```
PASS 1
PASS 2
  ####  PC-1 HEX-MONITOR  ####  PROTON 650X ASSEMBLER  V4.4   PAGE: 0001

0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;     @@@@@@@@     @@@@@@                  @@
0000        ;     @@      @@   @@                     @@@
0000        ;     @@      @@   @@                    @ @@
0000        ;     @@@@@@@@     @@        @@@@@@        @@
0000        ;     @@           @@                      @@
0000        ;     @@           @@                      @@
0000        ;     @@           @@@@@@                @@@@@@
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000        ;
0000                .TITLE '  ####  PC-1 HEX-MONITOR  ####'
```

ROVC
Postbus 117
6710 BC Ede
08380•32514

KURSUS
INDUSTRIËLE MICROPROCESSOR

```
0000          ;
0000          ;
0000          ;  REVISION DD: 24-09-82    SK     FILE=PC21$
0000          ;
0000          ;  *******************************************
0000          ;  *******************************************
0000          ;  *****                                 *****
0000          ;  *****    PC-1   HEXADECIMAL  MONITOR  *****
0000          ;  *****                                 *****
0000          ;  *****       PROTON ELECTRONICS        *****
0000          ;  *****       ENERGIESTRAAT   36        *****
0000          ;  *****       1411 AT     NAARDEN       *****
0000          ;  *****       TEL:   02159-48224        *****
0000          ;  *****                                 *****
0000          ;  *******************************************
0000          ;  *******************************************
0000          ;
0000          ;  KEYBOARD HAS INVERTED DRIVERS    25-08-1982
0000          ;

---------------------------------------------
 ###  COMMAND DESCRIPTION  ###

0000          ; COMMAND                   FUNKTION
0000          ;
0000          ;  DISPLAY NEXT/PREVIOUS LOCATION,ENTER ADDRESS
0000          ;  GO TO USER'S PROGRAM
0000          ;  INPUT ADDRES
0000          ;  INSERT 1 BYTE INTO TO PROGRAM
0000          ;  DELETE 1 BYTE FROM THE PROGRAM
0000          ;  CALCULATE BRANCH OFFSET
0000          ;  SINGLE STEP INSTRUCTION(S)
0000          ;  LOAD CODE FROM AUDIO-CASSETTE
0000          ;  DUMP CODE TO CASSETTE
0000          ;
0000          ; THE MONITOR DRIVES A 16 CHARACTER DISPLAY,
0000          ; HANDLES INTERRUPTS, BREAKPOINTS AND INCORPORATES
0000          ; EIGHT 10-MILISECOUND TIMERS.
0000          ;
0000          ;

---------------------------------------------
###  VIA ADDRESSING  ###

0000          ;
0000          ROMBAS:   =$F800           ; START OF PROGRAM
0000          IOSEL:    =$E000           ; I/O SELECT
0000          MONRAM:   =$F0
0000          ;
0000                  *=IOSEL+$10
E010          ;
E010          ;VIA #1
```

```
0096  E010        DRB1:      *=*+1
0097  E011        DRA1:      *=*+1
0098  E012        DDRB1:     *=*+1
0099  E013        DDRA1:     *=*+1
0100  E014        T1CL1:     *=*+1
0101  E015        T1CH1:     *=*+1
0102  E016        T1LL1:     *=*+1
0103  E017        T1LH1:     *=*+1
0104  E018        T2LL1:     *=*+1
0105  E019        T2CH1:     *=*+1
0106  E01A        SR1:       *=*+1
0107  E01B        ACR1:      *=*+1
0108  E01C        PCR1:      *=*+1
0109  E01D        IFR1:      *=*+1
0110  E01E        IER1:      *=*+1
0111  E01F        APORT1:    *=*+1
0112  E020        ;
0113  E020        ;

---------------------------------------------
   ###  ASSEMBLER CONSTANTS  ###

0115  E020        ENTER      =$0D
0116  E020        ESC        =$1B
0117  E020        ;
0118  E020        ; COMMAND-KEY VALUE'S
0119  E020        MEMCMD:    =$0C               ; FORW.SPACE
0120  E020        MINCMD:    =$08               ; BACK.SPACE
0121  E020        ADRCMD:    =$4D               ; 'M'
0122  E020        SSTCMD:    =$53               ; 'S'
0123  E020        GOCMD:     =$47               ; 'G'
0124  E020        LODCMD:    =$52               ; 'R'
0125  E020        DMPCMD:    =$57               ; 'W'
0126  E020        OFFCMD:    =$4F               ; 'O'
0127  E020        INSCMD:    =$11
0128  E020        DELCMD:    =$12
0129  E020        TSTCMD:    =$14               ; CTRL-T
0130  E020        ;
0131  E020        ; HARDWARE CONSTANTS
0132  E020        ;
0133  E020        NMICTR     =IOSEL+7           ; CONTROL-LATCH
0134  E020        ;
0135  E020        CHSEL      =IOSEL+4
0136  E020        SEGM0      =IOSEL+3
0137  E020        SEGM1      =IOSEL+2
0138  E020        ;
0139  E020        INKB       =IOSEL+6           ; KEYBOARD BUFFER
0140  E020        OUTKB      =IOSEL+5           ; KEYBOARD LATCH
0141  E020        ;
0142  E020                   *=MONRAM           ; ZERO PAGE
0143  00F0        NMIVEC     *=*+2              ; INDIRECT VECTORS
0144  00F2        INTVEC     *=*+2              ; IRQ BEFORE MONITOR
```

ROVC ©

```
00F4          TEMP     *=*+2
00F6          IRQVEC   *=*+2         ; IRQ AFTER MONITOR, NO 'BRK'
00F8          ADL      *=*+1         ; MEMORY POINTER
00F9          ADH      *=*+1
00FA          AC       *=*+1         ; USER REG  A
00FB          XR       *=*+1         ;           X
00FC          YR       *=*+1         ;           Y
00FD          PS       *=*+1         ; USER STATUS
00FE          SPTR     *=*+1         ; USER STACKPOINTER
00FF          PRVCMD   *=*+1         ; PREVIOUS COMMAND
0100          ;
0100          ;  ****  FROM HERE IN PAGE 1  ****
0100          ;
