

```

1 ; ASSEMBLER ASIDE 1.14
2 ; _FDCMON
3 ;
4 ; FLOPPY DISK UTILITY PROGRAM
5 ; AUTHOR: ANDREW LYNCH
6 ; CODE RE-ORGANIZED BY DAN WERNER
7 ; CODE FOR THE ASSEMBLER ASIDE 1.14 BY ROLF HARRMANN
8 ; HR = ROLF HARRMANN
9 ;
10 ;
11 ; DATA CONSTANTS
12 ;
13 ; REGISTER IO PORT ; FUNCTION
14 FMSR: EQU 036H ; ADDRESS OF MAIN STATUS REGISTER
15 FDMA: EQU 037H ; FLOPPY DATA REGISTER
16 FLATCH: EQU 03AH ; FLOPPY CONFIGURATION LATCH
17 FDMA: EQU 03CH ; PSEUDO DMA ADDRESS
18
19 ; CODE BY HR
20 ; 74LS574
21 FLATCH0: EQU 0000001B ; TC HIGH D0
22 FLATCH1: EQU 0000010B ; /MOTOR LOW D1
23 FLATCH2: EQU 0000100B ; MINI HIGH D2
24 FLATCH3: EQU 0001000B ; SETPS2 D3
25 FLATCH4: EQU 0010000B ; SETPS1 D4
26 FLATCH5: EQU 0100000B ; SETPS0 D5
27 FLATCH6: EQU 0100000B ; /REDWC D6
28 FLATCH7: EQU 1000000B ; /DISKCHG_L D7
29 ; *****
30 ; CODE BY HR
31 MOTOR_ON_OFF: EQU 1
32 ; *****
33
34
35 ;
36 ; FDC CONFIGURATION LATCH OUTPUT BIT PATTERNS
37 MOTOR: EQU %00000000 ; BIT PATTERN IN LATCH FOR MOTOR CONTROL (ON)
38
39 TERMEN: EQU %00000001 ; BIT PATTERN IN LATCH TO WRITE A TC STROBE
40
41 RESETL: EQU %00000010 ; BIT PATTERN IN LATCH TO RESET ALL BITS
42 MINI: EQU %00000100 ; BIT PATTERN IN LATCH TO SET MINI MODE FDC9229 LOW DENS=1, HIGH DEN
43 S=0
44 PRECOMP: EQU %00100000 ; BIT PATTERN IN LATCH TO SET WRITE PRECOMP 125 NS:
45 FDDENSITY: EQU %01000000 ; BIT PATTERN IN LATCH TO FLOPPY LOW DENSITY (HIGH IS 0)
46 FDREADY: EQU %10000000 ; BIT PATTERN IN LATCH TO FLOPPY READY (P-34):
47 ;
48 ; CP/M STRING RELATED CONSTANTS
49 CR: EQU 0Dh ; CARRIAGE RETURN CHARACTER
50 LF: EQU 0Ah ; LINE FEED CHARACTER
51 ESC: EQU 1Bh
52 ; CODE BY HR
53 ; ENDS instead of END because the Assembler does not allow END
54 ENDS: EQU '$' ; LINE TERMINATOR FOR CP/M STRINGS
55 ; *****
56 BS: EQU 08H ; ASCII backspace character
57 ;
58 ;
59 ; MAIN PROGRAM BEGINS HERE
60 ;
61 ORG 0100h
62 ;
63 CALL SETUPDRIVE ; SETUP DRIVE PARAMETERS
64 CALL OUTFLATCH ; OUTPUT TO CONTROLLER
65
66 MAIN_LOOP:
67 LD DE,MSG_START ;
68 LD C,09H ; CP/M WRITE START STRING TO CONSOLE CALL
69 CALL 0005H ;
70 LD C,01H ; CP/M console input call
71 CALL 0005H ; GET K.B. DATA
72 CP '1' ; MOTOR ON
73 JP Z,MENU_MOTOR_ON ;
74 CP '2' ; MOTOR OFF
75 JP Z,MENU_MOTOR_OFF ;
76 CP '3' ; SET TRACK
77 JP Z,MENU_SET_TRACK ;

```

```

78 ; CODE BY HR
79 ;*****
80 CP      '4'          ; FORMAT TRACK
81 JP      Z,YES_NO_FORMAT ; SURE YES OR NO
82 ;*****
83 CP      '5'          ; READ SECTOR
84 JP      Z,MENU_READ   ;
85 ; CODE BY HR
86 ;*****
87 CP      '6'          ; WRITE SECTOR
88 JP      Z,YES_NO_WRITE ; SURE YES OR NO
89 ;*****
90 CP      '7'          ; DUMP BUFFER
91 JP      Z,MENU_DUMP   ;
92 CP      '8'          ; SENSE INT
93 JP      Z,MENU_SENSE  ;
94 CP      '9'          ; SENSE INT
95 JP      Z,MENU_CLEAR  ;
96 CP      'Q'          ; IS QUIT
97 JP      Z,MENU_QUIT   ;
98 CP      'q'          ; IS QUIT
99 JP      Z,MENU_QUIT   ;
100 JP     MAIN_LOOP     ; LOOP TO MENU
101
102 ; CODE BY HR
103 ;*****
104 YES_NO_FORMAT:
105 LD      DE,MSG_YES_NO ;
106 LD      C,09H          ; CP/M WRITE START STRING TO CONSOLE CALL
107 CALL    0005H          ;
108 LD      C,01H          ; CP/M console input call
109 CALL    0005H          ; GET K.B. DATA
110 CP      'Y'            ;
111 JP      Z,MENU_FORMAT  ;
112 CP      'y'            ;
113 JP      Z,MENU_FORMAT  ;
114 CP      'Q'            ; IS QUIT
115 JP      Z,MENU_QUIT    ;
116 CP      'q'            ; IS QUIT
117 JP      Z,MENU_QUIT    ;
118 JP     MAIN_LOOP      ; LOOP TO MENU
119 ;*****
120 ; CODE BY HR
121 ;*****
122 YES_NO_WRITE:
123 LD      DE,MSG_YES_NO ;
124 LD      C,09H          ; CP/M WRITE START STRING TO CONSOLE CALL
125 CALL    0005H          ;
126 LD      C,01H          ; CP/M console input call
127 CALL    0005H          ; GET K.B. DATA
128 CP      'Y'            ;
129 JP      Z,MENU_WRITE   ;
130 CP      'y'            ;
131 JP      Z,MENU_WRITE   ;
132 CP      'Q'            ; IS QUIT
133 JP      Z,MENU_QUIT    ;
134 CP      'q'            ; IS QUIT
135 JP      Z,MENU_QUIT    ;
136 JP     MAIN_LOOP      ; LOOP TO MENU
137 ;*****
138
139 ; CODE BY HR
140 ;*****
141 TT:
142 PUSH    AF             ; Store AF
143 LD      A,"T"          ; LOAD A "T"
144 CALL    COUT           ; SCREEN IT
145 POP     AF             ; RESTORE AF
146 RET                     ; DONE
147 ;*****
148 ; CODE BY HR
149 ;*****
150 SS:
151 PUSH    AF             ; Store AF
152 LD      A,"S"          ; LOAD A "S"
153 CALL    COUT           ; SCREEN IT
154 POP     AF             ; RESTORE AF
155 RET                     ; DONE
156 ;*****
157
158 ; CODE BY HR
159 MOTOR_ON:
160 LD      HL,FLATCH_STORE ; POINT TO FLATCH

