```
NAM SBUG9
*
** MONITOR MEMORY MAP
*
RAM     EQU    $F0C0     START OF MONITOR RAM
ROM     EQU    $F800     START OF MONITOR ROM
EXTIO   EQU    $E000     OFF BOARD IO BASE
INTIO   EQU    $E400     ON BOARD DEVICES
*
* OPTION SWITCHES
*
VIDOPT EQU $00 NO VIDEO
PRTOPT EQU $FF PRINTER
CLKOPT EQU $FF REAL TIME CLOCK
FDCOPT EQU $FF FLOPPY DISK
*
*******************************************************
*
* IO EQUATES:
*
*******************************************************
*
** ACIA - TTY INTERFACE
*
ACIAS  EQU INTIO+$00
 IF PRTOPT
*
** PRINTER INTERFACE
*
PADATA EQU INTIO+$04
PACTRL EQU INTIO+$05
PBDATA EQU INTIO+$06
PBCTRL EQU INTIO+$07
*
** CB1  ACK.  I/P
** CB2  STB.  O/P
** PB0 - PB7   DATA 1 - 8   O/P
** PORT A BIT ASSIGNMENT
*
PBUSY  EQU $80 I/P
PEMPTY EQU $40 I/P
SELECT EQU $20 I/P
PERROR EQU $10 I/P
PRESET EQU %00000100 O/P PA3 = 0
AUTOFD EQU %00001000 O/P PA2 = 0
DIRMSK EQU %00001100
 ENDIF PRTOPT
 IF FDCOPT
*
** FLOPPY DISK CONTROLLER
*
DRVREG EQU EXTIO+$14
CMDREG EQU EXTIO+$18
SECREG EQU EXTIO+$1A
DATREG EQU EXTIO+$1B
 ENDIF FDCOPT
 IF VIDOPT
```

```
*
** VIDEO DISPLAY DEFINITIONS
*
SCREEN  EQU    $E800     START OF SCREEN MEMORY
LINLEN  EQU    64       LENGTH OF A LINE
NUMLIN  EQU    16       NUMBER OF LINES
SCNLEN  EQU    $400      LENGTH OF SCREEN
 ENDIF VIDOPT
 IF CLKOPT
*
*
* MM58167A REAL TIME CLOCK MEMORY MAP:
*
CLOCK  EQU $E040 CLOCK BASE ADDRESS AND REGISTERS
*
* CLOCK REGISTER OFFSETS:
* These register offsets are used for the CLOCK
* and comparitor ram CMPRAM.
*
S10000 EQU 0 TEN THOUNSANDTHS OF SECONDS
S100   EQU 1 HUNDRETHS AND TENTHS OF SECONDS
SECOND EQU 2
MINUIT EQU 3
HOUR   EQU 4
WKDAY  EQU 5
MTHDAY EQU 6
MONTH  EQU 7
*
* COUNTER AND COMPARITOR REGISTERS:
*
* Both the Clock Counter and Clock Comparitor
* consist of 8 registers for holding the time.
* The register offsets from the Counter and
* Comparitor registers are listed above.
*
COUNTR EQU CLOCK+0
CMPRAM EQU CLOCK+8 COMPARITOR REGISTERS
*
* INTERRUPT OUTPUT REGISTERS:
*
* An interrupt output may be generated at the
* following rates by setting the appropriate bit
* in the Interrupt Control Register (CINTCR).
* The Interrupt Status Register (CINTSR) must be
* read to clear the interrupt and will return
* the source of the interrupt.
*
* 1/Month      Bit 7
* 1/Week       Bit 6
* 1/Day        Bit 5
* 1/Hour       Bit 4
* 1/Minuite    Bit 3
* 1/Second     Bit 2
* 10/Second    Bit 1
* Comparitor   Bit 0
*
CINTSR  EQU CLOCK+16 INTERRUPT STATUS REGISTER
```

```
CINTCR  EQU CLOCK+17 INTERRUPT CONTROL REGISTER
*
* COUNTER AND RAM RESETS; GO COMMAND.
*
* The counter and comparitor may be reset
* by writing $FF into CTRRES and CMPRES
* respectivly.
* A write to the Go command register (GOCMND)
* will reset the 1/1000ths, 1/100ths and 1/10ths
* of a second counter.
*
CTRRES EQU CLOCK+18 COUNTER RESET
CMPRES EQU CLOCK+19 COMPARITOR RAM RESET
GOCMND EQU CLOCK+21 GO COMMAND
*
* CLOCK STATUS REGISTER.
*
* The counter takes 61 usec. to rollover for
* every 1KHz clock pulse. If the Status bit is
* set after reading the counter, the counter
* should be re-read to ensure the time is correct.