0100                   *=$100
0100          ID       *=*+1         ; ID NUMBER FOR CASSETTE
0101          TAPID    *=*+1         ; ID. OF CURRENT FILE
0102          EAL      *=*+1         ; END OF EDITING BUFFER
0103          EAH      *=*+1
0104          DIV1     *=*+1
0105          CURCMD   *=*+1         ; CURRENT  COMMAND
0106          GKX:     *=*+1         ; AUX SAVE .X
0107          TIMER    *=*+16        ; 10 MSEK TIMER (DECR.)
0117          DBCNTR   *=*+2         ; KEYBOARD DEBOUNCE
0119          ASAV     *=*+1         ; SAVE:  A
011A          XSAV     *=*+1         ;        X
011B          YSAV     *=*+1         ;        Y
011C          INDADR                 ; JUMP FOR COMMAND'S
011C          INDADL   *=*+1
011D          INDADH   *=*+1
011E          CHKSML   *=*+1         ; CHECKSUM
011F          CHKSMH   *=*+1
0120          ;
0120          DIBUFL   =16           ; DISPLAYBUFFER SIZE
0120          DIBUFF   *=*+DIBUFL    ; DISPLAY REFRESH BUFFER
0130          DCPTR    *=*+1         ; DISPLAY POINTER
0131          OMASK    *=*+1         ; MASK FOR KEYBOARD-OUTPUT
0132          IMASK    *=*+1         ; MASK FOR INPUT-DECODE
0133          KEYNR    *=*+1         ; LOGICAL NUMBER OF THE KEY
0134          LSTKEY   *=*+1         ; ASCII VALUE OF CURRENT & LAST KEY
0135          CPATRN   *=*+1         ; PATTERN FOR CONTROL-KEYS
0136          REPTO    *=*+1         ; REPEAT-TIME COUNTER
0137          ;
0000                   .EX1
0000          ;
0000          ;
0000                   *=ROMBAS
F800  A2FF    RESET:   LDX #$FF      ; INITIAL STACK VALUE
F802  9A               TXS
F803  78               SEI           ;  AND BLOCK INTERRUPTS
F804  2049FD           JSR BLKDIS    ; BLOCK DISPLAY-INTERRUPTS
F807  86FE             STX SPTR      ; USER STACK
F809  D8               CLD           ; BINAIRY MODE
```

```
F80A              ; TEST FOR WARM RESTART
F80A  A5F0                  LDA NMIVEC
F80C  CDA2F8                CMP VECTAB
F80F  D007                  BNE RS3A
F811  A5F1                  LDA NMIVEC+1
F813  CDA3F8                CMP VECTAB+1
F816  F022                  BEQ WARM          ; WARM RESTART
F818              ; COLD START
F818  A00D        RS3A      LDY #VECLEN
F81A  B9A2F8      RS1       LDA VECTAB,Y
F81D  99F000                STA MONRAM,Y
F820  88                    DEY
F821  10F7                  BPL RS1
F823  A900                  LDA #0
F825  A205                  LDX #TIMER+1-EAH
F827  9D0301      RS2       STA EAH,X
F82A  CA                    DEX
F82B  10FA                  BPL RS2
F82D  A970                  LDA #$70
F82F  8D0201                STA EAL
F832  A901                  LDA #1
F834  8D0001                STA ID            ; DEFAULT = 1
F837  8D0401                STA DIV1
F83A              ;
F83A  A0F8        WARM      LDY #>HDRMSG      ; PRINT HEADER-MESSAGE
F83C  A9B0                  LDA #<HDRMSG
F83E  2065FC                JSR TPRINT
F841  2007FE                JSR VIAINI        ; RELEASE PB7 BY CB2=LOW
F844              ;
```

MONITOR COMMAND LOOP

```
F844              ;
F844  A6FE        MONITR    LDX SPTR
F846  9A                    TXS
F847  2054FD                JSR RELDIS        ; RELEASE DISPLAY
F84A  D8                    CLD
F84B  20DFF9                JSR GETUKY        ; GET UPPER-CASE KEY
F84E              ;
F84E  A6FE        JUMPER    LDX SPTR          ; SET SYSTEM STACK
F850  9A                    TXS
F851  8D0501                STA CURCMD
F854  A20A                  LDX #NCMNDS
F856              JP1
F856  DD81F8                CMP CMDTAB,X
F859  F00E                  BEQ JP2
F85B  CA                    DEX
F85C  10F8                  BPL JP1
F85E  A93F                  LDA #'?
F860  8D0501                STA CURCMD        ; ILLEGAL COMMAND
F863  2034F9                JSR WCOM1
```

```
F866  4C44F8            JMP MONITR
F869            ;
F869  BA        JP2     TXA
F86A  0A                ASL A
F86B  AA                TAX
F86C  BD8CF8            LDA JMPTAB,X
F86F  8D1C01            STA INDADL
F872  BD8DF8            LDA JMPTAB+1,X
F875  8D1D01            STA INDADH
F878  207EF8            JSR CMD
F87B  4C44F8            JMP MONITR
F87E            ;
F87E  6C1C01 CMD        JMP (INDADR)
F881            ;
F881            CMDTAB
F881  0C                .BYT MEMCMD,MINCMD,ADRCMD,SSTCMD
F885  47                .BYT GOCMD,INSCMD,DELCMD,LODCMD
F889  57                .BYT DMPCMD,OFFCMD,TSTCMD
F88C            ;
F88C            NCMNDS=*-CMDTAB-1
F88C            ;
F88C            ; FUNKTIONS MUST RETURN WITH 'RTS'
F88C  D8F8      JMPTAB  .WOR PMEM,MMEM,INPADR,SSUSER
F894  ADF9              .WOR GOUSER,INSDEL,INSDEL,LOAD
F89C  DAFD              .WOR DUMP,OFFSET,TESTER
F8A2            ;
F8A2            ;
F8A2  D5FC      VECTAB  .WOR TERMNL,IRQHDL,NSUP,0
F8AA  0000              .WOR 0             ; PROGRAM COUNTER
F8AC  00                .BYT 0,0,0,0       ; REGISTERS
F8B0            VECLEN  =*-VECTAB-1
F8B0            ;
F8B0  5052      HDRMSG  .BYT 'PROTON PC-1 V1.0',0
F8C1            ;
F8C1            ;

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

 ### ENTRY FROM SINGLE-STEP ###

F8C1            ;
F8C1  85FA      SSTINT:  STA AC
F8C3  86FB               STX XR
F8C5  84FC               STY YR
F8C7  68                 PLA              ; STATUS
F8C8  85FD               STA FS
F8CA  68                 PLA              ; PROGRAM COUNTER
F8CB  85F8               STA ADL
F8CD  68                 PLA
F8CE  85F9               STA ADH
F8D0  BA                 TSX              ; RP
F8D1  86FE               STX SPTR
F8D3  AD18E0             LDA T2LL1        ; CLEAR SST-INT.
```

ROVC ©

```
F8D6  58                   CLI
F8D7  D8                   CLD
F8D8               ;
F8D8               ;   ###  DIRECT TO +MEM & -MEM  ###
F8D8               ;
F8D8  A93E    PMEM         LDA #'>
F8DA  8D0501               STA CURCMD
F8DD  D005                 BNE SEQADR
F8DF               ;
F8DF  A93C    MMEM         LDA #'<
F8E1  8D0501               STA CURCMD
F8E4               ;
F8E4               ; NOW SHOW NEXT LOCATION
F8E4               ;

-------------------------------------------
 ### INPUT DATA ON SEQ. ADDRESSES  ###

F8E4  2031F9 SEQADR        JSR WCOMND          ; CLEAR DISPLAY & WRITE COMMAND
F8E7  A200                 LDX #0
F8E9  A5F9                 LDA ADH
F8EB  200FFC               JSR WBYTE
F8EE  A5F8                 LDA ADL
F8F0  200FFC               JSR WBYTE
F8F3  A200    SQA1         LDX #0
F8F5  20FEFB               JSR RBYTE           ; GET ADDRESS
F8F8  85F9                 STA ADH
F8FA  20FEFB               JSR RBYTE
F8FD  85F8                 STA ADL
F8FF  A200                 LDX #0
F901  A1F8                 LDA (ADL,X)         ; AND IT'S DATA
F903  A206                 LDX #6
F905  200FFC               JSR WBYTE
F908  20E1FB               JSR GETBYT
F90B               ;
F90B  AE0501               LDX CURCMD          ; CHECK FOR +MEM/-MEM
F90E  E03C                 CPX #'<
F910  F009                 BEQ SQA2
F912  E6F8                 INC ADL             ; NEXT LOCATION
F914  D0CE                 BNE SEQADR
F916  E6F9                 INC ADH
F918  4CE4F8               JMP SEQADR
F91B  C6F8    SQA2         DEC ADL             ; PREVIOUS LOCATION
F91D  A5F8                 LDA ADL
F91F  C9FF                 CMP #$FF
F921  D0C1                 BNE SEQADR
F923  C6F9                 DEC ADH
F925  4CE4F8               JMP SEQADR
F928               ;

-------------------------------------------
 ### INPUT AN ADDRESS ###
```

```
F928  2031F9 INPADR    JSR WCOMND
F92B  20BBFB           JSR GETADR
F92E  4CF3F8           JMP SQA1