```

```

161      RES      1,(HL)          ; SET MOTOR ON
162      LD       A,(FLATCH_STORE) ; SET A TO SETTINGS
163      OUT      (FLATCH),A       ; OUTPUT TO CONTROLLER
164      RET
165 ; *****
166 ; CODE BY HR
167 MOTOR_OFF:
168      LD       HL,FLATCH_STORE  ; POINT TO FLATCH
169      SET      1,(HL)          ; SET MOTOR OFF
170      LD       A,(FLATCH_STORE) ; SET A TO SETTINGS
171      OUT      (FLATCH),A       ; OUTPUT TO CONTROLLER
172      RET
173 ; *****
174
175 ;
176 ;
177 ; MENU OPERATIONS
178 ;
179 MENU_MOTOR_ON:
180      LD       HL,FLATCH_STORE  ; POINT TO FLATCH
181      RES      1,(HL)          ; SET MOTOR ON
182      CALL     OUTFLATCH        ; OUTPUT TO CONTROLLER
183      JP       MAIN_LOOP       ; LOOP TO MENU
184
185 MENU_MOTOR_OFF:
186      LD       HL,FLATCH_STORE  ; POINT TO FLATCH
187      SET      1,(HL)          ; SET MOTOR OFF
188      CALL     OUTFLATCH        ; OUTPUT TO CONTROLLER
189      JP       MAIN_LOOP       ; LOOP TO MENU
190
191 MENU_SET_TRACK:
192      LD       DE,MSG_ENTER_TRACK_NUMBER
193      LD       C,09H           ; CP/M WRITE STRING TO CONSOLE CALL
194      CALL     0005H           ;
195      ;
196      LD       HL,KEYBUF        ; SET POINTER TO KEYBUF AREA
197      CALL     GETLN           ; GET A LINE OF INPUT FROM THE USER
198      LD       HL,KEYBUF        ; SET POINTER TO KEYBUF AREA
199      CALL     HEXIN           ; GET TRACK
200      LD       (TRACK),A        ; STORE TRACK
201      CALL     SETTRACK        ; DO IT
202      JP       MAIN_LOOP       ; LOOP TO MENU
203
204
205 MENU_FORMAT:
206      LD       DE,MSG_ENTER_TRACK_NUMBER
207      LD       C,09H           ; CP/M WRITE STRING TO CONSOLE CALL
208      CALL     0005H           ;
209      ;
210      LD       HL,KEYBUF        ; SET POINTER TO KEYBUF AREA
211      CALL     GETLN           ; GET A LINE OF INPUT FROM THE USER
212      LD       HL,KEYBUF        ; SET POINTER TO KEYBUF AREA
213      CALL     HEXIN           ; GET TRACK
214      LD       (TRACK),A        ; STORE TRACK
215      CALL     FORMAT          ; FORMAT A FLOPPY DISK TRACK
216      JP       MAIN_LOOP       ; LOOP TO MENU
217
218 MENU_READ:
219      LD       DE,MSG_ENTER_TRACK_NUMBER
220      LD       C,09H           ; CP/M WRITE STRING TO CONSOLE CALL
221      CALL     0005H           ;
222      ;
223      LD       HL,KEYBUF        ; SET POINTER TO KEYBUF AREA
224      CALL     GETLN           ; GET A LINE OF INPUT FROM THE USER
225      LD       HL,KEYBUF        ; SET POINTER TO KEYBUF AREA
226      CALL     HEXIN           ; GET TRACK
227      LD       (TRACK),A        ; STORE TRACK
228      LD       DE,MSG_ENTER_SECTOR_NUMBER
229      LD       C,09H           ; CP/M WRITE STRING TO CONSOLE CALL
230      CALL     0005H           ;
231      ;
232      LD       HL,KEYBUF        ; SET POINTER TO KEYBUF AREA
233      CALL     GETLN           ; GET A LINE OF INPUT FROM THE USER
234      LD       HL,KEYBUF        ; SET POINTER TO KEYBUF AREA
235      CALL     HEXIN           ; GET SECTOR
236      LD       (SECTOR),A       ; STORE SECTOR
237      ;
238      CALL     READ            ; READ A FLOPPY DISK SECTOR
239      CP       0FFH
240      JP       NZ,DOREADEXIT    ; NO ERROR IF READ RETURNS OTHER THAN -1
241      ; ELSE, PRINT ERROR MESSAGE

```

```

242                                     ;
243 ;*****
244         LD      DE,MSG_ERROR          ;
245         LD      C,09H                 ; CP/M WRITE ERROR STRING TO CONSOLE CALL
246         CALL    0005H                 ;
247 DOREADEXIT:
248         JP      MAIN_LOOP            ;
249
250 MENU_WRITE:
251         LD      DE,MSG_ENTER_TRACK_NUMBER
252         LD      C,09H                 ; CP/M WRITE STRING TO CONSOLE CALL
253         CALL    0005H                 ;
254                                     ;
255         LD      HL,KEYBUF              ; SET POINTER TO KEYBUF AREA
256         CALL    GETLN                 ; GET A LINE OF INPUT FROM THE USER
257         LD      HL,KEYBUF              ; SET POINTER TO KEYBUF AREA
258         CALL    HEXIN                 ; GET TRACK
259         LD      (TRACK),A              ; STORE TRACK
260         LD      DE,MSG_ENTER_SECTOR_NUMBER
261         LD      C,09H                 ; CP/M WRITE STRING TO CONSOLE CALL
262         CALL    0005H                 ;
263                                     ;
264         LD      HL,KEYBUF              ; SET POINTER TO KEYBUF AREA
265         CALL    GETLN                 ; GET A LINE OF INPUT FROM THE USER
266         LD      HL,KEYBUF              ; SET POINTER TO KEYBUF AREA
267         CALL    HEXIN                 ; GET SECTOR
268         LD      (SECTOR),A             ; STORE SECTOR
269                                     ;
270         CALL    WRITE                 ; WRITE A FLOPPY DISK SECTOR
271         CP      0FFH
272         JP      NZ,DOWRITEEXIT        ; NO ERROR IF WRITE RETURNS OTHER THAN -1
273                                     ; ELSE, PRINT ERROR MESSAGE
274                                     ;
275 ;*****
276         LD      DE,MSG_ERROR          ;
277         LD      C,09H                 ; CP/M WRITE ERROR STRING TO CONSOLE CALL
278         CALL    0005H                 ;
279 DOWRITEEXIT:
280         JP      MAIN_LOOP            ;
281
282 MENU_SENSE:
283         CALL    CHECKINT              ;
284         JP      MAIN_LOOP            ;
285
286 MENU_DUMP:
287         LD      HL,SECTOR_BUFFER      ; SET ADDRESS TO DUMP
288         LD      DE,SECTOR_BUFFER      ; SET END ADDRESS
289         INC     D                      ;
290         INC     D                      ;
291         CALL    DUMP_BUFFER           ; DUMP BUFFER TO CONSOLE
292         JP      MAIN_LOOP            ;
293
294 MENU_CLEAR:
295         LD      HL,SECTOR_BUFFER      ; SET ADDRESS TO DUMP
296
297         LD      C,00H
298 CLEAR_LOOP:
299         LD      A,(D1)
300         LD      (HL),A
301         DEC     C
302         INC     HL
303         JP      NZ,CLEAR_LOOP
304         LD      C,00H
305 CLEAR_LOOP1:
306         LD      A,(D1)
307         LD      (HL),A
308         DEC     C
309         INC     HL
310         JP      NZ,CLEAR_LOOP1
311         JP      MAIN_LOOP
312
313
314 MENU_QUIT:
315         LD      DE,MSG_END            ;
316         LD      C,09H                 ; CP/M WRITE END STRING TO CONSOLE CALL
317         CALL    0005H                 ;
318                                     ;
319         LD      C,00H                 ; CP/M SYSTEM RESET CALL
320         CALL    0005H                 ; RETURN TO PROMPT
321
322 ;

