

COLECO

ADAM TECHNICAL MANUAL

- IV - EOS 6 ABSOLUTE LISTINGS
- V - OS 7 ABSOLUTE LISTINGS



SECTION IV

EOS 6 ABSOLUTE LISTINGS



LOCATION OBJECT CODE LINE SOURCE LINE

```

1 ^Z80^
3 NAME ^Rev 06.1 - TF^
4
5 : Project: EOS 6 Absolute Listing
6 :
7 :
8 : EOS ABSOLUTE LISTING
9 :
10 :
11 : Electronic Development Group
12 : Coleco Industries Inc.
13 : 999 Quaker Lane South
14 : West Hartford, Connecticut
15 : 06110
16 :
17 :
18 :

```

```

19 : This absolute listing was generated to ease software
20 : development on ADAM. This listing provides the location
21 : of both released and unreleased entry points. Released
22 : entry points begin immediately in this file with the jump
23 : table and end before the first code segment listed.
24 : Released entry points include the jump table, common data
25 : areas(EOS_COMN), common data tables, and equates which
26 : describe the released data structures. Direct access
27 : to code segments is STRONGLY DISCOURAGED and may make
28 : your application incompatible with some ADAMs. There is
29 : more than one version of EOS on the market at this time
30 : and updates are planned.

```

```

31 :
32 : For further information on ADAM or EOS consult the
33 : ADAM Technical Reference Manual or send your questions
34 : in writing to:

```

Coleco Software Support

at the above address.

```

35 :
36 :
37 :
38 :
39 :
40 : COLECO MAKES NO REPRESENTATIONS OR WARRANTIES WHATSOEVER, INCLUDING
41 : WITHOUT LIMITATION ANY IMPLIED WARRANTIES OR MERCHANTABILITY AND
42 : FITNESS FOR A PARTICULAR PURPOSE, IN CONNECTION WITH THE MATERIALS
43 : CONTAINED HEREIN, AND SUCH MATERIALS ARE DISCLOSED AS IS. COLECO
44 : SHALL HAVE NO LIABILITY FOR ANY LOSSES CAUSED TO RECIPIENTS OF THESE
45 : MATERIALS BY REASON OF ANY CHANGES OR MODIFICATIONS MADE BY COLECO IN
46 : THESE MATERIALS AFTER THEIR DISCLOSURE HEREIN. IN ADDITION, COLECO
47 : SHALL HAVE NO LIABILITY FOR ANY CONSEQUENTIAL, SPECIAL, INDIRECT OR
48 : INCIDENTAL DAMAGES OR LOSSES WHATSOEVER, INCLUDING LOSS OF PROFITS, IN
49 : CONNECTION WITH THE USE OF THE MATERIALS DISCLOSED HEREIN.

```

```
LOCATION OBJECT CODE LINE SOURCE LINE
51 : .....
52 : .....
53 : .....
54 : .....
55 : .....
56 : .....
57 : .....
58 : .....
59 : [ code commented out by $:eee' was removed before EOS ]
60 : [ was grouped into one file, code commented out by simple ]
61 : [ $: was removed in the process of grouping EOS in order ]
62 : [ to avoid assembler complaints or LOCAL/GLOBAL conflicts ]
63 : [ $:....., separates code segments which were ]
64 : [ originally in separate files ]
65 : .....
66 : .....
67 : .....
```

Rev	Date	Name	Change
6.1	21Jun0844	af	Merge sources for absolute listing.
6.0	08oct1545	jk1	No changes made to code.
			Code released for production.

LOCATION OBJECT CODE LINE SOURCE LINE

```

69 NAME ^Rev 11 - RPD^
70 :
71 : Rev Date Made by Change
72 : 11 15oct425p RPD changed EOS_4 to CV_A
73 : 10 13oct1000p RPD made only 057 jump entries conditionally assembled
74 : 9 12oct2300 vsb Added __SCAN_FOR_FILE,__POSIT_FILE,__FILE_QUERY plus __EOS1 thru __EOS4
75 : 8 07oct1710 rfj __EFXOVER changed to __EFFECT_OVER
76 : Added __UPDATE_SPINNER
77 : 7 05oct1725 rfj Renamed WR_SPR_NM_TBL to WR_SPR_ATTRIBUTE
78 : 6 03oct1207 rfj Increased TEMP_STACK to 12 bytes.
79 : Removed SAVED_COUNT
80 : 5 30sept825p RPD replaced CONS_OUT data area with new CONS_OUT2
81 : 4 29sept235p RPD replaced PLAY_SONGS with SOUNDS
82 : 3 26sept1005a RPD added conditional assembly stuff for super games
83 : 2 25sept850a RPD merged all of EOS ram together into this file
84 : 1 20sept1005a RPD new memory mapping
85 : 0 mid-sept VB original map @ E800H
86 :
87 TRUE EQU OFFFFH
88 FALSE EQU O0000H
89
90 IN EOS EQU TRUE
91 SUPERGAME EQU FALSE
92 :
93 : CURRENT EOS MEMORY MAPPING
94 :
95 : FCB_S EQU O0390H :FCB area
96 : THREE1K_BLKS EQU O0400H :3 FCB 1K buffers
97 EOS_CODE EQU O0000H :address of EOS code segment
98 EOS_GLB_TBL EQU OFBFFH :address of EOS global tables
99 EOS_JMP_TBL EQU OFC30H :address of EOS jump table
100 EOS_GLB_RAM EQU OFD60H :address of EOS global data area
101 EOS_PCB_DCB EQU OFE00H :address of EOS PCB/DCB areas
102 :
103 :
104 : These are key record sizes, used in various EOS routines.
105 : See "INCLUDE FMGR_EQU" AND "INCLUDE P_DCB_EQU" for details.
106 :
107 :
108 DIR_ENT_LENGTH DEFN 26
109 P_SIZE DEFN 4 : THE NUMBER OF BYTES IN THE PCB
110 D_SIZE DEFN 21 : THE NUMBER OF BYTES IN THE DCB
111 :

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

113 :
114 : jump table area
115 :
116 : .....
117 :
118 :
119 :          ORG EOS_JMP_TBL
120 : .....
121 :
122 <FC30>      EQU $
123 FC30 C3F832  _EOS_START
124 <FC33>      EQU $
125 FC33 C3F627  _CONS_DISP
126 <FC36>      EQU $
127 FC36 C3F5DC  _CONS_INIT
128 <FC39>      EQU $
129 FC39 C3F60A  _CONS_OUT
130 <FC3C>      EQU $
131 FC3C C3F95F  _DLY_AFT_HRD_RES
132 <FC3F>      EQU $
133 FC3F C3F588  _END_PR_BUFF
134 <FC42>      EQU $
135 FC42 C3F57C  _END_PR_CH
136 <FC45>      EQU $
137 FC45 C3FAE2  _END_RD_1_BLOCK
138 <FC48>      EQU $
139 FC48 C3F8A5  _END_RD_CH_DEV
140 <FC4B>      EQU $
141 FC4B C3F4E0  _END_RD_KBD
142 <FC4E>      EQU $
143 FC4E C3F81B  _END_WR_1_BLOCK
144 <FC51>      EQU $
145 FC51 C3FBE1  _END_WR_CH_DEV
146 <FC54>      EQU $
147 FC54 C3F446  _FIND_DCB
148 <FC57>      EQU $
149 FC57 C3F446  _GET_DCB_ADDR
150 <FC5A>      EQU $
151 FC5A C3FA4C  _GET_PCB_ADDR
152 <FC5D>      EQU $
153 FC5D C3F8F6  _HARD_INIT
154 <FC60>      EQU $
155 FC60 C3F948  _HARD_RESET_NET
156 <FC63>      EQU $
157 FC63 C3F515  _PR_BUFF
158 <FC66>      EQU $
159 FC66 C3F4FC  _PR_CH
160 <FC69>      EQU $
161 FC69 C3FA9E  _RD_1_BLOCK
162 <FC6C>      EQU $
163 FC6C C3F4BA  _RD_KBD
164 <FC6F>      EQU $
165 FC6F C3FA7B  _RD_KBD_RET_CODE
166 <FC72>      EQU $
167 FC72 C3FA7F  _RD_PR_RET_CODE
168 <FC75>      EQU $
169 FC75 C3F      _RD_RET_CODE

```


LOCATION	OBJECT	CODE	LINE	SOURCE	LINE	EQU
FC78	C3FA83	<FC78>	170	RD_TAPE_RET_CODE	170	EQU \$
			171	JP	RD_TAPE_RET_CODE	
			172	RELOC_PCB		EQU \$
FC7B	C3FA2F	<FC7B>	173	JP	RELOC_PCB	
			174	REQUEST_STATUS		EQU \$
FC7E	C3F473	<FC7E>	175	JP	REQUEST_STATUS	
			176	REQ_KBD_STAT		EQU \$
FC81	C3F4CB	<FC81>	177	JP	REQ_KBD_STAT	
			178	REQ_PR_STAT		EQU \$
FC84	C3F5D2	<FC84>	179	JP	REQ_PR_STAT	
			180	REQ_TAPE_STAT		EQU \$
FC87	C3F5D7	<FC87>	181	JP	REQ_TAPE_STAT	
			182	SCAN_ACTIVE		EQU \$
FC8A	C3F9CB	<FC8A>	183	JP	SCAN_ACTIVE	
			184	SOFT_INIT		EQU \$
FC8D	C3F922	<FC8D>	185	JP	SOFT_INIT	
			186	SOFT_RES_DEV		EQU \$
FC90	C3FA5D	<FC90>	187	JP	SOFT_RES_DEV	
			188	SOFT_RES_KBD		EQU \$
FC93	C3FA51	<FC93>	189	JP	SOFT_RES_KBD	
			190	SOFT_RES_PR		EQU \$
FC96	C3FA55	<FC96>	191	JP	SOFT_RES_PR	
			192	SOFT_RES_TAPE		EQU \$
FC99	C3FA59	<FC99>	193	JP	SOFT_RES_TAPE	
			194	START_PR_BUFF		EQU \$
FC9C	C3F580	<FC9C>	195	JP	START_PR_BUFF	
			196	START_PR_CH		EQU \$
FC9F	C3F56D	<FC9F>	197	JP	START_PR_CH	
			198	START_RD_1_BLOCK		EQU \$
FCA2	C3FAC6	<FCA2>	199	JP	START_RD_1_BLOCK	
			200	START_RD_CH_DEF		EQU \$
FCA5	C3FB86	<FCA5>	201	JP	START_RD_CH_DEV	
			202	START_RD_KBD		EQU \$
FCA8	C3F4D0	<FCA8>	203	JP	START_RD_KBD	
			204	START_WR_1_BLOCK		EQU \$
FCA8	C3FAFF	<FCA8>	205	JP	START_WR_1_BLOCK	
			206	START_WR_CH_DEV		EQU \$
FCAE	C3F8C2	<FCAE>	207	JP	START_WR_CH_DEV	
			208	SYNC		EQU \$
FCB1	C3F970	<FCB1>	209	JP	SYNC	
			210	WR_1_BLOCK		EQU \$
FCB4	C3FAB2	<FCB4>	211	JP	WR_1_BLOCK	
			212	WR_CH_DEV		EQU \$
FCB7	C3FB75	<FCB7>	213	JP	WR_CH_DEV	

LOCATION OBJECT CODE LINE SOURCE LINE

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		215	FILE MANAGER ENTRIES
		216	
		217	
		218	FMGR_INIT EQU \$
FCBA	C3EEEA	219	JP FMGR_INIT EQU \$
		220	INIT_TAPE_DIR EQU \$
FCBD	C3F323	221	JP INIT_TAPE_DIR EQU \$
		222	OPEN_FILE EQU \$
FCCO	C3EA00	223	JP OPEN_FILE EQU \$
		224	CLOSE_FILE EQU \$
FCC3	C3EB04	225	JP CLOSE_FILE EQU \$
		226	RESET_FILE EQU \$
FCC6	C3EB6C	227	JP RESET_FILE EQU \$
		228	MAKE_FILE EQU \$
FCC9	C3E690	229	JP MAKE_FILE EQU \$
		230	QUERY_FILE EQU \$
FCCC	C3E61B	231	JP QUERY_FILE EQU \$
		232	SET_FILE EQU \$
FCCF	C3E651	233	JP SET_FILE EQU \$
		234	READ_FILE EQU \$
FCD2	C3EC17	235	JP READ_FILE EQU \$
		236	WRITE_FILE EQU \$
FCD5	C3ED8F	237	JP WRITE_FILE EQU \$
		238	SET_DATE EQU \$
FCD8	C3EEC5	239	JP SET_DATE EQU \$
		240	GET_DATE EQU \$
FCD8	C3EED4	241	JP GET_DATE EQU \$
		242	RENAME_FILE EQU \$
FCDE	C3F10F	243	JP RENAME_FILE EQU \$
		244	DELETE_FILE EQU \$
FCE1	C3F14E	245	JP DELETE_FILE EQU \$
		246	RD_DEV_DEP_STAT EQU \$
FCE4:	C3F488	247	JP RD_DEV_DEP_STAT EQU \$
		248	GOTO_WP EQU \$
FCE7	C3FA94	249	JP GOTO_WP EQU \$
		250	READ_EOS EQU \$
FCEA	C3FA90	251	JP READ_EOS EQU \$
		252	TRIM_FILE EQU \$
FCED	C3F241	253	JP TRIM_FILE EQU \$
		254	CHECK_FCB EQU \$
FCFO	C3F089	255	JP CHECK_FCB EQU \$
		256	READ_BLOCK EQU \$
FCF3	C3F17B	257	JP READ_BLOCK EQU \$
		258	WRITE_BLOCK EQU \$
FCF6	C3F1E6	259	JP WRITE_BLOCK EQU \$
		260	MODE_CHECK EQU \$
FCF9	C3F0D9	261	JP MODE_CHECK EQU \$
		262	SCAN_FOR_FILE EQU \$
FCFC	C3EF0B	263	JP SCAN_FOR_FILE EQU \$
		264	FILE_QUERY EQU \$
FCFF	C3E61B	265	JP FILE_QUERY EQU \$
		266	POSIT_FILE EQU \$
FD02	C3F442	267	JP POSIT_FILE EQU \$
		268	EOS_1 EQU \$
FD05	C3F442	269	JP EOS_1 EQU \$
		270	EOS_2 EQU \$
FD08	C3F	271	JP EOS_2 EQU \$

FILE: L...S:EDS_TF HEWLETT-PACKARD: EDS_COMM (c) Coleco 15- Confidential Sat, 8 Sep 1984, 23:38 .E 7

LOCATION OBJECT CODE LINE SOURCE LINE

<FDO8>	272	_EOS_3		EQU \$
FD08 C3F442	273	_UP	__EOS_3	
<FDOE>	274	_CV_A		EQU \$
FD0E C3F442	275	_UP	__CV_A	

LOCATION	OBJECT CODE	LINE	SOURCE	LINE	EQU \$
	<FD11>	277	PORT_COLLECTION		
FD11	C3E191	278	JP	PORT_COLLECTION	EQU \$
	<FD14>	279	SWITCH_MEM		
FD14	C3E185	280	JP	SWITCH_MEM	EQU \$
	<FD17>	281	PUT_ASCII		
FD17	C3E153	282	JP	PUT_ASCII	EQU \$
		283			
		284	micro OS7 entries		
		285			
		286			
		287			
	<FD1A>	288	WRITE_VRAM		
FD1A	C3E000	289	JP	WRITE_VRAM	EQU \$
	<FD1D>	290	READ_VRAM		
FD1D	C3E01A	291	JP	READ_VRAM	EQU \$
	<FD20>	292	WRITE_REGISTER		
FD20	C3E034	293	JP	WRITE_REGISTER	EQU \$
	<FD23>	294	READ_REGISTER		
FD23	C3E04F	295	JP	READ_REGISTER	EQU \$
	<FD26>	296	FILL_VRAM		
FD26	C3E059	297	JP	FILL_VRAM	EQU \$
	<FD29>	298	INIT_TABLE		
FD29	C3E066	299	JP	INIT_TABLE	EQU \$
	<FD2C>	300	PUT_VRAM		
FD2C	C3E0C9	301	JP	PUT_VRAM	EQU \$
	<FD2F>	302	GET_VRAM		
FD2F	C3E0CF	303	JP	GET_VRAM	EQU \$
	<FD32>	304	CALC_OFFSET		
FD32	C3E10A	305	JP	CALC_OFFSET	EQU \$
	<FD35>	306	PX_TO_PTRN_POS		
FD35	C3E129	307	JP	PX_TO_PTRN_POS	EQU \$
	<FD38>	308	LOAD_ASCII		
FD38	C3E149	309	JP	LOAD_ASCII	EQU \$
	<FD3B>	310	WR_SPR_ATTRIBUTE		
FD3B	C3E1C5	311	JP	WR_SPR_ATTRIBUTE	EQU \$
	<FD3E>	312	POLLER		
FD3E	C3E253	313	JP	POLLER	EQU \$
	<FD41>	314	UPDATE_SPINNER		
FD41	C3E2A4	315	JP	UPDATE_SPINNER	EQU \$
	<FD44>	316	DECLSN		
FD44	C3E355	317	JP	DECLSN	EQU \$
	<FD47>	318	DECM5N		
FD47	C3E35F	319	JP	DECM5N	EQU \$
	<FD4A>	320	MSNTOLSN		
FD4A	C3E369	321	JP	MSNTOLSN	EQU \$
	<FD4D>	322	ADD816		
FD4D	C3E374	323	JP	ADD816	EQU \$
	<FD50>	324	SOUND_INIT		
FD50	C3E3AB	325	JP	SOUND_INIT	EQU \$
	<FD53>	326	TURN_OFF_SOUND		
FD53	C3E3D1	327	JP	TURN_OFF_SOUND	EQU \$
	<FD56>	328	PLAY_IT		
FD56	C3E3E7	329	JP	PLAY_IT	EQU \$
	<FD59>	330	SOUNDS		
FD59	C3E406	331	JP	SOUNDS	EQU \$
	<FD5C>	332	EFFECT_OVER		
FD5C	C3E411	333	JP	EFFECT_OVER	EQU \$

: same as wr_spr_rm_tbi

: not globalized in os7?

: equals calls to play_songs and sound_man

LOCATION OBJECT CODE LINE SOURCE LINE

```
335 :  
336 : NOTE: See INTERRUPT_VECTORS  
337 :  
338 : ORG EOS_GLB_TBL  
339 : :GLB VECTOR_08H  
340 : EQU $+0  
341 : :GLB INT_VCTR_TBL  
342 : EQU VECTOR_08H  
343 : :GLB VECTOR_10H  
344 : EQU $+3  
345 : :GLB VECTOR_18H  
346 : EQU $+6  
347 : :GLB VECTOR_20H  
348 : EQU $+9  
349 : :GLB VECTOR_28H  
350 : EQU $+12  
351 : :GLB VECTOR_30H  
352 : EQU $+15  
353 : :GLB VECTOR_38H  
354 : EQU $+18  
355 : :GLB VECTOR_66H  
356 : EQU $+21
```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
358	:		NOTE: See SWITCH_TABLE	
359	:			
360	:			
361	:	MEM_CNFG00	:GLB	MEM_CNFG00
362	:	SWITCH_TABLE	EQU	SWITCH_TABLE
363	:	MEM_CNFG00	:GLB	MEM_CNFG00
364	:	SWITCH_TABLE	EQU	MEM_CNFG01
365	:	MEM_CNFG01	:GLB	MEM_CNFG01
366	:	MEM_CNFG01	EQU	MEM_CNFG02
367	:	MEM_CNFG02	:GLB	MEM_CNFG02
368	:	MEM_CNFG02	EQU	MEM_CNFG03
369	:	MEM_CNFG03	:GLB	MEM_CNFG03
370	:	MEM_CNFG03	EQU	MEM_CNFG04
371	:	MEM_CNFG04	:GLB	MEM_CNFG04
372	:	MEM_CNFG04	EQU	MEM_CNFG05
373	:	MEM_CNFG05	:GLB	MEM_CNFG05
374	:	MEM_CNFG05	EQU	MEM_CNFG06
375	:	MEM_CNFG06	:GLB	MEM_CNFG06
376	:	MEM_CNFG06	EQU	MEM_CNFG07
377	:	MEM_CNFG07	:GLB	MEM_CNFG07
378	:	MEM_CNFG07	EQU	MEM_CNFG08
379	:	MEM_CNFG08	:GLB	MEM_CNFG08
380	:	MEM_CNFG08	EQU	MEM_CNFG09
381	:	MEM_CNFG09	:GLB	MEM_CNFG09
382	:	MEM_CNFG09	EQU	MEM_CNFG0A
383	:	MEM_CNFG0A	:GLB	MEM_CNFG0A
384	:	MEM_CNFG0A	EQU	MEM_CNFG0B
385	:	MEM_CNFG0B	:GLB	MEM_CNFG0B
386	:	MEM_CNFG0B	EQU	MEM_CNFG0C
387	:	MEM_CNFG0C	:GLB	MEM_CNFG0C
388	:	MEM_CNFG0C	EQU	MEM_CNFG0D
389	:	MEM_CNFG0D	:GLB	MEM_CNFG0D
390	:	MEM_CNFG0D	EQU	MEM_CNFG0E
391	:	MEM_CNFG0E	:GLB	MEM_CNFG0E
392	:	MEM_CNFG0E	EQU	MEM_CNFG0F
393	:	MEM_CNFG0F	:GLB	MEM_CNFG0F
394	:	MEM_CNFG0F	EQU	MEM_CNFG0F

LOCATION OBJECT CODE LINE SOURCE LINE

```

396 ;
397 ;
398 ;
399 ;
400 ; MEM_SWITCH_PORT EQU :GLB MEM_SWITCH_PORT
401 ; :GLB $+40
402 ; PORT_TABLE EQU :GLB PORT_TABLE
403 ; :GLB MEM_SWITCH_PORT
404 ; NET_RESET_PORT EQU :GLB NET_RESET_PORT
405 ; :GLB $+41
406 ; VDP_CTRL_PORT EQU :GLB VDP_CTRL_PORT
407 ; :GLB $+42
408 ; VDP_DATA_PORT EQU :GLB VDP_DATA_PORT
409 ; :GLB $+43
410 ; CONTROLLER_O_PORT EQU :GLB CONTROLLER_O_PORT
411 ; :GLB $+44
412 ; CONTROLLER_1_PORT EQU :GLB CONTROLLER_1_PORT
413 ; :GLB $+45
414 ; STROBE_SET_PORT EQU :GLB STROBE_SET_PORT
415 ; :GLB $+46
416 ; STROBE_RESET_PORT EQU :GLB STROBE_RESET_PORT
417 ; :GLB $+47
418 ; SOUNDPORT EQU :GLB SOUNDPORT
419 ; :GLB $+48

```

NOTE: See PORT_TABLE

```

LOCATION OBJECT CODE LINE SOURCE LINE
420
421 :RAM DEFINITIONS FOR EOS
422
423 :REV 0 (V/D 8-24-83)
424
425
426 .....
427
428 ORG EOS_GLB_RAM
429
430 .....
431
432 :GLB CLEAR_RAM_START
433
434 CLEAR_RAM_START:
435
436 :
437 : EOS revision number initialized by EOS_START
438 :
439 :GLB REV_NUM
440 REV_NUM
441 DEFS 1
442
443
444 IF .NT.SUPERGAME
445 :
446 : OS7 equivalent global data structures
447 :
448 :GLB VDP_MODE_WORD
449 VDP_MODE_WORD
450 DEFS 2
451 :GLB VDP_STATUS_BYTE
452 VDP_STATUS_BYTE
453 DEFS 1
454 :GLB VRAM_ADDR_TABLE
455 VRAM_ADDR_TABLE
456 DEFS 2
457 :GLB SPRITEATTRIBL
458 SPRITEATTRIBL
459 DEFS 2
460 :GLB SPRITEGENTBL
461 SPRITEGENTBL
462 DEFS 2
463 :GLB PATTRNAMETBL
464 PATTRNAMETBL
465 DEFS 2
466 :GLB PATTRNGENTBL
467 PATTRNGENTBL
468 DEFS 2
469 :GLB COLORTABLE
470 COLORTABLE
471 DEFS 2
472 ELSE
473 DEFS 2*6+1
474 ENDIF
475 :GLB CUR_BANK
476 CUR_BANK
477 DEFS 1
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
471 ;
472 ; EOS global data structures
473 ;
474 ; GLB DEFAULT_BT_DEV
475 ; GLB CURRENT_DEV
476
477 DEFAULT_BT_DEV:
478 CURRENT_DEV:
479 DEFS 1
480
481 ; GLB CURRENT_PCB
482
483 CURRENT_PCB:
484 DEFS 2
485 IF .NT.SUPERGAME
486 ; GLB DEVICE_ID
487 ENDIF
488 DEVICE_ID:
489 DEFS 1
490
491 ; GLB FILE_NAME_ADDR
492 FILE_NAME_ADDR
493 DEFS 2
494
495 ; GLB KEYBOARD_BUFFER
496
497 KEYBOARD_BUFFER:
498 DEFS 1
499
500
501 ; GLB PRINT_BUFFER
502
503 PRINT_BUFFER:
504 DEFS 16
505
506 ; GLB SECTORS_TO_INIT,SECTOR_NO
507
508 SECTORS_TO_INIT:
509 DEFS 1
510
511 SECTOR_NO:
512 DEFS 4
513
514 ; GLB DCB_IMAGE
515
516 DCB_IMAGE:
517 DEFS 21
518
519 ; GLB QUERY_BUFFER
520 QUERY_BUFFER
521 DEFS DIR_ENT_LENGTH
522
523 ; GLB FCB_BUFFER
524 FCB_BUFFER
525 DEFS DIR_ENT_LENGTH
526
527 ; GLB FILE_COUNT,MOD_FILE_COUNT

```

; HOLDS THE START ADDRESS OF THE PCB

; SAVE DEVICE ID

; HOLDS THE KEY THAT IS READ FROM THE
; KEYBOARD

; HOLDS THE STRING TO BE PRINTED

SECTORS_TO_INIT,SECTOR_NO

FILE_COUNT,MOD_FILE_COUNT

LOCATION OBJECT CODE LINE SOURCE LINE

```

FDD4 528 FILE_COUNT
FDD4 529 DEFS 1
530
FDD5 531 MOD_FILE_COUNT
FDD5 532 DEFS 1
533
FDD6 534 ;GLB RETRY_COUNT,FILE_NUMBR
FDD6 535 RETRY_COUNT 1
536 DEFS
537
FDD7 538 FILE_NUMBR
FDD7 539 DEFS 1
540
FDD8 541 ;GLB FILENAME_CMPS
FDD8 542 FILENAME_CMPS 1
543 DEFS
544
FDD9 545 ;GLB DIR_BLOCK_NO,FOUND_AVAIL_ENT
FDD9 546 DIR_BLOCK_NO
FDD9 547 DEFS 2
548
FDD8 549 FOUND_AVAIL_ENT
FDD8 550 DEFS 1
551
FDDC 552 ;GLB VOL_BLK_SZ,BLK_STRT_PTR
FDDC 553 BLK_STRT_PTR:
FDDC 554 VOL_BLK_SZ:
FDDC 555 DEFS 4
556
FDE3 557 ; FILE MANAGER RAM STORAGE
558
FDE0 559 ;GLB EOS_YEAR,EOS_MONTH,EOS_DAY
FDE1 560 DEFS 1
FDE1 561 EOS_MONTH :FMGR'S DATE STORAGE
FDE2 562 DEFS 1
FDE2 563 DEFS 1
564
FDE3 564 FMGR_DIR_ENT
FDE3 565 DEFS FMGR_DIR_ENT
FDE3 566 DEFS DIR_ENT_LENGTH ;PLACE FOR FMGR TO PUT A DIR. ENTRY
567
FDFD 568 FCB_HEAD_ADDR
FDFD 569 DEFS FCB_HEAD_ADDR,FCB_DATA_ADDR
FDFD 570 DEFS 2
FDFD 571 DEFS 2 ;POINTER TO START OF FCB HEADS
FDFD 572 DEFS 2 ;POINTER TO START OF FCB BUFFERS
573
FE01 574 FNJM
FE01 575 DEFS FNJM,BYTES_REQ,BYTES_TO_GO,USER_BUF
FE02 576 DEFS BUF_START,BUF_END,BLOCKS_REQ
FE04 577 DEFS USER_NAME,START_BLOCK
FE06 578 DEFS 1 ;FILE NUMBER GIVEN TO READ/WRITE ROUTINES
FE08 579 DEFS 2 ;NUMBER OF BYTES REQUESTED BY CALLER
FE08 580 DEFS 2 ;NUMBER OF BYTES STILL TO GIVE TO CALLER
FE08 581 DEFS 2 ;ADDRESS OF CALLER'S BUFFER
FE08 582 DEFS 2 ;ADDRESS OF MY OWN BUFFER
FE0C 583 DEFS 2 ;ADDRESS OF END + 1 OF MY BUFFER
FE10 584 DEFS 4 ;NUMBER OF BLOCKS REQUESTED (MAKE_FILE)
FE12 585 DEFS 2 ;POINTER TO USER'S NAME STRING
586 DEFS 4 ;START_BLOCK # OF A FILE
587
588 ;GLB NEW_HOLE_START,NEW_HOLE_SIZE
589

```

LOCATION	OBJECT	CODE	LINE	SOURCE	LINE
FE16			585		
FE16			586	NEW_HOLE_START	
			587	DEFS	4
			588		
FE1A			589	NEW_HOLE_SIZE:	
FE1A			590	DEFS	2
			591		
FE1C			592	STACK_START:	
FE1C			593	DEFS	60
			594		
			595	:GLB	EDS_STACK
FE58			596	EDS_STACK:	

LOCATION	OBJECT CODE LINE	SOURCE LINE
598	:	
599	:GLB	SAVED_COUNT
600	:GLB	SPIN_SWO_CT
601	:GLB	SPIN_SWI_CT
602	:GLB	PERSONAL_DEBOUNCE_TABLE
603	:GLB	TEMP_STACK
604	:	
605	:GLB	PTR_TO_LST_OF_SND_ADDRS
606	:GLB	PTR_TO_S_ON_0
607	:GLB	PTR_TO_S_ON_1
608	:GLB	PTR_TO_S_ON_2
609	:GLB	PTR_TO_S_ON_3
610	:GLB	SAVE_CTRL
611	:	

: controller data area
: THESE TWO BYTES MUST BE IN THIS ORDER!!!

: used by put_ascii when bank switching
: sound data areas

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		613	:SAVED_COUNT	DEFS 2
		614	:	
		615	: controller data area	
		616	:	
FE58		617	SPIN_SWO_CT	DEFS 1
FE59		618	SPIN_SW1_CT	DEFS 1
FE5A		619	PERSONAL_DEBOUNCE_TABLE	DEFS 12
FE62		620	TEMP_STACK	EQU \$
	<FE6E>	621	TEMP_STACK	EQU \$
		622	:	
		623	: sound data areas	
		624	:	
		625	:	
FE6E		626	PTR_TO_LST_OF_SMD_ADDRS	DEFS 2
FE70		627	PTR_TO_S_ON_0	DEFS 2
FE72		628	PTR_TO_S_ON_1	DEFS 2
FE74		629	PTR_TO_S_ON_2	DEFS 2
FE76		630	PTR_TO_S_ON_3	DEFS 2
FE78		631	SAVE_CTRL	DEFS 1
		632	:	

THESE TWO BYTES MUST BE IN THIS ORDER!!!
 OR
 :USED BY TEMP_STACK
 :Used by put_ascll when bank switching

LOCATION OBJECT CODE LINE SOURCE LINE

```

634      : data area for CONS_OUT
635      :
636      :
637      : OLDCHAR_
638      : GLB X_MIN
639      : GLB X_MAX
640      : GLB Y_MIN
641      : GLB Y_MAX
642      : GLB LINEBUFFER_
643      : GLB NUM_LINES_
644      : GLB NUM_COLUMNS
645      : GLB UPPER_LEFT
646      : GLB PTRN_NAME_TBL
647      : GLB CURSOR
648      :
649      : OLDCHAR_
650      : GLB X_MIN
651      : GLB X_MAX
652      : GLB Y_MIN
653      : GLB Y_MAX
654      : GLB LINEBUFFER_
655      : GLB NUM_LINES_
656      : GLB NUM_COLUMNS
657      : GLB UPPER_LEFT
658      : GLB PTRN_NAME_TBL
659      : GLB CURSOR
660      :
661      : GLB CLEAR_RAM_SIZE
662      :
663      : CLEAR_RAM_SIZE      EQU      ($-CLEAR_RAM_START)

```

:Storage of char under cursor
:Absolute X coord of window

<0147>

FE79
FE7A
FE7B
FE7C
FE7D
FE7E
FE9F
FEA0
FEA1
FEA3
FEA5

LOCATION OBJECT CODE LINE SOURCE LINE

```

665 :
666 : this is the PCB/DCB area
667 :
668 :
669 :
670 : ORG EOS_PCB_DCB
671 :
672 :
673 :
674 : GLB PCB
675 :
FECO 676 PCB: DEFS P_SIZE : PCB
FECO 677 :
FECO 678 :
FECO 679 DCBS: DEFS 15*D_SIZE : 1 DCB PER NETWORK ADDRESS
FECO 680 :
FECO 681 :
FFFF 682 RESERVED_BYTE:
FFFF 683 DEFS 1 : SAVED IN CASE WE DO FAST DMA HERE
684 :
685 :
686 :
687 :

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		689	NAME ^Rev 00 - RPD^	
		690		
		691		
		692	
		693		
		694	ORG EDS_GLB_TBL	
		695		
		696	
		697		
		698	NOTE: these are defined global in EDS_COMN	
		699		
		700		
		701	:GLB VECTOR_08H	
		702	:GLB VECTOR_10H	
		703	:GLB VECTOR_18H	
		704	:GLB VECTOR_20H	
		705	:GLB VECTOR_28H	
		706	:GLB VECTOR_30H	
		707	:GLB VECTOR_38H	
		708	:GLB VECTOR_66H	
		709	VECTOR_08H	RET
		710		NOP
		711		NOP
		712	VECTOR_10H	RET
		713		NOP
		714		NOP
		715	VECTOR_18H	RET
		716		NOP
		717		NOP
		718	VECTOR_20H	RET
		719		NOP
		720		NOP
		721	VECTOR_28H	RET
		722		NOP
		723		NOP
		724	VECTOR_30H	RET
		725		NOP
		726		NOP
		727	VECTOR_38H	RET
		728		NOP
		729		NOP
		730	VECTOR_66H	RET
		731		NOP
FBFF	C9			
FC00	00			
FC01	00			
FC02	C9			
FC03	00			
FC04	00			
FC05	C9			
FC06	00			
FC07	00			
FC08	C9			
FC09	00			
FC0A	00			
FC0B	C9			
FC0C	00			
FC0E	C9			
FC0F	00			
FC10	00			
FC11	C9			
FC12	00			
FC13	00			
FC14	ED45			
FC16	00			

```

:GLB VECTOR_08H
:GLB VECTOR_10H
:GLB VECTOR_18H
:GLB VECTOR_20H
:GLB VECTOR_28H
:GLB VECTOR_30H
:GLB VECTOR_38H
:GLB VECTOR_66H

```

```

:rst 8
:rst 10
:rst 18
:rst 20
:rst 28
:rst 30
:rst 38
:rml vector

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

734 :
735 :
736 : NOTE: these are defined global in EOS_COMN
737 :
738 : SWITCH_TABLE
739 : MEM_CNFG00
740 : MEM_CNFG01
741 : MEM_CNFG02
742 : MEM_CNFG03
743 : MEM_CNFG04
744 : MEM_CNFG05
745 : MEM_CNFG06
746 : MEM_CNFG07
747 : MEM_CNFG08
748 : MEM_CNFG09
749 : MEM_CNFG0A
750 : MEM_CNFG0B
751 : MEM_CNFG0C
752 : MEM_CNFG0D
753 : MEM_CNFG0E
754 : MEM_CNFG0F
755 :
756 <0000> LO BOOT_ROM_N_ALPHA_MASK EQU 0000B
757 <0001> LO_INTRINSIC_RAM_0_TO_7FFFH_MASK EQU 0001B
758 <0002> LO_EXPANSION_0_TO_7FFFH_MASK EQU 0010B
759 <0003> LO_OS7_N_INTRINSIC_2000H_7FFFH_MASK EQU 0011B
760 :
761 <0000> HI_INTRINSIC_8000H_TO_OFFFHH_MASK EQU 0000B
762 <0004> HI_AUXILIARY_SLOT_8000H_OFFFHH EQU 0100B
763 <000B> HI_EXPANSION_8000H_TO_OFFFHH_MASK EQU 1000B
764 <000C> HI_COLECOVISION_CARTRIDGE_SLOT EQU 1100B
765 :

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
FC17		767	SWITCH_TABLE:	
		768		
FC17		769		
FC17 00		770	MEM_CNFG00	
		771		DEFB HI_INTRINSIC_8000H_TO_OFFFFH_MASK.OR.LO_BOOT_ROM_N_ALPHA_MASK
		772		
FC18		773	MEM_CNFG01	
FC18 01		774		DEFB HI_INTRINSIC_8000H_TO_OFFFFH_MASK.OR.LO_INTRINSIC_RAM_O_TO_7FFFH_MASK
		775		
FC19		776	MEM_CNFG02	
FC19 02		777		DEFB HI_INTRINSIC_8000H_TO_OFFFFH_MASK.OR.LO_EXPANSION_O_TO_7FFFH_MASK
		778		
FC1A		779	MEM_CNFG03	
FC1A 03		780		DEFB HI_INTRINSIC_8000H_TO_OFFFFH_MASK.OR.LO_057_N_INTRINSIC_2000H_7FFFH_MASK
		781		
FC1B		782	MEM_CNFG04	
FC1B 04		783		DEFB HI_AUXILLIARY_SLOT_8000H_OFFFFH.OR.LO_BOOT_ROM_N_ALPHA_MASK
		784		
FC1C		785	MEM_CNFG05	
FC1C 05		786		DEFB HI_AUXILLIARY_SLOT_8000H_OFFFFH.OR.LO_INTRINSIC_RAM_O_TO_7FFFH_MASK
		787		
FC1D		788	MEM_CNFG06	
FC1D 06		789		DEFB HI_AUXILLIARY_SLOT_8000H_OFFFFH.OR.LO_EXPANSION_O_TO_7FFFH_MASK
		790		
FC1E		791	MEM_CNFG07	
FC1E 07		792		DEFB HI_AUXILLIARY_SLOT_8000H_OFFFFH.OR.LO_057_N_INTRINSIC_2000H_7FFFH_MASK
		793		
FC1F		794	MEM_CNFG08	
FC1F 08		795		DEFB HI_EXPANSION_8000H_TO_OFFFFH_MASK.OR.LO_BOOT_ROM_N_ALPHA_MASK
		796		
FC20		797	MEM_CNFG09	
FC20 09		798		DEFB HI_EXPANSION_8000H_TO_OFFFFH_MASK.OR.LO_INTRINSIC_RAM_O_TO_7FFFH_MASK
		799		
FC21		800	MEM_CNFG0A	
FC21 0A		801		DEFB HI_EXPANSION_8000H_TO_OFFFFH_MASK.OR.LO_EXPANSION_O_TO_7FFFH_MASK
		802		
FC22		803	MEM_CNFG0B	
FC22 0B		804		DEFB HI_EXPANSION_8000H_TO_OFFFFH_MASK.OR.LO_057_N_INTRINSIC_2000H_7FFFH_MASK
		805		
FC23		806	MEM_CNFG0C	
FC23 0C		807		DEFB HI_COLECOVISION_CARTRIDGE_SLOT.OR.LO_BOOT_ROM_N_ALPHA_MASK
		808		
FC24		809	MEM_CNFG0D	
FC24 0D		810		DEFB HI_COLECOVISION_CARTRIDGE_SLOT.OR.LO_INTRINSIC_RAM_O_TO_7FFFH_MASK
		811		
FC25		812	MEM_CNFG0E	
FC25 0E		813		DEFB HI_COLECOVISION_CARTRIDGE_SLOT.OR.LO_EXPANSION_O_TO_7FFFH_MASK
		814		
FC26		815	MEM_CNFG0F	
FC26 0F		816		DEFB HI_COLECOVISION_CARTRIDGE_SLOT.OR.LO_057_N_INTRINSIC_2000H_7FFFH_MASK

LOCATION OBJECT CODE LINE SOURCE LINE

```

819 :
820 :
821 : NOTE: these are defined global in EOS_COMN
822 :
823 : PORT_TABLE
824 : :GLB MEM_SWITCH_PORT
825 : :GLB NET_RESET_PORT
826 : :GLB VDP_CTRL_PORT
827 : :GLB VDP_DATA_PORT
828 : :GLB CONTROLLER_0_PORT
829 : :GLB CONTROLLER_1_PORT
830 : :GLB STROBE_SET_PORT
831 : :GLB STROBE_RESET_PORT
832 : :GLB SOUNDPORT
833 :
834 :
835 : *****
836 : *** VDP ports must remain next to each other ***
837 : *** and in CTRL -> DATA order ***
838 : *****
839 : PORT_TABLE:
840 :
841 : MEM_SWITCH_PORT DEFB 07FH ;MEM_SWITCH_PORT defined here
842 :
843 :
844 : NET_RESET_PORT DEFB 03FH ;Adam_net reset port defined here
845 :
846 :
847 : VDP_CTRL_PORT DEFB 08FH ;VDP ctrl port 01D43H
848 :
849 :
850 : VDP_DATA_PORT DEFB 08EH ;VDP data port 01D47H
851 :
852 :
853 : CONTROLLER_0_PO DEFB 0FCH ;Controller 0 0114BH
854 :
855 :
856 : CONTROLLER_1_PO DEFB 0FFH ;Controller 1 01151H
857 :
858 :
859 : STROBE_SET_PORT DEFB 080H ;Controller strobe set 01157H
860 :
861 :
862 : STROBE_RESET_PO DEFB 0COH ;Controller strobe reset 01168H
863 :
864 :
865 : SOUNDPORT DEFB 0FFH ;Sound port 0018EH
866 :
867 :
868 : *****
869 : *****
870 : *****
871 :

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

874 ;
875 ;  globals routines defined
876 ;
877 ;
878 ;   WRITE_VRAM
879 ;   READ_VRAM
880 ;   WRITE_REGISTER
881 ;   READ_REGISTER
882 ;   FILL_VRAM
883 ;   INIT_TABLE
884 ;   PUT_VRAM
885 ;   GET_VRAM
886 ;   CALC_OFFSET
887 ;   PX_TO_PTRN_POS
888 ;   LOAD_ASCII
889 ;   PUT_ASCII
890 ;   WR_SPR_ATTRIBUTE
891 ;
892 ;   DECODER
893 ;   POLLER
894 ;   UPDATE_SPINNER
895 ;
896 ;   DECLSN
897 ;   DECMNSN
898 ;   MSNTOLSN
899 ;   ADDR16
900 ;   SOUND_INIT
901 ;   TURN_OFF_SOUND
902 ;   PLAY_IT
903 ;   SOUNDS
904 ;   EFFECT_OVER
905 ;
906 ;   SWITCH_MEM
907 ;   PORT_COLLECTION

