U. S. GEOLOGICAL SURVEY
OFFICE OF MARINE GEOLOGY

Woods Hole, Massachusetts
02543

UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

U.S. Geological Survey, Woods Hole, Mass.

Software listings and major electronic circuits and components of the Ocean Bottom Instrument Package (OBIP)

This program has been supported in part by the
U.S. Army Mobility Equipment Research and Development Command
Ft. Belvoir, Maryland

Open-File Report 84-267

1984

Table of Contents

Appendices, pocketed as follows:

Appendix 1 Schematic circuit diagram of the OBIP analog-to-digital circuit

Appendix 2 Schematic circuit diagram of the OBIP analog circuit

Appendix 3 Schematic circuit diagram of the OBIP control (cpu) circuit

Appendix 4 Schematic circuit diagram of the OBIP power interface circuit

Appendix 5 Schematic circuit diagram of the OBIP release circuit

Appendix 6 Circuit board silkscreen layout for the OBIP analog-to-digital circuit

Appendix 7 Circuit board component layout for the OBIP analog-to-digital circuit

Appendix 8 Circuit board wiring layout for the OBIP analog-to-digital circuit

Appendix 9 Circuit board silkscreen layout for the OBIP analog circuit

Appendix 10 Circuit board component layout for the OBIP analog circuit

Appendix 11 Circuit board wiring layout for the OBIP analog circuit

Appendix 12 Circuit board silkscreen layout for the OBIP control (cpu) circuit

Appendix 13 Circuit board component layout for the OBIP control (cpu) circuit

Appendix 14 Circuit board wiring layout for the OBIP control (cpu) circuit

Appendix 15 Circuit board silkscreen layout for the OBIP power interface circuit

Appendix 16 Circuit board component layout for the OBIP power interface circuit

Appendix 17 Circuit board wiring layout for the OBIP power interface circuit

Appendix 18 Circuit board component layout for the OBIP S-100 circuit

Appendix 19 Circuit board solder-side layout for the OBIP S-100 circuit

INTRODUCTION

The Ocean Bottom Instrument Package (OBIP) is a recoverable ocean-bottom data-capture device for seismic and other data. Its function and operation have been previously described in Ambuter and Davis (1981) and in listed references. Except for part of the instrument's release electronics and the magnetic tape cartridge controller, the electronics for the device were designed and implemented by the U.S. Geological Survey. Development of the OBIP was supported in part by the U.S. Army Mobility Equipment Research and Development Command, Ft. Belvoir, Maryland.

This report presents the details of the U.S. Geological Survey OBIP electronics, including schematics of the circuits, the circuit-board art work, software listings, parts lists, and all computer bus connections.

There are five major OBIP circuits presented in this report: (1) the analog-to-digital, (2) the analog, (3) the control (or cpu), and (4) the power-interface circuit. Each of these interfaces with (5) an S-100 bus circuit. The instrument release and magnetic-tape cartridge-controller circuits are also shown.

The instrument-release circuitry is partly proprietary to the Benthos Co., North Falmouth, MA 02556. Those aspects of the release circuitry that were developed by the U.S. Geological Survey are presented in this report. The magnetic-tape cartridge-controller circuitry is wholly proprietary to Alloy Engineering Co., 12 Mercer Road, Natick, MA 01760.

Details of other electronic and mechanical aspects of the OBIP not presented herein, i.e., mechanical drawings and specifications, testing procedures, additional device-component hookup wiring, and computer software operation, are on file at the U.S. Geological Survey offices in Woods Hole, Mass.

The reader should be cautioned that the circuit designs are prototypes. Various minor changes have been made to many of the circuits as adaptations to deployment conditions and data constraints, and these changes are not necessarily reflected in the presented figures.

DATA SUMMATION

Software

The software listings that begin on page 4 detail the software package currently in use (January 1984). It is anticipated that this software will be modified to accommodate new seismic-data acquisition modes and the acquisition of electromagnetic field data at very low sampling rates.

Parts lists and IC cross-reference

Figures 1-5 include the parts lists for the analog-to-digital, analog, control (cpu), power-interface, and release circuits, respectively. Figure 6 cross-references the integrated circuit (IC) identification numbers used by various manufacturers of the IC's in the analog-to-digital, analog, control (cpu), and power-interface circuits.

Computer bus connections

Figures 7-10 illustrate the S-100 hookups for the analog to digital, analog, control (cpu), and power interface, respectively. The reader should note that the hookups do not necessarily conform to standard S-100 protocol.

Schematics

Appendices 1 through 5 are the schematic drawings for the electronic components in the following circuits, respectively: analog to digital, analog, control (cpu), power interface, and release.

Circuit-board art work

Appendices 6 through 19 show the art work used to construct circuit boards for the following circuits: analog to digital (figs. 6-8), analog (figs. 9-11), control (cpu) (figs. 12-14), power interface (figs. 15-17), and S-100 (figs. 18-19).

REFERENCES

Ambuter, B.P., 1979, Low cost low power binary gain range amplifier operates over wide dynamic range: Electronics Design, February 1, 1979, p. 90-91.

Ambuter, B.P., and Davis, Ray, 1977, An ocean bottom instrument package (abs): Society of Exploration Geophysicists, Annual Meeting, 47th, Abstracts and Biographies, p. 58.

Ambuter, B.P., and Davis, R.E., 1981, Geophysical-data acquisition in remote environments: U.S. Geological Survey Open-File Report 81-1083, 14 p.

Ambuter, BP., and Godley, J.H., 1982, A microprocessor-based low power data acquisition system (abs): Abstracts of Proceedings, Oceans '82.

Ambuter, B.P., and Godley, J.H., 1983, A microprocessor-based low power data acquisition system: Seismological Society of America, Workshop on Remote Sensing Devices, Proceedings.

Davis, R.E., and Ambuter, B.P., 1979, An oceanographic data recorder (abs): Society of Exploration Geophysicists, Annual Meeting, 49th, New Orleans, 1979, Abstracts and Biographies, p. 37.