*
CLKSTA EQU CLOCK+20 STATUS BIT
SBYINT EQU CLOCK+22 STANDBY INTERRUPT
TSTMOD EQU CLOCK+31 TEST MODE REGISTER
 ENDIF CLKOPT
*******************************************************
*
* RAM ALLOCATION:
*
*******************************************************
*
** ALTERABLE INTERRUPT VECTOR TABLE
*
 ORG RAM
STACK RMB 2 $F0C0
SWI3 RMB 2 $F0C2
SWI2 RMB 2 $F0C4
FIRQ RMB 2 $F0C6
IRQ RMB 2 $F0C8
SWI RMB 2 $F0CA
SVCVO RMB 2 $F0CC
SVCVL RMB 2 $F0CE
*
** I/O CONTROL PARAMETERS
*
CPORT RMB 2 $F0E0 - REVECTORABLE CONTROL PORT
ECHO RMB 1 $F0E2 - ECHO FLAG
 IF VIDOPT
************************************************
*   DG640 MEMORY MAPPED DISPLAY DRIVER VARIABLES  *
************************************************

***** ALWAYS KEEP THESE TWO BYTES TOGETHER *****
COLADX  RMB    1         CURSOR COLUMN
ROWADX  RMB    1         CURSOR ROW
************************************************
```

```
CURSOR  RMB    2          ABSOLUTE SCREEN ADDRESS
NEWROW  RMB    1          NEW ROW TEMP FOR ESCAPE
ESCFLG  RMB    1          ESCAPE SEQUENCE ACTIVE
 ENDIF VIDOPT
*
* ACIA/VDU OUTPUT SWITCH
*
OUTFLG  RMB    1        ACIA/VDU OUTPUT SWITCH
VIDBIT EQU $01 VIDEO OUTPUT
SERBIT EQU $02 SERIAL OUTPUT
PRTBIT EQU $04 PRINTER OUTPUT
*
BPTBL RMB 24 $F0E3 - BREAK POINT TABLE BASE ADDRESS
*
** MONITOR ROUTINE JUMP VECTORS
*
 ORG ROM
 FDB MONITOR
 FDB NEXTCMD
 FDB INCH
 FDB INCHE
 FDB INCHEK
 FDB OUTCH
 FDB PDATA
 FDB PCRLF
 FDB PSTRNG
 FDB LRA
 IF PRTOPT
 FDB PCHK  CHECK FOR PRINTER INPUT
 FDB PINIZ INITIATE PRINTER
 FDB POUTCH OUTPUT CH. TO PRINTER
 ELSE
 FDB DUMRTS
 FDB DUMRTS
 FDB DUMRTS
 ENDIF PRTOPT
 IF VIDOPT
 FDB VINIZ
 FDB VOUTCH
 ELSE
 FDB DUMRTS
 FDB DUMRTS
 ENDIF VIDOPT
 FDB ACINIZ
 FDB AOUTCH
*
** POWER ON RESET
*
START LDS #STACK
*
** MONITOR ENTRY POINT
*
** MOVE ROM INTERRUPT JUMP VECTORS TO RAM
*
MONITOR LDX #RAMVEC
 LDY #STACK
 LDB #$10
```

```
LOOPA LDA ,X+
 STA ,Y+
 DECB
 BNE LOOPA
*
** SET UP CONTROL PORT POINTER
** (USUALLY AN ACIA)
*
 LDX #ACIAS
 STX CPORT
 LBSR XBKPNT
*
** CLEAR 12 ENTRIES ON STACK
*
 LDB #$0C
CLRSTK CLR ,-S
 DECB
 BNE CLRSTK
 LEAX MONITOR,PCR PRESET PC TO MONITOR
 STX $0A,S
 LDA #$D0 PRESET C.C.
 STA ,S
 TFR S,U
*
** INITIATE VDU AND ACIA
*
 LDA #SERBIT START UP ON ACIA
 STA OUTFLG
 LBSR ACINIZ
 IF VIDOPT
 LBSR VINIZ
 ENDIF VIDOPT
 IF CLKOPT
 LBSR CLKINZ
 ENDIF CLKOPT
 LDX #TTLMSG
 LBSR PDATA
*
** GET COMMAND FROM OPERATOR
*
NEXTCMD LDX #PMTMSG
 LBSR PSTRNG
 LBSR INCH
 ANDA #$7F
 CMPA #$0D
 BEQ NEXTCMD
 TFR A,B
 CMPA #$20
 BGE PRTCMD
 LDA #$5E
 LBSR OUTCH
 TFR B,A
 ADDA #$40
PRTCMD LBSR OUTCH
 LBSR OUT1S
*
** DO COMMAND TABLE LOOKUP
```

```
*
 LDX #JMPTAB
NXTCHR CMPB ,X+
 BEQ JMPCMD
 LEAX $02,X
 CMPX #RAMVEC
 BNE NXTCHR
 LDX #QRYMSG
 LBSR PDATA
 BRA NEXTCMD
JMPCMD JSR [,X]
 BRA NEXTCMD
*
** "M" MEMORY EXAMINE AND CHANGE
*
MEMCHG LBSR IN1ADR
 BVS CHRTN
 TFR X,Y
MEMC2 LDX #SEPMSG
 LBSR PSTRNG
 TFR Y,X
 LBSR OUT4H
 LBSR OUT1S
 LDA ,Y
 LBSR OUT2H
 LBSR OUT1S
 LBSR BYTE
 BVC CHANGE
 CMPA #$08
 BEQ MEMC2
 CMPA #$18
 BEQ MEMC2
 CMPA #$5E
 BEQ BACK
 CMPA #$0D
 BNE FORWRD
CHRTN RTS
*
CHANGE STA ,Y
 CMPA ,Y
 BEQ FORWRD
 LBSR OUT1S
 LDA #$3F
 LBSR OUTCH
FORWRD LEAY $01,Y
 BRA MEMC2
*
BACK LEAY -$01,Y
 BRA MEMC2
*
** "S" DISPLAY CONTENTS OF STACK
*
DISSTK LBSR PRTSP
 TFR U,Y
 LDX #STACK
 LEAX -$01,X
 BRA MDUMP1
```

```
*
MEMDUMP LBSR IN2ADR
 BVS EDPRTN
MDUMP1 PSHS Y
 CMPX ,S++
 BCC AJDUMP
EDPRTN RTS
*
* ADJUST LOWER AND UPPER ADDRESS LIMITS
* TO EVEN 16 BYTE BOUNDARIES.
