB 0D                    572             JMP     short ptr QEXIT
                                573     ;*******************************************************************************
                                574     ; OPRDIT - RETURN DIT ADDRESS FOR UNIT
                                575     ;           INPUT:  # [D_REQ+DIT] (dword) = seg:offset of current drive's DIT
                                576     ;                   # ZF = NZ if the drive number is bad
                                577     ;
00B5'  75 97                    578     OPRDIT: JNZ     CMDERR                  ; Return command error if there's error
00B7'  92                       579             XCHG    AX,DX                   ; AL := "New" current drive number
00B8'  98                       580             CBW                             ; Convert to word value
00B9'  D1 E0                    581             SHL     AX,1                    ; DI := Index to DIT table
00BB'  D1 E0                    582             SHL     AX,1                    ;
00BD'  93                       583             XCHG    AX,BX                   ; DI := New drive index
00BE'  C4 9F 0004"              584             LES     BX,dword ptr D_DIT[BX]  ; ES:AX := @ drive's DIT
00C2'  B4 00                    585     QEXIT:  MOV     AH,DKNORM               ; AH := Normal status
00C4'  F8                       586             CLC                             ; Set carry to indicate no error
00C5'  EB 9D                    587             JMP     short ptr POPRET        ; And return
                                588     ;*******************************************************************************
                                589
                                590             SUBTTL  ROM disk driver front end: REJUMP TO EXIT
                                591     ;*******************************************************************************
00C7'  EB 8A                    592     J_EXIT: JMP     short ptr EXIT          ; Rejump to EXIT
                                593     ;*******************************************************************************
                                594
                                595             SUBTTL  ROM disk driver front end: READ, WRITE & VERIFY
                                596     ;*******************************************************************************
                                597     ; INTERNAL FUNCTION PROCESSING ROUTINE
                                598     ; NAME:           DO_IO
                                599     ; ABSTRACT:       Process all disk I/O functions
                                600     ; PROCESSING:     * Interpret indicators from CHGDRV into errors (if any)
                                601     ;                 * Initialize status & retry status bytes to DKNORM
                                602     ;                 * Check for attempt to transfer past 64k boundry error
                                603     ;                 * Perform a disk I/O function
                                604     ;                 * If needed, adjust cylinder, track, sector & buffer address
                                605     ;                   parameters for repeating the I/O function on consecutive
                                606     ;                   sectors & re-issue the I/O request
                                607     ; INPUT:          * DL = Drive number
                                608     ;                 * CF =  C (8 for BSD) if drive is undefined
                                609     ;                 * ZF = NZ if drive is invalid
                                610     ; OUTPUT:         * [D_REQ+STAT] (byte) = Error code if error occurs
                                611     ;                 * [D_REQ+CYL, +TRK, +SEC & +BUF] = modified for multi-sectr I/O
                                612     ; REGISTERS:
                                613     ; STACK USED:
                                614     ;*******************************************************************************
00C9'  B0 01                    615     OPREAD: MOV     AL,RQ_READ              ; AL := Read sector function code
00CB'  EB 0A                    616             JMP     short ptr DO_IO
00CD'  B0 02                    617     OPWRIT: MOV     AL,RQ_WRIT              ; AL := Write sector function code
00CF'  EB 06                    618             JMP     short ptr DO_IO
```

```
00D1' B0 03                619 OPVERF: MOV   AL,RQ_VFC             ; AL := Verify CRC function code
00D3' EB 02                620         JMP   short ptr DO_IO
00D5' B0 04                621 OPVRFY: MOV   AL,RQ_VFD             ; AL := Verify data function code
00D7' A2 0000"             622 DO_IO:  MOV   D_REQ+FUNC,AL         ; Set function request to read or write
00DA' 75 95                623         JNZ   JCMDER                ; Return command error if drive # bad
00DC' 72 7A                624         JB    NRDYER                ; Return not ready error if drive undef
00DE' C6 06 0000" 00       625         MOV   D_REQ+STAT,DKNORM     ; Initialize status to normal
00E3' C6 06 0000" 00       626         MOV   D_REQ+RSTAT,DKNORM    ; Initialize retry status to normal
00E8' C4 1E 0000"          627         LES   BX,dword ptr D_REQ+DIT ; ES:BX := @ DIT for current drive
00EC' A0 0000"             628         MOV   AL,D_REQ+SECKNT       ; AL := number of sectors to transfer
00EF' 08 C0                629         OR    AL,AL                 ; Q: Is that value zero?
00F1' 74 D4                630         JZ    J_EXIT                ;    Y: All right, don't xfer anything
00F3' 30 E4                631         XOR   AH,AH                 ; Convert to unsigned word value
00F5' 26F7 67 04           632         MUL   word ptr ES:DITSEC[BX] ; DX,AX := number of bytes to transfer
00F9' 03 06 0000"          633         ADD   AX,word ptr D_REQ+BUF ; Add buffer offset
00FD' 83 D2 00             634         ADC   DX,0                  ;    (double word)
0100' 4A                   635         DEC   DX                    ; Q: Is that )64k?
0101' 7F 51                636         JQ    OVRMEM                ;    Y: Return 64k boundry error
0103' 7C 04                637         JL    IOLOOP                ;       If ( 64k, then perform the I/O
0105' 09 C0                638         OR    AX,AX                 ; Q: Is it = 64k?
0107' 75 48                639         JNZ   OVRMEM                ;    N: Return boundry error
0109' E8 00E8              640 IOLOOP: CALL  EXECMD                ; Execute the disk command
010C' 75 B9                641         JNZ   J_EXIT                ; Error exit if error occurred
010E' 80 3E 0000" 00       642         CMP   D_REQ+SECKNT,0        ; Q: device dep code xfer all the data?
0113' 74 B2                643         JZ    J_EXIT                ;    Y: Then exit
0115' FE 0E 0000"          644         DEC   D_REQ+SECKNT          ; Q: Have we I/O'd all the sectors?
0119' 74 AC                645         JZ    J_EXIT                ;    Y: Then exit
                           646                                     ;    N: Then increment to next sector:
                           647                                     ;
011B' C4 1E 0000"          648         LES   BX,dword ptr D_REQ+DIT ; ES:BX := @ current DIT
011F' 26 8B 47 04          649         MOV   AX,ES:DITSEC[BX]      ; AX := Sector size
0123' 01 06 0000"          650         ADD   word ptr D_REQ+BUF,AX ; Add to buffer offset
                           651                                     ;
0127' BE 0000"             652         MOV   SI,offset D_REQ+SEC   ; SI := @ req sector field of D_REQ
012A' FE 04                653         INC   byte ptr [SI]         ; Increment requested sector number
012C' 8A 04                654         MOV   AL,byte ptr [SI]      ; AL := new sector number
012E' 26 3A 47 06          655         CMP   AL,ES:DITTRK[BX]      ; Q: Is there such a sector?
0132' 76 D5                656         JBE   IOLOOP                ;    Y: Perform the I/O
                           657                                     ;    N: Increment to the next track:
                           658                                     ;
0134' C6 04 01             659         MOV   byte ptr [SI],1       ; Reset sector number to 1st sector
0137' 4E                   660         DEC   SI                    ; SI := @ req track field of D_REQ
0138' FE 04                661         INC   byte ptr [SI]         ; Increment requested track number
013A' 8A 04                662         MOV   AL,byte ptr [SI]      ; AL := new track number
013C' 26 3A 47 07          663         CMP   AL,ES:DITCYL[BX]      ; Q: Is there such a track?
0140' 72 C7                664         JB    IOLOOP                ;    Y: Perform the I/O
                           665                                     ;    N: Increment to the next cylinder:
                           666                                     ;
0142' C6 04 00             667         MOV   byte ptr [SI],0       ; Zero track number
0145' 4E                   668         DEC   SI                    ; SI := @ req cylinder field of D_REQ
0146' FE 04                669         INC   byte ptr [SI]         ; Increment cylinder number
0148' 8A 04                670         MOV   AL,byte ptr [SI]      ; AL := New cylinder number
```

```
014A'  263A 47 08            671             CMP     AL,ES:DITDSK[BX]        ; Q: Is there such a cylinder?
014E'  72 B9                 672             JB      IOLOOP                  ;    Y: Perform the I/O
0150'  B0 04                 673             MOV     AL,DKRNFE               ;    N: Return a record not found error
0152'  EB 06                 674             JMP     short ptr JEXITE        ;
                             675                                             ;
0154'  B0 09                 676     OVRMEM: MOV     AL,DKBOUN               ; Exit with 64k boundry error
0156'  EB 02                 677             JMP     short ptr JEXITE        ;
                             678                                             ;
0158'  B0 80                 679     NRDYER: MOV     AL,DKNRDY               ; Exit with disk not ready error
015A'  E9 FEF3               680     JEXITE: JMP     EXITE
                             681     ;*****************************************************************************
                             682
                             683             SUBTTL  ROM disk driver front end: CHGDRV (CHANGE CURRENT DRIVE)
                             684     ;*****************************************************************************
                             685     ; INTERNAL ROUTINE
                             686     ; NAME:           CHGDRV
                             687     ; ABSTRACT:       Check the requested drive number for validity and update the
                             688     ;                 disk driver data base, when appropriate, for accessing a new
                             689     ;                 drive (i.e., a different one from the last request).
                             690     ; INPUT:          * DL = New drive number
                             691     ; PROCESSING:     * Check for valid drive number
                             692     ;                 * Store current drive's position
                             693     ;                 * If no "indicators" are set, make new drive the current drive
                             694     ;                   by getting and storing:
                             695     ;                   - New drive number
                             696     ;                   - New drive's current position
                             697     ;                   - New drive's DIT address
                             698     ; OUTPUT:         * BX = Same as input
                             699     ;                 * SI = Same as input
                             700     ;                 * DL = New drive number
                             701     ;                 * CF = NC & ZF =  Z if no errors were found
                             702     ;                 * CF =  C & ZF =  Z If the drive is undefined
                             703     ;                 * CF = NC & ZF = NZ If drive # is invalid (# ( 0 or # ) NUMDRV)
                             704     ; REGISTERS:      AX, CX, DI, ES
                             705     ; STACK USED:     2 bytes
                             706     ; NOTES:          * Flag returns are called "indicators" rather than "errors" be-
                             707     ;                   cause the interpretation of the indicators as errors depends
                             708     ;                   on the command executed. For example, "drive undefined" isn't
                             709     ;                   an error for the "set DIT" functions, but it is for the I/O
                             710     ;                   functions. Therefore, the code for each function decides how
                             711     ;                   to interpret the indicators returned by this routine.
                             712     ;*****************************************************************************
015D'                        713     CHGDRV  PROC    NEAR
015D'  80 FA 00"             714             CMP     DL,NUMDRV-1             ; Q: Is the request for a valid drive?
0160'  77 4A                 715             JA      CHGRET                  ;    N: Return invalid (NC & NZ)
0162'  38 16 0000"           716             CMP     D_REQ+UNIT,DL           ; Q: Is this operation on a new drive?
0166'  74 44                 717             JE      CHGRET                  ;    N: Return OK (NC & Z)
0168'  A0 0000"              718     CHGD10: MOV     AL,D_REQ+UNIT           ; AL := "Old" current drive number
016B'  98                    719             CBW                             ; Convert to word value
016C'  8B F8                 720             MOV     DI,AX                   ; DI := Drive position index
016E'  40                    721             INC     AX                      ; Q: Is the old unit "INVALID"?
016F'  74 08                 722             JZ      DO_NEW                  ;    Y: Then go set for very 1st time
```

```
0171'  8A 0E 0000"            723             MOV     CL,D_REQ+CURCYL         ; CL := Current drive's position
0175'  88 8D 0006"            724             MOV     D_POS[DI],CL            ; Store current drive's position
0179'                         725     DO_NEW:                                 ;
                              726
0179'  8A CA                  727             MOV     CL,DL                   ; CL := Shift count for installed test
017B'  A0 000C"               728             MOV     AL,BYTE PTR SYSCON      ; AL := Installed bit for floppy drives
017E'  0C F0                  729             OR      AL,01111000B            ; Force all other drives to "installed"
0180'  D2 E8                  730             SHR     AL,CL                   ; Move installed bit to bit 0
0182'  A8 01                  731             TEST    AL,1                    ; Q: Is drive installed?
0184'  F9                    732             STC                             ;       Set CF (C) in case of error
0185'  74 25                  733             JZ      CHGRET                  ;    N: Return undefined (C & Z)
0187'  8A C2                  734             MOV     AL,DL                   ; AL := "New" current drive number
                              735     ;       CBW                             ; Convert to word value
0189'  97                     736             XCHG    AX,DI                   ; DI := New drive index
018A'  8A AD 0006"            737             MOV     CH,D_POS[DI]            ; CH := New drive's position
018E'  D1 E7                  738             SHL     DI,1                    ; DI := Index to DIT table
0190'  D1 E7                  739             SHL     DI,1                    ;
0192'  C4 85 0004"            740             LES     AX,dword ptr D_DIT[DI]  ; ES:AX := Q drive's DIT
0196'  09 C0                  741             OR      AX,AX                   ; Q: Is drive's DIT defined (( ) 0)?
0198'  F9                    742             STC                             ;       Set CF (C) in case of error
0199'  74 11                  743             JZ      CHGRET                  ;    N: Return undefined (C & Z)
019B'  88 16 0000"            744             MOV     D_REQ+UNIT,DL           ;    Y: Set new unit number as current
019F'  88 2E 0000"            745             MOV     D_REQ+CURCYL,CH         ;       Set new position as current
01A3'  A3 0000"               746             MOV     word ptr D_REQ+DIT+0,AX ;       Set new DIT address as current
01A6'  8C 06 0000"            747             MOV     word ptr D_REQ+DIT+2,ES ;
01AA'  31 C0                  748             XOR     AX,AX                   ; Return OK (NC & Z)
01AC'  C3                     749     CHGRET: RET                             ; Insure carry clear
                              750     ;*****************************************************************************
                              751
                              752             SUBTTL  ROM disk driver front end: DCALL (CALL QDO ROUTINE)
                              753     ;*****************************************************************************
                              754     ; PUBLIC ROUTINE
                              755     ; NAME:           DCALL
                              756     ; ABSTRACT:       Called by the interrupt level code to invoke device-dependent
                              757     ;                 General Disk Operation routines
                              758     ; PROCESSING:     * Converts the NEAR call to this routine into a FAR call
                              759     ;                 * Set BP register to the offset of the disk data area in ROMDAT
                              760     ;                 * Invokes a device-dependent Generic Disk Operation subroutine
                              761     ; INPUT:          * AL = Code for operation (passed on to interface routine)
                              762     ;                 * [D_REQ+DIT] (dword) = seg:offset of the Disk Interface Table
                              763     ; OUTPUT:         none from this routine
                              764     ; REGISTERS:      AX, BX, DX, DI, ES plus that used by the called routine
                              765     ; STACK USED:     This routine does not use any stack, it converts the NEAR call
                              766     ;                 to this routine to a FAR call, placing 4 bytes of return addr
                              767     ;                 information on the stack. However, the four bytes are included
                              768     ;                 in the stack usage of the invoked routines. Invoking floppy
                              769     ;                 QDO routines will use up to 12 bytes of stack.
                              770     ;*****************************************************************************
01AD'                         771     DCALL   PROC    NEAR
01AD'  5A                     772             POP     DX                      ; DX := return address
01AE'  0E                     773             PUSH    CS                      ; Push current CS
01AF'  52                     774             PUSH    DX                      ; Push return address
```

```
                                775                                     ; Call converted from NEAR to FAR
01B0'  C4 1E 0000"              776             LES     BX,dword ptr D_REQ+DIT  ; ES,BX := @ Disk Interface Table
01B4'  BD 0003"                 777             MOV     BP,offset D_DBGN        ; BP := base address of disk data area
01B7'  26FF 2F                  778             JMP     dword ptr ES:[BX]       ; Transfer to Disk Interface Routine
                                779     ;*******************************************************************************
                                780
                                781             SUBTTL  ROM disk driver front end: DFINISH (SIGNAL DISK DONE)
                                782     ;*******************************************************************************
                                783     ; PUBLIC ROUTINE
                                784     ; NAME:           DFINISH
                                785     ; ABSTRACT:       * Called by interrupt level code to indicate that it's done
                                786     ; PROCESSING:     * Set the disk driver wait flag to "READY"
                                787     ; INPUT:          none
                                788     ; OUTPUT:         * [D_WAIT] (byte) = "READY"
                                789     ; REGISTERS:      none
                                790     ; STACK USED:     2 bytes
                                791     ;*******************************************************************************
01BA'                           792     DFINISH PROC    NEAR
01BA'  C6 06 000A" 00           793             MOV     D_WAIT,READY            ; Clear the D_WAIT flag
01BF'  C3                       794             RET                             ; And return
                                795     ;*******************************************************************************
                                796
                                797             SUBTTL  ROM disk driver front end: DSKINI (DRIVER INITIALIZATION)
                                798     ;*******************************************************************************
                                799     ; PUBLIC ROUTINE
                                800     ; NAME:           DSKINI
                                801     ; ABSTRACT:       Initialize the disk data area
                                802     ; PROCESSING:     * Initialize the disk driver data area in ROMDAT
                                803     ; INPUT:          none
                                804     ; OUTPUT:         none
                                805     ; REGISTERS:      AX, BX, CX, SI, DI, ES
                                806     ; STACK USED:     2 bytes
                                807     ;*******************************************************************************
01C0'                           808     DSKINI  PROC    NEAR
01C0'  1E                       809             PUSH    DS                      ; ES := DS
01C1'  07                       810             POP     ES                      ;
01C2'  B8 0003"                 811             MOV     AX,offset D_DBGN        ; AX := @ beginning of disk data area
01C5'  8B F8                    812             MOV     DI,AX                   ; Copy that to DI
01C7'  B9 0000"                 813             MOV     CX,offset D_DEND+1      ; CX := just past end of disk data area
01CA'  29 C1                    814             SUB     CX,AX                   ; CX := Length of disk data area
01CC'  30 C0                    815             XOR     AL,AL                   ; AL := 0 for initialization
01CE'  F2                       816             REP                             ; Repeat:
01CF'  AA                       817             STOS    ES:D_DBGN               ; Zero disk data area
                                818                                             ; Initialize defaults:
                                819     ;       MOV     D_STAT,S_IDLE           ; Initialize driver state to "idle"
                                820                                             ; Note that S_IDLE = 0
01D0'  B9 0004                  821             MOV     CX,NUMFLP               ; Initialize table of DIT pointers
01D3'  31 DB                    822             XOR     BX,BX                   ;   for floppy disk drives
01D5'  C7 87 0000" 0220"        823     INIDIT: MOV     D_DIT+00[BX],offset SSSTD
01D9'  8C 8F 0000"              824             MOV     D_DIT+02[BX],CS
01DF'  83 C3 04                 825             ADD     BX,4
01E2'  E2 F1                    826             LOOP    INIDIT
```

```
                                827   ;*****************************************************************************
                                828   ; INTERNAL ROUTINE
                                829   ; NAME:           INVPOS
                                830   ; ABSTRACT:       Invalidate current disk positions and current drive. This
                                831   ;                 resets the disk system. The invalid disk position tells the
                                832   ;                 device specific code that a reset was requested.
                                833   ; PROCESSING:     * Set all disk current positions to "INVALID" (-1)
                                834   ;                 * Set current drive unit number to "INVALID" (-1)
                                835   ; INPUT:          none
                                836   ; OUTPUT:         * [D_POS] (byte array) = "INVALID"
                                837   ;                 * [D_REQ+UNIT] (byte) = "INVALID"
                                838   ; REGISTERS:      BX, CX
                                839   ; STACK USED:     2 bytes
                                840   ;*****************************************************************************
01E4' B9 000B*                  841   INVPOS: MOV       CX,NUMDRV                   ; CX := number of disk drives
01E7' 31 DB                     842           XOR       BX,BX                       ; BX := 0 (initialial index)
01E9' C6 87 0006" FF            843   INIPOS: MOV       D_POS[BX],INVALID           ; Put in the "nowhere" value
01EE' 43                        844           INC       BX
01EF' E2 F8                     845           LOOP      INIPOS
01F1' C6 06 0000" FF            846           MOV       D_REQ+UNIT,INVALID          ; Also invalidate the current unit
01F6' C3                        847           RET
                                848
                                849           SUBTTL  ROM disk driver front end: EXECMD (EXEC CMD & WAIT)
                                850   ;*****************************************************************************
                                851   ; INTERNAL ROUTINE
                                852   ; NAME:           EXECMD
                                853   ; ABSTRACT:       Invoke the disk Interrupt Service Routine to process a disk
                                854   ;                 request and wait for it to finish.
                                855   ; PROCESSING:     * Set the disk driver's wait flag to "BUSY"
                                856   ;                 * Initiate interrupt processing using the SOFINT routine
                                857   ;                 * Wait for wait flag to be set to "READY"
                                858   ;                 * Check the status byte for any error indication
                                859   ; INPUT:          none
                                860   ; OUTPUT:         * ZF = NZ if an error occurred during interrupt processing
                                861   ; REGISTERS:      none
                                862   ; STACK USED:     10 bytes
                                863   ;*****************************************************************************
01F7'                           864   EXECMD  PROC      NEAR
01F7' C6 06 000A" 01            865           MOV       D_WAIT,BUSY                 ; Set wait flag to "busy"
01FC' E8 0012                   866           CALL      SOFINT                      ; Initiate interrupt-driven code
01FF' 80 3E 000A" 00            867   WAITER: CMP       D_WAIT,READY                ; Wait for it to finish
0204' 75 F9                     868           JNE       WAITER
0206' 80 3E 0000" 00            869           CMP       D_REQ+STAT,DKNORM           ; Reset Z-flag (NZ condition) if error
020B' C3                        870           RET                                   ; And return
                                871   ;*****************************************************************************
                                872
                                873           SUBTTL  ROM disk driver front end: TIMOUT (TIME OUT DISK I/O)
                                874   ;*****************************************************************************
                                875   ; PUBLIC ROUTINE
                                876   ; NAME:           TIMOUT
                                877   ; ABSTRACT:       Called by the ROM timing services routine to signal a timeout
                                878   ; PROCESSING:     * Store the timeout error code in the disk driver status area
```

```
                                879   ;               * Invoke the disk Interrupt Service Routine via soft interrupt
                                880   ; INPUT:        none
                                881   ; OUTPUT:       * [D_REQ+STAT] (byte) = "DKNRDY" (timeout error)
                                882   ; REGISTERS:    none
                                883   ; STACK USED:   8 bytes, 6 for interrupt & 2 for storing the interrupted DS
                                884   ;*****************************************************************************
020C'                           885   TIMOUT  PROC    NEAR
020C'  C6 06 0000" 80           886           MOV     D_REQ+STAT,DKNRDY       ; Set status to timeout
                                887   ;       JMP     short ptr SOFINT        ; Invoke a software interrupt
                                888   ;*****************************************************************************
                                889
                                890           SUBTTL  ROM disk driver front end: SOFINT (SOFT INTERRUPT TO DSKISR)
                                891   ;*****************************************************************************
                                892   ; PUBLIC ROUTINE
                                893   ; NAME:         SOFINT
                                894   ; ABSTRACT:     Called by DSKIO routines and the FLTIME (motor-off) routine in
                                895   ;               FLPDIR to enter the interrupt-driven part of the disk driver.
                                896   ;               This particular subroutine may only be called by routines in
                                897   ;               the ROMCOD data segment, add-on ROMs will need their own copy.
                                898   ; PROCESSING:   * Set the soft interrupt indicator. NOTE: This is ABSOLUTELY
                                899   ;                 essential for the correct processing of a software interrupt.
                                900   ;                 If we don't, the interrupt driven logic will tell the system
                                901   ;                 interrupt controller that it has processed an interrupt which
                                902   ;                 the controller knows nothing about (the soft interrupt). This
                                903   ;                 would cause strange, but not wonderful things to happen.
                                904   ;               * Execute a software interrupt to the the disk driver.
                                905   ; INPUT:        none
                                906   ; OUTPUT:       * [D_SOFT] (byte) = 1 indicating a software interrupt issued
                                907   ; REGISTERS:    none
                                908   ; STACK USED:   8 bytes, 6 for interrupt & 2 for storing the interrupted DS
                                909   ;*****************************************************************************
0211'                           910   SOFINT  PROC    NEAR                    ; General Software Interrupt routine
0211'  FE 06 0008"             911           INC     D_SOFT                  ; Make software interrupt flag non-0
0215'  CD 46                   912           INT     IR6INT                  ; Invoke Disk Interrupt Service Routine
0217'  C3                      913           RET                             ; And return
                                914   ;*****************************************************************************
                                915
                                916           SUBTTL  ROM disk driver front end: STANDARD DITS TABLE
                                917   ;*****************************************************************************
                                918   ;                               FIXED DATA TABLES
                                919   ;*****************************************************************************
0218'                           920   DITTBL  LABEL   WORD                    ; TABLE OF STANDARD DIT'S (POINTERS)
0218'  0220"                   921           DW      SSSTD                   ; 0: Single-side, single track density
021A'  022B"                   922           DW      DSSTD                   ; 1: Double-side, single track density
021C'  0236"                   923           DW      SSDTD                   ; 2: Single-side, double track density
021E'  0241"                   924           DW      DSDTD                   ; 3: Double-side, double track density
                                925   ;*****************************************************************************
0220'                           926   SSSTD   LABEL   WORD                    ; SINGLE-SIDED, SINGLE TRACK DENSITY
0220'  0001" 0000"             927           DD      FLPDIR                  ; Floppy disk Interface Routine vector
0224'  0200                    928           DW      XXSEC                   ; Sector size in bytes
0226'  08                      929           DB      XXTRK                   ; Track size in sectors
0227'  01                      930           DB      SSCYL                   ; Cylinder size in tracks
```

```
0228'  28                      931           DB      STDSK                   ; Disk size in cylinders
0229'  02                      932           DB      FLPERR                  ; Maximum number of retries
022A'  14                      933           DB      STPRCMP                 ; Cylinder at which to change pre-comp
                               934   ;*****************************************************************************
022B'                          935   DSSTD   LABEL   WORD                    ; DOUBLE-SIDED, SINGLE TRACK DENSITY
022B'  0001" 0000"             936           DD      FLPDIR                  ; Floppy disk Interface Routine vector
022F'  0200                    937           DW      XXSEC                   ; Sector size in bytes
0231'  08                      938           DB      XXTRK                   ; Track size in sectors
0232'  02                      939           DB      DSCYL                   ; Cylinder size in tracks
0233'  28                      940           DB      STDSK                   ; Disk size in cylinders
0234'  02                      941           DB      FLPERR                  ; Maximum number of retries
0235'  14                      942           DB      STPRCMP                 ; Cylinder at which to change pre-comp
                               943   ;*****************************************************************************
0236'                          944   SSDTD   LABEL   WORD                    ; SINGLE-SIDED, DOUBLE TRACK DENSITY
0236'  0001" 0000"             945           DD      FLPDIR                  ; Floppy disk Interface Routine vector
023A'  0200                    946           DW      XXSEC                   ; Sector size in bytes
023C'  08                      947           DB      XXTRK                   ; Track size in sectors
023D'  01                      948           DB      SSCYL                   ; Cylinder size in tracks
023E'  50                      949           DB      DTDSK                   ; Disk size in cylinders
023F'  02                      950           DB      FLPERR                  ; Maximum number of retries
0240'  28                      951           DB      DTPRCMP                 ; Cylinder at which to change pre-comp
                               952   ;*****************************************************************************
0241'                          953   DSDTD   LABEL   WORD                    ; DOUBLE-SIDED, DOUBLE TRACK DENSITY
0241'  0001" 0000"             954           DD      FLPDIR                  ; Floppy disk Interface Routine vector
0245'  0200                    955           DW      XXSEC                   ; Sector size in bytes
0247'  08                      956           DB      XXTRK                   ; Track size in sectors
0248'  02                      957           DB      DSCYL                   ; Cylinder size in tracks
0249'  50                      958           DB      DTDSK                   ; Disk size in cylinders
024A'  02                      959           DB      FLPERR                  ; Maximum number of retries
024B'  28                      960           DB      DTPRCMP                 ; Cylinder at which to change pre-comp
                               961   ;*****************************************************************************
                               962           SUBTTL
                               963           END

No errors detected
```

```
                          1   ;/*************************************************************************/
                          2   ;/* TITLE    - DSKISR                                                     */
                          3   ;/* COMPUTER - 8088 C                                                     */
                          4   ;/* ABSTRACT - This is the drive independent, interrupt driven portion of */
                          5   ;/*            the PEGASUS disk driver.                                   */
                          6   ;/*                                                                       */
                          7   ;/*            All C code translated to assembly by the author. Some of the */
                          8   ;/*            routines are expanded in line - in particular the interrupt */
                          9   ;/*            overhead and entry/exit. This code probably only vaguely   */
                         10   ;/*            resembles what a real C compiler would generate since I    */
                         11   ;/*            could make use of special knowledge such as what registers */
                         12   ;/*            were in use. The C code reflects the actual logic of the   */
                         13   ;/*            program but probably cannot translate directly into a working */
                         14   ;/*            program without system-dependent re-writing.               */
                         15   ;/*                                                                       */
                         16   ;/* REGISTERS USED - All registers are restored intact                    */
                         17   ;/*                                                                       */
                         18   ;/* STACK USED - at least 24 bytes, probably more.                        */
                         19   ;/*                                                                       */
                         20   ;/*************************************************************************/
                         21           NAME    DSKISR
                         22   ;#include"peg:dskerr.ceq"                          /*Include disk error codes*/
                         60   ;#include"peg:dskop.ceq"                 /*Include disk generic operation codes*/
                         80   ;#include"peg:dskreq.ceq"                          /*Include disk requests*/
                         91   ;#include"peg:dsksta.ceq"                           /*Include disk states*/
                        103   ;                                                        /*Miscellaneous*/
                        104   ;#define RETRIES                 2              /*Max retries on an operation*/
=0002                   105   RETRIES EQU     2
                        106   ;
                        107   ;#define NULL                    0                        /*The null value*/
=0000                   108   NULL    EQU     0
                        109   ;#define NO                      0               /*Boolean value 'no' (false)*/
=0000                   110   NO      EQU     0
                        111   ;#define YES                     1               /*Boolean value 'yes' (true)*/
=0001                   112   YES     EQU     1
                        113   ;
                        114   ;*************************************************************************
                        115   ;       INCLUDE PEQ:INTCTLR.EQU
                        116   ;       INCLUDE PEQ:VECTOR.EQU
                        117   ;       INCLUDE PEQ:TIMEVT.EQU
                        118
                        257   ;*************************************************************************
=0080                   258   ACTEVT  EQU     1 shl EVTSTA            ; Mask for setting bit to activate evnt
=0032                   259   TIMITV  EQU     50                      ; Timeout interval = 1.25 seconds
                        260   ;/*************************************************************************/
                        261   ;#define BYTE char
                        262   ;#define BOOL char
                        263   ;#define ADDR long
                        264   ;
                        265   ;#include"peg:dskrib.ceq"                          /*Include defintion for RIB*/
                        281   ;
=0000                   282           SECT    ROMDAT
```

```
                        283   ;BYTE   D_EKNT ;                                  /*Counter for error retries*/
                        284           EXTRN   D_EKNT:BYTE
                        285   ;BYTE   D_NCMD ;                                       /*Next disk operation*/
                        286           EXTRN   D_NCMD:BYTE
                        287   ;BYTE   D_NSTA ;                                       /*Next state to enter*/
                        288           EXTRN   D_NSTA:BYTE
                        289   ;RIB    D_REQ ;                                   /*Request Information Block*/
                        290           EXTRN   D_REQ:BYTE
                        291   ;BOOL   D_SOFT ;                                    /*Software interrupt flag*/
                        292           EXTRN   D_SOFT:BYTE
                        293   ;BYTE   D_STAT ;                                    /*Current state of driver*/
                        294           EXTRN   D_STAT:BYTE
                        295           EXTRN   DSKEVT:BYTE                     ; Disk driver's timeout event
                        296           EXTRN   DSKISP:WORD                     ; Contains disk interrupt SP
                        297           EXTRN   DKSSSV:WORD                     ; For saving interrupted SS
                        298           EXTRN   DKSPSV:WORD                     ; For saving interrupted SP
                        299
=0000                   300           SECT    ROMCOD
                        301           ASSUME  CS:ROMCOD,DS:ROMDAT
                        302   ;
                        303           EXTRN   DCALL:NEAR
                        304           EXTRN   DFINIS:NEAR
                        305   ;
                        306   ;DSKISR()                        /*Interrupt received. Dispatch according to state*/
                        307           PUBLIC  DSKISR
0000'                   308   DSKISR  PROC    FAR
                        309           LOCAL
                        310   ;
                        311   ;{
                        312   ;global BOOL  D_BSY ;            /*Indicates driver is processing an interrupt*/
                        313   ;global RIB   D_REQ ;                                 /*Disk driver request block*/
                        314   ;global BYTE  D_EKNT ;                                /*Counter for error retries*/
                        315   ;global BYTE  D_NCMD ;                                     /*Next disk operation*/
                        316   ;global BYTE  D_NSTA ;                                     /*Next state to enter*/
                        317   ;global BYTE  D_STAT ;                                  /*Current state of driver*/
                        318   ;global BOOL  D_TIM ;                                             /*Timeout flag*/
                        319   ;
                        320   ;INTERRUPT_ENTRY() ;      /*Do system overhead things for interrupt processing*/
0000'  1E               321           PUSH    DS                              ;;; Save current DS on user stack
0001'  2EFE 06 C19A     322           INC     byte ptr CS:INTCTR+CSWRAP       ;;; Increment interrupt counter
0006'  2E8E 1E C180     323           MOV     DS, word ptr CS:DSADDR+CSWRAP   ;;; DS := @ ROM data segment
000B'  8C 16 0009"      324           MOV     DKSSSV,SS                       ;;; Save current SS
000F'  89 26 000A"      325           MOV     DKSPSV,SP                       ;;; Save current SP
0013'  8C DC            326           MOV     SP,DS                           ;;; SS := @ ROM data/stack seg
0015'  8E D4            327           MOV     SS,SP                           ;;;
0017'  BC 0008"         328           MOV     SP,offset DSKISP                ;;; SP := interrupt stack ptr
001A'  50               329           PUSH    AX                              ;;; Save registers used for
001B'  53               330           PUSH    BX                              ;;; interrupt overhead processing
001C'  06               331           PUSH    ES                              ;;;
001D'  51               332           PUSH    CX                              ;;; Save everything else that
001E'  52               333           PUSH    DX                              ;;; might get used also
001F'  56               334           PUSH    SI                              ;;;
```

```
0020'  57                      335          PUSH    DI                              ;;;
0021'  E4 19                   336          IN      AL,INTA01                       ;;; Mask level 6 interrupts
0023'  0C 40                   337          OR      AL,not ENAB6                    ;;;
0025'  E6 19                   338          OUT     INTA01,AL                       ;;;
                               339  ;
0027'  80 26 0000" 7F          340          AND     DSKEVT+IEVFLG,not ACTEVT        ;;;Abort timeout request
002C'  C7 06 0000" 0032        341          MOV     word ptr DSKEVT+IEVCTR,TIMITV   ;;;Reset Timer count
0032'  FB                      342          STI                                     ;  Enable interrupts
                               343  ;
0033'  80 3E 0000" 00          344          CMP     D_REQ+STAT,E_NORM               ;Q:  Timeout?
0038'  75 0C                   345          JNE     20$                             ; Y: Jump, don't get status.
003A'  80 3E 0006" 00          346          CMP     D_STAT,S_IDLE                   ;Q:  Beginning a request?
003F'  74 05                   347          JE      20$                             ; Y: Jump, don't get status.
0041'  B0 08                   348          MOV     AL,OP_RIS                       ;Get status of last operation.
0043'  E8 000B"                349          CALL    DCALL                           ;
       =0046                   350  20$     EQU     $                               ;
                               351  ;
0046'  C6 06 0003" 00          352          MOV     D_NSTA,NULL                     ;Clear next state indicator
004B'  C6 06 0002" FF          353          MOV     D_NCMD,OP_NUL                   ;Clear next command indicator
                               354  ;
0050'  8A 1E 0006"             355          MOV     BL,D_STAT                       ;Dispatch state
0054'  30 FF                   356          XOR     BH,BH                           ;Get state.
0056'  01 DB                   357          ADD     BX,BX                           ;Convert to a word table index.
0058'  80 3E 0006" 03          358          CMP     D_STAT,S_MAX                    ;Q:  Is this one defined?
005D'  77 40                   359          JA      L1DEF1                          ; N: Go Handle as an aux state.
005F'  80 3E 0000" 00          360          CMP     D_REQ+STAT,E_NORM               ; Ask this once for all states
                               361                                                  ; which need to know
0064'  2EFF A7 0069"           362          JMP     CS:40$[BX]                      ;
0069'  0071"                   363  40$     DW      L1IDLE                          ;
006B'  0076"                   364          DW      L1WAIT                          ;
006D'  007B"                   365          DW      L1SEEK                          ;
006F'  008F"                   366          DW      L1DONE                          ;
                               367  ;
       =0071                   368  L1IDLE  EQU     $                               ;Waiting for a request
0071'  B0 09                   369          MOV     AL,OP_BGN                       ;Go do any pre-request setup
0073'  E8 000B"                370          CALL    DCALL                           ;
       =0076                   371  L1WAIT  EQU     $                               ;Waiting to retry a request
0076'  E8 008D                 372          CALL    REQ_PR                          ;Then process the request
0079'  EB 29                   373          JMP     short ptr L1END1                ;and finish up
                               374  ;
       =007B                   375  L1SEEK  EQU     $                               ;Implicit seek pending
                               376  ;       CMP     D_REQ+STAT,E_NORM               ;Q:  Error from seek operation?
007B'  75 0B                   377          JNE     60$                             ; Y: jump and handle it.
007D'  A0 0000"                378          MOV     AL,D_REQ+CYL                    ; N: Set new current cylinder
0080'  A2 0000"                379          MOV     D_REQ+CURCYL,AL                 ;
0083'  E8 0080                 380          CALL    REQ_PR                          ;And process pending I/O request
0086'  EB 1C                   381          JMP     short ptr L1END1                ;
0088'  B0 02                   382  60$:    MOV     AL,OP_EP                        ;Process error.
008A'  E8 000B"                383          CALL    DCALL                           ;
008D'  EB 15                   384          JMP     short ptr L1END1                ;And finish up
                               385  ;
       =008F                   386  L1DONE  EQU     $                       ;General operation completion pending
```

```
                                387     ;       CMP     D_REQ+STAT,E_NORM       ;Q:  Any errors from operation?
008F'  75 07                    388             JNE     80$                     ; Y:  Jump and handle them.
0091'  C6 06 0003" 00           389             MOV     D_NSTA,S_IDLE           ; N:  Then we're finished
0096'  EB 0C                    390             JMP     short ptr L1END1        ;
0098'  B0 02                    391     80$:    MOV     AL,OP_EP                ;If there was an error
009A'  E8 000B"                 392             CALL    DCALL                   ;Process it
009D'  EB 05                    393             JMP     short ptr L1END1        ;
                                394     ;
       =009F                    395     L1DEF1  EQU     $                       ;Driver is in a device-dependent state
009F'  B0 01                    396             MOV     AL,OP_ASP               ;Invoke device-dependent
00A1'  E8 000B"                 397             CALL    DCALL                   ; auxiliary state processor
                                398     ;
       =00A4                    399     L1END1  EQU     $                       ;
00A4'  80 3E 0002" FF           400             CMP     D_NCMD,OP_NUL           ;Q:  New command to execute?
00A9'  74 06                    401             JE      110$                    ; N:  Jump and continue
00AB'  A0 0002"                 402             MOV     AL,D_NCMD               ; Y:  Go execute it
00AE'  E8 000B"                 403             CALL    DCALL                   ;
       =00B1                    404     110$    EQU     $                       ;
                                405     ;
00B1'  80 3E 0003" 00           406             CMP     D_NSTA,S_IDLE           ;Q:  Are we finished with op?
00B6'  75 0F                    407             JNE     140$                    ; N:  Jump and continue
00B8'  C6 06 0001" 00           408             MOV     D_EKNT,NULL             ; Y:  Do termination processing
00BD'  B0 0A                    409             MOV     AL,OP_TER               ;
00BF'  E8 000B"                 410             CALL    DCALL                   ;
00C2'  E8 000C"                 411             CALL    DFINISH                 ;Signal completion of the request
00C5'  EB 05                    412             JMP     short ptr 150$          ;
00C7'                           413     140$:
00C7'  80 0E 0000" 80           414             OR      DSKEVT+IEVFLG,ACTEVT    ;Start timeout event
       =00CC                    415     150$    EQU     $                       ;
                                416     ;
00CC'  A0 0003"                 417             MOV     AL,D_NSTA               ;Update state
00CF'  A2 0006"                 418             MOV     D_STAT,AL               ;
                                419     ;
00D2'  FA                       420             CLI                             ;;;Mask interrupts at processor level
                                421     ;                                       ;;;Exit from interrupt processing
00D3'  E4 19                    422             IN      AL,INTA01               ;;; Enable level 6 interrupts
00D5'  24 BF                    423             AND     AL,ENAB6                ;;;
00D7'  E6 19                    424             OUT     INTA01,AL               ;;;
00D9'  5F                       425             POP     DI                      ;;; Restore registers
00DA'  5E                       426             POP     SI                      ;;;
00DB'  5A                       427             POP     DX                      ;;;
00DC'  59                       428             POP     CX                      ;;;
00DD'  FE 0E 0005"              429             DEC     D_SOFT                  ;;; Q:  Is software interrupt?
00E1'  74 11                    430             JZ      SOFOUT                  ;;;     Y:  Take software int exit
00E3'  FE 06 0005"              431             INC     D_SOFT                  ;;;     N:  Restore flag to 0
00E7'  8E 06 0009"              432             MOV     ES,DKSSSV               ;;;         ES := interrupted SS
00EB'  8B 1E 000A"              433             MOV     BX,DKSPSV               ;;;         BX := interrupted SP
00EF'  2EFF 2E C164*            434             JMP     dword ptr CS:XITVEC*4+CSWRAP ;;;        Common interrupt exit
                                435                                             ;;;
00F4'  2EFE 0E C19A             436     SOFOUT: DEC     byte ptr CS:INTCTR+CSWRAP ;;; Decrement interrupt counter
00F9'  07                       437             POP     ES                      ;;; Restore interrupted ES
00FA'  5B                       438             POP     BX                      ;;; Restore interrupted BX
```

```
00FB'  58                    439          POP     AX                          ;;; Restore interrupted AX
00FC'  8B 26 000A"           440          MOV     SP,DK$P8V                   ;;; Restore interrupted SP
0100'  8E 16 0009"           441          MOV     SS,DK$SSV                   ;;; Restore interrupted SS
0104'  1F                    442          POP     DS                          ;;; Restore interrupted DS
0105'  CF                    443          IRET                                ;;; And return from the interrupt
                             444  ;/*****************************************************************************/
0106'                        445  REQ_PR  PROC    NEAR                    ;Process a user request
                             446          LOCAL
                             447  ;
                             448  ;global BYTE  D_NCMD ;                        /*Disk driver next command*/
                             449  ;global BYTE  D_NSTA ;                         /*Disk driver next state*/
                             450  ;global RIB   D_REQ ;                        /*Disk driver request block*/
                             451  ;
                             452  ;switch (D_REQ.FUNC) {                  /*Then dispatch request to handler*/
0106'  8A 1E 0000"           453          MOV     BL,D_REQ+FUNC             ; NOTE: We are trusting that bad
010A'  30 FF                 454          XOR     BH,BH                     ; function codes will never get
010C'  01 DB                 455          ADD     BX,BX                     ; this far
010E'  2EFF A7 0113"         456          JMP     CS:10$[BX]
0113'  0127"                 457  10$     DW      L$SEEK
0115'  012C"                 458          DW      L$READ
0117'  0131"                 459          DW      L$WRIT
0119'  011D"                 460          DW      L$VFC
011B'  0122"                 461          DW      L$VFD
                             462  ;
                             463  ;  case RQ_VFC  : D_NCMD = OP_VFC ; D_NSTA = S_DONE ; break ;
011D'  B8 0306               464  L$VFC:  MOV     AX,OP_VFC or (S_DONE shl 8)
0120'  EB 12                 465          JMP     short ptr COMMON
                             466  ;
                             467  ;  case RQ_VFD  : D_NCMD = OP_VFD ; D_NSTA = S_DEP  ; break ;
0122'  B8 FF07               468  L$VFD:  MOV     AX,OP_VFD or (S_DEP shl 8)
0125'  EB 0D                 469          JMP     short ptr COMMON
                             470  ;
                             471  ;  case RQ_SEEK : D_NCMD = OP_NUL ; D_NSTA = S_IDLE ; break ;
0127'  B8 00FF               472  L$SEEK: MOV     AX,OP_NUL or (S_IDLE shl 8)
012A'  EB 08                 473          JMP     short ptr COMMON
                             474  ;
                             475  ;  case RQ_READ : D_NCMD = OP_IR  ; D_NSTA = S_DEP  ; break ;
012C'  B8 FF03               476  L$READ: MOV     AX,OP_IR or (S_DEP shl 8)
012F'  EB 03                 477          JMP     short ptr COMMON
                             478  ;
                             479  ;  case RQ_WRITE : D_NCMD = OP_IW ; D_NSTA = S_DEP  ; break ;
0131'  B8 FF05               480  L$WRIT: MOV     AX,OP_IW or (S_DEP shl 8)
                             481  ;       JMP     short ptr COMMON
                             482  ;
0134'  A2 0002"              483  COMMON: MOV     D_NCMD,AL
0137'  88 26 0003"           484          MOV     D_NSTA,AH
                             485  ;
013B'  8A 16 0000"           486          MOV     DL,D_REQ+CURCYL         ;G;  Is Current cylinder correct?
013F'  3A 16 0000"           487          CMP     DL,D_REQ+CYL            ;
0143'  74 0A                 488          JE      40$                     ; Y; Jump and return to process req.
0145'  C6 06 0002" 04        489          MOV     D_NCMD,OP_IS            ; N; Change request to Seek.
014A'  C6 06 0003" 02        490          MOV     D_NSTA,S_ISEEK
```

```
     014F'  C3                  491   40$:    RET
                                492   ;/*****************************************************************************/
                                493           END
```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ; TITLE    - DSKTST
                        3   ; COMPUTER - 8088 ASSEMBLY LANGUAGE
                        4   ; ABSTRACT - Quick test of floppy disk sector buffer and controller hardware
                        5   ;            for power-up diagnostics. This version runs all the tests and
                        6   ;            sets bits for each failed test
                        7   ; REGISTERS USED - AX, BL, CX, DX
                        8   ; STACK USED - 4 Bytes maximum
                        9   ;*****************************************************************************
                       10           NAME    DSKTST
                       11           SUBTTL
=FFFF                  12   DEBUG   EQU     0FFFFH
                       13           SUBTTL  EXTERNALS and PUBLICS
                       14   ;*****************************************************************************
                       15           PUBLIC  DSKTST
                       16           EXTERN  CRTTST:NEAR             ;   The beginning of the CRT tests
                       17           EXTERN  DECLED:NEAR             ;   Decrement diagnostic LED pattern
                       18           EXTERN  DRVTST:NEAR             ;   Checks for connected drives
                       19           EXTERN  DSKINI:NEAR             ;   Disk driver initialization
                       20           EXTERN  DSPDIE:NEAR             ;   Display and die routine
                       21           EXTERN  E_CMD:ABS               ;   Command execution error
                       22           EXTERN  E_EVEN:ABS              ;   Even-order bit in secbuf bad error
                       23           EXTERN  E_ODD:ABS               ;   Odd-order bit in secbuf bad error
                       24           EXTERN  E_REG:ABS               ;   FDC register bad error
                       25           EXTERN  LED011:ABS              ;   Disk diagnostic LED pattern
                       26           EXTERN  TSTPAT:BYTE             ;   Table of test patterns for regs.
=0000                  27           SECT    ROMDAT
                       28           EXTERN  DSKC_M:BYTE             ;   RAM copy of disk control port
                       29           EXTERN  DSKS_M:BYTE             ;   RAM copy of disk select port
                       30           SUBTTL  Global equates
                       31   ;*****************************************************************************
                       32   ;       INCLUDE PEQ:PORTADDR.EQU
                       33   ;       INCLUDE PEQ:LATCHES.EQU
                       34   ;       INCLUDE PEQ:FLOPPY.EQU
                       35
                      134           SUBTTL  Local equates
                      135   ;*****************************************************************************
=0001                 136   BSYMSK  EQU     00000001B               ;   Busy bit mask for FDCSTA register
=0040                 137   C_STPN  EQU     01000000B               ;   Step-in/noupdate/unload/noverify/6
=00D0                 138   C_STOP  EQU     11010000B               ;   Force interrupt/nointerrupts
=000A                 139   D0_324  EQU     10                      ;   32.4 usec delay count
=01DC                 140   D2_000  EQU     476                     ;   20.000 msec delay count
=04A7                 141   D4_999  EQU     1191                    ;   4.9998 msec delay count
=000F                 142   DSELCT  EQU     0FH                     ;   Deselect-all code
=0055                 143   EVNMSK  EQU     01010101B               ;   Even bits mask for secbuf errors
=00AA                 144   ODDMSK  EQU     10101010B               ;   Odd  bits mask for secbuf errors
=0400                 145   SBFSIZ  EQU     1024                    ;   Size of sector buffer in bytes
                      146           SUBTTL  Top level of power-up disk diagnostics
                      147   ;*****************************************************************************
=0000                 148           SECT    ROMCOD
                      149           ASSUME  CS:ROMCOD,DS:ROMDAT
0000'                 150   DSKTST  PROC    NEAR
```

```
0000'  E8 0015          151           CALL    SBFTST                  ;   Perform floppy disk tests
0003'  08 D2            152           OR      DL,DL                   ;   Q: Did any tests fail?
0005'  74 05            153           JZ      NOFAIL                  ;      N: Continue DSKTST
0007'  B0 0A*           154           MOV     AL,LED011               ;   AL := Disk diagnostic LED pattern
0009'  E9 0005"         155           JMP     DSPDIE                  ;   Now, go display and die
000C'  E8 0003"         156  NOFAIL:  CALL    DRVTST                  ;   See what drives we've got
000F'  E8 0004"         157           CALL    DSKINI                  ;   Initialize disk driver
0012'  E8 0002"         158           CALL    DECLED                  ;   Decrement diagnostic LED pattern
0015'  E9 0001"         159           JMP     CRTTST                  ;   Continue with CRT test
                        160  ;*******************************************************************************
                        161           SUBTTL  SBFTST - Sector buffer test
                        162  ;*******************************************************************************
0018'                   163  SBFTST:
0018'  A0 000C"         164           MOV     AL,DSKC_M               ;   AL := RAM copy of control port
001B'  24 3F            165           AND     AL,MMASK                ;   AL := buffer control bits only
001D'  0C 00            166           OR      AL,ACCBUF               ;   AL := set up for CPU (--) BUFFER
001F'  E6 00            167           OUT     DSKC_P,AL               ;   Hit control port
0021'  A2 000C"         168           MOV     DSKC_M,AL               ;   Update RAM copy
                        169  ;
0024'  BA 0020          170           MOV     DX,SECBUF               ;   DX := Pointer to sector buffer
0027'  BD 0100          171           MOV     BP,0100000000B          ;   BP := test pattern
002A'  30 FF            172           XOR     BH,BH                   ;   BH := Error bit accumulater
                        173  ;
002C'  8B C5            174  SBF_LP:  MOV     AX,BP                   ;   BP := Copy of test pattern
002E'  E6 22            175           OUT     RSTBUF,AL               ;   Reset secbuf counter
0030'  B9 03FF          176           MOV     CX,SBFSIZ-1             ;   CX := secbuf size in bytes
0033'  D1 C8            177           ROR     AX,1                    ;   Set the carry flag correctly
0035'  EE               178  W_LOOP:  OUT     DX,AL                   ;   write out the data
0036'  D0 D0            179           RCL     AL,1                    ;   walk it
0038'  E2 FB            180           LOOP    W_LOOP                  ;   write to all of sector buffer
003A'  E6 22            181           OUT     RSTBUF,AL               ;   Reset sector buffer counter
003C'  B9 03FF          182           MOV     CX,SBFSIZ-1             ;   CX := sector buffer size in bytes
003F'  8B C5            183           MOV     AX,BP                   ;   Get starting test pattern
0041'  D1 C8            184           ROR     AX,1                    ;   Set the carry flag correctly
0043'  8A D8            185           MOV     BL,AL                   ;
0045'  EC               186  R_LOOP:  IN      AL,DX                   ;   AL := data from sector buffer
0046'  9F               187           LAHF                            ;   Save carry flag
0047'  30 D8            188           XOR     AL,BL                   ;   AL := bits in error (if any)
0049'  08 C7            189           OR      BH,AL                   ;   BH := Accumulated bad bits.
004B'  9E               190           SAHF                            ;   Retreive carry flag.
004C'  D0 D3            191           RCL     BL,1                    ;   Generate next test byte
004E'  E2 F5            192           LOOP    R_LOOP                  ;   continue till testing finished
0050'  81 F5 FFFF       193           XOR     BP,0FFFFH               ;   test the complement
0054'  78 D6            194           JS      SBF_LP                  ; Q:  Done with straight and comp?
                        195  ;                                        ;  N:  jump and do complement.
0056'  D1 ED            196           SHR     BP,1                    ; Q:  All nine loops?
0058'  73 D2            197           JNB     SBF_LP                  ;  N:  Jump and continue.
005A'  30 D2            198           XOR     DL,DL                   ;   Clear error accumulator.
005C'  8A C7            199           MOV     AL,BH                   ;   AH := copy of the bad bits
005E'  A8 55            200           TEST    AL,EVNMSK               ;   Q: Are any even-order bits bad?
0060'  74 03            201           JZ      EVN_OK                  ;      N: Go check the odd bits
0062'  80 CA 07*        202           OR      DL,E_EVEN               ;      Y: Set even bit bad indicator
```

```
0065'  A8 AA                203   EVN_OK: TEST    AL,ODDMSK               ;   Q: Are any odd-order bits bad?
0067'  74 03                204           JZ      ODD_OK                  ;      N: Go test another byte
0069'  80 CA 08*            205           OR      DL,E_ODD                ;      Y: Set odd bit bad indicator
006C'                       206   ODD_OK:
                            207   ;       JMP     SHORT REGTST            ;   Then go test the disk controller
                            208   ;*******************************************************************************
                            209           SUBTTL  REGTST - FDC register test
                            210   ;*******************************************************************************
006C'                       211   REGTST:
006C'  A0 000C"             212           MOV     AL,DSKC_M               ;   AL := RAM copy of control port
006F'  24 3F                213           AND     AL,MMASK                ;   AL := buffer control bits only
0071'  0C 80                214           OR      AL,ACCFDC               ;   AL := set up for CPU (--) FDC
0073'  E6 00                215           OUT     DSKC_P,AL               ;   Hit control port
0075'  A2 000C"             216           MOV     DSKC_M,AL               ;   Update RAM copy
                            217   ;
0078'  BE 0008"             218           MOV     SI,OFFSET TSTPAT        ;   SI := address of test pattern table
007B'  B9 0004              219           MOV     CX,4                    ;   CX := set test count for 4 tests
                            220   ;
007E'  2E8A 04              221   REG_LP: MOV     AL,CS:BYTE PTR[SI]      ;   Test pattern
0081'  E6 21                222           OUT     FDCTRK,AL               ;   Load pattern in track register
0083'  2E8A 44 01           223           MOV     AL,CS:BYTE PTR 1[SI]    ;   Test pattern
0087'  E6 22                224           OUT     FDCSEC,AL               ;   Load pattern in sector register
0089'  2E8A 44 02           225           MOV     AL,CS:BYTE PTR 2[SI]    ;   Test pattern
008D'  E6 23                226           OUT     FDCDAT,AL               ;   Load pattern in data register
008F'  E4 21                227           IN      AL,FDCTRK               ;   AL := Pattern from track register
0091'  2E3A 04              228           CMP     AL,CS:BYTE PTR [SI]     ;   Q: Is pattern ok?
0094'  75 15                229           JNE     REGBAD                  ;      N: Go indicate register failure
0096'  E4 22                230           IN      AL,FDCSEC               ;   AL := Pattern from sector register
0098'  2E3A 44 01           231           CMP     AL,CS:BYTE PTR 1[SI]    ;   Q: Is pattern ok?
009C'  75 0D                232           JNE     REGBAD                  ;      N: Go indicate register failure
009E'  E4 23                233           IN      AL,FDCDAT               ;   AL := Pattern from data register
00A0'  2E3A 44 02           234           CMP     AL,CS:BYTE PTR 2[SI]    ;   Q: Is pattern ok?
00A4'  75 05                235           JNE     REGBAD                  ;      N: Go indicate register failure
00A6'  46                   236           INC     SI                      ;   Point to next test pattern
00A7'  E2 D5                237           LOOP    REG_LP                  ;
00A9'  EB 03                238           JMP     SHORT REG_OK            ;   Continue register test
                            239   ;
00AB'  80 CA 09*            240   REGBAD: OR      DL,E_REG                ;   Set register failed indicator
                            241   ;
00AE'                       242   REG_OK:
                            243   ;       JMP     CMDTST                  ;   Then go test the disk controller
                            244   ;*******************************************************************************
                            245           SUBTTL  CMDTST - FDC command test
                            246   ;*******************************************************************************
00AE'                       247   CMDTST:
                            248   ;
                            249   ;       MOV     AL,DSELCT               ;   AL := deselect everything code
                            250   ;       OUT     DSKS_P,AL               ;   Hit floppy select port with that
                            251   ;       MOV     DSKS_M,AL               ;   Also update RAM copy
00AE'  B0 D0                252           MOV     AL,C_STOP               ;   AL := command to stop FDC
00B0'  E6 20                253           OUT     FDCCMD,AL               ;   Tell FDC to stop
                            254   ;
```

```
00B2'  B9 000A                255             MOV     CX,D0_324               ;   Wait about 32.4 usec
00B5'  E2 FE                  256             LOOP    $                       ;
                              257     ;
00B7'  B0 40                  258             MOV     AL,C_STPN               ;   AL := step-in command
00B9'  E6 20                  259             OUT     FDCCMD,AL               ;   Hit floppy command port with that
                              260     ;
00BB'  B9 01DC                261             MOV     CX,D2_000               ;   Wait about 2 msec
00BE'  E2 FE                  262             LOOP    $                       ;
00C0'  E4 20                  263             IN      AL,FDCSTA               ;   AL := floppy status
00C2'  24 01                  264             AND     AL,BSYMSK               ;Q: Is controller still busy?
00C4'  74 0C                  265             JZ      CMDBAD                  ; N: To early, go die
                              266     ;
00C6'  B9 04A7                267             MOV     CX,D4_999               ;   Wait total of 7 msec
00C9'  E2 FE                  268             LOOP    $                       ;
                              269     ;
00CB'  E4 20                  270             IN      AL,FDCSTA               ;   AL := FDC status
00CD'  24 01                  271             AND     AL,BSYMSK               ;Q: Is the controller still busy?
00CF'  75 01                  272             JNZ     CMDBAD                  ; Y: Should be finished now, dammit
                              273     ;
00D1'  C3                     274             RET                             ; N: All tests done, return
                              275     ;
00D2'                         276     CMDBAD:
00D2'  B0 D0                  277             MOV     AL,C_STOP               ;   AL := Command to stop FDC
00D4'  E6 20                  278             OUT     FDCCMD,AL               ;   Hit floppy command port with that
00D6'  B9 000A                279             MOV     CX,D0_324               ;   Wait about 32.4 usec            0.8
00D9'  E2 FE                  280             LOOP    $                       ;                                  31.6
                              281     ;                                                                         -------
                              282     ;                                                                          32.4
                              283     ;
00DB'  80 CA 06*              284             OR      DL,E_CMD                ;   Set command test failed indicator
00DE'  C3                     285             RET                             ;   And return
                              286     ;*******************************************************************************
                              287             END
```