```

:VDP support routines

:controller support routines

:sound chip support routines

:memory bank switch support routine

:STORE THE PORT ADDRESSES FROM OS_7

LOCATION OBJECT CODE LINE SOURCE LINE

```

908 ;
909 ; external data areas used
910 ;
911 :EXT VDP_MODE_WORD
912 :EXT VDP_STATUS_BYTE
913 :EXT VRAM_ADDR_TABLE
914 :EXT SPRITEATTRIBL
915 :*** :EXT SPRITEGENTBL
916 :*** :EXT PATRNAMETBL
917 :EXT :EXT PATRNGENTBL
918 :*** :EXT COLORTABLE
919 :EXT :EXT CUR_BANK
920
921
922 :EXT SPIN_SMO_CT
923 :*** :EXT SPIN_SWI_CT
924 :EXT :EXT PERSONAL_DEBOUNCE_TABLE
925 :EXT :EXT TEMP_STACK
926
927 :EXT PTR_TO_LST_OF_SMD_ADDRS
928 :EXT :EXT PTR_TO_S_ON_0
929 :EXT :EXT PTR_TO_S_ON_1
930 :EXT :EXT PTR_TO_S_ON_2
931 :EXT :EXT PTR_TO_S_ON_3
932 :EXT :EXT SAVE_CTRL
933 ;
934 ; Port address and bank select externals follow
935 ;
936 :EXT PORT_TABLE
937 :EXT MEM_SWITCH_PORT
938 :*** :EXT NET_RESET_PORT
939 :EXT :EXT VDP_CTRL_PORT
940 :*** :EXT VDP_DATA_PORT
941 :EXT :EXT CONTROLLER_0_PORT
942 :*** :EXT CONTROLLER_1_PORT
943 :EXT :EXT STROBE_SET_PORT
944 :*** :EXT STROBE_RESET_PORT
945 :EXT :EXT SOUNDPORT
946
947 :*** :EXT SWITCH_TABLE
948 :EXT :EXT MEM_CNFG00
949 :*** :EXT MEM_CNFG01
950 :*** :EXT MEM_CNFG02
951 :*** :EXT MEM_CNFG03
952 :*** :EXT MEM_CNFG04
953 :*** :EXT MEM_CNFG05
954 :*** :EXT MEM_CNFG06
955 :*** :EXT MEM_CNFG07
956 :*** :EXT MEM_CNFG08
957 :*** :EXT MEM_CNFG09
958 :*** :EXT MEM_CNFG0A
959 :*** :EXT MEM_CNFG0B
960 :*** :EXT MEM_CNFG0C
961 :*** :EXT MEM_CNFG0D
962 :*** :EXT MEM_CNFG0E
963 :*** :EXT MEM_CNFG0F

```

: controller data area
: THESE TWO BYTES MUST BE IN THIS ORDER!!!
: Used by put_ascii when bank switching
: sound data areas

LOCATION OBJECT CODE LINE SOURCE LINE

```
965 :  
966 : externals used  
967 :  
968 : EXT WRITE_REGISTER  
969 : EXT INIT_TABLE  
970 : EXT SWITCH_MEM  
971 : EXT PUT_VRAM  
972 : EXT WRITE_VRAM  
973 : EXT READ_VRAM  
974 :  
975 : local equates  
976 :  
<0102> 977 LOC_IN_ALPHA EQU 0102H  
978 : pointer to ascii gens in rev 28+ alpha
```

```

LOCATION OBJECT CODE LINE SOURCE LINE
981 ; Offsets into a volume descriptor (the first entry in the directory).
982 VOL_NAME EQU 0 ; LOGICAL VOLUME NAME
983 VOL_DIRSIZE EQU 12 ; *** 7 BITS ONLY *** # BLOCKS IN DIRECTORY
984 VOL_ATTR EQU 12 ; *** TOP BIT ONLY *** SET FOR DELETE PROTECTION
985 VOL_DIR_CHECK EQU 13 ; CONTAINS 4 UNIQUE BYTES FOR DIRECTORY EXISTENCE VERIFICATION
986 VOL_SIZE EQU 17 ; VOLUME SIZE (IN BLOCKS) (4 BYTES)
987 VOL_YEAR EQU 23 ; CREATION DATE -- YEAR
988 VOL_MONTH EQU 24 ; MONTH
989 VOL_DAY EQU 25 ; & DAY
990 VOL_DES_LENGTH EQU 26
991
992 ; Offsets into a directory entry.
993 DIR_NAME EQU 0 ; FILE NAME
994 DIR_ATTR EQU 12 ; FILE ATTRIBUTE BYTE
995 DIR_START_BLOCK EQU 13 ; STARTING BLOCK #
996 DIR_MAX_LENGTH EQU 17 ; TOTAL # BLOCKS ALLOCATED
997 DIR_USED_LENGTH EQU 19 ; # OF BLOCKS USED (FULL + 1 PARTIAL)
998 DIR_LAST_COUNT EQU 21 ; NUMBER OF BYTES IN LAST PARTIAL BLOCK
999 DIR_YEAR EQU 23 ; CREATION DATE -- YEAR
1000 DIR_MONTH EQU 24 ; MONTH
1001 DIR_DAY EQU 25 ; & DAY
1002 DIR_ENT_LENGTH DEFL 26
1003
1004 ENT_PER_BLOCK EQU 1024/26 ; NUMBER OF ENTRIES PER DIR BLOCK
1005
1006 ; Offsets into an FCB header.
1007
1008 ; Copy of DIR entry
1009
1010 FCB_NAME EQU 0 ; FILE NAME
1011 FCB_ATTR EQU 12 ; FILE ATTRIBUTE BYTE
1012 FCB_START_BLOCK EQU 13 ; STARTING BLOCK #
1013 FCB_FIRST_BLOCK EQU FCB_START_BLOCK
1014 FCB_MAX_LENGTH EQU 17 ; TOTAL # BLOCKS ALLOCATED
1015 FCB_USED_LENGTH EQU 19 ; # OF BLOCKS USED (FULL + 1 PARTIAL)
1016 FCB_LAST_COUNT EQU 21 ; NUMBER OF BYTES IN LAST PARTIAL BLOCK
1017
1018 FCB_STORED_BYTES EQU FCB_LAST_COUNT+1 ; NUMBER OF BYTES STORED ON DEVICE
1019 ; ... THE UPPER HALF OF THIS FCB EQU LIST
1020
1021 FCB_DEVICE EQU 26-3 ; NUMBER OF DEVICE CONTAINING FILE
1022 FCB_MODE EQU 27-3 ; FILE MODE (UNUSED, READ, WRITE, APPEND)
1023 FCB_BLOCK EQU 28-3 ; BLOCK NUMBER CURRENTLY IN BUFFER
1024 FCB_LAST_BLOCK EQU 32-3 ; LAST BLOCK NUMBER IN FILE
1025 FCB_POINTER EQU 36-3 ; POINTER INTO BLOCK BUFFER
1026 FCB_LENGTH EQU 38-3
1027
1028 ; Possible FCB modes.
1029 *
1030 * PROTECTED MODE EQUATES
1031 *
1032 MODE_UNUSED EQU 0 ; MANY PEOPLE ASSUME THIS IS ZERO!
1033 MODE_READ EQU MODE_UNUSED+1
1034 MODE_WRITE EQU MODE_READ+1
1035 MODE_UPDATE EQU MODE_WRITE+1
1036 MODE_EXEC EQU MODE_UPDATE+1
1037 MODE_MAX EQU MODE_EXEC+0

```

FILE: EDSABS:EDS_TF HEWLETT-PACKARD: INCLUDE FMGR_EQU5

```

LOCATION OBJECT CODE LINE SOURCE LINE
1038 *
<0005> 1039 MODE_REMAINDER_BIT EQU 5 : INDICATES TO ALLOCATE REST OF TAPE
<0006> 1040 MODE_DIRTY_BIT EQU 6 : TEST BIT 6
<0040> 1041 MODE_DIRTY EQU 0100000008 : INDICATES MODIFIED BUFFER
<0007> 1042 MODE_LAST_BLOCK_BIT EQU 7 : TEST BIT 7
<0080> 1043 MODE_LAST_BLOCK EQU 1000000008 : INDICATES LAST BLOCK OF FILE
<0007> 1044 MODE_MODE EQU 00000111B : BITS TO STORE MODES
1045
1046 : File attribute bits.
<0080> 1047 ATTR_PERMANENT EQU 1000000008
<0040> 1048 ATTR_WRITE_PROT EQU 0100000008
<0020> 1049 ATTR_READ_PROT EQU 0010000008
<0010> 1050 ATTR_USER EQU 0001000008
<0008> 1051 ATTR_SYSTEM EQU 0000100008
<0004> 1052 ATTR_DELETED EQU 0000010008
<0002> 1053 ATTR_DEL_BIT EQU 2
<0002> 1054 ATTR_EXECUTE EQU 0000001008
<0000> 1055 ATTR_HOLE_BIT EQU 0
<0001> 1056 ATTR_HOLE EQU 000000010B
1057
1058 : System-wide file name length.
<000C> 1059 NAME_LENGTH EQU 12
1060
1061 : Numbers of things we have.
<0003> 1062 NUM_FCBS EQU 3 : 1 FOR THE SYSTEM, 2 FOR THE USER

```


LOCATION	OBJECT	CODE	LINE	SOURCE	LINE
	<0001>		1065	DCB NOT FOUND	EQU 1
	<0002>		1066	DCB_BUSY	EQU 2
	<0003>		1067	DCB_IDLE_ERR	EQU 3
			1068		
	<0004>		1069	NO_DATE_ERR	EQU 4
	<0005>		1070	NO_FILE_ERR	EQU 5
	<0006>		1071	FILE_EXISTS_ERR	EQU 6
	<0007>		1072	NO_FCB_ERR	EQU 7
	<0008>		1073	MATCH_ERR	EQU 8
	<0009>		1074	BAD_FNJM_ERR	EQU 9
	<000A>		1075	EOF_ERR	EQU 10
	<000B>		1076	TOO_BIG_ERR	EQU 11
	<000C>		1077	FULL_DIR_ERR	EQU 12
	<000D>		1078	FULL_TAPE_ERR	EQU 13
	<000E>		1079	FILE_NM_ERR	EQU 14
	<000F>		1080	RENAME_ERR	EQU 15
	<0010>		1081	DELETE_ERR	EQU 16
	<0011>		1082	RANGE_ERR	EQU 17
			1083		
	<0012>		1084	CANT_SYNC1	EQU 18
	<0013>		1085	CANT_SYNC2	EQU 19
	<0014>		1086	PRT_ERR	EQU 20
			1087		
	<0015>		1088	RQ_TP_STAT_ERR	EQU 21
	<0016>		1089	DEVICE_DEPD_ERR	EQU 22
	<0017>		1090	PROG_NON_EXIST	EQU 23
	<0018>		1091	NO_DIR_ERR	EQU 24

: THERE WAS NO DCB FOR THE DEVICE REQUESTED.
: DCB IS BUSY
: DCB IS IDLE

:DLS(8/28/83)
:DLS(8/30/83)
:DLS(8/30/83)
:DLS(8/31/83)

: PROGRAM DOES NOT CURRENTLY EXIST
: NO DIRECTORY ON TAPE

LOCATION OBJECT CODE LINE SOURCE LINE

```

1094 ; THESE OUR EQUATES THAT ARE USED BY THE EOS PROGRAMS TO REFERRECE
1095 ; PCB AND DCB INFORMATION
1096
1097
1098 ; PCB EQUATES
1099
1100 P_COM_STAT EQU 0 ; THIS IS THE COMMAND/STATUS BYTE
1101
1102 P_REL_ADDR EQU 1 ; THIS IS THE RELOCATION ADDRESS
1103 P_REL_ADDR_LO EQU P_REL_ADDR+0
1104 P_REL_ADDR_HI EQU P_REL_ADDR+1
1105
1106 P_NUM_DCBS EQU 3 ; THIS IS THE NUMBER OF DCBS DEFINED
1107
1108
1109 P_SIZE DEFL 4 ; THE NUMBER OF BYTES IN THE PCB
1110
1111
1112
1113 ; DCB EQUATES
1114
1115 D_COM_STAT EQU 0 ; THE COMMAND STATUS BYTE
1116
1117 D_BUF_ADDR EQU 1 ; ADDRESS OF THE DATA BUFFER
1118 D_BUF_ADDR_LO EQU D_BUF_ADDR+0
1119 D_BUF_ADDR_HI EQU D_BUF_ADDR+1
1120
1121 D_BUF_LEN EQU 3 ; THE LENGTH OF THE DATA BUFFER
1122 D_BUF_LEN_LO EQU D_BUF_LEN+0
1123 D_BUF_LEN_HI EQU D_BUF_LEN+1
1124
1125 D_SECT_NUM EQU 5 ; THE BLOCK DEVICE SECTOR NUMBER
1126
1127 D_SEC_DEV_ID EQU 9 ; SECONDARY DEVICE ID
1128
1129 D_RET_COUNT EQU 14 ; THE NUMBER OF TIMES A COMMAND WILL
1130 ; BE RETRIED.
1131 D_RET_COUNT_LO EQU D_RET_COUNT+0
1132 D_RET_COUNT_HI EQU D_RET_COUNT+1
1133
1134 D_DEV_ADDR EQU 16 ; THE DEVICE ADDRESS (ID)
1135
1136 D_MAX_MSG_LEN EQU 17 ; THE MAX LENGTH OF A DATA STRING
1137 ; FOR THE DEVICE
1138 D_MAX_MSG_LEN_LO EQU D_MAX_MSG_LEN+0
1139 D_MAX_MSG_LEN_HI EQU D_MAX_MSG_LEN+1
1140
1141 D_DEV_TYPE EQU 19 ; THE DEVICE TYPE, BLOCKED OR CHARACTER
1142
1143 D_STATUS_FLAGS EQU 20 ; DEVICE DEPENDENT STATUS FLAGS
1144
1145 D_SIZE DEFL 21 ; THE NUMBER OF BYTES IN THE DCB
1146
1147
1148
1149 ; DEVICE ID'S FOR THE KEYBOARD, PRINTER, AND TAPE DRIVE
1150

```

<0001>	1151	KEYBOARD_ID	1	KYBD ID	EQU
<0002>	1152	PRINTER_ID	2	PRINTER ID	EQU
<0008>	1153	TAPE_ID	8	TAPE DRIVE ID	EQU
<0002>	1154	ERROR_RETRY	2	MAX RETRY ON ERRORS, READ_BLOCK AND WRITE_BLOCK	EQU
<000F>	1156	MAX_DEV_ADDR	15	HIGHEST POSSIBLE DEVICE ADDRESS	EQU
	1158			ON NETWORK	
	1159				
	1160				
	1161	:PCB_COMMAND EQUATES			
	1162				
<0000>	1164	PCB_IDLE	0	THIS IS AN IDLE STATE	EQU
<0001>	1165	PCB_SYNC1	1	SYNC BYTE 1	EQU
<0081>	1166	PCB_SYNC1_ACK		PCB_SYNC1+80H	EQU
<0002>	1167	PCB_SYNC2	2	SYNC BYTE 2	EQU
<0082>	1168	PCB_SYNC2_ACK		PCB_SYNC2+80H	EQU
<0003>	1171	PCB_SNA	3	SET NEW PCB ADDRESS	EQU
<0083>	1172	PCB_SNA_ACK		PCB_SNA+80H	EQU
<0004>	1174	PCB_RESET	4	RESET ALL NODES	EQU
<0084>	1175	PCB_RESET_ACK		PCB_RESET+80H	EQU
<0005>	1177	PCB_WAIT	5		EQU
<0085>	1178	PCB_WAIT_ACK		PCB_WAIT+80H	EQU
	1180				
	1181				
	1182				
	1183	:DCB_COMMAND EQUATES			
	1184				
<0000>	1185	DCB_IDLE	00		EQU
<0001>	1186	DCB_STATUS	01	REQUEST STATUS	EQU
<0002>	1187	DCB_RESET	02	RESET NODE	EQU
<0003>	1188	DCB_WR	03	WRITE DATA TO DEVICE	EQU
<0004>	1189	DCB_RD	04	READ DATA FROM DEVICE	EQU
	1190				
	1191				
<FECO>	1192	INIT_PCB_ADDR		INITIAL ADDRESS OF THE PCB	EQU
<D390>	1193	FCB_S		FCB HEADER AREA	EQU
<D400>	1194	THREE1K_BKLS		3K FCB DATA AREA	EQU
	1195				
	1196				
	1197				
	1198				
	1199	:GENERAL_USAGE EQUATES FOR USE WITH DCB INFO			
	1200				
<0007>	1201	CMND_COMPLETE_BIT	7	THIS IS THE BIT THAT INDICATES THE	EQU
<0080>	1202	CMND_FIN_STATUS	80H	COMMAND HAS BEEN PROCESSED.	EQU
<008C>	1203	KBD_NAK	8CH	THIS IS THE STATUS OF A COMMAND	EQU
	1204			THAT COMPLETED WITH NO ERRORS	
	1205			INDICATES NO KEY READY	
<0086>	1206	PR_NAK	86H	INDICATES THE PRINTER IS BUSY	EQU

FILE: EOSABS:EOS_TF' HEWLETT-PACKARD: INCLUDE P_DCB_EQU

LOCATION	OBJECT CODE	LINE	SOURCE	LINE	
		1208			
	<0003>	1209	ETX		
		1210			
	<009B>	1211	TIMEOUT		
		1212	:		
		1213	:		
		1214	:		
		1215	:		
		1216	:		
		1217	SKIP		

FC30

EQU 03H ; END OF DATA STRING INDICATOR

EQU 9BH ; DEVICE TIMED OUT

.....

.....

LOCATION OBJECT CODE LINE SOURCE LINE

```

1219 NAME ^Rev 07 - jk1^
1220
1221 De_A_u0S_00 MACRO :Header Rev. 5
1222 .GOTO Ede_A_u0S_00
1223
1224 Project: ADAM, 83-101
1225
1226 *****
1227 ****
1228 **** A_u0S_00 RPD
1229 ****
1230 *****
1231
1232
1233

```

```

Rev History
Rev. Date Name
7 08oct1545 jk1
6 08oct0003 rfj
5 07oct1708 rfj
4 05oct10:40 rfj
3 28sept1317 rfj
2 25sept900a RPD
1 24sept1250p RPD
0 24aug430p RPD

```

```

Change
make __LOAD_ASCII load chars from 0 to 7F
Fixed 100H bug with Bob Greenberg's
algorithm
__EFFECTOVER changed to __EFFECT_OVER
Decoder now saves Acc before getting
interrupt data
Removed excess documentation on Put/Load Ascii
Commented out all unreferenced symbols with ;***
Renamed WR_SPR_NM_TBL to WR_SPR_ATTRIBUTE
globalized __UPDATE_SPINNER
new address for the ASCII table referenced
but not approved
Multy changes after personal review
made ram external
addition of more routines & new port accessing
Initial creation date (READ & WRITE VRAM)

A_u0S_00 stands for ADAM micro Operating System version 0. This file
contains a scaled down model of COLECOVISION OS 7. Primary routines from
OS 7 are duplicated for ADAM application programs use. The functional areas
that are supported by these routines are the VDP, the controllers, the sound
chip, memory bank switch, port collection.

Ede_A_u0S_00 MEND

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

1260 .....
1261 .....
1262 .....
1263 .....
1264 .....
1265 .....
1266 .....
1267 .....
1268 .....
1269 .....
1270 .....
1271 .....
1272 .....
1273 .....
1274 .....
1275 .....
1276 .....
1277 .....
1278 .....
1279 .....
1280 .....
1281 .....
1282 .....
1283 .....
1284 .....
1285 .....
1286 .....
1287 .....
1288 .....
1289 .....
1290 .....

```

```

Name: WRITE_VRAM
Function: Writes to VRAM the contents of the data in a buffer area.
Entry: BC - number of bytes to be written
      DE - starting VRAM address to be written to
      HL - address of buffer containing the data
Exit: None.
Registers used: AF, BC, DE, HL
Size: ROM - 28 bytes
      RAM - 0 bytes

```

```

Comments: This version of WRITE_VRAM is provided for EDS operation.
          NOTE: The 100H bug found in DS 7 has been corrected.
Comparison: left out pascal entry point
            code compacted
            100H bug fixed
            edited for documentation
            reference port table

```

```

E000 C5
E001 EB
E002 CDE1E9
E005 69
E008 C1
E007 EB
E008 79
E009 4B
E00A 50
E00B 14
E00C 47
E00D B7
E00E 2806
E010
E010 EDA3
E012 00
E013 00
E014 20FA
E016
E016 15
E017 20F7
E019 C9

WRITE_VRAM:
PUSH BC
EX DE,HL
CALL SET_WRITE
LD L,C
POP BC
EX DE,HL
LD A,C
LD C,E
LD D,B
INC D
LD B,A
OR A
JR Z,OUT_DEC_HI_BYTE
OUTI
NOP
NOP
JR NZ,OUTPUT_LOOP
OUT_DEC_HI_BYTE:
DEC D
JR NZ,OUTPUT_LOOP
RET

BC has the number of bytes to xfer
HL<--vram addr,DE<--source address
pump the vram addr to the vdp
save C (the data_port_addr)
restore the xfer count
HL<--source address,port in E
Save low order of count in A
And get the port into C
Get HI order count in D
to free up B for low order count
Increment HI count to cover
for 0 in DEC D after low loop
Put low order in B for OUTI
Check if low order is zero the first time
If so then decrement HI count before low loop

DATA_PORT = buffer data write the data to the VDP
for OUTI: [C] <-- [HL], B = B - 1 and HL = HL + 1
delay for slow VDP
until byte count low (B reg) = 0
byte count high = byte count high - 1
until byte count high (A reg) = 0

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

1318 ;
1319 ; READ_VRAM
1320 ;
1321 ; Function: Reads from VRAM and puts the read data into a buffer area.
1322 ;
1323 ; Entry: BC - number of bytes to be read
1324 ; DE - starting VRAM address to be read from
1325 ; HL - address of buffer to receive the read data
1326 ;
1327 ; Exit: None.
1328 ;
1329 ; Registers used: AF, BC, DE, HL
1330 ;
1331 ; Size: ROM - 22 bytes
1332 ; RAM - 0 bytes
1333 ;
1334 ; Comments: This version of READ_VRAM is provided for stand alone operation.
1335 ; NOTE: The 100H bug found in OS 7 has been corrected.
1336 ;
1337 ; Comparison: left out pascal entry point
1338 ; code compacted
1339 ; 100H bug fixed
1340 ; edited for documentation
1341 ;
1342 ; READ_VRAM:
1343 ;
1344 ; BC has the number of bytes to xfer
1345 ; HL<--vram addr,DE<--source address
1346 ; pump the vram addr to the vdp
1347 ; save C (the data_port_addr)
1348 ; restore the xfer count
1349 ; HL<--source address,port in E
1350 ; Save low order of count in A
1351 ; And get the port into C
1352 ; Get HI order count in D
1353 ; to free up B for low order count
1354 ; Increment HI count to cover
1355 ; for 0 in DEC D after low loop
1356 ; Put low order in B for DUTI
1357 ; Check if low order is zero the first time
1358 ; If so then decrement HI count before low loop
1359 ;
1360 ; DATA_PORT = buffer data read the data from to the VDP
1361 ; for INI: [C] <-- [HL], B = B - 1 and HL = HL + 1
1362 ; delay for slow VDP
1363 ;
1364 ; until byte count low (B reg) = 0
1365 ;
1366 ; byte count high = byte count high - 1
1367 ; until byte count high (A reg) = 0
1368 ;
1369 ;
1370 ;
1371 ;
1372 ;
1373 ;
1374 ;
1375 ;
1376 ;
1377 ;
1378 ;
1379 ;
1380 ;
1381 ;
1382 ;
1383 ;
1384 ;
1385 ;
1386 ;
1387 ;
1388 ;
1389 ;
1390 ;
1391 ;
1392 ;
1393 ;
1394 ;
1395 ;
1396 ;
1397 ;
1398 ;
1399 ;
1400 ;
1401 ;
1402 ;
1403 ;
1404 ;
1405 ;
1406 ;
1407 ;
1408 ;
1409 ;
1410 ;
1411 ;
1412 ;
1413 ;
1414 ;
1415 ;
1416 ;
1417 ;
1418 ;
1419 ;
1420 ;
1421 ;
1422 ;
1423 ;
1424 ;
1425 ;
1426 ;
1427 ;
1428 ;
1429 ;
1430 ;
1431 ;
1432 ;
1433 ;
1434 ;
1435 ;
1436 ;
1437 ;
1438 ;
1439 ;
1440 ;
1441 ;
1442 ;
1443 ;
1444 ;
1445 ;
1446 ;
1447 ;
1448 ;
1449 ;
1450 ;
1451 ;
1452 ;
1453 ;
1454 ;
1455 ;
1456 ;
1457 ;
1458 ;
1459 ;
1460 ;
1461 ;
1462 ;
1463 ;
1464 ;
1465 ;
1466 ;
1467 ;
1468 ;
1469 ;
1470 ;
1471 ;
1472 ;
1473 ;
1474 ;
1475 ;
1476 ;
1477 ;
1478 ;
1479 ;
1480 ;
1481 ;
1482 ;
1483 ;
1484 ;
1485 ;
1486 ;
1487 ;
1488 ;
1489 ;
1490 ;
1491 ;
1492 ;
1493 ;
1494 ;
1495 ;
1496 ;
1497 ;
1498 ;
1499 ;
1500 ;
1501 ;
1502 ;
1503 ;
1504 ;
1505 ;
1506 ;
1507 ;
1508 ;
1509 ;
1510 ;
1511 ;
1512 ;
1513 ;
1514 ;
1515 ;
1516 ;
1517 ;
1518 ;
1519 ;
1520 ;
1521 ;
1522 ;
1523 ;
1524 ;
1525 ;
1526 ;
1527 ;
1528 ;
1529 ;
1530 ;
1531 ;
1532 ;
1533 ;
1534 ;
1535 ;
1536 ;
1537 ;
1538 ;
1539 ;
1540 ;
1541 ;
1542 ;
1543 ;
1544 ;
1545 ;
1546 ;
1547 ;
1548 ;
1549 ;
1550 ;
1551 ;
1552 ;
1553 ;
1554 ;
1555 ;
1556 ;
1557 ;
1558 ;
1559 ;
1560 ;
1561 ;
1562 ;
1563 ;
1564 ;
1565 ;
1566 ;
1567 ;
1568 ;
1569 ;
1570 ;
1571 ;
1572 ;
1573 ;
1574 ;
1575 ;
1576 ;
1577 ;
1578 ;
1579 ;
1580 ;
1581 ;
1582 ;
1583 ;
1584 ;
1585 ;
1586 ;
1587 ;
1588 ;
1589 ;
1590 ;
1591 ;
1592 ;
1593 ;
1594 ;
1595 ;
1596 ;
1597 ;
1598 ;
1599 ;
1600 ;
1601 ;
1602 ;
1603 ;
1604 ;
1605 ;
1606 ;
1607 ;
1608 ;
1609 ;
1610 ;
1611 ;
1612 ;
1613 ;
1614 ;
1615 ;
1616 ;
1617 ;
1618 ;
1619 ;
1620 ;
1621 ;
1622 ;
1623 ;
1624 ;
1625 ;
1626 ;
1627 ;
1628 ;
1629 ;
1630 ;
1631 ;
1632 ;
1633 ;
1634 ;
1635 ;
1636 ;
1637 ;
1638 ;
1639 ;
1640 ;
1641 ;
1642 ;
1643 ;
1644 ;
1645 ;
1646 ;
1647 ;
1648 ;
1649 ;
1650 ;
1651 ;
1652 ;
1653 ;
1654 ;
1655 ;
1656 ;
1657 ;
1658 ;
1659 ;
1660 ;
1661 ;
1662 ;
1663 ;
1664 ;
1665 ;
1666 ;
1667 ;
1668 ;
1669 ;
1670 ;
1671 ;
1672 ;
1673 ;
1674 ;
1675 ;
1676 ;
1677 ;
1678 ;
1679 ;
1680 ;
1681 ;
1682 ;
1683 ;
1684 ;
1685 ;
1686 ;
1687 ;
1688 ;
1689 ;
1690 ;
1691 ;
1692 ;
1693 ;
1694 ;
1695 ;
1696 ;
1697 ;
1698 ;
1699 ;
1700 ;
1701 ;
1702 ;
1703 ;
1704 ;
1705 ;
1706 ;
1707 ;
1708 ;
1709 ;
1710 ;
1711 ;
1712 ;
1713 ;
1714 ;
1715 ;
1716 ;
1717 ;
1718 ;
1719 ;
1720 ;
1721 ;
1722 ;
1723 ;
1724 ;
1725 ;
1726 ;
1727 ;
1728 ;
1729 ;
1730 ;
1731 ;
1732 ;
1733 ;
1734 ;
1735 ;
1736 ;
1737 ;
1738 ;
1739 ;
1740 ;
1741 ;
1742 ;
1743 ;
1744 ;
1745 ;
1746 ;
1747 ;
1748 ;
1749 ;
1750 ;
1751 ;
1752 ;
1753 ;
1754 ;
1755 ;
1756 ;
1757 ;
1758 ;
1759 ;
1760 ;
1761 ;
1762 ;
1763 ;
1764 ;
1765 ;
1766 ;
1767 ;
1768 ;
1769 ;
1770 ;
1771 ;
1772 ;
1773 ;
1774 ;
1775 ;
1776 ;
1777 ;
1778 ;
1779 ;
1780 ;
1781 ;
1782 ;
1783 ;
1784 ;
1785 ;
1786 ;
1787 ;
1788 ;
1789 ;
1790 ;
1791 ;
1792 ;
1793 ;
1794 ;
1795 ;
1796 ;
1797 ;
1798 ;
1799 ;
1800 ;
1801 ;
1802 ;
1803 ;
1804 ;
1805 ;
1806 ;
1807 ;
1808 ;
1809 ;
1810 ;
1811 ;
1812 ;
1813 ;
1814 ;
1815 ;
1816 ;
1817 ;
1818 ;
1819 ;
1820 ;
1821 ;
1822 ;
1823 ;
1824 ;
1825 ;
1826 ;
1827 ;
1828 ;
1829 ;
1830 ;
1831 ;
1832 ;
1833 ;
1834 ;
1835 ;
1836 ;
1837 ;
1838 ;
1839 ;
1840 ;
1841 ;
1842 ;
1843 ;
1844 ;
1845 ;
1846 ;
1847 ;
1848 ;
1849 ;
1850 ;
1851 ;
1852 ;
1853 ;
1854 ;
1855 ;
1856 ;
1857 ;
1858 ;
1859 ;
1860 ;
1861 ;
1862 ;
1863 ;
1864 ;
1865 ;
1866 ;
1867 ;
1868 ;
1869 ;
1870 ;
1871 ;
1872 ;
1873 ;
1874 ;
1875 ;
1876 ;
1877 ;
1878 ;
1879 ;
1880 ;
1881 ;
1882 ;
1883 ;
1884 ;
1885 ;
1886 ;
1887 ;
1888 ;
1889 ;
1890 ;
1891 ;
1892 ;
1893 ;
1894 ;
1895 ;
1896 ;
1897 ;
1898 ;
1899 ;
1900 ;
1901 ;
1902 ;
1903 ;
1904 ;
1905 ;
1906 ;
1907 ;
1908 ;
1909 ;
1910 ;
1911 ;
1912 ;
1913 ;
1914 ;
1915 ;
1916 ;
1917 ;
1918 ;
1919 ;
1920 ;
1921 ;
1922 ;
1923 ;
1924 ;
1925 ;
1926 ;
1927 ;
1928 ;
1929 ;
1930 ;
1931 ;
1932 ;
1933 ;
1934 ;
1935 ;
1936 ;
1937 ;
1938 ;
1939 ;
1940 ;
1941 ;
1942 ;
1943 ;
1944 ;
1945 ;
1946 ;
1947 ;
1948 ;
1949 ;
1950 ;
1951 ;
1952 ;
1953 ;
1954 ;
1955 ;
1956 ;
1957 ;
1958 ;
1959 ;
1960 ;
1961 ;
1962 ;
1963 ;
1964 ;
1965 ;
1966 ;
1967 ;
1968 ;
1969 ;
1970 ;
1971 ;
1972 ;
1973 ;
1974 ;
1975 ;
1976 ;
1977 ;
1978 ;
1979 ;
1980 ;
1981 ;
1982 ;
1983 ;
1984 ;
1985 ;
1986 ;
1987 ;
1988 ;
1989 ;
1990 ;
1991 ;
1992 ;
1993 ;
1994 ;
1995 ;
1996 ;
1997 ;
1998 ;
1999 ;
2000 ;

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

1369 : Name: WRITE_REGISTER
1370 :
1371 : Function: Writes a data byte value to a desired VDP register
1372 :
1373 : Entry: B - register number to write to
1374 : C - data byte value to be written
1375 :
1376 : Exit: if register number = 0 or 1, the respective byte
1377 : of the VDP_MODE_WORD is updated.
1378 :
1379 : Registers used: A,BC,E
1380 :
1381 : Size: ROM - 27 bytes
1382 : RAM - 0 bytes
1383 :
1384 : Comments: This version of WRITE_REGISTER is provided for stand alone operation.
1385 :
1386 : Comparison: left out pascal entry point
1387 : code compacted
1388 : edited for documentation
1389 :
1390 :
1391 : WRITE_REGISTER:
1392 : LD E,C
1393 : LD A,[VDP_CTRL_PORT]
1394 : LD C,A
1395 : OUT [C],E
1396 : LD A,B
1397 : DR 80H
1398 : OUT [C],A
1399 : LD A,B
1400 : DR A
1401 : LD A,E
1402 :
1403 : JR NZ,CHK_REG_1
1404 : LD [VDP_MODE_WORD],A
1405 :
1406 : RET
1407 :
1408 : CHK_REG_1:
1409 : DEC B
1410 : RET NZ
1411 : LD [VDP_MODE_WORD+1],A
1412 : RET
1413 :
E034 59
E035 3AFC29
E038 4F
E039 ED59
E038 78
E03C F680
E03E ED79
E040 78
E041 B7
E042 78
E043 2004
E045 32FD61
E048 C9
E049
E049 05
E04A C0
E048 32FD62
E04E C9

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
1415 : Name: READ_REGISTER
1416 :
1417 : Function: Reads a data byte value from the Colecovision CTRL_PORT.
1418 :
1419 : Entry: None.
1420 :
1421 : Exit: A - data byte value read in
1422 :
1423 : Registers used: A,C
1424 :
1425 : Size: ROM - 3 bytes
1426 : RAM - 0 bytes
1427 :
1428 :
1429 : Comments: This version of READ_REGISTER is provided for stand alone operation.
1430 : This routine does a direct access to the COLECOVISION I/O ports.
1431 :
1432 : Comparison: edited for documentation
1433 :
1434 : READ_REGISTER: LD A,[VDP_CTRL_PORT]
1435 : LD C,A
1436 : IN A,[C] ;get the data from CTRL_PORT
1437 : LD [VDP_STATUS_BYTE],A ;save a copy of VDP status data
1438 :
1439 : RET
1440 :
E04F 3AFC29
E04F 3AFC29
E052 4F
E053 ED78
E055 32FD63
E058 C9

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
1442	:	:	
1443	:	:	Name: FILL_VRAM
1444	:	:	Function: Fill a VRAM memory buffer with a constant byte value.
1445	:	:	
1446	:	:	Entry: A - constant byte value to be written
1447	:	:	DE - number of bytes in the VRAM memory buffer
1448	:	:	HL - starting VRAM address to be written to
1449	:	:	
1450	:	:	Exit: None.
1451	:	:	
1452	:	:	Registers used: AF, C, DE
1453	:	:	
1454	:	:	Size: ROM - 18 bytes
1455	:	:	RAM - 0 bytes
1456	:	:	
1457	:	:	
1458	:	:	Comments: This version of FILL_VRAM is provided for stand alone operation.
1459	:	:	This routine does a direct access to the COLECOVISION I/O ports.
1460	:	:	
1461	:	:	Comparison: edited for documentation
1462	:	:	needed call to READ_REGISTER taken out
1463	:	:	
1464	:	:	
1465	:	:	FILL_VRAM:
E059	F5	1466	PUSH AF
E05A	CDE1E9	1467	CALL SET_WRITE
E05D	E1	1468	POP HL
E05E		1469	FILL:
E05E	ED61	1470	OUT [C],H
E060	1B	1471	DEC DE
E061	7A	1472	LD A,D
E062	B3	1473	OR E
E063	20F9	1474	JR NZ,FILL
E065	C9	1475	JR NZ,FILL
		1476	RET
			: Save the data to fill with
			: Sets the ctrl port data and
			: popping AF into HL to restore the fill data
			: repeat
			: get back a copy of the saved constant
			: write the constant out to the VRAM buffer
			: byte count = byte count - 1
			: until byte count = 0
			: all done

```

LOCATION OBJECT CODE LINE SOURCE LINE
1478 : INIT_TABLE
1479 : Name:
1480 : Function:
1481 : INIT_TABLE initializes the addresses of the VRAM tables.
1482 : The passed address is converted to the correct format needed
1483 : to setup the VDP address registers. The following table codes
1484 : are use to indentify which table address is being setup:
1485 :
1486 : 0 - SPRITE ATTRIBUTE TABLE (SAT)
1487 : 1 - SPRITE GENERATOR TABLE (SGT)
1488 : 2 - PATTERN NAME TABLE (PNT)
1489 : 3 - PATTERN GENERATOR TABLE (PGT)
1490 : 4 - PATTERN COLOR TABLE (PCT)
1491 :
1492 : Entry: A - table code (see above)
1493 : HL - table address
1494 :
1495 : Exit: None.
1496 :
1497 : Registers used: AF, BC, HL, IX, IY
1498 :
1499 : Size: ROM - 99 bytes
1500 : RAM - 0 bytes
1501 :
1502 : Comments: This version of INIT_TABLE is provided for stand alone operation.
1503 :
1504 : Comparison: left out pascal entry point
1505 : code compacted
1506 : edited for documentation
1507 :
1508 :
1509 : INIT_TABLE:
1510 : LD C,A
1511 : LD B,0
1512 : LD IX,VRAM_ADDR_TABLE
1513 : ADD IX,BC
1514 : ADD IX,BC
1515 : LD [IX+0],L
1516 : LD [IX+1],H
1517 :
1518 : LD A,[VDP_MODE_WORD]
1519 : BIT 1,A
1520 : JR Z,INIT_TABLE80
1521 :
1522 : LD A,C
1523 : CP 3
1524 : JR Z,CASE_OF_GEN
1525 : CP 4
1526 : JR Z,CASE_OF_COLOR
1527 : JR INIT_TABLE80
1528 :
1529 : ;BC = index into VRAM_ADDR_TABLE table code
1530 : ;point to start of VRAM_ADDR_TABLE
1531 : ;add in the index
1532 : ;save the address
1533 : ;CHECK VDP GRAPHICS MODE
1534 : ;if VDP_MODE_WORD bit 1 mode bit 3 = 1
1535 :
1536 : ; its graphics mode 2, look for PGT or PCT first
1537 : ; get the table code
1538 : ; if table code = PGT
1539 : ; then setup PGT address
1540 : ; if table code = PCT
1541 : ; then setup PCT address
1542 : ; else not a special case, use normal setup

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		1530	:
		1531	: special case: graphics mode 2, address of PGT
		1532	:
E089		1533	CASE_OF_GEN:
E089 0604	LD	B,4	:register number = PGT base address register
E088 7D	LD	A,L	:if PGT address = 0000H
E08C B4	OR	H	
E08D 2004	JR	NZ,CASE_OF_GEN10	: data byte value for address 0000H
E08F 0E03	LD	C,3	
E091 1828	JR	INIT_TABLE90	
E093		1540	CASE_OF_GEN10:
E093 0E07	LD	C,7	:else PGT address <> 0000H
E095 1824	JR	INIT_TABLE90	: data byte value for address 2000H
		1542	:endif
		1543	:
		1544	: special case: graphics mode 2, address of PCT
		1545	:
		1546	CASE_OF_COLOR:
E097		1547	
E097 0603	LD	B,3	:register number = PCT base address register
E099 7D	LD	A,L	:if PCT address = 0000H
E09A B4	OR	H	
E098 2004	JR	NZ,CASE_OF_CLR10	
E09D 0E7F	LD	C,7FH	: data byte value for address 0000H
E09F 181A	JR	INIT_TABLE90	
E0A1		1553	CASE_OF_CLR10:
E0A1 0EFF	LD	C,OFFH	:else PCT address <> 0000H
E0A3 1816	JR	INIT_TABLE90	: data byte value for address 2000H
		1555	:endif
		1556	:
		1557	: not a special case (i.e. not mode 2, PGT or PCT)
		1558	:
		1559	INIT_TABLE80:
E0A5		1560	
E0A5 FD21E08F	LD	IY,BASE_FACTORS	:base address = table address / factor
E0A9 FD09	ADD	IY,BC	:point to start of the the factor table
E0AB FD09	ADD	IY,BC	:index into the table by the table code
E0AD FD7E00	LD	A,[IY+0]	:get the shift count
E080 FD4601	LD	B,[IY+1]	:get the register number
E083		1565	DIVIDE:
E083 C83C	SRL	H	:shift hi-byte
E085 CB1D	RR	L	:shift lo-byte
E087 3D	DEC	A	:shift count = shift count - 1
E088 20F9	JR	NZ,DIVIDE	
E08A 4D	LD	C,L	:get the data byte value
E08B		1571	INIT_TABLE90:
E08B CDFD20	CALL	WRITE_REGISTER	:setup the base register
E08E C9	RET		
		1572	
		1573	

LOCATION OBJECT CODE LINE SOURCE LINE

```

1575 ;
1576 ; base factor table, used to calculate base addresses
1577 ;
1578 BASE_FACTORS:
1579         DEFB 7           ; table code 0 base factor
1580         DEFB 5           ; register 5, SAT base address
1581         DEFB 11          ; table code 1 base factor
1582         DEFB 6           ; register 6, SGT base address
1583         DEFB 10          ; table code 2 base factor
1584         DEFB 2           ; register 2, PNT base address
1585         DEFB 11          ; table code 3 base factor
1586         DEFB 4           ; register 4, PGT base address
1587         DEFB 6           ; table code 4 base factor
1588         DEFB 3           ; register 3, PCT base address

```

EOBF 07
EOC0 05
EOC1 08
EOC2 06
EOC3 0A
EOC4 02
EOC5 08
EOC6 04
EOC7 06
EOC8 03

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
1590	:	:	PUT_VRAM	:
1591	:	:		:
1592	:	:		:
1593	:	:	Gets a block of data from a user buffer and puts it into VRAM.	:
1594	:	:	The following table codes are used to identify which VRAM table	:
1595	:	:	is being referenced:	:
1596	:	:	0 - SPRITE ATTRIBUTE TABLE (SAT)	:
1597	:	:	1 - SPRITE GENERATOR TABLE (SGT)	:
1598	:	:	2 - PATTERN NAME TABLE (PNT)	:
1599	:	:	3 - PATTERN GENERATOR TABLE (PGT)	:
1600	:	:	4 - PATTERN COLOR TABLE (PCT)	:
1601	:	:		:
1602	:	:		:
1603	:	:	A - table code (see above)	:
1604	:	:	DE - starting index into the table	:
1605	:	:	HL - address of user buffer	:
1606	:	:	IV - block size (or byte count)	:
1607	:	:		:
1608	:	:	Exit:	:
1609	:	:	None.	:
1610	:	:	Registers used: AF, DE, HL, IV	:
1611	:	:		:
1612	:	:	Routines used: SET_COUNT	:
1613	:	:		:
1614	:	:	ROM - 0 bytes	:
1615	:	:	RAM - 0 bytes	:
1616	:	:		:
1617	:	:	Comments: This version of PUT_VRAM is provided for stand alone operation.	:
1618	:	:		:
1619	:	:	Comparison: left out pascal entry point	:
1620	:	:	took out the mux sprites capability	:
1621	:	:	code compacted	:
1622	:	:	edited for documentation	:
1623	:	:		:
1624	:	:	PUT_VRAM:	:
EOC9	CDE005	1625	CALL SET_COUNT	:setup the actual byte count and the absolute VRAM address
EOC9	CDE005	1626	CALL WRITE_VRAM	:VRAM data = user buffer
EOC9	C3FD1A	1627	RET	
EOC9	C3FD1A	1628	JP WRITE_VRAM	
EOC9	C3FD1A	1629		