Davis, R.E., and Ambuter, B.P., 1981, An oceanographic data recorder: Geophysics, v. 46, no. 2, p. 203-207.

```
.Z80
NAME ('PROGFM')
TITLE   ARMY CONTROL BOARD (REV C)      01/11/83 JHG
SUBTTL  FM GAIN
INCLUDE ARMYREVC.EDU
        PAGE
PROGFM::
        LD      HL,FMCH1
        LD      B,4
        XOR     A
FTLP:   OR      (HL)
        INC     HL
        DJNZ    FTLP
        OR      A
        RET     Z
;
        LD      HL,FMACK        ; acknowledge here
        CALL    SNDMES##        ;
;
; ---- SET CONTROL AND POLARITY LINES LOW ----
;
        LD      A,8             ; init index
CLEAR:  DEC     A               ; for 7 - 0
        OUT     (FMPORT),A      ; set low
        JR      NZ,CLEAR        ; and loop
;
; ---- DELAY FOR APPROX 220 mS TO INTEGRATE
;
        LD      BC,10000        ; approx count
CLRDLY: DEC     BC              ;
        LD      A,B             ;
        OR      C               ;
        JP      NZ,CLRDLY       ; delay loop
;
; ---- MONITOR CNTS LINES FOR APPROX 2.5 SECS ----
;
        LD      BC,25000        ; approx count
ADJ:    IN      A,(FMPORT)      ; get byte
        LD      D,A             ; save
        LD      E,0             ; init index
NXTBIT: SRL     D               ; rot into carry
        JR      NC,OKAY         ; is low, skip next
        LD      A,E             ; get index
        ADD     A,84H           ; add offset
        OUT     (FMPORT),A      ; reverse polarity
;
OKAY:   INC     E               ; bump index
        LD      A,E             ; to accum
        CP      4               ; test for done
        JR      NZ,NXTBIT       ; not 4, next bit/chan
;
        DEC     BC              ; timeout ctr
        LD      A,B             ; test for zero
        OR      C               ;
        JR      NZ,ADJ          ; loop till
```

```
; ---- VERIFY CNTS LINES SET ----
;
        LD      HL,VERF         ; send diag mess
        CALL    SNDMES##        ;
;
        LD      BC,40000        ; approx 2 secs
VFMLP:  IN      A,(FMPORT)      ; get byte
        AND     0FH             ; isolate lo nibble
        JR      NZ,ERROUT       ; exit if not zero
        DEC     BC              ; else bump counter
        LD      A,B             ;
        OR      C               ;
        JR      NZ,VFMLP        ; and loop
;
ERROUT: ADD     A,30H           ; send code, '0' = ok
        RST     PUTBYT          ;
        CALL    CRLF            ;
;
; ---- PROGRAM SENSORS ----
;
        LD      HL,FMCH1        ; point to array
        LD      E,0             ; init index
;
CHLP:   LD      BC,30000        ; approx 2.5 secs
        LD      A,80H           ; get offset
        ADD     A,E             ; add index
        OUT     (FMPORT),A      ; reset sensor
;
TIMLP:  IN      A,(FMPORT)      ; get byte
        CALL    GETBIT          ; isolate bit
        JR      NC,NOTHI        ; skip next if still low
;
        CALL    PROG            ; begin programming
        LD      A,31H           ; set ascii '1'
        ADD     A,E             ; add index
        RST     PUTBYT          ; ack programmed
        JR      NXTCH           ; skip to next chan
;
NOTHI:  DEC     BC              ; bump timout ctr
        LD      A,B             ;
        OR      C               ;
        JR      NZ,TIMLP        ; loop till zero
;
; ---- TIMED OUT ----
;
        PUSH    HL              ; save pointer
        LD      HL,TIMOUT       ; print error mes
        CALL    SNDMES##        ;
        LD      A,31H           ; and channel #
        ADD     A,E             ;
        RST     PUTBYT          ;
        CALL    CRLF##          ;
;
NXTCH:  INC     E               ; bump chan index
```

```
        INC     HL              ; and array pointer
        LD      A,E             ; get index
        CP      4               ; test done
        JR      NZ,CHLP         ; loop till
        LD      HL,DONMES       ; ack done
        CALL    SNDMES##        ;
        RET                     ; and exit
;
; ---- PROGRAM GAIN ----
;
PROG:   LD      D,0             ; init LAST
WHILE:  LD      A,(HL)          ; get gain count
        OR      A               ; = zero ??
        RET     Z               ; yes, done
;
        LD      A,D             ; look at last
        OR      A               ; was it high
        JR      NZ,TEST0        ; yes, test for low
;
        IN      A,(FMPORT)      ; no, test for high
        CALL    GETBIT          ;
        JR      NC,WHILE        ; not high, loop
;
        DEC     D               ; went high, set LAST high
        DEC     (HL)            ; adj gain count
        JR      NZ,WHILE        ; loop if not zero
        LD      A,E             ; get chan index
        OUT     (FMPORT),A      ; inhibit ranging
        JR      WHILE           ; and loop
;
TEST0:  IN      A,(FMPORT)      ; get byte
        CALL    GETBIT          ; isolate bit
        JR      C,WHILE         ; still high, loop
        INC     D               ; else reset LAST
        JR      WHILE           ; and loop
;
; ---- ISOLATE BIT IN A ----
;
GETBIT: PUSH    DE              ; save reg
        INC     E               ; adj index
NXTB:   RRA                     ; shift to carry
        DEC     E               ; bump index
        JR      NZ,NXTB         ; loop for count
        POP     DE              ; restor reg
        RET                     ; ret with carry code
;
FMACK:  DB      CR,LF,'INITIALIZING SENSORS',CR,LF,0
VERF:   DB      'VERIFYING, CODE = ',0
TIMOUT: DB      'TIME OUT, CH ',0
DONMES: DB      'PROGRAMMING COMPLETE',CR,LF,0
;
        END
```

```
.Z80
NAME ('AQUIRE')
TITLE   ARMY CONTROL BOARD (REV D)      5/28/83 DHH
SUBTTL  AQUISITION
INCLUDE ARMYREVC.EQU
;*****************************************************************************
;THIS ROUTINE CONTROLS DATA ACQUISITION.ACTUAL ACQUISITION IS DONE BY THE
;INTERRUPT ROUTINE IN PAGE1.THIS DESIGN IS POOR, SINCE PROCESSOR UTILIZATION
;IS VERY LOW AND THE DE REGISTER(AS WELL AS HL) IS NOT FREE.
;*****************************************************************************
AQUIRE::
        LD      HL,AQUMES       ;POINT TO ACQUIRE MESSAGE
        CALL    SNDMES##        ;SEND IT
        XOR     A               ;CLEAR ACC AND FLAGS
        OUT     (AVGPRT),A      ;ENABLE LONG-TERM AVG.
;*****************************************************************************
;HERE THE REAL-TIME CLOCK IS READ.NOTE DELAY FOR WHOLE SECOND.
;
;*****************************************************************************
        LD      A,(TMEM+2)      ;CHECK CLOCK
RTRY:   LD      A,(TMEM+2)      ;AND AGAIN
        AND     0FH             ;MASK OUT GARBAGE
        CP      0FH             ;IF 0FH,IT CHANGED
        JR      Z,RTRY          ;SO TRY AGAIN
        INC     A               ;ADD 1 TO TIME
        DAA                     ;MAKE IT DECIMAL,NOT BINARY
        AND     0FH             ;MASK RESULT
        LD      E,A             ;SAVE IN E
WSEC:   LD      A,(TMEM+2)      ;NOW WAIT FOR CLOCK(1 SECOND)
        AND     0FH             ;MASK CLOCK
        CP      E               ;COMPARE TO LAST CLOCK+1
        JR      NZ,WSEC         ;NOT THERE YET IF NOT EQUAL
;END OF CLOCK READ
;*****************************************************************************
;                               ;IF HERE WE HAVE JUST CHANGED
;*****************************************************************************
;THIS WAS ADDED
;TO PREVENT CONFUSING DATA FROM TWO EVENTS
;
        LD      HL,1000H        ;NO. OF SAMPLES(8 BYTES EACH) IN BUFFER
        LD      DE,(PEVCNT)     ;# OF POST EVENT SAMPLES IN DE
        XOR     A               ;CLEAR CARRY
        SBC     HL,DE           ;#OF VALID PRE-EVENT SAMPLES NEEDED
        EX      DE,HL           ;PUT IN DE
;*****************************************************************************
;INITIALIZE BUFFER AND INTERRUPTS
;
        LD      HL,BUFMEM       ;SET UP HL TO 8000H(BUFFER START)
        LD      A,9             ;MASK TO ENABLE EARLY TERMINATOR & ACQISITION
        OUT     (INTPRT),A      ;PUT IT IN NSC 800
        LD      (INTMSK),A      ;SAVE A COPY?!
        LD      A,1H            ;PREPARE TO REMOVE RSTA INHIBIT
        LD      (PCCLRB),A      ;CLEAR PORT C PIN 0
        EI                      ;ENABLE INTERRUPTS
;
```

```
;*****************************************************************************
;INTERRUPT ROUTINE (SEE PAGE1.MAC) DECREMENTS DE PAIR EVERY TIME A SAMPLE
;IS TAKEN.A SAMPLE COMPRISES 2 BYTES EACH FROM 4 CHANNELS(FOR NOW FIXED).
;ABOVE CODE MUST BE CHANGED WHEN SAMPLE SIZE CAN VARY.
;*****************************************************************************
; CHECK H TO SEE IF BUFFER IS FULL ENOUGH
; THIS WAS ADDED
WABUFF: LD      A,D             ;MOVE D TO A
        OR      E               ;D&E MUST BE 0
        JR      NZ,WABUFF       ;WAIT TILL ENOUGH SAMPLES
;
;*****************************************************************************
;FIRST WAIT FOR AN EVENT
;
AQULP:  IN      A,(ANAPRT)      ;LOOK FOR EVENT
        RLA                     ;ROTATE BIT 7 TO CARRY
        JP      NC,AQULP        ;NO EVENT
;
;*****************************************************************************
;NOW QUALIFY EVENT
;
        LD      DE,20           ;EVENT LINE MUST BE HI FOR 20 SAMPLES
VRFLP:  IN      A,(ANAPRT)      ;LOOK FOR EVENT
        RLA                     ;ROTATE BIT 7 TO CARRY
        JP      NC,AQULP        ;NO EVENT
;THIS WAS ADDED
        LD      A,D             ;PUT D IN A
        OR      E               ;CHECK THAT BOTH D&E ARE 0
        JP      NZ,VRFLP        ;HAVE DONE 20 SAMPLES
;*****************************************************************************
;                               ;IT IS AN EVENT!
;*****************************************************************************
;THIS WAS ADDED
        LD      A,1             ;SO DISABLE AVERAGE
        OUT     (AVGPRT),A
;*****************************************************************************
;NOW TAKE POST EVENT SAMPLES
;
        LD      DE,(PEVCNT)     ;LOAD DE WITH # OF POST EVENT SAMPLES
PEVLP:  LD      A,D             ;D TO A(AS ABOVE)
        OR      E               ;BOTH ZERO?(REMEMBER,INT ROUTINE DECR. DE)
        JP      NZ,PEVLP        ;NO,TAKE MORE SAMPLES
;
;*****************************************************************************
;END OF OHH DOCUMENTATION,BEGIN JG
;*****************************************************************************
; ---- GET TIME OF EVENT -----
;
GETTIM: DI
        OUT     (INTPRT),A
        LD      (INTMSK),A
        EI                      ; Enable termination interrupt
        LD      (RECPTR),HL     ; Save end of record pointer
;
RETRY:  LD      B,11            ; Initialize counter for time codes
```

```
        LD      DE,TMEM+0CH     ; Point to too of RTC registers
        LD      HL,TIMBUF       ; Point to buffer area for time data
        LD      A,(DE)          ; Dummy read to clear missed time flags
TMLP:   LD      A,B             ; Give iteration count to accum
        CP      9               ; Ninth iteration ???
        JR      NZ,NTDOW        ; No, skip next
        DEC     B               ; Adjust iteration counter for skip
        DEC     DE              ; Push pointer past day of week code
NTDOW:  LD      A,(DE)          ; Get time nibble
        AND     0FH             ; Mask undefined high order
        CP      0FH             ; Time change during loop ???
        JR      Z,RETRY         ; Yes, Redo whole procedure
        LD      (HL),3          ; Preset buffer location for next instr
        RLD                     ; Converts BCD nibble in A to ascii in (HL)
        INC     HL              ; Ready pointer for next
        DEC     DE              ; and time pointer
        DJNZ    TMLP            ; Loop B times
        LD      (HL),0          ; Insert string terminator
        LD      HL,TIMBUF       ;
        CALL    SNDMES##
        IN      A,(ANAPRT)      ;
        LD      (GCODE),A       ;
        PAGE
;
; ---- CREATE FOUR RECORD HEADERS ----
;
        LD      B,4             ; Initialize loop counter
        LD      DE,HDRBUF       ; Point to header buffer
        LD      (FNPTR),DE      ; Pass pointer to WRTREC routine
FHDRLP: PUSH    BC              ; Save loop counter
        LD      HL,TIMBUF       ; Point to time data
        LD      BC,10           ; Initialize transfer counter
        LDIR                    ; Place in buffer
        EX      DE,HL           ; Give HL end of current buffer
        LD      DE,(RECPTR)     ; Get start of record pointer
        LD      (HL),E          ; Place lsb in header
        INC     HL              ; Bump pointer
        LD      (HL),D          ; Then place msb in header
        INC     HL              ; and bump pointer
        POP     BC              ; Restore loop counter
        LD      A,4             ; Determine current record #
        SUB     B               ; 0 to 3 corrosponds to rec's 1-4
        LD      (HL),A          ; Save record number in header
        INC     HL              ; Bump pointer
        XOR     A               ; Clear accum
        LD      C,2             ; Initialize loop counter to
FHDR0:  LD      (HL),A          ; fill reserved header bytes
        INC     HL              ; with zeroes
        DEC     C               ;
        JR      NZ,FHDR0        ; Loop till done (3 times)
        LD      A,(GCODE)       ;
        LD      (HL),A          ;
        INC     HL              ;
        EX      DE,HL           ; Give DE start of next header
        DJNZ    FHDRLP          ; Loop for four headers
```

```
        RET                     ; return to caller
AQUMES: DB      CR,LF,'AQUIRE',CR,LF,0

        END
```

```
.Z80
NAME ('ENDPRG')
TITLE   ARMY CONTROL BOARD (REV C)      11/20/82 JHG
SUBTTL  END OF PROGRAM
INCLUDE ARMYREVC.EDU
        PAGE
;
; ---- END OF PROGRAM LOOP (end of track 4) ----
;
ENDPRG::
        CALL    SLEEP##         ; Turn recorder off
        LD      A,0FFH          ; Set end of program flag
        LD      (EOPFLG),A      ;   true
        LD      A,1             ; allow only ETERM interrupts
        LD      (INTMSK),A      ;
        OUT     (INTPRT),A      ;
        LD      A,7             ; Select 32mS sample rate
        OUT     (ADPORT),A      ;
        EI                      ; Enable interrupts
ELP:    JP      ELP             ; Wait for interrupt
;
; ---- EARLY TERMINATION HANDLER ----
;
ETERM:: PUSH    AF              ;
        LD      A,BIT1          ; Disable power save
        LD      (PCCLRB),A      ;
;
        PUSH    BC              ;
        LD      BC,0            ;
TSTKEY: BIT     7,(IX+01H)      ;
        JR      Z,PRMON         ;
        DEC     BC              ;
        LD      A,B             ;
        OR      C               ;
        JR      NZ,TSTKEY       ;
        POP     BC              ;
        POP     AF              ;
;
        CALL    WAKEUP##        ; Power up cartridge and controller
        LD      A,(EOPFLG)      ; test end of program flag
        OR      A               ; to see if directory already written
        JR      NZ,NEWDIR       ; if so, adjust pointers and await tape
        CALL    EOTRK##         ; else finish off this track
NEWDIR: LD      HL,(STDIR)      ; Set up for new directory
        LD      (DYNDIR),HL     ;
        LD      (HL),1AH        ; Mark end of info. with ascii eof
        CALL    INOUT           ; Wait for new cartridge
        JP      RSTART##        ; and re-start experiment
;
PRMON:  POP     BC              ;
        POP     AF              ;
        JP      REST##          ;
;
; ---- AWAIT NEW TAPE ----
;
```

```
INOUT:: LD      A,8             ; Send current status command
        OUT     (CA),A          ; to controller
        CALL    RSTAT##         ; Get current status
        LD      A,(DS)          ; Look at drive status
        BIT     4,A             ; Cartridge still inserted ?
        JR      NZ,INOUT        ; Yes, looo til removed
$IN:    LD      A,8             ; Send current status command
        OUT     (CA),A          ; to controller
        CALL    RSTAT##         ; Get current status
        LD      A,(DS)          ; Look at drive status
        BIT     4,A             ; Still no cartridge
        JR      Z,$IN           ; Yes, loop till inserted
        RET                     ; Return to caller
        END
```

```
.Z80
NAME ('CARTLI')
TITLE   ARMY CONTROL BOARD [REV C]      11/19/82 JHG
SUBTTL  CARTRIDGE SUBROUTINES
INCLUDE ARMYREVC.EQU
Listing suppressed for equates file 'ARMYREVC.EQU'
NOTES TO REV C.:
1.      Routine RSFILE no longer sets IPA to 1, must be done by caller.
2.      WAKEUP and SLEEP assume cartridge power line is active high.
3.      New constant PWRDLY is count for BC for power on stabilization.
4.      In WRDATA routine, z test is for bit 7, not reg-pair values.
5.      Removed delay after power removal in sleep routine.
\
        PAGE
;
; ---- WAKE CARTRIDGE AND INTERFACE ----
;
WAKEUP::
        IF      DIAG
        PRINT   WAKMES
        ENDIF
        PUSH    BC              ; Save calling reg.
        LD      A,BIT2          ;
        LD      (PBCLRB),A      ;
        LD      A,83H           ;
        OUT     (PWRPRT),A      ; Apply power to cartridge
        LD      BC,PWRDLY       ; Delay for power distribution
        CALL    DELYBC          ;  100 uS / count in BC
        LD      A,80H           ;
        OUT     (PWRPRT),A
        LD      BC,PWRDLY
        CALL    DELYBC
        LD      A,BIT2          ;
        LD      (PBSETB),A      ;
        LD      BC,5000
        CALL    DELYBC
        LD      BC,REWAKE       ; Set abort vector for failure
        LD      (ABRVEC),BC     ;
        POP     BC              ; Restore BC
        CALL    GSTAT           ; Get status and return to caller
        PUSH    HL
        LD      HL,0
        LD      (ABRVEC),HL
        POP     HL
        RET
;
; ---- HERE ON ABORT DURING WAKE UP ----
;
REWAKE: IF      DIAG
        PRINT   REWMES
        ENDIF
        POP     AF              ; Clear stack from abort
        CALL    SLEEP           ; Power down again
        PUSH    BC              ; Save reg.
        LD      BC,0            ; Set up for delay
```

```
        CALL    DELYBC          ;  approx 6.5 secs
        POP     BC              ; Restore reg.
        JR      WAKEUP          ; and try again
;
; ---- POWER DOWN CARTRIDGE AND CONTROLLER ----
;
SLEEP:: IF      DIAG
        PRINT   SLPMES
        ENDIF
        PUSH    BC              ; Save reg
        LD      A,6FH           ; Select drive 4 (non-existent)
        OUT     (MA),A          ; Send to controller for de-select current
        LD      B,8             ; Delay for de-select (100uS)
SLPLP1: DJNZ    SLPLP1          ;  (don't know if this does any good)
        LD      A,3             ;
        OUT     (PWRPRT),A      ;
SLPLP2: DJNZ    SLPLP2          ;
SLPLP3: DJNZ    SLPLP3          ;
SLPLP4: DJNZ    SLPLP4          ;
        XOR     A               ;
        OUT     (PWRPRT),A      ; Remove power
        POP     BC              ; Restore reg
        RET                     ; and return to caller
        PAGE
; ---- POSITON TAPE BETWEEN LOAD POINT AND EOT ----
;
LPOINT::
        IF      DIAG
        PRINT   LPTMES
        ENDIF
        PUSH    BC              ; Save reg
        CALL    FORWRD          ; Get to load point
        CALL    FORWRD          ; Get past load point
        POP     BC              ; Restore reg
        RET                     ; and return
;
FORWRD: LD      A,80H           ; Set position arg. for 128 files
        LD      (IPA),A         ;
        CALL    SNDMPA          ; Send to controller
        CALL    GSTAT           ; Wait for port ready
        LD      A,5             ; Space forward files command
        OUT     (CA),A          ; Send to controller
        LD      BC,25000        ; Delay for forward motion
        CALL    DELYBC          ;  approx 2.5 sec's
        CALL    SLEEP           ; Power down cart and controller
        LD      BC,10000        ;
        CALL    DELYBC          ; Delay for cap's discharge
        CALL    WAKEUP          ; Wake it up again
        XOR     A               ;
        LD      (IPA),A         ; Reset position to 0 default
        RET                     ; and return
;
DELYBC::
        DEC     BC              ; Loop for count in BC
        LD      A,B             ; approx 100 uS per count
```

```
        OR      C               ; (99.1821uS @ 1,048,576 Hz)
        RET     Z               ;
        PUSH    BC              ; Save count
        LD      B,4             ; Extra delay
$DELBC: DJNZ    $DELBC          ;  to fill
        POP     BC              ; Restore count
        JP      DELYBC          ;  and loop
        PAGE
;
; ---- TEST PORT DATA AVAILABLE ----
;
ISTAT:: PUSH    AF              ;
$ISTAT: IN      A,(PS)          ; Read port status
        BIT     1,A             ; Data available ?
        JR      Z,$ISTAT        ; No, wait
        POP     AF              ;
        RET                     ; Yes, return to caller
;
; ---- TEST PORT READY FOR DATA ----
;
OSTAT:: PUSH    AF              ;
$OSTAT: IN      A,(PS)          ; Read port status
        BIT     0,A             ; Clear to send ?
        JR      Z,$OSTAT        ; No, wait
        POP     AF              ;
        RET                     ; Yes, return to caller
;
; ---- RETURN DRIVE AND INTERFACE STAUS ----
;
RSTAT:: CALL    ISTAT           ; Returns when data available
        IN      A,(CA)          ; Read drive status
        LD      (DS),A          ; and store internally.
        CALL    ISTAT           ; Returns when data available
        IN      A,(CA)          ; Read interface status
        LD      (IS),A          ; and store internally.
        IF      DIAG
        PRINT   DSISMS
        LD      A,(DS)
        CALL    TOHEX
        LD      A,(IS)