* ENTER WITH LOWER ADDRESS IN XREG
*            UPPER ADDRESS ON TOP OF STACK.
*
AJDUMP TFR X,D
 ADDD #$0010
 ANDB #$F0
 PSHS B,A
 TFR Y,D
 ANDB #$F0
 TFR D,X
NXTLIN CMPX ,S
 BEQ SKPDMP
 LBSR INCHEK
 BEQ EDUMP
SKPDMP LEAS $02,S
 RTS
*
** "E" DUMP MEMORY
*
EDUMP PSHS X
 LDX #SEPMSG
 LBSR PSTRNG
 LDX ,S
 LBSR OUT4H
 LBSR OUT2S
 LDB #$10
ELOOP LDA ,X+
 LBSR OUT2H
 LBSR OUT1S
 DECB
 BNE ELOOP
 LBSR OUT2S
 LDX ,S++
 LDB #$10
EDPASC LDA ,X+
 CMPA #$20
 BCS PERIOD
 CMPA #$7E
 BLS PRASC
PERIOD LDA #$2E
PRASC LBSR OUTCH
 DECB
 BNE EDPASC
 BRA NXTLIN
*
** GO TO SPECIFIED ADDRESS (SET PC AND CONTINUE)
*
```

```
GO      LBSR   ALTPC1    SET UP NEW PC
        BVC    CONTIN    ADDRESS OK, GO
        RTS              NOT OK, BOMB OUT
CONTIN  TFR U,S
RTI     RTI
*
** CHANGE OUTPUT DEVICE
*
SETOUT CLRB DISABLE ALL
SETOU1 LBSR INCH GET ARGUMENT
 CMPA #'S SERIAL ?
 BNE SETOU2
 LBSR ACINIZ YES,INITIATE
 ORB SERBIT YES, SET BIT
 BRA SETOU1

SETOU2 EQU *
 IF VIDOPT
 CMPA #'V VIDEO ?
 BNE SETOU3
 LBSR VINIZ
 ORB #VIDBIT
 BRA SETOU1

 ENDIF VIDOPT
SETOU3 EQU *
 IF PRTOPT
 CMPA #'P PRINTER ?
 BNE SETOU4
 LBSR PINIZ
 ORB #PRTBIT
 BRA SETOU1

 ENDIF PRTOPT
SETOU4 STB OUTFLG SET UP FLAG
 RTS
*
** "B" SET BREAKPOINT
*
BRKPNT LBSR IN1ADR
 BVS EXITBP
 CMPX #STACK
 BCC BPERR
 PSHS X
 LDX #$FFFF
 BSR BPTEST
 PULS X
 BEQ BPERR
 LDA ,X
 CMPA #$3F
 BEQ BPERR
 STA ,Y+
 STX ,Y
 LDA #$3F
 STA ,X
EXITBP RTS
*
```

```
BPERR LBSR OUT1S
 LDA #$3F
 LBRA OUTCH
XBKPNT LDY #BPTBL
 LDB #$08
XBPLP BSR RPLSWI
 DECB
 BNE XBPLP
 RTS
*
** SWI ENTRY POINT
*
SWIE TFR S,U
 LDX $0A,U
 LEAX -$01,X
 BSR BPTEST
 BEQ REGPR
 STX $0A,U
 BSR RPLSWI
REGPR LBSR REGSTR
 LBRA NEXTCMD
RPLSWI LDX $01,Y
 CMPX #STACK
 BHS FFSTBL
 LDA ,X
 CMPA #$3F
 BNE FFSTBL
 LDA ,Y
 STA ,X
FFSTBL LDA #$FF
 STA ,Y+
 STA ,Y+
 STA ,Y+
LRA EQU *
DUMRTS EQU *
 RTS
*
BPTEST LDY #BPTBL
 LDB #$08
FNDBP LDA ,Y+
 CMPX ,Y++
 BEQ BPADJ
 DECB
 BNE FNDBP
 RTS
BPADJ LEAY -$03,Y
 RTS
*
*
** "L" LOAD S1 FORMAT TAPE
*
LOAD LDA #$11
 BSR LDOUTC
 CLR ECHO
LOAD1 LBSR ECHON
LOAD2 CMPA #$53
 BNE LOAD1
```

```
 LBSR ECHON
 CMPA #$39
 BEQ LOAD21
 CMPA #$31
 BNE LOAD2
 LBSR BYTE
 PSHS A
 BVS LODERR
 LBSR IN1ADR
 BVS LODERR
 PSHS X
 LDB ,S+
 ADDB ,S+
 ADDB ,S
 DEC ,S
 DEC ,S
LOAD10 PSHS B
 LBSR BYTE
 PULS B
 BVS LODERR
 PSHS A
 ADDB ,S+
 DEC ,S
 BEQ LOAD16
 STA ,X+
 BRA LOAD10
*
LODERR CLRB
LOAD16 PULS A
 CMPB #$FF
 BEQ LOAD
 LDA #$3F
 BSR LDOUTC
LOAD21 COM ECHO
 LDA #$13
LDOUTC LBRA OUTCH
*
** "P" PUNCH S1 FORMAT TAPE
*
PUNCH CLR ,-S
 BSR IN2ADR
 PSHS Y,X
 BVS PUNEXT
 CMPX $02,S
 BCS PUNEXT
 LEAX $01,X
 STX ,S
 LDA #$12
 LBSR OUTCH
PUNCH2 LDD ,S
 SUBD $02,S
 BEQ PUNCH3
 CMPD #$0020
 BLS PUNCH4
PUNCH3 LDB #$20
PUNCH4 STB $04,S
 LDX #S1MSG
```

```
 LBSR PSTRNG
 ADDB #$03
 TFR B,A
 LBSR OUT2H
 LDX $02,S
 LBSR OUT4H
 ADDB $02,S
 ADDB $03,S
PUNCHL ADDB ,X