F931         ;
F931         ;    ####  WRITE COMMAND  ####
F931         ;
F931  2022FC WCOMND    JSR CLRDSP
F934  AD0501 WCOM1     LDA CURCMD
F937  8D2F01           STA DIBUFF+15
F93A  60               RTS
F93B         ;
F93B         ;

-------------------------------------------
 ###  INSERT & DELETE  ###

F93B  A000   INSDEL:   LDY #0
F93D  AD0201 ID0       LDA EAL
F940  C5F8             CMP ADL           ; TEST FOR END
F942  AD0301           LDA EAH
F945  E5F9             SBC ADH
F947  B005             BCS ID1
F949  F003             BEQ ID1
F94B  4CF3F8           JMP SQA1          ; OUT
F94E  AD0501 ID1       LDA CURCMD
F951  C912             CMP #DELCMD
F953  F00B             BEQ IDD
F955  B1F8             LDA (ADL),Y       ; INSERT
F957  48               PHA
F958  8A               TXA
F959  91F8             STA (ADL),Y
F95B  68               PLA
F95C  AA               TAX
F95D  4C66F9           JMP IDNEXT
F960          ;
F960  C8      IDD      INY
F961  B1F8             LDA (ADL),Y
F963  88               DEY
F964  91F8             STA (ADL),Y
F966          IDNEXT
F966  E6F8             INC ADL
F968  D0D3             BNE ID0
F96A  E6F9             INC ADH
F96C  D0CF             BNE ID0
F96E          ;

-------------------------------------------
 ###  CALCULATE BRANCH OFFSET  ###

F96E  2031F9 OFFSET    JSR WCOMND
F971  20BBFB           JSR GETADR
```

```
F974  A202            LDX #2
F976  20FEFB          JSR RBYTE
F979  18              CLC
F97A  E5F8            SBC ADL
F97C  A000            LDY #0
F97E  91F8            STA (ADL),Y
F980  4CE4F8          JMP SEQADR
F983          ;

------------------------------------------------
 ###  SINGLE STEP THE USER  ###

F983  ADA4F8 SSUSER   LDA VECTAB+2       ; SET INT-VECTOR ON 'SYSTEM-INT'
F986  85F2            STA INTVEC
F988  ADA5F8          LDA VECTAB+3
F98B  85F3            STA INTVEC+1
F98D  2049FD          JSR BLKDIS
F990  2054FD          JSR RELDIS
F993  AD1EE0          LDA IER1
F996  09A0            ORA #$A0
F998  8D1EE0          STA IER1
F99B  AD1BE0          LDA ACR1
F99E  29DF            AND #$DF
F9A0  8D1BE0          STA ACR1
F9A3  A924            LDA #$24           ; GENERATE 'IRQ' ON OPCODE-FETCH OF USER OPCOD
F9A5  8D18E0          STA T2LL1
F9A8  A900            LDA #0
F9AA  8D19E0          STA T2CH1
F9AD          ;

------------------------------------------------
 ###  GO TO THE USER-PROGRAM  ###

F9AD  A6FE   GOUSER   LDX SPTR           ; DEFINE STACK
F9AF  9A              TXS
F9B0  A5F9            LDA ADH
F9B2  48              PHA
F9B3  A5F8            LDA ADL
F9B5  48              PHA
F9B6  A5FD            LDA PS
F9B8  48              PHA
F9B9  A5FA            LDA AC
F9BB  A6FB            LDX XR
F9BD  A4FC            LDY YR
F9BF  40     NSUP     RTI
F9C0          ;

------------------------------------------------
 ###  ENTER MONITOR FROM "BRK"  ###

F9C0  85FA   BKUSER:  STA AC
F9C2  86FB            STX XR
```

```
F9C4  84FC                STY YR
F9C6  68                  PLA
F9C7  85FD                STA PS            ; STATUS
F9C9  D8                  CLD
F9CA  68                  PLA
F9CB  38                  SEC
F9CC  E901                SBC #1            ; CORRECT 1 FROM 'BRK'
F9CE  85F8                STA ADL           ; PC
F9D0  68                  PLA
F9D1  E900                SBC #0
F9D3  85F9                STA ADH
F9D5  BA                  TSX
F9D6  86FE                STX SPTR
F9D8  AD18E0              LDA T2LL1         ; CLEAR TIMER2
F9DB  58                  CLI
F9DC  4CE4F8              JMP SEQADR        ; SHOW NEXT LOCATION
F9DF              ;
F9DF              ;  ####  GET UPPER-CASE KEY  ####
F9DF              ;
F9DF  20EDF9 GETUKY:      JSR GETKEY
F9E2  C960                CMP #$60          ; TRANSFORM LOWERCASE TO UPPERCASE
F9E4  9006                BCC GUKY9
F9E6  C97B                CMP #$7B
F9E8  B002                BCS GUKY9
F9EA  29DF                AND #$DF
F9EC  60     GUKY9        RTS
F9ED              ;
F9ED              ;
```

```
 ###  GET A KEY FROM KEYBOARD  ###

F9ED              ;
F9ED              ;
F9ED              ;  THE GETKEY WILL PERFORM BOTH DOWN- & UP-DEBOUNCE
F9ED              ;  ON ANY CHAR-KEY. THE SHIFT, ALPHA-LOCK AND 'CTRL'
F9ED              ;  FUNKTIONS ARE PERFORMED LATER.
F9ED              ;
F9ED  8A     GETKEY:      TXA               ; SAVE .X & .Y
F9EE  48                  PHA
F9EF  98                  TYA
F9F0  48                  PHA
F9F1  201BFA              JSR ROVER         ; ROLL-OVER & REPEAT
F9F4  B01D                BCS GKEY9         ; HAVE A REPEAT.
F9F6  204AFA GKEY1        JSR SETDBN        ; START DEBOUNCE-TIMER
F9F9  2054FA GKEY2        JSR DECKEY        ; DECODE THE KEY(S)
F9FC  90F8                BCC GKEY1         ; NO KEY DEPRESSED
F9FE  2043FA GKEY3        JSR QHOLD         ; STILL DOWN ??
FA01  D0F3                BNE GKEY1         ; NO, RELEASED PREMATURELY
FA03  2050FA              JSR QDEBNC        ; TIME-OUT ?
FA06  D0F6                BNE GKEY3         ; NO. .
FA08  204AFA              JSR SETDBN        ; START TIMER FOR REPEAT
```

```
FA0B  A91B                LDA #27          ; START REPEAT AFTER 0.8 SEK
FA0D  8D3601              STA REPTO
FA10  209DFA              JSR KDECOD       ; DECODE KEY WITH SHIFT & CNTL
FA13  68       GKEY9      PLA              ; RESTORE .Y & .X
FA14  A8                  TAY
FA15  68                  PLA
FA16  AA                  TAX
FA17  AD3401              LDA LSTKEY
FA1A  60                  RTS
FA1B                ;
FA1B                ;  WAIT TILL KEY IS RELEASED (WITH DEBOUNCE)
FA1B                ;  & GENERATE REPEAT WHEN HOLDED (16 PER SEK).
FA1B                ;
FA1B  2043FA ROVER        JSR QHOLD        ; STILL DOWN ??
FA1E  D014                BNE ROVER5       ; NO.
FA20  2050FA              JSR QDEBNC       ; TIME-OUT
FA23  D0F6                BNE ROVER        ; NO. TRY AGAIN
FA25  204AFA              JSR SETDBN       ; RESTART TIMER
FA28  CE3601              DEC REPTO        ; COUNT INTERVALS (200MS)
FA2B  D0EE                BNE ROVER
FA2D  A903                LDA #3           ; 60MS
FA2F  8D3601              STA REPTO
FA32  38                  SEC              ; REPEAT!
FA33  60                  RTS
FA34                ;
FA34  204AFA ROVER5       JSR SETDBN       ; START TIMER
FA37  2043FA ROVER6       JSR QHOLD        ; NOW SEE IF THE KEY
FA3A  F0DF                BEQ ROVER        ; NOT RELEASED.
FA3C  2050FA              JSR QDEBNC       ; TIME-OUT ??
FA3F  D0F6                BNE ROVER6
FA41  18                  CLC
FA42  60                  RTS              ; KEY IS RELEASED
FA43                ;
FA43  AD3201 QHOLD        LDA IMASK
FA46  2C06E0              BIT INKB         ; .NE=RELEASED
FA49  60                  RTS
FA4A                ;
FA4A                ;  ####  START DEBOUNCE COUNTER  ####
FA4A                ;
FA4A                TDEBNC   =3            ; 20-30 MS
FA4A                ;
FA4A  A903          SETDBN    LDA #TDEBNC
FA4C  8D1701                  STA DBCNTR   ; START DEBOUNCE-TIMER
FA4F  60                      RTS
FA50                ;
FA50                ;  ####  TEST FOR DEBOUNCE-TIME-OUT  ####
FA50                ;
FA50  AD1701 QDEBNC       LDA DBCNTR
FA53  60                  RTS              ; .EQ = TIME-OUT
FA54                ;
FA54                ;
FA54                ;  ####  DECODE A KEY  ####
```

```
FA54                 ;
FA54                 ;  THIS ROUTINE DOES 2 THINGS:
FA54                 ;  1. SCAN THE 'SPECIAL' KEYS:
FA54                 ;       <L-SHIFT>,<R-SHIFT>,<ALPHA-LOCK>,<CNTL>
FA54                 ;     AND SAVE THESE BIT IN 'CPATRN'.