No errors detected

```
                        ;*****************************************************************************
                        ; TITLE    - FLPDIR
                        ; COMPUTER - 8088 ASSEMBLY LANGUAGE
                        ; ABSTRACT - Contains code which implements the Pegasus disk driver
                        ;            Generic Disk Operations for the floppy disk controller:
                        ;
                        ;            FLPFI  - Force-interrupt (not used for floppy disk)
                        ;            FLPIR  - Initiate Read
                        ;            FLPIW  - Initiate Write
                        ;            FLPVFC - Initiate Verify CRC
                        ;            FLPVFD - Initiate Verify Data
                        ;            FLPIS  - Initiate Seek
                        ;            FLPASP - Auxiliary State Processor
                        ;            FLPEP  - Error Processor
                        ;            FLPRIS - Read and Interpret Status
                        ;
                        ;            Also contains the following event entry points:
                        ;
                        ;            FLTIME - General floppy disk timing event
                        ;            MOTOFF - Floppy disk motor time-out handler
                        ;
                        ;            And the following entry point called during power-up testing:
                        ;
                        ;            FRCINT - Terminate FDC pending operation
                        ;
                        ; REGISTERS USED -
                        ;
                        ; STACK USED -
                        ;
                        ;*****************************************************************************
                                NAME    FLPDIR
                                SUBTTL  Floppy disk Generic Disk Operations
                        ;*****************************************************************************
                        ; The following fixes were made in version 1.01:
                        ;
                        ; * Problem: MSDOS 0001 - "disk-ROM no error reported on DRQ low".
                        ;   Solution: The "no data" error is now comprehended by FLPRS
                        ; * Problem: MSDOS 0003 - "disk-ROM Seek error reporting too global".
                        ;   Solution: The error processor, FLPEP, now does a "read address" operation
                        ;   when a "record not found" error occurs and uses the information gained
                        ;   thereby to better define the actual problem/error.
                        ; * Problem: the "force-interrupt-on-next-index-mark" trick to determine if
                        ;   media is present in the drive is not needed as the floppy disk controller
                        ;   counts index marks during I/O and times out after 5 have passed (1 second
                        ;   and returns "record not found" if the operation has not completed. That
                        ;   logic doesn't work anyway.
                        ;   Solution: That logic was removed to make room for the fix for MSDOS 0003
                        ;   which correctly returns error codes for the original situation as well.
                        ; * disk driver not waiting for disk motors to stabilize.
                        ;   The select logic, SELECT, has been modified to correctly determine the
                        ;   motor's current status and initiate a delay if needed.
                        ; * Need the 30 msec head-settling delay after seek.
```

```
                        ;   The seek logic, FLPIS, stores a mask to be used by the initiate I/O logic,
                        ;   INITIO, when issueing I/O commands to the floppy disk controller. INITIO
                        ;   clears this mask after using it so that the delay is induced only immedi-
                        ;   ately following seek operations.
                        ; The following changes were made in version 1.22:
                        ; * The motor off time has been reduced to 10 seconds and the select logic
                        ;   modified so that when an external drive is selected, the internal motors
                        ;   are not turned off.  Motors are turned on as needed and turned off only
                        ;   when no activity on any drive has been detected for 10 seconds.
                        ; * Some debug code was removed to make room for the changes.
                        ;*****************************************************************************
                        ;                               PUBLIC DEFINITIONS
                        ;*****************************************************************************
0000                            SECT    ROMCOD
                                PUBLIC  FLPDIR                  ; Floppy disk Disk Interface Routine
                        ;       PUBLIC  FLTIME                  ; Floppy timeout entry point
                                PUBLIC  FRCINT                  ; Terminate FDC function in progress
                                PUBLIC  MOTOFF                  ; Turn off all floppy drive motors
                        ;*****************************************************************************
                        ;                               EXTERNAL REFERENCES
                        ;*****************************************************************************
0000                            SECT    ROMCOD
                                EXTRN   SOFINT:NEAR             ; Software interrupt routine
                                                                ;   substituted for FLTIME
                                EXTRN   TIMOUT:NEAR             ; The timeout subroutine
                        ;*****************************************************************************
0000                            SECT    ROMDAT
                                EXTERN  D_EKNT:BYTE             ; Disk driver error counter
                                EXTERN  D_NCMD:BYTE             ; Disk driver next generic opcode
                                EXTERN  D_NSTA:BYTE             ; Disk driver next state
                                EXTERN  D_REQ:BYTE              ; Request Information Block
                                EXTERN  D_SOFT:BYTE             ; Software interrupt flag for DSKISR
                                EXTERN  D_STAT:BYTE             ; Disk driver current state
                                EXTERN  D_WKSP:BYTE             ; Disk driver general workspace
                                EXTERN  DSKC_M:BYTE             ; Memory image of floppy control port
                                EXTERN  DSKS_M:BYTE             ; Memory image of floppy select port
                                EXTERN  FL1EVT:BYTE             ; Floppy disk motor-off timing event
                                EXTERN  FL2EVT:BYTE             ; Floppy disk general usage timing evt
                                EXTERN  INVALID:ABS             ; Invalid position value
                        ;*****************************************************************************
                        ;                               PEGASUS INTERRUPT VECTORS
                        ;       INCLUDE PEG:VECTOR.EQU
                        ;*****************************************************************************
                        ;*****************************************************************************
                        ;                               PEGASUS DISK HARDWARE CONSTANT DEFINITIONS
                        ;       INCLUDE PEG:FLPADR.EQU
                        ;*****************************************************************************
                        ;*****************************************************************************
                        ; I/O and interface control port mask used by INITIO
000F                    IO_MSK  EQU     (not PRECMP) and (not SIDE_0) and MMASK
                        ;*****************************************************************************
                        ;                               FLOPPY DISK CONTROLLER DEFINITIONS
```

```
                        225   ;       INCLUDE PEQ:FLPCMD.EQU
                        226   ;*****************************************************************************
                        287   ;*****************************************************************************
=00D8                   288   FII     EQU     C_FOR or IMMEDI         ; Force-interrupt-immediate command
                        289   ;
=00D4                   290   INTIDX  EQU     C_FOR or INDEXP         ; Force-interrupt on index pulse
                        291   ;
=0082                   292   READ    EQU     C_RDS or T2SCHK         ; Read with side-select check
                        293   ;
=0001                   294   RESTOR  EQU     C_RES or T1ST06         ; Restore command
                        295   ;
                        296   ;               Seek    + update  + load    + 6 msec
                        297   ;                         track     head      step
                        298   ;                         reg               rate
=0018                   299   SEEK    EQU     C_SEE or T1UPDT or T1HLOD or T1ST06
                        300   ;
=0040                   301   STEPIN  EQU     C_STI or T1ST06         ; Step-in command
                        302   ;
=00A2                   303   WRITE   EQU     C_WRS or T2SCHK         ; Write with side-select check
                        304   ;*****************************************************************************
                        305   ;                       STATE DEFINITIONS (GENERAL & FLOPPY)
                        306   ;*****************************************************************************
                        307           INCLUDE PEQ:DSKSTA.EQU          ; Include disk driver state equates
                        308   ;                                       ; S_xxxx =) Driver states
=00FF                   309   S_DEP   EQU     255                     ; Device-dependent code must set state
=0000                   310   S_IDLE  EQU     0                       ; Waiting for a request
=0001                   311   S_WAIT  EQU     1                       ; Waiting to retry a request
=0002                   312   S_ISEEK EQU     2                       ; Implicit seek pending
=0003                   313   S_DONE  EQU     3                       ; Operation completed
=0003                   314   S_MAX   EQU     3                       ; The highest "standard" state number
                        315   ;                                       ; Other values are device-dependent
                        316   ;*****************************************************************************
=0004                   317   S_FRX   EQU     S_MAX+1                 ; Floppy read transfer state
=0005                   318   S_VFD   EQU     S_MAX+2                 ; Verify data state
=0006                   319   S_REC1  EQU     S_MAX+3                 ; Attempting error recovery state
=0007                   320   S_RDA   EQU     S_MAX+4                 ; Processing read address
=0007                   321   S_FMAX  EQU     S_MAX+4                 ; The highest valued floppy disk state
                        322   ;*****************************************************************************
                        323   ;                       DISK ERROR CODE DEFINITIONS
                        324   ;       INCLUDE PEQ:DSKERR.EQU
                        325   ;*****************************************************************************
                        363   ;*****************************************************************************
                        364   ;                       DIT STRUCTURE DEFINITION
                        365   ;       INCLUDE PEQ:DSKDIT.EQU
                        366   ;*****************************************************************************
                        381   ;*****************************************************************************
                        382   ;                       GENERIC OPCODE DEFINITIONS
                        383   ;       INCLUDE PEQ:DSKOP.EQU
                        384   ;*****************************************************************************
                        404   ;*****************************************************************************
                        405   ;                       I/O OPTIMIZATION CONSTANTS
                        406   ;*****************************************************************************
```

```
                        ;
                        ; The REPFAC constant is used for generating code in the loops which transfer
                        ; data to and from the sector buffer. "Loop packing" means that the contents of
                        ; the loop are repeated so that the loop count may be reduced. Loop overhead is
                        ; reduced, speeding up the operation overall.
                        ;
                        ; Packing factors are here expressed as the number of bits to adjust the sector
                        ; size (IN WORDS) for use as the loop counter in the data transfer loop.
                        ;
                        ; The number of times the data move operations is repeated within the data
                        ; transfer loop is called the repeat count. This quantity doubles for each bit
                        ; position the sector size count is shifted left. Therefore, the relationship
                        ; is: repeat count = 2 ^ packing factor.
                        ;
                        ; NOTE: sector size is stored in bytes, but data transfers are all in words.
                        ; Therefore, all figures are adjusted to work with word operations.
                        ;
                        ; The following table shows relative performance of various packing factors.
                        ; The single-byte data transfer loop figures are included as the baseline
                        ; performance reference.
                        ;
                        ; PACKING        DATA    CLOCK  RELATIVE     RELATIVE        LOOP
                        ;  FACTOR  TRANSFERED  CYCLES      TIME  PERFORMANCE        SIZE
                        ; ---------------------------------------------------------------
                        ;     0.5     1 BYTE    26112    100.0%         1.00     7 BYTES
                        ;       1     1 WORD    11520     44.1%         2.26     4 BYTES
                        ;       2    2 WORDS     9344     35.8%         2.79     6 BYTES
                        ;       3    4 WORDS     8256     31.6%         3.16    10 BYTES
                        ;       4    8 WORDS     7712     29.5%         3.39    18 BYTES
                        ;       5   16 WORDS     7440     28.5%         3.51    34 BYTES
                        ;       6   32 WORDS     7304     27.8%         3.59    66 BYTES
                        ;       7   64 WORDS     7236     27.7%         3.61   133 BYTES *
                        ;
                        ; * = loop must have has different terminating instruction sequence & requires
                        ;     a slightly different initialization (one more register) but does not
                        ;     incur any more time overhead in the loop than with the other examples.
                        ;
                        ;*******************************************************************************
=0004                   REPFAC  EQU     4                       ; Packing factor
=0008                   REPKNT  EQU     1 shl (REPFAC-1)        ; Repeat count
                        ;*******************************************************************************
                        ;                       RIB STRUCTURE DEFINITION
                        ;       INCLUDE PEQ:DSKRIB.EQU
                        ;*******************************************************************************
                        ;*******************************************************************************
                        ;                       TIMING EVENT DEFINITIONS
                        ;*******************************************************************************
                                INCLUDE PEQ:TIMEVT.EQU
                        ;
                        ; TIMING EVENT OFFSETS
                        ;
=0000                   IEVLNK  EQU     00H     ;       Link pointer
=0002                   IEVTID  EQU     02H     ;       Task id
```

```
=0003                   474   IEVFLG  EQU     03H     ;       Flags
=0004                   475   IEVCTR  EQU     04H     ;       Counter
=0006                   476   IEVSUB  EQU     06H     ;       Subroutine address
                        477   ;
                        478   ; Event flags
                        479   ;
=0004                   480   EVTTYP  EQU     4       ;       Event type          0/1 = Interrupt/Task
=0005                   481   EVTACK  EQU     5       ;       Event acknowledged  0/1 = Yes/No
=0006                   482   EVTEXP  EQU     6       ;       Event expired       0/1 = No/Yes
=0007                   483   EVTSTA  EQU     7       ;       Event status        0/1 = Inactive/Active
                        484
                        485   ;*****************************************************************************
=0080                   486   ACTEVT  EQU     1 shl EVTSTA                    ; Mask for setting bit to activate evnt
=000C                   487   INDXTI  EQU     12                              ; Index pulse wait interval = 300 msec
=0014                   488   MONTIM  EQU     20                              ; Disk motor-on wait = 1/2 sec
=0190                   489   MOFTIM  EQU     400                             ; Motor-off timeout interval = 10 sec
                        490   ;*****************************************************************************
                        491   ;                       WORKSPACE OFFSETS
                        492   ;*****************************************************************************
=0000                   493   F_LCMD  EQU     0                               ; For saving last FDC command issued
=0001                   494   F_DLAY  EQU     1                               ; For saving delay mask after seek
                        495   ;*****************************************************************************
                        496   ;                       MACROS
                        497   ;*****************************************************************************
                        498   EVENT   MACRO   %EVT,%TIM,%WHERE        ; Start an event
                        499           IFNB    (%WHERE)
                        500           MOV     word ptr %EVT+IEVSUB,offset %WHERE
                        501           ENDIF
                        502           MOV     word ptr %EVT+IEVCTR,%TIM
                        503           OR      %EVT+IEVFLG,ACTEVT
                        504           ENDM
                        505   ;*****************************************************************************
                        506   KILEVT  MACRO   %EVT                    ; Cancel an active event
                        507           AND     %EVT+IEVFLG, not ACTEVT
                        508           ENDM
                        509   ;*****************************************************************************
                        510   ;                       BEGINNING OF THE CODE AREA
                        511   ;*****************************************************************************
=0000                   512           SECT    ROMCOD
                        513           ASSUME  CS:ROMCOD,DS:ROMDAT,ES:NOTHING
                        514           SUBTTL  Floppy disk GDO routine: FLPDIR
                        515   ;*****************************************************************************
                        516   ; NAME:           FLPDIR - Floppy Disk Interface Routine
                        517   ; ABSTRACT:       This is the call interface to the floppy disk General Disk
                        518   ;                 Operation routines.
                        519   ; INPUTS:         * AL = function code (always valid)
                        520   ;                 * SS:BP = @ the disk driver data area
                        521   ; PROCESSING:     * Transfer control to the appropriate Generic Disk Operation
                        522   ; REGISTERS:      AX plus whatever transfered routine uses
                        523   ; STACK USED:     Whatevr the called routine uses (up to about 12 bytes)
                        524   ;*****************************************************************************
0000                    525   FLPDIR  PROC    FAR
```

```
0000'  98                    526           CBW                             ;   Convert generic opcode to word
0001'  01 C0                 527           ADD     AX,AX                   ;   Convert generic opcode to index
0003'  8B F0                 528           MOV     SI,AX                   ;   SI := jump table index
0005'  2EFF A4 000A"         529           JMP     CS:CMDTBL[SI]           ;   Branch to routine
                             530   ;*****************************************************************************
000A'                        531   CMDTBL  LABEL   WORD                    ;   Generic disk operations jump table
000A'  013B"                 532           DW      FLPFI                   ;   Vector: Force interrupt
000C'  01E8"                 533           DW      FLPASP                  ;   Vector: Auxiliary state processor
000E'  025D"                 534           DW      FLPEP                   ;   Vector: Error processor
0010'  01D1"                 535           DW      FLPIR                   ;   Vector: Initiate read
0012'  013C"                 536           DW      FLPIS                   ;   Vector: Initiate seek
0014'  015D"                 537           DW      FLPIW                   ;   Vector: Initiate write
0016'  01D6"                 538           DW      FLPVFC                  ;   Vector: Verify CRC
0018'  01DC"                 539           DW      FLPVFD                  ;   Vector: Verify data
001A'  018F"                 540           DW      FLPRS                   ;   Vector: Read status
001C'  029E"                 541           DW      FLPBGN                  ;   Vector: Request termination
001E'  0287"                 542           DW      FLPTER                  ;   Vector: Terminate floppy request
                             543   ;*****************************************************************************
                             544           SUBTTL  Floppy disk internal ODO routine: CPUxxx (MODIFY INTERFACE CTL)
                             545   ;*****************************************************************************
                             546   ; NAME:           CPUxxx
                             547   ; ABSTRACT:       Modifies the interface control bits so that the CPU can talk to
                             548   ;                 the Floppy Disk Controller (FDC) or the Sector Buffer (BUF).
                             549   ;                 - CPUFDC: Set CPU (--) FDC
                             550   ;                 - CPUBUF: Set CPU (--) sector buffer
                             551   ; REGISTERS:      AX
                             552   ; STACK USED:     4 bytes
                             553   ;*****************************************************************************
0020'                        554   CPUFDC  PROC    NEAR
0020'  B0 80                 555           MOV     AL,ACCFDC               ;   AH := bit pattern for CPU (--) FDC
0022'  EB 02                 556           JMP     short ptr CHGMOD        ;   Go change interface mode
                             557   ;*****************************************************************************
0024'                        558   CPUBUF  PROC    NEAR
0024'  B0 00                 559           MOV     AL,ACCBUF               ;   AH := bit pattern for CPU (--) BUF
                             560   ;       JMP     short ptr CHGMOD        ;   Go change interface mode
                             561   ;*****************************************************************************
0026'  B4 3F                 562   CHGMOD: MOV     AH,MMASK                ;   AH := Interface mode bit mask
0028'  EB 0E                 563           JMP     short ptr HITC_P        ;   Go change interface control port
                             564   ;*****************************************************************************
                             565           SUBTTL  Floppy disk internal ODO routine: FLTIME
                             566   ;*****************************************************************************
                             567   ; FLTIME is called by the system timing event routine (driven by the
                             568   ; system clock) when a general floppy timing event (associated with the event
                             569   ; data structure labeled: FL2EVT) expires. The two floppy timing events handled
                             570   ; are: the wait for a motor to come up to speed & the wait for an interrupt on
                             571   ; an index pulse when processing a timeout error.
                             572   ; PROCESSING: * Interrupt into the disk interrupt service routine
                             573   ; REGISTERS USED: none
                             574   ; STACK USED: 6 bytes
                             575   ;*****************************************************************************
                             576   ;FLTIME PROC    NEAR
                             577   ;       INC     D_SOFT                  ;   Make software interrupt flag non-0
```

```
                        578   ;       INT     IR6INT                  ;   Invoke DSKISR
                        579   ;       RET                             ;   Then return
                        580   ; NOTE: these timing events are now vectored directly to the routine SOFINT
                        581   ; found in the module, DSKIO which does exactly what you see here.
                        582   ;*****************************************************************************
                        583           SUBTTL  Floppy disk internal ODD routine: FRCINT
                        584   ;*****************************************************************************
                        585   ; PUBLIC/INTERNAL ROUTINE
                        586   ; NAME:           FRCINT - Force interrupt
                        587   ; ABSTRACT:       Clear the floppy disk controller to an idle condition.
                        588   ; PROCESSING:     * Send a "force interrupt" with no options to the FDC
                        589   ;                 * Impose a short delay for the command to take affect
                        590   ; OUTPUT:         * Interface mode set to CPU (--) FDC
                        591   ; REGISTERS:      AX
                        592   ; STACK USED:     6 bytes
                        593   ; NOTES:          This routine is misnamed because it is not intended to generate
                        594   ;                 an interrupt even though it does use a "force interrupt" com-
                        595   ;                 mand. In fact, "force interrupt" with no options does not cause
                        596   ;                 the FDC to interrupt.
                        597   ;*****************************************************************************
002A'                   598   FRCINT  PROC    NEAR
002A'  E8 FFF3          599           CALL    CPUFDC          ;   Set CPU (--) FDC
002D'  B0 D0            600           MOV     AL,C_FOR        ;   AL := force interrupt command
002F'  E6 20            601           OUT     FDCCMD,AL       ;   Tell FDC to do that
0031'  B0 0A            602           MOV     AL,10           ;   Delay after forced interrupt
0033'  FE C8            603   INTDLY: DEC     AL              ;   Does not affect carry
0035'  75 FC            604           JNZ     INTDLY
0037'  C3               605           RET                     ;   And return
                        606   ;*****************************************************************************
                        607           SUBTTL  Floppy disk internal ODD routine: HITC_P
                        608   ;*****************************************************************************
                        609   ; NAME:           HITC_P - Change interface control port
                        610   ; ABSTRACT:       Change floppy disk interface control port to new settings
                        611   ; PROCESSING:     * Disable interrupts for exclusive access (required because
                        612   ;                   the port is shared - i.e., some bits are used for other
                        613   ;                   hardware than the floppy disk interface)
                        614   ;                 * Create new pattern for interface control port using inputs
                        615   ;                 * Ouput new pattern to the control port
                        616   ;                 * Update RAM copy of the port
                        617   ; INPUT:          * AL = new bit fields for interface control port
                        618   ;                 * AH = bit mask for masking out fields to be changed
                        619   ; OUTPUT:         * [DSKC_M] = Updated control port image
                        620   ;                 * Interface control port updated as requested
                        621   ; REGISTERS USED: AX
                        622   ; STACK USED: 4 bytes
                        623   ; NOTE: ";77" comment delimiters indicate interrupts may be either on or off
                        624   ;*****************************************************************************
0038'                   625   HITC_P  PROC    NEAR
0038'  9C               626           PUSHF                           ;77 Save flags (esp. interrupt flag)
0039'  FA               627           CLI                             ;;; Critical code, disable interrupts
003A'  22 26 000A"      628           AND     AH,DSKC_M               ;;; AH := control port less new bits
003E'  08 E0            629           OR      AL,AH                   ;;; AL := updated port image
```

```
0040'  A2 000A"          630             MOV     DSKC_M,AL               ;;; Update control port image
0043'  E6 00             631             OUT     DSKC_P,AL               ;;; Hit control port
0045'  9D                632             POPF                            ;;; Restore flags (incl. interrupt)
0046'  C3                633             RET                             ;?? And return
                         634     ;*****************************************************************************
                         635             SUBTTL  Floppy disk internal QDO routine: HITS_P
                         636     ;*****************************************************************************
                         637     ; NAME:           HITS_P - Program select and motor control port
                         638     ; ABSTRACT:       Write new pattern to select and motor control port
                         639     ; PROCESSING:     * Output supplied pattern to the select and motor control port
                         640     ;                 * Save a copy of that pattern in the memory image copy of port
                         641     ; INPUT:          * AL = bit pattern for select and motor control control port
                         642     ; OUTPUT:         * [DSKC_M] = Updated control port image
                         643     ;                 * Interface control port updated as requested
                         644     ; REGISTERS:      none
                         645     ; STACK USED:     2 bytes
                         646     ; NOTES:          Unlike HITC_P, the select and motor control port is not shared.
                         647     ;                 Therefore, there is no need to protect accesses to it by dis-
                         648     ;                 abling interrupts.
                         649     ;*****************************************************************************
0047'                    650     HITS_P  PROC    NEAR
0047'  A2 000B"          651             MOV     DSKS_M,AL               ;   Update select port image
004A'  E6 04             652             OUT     DSKS_P,AL               ;   Hit select port
004C'  C3                653             RET                             ;   And return
                         654     ;*****************************************************************************
                         655             SUBTTL  Floppy disk internal QDO routine: HITFDC (ISSUE FDC COMMAND)
                         656     ;*****************************************************************************
                         657     ; NAME:           HITFDC - Issue FDC (Floppy Disk Controller) command
                         658     ; ABSTRACT:       Send a command to the floppy disk controller
                         659     ; PROCESSING:     * Output command to FDC command register
                         660     ;                 * Store that command as last one issued
                         661     ; INPUT:          * AL = FDC command to issue
                         662     ;                 * Interface mode MUST BE: CPU (--) FDC
                         663     ; OUTPUT:         * [D_WKSP+F_LCMD] (byte) = last FDC command issued
                         664     ; REGISTERS:      none
                         665     ; STACK USED:     2 bytes
                         666     ;*****************************************************************************
004D'                    667     HITFDC  PROC    NEAR
004D'  E6 20             668             OUT     FDCCMD,AL
004F'  A2 0000"          669             MOV     D_WKSP+F_LCMD,AL
0052'  C3                670             RET
                         671     ;*****************************************************************************
                         672             SUBTTL  Floppy disk internal QDO routine: INITIO
                         673     ;*****************************************************************************
                         674     ; NAME:           INITIO - Initiate I/O
                         675     ; ABSTRACT:       Performs processing to initiate an I/O operation
                         676     ; PROCESSING:     * Select disk drive
                         677     ;                 * Program interface and floppy disk controller for the request-
                         678     ;                   ed I/O
                         679     ; INPUTS:         * CH = I/O opcode (read or write) to execute
                         680     ;                 * CL = Next state to enter
                         681     ;                 * DH = Appropriate interface mode control bits:
```

```
                                682   ;                 FDC --) BUF for READ
                                683   ;                 BUF --) FDC for WRITE
                                684   ;               * ES:BX = @ DIT for the current disk drive
                                685   ;               * [D_REQ+CYL] (byte) = desired cylinder for the I/O
                                686   ;               * [D_REQ+TRK] (byte) = desired track for the I/O
                                687   ;               * [D_REQ+SEC] (byte) = desired sector for the I/O
                                688   ; OUTPUT:       * [D_NSTA] (byte) = Disk driver's next state
                                689   ; REGISTERS:    AX, BX, CX, DX, SI
                                690   ; STACK USED:   8 bytes
                                691   ;*******************************************************************************
0053'                           692   INITRD  PROC    FAR                     ;   Special entry for read operations
0053'  B5 82                   693           MOV     CH,READ                 ;   CH := Read command with options
0055'  B6 40                   694           MOV     DH,FDCBUF               ;   DH := bit pattern for FDC --) BUF
0057'                           695   INITIO:                                 ;   The normal entry point
0057'  E8 FFCA                 696           CALL    CPUBUF                  ;   Set interface mode to CPU (--) BUF
005A'  E6 22                   697           OUT     RSTBUF,AL               ;   Reset sector buffer counter
005C'  E8 0052                 698           CALL    SELECT                  ;   Select disk & insure motor is on
                               699                                           ;   Interface mode is CPU (--) FDC
005F'  72 3E                   700           JB      INIRET                  ;   If we must wait for the motor, exit
0061'  A0 0000"                701           MOV     AL,D_REQ+CYL            ;   AL := desired cylinder
0064'  E6 21                   702           OUT     FDCTRK,AL               ;   Set cylinder (FDC "track")
0066'  263A 47 0A              703           CMP     AL,ES:DITPRC[BX]        ;   @: Cylinder )= pre-comp threshold?
006A'  7C 03                   704           JL      NOPRCM                  ;      N: Don't set pre-comp bit
006C'  80 CE 10                705           OR      DH,PRECMP               ;      Y: Set pre-comp bit
006F'  80 CE 20                706   NOPRCM: OR      DH,SIDE_0               ;   Select side 0
0072'  A0 0000"                707           MOV     AL,D_REQ+TRK            ;   AL := desired track ("side")
0075'  3C 01                   708           CMP     AL,1                    ;   Is it track 1 (i.e., side 1)?
0077'  75 06                   709           JNE     NOSID1                  ;      N: Don't select side 1
0079'  80 E6 DF                710           AND     DH,not SIDE_0           ;      Y: Select side 1
007C'  80 CD 08                711           OR      CH,T2CHK1               ;         & set cmd to check for side 1
007F'  A0 0000"                712   NOSID1: MOV     AL,D_REQ+SEC            ;   AL := desired sector
0082'  E6 22                   713           OUT     FDCSEC,AL               ;   Set sector
0084'  8A C5                   714           MOV     AL,CH                   ;   AL := command to execute
0086'  0A 06 0000"             715           OR      AL,D_WKSP+F_DLAY        ;   Or in delay mask (if any)
008A'  FA                      716           CLI                             ;;; Critical code - mask interrupts
008B'  E8 FFBF                 717           CALL    HITFDC                  ;;; Tell FDC to execute the command
                               718                                           ;;; Now quickly set control port:
008E'  8A C6                   719           MOV     AL,DH                   ;;; AL := new bit fields for the port
0090'  B4 0F                   720           MOV     AH,IO_MSK               ;;; AH := bit mask for changed fields
0092'  E8 FFA3                 721           CALL    HITC_P                  ;;; Update interface control port
0095'  FB                      722           STI                             ;;; End critical code sequence
0096'  88 0E 0005"             723           MOV     D_NSTA,CL               ;   Set the driver's next state
009A'  C6 06 0000" 00          724           MOV     D_WKSP+F_DLAY,0         ;   Clear delay mask
009F'  CB                      725   INIRET: RET                             ;   And return
                               726   ;*******************************************************************************
                               727           SUBTTL  Floppy disk internal ODD routine: MOTOFF
                               728   ;*******************************************************************************
                               729   ; PUBLIC ROUTINE
                               730   ; NAME:         MOTOFF - Floppy motor off
                               731   ; ABSTRACT:     Called by the system timing event routine (driven by the timer)
                               732   ;               to turn off running drive motors after of period of non-use.
                               733   ; PROCESSING:   * Turn off all floppy disk drive motors and deselect all drives
```

```
                                734   ; INPUT:          none
                                735   ; OUTPUT:         none
                                736   ; REGISTERS:      none
                                737   ; STACK USED:     6 bytes
                                738   ;*****************************************************************************
00A0'                           739   MOTOFF  PROC      NEAR
00A0'  50                       740           PUSH      AX                      ;   Save AX register
00A1'  B0 0F                    741           MOV       AL,DRVMSK               ;   AL := all motors off bit pattern
00A3'  E8 FFA1                  742           CALL      HITS_P                  ;   Change select & motor control port
                                743+          KILEVT    FL1EVT                  ;   Cancel motor-off event
00AB'  58                       745           POP       AX                      ;   Restore AX
00AC'  C3                       746           RET                               ;
                                747   ;*****************************************************************************
                                748           SUBTTL  Floppy disk internal QDO routine: SELECT
                                749   ;*****************************************************************************
00AD'  3E                       750   SELTBL  DB        DRIV_0 or MOTR_0        ; Unit number select bit table
00AE'  3D                       751           DB        DRIV_1 or MOTR_1
00AF'  4B                       752           DB        DRIV_2 or MOTR_2
00B0'  87                       753           DB        DRIV_3 or MOTR_3
                                754   ;*****************************************************************************
                                755   ; NAME:           SELECT - Select disk drive
                                756   ; ABSTRACT:       Prepare floppy disk interface for accessing a particular drive
                                757   ; PROCESSING:     * Select disk drive and turn on its motor
                                758   ;                 * If the motor was not previously running:
                                759   ;                   - Initiates a wait for the motor to come up to speed
                                760   ;                   - Set driver state to S_WAIT which will cause the request to
                                761   ;                     be re-issued when the "motor-on wait" timing event expires
                                762   ;                   - Return an indication that the request must wait
                                763   ; INPUT:          * [D_REQ+UNIT] (byte) = Pegasus unit number of unit to select
                                764   ;                 * CH,CL,DH,ES:BX - Registers which must be preserved
                                765   ; OUTPUT:         * CH,CL,DH,ES:BX - Same as value on entry
                                766   ;                 * CF = 0 =) motor was running, 1 =) motor was not running
                                767   ;                 * If the motor was not running:
                                768   ;                   - [D_NCMD] = OP_NUL (do nothing command)
                                769   ;                   - [D_NSTA] = S_MOT (motor wait)
                                770   ;                 * Interface mode is set to CPU (--) FDC
                                771   ; REGISTERS:      AX
                                772   ; STACK USED:     6 bytes
                                773   ;*****************************************************************************
00B1'                           774   SELECT  PROC      NEAR
00B1'  E8 FF6C                  775           CALL      CPUFDC                  ;   Set interface mode to CPU (--) FDC
00B4'  53                       776           PUSH      BX                      ;   Save BX
00B5'  8A 1E 0000"              777           MOV       BL,D_REQ+UNIT           ;   BL := unit number
00B9'  30 FF                    778           XOR       BH,BH                   ;   Convert it to word for use as index
00BB'  A0 0000"                 779           MOV       AL,DSKS_M               ;
00BE'  24 F0                    780           AND       AL,not MOTMSK           ;   Mask out all but the motor data
00C0'  8A D0                    781           MOV       DL,AL                   ;   DL := image of select & motor port
00C2'  2E0A 87 00AD"            782           OR        AL,SELTBL[BX]           ;   AL := New select and motor bits
00C7'  E8 FF7D                  783           CALL      HITS_P                  ;   Change select & motor control port
00CA'  24 F0                    784           AND       AL,not MOTMSK           ;   Mask out all but the motor data
00CC'  30 D0                    785           XOR       AL,DL                   ;   Q: Was the motor already on?
00CE'  74 16                    786           JZ        SELRET                  ;       Y: Then we are ready to go
```

```
00D0'  C6 06 0005" 01          787             MOV     D_NSTA,S_WAIT           ;       N: Next state := general wait
00D5'  C6 06 0004" FF          788             MOV     D_NCMD,OP_NUL           ;          Change next operation to null
                               789+            EVENT   FL2EVT,MONTIM           ;          Start event for motor-on wait
00E5'  F9                      795             STC                             ;          Set CF indicating the wait
00E6'  5B                      796     SELRET: POP     BX                      ;   Restore BX
00E7'  C3                      797             RET                             ;   And return
                               798     ;*****************************************************************************
                               799             SUBTTL  Floppy disk internal ODD routine: VERIFY
                               800     ;*****************************************************************************
                               801     ; NAME:         VERIFY - Verify data
                               802     ; ABSTRACT:     Compares data in the sector buffer with memory data
                               803     ; PROCESSING:   * Compare the user's data with the disk data in sector buffer
                               804     ; INPUT:        * [D_REQ+BUF] (dword) = segment:offset for user's buffer
                               805     ;                * ES:BX = @ DIT for the current disk drive
                               806     ; OUTPUT:       * If a verify failure occurs:
                               807     ;                  - [D_REQ+STAT] (byte) = "DKVFYE" (Verify error)
                               808     ;                  - [D_REQ+BUF]  (dword) = address of WORD which doesn't match
                               809     ; REGISTERS:    AX, BX, CX, DX, SI
                               810     ; STACK USED:   6 bytes
                               811     ; NOTES:        Unlike the read and write data transfers, VERIFY *does not* use
                               812     ;                the loop packing method to speed up sector buffer accesses.
                               813     ;                This is for the sake of simplicity. Especially since the CMPS
                               814     ;                (compare string) instruction cannot be used in the same simple
                               815     ;                manner as STOS or LODS and, therefore, was not used.
                               816     ;*****************************************************************************
00E8'                          817     VERIFY  PROC    NEAR
00E8'  E8 FF39                 818             CALL    CPUBUF                  ;   Set interface mode to CPU (--) BUF
00EB'  E6 22                   819             OUT     RSTBUF,AL               ;   Reset sector buffer counter
00ED'  2688 4F 04              820             MOV     CX,ES:DITBEC[BX]        ;   CX := Sector length in bytes
00F1'  D1 E9                   821             SHR     CX,1                    ;   Adjust for word access to secbuf
00F3'  BA 0020                 822             MOV     DX,SECBUF               ;   DX := Address of sector buffer port
00F6'  1E                      823             PUSH    DS                      ;   Save current DS
00F7'  C5 36 0000"             824             LDS     SI,dword ptr D_REQ+BUF  ;   DS:SI := @ user buffer
                               825                                             ;
00FB'  ED                      826     VRLOOP: IN      AX,DX                   ;   AX := 1 word from sector buffer
00FC'  3B 04                   827             CMP     AX,word ptr DS:[SI]     ;   Q: Does the data match?
00FE'  75 06                   828             JNE     NO_VFY                  ;      N: Go fail
0100'  46                      829             INC     SI                      ;      Y: Keep checking until done
0101'  46                      830             INC     SI                      ;
0102'  E2 F7                   831             LOOP    VRLOOP                  ;
0104'  1F                      832             POP     DS                      ;   Restore DS
0105'  C3                      833             RET                             ;   And return
                               834                                             ;
0106'  1F                      835     NO_VFY: POP     DS                      ;   Restore DS
0107'  89 36 0000"             836             MOV     word ptr D_REQ+BUF+0,SI ;   Store address of bad word
010B'  C6 06 0000" 05          837             MOV     D_REQ+STAT,DKVFYE       ;   Set status to failure
0110'  C3                      838             RET                             ;   And return
                               839     ;*****************************************************************************
                               840             SUBTTL  Floppy disk internal ODD routine: XFRRDD
                               841     ;*****************************************************************************
                               842     ; NAME:         XFRRDD - Transfer read data
                               843     ; ABSTRACT:     Called to unload the contents of the sector buffer into memory.
```

```
                                844   ; PROCESSING:    * Transfer data from the sector buffer to the user's buffer
                                845   ; INPUT:         * [D_REQ+BUF] (dword) = segment:offset for user's buffer
                                846   ;                * ES:BX = @ DIT for the current disk drive
                                847   ; OUTPUT:        none
                                848   ; REGISTERS:     AX, BX, CX, DX, DI
                                849   ; STACK USED:    8 bytes
                                850   ; NOTES:         This routine employs the loop packing technique discussed
                                851   ;                earlier to speed up transferring data from the sector buffer
                                852   ;                to memory.
                                853   ;*******************************************************************************
0111'                           854   XFRRDD  PROC     NEAR
0111'  E8 FF10                 855           CALL     CPUBUF                  ;   Set interface mode to CPU (--) BUF
0114'  E6 22                   856           OUT      RSTBUF,AL               ;   Reset sector buffer counter
0116'  268B 47 04              857           MOV      AX,ES:DITSEC[BX]        ;   AX := Sector length in bytes
011A'  B1 04                   858           MOV      CL,REPFAC               ;   Adjust for loop packing and word...
011C'  D3 E8                   859           SHR      AX,CL                   ;     access to the sector buffer
011E'  8B C8                   860           MOV      CX,AX                   ;   CX := Repeat count for output loop
0120'  BA 0020                 861           MOV      DX,SECBUF               ;   DX := Address of sector buffer port
0123'  06                      862           PUSH     ES                      ;   Save calling ES register
0124'  C4 3E 0000"             863           LES      DI,dword ptr D_REQ+BUF  ;   ES:DI := @ user buffer
                               864                                           ;
0128'  =0008                   865   RDLOOP: REPT     REPKNT                  ;   Pack the output loop
                               866           IN       AX,DX                   ;   AX := 1 word from sector buffer
                               867           STOS     word ptr ES:[DI]        ;   Place data in the user's buffer
                               868           ENDR                             ;   End of repeated section
0138'  E2 EE                   869           LOOP     RDLOOP                  ;   Loop until the buffer is read
                               870                                           ;
013A'  07                      871           POP      ES                      ;   Restore calling ES register
013B'  C3                      872   RETINS: RET                              ;   And return
                               873   ;*******************************************************************************
                               874           SUBTTL   Floppy disk ODD routine: FLPFI
                               875   ;*******************************************************************************
                               876   ; NAME:          FLPFI - Force Interrupt
                               877   ; ABSTRACT:      Performs the processing needed to cause the disk controller to
                               878   ;                send an interrupt request.
                               879   ; PROCESSING:    * Send a "force interrupt immediate" to the disk controller
                               880   ; INPUT:         none
                               881   ; OUTPUT:        * [D_WKSP+F_LCMD] (byte) = Last command sent to FDC
                               882   ; REGISTERS:     AX
                               883   ; STACK USED:    6 bytes
                               884   ; NOTES:         I found that the "force interrupt" function was not needed for
                               885   ;                driving the floppy disk, so this code has been omitted to save
                               886   ;                space in the system ROM, which is CRITICALLY short of space.
                               887   ;                If this function is somehow called inadvertantly, it will
                               888   ;                return immediately.
                               889   ;*******************************************************************************
        =013B                  890   FLPFI   EQU      RETINS                  ;   Equate to location of a return
                               891   ;       CALL     CPUFDC                  ;   Set interface mode to CPU (--) FDC
                               892   ;       MOV      AL,FII                  ;   AL := Force-interrupt-immediate
                               893   ;       CALL     HITFDC                  ;   Hit FDC command register with that
                               894   ;       RET                              ;   And return
                               895   ;*******************************************************************************
```

```
                                912             SUBTTL  Floppy disk QDQ routine: FLPIB
                                913     ;*******************************************************************************
                                914     ; NAME:           FLPIB - Initiate Seek
                                915     ; ABSTRACT:       Performs the processing needed to seek a particular cylinder on
                                916     ;                 a disk. Also processes the "disk system reset" condition indi-
                                917     ;                 cated by the "INVALID" (-1) current position for the disk.
                                918     ; PROCESSINQ:     * Call SELECT to:
                                919     ;                   - Select the desired disk drive
                                920     ;                   - If the drive motor isn't on:
                                921     ;                     = Turn it on
                                922     ;                     = Initiate a wait for it to come up tp speed
                                923     ;                 * If the current position is "INVALID", initiate a recalibrate
                                924     ;                   operation before allowing the seek to begin
                                925     ;                 * Initiate a seek to the desired cylinder
                                926     ; INPUT:          * [D_REQ+CYL] (byte) = desired cylinder for the seek
                                927     ; OUTPUT:         * [D_NSTA] (byte) = Next state for disk driver (S_WAIT, S_REC1,
                                928     ;                   or "unchanged" (S_ISEEK))
                                929     ;                 * [D_WKSP+F_LCMD] (byte) = Last command sent to FDC
                                930     ;                 * [D_WKSP+F_DLAY] (byte) = 30 msec delay mask for next I/O
                                931     ; REGISTERS:      AX, BX
                                932     ; STACK USED:     8 bytes
                                933     ;*******************************************************************************
013C'                           934     FLPIB   PROC    FAR
013C'  E8 FF72                  935             CALL    SELECT                  ;   Select disk & insure motor on
                                936                                             ;   Interface mode is CPU (--) FDC
013F'  72 18                    937             JB      ISRET                   ;   If the motor wasn't on, wait for it
0141'  A0 0000"                 938             MOV     AL,D_REQ+CURCYL         ;   AL := Current cylinder
0144'  3C 0E*                   939             CMP     AL,INVALID              ;   Q: Was disk system reset?
0146'  74 12                    940             JE      RESET                   ;       Y: Qo reset & recalibrate
0148'  E6 21                    941             OUT     FDCTRK,AL               ;   Set track register to that
014A'  A0 0000"                 942             MOV     AL,D_REQ+CYL            ;   AL := Desired cylinder
014D'  E6 23                    943             OUT     FDCDAT,AL               ;   Select cylinder (FDC "track" )
014F'  B0 18                    944             MOV     AL,BEEK                 ;   AL := Seek command with options
0151'  E8 FEF9                  945             CALL    HITFDC                  ;   Hit command port with that
0154'  C6 06 0000" 04           946             MOV     D_WKSP+F_DLAY,T2DL30    ;   Set 30 msec delay mask for next I/O
0159'  CB                       947     ISRET:  RET                             ;   And return
                                948                                             ;
015A'  E9 011B                  949     RESET:  JMP     RECAL                   ;   Qo recalibrate
                                950     ;*******************************************************************************
                                951             SUBTTL  Floppy disk QDQ routine: FLPIW
                                952     ;*******************************************************************************
                                953     ; NAME:           FLPIW - Initiate Write
                                954     ; ABSTRACT:       Perform the processing needed to write data to the disk
                                955     ; PROCESSINQ:     * Transfer the data to the sector buffer
                                956     ;                 * Set parameters and call INITIQ to initiate a write operation
                                957     ; INPUT:          * ES:BX = @ DIT for the current disk drive
                                958     ;                 * [D_REQ+BUF] (dword) = segment:offset for user's buffer
                                959     ;                 * [D_REQ+CYL] (byte) = desired cylinder for the I/O
                                960     ;                 * [D_REQ+TRK] (byte) = desired track for the I/O
                                961     ;                 * [D_REQ+SEC] (byte) = desired sector for the I/O
                                962     ; OUTPUT:         * [D_NSTA] (byte) = next state for driver (S_WAIT or S_DONE)
                                963     ; REGISTERS:      AX, BX, CX, DX, SI
```

```
                                964   ; STACK USED:    10 bytes
                                965   ; NOTES:         This routine employs the loop packing discussed earlier to
                                966   ;                speed up transferring data between memory & the sector buffer.
                                967   ;*******************************************************************************
     015D'                      968   FLPIW   PROC    FAR
     015D'  E8 FEC4             969           CALL    CPUBUF                  ;   Set interface mode to CPU (--) BUF
     0160'  E6 22               970           OUT     RSTBUF,AL               ;   Reset sector buffer counter
     0162'  2688 47 04          971           MOV     AX,ES:DITSEC[BX]        ;   AX := Sector length in bytes
     0166'  B1 04               972           MOV     CL,REPFAC               ;   Adjust for loop packing and word...
     0168'  D3 E8               973           SHR     AX,CL                   ;     access to the sector buffer
     016A'  8B C8               974           MOV     CX,AX                   ;   CX := Repeat count for output loop
     016C'  BA 0020             975           MOV     DX,SECBUF               ;   DX := Address of sector buffer port
     016F'  1E                  976           PUSH    DS                      ;   Save current data segment register
     0170'  C5 36 0000"         977           LDS     SI,dword ptr D_REQ+BUF  ;   DS:SI := @ user buffer
                                978                                           ;
     0174'  =0008               979   IWLOOP: REPT    REPKNT                  ;   Pack the output loop
                                980           LODS    word ptr DS:[SI]        ;   AX := Word from user's buffer
                                981           OUT     DX,AX                   ;   Output that word to sector buffer
                                982           ENDR                            ;   End of repeated section
     0184'  E2 EE               999           LOOP    IWLOOP                  ;   Loop until the buffer is written
                               1000                                           ;
     0186'  1F                 1001           POP     DS                      ;   Restore data segment register
                               1002                                           ;
                               1003   ;       MOV     CH,WRITE                ;   CH := Write command with options
                               1004   ;       MOV     CL,S_DONE               ;   CL := next state (done)
     0187'  B9 A203            1005           MOV     CX,S_DONE or (WRITE shl 8) ;This replaces the above 2 instr's
     018A'  B6 C0              1006           MOV     DH,BUFFDC               ;   DH := bit pattern for BUF --) FDC
                               1007   ;       CALL    INITIO                  ;   Initiate the requested I/O
                               1008   ;       RET                             ;   And return
     018C'  E9 FEC8            1009           JMP     short INITIO            ;   This replaces the above 2 instr's
                               1010   ;*******************************************************************************
                               1011           SUBTTL  Floppy disk GDO routine: FLPRS
                               1012   ;*******************************************************************************
                               1013   ; NAME:          FLPRS - Read Status
                               1014   ; ABSTRACT:      Read the floppy disk controller status and interpret it into a
                               1015   ;                standard (IBM defined) error code.
                               1016   ; PROCESSING:    * Read the FDC status
                               1017   ;                * Synthesize and return a standard (i.e. IBM) error code
                               1018   ; INPUT:         * [D_WKSP+F_LCMD] (byte) = Last command sent to FDC
                               1019   ; OUTPUT:        * [D_REQ+STAT] (byte) status code (unchanged if no error)
                               1020   ; REGISTERS:     AX
                               1021   ; STACK USED:    6 bytes
                               1022   ; NOTES:         * Status on type 1 commands is not checked. We don't expect any
                               1023   ;                  meaningful status from them as no verify options are ever set
                               1024   ;                  for any type 1 command (restore, seek, step).
                               1025   ;                * The code for checking to see if a force interrupt command was
                               1026   ;                  issued has been commented out because this code never issues
                               1027   ;                  a force interrupt command which will actually cause an inter-
                               1028   ;                  rupt (see FRCINT elsewhere in this module).
                               1029   ;*******************************************************************************
     018F'                     1030   FLPRS   PROC    FAR
     018F'  E8 FE8E            1031           CALL    CPUFDC                  ;   Set CPU (--) FDC
```

```
0192'  E4 20                1032            IN      AL,FDCSTA               ;   AL := FDC current status
0194'  8A E0                1033            MOV     AH,AL                   ;   Save status in AH
0196'  A0 0000"             1034            MOV     AL,D_WKSP+F_LCMD        ;   AL := Last FDC command issued
0199'  C6 06 0000" 00       1035            MOV     D_WKSP+F_LCMD,0         ;   Null out last command
019E'  84 C0                1036            TEST    AL,AL                   ;   Q:  Is it a null command?
01A0'  74 2E                1037            JZ      RISRET                  ;       Y:  Don't do anything else
                            1038    ;       CMP     AL,C_FOR                ;   Q:  Is it a force interrupt?
                            1039    ;       JZ      RISRET                  ;       Y:  Don't do anything else
01A2'  A8 80                1040            TEST    AL,10000000B            ;   Q:  Is it a type 1 opcode?
01A4'  74 2A                1041            JZ      RISRET                  ;       Y:  Don't do anything else
01A6'  24 E0                1042            AND     AL,11100000B            ;   Mask down to bits which distinguish
                            1043                                            ;       type 2 & type 3 commands
01A8'  3C A0                1044            CMP     AL,C_WRS                ;   Q:  Was command a write sector?
01AA'  8A C4                1045            MOV     AL,AH                   ;           AL := status
01AC'  75 0C                1046            JNE     T2READ                  ;       N:  Go check bits for read
                            1047                                            ;       Y:  Check all type 2 & 3 bits
01AE'  B4 03                1048            MOV     AH,DKWPRT               ;   AH := Write protect error code
01B0'  A8 40                1049            TEST    AL,M_WPR                ;   Q:  Is write protect bit set?
01B2'  75 18                1050            JNZ     RISDON                  ;       Y:  Return write protect error
01B4'  B4 20                1051            MOV     AH,DKFAIL               ;   AH := Hardware failure error code
01B6'  A8 20                1052            TEST    AL,M_FAU                ;   Q:  Is write fault bit set?
01B8'  75 12                1053            JNZ     RISDON                  ;       Y:  Return hardware failure error
01BA'  B4 08                1054    T2READ: MOV     AH,DKDMAE               ;   AH := Data request errpr
01BC'  A8 04                1055            TEST    AL,M_LOS                ;   Q:  Is lost data bit set?
01BE'  75 0C                1056            JNZ     RISDON                  ;       Y:  Return data request error
01C0'  B4 10                1057            MOV     AH,DKCRCE               ;   AH := CRC error code
01C2'  A8 08                1058            TEST    AL,M_CRC                ;   Q:  Is CRC error bit set?
01C4'  75 06                1059            JNZ     RISDON                  ;       Y:  Return CRC error
01C6'  B4 04                1060            MOV     AH,DKRNFE               ;   AH := Record not found error
01C8'  A8 10                1061            TEST    AL,M_RNF                ;   Q:  Is record not found bit set?
01CA'  74 04                1062            JZ      RISRET                  ;       N:  Then return no error
01CC'  88 26 0000"          1063    RISDON: MOV     D_REQ+STAT,AH           ;       Y:  Return record not found error
01D0'  CB                   1064    RISRET: RET                             ;   And return
                            1065    ;*****************************************************************************
                            1066            SUBTTL  Floppy disk GDO routine: FLPIR
                            1067    ;*****************************************************************************
                            1068    ; NAME:         FLPIR - Initiate Read
                            1069    ; ABSTRACT:     Perform the processing needed to read data from disk
                            1070    ; PROCESSING:   * Set parameters and call INITIO to initiate a read operation
                            1071    ; INPUT:        * ES:BX = @ DIT for the current disk drive
                            1072    ;               * [D_REQ+CYL] (byte) = desired cylinder for the I/O
                            1073    ;               * [D_REQ+TRK] (byte) = desired track for the I/O
                            1074    ;               * [D_REQ+SEC] (byte) = desired sector for the I/O
                            1075    ; OUTPUT:       * [D_NSTA] (byte) = Next state for driver (S_WAIT or S_FRX)
                            1076    ; REGISTERS:    AX, BX, DX
                            1077    ; STACK USED:   10 bytes
                            1078    ;*****************************************************************************
01D1'                       1079    FLPIR   PROC    FAR
                            1080    ;       MOV     CH,READ                 ;   CH := Read command with options
01D1'  B1 04                1081            MOV     CL,S_FRX                ;   CL := next state (floppy read xfer)
                            1082    ;       MOV     DH,FDCBUF               ;   DH := bit pattern for FDC --) BUF
                            1083    ;       CALL    INITIO                  ;   Initiate the requested I/O
```

```
01D3'  E9 FE7D              1084   JNITRD: JMP     near INITRD             ;   This jump accomplishes everything
                            1085                                           ;     which is commented out here
                            1086   ;       RET                             ;   Return
                            1087   ;*****************************************************************************
                            1088           SUBTTL  Floppy disk GDO routine: FLPVFC
                            1089   ;*****************************************************************************
                            1090   ; NAME:          FLPVFC - VeriFy Crc
                            1091   ; ABSTRACT:      Perform the necessary processing to verify a sector CRC
                            1092   ; PROCESSING:    * Set parameters and call INITIO to initiate the reading of the
                            1093   ;                  data (during which the controller will check the CRC).
                            1094   ; INPUT:         * ES:BX = @ DIT for the current disk drive
                            1095   ;                * [D_REQ+CYL] (byte) = desired cylinder for the I/O
                            1096   ;                * [D_REQ+TRK] (byte) = desired track for the I/O
                            1097   ;                * [D_REQ+SEC] (byte) = desired sector for the I/O
                            1098   ; OUTPUT:        * [D_NSTA] (byte) = Next state for driver (S_WAIT or S_DONE)
                            1099   ; REGISTERS:     AX, BX, DX, SI
                            1100   ; STACK USED:    10 bytes
                            1101   ;*****************************************************************************
01D6'                       1102   FLPVFC  PROC    FAR
                            1103   ;       MOV     CH,READ                 ;   CH := Read command with options
01D6'  BA 0E 0005"          1104           MOV     CL,D_NSTA               ;   CL := next state
                            1105   ;       MOV     DH,FDCBUF               ;   DH := bit pattern for FDC --) BUF
                            1106   ;       CALL    INITIO                  ;   Initiate the requested I/O
01DA'  EB F7                1107           JMP     short JNITRD            ;   This jump accomplishes everything
                            1108                                           ;     which is commented out here
                            1109   ;       RET                             ;   Return
                            1110   ;*****************************************************************************
                            1111           SUBTTL  Floppy disk GDO routine: FLPVFD
                            1112   ;*****************************************************************************
                            1113   ; NAME:          FLPVFD - VeriFy Data
                            1114   ; ABSTRACT:      Perform the processing needed to compare data in memory with
                            1115   ;                data on disk.
                            1116   ; PROCESSING:    * Set parameters & call INITIO to inititate the read & verify
                            1117   ;                  of the data
                            1118   ; INPUT:         * ES:BX = @ DIT for the current disk drive
                            1119   ;                * [D_REQ+CYL] (byte) = desired cylinder for the I/O
                            1120   ;                * [D_REQ+TRK] (byte) = desired track for the I/O
                            1121   ;                * [D_REQ+SEC] (byte) = desired sector for the I/O
                            1122   ; OUTPUT:        * [D_NSTA] (byte) = next state for driver (S_WAIT or S_VFD)
                            1123   ; REGISTERS:     AX, DX, SI
                            1124   ; STACK USED:    10 bytes
                            1125   ;*****************************************************************************
01DC'                       1126   FLPVFD  PROC    FAR
                            1127   ;       MOV     CH,READ                 ;   CH := Read command with options
01DC'  B1 05                1128           MOV     CL,S_VFD                ;   CL := next state (verify data)
                            1129   ;       MOV     DH,FDCBUF               ;   DH := bit pattern for FDC --) BUF
                            1130   ;       CALL    INITIO                  ;   Initiate the requested I/O
01DE'  EB F3                1131           JMP     short JNITRD            ;   This jump accomplishes everything
                            1132                                           ;     which is commented out here
                            1133   ;       RET                             ;   Return
                            1134   ;*****************************************************************************
                            1135           SUBTTL  Floppy disk GDO routine: FLPASP
```

```
                                1136  ;*****************************************************************************
                                1137  ; NAME:           FLPASP - Auxiliary State Processor
                                1138  ; ABSTRACT:       Process disk driver states specific to the floppy disk inter-
                                1139  ;                 face. This routine is an extension of the general disk inter-
                                1140  ;                 rupt level code which processes the generic driver states.
                                1141  ; INPUT:          * ES:BX = @ DIT for current disk drive
                                1142  ;                 * [D_STAT] (byte) = Current state of the disk driver
                                1143  ; PROCESSING:     * Process disk driver states specific to the floppy disks
                                1144  ; NOTE:           Hits the floppy disk controller port directly rather than
                                1145  ;                 through the General Disk Operations "next command" interface.
                                1146  ;*****************************************************************************
01E0' 01FC"                     1147  ASPJMP  DW      L_FRX                   ;   VECTOR: Floppy read transfer
01E2' 0205"                     1148          DW      L_VFD                   ;   VECTOR: Verify data
01E4' 0210"                     1149          DW      L_REC1                  ;   VECTOR: Processing error recovery
01E6' 0230"                     1150          DW      L_RDA                   ;   VECTOR: Process read address
01E8'                           1151  FLPASP  PROC    FAR
01E8' A0 0008"                  1152          MOV     AL,D_STAT               ;   AL := current state of driver
01EB' 2C 04                     1153          SUB     AL,S_MAX+1              ;   Adjust for fixed states
01ED' 98                        1154          CBW                             ;   Convert to a word value
01EE' 01 C0                     1155          ADD     AX,AX                   ;   Convert to a jump table index
                                1156  ;       CMP     D_STAT,S_FMAX           ;   Q: Is current state a valid state?
                                1157  ;       JLE     ASPSEL                  ;      Y: Go process the state
                                1158  ;       INT     FATINT                  ;      N: Fatal error
01F0' 8B F8                     1159  ASPSEL: MOV     DI,AX                   ;   DI := jump table index
01F2' A0 0000"                  1160          MOV     AL,D_REQ+STAT           ;   AL := Status code
01F5' 3C 00                     1161          CMP     AL,DKNORM               ;   Q: Did an error occur?
01F7' 2EFF A5 01E0"             1162          JMP     ASPJMP[DI]              ;      Branch to state handler routine
                                1163                                          ;   FRX - Floppy read transfer
01FC' 9C                        1164  L_FRX:  PUSHF                           ;         Save comparison status
01FD' E8 FF11                   1165          CALL    XFRRDD                  ;         Transfer the read data
0200' 9D                        1166          POPF                            ;         Restore comparison status
                                1167  ;       CMP     AL,DKNORM               ;   Q: Did an error occur?
0201' 75 5A                     1168          JNE     FLPEP                   ;      Y: Go do something about it
0203' EB 05                     1169          JMP     short ptr FINISH        ;      N: Then we're done
                                1170                                          ;   VFD - Verify data
                                1171  ;       CMP     AL,DKNORM               ;   Q: Did an error occur?
0205' 75 56                     1172  L_VFD:  JNE     FLPEP                   ;      Y: Go do something about it
0207' E8 FEDE                   1173          CALL    VERIFY                  ;      N: Verify the data
020A' C6 06 0005" 00            1174  FINISH: MOV     D_NSTA,S_IDLE           ;   Finished, Set next state to "idle"
020F' CB                        1175          RET                             ;   And return
                                1176                                          ;   REC1 - Error recovery, stepin
0210' BA 0101                   1177  L_REC1: MOV     DX,RESTOR or (S_WAIT shl 8) ;cmd: restore, state: retry wait
0213' C6 06 0000" 00            1178          MOV     D_REQ+CURCYL,0          ;   Set current cylinder position to 0
0218' C6 06 0000" 00            1179  GO_FDC: MOV     D_REQ+STAT,DKNORM       ;   Reset operation status
021D' E8 FE00                   1180          CALL    CPUFDC                  ;   Set CPU (--) FDC
0220' 8A C2                     1181          MOV     AL,DL                   ;   Put command in AL
0222' E8 FE28                   1182          CALL    HITFDC                  ;   Issue command to FDC
0225' B8 3F40                   1183          MOV     AX,FDCBUF or (MMASK shl 8); AL := code for FDC --) BUF
                                1184                                          ;   AH := interface mode bit mask
0228' E8 FE0D                   1185          CALL    HITC_P                  ;   Set FDC --) BUF
022B' 88 36 0005"               1186          MOV     D_NSTA,DH               ;   Store next state
022F' CB                        1187          RET                             ;   And return
```

```
                                1188                                    ;   RDA - Process Read Address
0230'  B2 02                    1189    L_RDA:  MOV     DL,DKFMTE       ;   DL := Disk format error code
                                1190    ;       CMP     AL,DKNORM       ;   Q: Did an error occur?
0232'  74 08                    1191            JE      CHKADR          ;      N: Go interpret address data
0234'  3C 04                    1192            CMP     AL,DKRNFE       ;   Q: Was record not found?
0236'  74 1B                    1193            JE      STORER          ;      Y: Set disk format error
0238'  3C 10                    1194            CMP     AL,DKCRCE       ;   Q: Was there a CRC error?
023A'  75 2F                    1195            JNE     RECOVR          ;      N: Go attempt error recovery
                                1196                                    ;      Y: Accept CRC error
023C'  E8 FDE5                  1197    CHKADR: CALL    CPUBUF          ;   Set CPU (--) BUF
023F'  E6 22                    1198            OUT     RSTBUF,al       ;   Reset sector buffer
0241'  E5 20                    1199            IN      AX,SECBUF       ;   AL := cur cyl; AH := cur trk
                                1200    ;       MOV     DL,DKFMTE       ;   DL := Disk format error code
0243'  3A 26 0000"              1201            CMP     AH,D_REQ+TRK    ;   Q: Are we on the right trk (side)?
0247'  75 0A                    1202            JNE     STORER          ;      N: Set disk format error
0249'  B2 40                    1203            MOV     DL,DKSEEK       ;   DL := Seek error code
024B'  3A 06 0000"              1204            CMP     AL,D_REQ+CYL    ;   Q: Are we on the right cylinder?
024F'  75 02                    1205            JNE     STORER          ;      N: Set seek error
0251'  B2 04                    1206            MOV     DL,DKRNFE       ;      Y: Set record not found error
0253'  88 16 0000"              1207    STORER: MOV     D_REQ+STAT,DL   ;   Store error code as status
0257'  88 16 0000"              1208            MOV     D_REQ+RSTAT,DL  ;   Also store as the retry status
025B'  EB 0E                    1209            JMP     short RECOVR    ;   Now try to recover from the error
                                1210    ;*******************************************************************************
                                1211            SUBTTL  Floppy disk GDO routine: FLPEP
                                1212    ;*******************************************************************************
                                1213    ; NAME:           FLPEP - Error Processor
                                1214    ; ABSTRACT:       Process floppy disk errors
                                1215    ; INPUT:          * ES:BX = @ DIT for current disk drive
                                1216    ;                 * [D_REQ+STAT] (byte) = error status of operation
                                1217    ; PROCESSING:     * If the error was a timeout, just return a timeout error
                                1218    ;                 * If the error was a record not found:
                                1219    ;                   - Execute read address command
                                1220    ;                   - Process results of that (see FLPASP under L_RDA:)
                                1221    ;                   - Then come back and attempt error recovery:
                                1222    ;                 * Try to recover from other errors:
                                1223    ;                   - Increment errors counter
                                1224    ;                   - Give up if we have retried enough times
                                1225    ;                   - Step read/write head in
                                1226    ;                   - Restore drive to track 0
                                1227    ;                   - Re-issue request
                                1228    ; OUTPUT:         * [D_ERNT] (byte) = incremented count of times an error has
                                1229    ;                   occurred during the current request
                                1230    ;                 * [D_NSTA] = next state for disk driver to enter
                                1231    ; REGISTERS:
                                1232    ; STACK USED:     8 bytes
                                1233    ; NOTES:          * Hits the floppy disk controller port directly rather than
                                1234    ;                   through the General Disk Operations interface.
                                1235    ;*******************************************************************************
025D'                           1236    FLPEP   PROC    FAR
025D'  A0 0000"                 1237    ERRORP: MOV     AL,D_REQ+STAT   ;   AL := error code
0260'  A2 0000"                 1238            MOV     D_REQ+RSTAT,AL  ;   Copy to retry status
0263'  3C 04                    1239            CMP     AL,DKRNFE       ;   Q: Was it record not found?
```

```
0265'  74 16                1240            JE      CHKRNF                  ;       Y: Go pin the error down better
0267'  3C 80                1241            CMP     AL,DKNRDY               ;   Q: Was the error a timeout?
0269'  74 9F                1242            JE      FINISH                  ;       Y: Return the timeout error .
                            1243                                            ;       N: Attempt error recovery:
026B'  FE 06 0003"          1244    RECOVR: INC     D_EKNT                  ;   Increment error counter
026F'  A0 0003"             1245            MOV     AL,D_EKNT               ;   AL := new error count
0272'  263A 47 09           1246            CMP     AL,ES:DITERR[BX]        ;   Q: Have we exceeded error limit?
0276'  7F 92                1247            JG      FINISH                  ;       Y: Give up recovery attempt
0278'  BA 0640              1248    RECAL:  MOV     DX,STEPIN or (S_REC1 shl 8) ;   N: Tell FDC: Step-in
                            1249                                            ;          Next state := recovery state
027B'  EB 9B                1250            JMP     short ptr GO_FDC        ;          Go tell FDC to do it
                            1251    ;*****************************************************************************
027D'  E8 FDA4              1252    CHKRNF: CALL    CPUBUF                  ;   Set CPU (--) BUF
0280'  E6 22                1253            OUT     RSTBUF,AL               ;   Reset sector buffer
0282'  BA 07C0              1254            MOV     DX,C_RDA or (S_RDA SHL 8) ; Tell FDC: Read address
                            1255                                            ;   Next state := process read address
0285'  EB 91                1256            JMP     short ptr GO_FDC        ;   Go tell FDC to do that
                            1257    ;*****************************************************************************
                            1258            SUBTTL  Floppy disk QDO routine: FLPTER
                            1259    ;*****************************************************************************
                            1260    ; NAME:         FLPTER - TERmination processing
                            1261    ; ABSTRACT:     Perform device-specific termination processing for request
                            1262    ; PROCESSING:   * Insure that FDC is idle
                            1263    ;               * Deselect all drives so that all drive select lights are off
                            1264    ;               * Start the motor-off timeout event which will turn off the
                            1265    ;                 drive motors if nothing else is done until much later.
                            1266    ; REGISTERS:    AX
                            1267    ; STACK USED:   6 bytes
                            1268    ;*****************************************************************************
0287'                       1269    FLPTER  PROC    FAR
0287'  E8 FDA0              1270            CALL    FRCINT                  ;   Terminate FDC
                            1271+           EVENT   FL1EVT,MOFTIM           ;   Start motor-off timeout event
0295'  A0 000B"             1277            MOV     AL,DSKS_M               ;   AL := Image of motor/select port
0298'  0C 0F                1278            OR      AL,DRVMSK               ;   Or in deselect all drives mask
029A'  E8 FDAA              1279            CALL    HITS_P                  ;   Hit motor/select port with that
029D'  CB                   1280            RET
                            1281    ;*****************************************************************************
                            1282            SUBTTL  Floppy disk QDO routine: FLPBGN
                            1283    ;*****************************************************************************
                            1284    ; NAME:         FLPBGN - BeGIN processing request
                            1285    ; ABSTRACT:     Performs any initial processing for a disk I/O request
                            1286    ; PROCESSING:   * Cancel the motor-off timeout event
                            1287    ; REGISTERS:    none
                            1288    ; STACK USED:   4 bytes
                            1289    ;*****************************************************************************
029E'                       1290    FLPBGN  PROC    FAR
                            1291+           KILEVT  FL1EVT                  ;   Cancel motor-off event
02A3'  CB                   1293            RET
                            1294    ;*****************************************************************************
                            1295            SUBTTL
                            1296            END
```