 LDA ,X+
 LBSR OUT2H
 DEC $04,S
 BNE PUNCHL
 COMB
 TFR B,A
 LBSR OUT2H
 STX $02,S
 CMPX ,S
 BNE PUNCH2
PUNEXT LDA #$14
 BSR OUT1C
 LEAS $05,S
 RTS
*
** MEMORY BLOCK MOVE
*
BLKMOV BSR IN2ADR GET START AND END
 BVS BLKMV3 BAD ENTRY
 PSHS X SAVE SRC END
 LDA #'>
 BSR OUT1C
 BSR IN1ADR GET DESTINATION
 BVS BLKMV2 ERROR IN DEST
BLKMV1 LDA ,Y+ GET BYTE
 STA ,X+ STORE IT
 CMPY ,S REACH END ADDRESS
 BNE BLKMV1
BLKMV2 PULS X RESTORE STACK
BLKMV3 RTS
*
** GET TWO VALID ADDRESSES
*
IN2ADR BSR IN1ADR
 BVS NOTHEX
 TFR X,Y
 LDA #$2D
 BSR OUT1C
*
** GET ONE ADDRESS
*
IN1ADR BSR BYTE
 BVS NOTHEX
 TFR D,X
 BSR BYTE
 BVS NOTHEX
 PSHS X
 STA $01,S
```

```
 PULS X,PC
*
** READ A HEX BYTE
*
BYTE BSR INHEX
 BVS NOTHEX
 ASLA
 ASLA
 ASLA
 ASLA
 TFR A,B
 BSR INHEX
 BVS NOTHEX
 PSHS B
 ADDA ,S+
 RTS
*
** READ A SINGLE HEX DIGIT
*
INHEX LBSR ECHON
 CMPA #$30
 BCS NOTHEX
 CMPA #$39
 BHI INHEXA
 SUBA #$30
 RTS
*
INHEXA CMPA #$41
 BCS NOTHEX
 CMPA #$46
 BHI NOTHEX
 SUBA #$37
 RTS
*
NOTHEX ORCC #$02
 RTS
*
OUT2S BSR OUT1S
OUT1S LDA #$20
OUT1C LBRA OUTCH
*
** ALTER X INDEX REGISTER
*
ALTRX LBSR PRTIX
 BSR OUT1S
 BSR IN1ADR
 BVS ALTXD
 STX $04,U
ALTXD RTS
*
** ALTER A ACCUMULATOR
*
ALTRA LBSR PRTA
 BSR OUT1S
 BSR BYTE
 BVS ALTAD
 STA $01,U
```

```
ALTAD RTS
*
** ALTER DIRECT PAGE REGISTER
*
ALTRDP LBSR PRTDP
 BSR OUT1S
 BSR BYTE
 BVS ALTDPD
 STA $03,U
ALTDPD RTS
*
** ALTER PROGRAM COUNTER
*
ALTRPC BSR PRTPC
ALTPC1 BSR OUT1S
 BSR IN1ADR
 BVS ALTPCD
 STX $0A,U
ALTPCD RTS
*
** ALTER U STACK POINTER
*
ALTRU BSR PRTUP
 BSR OUT1S
 BSR IN1ADR
 BVS ALTUD
 STX $08,U
ALTUD RTS
*
** ALTER Y INDEX REGISTER
*
ALTRY BSR PRTIY
 BSR OUT1S
 LBSR IN1ADR
 BVS ALTYD
 STX $06,U
ALTYD RTS
*
** ALTER B ACCUMULATOR
*
ALTRB BSR PRTB
 BSR OUT1S
 LBSR BYTE
 BVS ALTBD
 STA $02,U
ALTBD RTS
*
** ALTER CONDITION CODES
*
ALTRCC BSR PRTCC
 BSR OUT1S
 LBSR BYTE
 BVS ALTCCD
 ORA #$80
 STA ,U
ALTCCD RTS
*
```

```
** DUMP REGISTERS
*
REGSTR LDX #SEPMSG
 LBSR PSTRNG
 BSR PRTSP
 BSR PRTIX
 BSR PRTA
 BSR PRTDP
 BSR PRTPC
 LDX #SEPMSG
 LBSR PSTRNG
 BSR PRTUP
 BSR PRTIY
 BSR PRTB
 BRA PRTCC
*
** DISPLAY CONTENTS OF STACK
*
PRTSP LDX #SPMSG
 LBSR PDATA
 TFR U,X
 BRA OUT4H
*
PRTIX LDX #IXMSG
 LBSR PDATA
 LDX $04,U
 BRA OUT4H
*
PRTA LDX #AMSG
 BSR PDATA
 LDA $01,U
 BRA OUT2H
*
PRTDP LDX #DPMSG
 BSR PDATA
 LDA $03,U
 BRA OUT2H
*
PRTPC LDX #PCMSG
 BSR PDATA
 LDX $0A,U
 BRA OUT4H
*
PRTUP LDX #UPMSG
 BSR PDATA
 LDX $08,U
 BRA OUT4H
*
PRTIY LDX #IYMSG
 BSR PDATA
 LDX $06,U
 BRA OUT4H
*
PRTB LDX #BMSG
 BSR PDATA
 LDA $02,U
 BRA OUT2H
```

```
*
PRTCC LDX #CCMSG
 BSR PDATA
 LDA ,U
 LDX #REGMSG
*
** BINARY TO ASCII CONVERSION
*
BIASCI PSHS A
 LDB #$08
OUTBA LDA ,X+
 ASL ,S
 BCS PRTBA
 LDA #$2D
PRTBA BSR OUTCH
 LBSR OUT1S
 DECB
 BNE OUTBA
 PULS A,PC
*
** OUTPUT A 4DIGIT HEX NUMBER
*
OUT4H PSHS X
 PULS A
 BSR OUT2H
 PULS A
*