FA54                 ;     BIT 4: <ALPHA-LOCK>
FA54                 ;         5: <R-SHIFT>
FA54                 ;         6: <L-SHIFT>
FA54                 ;         7: <CNTL>
FA54                 ;  2. SCAN THE REST OF THE MATRIX AND HOLD THE OUTPUT
FA54                 ;     WHEN A 'CLOSED-CONTACT' IS DETECTED. (.C=1)
FA54                 ;     WHEN NO KEY IS FOUND THE .C = 0
FA54                 ;
FA54  A901   DECKEY   LDA #1           ; #### INIT MASKS ####
FA56  8D3201          STA IMASK        ; (POSITIVE LOGIC)
FA59  8D3101          STA OMASK
FA5C  A204            LDX #4           ; WE HAVE 4 KEYS TO READ FIRST
FA5E  207DFA DECK1    JSR QKB          ; SETOUT OMASK & READ INPUTS & BUMP
FA61  38              SEC
FA62  D001            BNE *+3
FA64  18              CLC
FA65  6E3501          ROR CPATRN
FA68  CA              DEX
FA69  D0F3            BNE DECK1
FA6B  AD3201 DECK2    LDA IMASK        ; CHECK FOR END
FA6E  2D3101          AND OMASK
FA71  2901            AND #1
FA73  18              CLC
FA74  D006            BNE DECK9        ; NO KEY DEPRESSED
FA76  207DFA          JSR QKB          ; READ A MATRIX-KNOD
FA79  D0F0            BNE DECK2        ; NOT CLOSED
FA7B  38              SEC
FA7C  60     DECK9    RTS
FA7D                 ;
FA7D                 ;  ####  READ A MATRIX-KNOD  ####
FA7D                 ;
FA7D  4E3101 QKB      LSR OMASK        ; INCR. MASKS
FA80  D00B            BNE QKB1
FA82  6E3101          ROR OMASK        ; .C=1
FA85  4E3201          LSR IMASK
FA88  D003            BNE QKB1
FA8A  6E3201          ROR IMASK        ; .C=1
FA8D  AD3101 QKB1     LDA OMASK
FA90                 ;-INV-  EOR #$FF  ; <<< FOR NON-INV. OUTPUT
FA90  8D05E0          STA OUTKB
FA93  209CFA          JSR QKB2         ; DELAY 12 USEK
FA96  AD3201          LDA IMASK
FA99  2D06E0          AND INKB
FA9C  60     QKB2     RTS
FA9D                 ;
FA9D                 ;
FA9D                 ;  ### DECODE 'KEYNR' FROM MATRIX(X,Y) ###
```

```
FA9D                ;
FA9D  AD3201 KDECOD   LDA IMASK
FAA0  A200            LDX #0
FAA2  20ECFA          JSR BINHEX        ; CONVERT BIT-PATTERN TO HEX
FAA5  8A              TXA
FAA6  0A              ASL A             ; ADJUST FOR O-MASK
FAA7  0A              ASL A
FAA8  0A              ASL A
FAA9  AA              TAX
FAAA  AD3101          LDA OMASK
FAAD  20ECFA          JSR BINHEX
FAB0  BDF3FA          LDA KMATRX,X      ; GET LOGICAL NUMBER
FAB3  8D3301          STA KEYNR         ; SAVE THE KEY'S NUMBER
FAB6                ;
FAB6                ; HAVE A SHIFT ??
FAB6                ;
FAB6  AD3501          LDA CPATRN
FAB9  4960            EOR #$60
FABB  2960            AND #$60
FABD  F008            BEQ KDEC5         ; NO SHIFT
FABF  A204            LDX #SHIFT
FAC1  203AFB          JSR SPROC         ; CALL THE SHIFT-PROCESSOR
FAC4  4CDBFA          JMP KDEC8
FAC7                ;
FAC7  AD3501 KDEC5    LDA CPATRN
FACA  2910            AND #$10
FACC  D008            BNE KDEC7         ; NO ALPHA-LOCK
FACE  A200            LDX #LOCK
FAD0  203AFB          JSR SPROC         ; CALL THE SHIFT-PROCESSOR
FAD3  4CDBFA          JMP KDEC8
FAD6                ;
FAD6  A900   KDEC7    LDA #0            ; GET VALUE FROM KEYNR
FAD8  2057FB          JSR SPROC7
FADB                ;
FADB                ; NOW MODIFY THE CODE ON 'CTRL'
FADB                ;
FADB  AD3501 KDEC8    LDA CPATRN
FADE  3008            BMI KDEC9         ; NO CTRL
FAE0  AD3401          LDA LSTKEY
FAE3  291F            AND #%00011111
FAE5  8D3401          STA LSTKEY
FAE8  AD3401 KDEC9    LDA LSTKEY
FAEB  60              RTS
FAEC                ;
FAEC                ;
FAEC                ;   ####  CONVERT BIT (.A) TO HEX ADD TO .X  ####
FAEC                ;
FAEC  4A     BINHEX   LSR A
FAED  B003            BCS BHEX9
FAEF  E8              INX
FAF0  D0FA            BNE BINHEX
FAF2  60     BHEX9    RTS
```

```
FAF3             ;
FAF3             ;
FAF3             ;  ####  CONVERT MATRIX TO KEY-NUMBERS  ####
FAF3             ;
FAF3  01        KMATRX    .BYT 1,46,56,35,3,45,1,53
FAFB  2C                  .BYT 44,19,41,23,5,4,37,42
FB03  15                  .BYT 21,22,36,38,7,6,24,40
FB0B  14                  .BYT 20,25,43,39,9,8,26,32
FB13  1F                  .BYT 31,28,27,33,11,10,29,14
FB1B  10                  .BYT 16,30,34,15,12,2,13,17
FB23  01                  .BYT 1,52,55,0,18,48,49,51
FB2B  01                  .BYT 1,54,47,50
FB2F             ; -- ALL UNASSIGNED KEYS ARE CODED AS THE SPACE-BAR --
FB2F             ;
FB2F             ;
FB2F             ;  ####  THIS TABLE CONTAINS INFORMATION OVER  ####
FB2F             ;        WHAT OPERATION MUST BE PERFORMED ONTO
FB2F             ;        WHITCH KEY.
FB2F             ;
FB2F            SHFTBL    =*
FB2F             ;
FB2F            LOCK      =*-SHFTBL
FB2F  13                  .BYT 19,44          ; KEYS 19 TILL 44
FB31  20                  .BYT $20            ; EOR #$20
FB32  00                  .BYT 0
FB33             ;
FB33            SHIFT     =*-SHFTBL
FB33  03                  .BYT 3,17           ; KEYS 3 TILL 17
FB35  10                  .BYT $10            ; EOR #$10
FB36  12                  .BYT 18,48          ; AND KEYS 18 TILL 48
FB38  20                  .BYT $20            ; EOR #$20
FB39  00                  .BYT 0
FB3A             ;
FB3A             ;  ####  SHIFT-PROCESSOR  ####
FB3A             ;
FB3A             ;  PERFORMS THE SCANNING OF THE LIST (.X)
FB3A             ;  CHECKS IF KEYNR IS IN RANGE, GETS THE
FB3A             ;  ASCII VALUE AND DOES THE OPERATION.
FB3A             ;
FB3A  BD2FFB SPROC        LDA SHFTBL,X
FB3D  F018                BEQ SPROC7          ; END OF LIST, NO-OP
FB3F  E8                  INX
FB40  CD3301              CMP KEYNR
FB43  F002                BEQ SPROC3
FB45  B008                BCS SPROC1          ; NOT IN RANGE
FB47  BD2FFB SPROC3       LDA SHFTBL,X
FB4A  CD3301              CMP KEYNR
FB4D  B004                BCS SPROC2          ; IN RANGE . .