No errors detected

```
                                 1  ;*****************************************************************************
                                 2  ; TITLE    - HEADER - THIS MODULE MUST BE THE FIRST IN ROM.
                                 3  ; COMPUTER - 8088 ASSEMBLY LANGUAGE
                                 4  ; ABSTRACT - This module contains the System ROM header information.  It
                                 5  ;       must be the first module linked, so that it is at offset 0000
                                 6  ;       relative to the beginning of the ROM.  Each ROM in the system
                                 7  ;       must contain a header with this structure:
                                 8  ;
                                 9  ;               DW      xxxx            ; Size of the ROM (OFFFFH for System)
                                10  ;               DW      xxxx            ; Entry point of the ROM
                                11  ;               DB      xx              ; Length of ID message
                                12  ;               DB      CR,LF           ; First characters of message
                                13  ;               DB      'Vx.xx'         ; 5-byte ROM version in ASCII
                                14  ;               DB      xx,xx,...       ; Remainder of ID message
                                15  ;
                                16  ; REGISTERS USED - NONE
                                17  ;
                                18  ; STACK USED - NONE
                                19  ;
                                20  ;*****************************************************************************
                                21          NAME    HEADER
                                22          SUBTTL  SYSTEM ROM HEADER
                                24  ;*****************************************************************************
                                25  ;                       PUBLIC DEFINITIONS
                                26  ;*****************************************************************************
                                27  ;
                                28          PUBLIC  ROMVER
                                29  ;
                                30  ;*****************************************************************************
                                31  ;                       EXTERNAL REFERENCES
                                32  ;*****************************************************************************
                                33          EXTERN  PUPTST:NEAR             ;ENTRY POINT OF ROM
                                34          EXTERN  ROMSIZ:ABS              ;ROM SIZE
                                35  ;*****************************************************************************
                                36  ;                       LOCAL CONSTANTS
                                37  ;*****************************************************************************
                                38  ;
=000D                           39  CR      EQU     0DH
=000A                           40  LF      EQU     0AH
                                41  ;*****************************************************************************
                                42  ;                       MODULE ENTRY POINT
                                43  ;*****************************************************************************
=0000                           44          SECT    ROMCOD
                                45          ASSUME  CS:ROMCOD
                                46  ;
0000'  FFFF                     47          DW      0FFFFH                  ; Size of this ROM (Special for system)
0002'  0001"                    48          DW      OFFSET PUPTST           ; Entry point of this ROM
0004'  3A                       49          DB      OFFSET MSGSIZ           ; Size of the sign on message
0005'  0D 0A                   50  ROMMSG  DB      CR,LF
0007'  56 31 2E 32 33           51  ROMVER  DB      'V1.23'                 ; Version of the ROM
000C'  20 53 59 53 52 4F        52          DB      ' SYSROM (c) Copyright Texas Instruments Inc. 1983'
0012'  4D 20 28 63 29 20
```

```
0018'  43 6F 70 79 72 69
001E'  67 68 74 20 54 65
0024'  78 61 73 20 49 6E
002A'  73 74 72 75 6D 65
0030'  6E 74 73 20 49 6E
0036'  63 2E 20 31 39 38
003C'  33
003D'  0D 0A                    53            DB        CR,LF
       =003A                    54    MSGSIZ  EQU       $-ROMMSG
                                55    ;
                                56            END
```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ;
                        3   ;  TITLE    - INTTST - Interrupt controller power up diagnostics.
                        4   ;  COMPUTER - 8088 ASSEMBLY LANGUAGE
                        5   ;  ABSTRACT - This module contains the power up diagnostics that test the
                        6   ;         interrupt controller in the PEGASUS main board.  Only those interrupts
                        7   ;         that are generated by basic system components are tested.
                        8   ;*****************************************************************************
                        9           NAME    INTTST - PEGASUS power up interrupt controller test
                       10   ;       SUBTTL
                       12   ;*****************************************************************************
                       13   ;                           EXTERNAL PROCEDURAL REFERENCES
                       14   ;*****************************************************************************
                       15   ;
=0000                  16           SECT    ROMDAT
                       17           ASSUME  CS:ROMCOD
                       18   ;
                       19           EXTERN  DSKC_M:BYTE     ; DISK SELECT LATCH MEMORY IMAGE.       (ROMDAT)
                       20           EXTERN  PRCO_M:BYTE     ; PRINTER OUTPUT LATCH MEMORY IMAGE.    (ROMDAT)
                       21   ;*****************************************************************************
                       22   ;                           EXTERNAL DATA REFERENCES
                       23   ;*****************************************************************************
                       24   ;
=0000                  25           SECT    ROMCOD
                       26   ;
                       27           EXTERN  DSPDIE:NEAR     ; DISPLAY FATAL ERROR and DIE.          (OUTPUT)
                       28           EXTERN  FILVEC:NEAR     ; ENTRY TO VECTOR INITIALIZATION.       (VECINI)
                       29           EXTERN  DECLED:NEAR     ; DECREMENT LED SUBROUTINE.             (DECLED)
                       30           EXTERN  KEYRST:NEAR     ; KEYBOARD INTERRUPT RESET.             (KEYDSR)
                       31           EXTERN  FRCINT:NEAR     ; FLOPPY CONTROLLER RESET.              (DSKDSR)
                       32           EXTERN  TIMTST:NEAR     ; TIMER DIAGNOSTICS ERROR.              (TIMTST)
                       33           EXTERN  ICNERR:ABS      ; INTERRUPT CONTROLLER ERROR CODE.      (TSTERR)
                       34           EXTERN  IVIERR:ABS      ; INVALID INTERRUPT ERROR.              (TSTERR)
                       35           EXTERN  LED100:ABS      ; INTERRUPT LED ERROR PATTERN           (ROMERR)
                       36           EXTERN  NMIERR:ABS      ; NMI INTERRUPT FAILURE CODE.           (TSTERR)
                       37           EXTERN  TIMERR:ABS      ; TIMER INT FAILURE CODE.               (TSTERR)
                       38           EXTERN  KEYERR:ABS      ; KEYBOARD INT FAILURE CODE.            (TSTERR)
                       39           EXTERN  FLIERR:ABS      ; FLOPPY INT FAILURE CODE.              (TSTERR)
                       40   ;*****************************************************************************
                       41   ;                           PUBLIC  DEFINITIONS
                       42   ;*****************************************************************************
                       43           PUBLIC  INTTST          ; ENTRY TO INTERRUPT CONTROLLER DIAGNOSTICS.
                       44           PUBLIC  PRTRST          ; PRINTER AND PARITY INTERRUPT DISABLE SUBR.
                       45   ;*****************************************************************************
                       46   ;                           LOCAL CONSTANTS
                       47   ;       INCLUDE  PEQ:HARDWARE.EQU
                       48   ;       INCLUDE  PEQ:VECTOR.EQU
                       49   ;*****************************************************************************
                       50   ;
                      329   ;*****************************************************************************
                      330   ;                           BASE PAGE LOCATIONS
                      331   ;*****************************************************************************
```

```
                        332   ;
=0000                   333         SECT        ROMCOD
                        334         ASSUME      CS:ROMCOD, DS:ROMDAT, ES:ABS0
```

```
0000'  B2 09*           336  IMRER:  MOV     DL,ICNERR               ;;; SET ERROR CODE INTO DL
0002'  E9 00F2          337          JMP     FATAL                   ;;; AND REPORT ERROR
                        338  ;
                        339  ; This is the entry to the interrupt controller diagnostics.  First, the
                        340  ; controller is initialized along with all the interrupt vectors in memory.
                        341  ; The IMR register is tested with interrupts disabled.  With all vectors
                        342  ; set to "Invalid interrupt" the basic interrupts are unmasked.  If no
                        343  ; invalid interrupts occur, the test proceeds.
                        344  ;
                        345  ;       Disable speaker and timer interrupts.
                        346  ;
0005'  B0 1B            347  INTTST: MOV     AL,(ICW1 + IC_IW4 + IC_SNQ + IC_LTM) ; WRITE ICW1.
0007'  E6 18            348          OUT     INTA00,AL               ;;;
0009'  B0 40            349          MOV     AL,IROINT               ;;;   Hardware interrupts 40h-)47h
000B'  BA 0019          350          MOV     DX,INTA01               ;;;   Point to controller mask reg.
000E'  EE               351          OUT     DX,AL                   ;;;
000F'  B0 01            352          MOV     AL,(ICW4 + IC_M86)      ;;;   Select 8086 mode.
0011'  EE               353          OUT     DX,AL                   ;;;
0012'  B0 F0            354          MOV     AL,0F0H                 ;;;   Try an interrupt mask.
0014'  EE               355          OUT     DX,AL                   ;;;
0015'  EC               356          IN      AL,DX                   ;;;   Read interrupt mask.
0016'  3C F0            357          CMP     AL,0F0H                 ;;; Q: Is mask correct ?
0018'  75 E6            358          JNE     IMRER                   ;;;   No, fatal error.
001A'  B0 CC            359          MOV     AL,0CCH                 ;;;   Yes, try another mask.
001C'  EE               360          OUT     DX,AL                   ;;;
001D'  EC               361          IN      AL,DX                   ;;;   Read mask.
001E'  3C CC            362          CMP     AL,0CCH                 ;;; Q:  IS MASK CORRECT ?
0020'  75 DE            363          JNE     IMRER                   ;;;   NO, FATAL ERROR.
0022'  B0 55            364          MOV     AL,055H                 ;;;   Yes, try another mask.
0024'  EE               365          OUT     DX,AL                   ;;;
0025'  EC               366          IN      AL,DX                   ;;;   Read mask.
0026'  3C 55            367          CMP     AL,055H                 ;;; Q:  IS MASK CORRECT ?
0028'  75 D6            368          JNE     IMRER                   ;;;   NO, FATAL ERROR.
002A'  B0 AA            369          MOV     AL,0AAH                 ;;;   Yes, try another mask.
002C'  EE               370          OUT     DX,AL                   ;;;
002D'  EC               371          IN      AL,DX                   ;;;   Read mask.
002E'  3C AA            372          CMP     AL,0AAH                 ;;; Q:  IS MASK CORRECT ?
0030'  75 CE            373          JNE     IMRER                   ;;;   NO, FATAL ERROR.
0032'  B0 FF            374          MOV     AL,0FFH                 ;;;   MASK ALL INTERRUPTS.
0034'  EE               375          OUT     DX,AL                   ;;;
0035'  E8 00CA          376          CALL    PRTRST                  ;;;   RESET PRINTER
0038'  E8 00D2          377          CALL    TIMRST                  ;;;   RESET TIMER DEVICES.
003B'  E8 00BE          378          CALL    FLPRST                  ;;;   RESET FLOPPY CONTROLLER.
003E'  E8 0006"         379          CALL    KEYRST                  ;;;   RESET KEYBOARD UART INTERFACE.
0041'  B9 0162"         380          MOV     CX,OFFSET INVINT        ;;;   SET ADDRESS INVALID INTR VECTOR.
0044'  E8 0004"         381          CALL    FILVEC                  ;;;   SET INTERRUPT VECTORS TO INVALID.
0047'  30 D2            382          XOR     DL,DL                   ;;; SET INITIAL ERROR FLAG
0049'  FB               383          STI                             ;   ENABLE INTERRUPTS.
004A'  90               384          NOP                             ;   WINDOW
004B'  90               385          NOP                             ;   WINDOW
004C'  FA               386          CLI                             ;;;   DISABLE INTERRUPTS.
                        387  ;
```

```
                                388   ;         NOW TO CAUSE INDIVIDUAL INTERRUPTS IN THE FOLLOWING ORDER
                                389   ;         NMI, TIMERS, FLOPPY, KEYBOARD. THE NMI IS TESTED WITH THE
                                390   ;         CRT AND PARITY INTERRUPTS DISABLED.
                                391   ;
004D'  268B 0E 0008             392             MOV     CX,ES:WORD PTR (NMIINT*4) ;;; SAVE CURRENT NMI VECTOR.
0052'  26C7 06 0008 0134"      393             MOV     ES:WORD PTR (NMIINT*4),OFFSET NMIVCT ;;; NMI INTERRUPT VECTOR.
0059'  B0 22                   394             MOV     AL,(UC_RTS + UC_DTR)    ;;; RAISE DTR TO CAUSE NMI.
005B'  E6 11                   395             OUT     KBCMD,AL                ;;; DO IT.
005D'  B0 20                   396             MOV     AL,UC_RTS               ;;; RESTORE NMI DISABLE, JUST IN CASE.
005F'  E6 11                   397             OUT     KBCMD,AL                ;;;
0061'  2689 0E 0008            398             MOV     ES:WORD PTR (NMIINT*4),CX ;;; RESTORE NMI VECTOR.
0066'  80 0E 0002" 08          399             OR      PRC0_M,PARIEN           ;;; RE-ENABLE PARITY NMI INTERRUPT.
006B'  A0 0002"                400             MOV     AL,PRC0_M               ;;;
006E'  E6 03                   401             OUT     PRC0_P,AL               ;;;
                                402   ;
                                403   ;         NOW FOR TIMER INTERRRUPT. ONLY TIMER 1 IS TESTED SINCE TIMER2'S
                                404   ;         INTERRUPT IS SHARED WITH FOREIGN DEVICES ON THE BUS.
                                405   ;
0070'  B0 F7                   406             MOV     AL,ENAB3                ;;; ENABLE SYSTEM TIMER INTERRUPT.
0072'  E6 19                   407             OUT     INTA01,AL               ;;;
0074'  FB                      408             STI                             ;;; ALLOW INVALID INTERRUPT TO OCCUR
0075'  80 0E 0001" 02          409             OR      DSKC_M,TMR1EN           ;   SET TIMER INT ENABLE LATCH.
007A'  A0 0001"                410             MOV     AL,DSKC_M               ;
007D'  26C7 06 010C 013C"      411             MOV     ES:WORD PTR (IR3INT*4),OFFSET TIMRIN ; SET UP CORRECT VECTOR.
0084'  E6 00                   412             OUT     DSKC_P,AL               ; ENABLE TIMER 1.
0086'  B0 76                   413             MOV AL,(TC_SC1+TC_WRD+TC_MD3)   ; SET UP SYSTEM TIMER.
0088'  E6 17                   414             OUT     TIMCMD,AL               ; WRITE TO TIMER COMMAND REG.
008A'  B0 02                   415             MOV     AL,02                   ; LOW BYTE OF COUNT.
008C'  E6 15                   416             OUT     TIMER1,AL               ;
008E'  B0 00                   417             MOV     AL,00                   ; HIGH BYTE OF COUNT.
0090'  E6 15                   418             OUT     TIMER1,AL               ;
0092'  B3 03                   419             MOV     BL,03                   ; LOOP FOR AT LEAST 15 MICROSECS
0094'  FE CB                   420   DECLOP:   DEC     BL                      ;
0096'  75 FC                   421             JNE     DECLOP                  ;
0098'  FA                      422             CLI                             ;;;
0099'  26C7 06 010C 0162"      423   QOTIM:    MOV     ES:WORD PTR (IR3INT*4),OFFSET INVINT ;;; RESTORE INVALID VECTOR
00A0'  80 26 0001" FD          424             AND     DSKC_M,NOT TMR1EN       ;;; DISABLE TIMER INTERRUPT.
00A5'  B0 02                   425             MOV     AL,TMR1EN               ;;;
00A7'  E6 00                   426             OUT     DSKC_P,AL               ;;;
                                427   ;
                                428   ;         NOW FOR DISK CONTROLLER INTERRUPT. FIRST, THE INTERRUPT IS
                                429   ;         ENABLED TO DETERMINE IF AN INTERRUPT FROM THE BUS IS PENDING.
                                430   ;         IF SO, IT IS INVALID.
                                431   ;
00A9'  B0 BF                   432             MOV     AL,ENAB6                ;;;     SET ENABLE OF DISK INTERRUPT.
00AB'  E6 19                   433             OUT     INTA01,AL               ;;;     WRITE TO INT CONTROLLER.
00AD'  B9 4B1E                 434             MOV     CX,4B1EH                ;;;     DELAY 100 MS.
00B0'  FB                      435             STI                             ;
00B1'  E2 FE                   436             LOOP    $                       ;       WAIT JUST IN CASE.
00B3'  FA                      437             CLI                             ;;;
00B4'                          438   QOTST:
00B4'  26C7 06 0118 014E"      439             MOV     ES:WORD PTR (IR6INT*4),OFFSET FLPINT
```

```
00BB'  FB                      440           STI                                 ;
00BC'  E8 0059                 441           CALL    FORCIT                      ;     FORCE FLOPPY INTERRUPT.
00BF'  FA                      442           CLI                                 ;;;
00C0'  26C7 06 0118 0162"      443           MOV     ES:WORD PTR (IR6INT*4),OFFSET INVINT ;;; RESTORE VECTOR.
                               444   ;
                               445   ;       NOW TO TRY KEYBOARD INTERRUPT.
                               446   ;
00C7'                          447   KEYTST:
00C7'  B0 7F                   448           MOV     AL,ENAB7                    ;;;    ENABLE FOR KEYBOARD INT.
00C9'  E6 19                   449           OUT     INTA01,AL                   ;;;
00CB'  FB                      450           STI                                 ;;;     CAUSE INVALID INTERRUPT
00CC'  90                      451           NOP                                 ;
00CD'  90                      452           NOP                                 ;
00CE'  FA                      453           CLI                                 ;;;     IF IT OCCURS.
00CF'  26C7 06 011C 0159"      454           MOV     ES:WORD PTR (IR7INT*4),OFFSET KBDINT ;;;
00D6'  B0 21                   455           MOV     AL,(UC_RTS + UC_TXE)        ;;;     CAUSE KEYBOARD INTERRUPT.
00D8'  E6 11                   456           OUT     KBCMD,AL                    ;;;
00DA'  FB                      457           STI                                 ;;;
00DB'  B9 0014                 458           MOV     CX,20                       ;       LOOP FOR A WHILE.
00DE'  E2 FE                   459           LOOP    $                           ;
00E0'  FA                      460           CLI                                 ;;;
00E1'  B0 20                   461           MOV     AL,UC_RTS                   ;;;
00E3'  E6 11                   462           OUT     KBCMD,AL                    ;;;     RESTORE KEYBOARD UART.
00E5'  B0 FF                   463           MOV     AL,0FFH                     ;;;     DISABLE ALL INT IN MASK.
00E7'  E6 19                   464           OUT     INTA01,AL                   ;;;
00E9'  26C7 06 011C 0162"      465           MOV     ES:WORD PTR (IR7INT*4),OFFSET INVINT ;;;
                               466   ;
                               467   ;       GET AND REPORT ON ANY ERRORS
                               468   ;
00F0'  B0 1E                   469           MOV     AL,00011110B                ;;; AND OF NMI, TIMER, KEY AND FLOPPY
                               470   ;                                           ;;; INTERRUPT OCCURRENCES
00F2'  30 C2                   471           XOR     DL,AL                       ;;; SETS CORRECT BIT IN ERROR IF NO
                               472   ;                                           ;;; INTERRUPT OCCURRED.
00F4'  E9 0008"                473           JMP     TIMTST                      ;;; JUMP AND CONTINUE TESTING
                               474   ;                                           ;;; (TIMER TEST REPORTS ERRORS)
                               475   ;       COMMON FATAL EXIT FOR ALL INTERRUPT ERRORS.
                               476   ;
00F7'                          477   FATAL:
00F7'  B0 0B*                  478           MOV     AL,LED100                   ;   SET UP LED PATTERN FOR DSPDIE
00F9'  E9 0003"                479           JMP     DSPDIE                      ;
                               480   ;
                               481   ;       ROUTINE TO DISABLE FLOPPY CONTROLLER.
                               482   ;
00FC'                          483   FLPRST:
00FC'  E8 0007"                484           CALL    FRCINT                      ;       MAGIC CODE TO RESET FLOPPY
00FF'  E4 20                   485           IN      AL,FDCSTA                   ;       DISK CONTROLLER.
0101'  C3                      486           RET
                               487   ;
                               488   ;       ROUTINES TO DISABLE PRINTER, PARITY AND TIMER INTERRUPTS.
                               489   ;
0102'                          490   PRTRST:
0102'  80 26 0002" 77         491           AND     PRCO_M,NOT (PARIEN+PRINEN)      ;   DISABLE PARITY INTERRUPT.
```

```
0107'  A0 0002"                492             MOV     AL,PRC0_M               ;   DISABLE PARALLEL PORT INTERRUPT.
010A'  E6 03                   493             OUT     PRC0_P,AL               ;
010C'  C3                      494             RET
                               495     ;
010D'                          496     TIMRST:
010D'  80 26 0001" F9          497             AND     DSKC_M,NOT (TMR1EN+TMR2EN)      ;;; Disable timer 1 and 2.
0112'  A0 0001"                498             MOV     AL,DSKC_M               ;;;
0115'  E6 00                   499             OUT     DSKC_P,AL               ;;;
0117'  C3                      500             RET
                               501     ;
0118'                          502     FORCIT:
0118'  9C                      503             PUSHF                           ; Save carry and interrupt status.
0119'  FA                      504             CLI                             ;
011A'  A0 0001"                505             MOV     AL,DSKC_M               ;;; READ CURRENT MEMORY STATE.
011D'  24 3F                   506             AND     AL,MMASK                ;;; MASK OFF CONTROL BITS.
011F'  0C 80                   507             OR      AL,ACCFDC               ;;; SET UP FOR CPU (--) FDC.
0121'  E6 00                   508             OUT     DSKC_P,AL               ;;; SET THE LATCH.
0123'  A2 0001"                509             MOV     DSKC_M,AL               ;;; UPDATE MEMORY STATE.
0126'  9D                      510             POPF                            ;;; RESTORE INTERRUPT STATUS.
0127'  B0 D8                   511             MOV     AL,IMMINT               ;   CAUSE IMMEDIATE INTERRUPT.
0129'  E6 20                   512             OUT     FDCCMD,AL               ;
012B'  B0 0A                   513             MOV     AL,10                   ;   WAIT FOR CMD TO EXECUTE.
012D'                          514     INTDLY:
012D'  FE C8                   515             DEC     AL                      ;
012F'  75 FC                   516             JNZ     INTDLY                  ;
0131'  E4 20                   517             IN      AL,FDCSTA               ;   READ STATUS TO CLEAR INTERRUPT.
0133'  C3                      518             RET
                               519     ;
                               520     ;       ENTRY POINT FOR OCCURRENCE OF NMI
                               521     ;
0134'  B0 20                   522     NMIVCT: MOV     AL,UC_RTS               ;;; DISABLE FURTHER NMI'S.
0136'  E6 11                   523             OUT     KBCMD,AL                ;;;
0138'  80 CA 0C*               524             OR      DL,NMIERR               ;;; INDICATE CORRECT OCCURRENCE.
013B'  CF                      525             IRET
                               526     ;
                               527     ;       ENTRY FOR OCCURENCE OF TIMER INTERRUPT
                               528     ;
013C'  80 26 0001" FD          529     TIMRIN: AND     DSKC_M,NOT TMR1EN       ;;; DISABLE TIMER INTERRUPT.
0141'  A0 0001"                530             MOV     AL,DSKC_M               ;;;
0144'  E6 00                   531             OUT     DSKC_P,AL               ;;;
0146'  80 CA 0D*               532             OR      DL,TIMERR               ;;; INDICATE CORRECT OCCURRENCE
0149'  B0 20                   533     COMNI:  MOV     AL,EOI                  ;;;
014B'  E6 18                   534             OUT     INTA00,AL               ;;;
014D'  CF                      535             IRET
                               536     ;
                               537     ;       ENTRY FOR FLOPPY INTERRUPT
                               538     ;
014E'  E4 20                   539     FLPINT: IN      AL,FDCSTA               ;;; READ FLOPPY STATUS TO RESET INT.
0150'  B0 D0                   540             MOV     AL,FORCED               ;;; RESET FLOPPY INTERRUPT.
0152'  E6 20                   541             OUT     FDCCMD,AL               ;;;
0154'  80 CA 0F*               542             OR      DL,FLIERR               ;;; INDICATE CORRECT OCCURRENCE
0157'  EB F0                   543             JMP     COMNI                   ;;;
```

```
                        544   ;
                        545   ;       ENTRY FOR KEYBOARD INTERRUPT.
                        546   ;
0159'  B0 20            547   KBDINT: MOV     AL,(UC_RTS)             ;;;     DISABLE KEY INT.
015B'  E6 11            548           OUT     KBCMD,AL                ;;;
015D'  80 CA 0E*        549           OR      DL,KEYERR               ;;; INDICATE CORRECT OCCURRENCE
0160'  EB E7            550           JMP     COMNI                   ;;; COMMON INTERRUPT EXIT
                        551   ;
                        552   ;       ENTRY FOR AN INVALID INTERRUPT
                        553   ;
0162'  F6 D2            554   INVINT: NOT     DL                      ;;; SETS ANY UNTESTED BITS IN ERROR.
                        555   ;                                       ;;; NUMBER OF GOOD BITS INDICATES
                        556   ;                                       ;;; PROGRESS THROUGH THE TEST
                        557   ;       OR      DL,IVIERR               ;;; SIGNAL ERROR
0164'  EB 91            558           JMP     FATAL                   ;;; JUMP AND REPORT ERROR
                        559           END

No errors detected
```

```
                                1   ;*****************************************************************************
                                2   ; TITLE    - INTXIT (COMMON INTERRUPT EXIT LOGIC)
                                3   ; COMPUTER - 8088 ASSEMBLY LANGUAGE
                                4   ; ABSTRACT - This module contains the common interrupt exit logic executed
                                5   ;       by all interrupt processors.  The address of this routine is stored
                                6   ;       in an interrupt vector location (interrupt XITVEC) in segment ABSO
                                7   ;       (Absolute Zero) so that it is accessible to application programs.
                                8   ;       This logic decrements the interrupt counter and saves the location
                                9   ;       of the stack of the interrupted logic.  The interrupt exit logic
                               10   ;       then restores commonly used registers from the interrupt stack,
                               11   ;       switches back to the stack of the interrupted logic, and restores
                               12   ;       register DS from the user stack.
                               13   ;
                               14   ;       Also contained in this module is the dummy entry point used by
                               15   ;       soft interrupts SLCINT and TIMINT when ZEX is not present in the
                               16   ;       system.  This logic simply does an IRET.
                               17   ;
                               18   ; INPUTS   - ES,BX = SS,SP of interrupted logic
                               19   ;            Interrupt stack contains saved ES,BX,AX (ES at top of stack).
                               20   ;            Stack of interrupted logic contains saved DS.
                               21   ;
                               22   ; OUTPUTS  -
                               23   ;
                               24   ; REGISTERS USED -
                               25   ;
                               26   ; STACK USED -
                               27   ;
                               28   ;*****************************************************************************
                               29           NAME    INTXIT
                               30           SUBTTL
                               32   ;*****************************************************************************
                               33   ;                           PUBLIC DEFINITIONS
                               34   ;*****************************************************************************
                               35
                               36           PUBLIC  INTRET
                               37           PUBLIC  INTXIT
                               38
                               39   ;*****************************************************************************
                               40   ;                           EXTERNAL REFERENCES
                               41   ;*****************************************************************************
                               42
=0000                          43           SECT    ROMDAT
                               44           EXTERN  IXSPSV:WORD     ;       Where Interrupt Exit saves SP  (ROMDAT)
                               45           EXTERN  IXSSSV:WORD     ;       Where Interrupt Exit saves SS  (ROMDAT)
```

```
                        47  ;*****************************************************************************
                        48  ;                           LOCAL CONSTANTS
                        49  ;       INCLUDE PEQ:VECTOR.EQU
                        50  ;       INCLUDE PEQ:INTCTLR.EQU
                        51  ;*****************************************************************************
                       173          SUBTTL  MAIN
                       174  ;*****************************************************************************
                       175  ;                           MODULE ENTRY POINT
                       176  ;*****************************************************************************
                       177
      =0000            178          SECT    ROMCOD
                       179          ASSUME  CS:ROMCOD
                       180          ASSUME  DS:NOTHING
                       181
0000'                  182  INTXIT  PROC    FAR                     ;       ES,BX = ORIGINAL SS,SP
0000'  FA              183          CLI                             ;       DISABLE INTERRUPTS
0001'  B0 20           184          MOV     AL,EOI                  ;;;     RESET 8259
0003'  E6 18           185          OUT     INTA00,AL               ;;;
0005'  2EFE 0E C19A    186          DEC     BYTE PTR CS:INTCTR+CSWRAP;;;    DECREMENT INTERRUPT COUNTER
                       187
                       188          ASSUME  DS:ROMDAT
000A'  2E8E 1E C180    189          MOV     DS,WORD PTR CS:DSADDR+CSWRAP;;; DS = ROM DATA SEGMENT
000F'  8C 06 0002"     190          MOV     IXSSSV,ES               ;;;     SAVE SS,SP OF ORIGINAL LOGIC
0013'  89 1E 0001"     191          MOV     IXSPSV,BX               ;;;
0017'  07              192          POP     ES                      ;;;     RESTORE COMMONLY USED REGISTERS
0018'  5B              193          POP     BX                      ;;;       FROM INTERRUPT STACK
0019'  58              194          POP     AX                      ;;;
001A'  8E 16 0002"     195          MOV     SS,IXSSSV               ;;;     RESTORE ORIGINAL SS,SP
001E'  8B 26 0001"     196          MOV     SP,IXSPSV               ;;;
0022'  1F              197          POP     DS                      ;;;     RESTORE DS FROM ORIGINAL STACK
                       198  ;       IRET                            ;;;     ** INTERRUPT RETURN **
                       199
                       200
                       201  ;*****************************************************************************
                       202  ;
                       203  ; INTRET is the logic that is executed by a soft interrupts to SLCINT
                       204  ; and TIMINT
                       205  ;
                       206  ;*****************************************************************************
                       207
0023'                  208  INTRET  PROC    FAR                     ;       Dummy soft interrupt
0023'  CF              209          IRET                            ;       ** INTERRUPT RETURN **
                       210
                       211          SUBTTL
                       212          END
```

No errors detected

```
                ;*****************************************************************************
                ;  TITLE    - KBTABL - Standard keyboard encoding tables
                ;  COMPUTER - 8088 ASSEMBLY LANGUAGE  (BSO Assembler)
                ;  ABSTRACT - This module contains the data tables used by the Pegasus
                ;       keyboard routines for standard encoding of the keyboard.
                ;       As explained in the keyboard driver module, the keyboard sends
                ;       one or two bytes per key depression.  The first contains the
                ;       mode key information (CTRL, SHIFT, etc.).  The second is the
                ;       'scan code' corresponding to the key's physical position on the
                ;       keyboard.  These tables are arranged in order of ascending
                ;       scan codes.  The mode determination is handled in the keyboard
                ;       DSR code and the correct table accessed for encoding.  The
                ;       keyboard layout charts below map the physical locations of
                ;       the keys to the scan codes.  If the byte looked up is -1 (OFFH),
                ;       the key is to be ignored.  If the byte is greater than 7FH, but
                ;       less than OFCH (i.e. the high bit is set), then the low order 7
                ;       bits are used as an index into the KBFUNC table to retrieve
                ;       the function key code.  The other possible values are defined
                ;       as follows:  OFEH is the Program Pause function, OFDH is the
                ;       Program Break function, and OFCH is the Print Screen function.
                ;
                ;*****************************************************************************
                        NAME    KBTABL - Standard keyboard encoding tables
                ;       SUBTTL
                ;*****************************************************************************
                ;                       PUBLIC DEFINITIONS
                ;*****************************************************************************
                        PUBLIC  KB_ALT
                        PUBLIC  KBCTRL
                        PUBLIC  KBFUNC
                        PUBLIC  KBNUMK
                        PUBLIC  KBSHFT
                        PUBLIC  KBUNSH
                ;
                ;*****************************************************************************
                ;                       LOCAL CONSTANTS
                ;*****************************************************************************
                ;       INCLUDE     PEQ:ASCII.EQU
                ;       INCLUDE     PEQ:KBSPCL.EQU
                ;       INCLUDE     PEQ:KBMISC.EQU
                ;
                ;       GENERAL KEYBOARD POSITION NUMBERING:
                ;
                ;                   (left half)
                ;
                ;       ----------------    ----------------    ----------------
                ;       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
                ;       |101|102|103|104|   | 1 | 2 | 3 | 4 |   | 5 | 6 | 7 | 8 |
                ;       ----------------    ----------------    ----------------
                ;
                ;       ---------------------------------------------------------
```

```
;       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
;       |65 | 9 |10 |11 |12 |13 |14 |15 |16 |17 |18 |19 |20 |21 |  22 |
;       -----------------------------------------------------------------
;       |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
;       | 49  |50 |51 |52 |53 |54 |55 |56 |57 |58 |59 |60 |61 |   |  62 |
;       -------------------------------------------------------------+   |----
;       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |      |   |
;       |***|***|66 |67 |68 |69 |70 |71 |72 |73 |74 |75 |76 | 77|  78 |
;       -----------------------------------------------------------------
;       |   |     |   |   |   |   |   |   |   |   |   |   |   |     |
;       |***| *** |82 |83 |84 |85 |86 |87 |88 |89 |91 |92 | *** |
;       -------------------------------------------------------
;                       |                               |
;                       |               81              |
;                       ---------------------------------
;
;                                                       ----------------
;                                                       |   |   |   |   |
;   (right half)                                        |47 |48 |100|90 |
;                                                       ----------------
;                                                       ----------------
;                                                       |   |   |   |   |
;                                                       |23 |24 |25 |26 |
;                           -----                       ----------------
;                       (63)|   |(45)                   |   |   |   |   |
;                           |64 |                       |39 |40 |33 |34 |
;                        -------------                  ----------------
;                        |   |   |   |                  |   |   |   |   |
;                        |79 |80 |46 |                  |31 |32 |41 |42 |
;                        -------------                  ----------------
;                            |   |                      |   |   |   |   |
;                       (95) |96 | (94)                 |27 |35 |43 |30 |
;                            -----                      ------------|   |
;                                                       |       |   |   |
;                                                       | 29    |44 |   |
;                                                       ----------------
;
;  *** indicates the mode keys, for which no scan code is sent
;-------------------------------------------------------------------------------
;
;      STANDARD U.S. LAYOUT AND SYMBOLIZATION:
;
;                 (left half)
;
;      ----------------    ----------------    ----------------
;      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
;      |F1 |F2 |F3 |F4 |   |F5 |F6 |F7 |F8 |   |F9 |F10|F11|F12|
;      ----------------    ----------------    ----------------
;
;      ---------------------------------------------------------
;      |   | ! | @ | # | $ | % | ^ | & | * | ( | ) | _ | + |BACK| ~ |
```

```
                        ;       |ESC| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | - | = |BSPCE| ` |
                        ;       -----------------------------------------------------------------
                        ;       |     |   |   |   |   |   |   |   |   |   |   |   |   | { | } |   | L |
                        ;       | TAB | Q | W | E | R | T | Y | U | I | O | P | [ | ] |   | F |
                        ;       -------------------------------------------------------+   |---|
                        ;       |   |CAP|   |   |   |   |   |   |   |   |   |   | : | " | RET |   |   |
                        ;       |CTL|LOK| A | S | D | F | G | H | J | K | L | ; | ' |     | \ |
                        ;       -----------------------------------------------------------------
                        ;       |   |     |   |   |   |   |   |   |   |   | ( | ) | ? |     |
                        ;       |ALT|SHIFT| Z | X | C | V | B | N | M | , | . | / |SHIFT|
                        ;       -------------------------------------------------------
                        ;                       |                               |
                        ;                       |       ( SPACE BAR )           |
                        ;                       ---------------------------------
                        ;
                        ;
                        ;                                                 ----------------
                        ;                                                 |   |   |BRK|   |
                        ;   (right half)                                  |INS|DEL|PAU|PRT|
                        ;                                                 ----------------
                        ;                                                 ----------------
                        ;                                                 |   |   |   |   |
                        ;                                                 | = | + |BPC|TAB|
                        ;                   -----                         ----------------
                        ;                   | ^ |                         |   |   |   |   |
                        ;                   | | |                         | 7 | 8 | 9 | - |
                        ;               ------------                        ----------------
                        ;               |   |   |   |                     |   |   |   |   |
                        ;               |<--|HOM|-->|                     | 4 | 5 | 6 | , |
                        ;               ------------                        ----------------
                        ;                   | | |                         |   |   |   | E |
                        ;                   | V |                         | 1 | 2 | 3 | N |
                        ;                   -----                         ----------------| T |
                        ;                                                 |       |   | E |
                        ;                                                 |   0   | . | R |
                        ;                                                 ----------------
                        ;
                        ;
                        ;-----------------------------------------------------------------------------
                        ;                               THE TABLES:
                        ;*****************************************************************************
                        ;
=0000                   351             SECT    ROMCOD
                        352     ;
                        353     ;   UNSHIFTed (normal) table - Note that the value 'xx' in keys defined as
                        354     ;       'xx+80H' is to be used as an index into KBFUNC for the actual value.
                        355     ;
0000'                   356     KBUNSH  LABEL   BYTE            ; Key #   Lengend:
0000'  A0               357             DB      32+80H          ; 01    F5 key
0001'  A1               358             DB      33+80H          ; 02    F6 key
0002'  A2               359             DB      34+80H          ; 03    F7 key
0003'  A3               360             DB      35+80H          ; 04    F8 key
```