** OUTPUT 2DIGIT HEX NUMBER
*
OUT2H PSHS A
 LSRA
 LSRA
 LSRA
 LSRA
 BSR XASCII
 PULS A
 ANDA #$0F
XASCII ADDA #$30
 CMPA #$39
 BLE OUTC
 ADDA #$07
OUTC BRA OUTCH
*
** PRINT CR, LF, STRING
*
PSTRNG BSR PCRLF
 BRA PDATA
PCRLF PSHS X
 LDX #CRLFST
 BSR PDATA
 PULS X,PC
*
PRINT BSR OUTCH
PDATA LDA ,X+
 CMPA #$04
 BNE PRINT
 RTS
```

```
*
** INPUT CHARACTER WITH NO ECHO
*
INCH PSHS X
 LDX CPORT
GETSTA LDA ,X
 BITA #$01
 BEQ GETSTA
 LDA $01,X
 PULS X,PC
*
** INPUT CHARACTER AND ECHO
*
INCHEK PSHS A
 LDA [CPORT]
 BITA #$01
 PULS A,PC
*
** INPUT CHARACTER AND ECHO IF FLAG SET
*
ECHON TST ECHO
 BEQ INCH
INCHE BSR INCH
 ANDA #$7F
*
** OUTPUT CHARACTER IN ACCA
** FIRST DETERMINE WHICH OUTPUT ROUTINE(S) TO USE
*
OUTCH  PSHS   B
        LDB    OUTFLG
 IF PRTOPT
        BITB   #PRTBIT
        BEQ OUTCH2
        BSR POUTCH
 ENDIF PRTOPT
OUTCH2  BITB   #SERBIT
        BEQ    OUTCH3
        BSR   AOUTCH
OUTCH3 EQU *
 IF VIDOPT
        BITB   #VIDBIT
        BEQ    OUTCH4
        BSR   VOUTCH
 ENDIF VIDOPT
OUTCH4  PULS   B,PC
*
** INITIATE ACIA
*
ACINIZ LDX CPORT
 LDA #$03
 STA ,X
 LDA #$11
 STA ,X
 TST $01,X
 LDA #$FF
 STA ECHO
 RTS
```

```
*
** OUTPUT CHARACTER TO CONTROL PORT
*
AOUTCH PSHS B,X
 LDX CPORT
FETSTA LDB ,X
 BITB #$02
 BEQ FETSTA
 STA $01,X
 PULS B,X,PC
 IF PRTOPT
*
** PRINTER DRIVER ROUTINES
*
*
** PINIZ - INITIATE PRINTER PORT
*
PINIZ PSHS B
 LDD #DIRMSK*256+$04 ACCA=DIRMSK ACCB=$04
 STD PADATA SET DDR AND SELECT DATA
*
** RESET PRINTER
 LDB #PRESET
 STAB PADATA
RESTLP INCB DELAY FOR RESET
 BNE RESTLP
 STAA PADATA ACCA=DIRMSK
*
** INITALIZE PORT B (DATA PORT)
 LDAA #$2A
 STAA PBCTRL
 LDD #$FF2E ACCA=$FF ACCB =%00101110
 STD PBDATA PBDREG   PBCTRL
*
** SELECT 66 LINES/PAGE
 LDAA #$1B
 BSR POUTCH
 LDAA #'C
 BSR POUTCH
 LDAA #66
 PULS B
*
** OUTPUT A CHARACTER TO THE PRINTER
*
POUTCH PSHS B
 LDAB PBDATA CLEAR INTERRUPT BIT
*
** WAIT TILL NOT BUSY
BUSYLP LDAB PADATA
 BITB #PERROR
 BEQ PEXIT
 TSTB
 BMI BUSYLP
*
** NOW OUTPUT CHARACTER
 STAA PBDATA
PEXIT PULS B,PC
```

```
*
** PCHK TEST IF PRINTER READY
*
PCHK TST PBCTRL TEST STATE OF CRB7
 RTS SET ON ACKNOWLEDGE
 ENDIF PRTOPT
 IF VIDOPT
*****************************************************
*      TELEVIDEO-TYPE MEMORY-MAPPED EMULATOR        *
*                                                   *
* FOR HARD-WIRED MEMORY-MAPPED DISPLAYS USING THE   *
* HIGH ORDER BIT OF EACH BYTE FOR  REVERSE  VIDEO   *
* CURSORING  (SUCH  AS THE THOMAS INSTRUMENTATION   *
* 16x64 BOARD).                                     *
*****************************************************

*****************************************************
*               INITIALIZE EMULATOR                 *
*****************************************************

VINIZ   LDX    #0
        STX    COLADX    AND ROWADX
        STX    NEWROW    AND ESCFLG