LOCATION OBJECT CODE LINE SOURCE LINE

```

1631 : GET_VRAM
1632 : Name:
1633 :
1634 : Function:
1635 : Gets a block of data from VRAM and stores it into a user buffer.
1636 : The following table codes are used to identify which VRAM table
1637 : is being referenced:
1638 :
1639 : 0 - SPRITE ATTRIBUTE TABLE (SAT)
1640 : 1 - SPRITE GENERATOR TABLE (SGT)
1641 : 2 - PATTERN NAME TABLE (PNT)
1642 : 3 - PATTERN GENERATOR TABLE (PGT)
1643 : 4 - PATTERN COLOR LABEL (PCT)
1644 :
1645 : Entry:
1646 : A - table code (see above)
1647 : DE - starting index into the table
1648 : HL - address of user buffer
1649 : IY - block size (or byte count)
1650 :
1651 : Exit:
1652 : None.
1653 :
1654 : Registers used: AF, DE, HL, IY
1655 :
1656 : Routines used: SET_COUNT
1657 :
1658 : Size:
1659 : ROM - 0 bytes
1660 : RAM - 0 bytes
1661 :
1662 : Comments:
1663 : This version of GET_VRAM is provided for stand alone operation.
1664 :
1665 : Comparison:
1666 : left out pascal entry point
1667 : code compacted
1668 : edited for documentation
1669 :
1670 : GET_VRAM:
1671 : CALL SET_COUNT ;setup the actual byte count and the absolute VRAM address
1672 : CALL READ_VRAM ;user buffer = VRAM data
1673 : RET
1674 : JP READ_VRAM
1675 :
1676 : EOCF
1677 : EOCF C0E0D5
1678 : EOCF C3FD1D

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

1670 :
1671 : CALLED BY PUT_VRAM_ AND GET_VRAM_
1672 :
1673 : SETS BYTE COUNT AND INDEX FOR WRITES TO AND READS FROM VRAM.
1674 :
1675 : TABLE          BYTES/ITEM
1676 : SPRITE_NAME     4
1677 : SPRITE_GEN      8
1678 : PATTERN_NAME   1
1679 : PATTERN_GEN    8
1680 : COLOR (MODE 1) 1
1681 : COLOR (MODE 2) 8
1682 :
E005 SET_COUNT
1683 :
E005 FDE5 PUSH IY
1684 :
E007 4F LD C,A
1685 :
E008 FE04 CP 4
1686 :
E00A 2008 JR NZ,NOT_COLOR_TBL_ACCESS
1687 :
E00C 3AFD61 LD A,[VDP_MODE_WORD]
1688 :
E00F E602 AND O2H
1690 :
E0E1 2814 JR Z,ADD_TO_BASE
1691 :
E0E3 79 LD A,C
1692 :
1693 :
E0E4 NOT_COLOR_TBL_ACCESS
1694 :
E0E4 FE02 CP 2
1695 :
E0E6 280F JR Z,ADD_TO_BASE
1696 :
1697 :
1698 :
1699 :
1700 :
E0E8 EB EX DE,HL
E0E9 29 ADD HL,HL
1701 :
1702 :
1703 :
E0EA 29 ADD HL,HL
E0EB 87 OR A
1704 :
E0EC 2801 JR Z,HAVE_CNT
E0EE 29 ADD HL,HL
1705 :
1706 :
1707 :
1708 :
1709 :
1710 :
E0EF HAVE_CNT:
E0EF EB EX DE,HL
E0F0 E3 EX [SP],HL
1711 :
1712 :
1713 :
1714 :
1715 :
1716 :
1717 :
1718 :
E0F1 29 ADD HL,HL
E0F2 29 ADD HL,HL
E0F3 2801 JR Z,HAVE_CNT2
1719 :
1720 :
1721 :
1722 :
E0F5 29 ADD HL,HL
E0F6 HL,HL
E0F6 E3 EX [SP],HL
1723 :
1724 :
1725 :
1726 :
: Save count on stack
: Get table code #
: See if color table
: If not then check for name table
: Color Table
: Get mode word to check graphics type
: Bit 1 indicates mode 1 or 2
: If mode 1 then start index is not modified
: Get table code back
: Fall through to multiply by 8 for mode 2.
: Check for Name table
: If so then don't modify start index
: At this point the table code is:
: Sprite attribute table, Sprite gen table,
: Pattern gen table, or Color table in Mode 2
: Get count in HL
: *2
:
: *4
: Check for Sprite attr table
: If so then Index*4 is complete
: If not then Index*8 is required
: This is for the pattern gen table,
: sprite gen table, or color gen table
: in mode 2.
: Put updated index in DE
: Get count off stack
: Put address of RAM buffer on stack
: Now modify the count for:
: Sprite attribute table, Sprite gen table,
: Pattern gen table, or Color table in Mode 2
: If Sprite attr table then *4 is enough
: (the zero flag is still set from earlier)
: Count *8 for remaining tables
: Save updated count on stack
: Restore RAM buffer off stack

```



```

LOCATION OBJECT CODE LINE      SOURCE LINE
E0F7                            1727 ADD,_BASE:
E0F7 79                        LD                    A,C
E0F8 01FD64                    LD                    BC,VRAM_ADDR_TABLE
E0FB E5                        PUSH                  HL
E0FC 2600                      LD                    H,00H
E0FE 6F                        LD                    L,A
E0FF 29                        ADD                   HL,HL
E100 09                        ADD                   HL,BC
E101 7E                        LD                    A,[HL]
E102 23                        INC                   HL
E103 66                        LD                    H,[HL]
E104 6F                        LD                    L,A
E105 19                        ADD                   HL,DE
E106 EB                        EX                    DE,HL
E107 E1                        POP                   HL
E108 C1                        POP                   BC
E109 C9                        RET

```

```

: Get table code back
: Get top of address table
: Save RAM buffer on stack
: Put table code into HL

:*2 to offset for addresses in table
: Add to top of table
: Low byte of VRAM table address
: High byte of VRAM table address

: Add offset from start of table
: Absolute VRAM address in DE
: Restore RAM buffer address
: Restore Count into BC
: Registers are now ready for
: a call to the basic READ/WRITE VRAM routines
: Exit

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

1747 ..... CALC_OFFSET .....
1748 .....
1749 :DESCRIPTION: THIS ROUTINE CALCULATES THE PROPER OFFSET INTO THE NAME TABLE
1750 : FOR THE PATTERN POSITION GIVEN BY X_PAT_POS, Y_PAT_POS. THE
1751 : FORMULA USED IS: OFFSET = 32*Y_PAT_POS + X_PAT_POS
1752 .....
1753 :INPUT: D = Y_PAT_POS
1754 : E = X_PAT_POS
1755 .....
1756 :OUTPUT: DE = OFFSET
1757 .....
1758 .....
1759 :
1760 :__CALC_OFFSET:
1761 : PUSH HL
1762 : BIT 7,D
1763 : JR Z,ELSE_11
1764 : LD H,OFFH
1765 : JR END_IF_11
1766 ELSE_11:
1767 : LD H,0
1768 END_IF_11:
1769 .....
1770 :
1771 : L,D
1772 : ADD HL,HL
1773 : ADD HL,HL
1774 : ADD HL,HL
1775 : ADD HL,HL
1776 .....
1777 : BIT 7,E
1778 : JR Z,ELSE_12
1779 : LD D,OFFH
1780 : JR END_IF_12
1781 ELSE_12:
1782 : LD D,0
1783 END_IF_12:
1784 .....
1785 : ADD HL,DE
1786 : EX DE,HL
1787 : POP HL
1788 : RET

```

```

:save HL
:if Y_PAT_POS is minus
: extend negative sign
:else
: extend positive sign
:endif

:OFFSET = 32*Y_PAT_POS + X_PAT_POS
:HL=2*Y_PAT_POS
: 4*
: 8*
: 16*
: 32*

:if X_PAT_POS is minus
: extend negative sign
:else
: extend positive sign
:endif

:HL=32*Y_PAT_POS + X_PAT_POS
:restore HL intact

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
E129		1790	*****PX_TO_PTRN_POS*****
E129 E5		1791	:DESCRIPTION: DIVIDES REG DE BY 8, IF SIGNED RESULT > 127 THEN E := MAX SIGNED
E12A C5		1792	: POSITIVE NUMBER, IF RESULT < -128, THEN E := MIN NEGATIVE NUM
E12B 0603		1793	:INPUT: DE = 16 BIT SIGNED NUMBER
E12D		1794	:OUTPUT: DE/8 < -128 E = -128
E12D CB2A		1795	: -128 <= DE/8 <=+127 E = DE/8
E12F CB18		1796	: +127 < DE/8 E = +127
E131 10FA		1797	*****
E133 C1		1798	*****PX_TO_PTRN_POS:*****
E134 21FF80		1799	: HL USED TO TEST MAGNITUDE
E137 CB7A		1800	
E139 2006		1801	: PUSH HL
E13B 19		1802	
E13C E1		1803	: PUSH BC
E13D D0		1804	: LD B,3
E13E 1E7F		1805	:PX_2_P_P_1:
E140 C9		1806	: SRA D
E141 2600		1807	: RR E
E143 19		1808	:DJNZ PX_2_P_P_1
E144 E1		1809	: POP BC
E145 D8		1810	
E146 1E80		1811	: LD HL,OFF80H
E148 C9		1812	: BIT 7,D
		1813	: JR NZ,NEGTV
		1814	
		1815	: ADD HL,DE
		1816	: POP HL
		1817	: RET NC
		1818	: LD E,7FH
		1819	: RET
		1820	
		1821	:NEGTV: LD H,00H
		1822	
		1823	: ADD HL,DE
		1824	: POP HL
		1825	: RET C
		1826	: LD E,80H
		1827	: RET
			: B WILL BE USED AS A COUNTER IN THE DIVIDE
			: (THIS IS THE START OF THE DIVIDE LOOP)
			: 16 BIT SHIFT LEFT
			: DIVIDE BY 8 LOOP
			: RESTORE BC TO OLD VALUE
			: SET HL TO -128
			: IS RESULT OF DIVIDE NEGATIVE?
			: IF DE-128 < 0 (FNC) RETURN
			: same as (IF DE<=-127 RETURN)
			: IF > 128, THEN E := MAX SIGNED + NUM.
			: SET HL TO 128
			: IF DE+128 >= 0 RETURN
			: same as (IF DE>=-128 RETURN)
			: IF < -128, THE E := MIN SIGNED - NUM.

```

LOCATION OBJECT CODE LINE SOURCE LINE
<007F>
1829 DEL EQU 7FH ;ASCII code for lt
1830 :ASCII_GEN EQU LOC_IN_ALPHA ;1st byte of generator for NUL
1831 :
1832 : __LOAD_ASCII
1833 :
1834 : Loads the ASCII character generators into VRAM at the
1835 : current pattern generator table (NOTE - INIT_TABLE must
1836 : have been called prior to this). Only the characters
1837 : from the space to the DEL (20h to 7fh) are loaded.
1838 :
1839 : Input Parameters:
1840 : NONE
1841 :
1842 : Returns:
1843 : A, BC, HL, DE are destroyed
1844 :
1845 : Falls into __PUT_ASCII
1846 :
1847 : __LOAD_ASCII:
1848 :
1849 ED5BFD6A :LD HL,[PATTRGENTBL] ;get base of current generator table
1850 :LD DE,[PATTRGENTBL] ;get base of current generator table
1851 :ADD DE,00H+8 ;offset into it where SPACE will be
1852 :EX HL,DE ;leave VRAM address in DE
1853 :LD HL,00H ;want to start with the null
1854 :LD BC,(DEL-00H+1) ;load the entire ASCII set (0..7F)
1855 :
1856 : fall thru to PUT_ASCII
1857 :
E149
E149 ED5BFD6A
E14D 210000
E150 010080

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
1859	:	:	:
1860	:	:	__PUT_ASCII
1861	:	:	:
1862	:	:	Copys a specified number of ASCII character generators into VRAM.
1863	:	:	Swaps to ROM that contains the generators then swaps back, keeps
1864	:	:	track of callers stack and uses a local stack that is known to
1865	:	:	be in RAM when the ROM is swapped in.
1866	:	:	:
1867	:	:	Input Parameters:
1868	:	:	HL - Character to 1st load (generally in range 0..FFH)
1869	:	:	BC - Number of characters to load (Not the number of bytes)
1870	:	:	DE - Address in VRAM to load the 1st generator
1871	:	:	:
1872	:	:	A, BC, HL, DE, IX are destroyed
1873	:	:	:
1874	:	:	Calls WRITE_VRAM
1875	:	:	SWITCH_MEM
1876	:	:	:
1877	:	:	__PUT_ASCII:
E153			
E153 29	ADD	HL,HL	:multiply char to start loading by 8
E154 29	ADD	HL,HL	:so can use it as an offset into the
E155 29	ADD	HL,HL	:ASCII generator
E156 C5	PUSH	BC	:save number of chars desired
	:LD	BC,ASCII_GEN	
	:LD	BC,[LOC_IN_ALPHA]	:now HL points to the 1st generator desired
	:ADD	HL,BC	:get number of chars desired
E157 E3	EX	[SP],HL	:and multiply by 8
E158 29	ADD	HL,HL	: since each generator is 8 bytes
E159 29	ADD	HL,HL	: long
E15A 29	ADD	HL,HL	:get back pointer into ASCII generators
E15B E3	EX	[SP],HL	:get number of chars times 8
E15C C1	POP	BC	
	LD	IX,0000H	: to use to save callers SP
E15D DD210000	LD	IX,SP	
E161 DD39	ADD	SP,TEMP_STACK	:set stack into known RAM
E163 31FE6E	LD	IX	:save callers SP
E166 DDE5	PUSH	IX	
	LD	A,[CUR_BANK]	:remember which bank setting we
E168 3AFD6E	PUSH	AF	: are in
E168 F5	PUSH	A,[MEM_CNFG00]	:get value to let us get at the ALPHA roms
E16C 3AFC17	LD	BC	:count would get destroyed by SWITCH_MEM
E16F C5	PUSH	CALL	SWITCH_MEM
E170 CDFD14	CALL	POP	BC
E173 C1	POP	POP	DE
	LD	DE,[LDC_IN_ALPHA]	
E174 D5	PUSH	HL,DE	
E175 ED5B0102	LD	DE	
E179 19	ADD	HL,DE	
E17A D1	POP	POP	DE
	CALL	WRITE_VRAM	
E17B CDFD1A	CALL	WRITE_VRAM	
1910	:	:	:
1911	:	:	:
E17E F1	POP	AF	:get back callers mem configuration
E17F CDFD14	CALL	CALL	SWITCH_MEM
1913	:	:	:
1914	:	:	:
E182 E1	POP	HL	:get callers SP

FILE: EOSABS:EOS_TF

HEWLETT-PACKARD: A JOS_00 (c) Coleco 1983 Confidential

Sat. 8 Sep 1984, 23:38

PAGE 50

LOCATION OBJECT CODE LINE SOURCE LINE

E183 F9	1916	LD	SP,HL	:and restore it
E184 C9	1917	RET		: and back to caller

LOCATION OBJECT CODE LINE SOURCE LINE

```

1919 :
1920 : Switch_mem: memory bank switching routine
1921 :
1922 : Inputs:
1923 :     A = Appropriate input from SWITCH_TABLE
1924 :
1925 : Example:
1926 : Switch Table Offset  Memory configuration
1927 : -----
1928 :
1929 : 0 - Boot ROM and Alpha Roms
1930 : 1 - Intrinsic RAM (0000H thru 7FFFH)
1931 : 2 - Expansion memory (0000H thru 7FFFH)
1932 : 3 - OS_7 and Intrinsic RAM (2000H thru 7FFFH)
1933 : 4
1934 :
1935 : D
1936 : E
1937 : F
1938 :
1939 :
1940 :
1941 :
1942 :
1943 :
1944 :
1945 :
1946 :
1947 :
1948 :
1949 :
1950 :
1951 : SWITCH_MEM
1952 : LD
1953 : LD
1954 : LD
1955 : OUT
1956 : LD
1957 : LD
1958 : RET

```

B,A
 A,[MEM_SWITCH_PORT]
 C,A
 [C],B
 A,B
 [CUR_BANK],A

: SAVE THE PORT DATA TO WRITE
 : GET THE PORT NUMBER
 : SWITCH MEMORY BANKS
 : REMEMBER WHAT WAS SWITCHED TO.

CUR_BANK, a defined memory location will contain the updated input parameter for applications and EOS routines which need to determine which memory space is currently active.

Outputs:

***Note: The values in the table are subject to change in future releases of ADAM hardware.

For an example of the use of SWITCH_MEM see PORT_COLLECTION below.

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		1960	:
		1961	:
		1962	:
		1963	:
		1964	:
		1965	__PORT_COLLECTION:
E191	3AFD6E	1966	LD A.[CUR_BANK]
E194	F5	1967	PUSH AF
		1968	:
E195	3AFC1A	1969	LD A.[MEM_CNFG03]
E198	CFD14	1970	CALL SWITCH_MEM
E19B	21FC29	1971	LD HL,PORT_TABLE+2
		1972	:
E19E	3A1D43	1973	LD A.[O1D43H]
E1A1	77	1974	LD [HL],A
E1A2	23	1975	INC HL
		1976	:
E1A3	3A1D47	1977	LD A.[O1D47H]
E1A6	77	1978	LD [HL],A
E1A7	23	1979	INC HL
		1980	:
E1A8	3A114B	1981	LD A.[O114BH]
E1AB	77	1982	LD [HL],A
E1AC	23	1983	INC HL
		1984	:
E1AD	3A1151	1985	LD A.[O1151H]
E1B0	77	1986	LD [HL],A
E1B1	23	1987	INC HL
		1988	:
E1B2	3A1157	1989	LD A.[O1157H]
E1B5	77	1990	LD [HL],A
E1B6	23	1991	INC HL
		1992	:
E1B7	3A1168	1993	LD A.[O1168H]
E1BA	77	1994	LD [HL],A
E1BB	23	1995	INC HL
		1996	:
E1BC	3A018E	1997	LD A.[O018EH]
E1BF	77	1998	LD [HL],A
E1C0	F1	1999	POP AF
E1C1	CFD14	2000	CALL SWITCH_MEM
E1C4	C9	2001	RET

Port collection must be called during initialization
It switches to 057 ROM grabs the ports and switches back

:GET THE CURRENT BANK

:SWAP IN THE OS

:HL POINTS TO THE PORT TABLE

:VDP CONTROL PORT

:VDP DATA PORT

:CONTROLLER 0

:CONTROLLER 1

:CONTROLLER STROBE SET

:CONTROLLER STROBE RESET

:SOUND PORT

LOCATION OBJECT CODE LINE SOURCE LINE

```

2003 :
2004 :      WR_SPR_ATTRIBUTE
2005 :
2006 :      function:
2007 :      Transfers the local copy of the sprite attribute table
2008 :      to VRAM. WR_SPR_ATTRIBUTE will also reorder the
2009 :      sprite table to reduce 5th sprite priority problems.
2010 :
2011 :      Entry:
2012 :      DE - address of Local Sprite Table
2013 :      HL - address of Priority table
2014 :      A - number of sprites to be transferred (length of Local Sprite
2015 :      table divided by 4)
2016 :      None.
2017 :
2018 :      Registers used: AF, BC, DE, HL.
2019 :
2020 :      Data structures accessed:
2021 :      VRAM_ADDR_TABLE (SPRITEATTRIBL) for the starting address
2022 :      of the sprite attribute table in VRAM.
2023 :
2024 :      PORT_PTR_TABLE (CTRL_PORT_PTR, DATA_PORT_PTR) for
2025 :      VDP port access.
2026 :
2027 :      Local Sprite Table - User defined memory image of
2028 :      the sprite attribute table
2029 :
2030 :      Priority Table - User defined sprite priority list
2031 :      for the transfer of the sprite attributes
2032 :
2033 :      Size:      ROM - xx bytes
2034 :      RAM - xx bytes
2035 :
2036 :      Comments:
2037 :      left out pascal entry point
2038 :      added register parameters in place of defined memory pointers
2039 :      code compacted
2040 :      edited for documentation
2041 :
2042 :      WR_SPR_ATTRIBUTE
2043 :      PUSH AF
2044 :      PUSH HL
2045 :      LD HL, [SPRITEATTRIBL]
2046 :      CALL SET_WRITE
2047 :      POP HL
2048 :      POP AF
2049 :      LD B, A
2050 :
2051 :      E1C5 F5
2052 :      E1C6 E5
2053 :      E1C7 2AFD64
2054 :      E1CA CDE1E9
2055 :      E1CD E1
2056 :      E1CE F1
2057 :      E1CF 47
2058 :
2059 :      ; Initialize VDP address pointer
2060 :
2061 :      ; B = Sprite Count
2062 :      ; C = port address
2063 :      ; DE = address of Local Sprite Attribute Table
2064 :      ; HL = address of Priority table
2065 :      ; VDP is initialized to appropriate address

```

FILE: EOSABS:EOS_TF HEWLETT-PACKARD: A_U05_00 (c) Coleco 1983 Confidential

```

LOCATION OBJECT CODE LINE SOURCE LINE
E100          2056 LOOP EVERY_SPRITE:
2057
2058 *        Index to sprite with the next highest priority
2059
E100 7E      LD A,(HL]           ;Get current sprite as defined by the priority table
E101 87      ADD A,A           ;*2 Multiply by an attribute entry (4)
E102 87      ADD A,A           ;+4
2063
E103 23      INC HL           ;Index to next location in priority table
E104 E5      PUSH HL          ;Save for next sprite
2066
E105 6F      LD L,A           ;Offset into the Local Sprite Table by the current priority
E106 2600   LD H,O
E108 19      ADD HL,DE
2070
2071
E109 78      LD A,B           ;Save sprite count in Acc.
E10A 0604   LD B,A           ;Set count for write to VRAM
2073
2074         ;C has port address
2075         ;HL has the pointer to a sprite's attributes
2076 LOOP EVERY_BYTE
2077         ;Output a byte
E10C EDA3   NOP           ;Wait for VDP to catch up (worst case)
E10E 00     NOP
E10F 00     JR NZ,LOOP EVERY_BYTE
E1E0 20FA   LD B,A           ;Restore sprite count to B register
E1E2 47     POP HL           ;Restore index into the priority table
E1E3 E1     DJNZ LOOP EVERY_SPRITE ; If more sprites left then loop back
E1E4 10EA   RET
E1E6 C9     ;Else exit

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
2088	:	:	
2089	:	:	SET_READ / SET_WRITE
2090	:	:	Function: Initiates the VDP for reads or writes.
2091	:	:	
2092	:	:	Entry: HL - VRAM address to start read / writes
2093	:	:	
2094	:	:	Exit: C register contains the VDP_DATA_PORT value
2095	:	:	HL remains unchanged
2096	:	:	
2097	:	:	Registers used: AF, BC, HL.
2098	:	:	
2099	:	:	Data structures accessed:
2100	:	:	
2101	:	:	PORT_TABLE (CTRL_PORT_PTR, DATA_PORT_PTR) for
2102	:	:	VDP port access.
2103	:	:	
2104	:	:	Size: ROM - xx bytes
2105	:	:	RAM - xx bytes
2106	:	:	
2107	:	:	Comments:
2108	:	:	
2109	:	:	Comparison:
2110	:	:	
2111	:	:	
2112	SET_READ:		XOR A ;Clear accumulator for later OR which
2113			; determines a read or write
2114			
2115			
2116			DEFB 0C2H ;This is the beginning of a JP NZ which
2117			; will never be true but leaves the
2118			; accumulator intact over the next
2119			; two bytes (the LD A,40H is decoded
2120			; as the last two bytes of the jump
2121	SET_WRITE:		LD A,40H ;Set bit to tell VDP that write is requested
2122			
2123			
2124	E1EB ED4BFC29		LD BC,[VDP_CTRL_PORT] ;Get BOTH port addresses
2125	E1EF ED69		OUT [C],L ;Output low 8 bits of address
2126	E1F1 84		OR H ;OR read/write bit (Bit 6) into the high 8 bytes
2127	E1F2 ED79		OUT [C],A ;Output result
2128	E1F4 4B		LD C,B ;Put DATA_PORT value into the C register
2129			
2130	E1F5 C9		RET ;Exit to caller

```

LOCATION OBJECT CODE LINE      SOURCE LINE
2132 ;CONTROLLER SOFTWARE
2133 :
2134 *****DATA*****
2135 *
2136 * DECODER TABLE FOR THE KEYBOARD
2137
2138 *
2139 DEC_KBD_TBL      DEF8 KBD_NULL      : NULL ENTRY
2140      DEF8 6      : '6'
2141      DEF8 1      : '1'
2142      DEF8 3      : '3'
2143      DEF8 9      : '9'
2144      DEF8 0      : '0'
2145      DEF8 10      : '*'
2146      DEF8 12      : Purple Action Controller key (third one down)
2147      DEF8 2      : '2'
2148      DEF8 11      : '#'
2149      DEF8 7      : '7'
2150      DEF8 13      : Blue Action Controller key (fourth (or last) one down)
2151      DEF8 5      : '5'
2152      DEF8 4      : '4'
2153      DEF8 8      : '8'
2154      DEF8 KBD_NULL      : NULL ENTRY
2155
2156
2157 KBD_NULL      EQU      OFH
2158 KBD_MASK      EQU      OFH
2159 FIRE_MASK      EQU      40H
2160 ARM_MASK      EQU      40H
2161 JOY_MASK      EQU      OFH
2162
<000F>
<000F>
<0040>
<0040>
<000F>

```

LOCATION	OBJECT CODE LINE	SOURCE LINE
2164	:	
2165	:	DECODER
2166	:	Will read all the data, from both segments, for a given controller
2167	:	All data is returned in registers
2168	:	Segment 0 data is returned the same as with OS-7
2169	:	
2170	:	NOTE - some operations are done in an odd order so that a certain
2171	:	number of cycles (T states) pass between port addressing.
2172	:	Turns off interrupts if they were on at entry time for
2173	:	195 T states (approx. 54.5 micro secs), the interrupts
2174	:	are reenabled after this time if they were on before.
2175	:	
2176	:	Callers - The input parameters are different than OS-7 and
2177	:	both segments are returned so only one call is required
2178	:	for all the data from a given controller.
2179	:	The values returned are the same for segment 0 (H, L, E)
2180	:	but segment 1 values are returned in B and D.
2181	:	
2182	:	Stack Usage:
2183	:	Uses 2 words (pushes only)
2184	:	
2185	:	Input Parameters:
2186	:	A - controller number, 0 or 1
2187	:	
2188	:	Returns:
2189	:	H - fire button data
2190	:	L - joystick data
2191	:	B - arm button data
2192	:	C - raw data for segment 0, (Joystick, fire)
2193	:	D - keyboard data
2194	:	E - spinner count data
2195	:	A - raw data for segment 1, (keyboard, arm, super controller buttons)
2196	:	
2197	:	SPIN_SWx_CT - reset to 0 after read
2198	:	
2199	:	strobe port left in the reset condition
2200	:	
2201	:	Date:
2202	:	9/24/83 11:14 JKL
2203	:	

LOCATION OBJECT CODE LINE SOURCE LINE

```

E206      2205  _DECODER:
E206 4F      LD
E207 ED57    A,I
E209 F5      PUSH
E208 79      LD
E210 2210    A,C
E20C ED4BFC2D BC,[STROBE_SET_PORT]
E210 ED79    [C],A
E212 2212    D,B
E213 2213    LD
E214 2214    HL,SPIN_SWO_CT
E216 ED4BFC28 BC,[CONTROLLER_O_PORT]
E21A B7      OR A
E21B 2002    NZ,GOT_CORRECT_CONTROLLER_IN_C
E21D 203    JR
E218 2218    INC
E21E 48      LD
E21F 2220    GOT_CORRECT_CONTROLLER_IN_C:
E21F AF      XOR
E220 5E      LD
E221 77      E,[HL]
E221 77      [HL],A
E222 42      LD
E223 ED78    B,D
E224 2224    A,[C]
E225 ED78    IN
E225 2F      CPL
E227 :
E226 2600    LD
E228 6F      LD
E229 79      L,A
E230 48      A,C
E22A 48      C,B
E22B ED79    [C],A
E22C 4F      C,A
E22E F1      AF
E22F E2E233 PD,LEAVE_THEM_OFF
E232 FB      EI
E233 7D      LEAVE_THEM_OFF:
E233 7D      LD
E234 F5      PUSH
E235 E640    AND
E237 47      LD
E238 7D      A,L
E239 E60F    AND
E23B 6F      LD
E23C 05      L,A
E23D 11E1F6 DE,DEC_KBD_TBL
E240 19      HL,DE
E241 01      DE
E242 56      D,[HL]
E243 ED78    A,[C]
E245 2F      CPL
E246 F5      PUSH
E247 6F      LD
E248 E640    FIRE_MASK
E24A 67      AND
E24B 7D      H,A
E24C E60F    LD
E24E 6F      A,L
E24F F1      AND
E250 2F      JOY_MASK
E251 2F      LD
E252 :
E253 6F      POP
E254 67      L,A
E255 E640    AND
E256 67      H,A
E257 7D      A,L
E258 E60F    AND
E259 6F      JOY_MASK
E24E 6F      LD
E24F F1      POP
E250 2F      LD
E251 2F      C,A
E252 :
E253 6F      POP
E254 67      L,A
E255 E640    AND
E256 67      H,A
E257 7D      A,L
E258 E60F    AND
E259 6F      JOY_MASK
E24E 6F      LD
E24F F1      POP
E250 2F      LD
E251 2F      C,A

```

; Save controller #

; get current state of interrupts, on/off, into P/V flag

; and save for later so can turn them back on if needed

; Restore controller #

; C<--SET PORT ,B<--RST PORT

; STROBE TO SEGMENT 1 (SET)

; SAVE RESET PORT

; point at spinner count for controller 0

; C<--PORT 0, B<--PORT 1

; A contains controller number

;

; Point to spinner for ctrlr 1

; C<--PORT_1

;

; GOT_CORRECT_CONTROLLER_IN_C:

; HAVE SPIN SWITCH COUNT

; RESET COUNT

; B<--RST PORT

; READ DATA FROM SEG 1

;

; HL<--offset into DEC_KBD_TBL

; swap RST_PORT/DATA_PORT

;

; STROBE TO SEG 0 (reset)

; C<--DATA PORT

; get back entry state of interrupt reg

;

; turn interrupts back on only if they were on entry

;

; save raw data seg_1

; Test sleeve

; HAVE ARM DATA (in B)

;

; isolate KBD data

; Save spin switch count on stack (reg E)

; Get top of keypad decoder table

; point to DEC_KBD_TBL entry

; Restore Spinner count (reg E)

; HAVE KEYBOARD DATA

; segment 0 data

;

; save raw data

; isolate fire data

; HAVE FIRE_DATA

;

; isolate joystick data

; HAVE JOY_DATA

; get back raw data for seg 0

; C = raw data segment 0

;

LOCATION OBJECT CODE LINE SOURCE LINE

E251	F1	2262	POP	AF
E252	C9	2263	RET	

:A = raw data segment 1

LOCATION OBJECT CODE LINE SOURCE LINE

```

2265 :
2266 :   __POLLER
2267 :
2268 :   Used to fill a table of values for the 2 controllers
2269 :   Does some debouncing if called twice in succession
2270 :
2271 :   NOTE - Does not return at bottom of code, return is in middle
2272 :   Interrupts will be off for a while when call to DECODE is performed.
2273 :
2274 :   Callers - This has different input parameters than OS-7 and
2275 :   the controller map is ordered differently.
2276 :
2277 :
2278 : Stack Usage:
2279 :   Will use 4 words (2 pushes, 2 call)
2280 :
2281 : Calls:
2282 :   __DECODER
2283 :   DEBOUNCE
2284 :   READ_N_DEBOUNCE (a routine that is nested in __POLLER)
2285 :
2286 : Input Parameters:
2287 :   IX - pointer to 1st byte of users controller map (10 bytes)
2288 :   A - controller enabled and spinner enable
2289 :   bit 0 - set if want controller 0 enabled
2290 :   bit 1 - set if want controller 1 enabled
2291 :   bit 7 - set if want spinner enabled for controllers which are enabled
2292 :
2293 : Returns:
2294 :
2295 :   Data in users controller map is like this off of entry IX value
2296 :   IX+0 - Joystick 0
2297 :   IX+1 - fire 0
2298 :   IX+2 - arm 0
2299 :   IX+3 - keyboard 0
2300 :   IX+4 - spinner count 0
2301 :   IX+5 - Joystick 1
2302 :   IX+6 - fire 1
2303 :   IX+7 - arm 1
2304 :   IX+8 - keyboard 1
2305 :   IX+9 - spinner count 1
2306 :
2307 :   IX - points to 1 byte past users data table of highest enabled controller
2308 :
2309 :
2310 :   all but IY - destroyed
2311 :
2312 : Date:
2313 : 9/24/83 11:14 JKL
2314 :
2315 :
<0001> 2316 CONTROLLER_0_MASK EQU 01H
<0001> 2317 CONTROLLER_1_BIT EQU 1
<0007> 2318 SPINNER_ENABLE_BIT EQU 7

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

2320 __POLLER:
E253 21F5A LD HL,PERSONAL_DEBOUNCE_TABLE
E256 4F LD C,A
E257 E601 AND CONTROLLER_0_MASK
E259 2823 JR Z,NOT_CONTROLLER_0
E25B 3D 2325 A
E25C CDE264 CALL READ_N_DEBOUNCE
E25F CB49 BIT CONTROLLER_1_BIT.C
E261 C8 RET Z
E262 POLL_CONTROLLER_1:
E262 3E01 LD A,01H
E264 READ_N_DEBOUNCE:
E264 C5 PUSH BC
E265 E5 PUSH HL
E266 CDE206 CALL __DECODER
E269 4C LD C,H
E26A 7D LD A,L
E26B E1 POP HL
E26C CDE288 CALL DEBOUNCE
E26F C1 POP BC
E270 CB79 BIT SPINNER_ENABLE_BIT.C
E272 2807 JR Z,NEXT_IX
E274 7B LD A,E
E275 D08600 ADD A,[IX+0]
E278 D07700 LD [IX+0],A
E27B D023 INC IX
E27D C9 RET
E27E CB49 BIT CONTROLLER_1_BIT.C
E280 C8 RET Z
E281 110004 LD DE,0004H
E284 19 ADD HL,DE
E285 13 INC DE
E286 D019 ADD IX,DE
E288 C3E262 JIP POLL_CONTROLLER_1

```

```

:point to controller 0 debounce buffer
:save enabled controller input
:see if controller 0 is enabled
:if not then check to see if controller 1 is
:to 0 for parameter to __DECODER
:get the data and debounce it
:see if controller 1 is active
:return if not
:get data for controller 1

:save input parameter
:save debounce buffer ptr

:copy over raw data
: with fire and joystick data
:get pointer to debounce buffer
:debounce the data (on second call)
:get back input parameter
:see if spinners are enabled

:if so add
: new data to that in
: users buffer
:point to next users buffer

:see if controller 1 is enabled
:return if not

:point to next debounce buffer
:point to next user buffer

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2357 :
2358 : DEBOUNCE
2359 :
2360 : This will check for equal data on two passes, and if
2361 : the data is the same then the users buffer is updated.
2362 :
2363 : This is done by check for equal data, if the data is
2364 : different on each call then the data on this call is
2365 : stored so that the next call can use this new data.
2366 : If the data is equal then the data is stored with the
2367 : hi bit (bit 7) set so that the next call will yield
2368 : different data (IE there could never be a match on the
2369 : next call).
2370 :
2371 : Stack Usage:
2372 : 1 word (subroutine call)
2373 :
2374 : Input Parameters:
2375 : HL - Pointer to debounce buffer
2376 : IX - Pointer to users controller map
2377 : A - Joystick data
2378 : B - arm data
2379 : C - fire data
2380 : D - keyboard data
2381 :
2382 : Returns:
2383 : HL - Pointing to next buffer (entry value + 4)
2384 : IX - Pointing to next data area (entry value + 4)
2385 : A is changed
2386 : No others are modified
2387 :
2388 : Calls self as a nested subroutine CHECK
2389 :
2390 : Date: 9/24/83 11:14 JKL
2391 :
2392 :
2393 : DEBOUNCE:
2394 : CALL CHECK
2395 : LD A,C
2396 : CALL CHECK
2397 : LD A,B
2398 : CALL CHECK
2399 : LD A,D
2400 :
2401 : CHECK
2402 : CP [HL]
2403 : JR NZ,NOT_SAME
2404 : DR BOH
2405 :
2406 : NOT_SAME
2407 : LD [HL],A
2408 : INC IX
2409 : INC HL
2410 : RET

```

:debounce joystick data
:debounce fire data
:debounce arm data
:debounce keyboard data
:fall thru to check (return implied)
:see if data was the same as last time
:set hi_order bit if same, so not equal on next call
:update local debounce buffer
:point to next buffer location
: in debounce buffer as well

```

LOCATION OBJECT CODE LINE SOURCE LINE
2412 ;
2413 ;Update_Spinner - Controller spin switch interrupt service routine
2414 ;
2415 ; This routine processes the spinner switch interrupt and updates
2416 ; the data needed by both DECODER and POLLER
2417 ;
2418 ; Ram area used: Updates SPIN_SWO_CNT and SPIN_SWI_CNT
2419 ;
2420 ; The spinner switch maskable interrupt is RST 38H
2421 ;
2422 ; Destroys AF, BC, HL
2423 ;
2424 DIR_MASK EQU 001000008
2425 INT_BIT EQU 4
2426
2427 UPDATE_SPINNER
2428 LD BC,[CONTROLLER_O_PORT]
2429 IN A,[C]
2430 HL,SPIN_SWO_CT
2431 BIT INT_BIT,A
2432 JR NZ,GET_CONTROLLER_1
2433 AND DIR_MASK
2434 NZ,INCREMENT_O_DIRECTION
2435 [HL]
2436 DEC [HL]
2437
2438 INCREMENT_O_DIRECTION:
2439 INC [HL]
2440
2441 GET_CONTROLLER_1
2442 LD C,B
2443 IN A,[C]
2444 INC HL
2445 BIT INT_BIT,A
2446 RET NZ
2447
2448 AND DIR_MASK
2449 JR NZ,INCREMENT_1_DIRECTION
2450 DEC [HL]
2451 RET
2452
2453 INCREMENT_1_DIRECTION
2454 INC [HL]
2455 RET
<0020>
<0004>
E2A4 ED48FC28
E2A4 ED78
E2AA 21FE58
E2AD CB67
E2AF 2007
E2B1 E620
E2B3 2002
E2B5 35
E2B6 35
E2B7 34
E2B7 34
E2B8 48
E2B9 ED78
E2BB 23
E2BC CB67
E2BE C0
E2BF E620
E2C1 2002
E2C3 35
E2C4 C9
E2C5
E2C5 34
E2C6 C9
2412 ;
2413 ;Update_Spinner - Controller spin switch interrupt service routine
2414 ;
2415 ; This routine processes the spinner switch interrupt and updates
2416 ; the data needed by both DECODER and POLLER
2417 ;
2418 ; Ram area used: Updates SPIN_SWO_CNT and SPIN_SWI_CNT
2419 ;
2420 ; The spinner switch maskable interrupt is RST 38H
2421 ;
2422 ; Destroys AF, BC, HL
2423 ;
2424 DIR_MASK EQU 001000008
2425 INT_BIT EQU 4
2426
2427 UPDATE_SPINNER
2428 LD BC,[CONTROLLER_O_PORT]
2429 IN A,[C]
2430 HL,SPIN_SWO_CT
2431 BIT INT_BIT,A
2432 JR NZ,GET_CONTROLLER_1
2433 AND DIR_MASK
2434 NZ,INCREMENT_O_DIRECTION
2435 [HL]
2436 DEC [HL]
2437
2438 INCREMENT_O_DIRECTION:
2439 INC [HL]
2440
2441 GET_CONTROLLER_1
2442 LD C,B
2443 IN A,[C]
2444 INC HL
2445 BIT INT_BIT,A
2446 RET NZ
2447
2448 AND DIR_MASK
2449 JR NZ,INCREMENT_1_DIRECTION
2450 DEC [HL]
2451 RET
2452
2453 INCREMENT_1_DIRECTION
2454 INC [HL]
2455 RET
:bit 5 tells us the direction
:bit that tells which spinner interrupted
:Get port values for both controllers
:Input data from controller 0
:Point to spinner update data
:Was this the controller that interrupted ?
:If not then check controller 1
:Check direction bit
:If set then increment direction
:Else decrement direction
:Decrement once more to fall through increment
:Increment direction
:Get controller 1 port into C
:Get controller 1 data
:Point HL to controller 1 update area
:Is this the controller that interrupted me?
:Return if not, Also don't assume
: either controller if no bit is set
:Check direction bit
:If set then go increment direction
:Else decrement direction
:Then return
:Increment direction
:Exit

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2458 :
2459 : .....
2460 :
2461 :
2462 :
2463 :
2464 :
2465 :
2466 :
2467 :
2468 :
2469 :
2470 :
2471 :
2472 : .....
2473 :
2474 :
2475 :
2476 :
2477 :
2478 :
2479 :
2480 :
2481 :
2482 :
2483 :
2484 :
2485 :
2486 :
2487 :
2488 :
2489 :
2490 :
2491 :
2492 :
2493 :
2494 :
2495 :
2496 :
2497 :
2498 :
2499 :
2500 :
2501 :
2502 :
2503 :
2504 :
2505 :
2506 :
2507 :
2508 :
2509 :
2510 :
2511 :
2512 :
2513 :
2514 :

```

NOTES TO SOUND PROGRAMMERS

The only differences between the OS7 version and this ADAM version are transparent to the game programmer and important to the sound programmer. Formerly CALL's to PLAY_SONGS and SND_MANAGER were necessary to update the sounds. Now a CALL to SOUNDS runs the same code. Formerly CALL's to LEAVE_EFFECT and EFXOVER were needed to end a SPECIAL_EFFECT. Now EFFECT_OVER runs the same code. Independent CALL's to PLAY_SONGS, SND_MANAGER, LEAVE_EFFECT and EFXOVER do not run in this module as before!!!!

Operating system sound routine EQUATES

Special byte 0 codes

Offsets within an SxDATA song data area

song end codes

channel numbers, B7 -B6

FILE: L_JABS:EOS_TF HEWLETT-PACKARD: SOUND EQU#S (C) Coleco. 2 CONFIDENTIAL Sat. 8 Sep 1984, 23:39 .E 65

LOCATION OBJECT CODE LINE SOURCE LINE

2515	:000	CH1	EQU	010000008
2516	:000	CH2	EQU	100000008
2517	:000	CH3	EQU	110000008
2518				

Sat. 8 Sep 1984. 23:39

CONFIDENTIAL

HEWLETT-PACKARD: SOUND EQUIS (c) Coleco. 1902

FILE: L-SABS:EOS_TF

LOCATION OBJECT CODE LINE SOURCE LINE

2577 : END FREQSWE

LOCATION OBJECT CODE LINE SOURCE LINE