```
0004' A4                361     DB      36+80H          ; 05    F9 key
0005' A5                362     DB      37+80H          ; 06    F10 key
0006' F3                363     DB      115+80H         ; 07    F11 key
0007' F4                364     DB      116+80H         ; 08    F12 key
0008' 31                365     DB      '1'             ; 09    1 / !
0009' 32                366     DB      '2'             ; 10    2 / @
000A' 33                367     DB      '3'             ; 11    3 / #
000B' 34                368     DB      '4'             ; 12    4 / $
000C' 35                369     DB      '5'             ; 13    5 / %
000D' 36                370     DB      '6'             ; 14    6 / ^
000E' 37                371     DB      '7'             ; 15    7 / &
000F' 38                372     DB      '8'             ; 16    8 / *
0010' 39                373     DB      '9'             ; 17    9 / (
0011' 30                374     DB      '0'             ; 18    0 / )
0012' 2D                375     DB      '-'             ; 19    - / _
0013' 3D                376     DB      '='             ; 20    = / +
0014' 08                377     DB      BS              ; 21    BACK SPACE key
0015' 60                378     DB      '`'             ; 22    ` / ~
0016' 3D                379     DB      '='             ; 23    NUM =  (numeric pad)
0017' 2B                380     DB      '+'             ; 24    NUM +
0018' 20                381     DB      SPACE           ; 25    NUM SPACE key
0019' 09                382     DB      HT              ; 26    NUM TAB key
001A' 31                383     DB      '1'             ; 27    NUM 1
001B' FF                384     DB      -1              ; 28    ----    (not implemented)
001C' 30                385     DB      '0'             ; 29    NUM 0
001D' 0D                386     DB      CR              ; 30    NUM ENTER key
001E' 34                387     DB      '4'             ; 31    NUM 4
001F' 35                388     DB      '5'             ; 32    NUM 5
0020' 39                389     DB      '9'             ; 33    NUM 9
0021' 2D                390     DB      '-'             ; 34    NUM -
0022' 32                391     DB      '2'             ; 35    NUM 2
0023' FF                392     DB      -1              ; 36    ----    (not implemented)
0024' FF                393     DB      -1              ; 37    ----    (not implemented)
0025' FF                394     DB      -1              ; 38    ----    (not implemented)
0026' 37                395     DB      '7'             ; 39    NUM 7
0027' 38                396     DB      '8'             ; 40    NUM 8
0028' 36                397     DB      '6'             ; 41    NUM 6
0029' 2C                398     DB      ','             ; 42    NUM ,
002A' 33                399     DB      '3'             ; 43    NUM 3
002B' 2E                400     DB      '.'             ; 44    NUM .
002C' CE                401     DB      78+80H          ; 45    PRINT key   (duplicate of 90)
002D' AA                402     DB      42+80H          ; 46    Cursor RIGHT key
002E' AE                403     DB      46+80H          ; 47    INSert key
002F' AF                404     DB      47+80H          ; 48    DELete key
0030' 09                405     DB      HT              ; 49    TAB key
0031' 71                406     DB      'q'             ; 50    Q
0032' 77                407     DB      'w'             ; 51    W
0033' 65                408     DB      'e'             ; 52    E
0034' 72                409     DB      'r'             ; 53    R
0035' 74                410     DB      't'             ; 54    T
0036' 79                411     DB      'y'             ; 55    Y
0037' 75                412     DB      'u'             ; 56    U
```

```
0038' 69                413         DB      'i'             ; 57    I
0039' 6F                414         DB      'o'             ; 58    O
003A' 70                415         DB      'p'             ; 59    P
003B' 5B                416         DB      '['             ; 60    [ / {
003C' 5D                417         DB      ']'             ; 61    ] / }
003D' 0A                418         DB      LF              ; 62    LINE FEED key
003E' FE                419         DB      PAUFNC          ; 63    BREAK/PAUSE key   (duplicate of 100)
003F' A7                420         DB      39+80H          ; 64    Cursor UP key
0040' 1B                421         DB      ESC_            ; 65    ESC key
0041' 61                422         DB      'a'             ; 66    A
0042' 73                423         DB      's'             ; 67    S
0043' 64                424         DB      'd'             ; 68    D
0044' 66                425         DB      'f'             ; 69    F
0045' 67                426         DB      'g'             ; 70    G
0046' 68                427         DB      'h'             ; 71    H
0047' 6A                428         DB      'j'             ; 72    J
0048' 6B                429         DB      'k'             ; 73    K
0049' 6C                430         DB      'l'             ; 74    L
004A' 3B                431         DB      ';'             ; 75    ; / :
004B' 27                432         DB      ''''            ; 76    ' / "
004C' 0D                433         DB      CR              ; 77    RETURN key
004D' 5C                434         DB      '\'             ; 78    \ / | key
004E' A9                435         DB      41+80H          ; 79    Cursor LEFT key
004F' A6                436         DB      38+80H          ; 80    HOME key
0050' 20                437         DB      SPACE           ; 81    SPACE bar
0051' 7A                438         DB      'z'             ; 82    Z
0052' 78                439         DB      'x'             ; 83    X
0053' 63                440         DB      'c'             ; 84    C
0054' 76                441         DB      'v'             ; 85    V
0055' 62                442         DB      'b'             ; 86    B
0056' 6E                443         DB      'n'             ; 87    N
0057' 6D                444         DB      'm'             ; 88    M
0058' 2C                445         DB      ','             ; 89    , / <
0059' CE                446         DB      78+80H          ; 90    PRINT key
005A' 2E                447         DB      '.'             ; 91    . / >
005B' 2F                448         DB      '/'             ; 92    / / ?
005C' FF                449         DB      -1              ; 93    ----      (not implemented)
005D' AF                450         DB      47+80H          ; 94    DELete key    (duplicate of 48)
005E' AE                451         DB      46+80H          ; 95    INSert key    (duplicate of 47)
005F' AC                452         DB      44+80H          ; 96    Cursor DOWN key
0060' FF                453         DB      -1              ; 97    ----      (not implemented)
0061' FF                454         DB      -1              ; 98    ----      (not implemented)
0062' FF                455         DB      -1              ; 99    ----      (not implemented)
0063' FE                456         DB      PAUFNC          ; 100   BREAK/PAUSE key
0064' 9C                457         DB      28+80H          ; 101   F1 key
0065' 9D                458         DB      29+80H          ; 102   F2 key
0066' 9E                459         DB      30+80H          ; 103   F3 key
0067' 9F                460         DB      31+80H          ; 104   F4 key
                        461 ;
                        462 ;   SHIFTed Mode table - Note that the value 'xx' in keys defined as
                        463 ;       'xx+80H' is to be used as an index into KBFUNC for the actual value.
                        464 ;
```

```
006B'                   465   KBSHFT  LABEL   BYTE            ; Key #    Legend
006B'  B4               466           DB      52+80H          ; 01    F5 key
0069'  B5               467           DB      53+80H          ; 02    F6 key
006A'  B6               468           DB      54+80H          ; 03    F7 key
006B'  B7               469           DB      55+80H          ; 04    F8 key
006C'  B8               470           DB      56+80H          ; 05    F9 key
006D'  B9               471           DB      57+80H          ; 06    F10 key
006E'  F5               472           DB      117+80H         ; 07    F11 key
006F'  F6               473           DB      118+80H         ; 08    F12 key
0070'  21               474           DB      '!'             ; 09    1 / !
0071'  40               475           DB      '@'             ; 10    2 / @
0072'  23               476           DB      '#'             ; 11    3 / #
0073'  24               477           DB      '$'             ; 12    4 / $
0074'  25               478           DB      '%'             ; 13    5 / %
0075'  5E               479           DB      '^'             ; 14    6 / ^
0076'  26               480           DB      '&'             ; 15    7 / &
0077'  2A               481           DB      '*'             ; 16    8 / *
0078'  28               482           DB      '('             ; 17    9 / (
0079'  29               483           DB      ')'             ; 18    0 / )
007A'  5F               484           DB      '_'             ; 19    - / _
007B'  2B               485           DB      '+'             ; 20    = / +
007C'  08               486           DB      BB              ; 21    BACK SPACE key
007D'  7E               487           DB      '~'             ; 22    ` / ~
007E'  3D               488           DB      '='             ; 23    NUM =  (numeric pad)
007F'  2B               489           DB      '+'             ; 24    NUM +
0080'  20               490           DB      SPACE           ; 25    NUM SPACE key
0081'  81               491           DB      1+80H           ; 26    NUM TAB key
0082'  31               492           DB      '1'             ; 27    NUM 1
0083'  FF               493           DB      -1              ; 28    ----    (not implemented)
0084'  30               494           DB      '0'             ; 29    NUM 0
0085'  0D               495           DB      CR              ; 30    NUM ENTER key
0086'  34               496           DB      '4'             ; 31    NUM 4
0087'  35               497           DB      '5'             ; 32    NUM 5
0088'  39               498           DB      '9'             ; 33    NUM 9
0089'  2D               499           DB      '-'             ; 34    NUM -
008A'  32               500           DB      '2'             ; 35    NUM 2
008B'  FF               501           DB      -1              ; 36    ----    (not implemented)
008C'  FF               502           DB      -1              ; 37    ----    (not implemented)
008D'  FF               503           DB      -1              ; 38    ----    (not implemented)
008E'  37               504           DB      '7'             ; 39    NUM 7
008F'  38               505           DB      '8'             ; 40    NUM 8
0090'  36               506           DB      '6'             ; 41    NUM 6
0091'  2C               507           DB      ','             ; 42    NUM ,
0092'  33               508           DB      '3'             ; 43    NUM 3
0093'  2E               509           DB      '.'             ; 44    NUM .
0094'  FC               510           DB      PSCFNC          ; 45    PRINT key   (duplicate of 90)
0095'  F1               511           DB      113+80H         ; 46    Cursor RIGHT key
0096'  E5               512           DB      101+80H         ; 47    INSert key
0097'  E8               513           DB      104+80H         ; 48    DELete key
0098'  81               514           DB      01+80H          ; 49    TAB key
0099'  51               515           DB      'Q'             ; 50    Q
009A'  57               516           DB      'W'             ; 51    W
```

```
009B' 45                517 DB      'E'             ; 52    E
009C' 52                518 DB      'R'             ; 53    R
009D' 54                519 DB      'T'             ; 54    T
009E' 59                520 DB      'Y'             ; 55    Y
009F' 55                521 DB      'U'             ; 56    U
00A0' 49                522 DB      'I'             ; 57    I
00A1' 4F                523 DB      'O'             ; 58    O
00A2' 50                524 DB      'P'             ; 59    P
00A3' 7B                525 DB      '{'             ; 60    [ / {
00A4' 7D                526 DB      '}'             ; 61    ] / }
00A5' 0A                527 DB      LF              ; 62    LINE FEED key
00A6' FD                528 DB      PBKFNC          ; 63    BREAK/PAUSE key  (duplicate of 100)
00A7' EF                529 DB      111+80H         ; 64    Cursor UP key
00A8' 1B                530 DB      ESC_            ; 65    ESC key
00A9' 41                531 DB      'A'             ; 66    A
00AA' 53                532 DB      'S'             ; 67    S
00AB' 44                533 DB      'D'             ; 68    D
00AC' 46                534 DB      'F'             ; 69    F
00AD' 47                535 DB      'G'             ; 70    G
00AE' 48                536 DB      'H'             ; 71    H
00AF' 4A                537 DB      'J'             ; 72    J
00B0' 4B                538 DB      'K'             ; 73    K
00B1' 4C                539 DB      'L'             ; 74    L
00B2' 3A                540 DB      ':'             ; 75    ; / :
00B3' 22                541 DB      '"'             ; 76    ' / "
00B4' 0D                542 DB      CR              ; 77    RETURN key
00B5' 7C                543 DB      '|'             ; 78    \ / |
00B6' F2                544 DB      114+80H         ; 79    Cursor LEFT key
00B7' EE                545 DB      110+80H         ; 80    HOME key
00B8' 20                546 DB      SPACE           ; 81    SPACE bar
00B9' 5A                547 DB      'Z'             ; 82    Z
00BA' 58                548 DB      'X'             ; 83    X
00BB' 43                549 DB      'C'             ; 84    C
00BC' 56                550 DB      'V'             ; 85    V
00BD' 42                551 DB      'B'             ; 86    B
00BE' 4E                552 DB      'N'             ; 87    N
00BF' 4D                553 DB      'M'             ; 88    M
00C0' 3C                554 DB      '<'             ; 89    , / <
00C1' FC                555 DB      PSCFNC          ; 90    PRINT key
00C2' 3E                556 DB      '>'             ; 91    . / >
00C3' 3F                557 DB      '?'             ; 92    / / ?
00C4' FF                558 DB      -1              ; 93    ----    (not implemented)
00C5' E8                559 DB      104+80H         ; 94    DELete key  (duplicate of 48)
00C6' E5                560 DB      101+80H         ; 95    INSert key  (duplicate of 47)
00C7' F0                561 DB      112+80H         ; 96    Cursor DOWN key
00C8' FF                562 DB      -1              ; 97    ----    (not implemented)
00C9' FF                563 DB      -1              ; 98    ----    (not implemented)
00CA' FF                564 DB      -1              ; 99    ----    (not implemented)
00CB' FD                565 DB      PBKFNC          ; 100   BREAK/PAUSE key
00CC' B0                566 DB      48+80H          ; 101   F1 key
00CD' B1                567 DB      49+80H          ; 102   F2 key
00CE' B2                568 DB      50+80H          ; 103   F3 key
```

```
00CF'  B3                  569             DB      51+80H          ; 104   F4 key
                           570     ;
                           571     ;   CONTROL Mode table - Note that the value 'xx' in keys defined as
                           572     ;       'xx+80H' is to be used as an index into KBFUNC for the actual value.
                           573     ;
00D0'                      574     KBCTRL  LABEL   BYTE            ; Key #    Legend
00D0'  BE                  575             DB      62+80H          ; 01    F5 key
00D1'  BF                  576             DB      63+80H          ; 02    F6 key
00D2'  C0                  577             DB      64+80H          ; 03    F7 key
00D3'  C1                  578             DB      65+80H          ; 04    F8 key
00D4'  C2                  579             DB      66+80H          ; 05    F9 key
00D5'  C3                  580             DB      67+80H          ; 06    F10 key
00D6'  F7                  581             DB      119+80H         ; 07    F11 key
00D7'  F8                  582             DB      120+80H         ; 08    F12 key
00D8'  FF                  583             DB      -1              ; 09    1 / !
00D9'  80                  584             DB      00+80H          ; 10    2 / @
00DA'  FF                  585             DB      -1              ; 11    3 / #
00DB'  FF                  586             DB      -1              ; 12    4 / $
00DC'  FF                  587             DB      -1              ; 13    5 / %
00DD'  1E                  588             DB      RS              ; 14    6 / ^
00DE'  FF                  589             DB      -1              ; 15    7 / &
00DF'  FF                  590             DB      -1              ; 16    8 / *
00E0'  FF                  591             DB      -1              ; 17    9 / (
00E1'  FF                  592             DB      -1              ; 18    0 / )
00E2'  1F                  593             DB      US              ; 19    - / _
00E3'  FF                  594             DB      -1              ; 20    = / +
00E4'  7F                  595             DB      DEL             ; 21    BACK SPACE key
00E5'  FF                  596             DB      -1              ; 22    ` / ~
00E6'  3D                  597             DB      '='             ; 23    NUM =  (numeric pad)
00E7'  2B                  598             DB      '+'             ; 24    NUM +
00E8'  20                  599             DB      SPACE           ; 25    NUM SPACE key
00E9'  09                  600             DB      HT              ; 26    NUM TAB key
00EA'  31                  601             DB      '1'             ; 27    NUM 1
00EB'  FF                  602             DB      -1              ; 28    ----    (not implemented)
00EC'  30                  603             DB      '0'             ; 29    NUM 0
00ED'  0D                  604             DB      CR              ; 30    NUM ENTER key
00EE'  34                  605             DB      '4'             ; 31    NUM 4
00EF'  35                  606             DB      '5'             ; 32    NUM 5
00F0'  39                  607             DB      '9'             ; 33    NUM 9
00F1'  2D                  608             DB      '-'             ; 34    NUM -
00F2'  32                  609             DB      '2'             ; 35    NUM 2
00F3'  FF                  610             DB      -1              ; 36    ----    (not implemented)
00F4'  FF                  611             DB      -1              ; 37    ----    (not implemented)
00F5'  FF                  612             DB      -1              ; 38    ----    (not implemented)
00F6'  37                  613             DB      '7'             ; 39    NUM 7
00F7'  38                  614             DB      '8'             ; 40    NUM 8
00F8'  36                  615             DB      '6'             ; 41    NUM 6
00F9'  2C                  616             DB      ','             ; 42    NUM ,
00FA'  33                  617             DB      '3'             ; 43    NUM 3
00FB'  2E                  618             DB      '.'             ; 44    NUM .
00FC'  FF                  619             DB      -1              ; 45    PRINT key   (duplicate of 90)
00FD'  D0                  620             DB      80+80H          ; 46    Cursor RIGHT key
```

```
00FE'  E6                621         DB      102+80H         ; 47    INSert key
00FF'  E9                622         DB      105+80H         ; 48    DELete key
0100'  09                623         DB      HT              ; 49    TAB key
0101'  11                624         DB      DC1             ; 50    Q
0102'  17                625         DB      ETB             ; 51    W
0103'  05                626         DB      ENQ             ; 52    E
0104'  12                627         DB      DC2             ; 53    R
0105'  14                628         DB      DC4             ; 54    T
0106'  19                629         DB      EM              ; 55    Y
0107'  15                630         DB      NAK             ; 56    U
0108'  09                631         DB      HT              ; 57    I
0109'  0F                632         DB      SI_             ; 58    O
010A'  10                633         DB      DLE             ; 59    P
010B'  1B                634         DB      ESC_            ; 60    [ / {
010C'  1D                635         DB      GS              ; 61    ] / }
010D'  D1                636         DB      81+80H          ; 62    LINE FEED key
010E'  FF                637         DB      -1              ; 63    BREAK/PAUSE key   (duplicate of 100)
010F'  E0                638         DB      96+80H          ; 64    Cursor UP key
0110'  1B                639         DB      ESC_            ; 65    ESC key
0111'  01                640         DB      SOH             ; 66    A
0112'  13                641         DB      DC3             ; 67    S
0113'  04                642         DB      EOT             ; 68    D
0114'  06                643         DB      ACK             ; 69    F
0115'  07                644         DB      BEL             ; 70    G
0116'  08                645         DB      BS              ; 71    H
0117'  0A                646         DB      LF              ; 72    J
0118'  0B                647         DB      VT              ; 73    K
0119'  0C                648         DB      FF              ; 74    L
011A'  FF                649         DB      -1              ; 75    ; / :
011B'  FF                650         DB      -1              ; 76    ' / "
011C'  0D                651         DB      CR              ; 77    RETURN key
011D'  1C                652         DB      FS              ; 78    \ / |
011E'  CF                653         DB      79+80H          ; 79    Cursor LEFT key
011F'  D3                654         DB      83+80H          ; 80    HOME key
0120'  20                655         DB      SPACE           ; 81    SPACE bar
0121'  1A                656         DB      SUB_            ; 82    Z
0122'  18                657         DB      CAN             ; 83    X
0123'  03                658         DB      ETX             ; 84    C
0124'  16                659         DB      SYN             ; 85    V
0125'  02                660         DB      STX             ; 86    B
0126'  0E                661         DB      SO_             ; 87    N
0127'  0D                662         DB      CR              ; 88    M
0128'  FF                663         DB      -1              ; 89    , / <
0129'  FF                664         DB      -1              ; 90    PRINT key
012A'  FF                665         DB      -1              ; 91    . / >
012B'  FF                666         DB      -1              ; 92    / / ?
012C'  FF                667         DB      -1              ; 93    ----    (not implemented)
012D'  E9                668         DB      105+80H         ; 94    DELete key   (duplicate of 48)
012E'  E6                669         DB      102+80H         ; 95    INSert key   (duplicate of 47)
012F'  D2                670         DB      82+80H          ; 96    Cursor DOWN key
0130'  FF                671         DB      -1              ; 97    ----    (not implemented)
0131'  FF                672         DB      -1              ; 98    ----    (not implemented)
```

```
0132'  FF                673           DB      -1              ; 99    ----    (not implemented)
0133'  FF                674           DB      -1              ; 100   BREAK/PAUSE key
0134'  BA                675           DB      58+80H          ; 101   F1 key
0135'  BB                676           DB      59+80H          ; 102   F2 key
0136'  BC                677           DB      60+80H          ; 103   F3 key
0137'  BD                678           DB      61+80H          ; 104   F4 key
                         679   ;-------------------------------------------------------------------------
                         680   ;
                         681   ;   ALTERNATE Mode table - Note that the value 'xx' in keys defined as
                         682   ;       'xx+80H' is to be used as an index into KBFUNC for the actual value.
                         683   ;
0138'                    684   KB_ALT  LABEL   BYTE            ; Key #    Legend
0138'  C8                685           DB      72+80H          ; 01    F5 key
0139'  C9                686           DB      73+80H          ; 02    F6 key
013A'  CA                687           DB      74+80H          ; 03    F7 key
013B'  CB                688           DB      75+80H          ; 04    F8 key
013C'  CC                689           DB      76+80H          ; 05    F9 key
013D'  CD                690           DB      77+80H          ; 06    F10 key
013E'  F9                691           DB      121+80H         ; 07    F11 key
013F'  FA                692           DB      122+80H         ; 08    F12 key
0140'  D4                693           DB      84+80H          ; 09    1 / !
0141'  D5                694           DB      85+80H          ; 10    2 / @
0142'  D6                695           DB      86+80H          ; 11    3 / #
0143'  D7                696           DB      87+80H          ; 12    4 / $
0144'  D8                697           DB      88+80H          ; 13    5 / %
0145'  D9                698           DB      89+80H          ; 14    6 / ^
0146'  DA                699           DB      90+80H          ; 15    7 / &
0147'  DB                700           DB      91+80H          ; 16    8 / *
0148'  DC                701           DB      92+80H          ; 17    9 / (
0149'  DD                702           DB      93+80H          ; 18    0 / )
014A'  DE                703           DB      94+80H          ; 19    - / _
014B'  DF                704           DB      95+80H          ; 20    = / +
014C'  FF                705           DB      -1              ; 21    BACK SPACE key
014D'  FF                706           DB      -1              ; 22    ` / ~
014E'  E1                707           DB      97+80H          ; 23    NUM =  (numeric pad)
014F'  E2                708           DB      98+80H          ; 24    NUM +
0150'  E3                709           DB      99+80H          ; 25    NUM SPACE key
0151'  E4                710           DB      100+80H         ; 26    NUM TAB key
0152'  FF                711           DB      -1              ; 27    NUM 1
0153'  FF                712           DB      -1              ; 28    ----    (not implemented)
0154'  FF                713           DB      -1              ; 29    NUM 0
0155'  FF                714           DB      -1              ; 30    NUM ENTER key
0156'  FF                715           DB      -1              ; 31    NUM 4
0157'  FF                716           DB      -1              ; 32    NUM 5
0158'  FF                717           DB      -1              ; 33    NUM 9
0159'  FF                718           DB      -1              ; 34    NUM -
015A'  FF                719           DB      -1              ; 35    NUM 2
015B'  FF                720           DB      -1              ; 36    ----    (not implemented)
015C'  FF                721           DB      -1              ; 37    ----    (not implemented)
015D'  FF                722           DB      -1              ; 38    ----    (not implemented)
015E'  FF                723           DB      -1              ; 39    NUM 7
015F'  FF                724           DB      -1              ; 40    NUM 8
```

```
0160'  FF                725          DB      -1              ; 41    NUM 6
0161'  FF                726          DB      -1              ; 42    NUM ,
0162'  FF                727          DB      -1              ; 43    NUM 3
0163'  FF                728          DB      -1              ; 44    NUM .
0164'  FF                729          DB      -1              ; 45    PRINT key   (duplicate of 90)
0165'  EC                730          DB      108+80H         ; 46    Cursor RIGHT key
0166'  E7                731          DB      103+80H         ; 47    INSert key
0167'  EA                732          DB      106+80H         ; 48    DELete key
0168'  FF                733          DB      -1              ; 49    TAB key
0169'  82                734          DB      02+80H          ; 50    Q
016A'  83                735          DB      03+80H          ; 51    W
016B'  84                736          DB      04+80H          ; 52    E
016C'  85                737          DB      05+80H          ; 53    R
016D'  86                738          DB      06+80H          ; 54    T
016E'  87                739          DB      07+80H          ; 55    Y
016F'  88                740          DB      08+80H          ; 56    U
0170'  89                741          DB      09+80H          ; 57    I
0171'  8A                742          DB      10+80H          ; 58    O
0172'  8B                743          DB      11+80H          ; 59    P
0173'  FF                744          DB      -1              ; 60    [ / {
0174'  FF                745          DB      -1              ; 61    ] / }
0175'  AB                746          DB      43+80H          ; 62    LINE FEED key
0176'  FF                747          DB      -1              ; 63    BREAK/PAUSE key   (duplicate of 100)
0177'  A8                748          DB      40+80H          ; 64    Cursor UP key
0178'  FF                749          DB      -1              ; 65    ESC key
0179'  8C                750          DB      12+80H          ; 66    A
017A'  8D                751          DB      13+80H          ; 67    S
017B'  8E                752          DB      14+80H          ; 68    D
017C'  8F                753          DB      15+80H          ; 69    F
017D'  90                754          DB      16+80H          ; 70    G
017E'  91                755          DB      17+80H          ; 71    H
017F'  92                756          DB      18+80H          ; 72    J
0180'  93                757          DB      19+80H          ; 73    K
0181'  94                758          DB      20+80H          ; 74    L
0182'  FF                759          DB      -1              ; 75    ; / :
0183'  FF                760          DB      -1              ; 76    ' / "
0184'  FF                761          DB      -1              ; 77    RETURN key
0185'  FF                762          DB      -1              ; 78    \ / |
0186'  EB                763          DB      107+80H         ; 79    Cursor LEFT key
0187'  ED                764          DB      109+80H         ; 80    HOME key
0188'  20                765          DB      SPACE           ; 81    SPACE bar
0189'  95                766          DB      21+80H          ; 82    Z
018A'  96                767          DB      22+80H          ; 83    X
018B'  97                768          DB      23+80H          ; 84    C
018C'  98                769          DB      24+80H          ; 85    V
018D'  99                770          DB      25+80H          ; 86    B
018E'  9A                771          DB      26+80H          ; 87    N
018F'  9B                772          DB      27+80H          ; 88    M
0190'  FF                773          DB      -1              ; 89    , / (
0191'  FF                774          DB      -1              ; 90    PRINT key
0192'  FF                775          DB      -1              ; 91    . / )
0193'  FF                776          DB      -1              ; 92    / / ?
```

```
0194' FF          777           DB      -1              ; 93    ----    (not implemented)
0195' EA          778           DB      106+80H         ; 94    DELete key   (duplicate of 48)
0196' E7          779           DB      103+80H         ; 95    INSert key   (duplicate of 47)
0197' AD          780           DB      45+80H          ; 96    Cursor DOWN key
0198' FF          781           DB      -1              ; 97    ----    (not implemented)
0199' FF          782           DB      -1              ; 98    ----    (not implemented)
019A' FF          783           DB      -1              ; 99    ----    (not implemented)
019B' FF          784           DB      -1              ; 100   BREAK/PAUSE key
019C' C4          785           DB      68+80H          ; 101   F1 key
019D' C5          786           DB      69+80H          ; 102   F2 key
019E' C6          787           DB      70+80H          ; 103   F3 key
019F' C7          788           DB      71+80H          ; 104   F4 key
                  789   ;-----------------------------------------------------------------------
                  790   ;
                  791   ;     FUNCTION key lookup table - If the value in the normal table is
                  792   ;        greater than 7FH but less than OFCH, the lower 7 bits of the
                  793   ;        byte is used as an index into this table to find the actual
                  794   ;        function value.
                  795   ;
01A0'             796   KBFUNC  LABEL   BYTE            ; TABLE OFFSET:
01A0' 03          797           DB      KB_NUL          ; 0
01A1' 0F          798           DB      KBBKTB          ; 1
01A2' 10          799           DB      KBALTQ          ; 2
01A3' 11          800           DB      KBALTW          ; 3
01A4' 12          801           DB      KBALTE          ; 4
01A5' 13          802           DB      KBALTR          ; 5
01A6' 14          803           DB      KBALTT          ; 6
01A7' 15          804           DB      KBALTY          ; 7
01A8' 16          805           DB      KBALTU          ; 8
01A9' 17          806           DB      KBALTI          ; 9
01AA' 18          807           DB      KBALTO          ; 10
01AB' 19          808           DB      KBALTP          ; 11
01AC' 1E          809           DB      KBALTA          ; 12
01AD' 1F          810           DB      KBALTB          ; 13
01AE' 20          811           DB      KBALTD          ; 14
01AF' 21          812           DB      KBALTF          ; 15
01B0' 22          813           DB      KBALTG          ; 16
01B1' 23          814           DB      KBALTH          ; 17
01B2' 24          815           DB      KBALTJ          ; 18
01B3' 25          816           DB      KBALTK          ; 19
01B4' 26          817           DB      KBALTL          ; 20
01B5' 2C          818           DB      KBALTZ          ; 21
01B6' 2D          819           DB      KBALTX          ; 22
01B7' 2E          820           DB      KBALTC          ; 23
01B8' 2F          821           DB      KBALTV          ; 24
01B9' 30          822           DB      KBALTB          ; 25
01BA' 31          823           DB      KBALTN          ; 26
01BB' 32          824           DB      KBALTM          ; 27
01BC' 3B          825           DB      KB_F1           ; 28
01BD' 3C          826           DB      KB_F2           ; 29
01BE' 3D          827           DB      KB_F3           ; 30
01BF' 3E          828           DB      KB_F4           ; 31
```

```
01C0'  3F                829  DB      KB_F5           ; 32
01C1'  40                830  DB      KB_F6           ; 33
01C2'  41                831  DB      KB_F7           ; 34
01C3'  42                832  DB      KB_F8           ; 35
01C4'  43                833  DB      KB_F9           ; 36
01C5'  44                834  DB      KB_F10          ; 37
01C6'  47                835  DB      KB_HOM          ; 38
01C7'  48                836  DB      KB_CUP          ; 39
01C8'  49                837  DB      KBACUP          ; 40
01C9'  4B                838  DB      KB_CLF          ; 41
01CA'  4D                839  DB      KB_CRT          ; 42
01CB'  4F                840  DB      KBALF           ; 43
01CC'  50                841  DB      KB_CDN          ; 44
01CD'  51                842  DB      KBACDN          ; 45
01CE'  52                843  DB      KB_INS          ; 46
01CF'  53                844  DB      KB_DEL          ; 47
01D0'  54                845  DB      KBSF1           ; 48
01D1'  55                846  DB      KBSF2           ; 49
01D2'  56                847  DB      KBSF3           ; 50
01D3'  57                848  DB      KBSF4           ; 51
01D4'  58                849  DB      KBSF5           ; 52
01D5'  59                850  DB      KBSF6           ; 53
01D6'  5A                851  DB      KBSF7           ; 54
01D7'  5B                852  DB      KBSF8           ; 55
01D8'  5C                853  DB      KBSF9           ; 56
01D9'  5D                854  DB      KBSF10          ; 57
01DA'  5E                855  DB      KBCF1           ; 58
01DB'  5F                856  DB      KBCF2           ; 59
01DC'  60                857  DB      KBCF3           ; 60
01DD'  61                858  DB      KBCF4           ; 61
01DE'  62                859  DB      KBCF5           ; 62
01DF'  63                860  DB      KBCF6           ; 63
01E0'  64                861  DB      KBCF7           ; 64
01E1'  65                862  DB      KBCF8           ; 65
01E2'  66                863  DB      KBCF9           ; 66
01E3'  67                864  DB      KBCF10          ; 67
01E4'  68                865  DB      KBAF1           ; 68
01E5'  69                866  DB      KBAF2           ; 69
01E6'  6A                867  DB      KBAF3           ; 70
01E7'  6B                868  DB      KBAF4           ; 71
01E8'  6C                869  DB      KBAF5           ; 72
01E9'  6D                870  DB      KBAF6           ; 73
01EA'  6E                871  DB      KBAF7           ; 74
01EB'  6F                872  DB      KBAF8           ; 75
01EC'  70                873  DB      KBAF9           ; 76
01ED'  71                874  DB      KBAF10          ; 77
01EE'  72                875  DB      KBPTQL          ; 78
01EF'  73                876  DB      KBCCLF          ; 79
01F0'  74                877  DB      KBCCRT          ; 80
01F1'  75                878  DB      KBCLF           ; 81
01F2'  76                879  DB      KBCCDN          ; 82
01F3'  77                880  DB      KBCHOM          ; 83
```

```
01F4'  78                881             DB      KBALT1          ; 84
01F5'  79                882             DB      KBALT2          ; 85
01F6'  7A                883             DB      KBALT3          ; 86
01F7'  7B                884             DB      KBALT4          ; 87
01F8'  7C                885             DB      KBALT5          ; 88
01F9'  7D                886             DB      KBALT6          ; 89
01FA'  7E                887             DB      KBALT7          ; 90
01FB'  7F                888             DB      KBALT8          ; 91
01FC'  80                889             DB      KBALT9          ; 92
01FD'  81                890             DB      KBALT0          ; 93
01FE'  82                891             DB      KBADSH          ; 94
01FF'  83                892             DB      KBAEQU          ; 95
0200'  84                893             DB      KBCCUP          ; 96
0201'  8C                894             DB      KB_PF1          ; 97
0202'  8D                895             DB      KB_PF2          ; 98
0203'  8E                896             DB      KB_PF3          ; 99
0204'  8F                897             DB      KB_PF4          ; 100
0205'  28                898             DB      KBSINB          ; 101
0206'  29                899             DB      KBCINB          ; 102
0207'  2A                900             DB      KBAINS          ; 103
0208'  38                901             DB      KBSDEL          ; 104
0209'  39                902             DB      KBCDEL          ; 105
020A'  3A                903             DB      KBADEL          ; 106
020B'  4C                904             DB      KBACLF          ; 107
020C'  4E                905             DB      KBACRT          ; 108
020D'  85                906             DB      KBAHOM          ; 109
020E'  86                907             DB      KBSHOM          ; 110
020F'  88                908             DB      KBSCUP          ; 111
0210'  89                909             DB      KBSCDN          ; 112
0211'  8A                910             DB      KBSCRT          ; 113
0212'  8B                911             DB      KBSCLF          ; 114
0213'  45                912             DB      KB_F11          ; 115
0214'  46                913             DB      KB_F12          ; 116
0215'  08                914             DB      KBSF11          ; 117
0216'  09                915             DB      KBSF12          ; 118
0217'  0A                916             DB      KBCF11          ; 119
0218'  0B                917             DB      KBCF12          ; 120
0219'  0C                918             DB      KBAF11          ; 121
021A'  0D                919             DB      KBAF12          ; 122
                         920     ;
                         921     ;-----------------------------------------------------------------------
                         922     ;
                         923     ;   Table of the NUMeric keys in ascending order according to their legends.
                         924     ;       This table is used by the keyboard DSR to implement the special ALT
                         925     ;       NUM-pad function of allowing a 3-digit decimal value to be typed on
                         926     ;       the numeric pad while the ALT key is held down.  The resulting value
                         927     ;       passed directly to the DSR caller.
                         928     ;
021B'                    929     KBNUMK  LABEL   BYTE
021B'  1D                930             DB      29              ; Scan code for NUMeric 0 key
021C'  1B                931             DB      27              ; Scan code for NUMeric 1 key
021D'  23                932             DB      35              ; Scan code for NUMeric 2 key
```

```
021E'  28                933             DB      43              ; Scan code for NUMeric 3 key
021F'  1F                934             DB      31              ; Scan code for NUMeric 4 key
0220'  20                935             DB      32              ; Scan code for NUMeric 5 key
0221'  29                936             DB      41              ; Scan code for NUMeric 6 key
0222'  27                937             DB      39              ; Scan code for NUMeric 7 key
0223'  28                938             DB      40              ; Scan code for NUMeric 8 key
0224'  21                939             DB      33              ; Scan code for NUMeric 9 key
                         940     ;
                         941             END

No errors detected
```

```
                        1   ;*****************************************************************************
                        2   ;  TITLE    - KEYDSR - Keyboard device service routine
                        3   ;  COMPUTER - 8088 Assembly Language
                        4   ;
                        5   ;  ABSTRACT - This module contains the keyboard driver logic for the
                        6   ;       Pegasus system ROM.
                        7   ;
                        8   ;*****************************************************************************
                        9           NAME    KEYDSR - Keyboard device service routine        -
                       11   ;*****************************************************************************
                       12   ;                           PUBLIC DEFINITIONS
                       13   ;*****************************************************************************
                       14   ;
                       15           PUBLIC  FLUSH
                       16           PUBLIC  KEY_IO
                       17           PUBLIC  KEYIN
                       18           PUBLIC  KEYINI
                       19           PUBLIC  KEYISR
                       20           PUBLIC  KEYOUT
                       21           PUBLIC  KEYRST
                       22           PUBLIC  KPAUSE
                       23           PUBLIC  STATUS
                       24   ;
                       25   ;*****************************************************************************
                       26   ;                           EXTERNAL REFERENCES
                       27   ;*****************************************************************************
                       28   ;
=0000                  29           SECT    ROMCOD
                       30           EXTRN   DELAY:NEAR      ; 1 millisecond delay routine           (ROMUTL)
                       31           EXTRN   PWRUP:FAR       ; Keyboard reset causes system restart  (RESET)
                       32           EXTRN   SBEEP:NEAR      ; System beep routine                   (BELDSR)
                       33   ;
=0000                  34           SECT    ROMCOD
                       35           EXTRN   KB_ALT:BYTE     ; Keyboard Alternate-Mode lookup table  (KBTABL)
                       36           EXTRN   KBCTRL:BYTE     ; Keyboard Control-Mode lookup table    (KBTABL)
                       37           EXTRN   KBFUNC:BYTE     ; Keyboard Function key lookup table    (KBTABL)
                       38           EXTRN   KBNUMK:BYTE     ; Keyboard Numeric pad keys table       (KBTABL)
                       39           EXTRN   KBSHFT:BYTE     ; Keyboard Shifted-Mode lookup table    (KBTABL)
                       40           EXTRN   KBUNSH:BYTE     ; Keyboard Unshifted-Mode lookup table  (KBTABL)
                       41   ;
=0000                  42           SECT    ROMDAT          ; From the ROM's data area ...
                       43           EXTRN   CRTHLD:BYTE     ; CRT screen hold flag (set by KPAUSE) (ROMDAT)
                       44           EXTRN   ENDQ:WORD       ; End of the keyboard type-ahead buffer(ROMDAT)
                       45           EXTRN   KBMODE:BYTE     ; Status byte for Keyboard driver       (ROMDAT)
                       46           EXTRN   KBSPSV:WORD     ; Keyboard interrupt routine SP save    (ROMDAT)
                       47           EXTRN   KBSSSV:WORD     ; Keyboard interrupt routine SS save    (ROMDAT)
                       48           EXTRN   KEYISP:WORD     ; Keyboard interrupt routine stack      (ROMDAT)
                       49           EXTRN   NUMCNT:BYTE     ; Count of ALT/NUM keystrokes           (ROMDAT)
                       50           EXTRN   NUMVAL:BYTE     ; Accumulator for ALT/NUM keystrokes    (ROMDAT)
                       51           EXTRN   QDEPTH:BYTE     ; Current depth of queue (# chars)      (ROMDAT)
                       52           EXTRN   QFRONT:WORD     ; Pointer to 'front' of keyboard buffer(ROMDAT)
                       53           EXTRN   QREAR:WORD      ; Pointer to 'rear' of keyboard buffer  (ROMDAT)
```

```
                        54          EXTRN   QUEUE:WORD      ; Beginning of type-ahead buffer       (ROMDAT)
                        55  ;*****************************************************************************
                        56  ;                           LOCAL CONSTANTS
                        57  ;*****************************************************************************
                        58  ;       INCLUDE PEQ:PORTADDR.EQU
                        59  ;       INCLUDE PEQ:USART.EQU
                        60  ;       INCLUDE PEQ:INTCTLR.EQU
                        61  ;       INCLUDE PEQ:KBSPCL.EQU
                        62  ;       INCLUDE PEQ:KBCMD.EQU
                        63  ;       INCLUDE PEQ:KBMISC.EQU
                        64  ;       INCLUDE PEQ:VECTOR.EQU
                        65  ;
                       406  ;*****************************************************************************
                       407  ;                           CODE SEGMENT DEFINITION
                       408  ;*****************************************************************************
                       409  ;
=0000                  410          SECT    ROMCOD
                       411          ASSUME  CS:ROMCOD
                       412  ;
                       413  ;*****************************************************************************
                       414  ;                           MODULE ENTRY POINT
                       415  ;*****************************************************************************
                       416  ;
                       417  ;   KEYBOARD 'DSR'  - KEYIN, STATUS, FLUSH - INT 4AH
                       418  ;
                       419  ;   INPUT:  AH = Function (see individual routines for more detail):
                       420  ;                   0 - Read character from keyboard, return in AX
                       421  ;                   1 - Get status of keyboard:
                       422  ;                           ZF = 1 if no char available
                       423  ;                           ZF = 0 if character available & char returned in AX
                       424  ;                                (character remains in buffer)
                       425  ;                   2 - Read keyboard shift state - returns last valid
                       426  ;                       shift mode in register AL:
                       427  ;                           bit 0 = Control key
                       428  ;                           bit 1 = Alternate key
                       429  ;                           bit 2 = Shift key/s
                       430  ;                           bit 7 = Uppercase mode
                       431  ;                   3 - Flush keyboard buffer
                       432  ;                   4 - Send command character in AL to the keyboard
                       433  ;                           ZF = 1 if command accepted OK
                       434  ;                           ZF = 0 if error
                       435  ;                   5 - Insert 16-bit character in BX into the keyboard buffer
                       436  ;                        (simulate a key being typed)
                       437  ;                           ZF = 0 if key was placed into the buffer
                       438  ;                           ZF = 1 if buffer was full (key not placed)
                       439  ;                                  (BX preserved for retry, but not AH)
                       440  ;                   )5 - NULL command
                       441  ;   OUTPUT: as noted above
                       442  ;   USED:   AX (BX in #5)
                       443  ;   STACK:
                       444      ASSUME  CS:ROMCOD, DS:NOTHING
                       445  ;
```

```
                        446   ;-----------------------------------------------------------------------------
0000'                   447   KEY_IO  PROC    FAR
0000'  FB               448           STI                     ;;; interrupts back on
0001'  1E               449           PUSH    DS              ;;; save user's current DS
0002'  2E8E 1E C180     450           MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; set up my DS
0007'  FC               451           CLD                     ; forward strings
0008'  E8 0004          452           CALL    DO_KEY
000B'  1F               453           POP     DS
000C'  CA 0002          454           RET     2               ; *** RETURN *** (throw away saved flags)
                        455   ;
000F'                   456   DO_KEY  PROC    NEAR
000F'  08 E4            457           OR      AH,AH           ; AH = 0
0011'  74 15            458           JZ      KEYIN
0013'  FE CC            459           DEC     AH              ; AH = 1
0015'  74 2A            460           JZ      STATUS
0017'  FE CC            461           DEC     AH              ; AH = 2
0019'  74 47            462           JZ      SHIFTS
001B'  FE CC            463           DEC     AH              ; AH = 3
001D'  74 47            464           JZ      FLUSH
001F'  FE CC            465           DEC     AH              ; AH = 4
0021'  74 60            466           JZ      KEYOUT
0023'  FE CC            467           DEC     AH              ; AH = 5
0025'  74 54            468           JZ      KEYINQ
0027'  C3               469           RET                     ; INVALID
                        470   ;
                        471   ;*****************************************************************************
                        472   ;   KEYBOARD INPUT ROUTINE
                        473   ;
                        474   ;   INPUT:   (none)
                        475   ;   OUTPUT: AX = character (waits until one is available)
                        476   ;                (ASCII values return in AL with AH=0,
                        477   ;                 extended functions return with AL=0 and code in AH.)
                        478   ;   USED:    AX
                        479   ;   STACK:
                        480       ASSUME  CS:ROMCOD, DS:ROMDAT
                        481   ;-----------------------------------------------------------------------------
0028'                   482   KEYIN   PROC    NEAR
0028'  E8 0016          483           CALL    STATUS
002B'  74 FB            484           JZ      KEYIN           ; Loop til char ready
002D'  FA               485           CLI                     ;;; lights out
002E'  56               486           PUSH    SI              ;;;
002F'  8B 36 0013"      487           MOV     SI,QFRONT       ;;; Point to current character
0033'  E8 0020          488           CALL    INCQ            ;;; Permanently remove from queue
0036'  89 36 0013"      489           MOV     QFRONT,SI       ;;;    by updating front pointer
003A'  FE 0E 0012"      490           DEC     QDEPTH          ;;; And decrease Char-Waiting count
003E'  5E               491           POP     SI              ;;;
003F'  FB               492           STI                     ;;;
0040'  C3               493           RET                     ;;; *** RETURN ***
                        494   ;
                        495   ;*****************************************************************************
                        496   ;   KEYBOARD STATUS ROUTINE
                        497   ;
```

```
                        498   ;   INPUT:  (none)
                        499   ;   OUTPUT: ZF = reset (0) if ready, set (1) if not ready
                        500   ;           AX = char (remains in buffer)  (undefined if KB not ready)
                        501   ;   USED:   AX
                        502   ;   STACK:
                        503       ASSUME  CS:ROMCOD, DS:ROMDAT
                        504   ;-----------------------------------------------------------------------------
0041'                   505   STATUS  PROC    NEAR
0041'  FA               506           CLI                     ;;; raise the shields
0042'  56               507           PUSH    SI
0043'  8B 36 0013"      508           MOV     SI,QFRONT
0047'  3B 36 0014"      509           CMP     SI,QREAR        ;;; Q: Anything in queue ?
004B'  74 06            510           JZ      ST1             ;;;   N: return not-ready status
004D'  E8 0006          511           CALL    INCQ
0050'  AD               512           LODS    QUEUE           ;;;   Y: get the char
0051'  09 F6            513           OR      SI,SI           ;;; Reset the zero flag
0053'                   514   ST1:
0053'  5E               515           POP     SI
0054'  FB               516           STI                     ;;;
0055'  C3               517           RET                     ;;; *** RETURN ***
                        518   ;*****************************************************************************
                        519   ;   INCQ - increment queue pointer (assumes interrupts are disabled)
0056'                   520   INCQ:                           ;;; increment queue pointer
0056'  46               521           INC     SI
0057'  46               522           INC     SI
0058'  81 FE 000B"      523           CMP     SI,OFFSET ENDQ  ;;; Q: exceeded length of queue ?
005C'  72 03            524           JB      IQ1             ;;;   N: quit
005E'  BE 0015"         525           MOV     SI,OFFSET QUEUE ;;;   Y: wrap around to front
0061'                   526   IQ1:
0061'  C3               527           RET                     ;;; *** RETURN ***
                        528   ;
                        529   ;*****************************************************************************
                        530   ;   READ KEYBOARD SHIFT STATUS
                        531   ;
                        532   ;   INPUT:  (none)
                        533   ;   OUTPUT: AL = copy of keyboard mode byte
                        534   ;   USED:   AL
                        535   ;   STACK:
                        536       ASSUME  CS:ROMCOD, DS:ROMDAT
                        537   ;-----------------------------------------------------------------------------
0062'                   538   SHIFTS  PROC    NEAR
0062'  A0 000C"         539           MOV     AL,KBMODE               ; Pick up mode byte
0065'  C3               540           RET                             ; *** RETURN ***
                        541   ;
                        542   ;
                        543   ;*****************************************************************************
                        544   ;   FLUSH THE KEYBOARD BUFFER
                        545   ;
                        546   ;   INPUT:  (none)
                        547   ;   OUTPUT: (keyboard queue empty)
                        548   ;   USED:   (none)
                        549   ;   STACK:  4 bytes
```

```
                                550         ASSUME  CS:ROMCOD, DS:ROMDAT
                                551     ;-----------------------------------------------------------------------
0066'                           552     FLUSH   PROC    NEAR
0066'  9C                       553             PUSHF                           ; Save previous interrupt status
0067'  FA                       554             CLI                             ; Protect this
0068'  C7 06 0014" 0015"        555             MOV     QREAR,OFFSET QUEUE      ;;; Front = rear
006E'  C7 06 0013" 0015"        556             MOV     QFRONT,OFFSET QUEUE     ;;;
0074'  C6 06 0012" 00           557             MOV     BYTE PTR QDEPTH,00      ;;; Clear the Char-waiting counter
0079'  9D                       558             POPF                            ;;; Restore previous interrupt status
007A'  C3                       559             RET                             ;;; *** RETURN ***
                                560     ;***********************************************************************
                                561     ;   KEYINQ - Put the character in BX into the keyboard queue
                                562     ;
                                563     ;   INPUT:  BX = 16-bit character
                                564     ;   OUTPUT: ZF = 0 if character was placed into the buffer
                                565     ;           ZF = 1 if buffer was full (character not placed, in this case)
                                566     ;   USED:   AX
                                567     ;   STACK:
                                568         ASSUME  CS:ROMCOD, DS:ROMDAT
                                569     ;-----------------------------------------------------------------------
007B'                           570     KEYINQ  PROC    NEAR
007B'  56                       571             PUSH    SI
007C'  8B C3                    572             MOV     AX,BX
007E'  E8 023C                  573             CALL    KBFQ
0081'  5E                       574             POP     SI
0082'  C3                       575             RET
                                576     ;***********************************************************************
                                577     ;
                                578     ;   KEYOUT - Keyboard output routine - used to send commands to the keyboard.
                                579     ;       To use, place the command in register AL.  The command is not range-
                                580     ;       checked, so be sure to use a valid command.  Upon return, check the
                                581     ;       Z-flag.  If it is set (ZF = 1), then everything happened OK, and the
                                582     ;       keyboard has accepted and acted upon the command.  If the Z-flag is
                                583     ;       reset (ZF = 0), then some error has occurred.  To see what caused
                                584     ;       the error, look at AL.  If AL is 0FFH, then the routine timed-out
                                585     ;       while looking for the acknowledge char from the keyboard.  If AL is
                                586     ;       00H, then there was an error on the received character.  Otherwise,
                                587     ;       AL has the character returned by the keyboard.
                                588     ;
                                589     ;   INPUT:  AL = code to be sent to keyboard (see PEQ:KBCMD.EQU)
                                590     ;                There is no range-checking on the command
                                591     ;   OUTPUT: AL = 0FEH --) error in received character
                                592     ;              = 0FFH --) keyboard did not respond (time out)
                                593     ;                   else --) other error (ROM, RAM, etc.)
                                594     ;           ZF = 1 if OK (AL will be 70H in this case)
                                595     ;           ZF = 0 if error (then other reply codes apply)
                                596     ;
                                597     ;   USED:   AX
                                598     ;   STACK:  8 Bytes (including call)
                                599         ASSUME  DS:ROMDAT
                                600     ;
                                601     ;-----------------------------------------------------------------------
```

```
0083'                           602  KEYOUT  PROC    NEAR
0083'  53                       603          PUSH    BX
0084'  51                       604          PUSH    CX
0085'  B9 0003                  605          MOV     CX,3                    ; Number of retries
0088'  8A D8                    606          MOV     BL,AL                   ; Save keyboard command
008A'  FA                       607          CLI                             ; Protect this
008B'  E4 19                    608          IN      AL,INTA01
008D'  0C 80                    609          OR      AL,NOT ENAB7            ; Mask off the keyboard interrupt
008F'  E6 19                    610          OUT     INTA01,AL
0091'  FB                       611          STI
0092'                           612  KOLOOP:
0092'  E4 10                    613          IN      AL,KBDAT                ; Read any pending char (& throw away)
0094'  B0 35                    614          MOV     AL,(UC_ERR + UC_TXE + UC_RXE + UC_RTS)
0096'  E6 11                    615          OUT     KBCMD,AL                ; Reset any USART errors
0098'  D50A                     616          AAD                             ; Delay for the USART write-write
009A'  8A C3                    617          MOV     AL,BL                   ; Get the keyboard command back
009C'  E6 10                    618          OUT     KBDAT,AL                ; Put it in the transmit register
                                619  ;
                                620  ;  Simple timeout loop to keep from locking up if USART is bad
                                621  ;
009E'  51                       622          PUSH    CX
009F'  31 C9                    623          XOR     CX,CX                   ; Set up CX for about 1/3 second
00A1'  B4 05                    624          MOV     AH,5                    ; and AH for about 5*(1/3) = 1.6 sec
00A3'                           625  KO0:
00A3'  E4 11                    626          IN      AL,KBSTA                ;
00A5'  A8 01                    627          TEST    AL,US_TXR               ; Q: through transmitting ?
00A7'  E1 FA                    628          LOOPZ   KO0                     ;   N: loop til done OR timeout
00A9'  75 04                    629          JNZ     KO1                     ;   Y: continue
00AB'  FE CC                    630          DEC     AH                      ; decrement outer loop counter
00AD'  75 F4                    631          JNZ     KO0                     ; (fall thru if timeout)
00AF'                           632  KO1:
00AF'  59                       633          POP     CX
                                634  ;
00B0'  B0 24                    635          MOV     AL,(UC_RXE + UC_RTS)    ;   Y: turn transmitter off again
00B2'  E6 11                    636          OUT     KBCMD,AL
00B4'  B4 64                    637          MOV     AH,100                  ; Allow the keyboard up to 100 msec
00B6'                           638  KO2:                                    ;    to respond to the commmand
00B6'  E4 11                    639          IN      AL,KBSTA                ; Get USART status
00B8'  A8 02                    640          TEST    AL,US_RXR               ; Q: character received from keyboard ?
00BA'  75 0F                    641          JNZ     KO3                     ;   Y: continue
00BC'  51                       642          PUSH    CX                      ;   N: Delay for 1 msec
00BD'  B9 0126                  643          MOV     CX,294                  ; Value for 1 msec delay
00C0'  E2 FE                    644          LOOP    $                       ; (17 * .2 usec * 294) = 999.6 usec
00C2'  59                       645          POP     CX
00C3'  FE CC                    646          DEC     AH                      ; Q: timed out yet ?
00C5'  75 EF                    647          JNZ     KO2                     ;   N: loop again
00C7'  B0 FF                    648          MOV     AL,0FFH                 ;   Y: set up for 'timeout' error
00C9'  EB 1A                    649          JMP     SHORT KO_ERR            ; Report it
00CB'                           650  KO3:
00CB'  A8 38                    651          TEST    AL,(US_FE + US_OE + US_PE)  ; Q: any receiver errors ?
00CD'  75 0E                    652          JNZ     KO_BAD                  ;   Y: return the error
00CF'  E4 10                    653          IN      AL,KBDAT                ;   N: get the char (& reset interrupt)
```