FB4F  E8     SPROC1       INX
FB50  E8                  INX                 ; TRY ON THE NEXT BLOCK
FB51  D0E7                BNE SPROC
FB53             ;
```

```
FB53  E8      SPROC2   INX                    ; POINT TO THE OPERAND
FB54  BD2FFB           LDA SHFTBL,X
FB57  AC3301 SPROC7    LDY KEYNR
FB5A  5961FB           EOR ASCVAL,Y
FB5D  8D3401           STA LSTKEY             ; SAVE ASCII VALUE
FB60  60               RTS
FB61           ;
FB61           ; ####  TRANSLATION OF KEYNR TO ASCII VALUE  ####
FB61           ;
FB61  11      ASCVAL   .BYT $11               ; CHAR INS
FB62  20               .BYT $20               ; SPACE
FB63  3031             .BYT '0123456789:;,-./' ; KEY 2 - 17
FB73  4061             .BYT '@abcdefghijklmno'
FB83  7071             .BYT 'pqrstuvwxyz[\]^`'; KEY 18 - 48
FB92  12               .BYT $12               ; CHAR DEL
FB93  0D               .BYT $0D               ; KEY 50
FB94  08               .BYT $08
FB95  0A               .BYT $0A
FB96  0B               .BYT $0B
FB97  0C               .BYT $0C
FB98  1E               .BYT $1E               ; HOME KEY 55
FB99  1B               .BYT $1B               ; ESC
FB9A                                          ;
FB9A           ;
FB9A           ;

--------------------------------------------
 ###  GET HEX KEY OR DO COMMAND  ###

FB9A  20DFF9 GETHEX    JSR GETUKY             ; UPPER-CASE KEY
FB9D  C90D             CMP #ENTER
FB9F  F016             BEQ GHEX9
FBA1  C930             CMP #'0
FBA3  9013             BCC GHEX5              ; ERROR
FBA5  C93A             CMP #':
FBA7  900A             BCC GHEX1
FBA9  C941             CMP #'A
FBAB  900B             BCC GHEX5
FBAD  C947             CMP #'G
FBAF  B007             BCS GHEX5
FBB1  E906             SBC #6                 ; .C=0
FBB3  290F   GHEX1     AND #$F
FBB5  C910             CMP #$10               ; ALWAY .NE
FBB7  60     GHEX9     RTS
FBB8           ;
FBB8  4C4EF8 GHEX5     JMP JUMPER
FBBB           ;

--------------------------------------------
 ###  GET 4 DIGIT HEX ADDRESS  ###

FBBB  A900   GETADR    LDA #0
```

```
FBBD  AA                 TAX
FBBE  202DFC GA0         JSR WCHAR          ; CLEAR ADR-FIELD
FBC1  E8                 INX
FBC2  E004               CPX #4
FBC4  30F8               BMI GA0
FBC6  209AFB GA1         JSR GETHEX
FBC9  F015               BEQ GAEND
FBCB  A200               LDX #0
FBCD  48                 PHA                ; SAVE DATA
FBCE  BD2101 GA2         LDA DIBUFF+1,X     ; LEFT-SHIFT 4 DIGITS
FBD1  9D2001             STA DIBUFF,X
FBD4  E8                 INX
FBD5  E003               CPX #3
FBD7  30F5               BMI GA2
FBD9  68                 PLA
FBDA  202DFC             JSR WCHAR
FBDD  4CC6FB             JMP GA1
FBE0  60     GAEND       RTS
FBE1         ;

-------------------------------------------
 ###  GET BYTE IN DATA-FIELD ###

FBE1  209AFB GETBYT      JSR GETHEX
FBE4  F00E               BEQ GBEND
FBE6  AE2701             LDX DIBUFF+7       ; LEFT-SHIFT ON DATA-FIELD
FBE9  8E2601             STX DIBUFF+6
FBEC  A207               LDX #7
FBEE  202DFC             JSR WCHAR
FBF1  4CE1FB             JMP GETBYT
FBF4  A206   GBEND       LDX #6
FBF6  20FEFB             JSR RBYTE
FBF9  A200               LDX #0
FBFB  81F8               STA (ADL,X)
FBFD  60                 RTS
FBFE         ;

-------------------------------------------
 ###  READ A BYTE FROM DISPL(X)  ###

FBFE  203CFC RBYTE       JSR RCHAR
FC01  0A                 ASL A
FC02  0A                 ASL A
FC03  0A                 ASL A
FC04  0A                 ASL A
FC05  85F5               STA TEMP+1
FC07  E8                 INX
FC08  203CFC             JSR RCHAR
FC0B  05F5               ORA TEMP+1
FC0D  E8                 INX
FC0E  60                 RTS
FC0F         ;
```

```
-----------------------------------------------
 ###  WRITE BYTE TO DISPL(X)  ###

FC0F  85F5    WBYTE     STA TEMP+1   (00F5)
FC11  4A                LSR A
FC12  4A                LSR A
FC13  4A                LSR A
FC14  4A                LSR A
FC15  202DFC            JSR WCHAR
FC18  E8                INX
FC19  A5F5              LDA TEMP+1
FC1B  290F              AND #$0F
FC1D  202DFC            JSR WCHAR
FC20  E8                INX
FC21  60                RTS
FC22            ;

-----------------------------------------------
 ###  CLEAR DISPLAY TO BLANKS  ###

FC22  A920    CLRDSP    LDA #' '
FC24  A210    FILDSP    LDX #DIBUFL
FC26  9D2001 CD1        STA DIBUFF,X
FC29  CA                DEX
FC2A  10FA              BPL CD1
FC2C  60                RTS
FC2D            ;

-----------------------------------------------
 ###  WRITE CHR TO DISPL(X)  ###

FC2D  8C1B01 WCHAR      STY YSAV
FC30  A8                TAY
FC31  B952FC            LDA CHASET,Y
FC34  9D2001            STA DIBUFF,X
FC37  98                TYA
FC38  AC1B01            LDY YSAV
FC3B  60                RTS
FC3C            ;

-----------------------------------------------
 ###  READ CHR FROM DISPL(X)  ###

FC3C  8E1A01 RCHAR      STX XSAV
FC3F  BD2001            LDA DIBUFF,X
FC42  A213              LDX #NCHARS
FC44  DD52FC RC1        CMP CHASET,X
FC47  F004              BEQ RC2
FC49  CA                DEX
FC4A  10F8              BPL RC1
FC4C  E8                INX
```

```
FC4D  8A      RC2      TXA
FC4E  AE1A01           LDX XSAV
FC51  60               RTS
FC52                   ;
FC52  3031    CHASET   .BYT '01234'
FC57  3536             .BYT '56789'
FC5C  4142             .BYT 'ABCDEF'
FC62  202E3D           .BYT ' .='
FC65                   ;
FC65          NCHARS:  =*-CHASET
FC65                   ;
FC65                   ;  ####  PRINT A TEXT ON THE DISPLAY  ####
FC65                   ;
FC65  85F4    TPRINT   STA TEMP           ; TEXT-ADDRESS IN .YA
FC67  84F5             STY TEMP+1
FC69  2022FC           JSR CLRDSP
FC6C  A000             LDY #0
FC6E  B1F4    TPRNT1   LDA (TEMP)Y
FC70  F006             BEQ TPRNT9         ; *EOT*
FC72  992001           STA DIBUFF,Y
FC75  C8               INY
FC76  D0F6             BNE TPRNT1
FC78  60      TPRNT9   RTS
FC79                   ;
FC79                   ;  ####  TEST THE KEYBOARD/DISPLAY  &  I/O  ####
FC79                   ;
FC79  A900    TESTER   LDA #0             ; INIT VIA
FC7B  8D13E0           STA DDRA1
FC7E  A9FF             LDA #$FF
FC80  8D12E0           STA DDRB1
FC83  A9EC             LDA #$EC           ; CA2=0, CB2=1
FC85  8D1CE0           STA PCR1
FC88  2022FC           JSR CLRDSP
FC8B  A200    TEST1    LDX #0
FC8D  20EDF9  TEST2    JSR GETKEY
FC90  C91B             CMP #ESC
FC92  F012             BEQ TEST3
FC94  9D2001           STA DIBUFF,X
FC97  AD1CE0           LDA PCR1           ; TOGGLE CA2 & CB2
FC9A  4922             EOR #$22
FC9C  8D1CE0           STA PCR1
FC9F  E8               INX
FCA0  E010             CPX #DIBUFL
FCA2  B0E7             BCS TEST1
FCA4  90E7             BCC TEST2
FCA6  2007FE  TEST3    JSR VIAINI
FCA9  AD11E0  TEST4    LDA DRA1           ; ECHO INPUTS TO OUTPUTS
FCAC  8D10E0           STA DRB1
FCAF  B0F8             BCS TEST4          ; ALWAYS
FCB1                   ;
FCB1                   ;
FCB1                   ;
```

```
--------------------------------------------
 ###  INTERRUPT HANDLING  ###

FCB1              ;
FCB1              ;  ###  HANDLER FOR IRQ  ###
FCB1              ;
FCB1  85F4    IRQHDL:  STA TEMP
FCB3  68               PLA
FCB4  48               PHA
FCB5  2910             AND #$10
FCB7  D00F             BNE BRKINT
FCB9  AD1DE0           LDA IFR1
FCBC  2920             AND #$20
FCBE  D00D             BNE STEPIN        ; SST-INTERRUPT TIMER T2
FCC0  AD1DE0           LDA IFR1
FCC3  A5F4             LDA TEMP
FCC5  6CF600           JMP (IRQVEC)
FCC8              ;
FCC8  A5F4    BRKINT   LDA TEMP
FCCA  4CC0F9           JMP BKUSER
FCCD              ;
FCCD  A5F4    STEPIN   LDA TEMP
FCCF  4CC1F8           JMP SSTINT
FCD2              ;
FCD2              ;  ###  HANDLER FOR NMI  ####
FCD2              ;
FCD2  6CF000 NMI:      JMP (NMIVEC)
FCD5              ;
FCD5              ;

--------------------------------------------
 ####  DISPLAY ROUTINE  ####

FCD5              ;
FCD5              ;  ####  HANDLE THE INTERRUPTS  ####
FCD5              ;
FCD5  48      TERMNL:  PHA
FCD6  8A               TXA
FCD7  48               PHA
FCD8  98               TYA
FCD9  48               PHA
FCDA  2013FD           JSR DISPL         ; NEXT CHAR
FCDD  20E6FC           JSR TIMUPD        ; UPDATE TIMERS
FCE0  68               PLA
FCE1  A8               TAY
FCE2  68               PLA
FCE3  AA               TAX
FCE4  68               PLA
FCE5  40               RTI
FCE6              ;
FCE6              ;   UPDATE THE TIMERS
```

```
FCE6             ;
FCE6  CE0401 TIMUPD    DEC DIV1          ; EVERY 512 USEK
FCE9  1027             BPL TIMUP9
FCEB  A914             LDA #20           ; 10.24 MS INTERVAL
FCED  8D0401           STA DIV1
FCF0  A212             LDX #18           ; 8 TIMERS & DBCNTR
FCF2  D8     TIMUP1    CLD
FCF3  38               SEC
FCF4  BD0701           LDA TIMER,X
FCF7  E901             SBC #1
FCF9  9D0701           STA TIMER,X
FCFC  BD0801           LDA TIMER+1,X
FCFF  E900             SBC #0
FD01  9D0801           STA TIMER+1,X
FD04  B008             BCS TIMUP2        ; PAST '0000' ??