        CALL    TOHEX
        LD      A,(IS)
        RET
TOHEX:  PUSH    BC
        LD      C,A
        RRA
        RRA
        RRA
        RRA
        CALL    SHWHEX
        LD      A,C
        CALL    SHWHEX
        LD      A,' '
        RST     PUTBYT
        POP     BC
```

```
        RET
SHWHEX: AND     0FH
        ADD     A,30H
        CP      3AH
        JP      C,PUTBYT
        ADD     A,7
        JP      PUTBYT
        ENDIF
        RET                     ; Return to caller
;
; ---- SEND MODE AND POSITION ARGUMENTS
;
SNDMPA::
        PUSH    AF              ; Save reg.
        CALL    OSTAT           ; Returns when port available
        LD      A,(IMA)         ; Get internal mode argument
        OUT     (MA),A          ; Transfer to controller
        CALL    OSTAT           ; Returns when port available
        LD      A,(IPA)         ; Get internal mode argument
        OUT     (PA),A          ; Transfer to controller
        POP     AF              ; Restore reg.
        RET                     ; Return to caller
        PAGE
;
; ---- READ CARTRIDGE ----
; Z = read OK data follows, NZ = abort or syntax error
;
RDCART::
        LD      A,01H           ; Read record command
        JP      EXEC            ;
;
; ---- READ DATA FROM PORT BUFFER ----
; DE=byte count , HL-->transfer address
; on return, Z = OK, NZ = out of data
;
RDDATA::
        CALL    ISTAT           ; Wait for port ready
GBYTE:  IN      A,(CA)          ; Read data byte
        LD      (HL),A          ; Store at pointer
        INC     HL              ; Ready pointer for next
        DEC     DE              ; Decrement index
        LD      A,D             ; Test for end of transmit
        OR      E               ; Done ?
        JR      NZ,RDDATA       ; No, go back and read next
        RET                     ; Return with Z
        PAGE
;
; ---- WRITE CARTRIDE ----
; assumes MA,PA set and buffer full
; C = end of tape detected, Z = write OK, NZ = abort condition.
; *** Calling routine must test C flag first. ***
;
WRCART::
        PUSH    HL              ; Save reg
        LD      HL,(ABRVEC)     ; Get current abort vector
```

```
        PUSH    HL              ; Save it
        LD      HL,ABREDT       ; Point to write abort routine
        LD      (ABRVEC),HL     ; Place in pointer mem
        POP     HL              ; Restore old abort vector
        LD      A,02H           ; Write record command
        CALL    EXECN           ; Do it
        LD      A,(DS)          ; Test drive status for eot
        BIT     2,A             ;
        JR      Z,WRTOK         ; Normal exit if not
        SCF                     ; else set carry as marker
WRTOK:  LD      (ABRVEC),HL     ; Restore old abort vector
        POP     HL              ; and reg
        RET                     ; and return to caller
;
; ---- HERE ON ABORT DURING WRITE ----
;
ABREDT: LD      (ABRVEC),HL     ; Restore old abort vector
        POP     HL              ; Drop return address to WRCART
        POP     HL              ; Restore HL
        SCF                     ; Set carry to indicate trouble
        RET                     ; Return to caller
;
; ---- WRITE DATA TO PORT BUFFER ----
; BC=byte count, HL-->transfer address
;
WRDATA::
        CALL    OSTAT           ; Returns when port ready
        BIT     7,H             ; Aquisition buffer operation ?
        LD      A,(HL)          ; Get data byte
        OUT     (DA),A          ; Send byte to port buffer
        INC     HL              ; Next byte
        DEC     BC              ;
        JR      Z,NOTBUF        ; Skip next if not aquisition dump
        SET     7,H             ; Else fix for wrap-around
NOTBUF: LD      A,B             ; Go till all bytes transfered
        OR      C               ;
        JR      NZ,WRDATA       ;
        RET                     ; Return to caller
        PAGE
;
; ---- WRITE END OF FILE MARK ----
WREOF::
        LD      A,03H           ; Write end of file mark command
        JR      EXEC            ;
;
; ---- FORWARD SPACE RECORDS ----
FSREC::
        LD      A,04H           ; Space forward records comman
        JR      EXEC            ;
;
; ---- FORWARD SPACE FILES ----
FSFILE::
        LD      A,05H           ; Space forward files command
        JR      EXEC            ;
;
```

```
; ---- REVERSE SPACE RECORDS ----
RSREC::
        LD      A,06H           ; Space reverse records command
        JR      EXEC            ;
;
; ---- REVERSE SPACE FILES ----
RSFILE::
        LD      A,07H           ; Soace reverse files command
        JR      EXEC            ;
;
; ---- DRIVE AND INTERFACE STATUS ----
GSTAT:: LD      A,08H           ; Send current status command
        JR      EXEC            ;
;
; ---- REWIND DRIVE ----
REWIND::
        LD      A,40H           ; Rewind to bot command
        JR      EXEC            ;
;
; ---- MISC. COMMANDS ----
MISC::  LD      A,(ICA)         ; Get command and execute
        JR      EXEC            ;
        PAGE
; ---- COMMAND EXECUTOR ----
EXEC::  CALL    SNDMPA          ; Send mode and position args.
EXECN:: LD      (ICA),A         ; Save command in case of trouble
        CALL    OSTAT           ; Wait for port ready
        LD      A,(ICA)         ; Get command again
        OUT     (CA),A          ; Give to controller
        IF      DIAG
        PRINT   EXEMES
        LD      A,(ICA)
        ADD     A,30H
        RST     PUTBYT
        LD      A,' '
        RST     PUTBYT
        ENDIF
        CALL    RSTAT           ; Get current status
        AND     30H             ; Test for abort
        RET     Z               ; return if okay
;
; ---- ABORT ROUTINE ----
; NOTE: error handler must clear tos if not returning to original caller
ABORT:  IF      DIAG
        PRINT   ABRMES
        ENDIF
        PUSH    HL              ; Save reg.
        LD      HL,(ABRVEC)     ; Get abort vector address
        EX      (SP),HL         ; Place address on top of stack
        RET                     ; Jump to abort vector.
;
        IF      DIAG
WAKMES: DB      'WAKE   ',0
REWMES: DB      'REWAKE ',0
SLPMES: DB      'ZZZZ   ',0
```

```
LPTMES: DB      'LPOINT ',0
ABRMES: DB      'ABORT  ',0
EXEMES: DB      'EXEC CMD=',0
DSISMS: DB      'DS,IS=',0
        ENDIF
        END
```

```
.Z80
TITLE   ARMY CONTROL BOARD [REV C]      11/11/82 K.PRADA
SUBTTL  Z80 MONITOR ROM
INCLUDE ARMYREVC.EQU
        PAGE
;-----------------------------------------------------------------
;  NUMB:
;  DATE: 11 NOVEMBER 1982
;  AUTH: K. PRADA - UPPER CAPE SYSTEMS
;-----------------------------------------------------------------
;  REVISION HISTORY:
;
;       ADDED DRIVERS FOR USGS/ARMY REVC CONTROL BOARD 11/29/82 JHG
;       ADDED RST JUMP VECTORS FOR REVC CONTROL BOARD
;       FIXED BREAKPOINT RST (WAS 2 IN PAGE1, SHOULD BE 1) 12/3/82 JHG
;       ADDED ROUTINES FOR ROMABLE INPUT AND OUTPUT  12/6/82 JHG
;              (ASSUMES Z80 OR NSC 800 [ in r,(c) ])
;       ADDED ROUTINE FOR STORING 16 BIT WORDS 12/6/82 JHG
;-----------------------------------------------------------------
FALSE   EQU     0
TRUE    EQU     NOT FALSE
MONRAM  EQU     SYSRAM+800H     ;MUST BE SET BY END USER
RAMST   EQU     SYSRAM
RAMEND  EQU     SYSRAM+0FFFH
MONIT:: JP      CBOOT           ;COLD START AFTER RESET
        PAGE
;-----------------------------------------------------------------
; ENTRY AND COLD INITIALIZATION
;-----------------------------------------------------------------
CBOOT:  DI                      ;DISABLE INTERRUPTS
        LD      HL,RAMST
        LD      (DSADD),HL      ;SET DEFAULT DISPLAY ADDRESS
        LD      (LSADD),HL      ;AND DEFAULT LIST ADDRESS
        CALL    MEMSIZ          ;GET MEMORY PARAMETERS
        LD      SP,HL           ;SET STACK POINTER
        LD      DE,EXIT         ;MOVE EXIT AREA TO HIGH RAM
        EX      DE,HL
        LD      BC,ENDX-EXIT
MLOOP1: LD      A,(HL)
        LD      (DE),A
        INC     HL
        INC     DE
        DEC     BC
        LD      A,B
        OR      C
        JP      NZ,MLOOP1

        LD      BC,3*NBKPTS
        PUSH    DE
        POP     HL
        DEC     HL
MLOOP2: LD      A,(HL)
        LD      (DE),A
        INC     HL
        INC     DE
```

```
        DEC     BC
        LD      A,B
        OR      C
        JP      NZ,MLOOP2
        LD      HL,-24
        ADD     HL,SP
        PUSH    HL
        INC     HL              ;ADJUST USER STACK LOCATION
        INC     HL
        LD      (SPSV),HL       ;SAVE THE STACK INITIAL VALUE
        LD      D,10            ;CLEAR REGISTER STORAGE AREA
INIT2:  PUSH    BC
        DEC     D
        JP      NZ,INIT2
        LD      HL,LOGMSG       ;ISSUE LOGON MESSAGE
        CALL    PRTWD
        JP      WINIT           ;GO TO MONITOR EXECUTIVE
        PAGE
;-----------------------------------------------------------------
;  ROUTINE EXF READS ONE PARAMETER.  IT EXPECTS THE FIRST
;  CHARACTER OF THE PARAMETER TO BE IN THE A REGISTER
;  ON ENTRY.
;-----------------------------------------------------------------
EXF:    LD      B,1             ;SET UP FOR ONE PARAMETER
        LD      HL,0
        JP      EX1             ;FIRST CHARACTER IN A ALREADY
;-----------------------------------------------------------------
;  ROUTINE EXPR READS PARAMETERS FROM THE CONSOLE
;  AND DEVELOPS A 16 BIT HEXADECIMAL FOR EACH ONE.
;  THE NUMBER OF PARAMETERS WANTED IS IN THE B REG
;  ON ENTRY.  A CARRIAGE RETURN WILL TERMINATE THE
;  ENTRY SEQUENCE; A BLANK OR A COMMA WILL END THE
;  CURRENT PARAMETER ENTRY.  EACH PARAMETER ONLY
;  TAKES THE LAST 4 DIGITS TYPED IN; ANY EXCESS IS
;  DISCARDED.  A NON-HEX DIGIT WILL TERMINATE THE
;  ENTRY SEQUENCE AND CAUSE A WARM BOOT OF THE MON.
;-----------------------------------------------------------------
EX3:    JP      NZ,QPRT         ;NON-ZERO IS ERROR
EXPR1:  DEC     B               ;MORE PARAMETERS?
        RET     Z               ;NO, RETURN
EXPR:   LD      HL,0            ;INITIALIZE PARAMETER
EX0:    CALL    ECHO            ;GET NEXT NUMBER
EX1:    LD      C,A             ;SAVE CHAR FOR LATER USE
        CALL    NIBBLE
        JP      C,EX2           ;NOT A NUMBER, JUMP
        ADD     HL,HL           ;MULTIPLY BY 16
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        OR      L               ;ADD ON NEW DIGIT
        LD      L,A
        JP      EX0             ;GO GET NEXT DIGIT
EX2:    EX      (SP),HL         ;PUT UNDER RETURN ADDRESS ON STACK
        PUSH    HL              ;RESTORE RETURN ADDRESS
        LD      A,C             ;REGET THE LAST CHARACTER
```

```
        CALL    P2C             ;TEST FOR DELIMITER
        JP      NC,EX3          ;JUMP IF NOT CARRIAGE RETURN
        DEC     B
        JP      NZ,QPRT         ;CR WITH MORE PARAM MEANS ERROR
        RET
        PAGE
;---------------------------------------------------------------
; MAIN ACTION ROUTINES
;---------------------------------------------------------------
QPRT:   LD      HL,QMSG         ;GET ADDRESS OF WHAT MESSAGE
        CALL    PRTWD           ;PRINT IT AND FALL INTO WARM START
;---------------------------------------------------------------
;  THE WARM START CODE
;---------------------------------------------------------------
WINIT:  LD      HL,(SPSV)       ;RESET THE STACK
        LD      SP,HL
WINITA: LD      HL,WINIT        ;RESET RETURN AND WARM START VECTOR
        PUSH    HL
        CALL    CRLF            ;START A NEW LINE
        CALL    DECHO           ;GET THE COMMAND
        SUB     'A'             ;GET RID OF ASCII BITS
        JP      C,QPRT          ;BAD COMMAND
        CP      'Z'-'A'+1       ;CHECK UPPER LIMIT
        JP      NC,QPRT         ;BAD COMMAND
        ADD     A,A             ;DOUBLE IT FOR TABLE OFFSET
        LD      E,A             ;SET UP FOR DOUBLE ADD
        LD      D,0
        LD      B,2             ;SET UP FOR TWO PARAMETERS
        LD      HL,TBL          ;GET ACTION ROUTINE ADDRESS
        ADD     HL,DE
        LD      A,(HL)          ;LOAD H,L INDIRECT
        INC     HL
        LD      H,(HL)
        LD      L,A
        JP      (HL)            ;GO TO ACTION ROUTINE
        PAGE
;---------------------------------------------------------------
; THIS ROUTINE FILLS A BLOCK OF MEMORY WITH A USER-
; DETERMINED CONSTANT.  IT EXPECTS THREE PARAMETERS
; TO BE ENTERED IN THE FOLLOWING ORDER;
;
; START ADDRESS
; FINISH ADDRESS
; FILL VALUE
;---------------------------------------------------------------
FILL:   CALL    EXPR3           ;GET THREE PARAMETERS
FIO:    LD      (HL),C          ;PUT DOWN THE FILL VALUE
        CALL    HILO            ;INCREMENT AND CHECK THE POINTER
        JP      NC,FIO          ;NOT DONE YET, JUMP
        POP     DE              ;RESTORE STACK POINTER IN CASE
        JP      WINIT           ; STACK WAS OVERWRITTEN
;---------------------------------------------------------------
; THIS ROUTINE COMPARES TWO BLOCKS OF MEMORY AGAINST EACH
; OTHER.  IF A DIFFERENCE IN THE RELATIVE ADDRESS
; CONTENTS IS DETECTED, THE ADDRESS OF THE FIRST
```

```
; BLOCK IS DISPLAYED, ALONG WITH ITS CONTENTS AND
; THE CONTENTS OF THE OTHER BLOCK'S SAME RELATIVE
; ADDRESS.
;---------------------------------------------------------------
COMP:   CALL    EXPR3           ;GO GET THREE PARAMETERS
CMPA:   LD      A,(BC)          ;GET SOURCE 2 DATA
        PUSH    BC              ;SAVE SOURCE 2 POINTER
        LD      B,(HL)          ;READ SOURCE 1 DATA
        CP      B               ;COMPARE DATA
        JP      Z,CMPB          ;JUMP IF OK
        PUSH    AF              ;SAVE SOURCE 2 DATA
        CALL    LADRB           ;WRITE THE ADDRESS
        LD      A,B             ;GET SOURCE 1 DATA
        CALL    DASH1           ;FORMAT
        POP     AF              ;REGET SOURCE 2 DATA
        CALL    HEX1            ;OUTPUT IT
CMPB:   POP     BC
        CALL    HILOXB          ;INCREMENT SOURCE 1 POINTER AND SEE IF
        JP      CMPA            ;JUMP IF NOT DONE YET
        PAGE
;---------------------------------------------------------------
; THIS ROUTINE DISPLAYS A BLOCK OF MEMORY ON THE
; CURRENT CONSOLE DEVICE (CONSOLE DUMP).  THE USER
; MUST SPECIFY THE START AND FINISH ADDRESSES.
; THE DISPLAY IS ORGANIZED TO DISPLAY UP TO 16 BYTES
; PER DISPLAY LINE, WITH ALL COLUMNS ALIGNED SO
; EACH COLUMN HAS THE SAME LAST HEX DIGIT IN ITS ADDRESS
;---------------------------------------------------------------
DISP:   LD      B,1             ;GET START ADDRESS
        CALL    EXPR
        JP      C,DSALL         ;DISPLAY ALL
        LD      B,1             ;GET END ADDRESS
        CALL    EXPR
        JP      C,DSONE         ;DISPLAY ONE PAGE
        POP     DE
DISPX:  POP     HL
        CALL    CRLF
DIS1:   CALL    LADRB           ;DISPLAY THE START ADDRESS
        LD      A,L             ;SEE IF ON 16 BYTE BOUNDARY
        CALL    TRPLSP          ;SKIP OVER TO RIGHT COLUMN
        PUSH    HL              ;SAVE (H,L)
DIS2:   LD      A,(HL)          ;GET THE CONTENTS
        CALL    HEX1            ;OUTPUT IT
        CALL    HILO            ;INCREMENT, CHECK POINTER
        JP      C,DIS7          ;DONE IF CARRY SET
        CALL    BLK             ;MAKE COLUMNS
        LD      A,L             ;READY FOR NEW LINE?
        AND     0FH
        JP      NZ,DIS2
DIS3:   POP     HL              ;REGET LINE START ADDRESS
        LD      A,L             ;SKIP OVER TO RIGHT SPACE
        AND     0FH
        CALL    TRPL2
DIS4:   LD      A,(HL)          ;GET MEMORY VALUE
        LD      C,A             ;SET UP FOR OUTPUT
```

```
        CP      ' '             ;SEE IF PRINTABLE IN ASCII
        JP      C,DIS5          ;JUMP IF SO
        CP      7FH
        JP      C,DIS6
DIS5:   LD      C,'.'           ;ELSE, PRINT A DOT
DIS6:   CALL    TTYOUT
        CALL    HILOX           ;INCREMENT (H,L) AND SEE IF DONE
        LD      A,L             ;NOT DONE, READY FOR NEW LINE
        AND     0FH
        JP      NZ,DIS4         ;JUMP IF NOT
        JP      DIS1            ;DO THE NEXT LINE
DIS7:   SUB     E               ;SKIP OVER TO START ASCII PRINTOUT
        CALL    TRPLSP
        JP      DIS3            ;GO PRINT THE ASCII
        PAGE
TRPLSP: AND     0FH             ;ISOLATE THE LOW FOUR BITS
        LD      B,A             ;PREPARE TO SPACE OVER TO RIGHT COLUMN
        ADD     A,A             ;TRIPLE THE COUNT
        ADD     A,B
TRPL2:  LD      B,A             ;PUT BACK INTO B
        INC     B               ;ADJUST COUNTER
TRPL1:  CALL    BLK             ;DO THE SPACING
        DEC     B
        JP      NZ,TRPL1        ;NO, DO ANOTHER COLUMN
        RET
DSALL:  POP     HL              ;GET WHAT EXPR PUT ON STACK
        LD      A,H
        OR      L               ;ZERO?
        JP      NZ,DSAL1        ;YES, USE DEFAULT
        LD      HL,(DSADD)      ;DEFAULT DISPLAY ADDRESS
DSAL1:  EX      DE,HL           ;TO D-E
        LD      HL,256          ;ONE PAGE LENGTH
        ADD     HL,DE
        LD      (DSADD),HL      ;SET NEXT START ADDRESS
        DEC     HL              ;ENDING ADDRESS
        EX      DE,HL           ;SWAP INTO D-E
        CALL    CRLF
        JP      DIS1
DSONE:  POP     DE              ;GET END ADDRESS
        LD      A,D
        OR      E               ;ZERO?
        JP      NZ,DISPX        ;NO, DO START END
        POP     HL              ;OR GET START ADDRESS
        JP      DSAL1
        PAGE
;-----------------------------------------------------------------
; GOTO COMMAND TRANSFERS CONTROL TO A SPECIFIED ADDRESS.
; IT ALLOWS THE SELECTIVE SETTING OF UP TO TWO BREAKPOINTS
; AS WELL AS ALLOWING ANY CONSOLE INPUT TO BREAKPOINT
; THE RUN, AS LONG AS INTERRUPT 1 IS ACTIVE.
;-----------------------------------------------------------------
GOTO:   CALL    PCHK            ;SEE IF OLD ADDRESS WANTED
        JP      C,G03           ;  YES, JUMP
        JP      Z,G00           ;  YES, BUT SET SOME BREAKPOINTS
```

```
        CALL    EXF             ;GET NEW GOTO ADDRESS
        POP     DE
        LD      HL,PLOC         ;PUT ADDRESS IN PC LOCATION
        ADD     HL,SP
        LD      (HL),D          ;LOW BYTE
        DEC     HL
        LD      (HL),E          ;HIGH BYTE
        LD      A,C
        CP      CR              ;SEE IF A CR WAS LAST ENTERED
        JP      Z,GO3
GO0:    LD      B,NBKPTS
        LD      HL,TLOC         ;POINT TO TRAP STORAGE
        ADD     HL,SP
GO1:    PUSH    BC              ;SAVE NUMBER OF BREAKPOINTS
        PUSH    HL              ;SAVE STORAGE POINTER
        LD      B,2             ;SET UP TO GET TRAP ADDRESS
        CALL    EXPR1           ;GET A TRAP ADDRESS
        POP     DE              ;GET THE TRAP ADDRESS INTO (D,E)
        POP     HL              ;REGET THE STORAGE ADDRESS
        LD      A,D             ;INSURE THE TRAP ADDRESS ISN'T ZERO
        OR      E
        JP      Z,GO2           ;JUMP IF SO
        LD      (HL),E          ;SAVE THE BREAKPOINT ADDRESS
        INC     HL
        LD      (HL),D
        INC     HL
        LD      A,(DE)          ;SAVE THE INSTRUCTION FROM THE BP ADDR
        LD      (HL),A
        INC     HL
        LD      A,0CFH          ;INSERT THE BREAKPOINT
        LD      (DE),A
GO2:    LD      A,C             ;REGET THE DELIMITER TO SEE
        CP      CR              ;  IF WE ARE DONE SETTING BREAKPOINTS
        POP     BC              ;  UNLOAD THE STACK FIRST
        JP      Z,GO3           ;YES, JUMP
        DEC     B
        JP      NZ,GO1          ;JUMP IF NOT AT BP LIMIT
GO3:    CALL    CRLF
        POP     HL              ;GET RID OF STACK JUNK
        LD      HL,RS9
        PUSH    HL
        LD      HL,24           ;FIND REGISTER SET ROUTINE ADDRESSS
        ADD     HL,SP
        POP     DE              ;ADJUST THE STACK
        JP      (HL)            ;GO TO THE DESIRED PLACE
        PAGE
;-----------------------------------------------------------------
; GENERAL PURPOSE INPUT/OUTPUT ROUTINES
;
; THESE ROUTINES ALLOW BYTE-BY-BYTE INPUT OR OUTPUT FROM
; THE CURRENT CONSOLE DEVICE.  THEY ARE INVOKED BY
; THE MONITOR "I" OR "O" COMMAND.
;-----------------------------------------------------------------
INPT:   CALL    EXPR1           ;GET INPUT PORT NUMBER
        POP     BC              ;GET PORT # INTO C REGISTER
```

```
        IN      E,(C)
        PUSH    DE
        CALL    CRLF
        JP      BITS2           ;GO DO A BINARY PRINT OF THE VALUE
OUTX:   CALL    EXPR            ;GET THE ADDRESS AND DATA FOR OUTPUT
        POP     DE              ;DATA VALUE INTO E
        POP     BC              ;PORT INTO C
        OUT     (C),E
        RET
;---------------------------------------------------------------
; MOVE ROUTINE EXPECTS THREE PARAMETERS, ENTERED IN THE
; SOURCE FIRST BYTE ADDRESS
; SOURCE LAST BYTE ADDRESS
; DESTINATION FIRST BYTE ADDRESS
;---------------------------------------------------------------
MOVE:   CALL    EXPR3           ;GET THREE PARAMETERS
MOV1:   LD      A,(HL)          ;GET NEXT BYTE
        LD      (BC),A          ;MOVE IT
        CALL    HILOXB          ;GO INCREMENT, CHECK SOURCE POINTER
        JP      MOV1            ;NOT THERE YET, GO DO IT AGAIN
;-----------
; ROUTINE TO PERFORM WORD STORE FOR CHANGING VECTORS
;
WORDST: CALL    EXPR
        LD      (HL),E
        INC     HL
        LD      (HL),D
        RET
        PAGE
;---------------------------------------------------------------
; SUBSTITUTE ACTION ROUTINE
;
; THIS ROUTINE ALLOWS THE USER TO INSPECT ANY MEMORY LOCATION
; AND ALTER THE CONTENTS, IF DESIRED AND IF THE ADDRESS
; IS IN RAM.  THE CONTENTS MAY BE LEFT UNALTERED
; BY ENTERING A SPACE, COMMA, OR A CARRIAGE RETURN.  IF
; A CARRIAGE RETURN IS ENTERED, THE ROUTINE IS TERMINATED
; IF A SPACE OR COMMA IS ENTERED, THE ROUTINE
; PROCEEDS TO THE NEXT LOCATION AND PRESENTS THE USER
; WITH AN OPPORTUNITY TO ALTER IT.
;---------------------------------------------------------------
SUBS:   CALL    EXPR1           ;GO GET ONE PARAMETER