```
00D1' 3C 70              654          CMP     AL,KB__OK              ; Q: was it OK status ?
00D3' 74 12              655          JZ      KO_END                 ;   Y: return with zero flag set
00D5' A8 FF              656          TEST    AL,0FFH                ; Q: is it zero for some reason ?
00D7' 75 0C              657          JNZ     KO_ERR                 ;   N: take normal error exit
00D9' FE C0              658          INC     AL                     ;   Y: Then insure non-zero
00DB' EB 08              659          JMP     SHORT KO_ERR           ; and take normal error exit
                         660  ;
00DD'                    661  KO_BAD:
00DD' E4 10              662          IN      AL,KBDAT               ; Reset any pending interrupt
00DF' B0 34              663          MOV     AL,(UC_ERR + UC_RXE + UC_RTS)
00E1' E6 11              664          OUT     KBCMD,AL               ; Reset any USART errors
00E3' B0 FE              665          MOV     AL,0FEH                ; Set up error code (receive error)
00E5'                    666  KO_ERR:
00E5' 08 C0              667          OR      AL,AL                  ; Reset zero flag (error)
00E7'                    668  KO_END:
00E7' E0 A9              669          LOOPNZ  KOLOOP          ; Loop if ((ERROR) .AND. (RETRIES .GT. 0))
                         670  ;
00E9' 50                 671          PUSH    AX                     ; Save error status
00EA' 9C                 672          PUSHF                          ;    and flag
00EB' FA                 673          CLI                            ; Protect this
00EC' E4 19              674          IN      AL,INTA01
00EE' 24 7F              675          AND     AL,ENA87               ; Reenable keyboard interrupt
00F0' E6 19              676          OUT     INTA01,AL
00F2' 9D                 677          POPF                           ; Restore error flag (and interrupts)
00F3' 58                 678          POP     AX
                         679  ;
00F4' 59                 680          POP     CX
00F5' 5B                 681          POP     BX
00F6' C3                 682          RET                            ; *** RETURN ***
                         683  ;*****************************************************************************
                         684  ;   KEYBOARD INITIALIZATION LOGIC - resets keyboard and sets up local storage
                         685  ;       This routine assumes that the Interrupt controller, USART, and the
                         686  ;       interrupt vector have already been initialized.
                         687  ;
                         688  ;   INPUT:   DS = ROMDAT
                         689  ;   OUTPUT:  (keyboard queue empty)
                         690  ;   USED:    (none)
                         691  ;   STACK:   6 bytes
                         692      ASSUME  CS:ROMCOD, DS:ROMDAT
                         693  ;-----------------------------------------------------------------------------
00F7'                    694  KEYINI  PROC    NEAR
00F7' E8 FF6C            695          CALL    FLUSH                  ; Initialize the queue
00FA' 30 C0              696          XOR     AL,AL                  ; Clear a reg
00FC' A2 000C"           697          MOV     KBMODE,AL              ; Init mode/status byte
00FF' A2 0011"           698          MOV     NUMVAL,AL              ; Init ALT/NUM stuff
0102' A2 0010"           699          MOV     NUMCNT,AL
0105' E4 10              700          IN      AL,KBDAT               ; Read rcvr to reset any interrupt
0107' B0 34              701          MOV     AL,(UC_ERR + UC_RXE + UC_RTS)
0109' E6 11              702          OUT     KBCMD,AL               ; Reset any possible errors
010B' C3                 703          RET                            ; *** RETURN ***
                         704  ;
                         705  ;*****************************************************************************
```

```
                        706   ;   KEYRST - Keyboard USART initialization routine
                        707   ;
                        708   ;   INPUT:  (none)
                        709   ;   OUTPUT: (none)
                        710   ;   USED:   AX,CX,DX,SI
                        711   ;   STACK:  2 bytes
                        712       ASSUME  DS:ROMDAT
                        713   ;-----------------------------------------------------------------------
                        714   ;
                        715   ;   Table of bytes to send to USART for reset
                        716   ;
010C'                   717   KRTABL  LABEL   BYTE
010C'  00               718           DB      00H                     ; Null command
010D'  00               719           DB      00H                     ; Null command
010E'  00               720           DB      00H                     ; Null command
010F'  65               721           DB      (UC_IRS + UC_TXE + UC_RXE + UC_RTS)     ; Internal reset
0110'  7F               722           DB      7FH     ; MODE=8 data bits, even parity, 1 stop bit, 64X clock)
0111'  30               723           DB      (UC_ERR + UC_RTS)       ; Reset any errors
0112'                   724   KRTEND  LABEL   BYTE
       =0006            725   KRTLEN  EQU     KRTEND-KRTABL           ; Length of table
                        726   ;
                        727   ;   The routine itself
                        728   ;
0112'                   729   KEYRST  PROC    NEAR
0112'  BE 010C"         730           MOV     SI,OFFSET KRTABL        ; Point to table of reset commands
0115'  B9 0006          731           MOV     CX,OFFSET KRTLEN        ; Length of table to CX
0118'  BA 0011          732           MOV     DX,KBCMD                ; Set up command port I/O address
011B'                   733   KR1:
011B'  2EAC             734           LODB    CS:BYTE PTR [SI]        ; Pick up the command byte      12
011D'  EE               735           OUT     DX,AL                   ; Send it to the command port    8
011E'  E2 FB            736           LOOP    KR1                     ; 'Til they're all done         17
                        737                                           ;                               ----
                        738                                           ;                                27
0120'  C3               739           RET                             ; *** RETURN ***          (= 5.4 usec)
                        740   ;*************************************************************************
                        741   ;   KPAUSE - Keyboard default pause key handler.  Sets the CRTHLD (CRT hold)
                        742   ;       flag when the pause key is struck in order to freeze the screen.
                        743   ;       Note that striking any other key (including the PAUSE key) will
                        744   ;       restart the CRT.
                        745   ;   INPUT:  none
                        746   ;   OUTPUT: (CRT hold flag set
                        747   ;   USED:   none
                        748   ;   STACK:  8 bytes
                        749       ASSUME  DS:ROMDAT
                        750   ;-----------------------------------------------------------------------
0121'                   751   KPAUSE  PROC    FAR     ; (Interrupt Call)
0121'  1E               752           PUSH    DS
0122'  2E8E 1E C180\     753           MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; Point to my DS
0127'  F6 06 000A" FF   754           TEST    CRTHLD,0FFH             ; Q: Is the flag already set ?
012C'  C6 06 000A" FF   755           MOV     CRTHLD,-1               ; (pre)Set the flag
0131'  74 05            756           JZ      KP1                     ;   N: Then it should be set
0133'  C6 06 000A" 00   757           MOV     CRTHLD,00               ;   Y: Then it should be reset
```

```
0138'                          758   KP1:
0138'  1F                      759           POP     DS
0139'  FB                      760           STI                             ; Interrupts back on
013A'  F9                      761           STC                             ; Set carry so no code is sent
013B'  CA 0002                 762           RET     2                       ; Simulated IRET
                               763   ;
                               764   ;*****************************************************************************
                               765   ;
                               766   ;   KEYBOARD INTERRUPT HANDLER (no regs used) - responsible for all keyboard
                               767   ;       encoding functions.
                               768   ;
                               769   ;-----------------------------------------------------------------------------
                               770   ;  The information received from the keyboard is formatted as follows:
                               771   ;
                               772   ;       ---------------------------------
                               773   ;       | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |       FIRST BYTE   ('mode' byte)
                               774   ;       ---------------------------------       (not always sent)
                               775   ;         |   |   |   |   |   |   |   |
                               776   ;         |   |   |   |   |   |   |   -----)    Control
                               777   ;         |   |   |   |   |   |   |
                               778   ;         |   |   |   |   |   |   ---------)    Alternate
                               779   ;         |   |   |   |   |   |
                               780   ;         |   |   |   |   |   -------------)    Shift
                               781   ;         |   |   |   |   |
                               782   ;         |   -------------------------------)    = 1111 (denotes 1st byte)
                               783   ;         |
                               784   ;         ---------------------------------)    Uppercase
                               785   ;
                               786   ;
                               787   ;
                               788   ;       ---------------------------------
                               789   ;       | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |       SECOND BYTE   ('scan' byte)
                               790   ;       ---------------------------------
                               791   ;         |   |   |   |   |   |   |   |
                               792   ;         |   |   |   |   |   |   |   |
                               793   ;         |   -------------------------------)    Scan code
                               794   ;         |
                               795   ;         ---------------------------------)    Repeated code (typamatic)
                               796   ;
                               797       ASSUME  CS:ROMCOD, DS:ROMDAT
                               798   ;
013E'                          799   KEYISR  PROC    FAR
013E'  1E                      800           PUSH    DS                      ;;; save current DS
013F'  2E8E 1E C180            801           MOV     DS,WORD PTR CS:DSADDR+CSWRAP ;;; set up my DS
0144'  8C 16 000E"             802           MOV     KBSSV,SS                ;;; save user's SS, SP
0148'  89 26 000D"             803           MOV     KBSPSV,SP               ;;;
014C'  8C DC                   804           MOV     SP,DS                   ;;; (Use SP as temp reg)
014E'  8E D4                   805           MOV     SS,SP                   ;;; Local stack segment = DS
0150'  BC 000F"                806           MOV     SP,OFFSET KEYISP        ;;; Set up local stack
0153'  2EFE 06 C19A            807           INC     BYTE PTR CS:INTCTR+CSWRAP ;;; Bump the interrupt counter
0158'  50                      808           PUSH    AX
0159'  53                      809           PUSH    BX                      ;;; Save commonly used registers on the stack
```

```
0000  FF FF FF FF                       DD    -1
0004  8000                              DW    8000h
0006  0058 R                           DW    OFFSET init
0008  006C R                           DW    OFFSET doio
000A  47 45 54 4F 46 53                 DB    'GETOFS  '
      20 20

0012  0000                    bufbase    DW    0
0014  0000                    bufofs     DW    0
0016  0000                    intbase    DW    0
0018  0000                    intofs     DW    0
001A  00                      crow       DB    0
001B  00                      ccol       DB    0
001C     28 [                           DB    40 DUP(0)
               00
                  ]

0044  00                      stack      DB    0
0045  47 65 74 6F 66 73       starttxt   DB    'Getofs loaded X ',13,10,'$'
```

ABCDB:2 A 6

F400 BC9B

03 CE
0629

872E:0401 - [illegible]

D45D - code

06CC

B4 3D
B0 00
4BA

1 2 4
1 D D

16
6

68F2

6908
10F

E800:889

```
015A'  06                      810             PUSH    ES
                               811     ;
015B'  51                      812             PUSH    CX
015C'  57                      813             PUSH    DI              ;;; Locally used register (don't forget to POP)
015D'  56                      814             PUSH    SI
015E'  E4 11                   815             IN      AL,KBSTA        ;;; Look at status
0160'  A8 02                   816             TEST    AL,US_RXR       ;;; Q: Was it the receiver ?
0162'  74 1F                   817             JZ      KBXTSS          ;;;   N: ignore any other (xmit or coprocessor)
0164'  A8 38                   818             TEST    AL,(US_FE + US_OE + US_PE) ; Q: any errors ?
0166'  B0 34                   819             MOV     AL,(UC_ERR + UC_RXE + UC_RTS)
0168'  E6 11                   820             OUT     KBCMD,AL        ;;;   (Reset any errors)
016A'  75 17                   821             JNZ     KBXTSS          ;;;   Y: ignore the character
016C'  E4 10                   822             IN      AL,KBDAT        ;;;   N: get the char & reset the interrupt
016E'  FB                      823             STI                     ;;; Interrupts OK now
016F'  8A E0                   824             MOV     AH,AL           ; Save the character for a while
0171'  24 78                   825             AND     AL,NOT KBMMSK   ; Mask off the 'mode' field
0173'  3C 78                   826             CMP     AL,NOT KBMMSK   ; Q: is this a mode byte ?
0175'  75 0F                   827             JNE     KBIO            ;   N: continue
0177'  80 E4 87                828             AND     AH,KBMMSK       ;   Y: Mask off all but the mode bits
017A'  80 26 000C" 78          829             AND     KBMODE,NOT KBMMSK
017F'  08 26 C 0C"             830             OR      KBMODE,AH       ; Update mode/status byte
0183'                          831     KBXTSS:                         ; ('KBIXIT' stepping stone, jmp ) 128)
0183'  E9 0127                 832             JMP     KBIXIT          ; *** EXIT ***
0186'                          833     KBIO:
0186'  8A C4                   834             MOV     AL,AH           ; Get scan code in AL
0188'  80 E4 7F                835             AND     AH,07FH         ; Mask off the typamatic bit
018B'  80 FC 68                836             CMP     AH,MAXKEY       ; Q: scan code too high ?
018E'  77 F3                   837             JA      KBXTSS          ;   Y: ignore it (what was it, anyway ?)
0190'  3C 00                   838             CMP     AL,00           ; Q: is it zero (not allowed) ?
0192'  74 EF                   839             JE      KBXTSS          ;   Y: ignore it
0194'  3C 30                   840             CMP     AL,DELKEY       ; Q: Is this the Delete key ?
0196'  74 04                   841             JE      KBI1_5          ;   Y: Then check for CTRL/ALT
0198'  3C 5E                   842             CMP     AL,ADLKEY       ; Q: What about the other Delete key ?
019A'  75 11                   843             JNE     KBI1            ;   N: Then keep going
019C'                          844     KBI1_5:
019C'  8A 26 000C"             845             MOV     AH,KBMODE       ;   Y: Get mode/status byte
01A0'  80 E4 03                846             AND     AH,(ALTMSK + CTLMSK)
01A3'  80 FC 03                847             CMP     AH,(ALTMSK + CTLMSK) ; Q: Are CONTROL and ALT depressed also ?
01A6'  75 05                   848             JNE     KBI1            ;   N: Continue
01A8'  EA 0002" 0000"          849             JMP     FAR PWRUP       ;   Y: *** GO DO POWERUP RESET ***
01AD'                          850     KBI1:
01AD'  80 26 000C" F7          851             AND     KBMODE,NOT TYPMSK       ; Reset the typamatic bit
01B2'  A8 80                   852             TEST    AL,TYPBIT       ; Q: Typamatic character (high bit set) ?
01B4'  74 05                   853             JZ      KBI2            ;   N: Keep going (bit already reset)
01B6'  80 0E 000C" 08          854             OR      KBMODE,TYPMSK   ;   Y: Then set the typamatic bit in KBMODE
01BB'                          855     KBI2:
01BB'  8A 26 000C"             856             MOV     AH,KBMODE       ; AL has scan code;  AH has mode
01BF'  F9                      857             STC                     ; Set the carry-flag - If the user encodes the
                               858                                     ; key, he should reset the carry flag before
                               859                                     ; before returning; if the key is to be
                               860                                     ; ignored, return with carry set and AL=0FFH.
01C0'  CD 5B                   861             INT     MAPINT          ; Q: Does anybody want to take a shot at it ?
```

```
01C2'  73 44                  862             JNB     KBIQSS          ;   Y: User encoded key - put it in the queue
01C4'  3C FF                  863             CMP     AL,-1           ; Q: should this key be ignored ?
01C6'  74 BB                  864             JE      KBXTSS          ;   Y: go to exit
                              865     ;                               ;   N: Encode it
                              866     ;   Check for ALT-NUM fanciness
                              867     ;
01C8'  F6 C4 02               868             TEST    AH,ALTMSK       ; Q: Are we in ALT mode ?
01CB'  74 3E                  869             JZ      KBIENC          ;   N: Continue with normal encoding
01CD'  0E                     870             PUSH    CS
01CE'  07                     871             POP     ES              ; Set up ES to point to tables
01CF'  BF 0007"               872             MOV     DI,OFFSET KBNUMK ; Y: Point to Numeric key table
01D2'  B9 000A                873             MOV     CX,10           ; Table will always contain ten keys
01D5'  24 7F                  874             AND     AL,NOT TYPBIT   ; Strip off any typamatic bit so index is OK
01D7'  F2                     875             REPNE
01D8'  AE                     876             SCAB    BYTE PTR [DI]   ; Q: Is this a numeric key ?
01D9'  75 30                  877             JNE     KBIENC          ;   N: Then skip rest of numeric stuff
01DB'  F6 06 000C" 08         878             TEST    KBMODE,TYPMSK   ; Q: Was this a typamatic character ?
01E0'  75 A1                  879             JNZ     KBXTSS          ;   Y: No typamatics allowed in ALT/NUM mode
01E2'  81 EF 0007"            880             SUB     DI,OFFSET KBNUMK ;  N: Get offset into table
01E6'  4F                     881             DEC     DI              ; DI now has value of key (0-9)
01E7'  A0 0011"               882             MOV     AL,NUMVAL       ; Pick up the current value
01EA'  B4 0A                  883             MOV     AH,10           ;   and multiply it by 10
01EC'  F6 E4                  884             MUL     AH
01EE'  01 F8                  885             ADD     AX,DI           ; Add in the latest arrival
01F0'  A2 0011"               886             MOV     NUMVAL,AL       ;  ... and update that value in memory
01F3'  FE 06 0010"            887             INC     NUMCNT          ; Bump the keystroke counter
01F7'  80 3E 0010" 03         888             CMP     NUMCNT,3        ; Q: Have there been 3 'ALT/NUM' keys, yet ?
01FC'  75 85                  889             JNZ     KBXTSS          ;   N: Ignore this key and go get next one
01FE'                         890     KBI3:
01FE'  B4 00                  891             MOV     AH,0            ;   Y: Pass scan code of zero
0200'  88 26 0011"            892             MOV     NUMVAL,AH       ; And clear NUMVAL accumulator for next time
0204'  88 26 0010"            893             MOV     NUMCNT,AH       ;   ... also the keystroke counter
0208'                         894     KBIQSS:                         ; 'Stepping stone' for KBIQ (JMP ) 128)
0208'  E9 007E                895             JMP     KBIQ            ; Go put the created character in the queue
                              896     ;
                              897     ;   Normal encoding starts here
                              898     ;
020B'                         899     KBIENC:
020B'  86 06 0011"            900             XCHG    NUMVAL,AL       ; Get the current value of ALT/NUM accumulator
020F'  08 C0                  901             OR      AL,AL           ; Q: Is an incomplete ALT/NUM value waiting ?
0211'  75 EB                  902             JNE     KBI3            ;   Y: Then send it and blow off this keystroke
0213'  86 06 0011"            903             XCHG    NUMVAL,AL       ;   N: Continue normal encoding
0217'  FE C8                  904             DEC     AL              ; Tables start at 0 not 1, so adjust scan code
0219'  F6 C4 02               905             TEST    AH,ALTMSK       ; Q: is this an ALT function ?
021C'  BB 0004"               906             MOV     BX,OFFSET KB_ALT;    (Point to ALT encoding table)
021F'  75 13                  907             JNZ     KBIMAP          ;   Y: go to it
0221'  F6 C4 01               908             TEST    AH,CTLMSK       ; Q: is this a CONTROL function ?
0224'  BB 0005"               909             MOV     BX,OFFSET KBCTRL;    (Point to CONTROL encoding table)
0227'  75 0B                  910             JNZ     KBIMAP          ;   Y: go map it
0229'  F6 C4 04               911             TEST    AH,SHFMSK       ; Q: is either SHIFT key down ?
022C'  BB 0008"               912             MOV     BX,OFFSET KBSHFT;    (point to SHIFT encoding table)
022F'  75 03                  913             JNZ     KBIMAP          ;   Y: go map it
```

```
0231'  BB 0009"                914             MOV     BX,OFFSET KBUNSH;   N: then it must be NORMAL encoding
                               915     ;       JMP     SHORT KBIMAP    ; Go map it
                               916     ;
                               917     ;   Map the character according to the tables
                               918     ;
0234'                          919     KBIMAP:
0234'  24 7F                   920             AND     AL,NOT TYPBIT   ; Strip off the typamatic bit
0236'  2ED7                    921             XLAT    BYTE PTR CS:[BX] ; Map the key according to appropriate table
0238'  3C FF                   922             CMP     AL,-1           ; Q: Should this key be ignored ?
023A'  74 0D                   923             JE      KBXSS1          ;   Y: Exit
023C'  3C FE                   924             CMP     AL,PAUFNC       ; Q: Is this the Program Pause Key code ?
023E'  75 0B                   925             JNE     KBI4            ;   N: Continue
0240'  30 C0                   926             XOR     AL,AL           ;   Y: Flag this as extended code
0242'  B4 01                   927             MOV     AH,KBPPAU       ; Set up Program Pause function code
0244'  F8                      928             CLC                     ; reset carry flag in preparation
0245'  CD 5C                   929             INT     PAUINT          ; Call the Pause routine
0247'  73 40                   930             JNB     KBIQ            ; If user encoded the key, then put it in queue
0249'                          931     KBXSS1:                         ; KBIXIT stepping stone # 2
0249'  EB 62                   932             JMP     SHORT KBIXIT    ;   else exit
024B'                          933     KBI4:
024B'  3C FD                   934             CMP     AL,PBKFNC       ; Q: Is this the Program Break Key code ?
024D'  75 0B                   935             JNE     KBI5            ;   N: Continue
024F'  30 C0                   936             XOR     AL,AL           ;   Y: Flag this as an extended code
0251'  B4 00                   937             MOV     AH,KBPBRK       ; Set up Program Break function code
0253'  F8                      938             CLC                     ; reset carry flag in preparation
0254'  CD 5D                   939             INT     PBKINT          ; Call the Break routine
0256'  73 31                   940             JNB     KBIQ            ; If user encoded the key, then put it in queue
0258'  EB 53                   941             JMP     SHORT KBIXIT    ;   else exit
025A'                          942     KBI5:
025A'  3C FC                   943             CMP     AL,PSCFNC       ; Q: Is this the Print Screen Key code ?
025C'  75 07                   944             JNE     KBI6            ;   N: Continue
025E'  F9                      945             STC                     ;   Y: Set carry flag in preparation
025F'  CD 5E                   946             INT     PSCINT          ; Call the Print Screen routine
0261'  73 26                   947             JNB     KBIQ            ; If user encoded the key, then put it in queue
0263'  EB 48                   948             JMP     SHORT KBIXIT    ;   else exit
0265'                          949     KBI6:
0265'  A8 80                   950             TEST    AL,80H          ; Q: Should I look in the function key table ?
0267'  74 0D                   951             JZ      KBI7            ;   N: High bit not set so not a function key
0269'  BB 0006"                952             MOV     BX,OFFSET KBFUNC ;  Y: Point to function key table
026C'  24 7F                   953             AND     AL,7FH          ; Strip off the high bit
026E'  2ED7                    954             XLAT    BYTE PTR CS:[BX] ; and pick up the real function key code
0270'  8A E0                   955             MOV     AH,AL           ; Put it in AH
0272'  30 C0                   956             XOR     AL,AL           ; and zero into AL
0274'  EB 13                   957             JMP     SHORT KBIQ
0276'                          958     KBI7:
0276'  F6 C4 80                959             TEST    AH,UPCMSK       ; Q: is the UPPERCASE LOCK switch down ?
0279'  B4 00                   960             MOV     AH,0            ; (Here set up standard ASCII return w/AH=0)
027B'  74 0C                   961             JZ      KBIQ            ;   N: encoding done, put char in queue
027D'  3C 61                   962             CMP     AL,'a'          ;   Y: then check for lower case
027F'  72 08                   963             JB      KBIQ
0281'  3C 7B                   964             CMP     AL,'z'+1
0283'  73 04                   965             JNB     KBIQ
```

```
0285'  2C 20                966             SUB     AL,20H          ; ... and map to upper
0287'  EB 00                967             JMP     SHORT KBIQ      ; put it in the queue
                            968     ;
                            969     ;   Put the encoded character in the queue
                            970     ;
0289'                       971     KBIQ:
0289'  F6 06 000A" FF       972             TEST    CRTHLD,0FFH     ; Q: Is the CRT in the hold mode ?
028E'  74 05                973             JZ      KBIQO           ;   N: Then continue
0290'  C6 06 000A" 00       974             MOV     CRTHLD,00       ;   Y: Then reset the flag and unfreeze CRT
0295'                       975     KBIQO:
0295'  F6 06 000C" 08       976             TEST    KBMODE,TYPMSK   ; Q: Was this a typamatic character ?
029A'  74 09                977             JZ      KBIQ1           ;   N: Continue
029C'  80 3E 0012" 01       978             CMP     QDEPTH,1        ; Q: Is there more than one character waiting ?
02A1'  7E 02                979             JLE     KBIQ1           ;   N: Continue
02A3'  EB 08                980             JMP     SHORT KBIXIT    ;   Y: Then blow off the typamatic char
02A5'                       981     KBIQ1:                          ;Put the char into the queue
02A5'  E8 0015              982             CALL    KBFQ            ; Q: Did the character fit?
02A8'  75 03                983             JNZ     KBIXIT          ;     Y: exit
02AA'  E8 0003"             984             CALL    BBEEP           ;     N: ring bell
02AD'                       985     KBIXIT:
02AD'  5E                   986             POP     SI              ; Pop locally-used registers
02AE'  5F                   987             POP     DI
02AF'  59                   988             POP     CX
02B0'  8E 06 000E"          989             MOV     ES,KBSSSV       ; ES = SS of interrupted code
02B4'  8B 1E 000D"          990             MOV     BX,KBSPSV       ; BX = SP of interrupted code
02B8'  2EFF 2E C164         991             JMP     DWORD PTR CS:XITVEC*4+CSWRAP ; Take common interrupt exit
                            992     ;*****************************************************************************
                            993     ;   KBFQ - Routine to place 16-bit character in queue.  Called from KEYISR and
                            994     ;       as opcode 5 to simulate the keyboard putting key into queue.
                            995     ;
                            996     ;   INPUT:  AX = character to be placed into queue
                            997     ;           (QREAR),(QFRONT)
                            998     ;   OUTPUT: ZF = 0  if character was placed into the buffer
                            999     ;           ZF = 1 if buffer was full (char not placed, remains AX)
                           1000     ;   USED:   SI,(AX if put into buffer)
                           1001     ;   STACK:
                           1002     ;-----------------------------------------------------------------------------
02BD'                      1003     KBFQ    PROC    NEAR
02BD'  FA                  1004             CLI                     ;   Shields up
02BE'  8B 36 0014"         1005             MOV     SI,QREAR        ;;;
02C2'  E8 FD91             1006             CALL    INCQ            ;;; Bump queue pointer
02C5'  3B 36 0013"         1007             CMP     SI,QFRONT       ;;; Q: Any room in the queue ?
02C9'  74 10               1008             JZ      KBIQ2           ;;;   N: EXIT
                           1009                                     ;;;   Y: put character in queue
02CB'  89 04               1010             MOV     [SI],AX         ;;; Put character in queue
02CD'  89 36 0014"         1011             MOV     QREAR,SI        ;;; (added at rear of queue)
02D1'  FE 06 0012"         1012             INC     QDEPTH          ;;; Bump char-waiting count
02D5'  CD 5F               1013             INT     QUEINT          ;;; Tell the user (OS?) we just queued a char
02D7'  31 C0               1014             XOR     AX,AX           ;;;
02D9'  FE C0               1015             INC     AL              ;;; Insure Z-flag reset
02DB'  FB                  1016     KBIQ2:  STI                     ;;; Shields down
02DC'  C3                  1017             RET
```