```

```

323 ;
324 ; MAIN PROGRAM ENDS HERE
325 ;
326
327 ; __SETUPDRIVE__
328 ;
329 ;     SETUP DRIVE SETTINGS
330 ;
331 ;
332 ;
333 ;
334 SETUPDRIVE:
335     LD     A,RESETL           ; RESET SETTINGS
336     OR     MINI               ; SELECT MINI FLOPPY (low dens=1, high dens=0)
337     OR     PRECOMP            ; SELECT PRECOMP
338     OR     FDDENSITY          ; SELECT DENSITY
339     OR     FDREADY            ; SELECT READY SIGNAL
340     LD     (FLATCH_STORE),A   ; SAVE SETTINGS
341     LD     A,01H              ;
342     LD     (UNIT),A           ; SET UNIT 1
343     LD     A,2                ; DENSITY
344     LD     (DENS),A           ;
345     LD     A,09               ;
346     LD     (EOTSEC),A        ; LAST SECTOR OF TRACK
347     LD     A,7FH              ;
348     LD     (SRTHUT),A        ; STEP RATE AND HEAD UNLOAD TIME
349     LD     A,05H              ;
350     LD     (HLT),A           ; HEAD LOAD TIME
351     LD     A,0E5H             ;
352     LD     (D1),A            ; FILLER BYTE FOR FORMAT
353     LD     A,2AH              ;
354     LD     (FMTGAP),A        ; GAP FOR FORMAT
355     LD     A,0DH              ;
356     LD     (GAP),A           ; GAP
357     LD     A,80H              ;
358     LD     (SECSIZ),A        ; SECTOR SIZE /4
359     RET
360
361
362
363
364 ; __OUTFLATCH__
365 ;
366 ;     SEND SETTINGS TO FLOPPY CONTROLLER
367 ;
368 ;
369 OUTFLATCH:
370     LD     A,(FLATCH_STORE)   ; SET A TO SETTINGS
371     OUT    (FLATCH),A         ; OUTPUT TO CONTROLLER
372     RET
373
374
375 ; __READ__
376 ;
377 ;     READ A SECTOR
378 ;
379 ;
380 READ:
381     LD     A,46H              ; BIT 6 SETS MFM, 06H IS READ COMMAND
382     LD     (CMD),A
383     JP     DSKOP
384
385 ; __WRITE__
386 ;
387 ;     WRITE A SECTOR
388 ;
389 ;
390 WRITE:
391     LD     A,45H              ; BIT 6 SETS MFM, 05H IS WRITE COMMAND
392     LD     (CMD),A
393     JP     DSKOP
394
395

```

```

396
397 ; _FORMAT
398 ;
399 ;     FORMAT A TRACK
400 ;
401 ;
402 FORMAT:
403     LD     A,4DH                ; BIT 6 SETS MFM, 0DH IS FORMAT COMMAND
404     LD     (CMD),A
405     JP     DSKOP
406
407 ; _DSKOP
408 ;
409 ;     PERFORM A DISK OPERATION
410 ;
411 ;
412 DSKOP:
413
414     IF MOTOR_ON_OFF            ; CODE BY HR
415 ;***** TURN OFF MOTOR *****
416 ;*****
417     LD     HL,FLATCH_STORE      ; POINT TO FLATCH
418     SET    1,(HL)              ; SET MOTOR OFF
419     CALL   OUTFLATCH            ; OUTPUT TO CONTROLLER
420 ;*****
421     ENDIF                      ; CODE BY HR
422
423
424     CALL   CHECKINT             ; CHECK INTERRUPT STATUS, MAKE SURE IT IS CLEAR
425     CP     0FFH                ; DID IT RETURN WITH ERROR CODE?
426     JP     Z,DSKEXIT            ; IF YES, EXIT WITH ERROR CODE
427
428     LD     A,(UNIT)             ; GET DISK UNIT NUMBER
429     AND    03H                 ; MASK FOR FOUR DRIVES.
430     LD     B,A                 ; PARK IT IN B
431     LD     A,(HEAD)            ; GET HEAD SELECTION
432     AND    01H                 ; INSURE SINGLE BIT
433     RLA
434     RLA                        ; MOVE HEAD TO BIT 2 POSITION
435     OR     B                   ; OR HEAD TO UNIT BYTE IN COMMAND BLOCK
436     LD     (UNIT),A            ; STORE IN UNIT
437
438     IF MOTOR_ON_OFF            ; CODE BY HR
439 ;***** TURN ON MOTOR *****
440 ;*****
441     LD     HL,FLATCH_STORE      ; POINT TO FLATCH
442     RES    1,(HL)              ; SET MOTOR ON
443     CALL   OUTFLATCH            ; OUTPUT TO CONTROLLER
444 ;*****
445     ENDIF                      ; CODE BY HR
446
447     LD     A,03H               ; SPECIFY COMMAND
448     CALL   PFDATA               ; PUSH IT
449     LD     A,(SRTHUT)           ; STEP RATE AND HEAD UNLOAD TIME
450     CALL   PFDATA               ; PUSH THAT
451     LD     A,(HLT)              ;
452     CALL   PFDATA               ; PUSH THAT
453
454     CALL   SETTRACK              ; PERFORM SEEK TO TRACK
455
456     JP     NZ,DSKEXIT            ; IF ERROR, EXIT
457
458     LD     A,(CMD)              ; WHAT COMMAND IS PENDING?
459     OR     A                    ; SET FLAGS
460     CP     4DH                 ; IS IT A FORMAT COMMAND?
461     JP     Z,FMT                ; YES, DO FORMAT
462     JP     NZ,DOSO4             ; NO, MUST BE READ OR WRITE COMMAND
463 DSKEXIT:
464     IF MOTOR_ON_OFF            ; CODE BY HR
465 ;***** TURN OFF MOTOR *****
466 ;*****
467     LD     HL,FLATCH_STORE      ; POINT TO FLATCH
468     SET    1,(HL)              ; SET MOTOR OFF
469     CALL   OUTFLATCH            ; OUTPUT TO CONTROLLER
470 ;*****
471     ENDIF                      ; CODE BY HR
472     OR     0FFH                ; SET -1 IF ERROR
473     RET
474

```

```

475
476 ;*****
477 FMT:          LD      DE,MSG_BEGIN_FORMAT      ; FORMAT TRACK COMMAND
478              LD      C,09H                    ; CP/M WRITE START STRING TO CONSOLE CALL
479              CALL     0005H                    ;
480              ;
481              LD      A,00H                    ;
482              LD      B,A                      ;
483              LD      A,(EOTSEC)                ;
484              INC     A                        ;
485              LD      C,A                      ;
486              ;
487              LD      A,(CMD)                  ;
488              CALL    PFDATA                    ; PUSH FORMAT COMMAND TO I8272
489              LD      A,(UNIT)                ;
490              CALL    PFDATA                    ; WHICH DRIVE UNIT TO FORMAT
491              LD      A,(DENS)                ;
492              CALL    PFDATA                    ; WHAT DENSITY
493              LD      A,(EOTSEC)                ;
494              CALL    PFDATA                    ; ASSUME SC (SECTOR COUNT) EOT
495              LD      A,(FM TGAP)              ;
496              CALL    PFDATA                    ; WHAT GAP IS NEEDED
497              LD      A,(D1)                  ;
498              CALL    PFDATA                    ; FILLER BYTE FOR SECTORS
499              ;
500 FMT1:
501              IN      A,(FMSR)                  ;
502              OR      A                        ; test if byte ready RQM1
503              JP      P,FMT1                    ;
504              AND     20H                      ;
505              JP      Z,DSKOPEND                ; jump if in results mode
506              ;
507              LD      A,(TRACK)                ; UPDATE I8272 DURING FORMAT
508              ;
509              OUT     (FDATA),A                ; SEND CYLINDER NUMBER
510              ;
511 FMT1A:
512              IN      A,(FMSR)                  ;
513              OR      A                        ; test if byte ready RQM1
514              JP      P,FMT1A                  ;
515              AND     20H                      ;
516              JP      Z,DSKOPEND                ; jump if in results mode
517              ;
518              LD      A,(HEAD)                ;
519              OUT     (FDATA),A                ; WHICH DRIVE HEAD TO FORMAT
520              ;
521 FMT1B:
522              IN      A,(FMSR)                  ;
523              OR      A                        ; test if byte ready RQM1
524              JP      P,FMT1B                  ;
525              AND     20H                      ;
526              JP      Z,DSKOPEND                ; jump if in results mode
527              ;
528              LD      A,B                      ;
529              OUT     (FDATA),A                ; WHAT SECTOR NUMBER
530              ;
531 FMT1C:
532              IN      A,(FMSR)                  ;
533              OR      A                        ; test if byte ready RQM1
534              JP      P,FMT1C                  ;
535              AND     20H                      ;
536              JP      Z,DSKOPEND                ; jump if in results mode
537              ;
538              LD      A,(DENS)                ;
539              OUT     (FDATA),A                ; NUMBER OF BYTES PER SECTOR (N2)
540              ;
541              INC     B                        ; INCREASE SECTOR COUNT
542              ;
543              LD      A,C                      ; IS THIS LAST SECTOR OF TRACK?
544              CP      B                        ;
545              JP      NZ,FMT1                    ; IF NO, SEND ANOTHER SECTOR
546              JP      DSKOPEND                ;
547              ;
548              ;
549
550 ;***** RESULT *****
551 ;*****
552 RESULT:
553              LD      DE,MSG_END_OPERATION      ;
554              LD      C,09H                    ; CP/M WRITE START STRING TO CONSOLE CALL
555              CALL     0005H                    ;
556              ;