        POP     HL              ;GET THE START ADDRESS
        CALL    LADRB           ;PRINT THE ADDRESS
SUB1:   LD      A,(HL)          ;GET THE CONTENTS OF THE ADDRESS
        CALL    DASH1           ;DISPLAY IT ON CONSOLE AND A DASH
        CALL    PCHK            ;GET, CHECK CHARACTER
        JP      C,SUB2          ;NO CHANGE IF CR
        JP      Z,SUB2          ;OR BLANK OR ','
        CP      '-'             ;SEE IF PREVIOUS BYTE WANTED
        JP      Z,SUB3          ;YES, DO IT
        PUSH    HL              ;SAVE MEMORY POINTER
        CALL    EXF             ;GO GET REST OF NEW VALUE
        POP     DE              ;NEW VALUE TO E REGISTER
        POP     HL              ;RESTORE MEMORY POINTER
```

```
        LD      (HL),E          ;PUT DOWN NEW VALUE
        LD      A,C             ;GET THE DELIMITER
SUB2:   CP      ','             ;SEE IF DONE
        RET     Z               ;YES, RETURN TO MONITOR
        INC     HL              ;NO, INCREMENT MEMORY POINTER
        INC     HL              ;ALLOW A FALL-THROUGH ON THE NEXT INST
SUB3:   DEC     HL              ;ADJUST (H,L) AS APPROPRIATE
        CALL    LADRB           ;CALL IF ON THE BOUNDARY
        JP      SUB1            ;GO DO THE NEXT LOCATION
        PAGE
;-------------------------------------------------------------------
; MTEST ROUTINE TESTS A SPECIFIED BLOCK OF MEMORY TO
; SEE IF ANY HARD DATA BIT FAILURES EXIST.  IT IS
; NOT AN EXHAUSTIVE TEST, BUT JUST A QUICK INDICATION
; OF THE MEMORY'S OPERATIVENESS.
;-------------------------------------------------------------------
MTEST:  LD      B,1             ;ONE PARAMETER
        CALL    EXPR            ;GET START ADDRESS
        JP      C,MTALL         ;IF (CR) DO ALL OF RAM
        LD      B,1
        CALL    EXPR            ;GET END ADDRESS
        POP     DE
        POP     HL
MTEST1: PUSH    DE              ;SAVE END ADDRESS
        PUSH    HL              ;SAVE START ADDRESS
        LD      HL,MTMSG        ;OUTPUT MESSAGE
        CALL    PRTWD
        POP     HL              ;GET START ADDRESS
        PUSH    HL              ;BACK TO STACK
        CALL    LADR            ;PRINT THE VALUE
        CALL    DASH            ;AND A DASH
        POP     DE              ;START  ADDRESS
        POP     HL              ;END ADDRESS
        PUSH    HL
        PUSH    DE              ;BACK TO STACK
        CALL    LADR            ;PRINT END ADDRESS
        CALL    CRLF
        POP     HL              ;START ADDRESS
        POP     DE              ;END ADDRESS
MTESTX: LD      A,(HL)          ;READ A BYTE
        PUSH    AF              ;SAVE IT
        CPL                     ;COMPLEMENT IT
        LD      (HL),A          ;WRITE IT
        XOR     (HL)            ;RESULT SHOULD BE ZERO
        CALL    NZ,BITS         ;LOG ERROR IF NOT
MTEST2: POP     AF              ;RESTORE ORIGINAL BYTE
        LD      (HL),A
        CALL    HILOX           ;POINT TO NEXT AND SEE IF DONE
        JP      MTESTX          ;NO, CONTINUE
        PAGE
BITS:   PUSH    DE              ;SAVE (D,E)
        LD      E,A             ;SAVE ERROR PATTERN IN E
        CALL    LADRB           ;FIRST PRINT THE ADDRESS
BITS2:  LD      B,8             ;LOOP CONTROL FOR 8 BITS
BITS1:  LD      A,E             ;GET NEXT BIT
```

```
        RLCA                    ;  INTO CARRY
        LD      E,A             ;SAVE REST
        LD      A,'0'/2         ;BUILD ASCII 1 OR 0
        RLA                     ;  CARRY DETERMINES WHICH
        LD      C,A             ;NOW, OUTPUT IT
        CALL    TTYOUT
        DEC     B
        JP      NZ,BITS1        ;DO IT AGAIN
        POP     DE
        RET
MTALL:  POP     DE              ;GET EXPR RESULTS
        LD      A,D
        OR      E               ;ZERO?
        JP      Z,MTALX         ;YES, USE DEFAULT
        LD      HL,(RAMEND)     ;GET END ADDRESS
        EX      DE,HL
        JP      MTAL1           ;AND DO IT
MTALX:  LD      HL,RAMEND       ;LAST ADDRESS
        EX      DE,HL           ;TO D-E
        LD      HL,RAMST
MTAL1:  CALL    CRLF
        JP      MTEST1
        PAGE
;-----------------------------------------------------------------
; EXAMINE REGISTERS COMMAND INSPECTS THE VALUES OF THE
; REGISTERS STORED BY THE LAST ENCOUNTERED BREAKPOINT.
; THE VALUES MAY BE MODIFIED IF DESIRED.
;-----------------------------------------------------------------
XAA:    INC     HL              ;SKIP OVER TO NEXT ENTRY
        INC     HL
XA:     INC     (HL)            ;SEE IF AT END OF TABLE
        RET     Z               ;COULDN'T FIND MATCH, QUIT
        JP      P,XAB           ;SORT OUT BIT 7 OF TABLE
        OR      80H             ;SET IT ON TEST VALUE
        JP      XAC
XAB:    AND     7FH             ;RESET BIT 7
XAC:    DEC     (HL)            ;TO BE PULLED OUT IN ROM
        CP      (HL)            ;SEE IF THIS IS IT
        JP      NZ,XAA          ;NO, GO TRY AGAIN
        CALL    BLK             ;YES, PREPARE TO SHOW CURRENT VALUE
        CALL    PRTVAL          ;GO PRINT THE VALUE
        CALL    DASH            ;PROMPT A NEW VALUE
        CALL    PCHK            ;GET THE INPUT
        RET     C               ;DONE IF CARRIAGE RETURN
        JP      Z,XF            ;JUMP IF NO CHANGE DESIRED
        PUSH    HL              ;TO BE CHANGED, SAVE POINTER
        CALL    EXF             ;GET THE NEW VALUE
        POP     HL              ;  INTO (H,L)
        LD      A,L             ;GET THE NEW LOW BYTE
        INC     DE              ;ADJUST POINTER
        LD      (DE),A          ;PUT IT DOWN
        EX      (SP),HL         ;RECOVER THE TABLE POINTER
        LD      A,(HL)          ;GET THE ATTRIBUTES
        EX      (SP),HL         ;SET THE STACK STRAIGHT
        RLCA                    ;SEE IF 8 BIT REGISTER
```

```
        JP      NC,XE           ;JUMP IF SO
        INC     DE              ;REGISTER PAIR, DO OTHER 8 BITS
        LD      A,H
        LD      (DE),A
XE:     POP     HL              ;RESTORE THE TABLE POINTER
XF:     LD      A,C             ;SEE IF IT WAS A CR
        CP      CR
        RET     Z               ;DONE IF SO
XMNE:   LD      HL,ACTBL        ;GET ADDRESS OF REGISTER LOOK-UP TABLE
XMNE1:  CALL    PCHK            ;FIND OUT WHAT ACTION IS WANTED
        JP      C,XG            ;SHOW ALL IF CARRIAGE RETURN
        JP      Z,XMNE1         ;IGNORE BLANKS OR COMMAS
        CP      ''''            ;SEE IF PRIMES WANTED
        JP      NZ,XA           ;NO, MUST BE SINGLE REGISTER
        LD      HL,PRMTB        ;YES, SET TABLE ADDRESS
        JP      XMNE1           ;  AND FIND OUT WHICH ONE
        PAGE
XG:     LD      A,(HL)
        LD      C,A
        INC     A               ;SEE IF AT END OF TABLE
        RET     Z               ;DONE IF SO
        CALL    M,CRLF          ;START A NEW LINE IF BIT 7 IS SET
        CALL    TTYOUT
        CALL    DASH
        CALL    PRTVAL          ;GO PRINT THE VALUE
        CALL    BLK             ;AND A SPACE
        INC     HL              ;POINT TO NEXT ENTRY
        JP      XG              ;DO THE NEXT VALUE
PRTVAL: INC     HL              ;POINT TO NEXT ENTRY
        LD      A,(HL)          ;GET OFFSET AND ATTRIBUTES BYTE
        AND     3FH             ;ISOLATE THE OFFSET
        ADD     A,2             ;ALLOW FOR RETURN ADDRESS
        EX      DE,HL           ;SWAP POINTERS
        LD      L,A             ;BUILD THE ADDRESS OF THE REG CONTENTS
        LD      H,0
        ADD     HL,SP
        EX      DE,HL           ;RE-SWAP THE POINTERS
        LD      A,(HL)          ;NOW FIND OUT ATTRIBUTES
        LD      B,1             ;SET UP FOR SINGLE REG VALUE
        RLCA
        JP      NC,PV1          ;JUMP IF SINGLE REGISTER VALUE WANTED
        INC     B               ;SET UP FOR REGISTER PAIR
        RLCA
        JP      NC,PV1          ;JUMP IF REGISTER PAIR IS NEXT
        PUSH    HL              ;SPECIAL CASE FOR MEMORY REGISTER
        LD      A,(DE)          ;BUILD ADDRESS IN (H,L)
        LD      H,A
        DEC     DE
        LD      A,(DE)
        LD      L,A
        LD      A,(HL)          ;GET THE MEMORY VALUE
        POP     HL              ;RESTORE (H,L)
        DEC     B
        JP      NZ,PV2          ;ALWAYS JUMP
PV1:    LD      A,(DE)          ;GET THE REGISTER CONTENTS
```

```
PV2:    CALL    HEX1            ;OUTPUT THE VALUE
        DEC     DE              ;ADJUST THE MEMORY POINTER
        DEC     B
        JP      NZ,PV1
        RET
        PAGE
;---------------------------------------------------------------
; ROUTINE CONV CONVERTS THE LOW ORDER NIBBLE OF THE
; ACCUMULATOR TO ITS ASCII EQUIVALENT.  IT
; PUTS THE RESULT INTO C FOR LATER OUTPUT.
;---------------------------------------------------------------
CONV:   AND     0FH             ;STRIP OFF BITS 4-7
        ADD     A,90H           ;PUT ON THE ASCII ZONE
        DAA
        ADC     A,40H
        DAA
        LD      C,A             ;PUT IN OUTPUT PASS REGISTER
        RET
;---------------------------------------------------------------
; ROUTINE ECHO CAN READ A BYTE FROM A FULL-DUPLEX CONSOLE
; DEVICE, THEN ECHOES THE CHARACTER BACK TO THE
; CONSOLE.
;---------------------------------------------------------------
DECHO:  CALL    DASH            ;PRINT A DASH
ECHO:   CALL    TTYIN           ;CONSOLE READ, WRITE ROUTINE
        RET
;---------------------------------------------------------------
; ROUTINE EXPR3 GETS THREE PARAMETERS, DOES A CR, LF AND
; THEN LOADS (B,C), (D,E), AND (H,L) WITH THE PARAMETERS.
;---------------------------------------------------------------
EXPR3:  INC     B               ;2 IS ALREADY IN THE B REGISTER
        CALL    EXPR            ;GET THE PARAMETERS
        POP     BC              ;PUT PARAMETERS INTO REGISTERS
        POP     DE
        JP      CRLFA           ;GO DO THE CARRIAGE RETURN SEQUENCE
        PAGE
;---------------------------------------------------------------
; ROUTINE HILD INCREMENTS (H,L).  IT THEN CHECKS FOR (AND
; DISALLOWS) A WRAP-AROUND SITUATION.  IF IT OCCURS,
; THE CARRY BIT WILL BE SET ON RETURN.  IF NO WRAP-
; AROUND OCCURRED, (H,L) IS COMPARED TO (D,E) AND
; THE FLAG BITS SET ACCORDINGLY.
;---------------------------------------------------------------
HILD:   INC     HL              ;INCREMENT (H,L)
        LD      A,H             ;TEST IF ZERO
        OR      L               ;  IN (H,L)
        SCF                     ;SET CARRY FOR (H,L)=0
        RET     Z               ;RETURN IF (H,L) = 0
        LD      A,E             ;COMPARE (H,L) TO (D,E)
        SUB     L
        LD      A,D
        SBC     A,H
        RET                     ;RETURN WITH FLAGS SET
;---------------------------------------------------------------
; ROUTINE HILDX INCREMENTS (H,L), COMPARES IT TO (D,E) AND
```

```
; IF EQUAL, RETURNS CONTROL TO THE MONITOR EXECUTIVE.
; OTHERWISE, CONTROL RETURNS TO THE CALLING ROUTINE.
;-------------------------------------------------------------------
HILOXB: INC     BC              ;INCREMENT (B,C)
HILOX:  CALL    HILO            ;INC AND CHECK (H,L)
        JP      C,WINIT         ;DONE IF CARRY SET
        CALL    TTST            ;SEE IF CONSOLE BREAK PENDING
        OR      A
        RET     Z               ;NONE, RETURN TO CONTINUE
        JP      WINIT
        PAGE
;-------------------------------------------------------------------
; ROUTINE NIBBLE CONVERTS THE ASCII CHARACTERS 0-9 AND
; A-F TO THEIR EQUIVALENT HEXADECIMAL VALUE.  IF
; THE CHARACTER IS NOT IN RANGE, THE CARRY BIT IS SET TO
; FLAG THE ERROR.
;-------------------------------------------------------------------
NIBBLE: SUB     '0'             ;ASCII TO HEX CONVERSION
        RET     C               ;  DONE IF OUT OF RANGE
        CP      'G'-'0'         ;CHECK UPPER END
        CCF                     ;  TOGGLE THE CARRY BIT
        RET     C               ;  DONE IF OUT OF RANGE
        CP      '9'-'0'+1       ;SEE IF NUMERIC
        CCF                     ;  TOGGLE THE CARRY BIT
        RET     NC              ;  DONE IF SO
        SUB     'A'-'9'-1       ;SUBTRACT THE ALPHA BIAS
        CP      10              ;  SET CARRY FOR INVALID CHAR
        RET
;-------------------------------------------------------------------
; ROUTINE PCHK READS A CHARACTER FROM THE CONSOLE, THEN
; CHECKS IT FOR A DELIMITER.  IF IT IS NOT
; A DELIMITER, A NON-ZERO CONDITION IS RETURNED.
; IF IT IS A DELIMITER, A ZERO CONDITION IS RETURNED.
; FURTHER, IF THE DELIMITER IS A CARRIAGE RETURN,
; THE CARRY BIT IS SET.  A BLANK OR A COMMA RESETS
; THE CARRY BIT.
;-------------------------------------------------------------------
PCHK:   CALL    ECHO            ;GET, TEST FOR DELIMITER
P2C:    CP      ' '             ;  BLANK?
        RET     Z               ;  YES, DONE
        CP      ','             ;  NO, COMMA?
        RET     Z               ;  YES, DONE
        CP      '.'             ;PERIOD
        RET     Z               ;YES, SPECIAL DELIMITER
        CP      CR              ;  NO, CARRIAGE RETURN?
        SCF                     ;  SHOW IT IN CARRY BIT
        RET     Z               ;  DONE IF CR
        CCF                     ;CLEAR CARRY FOR NO DELIMITER
        RET
        PAGE
;-------------------------------------------------------------------
; ROUTINE REST TRAPS ALL OF THE REGISTER CONTENTS WHENEVER A
; RESTART 1 INSTRUCTION IS EXECUTED.  THE TRAPPED CONTENTS
; ARE STORED IN THE SYSTEM STACK AREA FOR LATER ACCESS AND
; USE BY THE GOTO AND THE EXAMINE REGISTERS COMMANDS.
```

```
;-----------------------------------------------------------------
REST::  DI                      ;DISABLE INTERRUPTS
        PUSH    HL              ;SAVE ALL THE REGISTERS
        PUSH    DE
        PUSH    BC
        PUSH    AF
        CALL    MEMSIZ          ;GET MEMORY PARAMETERS
        EX      DE,HL
        LD      HL,10           ;GO UP 10 BYTES IN THE STACK
        ADD     HL,SP           ;  TO SKIP OVER TEMP REGISTER SAVE
        LD      B,4             ;PICK OFF THE REGISTER VALUES
        EX      DE,HL
RS1:    DEC     HL
        LD      (HL),D          ;SAVE IN WORK AREA
        DEC     HL
        LD      (HL),E
        POP     DE
        DEC     B
        JP      NZ,RS1
        POP     BC              ;GET THE BREAKPOINT LOCATION
        DEC     BC
        LD      SP,HL           ;SET THE MONITOR STACK
        LD      HL,TLOCX        ;SET UP TO RESTORE BREAKPOINTS
        ADD     HL,SP
        PUSH    DE
        LD      D,NBKPTS        ;LOOP CONTROL FOR N BREAKPOINTS
RS2:    LD      A,(HL)
        SUB     C               ;SEE IF A SOFTWARE TRAP
        INC     HL
        LD      A,(HL)
        SBC     A,B             ;MAYBE, TRY REST OF ADDRESS
        JP      Z,RS5           ;FOUND ONE, JUMP TO RESET IT
RS3:    INC     HL              ;NOT FOUND, TRY NEXT ONE
        INC     HL
        DEC     D
        JP      NZ,RS2
        PAGE
RS4:    INC     BC              ;NONE FOUND
RS5:    LD      HL,LLOCX
        POP     DE
        ADD     HL,SP
        LD      (HL),E          ;STORE USER (H,L)
        INC     HL
        LD      (HL),D
        PUSH    BC              ;SAVE (B,C)
        LD      C,'*'           ;TYPE THE BREAK INDICATION
        CALL    TTYOUT
        POP     DE              ;REGET THE BREAKPOINT LOCATION
        LD      A,RS9/256
        CP      D               ;SEE IF A RET BREAKPOINT
        JP      Z,RS6
        INC     HL
        INC     HL
        LD      (HL),E          ;RESTORE USER PROGRAM COUNTER
        INC     HL
```

```
        LD      (HL),D
        EX      DE,HL           ;PRINT THE BREAKPOINT LOCATION
        CALL    LADR
RS6:    LD      HL,TLOCX
        ADD     HL,SP
        LD      BC,NBKPTS*256
RS7:    LD      E,(HL)          ;RESTORE BREAKPOINTED LOCATIONS
        LD      (HL),C          ;RESET SYSTEM BP SAVE AREA
        INC     HL
        LD      D,(HL)
        LD      (HL),C
        INC     HL
        LD      A,E
        OR      D
        JP      Z,RS8           ;DO NOTHING IF ZERO
        LD      A,(HL)
        LD      (DE),A
        PAGE
RS8:    INC     HL              ;SAME THING FOR OTHER
        DEC     B
        JP      NZ,RS7          ;  BREAKPOINT
        EX      AF,AF'                  ;NOW SAVE THE Z-80 UNIQUES
        EXX
        PUSH    HL
        PUSH    DE
        PUSH    BC
        PUSH    AF
        PUSH    IX
        PUSH    IY
        LD      A,I
        LD      B,A
        LD      A,R
        LD      C,A
        PUSH    BC
        JP      WINITA
RS9:    PUSH    HL              ;RET BREAKPOINT ENCOUNTERED
        RST     8*1             ;DO THE BREAKPOINT
EXIT:   POP     BC
        LD      A,C
        LD      R,A
        LD      A,B
        LD      I,A
        POP     IX
        POP     IY
        POP     AF
        POP     BC
        POP     DE
        POP     HL
        EX      AF,AF'
        EXX
        POP     DE
        POP     BC
        POP     AF
        POP     HL
        LD      SP,HL
```

```
        DEFB    0               ;ENABLE INTERRUPTS HERE
        LD      HL,0
        JP      0               ;WILL BE FILLED WITH PROPER ADDRESS
ENDX    EQU     $
        PAGE
;-----------------------------------------------------------------
MEMSIZ: PUSH    BC
        LD      HL,RAMEND       ;LAST RAM ADDRESS
        LD      BC,EXIT-ENDX-3*NBKPTS+1;TAKE OFF WORK SPACE
        ADD     HL,BC
        POP     BC
        RET
;-----------------------------------------------------------------
; TWO TYPES OF ERRORS ARE DETECTED:  A RESTART
; ERROR AND CERTAIN PROGRAM ERRORS (DETERMINED
; BY THE PARTICULAR ROUTINE WHERE THE ERROR
; CONDITION WAS ENCOUNTERED.)  EACH CAUSES
; A UNIQUE MESSAGE TO BE PRINTED.  A WARM
; RESTART IS THEN DONE.
;-----------------------------------------------------------------
IOER:   LD      HL,IOMSG        ;GET ADDRESS OF I/O ERROR AND
        JP      COMERR          ;GO PROCESS IT
DIV2:   OR      A               ;CLEAR THE CARRY BIT
        LD      A,H             ;DO A 16-BIT RIGHT SHIFT
        RRA
        LD      H,A
        LD      A,L
        RRA
        LD      L,A
        RET
;-----------------------------------------------------------------
; THIS ROUTINE ADDS AND SUBTRACTS TWO HEXADECIMAL 16-BIT
; UNSIGNED NUMBERS AND DISPLAYS THE RESULTS ON THE
; CONSOLE.
;-----------------------------------------------------------------
HEXN:   CALL    EXLF            ;GET THE TWO NUMBERS
        PUSH    HL              ;SAVE IT FOR THE SUBTRACT
        ADD     HL,DE           ;ADD THEM
        CALL    LADRB           ;OUTPUT THEM
        POP     HL              ;REGET THE FIRST NUMBER
        OR      A               ;CLEAR THE CARRY BIT
        SBC     HL,DE           ;DO THE SUBTRACT
        JP      LADR            ;GO OUTPUT THE RESULT
        PAGE
;-----------------------------------------------------------------
;  ROUTINE LADR PRINTS THE CONTENTS OF (H,L) ON THE
;  CURRENT CONSOLE, EITHER AT THE START OF A NEW
;  LINE (EP = LADRA) OR AT THE CURRENT LOCATION (EP = LADR).
;-----------------------------------------------------------------
LADRA:  CALL    CRLF            ;START A NEW LINE
LADR::  LD      A,H             ;GET HIGH TWO DIGITS
        CALL    HEX1            ;PRINT THEM
        LD      A,L             ;GET LOW TWO DIGITS
HEX1:   PUSH    AF              ;SAVE THE LOW DIGIT
        RRCA                    ;PUT HIGH NIBBLE INTO BITS 0-3
```

```
        RRCA
        RRCA
        RRCA
        CALL    HEX2            ;GO PRINT SINGLE DIGIT
        POP     AF              ;REGET THE LOW DIGIT
HEX2:   CALL    CONV            ;GO INSERT ASCII ZONE
        JP      TTYOUT          ;DO THE CHARACTER OUTPUT
;-----------------------------------------------------------------
;  ROUTINE DASH TYPES A DASH ON THE CURRENT CONSOLE DEVICE.
;-----------------------------------------------------------------
DASH1:  CALL    HEX1            ;FIRST, PRINT ACCUM AS TWO HEX DIGITS
DASH:   LD      C,'-'           ;GET AN ASCII DASH
        JP      TTYOUT          ;GO TYPE IT
;-----------------------------------------------------------------
LADRB:  CALL    LADRA           ;OUTPUT (H,L) AS 4 ASCII DIGITS
;-----------------------------------------------------------------
BLK:    LD      C,' '           ;OUTPUT A BLANK
        JP      TTYOUT
;-----------------------------------------------------------------
;  ROUTINE CONI READS THE CONSOLE AND STRIPS OFF THE ASCII
;  PARITY BIT.
;-----------------------------------------------------------------
CONI:   CALL    TTYIN           ;GET THE NEXT CHARACTER
        AND     7FH             ;STRIP OFF THE PARITY BIT
RTS:    RET
;-----------------------------------------------------------------
; ROUTINE PRTWD PRINTS AN ASCII STRING ONTO THE CONSOLE.
; THE STRING MUST BE TERMINATED BY A ZERO BYTE AS THE
; LAST CHARACTER OF THE STRING.  THE STRING WILL START
; A NEW LINE (EP = PRTWD) OR CONTINUE ON THE SAME
; LINE (EP = PRTWA)
;-----------------------------------------------------------------
PRTWD:  CALL    CRLF            ;START A NEW LINE
PRTWA:  PUSH    BC              ;SAVE (B,C)
PRTA:   LD      C,(HL)          ;GET NEXT CHARACTER FROM MEMORY
        CALL    TTYOUT          ;OUTPUT IT
        INC     HL              ;INCREMENT MEMORY POINTER
        LD      A,(HL)          ;GET NEXT VALUE
        OR      A               ;TEST FOR ZERO
        JP      NZ,PRTA         ;NOT ZERO, GO DO NEXT CHARACTER
PRTB:   POP     BC              ;RESTORE (B,C)
        RET
        PAGE
;-----------------------------------------------------------------
; ROUTINE EXLF READS TWO PARAMETERS, PUTS THEM INTO THE
; D,E AND H,L REGISTERS, THEN DOES A CARRIAGE RETURN,
; LINE FEED SEQUENCE.
;-----------------------------------------------------------------
EXLF:   CALL    EXPR            ;GO GET TWO PARAMETERS
        POP     DE
        POP     HL
;-----------------------------------------------------------------
; ROUTINE CRLF GENERATES A CARRIAGE RETURN, LINE FEED