FD06  A900             LDA #0
FD08  9D0701           STA TIMER,X
FD0B  9D0801           STA TIMER+1,X
FD0E  CA     TIMUP2    DEX
FD0F  CA               DEX
FD10  10E0             BPL TIMUP1
FD12  60     TIMUP9    RTS
FD13             ;
FD13             ;
FD13             ;  ####  REFRESH THE DISPLAY  ####
FD13             ;
FD13  AE3001 DISPL:    LDX DCPTR         ; SHOW NEXT CHR.
FD16  E8               INX
FD17  E010             CPX #DIBUFL
FD19  9002             BCC DISP1         ; MODULO DIBUFL
FD1B  A200             LDX #0
FD1D  8E3001 DISP1     STX DCPTR         ; SAVE THE PTR
FD20  A910             LDA #$10          ; BLANK CHR-SEL
FD22  8D04E0           STA CHSEL
FD25  BD2001           LDA DIBUFF,X
FD28  D8               CLD               ; CONVERT ASCII TO TABLE-INDEX
FD29  38               SEC
FD2A  E920             SBC #$20
FD2C  C940             CMP #$40
FD2E  9002             BCC DISP2
FD30  E920             SBC #$20          ; L-CASE --> U-CASE
FD32  0A     DISP2     ASL A             ; CHR.VAL 2*
FD33  AA               TAX
FD34  BD5AFD           LDA CHSET,X
FD37  8D03E0           STA SEGM0
FD3A  BD5BFD           LDA CHSET+1,X
FD3D  8D02E0           STA SEGM1
FD40  AD3001           LDA DCPTR
FD43  490F             EOR #$0F          ; INVERSED SEQUENCE
FD45  8D04E0           STA CHSEL
FD48  60               RTS
FD49             ;
```

```
FD49              ;  DISABLE & ENABLE DISPLAY-INTERRUPTS (NMI)
FD49              ;
FD49  A920    BLKDIS    LDA #$20
FD4B  8D07E0            STA NMICTR        ; CONTROL REGISTER
FD4E  A910              LDA #$10
FD50  8D04E0            STA CHSEL         ; BLANK THE DISPLAY
FD53  60                RTS
FD54              ;
FD54  A900    RELDIS    LDA #$00          ; RELEASE DISPLAY-INTERRUPTS
FD56  8D07E0            STA NMICTR
FD59  60                RTS
FD5A              ;
FD5A              ;
FD5A              ;    ####  CHARACTER SET  ####
FD5A              ;
FD5A              ;    FORMAT & HARDWARE : 1ST BYTE BIT 0-7  SEGM A-H  0-7
FD5A              ;                        2ND BYTE BIT 0-7  SEGM I-P  0-7
FD5A              ;
FD5A              ;    A LOW BIT TURNS THE CORRESPONDING SEGMENT ON.
FD5A              ;
FD5A  FFFF    CHSET     .DBYTE $FFFF      ; SPACE
FD5C  CEDB              .DBYTE $CEDB      ; !
FD5E  DFFE              .DBYTE $DFFE      ; "
FD60  31FC              .DBYTE $31FC      ; #
FD62  12FC              .DBYTE $12FC      ; $
FD64  1BC3              .DBYTE $1BC3      ; %
FD66  A2E6              .DBYTE $A2E6      ; &
FD68  FFFB              .DBYTE $FFFB      ; '
FD6A  FFEB              .DBYTE $FFEB      ; (
FD6C  FFD7              .DBYTE $FFD7      ; )
FD6E  3FC0              .DBYTE $3FC0      ; *
FD70  3FFC              .DBYTE $3FFC      ; +
FD72  FFDF              .DBYTE $FFDF      ; ,
FD74  3FFF              .DBYTE $3FFF      ; -
FD76  FF7F              .DBYTE $FF7F      ; .
FD78  FFDB              .DBYTE $FFDB      ; /
FD7A  C0DB              .DBYTE $C0DB      ; 0
FD7C  FFFC              .DBYTE $FFFC      ; 1
FD7E  24FF              .DBYTE $24FF      ; 2
FD80  70FF              .DBYTE $70FF      ; 3
FD82  19FF              .DBYTE $19FF      ; 4
FD84  96EF              .DBYTE $96EF      ; 5
FD86  02FF              .DBYTE $02FF      ; 6
FD88  F8FF              .DBYTE $F8FF      ; 7
FD8A  00FF              .DBYTE $00FF      ; 8
FD8C  10FF              .DBYTE $10FF      ; 9
FD8E  F6FF              .DBYTE $F6FF      ; :
FD90  BFDF              .DBYTE $BFDF      ; ;
FD92  F7DB              .DBYTE $F7DB      ; <
FD94  37FF              .DBYTE $37FF      ; =
FD96  F7E7              .DBYTE $F7E7      ; >
FD98  7CFD              .DBYTE $7CFD      ; ?