```

```

557      LD      C,07H          ; LOAD C WITH NUMBER OF STATUS BYTES
558      LD      HL,ST0         ; POINT TO STATS STORAGE
559 RS3:
560      CALL     GFDATA         ; GET FIRST BYTE
561      LD      (HL),A          ; SAVE IT
562      INC     HL              ; POINTER++
563      DEC     C               ; CC-1
564      JP      NZ,RS3          ; LOOP TIL C0
565
566 RSTEXIT:
567      CALL     PRINTRESULTS    ; PRINT RESULTS OF COMMAND
568      CALL     CHECKINT        ; CHECK INTERRUPT STATUS, MAKE SURE IT IS CLEAR
569      IF MOTOR_ON_OFF         ; CODE BY HR
570 ; ***** TURN OFF MOTOR *****
571 ; *****
572      LD      HL,FLATCH_STORE ; POINT TO FLATCH
573      SET     1,(HL)           ; SET MOTOR OFF
574      CALL     OUTFLATCH       ; OUTPUT TO CONTROLLER
575 ; *****
576      ENDIF                    ; CODE BY HR
577
578      RET                     ; DONE RETURN TO CALLER.
579
580
581 PRINTRESULTS:
582      PUSH     AF
583      CALL     CRLF
584      LD      A,(ST0)          ; GET BYTE
585      CALL     HXOUT            ; PRINT IT
586      CALL     SPACE           ;
587      LD      A,(ST1)          ; GET BYTE
588      CALL     HXOUT            ; PRINT IT
589      CALL     SPACE           ;
590      LD      A,(ST2)          ; GET BYTE
591      CALL     HXOUT            ; PRINT IT
592      CALL     SPACE           ;
593      LD      A,(SCYL)         ; GET BYTE
594      CALL     HXOUT            ; PRINT IT
595      CALL     SPACE           ;
596      LD      A,(SHEAD)        ; GET BYTE
597      CALL     HXOUT            ; PRINT IT
598      CALL     SPACE           ;
599      LD      A,(SREC)         ; GET BYTE
600      CALL     HXOUT            ; PRINT IT
601      CALL     SPACE           ;
602      LD      A,(SNBIT)        ; GET BYTE
603      CALL     HXOUT            ; PRINT IT
604      CALL     SPACE           ;
605      CALL     SPACE           ;
606 ; CODE BY HR
607      CALL     TT               ; PRINT "T"
608      LD      A,(TRACK)         ; TRACK
609      CALL     HXOUT            ; PRINT IT
610      CALL     SPACE           ;
611      CALL     SPACE           ;
612 ; *****
613 ; CODE BY HR
614      CALL     SS               ; PRINT "S"
615      LD      A,(SECTOR)        ; SECTOR
616      CALL     HXOUT            ; PRINT IT
617      CALL     SPACE           ;
618      CALL     SPACE           ;
619 ; *****
620 ; CODE BY HR
621      CALL     CRLF
622      POP      AF
623      RET
624 ; *****
625
626 ; *****
627 DOS04:
628      LD      DE,MSG_BEGIN_READ ;
629      LD      C,09H            ; CP/M WRITE START STRING TO CONSOLE CALL
630      CALL     0005H           ;
631
632
633      LD      HL,SECTOR_BUFFER ; GET BUFFER ADDRESS TO HL
634      LD      A,(SECSIZ)       ; XFERLEN
635      LD      C,A              ; C WILL BE THE NUMBER OF TRANSACTIONS
636
637
638
638      LD      A,(CMD)          ;
639      CALL     PFDATA           ; PUSH COMMAND TO I8272

```



```

640      LD      A,(UNIT)          ;
641      CALL    PFDATA            ;
642      LD      A,(TRACK)         ;
643      CALL    PFDATA            ;
644      LD      A,(HEAD)          ;
645      CALL    PFDATA            ;
646      LD      A,(SECTOR)        ;
647      CALL    PFDATA            ;
648      LD      A,(DENS)          ;
649      CALL    PFDATA            ; WHAT DENSITY
650      LD      A,(EOTSEC)         ;
651      CALL    PFDATA            ; ASSUME SC (SECTOR COUNT) EOT
652      LD      A,(GAP)           ;
653      CALL    PFDATA            ; WHAT GAP IS NEEDED
654      LD      A,(DTL)           ; DTL, IS THE LAST COMMAND BYTE TO I8272
655      CALL    PFDATA            ;
656      LD      A,(CMD)           ; READ IS 0 IS THIS A READ OR WRITE?
657      AND     %00000001         ; WRITE IS 1
658      JP      NZ,WRR            ; JMP WRITE IF 1
659 ;
660 ;
661 ;
662 ; PERFORM READ
663 ; LOOP EXECUTES 4X, THIS ALLOWS C RATHER THAN BC AS COUNTER
664 ; SAVING A FEW TSTATES. MAKES UP TO 1024 BYTE SECTORS POSSIBLE.
665 ; FROM READ TO READ MUST NOT EXCEED 25US WORST CASE MIN.
666 ; (76T STATES FOR 3MHZ 8085) or (100 T STATES FOR 4MHZ Z80)
667 ;
668 ;
669 RDD_POLL:
670 FDC_READP0:
671      IN      A,(FMSR)          ;
672      OR      A                 ; test if byte ready RQM1
673      JP      P,FDC_READP0     ;
674      ;
675      AND     20H               ;
676      JP      Z,DSKOPEND        ; jump if in results mode
677      ;
678      IN      A,(FDATA)         ;
679      LD      (HL),A            ;
680      INC     HL                ;
681 ;
682 FDC_READP1:
683      IN      A,(FMSR)          ;
684      OR      A                 ;
685      JP      P,FDC_READP1     ;
686      ;
687      AND     20H               ;
688      JP      Z,DSKOPEND        ;
689      ;
690      IN      A,(FDATA)         ;
691      LD      (HL),A            ;
692      INC     HL                ;
693 ;
694 FDC_READP2:
695      IN      A,(FMSR)          ;
696      OR      A                 ;
697      JP      P,FDC_READP2     ;
698      ;
699      AND     20H               ;
700      JP      Z,DSKOPEND        ;
701      ;
702      IN      A,(FDATA)         ;
703      LD      (HL),A            ;
704      INC     HL                ;
705 ;
706 FDC_READP3:
707      IN      A,(FMSR)          ; 11
708      OR      A                 ; 4
709      JP      P,FDC_READP3     ; 10
710      ;
711      AND     20H               ; 7
712      JP      Z,DSKOPEND        ; 10
713      ;
714      IN      A,(FDATA)         ; 11
715      LD      (HL),A            ; 10
716      INC     HL                ; 11
717      ;
718      DEC     C                 ; 4
719      JP      NZ,FDC_READP0     ; 11
720 ;
721 DSKOPEND:
722 ; ***** SET TC *****