        LDX    #SCREEN   POINT TO SCREEN
        STX    CURSOR    SET PROGRAM CURSOR
        LDA    #$1B      SEND ESCAPE
        BSR    VOUTCH
        LDA    #'Y       CLEAR TO END OF SCREEN
*
** VIDEO OUTPUT ROUTINE
*
VOUTCH  PSHS   A,B,X     SAVE REGISTERS
*
** CLEAR CURSOR
        LDX    CURSOR
        LDB   0,X
        ANDB   #$7F
        STB   0,X
*
** CHECK FOR ESCAPE SEQUENCE
        TST    ESCFLG    ESCAPE ACTIVE?
        BEQ    SOROU1    BRANCH IF NOT
        BSR   ESCAPE    ELSE DO ESCAPE
        BRA    RETURN    AND RETURN
*
** CHECK FOR CONTROL CHARACTERS
SOROU1  CMPA   #$20      CONTROL CODES?
        BHS    SOROU2
        BSR    CONTRL    BRANCH IF SO
        BRA    RETURN
*
** OUTPUT TEXT CHARACTER
SOROU2  LDX    CURSOR    ELSE GET CURSOR
        STAA   0,X       DISPLAY CHARACTER
        LBSR   NEWCOL    UPDATE COLUMN
*
** DISPLAY CURSOR AND RETURN
```

```
RETURN  LDX    CURSOR    AND DISPLAY IT
        LDB   X
        ORAB   #$80      WITH REVID
        STB   X
        PULS   A,B,X,PC  RESTORE REGISTERS AND RETURN

************************************************
*              CONTROL CODE HANDLERS           *
************************************************

CONTRL  CMPA   #$08      BACKSPACE ?
        BEQ    BACKSP
        CMPA   #$D       RETURN?
        LBEQ   CRETN
        CMPA   #$1B      ESCAPE SEQUENCE?
        BEQ    SETESC
        CMPA   #$0A      LINE FEED ?
        BNE    RETESC    NONE OF THESE, RETURN

************************************* LINE FEED

LINEFD  LDD    COLADX    GET CURRENT COLUMN AND ROW
        INCB             BUMP ROW
        CMPB   #NUMLIN   SCROLL TIME?
        LBNE   NEWCUR    POSITION CURSOR IF NOT
        LBRA   SCROLL    ELSE SCROLL IT

********************************** BACK SPACE

BACKSP  LDA    COLADX
        BEQ    RETESC      RETURN
        DECA
        LBRA   POSCOL    POSITION CURSOR

************************************************
*                 ESCAPE HANDLERS              *
************************************************

ESCAPE  LDAB   ESCFLG    GET FLAG
        CMPB   #'=       SETTING CURSOR?
        BEQ    ESCCUR    BRANCH IF SO
        CMPA   #'Y       CLEAR TO END OF SCREEN?
        LBEQ   ESCCLS
        CMPA   #'T       CLEAR TO END OF LINE?
        BEQ   ESCCLL
        CMPA   #'E       INSERT LINE?
        BEQ   ESCINL
        CMPA   #'R       DELETE LINE?
        BEQ   ESCDLL
        CMPA   #'=       STARTING CURSOR SET?
        BNE    CLRESC    BRANCH IF NOT

***************************** START ESCAPE SEQUENCE

SETESC  STAA   ESCFLG    ELSE START CURSORING
        RTS              AND RETURN
```

```
CLRESC  CLR    ESCFLG    NO OTHERS SUPPORTED
RETESC  RTS              SO RETURN

******************************** SET SCREEN CURSOR

ESCCUR  TST    NEWROW    ROW SET?