```

2579 ;*****
2580 ;***** ATN_SWEEP *****
2581 ;* ATN_SWEEP *
2582 ;*****
2583
2584 ;See User's Manual for description
2585 ;RETS Z SET: if byte 8 is 0 [means sweep is over, or note was never swept]
2586 ;RETS Z RESET: if sweep in progress
2587
2588 ATN_SWEEP
2589
2590 ; * RET with Z SET if byte 8 = 00
2591
2592 LD A,[IX+8] ;check byte 8 for no sweep code
2593 OR A ;Z is set if byte 8 = 0
2594 RET Z ;leave if Z set, sweep not going
2595
2596 ; * sweep going, so dec APSV
2597
2598 PUSH IX ;point HL to APSV
2599 POP HL
2600 LD DE,APSV
2601 ADD HL,DE
2602 CALL LOCL_DECLSN ;dec APSV [LSN of byte 9]
2603 JR NZ,L22 ;if PSW is zero APSV has timed out
2604 RET NZ ;Added u0S
2605
2606 ; * dec ALEN to see if sweep over
2607
2608 CALL LOCL_MSNTOLSN ;reload APSV from APS
2609 DEC HL ;point to ALEN [# of steps in the sweep]
2610 CALL LOCL_DECLSN ;dec ALEN [LSN byte 8]
2611 JR Z,L23 ;if PSW is non-zero sweep not over yet
2612
2613 ; * add ASTEP to ATN
2614
2615 LD A,[HL] ;MSN A = ASTEP
2616 AND OFOH ;mask LSN
2617 LD E,A ;E = ASTEP 0
2618 DEC HL ;point HL to ATN
2619 DEC HL
2620 DEC HL
2621 DEC HL
2622 LD A,[HL] ;MSN A = ATN
2623 AND OFOH ;A = ATN 0
2624 ADD A,E ;MSN A = [ASTEP + ATN] 0
2625 LD E,A ;E = [ASTEP + ATN] 0
2626 LD A,[HL] ;A = ATN freq or CTRL
2627 AND OFH ;mask old ATN A = 0 freq or CTRL
2628 OR E ;OR in new ATN
2629 LD [HL],A ;store updated value back into song data area
2630 OR OFFH ;RESET Z flag, sweep not over yet
2631 JR *****
2632 RET ;Added u0S
2633
2634 ; ELSE Z flag is SET: sweep over.
2635

```


LOCATION	OBJECT CODE	LINE	SOURCE	LINE
E325		2636	L23	
E325	3600	2637	LD	[HL].0
		2638		:set byte 8 to 0 to indicate end sweep
		2639	:	ENDIF
		2640	:	ENDIF
		2641		
		2642	:000	L22
		2643		RET
E327	C9	2644		
		2645	:	END ATNSWEE
		2646		

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		2648	;
		2649	;
		2650	UPATNCTRL *
		2651	;
		2652	;
		2653	;Perform single byte update of the snd chip noise control register or any
		2654	;attenuation register. IX is passed pointing to byte 0 of a song data area, MSN
		2655	;register C = formatted channel attenuation code.
		2656	;
E328		2657	UPATNCTRL
E328 D07E04		2658	LD A,[IX+4]
E328 CB61		2659	BIT 4,C
		2660	;
		2661	JR Z,L24
E32D 2804		2662	RRCA
E32F OF		2663	RRCA
E330 OF		2664	RRCA
E331 OF		2665	RRCA
E332 OF		2666	RRCA
		2667	;
		2668	ENDIF
		2669	;
E333		2670	L24
E333 E60F		2671	AND OFH
E335 B1		2672	C
		2673	CALL OUT_TO_SOUND_PORT ;output ATN or CTRL data
		2674	***** RET
		2675	JP OUT_TO_SOUND_PORT ;output ATN or CTRL data
E336 C3E600		2676	;

```

LOCATION OBJECT CODE LINE SOURCE LINE
2678 :*****
2679 :***** UPFREQ *****
2680 :* UPFREQ *
2681 :*****
2682 :*****
2683 :Perform double byte update of a sound chip frequency register. IX is passed
2684 :pointing to byte0 of a song data area. MSN register D = formatted channel
2685 :frequency code.
2686
E339 E339 DD7E03 2687 UPFREQ
E33C E33C E60F 2688 LD A,[IX+FREQ]
E33E B2 2689 AND OFH
E33F C0E600 2690 OR D
E342 DD7E03 2691 CALL OUT_TO_SOUND_PORT ;output first freq byte
E345 E6F0 2692 LD A,[IX+FREQ] ;A = F2 F3 F4 F5 F6 F7 F8 F9
E347 57 2693 AND OFOH ;A = F2 F3 F4 F5 0 0 0 0
E348 DD7E04 2694 LD D,A ;save in D
E348 E60F 2695 LD A,[IX+FREQ+1] ;LSN A = 0 0 FO F1
E34D B2 2696 OR OFH ;A = 0 0 0 0 0 FO F1
E34E OF 2697 RRCA ;A = F2 F3 F4 F5 0 0 FO F1
E34F OF 2698 RRCA ;swap nibbles
E350 OF 2700 RRCA
E351 OF 2701 RRCA
2702 **** OUT_TO_SOUND_PORT ;output 2nd [most significant] freq byte
2703 **** RET
2704 OUT_TO_SOUND_PORT ;output 2nd [most significant] freq byte
2705 JP

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2707 :*****
2708 :*****
2709 :*      LOCL_DECLSN      *
2710 :*****
2711 :
2712 :Without affecting the MSN, decrement the LSN of the byte pointed to by HL.
2713 :HL remains the same.
2714 :RET with Z flag set if dec LSN results in 0, reset otherwise.
2715 :RET with C flag set if dec LSN results in -1, reset otherwise.
2716 :
E355      2717  _DECLSN:
E355      2718  LOCL_DECLSN
E355  AF      2719      XOR      A
E356 ED67      2720      RRD
E358 D601      2721      SUB      1
E35A F5        2722      PUSH   AF
E358 ED6F      2723      RLD
E35D F1        2724      POP   AF
E35E C9        2725      RET
2726
                A
                1
                AF
                AF
                AF
                RET
                A
                1
                AF
                AF
                AF
                RET
                A = 0  LSN [HL]
                :Z flag set if dec to 0, C flag if dec to -1
                :save Z and C flags
                :[HL] = old MSN  new LSN
                :restore Z and C flags, A = 0  new LSN

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2728 ; .....
2729 ; .....
2730 ; *   LOCL_DECMSN *
2731 ; .....
2732
2733 ;Without affecting the LSN, decrement the MSN of the byte pointed to by HL.
2734 ;HL remains the same.
2735 ;RET with Z flag set if dec MSN results in 0, reset otherwise.
2736 ;RET with C flag set if dec MSN results in -1, reset otherwise.
2737
2738   _DECMSN
2739   LOCL_DECMSN
2740   XOR      A
2741
2742   RLD
2743   SUB     1
2744   PUSH   AF
2745   RRD
2746   POP    AF
2747   RET
2748
E35F
E35F AF
E360 ED6F
E362 D601
E364 F5
E365 ED67
E367 F1
E368 C9

```

```

; A = 0   MSN [HL]
; Z flag set if dec to 0, C flag set if dec -1
; save Z and C flags
; [HL] = new MSN   old LSN
; restore Z and C flags, A = 0   new MSN

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2749 ;.....
2750 ;*          LOCL_MSNTOLSN *
2751 ;*          *
2752 ;*          *
2753 ;*          *
2754 ;Copy MSN of the byte pointed to by HL to the LSN of that byte. HL remains
2755 ;the same.
2756 ;
2757 MSNTOLSN
2758 LOCL_MSNTOLSN
2759 LD
2760 AND
2761 LD
2762 RRCA
2763 RRCA
2764 RRCA
2765 RRCA
2766 OR
2767 LD
2768 RET
2769

E369
E369 7E
E36A E6FO
E36C 47
E36D OF
E36E OF
E36F OF
E370 OF
E371 B0
E372 77
E373 C9

A,[HL]
OF0H
B,A
;A = MSN LSN to be changed
;A = MSN 0
;save in B
;swap nibbles

B
[HL],A
;A = 0 MSN
;A = MSN MSN
;[HL] = MSN MSN

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2771 :*****
2772 :*****
2773 :* LOC_ADDR16 *
2774 :*****
2775
2776 :Adds 8 bit two's complement signed value passed in A to the 16 bit location
2777 :pointed to by HL.
2778
2779   _ADDR16
2780 LOC_ADDR16
2781 LD
2782 BIT
2783 JR Z,POS
2784 *****
2785 LD B,OFFH
2786 DEC B
2787 POS
2788 ADD A,[HL]
2789 LD [HL],A
2790 INC HL
2791 LD A,[HL]
2792 ADC A,B
2793 LD [HL],A
2794 DEC HL
2795 RET

E374
E374 0600
E376 C87F
E378 2801
E37A 05
E37B
E37B 8C
E37C 77
E37D 23
E37E 7E
E37F 88
E380 77
E381 28
E382 C9

: set B for positive value in A
: if A is positive
: skip
: A is neg: extend sign bit thru B
: Added u0S

: do 8 bit add [and set Carry]
: store result into LSB 16 bit number
: put MSB
: into A
: A = MSB + Carry + B [B is 0 or FF]
: store result into MSB
: re-point HL to LSB 16 bit number

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2797 ;.....
2798 ;.....
2799 ; * PT_IX_TO_SxDATA *
2800 ;.....
2801
2802 ;SONGNO passed in B.
2803 ;Point IX to byte 0 in SONGNO's song data area.
2804 ;RET with both DE and IX pointing to SxDATA.
2805 ;HL pointing to MSB SxDATA entry in LST_OF_SND_ADDRS.
2806
2807 PT_IX_TO_SxDATA
2808
2809 ; * IX & DE := addr of byte 0 in SONGNO's song data area.
2810 ; HL pointing to MSB SxDATA entry in LST_OF_SND_ADDRS.
2811
2812 LD HL,[PTR_TO_LST_OF_SND_ADDRS] ;point HL to start LST_OF_SND_ADDRS
2813 DEC HL ;init HL for addition
2814 DEC HL
2815 LD C,B ;form 4 * SONGNO in C
2816 LD B,O
2817 RLC C
2818 RLC C
2819 ADD HL,BC ;HL pts to SxDATA's entry in LST_OF_SND_ADDRS
2820 LD E,[HL] ;move addr SxDATA to IX thru DE
2821 INC HL
2822 LD D,[HL]
2823 PUSH DE
2824 POP IX
2825 RET
2826
E383
E386 2B 2AFE6E
E387 2B
E388 4B
E389 0600
E38B C801
E38D C801
E38F 09
E390 5E
E391 23
E392 56
E393 05
E394 00E1
E396 C9

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

2828 ;*****
2829 ;* AREA_SONG_IS *****
2830 ;* AREA_SONG_IS *
2831 ;*****
2832 ;
2833 ;The address of byte 0 of a song data area is passed in IX. The song number of
2834 ;the song using that area is returned in A [OFFH if inactive]. If a special
2835 ;effect was using that area, 62 is returned in A and HL is returned with the
2836 ;address of the special sound effect routine.
2837
2838 AREA_SONG_IS
2839 LD A,[IX+0] ;A := CH# SONGNO or 62, or A := FF
2840 CP OFFH
2841 RET Z ;leave with A = FF if area inactive
2842 AND 00111111B ;mask CH#
2843 CP 62
2844 RET NZ ;leave with A = SONGNO if not a special effect [62]
2845
2846 ; special effect, so set HL to addr effect, stored in bytes 1&2
2847
2848 PUSH IX ;point HL to byte 1
2849 POP HL
2850 INC HL
2851 LD E,[HL] ;save LSB effect addr in E
2852 INC HL ;HL to byte 2
2853 LD D,[HL] ;save MSB effect addr in D
2854 EX DE,HL ;HL := addr special effect
2855 RET
2856
2857 ; END UTIL
2858
E397 D07E00
E39A FEFF
E39C C8
E39D E63F
E39F FE3E
E3A1 C0

E3A2 DDE5
E3A4 E1
E3A5 23
E3A6 5E
E3A7 23
E3A8 56
E3A9 EB
E3AA C9

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
2860 ;
2861 ;.....
2862 ;* INIT_SOUND *
2863 ;.....
2864
2865 ;See Users' Manual for description; Includes ENTRY POINT ALL_OFF
2866 ;addr LST_OF_SND_ADDRS passed in HL
2867 ;n = # of song data areas to init, passed in B
2868
2869 ; *** Sound chip register code EQUATES ***
2870
2871 ; Tone generator frequency and attenuation formatted register codes
2872
2873 <0080> EQU SR1FRQ ;BIT7 = 1, BIT6-4 = TONE GEN 1 FREQ CODE
2874 <0090> EQU SR1ATN ;BIT7 = 1, BIT6-4 = TONE GEN 1 ATTN CODE
2875 <00A0> EQU SR2FRQ ;BIT7 = 1, BIT6-4 = TONE GEN 2 FREQ CODE
2876 <00B0> EQU SR2ATN ;BIT7 = 1, BIT6-4 = TONE GEN 2 ATTN CODE
2877 <00C0> EQU SR3FRQ ;BIT7 = 1, BIT6-4 = TONE GEN 3 FREQ CODE
2878 <00D0> EQU SR3ATN ;BIT7 = 1, BIT6-4 = TONE GEN 3 ATTN CODE
2879
2880 ; Noise generator control and attenuation formatted register codes
2881
2882 <00E0> EQU SRNCTL ;BIT7 = 1, BIT6-4 = NOISE GEN CONTROL CODE
2883 <00F0> EQU SRNATN ;BIT7 = 1, BIT6-4 = NOISE GEN ATTN CODE
2884
E3AB EQU SOUND_INIT ;
2885 EQU INIT_SOUND
2886
2887 ; * Initialize PTR_TO_LST_OF_SND_ADDRS with value passed in HL
2888 ;
2889 LD [PTR_TO_LST_OF_SND_ADDRS],HL
2890
2891 ; * store inactive code at byte 0 of each of the n data areas [B = n]
2892 ;
2893 INC HL ;pt HL to song 1 data area entry in LST_OF_SND_ADDRS
2894 INC HL
2895 LD E,[HL] ;pt DE to byte 0 in first song data area
2896 LD INC HL
2897 LD D,[HL]
2898 EX DE,HL ;pt HL to byte 0 in first song data area
2899 LD DE,10 ;set DE for 10 byte increment
2900 LD A,OFFH ;Added u0S
2901
2902 B1:
2903 LD [HL],A ;deactivate area *** Added u0S
2904 ADD HL,DE ;pt HL to byte 0 next area (10 bytes away)
2905 DJNZ B1 ;do this for the n (passed in B) data areas
2906
2907 ; * store end of data area code (0) at first byte after last song data area
2908 LD [HL],0 ;store end of data area code in byte 0 data area n + 1
2909
2910 ; * set the 4 channel data area pointers to a dummy, inactive data area
2911 ;
2912 LD HL,DUMAREA ;point HL to inactive byte below [after the RET]
2913 LD [PTR_TO_S_ON_0],HL ;store addr DUMAREA at PTR_TO_S_ON_0
2914 LD [PTR_TO_S_ON_1],HL ;store addr DUMAREA at PTR_TO_S_ON_1
2915 LD [PTR_TO_S_ON_2],HL ;store addr DUMAREA at PTR_TO_S_ON_2
2916
E3AB 22FE6E
E3AE 23
E3AF 23
E3B0 5E
E3B1 23
E3B2 56
E3B3 EB
E3B4 11000A
E3B7 3EFF
E3B9
E3B9 77
E3BA 19
E3BB 10FC
E3BD 3600
E3BF 21E3E6
E3C2 22FE70
E3C5 22FE72
E3C8 22FF

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
E3C8 22FE76 2917 LD [PTR_TO_S_ON_3].HL :store addr DUMAREA at PTR_TO_S_ON_3
2918
2919 : * Initialize SAVE_CTRL
2920
2921 ***** LD A,OFFH :note: this is only time MSN SAVE_CTRL will be non zero.
2922 ***** :Out u05 (loaded earlier)
2923
E3CE 32FE78 2924 LD [SAVE_CTRL].A :thus ensuring PLAY_SONGS will output 1st real CTRL data
E3D1 2925 _TURN_OFF_SOUND:
2926 :*** ALL_OFF
2927
2928 : * turn off all 4 sound generators
2929
E3D1 3AFC2F 2930 LD A,[SOUNDPORT] :Added u05
E3D4 4F 2931 LD C,A :Added u05
E3D5 3E9F 2932 LD A,OFF+SR1ATN :form off code for tone generator 1
E3D7 ED79 2933 OUT [C],A :send it out
E3D9 3EBF 2934 LD A,OFF+SR2ATN :form off code for tone generator 2
E3DB ED79 2935 OUT [C],A :send it out
E3DD 3EDF 2936 LD A,OFF+SR3ATN :form off code for tone generator 3
E3DF ED79 2937 OUT [C],A :send it out
E3E1 3EFF 2938 LD A,OFF+SRNATN :form off code for noise generator, N
E3E3 ED79 2939 OUT [C],A :send it out
E3E5 C8 2940 RET
E3E6 FF 2941
2942 DUMAREA DEFB INACTIVE
2943
2944 : END INITSO
2945

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2947 : .....
2948 : * JUKE_BOX .....
2949 : .....
2950 : .....
2951 : .....
2952 : see Users' Manual for description
2953 : SONGNO passed in B
2954 :
2955 : PLAY IT
2956 : JUKE_BOX
2957 :
2958 : * RET if song already in progress
2959 :
2960 : PUSH BC ; save SONGNO on stack
2961 : CALL PT_IX_TO_SXDATA ; point IX to SONGNO's song data area
2962 : LD A,[IX+0] ; A := CH# [if any] SONGNO [if any]
2963 : AND 3FH ; A := 0 0 SONGNO
2964 : POP BC ; B := SONGNO
2965 : CP B ; test if already in progress
2966 : RET Z ; if so, leave
2967 :
2968 : * load first note and set NEXT_NOTE_PTR [thru LOAD_NEXT_NOTE]
2969 :
2970 : LD [IX+0],B ; store SONGNO in byte 0
2971 : DEC HL ; -HL left by PT_IX_TO_SXDATA pointing to MSB SXDATA
2972 : DEC HL ; -entry in LST_OF_SND_ADDRS point HL to note list
2973 : LD D,[HL] ; -starting addr entry in LST_OF_SND_ADDRS and save this
2974 : DEC HL ; -addr in DE
2975 : LD E,[HL] ; DE now has the initial value for NEXT_NOTE_PTR
2976 : LD [IX+1],E ; set NEXT_NOTE_PTR for first note in song
2977 : LD [IX+2],D
2978 : CALL LOAD_NEXT_NOTE ; load note, byte 0 := CH# SONGNO, set new NEXT_NOTE_PTR
2979 : JR UP_CH_DATA_PTRS ; new song, so update channel data ptrs
2980 :
2981 :
2982 : END JUKEBOX

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
2984 :*****
2985 :* PLAY_SONGS_*****
2986 :* PLAY_SONGS_*****
2987 :*****
2988 :*****
2989 : *** Sound chip register code EQUATES
2990 :
2991 :*** WHITE EQU 000001008 ;BIT2 = 1, white noise code
2992 :*** PERIOD EQU 000000008 ;BIT2 = 0, periodic noise code
2993 :*** NSRHI EQU 000000008 ;BIT0-1 Set for highest noise shift rate [N/512]
2994 :*** NSRMED EQU 000000018 ;BIT0-1 Set for medium noise shift rate [N/1024]
2995 :*** NSRLOW EQU 000000108 ;BIT0-1 Set for lowest noise shift rate [N/2048]
2996 :*** NSRTG3 EQU 000000118 ;BIT0-1 Set for shift from tone gen 3 output
2997 :
2998 :SOUNDS
2999 :*** PLAY_SONGS_
3000 :
3001 : * output CH1 attenuation and frequency
3002 :
3003 LD A,OFF+SR1ATN ;format CH1 OFF byte into A
3004 LD C,SR1ATN ;format MSN C for CH1 attenuation
3005 LD D,SR1FRQ ;format MSN D for CH1 frequency
3006 LD IX,[PTR_TO_S_ON_1] ;point IX to byte 0 data area of song for CH1
3007 CALL TONE_OUT
3008 :
3009 : * output CH2 attenuation and frequency
3010 :
3011 LD A,OFF+SR2ATN ;format CH2 OFF byte into A
3012 LD C,SR2ATN ;format MSN C for CH2 attenuation
3013 LD D,SR2FRQ ;format MSN D for CH2 frequency
3014 LD IX,[PTR_TO_S_ON_2] ;point IX to byte 0 data area of song for CH2
3015 CALL TONE_OUT
3016 :
3017 : * output CH3 attenuation and frequency
3018 :
3019 LD A,OFF+SR3ATN ;format CH3 OFF byte into A
3020 LD C,SR3ATN ;format MSN C for CH3 attenuation
3021 LD D,SR3FRQ ;format MSN D for CH3 frequency
3022 LD IX,[PTR_TO_S_ON_3] ;point IX to byte 0 data area of song for CH3
3023 CALL TONE_OUT
3024 :
3025 : * output CHO [noise] ATN [and CTRL, if different from last time]
3026 :
3027 LD A,OFF+SRNATN ;format CHO OFF byte into A
3028 LD C,SRNATN ;format MSN C for CHO attenuation
3029 LD IX,[PTR_TO_S_ON_0] ;point IX to byte 0 data area of song for CHO
3030 LD E,[IX+0] ;look for inactive code, OFFH
3031 INC E ;this sets Z flag if E = OFFH
3032 JR NZ,L5 ;if PSW is zero song data area is inactive
3033 CALL OUT_TO_SOUND_PORT ;turn off CHO
3034 JR L6 ;SND_MANAGER
3035 :
3036 : ELSE
3037 :
3038 L5
3039 CALL UPATNCTRL ;send out current ATN
3040 LD A,[IX+CTRL] ;LSN A = current CTRL data

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
E446 E60F 3041 AND :mask MSN
E448 21FE78 3042 LD HL,SAVE_CTRL :point to last CTRL data sent
E44B BE 3043 CP [HL] :compare
E44C 2817 3044 JR Z,L6 :if PSW is zero CTRL has not changed (done)
E44E 77 3045 LD [HL],A :SAVE_CTRL = new CTRL data
E44F 0E00 3046 LD C,SRMCTL :send new CTRL data
E451 CDE328 3047 CALL UPATNCTRL
E454 180F 3048 JR L6 :SND_MANAGER
3049
3050 :
3051 :
3052
3053 TONE_OUT
3054 LD E,[IX+0] :look for inactive code, OFFH
E456 DD5E00 3055 INC E :this sets Z flag if E = OFFH
E459 1C 3056 JR NZ,L7 :if PSW is zero song data area is inactive
E45A 2003 3057 CALL OUT_TO_SOUND_PORT :turn off CHx
3058 **** JR L8
3059 **** JP OUT_TO_SOUND_PORT :turn off CHx
3060
E45C C3E600 3061 : ELSE send out current ATN and FREQ
3062
3063 L7
E45F CDE328 3064 CALL UPATNCTRL :send out attenuation
E462 C3E339 3065 JP UPFREQ :send out frequency
3066 **** CALL UPFREQ :send out frequency
E465 3067 L8
3068 **** RET
3069 : ENDIF

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

3071 :*****
3072 :* SND_MANAGER *
3073 :*****
3074
3075 :See Users' Manual for description
3076
E465 3077 L6
E465 3078 SND_MANAGER
3079
3080 : * IX := addr of song #1 data area [SIDATA]
3081
3082 LD B,1 :pt IX to byte 0 song data area for song # 1
3083 CALL PT_IX_TO_SxDATA
3084
E46A 3085 L1
E46A 3086 LD A,ENDSDATA :LOOP until end of song data areas
E46C 3087 CP [IX+0] :check for end of song data areas
E46F CB 3088 RET Z :set Z flag if inactive
3089 :leave [Z set], if all data areas have been processed
3090 :
3091 * process active song data areas
E470 3092 CALL PROCESS_DATA_AREA:update counters or call effect get next note
3093
3094 : * point IX to byte 0 next song data area
3095
E473 3096 LD DE,10
E476 3097 ADD IX,DE
E478 18FO 3098 JR L1 :REPEAT LOOP
3099

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

3101 :*****
3102 :* UP_CH_DATA_PTRS *
3103 :*****
3104 :*****
3105
3106 :For each active data area, starting with SIDATA and proceeding in order, load
3107 :the associated channel data area pointer [PTR_TO_S_ON_x] with the address of
3108 :byte 0. This routine is called by JUKE_BOX, when a song starts and
3109 :PROCESS_DATA_AREA when the channel using a data area has changed as a result of
3110 :calling LOAD_NEXT NOTE [this happens when a song finishes and when it switches
3111 :back and forth between noise and tone notes].
3112
3113 UP_CH_DATA_PTRS
3114 LD HL,DUMAREA ;save current IX
3115 LD [PTR_TO_S_ON_0],HL ;set all 4 ch data ptrs to dummy inactive area
3116 LD [PTR_TO_S_ON_1],HL
3117 LD [PTR_TO_S_ON_2],HL
3118 LD [PTR_TO_S_ON_3],HL
3119 LD B,1
3120 CALL PT_IX_TO_SxDATA ;set IX to byte 0 SIDATA
3121 ;RETS with IX addr byte 0 song 1
3122
3123 L2
3124 LD A,[IX+0] ;LOOP until end of song data areas
3125 OR A ;test for end of song data areas
3126 JR Z,DONE_SNDMAN ;**** CP ENDSDATA
3127 ;leave loop if all data areas checked
3128
3129 * If area active, set appropriate channel data area pointer
3130 INC A ;**** CP INACTIVE ;check for inactive data area: don't up date ptr if so
3131 JR Z,L9 ;if PSW is non-zero area is active: update channel data ptrs
3132 LD A,[IX+0] ;get CH# in A
3133 AND OCOH ;B7 - B6 in A = CH#
3134 RLCA ;form CH# * 2 in A, i.e., the offset from
3135 RLCA ;PTR_TO_S_ON_0 of the channel data area pointer
3136 RLCA ;that points to channel CH#
3137 LD E,A ;add offset to addr of PTR_TO_S_ON_0
3138 LD D,O
3139 LD HL,PTR_TO_S_ON_0
3140 ADD HL,DE ;HL points to proper channel data area pointer
3141 PUSH IX ;store this song data area's byte 0 addr there
3142 POP DE
3143 LD [HL],E
3144 INC HL
3145 LD [HL],D
3146
3147 ENDIF
3148
3149 ; * point IX to byte 0 next song data area
3150
3151 L9
3152 LD DE,10
3153 ADD IX,DE
3154 JR L2 ;REPEAT LOOP
3155
3156 DONE_SNDMAN
3157 POP IX ;restore IX

```


LOCATION OBJECT CODE LINE SOURCE LINE

E4B7 C9 3158 :000 L12
3159 RET
3160

LOCATION OBJECT CODE LINE SOURCE LINE

```

3162 : .....
3163 : * LEAVE_EFFECT *
3164 : .....
3165 : .....
3166 : .....
3167 : LEAVE_EFFECT, called by a special sound effect routine when it's finished,
3168 : restores the SONGNO of the song to which the effect note belongs to B5 - B0 of
3169 : byte 0 in the effect's data area, and loads bytes 1 and 2 with the address of
3170 : the next note in the song. The address of the 1 byte SONGNO (saved by the
3171 : effect when first called) is passed in DE. The 2 byte address of the next note
3172 : in the song, also saved by the effect, is passed in HL. IX is assumed to be
3173 : pointing to byte 0 of the data area to which the song number is to be restored.
3174 : Bits 7 and 6 of the saved SONGNO are ignored, and therefore may be used by the
3175 : effect to store flag information during the course of the note.
3176 : .....
3177 : ... EFFECT_OVER
3178 : ... LEAVE_EFFECT
3179 : LD [IX+1],L
3180 : LD [IX+2],H
3181 : LD A,[DE]
3182 : AND 03FH
3183 : LD B,A
3184 : LD A,[IX+0]
3185 : AND 0COH
3186 : DR B
3187 : LD [IX+0],A
3188 : JR EFXOVER
3189 : .....

E488
E488 DD7501
E488 DD7402
E48E 1A
E4BF E63F
E4C1 47
E4C2 DD7E00
E4C5 E6C0
E4C7 80
E4C8 DD7700
E4CB 1816

: LSB NEXT_NOTE_PTR := LSB addr next note in song
: MSB NEXT_NOTE_PTR := MSB addr next note in song
: A := x x SONGNO (i.e., the saved, original SONGNO)
: B := 0 0 SONGNO
: A := CH# 62 (all effect notes have SONGNO = 62)
: A := CH# 0 0 0 0
: A := CH# SONGNO
: restore the song number

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

3191 :*****
3192 :* PROCESS_DATA_AREA *
3193 :* *****
3194 :*****
3195 :*****
3196 :See Users' Manual for description
3197 :Terminology: SFX = address of sound effect routine
3198 :*****
3199 PROCESS_DATA_AREA
3200 CALL AREA_SONG_IS ;return area's SONGNO in A [and addr SFX in HL]
3201 CP INACTIVE ;test for inactive code
3202 RET Z ;RET, no processing if area inactive
3203 :*****
3204 : * if special effect, call it to process the data area
3205 :*****
3206 CP 62 ;test for special sound effect
3207 JR NZ,L10 ;if PSW is zero data area used by sound effect
3208 LD DE,7 ;pt HL to SFX+7, starting adr of the effect's code
3209 ADD HL,DE
3210 JP [HL] ;do 1 pass thru effect, RET from effect
3211 :*****
3212 : ENDIF
3213 :*****
3214 : * else process a non-effect note
3215 :*****
3216 L10
3217 CALL ATN_SWEEP ;process atn sweep data, if any
3218 CALL FREQ_SWEEP ;proc frq sweep data, if any, & note duration timer s
3219 ***** ;if PSW is zero note is over
3220 RET NZ
3221 EFXOVER
3222 LD A,[IX+0] ;A := CH# SONGNO this note
3223 PUSH AF ;save on stack
3224 CALL LOAD_NEXT_NOTE ;load data for next note
3225 POP BC ;B := CH# SONGNO previous note
3226 LD A,[IX+0] ;A := CH# SONGNO new note [may be inactive]
3227 CP B ;check against new note's CH# SONGNO
3228 ***** ;if PSW is non-zero change to/from tone/efx/noise
3229 RET Z
3230 JR UP_CH_DATA_PTRS ;to maintain data area priority system
3231 :*****
3232 : ENDIF
3233 : END SNDMAN
3234 :*****
E4CD CDE397
E4DO FEFF
E4D2 C8

E4D3 FE3E
E4D5 2005
E4D7 110007
E4DA 19
E4DB E9

E4DC CDE2F7
E4DF CDE2C7

E4E2 CO
E4E3
E4E3 DD7E00
E4E6 F5
E4E7 CDE4F2
E4EA C1
E4EB DD7E00
E4EE B8

E4EF C8
E4FO 1888

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
3236
3237 :*****
3238 :* LOAD_NEXT_NOTE *
3239 :*****
3240
3241 :see Users' Manual for description
3242 :SFX refers to the beginning address of a special sound effect routine
3243
E4F2 LOAD_NEXT_NOTE
3244
3245 : * deactivate area, save SONGNO on stack
3246 :
3247
3248 LD A,[IX+0] ; A := byte 0
E4F5 E63F AND 00111111B ;mask CH#, if any
E4F7 F5 PUSH AF ;save SONGNO on stack
E4F8 DD3600FF LD [IX+0],INACTIVE ;deactivate area
3251
3252 : * A := header new note
3253 :
3254
3255 LD L,[IX+1] ;HL := addr new note in ROM
E4FC DD6E01 LD H,[IX+2]
E4FF DD6602 LD A,[HL]
E502 7E ; A := header new note
3258
3259 : * save header of new note in song on stack and load its data CASE note type
3260
3261 LD B,A ;save header new note in B
E503 47 BIT 5,A ;test for rest
E504 CB6F JR Z,L13 ;if PSW is non-zero note is a rest
E506 281C
3264
3265 :*** REST
3266 PUSH BC ;save header on stack
E508 C5 AND 00011111B ;mask all but duration bits
E509 E61F
3268
3269 : * set up NEXT_NOTE_PTR
3270
3271 INC HL
E508 23 LD [IX+1],L
E50C DD7501 LD [IX+2],H
E50F DD7402
3274
3275 : * move this note's data and fill in bytes where necessary
3276
3277 LD [IX+ATN],OF0H ;set attn off
E512 DD3604FO ;NLEN := 5 bit duration
3278 LD [IX+NLEN],A ;indicate freq not to be swept
E516 DD7705 LD [IX+ASTEP],0
3279 LD [IX+ASTEP],0 ;indicate attn not to be swept
E519 DD360700 JP MODBO
3280 LD [IX+ASTEP],0
E51D DD360800
3281 C3E5EC
3282
3283 : ENDF
3284
3285 L13
3286 BIT 4,A ;test for end
E524 CB67 JR Z,L14 ;if PSW is non-zero end of song
3287 E526 280E BIT 3,A ;test for repeat
E528 CB5F JR Z,ENDNOREP ;if PSW is non-zero repeat song
3289 :*** ENDREP
3290 POP BC ; B := SONGNO
E52C C1 ; to reload if note of this song
E52D C3E:--

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
E56E DD7501 3350 LD [IX+1],L
E571 DD7402 3351 LD [IX+2],H
3352
3353 ; * move new note data and fill in bytes where necessary
3354
E574 2B 3354 DEC HL
E575 110005 3355 LD DE,05
E578 CDE603 3357 CALL DE,TO_DEST
E57B 010003 3358 LD BC,3
E57E E088 3359 LDDR
E580 DD360700 3360 LD [IX+fstep],0
E584 DD360800 3361 LD [IX+astep],0
E588 1862 3362 JR MODBO
3363
3364 ;
3365
3366 L16
E58A FE01 3367 CP 1
E58C 201A 3368 JR NZ,L17
3369
3370 ;*** TYPE1
E58E 110006 3371 LD DE,6
E591 19 3372 ADD HL,DE
E592 DD7501 3373 LD [IX+1],L
E595 DD7402 3374 LD [IX+2],H
3375
3376 ; * move new note data and fill in bytes where necessary
3377
E598 2B 3378 DEC HL
E599 1C 3379 INC E
E59A CDE603 3380 CALL DE,TO_DEST
E59D 010005 3381 LD BC,5
E5A0 E088 3382 LDDR
E5A2 DD360800 3383 LD [IX+astep],0
E5A6 1844 3384 JR MODBO
3385
3386 ;
3387
3388 L17
E5A8 FE02 3389 CP 2
E5AA 2026 3390 JR NZ,TYPE3
3391 ;*** TYPE2
E5AC 110006 3392 LD DE,6
E5AF 19 3393 ADD HL,DE
E5B0 F1 3394 POP AF
E5B1 F5 3395 PUSH AF
E5B2 E6C0 3396 AND 11000000B
E5B4 2001 3397 JR NZ,L18
E5B6 2B 3398 DEC HL
3399
3400 ;
3401
3402 L18
E5B7 DD7501 3403 LD [IX+1],L
E5BA DD7402 3404 LD [IX+2],H
3405
3406 ; * move new note data and fill in bytes where necessary

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
E58D 2B 3407
E5BE 1E09 3408 HL
E5C0 CDE603 3409 LD E,9
E5C3 010002 3410 CALL DE_TO_DEST
E5C6 EDB8 3411 LD BC,2
E5C8 AF 3412 LDDR
E5C9 12 3413 XOR A
E5CA 1B 3414 LD [DE],A
E5CB 1B 3415 DEC DE
E5CC 0E03 3416 DEC DE
E5CE EDB8 3417 LD C,3
E5D0 181A 3418 LDDR
3419 JR MODBO
3420
3421 ;
3422 ;
3423 ; if here, note is type 3: swept freq, swept attenuation
3424
3425 TYPE3
3426 LD DE,8
3427 ADD HL,DE
3428 LD [IX+1],L
3429 LD [IX+2],H
3430
3431 ; * move new note data and fill in bytes where necessary
3432
3433 DEC HL
3434 PUSH IX
3435 POP IY
3436 INC E
3437 ADD IY,DE
3438 PUSH IY
3439 POP DE
3440 LD BC,7
3441 LDDR
3442
3443 ; * modify byte 0 basis header new note
3444
3445 MODBO
3446 PUSH IX
3447 POP HL
3448 POP AF
3449 POP BC
3450 CP INACTIVE
3451 RET Z
3452 LD D,A
3453 AND 3FH
3454 CP 04
3455 JR NZ,L20_LOAD_NEX
3456 LD B,62
3457
3458 L20_LOAD_NEX:
3459 LD A,D
3460 AND OCOH
3461 OR B
3462 LD [HL],A
3463

:point HL back to 1st ROM data to move, APS
:point DE to destination: bytes 9,8,5 - 3
:move 2 bytes
:when done, DE points to FSTEP, HL to ROM NLEN
:FSTEP := 0 for no freq sweep
:pt DE to RAM NLEN
:move last 3 ROM bytes if this is a noise note, garbage
:will be loaded into byte 3, but's that's OK

;set up NEXT_NOTE_PTR
:note after this note is 8 bytes away.
:pt HL to it
:put addr in NEXT_NOTE_PTR

:point HL back to 1st ROM data to move, APS
:point DE to destination: bytes 9 - 3
:IY := addr byte 0 [and DE = 6]
:DE := 9
:IY := addr byte 9 [APS]
:DE := addr APS
:move 7 bytes

:pt HL to byte 0
:A := header new note
:B := SONGNO
:test for inactive [song over, as detected above]

:save header in D
:Rid channel bits
:Special effect

:restore A to header
:A := CH# 0 0 0 0
:A := new CH# SONGNO
:store back in byte 0

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		3464 :	ENDIF
		3465	
E602	C9	3466 :	eee L19
		3467	RET
		3468	
E603	DDE5	3469	DE_TO_DEST
E605	FDE1	3470	PUSH IX
E607	FD19	3471	POP IY
E609	FDE5	3472	ADD IY,DE
E608	D1	3473	PUSH IY
E60C	C9	3474	POP DE
		3475	RET
		3476	
		3477 :	END LOADNEX
		3478	

:DE passed = offset from byte 0. RETed w addr byte offset
 :IY := addr byte 0 [and DE = offset]
 :IY := addr byte 0 + offset
 :DE := addr of destination byte in SxDATA

LOCATION OBJECT CODE LINE SOURCE LINE

```

3480 :-----
3481 :
3482 : Out_to_sound_port
3483 : Input: a=sound port data to output
3484 : register usage: n.a.
3485 :
3486 : OUT_TO_SOUND_PORT:
3487 :   PUSH BC
3488 :   LD B,A
3489 :   LD A,[SOUNDPORT]
3490 :   LD C,A
3491 :   OUT [C],B
3492 :   LD A,B
3493 :   POP BC
3494 :   RET
3495 :-----
3496 :-----
3497 :-----
3498 :-----
3499 :-----
3500 :

```

E600 C5 ;SAVE BC
E60E 47 ;THE IDEA IS TO GET THE
E60F 3AFC2F ;INDIRECT PORT ADDRESSING
E612 4F ;THRU THE PORT TABLE IN EOS
E613 ED41
E615 78
E616 C1
E617 C9

LOCATION OBJECT CODE LINE SOURCE LINE

```

3502 :GLB QUERY_FILE, __SET_FILE, __MAKE_FILE
3503 :GLB __FILE_QUERY
3504 :EXT FCB_HEAD_ADDR,FCB_DATA_ADDR
3505 :EXT SCAN_FOR_FILE
3506 :EXT BLOCKS_REQ,USER_BUF,USER_NAME
3507 :EXT BUF_START,BUF_END
3508 :EXT STROMP,BASECMP
3509 :EXT NEW_HOLE_SIZE,NEW_HOLE_START,HOLE_FILE_NAME
3510 :EXT EOS_YEAR,EOS_MONTH,EOS_DAY
3511 :EXT BLK_STRT_PTR,VOL_BLK_SZ :DLS(8/29/83)
3512 :EXT READ_BLOCK,WRITE_BLOCK,CHECK_IF_DIRECTORY
3513 :EXT DIR_BLOCK_NO,FOUND_AVAIL_ENTRY
3514 :EXT FILENAME_CMPS
3515 -----
3516 -----
3517 -----
3518 QUERY_FILE -- Read the file's directory entry. (USES STROMP FOR FILE NAME COMPARISONS)
3519 __FILE_QUERY -- SAME AS ABOVE BUT SETS UP SCAN_FOR_FILE FOR BASE COMPARES ( USES BASECMP )
3520 -----
3521 CALLING PARAMETERS: Device number in A
3522 address of name string in DE
3523 address of buffer in HL
3524 -----
3525 EXIT PARAMETERS: If no errors -- Z = 1; A = 0; BCDE = file's start block;
3526 directory entry in caller's buffer
3527 If errors -- Z = 0; A = error code; DE = junk;
3528 caller's buffer undefined
3529 -----
3530 -----
3531 -----
3532 __FILE_QUERY:
3533 SCF
3534 JR QUEER
3535 -----
3536 QUERY_FILE
3537 SCF
3538 CCF
3539 QUEER:
3540 PUSH HL
3541 PUSH IX
3542 PUSH AF
3543
3544 JR C,SET_UP_A ; DID WE COME IN AT FILE_QUERY
3545
3546 XOR A ; USE STROMP FOR FILE NAME COMPARE
3547 SET_UP_A:
3548 LD [FILENAME_CMPS],A ; O = COMPARE COMPLETE FILE NAME
3549 ; NOT O - COMPARE ONLY THE BASE
3550 POP AF
3551
3552 LD [USER_BUF],HL ;SAVE USER'S DATA ADDRESS
3553
3554 LD IX,[FCB_HEAD_ADDR] ;POINT TO SYSTEM'S FCB
3555 LD H,D ;GET NAME POINTER IN HL
3556 LD L,E
3557 CALL SCAN_FOR_FILE ;LOOK IN THE DIRECTORY
3558

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
E634	2017	3559	JR NZ,Q_ERROR ;BRANCH IF ERROR RETURNED
E636	D5	3560	PUSH DE ;SAVE START BLOCK OF FILE
E637	C5	3561	PUSH BC
E638	ED5BFE06	3562	LD DE,[USER_BUF] ;GET BUFFER ADDRESS IN DE
E63C	DD6E21	3563	LD L,[IX+FCB_POINTER] ;GET ADDR OF ENTRY IN HL
E63F	DD6622	3564	LD H,[IX+FCB_POINTER+1]
E642	010017	3565	LD BC,DIR_ENT_LENGTH-3 ; SET BYTE COUNT TO LOAD ONLY VALID INFO THAT
		3566	; THAT WAS STORED ON THE DEVICE
		3567	
		3568	
E645	ED80	3569	LDIR ;COPY DATA TO CALLER'S BUFFER
E647	C1	3570	POP BC ;GET FILE'S START BLOCK
E648	D1	3571	POP DE
E649	AF	3572	XOR A ;SHOW NO ERROR
E64A	32FD08	3573	LD [FILENAME_CMPS].A ; DEFAULT FOR SCAN FOR FILE
E64D	DDE1	3574	LD ;RESTORE REG'S
E64F	E1	3575	POP IX
E650	C9	3576	POP HL
		3577	RET
		3578	
		3579	
		3580	
		3581	

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		3583	-----	
		3584	__SET_FILE --	Re-write the file's directory entry.
		3585		
		3586	CALLING PARAMETERS:	Device number in A
		3587		address of name string in DE
		3588		address of buffer in HL
		3589		
		3590	EXIT PARAMETERS:	If no errors -- Z = 1; A = 0;
		3591		directory entry updated
		3592		If errors -- Z = 0; A = error code;
		3593		directory entry unchanged
		3594	-----	
		3595		
		3596	-----	
		3597		
		3598	__SET_FILE	BC :SAVE REG'S
E651	C5	3599	PUSH	DE
E652	D5	3600	PUSH	HL
E653	E5	3601	PUSH	IX
E654	DDE5	3602	PUSH	[USER_BUF],HL :SAVE ADDR OF USER'S DATA
E656	22FE06	3603	LD	
		3604	LD	IX,[FCB_HEAD_ADDR] :POINT TO SYSTEM'S FCB
E659	DD2AFDFD	3605	LD	
		3606	LD	H,D :GET STRING ADDRESS IN HL
E65D	62	3607	LD	L,E
E65E	68	3608	CALL	SCAN_FOR_FILE :GET THE DIR ENTRY
E65F	CDEF08	3609	JR	NZ,S_ERROR :BRANCH IF ERROR
E662	2026	3610		
		3611	LD	HL,[USER_BUF] :GET BUFFER ADDRESS
E664	2AFE06	3612	LD	E,[IX+FCB_POINTER] :GET ENTRY ADDR IN DE
E667	DD5E21	3613	LD	D,[IX+FCB_POINTER+1]
E66A	DD5622	3614	LD	BC,DIR_ENT_LENGTH-3 :SET BYTE COUNT
E66D	010017	3615	LD	
E670	EDB0	3616	LDIR	:COPY THEIR BUFFER TO BLOCK BUFFER
		3617		
		3618	TAPE_TIMED_OUT:	
E672	DD7E17	3619	LD	A,[IX+FCB_DEVICE] :SET DEVICE NUMBER
E675	2AFDFD	3620	LD	HL,[FCB_DATA_ADDR] :GET ADDR OF MY BUFFER
E678	DD5E19	3621	LD	E,[IX+FCB_BLOCK] :GET BLOCK ADDRESS IN BCDE
E67B	DD561A	3622	LD	D,[IX+FCB_BLOCK+1]
E67E	DD4E1B	3623	LD	C,[IX+FCB_BLOCK+2]
E681	DD461C	3624	LD	B,[IX+FCB_BLOCK+3]
E684	DDF1E6	3625	CALL	WRITE_BLOCK :RE-WRITE THE DIRECTORY BLOCK
		3626		
E687	2001	3627	JR	NZ,WRITE_ERRORS
E689	AF	3628	XOR	A :SHOW NO ERRORS
		3629		
E68A		3630	WRITE_ERRORS:	
E68A		3631	S_ERROR	
		3632		
E68A	DDE1	3633	POP	IX :RESTORE REG'S
E68C	E1	3634	POP	HL
E68D	D1	3635	POP	DE
E68E	C1	3636	POP	BC
E68F	C9	3637	RET	
		3638		