```

```

723 ;*****
724     LD      HL,FLATCH_STORE      ; POINT TO FLATCH
725     SET     0,(HL)               ; SET TC
726     CALL    OUTFLATCH           ; OUTPUT TO CONTROLLER
727 ;*****
728     NOP
729     NOP                          ; 2 MICROSECOND DELAY
730
731 ;***** RESET TC *****
732     RES     0,(HL)               ; RESET TC
733     CALL    OUTFLATCH           ; OUTPUT TO CONTROLLER
734 ;*****
735     NOP
736     NOP                          ; 2 MICROSECOND DELAY
737     NOP
738     NOP                          ; 2 MICROSECOND DELAY
739     IF MOTOR_ON_OFF             ; CODE BY HR
740 ;***** TURN OFF MOTOR *****
741 ;*****
742     SET     1,(HL)               ; TURN OFF MOTOR
743     CALL    OUTFLATCH           ; OUTPUT TO CONTROLLER
744 ;*****
745     ENDIF                       ; CODE BY HR
746
747
748     JP      RESULT              ; GET STATUS BYTES <RESULT PHASE>
749
750
751 ;*****
752 WRR:
753 FDC_WRITEP0:
754     IN      A,(FMSR)             ;
755     OR      A                   ; test if byte ready RQM1
756     JP      P,FDC_WRITEP0       ;
757     ;
758     AND     20H                 ;
759     JP      Z,DSKOPEND          ; jump if in results mode
760     ;
761     LD      A,(HL)              ;
762     OUT     (FDATA),A           ;
763     INC     HL                  ;
764
765 FDC_WRITEP1:
766     IN      A,(FMSR)             ;
767     OR      A                   ;
768     JP      P,FDC_WRITEP1       ;
769     ;
770     AND     20H                 ;
771     JP      Z,DSKOPEND          ;
772     ;
773     LD      A,(HL)              ;
774     OUT     (FDATA),A           ;
775     INC     HL                  ;
776
777 FDC_WRITEP2:
778     IN      A,(FMSR)             ;
779     OR      A                   ;
780     JP      P,FDC_WRITEP2       ;
781     ;
782     AND     20H                 ;
783     JP      Z,DSKOPEND          ;
784     ;
785     LD      A,(HL)              ;
786     OUT     (FDATA),A           ;
787     INC     HL                  ;
788
789 FDC_WRITEP3:
790     IN      A,(FMSR)             ; 11
791     OR      A                   ; 4
792     JP      P,FDC_WRITEP3       ; 10
793     ;
794     AND     20H                 ; 7
795     JP      Z,DSKOPEND          ; 10
796     ;
797     LD      A,(HL)              ;
798     OUT     (FDATA),A           ;
799     INC     HL                  ; 11
800     ;
801     DEC     C                   ; 4
802     JP      NZ,FDC_WRITEP0      ; 11
803     JP      DSKOPEND            ; 10
804
805

```

```

806 ; __SETTRACK
807 ;
808 ; SEEK TO A TRACK ON GIVEN UNIT
809 ; A: TRACK #
810 ;
811 ;
812 SETTRACK:
813 ; ANY INTERRUPT PENDING
814 ; IF YES FIND OUT WHY/CLEAR
815 CALL CHECKINT ; CHECK INTERRUPT STATUS, MAKE SURE IT IS CLEAR
816 CP 0FFH ; DID IT RETURN WITH ERROR CODE?
817 JP Z,SETTRKEXIT ;
818 ;
819 LD A,(TRACK) ; GET TRACK
820 OR A ; SET FLAGS
821 JP Z,RECAL ; IF 0 PERFORM RECAL INSTEAD OF SEEK
822 LD A,0FH ; SEEK COMMAND
823 CALL PFDATA ; PUSH COMMAND
824 LD A,(UNIT) ; SAY WHICH UNIT
825 CALL PFDATA ; SEND THAT
826 LD A,(TRACK) ; TO WHAT TRACK
827 CALL PFDATA ; SEND THAT TOO
828 JP WAIT ; WAIT FOR INTERRUPT SAYING DONE
829 ;
830 ;*****
831 RECAL:
832 LD A,07H ; RECAL TO TRACK 0
833 CALL PFDATA ; SEND IT
834 LD A,(UNIT) ; WHICH UNIT
835 CALL PFDATA ; SEND THAT TOO
836 ;
837 WAIT:
838 ;
839 CALL DELAYHSEC ; DELAY TO LET HEADS SETTLE BEFORE READ
840 ;
841 ; WAIT HERE FOR INTERRUPT SAYING DONE
842 ; LOOP TIL INTERRUPT
843 CALL CHECKINT ; CHECK INTERRUPT STATUS
844 ;
845 SETTRKEXIT:
846 RET
847 ;
848 ; __PFDATAS
849 ;
850 ; WRITE A COMMAND OR PARAMETER SEQUENCE
851 ;
852 ; TRANSFERS ARE SYNCHRONIZED BYT MSR D7 <RQM> AND D6 <DIO>
853 ; RQM DIO
854 ; 0 0 BUSY
855 ; 1 0 WRITE TO DATA REGISTER PERMITTED
856 ; 1 1 BYTE FOR READ BY HOST PENDING
857 ; 0 1 BUSY
858 ;
859 ;
860 ;
861 PFDATAS:
862 PUSH AF ; STORE AF
863 PFDS1:
864 IN A,(FMSR) ; READING OR WRITING IS KEYS TO D7 RQM
865 AND 80H ; MASK OFF RQM BIT
866 JP Z,PFDS1 ; WAIT FOR RQM TO BE TRUE.
867 IN A,(FMSR) ; READ STATUS
868 AND 40H ; WAITING FOR INPUT?
869 ;*****
870 CALL NZ,ERRORT ; NO, SIGNAL ERROR
871 POP AF ; RESTORE AF
872 OUT (FDATA),A ; OUTPUT A TO CONTROLLER
873 RET
874 ;
875 ; __PFDATA
876 ;
877 ; WRITE A COMMAND OR PARAMETER SEQUENCE
878 ;
879 ; TRANSFERS ARE SYNCHRONIZED BYT MSR D7 <RQM> AND D6 <DIO>
880 ; RQM DIO
881 ; 0 0 BUSY
882 ; 1 0 WRITE TO DATA REGISTER PERMITTED
883 ; 1 1 BYTE FOR READ BY HOST PENDING

```

```

884 ;           0           1           BUSY
885 ;
886 ;_____

887 ;
888 PFDATA:
889     PUSH     AF           ; STORE AF
890 PFD1:
891     IN       A,(FMSR)     ; READING OR WRITING IS KEYS TO D7 RQM
892     AND      80H          ; MASK OFF RQM BIT
893     JP       Z,PFD1       ; WAIT FOR RQM TO BE TRUE.
894     IN       A,(FMSR)     ; READ STATUS
895     AND      40H          ; WAITING FOR INPUT?
896 ;*****
897     CALL     NZ,ERRORT     ; NO, SIGNAL ERROR
898     POP      AF           ; RESTORE AF
899     OUT      (FDATA),A     ; OUTPUT A TO CONTROLLER
900     NOP      ; WAIT 24 US BEFORE READING FMSR AGAIN
901     NOP      ;
902     NOP      ;
903     NOP      ;
904     NOP      ;
905     NOP      ;
906     NOP      ;
907     NOP      ;
908     NOP      ;
909     NOP      ;
910     NOP      ;
911     NOP      ;
912     NOP      ;
913     NOP      ;
914     NOP      ;
915     NOP      ;
916     NOP      ;
917     NOP      ;
918     NOP      ;
919     NOP      ;
920     NOP      ;
921     NOP      ;
922     NOP      ;
923     NOP      ;
924     RET
925
926
927 ;_CHECKINT_____

928 ;
929 ; CHECK FOR ACTIVE FDC INTERRUPTS BEFORE GIVING I8272 COMMANDS
930 ; POLL RQM FOR WHEN NOT BUSY AND THEN SEND FDC
931 ; SENSE INTERRUPT COMMAND. IF IT RETURNS WITH NON ZERO
932 ; ERROR CODE, PASS BACK TO CALLING ROUTINE FOR HANDLING
933 ;_____

934 ;
935 CHECKINT:
936     PUSH     AF           ; STORE AF
937 WTDONE:
938     IN       A,(FMSR)     ; READING OR WRITING IS KEYS TO D7 RQM
939     AND      80H          ; MASK OFF RQM BIT
940     JP       Z,WTDONE     ; WAIT FOR RQM TO BE TRUE. WAIT UNTIL DONE
941     IN       A,(FMSR)     ; READ STATUS
942     AND      40H          ; WAITING FOR INPUT?
943 ;*****
944     CALL     NZ,ERRORT     ; NO, SIGNAL ERROR
945     POP      AF           ; RESTORE AF
946     CALL     SENDINT       ; SENSE INTERRUPT COMMAND
947     PUSH     AF           ; STORE AF
948     ;
949     CP       %00000000     ; NORMAL TERMINATION CASE
950     LD       DE,MSG_NORMAL_TERM ; SET POINTER TO NORMAL TERM MESSAGE
951     JP       Z,EXITCI      ; EXIT
952     ;
953     CP       %01000000     ; ABNORMAL TERMINATION OF COMMAND CASE
954     LD       DE,MSG_ABNORMAL_TERM_CMD ; SET POINTER TO COMMAND STARTED BUT NOT SUCCESSFULLY
955 COMPLETED
956     JP       Z,EXITCI      ; EXIT
957     ;
958     CP       %11000000     ; ABNORMAL TERMINATION OF COMMAND CASE
959     LD       DE,MSG_READY_CHANGED ; SET POINTER TO READY SIGNAL FROM FDD CHANGED S
960 TATE
961     JP       Z,EXITCI      ; EXIT
962     ;
963     CP       %10000000     ; INVALID COMMAND