; SEQUENCE ON THE CURRENT CONSOLE TO START A NEW LINE
;-----------------------------------------------------------------
```

```
CRLF:   PUSH    HL              ;SAVE THE CONTENTS OF (H,L)
CRLFA:  LD      HL,CRMSG        ;ADDRESS OF CR,LF MESSAGE
        CALL    PRTWA           ;  OUTPUT IT
        POP     HL              ;RESTORE (H,L)
        RET
;
RSTER:: LD      HL,RSTMSG       ;GET ADDRESS OF RESTART ERROR MSG
COMERR: CALL    PRTWD           ;PRINT IT ON NEW LINE
        JP      WINIT           ;GO TO WARM BOOT
;
TTYIN:  JP      GETBYT
TTYOUT: LD      A,C
        JP      PUTBYT
DIRET:  DI
        RET
;
TTST:   LD      A,(PBDATA)
        CPL
        AND     80H
        RET
;-----------------------------------------------------------------
;  SYSTEM AND CONTROL EQUATES
;-----------------------------------------------------------------
NBKPTS  EQU     2               ;NUMBER OF MONITOR BREAKPOINTS
CTRLS   EQU     13H             ;ASCII DC3
BELL    EQU     7               ;DING-DONG
;-----------------------------------------------------------------
;  I/O PORT EQUATES
;-----------------------------------------------------------------
        ;USER I/O EQUATES GO HERE
        PAGE
;-----------------------------------------------------------------
;  REGISTER STORAGE DISPLACEMENTS FROM
;  NORMAL SYSTEM STACK LOCATION
;-----------------------------------------------------------------
ALOC    EQU     15H
BLOC    EQU     13H
CLOC    EQU     12H
DLOC    EQU     11H
ELOC    EQU     10H
FLOC    EQU     14H
HLOC    EQU     31H
LLOC    EQU     30H
PLOC    EQU     34H
SLOC    EQU     17H
TLOC    EQU     35H
TLOCX   EQU     25H
LLOCX   EQU     20H
APLOC   EQU     9
BPLOC   EQU     11
CPLOC   EQU     10
DPLOC   EQU     13
EPLOC   EQU     12
FPLOC   EQU     8
HPLOC   EQU     15
```

```
LPLOC   EQU     14
XLOC    EQU     7
YLOC    EQU     5
RLOC    EQU     2
ILOC    EQU     3
        PAGE
;-----------------------------------------------------------------
;  ROM TABLES AND CONSTANTS
;-----------------------------------------------------------------
;  MESSAGES
;-----------------------------------------------------------------
IOMSG:  DEFB    'I/O ERR',0
QMSG:   DEFB    CR,LF,'WHAT?',0
LOGMSG: DEFB    'UCS Z80 ROM MONITOR ver. 1.2 (11nov82)'
        DEFB    CR,LF,0
RSTMSG: DEFB    CR,LF,'RST ERR',0
CRMSG:  DEFB    CR,LF,0
MTMSG:  DEFB    'TEST MEMORY ADDRESSES: ',0
QUEST:  DEFB    '?? - ',0
        PAGE
;-----------------------------------------------------------------
;  ACTION ROUTINE ADDRESS TABLE
;-----------------------------------------------------------------
TBL:    DEFW    QPRT            ;A
        DEFW    QPRT            ;B
        DEFW    COMP            ;C
        DEFW    DISP            ;D
        DEFW    QPRT            ;E
        DEFW    FILL            ;F
        DEFW    GOTO            ;G
        DEFW    HEXN            ;H
        DEFW    INPT            ;I
        DEFW    QPRT            ;J
        DEFW    WORDST          ;K
        DEFW    QPRT            ;L
        DEFW    MOVE            ;M
        DEFW    QPRT            ;N
        DEFW    OUTX            ;O
        DEFW    QPRT            ;P
        DEFW    QPRT            ;Q
        DEFW    QPRT            ;R
        DEFW    SUBS            ;S
        DEFW    MTEST           ;T
        DEFW    QPRT            ;U
        DEFW    QPRT            ;V
        DEFW    QPRT            ;W
        DEFW    XMNE            ;X
        DEFW    QPRT            ;Y
        DEFW    QPRT            ;Z
        PAGE
;-----------------------------------------------------------------
;  Z80 REGISTER OFFSET TABLE
;-----------------------------------------------------------------
ACTBL:  DEFB    80H+'A',ALOC
        DEFB    'B',BLOC
```

```
        DEFB    'C',CLOC
        DEFB    'D',DLOC
        DEFB    'E',ELOC
        DEFB    'F',FLOC
        DEFB    'H',HLOC
        DEFB    'L',LLOC
        DEFB    80H+'M',HLOC+0C0H
        DEFB    'P',PLOC+80H
        DEFB    'S',SLOC+80H
        DEFB    'I',ILOC
;-----------------------------------------------------------------
;  REST OF Z-80 REGISTER OFFSETS
;-----------------------------------------------------------------
PRMTB:  DEFB    80H+'A',APLOC
        DEFB    'B',BPLOC
        DEFB    'C',CPLOC
        DEFB    'D',DPLOC
        DEFB    'E',EPLOC
        DEFB    'F',FPLOC
        DEFB    'H',HPLOC
        DEFB    'L',LPLOC
        DEFB    80H+'M',HPLOC+0C0H
        DEFB    'X',XLOC+80H
        DEFB    'Y',YLOC+80H
        DEFB    'R',RLOC
        DEFB    0FFH
        PAGE
ROMEND  EQU     $
;-----------------------------------------------------------------
;  VARIABLE STORAGE - UNINITIALIZED RAM
;-----------------------------------------------------------------
;       ORG     RAMST
;-----------------------------------------------------------------
;  MONITOR STORAGE AND WORK AREA RAM
;-----------------------------------------------------------------
        .PHASE  MONRAM
SADDR:  DEFS    2
PRINTF: DEFS    1
SPSV:   DEFS    2
DSADD:  DEFS    2
LSADD:  DEFS    2
NLINES: DEFS    1
        .DEPHASE
        END
```

```
; LAST UPDATE 03/02/83 JHG
; ---- ASSEMBLY CONTROL ----
FALSE   EQU     0               ; Initial definition
TRUE    EQU     NOT FALSE       ;
DIAG    EQU     TRUE            ; Print diagnostics during execution
PRINT   MACRO   STRING
        PUSH    HL              ; Usage:
        LD      HL,STRING       ;               PRINT DIAGMS
        CALL    SNDMES##        ;       DIAGMS: DB 'MESSAGE',0
        POP     HL              ; Note: alters AF
        ENDM
;
; ---- REVC CONSTANTS ----
BIT0    EQU     01H             ; \
BIT1    EQU     02H             ;  \
BIT2    EQU     04H             ;   \
BIT3    EQU     08H             ;    \ Use as values for
BIT4    EQU     10H             ;    /  bit masks
BIT5    EQU     20H             ;   /
BIT6    EQU     40H             ;  /
BIT7    EQU     80H             ; /
;
NSCRAM  EQU     NSCIOT-80H      ; NSC-810 ram area
SYSRAM  EQU     4000H           ; Start of free ram
BUFMEM  EQU     8000H           ; 32K aquisition buffer
PWRDLY  EQU     1500            ; Cartridge power dist delay
DIRLEN  EQU     0800H           ; Max directory length
RECLEN  EQU     2000H           ; Physical rec len (excl hdr)
;
EVENT   EQU     7               ; For bit test in aquire routine
CR      EQU     0DH             ; Carriage return
LF      EQU     0AH             ; Line feed
;
PUTBYT  EQU     10H             ; RST vector: CON = A
GETBYT  EQU     18H             ; RST vector: A = CON
;
; ---- PORT ASSIGNMENTS ----
ANAPRT  EQU     010H            ; Analog board STA & Threshold
AVGPRT  EQU     ANAPRT+2        ; Analog board Feedback control
ADPORT  EQU     018H            ; A-D board base address
FMPORT  EQU     28H             ; FM board base address
INTPRT  EQU     0BBH            ; NSC-800 interrupt mask port
CPORT   EQU     0F0H            ; Cartridge controller base
MA      EQU     CPORT           ; Mode argument
PA      EQU     CPORT+1         ; Position argument
CA      EQU     CPORT+2         ; Command argument
DA      EQU     CPORT+3         ; Data argument
PS      EQU     CPORT+1         ; Port status
PWRPRT  EQU     0FFH            ; Power Interface board base
```

```
; ---- MEMORY MAPPED I/O ----
TMEM    EQU     2000H           ; RTC base address
NSCIOT  EQU     3000H           ; NSC-810 I/O-Timer base
PBDATA  EQU     NSCIOT+01H      ; Port B data reg
PCDATA  EQU     NSCIOT+02H      ; Port C data reg
PBDDIR  EQU     NSCIOT+05H      ; Port B data direction reg
PCDDIR  EQU     NSCIOT+06H      ; Port C data direction reg
NSCMDR  EQU     NSCIOT+07H      ; Mode definition reg
PBCLRB  EQU     NSCIOT+09H      ; Port B bit clear reg
PCCLRB  EQU     NSCIOT+0AH      ; Port C bit clear reg
PBSETB  EQU     NSCIOT+0DH      ; Port B bit set reg
PCSETB  EQU     NSCIOT+0EH      ; Port C bit set reg
TMR0LS  EQU     NSCIOT+10H      ; Timer 0 lsb
TMR0MS  EQU     NSCIOT+11H      ; Timer 0 msb
TMR1LS  EQU     NSCIOT+12H      ; Timer 1 lsb
TMR1MS  EQU     NSCIOT+13H      ; Timer 1 msb
STOPT0  EQU     NSCIOT+14H      ; Timer 0 stop
STRT0   EQU     NSCIOT+15H      ; Timer 0 start
STOPT1  EQU     NSCIOT+16H      ; Timer 1 stop
STRT1   EQU     NSCIOT+17H      ; Timer 1 start
CMDT0   EQU     NSCIOT+18H      ; Timer 0 command reg
CMDT1   EQU     NSCIOT+19H      ; Timer 1 command reg
;
; ---- GLOBAL VARIABLES ----
TIMBUF  EQU     SYSRAM+1780H    ; RTC time buffer
HDRBUF  EQU     SYSRAM+17A0H    ; Record header buffer
FNAME   EQU     SYSRAM+1800H    ; Dir header buffer
HDRMEM  EQU     FNAME+10H       ; Directory ram
;
STACK   EQU     NSCRAM+50H      ; Run time stack
FNPTR   EQU     NSCRAM+50H      ; [2] --) event header
RECPTR  EQU     NSCRAM+52H      ; [2] --) start of event
RSBJMP  EQU     NSCRAM+5AH      ; [1] z80 jump instruction
RSBVEC  EQU     NSCRAM+5BH      ; [2] rst B interrupt vector
RSCJMP  EQU     NSCRAM+5DH      ; [1] z80 jump instruction
RSCVEC  EQU     NSCRAM+5EH      ; [2] rst C interrupt vector
DS      EQU     NSCRAM+60H      ; [1] stored drive status
IS      EQU     DS+1            ; [1] stored interface status
IMA     EQU     DS+2            ; [1] mode arg copy
IPA     EQU     DS+3            ; [1] pos. arg copy
ICA     EQU     DS+4            ; [1] cmd. arg copy
GCODE   EQU     NSCRAM+65H      ; [1] analog board gain code
ABRVEC  EQU     NSCRAM+66H      ; [2] cartridge abort vector
FMCH1   EQU     NSCRAM+68H      ; [1] \
FMCH2   EQU     NSCRAM+69H      ; [1]  \ Gain code for
FMCH3   EQU     NSCRAM+6AH      ; [1]  / FM channels
FMCH4   EQU     NSCRAM+6BH      ; [1] /
SDEL    EQU     NSCRAM+6DH      ; [2] Start delay in minutes
EOPFLG  EQU     NSCRAM+6FH      ; [1] End of program (tr4) flag
DYNDIR  EQU     NSCRAM+74H      ; [2] --) next free for dir
STDIR   EQU     NSCRAM+76H      ; [2] --) end of entry params
SRCODE  EQU     NSCRAM+78H      ; [1] Sample rate value for A-D board
INTMSK  EQU     NSCRAM+7BH      ; [1] Copy of interruot mask
PEVCNT  EQU     NSCRAM+7CH      ; [2] # post event samples
BAUD    EQU     NSCRAM+7EH      ; [2] Baud rate constant
```

```
.Z80
NAME ('TAPLOG')
TITLE   ARMY CONTROL BOARD (REV C)      11/20/82 JHB
SUBTTL  LOG EVENTS
INCLUDE ARMYREVC.EDU
        PAGE
;
; ---- WRITE AQUISITION BUFFER TO TAPE ----
;
WRTREC::
        CALL    WAKEUP##        ; Power up recorder and controller
        LD      A,(DS)          ;
        BIT     3,A             ;
        JR      Z,NOTBOT        ;
        CALL    SLEEP##         ;
        LD      B,0             ;
LP:     LD      A,7             ;
        RST     PUTBYT          ;
        DJNZ    LP
        JR      WRTREC
;
NOTBOT: LD      (IX-1DH),0      ; Set PA for 1
        CALL    RSFILE##        ; Backspace over last file mark
        LD      A,(DS)
        BIT     3,A
        JR      Z,DUMPAQ
        CALL    SLEEP##
        LD      BC,0
DLLL:   LD      A,B
        OR      C
        JR      NZ,DLLL
        CALL    WAKEUP##
        JP      AT.EOT##
;
DUMPAQ::
        LD      HL,(FNPTR)      ; Get pointer to time header
        LD      DE,(RECPTR)     ; and start of record
        LD      B,4             ; Set index for 4 records per event
DMPLP:  CALL    SNDMPA##        ; Reset cartridge buffer pointer
        PUSH    BC              ; Save index
        PUSH    DE              ; and start pointer
        LD      BC,10H          ; Select transfer swath
        CALL    WRDATA##        ; Send time header
        POP     DE              ; Restore start pointer
        PUSH    HL              ; save header pointer
        EX      DE,HL           ; set up for transfer
        LD      BC,RECLEN       ; Select swath
        CALL    WRDATA##        ; Send 8k data block
        CALL    WRCART##        ; Write 1 record to tape
        EX      DE,HL           ; Restore start pointer to next block
        POP     HL              ; Restore header pointer
        POP     BC              ; and index
        JP      C,AT.EOT##      ; Exit loop if at end of tape
        DJNZ    DMPLP           ; else loop for 4 records
        CALL    WREOF##         ; Write end of file mark
```

```
        LD      A,(DS)          ;
        BIT     2,A             ;
        JP      NZ,AT.EOT##     ;
        CALL    SLEEP##         ; Power down cartridge and controller
;
; ---- APPEND EVENT TIME TO DIRECTORY ----
;
APPEND: LD      HL,(FNPTR)      ; Point HL at filename (time)
        PUSH    HL              ; Save start position
        LD      BC,10           ; Time data is ten bytes long
        ADD     HL,BC           ; Get offset to end
        LD      (HL),0          ; Add terminator to string
        POP     HL              ; Restore front pointer
        LD      BC,(DYNDIR)     ; Point BC at current end of directory
        LD      A,CR            ; Initialize A for CR,LF terminator
        CALL    FILHDR##        ; Append event to directory
        LD      A,1AH           ; and mark end with ASCII end of file
        LD      (BC),A          ;
        LD      (DYNDIR),BC     ; Store new directory pointer
        RET                     ; Setup for next event
        END
```

```
.Z80
NAME ('PAGE0')
TITLE   ARMY CONTROL BOARD (REV F)      COPYRIGHT BY OHH 7/24/83
SUBTTL  PAGE ZERO INTERRUPT AND RST VECTORS
;
INCLUDE REVF.EQU
;THIS MACRO DELAYS N HUNDRED T-STATES
;
DELAY   MACRO   NHUNDRED
        PUSH    BC              ;11 T --SAVE B REGISTER
        LD      B,NHUNDRED-1    ; 7 T --SET UP B FOR DJNZ IN DELAYR
        CALL    DELAYR  ;17 T --CALL DELAY SUBROUTINE
        POP     BC              ;10 T --RESTORE B AND CONTINUE
        ENDM                    ;END OF MACRO
;
;MACRO USES 45 TSTATES.DELAYR WASTES 55 TSTATES AND CALLS A 100 T-STATE
;DELAY N-1 TIMES FOR A TOTAL DELAY OF N HUNDRED TSTATES.  NOTE THAT MAX.
;DELAY FOR THIS MACRO IS 25.6 MS.  DELAY TIMES ARE BASED ON T-STATE=1 MICRO-
;SEC WHICH IS ONLY APPROXIMATE.  APPROXIMATION IS NOT USED FOR CRITICAL DELAYS.;
;*****************************************************************************

ASEG
        ORG     0000H
;
*****************************************************************************
*   PROCESSOR STARTS HERE ON RESET.INTERRUPT AND REFRESH REGS ARE           *
*   CLEARED.ALL INTERRUPTS ARE DISABLED(HARDWARE DI INSTRUCTION).INTERRUPT  *
*   CONTROL REGISTER IS SET TO 01,WHICH ENABLES "NOT INTR" AND MASKS OFF    *
*   "NOT" RSTA,RSTB,RSTC.REMEMBER -- NO INTERRUPT CAN WORK UNLESS BOTH      *
*   AN EI INSTRUCTION HAS BEEN ISSUED AND THERE IS A ONE IN THE APPROPIATE  *
*   SPOT IN THE INTERRUPT MASK REGISTER,WHICH IS A WRITE ONLY REGISTER       *
*   ADDRESSED AS AN OUTPUT PORT(LOWER 4 BITS ONLY) BY AN OUT BBH OR EQUIV.  *
*   INSTRUCTION.THE FOLLOWING VALUES ENABLE THE CORRESPONDING INT. LINES:   *
*           08H             ENABLES RSTA                                    *
*           04H             ENABLES RSTB                                    *
*           02H             ENABLES RSTC                                    *
*           01H             ENABLES INTR                                    *
*   NOTE THAT 0FH ENABLES ALL INT. LINES                                    *
*   8080 INTERRUPT MODE IS AUTOMATICALLY SELECTED ON RESET                  *
*****************************************************************************
        RESTART 0       (11000111)!
START:  DI                      ; NOT NECESSARY
        JP      MAIN##          ;
;
;       END OF RESET CODE
*****************************************************************************
THESE ARE THE 8 NORMAL Z80 RESTART LOCATIONS.IN 8080 MODE ,THESE
CONSTITUTE A MEANS FOR EXTERNAL DEVICES TO FORCE A CALL TO ONE OF 8 LOCATIONS
WITH ONE INSTRUCTION PUT ON THE BUS DURING AN INTACK CYCLE.ALSO CAN BE USED
AS A VERY SHORT CALL VIA RST INSTRUCTION.
NOTE THAT A DI INSTRUCTION IS AUTOMATICALLY EXECUTED
SEE NATIONAL NSC 800 PAGE 4-17.(ALSO STANDARD ON 8080 AND Z80)
*****************************************************************************
!
;       RESTART 1       (11001111)
```

```
;       THIS ROUTINE USED BY MONITOR PROGRAM TO DO BREAKPOINTS
        ORG     0008H
        JP      REST##          ; GOTO REST ROUTINE IN MONITOR
;
;*****************************************************************************
;NOTE:THIS SHOULD BE ALTERED SINCE THERE IS NO REASONABLE WAY
;WE CAN GET THE CONSOLE TO TRIGGER INTERRUPT 1.MONITOR SHOULD USE
;RST7 AND THE TERMINATOR SHOULD USE RST8 OR RSTC.(MONITOR REQUIRES
;BOTH AN INSTRUCTION AND A HARDWARE RESTART.)
;
;*****************************************************************************
;
;       RESTART 2       (11010111)
        ORG     0010H
        JP      ROMUL8          ;ROMULATOR DISPLAY REGS. ROUTINE(SEE BELOW)
;
;IN Z80 RST18.
;
;       RESTART 3       (11011111)
        ORG     0018H           ;
        JP      ROMUL8          ;ROMULATOR DISPLAY REGS. ROUTINE(SEE BELOW)
;
;*****************************************************************************
;       RESTART 4       (11100111)
                        ;THESE ARE JUMPS TO A RESTART ERROR ROUTINE IN
                        ;THE MONITOR
        ORG     0020H
        JP      ROMUL8          ;ROMULATOR DISPLAY REGS. ROUTINE(SEE BELOW)
;
;       RESTART 5       (11101111)
        ORG     0028H
        JP      ROMUL8          ;ROMULATOR DISPLAY REGS. ROUTINE(SEE BELOW)
;
;       RESTART 6       (11110111)
        ORG     0030H
        JP      ROMUL8          ;ROMULATOR DISPLAY REGS. ROUTINE(SEE BELOW)
;
;*****************************************************************************
;       RESTART 7       (11111111)
                        ;IN MODE 1,WILL COME HERE ON NOT INTR GOING LOW
        ORG     0038H
        JP      ROMUL8          ;ROMULATOR DISPLAY REGS. ROUTINE(SEE BELOW)
;       JP      ETERM##         ; EARLY TERMINATION ROUTINE
;
;NOTE THAT THIS REQUIRES HARDWARE OR MODE1 INTERRUPTS!
;*****************************************************************************
;       NSC     800 SPECIAL INTERRUPT LOCATIONS
;       IF INT. ENABLED AND MASK OK WILL COME HERE AS SHOWN.
;       NO INSTRUCTION NEEDED ON BUS-- JUST PULL THE APPROPIATE PIN LOW.
;*****************************************************************************
;       RESTART C
        ORG     002CH
        RET
;
;       RESTART B
```

```
        ORG     0034H
        RET
;
;*****************************************************************************
;THESE TWO INTERRUPTS ARE NOT USED.
;THESE HAVE NOW BEEN MADE RETURNS FOR SAFETY.7/16/83
;*****************************************************************************
;*****************************************************************************
;                               RESTART A
;
;THIS INTERRUPT IS GENERATED ON THE A/D BOARD.IT IS CONTROLLED BY A NUMBER
;OF CPU BOARD OUTPUTS.THE NSC 810 PORT C PIN 0 CONTROLS AN OR GATE WHICH
;DISABLES THE INTERUPT COMING FROM THE A/D BOARD ON S-100 PIN 4 IF C-0 IS
;HIGH.NOTE THAT C-0 IS NOT CURRENTLY INITIALIZED AND IS THUS RANDOM.
;FURTHERMORE, THE A/D BOARD WILL NOT GENERATE AN INTERRUPT IF ITS FIRST
;INTERRUPT WAS NOT ACKNOWLEDGED.THE A/D INTERRUPT OCCURS IMMEDIATELY AFTER
;CONVERSION COMPLETE FOLLOWING A TIME-OUT OF THE SOFTWARE SETTABLE COUNTER
;WHICH GIVES THE APPROXIMATE SAMPLE(WILL HAVE ABOUT 200 MICROSECONDS OF SLOP)
;TIME.
;*****************************************************************************
        ORG     003CH
AQUINT: PUSH    AF              ; SAVE FLAGS AND ACC.
        LD      BC,0817H        ; SET UP B&C REGS FOR BLOCK INPUT
;*****************************************************************************
;C REGISTER IS THE IO PORT ADDRESS;B REGISTER IS THE COUNTER
;SO FETCH DATA FROM IO 18 THROUGH 25--8 BYTES TOTAL
;*****************************************************************************
AQLP:   INC     C               ; IO PORT=IO PORT + 1
        INI                     ; BLOCK INPUT INSTR.
;
;*****************************************************************************
;FETCH FROM IO PORT (C),STORE AT HL,DECR. B, INCR. HL .NOTE: B IS PUT ON
;THE HIGH ORDER ADDRESS LINES,WHICH ARE GENERALLY NOT LOOKED AT DURING IO.
;*****************************************************************************
;
        JP      NZ,AQLP         ; DO UNTIL B=0 (8 TIMES)
        SET     7,H             ; RAM IS FROM 8000H TO FFFFH.
;
;*****************************************************************************
;TO MAINTAIN CIRCULAR 32K BUFFER,MAKE SURE HL ALWAYS HAS HIGH BIT SET.
;THUS FFFF INCREMENTS TO 0000,BUT THIS SETS IT BACK TO 8000H.
;*****************************************************************************
;
        DEC     DE              ; DE IS USED AS A MULTI-PURPOSE COUNTER IN
                                ; AQUIRE.A RAM LOCATION SHOULD BE USED INSTEAD.
;                               ; DE COUNTS THE NO. OF TIMES
                                ; THAT PROCESSOR HAS BEEN INTERRUPTED.
        POP     AF              ; RESTORE FLAGS&ACC
        EI                      ; ENABLE INTERUPTS
        RET                     ; RETURN
;
;*****************************************************************************
;               SEND A MESSAGE TO THE TERMINAL
;       (THIS SUBROUTINE MUST BE HERE TO AVOID PROBLEMS WITH MACROS)
;
```

```
SNDMES::
        PUSH    AF
MESSND: LD      A,(HL)          ; Get byte at HL.
        CP      0               ; Test for terminater.
        JP      Z,SNDRET        ; Done if 0
        CALL    PBYT##          ; Else print character.
        INC     HL              ; Point at next.
        JR      MESSND          ; and continue.
SNDRET: POP     AF
        RET
;
;*****************************************************************************
;       ROMULATOR ROUTINE
;THIS SUBROUTINE SAVES,DISPLAYS, AND RESTORES ALL REGISTERS.REGISTERS MAY BE
;ALTERED BY:
;       1)TYPING A CONTROL A INSTEAD OF A RETURN AT PROMPT