```
                                1018            END

```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ;  TITLE    - KEYTST - Keyboard initialization logic
                        3   ;  COMPUTER - 8088 Assembly Language
                        4   ;  ABSTRACT - This module contains the powerup initialization code for the
                        5   ;       keyboard.
                        6   ;*****************************************************************************
                        7           NAME    KEYTST - Keyboard initialization logic
                        9   ;*****************************************************************************
                       10   ;                           PUBLIC DEFINITIONS
                       11   ;*****************************************************************************
                       12   ;
                       13           PUBLIC  KEYTST
                       14   ;
                       15   ;*****************************************************************************
                       16   ;                           EXTERNAL REFERENCES
                       17   ;*****************************************************************************
                       18   ;
                       19           EXTRN   DSPKER:NEAR     ; Keyboard error display routine        (OUTPUT)
                       20           EXTRN   KEYINI:NEAR     ; Keyboard driver initialization        (KEYDSR)
                       21           EXTRN   KEYOUT:NEAR     ; Keyboard output routine               (KEYDSR)
                       22           EXTRN   PUPTS1:NEAR     ; Continuation of powerup test logic    (PUPTS1)
                       23   ;
                       24   ;   Keyboard error codes
                       25   ;
                       26           EXTRN   KNIERR:ABS      ; Keyboard 'not installed' error code   (ROMERR)
                       27           EXTRN   KNRERR:ABS      ; Keyboard 'no response' error code     (ROMERR)
                       28           EXTRN   KRAERR:ABS      ; Keyboard 'RAM failure' error code     (ROMERR)
                       29           EXTRN   KRCERR:ABS      ; Keyboard 'receive error' error code   (ROMERR)
                       30           EXTRN   KROERR:ABS      ; Keyboard 'ROM failure' error code     (ROMERR)
                       31           EXTRN   KUNERR:ABS      ; Keyboard 'unknown' error code         (ROMERR)
                       32   ;
                       33   ;*****************************************************************************
                       34   ;                           LOCAL CONSTANTS
                       35   ;       INCLUDE PEQ:PORTADDR.EQU
                       36   ;       INCLUDE PEQ:VECTOR.EQU
                       37   ;       INCLUDE PEQ:INTCTLR.EQU
                       38   ;       INCLUDE PEQ:USART.EQU
                       39   ;       INCLUDE PEQ:KBCMD.EQU
                       40   ;*****************************************************************************
                      218   ;*****************************************************************************
                      219   ;                           CODE SEGMENT DEFINITION
                      220   ;*****************************************************************************
                      221   ;
=0000                 222           SECT    ROMCOD
                      223           ASSUME  CS:ROMCOD
                      224   ;
                      225   ;*****************************************************************************
                      226   ;                           MODULE ENTRY POINT
                      227   ;*****************************************************************************
                      228   ;
                      229           ASSUME  DS:ROMDAT
                      230   ;
```

```
0000'                           231   KEYTST  PROC    NEAR
                                232   ;
0000'  2E8E 1E C180             233           MOV     DS,WORD PTR CS:DSADDR+CSWRAP
                                234   ;
                                235   ;   'KEYRST' has already been called (from INTTST) so that the
                                236   ;   USART is already set up.
                                237   ;
                                238   ;       CALL    KEYRST                  ; Init the USART
0005'  E4 10                    239           IN      AL,KBDAT                ; Read RCVR to reset any interrupts
0007'  B0 34                    240           MOV     AL,(UC_ERR + UC_RXE + UC_RTS)
0009'  E6 11                    241           OUT     KBCMD,AL                ; Reset any errors
                                242   ;
                                243   ;   Check for keyboard installed (plugged in) by looking at transmit
                                244   ;   signal looped-back through the DSR pin.  Simple test, just looks
                                245   ;   for a wiggle, doesn't time it.
                                246   ;
000B'  E4 11                    247           IN      AL,KBSTA                ; Look at USART status
000D'  A8 80                    248           TEST    AL,US_DSR               ; Q: is line idle (pulled low)
000F'  75 19                    249           JNZ     KEYNIS                  ;   N: then keyboard isn't there
0011'  B0 55                    250           MOV     AL,55H                  ; This 'command' shouldn't do
0013'  E6 10                    251           OUT     KBDAT,AL                ;    anything to the keyboard
0015'  B0 25                    252           MOV     AL,(UC_RXE + UC_TXE + UC_RTS)
0017'  E6 11                    253           OUT     KBCMD,AL                ; Enable the transmitter
                                254   ;
0019'  B4 05                    255           MOV     AH,5                    ; If line hasn't wiggled in 5 msec,
                                256                                           ;    then give it up
001B'                           257   KTO:
001B'  E4 11                    258           IN      AL,KBSTA                ; Look at DSR (xmitter loopback)
001D'  A8 80                    259           TEST    AL,US_DSR               ; Q: Has it gone high ?
001F'  75 0D                    260           JNZ     KT1                     ;   Y: Keyboard there, continue
0021'  B9 0126                  261           MOV     CX,294                  ;   N: delay a millisecond
0024'  E2 FE                    262           LOOP    $
0026'  FE CC                    263           DEC     AH                      ; Unbump msec counter
0028'  75 F1                    264           JNZ     KTO                     ; Loop if not timed out
002A'  B4 05*                   265   KEYNIS: MOV     AH,KNIERR               ; Report keyboard 'Not installed'
002C'  EB 39                    266           JMP     SHORT KEYERR
                                267   ;
                                268   ;   Simple timeout loop to keep from locking up if USART is bad
                                269   ;
002E'                           270   KT1:
002E'  E4 11                    271           IN      AL,KBSTA                ; Look at status
0030'  A8 01                    272           TEST    AL,US_TXR               ; Q: through transmitting ?
0032'  E1 FA                    273           LOOPZ   KT1                     ;   N: loop til done OR timeout
0034'  75 04                    274           JNZ     KT2                     ;   Y: continue
0036'  FE CC                    275           DEC     AH                      ; decrement outer loop counter
0038'  75 F4                    276           JNZ     KT1                     ; (fall thru if timeout)
                                277   ;
003A'                           278   KT2:
003A'  B0 24                    279           MOV     AL,(UC_RXE + UC_RTS)    ; Then shut it off
003C'  E6 11                    280           OUT     KBCMD,AL
                                281   ;
003E'  FA                       282           CLI                             ; Protect this
```

```
003F'  E4 19            283             IN      AL,INTA01
0041'  24 7F            284             AND     AL,ENAB7                ; Enable the keyboard interrupt
0043'  E6 19            285             OUT     INTA01,AL
0045'  FB               286             STI
                        287     ;
0046'  B0 00            288             MOV     AL,KBPWRU
0048'  E8 0003"         289             CALL    KEYOUT                  ; Send reset code to keyboard
004B'  74 21            290             JZ      KEYFIN                  ; Exit if no errors
004D'  B4 06*           291             MOV     AH,KNRERR               ; Report keyboard 'No response'
004F'  3C FF            292             CMP     AL,0FFH                 ; Q: did KB answer at all ?
0051'  74 14            293             JZ      KEYERR                  ;   N: report 'No Response'
0053'  B4 08*           294             MOV     AH,KRCERR               ; Report keyboard 'Receiver error'
0055'  3C FE            295             CMP     AL,0FEH                 ; Q: any receive errors ?
0057'  74 0E            296             JZ      KEYERR                  ;   Y: report 'Receiver error'
                        297     ;
                        298     ;   At this point, assume its a 'regular' error
                        299     ;
0059'  B4 09*           300             MOV     AH,KROERR               ; Report keyboard 'ROM failure'
005B'  3C 71            301             CMP     AL,KB_ROM               ; Q: Was it a ROM failure ?
005D'  74 08            302             JZ      KEYERR                  ;   Y: report the error
005F'  B4 07*           303             MOV     AH,KRAERR               ; Report keyboard 'RAM failure'
0061'  3C 72            304             CMP     AL,KB_RAM               ; Q: Was it a RAM failure ?
0063'  74 02            305             JZ      KEYERR                  ;   Y: report the error
                        306     ;                                       ;   N: report 'Unknown' error
0065'  B4 0A*           307             MOV     AH,KUNERR               ; Report keyboard 'Unknown error'
                        308     ;
0067'  30 D2            309     KEYERR: XOR     DL,DL                   ; Nothing for parallel port
0069'  8A C4            310             MOV     AL,AH                   ;
006B'  E8 0001"         311             CALL    DSPKER                  ; Put error on the screen
                        312     ;
006E'  E8 0002"         313     KEYFIN: CALL    KEYINI                  ; Do driver initialization in KEYDSR
                        314     ;       CALL    DECLED                  ; *** NO LED code for keyboard ***
0071'  E9 0004"         315             JMP     PUPTS1                  ; Go to next
                        316     ;
                        317             END

No errors detected
```

```
                                1  ;*****************************************************************************
                                2  ;  TITLE    - MSCTST - Miscellaneous initialization
                                3  ;  COMPUTER - 8088 Assembly Language
                                4  ;  ABSTRACT - This module is responsible for initializing the system
                                5  ;       timer, the clock DSR, and updating the System Configuration
                                6  ;       word with various information.  These routines assume the
                                7  ;       System Config words were initialized to 0000H by RAMINI.
                                8  ;*****************************************************************************
                                9          NAME    MSCTST - Miscellaneous initialization
                               11  ;*****************************************************************************
                               12  ;                           PUBLIC DEFINITIONS
                               13  ;*****************************************************************************
                               14  ;
                               15          PUBLIC  MSCTBT
                               16  ;
                               17  ;*****************************************************************************
                               18  ;                           EXTERNAL REFERENCES
                               19  ;*****************************************************************************
                               20  ;
                               21          EXTRN   KEYTST:NEAR     ; Keyboard test logic                   (KEYTST)
                               22          EXTRN   SETDAT:NEAR     ; Date setting logic                    (CLKDSR)
                               23          EXTRN   SETTIM:NEAR     ; Time setting logic                    (CLKDSR)
                               24  ;
=0000                          25          SECT    ROMDAT
                               26          EXTRN   DSKC_M:BYTE     ; Memory copy of disk control latch     (ROMDAT)
                               27          EXTRN   QUEUE:WORD      ; Keyboard queue area (used for temp)   (ROMDAT)
                               28          EXTRN   SYSCON:WORD     ; System Configuration word             (ROMDAT)
                               29  ;
                               30  ;*****************************************************************************
                               31  ;                           LOCAL CONSTANTS
                               32  ;*****************************************************************************
                               33  ;
=0000                          34  TBTLOC  EQU     0000H           ; Offset used for installation determination
=1144                          35  TSTVAL  EQU     1144H           ; Read/write value for installation testing
                               36
                               37  ;       INCLUDE PEQ:PORTADDR.EQU
                               38  ;       INCLUDE PEQ:INTCTLR.EQU
                               39  ;       INCLUDE PEQ:TIMERS.EQU
                               40  ;       INCLUDE PEQ:LATCHES.EQU
                               41  ;       INCLUDE PEQ:SYSCON.EQU
                               42  ;       INCLUDE PEQ:VECTOR.EQU
                               43  ;       INCLUDE PEQ:COMM.EQU
                               44  ;       INCLUDE PEQ:CRT.EQU
                               45  ;
                              429  ;*****************************************************************************
                              430  ;                           CODE SEGMENT DEFINITION
                              431  ;*****************************************************************************
                              432  ;
=0000                         433          SECT    ROMCOD
                              434          ASSUME  CS:ROMCOD
0000' 00                      435  SCCTBL  DB      SELWR0                  ;;; Null command to SCC
0001' 09                      436          DB      SELWR9                  ;;; Select register 9 (WR9)
```

```
                        1   ;*****************************************************************************
                        2   ;  TITLE    - OUTPUT - Various screen output and error-handling routines
                        3   ;  COMPUTER - 8088 Assembly Language
                        4   ;  ABSTRACT - This module contains the error-reporting and the general-
                        5   ;       purpose screen output routines.
                        6   ; * Add entry point for external users of DSPERR which accepts error code in BX
                        7   ;*****************************************************************************
                        8           NAME    OUTPUT - Screen output and error-handling routines  -
                        9   ;*****************************************************************************
                       10   ;                           PUBLIC DEFINITIONS
                       11   ;*****************************************************************************
                       12   ;
                       13           PUBLIC  ARGHHH
                       14           PUBLIC  DSPDIE
                       15           PUBLIC  DSPERR
                       16           PUBLIC  DSPKER
                       17           PUBLIC  DSPERZ
                       18           PUBLIC  FATAL
                       19           PUBLIC  MSG
                       20           PUBLIC  NMIGO
                       21           PUBLIC  PUPNMI
                       22           PUBLIC  WILD$$
                       23   ;
                       24   ;*****************************************************************************
                       25   ;                           EXTERNAL REFERENCES
                       26   ;*****************************************************************************
                       27   ;
=0000                  28           SECT    ROMCOD
                       29           EXTRN   CRTOUT:NEAR     ; CRT character output routine          (CRTDSR)
                       30           EXTRN   FATERR:ABS      ; Fatal software error code             (ROMERR)
                       31           EXTRN   IVIERR:ABS      ; Invalid interrupt Interrupt port code(TSTERR)
                       32           EXTRN   LED100:ABS      ; Interrupt error code                  (ROMERR)
                       33           EXTRN   LED101:ABS      ; LED error code (used for PUPNMI)      (ROMERR)
                       34           EXTRN   PWRUP:FAR       ; Powerup reset routine                 (RESET)
                       35           EXTRN   RMPERR:ABS      ; 'RAM parity error'                    (ROMERR)
                       36           EXTRN   WLDERR:ABS      ; Wild interrupt error code             (ROMERR)
                       37           EXTRN   XNMERR:ABS      ; 'Unexpected NMI' error code           (ROMERR)
                       38   ;
=0000                  39           SECT    ROMDAT
                       40           EXTRN   DSKC_M:BYTE     ; RAM copy of disk control port         (ROMDAT)
                       41           EXTRN   PRCO_M:BYTE     ; RAM copy of printer control port      (ROMDAT)
```

```
                                43  ;*****************************************************************************
                                44  ;                               LOCAL CONSTANTS
                                45  ;*****************************************************************************
                                46  ;
          =061B                 47  BELTMR   EQU     1563            ; 1563 ==) 800 Hz for system beeper
          =000D                 48  CR       EQU     0DH
          =000A                 49  LF       EQU     0AH
                                50
                                51  ;       INCLUDE PEQ:CRTOP.EQU
                                52  ;       INCLUDE PEQ:INTCTLR.EQU
                                53  ;       INCLUDE PEQ:CRT.EQU
                                54  ;       INCLUDE PEQ:LATCHES.EQU
                                55  ;       INCLUDE PEQ:PORTADDR.EQU
                                56  ;       INCLUDE PEQ:TIMERS.EQU
                                57  ;       INCLUDE PEQ:USART.EQU
                                58  ;       INCLUDE PEQ:VECTOR.EQU
                                59  ;
                               384  ;*****************************************************************************
                               385  ;                           CODE SEGMENT DEFINITION
                               386  ;*****************************************************************************
                               387  ;
          =0000                388          SECT    ROMCOD
                               389          ASSUME  CS:ROMCOD
                               390  ;
                               391  ;*****************************************************************************
                               392  ;                           LOCAL DATA AREA
                               393  ;*****************************************************************************
                               394  ;
0000'  0D 0A 0A                395  KERMSG   DB      CR,LF,LF
0003'  2A 2A 20 48 65 79       396          DB      '** Keyboard Error **  '
0009'  62 6F 61 72 64 20
000F'  45 72 72 6F 72 20
0015'  2A 2A 20 20
0019'  00                      397          DB      00                              ; End of string
                               398  ;
001A'  0D 0A 0A                399  SER1MS   DB      CR,LF,LF
001D'  2A 2A 20 53 79 73       400  SER2MS   DB      '** System Error **  '
0023'  74 65 6D 20 45 72
0029'  72 6F 72 20 2A 2A
002F'  20 20
0031'                          401  SEREND   LABEL   BYTE
          =0014                402  SERMSL   EQU     SEREND-SER2MS
0031'  00                      403          DB      00              ; End of string
                               404  ;
0032'                          405  BINASC   LABEL   BYTE            ; Binary to ASCII lookup table - used by XLAT
0032'  30 31 32 33 34 35       406          DB      '0123456789ABCDEF'
0038'  36 37 38 39 41 42
003E'  43 44 45 46 \
                               407  ;
0042'                          408  DIESTK   LABEL   WORD
0042'  0082"                   409          DW      OFFSET DD0      ; Fake return address for DSPDIE
0044'  0087"                   410          DW      OFFSET DD1      ; Fake return address for DSPDIE
```

```
0046'  008E"                411          DW      OFFSET DD2      ; Fake return address for DSPDIE
0048'  0094"                412          DW      OFFSET DD3      ; Fake return address for DSPDIE
004A'  0099"                413          DW      OFFSET DD4      ; Fake return address for DSPDIE
004C'  00A0"                414          DW      OFFSET DD5      ; Fake return address for DSPDIE
                            415  ;
                            416  ;*****************************************************************************
                            417  ;                           MODULE ENTRY POINT
                            418  ;*****************************************************************************
                            419  ;
                            420  ;   DSPDIE - Display error message and 'die'
                            421  ;       This routine is intended for use by the powerup diagnostics.
                            422  ;       If an error occurs during powerup tests, this routine should be
                            423  ;       'jumped to' with the low-order byte of the error code in AL.
                            424  ;       It displays the message ' ** System Error **  - 00xx' by
                            425  ;       writing directly to screen RAM (because during powerup the CRT
                            426  ;       DSR has not yet been initialized).  If the factory test interface
                            427  ;       is connected (signified by 'PAPER OUT' and 'NOT BUSY'), the error
                            428  ;       code is also sent out the parallel printer port.  A code passed
                            429  ;       in register DL will be written to the printer port (as one character)
                            430  ;       after the ASCII error code.  The system 'bell' is then sounded and
                            431  ;       the CPU halts with interrupts disabled.  Any restart attempts
                            432  ;       must be by cycling power.
                            433  ;
                            434  ;   INPUT:  AL = 2-digit error code to be displayed (DSPDIE adds leading 00)
                            435  ;                Error code is not range-checked, but should be in the
                            436  ;                range from 0 to 9 because of simple ASCII conversion.
                            437  ;           DL = LED value to be place on the parallel port before dying.
                            438  ;   OUTPUT: (error message to screen/parallel port, beep)
                            439  ;   USED:   (at this point, nobody cares anyway)
                            440  ;   STACK:
                            441      ASSUME  CS:ROMCOD, DS:ROMDAT, ES:CRTDAT
                            442  ;-----------------------------------------------------------------------------
004E'                       443  DSPDIE  PROC    NEAR
004E'  FC                   444          CLD                                 ;;; Forward strings
004F'  FA                   445          CLI                                 ;;; Quit listening
0050'  2E8E 1E C180         446          MOV     DS,WORD PTR CS:DSADDR+CSWRAP ;;; Set up local DS
0055'  BE 001D"             447          MOV     SI,OFFSET SER2MS            ;;; Point to 'SYSTEM ERROR' message
0058'  BB 0000'             448          MOV     BX,CRTDAT                   ;;;
005B'  8E C3                449          MOV     ES,BX                       ;;; ES =  screen RAM segment
005D'  26C6 06 1800 0F      450          MOV     BYTE PTR ES:ATTLAT,0FH      ;;; Set attribute latch to default
0063'  BF 0000              451          MOV     DI,0000                     ;;; Start in up left corner of screen
0066'  B9 0014              452          MOV     CX,OFFSET SERMSL            ;;; Get length of message
0069'  F2                   453          REP                                 ;;;
006A'  2EA4                 454          MOVS    BYTE PTR [DI],CS:BYTE PTR [SI]  ;;; Put string on screen
006C'  8C C9                455          MOV     CX,CS                       ;;;
006E'  8E D1                456          MOV     SS,CX                       ;;;
0070'  BC 0042"             457          MOV     SP,OFFSET DIESTK            ;;; Set up fake stack for no-RAM calls
0073'  0C 70                458          OR AL,(PRAUTO+PRSTRB+PRINIT)        ;;; Force printer interface to active
0075'  8A E8                459          MOV     CH,AL                       ;;; Save LED state for now.
0077'  F6 D0                460          NOT     AL                          ;;; Invert the error code byte
0079'  24 07                461          AND     AL,07H                      ;;; Mask off low-order 3 bits
007B'  8A F0                462          MOV     DH,AL                       ;;; Save AL error
```

```
                        463   ;
007D'  BB 202D          464           MOV     AX,' -'                 ;;; Reverses to '- '
0080'  EB 27            465           JMP     SHORT PRCWRD            ;;; Data to screen & test I/F (call)
0082'                   466   DD0:                                    ;;; Fake stack causes RETurn to here
0082'  BB 3030          467           MOV     AX,'00'                 ;;; Leading zeros to the screen
0085'  EB 22            468           JMP     SHORT PRCWRD            ;;;    and to the test I/F (fake call)
0087'                   469   DD1:                                    ;;; Fake stack causes RETurn to here
0087'  BB 3030          470           MOV     AX,'00'                 ;;;
008A'  00 F4            471           ADD     AH,DH                   ;;; Convert binary in AL to ASCII in AX
008C'  EB 1B            472           JMP     SHORT PRCWRD            ;;; Data to screen & test I/F (call)
008E'                   473   DD2:                                    ;;; Fake stack causes RETurn to here
008E'  B0 0D            474           MOV     AL,CR                   ;;; Carriage return
0090'  8A E2            475           MOV     AH,DL                   ;;;    and the LED port code
0092'  EB 16            476           JMP     SHORT PRTWRD            ;;; To the test I/F (fake call)
0094'                   477   DD3:                                    ;;; Fake stack causes RETurn to here
0094'  BB 0A0A          478           MOV     AX,0A0AH                ;;; Two line feeds
0097'  EB 11            479           JMP     SHORT PRTWRD            ;;; To the test I/F (fake call)
0099'                   480   DD4:                                    ;;; Fake stack causes RETurn to here
0099'  8A C2            481           MOV     AL,DL                   ;;;
009B'  E6 02            482           OUT     PRDA_P,AL               ;;; Always send LED code to port
009D'  E9 0122          483           JMP     PUPBEL                  ;;; Beep the bell (fake call)
00A0'                   484   DD5:                                    ;;; Fake stack causes RETurn to here
00A0'  F4               485           HLT                             ;;; *** DIE TOTALLY ***
00A1'  EB F6            486           JMP     DD4                     ;;; Shoot him again to make sure
                        487   ;*****************************************************************************
                        488   ;   PRTWRD - Send the word in AX out the parallel port if the factory test
                        489   ;       interface is connected (denoted by PAPER OUT being true when NOT BUSY).
                        490   ;       Note that the data is strobed using the AUTOFEED signal rather than
                        491   ;       the normal STROBE signal.
                        492   ;   PRCWRD - Alternate entry point which writes the word to memory at ES:DI
                        493   ;       (typically screen RAM) before sending it to the parallel port.
                        494   ;
                        495   ;   INPUT:  AL = 1st character to be sent to parallel port
                        496   ;           AH = 2nd character to be sent to parallel port
                        497   ;           CH = PRCO_M (or what should be in PRCO_M)
                        498   ;           (Assumes interrupts are disabled)
                        499   ;   OUTPUT: Two bytes out the parallel port if test I/F is there
                        500   ;   USED:   AX,BX,CX
                        501   ;   STACK:  2 bytes
                        502       ASSUME  DS:ROMDAT
                        503   ;-----------------------------------------------------------------------------
00A3'                   504   PRTWDX  PROC    NEAR
00A3'  8A 2E 000B"      505           MOV     CH,BYTE PTR PRCO_M      ; printer control latch
00A7'  EB 01            506           JMP     SHORT PRTWRD            ; Do print word operation
                        507   ;
00A9'                   508   PRCWRD  PROC    NEAR
00A9'  AB               509           STOS    WORD PTR [DI]           ; Write the word to screen memory
                        510   ;
00AA'                   511   PRTWRD  PROC    NEAR
00AA'  8B D8            512           MOV     BX,AX                   ; Save the characters temporarily
00AC'  E4 01            513           IN      AL,PRCI_P               ; Look at parallel port status latch
00AE'  24 30            514           AND     AL,PRPAPO + PRBUSY      ; Mask off the Paperout and Busy bits
```

```
00B0'  3C 20                515             CMP     AL,PRPAPO AND (NOT PRBUSY) ; Q: Is the factory connected ?
00B2'  75 1E                516             JNE     PWEXIT                     ;    N: Then blow it off
00B4'  B1 02                517             MOV     CL,02                      ;    Y: Set up loop for 2 characters
00B6'  8A C3                518             MOV     AL,BL                      ; Get first character
00B8'                       519     PW1:
00B8'  E6 02                520             OUT     PRDA_P,AL                  ; Put it out to the data latch
00BA'  8A C5                521             MOV     AL,CH                      ; Get copy of control latch from memory
00BC'  24 EF                522             AND     AL,NOT PRAUTO              ; (Use AUTOFEED signal as data strobe)
00BE'  E6 03                523             OUT     PRCO_P,AL                  ; Set the AUTOFEED signal low
00C0'  90                   524             NOP                                ; Delay
00C1'  34 10                525             XOR     AL,PRAUTO                  ;
00C3'  E6 03                526             OUT     PRCO_P,AL                  ; Bring it back high
00C5'  90                   527             NOP                                ; Delay for BUSY on test interface
00C6'                       528     PW2:
00C6'  E4 01                529             IN      AL,PRCI_P                  ; Look at status latch
00C8'  A8 10                530             TEST    AL,PRBUSY                  ; Q: Still BUSY ?
00CA'  75 FA                531             JNZ     PW2                        ;    Y: Loop til not busy
00CC'  8A C7                532             MOV     AL,BH                      ;    N: Send next character
00CE'  FE C9                533             DEC     CL                         ; SENT ALL CHARS?
00D0'  75 E6                534             JNZ     PW1                        ; (if I haven't already)
00D2'                       535     PWEXIT:
00D2'  C3                   536             RET                                ; *** RETURN ***
                            537     ;*************************************************************************
                            538     ;
                            539     ;   DSPERR - System error display subroutine
                            540     ;   DSPKER - Keyboard error display subroutine
                            541     ;       These subroutines perform a function similar to DSPDIE, except
                            542     ;       that they are callable NEAR subroutines, returning rather than
                            543     ;       'dying'.  Also, they use the CRT DSR to do the screen output.
                            544     ;       DSPERR outputs the message '** System Error **  - 00xx'
                            545     ;
                            546     ;   INPUT:  AL = 2-digit error code to be displayed (DSPERR adds leading 00)
                            547     ;  (DSPERX) AX = 4-digit error code to be displayed
                            548     ;   OUTPUT: (error message to screen/parallel port, beep)
                            549     ;   USED:   AX,BX,CX
                            550     ;   STACK:
                            551         ASSUME  CS:ROMCOD
                            552     ;-------------------------------------------------------------------------
00D3'                       553     DSPERZ  PROC    NEAR
00D3'  8B C3                554             MOV     AX,BX                      ; Put error code into AX.
00D5'  EB 0A                555             JMP     SHORT DSPERX               ; and continue
                            556     ;
00D7'                       557     DSPKER  PROC    NEAR
00D7'  30 E4                558             XOR     AH,AH                      ; Set up leading zero of message
00D9'  56                   559             PUSH    SI
00DA'  BE 0000"             560             MOV     SI,OFFSET KERMSG           ; Point to Keyboard error message
00DD'  EB 06                561             JMP     SHORT DSPER1               ; Jump to rest of code
                            562     ;
00DF'                       563     DSPERR  PROC    NEAR
00DF'  30 E4                564             XOR     AH,AH                      ; Set up leading zero of message
00E1'  56                   565     DSPERX: PUSH    SI
00E2'  BE 001A"             566             MOV     SI,OFFSET SER1MS           ; Point to System error message
```

```
00E5'                           567   DSPER1:
00E5'  9C                       568           PUSHF                           ; Save flags.
00E6'  1E                       569           PUSH    DS                      ; SAVE current DS.
00E7'  2E8E 1E C180             570           MOV     DS,CS:WORD PTR DSADDR+CSWRAP
00EC'  50                       571           PUSH    AX                      ; Save the error code
00ED'  E8 00BF                  572           CALL    MSG                     ; Put it to the screen
00F0'  B8 202D                  573           MOV     AX,' -'                 ; Reversed to '- '
00F3'  E8 00F9                  574           CALL    OUT2CH                  ; Send to screen and test I/F
00F6'  58                       575           POP     AX                      ; Get leading error code
00F7'  50                       576           PUSH    AX
00F8'  86 E0                    577           XCHG    AH,AL                   ; SWAP
00FA'  E8 00EF                  578           CALL    OUTBYT                  ; Convert it and display it
00FD'  58                       579           POP     AX                      ; Get the error code back
00FE'  E8 00EB                  580           CALL    OUTBYT                  ; Convert it and display it
0101'  B8 0A0D                  581           MOV     AX,0A0DH                ; Carriage return
0104'  E8 FF9C                  582           CALL    PRTWDX                  ; Send to test I/F (only)
0107'  E8 00B8                  583           CALL    PUPBEL                  ; Make 'em notice
010A'  1F                       584           POP     DS                      ; Restore state
010B'  9D                       585           POPF
010C'  5E                       586           POP     SI
010D'  C3                       587           RET                             ; *** RETURN ***
                                588   ;*************************************************************************
                                589   ;       NMI interrupt entry point for normal processing
                                590   ;
                                591   ;       Checks for parity interrupt and shuts down and reports error if so,
                                592   ;       reports unexpected NMI otherwise.   If anyone ever uses NMI for
                                593   ;       anything else, they should intercept the vector to the beginning of
                                594   ;       the routine.
                                595   ;
                                596   ;       STACK:  Who cares?
                                597   ;-------------------------------------------------------------------------
010E'  20 20 49 67 6E 6F        598   PARMSG  DB      '  Ignore, Reboot?',00
0114'  72 65 2C 20 52 65
011A'  62 6F 6F 74 3F 00
                                599   ;
0120'                           600   NMIQQ   PROC    FAR
0120'  50                       601           PUSH    AX                      ; Save registers
0121'  B0 26                    602           MOV     AL,UC_DTR+UC_RXE+UC_RTS ; Force interrupt on permanently
0123'  E6 11                    603           OUT     KBCMD,AL                ;
0125'  E4 01                    604           IN      AL,PRCI_P               ; Get parity interrupt status.
0127'  24 08                    605           AND     AL,PARINT               ; Q: Parity interrupt?
0129'  B0 09*                   606           MOV     AL,XNMERR               ;       Set up unexpected NMI error code
012B'  74 07                    607           JZ      NMIQ4                   ;   N: Jump and exit.
012D'  B0 07*                   608           MOV     AL,RMPERR               ;   Y: Display the RAM parity error.
012F'  2EFF 0E C198             609           DEC WORD PTR CS:MEMSIZ+CSWRAP   ; require full powerup on warm reset.
0134'  53                       610   NMIQ4:  PUSH    BX                      ; Save the user registers
0135'  51                       611           PUSH    CX
0136'  52                       612           PUSH    DX
0137'  56                       613           PUSH    SI
0138'  B4 10                    614           MOV     AH,10H                  ; 104X
013A'  E8 FFA4                  615           CALL    DSPERX                  ; Display the error message.
013D'  BE 010E*                 616           MOV     SI,OFFSET PARMSG        ; Parity error message.
```

```
0140'  E8 006C              617             CALL    MSG                     ; Display prompt
0143'  30 E4                618     NMIQ6:  XOR     AH,AH                   ; Get a char from keyboard.
0145'  CD 4A                619             INT     KEYINT                  ;
0147'  24 DF                620             AND     AL,0DFH                 ; Convert to uppercase
0149'  3C 52                621             CMP     AL,'R'                  ; Q: Reboot?
014B'  75 05                622             JNZ     NMIQ8                   ;   N: check for others
014D'  EA 0006" 0000"       623             JMP     PWRUP                   ;   Y: Reset the system
0152'  3C 49                624     NMIQ8:  CMP     AL,'I'                  ; Q: Ignore?
0154'  75 ED                625             JNZ     NMIQ6                   ;   N: Wait for another char.
0156'  5E                   626             POP     SI                      ;
0157'  5A                   627             POP     DX                      ;   Y: Return to user program
0158'  59                   628             POP     CX
0159'  5B                   629             POP     BX
015A'  58                   630             POP     AX
015B'  CF                   631             IRET
                            632     ;
                            633     ;
                            634     ;*****************************************************************************
                            635     ;       NMI interrupt entry point for powerup
                            636     ;
                            637     ;       Checks for parity interrupt and shuts down and reports error if so,
                            638     ;       reports unexpected NMI otherwise.   If anyone ever uses NMI for
                            639     ;       anything else, they should intercept the vector to the beginning of
                            640     ;       the routine.
                            641     ;
                            642     ;       STACK:  Who cares?
                            643     ;-----------------------------------------------------------------------------
015C'                       644     PUPNMI  PROC    FAR
015C'  B0 26                645             MOV     AL,UC_DTR+UC_RXE+UC_RTS ; Force interrupt on permanently
015E'  E6 11                646             OUT     KBCMD,AL                ;
0160'  E4 01                647             IN      AL,PRCI_P               ; Get parity interrupt status.
0162'  24 08                648             AND     AL,PARINT               ; Q:  Parity interrupt ?
0164'  B0 04*               649             MOV     AL,LED100               ;   Y: Set up unexpected NMI error code
0166'  B2 03*               650             MOV     DL,IVIERR               ;      And LED state.
0168'  74 04                651             JZ      PUPNM4                  ;   N: Jump and exit.
016A'  B0 05*               652             MOV     AL,LED101               ; Display the RAM parity error.
016C'  30 D2                653             XOR     DL,DL                   ; Zero to printer port
016E'  E9 FEDD              654     PUPNM4: JMP     DSPDIE                  ; Quit.
                            655     ;*****************************************************************************
                            656     ;   WILD$$ - Illegal (soft or hard) interrupt exit
                            657     ;       This exit is taken if an interrupt occurs from any of the unused
                            658     ;       interrupt vectors.  The cause could be either a software interrupt
                            659     ;       or an errant hardware interrupt.  This routine puts a fixed error
                            660     ;       code into AL and jumps to ARGHHH.
                            661     ;   INPUT:  (none)
                            662     ;   OUTPUT: Error message to the screen (fixed error code), PC, beep
                            663     ;   USED:
                            664     ;   STACK:
                            665             ASSUME  CS:ROMCOD
                            666     ;-----------------------------------------------------------------------------
0171'                       667     WILD$$  PROC    FAR
0171'  B0 08*               668             MOV     AL,WLDERR               ; Set up 'Wild' interrupt error code
```

```
0173' EB 02                669             JMP     SHORT ARGNER            ; Exit the standard way
                           670     ;
                           671     ;
                           672     ;*****************************************************************************
                           673     ;   FATAL - Fatal error exit (software buggie)
                           674     ;       This exit is taken as the last resort by any software routine
                           675     ;       which has given up on the normal error-recovery procedures.  It
                           676     ;       puts a fixed error code into AL and jumps to ARGHHH.  Typical user
                           677     ;       would be ZEX.
                           678     ;   INPUT:  (none)
                           679     ;   OUTPUT: Error message to the screen (fixed error code), PC, beep
                           680     ;   USED:
                           681     ;   STACK:
                           682         ASSUME  CS:ROMCOD
                           683     ;-----------------------------------------------------------------------------
0175'                      684     FATAL   PROC    FAR
0175' B0 02*               685             MOV     AL,FATERR               ; Set up fatal error code
                           686     ;       JMP     SHORT ARGNER            ; Do the standard thing
                           687     ;*****************************************************************************
                           688     ;   ARGHHH - (Just like it sounds) Display error message and wait.
                           689     ;       This routine is used by anyone that wants to halt with an
                           690     ;       error message but still be alive enough to restart from the
                           691     ;       keyboard.  This should be used for all cases except during
                           692     ;       powerup.  In addition to the standard 'SYSTEM ERROR' message
                           693     ;       and beep, the  calling CS:IP is displayed as a debugging aid.
                           694     ;
                           695     ;   INPUT:  AL = 2-digit error code to be displayed (ARGHHH adds leading 10)
                           696     ;   OUTPUT: Error message to the screen, PC, beep
                           697     ;   USED:
                           698     ;   STACK:
                           699         ASSUME  CS:ROMCOD
                           700     ;-----------------------------------------------------------------------------
0177'                      701     ARGNER  PROC    NEAR                    ; Dummy def for assembler
0177'                      702     ARGHHH  PROC    FAR
0177' B4 10                703             MOV     AH,10H                  ; Set high byte of error to 10
0179' B2 0A                704             MOV     DL,LF                   ;
017B' E8 FF63              705             CALL    DSPERX                  ; Print error
017E' BB 4020              706             MOV     AX,'@ '                 ; Delimiter (' @')
0181' E8 0068              707             CALL    OUT2CH
0184' 8B EC                708             MOV     BP,SP                   ; Set up BP
0186' 8B 46 02             709             MOV     AX,2[BP]                ; Get user's CS
0189' E8 0059              710             CALL    OUTWRD                  ; Display it
018C' B0 3A                711             MOV     AL,':'                  ; Delimiter
018E' B4 0E                712             MOV     AH,CRTWTY               ; Use TTY screen
0190' E8 0001*             713             CALL    CRTOUT                  ; Output it (to screen only)
0193' 8B 46 00             714             MOV     AX,[BP]                 ; Get user's IP
0196' E8 004C              715             CALL    OUTWRD                  ; Display it
0199' B8 0A0D              716             MOV     AX,CR + (LF SHL 8)      ; CRLF
019C' E8 0050              717             CALL    OUT2CH
019F' B0 7F                718             MOV     AL,(ENAB7)              ; Leave only keyboard enabled
01A1' E6 19                719             OUT     INTA01,AL               ; Set mask register
01A3' B0 C6                720             MOV     AL,11000110B            ; Readjust priority of controler
```

```
01A5'  E6 18                721             OUT     INTA00,AL               ; So keyboard is highest priority
01A7'  B0 0F                722             MOV     AL,0FH                  ; Turn off floppy motors
01A9'  E6 04                723             OUT     DSKS_P,AL
01AB'  FB                   724             STI                             ; Turn interrupts back on
01AC'                       725     ARHALT:
01AC'  F4                   726             HLT                             ; Wait.....
01AD'  EB FD                727             JMP     SHORT ARHALT            ; ...and wait
                            728     ;--------------------------------------------------------------------------
                            729     ;   MSG - Output string of characters in the current CS to the CRT.
                            730     ;       The string should be terminated with a zero byte.
                            731     ;   INPUT:  SI = string offset in current CS
                            732     ;   OUTPUT: (screen)
                            733     ;   USED:   AX,SI
                            734     ;   STACK:
                            735         ASSUME  CS:ROMCOD, DS:NOTHING
                            736     ;--------------------------------------------------------------------------
01AF'                       737     MSG     PROC    NEAR
01AF'  51                   738             PUSH    CX
01B0'  52                   739             PUSH    DX
01B1'  FC                   740             CLD                             ; forward strings
01B2'                       741     MSG0:
01B2'  B4 0E                742             MOV     AH,CRTWTY               ; Set CRT for TTY output
01B4'  2EAC                 743             LODS    CS:BYTE PTR [SI]        ; Pickup byte
01B6'  08 C0                744             OR      AL,AL                   ; Check for end
01B8'  74 05                745             JZ      MSG1
01BA'  E8 0001"             746             CALL    CRTOUT                  ; Print char
01BD'  EB F3                747             JMP     SHORT MSG0
01BF'                       748     MSG1:
01BF'  5A                   749             POP     DX
01C0'  59                   750             POP     CX
01C1'  C3                   751             RET                             ; *** RETURN ***
                            752     ;--------------------------------------------------------------------------
                            753     ;   PUPBEL - This routine is used to sound the system beeper without
                            754     ;       using timing services or any stack other than that used to call
                            755     ;       this subroutine (which can be faked).
                            756     ;
                            757     ;       NOTE THAT INTERRUPTS ARE DISABLED IN THIS ROUTINE FOR THE DURATION
                            758     ;       OF THE 1 SECOND BEEP DELAY AND NOT REENABLED.  THIS IS NOT
                            759     ;       INTENDED TO BE A GENERAL PURPOSE ROUTINE.
                            760     ;
                            761     ;   OUTPUT: (*** BEEP ***)
                            762     ;   USED:   AL
                            763     ;   STACK:  2 bytes (usually faked)
                            764         ASSUME  CS:ROMCOD, DS:NOTHING
                            765     ;--------------------------------------------------------------------------
01C2'                       766     PUPBEL  PROC    NEAR
01C2'  FA                   767             CLI                             ; Protect this
01C3'  B0 36                768             MOV     AL,(TC_SCO + TC_WRD + TC_MD3) ;;; Set the timer
01C5'  E6 17                769             OUT     TIMCMD,AL               ;;;
01C7'  B0 1B                770             MOV     AL,LOW BELTMR           ;;; 1.25 MHz / BELTMR = frequency
01C9'  E6 14                771             OUT     TIMER0,AL               ;;;
01CB'  B0 06                772             MOV     AL,HIGH BELTMR          ;;;
```

```
01CD'  E6 14              773          OUT     TIMER0,AL               ;;;
01CF'  A0 000A"           774          MOV     AL,DSKC_M               ;;;
01D2'  0C 01              775          OR      AL,SPKREN               ;;; Turn the speaker on
01D4'  E6 00              776          OUT     DSKC_P,AL               ;;;
01D6'  31 C9              777          XOR     CX,CX                   ;;; Clear CX
01D8'  E2 FE              778          LOOP    $                       ;;; 223 msec +
01DA'  E2 FE              779          LOOP    $                       ;;; 223 msec +
01DC'  E2 FE              780          LOOP    $                       ;;; 223 msec +
01DE'  E2 FE              781          LOOP    $                       ;;; 223 msec = approx. 1 second
01E0'  34 01              782          XOR     AL,SPKREN               ;;; Flip the enable bit
01E2'  E6 00              783          OUT     DSKC_P,AL               ;;; Turn the speaker off
01E4'  C3                 784          RET                             ;;; *** RETURN ***
                          785  ;*****************************************************************************
                          786  ;   OUTWRD/OUTBYT - output word(byte) in AX(AL) to the screen in hex
                          787  ;
                          788  ;   INPUT:   AX/AL = number
                          789  ;   OUTPUT:  (display)
                          790  ;   USED:    AX,BX
                          791  ;   STACK:              (LOTS)
                          792      ASSUME  CS:ROMCOD
                          793  ;-----------------------------------------------------------------------------
01E5'                     794  OUTWRD  PROC    NEAR
01E5'  50                 795          PUSH    AX                      ; Work on left byte first
01E6'  8A C4              796          MOV     AL,AH                   ; Copy to AL
01E8'  E8 0001            797          CALL    OUTBYT                  ; Convert & output
01EB'  58                 798          POP     AX                      ; Now do right byte
                          799  ;
01EC'                     800  OUTBYT  PROC    NEAR
01EC'  E8 0018            801          CALL    CNVBYT                  ; Convert byte to hex-ASCII
                          802  ;       JMP     OUT2CH                  ; Display result
                          803  ;-----------------------------------------------------------------------------
                          804  ;   OUT2CH - Output two chars to console, and factory test interface,
                          805  ;       if connected.
                          806  ;
                          807  ;   INPUT:   AL = 1st char to be output
                          808  ;            AH = 2nd char to be output
                          809  ;   OUTPUT:  (display & parallel port)
                          810  ;   USED:    AX,BX
                          811  ;   STACK:
                          812      ASSUME  CS:ROMCOD
                          813  ;-----------------------------------------------------------------------------
01EF'                     814  OUT2CH  PROC    NEAR
01EF'  51                 815          PUSH    CX
01F0'  52                 816          PUSH    DX
01F1'  8B D8              817          MOV     BX,AX                   ; Save the characters
01F3'  B4 0E              818          MOV     AH,CRTWTY               ; Set up for TTY output
01F5'  E8 0001"           819          CALL    CRTOUT                  ; Output char in AL as usual
01F8'  8A C7              820          MOV     AL,BH                   ; Now do char in AH
01FA'  B4 0E              821          MOV     AH,CRTWTY
01FC'  E8 0001"           822          CALL    CRTOUT
01FF'  8B C3              823          MOV     AX,BX                   ; Restore AX
0201'  E8 FE9F            824          CALL    PRTWDX                  ;    and send it to the test I/F
```

```
0204'  5A                825          POP     DX
0205'  59                826          POP     CX
0206'  C3                827          RET                             ; *** RETURN ***
                         828  ;
                         829  ;--------------------------------------------------------------------------
                         830  ;   CNVBYT - Convert binary byte to Hex-ASCII
                         831  ;
                         832  ;   INPUT:   AL = byte to be converted
                         833  ;   OUTPUT:  AL = ASCII representation for high-order nibble
                         834  ;            AH = ASCII representation for low-order nibble
                         835  ;   USED:    AX,BX
                         836  ;   STACK:   4
                         837      ASSUME  CS:ROMCOD
                         838  ;--------------------------------------------------------------------------
0207'                    839  CNVBYT  PROC    NEAR
0207'  51                840          PUSH    CX                      ; Must save CX
0208'  BB 0032"          841          MOV     BX,OFFSET BINASC        ; Address of lookup table
020B'  8A C8             842          MOV     CL,AL                   ; Save other nibble
                         843  ;
                         844  ;   Right nibble first
                         845  ;
020D'  24 0F             846          AND     AL,0FH                  ; Extract (3:0)
020F'  2ED7              847          XLAT    BYTE PTR CS:[BX]        ; Translate AL (-- [BX+AL]
0211'  8A E0             848          MOV     AH,AL                   ; Save result
                         849  ;
                         850  ;   Then left nibble
                         851  ;
0213'  8A C1             852          MOV     AL,CL                   ; Retrieve other nibble
0215'  B1 04             853          MOV     CL,4                    ; Prepare for 4 shifts
0217'  D2 E8             854          SHR     AL,CL                   ; Extract (7:4)
0219'  2ED7              855          XLAT    BYTE PTR CS:[BX]        ; translate:    AL (-- [BX+AL]
021B'  59                856          POP     CX                      ; restore
021C'  C3                857          RET                             ; *** RETURN ***
                         858  ;
                         859          END

No errors detected
```

```
                        1   ;*****************************************************************************
                        2   ;  TITLE    - PRTDSR - Printer DSR
                        3   ;  COMPUTER - 8088 Assembly Language
                        4   ;  ABSTRACT - This module contains the interface logic for the parallel
                        5   ;       printer port.
                        6   ;*****************************************************************************
                        7           NAME    PRTDSR - Printer interface routines             -
                        8   ;       SUBTTL
                       10   ;*****************************************************************************
                       11   ;                           PUBLIC DEFINITIONS
                       12   ;*****************************************************************************
                       13   ;
                       14           PUBLIC  PRINT
                       15           PUBLIC  PRT_IO
                       16           PUBLIC  PRTINI
                       17   ;
                       18   ;*****************************************************************************
                       19   ;                           EXTERNAL REFERENCES
                       20   ;*****************************************************************************
                       21   ;
=0000                  22           SECT    ROMDAT
                       23           EXTRN   PRCO_M:BYTE     ; Printer control latch memory          (ROMDAT)
                       24   ;
                       25   ;*****************************************************************************
                       26   ;                           LOCAL CONSTANTS
                       27   ;*****************************************************************************
                       28   ;
                       29   ;       INCLUDE PEQ:PORTADDR.EQU
                       30   ;       INCLUDE PEQ:LATCHES.EQU
                       31   ;       INCLUDE PEQ:VECTOR.EQU
                       32
                      187
=00F0                 188   PRMASK  EQU     (PRBUSY+PRPAPO+PRSLCT+PRFAUL)   ; Make a printer mask
                      189
                      190   ;*****************************************************************************
                      191   ;                           CODE SEGMENT DEFINITION
                      192   ;*****************************************************************************
                      193   ;
=0000                 194           SECT    ROMCOD
                      195           ASSUME  CS:ROMCOD, DS:ROMDAT
                      196   ;
                      197   ;*****************************************************************************
                      198   ;                           MODULE ENTRY POINT
                      199   ;*****************************************************************************
                      200   ;
                      201   ;   PARALLEL PRINT PORT 'DSR' - INTerrupt 4BH
                      202   ;
                      203   ;   INPUT:  AH = function: (see the file PEQ:PRTOP.EQU)
                      204   ;                   0 - output char to printer
                      205   ;                   1 - initialize the printer
                      206   ;                   2 - return printer status
                      207   ;           AL = character to be printed (function 0 only)
```

```
                        208   ;   OUTPUT: AH = Status of printer as follows:
                        209   ;
                        210   ;                Bit: 7 - 1 = FAULT
                        211   ;                     6 - 1 = SELECTED
                        212   ;                     5 - 1 = PAPER OUT
                        213   ;                     4 - 1 = BUSY
                        214   ;                     0 - 1 = PRINTER TIME OUT
                        215   ;
                        216   ;           AL = character (unchanged)
                        217   ;   USED:   AX
                        218   ;   STACK:  12 bytes + what the function uses
                        219   ;-----------------------------------------------------------------------------
0000'                   220   PRT_IO  PROC    FAR
0000'  FB               221           STI                     ;;; Interrupts back on
0001'  1E               222           PUSH    DS              ; Save current DS
0002'  51               223           PUSH    CX
0003'  53               224           PUSH    BX
0004'  2E8E 1E C180     225           MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; Set my DS
0009'  E8 0004          226           CALL    DO_PRT          ; Do the function
000C'  5B               227           POP     BX
000D'  59               228           POP     CX
000E'  1F               229           POP     DS
000F'  CF               230           IRET                    ; *** RETURN ***
                        231   ;
                        232   ;
0010'                   233   DO_PRT  PROC    NEAR            ; dispatch to function specified by AH
0010'  08 E4            234           OR      AH,AH           ; AH = 0
0012'  74 2A            235           JZ      PRINT
0014'  FE CC            236           DEC     AH              ; AH = 1
0016'  74 05            237           JZ      PRTINI
0018'  FE CC            238           DEC     AH              ; AH = 2
001A'  74 15            239           JZ      PRSTAT
001C'  C3               240           RET                     ; *** RETURN *** (invalid command)
                        241
                        242   ;*****************************************************************************
                        243   ;   PRINTER INITIALIZATION ROUTINE
                        244   ;
                        245   ;   INPUT:  DS points to ROMDAT
                        246   ;   OUTPUT: AH = printer status (see above)
                        247   ;   USED:   AX,CX
                        248   ;   STACK:  4 bytes
                        249   ;-----------------------------------------------------------------------------
001D'                   250   PRTINI  PROC    NEAR
001D'  FA               251           CLI                     ; Protect this
001E'  A0 0001"         252           MOV     AL,PRCO_M       ;;; Get copy of latch from RAM
0021'  24 BF            253           AND     AL,NOT PRINIT   ;;;
0023'  E6 03            254           OUT     PRCO_P,AL       ;;; Diddle the INIT line low
0025'  B9 0096          255           MOV     CX,150          ;;; Delay value for about 500 usec
0028'                   256   PRI1:
0028'  E2 FE            257           LOOP    PRI1            ;;; Delay for INIT to take effect
002A'  0C 40            258           OR      AL,PRINIT       ;;;
002C'  E6 03            259           OUT     PRCO_P,AL       ;;; Flip INIT back high
```

```
002E' FB                260         STI                     ;;; Restore interrupt status
002F' B4 00             261         MOV     AH,00           ; Set 'no error status'
                        262 ;       JMP     SHORT PRSTAT    ; Return with printer status
                        263 ;
                        264 ;---------------------------------------------------------------------------
                        265 ;   PRINTER STATUS ROUTINE
                        266 ;
                        267 ;   INPUT:  AH = bit zero has timeout status
                        268 ;           (Note that AH=0 when STATUS called as a function thru PRT_IO)
                        269 ;   OUTPUT: AH = status (see above)
                        270 ;   USED:   AH,CX
                        271 ;   STACK:  4
                        272 ;---------------------------------------------------------------------------
0031'                   273 PRSTAT  PROC    NEAR
0031' 50                274         PUSH    AX              ; Save character
0032' E4 01             275         IN      AL,PRCI_P       ; Read printer status port
0034' 24 F0             276         AND     AL,PRMASK       ; Mask off the non-printer stuff
0036' 34 80             277         XOR     AL,PRFAUL       ; Flip the (low true) FAULT bit
0038' 08 C4             278         OR      AH,AL           ; OR AL (real status) into AH (has timeout)
003A' 59                279         POP     CX
003B' 8A C1             280         MOV     AL,CL           ; Restore the character
003D' C3                281         RET                     ; *** RETURN ***
                        282
                        283 ;---------------------------------------------------------------------------
                        284 ;   PRINTER OUTPUT ROUTINE - Contains factory interface logic.  If PAPER OUT
                        285 ;       remains active and BUSY is inactive, then the data is strobed with
                        286 ;       the AUTOFEED line instead of the normal STROBE signal.
                        287 ;
                        288 ;   INPUT:  AL = character to be printed
                        289 ;   OUTPUT: AL = the character (unchanged)
                        290 ;           AH = printer status (see above)
                        291 ;   USED:   AH,BL,CX
                        292 ;   STACK:  4 bytes
                        293 ;---------------------------------------------------------------------------
003E'                   294 PRINT   PROC    NEAR
003E' 8A D8             295         MOV     BL,AL           ; Save character
0040' B4 00             296         MOV     AH,00           ; Reset status initially
0042' B9 C350           297         MOV     CX,50000        ; Delay count for about 1/3 second
0045' E6 02             298         OUT     PRDA_P,AL       ; Put printer data out to latch
0047'                   299 PR1:
0047' E4 01             300         IN      AL,PRCI_P       ; Read printer status latch             10
0049' A8 10             301         TEST    AL,PRBUSY       ; Q: Printer busy ?                      4
004B' 74 06             302         JZ      PR2             ;   N: Continue                          4
004D' E2 F8             303         LOOP    PR1             ;   Y: Loop                             17
004F' B4 01             304         MOV     AH,1            ; TIMEOUT: Set timeout bit              ----
0051' EB 20             305         JMP     SHORT PRXIT     ; and exit                              35
0053'                   306 PR2:                            ;                                  (= 7 usec)
0053' B9 000A           307         MOV     CX,10           ; do a short wait before writing data
0056' E2 FE             308         LOOP    $               ;
                        309 ;       IN      AL,PRCI_P       ; Look at printer status latch (already in AL)
0058' A8 20             310         TEST    AL,PRPAPO       ; Q: Paper Out ? (Paperout AND Not Busy is
                        311                                 ;    indication of factory connection)
```

```
005A' FA                312             CLI                     ;    (Protect this)
005B' A0 0001"          313             MOV     AL,PRCO_M       ;;; Get copy of latch from memory
005E' 74 09             314             JZ      PR3             ;;; N: Use normal interface
                        315                                     ;;; Y: Use factory interface
0060' 24 EF             316             AND     AL,NOT PRAUTO   ;;; (Use the Autofeed signal as data strobe)
0062' E6 03             317             OUT     PRCO_P,AL       ;;; Set the autofeed signal low
0064' 90                318             NOP                     ;;; Delay
0065' 34 10             319             XOR     AL,PRAUTO       ;;;
0067' EB 07             320             JMP     SHORT PR4       ;;; Finish up like normal
0069'                   321     PR3:
0069' 24 DF             322             AND     AL,NOT PRSTRB   ;;;
006B' E6 03             323             OUT     PRCO_P,AL       ;;; Set the strobe low
006D' 90                324             NOP                     ;;; Wait a bit
006E' 34 20             325             XOR     AL,PRSTRB       ;;;
0070'                   326     PR4:
0070' E6 03             327             OUT     PRCO_P,AL       ;;; Bring it back high
0072' FB                328             STI                     ;;;
0073'                   329     PRXIT:
0073' 8A C3             330             MOV     AL,BL           ; Restore saved character
0075' EB BA             331             JMP     SHORT PRSTAT    ; Go get printer status
                        332     ;
                        333             END
```

No errors detected

```
                         1  ;*****************************************************************************
                         2  ; TITLE    - PUPTS1 - PEGASUS POWERUP TEST CODE, OPTION RAM AND ROMS TEST.
                         3  ; ABSTRACT - THE TEST IS DESIGNED TO CHECK OUT THE OPTION ROM AND RAM.
                         4  ;        ANY ADDITIONAL ROMS ARE CALLED IF THEY TEST OUT OK TO ALLOW
                         5  ;        INITIALIZATION OR BOOTING OF OTHER DEVICES IN THE SYSTEM.
                         6  ;
                         7  ; REGISTERS USED - ALL
                         8  ;
                         9  ; STACK USED - 4 BYTES UNLESS THERE IS A ROM ERROR, THEN "DSPERR"+2.
                        10  ;
                        11  ; * Register is saved during call to DSPERR to fix crash after bad option ROM
                        12  ;   CRC.
                        13  ; * EXTERNs are put into SECT sections to correct bug causing incorrect value
                        14  ;   of PRCO_M to be linked which caused unexpected printer reset.
                        15  ; * powerup RAM test changed to allow 786432 bytes of main memory (originaly
                        16  ;        524288 bytes) also fixes bugs concerning large amounts of main RAM
                        17  ;*****************************************************************************
                        18          NAME    PUPTS1
                        19          SUBTTL  PEGASUS POWERUP DIAGNOSTIC TESTS
                        21  ;*****************************************************************************
                        22  ;                           PUBLIC DEFINITIONS
                        23  ;*****************************************************************************
                        24          PUBLIC  PUPTS1          ;MODULE ENTRY POINT
                        25  ;*****************************************************************************
                        26  ;                           EXTERNAL REFERENCES
                        27  ;*****************************************************************************
                        28          EXTERN  OPRERR:ABS      ;OPTION RAM FAIL DURING PUPTST
                        29          EXTERN  RMPERR:ABS      ;RAM PARITY ERROR
                        30          EXTERN  ROMERX:ABS      ;ROM ERRORS ERROR CODE
                        31          EXTERN  ROMSIZ:ABS      ;SIZE OF ROM SPACE.
                        32          EXTERN  XNMERR:ABS      ;UNEXPECTED NMI
=0000                   33          SECT    ROMCOD
                        34          ASSUME CS:ROMCOD
                        35          EXTERN  ARGHHH:NEAR     ;ADDRESS OF FAILURE ROUTINE
                        36          EXTERN  DECLED:NEAR     ;ADDRESS OF DECREMENT LED ROUTINE
                        37          EXTERN  DKBOOT:NEAR     ;ADDRESS OF POWERUP BOOT ROUTINE
                        38          EXTERN  DSPERR:NEAR     ;ADDRESS OF FAILURE ROUTINE
                        39          EXTERN  NMIGO:FAR       ;NMI INTERRUPT ENTRY
                        40          EXTERN  ROMTST:NEAR     ;ADDRESS ROM TEST SUBROUTINE
                        41          EXTERN  SYSROM:BYTE     ;ADDRESS OF SYSTEM ROM
=0000                   42          SECT    ROMDAT
                        43          EXTERN  PRCO_M:BYTE     ;ADDRESS OF LATCH SAVE BYTE
                        44  ;*****************************************************************************
                        45  ;                           LOCAL CONSTANTS
                        46  ;*****************************************************************************
                        47  ;       INCLUDE PEG:PORTADDR.EQU
                        48  ;       INCLUDE PEG:LATCHES.EQU
                        49  ;       INCLUDE PEG:VECTOR.EQU
                        50
                       205
=0008                  206  NMIVEC  EQU     NMIINT*4        ;NMI INTERRUPT VECTOR LOCATION
                       207  ;*****************************************************************************
```

```
                                208  ;                          MODULE ENTRY POINT
                                209  ;*****************************************************************************
          =0000                 210          SECT    ROMCOD                  ;ALL CODE IS PLACE HERE
                                211          ASSUME  CS:ROMCOD,DS:ABS0,ES:ABS0
                                212  ;
                                213  ;       DO NOTHING IF WARM RESET, ELSE CHECK OPTION RAM.
                                214  ;
0000'                           215  PUPTS1  PROC    NEAR
0000'  2EA1 C198                216          MOV     AX,WORD PTR CS:MEMSIZ+CSWRAP    ;GET SIZE OF RAM.
                                217  ;                                       ;1000H (= RAMSIZ (= C000H IF WARM RESET
0004'  B1 04                    218          MOV     CL,4                    ;ADJUST SIZE SO THAT RANGE IS FROM
0006'  D3 C0                    219          ROL     AX,CL                   ; 1 (= AX (= 0CH
0008'  48                       220          DEC     AX                      ; 0 (= AX (= 0BH
0009'  3D 000C                  221          CMP     AX,000CH                ;Q:  WARM RESET?
000C'  72 75                    222          JB      OPRXIT                  ; Y; EXIT TEST ROUTINE
                                223  ;
                                224  ;       TEST OPTION RAM.  SET UP PARITY VECTOR AND FILL RAM.
                                225  ;
000E'  FA                       226  OPTRM:  CLI                             ;DISABLE INTERRUPTS
000F'  2EC7 06 C008 0052"       227          MOV     WORD PTR CS:NMIVEC+CSWRAP,OFFSET DERROR ;CHANGE PARITY VECTOR.
0016'  BE 1000                  228          MOV     SI,1000H                ;INITIAL OPTION RAM SEGMENT
0019'  8E DE                    229  OPRAM:  MOV     DS,SI                   ;SET UP RAM SEGMENT
001B'  8E C6                    230          MOV     ES,SI
001D'  BD 0002                  231          MOV     BP,000000010B           ;WALKING ONE
0020'  31 FF                    232  OPR0:   XOR     DI,DI                   ;SET UP START OF RAM TO TEST
0022'  8B C5                    233          MOV     AX,BP                   ;GET WALKING ONE TO AX.
0024'  D1 C8                    234          ROR     AX,1                    ;SET UP THE CARRY BIT
0026'  B9 8000                  235          MOV     CX,8000H                ;SIZE OF OPTION RAM
0029'  AB                       236  OPR1:   STOS    WORD PTR[DI]            ;STORE WALKING ONE
002A'  D1 D0                    237          RCL     AX,1                    ;WALK IT
002C'  E2 FB                    238          LOOP    OPR1                    ;DO THE WHOLE SEGMENT.
                                239
                                240  ;
                                241  ;       READ AND CHECK THE OPTION RAM SEGMENT
                                242  ;
002E'  8B DD                    243          MOV     BX,BP                   ;GET INITIAL PATTERN
0030'  D1 CB                    244          ROR     BX,1                    ;SET UP THE CARRY BIT
0032'  9F                       245  OPR2:   LAHF                            ;SAVE FLAGS
0033'  8B D3                    246          MOV     DX,BX                   ;CHECK BYTE
0035'  33 15                    247          XOR     DX,[DI]                 ;IS THIS OK?
0037'  75 19                    248          JNZ     DERROR                  ;N:  JUMP
0039'  9E                       249          SAHF                            ;GET FLAGS
003A'  D1 D3                    250          RCL     BX,1                    ;NEXT PATTERN TO CHECK
003C'  47                       251          INC     DI                      ;BUMP POINTERS
003D'  47                       252          INC     DI                      ;BUMP POINTER
003E'  75 F2                    253          JNZ     OPR2                    ;CONTINUE TILL FINISHED
0040'  F7 D5                    254          NOT     BP                      ;SET UP FOR WALKING ZERO
0042'  21 ED                    255          AND     BP,BP                   ;Q; SECOND LOOP?
0044'  78 DA                    256          JS      OPR0                    ;N; GO FOR NEXT LOOP
0046'  81 C6 1000               257          ADD     SI,1000H                ;BUMP TO NEXT SEGMENT.
004A'  81 FE C000               258          CMP     SI,0C000H               ;Q;  END OF POSSIBLE RAM?
004E'  75 C9                    259          JNZ     OPRAM                   ; N; JUMP AND TEST IT.
```

```
0050'  EB 2C                    260             JMP     SHORT OPREND            ; Y; SAVE MEM SIZE AND CONTINUE
                                261
                                262     ;
                                263     ;       OPTION RAM END OR FAILURE ROUTINE
                                264     ;
0052'  F7 DB                    265     OERROR: NEG     BX                      ;TRY WORD WITH INVERSE VALUE ALSO
0054'  89 1D                    266             MOV     WORD PTR[DI],BX         ;WRITE INVERTED PATTERN
0056'  33 1D                    267             XOR     BX,WORD PTR[DI]         ;GET BITS IN ERROR INTO BX
0058'  09 DA                    268             OR      DX,BX                   ;COMBINE WITH ORIGINAL ERROR BITS
005A'  B9 0010                  269             MOV     CX,16                   ;DO WHOLE WORD.
005D'  D1 EA                    270     OERRO:  SHR     DX,1                    ;Q;  ERROR BIT?
005F'  73 02                    271             JNB     OERR1                   ; N; CONTINUE CHECK
0061'  FE C5                    272             INC     CH                      ; Y; INC ERROR BIT COUNT
0063'  FE C9                    273     OERR1:  DEC     CL                      ;Q;  16 BITS YET?
0065'  75 F6                    274             JNZ     OERRO                   ; N; JUMP.
0067'  80 FD 04                 275             CMP     CH,4                    ;Q;  MORE THAN FOUR ERRORS?
006A'  77 12                    276             JA      OPREND                  ; Y: JUMP, MUST BE NO RAM.
006C'  8B C6                    277             MOV     AX,SI                   ;CALCULATE ERROR CODE FROM ADDRESS.
006E'  4E                       278             DEC     SI                      ;SHORTEN AVAILABLE RAM TO REQUIRE TEST
006F'  B1 0C                    279             MOV     CL,12                   ;SHIFT RIGHT 12 BITS
0071'  D3 E8                    280             SHR     AX,CL
0073'  04 00"                   281             ADD     AL,OPRERR-1             ;PUT ERROR CODE IN AL
0075'  3C 00"                   282             CMP     AL,OPRERR+6             ;Q;  LEGAL ERROR CODE?
0077'  76 02                    283             JBE     OERR2                   ; Y; JUMP AND DISPLAY IT
0079'  B0 00"                   284             MOV     AL,OPRERR+7             ; N; INDICATE ERROR IN LAST 4 BANKS.
007B'  E8 0009"                 285     OERR2:  CALL    DSPERR                  ;DISPLAY IT
                                286     ;
                                287     ;       FINISHED WITH RAM TEST.
                                288     ;
007E'  2E89 36 C198             289     OPREND: MOV     WORD PTR CS:MEMSIZ+CSWRAP,SI    ;SAVE THE TOP OF RAM VALUE.
0083'                           290     OPRXIT:                                 ;RESTORE NMI INTERRUPT VECTOR
                                291             ASSUME  DS:ROMDAT
0083'  2EC7 06 C008 000A"       292             MOV     WORD PTR CS:NMIVEC+CSWRAP,OFFSET NMIGO
008A'  2E8E 1E C180             293             MOV     DS,WORD PTR CS:CSWRAP+DSADDR    ;SET UP DS TO ROMDAT.
008F'  A0 000D"                 294             MOV     AL,PRC0_M               ;GET LATCH VALUE
0092'  24 F7                    295             AND     AL,NOT PARIEN           ;DISABLE PARITY INTERRUPTS
0094'  E6 03                    296             OUT     PRC0_P,AL
0096'  0C 08                    297             OR      AL,PARIEN               ;REENABLE THEM
0098'  E6 03                    298             OUT     PRC0_P,AL
009A'  FB                       299             STI                             ;RESTORE INTERRUPT CONDITION
                                300
                                301     ;*******************************************************************************
                                302     ;       THIS ROUTINE CHECKS ROMS TO SEE IF THEY ARE THERE, AND OK, AND THEN
                                303     ;       CALLS THEM TO ALLOW THEM TO INITIALIZE ANY HARDWARE OR BOOT.
                                304     ;
                                305     ;       THE FIRST WORD IN A ROM SHOULD BE THE SIZE OF THE ROM IN BYTES.
                                306     ;       THE SECOND WORD SHOULD BE THE ADDRESS OF A POWERUP BOOT SUBROUTINE
                                307     ;       WHICH IS CONTAINED IN THE ROM.
                                308     ;*******************************************************************************
009B'  31 DB                    309             XOR     BX,BX                   ;POINT TO FIRST ROM IN SEGMENT
009D'  BE 0003*                 310             MOV     SI,ROMERX               ;GET ERROR CODE
00A0'  2E8B 07                  311     ROMT2:  MOV     AX,CS:[BX]              ;GET ROM SIZE WORD
```

```
00A3' 8B E8                312             MOV     BP,AX                   ;SAVE FOR ROM TEST
00A5' B1 0D                313             MOV     CL,13                   ;NUMBER OF BITS TO CHECK.
00A7' D1 E8                314   ROMT3:    SHR     AX,1                    ;CHECK FOR VALID SIZE
00A9' 72 04                315             JB      ROMT4                   ;IF CARRY, JUMP
00AB' FE C9                316             DEC     CL                      ;Q:  DONE YET?
00AD' 75 F8                317             JNZ     ROMT3                   ; N: JUMP.
00AF' 75 18                318   ROMT4:    JNZ     ROMT7                   ;IF NOT ZERO, NO ROM HERE
00B1' 8C C8                319             MOV     AX,CS
00B3' 8E C0                320             MOV     ES,AX                   ;SET UP ROM SEGMENT FOR 'ROMTST'
00B5' 53                   321             PUSH    BX                      ;SAVE THE ROM STARTING ADDRESS.
00B6' E8 000B"             322             CALL    ROMTST                  ;GO TEST THE ROM
00B9' 5B                   323             POP     BX
00BA' 74 09                324             JZ      ROMT6                   ;IF OK, JUMP.
00BC' 8B C6                325             MOV     AX,SI                   ;GET ERROR CODE
00BE' 53                   326             PUSH    BX                      ;SAVE STARTING ADDRESS
00BF' E8 0009"             327             CALL    DSPERR                  ; DISPLAY IT
00C2' 5B                   328             POP     BX                      ;
00C3' EB 04                329             JMP     SHORT ROMT7             ;GO TO NEXT ROM
                           330   ;
                           331   ;         SECOND WORD OF ROM OR EPROM CONTAINS ADDRESS OF BOOT ROUTINE
                           332   ;         THIS ROUTINE SHOULD SAVE SI AND BX.  IF NO BOOT ROUTINE IS DESIRED,
                           333   ;         THEN THE WORD SHOULD POINT TO A NEAR RETURN INSTRUCTION.
                           334   ;
00C5' 2EFF 57 02           335   ROMT6:    CALL    CS:WORD PTR 2[BX]       ;GO TO CODE POINTED TO BY SECOND WORD
00C9' 46                   336   ROMT7:    INC     SI                      ;NEXT ERROR CODE.
00CA' 80 C7 20             337             ADD     BH,20H                  ;ROMS ARE ON ROMSIZ BOUNDARYS.
00CD' 81 FB 000C"          338             CMP     BX,OFFSET SYSROM        ;Q:  ALL DONE?
00D1' 75 CD                339             JNZ     ROMT2                   ; N: JUMP AND CONTINUE.
00D3' E8 0007"             340             CALL    DECLED                  ; Decrement LEDs for final time
00D6' E9 0008"             341             JMP     DKBOOT
                           342             END

No errors detected
```

```
                        1   ;*****************************************************************************
                        2   ; TITLE    - PUPTST - PEGASUS POWERUP TEST CODE
                        3   ; ABSTRACT - THE TEST IS DESIGNED TO CHECK OUT THE SYSTEM ROM AND RAM AND TO
                        4   ;        ALLOW THE FACTORY TO ACTIVATE ANOTHER EPROM FOR TEST PURPOSES.  THE
                        5   ;        TEST IS RUN WITH INTERRUPTS DISABLED AND NEEDS NO RAM FOR INITIAL
                        6   ;        CHECKOUT.
                        7   ;
                        8   ; REGISTERS USED - ALL
                        9   ;
                       10   ; STACK USED - SS:SP POINTS TO A ROM RETURN TABLE TO AVOID USING RAM.
                       11   ;
                       12   ;*****************************************************************************
                       13           NAME    PUPTST
                       14           SUBTTL  PEGASUS POWERUP DIAGNOSTIC TESTS
                       16   ;*****************************************************************************
                       17   ;                               PUBLIC DEFINITIONS
                       18   ;*****************************************************************************
                       19           PUBLIC  ROMTST          ;ROM CRC TEST ROUTINE
                       20           PUBLIC  PUPTST          ;POWER UP TEST ENTRY
                       21   ;*****************************************************************************
                       22   ;                               EXTERNAL REFERENCES
                       23   ;*****************************************************************************
=0000                  24           SECT    ROMCOD
                       25           EXTERN  INICRT:NEAR     ;CRT HARDWARE INITIALIZATION ROUTINE
                       26           EXTERN  EPROM:BYTE      ;ADDRESS OF BEGINNING OF EPROM
                       27           EXTERN  SYSROM:BYTE     ;ADDRESS OF BEGINNING OF SYSTEM ROM
                       28           EXTERN  DSPDIE:NEAR     ;ADDRESS OF FAILURE ROUTINE
                       29           EXTERN  LED110:ABS      ;ROM FAILED ERROR CODE
                       30           EXTERN  LED101:ABS      ;SYSTEM RAM FAILED ERROR CODE
                       31           EXTERN  LED100:ABS      ;PASSED SYSTEM RAM TEST , NEXT.
                       32           EXTERN  RAMINI:NEAR     ;ADDRESS OF RAM INIT ROUTINE.
                       33           EXTERN  ROMSIZ:ABS      ;SIZE OF THE SYSTEM ROM
                       34           EXTERN  ROMF4:NEAR      ;ADDRESS OF FACTORY ROM
                       35   ;*****************************************************************************
                       36   ;                               LOCAL CONSTANTS
                       37   ;       INCLUDE PEG:PORTADDR.EQU
                       38   ;       INCLUDE PEG:LATCHES.EQU
                       39   ;       INCLUDE PEG:FLOPPY.EQU
                       40   ;       INCLUDE PEG:VECTOR.EQU
                       41   ;       INCLUDE PEG:WINCHSTR.EQU
                       42   ;       INCLUDE PEG:
                       43   ;*****************************************************************************
                      237   ;
=000C                 238   RAMBEG  EQU     BRKINT*4        ;START OF VALID RAMTEST
=0008                 239   NMIVEC  EQU     NMIINT*4        ;NMI INTERRUPT VECTOR LOCATION
=7FFA                 240   RMTSSZ  EQU     (0-RAMBEG) SHR 1        ;SIZE OF RAM IN WORDS
                      241   ;
                      242   ;*****************************************************************************
                      243   ;                               MODULE ENTRY POINT
                      244   ;*****************************************************************************
=0000                 245           SECT    ROMCOD          ;ALL CODE IS PLACE HERE
                      246           ASSUME  CS:ROMCOD
```

```
                                247   ;
0000'                           248   PUPTST  PROC    FAR                     ;POWER UP TEST ENTRY POINT
0000'  2E81 3E 000A" 4946       249           CMP     CS:WORD PTR ROMF4,'IF'  ;Q:  FACTORY INTERFACE?
0007'  75 03                    250           JNZ     T1                      ; N: CONTINUE POWER UP.
0009'  E9 0000"                 251           JMP     NEAR PTR ROMF4+2        ; Y: GO TO FACTORY EPROM.
                                252   ;
                                253   ;       INITIALIZE THE I/O SYSTEM TO PREVENT PROBLEMS.
                                254   ;
000C'  B0 00                    255   T1:     MOV     AL,00000000B
000E'  E6 00                    256           OUT     DSKC_P,AL               ;SHUT OFF TIMERS
0010'  E6 02                    257           OUT     PRDA_P,AL               ;CLEAR PRINTER DATA LATCH
0012'  B0 70                    258           MOV     AL,(PRAUTO+PRSTRB+PRINIT)       ; 'PRCO_P' latch
0014'  E6 03                    259           OUT     PRCO_P,AL
0016'  B0 0F                    260           MOV     AL,0FH                  ; 'DSKS_P' latch.  Deselect all drives
0018'  E6 04                    261           OUT     DSKS_P,AL
001A'  B0 10                    262           MOV     AL,WRSTCM               ;RESET THE WINCHESTER
001C'  E6 31                    263           OUT     WINRST,AL
001E'  DB E3                    264           DB      0DBH,0E3H               ; 8087 'FNINIT' INSTRUCTION - TO RESET
                                265                                           ;   A POSSIBLE 8087
                                266   ;
                                267   ;       INITIALIZE THE CRT HARDWARE
                                268   ;
0020'  8C C8                    269           MOV     AX,CS                   ;SET UP A TEMPORARY STACK
0022'  8E D0                    270           MOV     SS,AX                   ; SEGMENT SAME AS CODE
0024'  BC 00DF"                 271           MOV     SP,OFFSET TSTACK        ; TOP OF LIST OF RETURN WORDS IN ROM
0027'  E9 0001"                 272           JMP     INICRT                  ;GO SET UP THE CRT HARDWARE
                                273   ;
                                274   ;       CHECK OUT THE SYSTEM ROM
                                275   ;
002A'  BD 0009*                 276   T1P5:   MOV     BP,ROMSIZ               ;GET THE SIZE OF THE ROM
002D'  8C CB                    277           MOV     BX,CS
002F'  8E C3                    278           MOV     ES,BX                   ;START SEGMENT OF ROM
                                279           ASSUME ES:ROMCOD
0031'  BB 0003"                 280           MOV     BX,OFFSET SYSROM        ;START ADDRESS OF ROM
0034'  B0 00"                   281           MOV     AL,(LED110+PRAUTO+PRSTRB+PRINIT)        ;TURN OFF FIRST LED
0036'  E6 03                    282           OUT     PRCO_P,AL
0038'  E9 0083                  283           JMP     ROMTST                  ;GO CHECK IT
                                284                                           ;Q:  ROM OK?
003B'  74 0D                    285   T1P6:   JZ      RAMTST                  ; Y: JUMP.
                                286
                                287   ;
                                288   ;       ROM ERROR SHUTDOWN ROUTINES
                                289   ;
003D'  B0 05*                   290           MOV     AL,LED110       ;ERROR CODE FOR ROM FAIL.
003F'  30 D2                    291           XOR     DL,DL           ;NOTHING FOR PARALLEL PORT
0041'  E9 0004"                 292   ERROR:  JMP     DSPDIE          ;STOP POWERUP
                                293   ;
                                294   ;       RAM ERROR SHUTDOWN ROUTINE
                                295   ;
0044'                           296   RERROR  PROC    FAR
0044'  08 F2                    297           OR      DL,DH           ;COMBINE BYTES IN THE WORD FOR PARALLEL PORT
0046'  B0 06*                   298           MOV     AL,LED101       ;ERROR CODE FOR SYSTEM RAM FAIL.
```

```
0048'  EB F7                    299             JMP     ERROR
                                300     ;*******************************************************************************
                                301     ;       RAMTEST         THIS IS A SIMPLE TEST OF THE SYSTEM RAM.  IT IS
                                302     ;                       USED TO CLEAR THE PARITY BIT IN ALL OF THE RAM, QUICKLY CHECK
                                303     ;                       THE RAM FOR ERRORS.
                                304     ;*******************************************************************************
004A'  B0 00"                   305     RAMTST: MOV     AL,(LED101+PRAUTO+PRSTRB+PRINIT) ;ADVANCE LED INDICATION
004C'  E6 03                    306             OUT     PRC0_P,AL                       ;    FOR NEXT TEST
004E'  31 E4                    307             XOR     SP,SP                           ;CLEAR STACK POINTER
0050'  8E D4                    308             MOV     SS,SP                           ;SET UP STACK SEGMENT
0052'  8E DC                    309             MOV     DS,SP                           ;SET UP DATA SEGMENT
0054'  8E C4                    310             MOV     ES,SP                           ;SET UP EXTRA SEGMENT
                                311     ;
                                312             ASSUME  DS:ABS0, ES:ABS0, SS:ABS0
0056'  8B 26 0198               313             MOV     SP,WORD PTR MEMSIZ              ;GET WARM RESET FLAG INTO SP.
005A'  BE 0008                  314             MOV     SI,OFFSET NMIVEC                ;POINTER TO NMI VECTOR
005D'  C7 04 0044"              315             MOV WORD PTR[SI],OFFSET RERROR  ;STORE THE NMI ERROR VECTOR OFFSET
0061'  8C 4C 02                 316             MOV     2[SI],CS                        ;STORE THE SEGMENT(SAME AS THIS CODE)
0064'  81 3C 0044"              317             CMP WORD PTR[SI],OFFSET RERROR  ;Q:  NMI ERROR VECTOR OFFSET OK?
0068'  75 07                    318             JNZ     VECERR                          ; N: JUMP AND FLAG ERROR.
006A'  81 7C 02 0000'           319             CMP     WORD PTR 2[SI],ROMCOD   ;Q:  NMI ERROR VECTOR SEGMENT OK?
006F'  74 05                    320             JZ      R000                    ; Y: CONTINUE THE TEST.
0071'  BC 55AA                  321     VECERR: MOV     SP,55AAH                        ;FLAG VECTOR ERROR OCCURRED.
0074'  EB 04                    322             JMP     SHORT R00
0076'  B0 00"                   323     R000:   MOV     AL,(PARIEN+LED101+PRAUTO+PRSTRB+PRINIT) ;TURN ON PARITY INTRs
0078'  E6 03                    324             OUT     PRC0_P,AL
007A'  8B C4                    325     R00:    MOV     AX,SP                           ;1000H (= RAMSIZ (= 8000H IF WARM RESET
007C'  80 EC 10                 326             SUB     AH,10H                          ;Q:  WARM RESET?
007F'  25 8FFF                  327             AND     AX,8FFFH                        ;
0082'  74 37                    328             JZ      REND                            ; Y: EXIT TEST ROUTINE
0084'  BD 0002                  329             MOV     BP,00000010B                    ;WALKING ONE
0087'  B9 7FFA                  330     R0:     MOV     CX,OFFSET RMTSSZ                ;SET UP SIZE OF RAM TO TEST
008A'  BF 000C                  331             MOV     DI,RAMBEG                       ;SET UP START OF RAM.
008D'  8B F7                    332             MOV     SI,DI                           ;COPY TO SI FOR LATER.
008F'  8B C5                    333             MOV     AX,BP                           ;GET WALKING ONE TO AX.
0091'  D1 C8                    334             ROR     AX,1                            ;SET UP THE CARRY BIT
0093'  AB                       335     R1:     STOS    WORD PTR [DI]                   ;STORE WALKING ONE
0094'  D1 D0                    336             RCL     AX,1                            ;WALK IT
0096'  E2 FB                    337             LOOP    R1
0098'  E2 FE                    338             LOOP    $                       ;ALLOW RAM TO FORGET IF REFRESH IS NOT WORKING
                                339
                                340     ;
                                341     ;       READ AND CHECK THE STORED PATTERN
                                342     ;
009A'  8B DD                    343             MOV     BX,BP                           ;GET INITIAL PATTERN
009C'  B9 7FFA                  344             MOV     CX,OFFSET RMTSSZ                ;SIZE OF RAM IN WORDS
009F'  D1 CB                    345             ROR     BX,1                            ;SET UP THE CARRY BIT
00A1'  9F                       346     R2:     LAHF                                    ;SAVE FLAGS
00A2'  8B D3                    347             MOV     DX,BX                           ;GET THE WORD THAT WAS WRITTEN.
00A4'  33 14                    348             XOR     DX,[SI]                         ;Q:  IS THIS OK?
00A6'  75 9C                    349             JNZ     RERROR                          ; N: JUMP
00A8'  46                       350             INC     SI                              ;BUMP POINTERS
```

```
00A9'  46                    351             INC     SI
00AA'  9E                    352             SAHF                            ;GET FLAGS
00AB'  D1 D3                 353             RCL     BX,1                    ;NEXT PATTERN TO CHECK
00AD'  E2 F2                 354             LOOP    R2                      ;CONTINUE TILL FINISHED
00AF'  F7 D5                 355             NOT     BP                      ;SET UP FOR WALKING ZERO
00B1'  21 ED                 356             AND     BP,BP                   ;Q; SECOND LOOP?
00B3'  78 D2                 357             JS      RO                      ;N; GO FOR NEXT LOOP
00B5'  81 FC 55AA            358             CMP     SP,55AAH                ;Q;  WERE VECTORS BAD?
00B9'  74 89                 359             JZ      RERROR                  ; Y; GO REPORT ERROR.
                             360     ;
                             361     ;       MAIN RAM OK.
                             362     ;
00BB'  E9 0008"              363     REND:   JMP     RAMINI                  ;RUN THE RAM INITIALIZATION
                             364
                             365     ;*******************************************************************************
                             366     ;                       LOCAL SUBROUTINES
                             367     ;*******************************************************************************
                             368     ;       ROM TEST        THIS TEST CHECKS OUT THE SYSTEM ROM USING A CRC 16
                             369     ;       ROUTINE         TYPE CHECK.  THE LAST WORD IN THE ROM IS EXPECTED TO
                             370     ;                       CONTAIN THE CORRECT CRC REMAINDER SUCH THAT WHEN THE
                             371     ;                       TEST IS MADE, THE REMAINDER IS ZERO.
                             372     ;       ON ENTRY        BP= SIZE OF ROM
                             373     ;                       BX= OFFSET OF ROM
                             374     ;                       ES= SEGMENT OF ROM
                             375     ;                       AX,CX ARE USED
                             376     ;                       SI,DI,DS,ES ARE NOT CHANGED
                             377     ;       ON EXIT         DX=REMAINDER
                             378     ;                       ZF=SET IF DX=0
                             379     ;*******************************************************************************
00BE'                        380     ROMTST  PROC    NEAR                    ;ENTRY POINT
00BE'  31 D2                 381             XOR     DX,DX                   ;CLEAR THE INITIAL REMAINDER
00C0'  268A 27               382     ROMTS1: MOV     AH,ES:[BX]              ;GET BYTE
                             383     ;
                             384     ;       THIS IS THE KERNAL OF THE CRC16 ROUTINE (FOR A BYTE).
                             385     ;
00C3'  B9 0008               386             MOV     CX,8                    ;LENGTH OF BYTE
00C6'  8A C4                 387     T3:     MOV     AL,AH                   ;GET BYTE TO TEMP REGISTER
00C8'  30 D0                 388             XOR     AL,DL                   ;CHECK IF THE LOW BITS ARE EQUAL
00CA'  D1 EA                 389             SHR     DX,1                    ;SHIFT REMAINDER RIGHT
00CC'  D1 E8                 390             SHR     AX,1                    ;MOV LOW BIT TO CARRY AND SHIFT BYTE RIGHT
00CE'  73 04                 391             JNB     T4                      ;IF LOW BIT OFF, JUMP.
00D0'  81 F2 A001            392             XOR     DX,0A001H               ;XOR GENERATOR AND REMAINDER.
00D4'  E2 F0                 393     T4:     LOOP    T3                      ;FINISH BYTE
                             394     ;
00D6'  43                    395             INC     BX                      ;POINT TO NEXT BYTE IN ROM
00D7'  4D                    396             DEC     BP                      ;DECREMENT COUNT
00D8'  75 E6                 397             JNZ     ROMTS1
00DA'  31 D2                 398             XOR     DX,DX                   ;     FORCE ROM TO BE CORRECT
00DC'  21 D2                 399             AND     DX,DX                   ;Q;   IS THE ROM CORRECT?
00DE'  C3                    400             RET
                             401     ;*******************************************************************************
                             402     ;                               MODULE DATA DEFINITIONS
```

```
                          403   ;*****************************************************************************
                          404   ;
00DF'                     405   TSTACK  LABEL   WORD            ;A STACK WHICH IS USED DURING POWERUP TO RETURN
00DF'  002A"              406           DW      OFFSET T1P5     ;RETURN ADDRESS FOR INICRT ROUTINE
00E1'  003B"              407           DW      OFFSET T1P6     ;RETURN ADDRESS FOR ROM TEST ROUTINE
                          408   ;
                          409           END
```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ;  TITLE    - RAMINI - ROMDAT data area initialization
                        3   ;  COMPUTER - 8088 Assembly Language  (BSO Assembler)
                        4   ;  ABSTRACT - This module is responsible for setting up the ROM's data
                        5   ;       area in RAM.  This consists of initializing the timing event blocks,
                        6   ;       initializing the three hardware latches and their corresponding
                        7   ;       memory locations, and setting up a stack for the rest of the
                        8   ;       powerup code to use.  It also sets up the 8088's normal system
                        9   ;       interrupt vectors (types 0-4).
                       10   ;
                       11   ;       Note that the floating DS (in DSADDR) is set up at this time also.
                       12   ;       ROMDAT is 'temporarily' located at 40:0 by the linker, used as
                       13   ;       ROMDAT by the ROM, and as far as the assembler is concerned, is
                       14   ;       in fact the segment ROMDAT.  The disk boot may find that this area
                       15   ;       is needed by whatever DOS is being loaded, and therefore must be
                       16   ;       moved.  This move is described in the module 'SYSORQ'.
                       17   ;
                       18   ;*****************************************************************************
                       19           NAME    RAMINI - ROMDAT data area initialization
                       20   ;       SUBTTL
=FFFF                  22   DEBUG   EQU     0FFFFH
                       23
                       24   ;*****************************************************************************
                       25   ;                           PUBLIC DEFINITIONS
                       26   ;*****************************************************************************
                       27   ;
                       28           PUBLIC  RAMINI
                       29   ;
                       30   ;*****************************************************************************
                       31   ;                          EXTERNAL REFERENCES
                       32   ;*****************************************************************************
                       33   ;
                       34   ;
=0000                  35           SECT    ABS0
                       36           EXTRN   BOOTMV:BYTE     ; Load address of disk boot sector     (SYSORQ)
                       37                                   ; (Used as base of init stack)
=0000                  38           SECT    ROMCOD
                       39           EXTRN   VECTBO:WORD     ; 8088 vector table                    (VECINI)
                       40           EXTRN   VTOLEN:ABS      ; Size of 8088 vector table in words   (VECINI)
                       41           EXTRN   BEEPOF:NEAR     ; System beeper turnoff subroutine     (BELDSR)
                       42           EXTRN   SOFINT:NEAR     ;* Used in place of the above          (DSKIO )
                       43           EXTRN   INTTST:NEAR     ; Next routine in powerup sequence     (INTTST)
                       44           EXTRN   LED100:ABS      ; Interrupt/timer failure LED pattern  (ROMERR)
                       45           EXTRN   MOTOFF:NEAR     ;* Floppy disk motor turnoff routine   (FLPXXX)
                       46           EXTRN   TIMOUT:NEAR     ;* Disk controller timeout subroutine  (DSKIO )
                       47   ;
=0000                  48           SECT    ROMDAT
                       49           EXTRN   BELEVT:WORD     ; System bell timing event block       (ROMDAT)
                       50           EXTRN   DSKC_M:BYTE     ; Disk control latch memory            (ROMDAT)
                       51           EXTRN   DSKS_M:BYTE     ; Disk select latch memory             (ROMDAT)
                       52           EXTRN   DSKEVT:WORD     ;* Disk driver timeout event           (ROMDAT)
                       53           EXTRN   FL1EVT:WORD     ;* Floppy disk motor-off event         (ROMDAT)
```