```

```

962          LD      DE,MSG_INVALID_COMMAND      ; SET POINTER TO INVALID COMMAND MESSAGE
963
964
965 EXITCI:
966          LD      C,09H                        ; CP/M WRITE END STRING TO CONSOLE CALL
967          CALL    0005H                        ;
968          POP     AF                          ; RESTORE AF
969          CALL    PRINTRESULTS                 ; PRINT RESULTS OF COMMAND
970
971          RET
972
973 ; _DELAYHSEC_


---


974 ;
975 ; DELAY FOR 1/2 SECOND
976 ;


---


977 ;
978 DELAYHSEC:
979          LD      HL,00000H                    ; 65536
980 DELDM:
981          NOP
982          NOP
983          NOP
984          NOP
985          DEC     L
986          JP      NZ,DELDM                     ; (10 T) 24 T 8 MICROSECONDS AT 4 MHZ
987          DEC     H
988          JP      NZ,DELDM                     ; (10 T) (8 US * 256) * 256 524288 US .5 SECONDS
989          RET
990
991 ; _ERRORT_


---


992 ;
993 ; ERROR HANDLING
994 ;


---


995 ;
996 ERRORT:
997          POP     AF
998 ERRORT1:
999 ; CODE BY HR
1000         LD      DE,MSG_ERRORX                ; POINT TO ERROR STRING
1001 ; *****
1002         LD      C,09H                        ; CP/M WRITE ERROR STRING TO CONSOLE CALL
1003         CALL    0005H                        ;
1004
1005 ERRCLR:
1006         IN      A,(FDATA)                    ; CLEAR THE JUNK OUT OF DATA REGISTER
1007         IN      A,(FMSR)                     ; CHECK WITH RQM
1008         AND     80H                          ; IF STILL NOT READY, READ OUT MORE JUNK
1009
1010         JP      Z,ERRCLR                     ;
1011
1012         LD      A,0FFH                       ; RETURN ERROR CODE -1
1013
1014         RET
1015 ; _SENDINT_


---


1016 ;
1017 ; SENSE INTERRUPT COMMAND
1018 ;


---


1019 ;
1020 SENDINT:
1021         LD      A,08H                        ; SENSE INTERRUPT COMMAND
1022         CALL    PFDATA                       ; SEND IT
1023         CALL    GFDATA                       ; GET RESULTS
1024         LD      (ST0),A                      ; STORE THAT
1025         AND     0C0H                          ; MASK OFF INTERRUPT STATUS BITS
1026         CP      80H                          ; CHECK IF INVALID COMMAND
1027         JP      Z,ENDSENDINT                 ; YES, EXIT
1028         CALL    GFDATA                       ; GET ANOTHER (STATUS CODE 1)
1029         LD      (ST1),A                      ; SAVE THAT
1030         LD      A,(ST0)                      ; GET FIRST ONE
1031         AND     0C0H                          ; MASK OFF ALL BUT INTERRUPT CODE 00 IS NORMAL
1032 ENDSENDINT:
1033         RET
1034
1035
1036 ; _GFDATA_


---



```

```

1037 ;
1038 ; GET DATA FROM FLOPPY CONTROLLER
1039 ;
1040 ; TRANSFERS ARE SYNCHRONIZED BYT MSR D7 <RQM> AND D6 <DIO>
1041 ;      RQM  DIO
1042 ;      0      0      BUSY
1043 ;      1      0      WRITE TO DATA REGISTER PERMITTED
1044 ;      1      1      BYTE FOR READ BY HOST PENDING
1045 ;      0      1      BUSY
1046 ;
1047 ;


---


1048 ;
1049 ;***** GFDATA *****
1050 ;*****
1051 GFDATA:
1052      IN      A,(FMSR)      ; READ STATUS BYTE
1053      AND     80H          ; MASK OFF RQM
1054      JP      Z,GFDATA     ; LOOP WHILE BUSY
1055      IN      A,(FMSR)     ; READ STATUS BYTE
1056      AND     40H          ; MASK OFF DIO
1057      CALL    Z,ERRORT1    ; IF WRITE EXPECTED RUN ERRORRT
1058      IN      A,(FDATA)    ; READ DATA
1059      NOP     ; WAIT 24 US BEFORE READING FMSR AGAIN
1060      NOP
1061      NOP
1062      NOP
1063      NOP
1064      NOP
1065      NOP
1066      NOP
1067      NOP
1068      NOP
1069      NOP
1070      NOP
1071      NOP
1072      NOP
1073      NOP
1074      NOP
1075      NOP
1076      NOP
1077      NOP
1078      NOP
1079      NOP
1080      NOP
1081      NOP
1082      NOP
1083      RET
1084
1085
1086 ; _DUMP


---


1087 ;
1088 ;      Print a Memory Dump (LOW)
1089 ;


---


1090 ;
1091 DUMP_BUFFER:
1092      CALL    CRLF          ;
1093 BLKRD:
1094      CALL    PHL          ; PRINT START LOCATION
1095      LD      C,16         ; SET FOR 16 LOCS
1096      PUSH    HL          ; SAVE STARTING HL
1097 NXTONE:
1098      LD      A,(HL)       ; GET BYTE
1099      CALL    HXOUT        ; PRINT IT
1100      CALL    SPACE        ;
1101 UPDH:
1102      INC     HL          ; POINT NEXT
1103      DEC     C           ; DEC. LOC COUNT
1104      JR      NZ,NXTONE    ; IF LINE NOT DONE
1105      ; NOW PRINT 'DECODED' DATA TO RIGHT OF DUMP
1106 PCRLF:
1107      CALL    SPACE        ; SPACE IT
1108      LD      C,16         ; SET FOR 16 CHARS
1109      POP     HL          ; GET BACK START
1110 PCRLF0:
1111      LD      A,(HL)       ; GET BYTE
1112      AND     060H         ; SEE IF A 'DOT'
1113      LD      A,(HL)       ; O.K. TO GET
1114      JR      NZ,PDOT      ;
1115 DOT:
1116      LD      A,2EH        ; LOAD A DOT