```

LOCATION OBJECT CODE LINE      SOURCE LINE
-----
3640 :
3641 :
3642 :   __MAKE_FILE: creates a file in the directory.
3643 :
3644 :   ENTRY PARAMETERS: called with device ID in A; address of name
3645 :   string in HL; file size (in bytes) in BC DE.
3646 :
3647 :   NOTE: IF FILE SIZE = 0 (BC DE), THEN THE REMAINDER OF THE TAPE IS
3648 :   ALLOCATED TO THE FILE
3649 :
3650 :   EXIT PARAMETERS: CONDITION FLAGS
3651 :   Z - NO ERRORS
3652 :   NZ - ERRORS
3653 :   A = ERROR CODE
3654 :
3655 :   ALL REGISTERS ARE PRESERVED EXCEPT AF
3656 :
3657 :
-----
3659 FOUND_ENTRY EQU 0 ; BIT INDICATING DELETED FILE FOUND FOR OVERLAYING
3660
3661 __MAKE_FILE
3662     IV          ;SAVE REGISTERS
3663     PUSH IX
3664     PUSH HL
3665     PUSH DE
3666     PUSH BC
3667
3668     LD IX,[FCB_HEAD_ADDR] ;POINT TO SYSTEM FCB
3669     LD [IX+FCB_MODE],0 ; INIT IT IN CASE IT WAS SET
3670
3671     LD [IX+FCB_DEVICE],A ;SAVE DEVICE NUMBER
3672     LD [USER_NAME],HL ;SAVE POINTER TO NAME
3673
3674     LD A,B ; CHECK IF TO ALLOCATE REST OF TAPE
3675     OR C
3676     JR NZ,GOT_FILE_SIZE ; NOPE, GOT SIZE
3677
3678     LD A,E
3679     OR D
3680     JR NZ,GOT_FILE_SIZE
3681     SET MODE_REMAINDER_BIT,[IX+FCB_MODE] ; FLAG IT
3682
3683     JR SET_UP_DIR
3684
3685
3686 ; SINCE WE ARE CALLED WITH A BYTE COUNT, DIVIDE IT BY 1024
3687 ; (SHIFT RIGHT 10 BITS) TO GET A BLOCK COUNT.
3688
3689 GOT_FILE_SIZE:
3690     LD E,D ;DO A QUICK SHIFT BY 8
3691     LD D,C
3692     LD C,B
3693     LD B,0
3694     SRL C
3695     RR D
3696     RR E

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

E68E CB39          3697      SRL          C          : THEN ONE LAST TIME
E6C0 CB1A          3698      RR           D
E6C2 CB1B          3699      RR           E
3700
3701 : FOR THIS TO BE A LEGAL FILE SIZE, IT MUST FIT INTO DE.
3702
3703      LD           A,C          : CHECK FOR TOO BIG
3704      OR           A
3705      JP           NZ,TOO_BIG   : JUMP IF TOO BEEG, SENOR
3706      INC          DE           : ROUND UP IN CASE THERE WAS A FRACTION
3707      LD           A,E          : CHECK FOR TOO BIG AGAIN
3708      OR           D
3709      JP           Z,TOO_BIG   :
3710      LD           [BLOCKS_REQ],DE : ELSE SAVE THE BLOCK COUNT
3711
3712 SET_UP_DIR:
3713      LD           DE,0          : ZERO OUT HI ADDRESS
3714      LD           [BLOCKS_REQ+2],DE
3715      LD           [DIR_BLOCK_NO],DE
3716      XOR          A
3717      LD           [FOUND_AVAIL_ENTRY],A
3718
3719      LD           [IX+FCB_BLOCK],1 : INIT NUMBER OF FIRST DIR BLOCK
3720      LD           [IX+FCB_BLOCK+1],A : ZERO IT OUT
3721      LD           [IX+FCB_BLOCK+2],A
3722      LD           [IX+FCB_BLOCK+3],A
3723
3724      LD           [IX+FCB_START_BLOCK],1 : INIT NUMBER OF FIRST DIR BLOCK
3725      LD           [IX+FCB_START_BLOCK+1],A
3726      LD           [IX+FCB_START_BLOCK+2],A
3727      LD           [IX+FCB_START_BLOCK+3],A
3728
3729      LD           DE,[FCB_DATA_ADDR] : INIT POINTER TO BUFFER
3730      LD           [IX+FCB_POINTER],E
3731      LD           [IX+FCB_POINTER+1],D
3732
3733      LD           [BUF_START],DE   : ALSO INIT HERE WITH SAME
3734      LD           HL,1024         : CALC ADDR OF END OF BUFFER TOO
3735      ADD          HL,DE
3736      LD           [BUF_END],HL   : AND SAVE IT
3737
3738 : THIS LABEL NOT USED IN THIS ROUTINE, NOT DECLARED GLOBAL
3739 : REMOVED BECAUSE IT CONFLICTS WITH IDENTICAL LABEL ELSEWHERE
3740 : READ_TIMEOUT:
3741      LD           A,[IX+FCB_DEVICE] : GET PARAMETERS FROM FCB
3742      LD           L,[IX+FCB_POINTER]
3743      LD           H,[IX+FCB_POINTER+1]
3744      LD           E,[IX+FCB_BLOCK]
3745      LD           D,[IX+FCB_BLOCK+1]
3746      LD           C,[IX+FCB_BLOCK+2]
3747      LD           B,[IX+FCB_BLOCK+3]
3748      CALL        READ_BLOCK      : THEN READ THE FIRST BLOCK
3749
3750      JP           NZ,MAKE_ERROR   : BRANCH IF ERROR
3751
3752 SET_UP_FCB:
3753      LD           IY,[FCB_DATA_ADDR] :POINT THE BUFFER
    
```

LOCATION OBJECT CODE LINE SOURCE LINE

```

3754 *
E730 FD7EOC          LD      A,[IY+VOL_DIRSIZE] ;GET THE DIR SIZE BYTE
E733 E67F          AND     7FH           ;ZERO THE PERM BIT
                 INC     A           ;BECAUSE WE START IN BLOCK 1
                 DEC     A           ;TO SOLVE PICKET FENCE PROBLEM
E735 DD771D          LD      A,[IX+FCB_LAST_BLOCK],A ;SET THE LAST BLOCK #
E738 DD361E00        LD      [IX+FCB_LAST_BLOCK+1],0 ;OTHER BYTES WILL BE ZERO
E73C DD361F00        LD      [IX+FCB_LAST_BLOCK+2],0
E740 DD362000        LD      [IX+FCB_LAST_BLOCK+3],0
                 CALL    CHECK_IF_DIRECTORY ; CHECK IF DIRECTORY EXISTS
E744 CDF035          JP      NZ,MAKE_ERROR
E747 C2E989
E74A 0626          LD      B,ENT_PER_BLOCK-1 ;INIT ENTRY COUNT, ALLOW FOR VOL ID
                 LD      NEXT_ENT
E74C              LD      L,[IX+FCB_POINTER] ;GET CURRENT POINTER
E74F DD6E21          LD      H,[IX+FCB_POINTER+1]
E752 11001A        LD      DE,DIR_ENT_LENGTH ;GET LENGTH OF AN ENTRY
E755 19            ADD     HL,DE           ;ADVANCE POINTER TO NEXT ENTRY
E756 DD7521          LD      [IX+FCB_POINTER],L ;AND SAVE IT AGAIN
E759 DD7422          LD      [IX+FCB_POINTER+1],H
E75C              LD      CHECK_ENT
E75F DD6E21          LD      L,[IX+FCB_POINTER] ;GET POINTER INTO IY THE LONG WAY
E762 E5            LD      H,[IX+FCB_POINTER+1]
E763 FDE1          PUSH   HL
E765 FD7EOC          POP    IY
E768 4F            LD      A,[IY+DIR_ATTR] ;GET THE ATTRIBUTE BYTE FOR THIS ENTRY
E769 C847          LD      C,A           ;SAVE IT HERE FOR NOW
E76B C2E807        BIT     ATTR_HOLE_BIT,A ;CHECK IF HOLE
E76E DDCB186E      JR      NZ,FOUND_HOLE ;BRANCH IF IT'S A HOLE!!! <<-----
E772 2037          JR      MODE_REMAINDER_BIT,[IX+FCB_MODE] ; DO WE ALLOCATE REST OF TAPE?
E774 C851          LD      NZ,D_FILE    ; YUP, DON'T BOTHER CHECKING
E776 2824          BIT     ATTR_DEL_BIT,C ;IS IT A DELETED FILE?
                 JR      Z,ACTIVE_FILE ;NOPE, AN ACTIVE ONE
                 LET'S LOOK FOR DELETED FILE TO USE
E778 FD6E11          LD      L,[IY+DIR_MAX_LENGTH] ; GET ALLOCATED SIZE
E77B FD6612          LD      H,[IY+DIR_MAX_LENGTH+1]
E77E ED5BFEOC      LD      DE,[BLOCKS_REQ] ; MIN NO OF BLOCKS NEEDED
E782 87            OR     A           ;clear carry flag
E783 ED52          LD      HL,DE        ; WILL WE FIT?
E785 3824          JR      C,D_FILE   ; NOPE, LOOK FOR ANOTHER ENTRY
E787 21F0DB        LD      HL,FOUND_AVAIL_ENTRY ; FLAG BYTE
E78A C846          BIT     FOUND_ENTRY,[HL] ; DO WE ALREADY HAVE FIRST FIT?
E78C 201D          JR      NZ,D_FILE  ; YUP, DON'T EVEN CONSIDER THIS ONE
E78E C8C6          SET     FOUND_ENTRY,[HL]
E790 DD5E19        LD      E,[IX+FCB_BLOCK]
E793 DD561A        LD      D,[IX+FCB_BLOCK+1]

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
3811
E796 ED53FDD9 LD [DIR_BLOCK_NO],DE ; SAVE IT
E79A 180F JR D_FILE
3812
3813
3814
3815
3816 ; IF THIS IS A REAL FILE, WE CAN SEE IF ITS NAME MATCHES THE
3817 ; ONE WE ARE ABOUT TO MAKE -- IT'S AN ERROR IF SO.
3818
3819 ACTIVE_FILE:
E79C DD5E21 E,[IX+FCB_POINTER] ;POINT TO THIS ENTRY
E79F DD5622 D,[IX+FCB_POINTER+1]
E7A2 2AFE10 HL,[USER_NAME] ;POINT TO DESIRED NAME
E7A5 CDF054 BASECMP ;COMPARE THE BASE OF THE FILENAMES
E7A8 CAE996 Z,FILE_EXISTS ;BRANCH IF THEY'RE THE SAME
E7AB 109F DJNZ NEXT_ENT ;ELSE GO TO NEXT ENTRY (IF IT EXISTS)
3820
3821
3822
3823
3824
3825 D_FILE
3826
3827
3828
E7AD 2AFE08 LD [BUF_START] ;IF NOT RESET FCB FOR NEXT BLOCK
E7B0 DD7521 LD [IX+FCB_POINTER],L
E7B3 DD7422 LD [IX+FCB_POINTER+1],H
E7B6 DD3419 INC [IX+FCB_BLOCK] ;INC THE BLOCK -- NO CARRY OUT
3832
3833 ; DO 4-BYTE COMPARE OF FCB_BLOCK VS FCB_LAST_BLOCK TO SEE
3834 ; IF THERE'S MORE FILE LEFT.
3835
3836
E7B9 DD7E20 LD A,[IX+FCB_LAST_BLOCK+3] ;COMPARE MS BYTES
E7BC DDBE1C CP [IX+FCB_BLOCK+3]
E7BF DAE7FD JP C,FULL_DIR ;BRANCH IF BLOCK IS LARGER
E7C2 DD7E1F LD A,[IX+FCB_LAST_BLOCK+2] ;COMPARE NEXT BYTES
E7C5 DDBE1B CP [IX+FCB_BLOCK+2]
E7C8 DAE7FD JP C,FULL_DIR ;BRANCH IF BLOCK IS LARGER
E7CB DD7E1E LD A,[IX+FCB_LAST_BLOCK+1] ;COMPARE NEXT BYTES
E7CE DDBE1A CP [IX+FCB_BLOCK+1]
E7D1 DAE7FD JP C,FULL_DIR ;BRANCH IF BLOCK IS LARGER
E7D4 DD7E1D LD A,[IX+FCB_LAST_BLOCK+0] ;COMPARE LS BYTES
E7D7 DDBE19 CP [IX+FCB_BLOCK+0]
E7DA DAE7FD JP C,FULL_DIR ;BRANCH IF BLOCK IS LARGER
E7DD
3849 DO_READ_AGAIN:
3850
E7DD DD7E17 LD A,[IX+FCB_DEVICE] ;SET UP THE PARAMETERS
E7E0 DD6E21 LD L,[IX+FCB_POINTER]
E7E3 DD6622 LD H,[IX+FCB_POINTER+1]
E7E6 DD5E19 LD E,[IX+FCB_BLOCK]
E7E9 DD561A LD D,[IX+FCB_BLOCK+1]
E7EC DD4E1B LD C,[IX+FCB_BLOCK+2]
E7EF DD461C LD B,[IX+FCB_BLOCK+3]
E7F2 CDF178 READ_BLOCK ;AND READ THE NEXT BLOCK OF ENTRIES
E7F5 C2E989 NZ,MAKE_ERROR ;NOPE, A REAL ERROR
3860
3861 NO_TIMEOUT:
E7F8 0627 LD B,ENT_PER_BLOCK ;SET THE ENTRY COUNTER
E7FA C3E75C CHECK_ENT ;AND GO CHECK THE NEW ENTRY
3864
3865 FULL_DIR:
E7FD 21FDD8 LD HL,FOUND_AVAIL_ENTRY
E800 5 FOUND_ENTRY,[HL] ; E WE FOUND AN ENTRY?

```


LOCATION	OBJECT CODE	LINE	SOURCE LINE
E802	CAE99A	3868	JP Z,FULL_DIR_EXIT ; NOPE, EXIT
E805	1807	3869	JR USE_ENTRY ; YUP, CREATE NEW ENTRY THERE
		3870	
		3871	: ARRIVE HERE WHEN WE'VE FOUND THE HOLE ENTRY IN THE DIRECTORY.
		3872	: IV IS STILL POINTING TO THE ENTRY IN THE BUFFER.
		3873	
E807		3874	FOUND_HOLE
E807	21FD08	3875	LD HL,FOUND_AVAIL_ENTRY
E80A	CB46	3876	BIT FOUND_ENTRY,[HL]
E80C	2864	3877	JR Z,USE_HOLE
		3878	
E80E		3879	USE_ENTRY:
E80E	CB86	3880	RES FOUND_ENTRY,[HL] ; RESET FOR NEXT TIME
		3881	
E810	D06E19	3882	LD L,[IX+FCB_BLOCK] ; GET THE CURRENT BLOCK
E813	D0661A	3883	LD H,[IX+FCB_BLOCK+1]
		3884	
E816	ED58FD09	3885	LD DE,[DIR_BLOCK_NO] ; BLOCK NO WHERE WE HAVE ENTRY TO USE
E81A	B7	3886	OR A ; clear carry flag
E81B	ED52	3887	SBC HL,DE ; SAME BLOCK?
		3888	
E81D	2AFE08	3889	LD HL,[BUF_START] ; POINT TO START OF BLOCK
E820	D07621	3890	LD [IX+FCB_POINTER],L
E823	D07422	3891	LD [IX+FCB_POINTER+1],H
		3892	
E826	2824	3893	JR Z,GOT_BLOCK ; YUP, DON'T NEED TO READ IN ANOTHER BLOCK
		3894	
E828	D07319	3895	LD [IX+FCB_BLOCK],E ; GET THIS BLOCK
E82B	D0721A	3896	LD [IX+FCB_BLOCK+1],D
		3897	
E82E	010000	3898	LD BC,0
E831	D07E17	3899	LD A,[IX+FCB_DEVICE] ; DEVICE ID
		3900	
E834	CD178	3901	CALL READ_BLOCK
E837	C2E989	3902	JP NZ,MAKE_ERROR
E83A	1810	3903	JR GOT_BLOCK ; READ IN NEW DIRECTORY
		3904	
E83C		3905	NEXT_DIR:
E83C		3906	TOO_SMALL:
E83C	D06E21	3907	LD L,[IX+FCB_POINTER] ; GET POINTER TO DIR ENTRY
E83F	D06622	3908	LD H,[IX+FCB_POINTER+1]
		3909	
E842	11001A	3910	LD DE,DIR_ENT_LENGTH ; LENGTH OF ENTRY
		3911	
E845	19	3912	ADD HL,DE ; POINT TO NEXT ENTRY
		3913	
E846	D07521	3914	LD [IX+FCB_POINTER],L
E849	D07422	3915	LD [IX+FCB_POINTER+1],H
		3916	
E84C		3917	GOT_BLOCK:
E84C	E5	3918	HL PUSH
E84D	FDE1	3919	POP IV
		3920	
E84F	FDCB0C56	3921	BIT ATTR_DEL_BIT,[IY+DIR_ATTR] ; IS IT DELETED?
E853	28E7	3922	JR Z,NEXT_DIR ; NOPE, LOOK FOR ANOTHER DELETED ENTRY
		3923	
E855	FD6E11	3924	LD L,[IY+DIR_MAX_LENGTH] ; GET ALLOCATED AMOUNT OF DELETED FILE

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
E858	FD6612	3925	LD	H, [IY+DIR_MAX_LENGTH+1]
		3926		
E858	ED58FEOC	3927	LD	DE, [BLOCKS_REQ]
		3928		: MIN AMOUNT NEEDED
E85F	E5	3929	PUSH	HL
E860	B7	3930	OR	A
E861	ED62	3931	SBC	HL, DE
E863	E1	3932	POP	HL
		3933		
E864	38D6	3934	JR	C, TOO_SMALL
		3935		: NOPE, TRY ANOTHER ONE
E866	22FEOC	3936	LD	[BLOCKS_REQ], HL
		3937		: FORCE ALLOCATED AMOUNT
E869	CDE98F	3938	CALL	LOAD_NEW_ENTRY_INFO
E86C	C2E989	3939	JP	NZ, MAKE_ERROR
		3940		
E86F	C3E971	3942	JP	TIME_TO_WRITE
		3943		
		3944		: COMPARE FCB BLOCK TO FCB_LAST_BLOCK. GO TO MORE_BLOCKS
		3945		: IF FCB_LAST_BLOCK > FCB_BLOCK.
		3946		
		3947	USE_HOLE:	
E872	D07E1C	3947	LD	A, [IX+FCB_BLOCK+3]
E875	D08E20	3948	CP	[IX+FCB_LAST_BLOCK+3]
E878	381E	3950	JR	C, MORE_BLOCKS
E87A	D07E1B	3951	LD	A, [IX+FCB_BLOCK+2]
E87D	D08E1F	3952	CP	[IX+FCB_LAST_BLOCK+2]
E880	3816	3953	JR	C, MORE_BLOCKS
E882	D07E1A	3954	LD	A, [IX+FCB_BLOCK+1]
E885	D08E1E	3955	CP	[IX+FCB_LAST_BLOCK+1]
E888	380E	3956	JR	C, MORE_BLOCKS
E88A	D07E19	3957	LD	A, [IX+FCB_BLOCK]
E88D	D08E1D	3958	CP	[IX+FCB_LAST_BLOCK]
E890	3806	3959	JR	C, MORE_BLOCKS
		3960		
		3961		: FALL THROUGH IF THIS IS THE LAST BLOCK OF THE FILE.
		3962		
E892	78	3963	LD	A, B
E893	FE01	3964	CP	1
E895	CAE99E	3965	JP	Z, TAPE_FULL
		3966		: ERROR IF NO ENTRIES LEFT VACANT
		3967	MORE_BLOCKS	
E898		3968	LD	L, [IY+DIR_MAX_LENGTH]
E898	FD6E11	3968	LD	H, [IY+DIR_MAX_LENGTH+1]
E898	FD6612	3969		
		3970		
E89E	DDCB186E	3971	BIT	MODE_REMAINDER_BIT, [IX+FCB_MODE]
E8A2	280C	3972	JR	Z, CHECK_HOLE_SIZE
		3973		
E8A4	7C	3974	LD	A, H
E8A5	B5	3975	DR	L
E8A6	CAE99E	3976	JP	Z, TAPE_FULL
		3977		: TAPE FULL
E8A9	22FEOC	3978	LD	[BLOCKS_REQ], HL
E8AC	DDCB18AE	3979	RES	MODE_REMAINDER_BIT, [IX+FCB_MODE]
		3980		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		: NOPE
		3981		: YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
		3981		: TAPE FULL
		3981		: NOT FULL, REQUEST REST OF TAPE
		3981		

LOCATION	OBJECT CODE	LINE	SOURCE LINE
E880	ED58FE0C	3982	LD DE,[BLOCKS_REQ] ;GET THE REQUESTED SIZE
E884	B7	3983	OR A
E885	ED52	3984	SBC HL,DE ;COMPARE THEM
E887	DAE99E	3985	JP C,TAPE_FULL ;BRANCH IF THE HOLE IS TOO SMALL
E88A	22FE1A	3986	LD [NEW_HOLE_SIZE],HL ;ELSE SAVE THE RESULT
		3987	
		3988	
E88D	21FE0C	3989	; ADD BLOCKS_REQ TO DIR_START_BLOCK TO FIND THE HOLE'S NEW
E8C0	FD7E0D	3990	; START BLOCK. NOTICE HOW STOOPID THE Z80 IS WITH ADDITION!
E8C3	86	3991	LD HL,BLOCKS_REQ ;POINT TO THE NUMBER TO ADD
E8C4	23	3992	LD A,[IV+DIR_START_BLOCK]
E8C5	32FE16	3993	ADD A,[HL]
E8C8	FD7E0E	3994	INC HL
E8CB	8E	3995	LD [NEW_HOLE_START],A
E8CC	23	3996	LD A,[IV+DIR_START_BLOCK+1]
E8CD	32FE17	3997	ADC A,[HL]
E8D0	FD7E0F	3998	INC HL
E8D3	8E	3999	LD [NEW_HOLE_START+1],A
E8D4	23	4000	LD A,[IV+DIR_START_BLOCK+2]
E8D5	32FE18	4001	ADC A,[HL]
E8D8	FD7E10	4002	INC HL
E8DB	8E	4003	LD [NEW_HOLE_START+2],A
E8DC	23	4004	LD A,[IV+DIR_START_BLOCK+3]
E8DD	32FE19	4005	ADC A,[HL]
		4006	INC HL
		4007	LD [NEW_HOLE_START+3],A
E8E0	C5	4008	PUSH BC ;SAVE ENTRY COUNT (IN B)
E8E1	CDE98F	4009	CALL LOAD_NEW_ENTRY_INFO
E8E4	C1	4010	POP BC ;RESTORE ENTRY COUNT
E8E5	C2E989	4011	JP NZ,MAKE_ERROR
E8E8	D06E21	4012	LD L,[IX+FCB_POINTER] ;GET CURRENT POINTER
		4013	
		4014	
E8EB	D06E22	4015	H,[IX+FCB_POINTER+1]
E8EE	11001A	4016	LD DE,DIR_ENT_LENGTH ;GET LENGTH OF AN ENTRY
E8F1	19	4017	ADD HL,DE ;ADVANCE POINTER TO NEXT ENTRY
E8F2	D07521	4018	LD [IX+FCB_POINTER],L ;AND SAVE IT AGAIN
E8F5	D07422	4019	LD [IX+FCB_POINTER+1],H
		4020	
E8F8	1044	4021	DJNZ NOT_END ;JUMP IF THERE ARE MORE ENTRIES
		4022	
		4023	
		4024	
		4025	
		4026	
		4027	
		4028	;THIS LABEL NOT USED IN THIS ROUTINE, NOT DECLARED GLOBAL
		4029	;REMOVED BECAUSE IT CONFLICTS WITH IDENTICAL LABEL ELSEWHERE
		4030	;WRITE_AGAIN:
E8FA	DD7E17	4031	LD A,[IX+FCB_DEVICE] ;GET PARAMETERS FROM FCB
E8FD	2AFE08	4032	LD HL,[BUF_START]
E900	DD5E19	4033	LD E,[IX+FCB_BLOCK]
E903	DD561A	4034	LD D,[IX+FCB_BLOCK+1]
E906	DD4E1B	4035	LD C,[IX+FCB_BLOCK+2]
E909	DD461C	4036	LD B,[IX+FCB_BLOCK+3]
E90C	DD1E16	4037	CALL WRITE_BLOCK

```

LOCATION OBJECT CODE LINE SOURCE LINE
E90F C2E989 4039 JP NZ.MAKE_ERROR ; NOPE, A REAL LIVE ERROR
4040
4041
4042
E912 4043 RESET_THE_POINTER:
E912 2AFE08 4044 HL.[BUF_START]:RESET THE POINTER
E915 DD7521 4045 [IX+FCB_POINTER].L
E918 DD7422 4046 LD [IX+FCB_POINTER+1].H
4047
E91B DD3419 4048 INC [IX+FCB_BLOCK] ;INC BLOCK -- NO CARRY OUT
4049
E91E 4050 TIMEOUT_IN_READ:
E91E DD7E17 4051 LD A.[IX+FCB_DEVICE] ;GET PARAMETERS FROM FCB
E921 DD6E21 4052 LD L.[IX+FCB_POINTER]
E924 DD6622 4053 LD H.[IX+FCB_POINTER+1]
E927 DD5E19 4054 LD E.[IX+FCB_BLOCK]
E92A DD561A 4055 LD D.[IX+FCB_BLOCK+1]
E92D DD4E18 4056 LD C.[IX+FCB_BLOCK+2]
E930 DD461C 4057 LD B.[IX+FCB_BLOCK+3]
E933 CDF17B 4058 CALL READ_BLOCK ;AND READ THE NEXT BLOCK
4059
E936 2051 4060 JR NZ.MAKE_ERROR ; NOPE, ERROR!!!!!!
4061
4062
E938 DD6E21 4063 LD L.[IX+FCB_POINTER] ;GET POINTER INTO IY THE LONG WAY
E938 DD6622 4064 LD H.[IX+FCB_POINTER+1]
4065
E93E 4066 NOT_END:
E93E E5 4067 PUSH HL
E93F FDE1 4068 POP IY ;!!!
4069
4070 *
4071 * STILL TRYING TO CALC THE NEW START ADDR
4072 *
E941 2AFE1A 4073 LD HL.[NEW_HOLE_SIZE]
E944 FD7511 4074 LD [IY+DIR_MAX_LENGTH].L
E947 FD7412 4075 LD [IY+DIR_MAX_LENGTH+1].H
4076
E94A 3AFE16 4077 LD A.[NEW_HOLE_START]
E94D FD7700 4078 LD [IY+DIR_START_BLOCK].A
E950 3AFE17 4079 LD A.[NEW_HOLE_START+1]
E953 FD770E 4080 LD [IY+DIR_START_BLOCK+1].A
E956 3AFE18 4081 LD A.[NEW_HOLE_START+2]
E959 FD770F 4082 LD [IY+DIR_START_BLOCK+2].A
E95C 3AFE19 4083 LD A.[NEW_HOLE_START+3]
E95F FD7710 4084 LD [IY+DIR_START_BLOCK+3].A
4085
E962 FD360C01 4086 LD [IY+DIR_ATTR].ATTR_HOLE ; SET HOLE UP
4087
E966 FDE5 4088 PUSH IY
E968 D1 4089 POP DE
4090
E969 21F428 4091 LD HL,HOLE_FILE_NAME
E96C 01000C 4092 LD BC,12
E96F EDB0 4093 LDIR
4094
4095 TIME_TO_WRITE:

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

E971 D07E17 4096 LD A,[IX+FCB_DEVICE] ;GET PARAMETERS FROM FCB
E974 2AFE08 4097 LD HL,[BUF_START]
E977 D05E19 4098 E,[IX+FCB_BLOCK]
E97A D0561A 4099 D,[IX+FCB_BLOCK+1]
E97D D04E1B 4100 C,[IX+FCB_BLOCK+2]
E980 D0461C 4101 B,[IX+FCB_BLOCK+3]
E983 CDF1E6 4102 CALL WRITE_BLOCK ;WRITE THE BLOCK OUT
4103
E986 2001 4104 JR NZ,MAKE_ERROR ; IT'S A BOO BOO!!
E988 AF 4106 XOR A ;SHOW NO ERROR
4107 LETS_GET_OUT:
E989 4108 MAKE_ERROR:
4109
E989 B7 4110 OR A
E98A C1 4111 POP BC
E98B D1 4112 POP DE
E98C E1 4113 POP HL
E98D D0E1 4114 POP IX
E98F FDE1 4115 POP IY
E991 C9 4116 RET
4117
E992 4118 TOO_BIG
E992 3E08 4119 LD A,TOO_BIG_ERR ;SHOW AN ERROR
E994 18F3 4120 JR MAKE_ERROR
4121
E996 4122 FILE_EXISTS
E996 3E06 4123 LD A,FILE_EXISTS_ERR
E998 18EF 4126 JR MAKE_ERROR
4127 FULL_DIR_EXIT:
E99A 3E0C 4128 LD A,FULL_DIR_ERR
E99C 18EB 4129 JR MAKE_ERROR
4130
E99E 4131 TAPE_FULL
E99E 3E00 4132 LD A,FULL_TAPE_ERR
E9A0 18E7 4133 JR MAKE_ERROR
4134 *
4135 *
4136 * SUBROUTINE NAME: GET FILE NAME LENGTH
4137 *
4138 * ENTRY: HL- POINTS TO TEXT STRING
4139 * EXIT: Z=1-FOUND AND PROPER SIZE (1-12)
4140 * BC=BYTE COUNT
4141 * A=TRASHED
4142 *
4143 * Z=0=ERROR
4144 * BC=TRASH
4145 * A=FILE_NAME_TOO_LONG
4146 * DLS(B/28/83)
4147 *
4148 GET_FILE_NM_LEN:
E9A2 4149 PUSH HL
E9A3 060C 4150 LD B,12
E9A5 0E01 4151 LD C,1
E9A7 4152 SRCH_LOOP:

```

;SAVE THE POINTER TO FILE NAME
;SCAN UP TO 12 BYTE
;SET COUNT TO 1

HEWLETT-PACKARD: DIR_HANDL (c) Coleco 1983 Confidential

FILE: EOSABS:EOS_TF

LOCATION	OBJECT CODE	LINE	SOURCE LINE
E9A7	TE	4153	
E9A8	FE03	4154	LD A,[HL]
E9AA	2809	4155	CP O3
		4156 *	JR Z,GOT_IT
E9AC	OC	4157	INC C
E9AD	23	4158	INC HL
E9AE	10F7	4159	DJNZ SRCH_LOOP
		4160 *	
E9B0		4161	LD A,FILE_NM_ERR
E9B0	3E0E	4162	OR A
E9B2	B7	4163	POP HL
E9B3	E1	4164	RET
E9B4	C9	4165	
		4166 *	
E9B5		4167	LD GOT_IT
E9B5	79	4168	LD A,C
E9B6	FE01	4169	CP 1
E9B8	28F6	4170	JR Z,ERR_GFN
E9BA	0600	4171	LD B,0
E9BC	AF	4172	OR A
E9BD	E1	4173	POP HL
E9BE	C9	4174	RET
		4175	
		4176	

:GET FN(1)
:ETX

:ADVANCE POINTER TO FILE NAME STRING

:SHOW ERROR

:ETX ALONE IS NOT VALID

:SHOW OK

LOCATION OBJECT CODE LINE SOURCE LINE

```

4178 LOAD_NEW_ENTRY_INFO:
E9BF          PUSH_IV          :GET NAME ADDRESS INTO DE
E9BF FDE5     POP              DE
E9C1 D1
4182
E9C2 2AFE10  LD             HL,[USER_NAME] ;GET USER'S STRING ADDRESS
4184 *
E9C5 CDE9A2  CALL          GET_FILE_NM_LEN ; DLS(0/28/83)
E9C8 C2E9FF  JP             NZ,MAKE_ERR_1
4187 *
E9CB ED80    LDIR
4189
E9CD 3E10    LD             A,ATTR_USER      ;SET THE DEFAULT ATTRIBUTE
E9CF FD770C  LD             [IV+DIR_ATTR],A
4192
E9D2 ED4BFEOC LD             BC,[BLOCKS_REQ] ;GET NEW FILE'S MAX SIZE
E9D6 FD7111 LD             [IV+DIR_MAX_LENGTH],C ;PUT INTO DIR
E9D9 FD7012 LD             [IV+DIR_MAX_LENGTH+1],B
4196.
4197
E9DC FD361301 LD             [IV+DIR_USED_LENGTH],1 ;INIT COUNT OF BLOCKS USED
E9E0 FD361400 LD             [IV+DIR_USED_LENGTH+1],0
4200
E9E4 FD361500 LD             [IV+DIR_LAST_COUNT],0 ;INIT BYTECOUNT IN LAST BLOCK
E9E8 FD361600 LD             [IV+DIR_LAST_COUNT+1],0
4203
E9EC 3AFDE0  LD             A,[EDS_YEAR]    ;INSERT THE DATE
E9EF FD7717 LD             [IV+DIR_YEAR],A
E9F2 3AFDE1  LD             A,[EDS_MONTH]
E9F5 FD7718 LD             [IV+DIR_MONTH],A
E9F8 3AFDE2  LD             A,[EDS_DAY]
E9FB FD7719 LD             [IV+DIR_DAY],A
4210
E9FE AF      XOR             A
4211
4212
E9FF C9      MAKE_ERR_1:
E9FF C9      RET
4214
4215
4216 ;
4217 ;
4218 ;
4219 ;
4220 ;
4221

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		4223	: GLB
		4224	: EXT
		4225	: EXT
		4226	: EXT
		4227	: EXT
		4228	: EXT
		4229	: EXT
		4230	-----
		4231	-----
		4232	__OPEN_FILE -- Sets up an FCB for the caller to access a file.
		4233	
		4234	CALLING PARAMETERS: device number in A; address of name string
		4235	in HL; mode in B.
		4236	
		4237	EXIT PARAMETERS: if no error -- Z = 1; A = file number
		4238	if error -- Z = 0; A = error code; B = junk
		4239	-----
		4240	-----
		4241	
		4242	: NEXT_FCB CHANGED TO 0_NEXT_FCB TO PREVENT CONFLICT WITH
		4243	: IDENTICAL LABEL ELSEWHERE
		4244	-----
		4245	__OPEN_FILE
EA00	FDE5	4246	PUSH IY ; SAVE REGISTERS
EA02	E5	4247	PUSH HL ;
EA03	DOE5	4248	PUSH IX ;
EA06	D5	4249	PUSH DE ;
		4250	
EA06	F5	4251	PUSH AF ; SAVE DEVICE NUMBER
EA07	C5	4252	PUSH BC ; SAVE MODE
		4253	
		4254	: FIRST, WE HAVE TO LOCATE A FREE FCB.
EA08	DD2AFDFD	4255	LD IX, [FCB_HEAD_ADDR] ; GET POINTER TO FIRST FCB'S HEAD
EA0C	110023	4256	LD DE, FCB_LENGTH
EA0F	DD19	4257	ADD IX, DE ; SKIP OVER IT -- BELONGS TO SYSTEM
		4258	
EA11	FD2AFDFD	4259	LD IY, [FCB_DATA_ADDR] ; GET POINTER TO FIRST FCB'S BODY
EA15	110400	4260	LD DE, 1024
EA18	FD19	4261	ADD IY, DE ; SKIP OVER IT TOO
		4262	
EA1A	0601	4263	LD B, 1 ; SET FCB NUMBER
EA1C		4264	O_NEXT_FCB
EA1C	DD7E18	4265	LD A, [IX+FCB_MODE] ; GET THE MODE BYTE FROM FCB
EA1F	B7	4266	OR A ; CHECK IT
EA20	281C	4267	JR Z, GOT_ONE ; BRANCH IF THIS ONE IS FREE
		4268	
EA22	110023	4269	LD DE, FCB_LENGTH ; ELSE SKIP OVER IT
EA25	DD19	4270	ADD IX, DE
EA27	110400	4271	LD DE, 1024
EA2A	FD19	4272	ADD IY, DE
		4273	
EA2C	04	4274	INC B ; INC THE FCB NUMBER
EA2D	78	4275	LD A, B
EA2E	FE03	4276	CP ; ARE THERE MORE FCBS TO CHECK?
EA30	38EA	4277	JR C, O_NEXT_FCB ; LOOP UNTIL WE'VE SEEN THEM ALL
EA32		4278	NO_FCBS
EA32		4279	POP BC ; RESTORE P
EA32		4279	

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
EA33	F 1	4280	POP	AF
EA34	D 1	4281	POP	DE
EA35	DDE1	4282	POP	IX
EA37	E 1	4283	POP	HL
EA38	FDE1	4284	POP	IY
EA3A	3E07	4285	LD	A,NO_FCB_ERR ;SET THE ERROR CODE
EA3C	87	4286	OR	A ;SET THE CONDITIONS
EA3D	C 9	4287	RET	
EA3E		4288		
EA3E		4290	GOT_ONE	
EA3E	FD22FE08	4291	LD	[BUF_START].IY ;SAVE THE FCB BUFFER'S ADDRESS
		4292		
EA42	F 1	4293	POP	AF ;GET THE MODE
EA43	D07718	4294	LD	[IX+FCB_MODE].A ;PUT IT INTO FCB
EA46	F 1	4295	POP	AF ;GET THE DEVICE
EA47	D07717	4296	LD	[IX+FCB_DEVICE].A ;PUT IT INTO FCB
		4297		
EA4A	C 5	4298	PUSH	BC ;SAVE THE FCB NUMBER
		4299		
EA4B	E 5	4300	PUSH	HL ; ADDRESS OF NAME STRING
EA4C	D 1	4301	POP	DE
		4302		
EA4D	DDE5	4303	PUSH	IX ;POINT TO FCB NAME (ASSUME OFFSET = 0)
EA4F	E 1	4304	POP	HL
EA50	D07E17	4305	LD	A,[IX+FCB_DEVICE] ;GET DEVICE NUMBER
EA53	CDE618	4306	CALL	_QUERY_FILE ;GET THE FILE'S DIR ENTRY
EA56	C2EAF8	4307	JP	NZ,OP_ERR ;BRANCH IF THERE WAS AN ERROR
		4308		*** SET PARAMS ***
EA59	CF009	4309	CALL	MODE_CHECK ;SEE IF MODE MATCHES ATTRIBUTES
EA5C	C2EAF8	4310	JP	NZ,OP_ERR ;BRANCH IF NOT
		4311		
EA5F	D07E13	4312	LD	A,[IX+FCB_USED_LENGTH] ;CALC. & SET LAST BLOCK
EA62	D08600	4313	ADD	A,[IX+FCB_FIRST_BLOCK]
EA65	D0771D	4314	LD	[IX+FCB_LAST_BLOCK].A
EA68	D07E0E	4315	LD	A,[IX+FCB_FIRST_BLOCK+1]
EA68	D08E14	4316	ADC	A,[IX+FCB_USED_LENGTH+1]
EA6E	D0771E	4317	LD	[IX+FCB_LAST_BLOCK+1].A
EA71	D07E0F	4318	LD	A,[IX+FCB_FIRST_BLOCK+2]
EA74	CE00	4319	ADC	A,0
EA76	D0771F	4320	LD	[IX+FCB_LAST_BLOCK+2].A
EA79	D07E10	4321	LD	A,[IX+FCB_FIRST_BLOCK+3]
EA7C	CE00	4322	ADC	A,0
EAT7E	D07720	4323	LD	[IX+FCB_LAST_BLOCK+3].A
		4324		
EA81	D07E1D	4325	LD	A,[IX+FCB_LAST_BLOCK+0]
EA84	D601	4326	SUB	1
EA86	D0771D	4327	LD	[IX+FCB_LAST_BLOCK+0].A
		4328		
EA89	D07E1E	4329	LD	A,[IX+FCB_LAST_BLOCK+1]
EA8C	DE00	4330	SBC	A,0
EABE	D0771E	4331	LD	[IX+FCB_LAST_BLOCK+1].A
		4332		
EA91	D07E1F	4333	LD	A,[IX+FCB_LAST_BLOCK+2]
EA94	DE00	4334	SBC	A,0
EA96	D0771F	4335	LD	[IX+FCB_LAST_BLOCK+2].A
		4336		