```
                        54          EXTRN   FL2EVT:WORD     ;* Floppy disk general timing event    (ROMDAT)
                        55          EXTRN   PRCO_M:BYTE     ; Printer control latch memory         (ROMDAT)
                        56          EXTRN   SYSCON:WORD     ; System Configuration word            (ROMDAT)
                        57          EXTRN   TIKCTR:BYTE     ; System timer tick counter (load loc) (ROMDAT)
                        58  ;
                        59  ;*****************************************************************************
                        60  ;                           LOCAL CONSTANTS
                        61  ;       INCLUDE PEQ:PORTADDR.EQU
                        62  ;       INCLUDE PEQ:VECTOR.EQU
                        63  ;       INCLUDE PEQ:LATCHES.EQU
                        64  ;*****************************************************************************
                        65  ;
                       220
                       221  ;*****************************************************************************
                       222  ;                           CODE SEGMENT DEFINITION
                       223  ;*****************************************************************************
                       224  ;
      =0000            225          SECT    ROMCOD
                       226          ASSUME  CS:ROMCOD
                       227  ;
                       228  ;*****************************************************************************
                       229  ;                           LOCAL DATA AREA
                       230  ;*****************************************************************************
                       231  ;
                       232  ;   This table is used to initialize the timing event blocks in ROMDAT by
                       233  ;   being loaded at offset TIKCTR.
                       234  ;
                       235  ; ***************************************************************
                       236  ; ***** THIS TABLE MUST MATCH THE CORRESPONDING DEFINITION    *****
                       237  ; ***** STRUCTURE IN THE MODULE 'ROMDAT' (STARTING AT TIKCTR) *****
                       238  ; ***************************************************************
                       239  ;
0000'                  240  EVTINI  LABEL   BYTE
                       241  ;
                       242  ;   System timer data area
                       243  ;
                       244  ;TIKCTR LABEL   BYTE            ; System timer (25 msec) tick counter
0000'  00              245          DB      00
                       246  ;
                       247  ;   Timing services permanent queue entries
                       248  ;
                       249  ;ROMEVT LABEL   WORD            ; Start of timing events linked-list
0001'  000D"           250          DW      OFFSET DSKEVT   ;
                       251  ;
                       252  ;DSKEVT LABEL   DWORD           ;* Disk driver timeout event
0003'  000E"           253          DW      offset FL1EVT   ;* Linked list pointer
0005'  FF              254          DB      0FFH            ;* Task ID (=0FFH for permanent events)
0006'  00              255          DB      00              ;* Flags
0007'  0000            256          DW      0000            ;* Counter
0009'  0009"           257          DW      offset TIMOUT   ;* Subroutine address (disk TIMeOUT)
                       258  ;
                       259  ;FL1EVT LABEL   DWORD           ;* Floppy disk motor-off event
```

```
000B'  000F"                260           DW      offset FL2EVT   ;* Linked list pointer
000D'  FF                   261           DB      OFFH            ;* Task ID (=OFFH for permanent events)
000E'  00                   262           DB      00              ;* Flags
000F'  0000                 263           DW      0000            ;* Counter
0011'  0008"                264           DW      offset MOTOFF   ;* Subroutine address (disk MOTor-OFF)
                            265   ;
                            266   ;FL2EVT LABEL   DWORD           ;* Floppy disk timing event
0013'  000A"                267           DW      offset BELEVT   ;* Linked list pointer (last entry)
0015'  FF                   268           DB      OFFH            ;* Task ID (=OFFH for permanent events)
0016'  00                   269           DB      00              ;* Flags
0017'  0000                 270           DW      0000            ;* Counter
0019'  0005"                271           DW      offset SOFINT   ;* Subroutine address (FLoppy TIME event)
                            272   ;                               ;* Note that we have actually reference SOFINT
                            273   ;BELEVT LABEL   WORD
001B'  0000                 274           DW      0000            ;* Linked list pointer
001D'  FF                   275           DB      OFFH            ; Task ID (=OFFH for permanent events)
001E'  00                   276           DB      00              ; Flags
001F'  0000                 277           DW      0000            ; Counter
0021'  0004"                278           DW      OFFSET BEEPOF   ; Subroutine address (system BEEPer OFf)
0023'                       279   EVTEND  LABEL   BYTE
       =0023                280   EVTLEN  EQU     EVTEND-EVTINI   ; Length of initialization table
                            281
                            282   ;*****************************************************************************
                            283   ;                               MODULE ENTRY POINT
                            284   ;*****************************************************************************
                            285   ;
                            286           ASSUME  DS:ROMDAT, ES:ABS0
                            287   ;
0023'                       288   RAMINI  PROC    NEAR
                            289   ;       CLD                             ; forward strings
                            290   ;       CLI                             ; make sure interrupts are disabled
                            291   ;
                            292   ;   Set up initialization stack
                            293   ;
0023'  31 C0                294           XOR     AX,AX
0025'  8E D0                295           MOV     SS,AX                   ;;; Stack segment is ABS0
0027'  BC 0001"             296           MOV     SP,OFFSET BOOTMV        ;;; Stack is right below boot loc
002A'  0E                   297           PUSH    CS                      ;;;
002B'  1F                   298           POP     DS                      ;;; DS = CS
                            299   ;
                            300   ;   Initialize the 8088 vectors
                            301   ;                                           DS = CS = ROMCOD
                            302   ;       MOV     AX,ABS0                 ;;; (VECTBO is in ROMCOD)
002C'  8E C0                303           MOV     ES,AX                   ;;; ES points to ABS0
002E'  BE 0002"             304           MOV     SI,OFFSET VECTBO        ;;; DS:SI have source for move
0031'  BF 0000              305           MOV     DI,OFFSET ZDVINT*4      ;;; ES:DI have destination for move
0034'  B9 0003*             306           MOV     CX,VTOLEN               ;;; CX has size to move
0037'  F2                   307           REP                             ;;;
0038'  A5                   308           MOVB    WORD PTR [DI],WORD PTR [SI] ;;; Do it
                            309   ;
                            310   ;   Now initialize Timing Event Blocks in ROMDAT
                            311   ;
```

```
0039'  B8 0000'          312          MOV     AX,ROMDAT                ;;; Note that DS = CS = ROMCOD
003C'  8E C0             313          MOV     ES,AX                    ;;; ES points to initial ROMDAT loc.
003E'  BE 0000"          314          MOV     SI,OFFSET EVTINI         ;;; DS:SI have source for move
0041'  BF 0012"          315          MOV     DI,OFFSET TIKCTR         ;;; ES:DI have destination for move
0044'  B9 0023           316          MOV     CX,OFFSET EVTLEN         ;;; CX has size to move
0047'  F2                317          REP                              ;;;
0048'  A4                318          MOVS    BYTE PTR [DI], BYTE PTR [SI]    ;;; Do the move
0049'  2EA3 C180         319          MOV     WORD PTR CS:DSADDR+CSWRAP,AX    ;;; Set floating DS (ROMDAT)
004D'  8E D8             320          MOV     DS,AX                    ;;; Set new DS (ROMDAT)
                         321  ;
                         322  ;   Initialize the ports to match their RAM counterparts
                         323  ;
004F'  B0 A0             324          MOV     AL,(SIDE_0+ACCFDC)       ;;; Set up initial value
0051'  A2 000B"          325          MOV     DSKC_M,AL                ;;; Set the RAM version
0054'  E6 00             326          OUT     DSKC_P,AL                ;;; Set the latch
0056'  B0 00"            327          MOV     AL,(LED100+PARIEN+PRAUTO+PRSTRB+PRINIT)   ;;; Initial value
0058'  A2 0010"          328          MOV     PRCO_M,AL                ;;;
005B'  E6 03             329          OUT     PRCO_P,AL                ;;; (LEDs still OK)
005D'  B0 FF             330          MOV     AL,0FFH                  ;;; Initial value
                         331  ;       MOV     DSKS_M,AL                ;;;
005F'  E6 04             332          OUT     DSKS_P,AL                ;;;
0061'  31 C0             333          XOR     AX,AX                    ;;;
0063'  A3 0011"          334          MOV     SYSCON,AX                ;;; Clear the configuration word
                         335  ;
                         336  ;   Go to next
                         337  ;
                         338  ;       MOV     AX,ABS0                  ;;; INTTST assumes that
0066'  8E C0             339          MOV     ES,AX                    ;;;    ES points to ABS0
0068'  E9 0006"          340          JMP     INTTST                   ;;; (Interrupts still disabled)
                         341  ;
                         342          END

No errors detected
```

```
                                1   ;*****************************************************************************
                                2   ;  TITLE    - RESET - Powerup reset code
                                3   ;  COMPUTER - 8088 Assembly Language  (BSO Assembler)
                                4   ;  ABSTRACT - The module contains the code that goes in the RESET segment
                                5   ;*****************************************************************************
                                6           NAME    RESET - Powerup reset code
                                7   ;       SUBTTL
                                9   ;*****************************************************************************
                               10   ;                           PUBLIC DEFINITIONS
                               11   ;*****************************************************************************
                               12   ;
                               13           PUBLIC  PWRUP
                               14   ;
                               15   ;*****************************************************************************
                               16   ;                           EXTERNAL REFERENCES
                               17   ;*****************************************************************************
                               18   ;
                               19           EXTRN   PUPTST:FAR      ; Start of powerup test code          (PUPTST)
                               20   ;
                               21   ;*****************************************************************************
                               22   ;                           MODULE ENTRY POINT
                               23   ;*****************************************************************************
                               24   ;
                               25   ;   RESET is located at absolute memory location OFFFF0H.  This is where the
                               26   ;   processor starts after being reset (with CS = FFFFH, other seg-regs = 0)
                               27   ;
                    =0000      28           SECT    RESET
                               29           ASSUME  CS:RESET
                               30   ;
                    =0000      31           ORG     0000H
                               32   ;
0000'                          33   PWRUP   PROC    FAR
0000'  FA                      34           CLI                     ; Disable interrupts (for keyboard restart)
0001'  FC                      35           CLD                     ; Start out with forward strings
0002'  EA 0001" 0000"          36           JMP     PUPTST          ; Intersegment jump to powerup test code
                               37   ;
0007'  50 65 67 61 73 75       38           DB      'Pegasus'
000D'  73
                    =000E      39           ORG     000EH
000E'  9CEA                    40           DW      9CEAH           ; CRC of the system ROM (this is determined
                               41                                   ; empirically with the procedure described
                               42                                   ; in the file README.DOC)
                               43           END
```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ;
                        3   ;  TITLE    - ROMDAT - Pegasus system ROM data areas
                        4   ;  COMPUTER - 8088 Assembly Language  (BSO Assembler)
                        5   ;  ABSTRACT - This module contains the RAM data areas for the various
                        6   ;       ROM-based system routines.  Note that the actual location of these
                        7   ;       variables in memory is variable, depending on the current value
                        8   ;       for the ROM DS pointer, 'DSADDR'.  This module is just a skeleton
                        9   ;       used to define the offsets of the variables.  The initialization
                       10   ;       is handled by the various routines that actually use the RAM.
                       11   ;       One exception to this is the routine 'RAMINI' which sets up some
                       12   ;       of the constant data (TIKCTR and Timing Event blocks).
                       13   ;*****************************************************************************
                       14           NAME    ROMDAT - TIPC system ROM data areas             -
                       15   ;       SUBTTL
=FFFF                  17   DEBUG   EQU     OFFFFH
                       18   ;*****************************************************************************
                       19   ;                           PUBLIC DEFINITIONS
                       20   ;*****************************************************************************
                       21   ;
                       22           PUBLIC  ATTSAV
                       23           PUBLIC  BELEVT
                       24           PUBLIC  CRTCNT
                       25           PUBLIC  CRTERR
                       26           PUBLIC  CRTHLD
                       27           PUBLIC  CURPOS
                       28           PUBLIC  CURTYP
                       29           PUBLIC  D_DBGN                  ;* Beginning of disk driver data area
                       30           PUBLIC  D_DEND                  ;* End of disk driver data area
                       31           PUBLIC  D_DIT                   ;* Disk table of Disk Interface Tables
                       32           PUBLIC  D_EKNT                  ;* Disk error counter
                       33           PUBLIC  D_NCMD                  ;* Disk next operation to perform
                       34           PUBLIC  D_NSTA                  ;* Disk next state for driver
                       35           PUBLIC  D_POS                   ;* Disk table of current positions
                       36           PUBLIC  D_REQ                   ;* Disk Request Information Block
                       37           PUBLIC  D_SOFT                  ;* Disk software interrupt flag
                       38           PUBLIC  D_STAT                  ;* Disk driver current state
                       39           PUBLIC  D_WAIT                  ;* Disk waiting for I/O completion flag
                       40           PUBLIC  D_WKSP                  ;* Disk driver workspace
                       41           PUBLIC  DATE
                       42           PUBLIC  DISBEG
                       43           PUBLIC  DISEND
                       44           PUBLIC  DKSPSV
                       45           PUBLIC  DKSSSV
                       46           PUBLIC  DSKC_M
                       47           PUBLIC  DSKEVT                  ;* Disk driver timeout event
                       48           PUBLIC  DSKISP
                       49           PUBLIC  DSKS_M
                       50           PUBLIC  ENDG
                       51           PUBLIC  FL1EVT                  ;* Floppy disk motor-off event
                       52           PUBLIC  FL2EVT                  ;* Floppy general timing event
                       53           PUBLIC  HOURS
```

```
                        54          PUBLIC  HUNS
                        55          PUBLIC  IXSPSV
                        56          PUBLIC  IXSSSV
                        57          PUBLIC  KBMODE
                        58          PUBLIC  KBSSSV
                        59          PUBLIC  KBSPSV
                        60          PUBLIC  KEYISP
                        61          PUBLIC  MINS
                        62          PUBLIC  NUMCNT
                        63          PUBLIC  NUMDRV                  ;* Number of disk drive units
                        64          PUBLIC  NUMVAL
                        65          PUBLIC  PRCO_M
                        66          PUBLIC  QDEPTH
                        67          PUBLIC  QFRONT
                        68          PUBLIC  QREAR
                        69          PUBLIC  QUEUE
                        70          PUBLIC  RETFLQ
                        71          PUBLIC  RMDTBQ
                        72          PUBLIC  RMDTSZ
                        73          PUBLIC  ROMEVT
                        74          PUBLIC  SECS
                        75          PUBLIC  STATLN
                        76          PUBLIC  SYSCON
                        77          PUBLIC  THSLIN
                        78          PUBLIC  TIKCTR
                        79          PUBLIC  TIMISP
                        80          PUBLIC  TMSPSV
                        81          PUBLIC  TMSSSV
                        82  ;
                        83  ;                       EXTERNAL REFERENCES
                        84  ;
                        85  ;       EXTRN   BEEPOF:NEAR     ; System beeper turnoff subroutine      (OUTPUT)
                        86  ;       EXTRN   FLTIME:NEAR     ;* Floppy disk timing event routine     (FLPXXX)
                        87  ;       EXTRN   LED100:ABS      ; Interrupt/timer failure LED pattern   (ROMERR)
                        88  ;       EXTRN   MOTOFF:NEAR     ;* Floppy disk motor turnoff routine    (FLPXXX)
                        89  ;       EXTRN   SSSTD:NEAR      ;* Single-side, single track dens DIT   (FLPDIT)
                        90  ;       EXTRN   TIMOUT:NEAR     ;* Disk controller timeout subroutine   (DSKUTL)
                        91  ;
                        92  ;
                        93  ;                       DISK DRIVER STATE DEFINITIONS
                        94  ;                           INCLUDE PEQ:DSKSTA.EQU
                       106  ;
                       107  ;                       REQUEST INFORMATION BLOCK STRUCTURE DEFINITION
                       108  ;                               INCLUDE PEQ:DSKRIB
                       109
                       125  ;
                       126  ;                       LOCAL CONSTANTS
                       127  ;
                       128  ;
=0008                  129  NUMDRV  EQU     8               ;* Max number of disk drives
=0010                  130  QSIZE   EQU     16              ; Keyboard type-ahead buffer size
                       131  ;
```

```
                        ;*****************************************************************************
                        ;                          BEGINNING OF DATA AREA
                        ;*****************************************************************************
                        ;
                        ;    N O T E ===))) The memory locations from DSKC_M through DISBEQ must
                        ;       remain in their relative locations in later versions of the ROM.
                        ;
=0000                   ;          SECT    ROMDAT
                        ;
0000'                   RMDTBQ  LABEL   BYTE
                        ;
                        ;   Hardware 'memory' - NOTE: These three bytes must be the first in ROMDAT
                        ;
0000'  =0001            DSKC_M  DB      1       ; BYTE - 'DSKC_P' latch
0001'  =0001            PRCO_M  DB      1       ; BYTE - 'PRCO_P' latch
0002'  =0001            DSKS_M  DB      1       ; BYTE - 'DSKS_P' latch
                        ;
                        ;-------------------------------------------------------------------------
                        ;   System configuration word - See 'PEQ:SYSCON.EQU' for bit definitions
                        ;
0003'  =0002            SYSCON  DB      2       ; WORD - System config - initialized during powerup
                        ;
                        ;-------------------------------------------------------------------------
                        ;   Timing services permanent queue entries - This area is initialized from
                        ;       a table in the module 'RAMINI'.  Any changes made here should also
                        ;       be made in RAMINI.
                        ;
                        ;       ********************************************************
                        ;       ****   The following data items (up through DISBEQ)   ****
                        ;       ****   must remain as contiguous memory locations.    ****
                        ;       ********************************************************
                        ;
0005'  =0001            TIKCTR  DB      1       ; BYTE - System timer (25 msec) tick counter
                        ;
0006'                   ROMEVT  LABEL   WORD    ; Start of timing events linked-list
=0002                           DB      2       ; WORD - Pointer to first event block
                        ;
0008'                   DSKEVT  LABEL   WORD    ;* Disk driver timeout event
=0002                           DB      2       ;* WORD - Linked list pointer
=0001                           DB      1       ;* BYTE - Task ID (=0FFH for permanent events)
=0001                           DB      1       ;* BYTE - Flags
=0002                           DB      2       ;* WORD - Counter
=0002                           DB      2       ;* WORD - Subroutine address (disk TIMeOUT)
                        ;
0010'                   FL1EVT  LABEL   WORD    ;* Floppy disk motor-off event
=0002                           DB      2       ;* WORD - Linked list pointer
=0001                           DB      1       ;* BYTE - Task ID (=0FFH for permanent events)
=0001                           DB      1       ;* BYTE - Flags
=0002                           DB      2       ;* WORD - Counter
=0002                           DB      2       ;* WORD - Subroutine address (floppy MOTor OFF)
                        ;
0018'                   FL2EVT  LABEL   WORD    ;* Floppy disk general timing event
```

```
                =0002           184             DS      2       ;* WORD - Linked list pointer (last entry)
                =0001           185             DS      1       ;* BYTE - Task ID (=0FFH for permanent events)
                =0001           186             DS      1       ;* BYTE - Flags
                =0002           187             DS      2       ;* WORD - Counter
                =0002           188             DS      2       ;* WORD - Subroutine address (FLoppy TIMEing)
                                189     ;
0020'                           190     BELEVT  LABEL   WORD
                =0002           191             DS      2       ; WORD - Linked list pointer
                =0001           192             DS      1       ; BYTE - Task ID (=0FFH for permanent events)
                =0001           193             DS      1       ; BYTE - Flags
                =0002           194             DS      2       ; WORD - Counter
                =0002           195             DS      2       ; WORD - Subroutine address (system BEEPer OFF)
                                196     ;
                                197     ;-------------------------------------------------------------------------------
                                198     ;   CRT DSR data area
                                199     ;
0028'   =0002                  200     DISBEG  DS      2       ; WORD - Begin address of display.
002A'   =0002                  201     DISEND  DS      2       ; WORD - End address of display.
002C'   =0002                  202     CURPOS  DS      2       ; WORD - Address of current cursor position.
002E'   =0002                  203     THSLIN  DS      2       ; WORD - Address of current line of cursor.
0030'   =0002                  204     CRTCNT  DS      2       ; WORD - Number of chars in current line.
0032'   =0002                  205     CURTYP  DS      2       ; WORD - Cursor type (block & 1/4 sec. blink @ pwrup).
0034'   =0002                  206     STATLN  DS      2       ; WORD - TTY status line begin = no stat line @ pwrup.
0036'   =0001                  207     RETFLG  DS      1       ; BYTE - Cond. ret. flag for internal use only !!!
0037'   =0001                  208     CRTERR  DS      1       ; BYTE - Current CRT errors (used by CRTTST).
0038'   =0001                  209     CRTHLD  DS      1       ; BYTE - Status byte for pausing CRT I/O operations.
0039'   =0001                  210     ATTSAV  DS      1       ; BYTE - Attribute latch save.
                                211     ;
                                212     ;-------------------------------------------------------------------------------
                                213     ;   Keyboard DSR data area
                                214     ;
003A'   =0002                  215     QFRONT  DS      2       ; WORD - Pointer to current 'front' of keyboard queue
003C'   =0002                  216     QREAR   DS      2       ; WORD - Pointer to current 'rear' of keyboard queue
003E'   =0001                  217     QDEPTH  DS      1       ; BYTE - Current number of chars waiting in queue
003F'   =0001                  218     KBMODE  DS      1       ; BYTE - Keymode mode/status byte
0040'   =0001                  219     NUMVAL  DS      1       ; BYTE - ALT/NUM accumulator
0041'   =0001                  220     NUMCNT  DS      1       ; BYTE - ALT/NUM keystroke counter
                                221     ;
0042'                           222     QUEUE   LABEL   WORD    ; The keyboard queue itself
                =0020           223             DS      QSIZE*2 ; (*2 because queue entries are words)
0062'                           224     ENDQ    LABEL   WORD
                                225     ;
                                226     ;-------------------------------------------------------------------------------
                                227     ;   Floppy disk DSR data area
                                228     ;
0062'   =0020                  229     D_DIT   DS      4*NUMDRV        ;* DWORD*NUMDRV - Table of disk unit DIT's
0082'   =0001                  230     D_EKNT  DS      1               ;* BYTE - Error counter
0083'   =0001                  231     D_NCMD  DS      1               ;* BYTE - Next disk operation to perform
0084'   =0001                  232     D_NSTA  DS      1               ;* BYTE - Next state for driver
0085'   =0008                  233     D_POS   DS      NUMDRV          ;* BYTE - Table of current disk positions
008D'   =0011 => offset 2B      234     D_REQ   DS      RIBL            ;* BYTE*RIBL - Request Information Block
009E'   =0001                  235     D_SOFT  DS      1               ;* BYTE - Non-zero indicates software interrupt
```

```
                        236                                     ;*        was used to enter DSKISR
009F'  =0001            237  D_STAT  DS      1                  ;* BYTE - Current state of driver
00A0'  =0001            238  D_WAIT  DS      1                  ;* BYTE - Waiting for disk I/O completion flag
00A1'  =0004            239  D_WKSP  DS      4                  ;* BYTE*4 - Disk driver workspace
                        240  ;
       =0062            241  D_DBGN  EQU     D_DIT              ;* Beginning of disk driver data area
       =00A5            242  D_DEND  EQU     D_WKSP+4           ;* End of disk driver data area
                        243  ;
                        244  ;-------------------------------------------------------------------------
       =0001            245          DS      1                  ; This forces ROMDAT size to be multiple of 16
                        246  ;                                   ; and stacks to be on an even word boundary
                        247  ;-------------------------------------------------------------------------
                        248  ;   Interrupt handler stack save areas:
                        249  ;
00A6'  =0002            250  KBSSSV  DS      2                  ; Keyboard interrupt stack segment save
00A8'  =0002            251  KBSPSV  DS      2                  ; Keyboard interrupt stack pointer save
                        252  ;
00AA'  =0002            253  DKSSSV  DS      2                  ; Disk interrupt stack segment save
00AC'  =0002            254  DKSPSV  DS      2                  ; Disk interrupt stack pointer save
                        255  ;
00AE'  =0002            256  TMSSSV  DS      2                  ; Timer interrupt stack segment save
00B0'  =0002            257  TMSPSV  DS      2                  ; Timer interrupt stack pointer save
                        258  ;
00B2'  =0002            259  IXSSSV  DS      2                  ; Common interrupt exit stack segment save
00B4'  =0002            260  IXSPSV  DS      2                  ; Common interrupt exit stack pointer save
                        261  ;
       =0030            262          DS      48                 ; Keyboard interrupt service routine's stack
00E6'                   263  KEYISP  LABEL   WORD
                        264  ;
       =0030            265          DS      48                 ;* Disk interrupt service routine's stack
0116'                   266  DSKISP  LABEL   WORD
                        267  ;
       =0024            268          DS      36                 ; System time interrupt routine's stack
013A'                   269  TIMISP  LABEL   WORD
                        270  ;
                        271  ;-------------------------------------------------------------------------
                        272  ;   Time-of-day clock DSR data area
                        273  ;
                        274  ;   *** HUNS thru DATE must be contiguous ***
                        275  ;
013A'  =0001            276  HUNS    DS      1                  ; Hundredths of seconds (tenth sec resolution)
013B'  =0001            277  SECS    DS      1                  ; Seconds
013C'  =0001            278  MINS    DS      1                  ; Minutes
013D'  =0001            279  HOURS   DS      1                  ; Hours
013E'  =0002            280  DATE    DS      2                  ; Days since 1-1-80
                        281  ;
0140'                   282  RMDTEN  LABEL   BYTE               ; Last location of ROMDAT
       =0140            283  RMDTSZ  EQU     RMDTEN-RMDTBG      ; The size of ROMDAT
                        284  ;
       =0000            285      IF  (RMDTSZ MOD 10H) NE 0000
                        286          BARFO                  ; Blow up if ROMDAT size is not a multiple of 10H
                        287      ENDIF
```

```
                                288     ;
                                289             END
```

No errors detected

```
                        1   ;*******************************************************************************
                        2   ; TITLE    - ROMERR (ROM SYSTEM ERROR CODES)
                        3   ; ABSTRACT - This module contains the definitions of ROM based system
                        4   ;       error codes.
                        5   ;*******************************************************************************
                        6           NAME    ROMERR (ROM BASED SYSTEM ERROR CODES)
                        7           SUBTTL
                        9   ;*******************************************************************************
                       10   ;                           PUBLIC DEFINITIONS
                       11   ;*******************************************************************************
                       12           PUBLIC  DBCERR          ; Boot - Bad CRC on boot sector
                       13           PUBLIC  DCRERR          ; Boot - CRC error
                       14           PUBLIC  DFMERR          ; Boot - Disk format error
                       15           PUBLIC  DNIERR          ; Boot - No drives installed
                       16           PUBLIC  DNRERR          ; Boot - Disk not ready
                       17           PUBLIC  DNSERR          ; Boot - Not a TI system disk
                       18           PUBLIC  DSKERR          ; Boot - Seek error
                       19           PUBLIC  DSNERR          ; Boot - Sector-not-found error
                       20           PUBLIC  DUNERR          ; Boot - Data error (should not occur)
                       21           PUBLIC  DDMERR          ; Boot - DRQ error (hardware failure)
                       22           PUBLIC  FATERR          ; Fatal software error
                       23           PUBLIC  KNIERR          ; Keyboard not installed
                       24           PUBLIC  KNRERR          ; Keyboard - no response
                       25           PUBLIC  KRAERR          ; Keyboard RAM failure
                       26           PUBLIC  KRCERR          ; Keyboard acknowledge char receipt error
                       27           PUBLIC  KROERR          ; Keyboard ROM failure
                       28           PUBLIC  KUNERR          ; Keyboard - Unknown error
                       29           PUBLIC  LED111          ; All LEDs on, complete system fail.
                       30           PUBLIC  LED110          ; System ROM failure
                       31           PUBLIC  LED101          ; System RAM failure (P port CONTAINS BITS BAD)
                       32           PUBLIC  LED100          ; Interrupt or Timer failure
                       33           PUBLIC  LED011          ; Floppy controller failure
                       34           PUBLIC  LED010          ; CRT hardware failure
                       35           PUBLIC  LED001          ; Option failure (used by PUPNMI)
                       36           PUBLIC  LED000          ; Basic system test passed
                       37           PUBLIC  OPRERR          ; Optional RAM has an error during powerup test
                       38           PUBLIC  PURMPE          ; RAM parity error during powerup
                       39           PUBLIC  PUXNME          ; Unexpected NMI during powerup
                       40           PUBLIC  RMPERR          ; RAM parity error
                       41           PUBLIC  ROMERX          ; Option ROM at 0F4000H is in error
                       42           PUBLIC  XNMERR          ; Unexpected NMI
                       43           PUBLIC  WLDERR          ; 'Wild' interrupt
                       44
                       45   ;*******************************************************************************
                       46   ;       POWERUP AND DIE ERROR CODES - '00xx'
                       47   ;       'xx' in the range  00H --) 09H
                       48   ;
                       49   ;       Note that LED bits are inverted, since LEDs are low = 'on'
                       50   ;*******************************************************************************
=0000                  51   LED111  EQU     00000000B       ; "7" All LEDs on, complete system fail.
=0001                  52   LED110  EQU     00000001B       ; "6" System ROM failure
=0002                  53   LED101  EQU     00000010B       ; "5" System RAM failure
```

```
=0003                    54   LED100  EQU     00000011B       ; "4" Interrupt or timers failed
=0004                    55   LED011  EQU     00000100B       ; "3" Floppy controller failed
=0005                    56   LED010  EQU     00000101B       ; "2" CRT hardware failed
=0006                    57   LED001  EQU     00000110B       ; "1" Keyboard failed (not used yet)
=0007                    58   LED000  EQU     00000111B       ; "0" Basic system test passed
                         59   ;
=0008                    60   PUXNME  EQU     8               ; Unexpected NMI during powerup sequence
=0009                    61   PURMPE  EQU     9               ; RAM parity error during powerup sequence
                         62   ;
                         63   ;
                         64   ;*****************************************************************************
                         65   ;       KEYBOARD ERRORS - '00xx'
                         66   ;       'xx' in the range 10H --) 19H
                         67   ;*****************************************************************************
                         68   ;
=0010                    69   KNIERR  EQU     10H             ; Keyboard not installed
=0011                    70   KNRERR  EQU     11H             ; Keyboard - installed but no response
=0012                    71   KRAERR  EQU     12H             ; Keyboard RAM failure
=0013                    72   KROERR  EQU     13H             ; Keyboard ROM failure
=0014                    73   KUNERR  EQU     14H             ; Keyboard Unknown failure(unexpected response)
=0015                    74   KRCERR  EQU     15H             ; Keyboard acknowledge char receipt error
                         75   ;
                         76   ;
                         77   ;*****************************************************************************
                         78   ;       OPTIONAL MEMORY ERRORS - '00xx'
                         79   ;       'xx' in the range 20H --) 2FH
                         80   ;*****************************************************************************
=0020                    81   OPRERR  EQU     20H             ; Option RAM bank 1 has an error in puptst.
                         82   ;       EQU     21H             ; Option RAM bank 2 has an error in puptst.
                         83   ;       EQU     22H             ; Option RAM bank 3 has an error in puptst.
                         84   ;       EQU     23H             ; Option RAM bank 4 has an error in puptst.
                         85   ;       EQU     24H             ; Option RAM bank 5 has an error in puptst.
                         86   ;       EQU     25H             ; Option RAM bank 6 has an error in puptst.
                         87   ;       EQU     26H             ; Option RAM bank 7 has an error in puptst.
=0028                    88   ROMERX  EQU     28H             ; Option ROM at OF4000H is in error
                         89   ;       EQU     29H             ; Option ROM at OF6000H is in error
                         90   ;       EQU     2AH             ; Option ROM at OF8000H is in error
                         91   ;       EQU     2BH             ; Option ROM at OFA000H is in error
                         92   ;       EQU     2CH             ; Option ROM at OFC000H is in error
                         93   ;       EQU     2DH             ; MAIN ROM at OFE000H is in error
                         94
                         95   ;*****************************************************************************
                         96   ;       DISK BOOT ERRORS - '00xx'
                         97   ;       'xx' in the range 30H --) 3FH
                         98   ;*****************************************************************************
=0030                    99   DNIERR  EQU     30H             ; No drives installed (must have at least one)
=0031                   100   DNRERR  EQU     31H             ; Disks not ready or not bootable
=0032                   101   DCRERR  EQU     32H             ; CRC error
=0033                   102   DSKERR  EQU     33H             ; Seek error
=0034                   103   DSNERR  EQU     34H             ; Sector-not-found error
=0035                   104   DUNERR  EQU     35H             ; 'Disk' (unknown) error (controller failure)
=0036                   105   DNSERR  EQU     36H             ; Not a TI system disk
```

```
=0037                   106   DFMERR  EQU     37H             ; Disk format error
=0038                   107   DBCERR  EQU     38H             ; Bad boot sector CRC or bad sector buffer I/F
=0039                   108   DDMERR  EQU     39H             ; DRQ error (controller failure)
                        109   ;
                        110   ;
                        111   ;*****************************************************************************
                        112   ;       INTERRUPT ERROR CODES - '10xx'
                        113   ;       'xx' in the range 40H --) 4FH
                        114   ;*****************************************************************************
=0040                   115   XNMERR  EQU     40H             ; Unexpected non-maskable interrupt (NMI)
=0041                   116   RMPERR  EQU     41H             ; RAM parity error detected during operation.
=0042                   117   WLDERR  EQU     42H             ; Unexpected (Wild) interrupt
                        118   ;
                        119   ;
                        120   ;*****************************************************************************
                        121   ;       SOFTWARE PROBLEM ERROR CODES - '10xx'
                        122   ;       'xx' in the range 50H --) 5FH
                        123   ;*****************************************************************************
=0050                   124   FATERR  EQU     50H             ; Fatal software bug encountered
                        125   ;
                        126   ;
                        127   ;*****************************************************************************
                        128   ;       RESERVED ERROR CODES
                        129   ;*****************************************************************************
                        130   ;
=00FF                   131   UNUSED  EQU     0FFH            ; Do not use 0FFH as an error code
                        132   ;
                        133           SUBTTL
                        134           END
```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ; TITLE    - ROMTIM (ROM TIMING SERVICES)
                        3   ; COMPUTER - 8088 ASSEMBLY LANGUAGE
                        4   ; ABSTRACT - The ROMTIM subroutine is called by the system timer interrupt
                        5   ;       processor every 100ms (every 4 clock ticks).  This routine scans
                        6   ;       the list of fixed timing events to determine if any events have
                        7   ;       expired.  When an event does expire, the subroutine defined in
                        8   ;       the event structure is called.
                        9   ;
                       10   ; INPUT    - DS = ROM DATA SEGMENT
                       11   ;
                       12   ; OUTPUT   - NONE
                       13   ;
                       14   ; REGISTERS USED - AX,DI,DS,FLAGS
                       15   ;
                       16   ; STACK USED - 6 BYTES
                       17   ;
                       18   ;*****************************************************************************
                       19           NAME    ROMTIM (ROM TIMING SERVICES SUBROUTINE)
                       20           SUBTTL
                       22   ;*****************************************************************************
                       23   ;                           PUBLIC DEFINITIONS
                       24   ;*****************************************************************************
                       25
                       26           PUBLIC  ROMTIM
                       27
                       28   ;*****************************************************************************
                       29   ;                           EXTERNAL REFERENCES
                       30   ;*****************************************************************************
                       31
                       32
=0000                  33           SECT    ROMDAT
                       34           EXTERN  ROMEVT:WORD     ;       POINTER TO 1ST EVENT IN LIST   (ROMDAT)
                       35   ;*****************************************************************************
                       36   ;                           LOCAL CONSTANTS
                       37   ;
                       38   ;       INCLUDE PEQ:TIMEVT.EQU
                       39   ;*****************************************************************************
                       59   ;*****************************************************************************
                       60   ;                           LOCAL MACROS
                       61   ;*****************************************************************************
                       62
                       63           INCLUDE PEQ:BIT.MAC
                       89
                       90   BIT     MACRO   %BITNUM,%TARGET
                       91           TEST    BYTE PTR %TARGET,1 SHL %BITNUM  ;  TEST BIT, GET Z-FLAG
                       92           ENDM
                       93
                       94           INCLUDE PEQ:RES.MAC
                      116
                      117   RES     MACRO   %BITNUM,%TARGET
                      118           PUSHF                           ;       SAVE FLAGS
```

```
                        119             AND     BYTE PTR %TARGET,NOT(1 SHL %BITNUM) ; RESET THE BIT
                        120             POPF                                ;       RESTORE FLAGS
                        121             ENDM
                        122
```

```
                                124             SUBTTL  MAIN
                                125     ;*******************************************************************************
                                126     ;                               MODULE ENTRY POINT
                                127     ;*******************************************************************************
                                128
          =0000                 129             SECT    ROMCOD
                                130             ASSUME  CS:ROMCOD
                                131             ASSUME  DS:ROMDAT
                                132
0000'                           133     ROMTIM  PROC    NEAR                    ;;      DS = ROM DATA SEGMENT
0000'  BF 0001"                 134             MOV     DI,OFFSET ROMEVT        ;;      DS,DI = ADDR PTR TO 1ST EVENT
0003'                           135     NXTEVT:                                 ;;      DS,DI = ADDR PTR TO NEXT EVENT
0003'  8B 3D                    136             MOV     DI,[DI]                 ;       DS,DI = ADDR OF POSSIBLE EVENT
0005'  09 FF                    137             OR      DI,DI                   ;Q:     END OF LIST ENCOUNTERED ?
0007'  74 1A                    138             JZ      EXIT                    ;; Y:   DONE
                                139                                             ;; N:   DS,DI = ADDR OF EVENT
                                140+            BIT     EVTSTA,IEVFLG[DI]       ;;Q:    IS THIS EVENT ACTIVE ?
000D'  74 F4                    142             JZ      NXTEVT                  ;; N:   GET NEXT EVENT
000F'  FF 4D 04                 143             DEC     WORD PTR IEVCTR[DI]     ;; Y:   DECREMENT EVENT'S COUNTER
                                144                                             ;;Q:    EVENT EXPIRED ?
0012'  75 EF                    145             JNZ     NXTEVT                  ;; N:   GET NEXT EVENT
                                146+            RES     EVTSTA,IEVFLG[DI]       ;; Y:   NEW STATUS = INACTIVE
001A'  1E                       150             PUSH    DS                      ;;      SAVE EVENT ADDRESS
001B'  57                       151             PUSH    DI                      ;;
001C'  FF 55 06                 152             CALL    WORD PTR IEVSUB[DI]     ;;      CALL EVENT'S SUBROUTINE
001F'  5F                       153             POP     DI                      ;;
0020'  1F                       154             POP     DS                      ;;      DS,DI = EVENT ADDRESS
0021'  EB E0                    155             JMP     NXTEVT                  ;;      GET NEXT EVENT
                                156
0023'                           157     EXIT:                                   ;;      SCAN COMPLETE
0023'  C3                       158             RET                             ;;      ** RETURN **
                                159
                                160             SUBTTL
                                161             END
```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ;  TITLE    - ROMUTL - System ROM utility routines
                        3   ;  COMPUTER - 8088 Assembly Language
                        4   ;  ABSTRACT - This module contains several commonly used subroutines
                        5   ;*****************************************************************************
                        6           NAME    ROMUTL - System ROM utility routines
                        7   ;       SUBTTL
                        9   ;*****************************************************************************
                       10   ;                           PUBLIC DEFINITIONS
                       11   ;*****************************************************************************
                       12   ;
                       13           PUBLIC  DECLED
                       14           PUBLIC  DELAY
                       15   ;
                       16   ;*****************************************************************************
                       17   ;                           EXTERNAL REFERENCES
                       18   ;*****************************************************************************
                       19   ;
=0000                  20           SECT    ROMDAT
                       21           EXTRN   PRCO_M:BYTE     ; Parallel control port memory (ROMDAT)
                       22   ;
                       23   ;*****************************************************************************
                       24   ;                           LOCAL CONSTANTS
                       25   ;       INCLUDE PEQ:PORTADDR.EQU
                       26   ;       INCLUDE PEQ:VECTOR.EQU
                       27   ;*****************************************************************************
                       28   ;
                      134   ;*****************************************************************************
                      135   ;                           CODE SEGMENT DEFINITION
                      136   ;*****************************************************************************
                      137   ;
=0000                 138           SECT    ROMCOD
                      139           ASSUME  CS:ROMCOD
                      140   ;
                      141   ;*****************************************************************************
                      142   ;                           MODULE ENTRY POINT
                      143   ;*****************************************************************************
                      144   ;
                      145   ;   SETPRT - Initialize data ports according to table (in the current CS)
                      146   ;       The format of the table is:
                      147   ;
                      148   ;               DB      (port),(data)
                      149   ;               DB      OFFH                    ; Table delimiter
                      150   ;
                      151   ;       Where port is an 8-bit port address.  End of table denoted by a
                      152   ;       port value of OFFH.  It is recommended this routine be executed
                      153   ;       with interrupts disabled.  This method saves 1 byte over
                      154   ;       conventional "MOV  AL,XX;  OUT  XX,AL" initialization if the
                      155   ;       number of ports to be initialized is 4 (9-byte table).  There
                      156   ;       are corresponding savings for larger tables
                      157   ;
                      158   ;   INPUT:  CS:SI = Address of table
```

```
                        ;   OUTPUT: (none) ports are initialized
                        ;   USED:
                        ;   STACK:  2
                        ;    ASSUME  CS:ROMCOD
                        ;
                        ;SETPRT PROC    NEAR
                        ;       MOV     DH,00                   ; Ports are below 100H
                        ;SP1:
                        ;       LODS    BYTE PTR CS:[SI]        ; Get the port
                        ;       CMP     AL,0FFH                 ; Q: End of the list ?
                        ;       JZ      SP2                     ;   Y: That's all, then
                        ;       MOV     DL,AL                   ;   N: Set up DX to point to port
                        ;       LODS    BYTE PTR CS:[SI]        ; Get the data
                        ;       OUT     DX,AL                   ; Spit it out
                        ;       LOOP    SP1                     ; Do 'em all
                        ;SP2:
                        ;       RET                             ; *** RETURN ***

                        ;*****************************************************************************
                        ;
                        ;   DELAY - delay for (CX) milliseconds
                        ;
                        ;   INPUT:  CX = length of delay in milliseconds
                        ;   OUTPUT: (none)
                        ;   USED:   CX
                        ;   STACK:  4
                        ;       ASSUME  CS:ROMCOD
                        ;
0000'                   DELAY   PROC    NEAR
0000'  50                       PUSH    AX
0001'                   DEL1:
0001'  B0 A0                    MOV     AL,0A0H         ; Empirically derived using MDS timer
0003'                   DEL2:
0003'  FE C8                    DEC     AL              ;
0005'  75 FC                    JNZ     DEL2            ; if not 1 msec
0007'  E2 F8                    LOOP    DEL1            ; if not desired count
0009'  58                       POP     AX
000A'  C3                       RET                     ; *** RETURN ***

                        ;
                        ;*****************************************************************************
                        ;
                        ;   DECLED - This subroutine reads the state of the LED's from RAM,
                        ;            decrements it, writes it back to RAM and outputs the
                        ;            new state to the LED port.
                        ;
                        ;   INPUT - (none)
                        ;
                        ;   OUTPUT - AL = New state of LED'S (i.e. PRCO_M-1)
                        ;            LED'S decremented and LED ram byte updated
                        ;
                        ;   USED - AL
```

```
                        211   ;
                        212   ;    STACK - NONE
                        213   ;
                        214   ;*******************************************************************************
                        215   ;
                        216           ASSUME  CS:ROMCOD
                        217   ;
000B'                   218   DECLED  PROC    NEAR
000B'  2E8E 1E C180     219           MOV     DS,WORD PTR CS:DSADDR+CSWRAP    ; Set up DS as ROMDAT
0010'  9C               220           PUSHF                                   ; Save callers flags (interrupt state)
0011'  FA               221           CLI                                     ; Interrupts off
0012'  A0 0001"         222           MOV     AL,PRCO_M                       ; Get copy of LED's from RAM
0015'  8A E0             223           MOV     AH,AL                           ; Save LED state in AH
0017'  24 07             224           AND     AL,07H                          ; Mask off last three bits
0019'  3C 07             225           CMP     AL,07H                          ; Q: Are they all "zero" ?
001B'  74 09             226           JE      NODEC                           ; Y: Then return (no decrement)
001D'  8A C4             227           MOV     AL,AH                           ; N: Get copy of LED's back
                        228   ;       CALL    SWPLED                          ; Swap the LED's for the INC
001F'  FE C0             229           INC     AL                              ; ...and decrement LED's (inverted)
                        230   ;       CALL    SWPLED                          ; Go SWAP bits 0 & 2
0021'  E6 03             231           OUT     PRCO_P,AL                       ; Output new LED's
0023'  A2 0001"          232           MOV     PRCO_M,AL                       ; Save new LED state in RAM
                        233   ;
0026'                   234   NODEC:                                          ;
0026'  9D               235           POPF                                    ; Restore callers flags
0027'  C3               236           RET                                     ; All done !!!
                        237   ;
                        238   ;*******************************************************************************
                        239   ;       THIS ROUTINE IS USED TO SWAP BITS 0 & 2 IN SOFTWARE
                        240   ;
                        241   ;       INPUT - AL = STATE OF THE LED'S
                        242   ;
                        243   ;       OUTPUT - AL = CORRECT STATE OF THE LED'S
                        244   ;
                        245   ;*******************************************************************************
                        246   ;
                        247   ;SWPLED PROC    NEAR                            ;
                        248   ;       MOV     BL,AL                           ; SAVE LED'S
                        249   ;       XOR     CL,CL                           ; CLEAR CL
                        250   ;       AND     BL,7H                           ; MASK OUT LAST THREE BITS
                        251   ;       TEST    BL,2H                           ; Q: IS SECOND BIT SET ?
                        252   ;       JNZ     KEEP1                           ; Y: THEN KEEP IT A ONE.
                        253   ;TEST0: TEST    BL,1H                           ; Q: IS FIRST BIT SET ?
                        254   ;       JNZ     SWAP0                           ; Y: THEN SWAP BIT 0 WITH BIT 2
                        255   ;TEST2: TEST    BL,4H                           ; Q: IS THIRD BIT SET ?
                        256   ;       JNZ     SWAP2                           ; Y: THEN SWAP BIT 2 WITH BIT 0
                        257   ;UPBITS:        AND     AL,0F8H                         ; CLEAR LAST THREE BITS
                        258   ;       OR      AL,CL                           ; OR IN NEW CORRECT LED'S
                        259   ;       RET                                     ; ALL DONE
                        260   ;
                        261   ;KEEP1: OR      CL,00000010B                    ; KEEP SECOND BIT(1) A ONE
                        262   ;       JMP     TEST0                           ; GO TEST THE OTHER BITS
```

```
                        263     ;
                        264     ;SWAP0:  OR      CL,00000100B            ; CHANGE THIRD BIT(2) TO A ONE
                        265     ;        JMP     TEST2                   ; GO TEST BIT THREE
                        266     ;
                        267     ;SWAP2:  OR      CL,00000001B            ; CHANGE FIRST BIT(0) TO A ONE
                        268     ;        JMP     UPBITS                  ; GO UPDATE THE NEW BITS
                        269     ;
                        270             END
```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ;  TITLE    - SYSORG - System organization module
                        3   ;  COMPUTER - 8088 Assembly Language  (BSO Assembler)
                        4   ;  ABSTRACT - This module defines the 'layout' of the Pegasus system memory.
                        5   ;*****************************************************************************
                        6           NAME    SYSORG - System organization module
                        7   ;       SUBTTL
                        9   ;*****************************************************************************
                       10   ;                               PUBLIC DEFINITIONS
                       11   ;*****************************************************************************
                       12   ;
                       13           PUBLIC  BOOTLO
                       14           PUBLIC  BOOTMV
                       15           PUBLIC  BOOTSZ
                       16           PUBLIC  EPROM
                       17           PUBLIC  ROMF4
                       18           PUBLIC  ROMF6
                       19           PUBLIC  ROMF8
                       20           PUBLIC  ROMFA
                       21           PUBLIC  ROMSIZ
                       22           PUBLIC  SYSROM
                       23   ;
                       24   ;*****************************************************************************
                       25   ;                               LOCAL CONSTANTS
                       26   ;*****************************************************************************
                       27   ;
=0200                  28   BOOTSZ  EQU     200H            ; Max size of boot sector loaded at BOOTLO
=1820                  29   CRTMSZ  EQU     1820H           ; Size of memory that Alpha board talks to
=2000                  30   ROMSIZ  EQU     2000H           ; ROM boundaries are 8K
                       31
                       32   ;*****************************************************************************
                       33   ;                               SEGMENT DEFINITION
                       34   ;*****************************************************************************
                       35   ;
                       36   ;   ABS0 is located at absolute memory address 0.  It is used during system
                       37   ;   initialization and diagnostics to access system constants and interrupt
                       38   ;   vectors in low RAM (see VECTOR.EQU).
                       39   ;
=0000                  40           SECT    ABS0
                       41           ASSUME  CS:ABS0, DS:ABS0
                       42   ;
=C000                  43           ORG     0C000H          ; BOOT SECTOR LOADED HERE
                       44   ;
C000'                  45   BOOTMV  LABEL   BYTE            ; This label used as ref to load boot sector
                       46                                   ;   (also as base for ROM init stack)
C000'                  47   BOOTLO  PROC    FAR             ; DKBOOT jumps to here after boot is loaded
=0200                  48           DS      BOOTSZ          ; space
                       49   ;
                       50   ;-----------------------------------------------------------------------------
                       51   ;
                       52   ;   ROMDAT is the dedicated RAM area allocated to the 8K System ROM on the
                       53   ;   main board.  The optional ROMs are each allocated a separate data area.
```