        BNE    ESCCU1    BRANCH IF SO
        STAA   NEWROW    ELSE SET NEW ROW
        RTS              AND RETURN

ESCCU1  CLR    ESCFLG
        SUBA   #$20      ADJUST COLUMN ADDRESS
        CMPA   #LINLEN-1 CHECK FOR ACCEPTABLE COLUM
        BHI    RETESC    NOT OK, DO NOTHING

ESCCU2  LDAB   NEWROW
        CLR    NEWROW
        SUBB   #$20      ADJUST TO ROW ADDRESS
        CMPB   #NUMLIN-1 CHECK FOR ACCEPTABLE ROW
        BHI    RETESC    ELSE RETURN DOING NOTHING
        BRA   NEWCUR    GO SET NEW CURSOR IF SO
*
************************** DELETE LINE FROM SCREEN

ESCDLL  BSR    CRETN     GO COL. ZERO
        LDB    ROWADX
        CMPB   #NUMLIN-1
        BEQ   SCROL3
        BRA   SCROL1    AND DELETE THIS LINE

************************** INSERT LINE INTO SCREEN

ESCINL  BSR    CRETN    GO TO COL. ZERO
        LDAB   ROWADX
        CMPB   #NUMLIN-1
        BEQ    ESCCLL
*
** SCROLL SCREEN DOWN FROM CURSOR
*
        LDX    #SCREEN+SCNLEN-LINLEN
ESCIN0  LDAA   0,-X
        STAA   LINLEN,X
        LDA    SCNLEN,X
        STA    SCNLEN+LINLEN,X
        CPX    CURSOR
        BNE    ESCIN0

****************** CLEAR FROM CURSOR TO END OF LINE

ESCCLL  LDA    COLADX    GET CURRENT COLUMN
        LDX    CURSOR    GET CURSOR
        LDB    #$20      AND CLEAR CHAR
ESCLL1  STB    SCNLEN,X  CLEAR ATTRIBUTE
        STB    ,X+       CLEAR TEXT
        INCA
        CMPA   #LINLEN   UNTIL END OF LINE
        BNE    ESCLL1
```

```
        CLR    ESCFLG
        RTS

********************************** CARRIAGE RETURN

CRETN   CLRA               SET COLUMN ZERO
POSCOL  LDB    ROWADX      GET CURRENT ROW

*********** GENERATE NEW CURSOR POSITION AND RETURN

NEWCUR  STD    COLADX      SAVE NEW ROW AND COLUMN
        LDA    #LINLEN     ELSE ADD A LINE
        MUL                LINLEN * ROWADX
        ADDB   COLADX
        ADCA   #0
        ADDD   #SCREEN     ADD SCREEN BASE.
        STD    CURSOR      SAVE NEW CURSOR
        TFR    D,X         GET CURSOR IN X
        RTS                AND RETURN

********************* UPDATE CURRENT COLUMN AND ROW

NEWCOL  LDD    COLADX      GET ROW AND COLUMN
        INCA               BUMP COLUMN
        CMPA   #LINLEN     ROLL?
        BNE    NEWCUR      BRANCH IF NOT
        CLRA               ELSE RESET TO ZERO
        INCB               AND BUMP ROW
        CMPB   #NUMLIN
        BNE    NEWCUR
        DECB               BOTTOM ROW
        BSR    NEWCUR

******************************** SCROLL THE SCREEN

SCROLL  LDX    #SCREEN     POINT TO SCREEN
SCROL1  LDA    SCNLEN+LINLEN,X
        STA    SCNLEN,X
        LDAA   LINLEN,X    MOVE TWO BYTES
        STAA   0,X+        UP ONE LINE
        CMPX   #SCREEN+SCNLEN-LINLEN
        BNE    SCROL1      LOOP UNTIL DONE
        BRA SCROL3

*************** CLEAR FROM CURSOR TO END OF SCREEN

ESCCLS  LDX    CURSOR      GET CURSOR
SCROL3  LDAA   #$20        GET A SPACE
SCROL2  STA    SCNLEN,X    CLEAR ATTRIBUTES
        STA    ,X+         AND TEXT
        CMPX   #SCREEN+SCNLEN
        BNE    SCROL2      UNTIL DONE
        CLR    ESCFLG
     RTS
 ENDIF VIDOPT
 IF FDCOPT
*
```

```
** "D" MINI DISK BOOT
*
MINBOOT TST CMDREG
 CLR DRVREG
 LDX #$0000
LOOP LEAX $01,X