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
EA99	DD7E20	4337	LD	A, [IX+FCB_LAST_BLOCK+3]
EABC	DE00	4338	SBC	A, 0
EA9E	DD7720	4339	LD	[IX+FCB_LAST_BLOCK+3], A
		4340		
EAA1	ED5BFE08	4341	LD	DE, [BUF_START] ; INIT BUFFER POINTER
EAA5	DD7321	4342	LD	[IX+FCB_POINTER], E
EAA8	DD7222	4343	LD	[IX+FCB_POINTER+1], D
		4344		
EAA8	DD7E00	4345	LD	A, [IX+FCB_FIRST_BLOCK] ; SET BLOCK TO FIRST_BLOCK
EAAE	DD7719	4346	LD	[IX+FCB_BLOCK], A
EAB1	DD7E0E	4347	LD	A, [IX+FCB_FIRST_BLOCK+1]
EAB4	DD771A	4348	LD	[IX+FCB_BLOCK+1], A
EAB7	DD7E0F	4349	LD	A, [IX+FCB_FIRST_BLOCK+2]
EABA	DD771B	4350	LD	[IX+FCB_BLOCK+2], A
EABD	DD7E10	4351	LD	A, [IX+FCB_FIRST_BLOCK+3]
EACO	DD771C	4352	LD	[IX+FCB_BLOCK+3], A
		4353		
EAC3	DD7E18	4354	LD	A, [IX+FCB_MODE] ; LOOK AT THE MODE
EAC6	E607	4355	AND	MODE_MODE
EAC8	FE02	4356	CP	MODE_WRITE ; IS IT WRITE MODE?
EACA	2827	4357	JR	Z, OPENS ; JUMP IF SO -- NO PRE-READ NECESSARY
		4358		
EACC		4359	READ_TIMEOUT:	
		4360	*	
EACC	AF	4361	XOR	IF FCB_USED_LENGTH - 1 (file size of one block)
EACD	DD8E14	4362	CP	A ; load Acc with zero
EADO	200A	4363	JR	[IX+FCB_USED_LENGTH+1] ; High order must be zero
EAD2	3C	4364	INC	NZ, NOT_SIZE_1 ; If not then past bitset
EAD3	DD8E13	4365	CP	A ; INC Acc to a 1
EAD6	2004	4366	JR	[IX+FCB_USED_LENGTH] ; Compare with low order
		4367		NZ, NOT_SIZE_1 ; If not equal then exit
		4368	*	
EAD8	DDCB18FE	4369	SET	MODE_LAST_BLOCK_BIT, [IX+FCB_MODE] ; Set it
		4370	*	EndIf
EADC		4371	NOT_SIZE_1:	
		4372		
EADC	DD7E17	4373	LD	A, [IX+FCB_DEVICE] ; GET THE DEVICE NUMBER
EADF	2AFE08	4374	LD	HL, [BUF_START] ; GET BUFFER ADDRESS
EAE2	DD5E19	4375	LD	E, [IX+FCB_BLOCK] ; GET BLOCK NUMBER
EAE5	DD561A	4376	LD	D, [IX+FCB_BLOCK+1]
EAE8	DD4E1B	4377	LD	C, [IX+FCB_BLOCK+2]
EAEB	DD461C	4378	LD	B, [IX+FCB_BLOCK+3]
EAEF	DDF178	4379	CALL	READ_BLOCK ; READ THE FIRST BLOCK
		4380		
EAF1	2005	4381	JR	NZ, OP_ERR ; IT'S AN ERROR
		4382		
EAF3	AF	4383	OPENS	
EAF3	AF	4384	XOR	A ; SHOW NO ERROR
EAF4	C1	4385	POP	BC ; GET THE FILE NUMBER IN B
EAF5	78	4386	LD	A, B
		4387		
EAF6	1805	4388	JR	REST_REGS ; RESTORE REGISTERS
		4389	POP	DE
		4390	POP	IX
		4391	POP	HL
		4392	POP	IV
		4393	RET	

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		4394		
EAF8	OP_ERR	4395	LD	
EAF8	DD361800	4396		[IX+FCB_MODE],MODE_UNUSED;MAKE THE FCB UNUSED AGAIN
		4397		
EAF8	C1	4398	POP	BC
EAF8	D1	4399	REST_REGS:	
EAF8	DOE1	4400	POP	DE
EBO0	E1	4401	POP	IX
EBO1	FDE1	4402	POP	HL
EBO3	CB	4403	POP	IV
		4404	RET	

: RESTORE REGISTERS

LOCATION OBJECT CODE LINE SOURCE LINE

```

4406 -----
4407  __CLOSE_FILE -- does the necessary clean-up and marks the FCB
4408      as unused.
4409
4410  ENTRY PARAMETERS -- file number in A.
4411
4412  EXIT PARAMETERS -- If no error -- Z = 1; A = 0
4413                  If error -- Z = 0; A = error code
4414
4415 -----
4416  __CLOSE_FILE
4417
4418  EB04 DOE5      IX      ;SAVE REGISTERS
4419  EB04 E5        HL      ;
4420  EB06 D5        PUSH   ;
4421  EB07 D5        PUSH   ;
4422  EB08 C5        PUSH   ;
4423
4424  EB09 B7        OR      ;CHECK FILE NUMBER
4425  EBOA 2850      JR      ;BRANCH IF ZERO
4426  EB0C FE03      CP      ;CHECK FOR MAX RANGE
4427  EBOE 304C     JR      ;BRANCH IF TOO LARGE
4428
4429  EB10 47        LD      ;PUT IT INTO B
4430  EB11 DD2AF0FD  LD      ;GET ADDRESS OF OTH FCB
4431  EB15 2AF0FF   LD      ;GET ADDRESS OF OTH BUFFER
4432
4433  EB18          LD      ;SET OFFSET
4434  EB18 DD19     ADD     ;ADVANCE POINTER TO NEXT FCB
4435  EB1D 110400   LD      ;DE.1024
4436  EB20 19       ADD     ;ADVANCE POINTER TO NEXT BUFFER
4437  EB21 10F5     D.JNZ  ;UNTIL WE GET TO THE ONE WE WANT
4438
4439  EB23 DD7E18   LD      ;LOOK AT THE MODE BYTE
4440  EB26 B7       OR      ;
4441  EB27 2833     JR      ;BRANCH IF FCB NOT IN USE
4442
4443  EB29 EG40     AND     ;MODE DIRTY
4444  EB28 2825     JR      ;Z.CLOSE3
4445
4446  EB2D DD22FD73 LD      ;[FILE_NAME_ADDR].IX ; POINTER TO FILE NAME IN FCB
4447  EB31          LD      ;WRITE_TIMEOUT:
4448  EB31 DD7E17   LD      ;A.[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
4449  EB34 DD5E19   LD      ;E.[IX+FCB_BLOCK] ;GET THE BLOCK NUMBER
4450  EB37 DD5B1A   LD      ;D.[IX+FCB_BLOCK+1]
4451  EB3A DD4E1B   LD      ;C.[IX+FCB_BLOCK+2]
4452  EB3D DD461C   LD      ;B.[IX+FCB_BLOCK+3]
4453
4454  EB40 CDF1E6   CALL   ;HL STILL HAS ADDRESS OF BUFFER
4455  EB43 2020     JR      ;FLUSH THE BLOCK TO TAPE
4456
4457  EB45          LD      ;CLOSE4:
4458  EB45 DD7E17   LD      ;A.[IX+FCB_DEVICE] ; GET DEVICE ID
4459  EB48 DDE5     PUSH  ;IX
4460  EB4A E1       POP   ;HL
4461  EB4B ED5BFD73 LD      ;DE.[FILE_NAME_ADDR]
4462

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
EB4F	CDE851	4463	CALL _SET_FILE
		4464	
EB52		4465	CLOSE3
EB52	DD361800	4466	LD [IX+FCB_MODE],MODE_UNUSED; MARK THE FCB EMPTY
		4467	
EB56	C1	4468	POP BC; RESTORE REGISTERS
EB57	D1	4469	POP DE
EB58	E1	4470	POP HL
EB59	DOE1	4471	POP IX
EB58	C9	4472	RET
EB5C		4473	C_ERROR
EB6C	C1	4474	POP BC; RESTORE REGISTERS
EB6D	D1	4475	POP DE
EB5E	E1	4476	POP HL
EB5F	DOE1	4477	POP IX
EB61	3E09	4478	LD A,BAD_FNUM_ERR; SET THE ERROR
EB63	B7	4479	OR A; SET THE CONDITIONS
EB64	C9	4480	RET
EB65		4481	C_ERROR2
EB65	C1	4482	POP BC; RESTORE REGISTERS
EB66	D1	4483	POP DE
EB67	E1	4484	POP HL
EB68	DOE1	4485	POP IX
EB6A	B7	4486	OR A; SET CONDITIONS
EB68	C9	4487	RET

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		4489	-----
		4490	__RESET_FILE -- rewinds the file back to the first byte.
		4491	-----
		4492	ENTRY PARAMETERS: file number in A
		4493	-----
		4494	EXIT PARAMETERS: no error -- Z = 1; A = 0
		4495	if error -- Z = 0; A = error code
		4496	-----
		4497	-----
		4498	-----
		4499	-----
		4500	__RESET_FILE
EB6C	DDE5	4501	PUSH IX ; SAVE REGISTERS
EB6E	D5	4502	PUSH DE
EB6F	C5	4503	PUSH BC
EB70	E5	4504	PUSH HL
		4505	OR A ; CHECK FILE NUMBER
EB71	B7	4506	Z,R_ERROR ; BRANCH IF ZERO
EB72	CAEC07	4507	JP ; CHECK FOR MAX RANGE
EB75	FE03	4508	CP NUM_FCBS ; BRANCH IF TOO LARGE
EB77	D2EC07	4509	JP NC,R_ERROR
		4510	
		4511	B,A ; PUT IT INTO B
EB7A	47	4511	LD ;
EB7B	DD2AFDFD	4512	LD IX,[FCB_HEAD_ADDR] ;GET ADDRESS OF OTH FCB
EB7F	2AFDFD	4513	LD HL,[FCB_DATA_ADDR] ;GET ADDRESS OF OTH BUFFER
		4514	RES1
EB82		4515	LD DE,FCB_LENGTH ; SET OFFSET
EB85	DD19	4516	ADD IX,DE ; ADVANCE POINTER TO NEXT FCB
EB87	110400	4517	LD DE,1024 ; SET BUFFER SIZE
EB8A	19	4518	ADD HL,DE ; ADVANCE POINTER TO NEXT BUFFER
EB8B	10F5	4519	DJNZ RES1 ; UNTIL WE GET TO THE ONE WE WANT
		4520	
EB8D	DD7E18	4521	LD A,[IX+FCB_MODE] ; LOOK AT THE MODE BYTE
EB90	B7	4522	OR A
EB91	2874	4523	JR Z,R_ERROR ; BRANCH IF FCB NOT IN USE
		4524	
EB93	E640	4525	AND MODE_DIRTY ; ZERO UNWANTED BITS
EB95	2818	4526	JR Z,RES2 ; BRANCH IF BUFFER NOT DIRTY
		4527	
		4528	TIMED_OUT:
EB97	DD7E17	4529	LD A,[IX+FCB_DEVICE] ; GET THE DEVICE NUMBER
EB9A	DD5E19	4530	E,[IX+FCB_BLOCK] ; GET THE BLOCK NUMBER
EB9D	DD561A	4531	D,[IX+FCB_BLOCK+1]
EBA0	DD4E18	4532	C,[IX+FCB_BLOCK+2]
EBA3	DD461C	4533	B,[IX+FCB_BLOCK+3]
		4534	HL STILL HAS ADDRESS OF BUFFER
EBA6	DDF1E6	4535	CALL WRITE_BLOCK ; FLUSH THE BLOCK TO TAPE
EBA9	2065	4536	JR NZ,R_ERROR2 ; BRANCH IF ERROR
		4537	
		4538	RESET_BIT:
EBAB	DDCB1886	4539	RES MODE_DIRTY_BIT,[IX+FCB_MODE] ; CLEAR THE DIRTY BIT
EBAF	DDCB188E	4540	RES MODE_LAST_BLOCK_BIT,[IX+FCB_MODE] ; BACK TO FIRST BLOCK
EBB3	DD7E00	4542	A,[IX+FCB_FIRST_BLOCK] ; SET BLOCK TO FIRST_BLOCK
EBB6	DD7719	4543	[IX+FCB_BLOCK],A
EBB9	DD77	4544	A,[IX+FCB_FIRST_BLOCK+1]
EBBC		4545	[IX+FCB_BLOCK+1],A

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
EBBF	DD7E0F	4546	LD	A,[IX+FCB_FIRST_BLOCK+2]
EBC2	DD771B	4547	LD	[IX+FCB_BLOCK+2],A
EBC5	DD7E10	4548	LD	A,[IX+FCB_FIRST_BLOCK+3]
EBC8	DD771C	4549	LD	[IX+FCB_BLOCK+3],A
EBCB	DD7521	4550	LD	[IX+FCB_POINTER+0],L
EBCE	DD7422	4551	LD	[IX+FCB_POINTER+1],H
		4552	LD	
		4553	LD	
EBD1	DD7E18	4554	LD	A,[IX+FCB_MODE] ;GET THE MODE AGAIN
EBD4	E607	4555	AND	MODE_MODE ;LOOK AT MODE TYPE ONLY
EBD6	FE02	4556	CP	MODE_WRITE ;ARE WE WRITING?
EBD8	2J12	4557	JR	NZ,READ_IT ;NOPE, NEED TO PRE-READ
		4558		
EBDA	DD361500	4559	LD	[IX+FCB_LAST_COUNT],0
EBDE	DD361600	4560	LD	[IX+FCB_LAST_COUNT+1],0
		4561	LD	
EBE2	DD361301	4562	LD	[IX+FCB_USED_LENGTH],1
EBE6	DD361400	4563	LD	[IX+FCB_USED_LENGTH+1],0
		4564		
EBEA	1814	4565	JR	NO_READ
		4566		
EBEC		4567	READ_IT:	
EBEC	DD7E17	4568	LD	A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
EBEF	DD5E19	4569	LD	E,[IX+FCB_BLOCK] ;GET THE BLOCK NUMBER
EBF2	DD561A	4570	LD	D,[IX+FCB_BLOCK+1]
EBF5	DD4E1B	4571	LD	C,[IX+FCB_BLOCK+2]
EBF8	DD461C	4572	LD	B,[IX+FCB_BLOCK+3]
		4573		
EBFB	DDF17B	4574	CALL	READ_BLOCK ;HL STILL HAS ADDRESS OF BUFFER
		4575		
EBFE	2010	4576	JR	NZ,R_ERROR2 ;BRANCH ON ERROR
		4577		
EC00		4578	NO_READ	
EC00	E1	4579	POP	HL ;RESTORE THE REGISTERS
EC01	C1	4580	POP	BC ;RESTORE THE REGISTERS
EC02	D1	4581	POP	DE ;RESTORE THE REGISTERS
EC03	DDE1	4582	POP	IX ;RESTORE THE REGISTERS
EC05	AF	4583	XOR	A ;SHOW NO ERROR
EC06	C9	4584	RET	
		4585		
EC07		4586	R_ERROR	
EC07	E1	4587	POP	HL ;RESTORE THE REGISTERS
EC08	C1	4588	POP	BC ;RESTORE THE REGISTERS
EC09	D1	4589	POP	DE ;RESTORE THE REGISTERS
EC0A	DDE1	4590	POP	IX ;RESTORE THE REGISTERS
EC0C	3E09	4591	LD	A,BAD_FNUM_ERR ;SHOW THE ERROR
EC0E	B7	4592	OR	A ;SET CONDITIONS
EC0F	C9	4593	RET	
		4594	R_ERROR2	
EC10		4594	R_ERROR2	
EC10	E1	4595	POP	HL ;RESTORE REGISTERS
EC11	C1	4596	POP	BC ;RESTORE REGISTERS
EC12	D1	4597	POP	DE ;RESTORE REGISTERS
EC13	DDE1	4598	POP	IX ;RESTORE REGISTERS
EC15	B7	4599	OR	A ;SET CONDITION
EC16	C9	4600	RET	
		4601		
		4602		

LOCATION OBJECT CODE LINE SOURCE LINE

4603 ;
4604 ;
4605 ;
4606 ;


```

LOCATION OBJECT CODE LINE SOURCE LINE
4608
4609 ;Packed some code to allow room for
4610
4611 ;GLB READ_FILE,WRITE_FILE
4612 :EXT BUF_START,BUF_END,USER_BUF,BYTES_REQ,BYTES_TO_GO
4613 :EXT FNJM
4614 :EXT FCB HEAD ADDR,FCB_DATA_ADDR
4615 :EXT READ_BLOCK,WRITE_BLOCK
4616 :EXT MODE_CHECK
4617
4618 -----
4619
4620 ;__READ_FILE -- read some data from a file into the user's buffer.
4621
4622 ENTRY PARAMETERS: device number in A; buffer address in HL;
4623 desired number of bytes in BC.
4624
4625 EXIT PARAMETERS: no errors -- Z = 1; A = 0
4626 if errors -- Z = 0; A = error code;
4627 BC = number of bytes transferred
4628
4629 -----
4630
4631 ;GET_OUT CHANGED TO R_GET_OUT TO PREVENT CONFLICT WITH
4632 ;IDENTICAL LABEL ELSEWHERE
4633
4634 __READ_FILE BC ;SAVE REGISTERS
4635 PUSH DE
4636 PUSH HL
4637 PUSH IX
4638
4639
4640 LD [USER_BUF],HL ;SAVE USER'S BUFFER ADDRESS
4641 LD [BYTES_REQ],BC ;SAVE REQUESTED COUNT
4642 LD [FNJM],A ;SAVE FILE NUMBER
4643 LD [BYTES_TO_GO],BC ;INIT COUNTER (COUNTS DOWN)
4644
4645 OR A ;CHECK FILE NUMBER
4646 JP Z,READ_ERR2 ;BRANCH IF ZERO
4647 CP NUM_FCBS ;CHECK FOR MAX RANGE
4648 JP NC,READ_ERR2 ;BRANCH IF TOO LARGE
4649
4650 LD B,A ;PUT IT INTO B
4651 LD IX,[FCB_HEAD_ADDR] ;GET ADDRESS OF OTH FCB
4652 LD HL,[FCB_DATA_ADDR] ;GET ADDRESS OF OTH BUFFER
4653 READ1
4654 LD DE,FCB_LENGTH ;ADVANCE POINTER TO NEXT FCB
4655 ADD IX,DE
4656 LD DE,1024 ;ADVANCE POINTER TO NEXT BUFFER
4657 ADD HL,DE
4658 DJNZ READ1 ;UNTIL WE GET TO THE ONE WE WANT
4659
4660 LD [BUF_START],HL ;SAVE ADDRESS OF MY BUFFER
4661 ADD HL,DE ;POINT HL PAST END OF MY BUFFER
4662 LD [BUF_END],HL ;SAVE IT TOO
4663
4664

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE	
EC4D	DDE5	4665	PUSH	IX	
EC4F	E1	4666	POP	HL	
		4667			
EC50	CDFO09	4668	CALL	MODE_CHECK	:SEE IF THE MODE IS OK
EC53	C2E081	4669	JP	NZ,READ_ERR1	:BRANCH IF NOT
		4670			
EC56		4671	READ3		
EC56	DDCB187E	4672	BIT	MODE_LAST_BLOCK_BIT,[IX+FCB_MODE]	:LOOK AT LAST BLOCK BIT
EC5A	C2ED2B	4673	JP	NZ,READ_LAST	:BRANCH IF THIS IS THE LAST FILE BLOCK
		4674			
EC5D	2AFE0A	4675	LD	HL,[BUF_END]	:GET END ADDRESS OF BUFFER
EC60	DD5E21	4676	LD	E,[IX+FCB_POINTER]	:GET POINTER INTO BUFFER
EC63	DD5622	4677	LD	D,[IX+FCB_POINTER+1]	
EC66	B7	4678	OR	A	
EC67	ED52	4679	SBC	HL,DE	:CALC. BYTES LEFT IN BUFFER
EC69	2825	4680	JR	Z,_O_BYTES_LEFT2	
		4681			
EC68	44	4682	LD	B,H	:SAVE BYTES LEFT IN CASE WE NEED IT
EC6C	4D	4683	LD	C,L	
		4684			
EC6D	ED58FE04	4685	LD	DE,[BYTES_TO_GO]	:GET BYTES LEFT TO SEND
EC71	B7	4686	OR	A	
EC72	ED52	4687	SBC	HL,DE	:COMPARE TO WHAT WE HAVE
EC74	D2ECEA	4688	JP	NC,ENOUGH	:BRANCH IF WE HAVE ENOUGH DATA HERE
		4689			
		4690			: IF WE FALL THROUGH HERE, WE KNOW WE WILL HAVE TO GET ANOTHER
		4691			: BLOCK AFTER THIS ONE. FIRST, LET'S TRANSFER WHAT WE HAVE.
		4692			
EC77	2AFE04	4693	LD	HL,[BYTES_TO_GO]	:UPDATE NUMBER OF BYTES TO GO
EC7A	B7	4694	OR	A	
EC7B	ED42	4695	SBC	HL,BC	:SUBTRACT NUM OF BYTES WE ARE SENDING HERE
EC7D	22FE04	4696	LD	[BYTES_TO_GO],HL	
		4697			
EC80	DD6E21	4698	LD	L,[IX+FCB_POINTER]	:POINT INTO MY BUFFER
EC83	DD6622	4699	LD	H,[IX+FCB_POINTER+1]	
EC86	ED58FE06	4700	LD	DE,[USER_BUF]	:POINT INTO THE CALLER'S BUFFER
		4701			:BC HAS COUNT FROM ABOVE
EC8A	ED80	4702	LDIR		:GIVE WHAT WE HAVE
		4703			
EC8C	ED53FE06	4704	LD	[USER_BUF],DE	:POINT TO NEXT SPOT
		4705			
EC90		4706			_O_BYTES_LEFT2:
EC90	2AFE08	4707	LD	HL,[BUF_START]	:RESET THE POINTER
EC93	DD7521	4708	LD	[IX+FCB_POINTER],L	
EC96	DD7422	4709	LD	[IX+FCB_POINTER+1],H	
		4710			
EC99	DD3419	4711	INC	[IX+FCB_BLOCK]	:INC THE BLOCK NUMBER
EC9C	200D	4712	JR	NZ,READ4	:BRANCH IF NO CARRY OUT
EC9E	DD341A	4713	INC	[IX+FCB_BLOCK+1]	
ECA1	2008	4714	JR	NZ,READ4	
ECA3	DD341B	4715	INC	[IX+FCB_BLOCK+2]	
ECA6	2003	4716	JR	NZ,READ4	
ECA8	DD341C	4717	INC	[IX+FCB_BLOCK+3]	
		4718			
ECAB		4719	READ4		
		4720			
ECAB		4721	LD	A,[IX+FCB_DEVICE]	:DEVICE ID

```

LOCATION OBJECT CODE LINE SOURCE LINE
ECAA 2AFE08 4722 LD HL,[BUF_START] ; GET ADDRESS OF USERS BUFFER
ECB1 D05E19 4723 LD E,[IX+FCB_BLOCK] ; GET BLOCK NUMBER
ECB4 D0561A 4724 LD D,[IX+FCB_BLOCK+1]
ECB7 D04E1B 4725 LD C,[IX+FCB_BLOCK+2]
ECBA D0461C 4726 LD B,[IX+FCB_BLOCK+3]
4727
ECBD CDF17B 4728 CALL READ_BLOCK ;READ THE NEXT BLOCK
ECC0 C2ED81 4729 JP NZ,READ_ERR1 ;BRANCH IF ERROR
4731
ECC3 4732 NOT TAPE1:
ECC3 D07E20 4733 LD A,[IX+FCB_LAST_BLOCK+3] ;SEE IF THIS IS THE LAST BLOCK
ECC8 D08E1C 4734 CP [IX+FCB_BLOCK+3]
ECC9 201C 4735 JR NZ,JP_TO_READ3 ;BRANCH IF NOT
ECCB D07E1F 4736 LD A,[IX+FCB_LAST_BLOCK+2]
ECCE D08E1B 4737 CP [IX+FCB_BLOCK+2]
ECD1 2014 4738 JR NZ,JP_TO_READ3
ECD3 D07E1E 4739 LD A,[IX+FCB_LAST_BLOCK+1]
ECD6 D08E1A 4740 CP [IX+FCB_BLOCK+1]
ECD9 200C 4741 JR NZ,JP_TO_READ3
ECDB D07E1D 4742 LD A,[IX+FCB_LAST_BLOCK]
ECDE D08E19 4743 CP [IX+FCB_BLOCK]
ECE1 2004 4744 JR NZ,JP_TO_READ3
ECE3 D0CB18FE 4745 SET MODE_LAST_BLOCK_BIT,[IX+FCB_MODE] ;ELSE SET THE LAST BLOCK BIT
ECE7 4746 JP_TO_READ3:
ECE7 C3EC86 4747 JP READ3 ;AND DEAL WITH THE NEXT BLOCK
4748
4749 ; ARRIVE HERE IF THIS BLOCK CONTAINS ENOUGH BYTES TO SATISFY
4750 ; THIS CALL.
4751
4752 ENOUGH:
ECEA D07E20 4753 LD A,[IX+FCB_LAST_BLOCK+3] ;SEE IF THIS IS THE LAST BLOCK
ECED D08E1C 4754 CP [IX+FCB_BLOCK+3]
ECFO 2018 4755 JR NZ,READ_LAST1 ;BRANCH IF NOT
ECF2 D07E1F 4756 LD A,[IX+FCB_LAST_BLOCK+2]
ECF5 D08E1B 4757 CP [IX+FCB_BLOCK+2]
ECF8 2010 4758 JR NZ,READ_LAST1
ECFA D07E1E 4759 LD A,[IX+FCB_LAST_BLOCK+1]
ECFD D08E1A 4760 CP [IX+FCB_BLOCK+1]
ED00 2008 4761 JR NZ,READ_LAST1
ED02 D07E1D 4762 LD A,[IX+FCB_LAST_BLOCK]
ED05 D08E19 4763 CP [IX+FCB_BLOCK]
ED08 281D 4764 JR Z,LAST_BLOCK
4765
ED0A 4766 READ_LAST1:
ED0A ED48FE04 4767 LD BC,[BYTES_TO_GO] ; GET BYTE COUNT TO MOVE
4768
4769
ED0E D06E21 4770 LD L,[IX+FCB_POINTER] ;POINT INTO MY BUFFER
ED11 D06622 4771 LD H,[IX+FCB_POINTER+1]
ED14 ED58FE06 4772 LD DE,[USER_BUF] ;POINT TO THE USER'S BUFFER
ED18 ED80 4773 LDIR ;MOVE THE DATA
4774
ED1A D07521 4775 LD [IX+FCB_POINTER].L ;UPDATE MY POINTER
ED1D D07422 4776 LD [IX+FCB_POINTER+1].H
4777
4778

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
ED20	DOE1	4779	POP
ED22	E1	4780	POP
ED23	D1	4781	DE
ED24	C1	4782	POP
ED25	AF	4783	XOR
ED26	C9	4784	RET
		4785	
ED27		4786	: ARRIVE HERE IF THIS IS THE LAST BLOCK OF THE FILE.
ED27	DOCB18FE	4787	LAST_BLOCK:
		4788	SET
		4789	
ED28		4790	READ_LAST
ED28	2AFE08	4791	LD
ED2E	DO8E15	4792	LD
ED31	DO8E16	4793	LD
ED34	19	4794	ADD
ED36	DO8E21	4795	LD
ED38	DO8E22	4796	LD
ED38	B7	4797	OR
ED3C	ED82	4798	SBC
ED3E	44	4799	LD
ED3F	40	4800	LD
ED40	282A	4801	JR
		4802	
ED42	ED8BFE04	4803	LD
ED46	B7	4804	OR
ED47	ED52	4805	SBC
ED49	308F	4806	JR
		4807	
ED4B	2AFE04	4808	LD
ED4E	B7	4809	OR
ED4F	ED42	4810	SBC
ED51	22FE04	4811	LD
		4812	
ED54	DO6E21	4813	LD
ED57	DO6E22	4814	LD
ED5A	ED6BFE06	4815	LD
		4816	
ED5E	ED80	4817	LDJR
		4818	
ED60	2AFE02	4819	LD
ED63	ED4BFE04	4820	LD
ED67	B7	4821	OR
ED68	ED42	4822	SBC
ED6A	44	4823	LD
ED6B	40	4824	LD
		4825	
ED6C		4826	_O_BYTES_LEFT:
ED6C	3E0A	4827	LD
ED6E		4828	R_GET_OUT:
ED6E	DOE1	4829	POP
ED70	E1	4830	POP
ED71	D1	4831	POP
ED72	33	4832	INC
ED73	33	4833	INC
		4834	
ED74	B7	4835	DR
			A

: RESTORE REGISTERS

: SHOW NO ERROR

MODE_LAST_BLOCK_BIT.[IX+FCB_MODE]

HL.[BUF_START] :POINT TO MY BUFFER
 E.[IX+FCB_LAST_COUNT] :GET # BYTES IN IT
 D.[IX+FCB_LAST_COUNT+1]
 HL,DE :CALC ADDR OF LAST BYTE + 1
 E.[IX+FCB_POINTER] :GET CURRENT POINTER
 D.[IX+FCB_POINTER+1]
 A :CLEAR CARRY
 HL,DE :CALC # BYTES LEFT IN BUFFER
 B,H :SAVE IT IN CASE IT'S NEEDED
 C,L
 Z,_O_BYTES_LEFT

DE.[BYTES_TO_GO] :GET REQUESTED COUNT
 A :CLEAR CARRY (UGLY Z80)
 HL,DE :SEE IF WE HAVE ENOUGH BYTES
 NC,READ_LAST1 :BRANCH IF SO -- NO SWEAT

HL.[BYTES_TO_GO] :UPDATE NUMBER OF BYTES TO GO
 A
 HL,BC :SUBTRACT NUM OF BYTES WE ARE SENDING HERE
 [BYTES_TO_GO],HL

L.[IX+FCB_POINTER] :GET POINTER TO MY BUFFER
 H,[IX+FCB_POINTER+1]
 DE.[USER_BUF] :GET POINTER TO USER'S BUFFER
 :BC HAS COUNT FROM ABOVE
 :MOVE WHAT WE HAVE TO USER

HL.[BYTES_REQ] :GET # BYTES REQUESTED
 BC.[BYTES_TO_GO] :GET # BYTES WE DIDN'T DO
 A
 HL,BC :CALC. # OF BYTES WE DID DO
 B,H :PUT IT INTO BC
 C,L

A.EOF_ERR :SHOW AN ERROR
 IX :RESTORE REGISTERS

:DISCARD OLD BC

LOCATION	OBJECT CODE	LINE	SOURCE LINE
ED75	C9	4836	RET
ED76		4837	
ED76	D0E1	4838	RO_STAT_ERR:
ED78	E1	4839	POP
ED79	D1	4840	HL
ED7A	C1	4841	DE
		4842	POP
		4843	BC
ED7B	3E16	4844	A,RO_TP_STAT_ERR
ED7D	B7	4845	A
ED7E	C9	4846	RET
ED7F		4847	
ED7F	3E09	4848	READ_ERR2:
		4849	LD
		4850	A,BAD_FNUM_ERR
ED81		4851	READ_ERR1
ED81	2AFE02	4852	LD
ED84	ED4BFEO4	4853	LD
ED88	B7	4854	OR
ED89	ED42	4855	SBC
ED8B	44	4856	LD
ED8C	4D	4857	LD
ED8D	18DF	4858	JR
		4859	R_GET_OUT
		4860	
		4861 *	POP
		4862 *	POP
		4863 *	POP
		4864 *	INC
		4865 *	INC
		4866 *	OR
		4867 *	RET
		4868 *	
		4869 *	READ_ERR2
		4870 *	LD
		4871 *	LD
		4872 *	OR
		4873 *	SBC
		4874 *	LD
		4875 *	LD
		4876 *	
		4877 *	POP
		4878 *	POP
		4879 *	POP
		4880 *	INC
		4881 *	INC
		4882 *	LD
		4883 *	OR
		4884 *	RET
			HL,[BYTES_REQ] ;GET # BYTES REQUESTED
			BC,[BYTES_TO_GO] ;GET # BYTES WE DIDN'T DO
			A
			HL,BC ;CALC. # OF BYTES WE DID DO
			B,H ;PUT IT INTO BC
			C,L
			R_GET_OUT
			IX
			HL
			DE
			SP
			SP
			A
			HL,[BYTES_REQ] ;GET # BYTES REQUESTED
			BC,[BYTES_TO_GO] ;GET # BYTES WE DIDN'T DO
			A
			HL,BC ;CALC. # OF BYTES WE DID DO
			B,H ;PUT IT INTO BC
			C,L
			IX
			HL
			DE
			SP
			SP
			A,BAD_FNUM_ERR ;SET THE ERROR CODE
			A ;SET CONDITIONS

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		4886	-----
		4887	-----
		4888	WRITE_FILE -- write some bytes to a file.
		4889	
		4890	ENTRY PARAMETERS: file number in A; byte count in BC; pointer
		4891	to data in HL.
		4892	
		4893	EXIT PARAMETERS: no errors -- Z = 1; A = 0
		4894	if errors -- Z = 0; A = error code
		4895	-----
		4896	-----
		4897	-----
		4898	WRITE_FILE BC ;SAVE REGISTERS
ED8F	C5	4899	PUSH DE
ED90	D8	4900	PUSH HL
ED91	E8	4901	PUSH IX
ED92	DOE5	4902	
ED94	22FE06	4903	LD [USER_BUF],HL ;SAVE USER'S BUFFER ADDRESS
ED97	ED43FE02	4904	LD [BYTES_REQ],BC ;SAVE REQUESTED COUNT
ED98	32FE01	4905	LD [FNAM],A ;SAVE FILE NUMBER
ED9E	ED43FE04	4906	LD [BYTES_TO_GO],BC ;INIT COUNTER (COUNTS DOWN)
ED9E		4907	
ED9E		4908	
EDA2	B7	4909	OR A ;CHECK FILE NUMBER
EDA3	CAEEBC	4910	JP Z,WRITE_ERR2 ;BRANCH IF ZERO
EDA6	FE03	4911	CP NUM_FCB5 ;CHECK FOR MAX RANGE
EDAB	D2EEBC	4912	JP NC,WRITE_ERR2 ;BRANCH IF TOO LARGE
EDAB	47	4913	
EDAB	47	4914	LD B,A ;PUT IT INTO B
EDAC	D02AFDFD	4915	LD IX,[FCB_HEAD_ADDR] ;GET ADDRESS OF OTH FCB
EDB0	110023	4916	LD DE,FCB_LENGTH ;SET OFFSET
EDB3		4917	WRITE1
EDB3	D019	4918	ADD IX,DE ;ADVANCE POINTER TO NEXT FCB
EDB5	10FC	4919	DJNZ WRITE1 ;UNTIL WE GET TO THE ONE WE WANT
EDB7	47	4920	
EDB7	47	4921	LD B,A ;POINT HL AT MY BUFFER
EDB8	110400	4922	LD DE,1024
EDB8	2AFDFD	4923	LD HL,[FCB_DATA_ADDR]
EDBE	19	4924	WRITE2
EDBE	19	4925	ADD HL,DE
EDBF	10FD	4926	DJNZ WRITE2
EDC1	22FE08	4927	
EDC4	19	4928	LD [BUF_START],HL ;SAVE ADDRESS OF MY BUFFER
EDC5	22FE0A	4929	ADD HL,DE ;POINT HL PAST END OF MY BUFFER
EDC5	22FE0A	4930	LD [BUF_END],HL ;SAVE IT TOO
EDC5		4931	
EDC5		4932	
EDC8	DDE5	4933	PUSH IX ;*** CHECK FOR MODE ***
EDCA	E1	4934	POP HL
EDCB	CDFO09	4935	
EDCE	C2EEBE	4936	CALL MODE_CHECK
EDCE		4937	JP NZ,WRITE_ERR1 ;JUMP IF WRONG MODE
EDCE		4938	
EDD1		4939	WRITE3
EDD1	D07E14	4940	LD A,[IX+FCB_USED_LENGTH+1]
EDD4	DBE12	4941	CP [IX+FCB_MAX_LENGTH+1]
EDD7	37	4942	JR C,SET_UP

LOCATION	OBJECT CODE	LINE	SOURCE LINE
EDD9	D07E13	4943	LD A,[IX+FCB_USED_LENGTH+0]
EDDC	D0BE11	4944	CP [IX+FCB_MAX_LENGTH]
EDDF	3804	4945	JR C,SET_UP
		4946	
EDE1	DDCB18FE	4947	SET MODE_LAST_BLOCK_BIT,[IX+FCB_MODE]
EDES		4948	SET_UP:
EDES	2AFE0A	4949	LD HL,[BUF_END] ;GET LAST ADDR + 1 OF BUFFER
EDF8	D05E21	4950	LD E,[IX+FCB_POINTER] ;GET CURRENT POINTER
EDEB	D05E22	4951	LD D,[IX+FCB_POINTER+1]
EDEE	B7	4952	OR A ;CLEAR CARRY
EDEF	ED52	4953	SBC HL,DE ;CALC. ROOM LEFT IN BUFFER
EDF1	44	4954	LD B,H ;SAVE IT IN CASE WE NEED IT
EDF2	4D	4955	LD C,L
EDF3	ED58FE04	4956	LD DE,[BYTES_TO_GO] ;GET BYTES LEFT TO TRANSFER
EDF7	B7	4957	OR A
EDF8	ED52	4958	SBC HL,DE ;SEE IF WE HAVE ROOM FOR IT ALL
EDFA	D2EE93	4959	JP NC,ROOM_OK ;BRANCH IF YES
		4960	
		4961	; WE FALL THROUGH HERE IF THE NUMBER OF BYTES TO BE WRITTEN
		4962	; IS MORE THAN THE SPACE WE HAVE IN THE CURRENT BLOCK.
		4963	; FILL THE BLOCK, WRITE IT OUT, THEN DECIDE WHAT TO DO.
		4964	
EDFD	78	4965	LD A,B ; CHECK IF ANY BYTES AVAILABLE
EDFE	B1	4966	OR C
EDFF	2818	4967	JR Z,CHECK_LAST_BLOCK ; NOPE, EMPTY BUFFER FIRST
		4968	
EE01	2AFE04	4969	LD HL,[BYTES_TO_GO] ;GET THE COUNT
EE04	B7	4970	OR A
EE06	ED42	4971	SBC HL,BC ;SUB THE ROOM IN BUFFER TO UPDATE IT
EE07	22FE04	4972	LD [BYTES_TO_GO],HL
		4973	
EE0A	D05E21	4974	LD E,[IX+FCB_POINTER] ;GET POINTER INTO BUFFER
EE0D	D05E22	4975	LD D,[IX+FCB_POINTER+1]
EE10	2AFE06	4976	LD HL,[USER_BUF] ;GET POINTER TO USER DATA
		4977	;BC STILL HAS COUNT FROM ABOVE
		4978	;MOVE DATA INTO THE BUFFER
EE13	ED80	4979	LDIR
		4980	LD [USER_BUF],HL ; SAVE WHERE WE LEFT OFF
EE15	22FE06	4981	
		4982	
EE18	DDCB18F6	4983	SET MODE_DIRTY_BIT,[IX+FCB_MODE] ;SET THE DIRTY BIT
		4984	
EE1C		4985	CHECK_LAST_BLOCK:
EE1C	DDCB187E	4986	LD MODE_LAST_BLOCK_BIT,[IX+FCB_MODE] ;CHECK FOR LAST BLOCK
EE20	2809	4987	JR Z,MORE_FILE ;JUMP IF THIS ISN'T LAST BLOCK
		4988	
		4989	; IF THIS IS THE LAST BLOCK (AND IT'S NOW FULL) THERE'S NOTHING
		4990	; MORE WE CAN DO. WE LEAVE THE DATA IN PLACE (IT WILL GET
		4991	; FLUSHED WHEN THE FILE IS CLOSED) AND REPORT AN EOF ERROR.
		4992	; MEANING THAT WE COULDN'T WRITE ALL THE DATA.
		4993	
EE22	DDE1	4994	POP IX ;RESTORE REGISTERS
EE24	E1	4995	POP HL
EE25	D1	4996	POP DE
EE26	C1	4997	POP BC
EE27	3E0A	4998	LD A,EOF_ERR ;SHOW THE ERROR
EE29	B7	4999	OR A ;SET CONDITIONS

```

LOCATION OBJECT CODE LINE SOURCE LINE
EE2A C9
5000 RET
5001
5002 ; WE COME HERE IF WE HAVE A FULL BUFFER AND THERE'S MORE
5003 ; DATA TO WRITE AND THERE ARE MORE BLOCKS TO THE FILE.
5004
5005 MORE_FILE
5006 INC [IX+FCB_USED_LENGTH] ; INCREMENT SECTORS WRITTEN TO
5007 JR NZ,MORE_FILE1
5008 INC [IX+FCB_USED_LENGTH+1]
5009 MORE_FILE1:
5010 LD A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
5011 LD HL,[BUF_START] ;GET BUFFER ADDRESS
5012 LD E,[IX+FCB_BLOCK] ;GET BLOCK NUMBER
5013 LD D,[IX+FCB_BLOCK+1]
5014 LD C,[IX+FCB_BLOCK+2]
5015 LD B,[IX+FCB_BLOCK+3]
5016 CALL WRITE_BLOCK ;WRITE OUT THE FULL BLOCK
5017
5018 JP NZ,WRITE_ERR1 ;BRANCH IF ERROR
5019
5020 CLEAR_DIRTY_BIT:
5021 RES MODE_DIRTY_BIT,[X+FCB_MODE] ;CLEAR THE DIRTY BIT
5022 LD HL,[BUF_START] ;RESET THE BUFFER POINTER
5023 LD [IX+FCB_POINTER],L
5024 LD [IX+FCB_POINTER+1],H
5025
5026 LD [IX+FCB_LAST_COUNT],0
5027 LD [IX+FCB_LAST_COUNT+1],0
5028
5029 INC [IX+FCB_BLOCK] ;INC THE BLOCK NUMBER
5030 JR NZ,WRITE4 ;BRANCH IF NO CARRY OUT
5031 INC [IX+FCB_BLOCK+1]
5032 JR NZ,WRITE4
5033 INC [IX+FCB_BLOCK+2]
5034 JR NZ,WRITE4
5035 INC [IX+FCB_BLOCK+3]
5036
5037 WRITE4
5038 LD A,[IX+FCB_MODE] ;LOOK AT THE MODE BYTE
5039 AND MODE_MODE ;GET RID OF EXTRA BITS
5040 CP MODE_UPDATE ;DO WE HAVE TO PRE_READ THE NEXT BLOCK?
5041 JP NZ,WRITE3 ;LOOP TO TOP IF NOT
5042
5043 WRITE_AGAIN:
5044 LD A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
5045 LD HL,[BUF_START] ;GET BUFFER ADDRESS
5046 LD E,[IX+FCB_BLOCK] ;GET BLOCK NUMBER
5047 LD D,[IX+FCB_BLOCK+1]
5048 LD C,[IX+FCB_BLOCK+2]
5049 LD B,[IX+FCB_BLOCK+3]
5050 CALL READ_BLOCK ;PRE-READ THE NEXT FILE BLOCK
5051
5052 JR NZ,WRITE_ERR1 ;BRANCH IF ERROR
5053
5054
5055 ; ARRIVE HERE IF THE DATA BEING WRITTEN WILL NOT FILL THE BUFFER.
5056 ; EVERYTHING IS VERY EASY -- JUST COPY IT FF -- USER'S ADDRESS

```


LOCATION	OBJECT	CODE	LINE	SOURCE	LINE
			5057	: INTO THE BUFFER AND GO AWAY.	
EE93			5058		
EE93	ED4BFE04		5059	ROOM_OK LD	BC.[BYTES_TO_GO] ;GET THE BYTE COUNT
EE97	DD6E15		5061	LD	L.[IX+FCB_LAST_COUNT]
EE9A	DD6616		5062	LD	H.[IX+FCB_LAST_COUNT+1]
EE9D	09		5063	ADD	HL,BC
EE9E	DD7515		5064	LD	[IX+FCB_LAST_COUNT],L
EEA1	DD7416		5065	LD	[IX+FCB_LAST_COUNT+1],H
EEA4	DD9E21		5066	LD	E.[IX+FCB_POINTER] ;GET POINTER INTO BUFFER
EEA7	DD9622		5067	LD	D.[IX+FCB_POINTER+1]
EEAA	2AFE06		5068	LD	HL.[USER_BUF] ;GET POINTER TO USER'S DATA
EEAD	ED80		5069	LDIR	;MOVE THE DATA
			5070		
EEAF	DD7321		5071	LD	[IX+FCB_POINTER],E ;UPDATE THE POINTER
EEB2	DD7222		5072	LD	[IX+FCB_POINTER+1],D
EEB5	DDC818F6		5073	SET	MODE_DIRTY_BIT,[IX+FCB_MODE] ;SET THE DIRTY BIT
			5074		
EEB9			5075	ROOM1_OK:	
EEB9	AF		5076	XOR	A
EEBA	1803		5077	JR	PAST_WRITE_ERR
EEBC			5078	WRITE_ERR2:	
EEBC	3E09		5079	LD	A,BAD_FNUM_ERR
EEBE			5080	WRITE_ERR1:	
EEBE	B7		5081	OR	A
EEBF			5082	PAST_WRITE_ERR:	
EEBF	DDE1		5083	POP	IX ;RESTORE REGISTERS
EEC1	E1		5084	POP	HL
EEC2	D1		5085	POP	DE
EEC3	C1		5086	POP	BC
EEC4	C9		5087	RET	
			5088	*WRITE_ERR1	
			5089	*POP	IX ;RESTORE REGISTERS
			5090	*POP	HL
			5091	*POP	DE
			5092	*POP	BC
			5093	*OR	A
			5094	*RET	
			5095	*WRITE_ERR2	
			5096	*POP	IX ;RESTORE REGISTERS
			5097	*POP	HL
			5098	*POP	DE
			5099	*POP	BC
			5100	*LD	A,BAD_FNUM_ERR ;SHOW AN ERROR
			5101	*OR	A
			5102	*RET	
			5103	:	
			5104	:	
			5105	:	
			5106	:	
			5107	:	
			5108	:	

LOCATION OBJECT CODE LINE SOURCE LINE

```

5110 :GLB      __SET_DATE,__GET_DATE
5111 :EXT      EOS_YEAR,EOS_MONTH,EOS_DAY
5112
5113
5114 -----
5115 :
5116 :__SET_DATE -- Set the current date.
5117
5118 ENTRY PARAMETERS: B = day (1..31)
5119                  C = month (1..12)
5120                  D = year (83..99)
5121
5122 EXIT PARAMETERS: none
5123
5124 -----
5125

```

```

EEC5 5126 __SET_DATE      AF      :SAVE THE CURRENT A
EEC5 5127 PUSH          AF
EEC6 5128 LD            A,B      :STORE THE DAY
EEC7 5129 LD            [EOS_DAY],A
EECA 5130 LD            A,C      :STORE THE MONTH
EECB 5131 LD            [EOS_MONTH],A
EECE 5132 LD            A,D      :STORE THE YEAR
EECF 5133 LD            [EOS_YEAR],A
EED2 5134 LD
EED3 5135 POP          AF      :RESTORE A
EED3 5136 RET
EED3 5137

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

5139 :-----
5140 :
5141 : __GET_DATE -- Reads the current date.
5142 :
5143 : ENTRY PARAMETERS: none
5144 :
5145 : EXIT PARAMETERS: no errors -- Z = 1; A = 0
5146 :                   B = day
5147 :                   C = month
5148 :                   D = year
5149 :
5150 : error (date never set) -- Z = 0; A = error code
5151 :                   B = 0
5152 :                   C = 0
5153 :                   D = 0
5154 :
5155 :-----
5156 :
5157 : __GET_DATE
5158 : LD A,[EOS_DAY]
5159 : LD B,A
5160 : LD A,[EOS_MONTH]
5161 : LD C,A
5162 : LD A,[EOS_YEAR]
5163 : LD D,A
5164 :
5165 : OR B
5166 : OR C
5167 : JR Z,GET_ERROR
5168 : XOR A
5169 : RET
5170 :
5171 : GET_ERROR
5172 : LD A,NO_DATE_ERR
5173 : OR A
5174 : RET
5175 :
5176 :-----
5177 :-----
5178 :-----
5179 :-----
5180 :

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

5182 : Rev. 1 07oct515p RPD made __DELETE__FILE check for protected file
5183 :
5184 :
5185 :
5186 : FMGR INIT,SCAN FOR FILE,STRCMP,MODE_CHECK
5187 : READ_BLOCK,WRITE_BLOCK,BASECMP,__WRITE_BLOCK
5188 : MODE_CHECK,__READ_BLOCK,__SCAN_FOR_FILE
5189 : __RD_1_BLOCK,__WR_1_BLOCK,__REQUEST_STATUS
5190 : QUERY_FILE,__SET_FILE
5191 : QUERY_BUFFER,FILENAME_CMPS
5192 : FCB HEAD,ADDR,FCB DATA_ADDR
5193 : USER_NAME,BUF_START
5194 : RETRY_COUNT
5195 : FILE_COUNT,MOD_FILE_COUNT
5196 :
5197 :
5198 :
5199 :
5200 :
5201 :
5202 :
5203 :
5204 :
5205 :
5206 :
5207 :

```

__FMGR_INIT: sets up everything to start with.
ENTRY PARAMETERS: DE contains address to place FCB buffers.
and HL contains address to place FCB headers.

EXIT PARAMETERS: none.