;               (THIS PUTS YOU IN MONITOR)
;       2)ALTERING MEMORY LOCATIONS SHOWN IN EQUATES BELOW AS DESIRED
;       3)TYPING G FOLOWED BY THE ADDRESS OF RESUME(GET FROM GLOBAL LISTING)
;       4)TYPING A RETURN AT PROMPT
;
;       NOTE THAT BREAKPOINT MUST BE CHANGED IN ROMULATOR PRIOR TO TYPING
;       RETURN.BREAKPOINTS ARE SET BY SUBSTITUTING A RST 2(D7H) FOR FIRST
;       BYTE OF AN INSTRUCTION.REMEMBER TO WRITE DOWN O_D VALUE WHEN YOU
;       DO THIS, SO THAT YOU CAN RESTORE IT BEFORE TYPING RETURN.
;*****************************************************************************
;       ROMULATOR EQUATES
STPTR   EQU     0FFFEH          ;STACK POINTER LOW BYTE(HIGH BYTE IS ALWAYS
;                                ONE UP)
PAF     EQU     0FFFCH          ;AF REGISTERS(F=C,A=D)
PBC     EQU     0FFFAH          ;BC (C=A,B=B)
PDE     EQU     0FFF8H          ;DE
PHL     EQU     0FFF6H          ;HL
PIX     EQU     0FFF4H          ;IX
PIY     EQU     0FFF2H          ;IY
PAFPRIME        EQU     0FFF0H          ;AF'
PBCPRIME        EQU     0FFEEH          ;BC'
PDEPRIME        EQU     0FFECH          ;DE'
PHLPRIME        EQU     0FFEAH          ;HL'
PPC     EQU     0FFE8H          ;PC AT BREAK
BSTACK  EQU     PPC             ;TEMPORARY STACK FOR DISPLAY
PSTPTR  EQU     STPTR           ;TO FIT MACRO
;
PRINTREG        MACRO   STRING
        LD      HL,STRING&M
        CALL    SNDMES
        LD      HL,(P&STRING)
        CALL    LADR##
        CALL    CRLF##
        ENDM
;
MSG     MACRO   STRING
STRING&M::      DB      '&STRING&= '
        DB      0
        ENDM
```

```
;*******************************************************************************
ROMUL8::
        LD      (STPTR),SP      ;SAVE STACK POINTER
        LD      SP,STPTR        ;SET STACK POINTER UP TO SAVE REGS
        PUSHALL                 ;SAVE ALL REGULAR REGS
        PUSH    IX              ;NOW IX
        PUSH    IY              ;AND IY
        EXX                     ;GET ALTERNATE REGS
        EX      AF,AF'          ;AND FLAGS
        PUSHALL                 ;SAVE THEM TOO
;
;ALL REGS AND STACK POINTER NOW SAVED.NOW FIX AND SAVE RETURN ADDRESS
        LD      SP,(STPTR)      ;BACK TO OLD STACK
        POP     HL              ;TAKE RETURN ADDRESS OFF AND PUT IN HL
        DEC     HL              ;ACCOUNT FOR BREAKPOINT
        LD      (PPC),HL        ;SAVE PC
RESUME::        LD      SP,BSTACK       ;TEMPORARY STACK
        LD      HL,CLRSCR##
        CALL    SNDMES
        LD      HL,CNOT
        CALL    SNDMES
;*******************************************************************************
        PRINTREG        PC      ;PC OF BREAK
;PC PRINTED,SO PRINT REST OF REGS
        PRINTREG        STPTR   ;STACK POINTER
;
        PRINTREG        AF      ;AF
;
        PRINTREG        BC      ;BC
;
        PRINTREG        DE      ;DE
;
        PRINTREG        HL      ;HL
;
        PRINTREG        IX      ;IX
;
        PRINTREG        IY      ;IY
;
        PRINTREG        AFPRIME ;AF'
;
        PRINTREG        BCPRIME ;BC'
;
        PRINTREG        DEPRIME ;DE'
;
        PRINTREG        HLPRIME ;HL'
;
;*******************************************************************************
;       NOW PRINT PROMPT
        LD      HL,PROMPT
        CALL    SNDMES
        LD      HL,BLOCKADDR    ;SET UP TO MOVE CODE TO RAM
        LD      DE,BBLOCK
        LD      BC,EBLOCK-BBLOCK
        LDIR                    ;MOVE IT!
        JP      BBLOCK          ;CALL AND WAIT FOR CR IN ANOTHER CHIP!
```

```
;*****************************************************************************
;       THIS BLOCK OF CODE IS LOADED INTO RAM AND EXECUTED THERE
BLOCKADDR:
BBLOCK:
FRAMERR:LD      B,8             ; Set index for number of bits to input.
;
;*****************************************************************************
;
;       WE MUST DETECT THE BEGINNING OF THE START BIT-SO THE FIRST STEP IS TO
;       MAKE SURE THAT IT HAS NOT BEGUN!
;
MARK:   LD      A,(PBDATA)      ; GET INPUT AND MAKE SURE IT IS A MARK
        RLA                     ; INPUT TO CARRY(INPUT IS INVERTED)
        JP      NC,MARK         ; WAIT TILL MARK
;
;       NOW GET TRANSITION
;
GTSTRT: LD      A,(PBDATA)      ;13T-- Look for start bit.
        RLA                     ; 4T-- Move to carry flag.
        JP      C,GTSTRT        ;10T-- IF   start bit not present
                                ; THEN go back and look again.
;
;       MAXIMUM ERROR IN START BIT TIME IS 27 T-STATES
;
;*****************************************************************************
;
;       FOR THIS SYSTEM THE CLOCK IS 1.0486*10^6
;       AND THE NO. OF T-STATES IN ONE BIT TIME FOR COMMON BAUD RATES ARE:
;
;       9600            109.23 T-STATES
;       4800            218.46
;       2400            436.92
;       1200            873.83
;        300           3495.30
;
;       FOR NOW WE ARE RUNNING AT 1200 BAUD, FIXED
;       1/2 BIT TIME = 437 TSTATES FOR ALL INTENTS AND PURPOSES
;
;*****************************************************************************
;*****************************************************************************
;
;       WE NOW HAVE A HYPOTHETICAL START BIT
;
;       SO NOW WAIT 1/2 BIT TIME AND SEE IF IT IS STILL THERE
;
;       TIME NOW=TRANSITION+14 T
        DELAY   4               ; MACRO TO DELAY 400 TSTATES
        JP      CHSTRT          ;10T
;       TIME NOW=TRANS.+424T(13T TO READ LINE MAKES 437)
CHSTRT: LD      A,(PBDATA)      ;13T-- Look for start bit again.
        RLA                     ; 4T-- Move it to carry flag.
        JP      C,GTSTRT        ;10T-- IF   start bit is gone, TRY AGAIN
;
;*****************************************************************************
;
```

```
;       PREPARE TO ECHO START BIT
;
;       TIME IS 451 TSTATES AFTER START BIT EDGE DETECTED
;
;       WE WILL NOW NEED TO DELAY ABOUT 1 BIT TIME SO THAT WE CONTINUE TO HIT
;       THE MIDDLE OF EACH BIT AND SO THAT THE BITS WE ECHO ARE THE RIGHT
;       LENGTH.
;
        LD      A,1             ; 7T-- Prepare A as start bit
        CALL    D50T            ;50T-- DELAY 50T STATES
        LD      (PBCLRB),A      ;13T-- and send it.
        CALL    D27T            ;27T-- WAIT 27 T
;
;*******************************************************************************
;
;       TIME IS NOW 111 T-STATES AFTER MIDDLE OF BIT
;       763 TO GO BEFORE WE READ NEXT,847 (IDEALLY) BEFORE NEXT ECHO
;
;*******************************************************************************
XMIT:   DELAY   7               ; Delay 700 T-STATES
        CALL    D50T            ; DELAY 50 MORE
;763-750=13 WHICH IS HOW LONG IT TAKES TO READ
        LD      A,(PBDATA)      ;13T Get receive bit 7 in A.
;                               ;84T TO ECHO
        RLCA                    ; 4T Rotate into carry flag AND BIT 0
        RR      C               ; 8T then into C.
        AND     1               ; 7T Mask other bits AND SET FLAGS FOR TEST
        LD      A,1             ; 7T Prepare A to set or reset line.
;                               ;FLAGS UNALTERED FROM THE AND ABOVE
        NOP                     ; 8 MORE TO ADD
        NOP
;50 T-STATES TO GO .LESS 13T TO DO IT AND 10T FOR THE JUMP LEAVES 27
        CALL    D27T            ; 27T SO DELAY IT
        JP      Z,GTCLRB        ; 10T TEST FLAG AND GOTO APPROPIATE ECHO
;
;*******************************************************************************
;
;       ECHO A ONE              NOTE THAT BOTH INPUT AND OUTPUT ARE INVERTED
;                               SO THAT THIS IS REALLY A ZERO
;
GTSETB: LD      (PBSETB),A      ; 13T SET OUTPUT HIGH
        JP      GTREST          ; 10T AND CONTINUE,KEEPING TIMING THE SAME
;
;       23 T-STATES FOR THIS BRANCH
;*******************************************************************************
;
;       ECHO A ZERO
;
GTCLRB: LD      (PBCLRB),A      ; 13T SET OUTPUT LOW
        JP      GTREST          ; 10T and keep timing constant.
;
;*******************************************************************************
;
GTREST: NOP                     ; 4T--DELAY
        DJNZ    XMIT            ;13T Go till all bits in.
```

```
;
;TIMING:SINCE LAST READ=4+8+7+7+8+27+10+13+10+4+13=111
;       SINCE LAST WRITE=10+4+13=27
;       NOTE TIMING PRESERVED THROUGH XMIT LOOP(SEE VALUES AT START OF LOOP)
;
                                ; 8T WHEN WE FALL THROUGH
;TIMING:SINCE LAST READ=111-13+8=106
;       SINCE LAST WRITE=27-13+8=22
;
;*******************************************************************************
;
        DELAY   7
        CALL    D47T
        NOP
        NOP
;
;*******************************************************************************
;TIMING=755+106=861 SINCE LAST READ(+13 FOR READ=874)
;TIMING=755+22=777 SINCE LAST WRITE
;*******************************************************************************
;
        LD      A,(PBDATA)      ;13T Get receive bit 7 in A.
        RLA                     ; 4T TO CARRY
        JP      NC,FRAMERR      ;10T FRAMING ERROR IF NOT A STOP BIT
;
;*******************************************************************************
;
;TIMING=777+27=804 SINCE LAST WRITE
;
;*******************************************************************************
;
        CALL    D50T            ;50T
        LD      A,1             ; 7T Prepare A as stop bit.
        LD      (PBSETB),A      ;13T Echo stop bit AT 874T
;
;*******************************************************************************
        DELAY   9               ;TIMING NO LONGER CRITICAL-IT JUST MUST BE
                                ;LONG ENOUGH
        LD      A,C             ; Transfer assembled byte to A
;*******************************************************************************
;
        CP      1               ; IS ACC = 1?
        JP      Z,MONIT##       ; GO TO MONITOR IF SO
        CP      CR              ; CARRIAGE RETURN?
        JP      NZ,FRAMERR      ; NOT VALID IF NOT
        JP      RESTORE         ; and return to caller.
;*******************************************************************************
;*******************************************************************************
;DELAY SUBROUTINES
;
;*******************************************************************************
;
DELAYR:CALL     D87T            ;17T--THIS CALL INCLUDED IN 87T TIME
        DJNZ    DELAYR          ;13T--THIS+87=100 PER LOOP
        POP     BC              ;10T--WASTE TIME.NOTE THAT WE HAVE 8 WHEN WE
```

```
;                               FALL THROUGH
        PUSH    BC              ;11T--RESTORE
        NOP                     ;4T
        NOP                     ;A TOTAL OF 8+21+16+10=55 TSTATES
        NOP
        NOP
        RET                     ;10T--RETURN.SEE MACROS FOR MORE DOCUMENTATION
;
;*******************************************************************************
;
;       50 TSTATE DELAY
;CALL TO COME HERE=17 TSTATES
;
D50T:   EX      AF,AF'          ; 4T--SWITCH REGS SO AS TO SAVE FLAGS
        OR      0FFH            ; 7T--THIS ONLY SCREWS F'
        EX      AF,AF'          ; 4T--RESTORE FLAGS
        NOP                     ; 4T--WASTE TIME
        NOP                     ; 4T--4+4+4+4+7+10+CALL TO COME(17)=50
        RET                     ;10T
;
;*******************************************************************************
;
;       87 TSTATE DELAY
;
;CALL TO COME HERE=17 T
;
D87T:   JP      D77T            ;10T--WASTE TIME
D77T:   JP      D67T            ;10T--HERE DOWN=77 INCLUDING THE CALL IN
D67T:   JP      D57T            ;10T--AND SO ON
D57T:   JP      D47T
D47T:   JP      D37T
D37T:   JP      D27T
D27T:   RET                     ;10T--NOTE THAT THIS FITS WELL WITH DJNZ WHICH
;                                IS 13T
;
;*******************************************************************************
;
;       100 TSTATE DELAY
;
;CALL TO COME HERE=17 T
;
D100T:  CALL    D50T            ;THIS CALL IS INCLUDED IN 50T'S TIMING
        EX      AF,AF'          ;SAME TRICKS AS IN 50T
        OR      0FFH
        EX      AF,AF'
        NOP
        NOP
        RET
;
;*******************************************************************************
EBLOCK:
;
;*******************************************************************************
;       RESTORE ALL REGISTERS
```

```
RESTORE:        LD      SP,PHLPRIME     ;SET STACK TO POINT TO RIGHT REGS
                POPALL                  ;FIRST PRIMES
                EX      AF,AF'          ;NOW GO TO REGULAR SET
                EXX
                POP     IY              ;THIS IS THE REVERSE ORDER OF ENTRY
                POP     IX
                POPALL
                LD      SP,(STPTR)      ;RESTORE OLD STACK
                LD      HL,(PPC)        ;GET PC
                EX      (SP),HL         ;PUT PC ON STACK
                LD      HL,(PHL)        ;RESTORE HL
                RET                     ;RETURN
;*******************************************************************************
;       MESSAGES
                MSG     AF
                MSG     BC
                MSG     DE
                MSG     HL
                MSG     IX
                MSG     IY
                MSG     AFPRIME
                MSG     BCPRIME
                MSG     DEPRIME
                MSG     HLPRIME
                MSG     STPTR
                MSG     PC
PROMPT: DB      'RESTORE BREAKPOINT THEN HIT RETURN'
                DB      CR
                DB      LF
                DB      'TO CHANGE REGISTERS AND/OR PC SEE LISTING'
                DB      CR
                DB      LF
                DB      0
CNOT:   DB      '       BREAKPOINT ROUTINE'
                DB      CR
                DB      LF
                DB      'COPYRIGHT 1983 BY OGDEN H. HAMMOND'
                DB      CR
                DB      LF
                DB      0
                END
```

```
        .Z80
;
NAME ('PAGE1')
TITLE   ARMY CONTROL BOARD [REV C]      COMMENTED BY OHH 5/23/83
SUBTTL  PAGE ONE INTERRUPT AND RST VECTORS
;
;
INCLUDE ARMYREVC.EQU
ASEG
        ORG     0000H
;
*****************************************************************************
*   PROCESSOR STARTS HERE ON RESET. INTERRUPT AND REFRESH REGS ARE           *
*   CLEARED. ALL INTERRUPTS ARE DISABLED(HARDWARE DI INSTRUCTION). INTERRUPT  *
*   CONTROL REGISTER IS SET TO 01, WHICH ENABLES "NOT INTR" AND MASKS OFF     *
*   "NOT" RSTA, RSTB, RSTC. REMEMBER -- NO INTERRUPT CAN WORK UNLESS BOTH    *
*   AN EI INSTRUCTION HAS BEEN ISSUED AND THERE IS A ONE IN THE APPROPIATE    *
*   SPOT IN THE INTERRUPT MASK REGISTER, WHICH IS A WRITE ONLY REGISTER       *
*   ADDRESSED AS AN OUTPUT PORT(LOWER 4 BITS ONLY) BY AN OUT BBH OR EQUIV.    *
*   INSTRUCTION. THE FOLLOWING VALUES ENABLE THE CORRESPONDING INT. LINES:    *
*           08H             ENABLES RSTA                                      *
*           04H             ENABLES RSTB                                      *
*           02H             ENABLES RSTC                                      *
*           01H             ENABLES INTR                                      *
*   NOTE THAT 0FH ENABLES ALL INT. LINES                                      *
*   8080 INTERRUPT MODE IS AUTOMATICALLY SELECTED ON RESET                    *
*****************************************************************************
        RESTART 0       (11000111)!
START:  DI                      ; NOT NECESSARY
        JP      MAIN##          ;
;
;       END OF RESET CODE
*****************************************************************************
THESE ARE THE 8 NORMAL Z80 RESTART LOCATIONS. IN 8080 MODE , THESE
CONSTITUTE A MEANS FOR EXTERNAL DEVICES TO FORCE A CALL TO ONE OF 8 LOCATIONS
WITH ONE INSTRUCTION PUT ON THE BUS DURING AN INTACK CYCLE. ALSO CAN BE USED
AS A VERY SHORT CALL VIA RST INSTRUCTION.
;NOTE THAT ANY INTERRUPT AUTOMATICALLY EXECUTES A DI INSTRUCTION
;SEE NATIONAL NSC 800 PAGE 4-17. (ALSO STANDARD ON 8080 AND Z80)
*****************************************************************************
!
;       RESTART 1       (11001111)
;       THIS ROUTINE USED BY MONITOR PROGRAM TO DO BREAKPOINTS
        ORG     0008H
        JP      REST##          ; GOTO REST ROUTINE IN MONITOR
;
;*****************************************************************************
;NOTE:THIS SHOULD BE ALTERED SINCE THERE IS NO REASONABLE WAY
;WE CAN GET THE CONSOLE TO TRIGGER INTERRUPT 1. MONITOR SHOULD USE
;RST7 AND THE TERMINATOR SHOULD USE RSTB OR RSTC. (MONITOR REQUIRES
;BOTH AN INSTRUCTION AND A HARDWARE RESTART.)
;
;*****************************************************************************
;IN Z80 RST10 . USED BY GODLEY TO SAVE CODE(ALL OF TWO BYTES)
```

```
;
        RESTART 2       (11010111)
        ORG     0010H
        JP      PBYT##          ; SEND ACC TO CONSOLE
;
;IN Z80 RST18.USED BY GODLEY AS ABOVE
;
;       RESTART 3       (11011111)
        ORG     0018H           ; SEE GTEST DOCUMENTATION BELOW
        CALL    GBYT##          ; GET CONSOLE CHARACTER
        JP      GTEST           ; GOTO GTEST
;
;*******************************************************************************
;       RESTART 4       (11100111)
                                ;THESE ARE JUMPS TO A RESTART ERROR ROUTINE IN
                                ;THE MONITOR
        ORG     0020H
        JP      RSTER##         ;
;
;       RESTART 5       (11101111)
        ORG     0028H
        JP      RSTER##         ;
;
;       RESTART 6       (11110111)
        ORG     0030H
        JP      RSTER##         ;
;
;*******************************************************************************
;       RESTART 7       (11111111)
                                ;IN MODE 1,WILL COME HERE ON NOT INTR GOING LOW
        ORG     0038H
        JP      ETERM##         ; EARLY TERMINATION ROUTINE
;
;NOTE THAT THIS REQUIRES HARDWARE OR MODE1 INTERRUPTS!
;*******************************************************************************
;       NSC     800 SPECIAL INTERRUPT LOCATIONS
;       IF INT. ENABLED AND MASK OK WILL COME HERE AS SHOWN.
;       NO INSTRUCTION NEEDED ON BUS-- JUST PULL THE APPROPIATE PIN LOW.
;*******************************************************************************
;       RESTART C
        ORG     002CH
        RET
;
;       RESTART B
        ORG     0034H
        RET
;
;*******************************************************************************
;THESE TWO INTERRUPTS ARE NOT USED.RSBJMP AND RSCJMP ARE LOCATIONS IN
;RAM-NSCRAM+5AH AND+5DH RESPECTIVELY.THESE LOCATIONS ARE NOT CURRENTLY
;INITIALIZED.ACCORDINGLY,A DISASTER WILL HAPPEN IF RSTB OR RSTC GO LOW
;WHILE THE INTERRRUPT MASK ENABLES THEM.GRANTED THIS SHOULD NOT HAPPEN.
;THESE HAVE NOW BEEN MADE RETURNS FOR SAFETY.7/16/83
;*******************************************************************************
;*******************************************************************************
```

```
;THIS INTERRUPT IS GENERATED ON THE A/D BOARD.IT IS CONTROLLED BY A NUMBER
;OF CPU BOARD OUTPUTS.THE NSC 810 PORT C PIN 0 CONTROLS AN OR GATE WHICH
;DISABLES THE INTERUPT COMING FROM THE A/D BOARD ON S-100 PIN 4 IF C-0 IS
;HIGH.NOTE THAT C-0 IS NOT CURRENTLY INITIALIZED AND IS THUS RANDOM.
;FURTHERMORE, THE A/D BOARD WILL NOT GENERATE AN INTERRUPT IF ITS FIRST
;INTERRUPT WAS NOT ACKNOWLEDGED.THE A/D INTERRUPT OCCURS IMMEDIATELY AFTER
;CONVERSION COMPLETE FOLLOWING A TIME-OUT OF THE SOFTWARE SETTABLE COUNTER
;WHICH GIVES THE APPROXIMATE SAMPLE(WILL HAVE ABOUT 200 MICROSECONDS OF SLOP)
;TIME.
;*****************************************************************************
        ORG     003CH
AQUINT: PUSH    AF              ; SAVE FLAGS AND ACC.
        LD      BC,0817H        ; SET UP B&C REGS FOR BLOCK INPUT
;*****************************************************************************
;C REGISTER IS THE IO PORT ADDRESS:B REGISTER IS THE COUNTER
;SO FETCH DATA FROM IO 18 THROUGH 25--8 BYTES TOTAL
;*****************************************************************************
AQLP:   INC     C               ; IO PORT=IO PORT + 1
        INI                     ; BLOCK INPUT INSTR.
;
;*****************************************************************************
;FETCH FROM IO PORT (C),STORE AT HL,DECR. B, INCR. HL .NOTE: B IS PUT ON
;THE HIGH ORDER ADDRESS LINES,WHICH ARE GENERALLY NOT LOOKED AT DURING IO.
;*****************************************************************************
;
        JP      NZ,AQLP         ; DO UNTIL B=0 (8 TIMES)
        SET     7,H             ; RAM IS FROM 8000H TO FFFFH.
;
;*****************************************************************************
;TO MAINTAIN CIRCULAR 32K BUFFER,MAKE SURE HL ALWAYS HAS HIGH BIT SET.
;THUS FFFF INCREMENTS TO 0000,BUT THIS SETS IT BACK TO 8000H.
;*****************************************************************************
;
        DEC     DE              ; DE IS USED AS A MULTI-PURPOSE COUNTER IN
                                ; AQUIRE.A RAM LOCATION SHOULD BE USED INSTEAD.
;                               ; DE COUNTS THE NO. OF TIMES
                                ; THAT PROCESSOR HAS BEEN INTERRUPTED.
        POP     AF              ; RESTORE FLAGS&ACC
        EI                      ; ENABLE INTERUPTS
        RET                     ; RETURN
;
;*****************************************************************************
;THIS ROUTINE APPARENTLY JUMPS TO MONITOR IF A CTL A IS TYPED
;SEQUENCE IS INITIATED BY A RST3,FOLLOWED BY A CALL TO GBYT,WHICH
;RETURNS WITH THE CONSOLE CHARACTER(ASCII) IN ACC.
;*****************************************************************************
;
GTEST:  PUSH    AF              ; SAVE FLAGS&ACC
        CP      1               ; IS ACC = 1?
        JP      Z,MONIT##       ; GO TO MONITOR IF SO
        POP     AF              ; RESTORE FLAGS&ACC
        RET                     ; RETURN
;

        END
```