```
                        ;   There is a set of words in the interrupt vector area of memory (ABS0),
                        ;   that contains the location and size of the blocks.  Each ROM has its own
                        ;   2 words (see VECTOR.EQU).  The location DSADDR contains a pointer to
                        ;   ROMDAT, and the location DSSIZR contains the size (in bytes) of ROMDAT.
                        ;   The other locations (DSADDx and DSSIZx) contain corresponding pointers
                        ;   and sizes for the option ROMs.
                        ;
                        ;   These values (except for DSADDR/DSSIZR) are initialized to zero during
                        ;   System ROM powerup.  Each optional ROM is responsible for allocating
                        ;   its own data area when it is called from PUPTSI.  This is accomplished by
                        ;   looking at the DSSIZx locations prior to the current one, scanning
                        ;   backwards until a non-zero DSSIZx is encountered.  This value is adjusted
                        ;   and added to the corresponding DSADDx to form the next contiguous segment
                        ;   address and the current DSSIZx is filled in.  The size must be a multiple
                        ;   of 16, i.e. the low order 4 bits must be zero.  This is for ease of
                        ;   conversion to segment addresses for calculating total RAM data area size,
                        ;   etc.  Note that the ROM initialization routines are called in the order
                        ;   0,2,4,6,8 where the numbers correspond to the ROMs at offsets of 0000,
                        ;   2000, 4000, 6000, and 8000 from the beginning of ROMCOD (the System ROM
                        ;   is at offset A000).
                        ;
                        ;   The memory is allocated sequentially from DSADDR such that the data area
                        ;   for the entire set of ROMs in ROMCOD (the aforementioned optional ROMs)
                        ;   begins at DSADDR.  The total size of the data area is determined by
                        ;   adding all the DSSIZx values together, since each size has been rounded
                        ;   up to the nearest segment.  This implies that the values in the DSSIZx
                        ;   locations for non-installed ROMs must be zero.
                        ;
                        ;   Although ROMDAT is initially loaded here, the boot routine may change its
                        ;   location to anywhere within the 1 Mbyte address range of the processor.
                        ;   This is done by changing all the DSADDx locations to point to the new
                        ;   locations (by adding/subtracting a constant) and then copying the data
                        ;   from DSADDR to the new location.  The length for the move is determined
                        ;   as shown above, being the total size of allocated RAM area.  Interrupts
                        ;   should be disabled during this process.
                        ;
                        ;   NOTE:  Remember that the DSADDx values are segment values, while the
                        ;          DSSIZx values are in bytes.
                        ;
=0000                       SECT    ROMDAT
                            ASSUME  CS:ROMDAT
                        ;
0000'                   INRMDT  LABEL   BYTE            ; ROMDAT data initially loaded here.
                        ;
                        ;-------------------------------------------------------------------------
                        ;
                        ;   CRTDAT is the Alpha-board screen RAM.
                        ;
=0000                       SECT    CRTDAT
                            ASSUME  CS:CRTDAT
                        ;
=1820                       DS      CRTMSZ
```

```
                        106
                        107   ;-----------------------------------------------------------------------
                        108   ;
                        109   ;   ROMCOD is located at absolute memory address 0F4000H.  Although the
                        110   ;   system ROM is located at 0FC000H, this leaves space for option ROM to
                        111   ;   be in the same segment as the system ROM... useful, for example, if
                        112   ;   BASIC were to be put in ROM.  Note that this is 48K from the end of RAM,
                        113   ;   allowing direct access to the first 16K of RAM by taking advantage of
                        114   ;   the fact that the hardware 'wraps' memory (the address following
                        115   ;   0FFFFFH is 00000H).  This memory is accessed by adding the offset
                        116   ;   (from absolute zero) to the constant CSWRAP defined in VECTOR.EQU.
                        117   ;
        =0000           118           SECT    ROMCOD
                        119           ASSUME  CS:ROMCOD
                        120   ;
                        121   ;   The following labels correspond to the starting locations for the option
                        122   ;   ROMs.  They are used during system initialization to determine the ROM's
                        123   ;   presence.  Note that the ROMs are assumed to be on 8K boundaries.
                        124   ;
0000'                   125   ROMF4   LABEL   BYTE            ; ROM0 (no socket on main board; future option)
        =2000           126           DB      ROM8IZ
                        127   ;
2000'                   128   ROMF6   LABEL   BYTE            ; ROM2 (no socket on main board; future option)
        =2000           129           DB      ROM8IZ
                        130   ;
4000'                   131   ROMF8   LABEL   BYTE            ; ROM4 (no socket on main board; future option)
        =2000           132           DB      ROM8IZ
                        133   ;
6000'                   134   ROMFA   LABEL   BYTE            ; ROM6 (no socket on main board; future option)
        =2000           135           DB      ROM8IZ
                        136   ;
8000'                   137   EPROM   LABEL   BYTE            ; ROM8 Starting loc of main board option ROM
        =2000           138           DB      ROM8IZ
                        139   ;
A000'                   140   SYSROM  LABEL   BYTE            ; Starting loc of system ROM
                        141
                        142   ;-----------------------------------------------------------------------
                        143   ;
                        144   ;   RESET is located at absolute memory location 0FFFF0H.  This is where the
                        145   ;   processor starts after being reset (with CS = FFFFH, other seg-regs = 0)
                        146   ;
        =0000           147           SECT    RESET
                        148           ASSUME  CS:RESET
                        149   ;
                        150           END
```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ; TITLE    - TIMISR (SYSTEM TIMER INTERRUPT SERVICE ROUTINE)
                        3   ; COMPUTER - 8088 ASSEMBLY LANGUAGE
                        4   ; ABSTRACT - The system timer interrupt service routine logic is executed
                        5   ;       each time a clock tick occurs (25ms).  Register DS is saved on the
                        6   ;       stack of the interrupted logic.  The timer interrupt processor
                        7   ;       stack is then used to save registers that are commonly used by
                        8   ;       interrupt processors (AX,BX,and ES) and registers that are used
                        9   ;       locally.
                       10   ;
                       11   ;       At 25ms intervals (each tick), the ROMTIM subroutine is called
                       12   ;       to perform timing services for the fixed events defined in ROM.
                       13   ;       A tick counter is incremented and checked to see if 100ms has
                       14   ;       elapsed.  If so, the time of day is updated by calling CLKSRV.
                       15   ;       If ZEX is present in the system, TIMISR does soft interrupts to
                       16   ;       ZEX logic to perform timing services for ZEX (dynamic) timing
                       17   ;       events and to do time slicing operations.  If ZEX is not in
                       18   ;       the system, the soft interrupt logic simply does an IRET.
                       19   ;
                       20   ;       On exit, this logic restores all registers used locally from the
                       21   ;       interrupt processor stack and jumps to the common interrupt exit
                       22   ;       logic, passing the location of the interrupted logic's stack in
                       23   ;       registers ES and BX.
                       24   ;
                       25   ; REGISTERS USED - none
                       26   ;
                       27   ; STACK USED -  8 bytes (user stack)
                       28   ;              16 bytes (interrupt stack)
                       29   ;
                       30   ;*****************************************************************************
                       31           NAME    TIMISR
                       32           SUBTTL
                       34   ;*****************************************************************************
                       35   ;                           PUBLIC DEFINITIONS
                       36   ;*****************************************************************************
                       37
                       38           PUBLIC  TIMISR
                       39
                       40   ;*****************************************************************************
                       41   ;                           EXTERNAL REFERENCES
                       42   ;*****************************************************************************
                       43
=0000                  44           SECT    ROMCOD
                       45           EXTERN  CLKSRV:NEAR     ;       CLOCK SERVICES                  (CLKDSR)
                       46           EXTERN  ROMTIM:NEAR     ;       TIMING SERVICES                 (ROMTIM)
                       47
=0000                  48           SECT    ROMDAT
                       49           EXTERN  DSKC_M:BYTE     ;       MEMORY COPY OF TICK CTR LATCH   (ROMDAT)
                       50           EXTERN  TIKCTR:BYTE     ;       TICK COUNTER                    (ROMDAT)
                       51           EXTERN  TIMISP:WORD     ;       TIMER INTERRUPT STACK SP        (ROMDAT)
                       52           EXTERN  TMSPSV:WORD     ;       WORD TO SAVE USER SP            (ROMDAT)
                       53           EXTERN  TMSSSV:WORD     ;       WORD TO SAVE USER SS            (ROMDAT)
```

```
                                54
                                55  ;*****************************************************************************
                                56  ;                           LOCAL CONSTANTS
                                57  ;       INCLUDE PEQ:VECTOR.EQU
                                58  ;       INCLUDE PEQ:PORTADDR.EQU
                                59  ;       INCLUDE PEQ:LATCHES.EQU
                                60  ;*****************************************************************************
                                61
        =0004                   62  LIMIT   EQU     4               ;       NUMBER OF TICKS TO MAKE TIMING
                                63                                  ;         SERVICES, CLOCK SERVICES RESOLUTION
                               218
                               219          SUBTTL  MAIN
                               220  ;*****************************************************************************
                               221  ;                           MODULE ENTRY POINT
                               222  ;*****************************************************************************
                               223
        =0000                  224          SECT    ROMCOD
                               225          ASSUME  CS:ROMCOD
                               226          ASSUME  DS:ROMDAT
                               227
0000'                          228  TIMISR  PROC    FAR                         ;;;
0000'  1E                      229          PUSH    DS                          ;;;     SAVE CURRENT DS ON USER STACK
0001'  2EFE 06 C19A            230          INC     BYTE PTR CS:INTCTR+CSWRAP;;;    INCREMENT INTERRUPT COUNTER
0006'  2E8E 1E C180            231          MOV     DS,WORD PTR CS:DSADDR+CSWRAP;;; DS = ROM DATA SEGMENT
000B'  8C 16 0007"             232          MOV     TMSSSV,SS                   ;;;     SAVE CURRENT SS,SP
000F'  89 26 0006"             233          MOV     TMSPSV,SP                   ;;;
0013'  8C DC                   234          MOV     SP,DS                       ;;;     STACK SEG = ROM DATA SEGMENT
0015'  8E D4                   235          MOV     SS,SP                       ;;;
0017'  BC 0005"                236          MOV     SP,OFFSET TIMISP            ;;;     SS,SP = INTERRUPT STACK PTR
001A'  50                      237          PUSH    AX                          ;;;     SAVE COMMONLY USED REGISTERS
001B'  53                      238          PUSH    BX                          ;;;       ON INTERRUPT STACK
001C'  06                      239          PUSH    ES                          ;;;
001D'  A0 0003"                240          MOV     AL,DSKC_M                   ;;;     AL = COPY OF LATCH FROM MEMORY
0020'  24 FD                   241          AND     AL,NOT TMR1EN               ;;;
0022'  E6 00                   242          OUT     DSKC_P,AL                   ;;;     CLEAR THE TIMER INTERRUPT
0024'  34 02                   243          XOR     AL,TMR1EN                   ;;;
0026'  E6 00                   244          OUT     DSKC_P,AL                   ;;;
0028'  FB                      245          STI                                 ;;;     ENABLE INTERRUPTS
0029'  57                      246          PUSH    DI                          ;;;     SAVE REGISTERS USED LOCALLY
                               247
002A'  E8 0002"                248          CALL    ROMTIM                      ;       HANDLE FIXED TIMING SERVICES
002D'  FE 06 0004"             249          INC     TIKCTR                      ;       INCREMENT TICK COUNTER
0031'  80 3E 0004" 04          250          CMP     TIKCTR,LIMIT                ;Q:     REACH TIMING INTERVAL LIMIT ?
0036'  75 0A                   251          JNZ     NOTYET                      ; N:    NO TIMING OR CLOCK SERVICES YET
0038'  C6 06 0004" 00          252          MOV     TIKCTR,0                    ; Y:    RESET TICK COUNTER
003D'  E8 0001"                253          CALL    CLKSRV                      ;       UPDATE TIME OF DAY
0040'  CD 5A                   254          INT     TIMINT                      ;       HANDLE DYNAMIC TIMING SERVICES
0042'                          255  NOTYET:                                     ;
0042'  CD 58                   256          INT     SLCINT                      ;       DO TIME-SLICING IF NEEDED
                               257
0044'  5F                      258          POP     DI                          ;       RESTORE REGISTERS USED LOCALLY
0045'  2E8E 1E C180            259          MOV     DS,WORD PTR CS:DSADDR+CSWRAP;   DS = ROM DATA SEGMENT
```

```
004A'  8E 06 0007"            260          MOV     ES,TMSSSV                ;       ES = SS OF INTERRUPTED LOGIC
004E'  8B 1E 0006"            261          MOV     BX,TMSPSV                ;       BX = SP OF INTERRUPTED LOGIC
0052'  2EFF 2E C164           262          JMP     DWORD PTR CS:XITVEC*4+CSWRAP;    TAKE COMMON INTERRUPT EXIT
                              263          SUBTTL
                              264          END
```

No errors detected

```
                                1   ;*****************************************************************************
                                2   ; TITLE    - TIMSUB (TIMER SUBROUTINES)
                                3   ; COMPUTER - 8088 ASSEMBLY LANGUAGE
                                4   ; ABSTRACT - This module contains subroutines that are called via a
                                5   ;       soft interrupt from the TIMING macro.  The following subroutines
                                6   ;       are provided :
                                7   ;
                                8   ;               TIMCAN - Cancel a timing event
                                9   ;               TIMRST - Restart a timing event
                               10   ;
                               11   ;*****************************************************************************
                               12           NAME    TIMSUB (TIMER SUBROUTINES)
                               13           SUBTTL
                               15   ;*****************************************************************************
                               16   ;                           PUBLIC DEFINITIONS
                               17   ;*****************************************************************************
                               18
                               19           PUBLIC  TIMCAN
                               20           PUBLIC  TIMRST
                               21   ;*****************************************************************************
                               22   ;                           LOCAL CONSTANTS
                               23   ;       INCLUDE PEG:TIMEVT.EQU
                               24   ;*****************************************************************************
                               25
                               45           SUBTTL  TIMCAN (CANCEL EVENT SUBROUTINE)
                               46
     =0000                     47           SECT    ROMCOD
                               48           ASSUME  CS:ROMCOD
                               49   ;*****************************************************************************
                               50   ;
                               51   ; TITLE    - TIMCAN (TIMER CANCEL EVENT SUBROUTINE)
                               52   ; ABSTRACT - This subroutine allows the caller to make an event
                               53   ;       inactive.  The event status bit is reset to indicate that
                               54   ;       the event is now inactive.  Since only interrupt-oriented
                               55   ;       events are supported at this time, the event acknowledgement
                               56   ;       flag is not checked.
                               57   ;
                               58   ; INPUT    - DS,DI = EVENT ADDRESS
                               59   ;
                               60   ; OUTPUT   -  Event is now inactive
                               61   ;
                               62   ; REGISTERS USED - none
                               63   ;
                               64   ; STACK USED - 6 BYTES
                               65   ;
                               66   ;*****************************************************************************
                               67
 0000'                         68   TIMCAN  PROC    FAR                         ;;;     DS,DI = EVENT ADDRESS
 0000'  80 65 03 7F '          69           AND     BYTE PTR IEVFLG[DI],NOT(1 SHL EVTSTA);;; RESET STATUS BIT
 0004'  CF                     70           IRET                                ;;;     ** INTERRUPT RETURN **
                               71
                               72           SUBTTL  TIMRST (RESTART EVENT SUBROUTINE)
```

```
                        73   ;*****************************************************************************
                        74   ;
                        75   ; TITLE    - TIMRST (TIMER RESTART EVENT SUBROUTINE)
                        76   ; ABSTRACT - This subroutine allows the caller to restart a timing event.
                        77   ;       The event status bit is reset to indicate that the event is now
                        78   ;       inactive.  Since only interrupt-oriented events are supported at
                        79   ;       this time, the event acknowledgement flag is not checked.
                        80   ;
                        81   ; INPUT    - DS,DI = EVENT ADDRESS
                        82   ;            AX = INITIAL COUNTER VALUE
                        83   ;
                        84   ; OUTPUT   - Event restarted
                        85   ;
                        86   ; REGISTERS USED - none
                        87   ;
                        88   ; STACK USED - 6 BYTES
                        89   ;
                        90   ;*****************************************************************************
                        91
0005'                   92   TIMRST  PROC    FAR                         ;;;     DS,DI = EVENT ADDR; AX = COUNT
0005'  89 45 04         93           MOV     IEVCTR[DI],AX               ;;;     SET INITIAL COUNT IN EVENT
0008'  80 4D 03 80      94           OR      BYTE PTR IEVFLG[DI],1 SHL EVTSTA;;; SET STATUS BIT
                        95                                               ;;;     NEW STATUS = ACTIVE
000C'  CF               96           IRET                                ;;;     ** INTERRUPT RETURN **
                        97
                        98           SUBTTL
                        99           END
```

No errors detected

```
                                1  ;*****************************************************************************
                                2  ;  TITLE    - TIMTST - timers power up diagnostics.
                                3  ;  COMPUTER - 8088 ASSEMBLY LANGUAGE
                                4  ;  ABSTRACT - This module contains the power up diagnostics that test the
                                5  ;       three timers present in the PEGASUS main board.  It is assumed that
                                6  ;       the ability to generate interrupts has been previosuly tested.
                                7  ;       This test emphasizes the testing of the timer accuraccy in various
                                8  ;       timer intervals.
                                9  ;*****************************************************************************
                               10          NAME    TIMTST - PEGASUS power up timer test
                               11  ;       SUBTTL
                               13  ;*****************************************************************************
                               14  ;                           EXTERNAL REFERENCES
                               15  ;*****************************************************************************
     =0000                     16          SECT    ROMDAT
                               17          EXTERN  DSKC_M:BYTE     ; DISK SELECT LATCH MEMORY IMAGE.       (ROMDAT)
                               18          EXTERN  PRCO_M:BYTE     ; PRINTER OUTPUT LATCH MEMORY IMAGE.    (ROMDAT)
     =0000                     19          SECT    ROMCOD
                               20          ASSUME CS:ROMCOD
                               21          EXTERN  DSPDIE:NEAR     ; DISPLAY FATAL ERROR and DIE.          (OUTPUT)
                               22          EXTERN  VECINI:NEAR     ; ENTRY TO VECTOR INITIALIZATION.       (VECINI)
                               23          EXTERN  DECLED:NEAR     ; DECREMENT LED SUBROUTINE.             (DECLED)
                               24          EXTERN  T0ERR:ABS       ; TIMER 0 ERROR CODE.                   (TSTERR)
                               25          EXTERN  T1ERR:ABS       ; TIMER 1 ERROR CODE.                   (TSTERR)
                               26          EXTERN  T2ERR:ABS       ; TIMER 2 ERROR CODE.                   (TSTERR)
                               27          EXTERN  LED100:ABS      ; LED ERROR CODE.                       (ROMERR)
                               28  ;*****************************************************************************
                               29  ;                           PUBLIC  DEFINITIONS
                               30  ;*****************************************************************************
                               31          PUBLIC  TIMTST          ; ENTRY TO TIMER POWER UP DIAGNOSTICS.
                               32  ;*****************************************************************************
                               33  ;                           LOCAL CONSTANTS
                               34  ;       INCLUDE  PEQ:HARDWARE.EQU
                               35  ;       INCLUDE  PEQ:VECTOR.EQU
                               36  ;*****************************************************************************
                               37  ;
                              316  ;*****************************************************************************
                              317  ;                           BASE PAGE LOCATIONS
                              318  ;*****************************************************************************
                              319  ;
     =0000                    320      SECT        ROMCOD
                              321      ASSUME      CS:ROMCOD, DS:ROMDAT, ES:ABS0
                              322  ;
                              323  ;               Table of test cases.
                              324  ;
     0000'                    325  TMTTAB  LABEL   WORD            ;       Table of cases tested.
                              326  ;
     =0000                    327  TCMOD   EQU     0               ;       OFFSET TO MODE BYTE
     =0001                    328  TCLAT   EQU     1               ;       OFFSET TO LATCH BYTE
     =0002                    329  TCERR   EQU     2               ;       OFFSET TO ERROR CODE
     =0003                    330  TCPORT  EQU     3               ;       PORT ADDRESS
     =0005                    331  TCPROG  EQU     5               ;       OFFSET TO PROGRAM VALUE
```

```
                =0007                   332   TCCHEK  EQU     7               ;       OFFSET TO CHECK VALUE
                =0009                   333   TCLIM   EQU     9               ;       OFFSET TO LIMIT READ.
                =000B                   334   TMTSIZ  EQU     11
                                        335   ;
                                        336   ; (TCPROG)  PROG = PROGRAMMED TIMER COUNT
                                        337   ; (TCLIM)   TIMER READ VALUE = (PROG AND OFFE0H) - 20H
                                        338   ;           RUN TIME = PROG + (PROG - READ + 10) / 2) * CLOCK RATE
                                        339   ; (TCCHEK)  LOOP COUNT = RUN TIME / LOOP TIME
                                        340   ;           LOOP TIME = 21 * 200nsec    (LOOP $ time)
                                        341   ;
0000' 36                                342           DB      TC_SC0+TC_WRD+TC_MD3;   Timer 0 mode 3
0001' 00                                343           DB      TC_SC0+TC_LAT   ;       Latch timer 0
0002' 06*                               344           DB      T0ERR           ;       Timer 0 fail code
0003' 0014                              345           DW      TIMER0          ;       Port address for timer 0
0005' FFFF                              346           DW      0FFFFH          ;       Timer count
0007' 30C6                              347           DW      030C6H          ;       Loop count = 52.428 ms.
0009' FFC0                              348           DW      0FFC0H          ;       Timer read value
                                        349   ;
000B' 36                                350           DB      TC_SC0+TC_WRD+TC_MD3;   Timer 0 mode 3
000C' 00                                351           DB      TC_SC0+TC_LAT   ;       Latch timer 0
000D' 06*                               352           DB      T0ERR           ;       Timer 0 fail code
000E' 0014                              353           DW      TIMER0          ;       Port address for timer 0
0010' 30D4                              354           DW      030D4H          ;       Timer count
0012' 094F                              355           DW      094FH           ;       Loop count = 10 ms.
0014' 30A0                              356           DW      030A0H          ;       Timer read value
                                        357   ;
0016' B6                                358           DB      TC_SC2+TC_WRD+TC_MD3;   Timer 2 mode 3
0017' 80                                359           DB      TC_SC2+TC_LAT   ;       Latch timer 2
0018' 08*                               360           DB      T2ERR           ;       Timer 2 fail code
0019' 0016                              361           DW      TIMER2          ;       Port address for timer 2
001B' FFFF                              362           DW      0FFFFH          ;       Timer count
001D' 618D                              363           DW      0618DH          ;       Loop count = 104.8 ms.
001F' FFC0                              364           DW      0FFC0H          ;       Timer read value
                                        365   ;
0021' B6                                366           DB      TC_SC2+TC_WRD+TC_MD3;   Timer 2 mode 3
0022' 80                                367           DB      TC_SC2+TC_LAT   ;       Latch timer 2
0023' 08*                               368           DB      T2ERR           ;       Timer 2 fail code
0024' 0016                              369           DW      TIMER2          ;       Port address for timer 2
0026' 186A                              370           DW      0186AH          ;       Timer count
0028' 0951                              371           DW      0951H           ;       Loop count = 10.0 ms.
002A' 1840                              372           DW      01840H          ;       Timer read value
                                        373   ;
002C' 76                                374           DB      TC_SC1+TC_WRD+TC_MD3;   Timer 1 mode 3
002D' 40                                375           DB      TC_SC1+TC_LAT   ;       Latch timer 1
002E' 07*                               376           DB      T1ERR           ;       Timer 1 fail code
002F' 0015                              377           DW      TIMER1          ;       Port address for timer 1
0031' FFFF                              378           DW      0FFFFH          ;       Timer count
0033' 618D                              379           DW      0618DH          ;       Loop count = 104.8 ms.
0035' FFC0                              380           DW      0FFC0H          ;       Timer read value
                                        381   ;
0037' 76                                382           DB      TC_SC1+TC_WRD+TC_MD3;   Timer 1 mode 3
0038' 40                                383           DB      TC_SC1+TC_LAT   ;       Latch timer 1
```

```
0039' 07*                384           DB      T1ERR           ;       Timer 1 fail code
003A' 0015               385           DW      TIMER1          ;       Port address for timer 1
003C' 3D09               386           DW      03D09H          ;       Timer count
003E' 1744               387           DW      01744H          ;       Loop count = 25.0 ms.
0040' 3CE0               388           DW      03CE0H          ;       Timer read value
                         389   ;
0042'                    390   TMTEND  LABEL   BYTE
                         391   ;
                         392
                         393   ;
                         394   ; This is the entry to the timer diagnostics.  The speaker time is tested
                         395   ; first, followed by the test of the interrupt driven timers. SI is used as
                         396   ; a pointer to the table of test cases.
                         397   ; DL contains any errors found from the interrupt test.
                         398   ;
                         399   ;       Disable timer interrupts.
                         400   ;
0042' 8A DA              401   TIMTST: MOV     BL,DL                   ;;; Get OR'D error bits into BL.
0044' A0 0001"           402           MOV     AL,DSKC_M               ;;; Disable interrupts, enable speaker
0047' 24 F9              403           AND     AL,(NOT TMR1EN) AND (NOT TMR2EN)        ;;;
0049' 0C 01              404           OR      AL,SPKREN               ;;;
004B' A2 0001"           405           MOV     DSKC_M,AL               ;;; Get latch memory image.
004E' E6 00              406           OUT     DSKC_P,AL               ;;; Write image to latch.
0050' E8 001B            407           CALL    TSTMR                   ;;; Test timers.
0053' A0 0001"           408           MOV     AL,DSKC_M               ;;; Disable the speaker
0056' 24 FE              409           AND     AL,(NOT SPKREN)         ;;;
0058' A2 0001"           410           MOV     DSKC_M,AL               ;;; Save latch memory image
005B' E6 00              411           OUT     DSKC_P,AL               ;;; Write image to latch
005D' 08 DB              412           OR      BL,BL                   ;;;Q: Any errors?
005F' 75 06              413           JNZ     INFAIL                  ;;; Y: go report them.
0061' E8 0005"           414           CALL    DECLED                  ;;; N: decrement LED count.
0064' E9 0004"           415           JMP     VECINI                  ;;; Go execute floppy test.
                         416   ;
                         417   ;  This is the timer failure logic.  It places a failure code on the
                         418   ;  parallel port for factory diagnostics purposes.  This code indicates
                         419   ;  that a failure of one of the timers has occurred.
                         420   ;
0067' 8A D3              421   INFAIL: MOV     DL,BL                   ;;;put error code into  DL.
0069' B0 09*             422           MOV     AL,LED100               ;;;AL =  LED FAIL CODE.
006B' E9 0003"           423           JMP     DSPDIE                  ;;;Fatal error stops the CPU.
                         424
                         425   ;
                         426   ; This PROC is used to test all the timers.
                         427   ; The method for verifying proper operation of these timers is by
                         428   ; tracking the elapsed time with a software loop/counter and observing
                         429   ; the timer internal counter at the proper time to see if it matches its
                         430   ; expected value.
                         431   ;
006E'                    432   TSTMR   PROC    NEAR
006E' BE 0000"           433           MOV     SI,OFFSET TMTTAB        ;;; Get table of test cases.
0071'                    434   TOLOOP:
0071' 2EBB 4C 07         435           MOV     CX,CS:TCCHEK[SI]        ;;; CX = software counter value.
```

```
0075'  2E8B 54 03          436             MOV     DX,CS:TCPORT[SI]        ;;; DX = Timer port address.
                           437     ;               Initialize timer counter.
0079'  2E8A 04             438             MOV     AL,CS:TCMOD[SI]         ;;; Select timer in mode 3.
007C'  E6 17               439             OUT     TIMCMD,AL               ;;;
007E'  2E8A 44 05          440             MOV     AL,CS:TCPROG[SI]        ;;; BL= Low byte of counter value.
0082'  EE                  441             OUT     DX,AL                   ;;;
0083'  2E8A 44 06          442             MOV     AL,CS:TCPROG+1[SI]      ;;; BH= High byte of counter value.
0087'  EE                  443             OUT     DX,AL                   ;;;
0088'                      444     TOCNTL:
0088'  E2 FE               445             LOOP    $                       ;;;G: Software counter value = 0 ?
                           446                                             ;;; N: Continue in loop.
008A'  2E8A 44 01          447             MOV     AL,CS:TCLAT[SI]         ;;; Y: read timer counter value.
008E'  E6 17               448             OUT     TIMCMD,AL               ;;;
0090'  EC                  449             IN      AL,DX                   ;;; Read timer counter value.
0091'  8A E0               450             MOV     AH,AL                   ;;; Save LSB in AH.
0093'  EC                  451             IN      AL,DX                   ;;;
0094'  86 C4               452             XCHG    AL,AH                   ;;; Exchange LSB and MSB.
0096'  25 FFE0             453             AND     AX,0FFE0H               ;;; Provide for small window (16 count)
0099'  2E3B 44 09          454             CMP     AX,CS:TCLIM[SI]         ;;; G: counter value within limits ?
009D'  75 0A               455             JNZ     OVERLM                  ;;;  N: counter value wrong.
009F'  83 C6 0B            456     NXTTST: ADD     SI,TMTSIZ               ;;;  Y: Point to next test case
00A2'  81 FE 0042"         457             CMP     SI,OFFSET TMTEND        ;;; G: End of test?
00A6'  75 C9               458             JNZ     TOLOOP                  ;;;  No, continue testing of next case.
00A8'  C3                  459             RET                             ;;;  Yes, return to caller.
                           460     ;
00A9'                      461     OVERLM:                                 ;;;
00A9'  2E0A 5C 02          462             OR      BL,CS:TCERR[SI]         ;;;No, report timer error
00AD'  EB F0               463             JMP     NXTTST                  ;;;And do next test
                           464     ;
                           465             END

No errors detected
```

```
                        1   ;*****************************************************************************
                        2   ; TITLE    - TSTERR (POWERUP TEST ERROR CODES)
                        3   ; COMPUTER - 8088 ASSEMBLY LANGUAGE
                        4   ; ABSTRACT - THIS MODULE CONSISTS OF THE VARIOUS ERROR CODES THAT THE
                        5   ;            DIFFERENT POWERUP TESTS MIGHT PUT TO THE PARALLEL PRINTER
                        6   ;            PORT TO CONVEY ADDITIONAL INFORMATION CONCERNING THE NATURE
                        7   ;            OF THE LED INDICATED FAILURE.
                        8   ;
                        9   ; REGISTERS USED - NONE
                       10   ;
                       11   ; STACK USED - NONE
                       12   ;
                       13   ;*****************************************************************************
                       14           NAME    TSTERR
                       15           SUBTTL
                       17   ;*****************************************************************************
                       18   ;                           PUBLIC DEFINITIONS
                       19   ;*****************************************************************************
                       20   ;
                       21           PUBLIC  ATMERR
                       22           PUBLIC  ATTERR
                       23           PUBLIC  CINTER
                       24           PUBLIC  CURERR
                       25           PUBLIC  CRAMER
                       26           PUBLIC  E_EVEN
                       27           PUBLIC  E_ODD
                       28           PUBLIC  E_REQ
                       29           PUBLIC  E_CMD
                       30           PUBLIC  VIDERR
                       31           PUBLIC  VBLERR
                       32           PUBLIC  TOERR
                       33           PUBLIC  T1ERR
                       34           PUBLIC  T2ERR
                       35           PUBLIC  ICNERR
                       36           PUBLIC  IVIERR
                       37           PUBLIC  TIMERR
                       38           PUBLIC  NMIERR
                       39           PUBLIC  KEYERR
                       40           PUBLIC  FLIERR
                       41
                       42           SUBTTL  MAIN
                       43   ;*****************************************************************************
                       44   ;                           MODULE ENTRY POINT
                       45   ;*****************************************************************************
                       46   ;       INTERRUPT CTRL ERROR CODES FOR PARALLEL PRINTER PORT
                       47   ;
=0000                  48   ICNERR  EQU     00H             ;       CONTROLLER ERROR (IMR).
=0001                  49   IVIERR  EQU     01H             ;       INVALID INTERRUPT ERROR.
=0002                  50   NMIERR  EQU     02H             ;       NMI ERROR FAILURE.
=0004                  51   TIMERR  EQU     04H             ;       TIMER INTERRUPT FAILURE.
=0008                  52   FLIERR  EQU     08H             ;       FLOPPY INTERRUPT FAILURE.
=0010                  53   KEYERR  EQU     10H             ;       KEYBOARD INTERRUPT ERROR.
```

```
                        54  ;
                        55  ;*****************************************************************************
                        56  ;       TIMER ERROR CODES FOR PARALLEL PRINTER PORT
                        57  ;
=0020                   58  TOERR   EQU     20H             ;       TIMER 0 ERROR.
=0040                   59  T1ERR   EQU     40H             ;       TIMER 1 ERROR.
=0080                   60  T2ERR   EQU     80H             ;       TIMER 2 ERROR.
                        61  ;
                        62  ;*****************************************************************************
                        63  ;       DISK CTRL ERROR CODES FOR PARALLEL PRINTER PORT
                        64  ;
=0001                   65  E_EVEN  EQU     01H                     ;   Even-order bit in secbuf bad error
=0002                   66  E_ODD   EQU     02H                     ;   Odd-order bit in secbuf bad error
=0004                   67  E_REQ   EQU     04H                     ;   FDC register bad error
=0008                   68  E_CMD   EQU     08H                     ;   Command execution error
                        69  ;
                        70  ;*****************************************************************************
                        71  ;       CRT CTRL ERROR CODES FOR PARALLEL PRINTER PORT
                        72  ;
=0001                   73  ATMERR  EQU     01H                     ;   Attribute memory failure.
=0002                   74  ATTERR  EQU     02H                     ;   Attribute latch failure.
=0004                   75  CURERR  EQU     04H                     ;   Controller failure (cursor).
=0008                   76  CRAMER  EQU     08H                     ;   Character memory failure.
=0010                   77  VIDERR  EQU     10H                     ;   Video output failure.
=0020                   78  CINTER  EQU     20H                     ;   CRT interrupt failure.
=0040                   79  VBLERR  EQU     40H                     ;   Vertical blank interrupt failure.
                        80  ;*****************************************************************************
                        81          END
```

No errors detected

```
                        1   ;*****************************************************************************
                        2   ;  TITLE    - VECINI - ABSO initialization, vectors, system info
                        3   ;  COMPUTER - 8088 Assembly Language
                        4   ;  ABSTRACT - This routine is responsible for initializing the interrupt
                        5   ;       vectors, BIOS interface vectors, and various system information
                        6   ;       contained in the first 1K of RAM.  Note that all unused interrupt
                        7   ;       vectors are set to WILD$$.
                        8   ;
                        9   ;       *** BE WARNED:  THIS MODULE IS AFFECTED BY ANY CHANGES IN THE
                       10   ;       FILE 'VECTOR.EQU' DUE TO THE FACT THAT THE VECTOR INITIALIZATION
                       11   ;       TABLES MUST MATCH THE VECTOR EQUATES.
                       12   ;
                       13   ;*****************************************************************************
                       14        NAME    VECINI - ABSO initialization, vectors, system info
                       15   ;       SUBTTL
                       17   ;*****************************************************************************
                       18   ;                          PUBLIC DEFINITIONS
                       19   ;*****************************************************************************
                       20   ;
                       21        PUBLIC  FILVEC
                       22        PUBLIC  VECINI
                       23        PUBLIC  VECTBO
                       24        PUBLIC  VTOLEN
                       25   ;
                       26   ;*****************************************************************************
                       27   ;                          EXTERNAL REFERENCES
                       28   ;*****************************************************************************
                       29   ;
                       30        EXTRN   BEEPIO:FAR      ; System beeper I/O handler             (BELDSR)
                       31        EXTRN   CLK_IO:FAR      ; Time-of-day clock I/O handler         (CLKDSR)
                       32        EXTRN   CFG_IO:FAR      ; System Configuration function         (CONFIG)
                       33        EXTRN   CRT_IO:FAR      ; Screen I/O handler                    (CRTDSR)
                       34        EXTRN   DSK_IO:FAR      ; Floppy disk I/O handler               (DSKDSR)
                       35        EXTRN   DSKISR:FAR      ; Disk interrupt service routine        (DSKDSR)
                       36        EXTRN   DSKTST:NEAR     ; Entry point for the FDC test routine (DSKTST)
                       37        EXTRN   FATAL:FAR       ; Fatal error handler                   (OUTPUT)
                       38        EXTRN   INTRET:FAR      ; Common IRET instruction (for dummies)(INTXIT)
                       39        EXTRN   INTXIT:FAR      ; Common interrupt exit logic           (INTXIT)
                       40        EXTRN   KEY_IO:FAR      ; Keyboard I/O handler                  (KEYDSR)
                       41        EXTRN   KEYISR:FAR      ; Keyboard interrupt service routine    (KEYDSR)
                       42        EXTRN   KPAUSE:FAR      ; Keyboard default pause key handler    (KEYDSR)
                       43        EXTRN   PUPNMI:FAR      ; Powerup NMI interrupt service routine(OUTPUT)
                       44        EXTRN   PRT_IO:FAR      ; Printer I/O handler                   (PRTDSR)
                       45        EXTRN   PWRUP:FAR       ; Powerup reset starting location       (RESET)
                       46        EXTRN   TIMCAN:FAR      ; Timing services - cancel event        (TIMSUB)
                       47        EXTRN   TIMISR:FAR      ; Timer interrupt service routine       (TIMISR)
                       48        EXTRN   TIMRST:FAR      ; Timing services - restart event       (TIMSUB)
                       49        EXTRN   WILD$$:FAR      ; Unexpected (Wild) interrupt handler   (OUTPUT)
                       50   ;
=0000                  51        SECT    ROMDAT
                       52        EXTRN   RMDTSZ:ABS      ; Size of ROMDAT data area              (ROMDAT)
                       53   ;
```

```
                                54  ;*****************************************************************************
                                55  ;                           LOCAL CONSTANTS
                                56  ;       INCLUDE PEQ:VECTOR.EQU
                                57  ;*****************************************************************************
                                58  ;
                               152  ;
                               153  ;-----------------------------------------------------------------------------
                               154  ;
          =0014                155  TP5VEC  EQU     (OVFINT*4)+4
          =0018                156  TP6VEC  EQU     TP5VEC+4
                               157
                               158  ;*****************************************************************************
                               159  ;                           CODE SEGMENT DEFINITION
                               160  ;*****************************************************************************
                               161  ;
          =0000                162          SECT    ROMCOD
                               163          ASSUME  CS:ROMCOD
                               164  ;
                               165  ;*****************************************************************************
                               166  ;                           LOCAL DATA AREA
                               167  ;*****************************************************************************
                               168  ;
                               169  ;   These sets of vectors are moved into the 8088 interrupt area
                               170  ;   (Table VECTBO is moved by the routine RAMINI)
                               171  ;
0000'                          172  VECTBO  LABEL   WORD            ; Table 0 - Processor traps
                               173  ;
0000'  0014" 0000"             174          DD      WILD$$          ; Interrupt 0 - Divide-by-zero trap
0004'  0014" 0000"             175          DD      WILD$$          ; Interrupt 1 - Single-step trap
0008'  000E" 0000"             176          DD      PUPNMI          ; Interrupt 2 - Non-maskable interrupt
000C'  0014" 0000"             177          DD      WILD$$          ; Interrupt 3 - Break (single-byte) soft intr
0010'  0014" 0000"             178          DD      WILD$$          ; Interrupt 4 - Overflow trap
                               179  ;
0014'                          180  VTOEND  LABEL   WORD
          =000A                181  VTOLEN  EQU     (VTOEND-VECTBO)/2
                               182
                               183  ;-----------------------------------------------------------------------------
0014'                          184  VECTB1  LABEL   WORD            ; Table 1 - Pegasus system vectors
                               185  ;
                               186  ;   Hardware interrupts:
                               187  ;
0014'  0014" 0000"             188          DD      WILD$$          ; Interrupt 40H - IR0 (unused)
0018'  0014" 0000"             189          DD      WILD$$          ; Interrupt 41H - IR1 (unused)
001C'  0014" 0000"             190          DD      WILD$$          ; Interrupt 42H - IR2 (unused)
0020'  0012" 0000"             191          DD      TIMISR          ; Interrupt 43H - IR3 (Timer 1)
0024'  0014" 0000"             192          DD      WILD$$          ; Interrupt 44H - IR4 (unused)
0028'  0014" 0000"             193          DD      WILD$$          ; Interrupt 45H - IR5 (Parallel port ACK)
002C'  0006" 0000"             194          DD      DSKISR          ; Interrupt 46H - IR6 (Disk controller)
0030'  000C" 0000"             195          DD      KEYISR          ; Interrupt 47H - IR7 (Keyboard USART)
                               196  ;
                               197  ;   BIOS interface vectors
                               198  ;
```

```
0034'  0001" 0000"          199             DD      BEEPIO          ; Interrupt 48H - System beeper I/O
0038'  0004" 0000"          200             DD      CRT_IO          ; Interrupt 49H - Screen I/O
003C'  0008" 0000"          201             DD      KEY_IO          ; Interrupt 4AH - Keyboard I/O
0040'  000F" 0000"          202             DD      PRT_IO          ; Interrupt 4BH - Printer port I/O
0044'  0014" 0000"          203             DD      WILD$$          ; Interrupt 4CH - *** UNUSED ***
0048'  0005" 0000"          204             DD      DSK_IO          ; Interrupt 4DH - Floppy disk interface
004C'  0002" 0000"          205             DD      CLK_IO          ; Interrupt 4EH - Time-of-day clock I/O
0050'  0003" 0000"          206             DD      CFG_IO          ; Interrupt 4FH - System configuration
                            207     ;
0054'  0008" 0000"          208             DD      FATAL           ; Interrupt 50H - Fatal error trap
0058'  0013" 0000"          209             DD      TIMRST          ; Interrupt 51H - Restart timing event
005C'  0011" 0000"          210             DD      TIMCAN          ; Interrupt 52H - Cancel timing event
0060'  0014" 0000"          211             DD      WILD$$          ; Interrupt 53H - (unused)
0064'  0014" 0000"          212             DD      WILD$$          ; Interrupt 54H - (unused)
0068'  0014" 0000"          213             DD      WILD$$          ; Interrupt 55H - (unused)
006C'  0014" 0000"          214             DD      WILD$$          ; Interrupt 56H - (unused)
0070'  0009" 0000"          215             DD      INTRET          ; Interrupt 57H - CRT character mapping vector
                            216     ;
0074'  0009" 0000"          217             DD      INTRET          ; Interrupt 58H - Time slicing (every 25 msec)
0078'  000A" 0000"          218             DD      INTXIT          ; Interrupt 59H - Common interrupt exit routine
007C'  0009" 0000"          219             DD      INTRET          ; Interrupt 5AH - Dyn. timing services (100 ms)
0080'  0009" 0000"          220             DD      INTRET          ; Interrupt 5BH - Keyboard mapping vector
0084'  000D" 0000"          221             DD      KPAUSE          ; Interrupt 5CH - Keyboard pause key vector
0088'  0009" 0000"          222             DD      INTRET          ; Interrupt 5DH - Keyboard break key vector
008C'  0009" 0000"          223             DD      INTRET          ; Interrupt 5EH - Keyboard print screen vector
0090'  0009" 0000"          224             DD      INTRET          ; Interrupt 5FH - Keyboard queuing vector
                            225     ;
0094'  0000'                226             DW      ROMDAT          ; DSADDR - System ROM DS segment address
0096'  0015*                227             DW      RMDTSZ          ; DSSIZR - System ROM DS size in bytes
0098'  0000 0000            228             DD      00000000        ; DSADD0/DSSIZ0 - Option ROM 0 DS info
009C'  0000 0000            229             DD      00000000        ; DSADD2/DSSIZ2 - Option ROM 2 DS info
00A0'  0000 0000            230             DD      00000000        ; DSADD4/DSSIZ4 - Option ROM 4 DS info
00A4'  0000 0000            231             DD      00000000        ; DSADD6/DSSIZ6 - Option ROM 6 DS info
00A8'  0000 0000            232             DD      00000000        ; DSADD8/DSSIZ8 - Option ROM 8 DS info
00AC'  0000                 233             DW      0000            ; MEMSIZ
00AE'  00                   234             DB      00              ; INTCTR
00AF'  00                   235             DB      00              ; DRVBYT
                            236     ;
00B0'  0000                 237             DW      0000            ; SYSC01
00B2'  0000                 238             DW      0000            ; SYSC02
00B4'                       239     VT1END  LABEL   WORD
       =0050                240     VT1LEN  EQU     (VT1END-VECTB1)/2
                            241
                            242     ;*****************************************************************************
                            243     ;                           MODULE ENTRY POINT
                            244     ;*****************************************************************************
                            245     ;
                            246     ;   VECINI - Initialize the interrupt vectors and system info
                            247     ;
                            248     ;   INPUT:  (MEMSIZ) & (DSADDR)
                            249     ;   OUTPUT: (none) Vectors and sys info initialized
                            250     ;           (MEMSIZ) & (DSADDR) unchanged
```

```
                        ;   USED:
                        ;   STACK:
                            ASSUME  DS:ROMCOD, ES:ABS0
                        ;
00B4'                   VECINI  PROC    NEAR
00B4'  FA               ;       CLI                             ; Protect this
00B5'  0E               ;       PUSH    CS                      ;;;
00B6'  1F               ;       POP     DS                      ;;; DS = ROMCOD
00B7'  31 C0            ;       XOR     AX,AX                   ;;;
00B9'  8E C0            ;       MOV     ES,AX                   ;;; ES = ABS0
00BB'  B9 0014"         ;       MOV     CX,OFFSET WILD$$        ;;; Set all vectors to WILD$$
00BE'  E8 0015          ;       CALL    FILVEC                  ;;; Do it
                        ;                                       ;;;    ES=ABS0, DS=ROMCOD ...
00C1'  BE 0014"         ;       MOV     SI,OFFSET VECTB1        ;;; Source for move in DS
00C4'  BF 0100          ;       MOV     DI,OFFSET IROINT*4      ;;; Destination for move in ES
00C7'  B9 0050          ;       MOV     CX,OFFSET VT1LEN                ;;; Length of move (in words)
00CA'  F2               ;       REP
00CB'  A5               ;       MOVS    WORD PTR [DI],WORD PTR [SI]     ;;; Do it
                        ;
00CC'  2689 2E 0198     ;       MOV     ES:WORD PTR MEMSIZ,BP   ;;; Restore MEMSIZ
00D1'  8E DA            ;       MOV     DS,DX                   ;;; Set the DS back to normal
00D3'  E9 0007"         ;       JMP     DSKTST                  ;;; Go to next
                        ;
                        ;-----------------------------------------------------------------------
                        ;
                        ;   FILVEC - Fill 8088 vector space with 'constant' vector.  (Fills entire
                        ;       vector space except for first 5 (8088) vectors, DSADDR, & MEMSIZ
                        ;       which are init'ed by RAMINI.)
                        ;
                        ;   INPUT:  CX = The vector's offset (segment assumed to be ROMCOD)
                        ;           ** Interrupts disabled
                        ;   OUTPUT: (none) vector space initialized
                        ;   USED:   CX,DI,SI
                        ;   STACK:  2 bytes
                            ASSUME  DS:ABS0, ES:ABS0
                        ;
00D6'                   FILVEC  PROC    NEAR
00D6'  FC               ;       CLD                             ;;; Forward strings
00D7'  1E               ;       PUSH    DS                      ;;; Save caller's segs
00D8'  06               ;       PUSH    ES                      ;;;
00D9'  31 FF            ;       XOR     DI,DI                   ;;;
00DB'  8E DF            ;       MOV     DS,DI                   ;;; Set up segment registers
00DD'  8E C7            ;       MOV     ES,DI                   ;;;
00DF'  BA 0000'         ;       MOV     DX,ROMDAT               ;;; Save DSADDR
00E2'  8B 2E 0198       ;       MOV     BP,WORD PTR MEMSIZ      ;;; Save MEMSIZ
00E6'  89 0E 0014       ;       MOV     WORD PTR TP5VEC,CX      ;;; Set initial offset
00EA'  8C 0E 0016       ;       MOV     WORD PTR TP5VEC+2,CS    ;;; Set initial segment
00EE'  BE 0014          ;       MOV     SI,OFFSET TP5VEC        ;;; Copy vector from 1st location
00F1'  BF 0018          ;       MOV     DI,OFFSET TP6VEC        ;;;   ... to next location
00F4'  B9 01F4          ;       MOV     CX,2*(251-1)            ;;; All 251 (DWORD) locations
00F7'  F2               ;       REP                             ;;;   a word at a time
00F8'  A5               ;       MOVS    WORD PTR [DI], WORD PTR [SI]
```

```
00F9'  89 16 0180          303          MOV     WORD PTR DBADDR,DX      ;;; Restore DBADDR
00FD'  89 2E 0198          304          MOV     WORD PTR MEMSIZ,BP      ;;; Restore MEMSIZ
0101'  07                  305          POP     ES                      ;;; Restore seg regs
0102'  1F                  306          POP     DS                      ;;;
0103'  C3                  307          RET                             ;;; *** RETURN ***
                           308  ;
                           309          END

No errors detected
```