```

```
1021  FD9A  A0FD           .DBYTE $A0FD      ; @
1022  FD9C  08FF           .DBYTE $08FF      ; A
1023  FD9E  70FC           .DBYTE $70FC      ; B
1024  FDA0  C6FF           .DBYTE $C6FF      ; C
1025  FDA2  F0FC           .DBYTE $F0FC      ; D
1026  FDA4  06FF           .DBYTE $06FF      ; E
1027  FDA6  0EFF           .DBYTE $0EFF      ; F
1028  FDA8  42FF           .DBYTE $42FF      ; G
1029  FDAA  09FF           .DBYTE $09FF      ; H
1030  FDAC  F6FC           .DBYTE $F6FC      ; I
1031  FDAE  E1FF           .DBYTE $E1FF      ; J
1032  FDB0  8FEB           .DBYTE $8FEB      ; K
1033  FDB2  C7FF           .DBYTE $C7FF      ; L
1034  FDB4  C9F3           .DBYTE $C9F3      ; M
1035  FDB6  C9E7           .DBYTE $C9E7      ; N
1036  FDB8  C0FF           .DBYTE $C0FF      ; O
1037  FDBA  0CFF           .DBYTE $0CFF      ; P
1038  FDBC  C0EF           .DBYTE $C0EF      ; Q
1039  FDBE  0CEF           .DBYTE $0CEF      ; R
1040  FDC0  12FF           .DBYTE $12FF      ; S
1041  FDC2  FEFC           .DBYTE $FEFC      ; T
1042  FDC4  C1FF           .DBYTE $C1FF      ; U
1043  FDC6  CFDB           .DBYTE $CFDB      ; V
1044  FDC8  C9CF           .DBYTE $C9CF      ; W
1045  FDCA  FFC3           .DBYTE $FFC3      ; X
1046  FDCC  FFF1           .DBYTE $FFF1      ; Y
1047  FDCE  F6DB           .DBYTE $F6DB      ; Z
1048  FDD0  F9EB           .DBYTE $F9EB      ; [
1049  FDD2  FFE7           .DBYTE $FFE7      ; \
1050  FDD4  CFD7           .DBYTE $CFD7      ; ]
1051  FDD6  FFCF           .DBYTE $FFCF      ; ^
1052  FDD8  F7FF           .DBYTE $F7FF      ; _
1053  FDDA        ;
1054  FDDA        ;
```

```
-----------------------------------------------
 ###  CASSETTE DUMP  ###

FDDA             ;
FDDA             ;  *********************************
FDDA             ;  *****                       *****
FDDA             ;  *****  DUMP MEMORY TO TAPE  *****
FDDA             ;  *****   IN  KIM-6502 FORMAT *****
FDDA             ;  *****                       *****
FDDA             ;  *********************************
FDDA             ;
FDDA             ;
FDDA             F3700     =138              ; FOR 3700 HERTZ
FDDA             F2400     =207              ; FOR 2400 HERTZ
FDDA             ;
FDDA             ;

-----------------------------------------------
 ###  DUMP  MAIN ROUTINE  ###

FDDA  2049FD DUMP:        JSR BLKDIS         ; BLOCK DISPLAY
FDDD  20ECFD              JSR DMPOPN
FDE0  2036FE              JSR DMPMEM
FDE3  2054FE              JSR DMPCLS
FDE6  2054FD              JSR RELDIS         ; RELEASE DISPLAY
FDE9  4CE4F8              JMP SEQADR
FDEC             ;
FDEC             ;  ###  OPEN FOR DUMP  ###
FDEC  20F3FD DMPOPN       JSR DMPINI
FDEF  200DFE              JSR DMPHED
FDF2  60                  RTS
FDF3             ;
FDF3             ;  ###  INIT HARDWARE FOR DUMP  ###
FDF3  A9C0   DMPINI       LDA #$C0           ; T1 FREE-RUNNING, PB7 PULS
FDF5  8D1BE0              STA ACR1
FDF8  2007FE              JSR VIAINI
FDFB  A900                LDA #0
FDFD  8D1E01              STA CHKSML
FE00  8D1F01              STA CHKSMH
FE03  8D15E0              STA T1CH1
FE06  60                  RTS
FE07             ;
FE07  A9C0   VIAINI       LDA #$C0           ; SET CB2 LOW
FE09  8D1CE0              STA PCR1
FE0C  60                  RTS
FE0D             ;
FE0D             ;  ###  DUMP HEADER  ###
FE0D  A264   DMPHED       LDX #100
FE0F  86F4                STX TEMP
FE11  A916                LDA #$16
FE13  208DFE NXTSNC       JSR DMPACC         ; PRE-AMBLE
```

```
FE16  C6F4             DEC TEMP
FE18  D0F9             BNE NXTSNC
FE1A           ;
FE1A  A92A             LDA #'*
FE1C  208DFE           JSR DMPACC        ; TRIGGER CHR
FE1F  AD0001           LDA ID
FE22  2080FE           JSR DMPBYT        ; ID. NUMBER
FE25  A5F8             LDA ADL
FE27  206EFE           JSR ADDCK         ; START ADDRESS
FE2A  2080FE           JSR DMPBYT
FE2D  A5F9             LDA ADH
FE2F  206EFE           JSR ADDCK
FE32  2080FE           JSR DMPBYT
FE35  60               RTS
FE36           ;
FE36           ; ###  DUMP BODY (MEMORY)  ###
FE36  A000     DMPMEM  LDY #0
FE38  B1F8             LDA (ADL),Y
FE3A  206EFE           JSR ADDCK
FE3D  2080FE           JSR DMPBYT
FE40  E6F8             INC ADL
FE42  D002             BNE DM1
FE44  E6F9             INC ADH
FE46  38       DM1     SEC
FE47  AD0201           LDA EAL
FE4A  E5F8             SBC ADL
FE4C  AD0301           LDA EAH
FE4F  E5F9             SBC ADH
FE51  B0E3             BCS DMPMEM
FE53  60               RTS
FE54           ;
FE54           ; ###  DUMP END-RECORD  ###
FE54  A92F     DMPCLS  LDA #'/
FE56  208DFE           JSR DMPACC
FE59  AD1E01           LDA CHKSML
FE5C  2080FE           JSR DMPBYT        ; CHECKSUM
FE5F  AD1F01           LDA CHKSMH
FE62  2080FE           JSR DMPBYT
FE65  A904             LDA #04
FE67  208DFE           JSR DMPACC        ; 'EOT'
FE6A  208DFE           JSR DMPACC
FE6D  60               RTS
FE6E           ;
FE6E           ; ###  ADD BYTE TO CHECKSUM  ###
FE6E  48       ADDCK   PHA
FE6F  18               CLC
FE70  6D1E01           ADC CHKSML
FE73  8D1E01           STA CHKSML
FE76  AD1F01           LDA CHKSMH
FE79  6900             ADC #0
FE7B  8D1F01           STA CHKSMH
FE7E  68               PLA
```

```
FE7F  60                     RTS
FE80                 ;
FE80                 ;  ###  DUMP ONE BYTE  ###
FE80  20C3FE DMPBYT          JSR HXASCH         ; HEX --> ASCII  HIGH-BYTE
FE83  208DFE                 JSR DMPACC
FE86  20CBFE                 JSR HXASCL         ; HEX --> ASCII  LOW-BYTE
FE89  208DFE                 JSR DMPACC
FE8C  60                     RTS
FE8D                 ;
FE8D                 ;  ###  DUMP ACCUMULATOR  ###
FE8D  A208   DMPACC          LDX #8
FE8F  48                     PHA
FE90  48                     PHA
FE91  18     NXTBIT          CLC
FE92  20A5FE                 JSR CPULSE         ; 3700 HZ
FE95  68                     PLA
FE96  4A                     LSR A
FE97  48                     PHA
FE98  20A5FE                 JSR CPULSE         ; 3700 OR 2400 HZ
FE9B  38                     SEC
FE9C  20A5FE                 JSR CPULSE         ; 2400 HZ
FE9F  CA                     DEX
FEA0  D0EF                   BNE NXTBIT
FEA2  68                     PLA
FEA3  68                     PLA
FEA4  60                     RTS
FEA5                 ;
FEA5                 ;  ###  MAKE A PULSE  ###
FEA5  A900   CPULSE          LDA #0             ; BIT IS IN CARRY
FEA7  2A                     ROL A
FEA8  A8                     TAY
FEA9  B9BFFE                 LDA FREQ,Y
FEAC  8D16E0                 STA T1LL1
FEAF  B9C1FE                 LDA NPLS,Y
FEB2  A8                     TAY
FEB3  2C1DE0 WAITPL          BIT IFR1
FEB6  50FB                   BVC WAITPL
FEB8  AD14E0                 LDA T1CL1          ; CLEAR INT. FLAG
FEBB  88                     DEY
FEBC  D0F5                   BNE WAITPL
FEBE  60                     RTS
FEBF                 ;
FEBF  88     FREQ           .BYTE F3700-2
FEC0  CD                    .BYTE F2400-2
FEC1  12     NPLS           .BYTE 18            ; NUMBER OF HALF-PULSES
FEC2  0C                    .BYTE 12
FEC3                 ;
FEC3                 ;  ###  CONVERTION HEX --> ASCII  ###
FEC3  85F5   HXASCH          STA TEMP+1         ; ENTRY FOR HIGH-BYTE
FEC5  4A                     LSR A
FEC6  4A                     LSR A
FEC7  4A                     LSR A
```

```
FEC8  4A                LSR A
FEC9  1004              BPL HEXASC
FECB  A5F5    HXASCL    LDA TEMP+1          ; ENTRY FOR LOW-BYTE
FECD  290F              AND #$0F
FECF          HEXASC
FECF  C90A              CMP #$0A
FED1  9002              BCC HA1
FED3  6906              ADC #6
FED5  6930    HA1       ADC #$30
FED7  60                RTS
FED8          ;
```

```
--------------------------------------------
 ###  CASSETTE LOAD  ###

FED8            ;
FED8            ;   ***********************************
FED8            ;   ***********************************
FED8            ;   *****                         *****
FED8            ;   *****  LOAD MEMORY FROM TAPY  *****
FED8            ;   *****  IN 'KIM-6502'  FORMAT  *****
FED8            ;   *****                         *****
FED8            ;   ***********************************
FED8            ;   ***********************************
FED8            ;
FED8            ;

--------------------------------------------
 ###  LOAD   MAIN ROUTINE  ###

FED8  2049FD LOAD:     JSR BLKDIS          ; BLOCK DISPLAY
FEDB  A9E0             LDA #$E0            ; CB2 HIGH
FEDD  8D1CE0           STA PCR1
FEE0  A97F             LDA #$7F            ; PB7 FOR INPUT
FEE2  8D12E0           STA DDRB1
FEE5  A900             LDA #0
FEE7  8D1BE0           STA ACR1            ; T2 ONE-SHOT
FEEA  8D1E01           STA CHKSML          ; CLEAR CHECKSUM
FEED  8D1F01           STA CHKSMH
FEF0            ;
FEF0            ;  ###  READ THE SYNC-BYTE'S  ###
FEF0  20BBFF LODSNC    JSR RDBIT
FEF3  6A               ROR A
FEF4  C916             CMP #$16
FEF6  D0F8             BNE LODSNC
FEF8  A909             LDA #9
FEFA  85F5             STA TEMP+1
FEFC  20AFFF LS1       JSR LDACC
FEFF  C916             CMP #$16
FF01  D0ED             BNE LODSNC
FF03  C6F5             DEC TEMP+1
FF05  10F5             BPL LS1
FF07            ;
FF07            ;  ###  FIND TRIGGER-WORD  ###
FF07  20AFFF LODSTR    JSR LDACC
FF0A  C92A             CMP #'*
FF0C  F006             BEQ LS2
FF0E  C916             CMP #$16
FF10  F0F5             BEQ LODSTR
FF12  D0DC             BNE LODSNC
FF14  2086FF LS2       JSR RDBYTE
FF17  8D0101           STA TAPID           ; LEAVE CURRENT ID.