 CMPX #$0000
 BNE LOOP
 LDA #$0F
 STA CMDREG
 BSR DELAY
LOOP1 LDB CMDREG
 BITB #$01
 BNE LOOP1
 LDA #$01
 STA SECREG
 BSR DELAY
 LDA #$8C
 STA CMDREG
 BSR DELAY
 LDX #$C000
 BRA LOOP3
LOOP2 BITB #$02
 BEQ LOOP3
 LDA DATREG
 STA ,X+
LOOP3 LDB CMDREG
 BITB #$01
 BNE LOOP2
 BITB #$2C
 BEQ LOOP4
 RTS
*
LOOP4 LDX #$C000
 STX $0A,U
 TFR U,S
 RTI
*
DELAY LDB #$04
LOOP5 DECB
 BNE LOOP5
 RTS
 ENDIF FDCOPT
*
** COMMAND JUMP TABLE
*
JMPTAB FCB $01
 FDB ALTRA
 FCB $02
 FDB ALTRB
 FCB $03
 FDB ALTRCC
 FCB $04
 FDB ALTRDP
 FCB $10
 FDB ALTRPC
 FCB $15
```

```
 FDB ALTRU
 FCB $18
 FDB ALTRX
 FCB $19
 FDB ALTRY
*
 FCC "B"
 FDB BRKPNT
 IF FDCOPT
 FCC "D"
 FDB MINBOOT
 ENDIF FDCOPT
 FCC "E"
 FDB MEMDUMP
 FCC "G"
 FDB GO
 FCC "L"
 FDB LOAD
 FCC "M"
 FDB MEMCHG
 FCC "O"
 FDB SETOUT
 FCC "P"
 FDB PUNCH
 FCC "R"
 FDB REGSTR
 FCC "S"
 FDB DISSTK
 IF CLKOPT
 FCC "T"
 FDB TIMSET
 ENDIF CLKOPT
 FCC "X"
 FDB XBKPNT
 FCC "Z"
 FDB BLKMOV
*
RAMVEC FDB RTI
 FDB RTI
 FDB RTI
 FDB RTI
 FDB SWIE
 FDB $FFFF
 FDB $FFFF
*
** CHARATER STRINGS
*
TTLMSG FCB $00
 FCB $00
 FCB $00
 FCB $0D
 FCB $0A
 FCB $00
 FCB $00
 FCB $00
 FCC "K-BUG9 V1.0 "
```

```
 FCB $04
*
CRLFST FCB $0D
 FCB $0A
 FCB $00
 FCB $00
 FCB $00
 FCB $04
*
PMTMSG FCC ">"
 FCB $04
QRYMSG FCC "WHAT?"
 FCB $04
SEPMSG FCC " - "
 FCB $04
*
SPMSG FCC "  SP="
 FCB $04
PCMSG FCC "  PC="
 FCB $04
UPMSG FCC "  UP="
 FCB $04
IYMSG FCC "  IY="
 FCB $04
IXMSG FCC "  IX="
 FCB $04
DPMSG FCC "  DP="
 FCB $04
AMSG FCC "  A="
 FCB $04
BMSG FCC "  B="
 FCB $04
CCMSG FCC "  CC: "
 FCB $04
*
REGMSG FCC "EFHINZVC"
S1MSG FCC "S1"
 FCB $04
 IF CLKOPT
*
* CLOCK INTER FACE UTILITY
*
* TIME <Hours> <Minuits> <Seconds>
* If no argument is specified, the current time
* will be displayed.
*
* READ A REGISTER FROM THE COUNTER.
* The X Index rgister points to the register
* to be read. The Status Register is checked
* before and after the register is read before
* returning a value in accumulator A
*
RDCLK  TST CLKSTA
       BNE RDCLK
RDCLK1 LDA 0,X
       TST CLKSTA
       BNE RDCLK1
```

```
       RTS
*
* MAIN PROGRAM:
*
TIMSET LDX #COUNTR POINT TO TIMER
      LBSR BYTE READ HOURS
      BVS  SHOWTM NO ARG, DISP TIME
      STA HOUR,X
      LBSR OUT1S
      LBSR BYTE READ MINUITES
      BVS  SHOWTM
      STA MINUIT,X
      LBSR OUT1S
      LBSR BYTE SECONDS.
      BVS SHOWTM
      STA SECOND,X
*
* DISPLAY CURRENT TIME
*
SHOWTM LBSR PCRLF
       LDX #COUNTR+HOUR
       LDB #3
SHOWLP BSR RDCLK
       LBSR OUT2H
       LDA #':
       LBSR OUTCH
       LEAX -1,X
       DECB
       BNE SHOWLP
       RTS
*
* INITIATE CLOCK.
* MASK INTERRUPTS.
*
CLKINZ CLR CINTCR  MASK ALL INTERRUPTS
       TST CINTSR  CLEAR ANY INTERRUPTS
       RTS
 ENDIF CLKOPT
 ORG ROM+$07B2
*
** INTERRUPT JUMP TABLE
*
V1 JMP [STACK]
V2 JMP [SWI2]
V3 JMP [FIRQ]
V4 JMP [IRQ]
V5 JMP [SWI]
*
** SPECIAL CALL TO MONITOR
*
SWI3E TFR S,U
 LDX $0A,U
 LDB ,X+
 STX $0A,U
 CLRA
 ASLB
 ROLA
```

```
 LDX SVCVO
 CMPX #$FFFF
 BEQ SWI3Z
 LEAX D,X
 CMPX SVCVL
 BHI SWI3Z
 PSHS X
 LDD ,U
 LDX $04,U
 JMP [,S++]
SWI3Z PULU X,DP,B,A,CC
 LDU $02,U
 JMP [SWI3]
*
** 6809 INTERRUPT VECTORS
*
 FDB V1
 FDB SWI3E
 FDB V2
 FDB V3
 FDB V4
 FDB V5
 FDB START
 FDB START
 END START
```