value	part #	qty.	notes
Capacitors			
100 pf	C1, C4, C20, C33	4	25v ceramic
0.01 uf	C56-C59	4	Elpac 1% poly.
0.1 uf	C29	1	25v ceramic
0.22 uf	C3, C5-C15, C19, C21-C28, C30, C32, C34-C44, C46-C55	44	25v ceramic
2.2 uf	C2, C18, C31, C45	4	" "
100.0 uf	C16, C17	2	10v tantalum
Resistors			
332 ohm	R18, R19	2	1/4watt 5%
1k ohm	R101, R201, R301 R401	4	" "
3.32k ohm	R17, R20	2	" "
4.42k ohm	R16, R21	2	" "
10k ohm	R104, R204, R304 R404	4	" "
100k ohm	R12-R15, R22, R23, R105-R108, R205-R208, R305-R308, R405-R408,	22	" "
200k ohm	R102-402, R109-409	8	" "
Transistors			
2N2907	Q1	1	GE or equiv.
Diodes			
1N914	D1	1	GE or equiv.

Figure 1.--Analog to digital converter board parts list.

Integrated Circuits

DP21EP	U104,U105,U204 U205,U304,U305, U404,U405	8	IC- PMI
74C00	U110,U210,U320 U410	4	Integrated
74C04	U19	1	Circuits
74C08	U21,U109,U209 U309,U409	5	National Semiconductor
74C10	U16	1	or
74C85	U14	1	Equivalent
74C138	U12	1	
74C374	U13,U102,U103,U202, U203,U302,U303, U402,U403	9	
ADC1210HCD	U101,U201,U301, U401	4	National
HI-7541JN	U106,U206,U306 U406	4	Analog Devices
CD4017BE	U17	1	RCA or equiv.
CD4028	U107,U207,U307 U407	4	
CD4029*	U108,U208,U308 U408	4	
CD4040	U22	1	
CD4043	U18	1	
CD4063	U14	1	
CD4536	U15	1	
L161CJ	U111,U211,U311 U411	4	Siliconix

Switches

76RSB04S	SW1	1	Grayhill

Connectors

2x13 .1"	J1	1	Amp
rt. angle 5pin .1"	J2,J3	2	"

Sockets

8 pin	9	Augat
14 pin	13	"
16 pin	18	"
18 pin	4	"
20 pin	9	"
24 pin	4	"

*Must not be TOSHIBA;
Use FAIRCHILD only.