FF1A  CD0001           CMP ID
```

```
FF1D  F01A              BEQ LODSAD        ; LOAD EVERY FILE
FF1F  AD0001            LDA ID
FF22  C900              CMP #0
FF24  F013              BEQ LODSAD
FF26  C9FF              CMP #$FF          ; ID=$FF 'RELOCATE & LOAD'
FF28  D0C6              BNE LODSNC        ; FIND NEXT FILE
FF2A              ;
FF2A  2086FF            JSR RDBYTE
FF2D  206EFE            JSR ADDCK
FF30  2086FF            JSR RDBYTE
FF33  206EFE            JSR ADDCK
FF36  4C49FF            JMP LODDAT
FF39              ;
FF39              ;  ###  GET START ADDRESS  ###
FF39  2086FF LODSAD     JSR RDBYTE
FF3C  206EFE            JSR ADDCK
FF3F  85F8              STA ADL
FF41  2086FF            JSR RDBYTE
FF44  206EFE            JSR ADDCK
FF47  85F9              STA ADH
FF49              ;
FF49              ;  ###  LOAD DATA TO MEM  ###
FF49  2086FF LODDAT     JSR RDBYTE
FF4C  B00F              BCS LODEND        ; *EOF*
FF4E  206EFE            JSR ADDCK
FF51  A000              LDY #0
FF53  91F8              STA (ADL),Y
FF55  E6F8              INC ADL
FF57  D0F0              BNE LODDAT
FF59  E6F9              INC ADH
FF5B  D0EC              BNE LODDAT
FF5D              ;
FF5D              ;  ###  TEST CHECKSUM  ###
FF5D  2086FF LODEND     JSR RDBYTE
FF60  CD1E01            CMP CHKSML
FF63  D008              BNE LE1
FF65  2086FF            JSR RDBYTE
FF68  CD1F01            CMP CHKSMH
FF6B  F00D              BEQ LE2
FF6D  A980   LE1        LDA #$80
FF6F  0D0101            ORA TAPID         ; MARK TAPE-ERROR
FF72  8D0101            STA TAPID
FF75  A945              LDA #'E
FF77  8D0501            STA CURCMD
FF7A  AD18E0 LE2        LDA T2LL1         ; CLEAR T2 INT. FOR 'SST'
FF7D  2054FD            JSR RELDIS        ; RELEASE DISPLAY
FF80  2007FE            JSR VIAINI
FF83  4CE4F8            JMP SEQADR
FF86              ;
FF86              ;  ###  READ ONE BYTE  ###
FF86  209AFF RDBYTE     JSR RDHEX
FF89  B00E              BCS RDB1          ; *EOF*
```

```
FF8B  0A            ASL A
FF8C  0A            ASL A
FF8D  0A            ASL A
FF8E  0A            ASL A
FF8F  85F4          STA TEMP
FF91  209AFF        JSR RDHEX
FF94  B003          BCS RDB1          ; *EOF*
FF96  05F4          ORA TEMP
FF98  18            CLC
FF99  60     RDB1   RTS
FF9A         ;
FF9A         ; ###  READ A HEX NUMBER  ###
FF9A  20AFFF RDHEX  JSR LDACC
FF9D  C92F          CMP #'/
FF9F  F00C          BEQ RH2           ; '*EOF*'
FFA1  38            SEC
FFA2  E930          SBC #$30
FFA4  C90A          CMP #$0A
FFA6  3004          BMI RH1
FFA8  38            SEC
FFA9  E907          SBC #$07
FFAB  18            CLC               ; RETURN WITH HEX NUMBER
FFAC  60     RH1    RTS
FFAD  38     RH2    SEC               ; RETURN WITH *EOF* MARKER
FFAE  60            RTS
FFAF         ;
FFAF         ; ###  READ 8 BIT'S TO ACC  ###
FFAF  A900   LDACC  LDA #0
FFB1  A207          LDX #7
FFB3  20BBFF LDA1   JSR RDBIT
FFB6  6A            ROR A
FFB7  CA            DEX
FFB8  10F9          BPL LDA1
FFBA  60            RTS
FFBB         ;
FFBB         ; ###  READ ONE BIT  ###
FFBB  48     RDBIT  PHA
FFBC  20CFFF RB0    JSR GETFRQ
FFBF  F0FB          BEQ RB0           ; WAIT FOR 2400 HZ
FFC1  A000          LDY #0
FFC3  20CFFF RB1    JSR GETFRQ
FFC6  F003          BEQ RB2           ; WAIT FOR 3700 HZ
FFC8  C8            INY
FFC9  10F8          BPL RB1
FFCB  C009   RB2    CPY #9            ; BIT IS IN CARRY
FFCD  68            PLA
FFCE  60            RTS
FFCF         ;
FFCF         ; ###  TIME PULSE LENGTH  ###
FFCF         ;
FFCF         ;   THE COMPLETE PERIODE-TIME IS DETERMINED,
FFCF         ;   SO THE LOGIC IS INSENITIVE FOR FASE-SHIFTS.
FFCF          ;
FFCF          TPERD    =316              ; PERIODE-CENTER-TIME
FFCF          ;
FFCF  2C10E0  GETFRQ   BIT DRB1
FFD2  30FB             BMI GETFRQ        ; WAIT TILL LOW
FFD4  2C10E0           BIT DRB1          ; & DEBOUNCE
FFD7  30F6             BMI GETFRQ
FFD9  2C10E0  GFO      BIT DRB1
FFDC  10FB             BPL GFO           ; WAIT TILL HIGH
FFDE  2C10E0           BIT DRB1          ; & DEBOUNCE
FFE1  10F6             BPL GFO
FFE3  AD1DE0           LDA IFR1          ; READ T2-STATUS OF LAST
FFE6  48               PHA               ; PERIODE
FFE7  A93C             LDA #<TPERD       ; AND RETRIGGER
FFE9  8D18E0           STA T2LL1
FFEC  A901             LDA #>TPERD
FFEE  8D19E0           STA T2CH1
FFF1  68               PLA
FFF2  2920             AND #$20
FFF4  60               RTS
FFF5          ;
FFF5          ;
FFF5          ;  ####  RESET & INTERRUPT VECTORS  ####
FFF5          ;
FFF5                   *=ROMBAS-7+$800   ; FOR 2K (P)ROM
FFF9          ;
FFF9  FA               .BYT $FA
FFFA  D2FC             .WOR NMI,RESET,IRQHDL
0000          ;
0000          REND:
0000          ;
0000          ;
0000                   .OPT NOOBJECT
0000          ;
0000          ;  ####  LINK IRQ TO THE 65-SYSTEM VECTOR'S  ####
0000          ;
0000                   *=$C400           ; FOR DEBUG-PACKAGE LINKING
C400  B1FC             .WOR IRQHDL,NMI,IRQHDL
C406          ;
C406          ;
C406                   .END

ERRORS: 0000                          <0000>
```