```

EEEE EEEA
EEEE ED53FDFF
EEEE 22FD0D

EEF1 C5
EEF2 D5
EEF3 D0E5

EEF5 0603
EEF7 110023
EEFA D02AFD0D

EEFE EEEF DD361800
EFO2 DD19
EFO4 10F8

EFO6 DDE1
EFO8 D1
EFO9 C1
EFOA C9

__FMGR_INIT
LD [FCB_DATA_ADDR],DE ;SAVE BUFFER ADDRESS
LD [FCB_HEAD_ADDR],HL ;SAVE HEADER ADDRESS

BC ;SAVE REGISTERS
DE
IX

B,NUM_FCBS ;SET TO # FCBS
DE,FCB_LENGTH ;SET FOR ADDITION
IX,[FCB_HEAD_ADDR] ;GET ADDR OF OTH FCB

INIT_LOOP
LD [IX+FCB_MODE],MODE UNUSED ;SHOW THE FCB IS EMPTY
IX,DE ;ADVANCE POINTER TO NEXT FCB
DUNZ ;LOOP FOR ALL FCBS

POP ;RESTORE REGISTERS
POP DE
POP BC
RET

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

5231 :-----
5232 :
5233 : SCAN_FOR_FILE -- Look in the directory for a file name.
5234 : Reads each block of the directory into a buffer
5235 : (in the system's FCB) and scans through for a
5236 : match with the caller's string.
5237 :
5238 : ENTRY PARAMETERS: address of name string in HL; device # in A
5239 : ASSUMES FILENAME_CMP SET UP FOR FILE COMPARIS
5240 : O - USE STRCMP, NOT O - USE BASECMP FOR FILE NAME COMPARISONS
5241 :
5242 : EXIT PARAMETERS: if match found -- Z = 1; A = 0; file's start
5243 : block in BCDE; directory block left in buffer,
5244 : address of entry left in FCB_POINTER.
5245 :
5246 : if no match found -- Z = 0; A = error code;
5247 : BCDE = junk; other stuff = junk.
5248 :-----
5249 :
5250 :
5251 : SCAN_FOR_FILE:
5252 : SCAN_FOR_FILE
5253 : PUSH HL ;SAVE REGISTERS
5254 : PUSH IX
5255 : PUSH IY
5256 :
5257 : LD [USER_NAME],HL ;SAVE USER'S STRING ADDRESS
5258 :
5259 : LD IX,[FCB_HEAD_ADDR] ;GET ADDRESS OF SYSTEM'S FCB
5260 :
5261 : First, set up as much of the FCB as we can.
5262 : LD [IX+FCB_DEVICE],A ;SET THE DEVICE ADDRESS PASSED TO US
5263 :
5264 : LD [IX+FCB_BLOCK],1 ;SET BLOCK TO 1
5265 : LD [IX+FCB_BLOCK+1],0
5266 : LD [IX+FCB_BLOCK+2],0
5267 : LD [IX+FCB_BLOCK+3],0
5268 :
5269 : LD DE,[FCB_DATA_ADDR] ;SET BUFFER ADDRESS
5270 : LD [IX+FCB_POINTER],E
5271 : LD [IX+FCB_POINTER+1],D
5272 :
5273 : RD_TIMEOUT:
5274 : LD HL,FILE_COUNT
5275 : LD [HL],0
5276 :
5277 : LD A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
5278 : LD HL,[FCB_DATA_ADDR] ;GET BUFFER ADDRESS
5279 : LD E,[IX+FCB_BLOCK] ;GET BLOCK NUMBER
5280 : LD D,[IX+FCB_BLOCK+1]
5281 : LD C,[IX+FCB_BLOCK+2]
5282 : LD B,[IX+FCB_BLOCK+3]
5283 : CALL READ_BLOCK ;READ FIRST BLOCK OF DIRECTORY
5284 : JP NZ,SCAN_ERROR ;BRANCH IF ERROR
5285 :
5286 : GET_BUFF_ADDR:
5287 :

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

EF51 E5      5288      PUSH HL      ; GET BUFFER ADDRESS INTO IY T00
EF52 FDE1    5289      POP  IY
5290
EF54 F07E0C  5291      LD  A,[IY+VOL_DIRSIZE] ;GET THE DIRECTORY SIZE
EF57 E67F    5292      O7FH      ;ZERO PERM BIT
5293      ;
5294      ;
EF59 D0771D  5295      LD  A,[IX+FCB_LAST_BLOCK],A ;SET LAST BLOCK IN FCB
EF5C D0361E00 5296      LD  A,[IX+FCB_LAST_BLOCK+1],0 ;(ALL MSBYTES WILL BE 0)
EF60 D0361F00 5297      LD  A,[IX+FCB_LAST_BLOCK+2],0
EF64 D0362000 5298      LD  A,[IX+FCB_LAST_BLOCK+3],0
5299
EF66 C0F035  5300      CALL CHECK_IF_DIRECTORY ; CHECK IF DIRECTORY EXIST
EF68 C2F026  5301      JP   NZ,SCAN_ERROR
5302
EF6E D07E21  5303      LD  A,[IX+FCB_POINTER] ;ADVANCE POINTER TO NEXT ENTRY
EF71 C61A    5304      ADD  A,DIR_ENT_LENGTH
5305
EF73 D07721  5306      LD  A,[IX+FCB_POINTER],A
5307
EF76 D07E22  5307      LD  A,[IX+FCB_POINTER+1]
5308
EF78 D07722  5308      LD  A,0
5309
EF7E 21FDD4  5310      LD  HL,FILE_COUNT
EF81 34      5311      INC  [HL]
5312
EF82 0626    5313      LD  B,ENT_PER_BLOCK-1 ;SET DIR ENTRY COUNT FOR FIRST BLOCK
EF84 1848    5314      JR   SCAN2 ;JUMP INTO THE LOOP
5315
5316      SCAN1
EF86      5316      INC  [IX+FCB_BLOCK] ;INC THE BLOCK NUMBER
EF88 D03419  5317      ;WE CAN ASSUME THERE'S NO CARRY OUT
5318
5319
5320      ; SEE IF THERE IS MORE DIRECTORY: I.E. IS FCB_BLOCK > FCB_LAST_BLOCK?
5321      ; IF TRUE, GO TO NO_ENT.
5322
EF89 D07E20  5323      LD  A,[IX+FCB_LAST_BLOCK+3]
EF8C D0BE1C  5324      CP   [IX+FCB_BLOCK+3]
EF8F DAF02C  5325      JP   C,NO_ENT ;BRANCH IF NO MORE DIRECTORY
EF92 D07E1F  5326      LD  A,[IX+FCB_LAST_BLOCK+2]
EF95 D0BE1B  5327      CP   [IX+FCB_BLOCK+2]
EF98 DAF02C  5328      JP   C,NO_ENT ;BRANCH IF NO MORE DIRECTORY
EF9E D0BE1A  5329      LD  A,[IX+FCB_LAST_BLOCK+1]
FA01 DAF02C  5330      CP   [IX+FCB_BLOCK+1]
FA04 D07E1D  5331      JP   C,NO_ENT ;BRANCH IF NO MORE DIRECTORY
FA07 D0BE19  5332      LD  A,[IX+FCB_LAST_BLOCK]
FA0A DAF02C  5333      CP   [IX+FCB_BLOCK]
FA0D DAF02C  5334      JP   C,NO_ENT ;BRANCH IF NO MORE DIRECTORY
5335
5336      RD_TIMEOUT1:
EFAD D07E17  5337      LD  A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
EFB0 2AFDFF  5338      HL,[FCB_DATA_ADDR] ;GET BUFFER ADDRESS
EFB3 D05E19  5339      E,[IX+FCB_BLOCK] ;GET BLOCK NUMBER
EFB6 D0561A  5340      LD  D,[IX+FCB_BLOCK+1]
EFB9 D04E1B  5341      LD  C,[IX+FCB_BLOCK+2]
EFBC D0461C  5342      LD  B,[IX+FCB_BLOCK+3]
EFBF C8E1      5343      CALL READ_BLOCK ;READ THE NEXT BLOCK
5344      ;BRANCH IF

```

LOCATION OBJECT CNOE LINE SOURCE LINE

```

5345
5346 GET_BUFF_ADDR1:
5347
5348
5349 LD DE.[FCB DATA ADDR] ;GET BUFFER ADDRESS
5350 LD [IX+FCB_POINTER].E ;PUT IT INTO THE FCB
5351 LD [IX+FCB_POINTER+1].D
5352
5353 LD B.ENT_PER_BLOCK ;SEI ENTRY COUNT
5354
5355 SCAN2
5356 LD HL.FILE_COUNT
5357 INC [HL]
5358
5359 LD E.[IX+FCB_POINTER] ;GET POINTER INTO DE
5360 LD D.[IX+FCB_POINTER+1]
5361
5362 LD L.DIR_ATTR ; OFFSET (FOR ATTR) INTO DIR ENTRY
5363 LD H.O ; NOT USED
5364 ADD ; SET UP HL TO POINT TO IT
5365 BIT ATTR_HOLE_BIT.[HL] ; IS IT THE HOLE?
5366 JR NZ.NO_ENT ; YUP, DON'T BOTHER SEARCHING ANY MORE
5367 BIT ATTR_DEL_BIT.[HL] ; IS THIS FILE DELETED?
5368 JR NZ.FILE_DELETED ; YUP, DON'T EVEN BOTHER CHECK FILE NAMES
5369
5370 LD HL.[USER_NAME]
5371
5372 LD A.[FILENAME_CMPS] ;
5373 OR A ; IS IT ZERO?
5374 JR Z.FULL_FILENAME ; YUP, COMPARE THE FULL FILE NAME
5375 CALL BASECMP ; NOPE, COMPARE JUST THE BASE
5376 JR ERROR_CHECK
5377 FULL_FILENAME:
5378 CALL STRCMP ; COMPARE THE FULL FILE NAME (BASE PLUS EXTENSION)
5379 ERROR_CHECK: ; BRANCH IF YES
5380 JR Z.SCAN3
5381
5382 FILE_DELETED:
5383 LD A.[IX+FCB_POINTER] ;ADVANCE POINTER TO NEXT ENTRY
5384 ADD A.DIR_ENT_LENGTH
5385 LD [IX+FCB_POINTER].A
5386 LD A.[IX+FCB_POINTER+1]
5387 ADC A.O
5388 LD [IX+FCB_POINTER+1].A
5389
5390 DJNZ SCAN2 ;LOOP IF MORE ENTRIES TO CHECK IN THIS BLOCK
5391
5392 JP SCAN1 ;ELSE GET NEXT BLOCK
5393
5394 SCAN3
5395 PUSH DE
5396 POP IV ; GET DIR ENTRY PTR INTO IV
5397
5398 LD E.[IX+DIR_START_BLOCK+0]
5399 LD D.[IX+DIR_START_BLOCK+1]
5400 LD C.[IX+DIR_START_BLOCK+2]
5401 LD B.[IX+DIR_START_BLOCK+3]

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

F01F FDE1      5402      POP      IV      :RESTORE REGISTERS
F021 DDE1      5403      POP      IX
F023 E1        5404      POP      HL
F024 AF        5405      POP      HL
F025 C9        5406      XOR      A      :SHOW NO ERROR
                5407      RET
                5408
F026           5409      SCAN_ERROR
F026 FDE1      5410      POP      IV      :RESTORE REGISTERS
F028 DDE1      5411      POP      IX
F02A E1        5412      POP      HL
F02B C9        5413      POP      HL
                5414      RET
                5415      NO_ENT
F02C           5416      POP      IV      :RESTORE REGISTERS
F02C FDE1      5417      POP      IX
F02E DDE1      5418      POP      HL
F030 E1        5419      LD      A,NO_FILE_ERR
F031 3E08      5420      OR      A
F033 B7        5421      OR      A
F034 C9        5422      RET
                5423

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
5425 .....
5426 .....
5427 .....
5428 * .....
5429 * CHECK_IF_DIRECTORY_VERIFIES_THE_EXISTENCE_OF_A_DIRECTORY
5430 * CHECK_IF_DIRECTORY_VERIFIES_THE_EXISTENCE_OF_A_DIRECTORY
5431 * CHECK_IF_DIRECTORY_VERIFIES_THE_EXISTENCE_OF_A_DIRECTORY
5432 * INPUT: IY - POINTS TO THE VOLUME ENTRY (FIRST ENTRY INTO THE DIRECTORY)
5433 * .....
5434 * OUTPUT: CONDITION FLAGS
5435 * Z - DIRECTORY EXISTS
5436 * NZ - NO DIRECTORY
5437 * A - ERROR CODE
5438 * .....
5439 * .....
5440 * ALL REGISTERS PRESERVED EXCEPT AF
5441 * .....
5442 .....
5443 .....
5444 .....
5445 :GLB CHECK_IF_DIRECTORY
5446 :EXT DIRECTORY_CHECK
5447 .....
5448 CHECK_IF_DIRECTORY:
5449 PUSH IY
5450 PUSH BC
5451 PUSH HL
5452 .....
5453 LD HL, DIRECTORY_CHECK
5454 .....
5455 .....
5456 LD B, 4
5457 NEXT_CHECK:
5458 LD A, [HL]
5459 CP [IY+VOL_DIR_CHECK]
5460 JR NZ, NOT_DIR
5461 .....
5462 INC HL
5463 INC IY
5464 DJNZ NEXT_CHECK
5465 .....
5466 XOR A
5467 .....
5468 GET_OUT_HERE:
5469 POP HL
5470 POP BC
5471 POP IY
5472 .....
5473 RET
5474 .....
5475 NOT_DIR:
5476 LD A, NO_DIR_ERR
5477 JR GET_OUT_HERE
5478 .....

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

5480 -----
5481 : STRCMP -- compare two strings (file names).
5482 :
5483 : ENTRY PARAMETERS: address of name string in HL; address of
5484 : name within dir entry in DE. NOTE: string
5485 : pointed to by HL must be terminated with ETX.
5486 : Maximum of NAME_LENGTH characters are compared.
5487 :
5488 : EXIT PARAMETERS: Z = 1; A = 0 if match or
5489 : Z = 0; A = error code if no match.
5490 :
5491 :-----
5492 :
5493 BASECMP: EQU $+1      ; EXECUTES SCF (37H) - COMPARE ONLY THE BASE
5494 : OF THE FILE NAME
5495 : "AND" INSTRUCTION CLEARS CARRY - COMPARE BASE &
5496 : EXTENSION
5497 : PRESERVE REG'S
5498 :
5499 STRCMP:      AND      037H      ; SAVE CARRY FLAGGING EXECUTED ABOVE
5500      PUSH      HL      ; INIT MAX COUNT
5501      PUSH      DE      ; GET CHAR FROM STRING
5502      PUSH      BC      ; HAVE WE REACHED END OF STRING?
5503      PUSH      AF      ; BRANCH IF YES
5504      LD      B,NAME_LENGTH      ; GET CHAR FROM DIR ENTRY
5505      LD      A,[HL]      ; COMPARE TO STRING
5506      CP      Z,EOS      ; BRANCH IF DIFFERENT
5507      JR      C,      ; ELSE INC. POINTERS
5508      LD      A,[DE]      ; LOOP IF MORE CHARS TO LOOK AT
5509      CP      HL      ; POP AF INTO BC
5510      JR      NC,NO_MATCH      ; ALL 12 MATCH IF WE GET HERE!
5511      INC      DE      ; COMPARE BASE OR BASE&EXTENSION
5512      INC      NEXT_CHAR      ; NEEDED TO COMPARE THE WHOLE FILE NAME
5513      DJNZ      BC      ; ONLY COMPARE THE BASE
5514      POP      MATCH      ; POINT TO NEXT CHAR OF NAME STRING
5515      JR      C,      ; AND DIR ENTRY NAME
5516      MATCHLESS      ; IS IT ETX?
5517      POP      AF      ; NOPE. DOES NOT MATCH
5518      JR      NC,NO_MATCH      ; AF INTO BC TO PRESERVE AF
5519      POP      BC      ; LOOK AT CHAR IN DIR ENTRY
5520      INC      DE      ; DLS(B/29/83) CHANGED SPACE TO ETX.
5521      INC      A,[HL]      ; END OF FILE NAME?
5522      LD      ETX      ; BRANCH IF NOT
5523      CP      NZ,NO_MATCH      ; RESTORE REG'S
5524      JR      C,      ;
5525      JR      AROUND_POP      ;
5526      EOS      ;
5527      POP      BC      ;
5528      AROUND_POP:      ;
5529      LD      A,[DE]      ;
5530      CP      ETX      ;
5531      JR      C,      ;
5532      JR      NZ,NO_MATCH      ;
5533      POP      BC      ;
5534      MATCH      ;
5535      POP      POP      ;
5536      ;

```

<F054>

F053
F053 E637
F055 EB
F056 DB
F057 CB
F058 FB

F059 O60C
F068
F068 7E
F06C FE03
F05E 2B17
F060 1A
F061 BE
F062 2007
F064 23
F065 13
F066 10F3
F068 C1
F069 1812
F06B
F06B F1
F06C 3014

F06E 23
F06F 13
F070 7E
F071 FE03
F073 200D
F075 1801

F077
F077 C1
F078
F078 1A

F079 FE03
F07B 2005
F07D
F07D C1
F07E

LOCATION OBJECT CODE LINE SOURCE LINE

F07F E1	5537	POP	HL	
F080 AF	5538	XOR	A	
F081 C9	5539	RET		: SHOW THE MATCH
F082	5540	NO_MATCH		
F082 C1	5541	POP	BC	: RESTORE REG'S
F083 D1	5542	POP	DE	
F084 E1	5543	POP	HL	
F085 3E08	5544	LD	A, MATCH_ERR	
F087 B7	5545	OR	A	
F088 C9	5546	RET		: RETURN WITH NZ CONDITION

```

LOCATION OBJECT CODE LINE SOURCE LINE
5548 .....
5549 .....
5550 *
5551 *
5552 * THIS ROUTINE CHECKS TO SEE IF THE FILENAME (POINTED TO BY HL)
5553 * IS ALREADY OPEN AND EXIST IN THE FCB
5554 *
5555 * INPUT: HL POINTER TO FILENAME (ETX TERMINATED)
5556 *
5557 * OUTPUT: CONDITION FLAGS
5558 * NZ NOT IN FCB (NOT OPENED)
5559 * A = ERROR CODE
5560 * HL = ORIGINAL POINTER TO FILENAME
5561 * Z IN FCB
5562 * A = FILE NUMBER
5563 * B = MODE
5564 * HL = POINTER TO START OF FCB *
5565 *
5566 .....
5567 .....
5568 : GET OUT U_GET_OUT
5569 : NEXT_FCB CHANGED TO U_NEXT_FCB TO PREVENT CONFLICT WITH
5570 : IDENTICAL LABEL ELSEWHERE
5571
5572 :GLB CHECK_FCB
5573 :EXT FILE_NUMBR
5574
5575 CHECK_FCB:
5576 PUSH IX
5577 PUSH DE
5578 :
5579 PUSH BC
5579 PUSH HL
5580
5581 LD HL,[FCB_HEAD_ADDR]
5582 XOR A
5583 U_NEXT_FCB:
5584 LD [FILE_NUMBR].A
5585
5586 LD DE,FCB_LENGTH
5587 ADD HL,DE
5588
5589 PUSH HL
5590 POP IX
5591
5592 LD A,[IX+FCB_MODE]
5593 AND MODE_MODE
5594 CP O
5595
5596 JR Z,END_FCB
5597
5598 EX DE,HL
5599 POP HL
5600 PUSH HL
5601
5602 PUSH DE
5603
5604

```

: ZERO OUT A
: CONTAINS THE FILE NUMBER
: (THE POSITION ON THE TAPE)
: NUMBER OF BYTES IN EACH FCB
: POINT TO NEXT FCB ENTRY
: IX POINTS TO START OF FCB
: GET MODE
: IT IS UNUSED?
: YUP, LOOK AT NEXT FCB
: DE = POINTER TO FCB NAME STRING
: HL = POINTER TO FILENAME TO SEARCH FOR

LOCATION OBJECT CODE LINE SOURCE LINE

```

FOA9 01000C      5605 LD      BC,NAME_LENGTH      ; NO. OF CHARACTERS IN NAME
5606              ; HL = FILE NAME STRING
FOAC              5607 NEXT_CHAR1:
FOAC 1A          5608 LD      A,[DE]
FOAD FE03       5609 CP      ETX
5610              ; DE = FCB FILE NAME
FOAF 2813       5611 JR      Z,ITS_ETX1
5612              ; IS IT ETX?
FOB1 EDA1       5612 CPI
5613              ; YUP
FOB3 13         5614 INC
5615              ; DO CHARS MATCH ([HL] = A)?
FOB4 28F6       5615 JR      DE
5616              ; YUP, CHECK NEXT CHAR OF FILE NAME
5617              ; Z, NEXT_CHAR1
FOB6 E1         5616 POP
5617              ; HL
FOB7              5618 END_FCB:
FOB7 3AFD07     5619 LD      A,[FILE_NUMBER]
FOBA FE02       5620 CP      NUM_FCBS-1
5621              ; ARE WE AT 2ND FILE(FCB)?
FOBC 2003       5622 JR      NZ,U_NEXT_FCB
5623              ; NOPE, NOT YET.
5624              ; HL = ORIGINAL POINTER TO FILE NAME
FOBE E1         5625 NO_MATCH1:
FOBE E1         5626 POP
5627              ; HL
FOBF 3E05       5627 LD      A,NO_FILE_ERR
FOC1 B7         5628 OR      A
5629              ; NO MATCH FOUND
FOC2 1811       5629 JR      GET_OUT_ERR
5630              ; HL
FOC4              5631 ITS_ETX1:
FOC4 7E         5632 LD
FOC5 FE03       5633 CP
FOC7 E1         5634 CP
FOC8 20F4       5635 POP
5636              ; GET ASCII CHAR
5637              ; IS IT ETX?
5638              ; *pointer to FCB
5639              ; NOPE
5640              ; copy pointer to FCB so can index to mode
FOCA DD7E18     5638 PUSH
FOCD E607       5639 POP
FOCF 47         5640 LD
5641              ; HL
FOCF 47         5641 LD      A,[IX+FCB_MODE]
FOCF 47         5642 AND     MODE_MODE
5643              ; GET MODE THIS FILE WAS ORIGINALLY
5644              ; OPENED FOR AND MASK OFF FLAG BITS
FOCF 47         5643 LD      B,A
5644              ; PASS IT BACK IN B
FOOF AF         5644 XOR
FOO1 3AFD07     5645 LD      A
5646              ; INDICATE Z CONDITION
5647              ; GET FILE NUMBER
FOO4              5647 U_GET_OUT:
FOO4 D1         5648 POP
FOO5 D1         5649 POP
FOO6 DDE1       5650 POP
5651              ; *clean off pointer to name
FOO6 DDE1       5651 GET_OUT_ERR:
FOO6 DDE1       5652 POP
5653              ; DE
5654              ; IX
FOOB C9         5653 RET
5655              ; DE
5656              ; BC
5656              ; POP
5656              ; GET_OUT_ERR:
5656              ; POP
5656              ; DE
5656              ; IX
5656              ; RET

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

5658 -----
5659
5660
5661 : MODE_CHECK: sees if an attribute byte matches a permission request.
5662 -----
5663
5664 : ENTRY PARAMETERS: IX- FCB POINTER
5665 : HL- DIR ENTRY
5666 : EXIT PARAMETERS: Z=1=OK
5667 : Z=0=NO WAY
5668 : A=TRASH
5669 : IX AND HL PRESERVED
5670 -----
5671
5672 : OK CHANGED TO U_OK TO PREVENT CONFLICT WITH IDENTICAL
5673 : LABEL ELSEWHERE
5674
5675 : MODE_CHECK:
5676 : MODE_CHECK:
5677
5678 : PUSH IV :SAVE REG IV
5679 : PUSH HL :SAVE DIR ENTRY
5680 : LD A, [IX+FCB_MODE] :GET THE MODE THE USER PASSED IN
5681 *
5682 * RANGE CHECK THIS REQUEST
5683 *
5684 : AND MODE_MODE : AND OFF TEST BITS
5685 : CP O :MAKE SURE HE'S IN RANGE
5686 : JR Z,RANGE_NONO
5687 : CP MODE_MAX+1
5688 : JR NC,RANGE_NONO
5689 *
5690 : LD HL,ATTRIB_TBL : INDEX INTO THE MASK TABLE
5691 : ADD A,L
5692 : LD L,A
5693 : JR NC,ADJUST
5694 : INC H
5695 : ADJUST
5696 : POP IV :GET THE DIR ENTRY POINTER INTO IV
5697 : PUSH IV
5698 : LD A,[IY+DIR_ATTR] :GRAB HIS ATTRIBUTE BYTE
5699 : AND [HL] :COMPARE IT TO THE USER'S REQUEST
5700 : JR NZ,PROTECT_FAULT
5701 : U_OK
5702 : POP HL
5703 : POP IV
5704 : XOR A
5705 : RET
5706 : PROTECT_FAULT
5707 : LD A,PRT_ERR
5708 : JR PROTECT
5709 : RANGE_NONO
5710 : LD A,RANGE_ERR
5711 : PROTECT
5712 : POP HL
5713 : POP IV
5714 : OR A

```

LOCATION OBJECT CODE LINE SOURCE LINE

F10A C9	5715	RET	
<F10A>	5716	ATTRIB_TBL EQU	\$-1
F10B 20	5717	DEFB	ATTR_READ_PROT
F10C 40	5718	DEFB	ATTR_WRITE_PROT
F10D 80	5719	DEFB	ATTR_PERMANENT
F10E 02	5720	DEFB	ATTR_EXECUTE
	5721		
	5722		

LOCATION OBJECT CODE LINE SOURCE LINE

```

5724 :.....
5725 :..... EOS_RENAME_FILE .....
5726 :.....
5727 :
5728 : ON ENTRY: A = DEVICE NUMBER (8=TAPE1)
5729 : DE = POINTER TO OLD FILENAME
5730 : HL = POINTER TO NEW FILENAME
5731 :
5732 : ON EXIT: Z = 1, FILE RENAMED, A = 0
5733 : Z = 0, FILE NOT RENAMED, A = ERROR CODE
5734 :
5735 :
5736 :
5737 :
5738 :
5739 :
5740 :
5741 :
5742 :
5743 :
5744 :
5745 :
5746 :
5747 :
5748 :
5749 :
5750 :
5751 :
5752 :
5753 :
5754 :
5755 :
5756 :
5757 :
5758 :
5759 :
5760 :
5761 :
5762 :
5763 :
5764 :
5765 :
5766 :
5767 :
5768 :
5769 :
5770 :
5771 :
5772 :
5773 :
5774 :
5775 :
5776 :
5777 :
5778 :
5779 :
5780 :
5781 :
5782 :
5783 :
5784 :
5785 :
5786 :
5787 :
5788 :
5789 :
5790 :
5791 :
5792 :
5793 :
5794 :
5795 :
5796 :
5797 :
5798 :
5799 :
5800 :
5801 :
5802 :
5803 :
5804 :
5805 :
5806 :
5807 :
5808 :
5809 :
5810 :
5811 :
5812 :
5813 :
5814 :
5815 :
5816 :
5817 :
5818 :
5819 :
5820 :
5821 :
5822 :
5823 :
5824 :
5825 :
5826 :
5827 :
5828 :
5829 :
5830 :
5831 :
5832 :
5833 :
5834 :
5835 :
5836 :
5837 :
5838 :
5839 :
5840 :
5841 :
5842 :
5843 :
5844 :
5845 :
5846 :
5847 :
5848 :
5849 :
5850 :
5851 :
5852 :
5853 :
5854 :
5855 :
5856 :
5857 :
5858 :
5859 :
5860 :
5861 :
5862 :
5863 :
5864 :
5865 :
5866 :
5867 :
5868 :
5869 :
5870 :
5871 :
5872 :
5873 :
5874 :
5875 :
5876 :
5877 :
5878 :
5879 :
5880 :
5881 :
5882 :
5883 :
5884 :
5885 :
5886 :
5887 :
5888 :
5889 :
5890 :
5891 :
5892 :
5893 :
5894 :
5895 :
5896 :
5897 :
5898 :
5899 :
5900 :
5901 :
5902 :
5903 :
5904 :
5905 :
5906 :
5907 :
5908 :
5909 :
5910 :
5911 :
5912 :
5913 :
5914 :
5915 :
5916 :
5917 :
5918 :
5919 :
5920 :
5921 :
5922 :
5923 :
5924 :
5925 :
5926 :
5927 :
5928 :
5929 :
5930 :
5931 :
5932 :
5933 :
5934 :
5935 :
5936 :
5937 :
5938 :
5939 :
5940 :
5941 :
5942 :
5943 :
5944 :
5945 :
5946 :
5947 :
5948 :
5949 :
5950 :
5951 :
5952 :
5953 :
5954 :
5955 :
5956 :
5957 :
5958 :
5959 :
5960 :
5961 :
5962 :
5963 :
5964 :
5965 :
5966 :
5967 :
5968 :
5969 :
5970 :
5971 :
5972 :
5973 :
5974 :
5975 :
5976 :
5977 :
5978 :
5979 :
5980 :
5981 :
5982 :
5983 :
5984 :
5985 :
5986 :
5987 :
5988 :
5989 :
5990 :
5991 :
5992 :
5993 :
5994 :
5995 :
5996 :
5997 :
5998 :
5999 :
6000 :

```


LOCATION	OBJECT CODE	LINE	SOURCE	LINE
F14A	D1	5781	POP	DE
F14B	C1	5782	POP	BC
F14C	C1	5783	POP	BC
F14D	C9	5784	RET	
		5785		

: REALLY AF BUT NEED TO PRESERVE ERR MSG.

LOCATION OBJECT CODE LINE SOURCE LINE

```

5787
5788
5789 :.....EOS_DELETE_FILE.....
5790 :.....EOS_DELETE_FILE.....
5791
5792 : ON ENTRY: A = DEVICE NUMBER (8=TAPE1)
5793 : HL = POINTER TO FILENAME
5794
5795 : ON EXIT: Z = 1, FILE DELETED, A = 0
5796 : Z = 0, FILE NOT DELETED, A = ERROR CODE
5797
5798 :GLB      __DELETE_FILE
5799
5800
5801 __DELETE_FILE:
5802 PUSH
5803 HL      : HL = POINTER TO FILENAME
5804 PUSH  AF      : A = DEVICE ID
5805 EX     DE,HL  : DE = POINTER TO FILENAME
5806 LD     HL,QUERY_BUFFER      : HL = POINTER TO BUFFER
5807 CALL  __QUERY_FILE
5808 JR
5809 NZ,OTHER_ERR
5810 LD     A,[QUERY_BUFFER+FCB_ATTR] : GET ATTRIBUTE BYTE
5811 BIT   7,A      :see if permanent protected
5812 OR
5813 NZ,NOT_DELETED      : SET "DELETED" BIT
5814 LD     ATTR_DELETED
5815 POP   AF
5816 DE
5817 DE
5818 AF      : GET FILENAME POINTER
5819
5820 CALL  __SET_FILE
5821 JR
5822 NZ,OTHER_ERR
5823 A
5824 HL
5825 A
5826 HL
5827 HL
5828 DE
5829
5830 A,DELETE_ERR      : ERROR RETURN
5831 A
5832
5833 HL
5834 HL
5835 DE
5836
5837 OTHER_ERR:
5838 POP
5839 POP
5840 POP
5841 POP
5842 POP
5843 RET
5844
5845 NOT_DELETED:
5846 LD
5847 LD
5848 OR
5849
5850 POP
5851 POP
5852 POP
5853 POP
5854 RET
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

5836 *****
5837 *****
5838 *
5839 *
5840 *
5841 *
5842 *
5843 *
5844 *
5845 *
5846 *
5847 *
5848 *
5849 *
5850 *
5851 *
5852 *
5853 *
5854 *
5855 *
5856 *
5857 *
5858 *
5859 *
5860 *
5861 *
5862 *
5863 *****
5864 *****
5865 *****
5866 READ_BLOCK:
5867 READ_BLOCK:
5868 PUSH BC
5869 PUSH DE
5870 PUSH HL
5871 PUSH IX
5872 PUSH IY
5873 PUSH AF
5874
5875 LD A,ERROR_RETRY
5876 LD [RETRY_COUNT],A
5877 READ_LOOP:
5878 POP AF
5879 PUSH AF
5880
5881 CALL __RD_1_BLOCK
5882
5883 JR Z,NO_READ_ERRORS
5884
5885 CP TIMEOUT
5886
5887 JR Z,NO_READ_ERRORS
5888
5889 PUSH HL
5890 LD HL,RETRY_COUNT
5891 DEC [HL]
5892 POP HL

```

READ_BLOCK:
1) READS 1 BLOCK OFF OF SPECIFIED DEVICE
2) REQUESTS FOR THE STATUS
3) AND SENDS ANOTHER READ TO THE DEVICE

INPUT: SAME AS FOR __RD_1_BLOCK
A DEVICE ID
LOW NIBBLE - DEVICE ADDRESS
HI NIBBLE - SECONDARY DEVICE ID

HL DESTINATION IN RAM

REGISTER PAIR BC DE
SECTOR NUMBER ON DEVICE

OUTPUT: CONDITION FLAGS
Z: NO ERRORS
NZ: ERROR OCCURED
A - ERROR CODE

ALL REGISTERS ARE PRESERVED EXCEPT FOR AF

: RETRY COUNT FOR ERRORS
: GET DEVICE ID BACK IN A
: READ WENT OK
: WAS IT A TIMEOUT?
: YUP.
: DEC THE NO OF RETRY'S LEFT

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
F199	283F	5893	JR	Z_READ_ERROR
F19B	18EB	5894	JR	READ_LOOP
		5895		: NO RETRY'S LEFT ; TRY AGAIN
F19D		5896	LD	B_ERROR_RETRY
F19D	0602	5897		: NUMBER OF TIMES TO RETRY
		5898		
F19F		5899	POP	AF
F19F	F1	5900	PUSH	AF
F1A0	F5	5901		: RESTORE DEVICIE ID
		5902		
F1A1	CDF473	5903	CALL	REQUEST_STATUS
		5904		: GET STATUS
F1A4	2808	5905	JR	Z_READ_COMMAND_COMPLETED
		5906		: READ ALL DONE
F1A6	FE9B	5907	CP	TIMEOUT
F1A8	28F5	5908	JR	Z_RETRY_FOR_ERROR
		5909		: WAS IT A TIMEOUT? ; NOT A REAL ERROR, DON'T NEED TO DEC COUNTER
F1AA	10F3	5910	DJNZ	RETRY_FOR_ERROR
		5911		: IT'S A REAL ERROR, TRY AGAIN
F1AC	182C	5912	JR	READ_ERROR
		5913		: CAN'T GET STATUS , GET OUT
F1AE		5914	CALL	READ_COMMAND_COMPLETE
F1AE	F1	5915	POP	AF
F1AF	F5	5916	PUSH	AF
		5917		: RESTORE DEVICE ID
F1B0	CDF225	5918	CALL	GET_STATUS_FLAG
		5919		: GET THE STATUS FOR THE PROPER ; DEVICE IN THE A REGISTER
F1B3	FE00	5920	CP	O
		5921		: ANY ERRORS WITH DRIVE?
F1B5	2023	5923	JR	NZ_READ_ERROR
		5924		: YUP
F1B7	3E02	5925	LD	A_ERROR_RETRY
F1B9	32FDD6	5926	LD	[RETRY_COUNT].A
		5927		
F1BC	F1	5928	POP	AF
F1BD	FDE1	5929	POP	IY
F1BF	DDE1	5930	POP	IX
F1C1	E1	5931	POP	HL
F1C2	D1	5932	POP	DE
F1C3	C1	5933	POP	BC
F1C4	C5	5934	PUSH	BC
F1C5		5935	CALL	RD_1_BLOCK
F1C5	F5	5936	PUSH	Z_NO_ERRORS
		5937		: SEND ANOTHER READ COMMAND
F1C6	CDFA9E	5938	CALL	RD_1_BLOCK
F1C9	280C	5939	JR	Z_NO_ERRORS
F1CB	E5	5940	PUSH	HL
F1CC	21FDD6	5941	LD	HL,RETRY_COUNT
F1CF	35	5942	DEC	[HL]
F1D0	E1	5943	POP	HL
F1D1	2803	5944	JR	Z_2ND_READ_ERROR
F1D3	F1	5945	POP	AF
		5946		: THE NO OF RETRY'S ; NO RETRY'S LEFT
F1D4	10FF	5947	JR	Z_2ND_READ_ERROR
		5948		: GO TRY AGAIN
F1D5		5949		
		5950		

END_READ_ERRORS

LOCATION	OBJECT	CODE	LINE	SOURCE	LINE
F1D6	B7		5950		
F1D7	C1		5951	OR	A
F1D7	C1		5952	NO_ERRORS:	
F1D8	C1		5953	POP	BC
			5954	POP	BC
F1D9	C9		5955	RET	
			5956		
F1DA	FDE1		5957	READ_ERROR:	
F1DC	FDE1		5958	POP	IV
F1DE	DDE1		5959	POP	IY
F1EO	E1		5960	POP	IX
F1E1	D1		5961	POP	HL
F1E2	C1		5962	POP	DE
			5963	POP	BC
			5964		
F1E3	3E16		5965	LD	A.DEVICE_DEPD_ERR
			5966		: INDICATE ERROR, LET APPL. HANDLE IT
F1E5	C9		5967	RET	
			5968		
			5969		

: NZ CONDITION

: POP AF INTO IY TO PRESERVE FLAGS

LOCATION OBJECT CODE LINE SOURCE LINE

5971 *****
5972 *****
5973 *****
5974 *

WRITE_BLOCK:

- 1) WRITES A BLOCK OUT TO TAPE
- 2) AND CHECKS STATUS AFTER WRITE IS COMPLETED

INPUT: SAME AS FOR __WR_1_BLOCK

A DEVICE ID
 LOW NIBBLE - DEVICE ADDRESS
 HI NIBBLE - SECONDARY DEVICE ID

HL SOURCE ADDRESS OF BUFFER

REGISTER PAIR BC DE
 SECTOR NUMBER ON DEVICE

OUTPUT: CONDITION FLAGS

- Z: NO ERRORS
- NZ: ERRORS OCCURED
- A - ERROR CODE

ALL REGISTERS PRESERVED EXCEPT FOR AF

5997 *
5998 *
5999 *****
6000 *****
6001 *****
6002 *****

F1E6

F1E6 C5

F1E7 FDE5

F1E9 F5

F1EA 3E02

F1EC 32FDD6

F1EF F1

F1FO F5

F1F1 CDFAB2

F1F4 280E

F1F6 FE9B

F1F8 280A

F1FA E5

F1FB 21FDD6

F1FE 3'

F1FF E

WRITE_BLOCK:

WRITE_BLOCK:

PUSH BC

PUSH IV

PUSH AF

LD A,ERROR_RETRY

LD [RETRY_COUNT],A

TRY_WRITE_AGAIN:

POP AF

PUSH AF

CALL __WR_1_BLOCK

JR Z,NO_WRITE_ERRORS ; NO ERRORS

CP TIMEOUT

JR Z,NO_WRITE_ERRORS ; ERROR OCCURED IN TRANSMISSION

PUSH HL

LD HL,RETRY_COUNT

DEC [HL]

POP HL

LOCATION OBJECT CODE LINE SOURCE LINE