Figure 1, cont'd.

value	part #	qty.	notes
Capacitors			
0.00627 uf	C8,C17,C26,C35,	4	Elpac
0.0178 uf	C6,C15,C24,C33	4	1% polycarbonate
0.0267 uf	C4,C13,C22.C31	4	"
0.0315 uf	C2,C11,C20,C29	4	"
0.0328 uf	C1,C10,C19,C28	4	"
0.0387 uf	C3,C12,C21,C32	4	"
0.0581 uf	C5,C14,C23,C32	4	"
0.1690 uf	C7,C16,C25.C34	4	"
470 pf	C54,C56	2	ceramic
0.0015 uf	C53	1	"
0.00457 uf	C52	1	"
0.018 uf	C51	1	"
0.1 uf	C65	1	"
0.22 uf	C66-C90	25	"
2.2 uf	C9,C18,C27,C50, C55,C57	6	ceramic
39.0 uf	C60	1	10v tantalum
100.0 uf	C58,C59,C92,C93	4	10v tantalum
on header CN8:			
0.22 uf	C64	1	ceramic
0.47 uf	C61	1	"
1.0 uf	C62	1	"
2.2 uf	C63	1	"
Resistors			
1k ohm	R1,R12,R33,R34,R85	5	1/4watt 1%
4.7k	R108	1	"
5.11k	R87	1	"
10k	R84	1	"
22.1k	R83	1	"
33.2k	R64	1	"
47.5k	R70,R72,R75	3	"
56.2k	R81,R94	2	"
82.5k	R77,R80,R90,R93, R95	5	"
100k	R62,R63,R65-R67 R74,R86,R104, R105,R109-R112	13	"

Figure 2.--Analog board parts list

165k	R76, R74, R79, R82 R89, R91, R92, R95	8	1/4 watt, 5%
200k	R107	1	"
2m	R71, R73	2	"
6.6m	R88	1	"
or header CY8:			
5.49k	R102	1	"
11.3k	R101	1	"
26.7k	R100	1	"
10-1-104	RP1	1	Bourns resistor net.
16-1-154*	CM1-CM4	4	"
????????*	CM5	1	set to specific frequency

Transistors

2N2222	Q1	1	GE or equiv.

Diodes

1N914	D3-D6	4	Ge or equiv.

Integrated Circuits

AD7110	U22	1	Analog Devices
OP421HY	U1-U5, U13, U15, U19	8	PMI
L161CJ	U12	1	Siliconix
MM74C04N	U9	1	National
MM74C10	U8	1	Semiconductor
MM74C32	U10	1	"
MM74C74	U24	1	"
MM74C85	U11	1	"
MM74C244	U20	1	"
MM74C373	U14	1	"
CD4029BCN	U25	1	"
MC14521BCP	U22	1	Motorola

Switches

TP11	SW1	1	C&K
76RSB04S	SW2	1	Grayhill
76RSB08S	SW3	1	"

Connectors

* USGS only

Figure 2, cont'd.

2x13 .1"	J6,J7	2	Amp
2x3 .1"	J3,J5	2	"
1x3 .1"	J4,J8	2	"
5 pin rt.ang.	J1,J2	2	"

Sockets
---

8 pin	1	Augat
14 pin	19	"
16 pin	16	"
20 pin	2	"

Figure 2, cont'd.

value	part #	qty.	notes
Capacitors			
0.22 uf	C4-C31	28	ceramic 25v
39.0 uf	C1	1	10v tantalum
100.0 uf	C2,C3	2	10v tantalum
Resistors			
1k ohm	R5	1	1/4 watt 5%
10k ohm	R1,R6,R7,R12,R13	5	" " "
27.4k ohm	R8,R9	2	" " "
50k ohm	R10	1	" " "
68.1k ohm	R11	1	" " "
10-1-104	RP1	1	Bournes resistor net.
Semiconductors			
74C00	U20	1	Integrated Circuits
74C02	U12	1	
74C08	U13	1	
74C30	U20	1	National Semiconductor or ecuivalent
74C32	U22,U27	2	
74C74*	U26	1	
74C86*	U25	1	
74C138	U23,U24	2	
74C240	U21	1	
74C244	U18,U19	2	
74C373	U17	1	
5817A	U10	1	
NSC800N	U7	1	National
NSC810N	U8	1	National
CD4040	U14	1	RCA or ecuiv.
HM5-6564	U2-U6	5	Harris
MBM27C64-30	U16	1	Fujitsu
OP220HZ	U15	1	PMI
1N914	D1-D4	4	diode GE
Sockets			
8 pin		1	Augat
14 pin		8	"
16 pin		4	"
20 pin		4	"
28 pin		1	"
40 pin		2	"
20 pin sio		10	"

* not needed for some EPROMs. use 14 pin dip header with pins 2&3, 5&6 tied together in place of U25.

Figure 3.--Control (cpu) board parts list

value/ part #	board #	qty.	notes
Capacitors			
0.1 uf	C7	1	ceramic 50v
0.22 uf	C8,C9	2	ceramic 50v
2.2 uf	C3,C4	2	ceramic 50v
10.0 uf	C1,C2,C6	3	10v tantalum
100.0 uf	C5	1	10v tantalum
Resistors			
120 ohm*	R3	1	1/4 watt, 5%
360 ohm	R4	1	" " "
3.3k ohm	R1,R2	2	" " "
4.7k ohm	R5,R6	2	" " "
Semiconductors			
LM338K steel	Q1	1	12v adj. reg.
UA7805	Q2	1	+5volt reg.
UA7905	Q3	1	-5volt reg.
2N2222	Q4,Q5	2	transistor
1N4001	CR1-CR12	12	diode 1a 50v
74C10**	U1	1	3-inout nand gate
74C30	U2	1	8-inout nand gate
CD4724	U3	1	8-bit latch
Fuses			
2A Micro	F2-F5	5	Littlefuse
3A Micro	F1	1	"
Relay			
S4E-12V	K1,K2	2	Aromat

Figure 4.--Power interface board parts list

Oscillator

251-5907	Y1	1	Vectron

Switches

TP11	SW1	1	C&K
T203HMCBE	SW2	1	"

Hardware

65823-073	P1	1	Berg connector
26 pin .01c	P2, P3	2	Bendix "
13 pin .01	P4	1	" "
14 pin		2	IC sockets
16 pin		1	"
856		2	Keystone klips
281-007		5	Littlefuse fuse holder
401-T03		1	Ahamtor heat sink
BC9B		2	GE battery
6-32x1/2" Binder hd		2	screw
4-40x3/8" Binder hd		6	screw
#6 lock		2	washer
6-32 hex		2	nut
4-40 hex		6	nut

* R3 adjusted for output of Q1 = 5.2volts
** or equivalent

Figure 4, Cont'd.

part #	board #	qty	notes
Capacitors			
37,000 uf	C1,C2	2	50v electrolytic
Resistors			
VC5E-10	R5	1	Clarostat
RN60D-1001F	R3,R4	2	TRW
RN60D-1002F	R2	1	"
RN60D-1003F	R1,R6	2	"
Diodes			
1N4001	D1,D4	2	GE or equiv.
3AF4	D2,D3,D5	3	ST. Semicon
Transistors			
MPSA14	Q1	1	GE or equiv.
2N3716	Q3	1	"
2N5194	Q2	1	"
Connectors			
09-50-3041	C1	1	Molex
09-50-3061	C2	1	"
09-50-3081	C3	1	"

Figure 5
Figure 5 --OBIP Release firing board parts list
Parts List

National	RCA	Motorola	Fairchild	Toshiba	Type	Power	CPU	A-D	Anal	FM
MM74C00N	---	MC74HC00N	---	TC40HC000P	Quad 2-input Nand Gate	-	1	4	-	-
MM74C02N	---	MC74HC02N	---	TC40HC002P	Quad 2-input Nor Gate	-	1	-	-	-
MM74C04N	C4069UBE	MC14069UBCP	F4069UBPC	---	Hex Inverters	-	-	1	1	-
MM74C08N	---	---	---	TC48HC008P	Quad 2-input And Gate	-	1	5	-	-
MM74C10N	---	MC74HC10N	---	---	Triple 3-input Nand Gate	1	-	1	1	1
MM74C30N	---	---	---	---	8-input Nand Gate	1	1	-	-	-
MM74C32N	---	---	---	TC40HC032P	Quad 2-input Or Gate	-	2	-	1	-
MM74C74N	---	MC74HC74N	---	TC40HC074P	Dual D-type Flip Flop	-	-	-	1	-
MM74C86N	---	MC74HC86N	---	---	Quad 2-input Excl-Or Gate	-	-	-	-	-
MM74C85N	---	---	F40085BPC	---	4-Bit Magnitude Comparator	-	-	-	1	1
MM74HC138N	---	MC74HC138	---	TC40HC138P	1-of-8 Decoder/Demultiplexer	-	2	1	-	-
MM74HC240N	---	MC74HC240	---	TC40HC240P	3-state Octal Buffer	-	1	-	-	-
MM74HC244N	---	MC74HC244	---	---	3-state Octal Buffer	-	2	-	1	1
MM74HC373N	---	MC74HC373	---	TC40HC373P	3-state Octal D-type Latch	-	1	-	1	-
MM74HC374N	---	MC74HC374	---	---	3-state Octal D-type Latch	-	-	9	-	-
CD4001BCN	CD4001BE	MC14001BCP	F4001BPC	---	Quad 2-input Nor Gate	-	-	-	-	1
CD4017BCN	CD4017BE	MC14071BCP	F4017BPC	TC4017BP	5-stage Decade Counter	-	-	1	-	-
CD4028BCN	CD4028BE	MC14028BCP	F4028BPC	TC4028BP	1 of 10 Decoder	-	-	4	-	-
CD4029BCN	CD4029BE	MC14029BCP	F4029BPC	---	Up/Down Binary Counter	-	-	4	-	-
CD4040BCN	CD4040BE	MC14040BCP	F4040BPC	TC4040BP	12-stage Binary Counter	-	-	1	-	-
CD4043BCN	CD4043BE	MC14043BCP	F4043BPC	TC4043BP	3-state Quad R/S Latch	-	-	1	-	-
CD4046BCN	CD4046BE	MC14046BCP	F4046BPC	---	Micropower Phased Locked Loop	-	-	-	-	4
---	CD4063BE	---	---	TC4063BP	4-Bit Magnitude Comparator	-	-	1	-	-
CD4066BCN	CD4066BE	MC14066BCP	F4066BPC	TC4066BP	Quad Bilateral Switch	-	-	1	-	-
CD4093BCN	CD4093BE	MC14093BCP	F4093BPC	---	Quad Nand Schmitt Trigger	-	-	-	-	4
---	---	MC14521BCP	F4521BPC	---	24-stage Binary Counter	-	-	-	1	-
---	CD4536BE	MC14536BCP	---	---	Programmable Timer	-	-	1	-	-
CD47C24BCN	---	---	F4724BPC	---	8-Bit Addressable Latch	1	-	-	-	-
LM339AN	LM339N	LM339N	uA339	---	Operational Amplifier	-	-	-	-	1

Harris	Analog Dev.	Datel-Intersil								
HI7541KN	AD7541KN	DAC-HA12BC			12-Bit D-A Converter	-	-	4	-	-

Sole Source IC's

NSC800N	National Semiconductor				CMOS CPU	-	1	-	-	-
NSC810N	"	"			CMOS Ram-I/O-Timer	-	1	-	-	-
MM58174AN	"	"			Real Time Clock	-	1	-	-	-
ADC1210CD	"	"			12-Bit A-D Converter	-	-	4	-	-
HA7-2725-5	Harris Semiconductor				Programmable Op Amp	-	-	-	-	4
HM5-6564	"	"			8K x 8 Static Ram	-	5	-	-	-
AD7110KN	Analog Devices				Audio Attenuator	-	-	-	1	-
MBM27C64-30	Fujitsu Microelectronics				2K x 8 EPROM	-	1	-	-	-
OP21EP	Precision Monolithics				Precision Op Amp	-	-	8	-	-
OP220HZ	"	"			Dual Precision Op Amp	-	1	-	-	-
OP420HY	"	"			Quad Precision Op Amp	-	-	-	8	-
L161CJ	Siliconix				Quad Comparator	-	-	4	-	-

Figure 6:—IC Master Cross-reference Guide

Pin #	Function	Pin #	Function
1	+5V	51	+5V
2		52	
3		53	
4	VI0	54	-5v
5	CH. 1 IN	55	CH. 1 IN
6	CH. 1 IN	56	CH. 1 IN
7	CH. 1 IN	57	CH. 1 IN
8	CH. 2 IN	58	CH. 2 IN
9	CH. 2 IN	59	CH. 2 IN
10	CH. 2 IN	60	CH. 2 IN
11	CH. 3 IN	61	CH. 3 IN
12	CH. 3 IN	62	CH. 3 IN
13	CH. 3 IN	63	CH. 3 IN
14	CH. 4 IN	64	CH. 4 IN
15	CH. 4 IN	65	CH. 4 IN
16	CH. 4 IN	66	CH. 4 IN
17	GND	67	GND
18		68	
19		69	
20		70	
22		71	
21		72	
23		73	INT*
24		74	
25		75	RESET*
26		76	
27		77	oWR*
28		78	oDIN
29	A5	79	A0
30	A4	80	A1
31	A3	81	A2
32		82	A6
33		83	A7
34		84	
35	DO1	85	
36	DO0	86	
37		87	
38	DO4	88	DO2
39	DO5	89	DO3
40	DO6	90	DO7
41	DI2	91	DI4
42	DI3	92	DI5
43	DI7	93	DI6
44		94	DI1
45	sOUT	95	DI0
46	sINP	96	
47		97	
48		98	
49	CLOCK	99	
50	GND	100	GND

Figure 7: A-D Board S-100 Bus Connections

Pin #	Function	Pin #	Function
1	+5v	51	+5v
2		52	
3		53	
4		54	-5v
5	CH. 1 IN	55	CH. 1 OUT
6	CH. 1 IN	56	CH. 1 OUT
7	CH. 1 IN	57	CH. 1 OUT
8	CH. 2 IN	58	CH. 2 OUT
9	CH. 2 IN	59	CH. 2 OUT
10	CH. 2 IN	60	CH. 2 OUT
11	CH. 3 IN	61	CH. 3 OUT
12	CH. 3 IN	62	CH. 3 OUT
13	CH. 3 IN	63	CH. 3 OUT
14	CH. 4 IN	64	CH. 4 OUT
15	CH. 4 IN	65	CH. 4 OUT
16	CH. 4 IN	66	CH. 4 OUT
17	GND	67	GND
18		68	
19		69	
20		70	
21		71	
22		72	
23		73	INT*
24		74	
25		75	RESET*
26		76	
27		77	pWR*
28		78	pDIN
29	A5	79	A0
30	A4	80	
31	A3	81	
32		82	A6
33		83	A7
34		84	
35	DO1	85	
36	DO0	86	
37		87	
38	DO4	88	DO2
39	DO5	89	DO3
40	DO6	90	DO7
41	DI2	91	DI4
42	DI3	92	DI5
43	DI7	93	DI6
44		94	DI1
45	sOUT	95	DI0
46	sINP	96	
47		97	
48		98	
49	CLOCK	99	POC*
50	GND	100	GND

Figure 8.--Analog board S-100 bus connections

Pin #	Function	Pin #	Function
1	+5V	51	+5V
2	+12V	52	
3		53	
4	VI0	54	-5V
5		55	
6		56	
7		57	
8		58	
9		59	
10		60	
11		61	
12		62	
13		63	
14		64	
15		65	CART RESET
16		66	
17		67	
18		68	
19		69	EXT START (RFV)
20	GND	70	GND
21	-5V (NDEF)	71	
22		72	RDY
23		73	INT*
24		74	
25		75	RESET*
26		76	
27	RCV (RFV)	77	oWR*
28	XMIT (RFV)	78	oDBIN
29	A5	79	A0
30	A4	80	A1
31	A3	81	A2
32		82	A6
33		83	A7
34		84	
35	DO1	85	
36	DO0	85	
37		87	
38	DO4	88	DO2
39	DO5	89	DO3
40	DO6	90	DO7
41	DI2	91	DI4
42	DI3	92	DI5
43	DI7	93	DI6
44		94	DI1
45	sOUT	95	DI0
46	sINP	96	INTA
47		97	
48		98	
49	CLOCK	99	
50	GND	100	GND

Figure 9:--Control (cpu) board, S-100 bus connections

Pin #	Function	Pin #	Function
1	+5V	51	+5V
2	-5V	52	-12V
3		53	
4	+5V CART	54	-5V
5	+5V CART	55	CH1
6	+5V CART	56	CH1
7	+24V CART	57	CH1
8	+24V CART	58	CH2
9	-24V CART	59	CH2
10	-24V CART	60	CH2
11	RCV	61	CH3
12	XMIT	62	CH3
13	CNTRCLK	63	CH3
14	CNTRSET	64	CH4
15	EXT START	65	CH4
16	TERM	66	CH4
17	EXTRES	67	GND
18		68	
19		69	EXT START
20	GND	70	GND
21	-5V	71	
22		72	
23		73	INT*
24		74	
25		75	RESET*
26		76	
27	RCV	77	pWR*
28	XMIT	78	
29	A5	79	A0
30	A4	80	A1
31	A3	81	A2
32		82	A6
33		83	A7
34		84	
35	DO1	85	
36	DO0	86	
37		87	
38		88	
39		89	
40		90	DO7
41		91	
42		92	
43		93	
44		94	
45	sOUT	95	
46		96	
47		97	
48		98	
49	CLOCK	99	
50	GND	100	GND

Figure 10.--Power interface board, S-100 bus connections

ACCOPRESS®

25070	YELLOW
25071	BLACK
25072	LIGHT BLUE
25073	DARK BLUE
25074	LIGHT GRAY
25075	LIGHT GREEN
25076	DARK GREEN
25077	TANGERINE
25078	RED
25079	EXECUTIVE RED

ACCO INTERNATIONAL INC.
CHICAGO, ILLINOIS 60619