```

```

1117 PDOT:
1118     CALL    COUT           ; PRINT IT
1119     INC     HL             ;
1120     LD      A,D            ;
1121     CP      H              ;
1122     JR      NZ,UPDH1       ;
1123     LD      A,E            ;
1124     CP      L              ;
1125     JP      Z,DUMP_END     ;
1126 ;
1127 ;IF BLOCK NOT DUMPED, DO NEXT CHARACTER OR LINE
1128 UPDH1:
1129     DEC     C              ; DEC. CHAR COUNT
1130     JR      NZ,PCRLF0      ; DO NEXT
1131 CONTD:
1132     CALL    CRLF           ;
1133     JP      BLKRD          ;
1134 ;
1135 DUMP_END:
1136     RET                   ;
1137 ;
1138 ;
1139 ;_HXOUT_


---


1140 ;
1141 ;     PRINT THE ACCUMULATOR CONTENTS AS HEX DATA ON THE SERIAL PORT
1142 ;


---


1143 ;
1144 HXOUT:
1145     PUSH    BC             ; SAVE BC
1146     LD      B,A            ;
1147     RLC     A              ; DO HIGH NIBBLE FIRST
1148     RLC     A              ;
1149     RLC     A              ;
1150     RLC     A              ;
1151     AND     0FH            ; ONLY THIS NOW
1152     ADD     A,30H          ; TRY A NUMBER
1153     CP      3AH            ; TEST IT
1154     JR      C,OUT1         ; IF CY SET PRINT 'NUMBER'
1155     ADD     A,07H          ; MAKE IT AN ALPHA
1156 OUT1:
1157     CALL    COUT           ; SCREEN IT
1158     LD      A,B            ; NEXT NIBBLE
1159     AND     0FH            ; JUST THIS
1160     ADD     A,30H          ; TRY A NUMBER
1161     CP      3AH            ; TEST IT
1162     JR      C,OUT2         ; PRINT 'NUMBER'
1163     ADD     A,07H          ; MAKE IT ALPHA
1164 OUT2:
1165     CALL    COUT           ; SCREEN IT
1166     POP     BC             ; RESTORE BC
1167     RET                   ;
1168 ;
1169 ;
1170 ;_SPACE_


---


1171 ;
1172 ;     PRINT A SPACE CHARACTER ON THE SERIAL PORT
1173 ;


---


1174 ;
1175 SPACE:
1176     PUSH    AF             ; Store AF
1177     LD      A,20H          ; LOAD A "SPACE"
1178     CALL    COUT           ; SCREEN IT
1179     POP     AF             ; RESTORE AF
1180     RET                   ; DONE
1181 ;
1182 ;_CRLF_


---


1183 ;
1184 ;     PRINT A cr/lf
1185 ;


---


1186 ;
1187 CRLF:
1188     PUSH    AF             ; Store AF
1189     LD      A,0DH          ; LOAD A 0DH
1190     CALL    COUT           ; SCREEN IT
1191     LD      A,0AH          ; LOAD A 0AH
1192     CALL    COUT           ; SCREEN IT
1193     POP     AF             ; RESTORE AF

```

```

1194          RET                      ; DONE
1195
1196 ; _COUT_


---


1197 ;
1198 ;          PRINT CONTENTS OF A
1199 ;


---


1200 ;
1201 COUT:
1202     PUSH     BC                      ;
1203     PUSH     AF                      ;
1204     PUSH     HL                      ;
1205     PUSH     DE                      ;
1206
1207     LD       (COUT_BUFFER),A ;
1208     LD       DE,COUT_BUFFER ;
1209     LD       C,09H                ; CP/M WRITE START STRING TO CONSOLE CALL
1210     CALL     0005H
1211     POP      DE                      ;
1212     POP      HL                      ;
1213     POP      AF                      ;
1214     POP      BC                      ;
1215     RET                      ; DONE
1216
1217
1218
1219
1220 ; _PHL_


---


1221 ;
1222 ;          PRINT THE HL REG ON THE SERIAL PORT
1223 ;


---


1224 ;
1225 PHL:
1226     LD       A,H                    ; GET HI BYTE
1227     CALL     HXOUT                  ; DO HEX OUT ROUTINE
1228     LD       A,L                    ; GET LOW BYTE
1229     CALL     HXOUT                  ; HEX IT
1230     CALL     SPACE                  ;
1231     RET                      ; DONE
1232
1233
1234 ; _GETLN_


---


1235 ;
1236 ;          Read a line(80) of text from the Serial Port, handle <BS>, term on <CR>.
1237 ;          Exit if too many chars.  Store result in HL.  Char count in C.
1238 ;


---


1239 ;
1240 GETLN:
1241     LD       C,00H                  ; ZERO CHAR COUNTER
1242     PUSH     DE                      ; STORE DE
1243 GETLNLOP:
1244     CALL     KIN                    ; GET A KEY
1245     CP       CR                     ; IS <CR>?
1246     JR       Z,GETLNDONE            ; YES, EXIT
1247     CP       BS                     ; IS <BS>?
1248     JR       NZ,GETLNSTORE          ; NO, STORE CHAR
1249     LD       A,C                    ; A=C
1250     CP       0                      ;
1251     JR       Z,GETLNLOP             ; NOTHING TO BACKSPACE, IGNORE & GET NEXT KEY
1252     DEC      HL                     ; PERFORM BACKSPACE
1253     DEC      C                      ; LOWER CHAR COUNTER
1254     LD       A,0                    ;
1255     LD       (HL),A                 ; STORE NULL IN BUFFER
1256     LD       A,20H                  ; BLANK OUT CHAR ON TERM
1257     CALL     COUT                    ;
1258     LD       A,BS                    ;
1259     CALL     COUT                    ;
1260     JR       GETLNLOP              ; GET NEXT KEY
1261 GETLNSTORE:
1262     LD       (HL),A                 ; STORE CHAR IN BUFFER
1263     INC      HL                     ; INC POINTER
1264     INC      C                      ; INC CHAR COUNTER
1265     LD       A,C                    ; A=C
1266     CP       4DH                     ; OUT OF BUFFER SPACE?
1267     JR       NZ,GETLNLOP            ; NOPE, GET NEXT CHAR
1268 GETLNDONE:
1269     LD       (HL),00H                ; STORE NULL IN BUFFER
1270     POP      DE                      ; RESTORE DE

```



```

1271          RET                ;
1272
1273
1274 ; _KIN_


---


1275 ;
1276 ;      Read from the Serial Port and Echo & convert input to UCASE
1277 ;


---


1278 ;
1279 KIN:
1280     PUSH     BC                ;
1281     PUSH     HL                ;
1282     PUSH     DE                ;
1283     LD       C,01H            ; CP/M console input call
1284     CALL     0005H            ; GET K.B. DATA
1285     POP      DE                ;
1286     POP      HL                ;
1287     POP      BC                ;
1288     RET
1289
1290 ; _LDHL_


---


1291 ;
1292 ;      GET ONE WORD OF HEX DATA FROM BUFFER POINTED TO BY HL SERIAL PORT, RETURN IN HL
1293 ;


---


1294 ;
1295 LDHL:
1296     PUSH     DE                ; STORE DE
1297     CALL     HEXIN            ; GET K.B. AND MAKE HEX
1298     LD       D,A              ; THATS THE HI BYTE
1299     CALL     HEXIN            ; DO HEX AGAIN
1300     LD       L,A              ; THATS THE LOW BYTE
1301     LD       H,D              ; MOVE TO HL
1302     POP      DE                ; RESTORE BC
1303     RET                  ; GO BACK WITH ADDRESS
1304
1305
1306 ; _HEXIN_


---


1307 ;
1308 ;      GET ONE BYTE OF HEX DATA FROM BUFFER IN HL, RETURN IN A
1309 ;


---


1310 ;
1311 HEXIN:
1312     PUSH     BC                ; SAVE BC REGS.
1313     CALL     NIBL              ; DO A NIBBLE
1314     RLC      A                ; MOVE FIRST BYTE UPPER NIBBLE
1315     RLC      A                ;
1316     RLC      A                ;
1317     RLC      A                ;
1318     LD       B,A              ; SAVE ROTATED BYTE
1319     CALL     NIBL              ; DO NEXT NIBBLE
1320     ADD      A,B              ; COMBINE NIBBLES IN ACC.
1321     POP      BC                ; RESTORE BC
1322     RET                  ; DONE
1323 NIBL:
1324     LD       A,(HL)           ; GET K.B. DATA
1325     INC      HL                ; INC KB POINTER
1326     CP      40H              ; TEST FOR ALPHA
1327     JR       NC,ALPH          ;
1328     AND      0FH              ; GET THE BITS
1329     RET
1330 ALPH:
1331     AND      0FH              ; GET THE BITS
1332     ADD      A,09H            ; MAKE IT HEX A-F
1333     RET
1334
1335 ; _FILL_MEM_


---


1336 ;
1337 ;      Function : fill memory with a value
1338 ;      Input      : HL = start address block
1339 ;                  : BC = length of block
1340 ;                  : A = value to fill with
1341 ;      Uses       : DE, BC
1342 ;


---


1343 ;
1344
1345 FILL_MEM:

```

```

1346                                ; HL = start address of block
1347                                ; DE = HL + 1
1348    ld        e,l                ;
1349    ld        d,h                ;
1350    inc       de                 ;
1351                                ; initialise first byte of block
1352                                ; with data byte (&00)
1353                                ; with data byte in A
1354    ld        (hl),A             ;
1355    ldir      ;
1356    RET      ; return to caller
1357
1358
1359
1360 ;-----
1361
1362 ; TEXT CONSTANTS
1363
1364 MSG_START:
1365    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1366    DEFM      "FLOPPY TEST PROGRAM"
1367    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1368
1369    DEFM      "1->START MOTOR"
1370    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1371    DEFM      "2->STOP MOTOR"
1372    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1373    DEFM      "3->SELECT TRACK"
1374    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1375    DEFM      "4->FORMAT TRACK"
1376    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1377    DEFM      "5->READ SECTOR"
1378    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1379    DEFM      "6->WRITE SECTOR"
1380    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1381    DEFM      "7->DUMP BUFFER"
1382    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1383    DEFM      "8->SENSE INT"
1384    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1385    DEFM      "9->CLEAR BUFFER"
1386    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1387    DEFM      "Q->Quit"
1388    DEFB      LF, CR             ; LINE FEED AND CARRIAGE RETURN
1389    DEFB      ENDS              ; LINE TERMINATOR
1390 MSG_NORMAL_TERM:
1391    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1392    DEFM      "INTERRUPT NORMAL TERMINATION"
1393    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1394    DEFB      ENDS              ; LINE TERMINATOR
1395 ;
1396 MSG_ABNORMAL_TERM_CMD:
1397    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1398    DEFM      "ABNORMAL TERMINATION OF COMMAND: STARTED BUT RETURNED ERROR"
1399    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1400    DEFB      ENDS              ; LINE TERMINATOR
1401 ;
1402 MSG_INVALID_COMMAND:
1403    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1404    DEFM      "INVALID COMMAND"
1405    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1406    DEFB      ENDS              ; LINE TERMINATOR
1407 ;
1408 MSG_READY_CHANGED:
1409    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1410    DEFM      "ABNORMAL TERMINATION OF COMMAND: FDD READY STATE CHANGED"
1411    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1412    DEFB      ENDS              ; LINE TERMINATOR
1413 ;
1414 MSG_ERROR:
1415    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1416    DEFM      "TERMINATION ERROR"
1417    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1418    DEFB      ENDS              ; LINE TERMINATOR
1419 ; CODE BY HR
1420 MSG_ERRORX:
1421    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1422    DEFM      "TERMINATION ERRORX"
1423    DEFB      LF,CR              ; LINE FEED AND CARRIAGE RETURN
1424    DEFB      ENDS              ; LINE TERMINATOR
1425 ;*****
1426
1427 MSG_ENTER_TRACK_NUMBER:
1428    DEFB      LF, CR              ; LINE FEED AND CARRIAGE RETURN

```

```

1429      DEFM    "ENTER TRACK NUMBER (00):"
1430      DEFB     ENDS                                ; LINE TERMINATOR
1431 MSG_ENTER_SECTOR_NUMBER:
1432      DEFB     LF, CR                               ; LINE FEED AND CARRIAGE RETURN
1433      DEFM    "ENTER SECTOR NUMBER (00):"
1434      DEFB     ENDS                                ; LINE TERMINATOR
1435 MSG_BEGIN_FORMAT:
1436      DEFB     LF, CR                               ; LINE FEED AND CARRIAGE RETURN
1437      DEFM    "FORMAT BEGINNING. . ."
1438      DEFB     ENDS                                ; LINE TERMINATOR
1439 MSG_BEGIN_READ:
1440      DEFB     LF, CR                               ; LINE FEED AND CARRIAGE RETURN
1441      DEFM    "READ/WRITE BEGINNING. . ."
1442      DEFB     ENDS                                ; LINE TERMINATOR
1443 MSG_END_OPERATION:
1444      DEFB     LF, CR                               ; LINE FEED AND CARRIAGE RETURN
1445      DEFM    "OPERATION ENDED. . ."
1446      DEFB     ENDS                                ; LINE TERMINATOR
1447
1448 MSG_END:
1449      DEFB     LF, CR                               ; LINE FEED AND CARRIAGE RETURN
1450      DEFM    "END FLOPPY TEST PROGRAM"
1451      DEFB     LF, CR                               ; LINE FEED AND CARRIAGE RETURN
1452      DEFB     ENDS                                ; LINE TERMINATOR
1453 ; CODE BY HR
1454 MSG_YES_NO:
1455      DEFB     LF, CR                               ; LINE FEED AND CARRIAGE RETURN
1456      DEFM    "YES or NO"
1457      DEFB     LF, CR                               ; LINE FEED AND CARRIAGE RETURN
1458      DEFB     ENDS                                ; LINE TERMINATOR
1459 ; *****
1460
1461
1462
1463
1464 COUT_BUFFER:
1465      DEFB     00
1466      DEFB     "$"
1467 FLATCH_STORE:
1468      DEFB     00
1469 ;
1470 ; DISK COMMAND BLOCK
1471 ;
1472 CMD:      DEFB     0                                ; COMMAND READ OR WRITE,
1473 UNIT:     DEFB     0                                ; PHYSICAL DRIVE 0->3
1474 TRACK:    DEFB     0                                ; PHYSICAL TRACK 0->256
1475 HEAD:     DEFB     0                                ; HEAD SEL 0 OR 1
1476 SECTOR:   DEFB     1                                ; PHYSICAL SECTOR, ALWAYS 1 TO MAX SECTOR
1477 DENS:     DEFB     2                                ; DENSITY
1478 EOTSEC:   DEFB     09                               ; LAST SECTOR OF TRACK
1479 GAP:       DEFB     1BH                             ; VALUE FOR IRG <GAP3>
1480 SECSIZ:   DEFB     080H                             ; HOW MANY BYTES TO TRANSFER/4
1481 DTL:      DEFB     0FFH                             ; SIZE OF SECTOR
1482 ;
1483 SRTHUT:   DEFB     7FH                               ; STEP RATE AND HEAD UNLOAD TIME
1484 HLT:      DEFB     05H                               ; HEAD LOAD TIME
1485 ;
1486 MIN:      DEFB     MINI                             ; LATCH BIT PATTERN FOR FDC9229 MINITRUE
1487 ; CODE BY HR
1488 ; D1 instead of D
1489 ; D1 instead of D because the Assembler does not allow D
1490 D1:       DEFB     0E5H                             ; FILLER BYTE FOR FORMAT
1491 ; *****
1492 FMTGAP:   DEFB     054H                             ; GAP FOR FORMAT
1493 PRE:      DEFB     PRECOMP                           ; LATCH BIT PATTERN FOR FDC9229 PRECOMP125NS
1494 ;
1495 ; STATUS RESULT STORAGE
1496 ;
1497 ST0:      DEFB     0                                ; STORE STATUS 0
1498 ST1:      DEFB     0                                ; ST1
1499 ST2:      DEFB     0                                ; ST2
1500 SCYL:     DEFB     0                                ; TRACK
1501 SHEAD:    DEFB     0                                ; HEAD 0 OR 1
1502 SREC:     DEFB     0                                ; SECTOR
1503 SNBIT:    DEFB     0                                ; DENSITY
1504
1505 ; CODE BY HR
1506 KEYBUF:   DEFS     80,20h                           ; 80 x SPACE
1507 ; *****
1508 ; KEYBUF:   DEFM    "
1509 KEYEND:
1510 TKEYEND:  EQU     KEYEND-KEYBUF

```

```
1511 ;*****
1512      ORG          2500h
1513  SECTOR_BUFFER:      DEFS      0200h
1514
1515      END
1516
1517
```