```

F200 281C      6028 JR      Z.WRITE_ERROR
6029
F202 18EB      6030 JR      TRY_WRITE_AGAIN
6031
F204          6032 NO_WRITE_ERRORS:
6033
F204 0602      6034 LD      B.ERROR_RETRY      ; MAX NO OF RETRIES
6035
F206          6036 RETRY_WRITE:
F206 F1        6037 POP     AF      ; RESTORE DEVICE ID
F207 F5        6038 PUSH    AF
6039
F208 CDF473    6040 CALL    __REQUEST_STATUS
6041
F208 2808      6042 JR      Z.WRITE_COMMAND_COMPLETE ; STATUS BACK
6043
F200 FE9B      6044 CP      TIMEOUT      ; CHECK FOR TIMEOUT
6045
F20F 28F5      6046 JR      Z.RETRY_WRITE      ; TIMEOUT, NOT A 'REAL' ERROR
6047
F211 10F3      6048 DJNZ   RETRY_WRITE      ; REAL ERROR, TRY AGAIN
6049
F213 1809      6050 JR      WRITE_ERROR      ; DONE RETRYING, IT'S AN ERROR
6051
F215          6052 WRITE_COMMAND_COMPLETE:
F215 F1        6053 POP     AF      ; RESTORE DEVICE ID
F216 F5        6054 PUSH    AF
6055
F217 CDF225    6056 CALL    GET_STATUS_FLAG    ; GET STATUS FOR SPECIFIED DEVICE
6057
F21A FE00      6058 CP      0      ; WRITE OK?
F21C 2802      6059 JR      Z.WRITE_OK      ; YUP
6060
F21E          6061 WRITE_ERROR:
F21E 3E16      6062 LD      A.DEVICE_DEPD_ERR
6063
F220          6064 WRITE_OK:
F220 C1        6065 POP     BC      ; REALLY AF, BUT PRESERVE CURRENT FLAGS
F221 FDE1      6066 POP     IX
F223 C1        6067 POP     BC
6068
F224 C9        6069
6070
6071
6072

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

6074
6075 GET_STATUS_FLAG:
6076
6077 SRL A
6078 SRL A : GET SECONDARY DEVICE ID (HI NIBBLE)
6079 SRL A
6080 SRL A
6081
6082 CP O : WAS IT UNIT 0?
6083
6084 LD A,[IY+D_STATUS_FLAGS] : BY THE WAY, GET THE STATUS
6085
6086 JR NZ,UNIT1 : NOPE, UNIT 1
6087
6088 AND OFH : MASK OUT STATUS OF UNIT 1
6089
6090 JR CHECK_STATUS
6091
6092 UNIT1:
6093 SRL A
6094 SRL A
6095 SRL A
6096 SRL A
6097
6098 CHECK_STATUS:
6099
6100 RET
6101

```

: RETURN WITH THE STATUS NIBBLE IN A

LOCATION OBJECT CODE LINE SOURCE LINE

6103 *****
6104 *****
6105 *
6106 *
6107 *
6108 *
6109 *
6110 *
6111 *
6112 *
6113 *
6114 *
6115 *
6116 *
6117 *
6118 *
6119 *
6120 *
6121 *
6122 *****
6123 *****
6124 *****

TRIM_FILE RELEASES THE UNUSED BLOCKS IN THE SPECIFIED FILE

NOTE: IF FILE BEING TRIMMED IS THE LAST FILE, THE UNUSED BLOCKS ARE ALLOCATED AS PART OF THE REMAINING AVAILABLE BLOCKS (RETURNED TO THE HOLE).

INPUT: A DEVICE ID
DE POINTER TO FILE NAME STRING

OUTPUT: CONDITION FLAGS
Z: NO ERRORS
NZ: ERRORS
A - ERROR CODE

6125 :GLB __TRIM_FILE *****
6126 :EXT FCB_BUFFER *****
6127 :EXT NEW_HOLE_SIZE *****
6128 :EXT HOLE_FILE_NAME *****
6129 *****
6130 *****

6131 __TRIM_FILE:

6132 PUSH HL
6133 PUSH IX
6134 PUSH IY
6135 PUSH DE
6136 PUSH AF
6137 LD HL,FCB_BUFFER
6138 CALL __QUERY_FILE ; FIND FILE IN DIRECTORY
6139 JP NZ,ERROR ; HAD TROUBLE FINDING IT
6140 LD IX,FCB_BUFFER ; POINT TO THE BUFFER
6141 LD E,[IX+FCB_USED_LENGTH]
6142 LD D,[IX+FCB_USED_LENGTH+1] ; GET THE NO OF USED BLOCKS
6143 LD L,[IX+FCB_MAX_LENGTH]
6144 LD H,[IX+FCB_MAX_LENGTH+1] ; GET THE NO ALLOCATED
6145 OR A
6146 SBC HL,DE ; FIND OUT HOW MANY EMPTY
6147 LD A,H
6148 OR L
6149 JP Z,NO_TRIM ; ALL USED

F241 E5

F242 DDE5

F244 FDE5

F246 D5

F247 F5

F248 21FDBA

F24B CDE61B

F24E C2F31A

F251 DD21FDBA

F255 DD5E13

F258 DD5614

F25B DD6E11

F25E DD6612

F261 B7

F262 ED52

F264 7C

F265 B5

F266 CAF311

LOCATION OBJECT CODE LINE SOURCE LINE

```

6160
F269 22FE1A LD [NEW_HOLE_SIZE],HL ; SAVE THE NUMBER OF EMPTY BLOCKS
F26C DD7311 LD [IX+FCB_MAX_LENGTH],E ; ALLOCATED = USED, NOW
F26F DD7212 LD [IX+FCB_MAX_LENGTH+1],D
F272 3AFDD4 LD A,[FILE_COUNT] ; FILE ENTRY POSITION IN DIRECTORY
F275 0627 LD B,ENT_PER_BLOCK
F277 TRY_AGAIN:
F277 90 SUB B ; IS IT THE LAST ENTRY IN THIS BLOCK
F278 2824 JR Z,LAST_ENTRY ; YUP
F27A 30FB JR NC,TRY_AGAIN ; NOPE, MUST BE IN NEXT BLOCK
F27C NOT_LAST_ENTRY:
F27C DD2AFDFD LD IX,[FCB_HEAD_ADDR] ; GET START OF FCB'S
F280 DD5E21 LD E,[IX+FCB_POINTER] ; GET ADDRESS OF THE MODIFIED FCB
F283 DD5622 LD D,[IX+FCB_POINTER+1]
F286 D5 PUSH DE
F287 01001A LD BC,DIR_ENT_LENGTH ; OFFSET TO NEXT DIR ENTRY'S ATTR BYTE
F28A C5 PUSH BC
F28B 21FDBA LD HL,FCB_BUFFER
F28E EDB0 LDIR
F290 D1 POP DE
F291 E1 POP HL
F292 19 ADD HL,DE ; POINT TO ATTR BYTE
F293 E5 PUSH HL
F294 FDE1 POP IY
F296 FDCB0C46 BIT ATTR_HOLE_BIT,[IY+DIR_ATTR] ; IS IT A HOLE?
F29A 285E JR Z,NOT_HOLE ; NOPE
F29C 183C JR ITS_HOLE
F29E LAST_ENTRY:
F29E 3AFDD4 LD A,[FILE_COUNT]
F2A1 32FDD5 LD [MOD_FILE_COUNT],A ; SAVE IT
F2A4 F1 POP AF
F2A5 D1 POP DE
F2A6 D5 PUSH DE
F2A7 F5 PUSH AF
F2AB 21FDBA LD HL,FCB_BUFFER
F2AB CDE651 CALL _SET_FILE ; PUT CHANGED DIRECTOR: AWAY
F2AE 206A JR NZ,ERRGR
F2B0 F1 POP AF
F2B1 D1 POP DE
6216
    
```

LOCATION OBJECT CODE LINE	SOURCE LINE				
F2B2 11F428	LD	DE,HOLE_FILE_NAME	:	NAME OF HOLE	
6217					
6218					
F2B5 D5	PUSH	DE			
F2B6 F5	PUSH	AF	:	SAVE DEVICE ID	
6221					
F2B7 21FDBA	LD	HL,FCB_BUFFER			
6222					
6223					
F2BA CDE61B	CALL	QUERY_FILE			
F2BD DD2AFDFD	LD	IX,[FCB_HEAD_ADDR]			
F2C1 DD6E21	LD	L,[IX+FCB_POINTER]	:	GET ADDRESS OF HOLE IN DIR BLOCK	
F2C4 DD6622	LD	H,[IX+FCB_POINTER+1]			
6227					
6228					
F2C7 E5	PUSH	HL			
F2C8 FDE1	POP	IX			
6230					
6231					
F2CA FDCB0C46	BIT	ATTR_HOLE_BIT,[IY+DIR_ATTR]	:	WAS IT A HOLE	
F2CE 284A	JR	Z,ERROR			
6232					
6233					
6234					
F2D0 21FDD4	LD	HL,FILE_COUNT			
F2D3 3AFDD5	LD	A,[MOD_FILE_COUNT]			
6236					
6237					
F2D6 3C	INC	A			
6238					
6239					
F2D7 BE	CP	[HL]			
F2D8 2037	JR	NZ,NO_TRIM			
6241					
6242					
6243					
F2DA	ITS_HOLE:	L,[IY+DIR_MAX_LENGTH]	:	GET CURRENT HOLE SIZE	
F2DA FD6E11	LD	H,[IY+DIR_MAX_LENGTH+1]			
F2DD FD6612	LD				
6246					
6247					
F2E0 ED58FE1A	LD	DE,[NEW_HOLE_SIZE]	:	BLOCKS TO BE RETURNED TO HOLE	
F2E4 19	ADD	HL,DE	:	NEW HOLE SIZE	
6249					
6250					
F2E5 FD7511	LD	[IY+DIR_MAX_LENGTH],L			
F2E8 FD7412	LD	[IY+DIR_MAX_LENGTH+1],H			
6252					
6253					
F2EB FD6E0D	LD	L,[IY+DIR_START_BLOCK]			
F2EE FD660E	LD	H,[IY+DIR_START_BLOCK+1]			
6254					
6255					
6256					
F2F1 B7	OR	A			
F2F2 ED52	SBC	HL,DE			
6257					
6258					
6259					
F2F4 FD750D	LD	[IY+DIR_START_BLOCK],L			
F2F7 FD740E	LD	[IY+DIR_START_BLOCK+1],H			
6260					
6261					
6262					
F2FA	NOT_HOLE:				
6263					
6264					
F2FA DD7E17	LD	A,[IX+FCB_DEVICE]	:	DEVICE NUMBER	
F2FD 2AFDFD	LD	HL,[FCB_DATA_ADDR]	:	GET ADDR OF MY BUFFER	
F300 DD5E19	LD	E,[IX+FCB_BLOCK]	:	GET BLOCK ADDRESS IN BCDE	
F303 DD561A	LD	D,[IX+FCB_BLOCK+1]			
F306 DD4E1B	LD	C,[IX+FCB_BLOCK+2]			
F309 DD461C	LD	B,[IX+FCB_BLOCK+3]			
6268					
6269					
6270					
6271					
F30C CDF1E6	CALL	WRITE_BLOCK	:	WRITE THE DIRECTORY BLOCK	
F30F 2009	JR	NZ,ERROR			
6272					
6273					

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
F311	D1	6274		
F312	D1	6275	NO_TRIM:	
F313	FDE1	6276	POP	DE
F315	DDE1	6277	POP	DE
F317	E1	6278	POP	IY
		6279	POP	IX
		6280	POP	HL
		6281		
F318	AF	6282	XOR	A
		6283		
F319	C9	6284	RET	
		6285		
F31A	B7	6286	ERROR:	
F31B	D1	6287	OR	A
F31C	D1	6288	POP	DE
F31D	FDE1	6289	POP	DE
F31F	DDE1	6290	POP	IY
		6291	POP	IX
		6292	POP	HL
		6293		
F322	C9	6294	RET	
		6295		
		6296		

: REALLY WAS AF

: REALLY AF

LOCATION OBJECT CODE LINE SOURCE LINE

```

6298 *****
6299 *
6300 *
6301 *
6302 *
6303 *
6304 *
6305 *
6306 *
6307 *
6308 *
6309 *
6310 *
6311 *
6312 *
6313 *
6314 *
6315 *
6316 *
6317 *
6318 *****
6319 *****
6320 *
6321 *
6322 *
6323 *
6324 *
6325 *
6326 *
6327 *
6328 *
6329 *
6330 *
6331 *
6332 *
6333 *
6334 *
6335 *
6336 *
6337 *
6338 *
6339 *
6340 *
6341 *
6342 *
6343 *
6344 *
6345 *
6346 *
6347 *
6348 *
6349 *
6350 *
6351 *
6352 *
6353 *
6354 *

        ___INIT_TP_DIR INITIALIZES THE TAPE DIRECTORY
INPUT:  A      DEVICE ID
        LOW NIBBLE - DEVICE ADDRESS
        HI NIBBLE - SECONDARY DEVICE ID
C      DE      NUMBER OF SECTORS IN DIRECTORY
        DE      NUMBER OF SECTORS ON DEVICE
HL     HL     POINTER TO ASCII ETX TERMINATED STRING
        ( VOLUME ID) (SHOULD BE BLANK-PADDED IF
        LESS THAN 12 CHARACTERS)
OUTPUT:
        CONDITION FLAGS
Z      NO ERROR
NZ     ERROR
A      A = ERROR CODE

        ___INIT_TAPE_DIR
:GLB  ___INIT_TAPE_DIR
:GLB  HOLE_FILE_NAME
:GLB  DIRECTORY_CHECK
:EXT  SECTOR_NO
:EXT  DEVICE_ID
:EXT  FCB_DATA_ADDR
:EXT  SECTORS_TO_INIT
:EXT  WRITE_BLOCK

F323  F323 FDE5
F325  F325 C5
F326  F326 D5
F327  F327 E5
F328  F328 32FD72
F328  F328 79
F32C  F32C 32FD86
F32F  F32F E5
F330  F330 CDF3AF
F333  F333 21F3DA
F336  F336 ED58FDF
F33A  F33A 010068
F33D  F33D EDB0
F33F  F33F E1
F340  F340 ED58FDF
F344  F344 060C

        [DEVICE_ID].A ; SAVE THE DEVICE ID
        A.C
[SECTORS_TO_INIT].A ; NO OF SECTORS TO INITIALIZE
HL
PUSH
CALL ZERO_OUT_BUFFER ; ZERO OUT THE 1K BUFFER
LD HL,INITIAL_DIRECTORY ; POINTER TO INITIAL SETUP
LD DE,[FCB_DATA_ADDR] ; ADDRESS OF 1K BUFFER
LD BC,DIR_ENT_LENGTH*3+VOL_DES_LENGTH ; # BYTES TO MOVE INTO BUFFER
LDIR
POP HL
LD DE,[FCB_DATA_ADDR]
LD B,12
        ; GET ADDR OF ETX TERMINATED ASCII STRING
        ; GET ADDRESS OF 1K BUFFER
        ; MAX OF 12 BYTES IN STRING
    
```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
F346	LOAD_DIR:	6355	
F346 7E	LD	6356	A.[HL]
F347 FE03	CP	6357	ETX
		6358	: GET ASCII CHAR OF VOLUME ID
		6359	: IS IT ETX?
F349 2808	JR	6360	Z.ITS_AN_ETX
		6361	: YUP
F34B 12	LD	6362	[DE].A
		6363	: NOPE,PUT IN THE 1K BUFFER
F34C 23	INC	6364	HL
F34D 13	INC	6365	DE
		6366	LOAD_DIR
F34E 10F6	DJNZ	6367	
		6368	
F350 3E03	LD	6369	A.ETX
F352 1B	DEC	6370	DE
		6371	
F353	ITS_AN_ETX:	6372	
F353 12	LD	6373	[DE].A
F354 FD2AFDF	LD	6374	IY,[FCB_DATA_ADDR]
		6375	: ETX TERMINATED DIRECTORY NAME
F358 3AFD86	LD	6376	A.[SECTORS_TO_INIT]
F35B F680	OR	6377	OBOH
F35D FD770C	LD	6378	[IY+VOL_DIRSIZE].A
		6379	: GET THE NO OF SECTORS TO INITIALIZE
F360 E1	POP	6380	HL
F361 D1	POP	6381	DE
F362 C1	POP	6382	BC
		6383	
F363 C5	PUSH	6384	BC
F364 D5	PUSH	6385	DE
F365 E5	PUSH	6386	HL
		6387	[IY+VOL_SIZE+0].E
F366 FD7311	LD	6388	[IY+VOL_SIZE+1].D
F369 FD7212	LD	6389	
		6390	
F36C D5	PUSH	6391	DE
F36D 11004E	LD	6392	DE,VOL_DES_LENGTH+DIR_ENT_LENGTH*2
F370 FD19	ADD	6393	IY,DE
F372 D1	POP	6394	DE
		6395	: GET TO FIRST DIR ENTRY
F373 FD71F7	LD	6396	[IY+DIR_MAX_LENGTH-DIR_ENT_LENGTH].C
F376 FD71F9	LD	6397	[IY+DIR_USED_LENGTH-DIR_ENT_LENGTH].C
		6398	: BLOCKS ALLOCATED FOR DIRECTORY
F379 OC	INC	6399	C
F37A FD710D	LD	6400	[IY+DIR_START_BLOCK].C
		6401	: FIRST "FILE" STARTS AFTER
		6402	: LAST SECTOR OF DIR
F37D 0600	LD	6403	B.O
		6404	: WANT BC = # SECTORS USED
		6405	: (# DIR SECT + 1 TO COVER
		6406	: SECT O)
F37F EB	EX	6407	DE,HL
F380 B7	OR	6408	A
F381 ED42	SBC	6409	HL,BC
		6410	[IY+DIR_MAX_LENGTH+0].L
F383 F1 11	LD	6411	[IY+DIR_MAX_LENGTH+1].H
F386 F4 12	LD	6412	

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		6412	
F389	3E01	LD	A,1 ; FIRST SECTOR
F38B	32FD87	LD	[SECTOR_NO],A
		6415	
F38E	CD3C5	CALL	WRITE_1_BLOCK ; WRITE BLOCK TO TAPE
		6417	
F391	2016	JR	NZ,WE_GOT_ERRORS ; CAME BACK WITH ERROR
		6419	
F393	3AFD86	LD	A,[SECTORS_TO_INIT] ; GET NO OF SECTORS TO INITIALIZE
		6421	
F396	47	LD	B,A ; ONE LESS BECAUSE ONE WAS JUST INITIALIZED
F397	05	DEC	B ; WAS ONLY ONE TO INIT
		6424	
F398	280E	JR	Z,NO_SECTORS_TO_INIT ; CLEAR 1K BUFFER
		6426	
F39A	CD3AF	CALL	ZERO_OUT_BUFFER ; CLEAR 1K BUFFER
		6428	
F39D			SECT_INIT_LOOP:
F39D	21FD87	LD	HL,SECTOR_NO ; SECTOR NO.
F3A0	34	INC	[HL] ; NEXT SECTOR
		6432	
F3A1	CD3C5	CALL	WRITE_1_BLOCK ; WRITE BLOCK OF ZEROS OUT
F3A4	2003	JR	NZ,WE_GOT_ERRORS ; GOT A ERROR IN WRITE
		6435	
F3A6	10F5	DJNZ	SECT_INIT_LOOP
		6437	
F3A8			NO_SECTORS_TO_INIT:
F3A8	AF	XOR	A ; ZERO CONDITION
		6440	
F3A9			WE_GOT_ERRORS:
		6443	
F3A9	E1	POP	HL
F3AA	D1	POP	DE
F3AB	C1	POP	BC
F3AC	FDE1	POP	IY
		6448	
F3AE	C9	RET	
		6449	
		6450	
		6451	

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		6453	
F3AF	ZERO_OUT_BUFFER:	6454	
		6455	
F3AF C5	PUSH BC	6456	
F3B0 D5	PUSH DE	6457	
F3B1 E5	PUSH HL	6458	
		6459	
F3B2 2AFDFF	LD HL,[FCB_DATA_ADDR]	6460	: POINTER TO 1K BUFFER
F3B5 ED5BDFDF	LD DE,[FCB_DATA_ADDR]	6461	
F3B9 13	INC DE	6462	
F3BA 0103FF	LD BC,1023	6463	
		6464	
F3BD 3600	LD [HL],0	6465	: ZERO IT OUT
		6466	
F3BF EDB0	LDIR	6467	
		6468	
F3C1 E1	POP HL	6469	
F3C2 D1	POP DE	6470	
F3C3 C1	POP BC	6471	
		6472	
F3C4 C9	RET	6473	
		6474	
		6475	
F3C5	WRITE_1_BLOCK:	6476	
F3C5 C5	PUSH BC	6477	
		6478	
F3C6 2AFDFF	LD HL,[FCB_DATA_ADDR]	6479	: ADDRESS OF 1K BUFFER
F3C9 3AFD87	LD A,[SECTOR_NO]	6480	: GET SECTOR NUMBER
F3CC 5F	LD E,A	6481	
F3CD 1600	LD D,O	6482	
F3CF 010000	LD BC,O	6483	
		6484	
F3D2 3AFD72	LD A,[DEVICE_ID]	6485	
		6486	
F3D5 CDF1E6	CALL WRITE_BLOCK	6487	: WRITE IT OUT
		6488	
F3D8 C1	POP BC	6489	
		6490	
F3D9 C9	RET	6491	
		6492	
		6493	
F3DA	INITIAL_DIRECTORY:	6494	
F3DA 20202020	DEFB ,	6495	
F3DF 20202020			
F3E4 2020			
F3E6 80	DEFB 10000000B	6496	
F3E7	DIRECTORY_CHECK:	6497	
F3E7 55AA00FF	DEFB 055H,0AAH,000H,0FFH	6498	
F3EB 0000	DEFB 0,0	6499	
F3ED 0000	DEFB 0,0	6500	
F3EF 0000	DEFB 0,0	6501	
F3F1 000000	DEFB 0,0,0	6502	
		6503	
F3F4 424F5403	DEFB 'BOOT',ETX,'	6504	
F3F9 20202020			
F3FE 20			
F400 80	DEFB ATTR_PERMANENT.OP_ATTR_SYS,	6505	

LOCATION OBJECT CODE LINE SOURCE LINE

```

F401 00000000 6506 DEFB 0,0,0,0
F405 0001 6507 DEFW 1
F407 0001 6508 DEFW 1
F409 0000 6509 DEFW 0
F40B 000000 6510 DEFB 0,0,0
F40E 449524543 6511 DEFB 'DIRECTORY',ETX,'
F413 544F525903 6512 DEFB
F418 2020
F41A C8
F41B 01000000 6513 DEFB ATTR_PERMANENT.OR.ATTR_WRITE_PROT.OR.ATTR_SYSTEM
F41F 0080 6514 DEFB 1,0,0,0
F421 0001 6515 DEFW 128
F423 0400 6516 DEFW 1
F425 000000 6517 DEFW 1024
F428 424C4F434B 6518 DEFB 0,0,0
F42D 53204C4546 6519 DEFB
F432 5403
F434 01
F435 00000000 6520 HOLE_FILE_NAME:
F439 0000 6521 DEFB 'BLOCKS LEFT',ETX
F43B 0000 6522 DEFB ATTR_HOLE
F43D 0000 6523 DEFB 0,0,0,0
F43F 570711 6524 DEFB 0,0
6525 DEFB 0,0
6526 DEFB 0,0
6527 DEFB 57H,07H,11H
6528
6529 INIT_INFO_SIZE EQU $-INITIAL_DIRECTORY
6530 ;
6531 ;
6532 ;
6533 ;
6534 ;
6535 ;

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

6537 NAME "REV 1 - RPD*"
6538 *
6539 * RETURN ERROR CODES UNTIL THESE ROUTINES ARE WRITTEN
6540 *
6541 * 10/12/83 VSB
6542 * 10/14/83 RPD changed __EOS_4 to __CV_A
6543 *
6544
6545 :GLB __POSIT_FILE,__EOS_1,__EOS_2,__EOS_3,__CV_A
6546 POSIT_FILE:
6547 __EOS_1:
6548 __EOS_2:
6549 __EOS_3:
6550 __CV_A:
6551 LD A,PROG_NON_EXIST ; SIGNIFY PROGRAM DOES NOT EXIST
6552 OR A ; NZ CONDITION
6553 RET
6554 ;
6555 ;
6556 ;
6557 ;
6558 ;
6559 ;

```

```

F442
F442
F442
F442
F442 3E17
F444 B7
F445 C9

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

6561 ;REV O (V/D 8/24/83)
6562
6563
6564 *****
6565 * THIS ROUTINE WILL FIND THE DCB FOR THE DEVICE DESIRED *****
6566 *
6567 * INPUT: A ==> POINTS TO THE DEVICE ID
6568 * OUTPUT:
6569 *
6570 * CONDITION FLAG:
6571 * Z : NO ERRORS
6572 * IY ==> POINTS TO THE START OF THE DEVICE DCB
6573 * NZ : ERROR OCCURED
6574 * IY ==> IS DESTROYED
6575 *****
6576 *****
6577 *****
6578 ;GLB GET_DCB_ADDR
6579 GLOBAL _FIND_DCB
6580 ;EXT CURRENT_PCB
6581
6582 _GET_DCB_ADDR:
6583 _FIND_DCB:
6584 PUSH BC
6585 PUSH DE
6586
6587 LD C,A
6588 LD IY,[CURRENT_PCB]
6589 LD B,[IY+P_NUM_DCBS]
6590 XOR A
6591 CP B
6592 JR Z,DCB_NO_FIND
6593
6594 LD DE,P_SIZE
6595 ADD IY,DE
6596 LD DE,D_SIZE
6597 LD A,C
6598 AND OFH
6599
6600
6601 DCB_LOOP:
6602 CP [IY+D_DEV_ADDR]
6603 JR Z,FOUND_IT
6604
6605 ADD DJNZ DCB_LOOP
6606
6607 JR DCB_NO_FIND
6608
6609
6610
6611 FOUND_IT:
6612 LD A,C
6613 JR FINISHED
6614
6615 DCB_NO_FIND:
6616 LD A,DCB_NOT_FOUND
6617 OR A

```

```

: SAVE THE DEVICE ID
: GET THE START ADDRESS OF PCB
: GET THE NUMBER OF DCB'S
: ARE THERE ZERO DCBS?
: YES
: NO
: GET THE SIZE OF THE PCB
: ADD TO IY TO GET TO THE START OF THE DCB
: GET THE DCB SIZE
: RELOAD THE DEVICE ID
: AND OF THE HIGH BITS TO GET ONLY
: THE DEVICE ID

: IS THIS THE CORRECT DEVICE DCB?
: YES
: NO
: GET TO THE NEXT DCB AND CHECK IT
: GO CHECK THEM ALL UNTIL ONE IS FOUND

: RESTORE A TO ITS ORIGINAL STATE

```

LOCATION OBJECT CODE LINE SOURCE LINE

F470	6618		
F470 D1	6619	FINISHED:	
F471 C1	6620	POP	DE
	6621	POP	BC
	6622		
F472 C9	6623	RET	
	6624		
	6625		

```

LOCATION OBJECT CODE LINE SOURCE LINE
6627
6628 *****
6629 * THIS ROUTINE WILL INITIATE A STATUS REQUEST COMMAND, AND WILL RETURN *****
6630 * THE RESULT OF THE COMMAND.
6631 * INPUT: A=> DEVICE ID
6632 * OUTPUT: CONDITION BITS
6633 * Z: NO ERRORS
6634 * IY CONTAINS START ADDRESS OF DCB
6635 * NZ: ERROR OCCURED
6636 * A=> ERROR CODE
6637 * IY IS DESTROYED (MAY BE ADDRESS OF DCB
6638 * UNDER CERTAIN ERROR CONDITIONS)
6639 *
6640 *****
6641 ;GLB __REQUEST_STATUS
6642
6643
6644
6645 __REQUEST_STATUS:
6646 CALL __FIND_DCB ; GET THE CORRECT DEVICE DCB
6647
6648 JR NZ,END_R_S
6649
6650 LD [IY+D_COM_STAT],DCB_STATUS ; SEND THE COMMAND TO REQUEST
6651 ; A STATUS
6652
6653 REQUEST_LOOP:
6654 CMND_COMPLETE_BIT,[IY+D_COM_STAT] ; HAS THE COMMAND COMPLETED?
6655 JR Z,REQUEST_LOOP ; NO LOOP TILL IT DOES
6656
6657 LD A,[IY+D_COM_STAT] ; CHECK THE STATUS
6658 CP CMND_FIN_STATUS ; DID IT FINISH WITH NO ERRORS?
6659 ; IF ZERO EVERYTHING IS OK
6660 ; IF NZ THERE WAS AN ERROR
6661
6662 END_R_S:
6663 F487 C9 RET
6664
6665
F473
F473 CDF446
F476 200F
F478 FD360001
F47C
F47C FDCB007E
F480 28FA
F482 FD7E00
F485 FE80
F487 C9

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

6667
6668 :GLB __RD_DEV_DEP_STAT
6669
F488 __RD_DEV_DEP_STAT:
F488 FDE5 PUSH IY ; SAVE THIS REGISTER
F48A CDF446 CALL __FIND_DCB ; GET THE CORRECT DEVICE DCB
6673
F48D 2004 JR NZ.END_R_D_D_S
6674
F48F AF XOR A ; NO ERROR
6677
F490 FD7E14 LD A,[IY+D_STATUS_FLAGS] ; READ DEVICE-DEPENDENT STATUS FLAGS
F493
F493 FDE1 END_R_D_D_S:
F495 C9 POP IY ; RESTORE THE REGISTER
6683
6683 RET

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
6685 *****
6686 * THIS ROUTINE WILL CHECK AND SEE IF THE DEVICE DCB IS CURRENTLY ACTIVE.
6687 * IF IT IS THEN A NZ CONDITION WILL BE SENT BACK TO THE CALLING PROGRAM.
6688 * INPUT: IY ==> POINTS TO START OF DEVICE DCB
6689 * OUTPUT: CONDITION FLAGS
6690 * Z : DCB IS INACTIVE, IN OTHERWORDS IT IS AVAILABLE
6691 * A ==> IS PRESERVED
6692 * NZ: DCB IS ACTIVE
6693 * A ==> ERROR CODE
6694 *
6695 *****
6696
6697 ; OK CHANGED TO G_OK TO PREVENT CONFLICT WITH IDENTICAL
6698 ; LABEL ELSEWHERE
6699
6700 ;GLB CHK_IF_INACTIVE
6701
6702 CHK_IF_INACTIVE:
6703 _PUSH BC
6704
6705 LD C,A
6706 LD A,[IY+D_COM_STAT]
6707 CP DCB_IDLE
6708 JR Z,G_OK
6709
6710 BIT CMND_COMPLETE_BIT,[IY+D_COM_STAT]
6711
6712 JR Z,DEVICE_BUSY
6713
6714 G_OK:
6715 XOR A
6716 LD A,C
6717 JR DONE
6718
6719 DEVICE_BUSY:
6720 _INC
6721 LD A,DCB_BUSY
6722
6723 DONE:
6724 POP BC
6725 RET
6726
6727

```

```

; PRESERVE A
; IS THE DCB AVAILABLE?
; YES IT IS INACTIVE
; NO IT IS NOT
; PROCESSED
; NO, THE DEVICE IS STILL BUSY
; YES, THE DCB IS AVAILABLE
; INSURE ZERO CONDITION
; RESTORE A
;
; INSURE A NON ZERO CONDITION FOR FLAG
; SEND BACK THE ERROR CODE

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

6729 *****
6730 *****
6731 *
6732 *   CHK FOR IDLE CHECKS IF THE REQUESTED DEVICE IS IDLE
6733 *   SENDING_BACK AN ERROR CODE
6734 *
6735 *   INPUT:  IY   POINTER TO THE DCB ,
6736 *
6737 *   OUTPUT: CONDITION FLAGS
6738 *           Z   DEVICE NOT IDLE
6739 *           NZ  DEVICE IDLE
6740 *
6741 *****
6742 *****
6743 *****
6744 *****
6745 *****
6746 *****
6747 *****
6748 *****
6749 *****
6750 *****
6751 *****
6752 *****
6753 *****
6754 *****
6755 *****
6756 *****
6757 *****
6758 *****
6759 *****
6760 *****
6761 *****
6762 *****
6763 *****
6764 *****
6765 *****
6766 *****
6767 *****
6768 *****

F4AE          :GLB      CHK_IF_IDLE
F4AE FD7E00   LD      A,[IY+D_COM_STAT]      : GET STATUS BYTE
F4B1 B7       OR      A                      : IS IT IDLE(ZERO)?
F4B2 2004     JR      NZ,NOT_IDLE           : NOPE
F4B4 3E03     LD      A,DCB_IDLE_ERR        : ERROR CODE FOR DEVICE IDLE
F4B6 B7       OR      A                      : CREATE NZ CONDITION
F4B7 C9       RET
F4B8          NOT_IDLE:
F4B8 AF       XOR      A                      : CLEAR ZERO FLAG
F4B9 C9       RET
6764 ;
6765 ;
6766 ;
6767 ;
6768 ;

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

6770 :REV 0 (V/D 8-24-83)
6771 *****
6772 * THIS ROUTINE WILL READ A CHARACTER FROM THE KEYBOARD *****
6773 *
6774 *
6775 * INPUT: NOTHING
6776 * OUTPUT: CONDITION FLAGS
6777 * Z : NO ERRORS
6778 * A ==> CONTAINS THE KEY READ
6779 * NZ: ERROR OCCURED
6780 * A ==> ERROR CODE
6781 *
6782 *
6783 *****
6784 *****
6785 GLOBAL RD_KBD
6786 :EXT KEYBOARD_BUFFER
6787 :EXT RD_CH_DEV
6788
6789 RD_KBD:
6790 PUSH BC
6791 PUSH DE
6792
6793 RD_KBD_LOOP:
6794
6795 CALL __START_RD_KBD
6796
6797 JR NZ,ERROR_OCCURED
6798
6799 LD C,A
6800
6801 WAITING:
6802 LD A,C
6803
6804 CALL __END_RD_KBD
6805
6806 JR NC,WAITING
6807
6808
6809 ERROR_OCCURED:
6810 POP DE
6811 POP BC
6812
6813 RET
6814

```

```

: PRESERVE THE DEVICE ID
: RESTOR THE DEVICE ID
: BC IS SAVED IN ALL CALLED MODULES
: THE OPERATION HAS NOT COMPLETED YET
: AT THIS POINT WE EITHER HAVE THE
: KEY OR AN ERROR CODE IN A

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

6816 *****
6817 * THIS ROUTINE WILL REQUEST A STATUS ON THE KEYBOARD *****
6818 * INPUT: NOTHING
6819 * OUTPUT: CONDITION FLAG
6820 * Z: NO ERRORS
6821 * NZ: ERROR OCCURED
6822 * A ==> ERROR CODE
6823 *
6824 *****
6825 GLOBAL __REQ_KBD_STAT
6826
6827
6828 :EXT __REQUEST_STATUS
6829
6830 __REQ_KBD_STAT:
6831 LD A,KEYBOARD_ID ; WANT STATUS ON THE KEYBOARD
6832 JP __REQUEST_STATUS ;
6833

```

F4CB
F4CB 3E01
F4CD C3F473

LOCATION OBJECT CODE LINE SOURCE LINE

```

6835 *****
6836 * THIS ROUTINE WILL START A READ DATA COMMAND ON THE KEYBOARD *****
6837 * INPUT: NOTHING
6838 * OUTPUT: CONDITION FLAGS
6839 * Z: NO ERROR
6840 * NZ: ERROR OCCURED
6841 * A ==> ERROR CODE
6842 *****
6843
6844 ;GLB __START_RD_KBD
6845
6846 ;EXT __START_RD_CH_DEV
6847
6848 __START_RD_KBD:
6849 PUSH BC
6850 PUSH DE
6851
6852 STRT_RD_KBD_LOOP:
6853 LD A,KEYBOARD_ID
6854 LD DE,KEYBOARD_BUFFER
6855 LD BC,1
6856 CALL __START_RD_CH_DEV
6857
6858 POP DE
6859 POP BC
6860
6861 RET
6862
F4D0
F4D0 C5
F4D1 D5
F4D2
F4D2 3E01
F4D4 11FD75
F4D7 010001
F4DA CDFB86
F4DD D1
F4DE C1
F4DF C9

```

```

: DESTINATION OF WHERE TO PLACE KEY
: GET 1 CHARACTER
: GO INITIATE THE COMMAND

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
6864
6865 *****
6866 * THIS ROUTINE WILL CHECK THE RESULT OF A COMMAND THAT WAS SENT. IF THE
6867 * RESULT IS A NAK THEN THE ROUTINE WILL ISSUE A NEW READ COMMAND.
6868 * INPUT: NOTHING
6869 * OUTPUT: CONDITION FLAG
6870 * C : COMMAND HAS FINISHED
6871 * NC: COMMAND HAS NOT BEEN FINISHED
6872 * Z : NO ERROR OCCURED
6873 * NZ: ERROR OCCURED
6874 * A ==> ERROR CODE
6875 *****
6876 :GLB _END_RD_KBD
6877 :EXT _END_RD_CH_DEV
6878
6879
6880 _END_RD_KBD:
6881 LD A,KEYBOARD_ID
6882 CALL _END_RD_CH_DEV
6883
6884 JR NC,ROUTINE_DONE
6885
6886 JR Z,HAVE_THE_KEY
6887
6888
6889 CP KBD_NAK
6890 JR NZ,NO_NAK
6891
6892 CALL _START_RD_KBD
6893 JR Z,SHOW_COMPLETE
6894
6895
6896
6897
6898
6899
6900 NO_NAK: SCF
6901 JR ROUTINE_DONE
6902
6903
6904 HAVE_THE_KEY:
6905 LD A,[KEYBOARD_BUFFER]
6906 JR ROUTINE_DONE
6907
6908 SHOW_COMPLETE:
6909 OR A
6910
6911 ROUTINE_DONE:
6912 RET
6913 :
6914 :
6915 :
6916 :
6917 :

```

```

: SEND THIS PARM TO _END_RD_CH_DEV
: CHECK THE RESULT
: CARRY FLAG IS CLEARED, CMND IS NOT COMPLETE
: THE COMMAND IS COMPLETE
: ROUTINE COMPLETED WITH NO ERRORS.
: CARRY FLAG IS SET. ZERO CONDITION
: THERE WAS AN ERROR
: WAS THE ERROR A NAK?
: NO, GO SET C FLAG TO SHOW COMMAND COMPLETED
: YES, TRY THE COMMAND AGAIN
:
: THERE WAS AN ERROR IN THE COMMAND RETRY
: SET THE CARRY FLAG TO SHOW THAT THE
: OPERATION COMPLETED. THE FLAG
: CONDITION WILL BE NON-ZERO TO
: INDICATE AN ERROR OCCURED. A CONTAINS
: THE ERROR CODE
:
: PUT THE KEY IN THE KEYBOARD BUFFER
:
: CLEAR CARRY FLAG BECAUSE THE RETRY WAS STARTED
: WITH NO PROBLEMS. THE COMMAND IS NOT COMPLETE

```

```

LOCATION OBJECT CODE LINE      SOURCE LINE
6920
6921 ;REV 1                    (V/D 8-24-83)
6922 *****
6923 * PRINT_CHAR ALLOWS THE CALLER TO PRINT ONE CHARACTER
6924 * INPUT: A ==> CHARACTER TO BE PRINTED
6925 * OUTPUT: CONDITION FLAGS
6926 *        Z :        NO ERROR
6927 *        NZ:        ERROR OCCURED
6928 *        A ==> ERROR CODE
6929 *
6930 *****
6931
6932 ;GLOBAL            PR_CH
6933 ;EXT              PRINT_BUFFER
6934 ;EAT              CHK_IF_INACTIVE
6935
6936                    PR_CH:
6937                    PUSH    BC
6938                    PUSH    DE
6939                    PUSH    HL
6940                    PUSH    IY
6941
6942                    LD        [PRINT_BUFFER].A
6943                    LD        A,ETX
6944                    LD        [PRINT_BUFFER+1].A
6945                    LD        HL,PRINT_BUFFER
6946
6947                    CALL     __PR_BUFF
6948
6949                    POP        IY
6950                    POP        HL
6951                    POP        DE
6952                    POP        BC
6953
6954                    RET
6955
F4FC                    F501 32FD76                    ; TERMINATE STRIN W/ETX
F4FD C5                F504 3E03                    ;
F4FE D5                F506 32FD77                    ; ADDRESS OF PRINT BUFFER
F4FE E5                F509 21FD76                    ;
F4FF FDE5              F50C CDF515
F50F FDE1              F50F FDE1
F511 E1                F511 E1
F512 D1                F512 D1
F513 C1                F513 C1
F514 C9                F514 C9

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

6957 *****
6958 * PRINT_BUFFER ALLOWS THE CALLER TO PRINT AN ASCII ETX-TERMINATED STRING *****
6959 * INPUT: HL ==> ADDRESS OF ETX-TERMINATED STRING
6960 * OUTPUT: CONDITION FLAGS
6961 * Z : NO ERROR
6962 * NZ: ERROR OCCURED
6963 * A ==> ERROR CODE
6964 *
6965 *****
6966 GLOBAL __PR_BUFF
6967
6968 :EXT __FIND_DCB,__WR_CH_DEV
6969
6970
6971 __PR_BUFF.
6972 PUSH BC
6973 PUSH DE
6974 PUSH HL
6975 PUSH IY
6976
6977 PR_BUFF2:
6978 LD A,PRINTER_ID
6979 CALL __FIND_DCB
6980
6981 JR NZ,END_PRINT_ROUTINE
6982
6983 CALL CHK_IF_INACTIVE
6984 JR NZ,PRTR_DCB_ACTIVE
6985
6986 CHECK_FOR_ETX:
6987 LD BC,O
6988 LD E,L
6989 LD D,H
6990
6991 GET_NEXT_ASCII:
6992 LD A,ETX
6993 CP [HL]
6994 JR Z,ITS_ETX
6995
6996
6997 INC HL
6998 INC C
6999 LD A,C
7000 CP 16
7001 JR NZ,GET_NEXT_ASCII
7002
7003 EX DE,HL
7004
7005 WRITE_COMMAND_AGAIN:
7006
7007 LD A,PRINTER_ID
7008 CALL __WR_CH_DEV
7009
7010 JR Z,WRITE_DONE
7011
7012 CP PR_NAK
7013 JR NZ,PRINT_CMND_FAILED

```

```

: WANT THE PRINTERS DCB
: GET THE DEVICE'S DCB
: THERE WAS A PROBLEM FINDING THE DCB
: CHARACTER COUNT
: SAVE THE STRING
: DE=HL= PTR TO UNPRINTED STRING
: IS IT AN ETX?
: YES
: NO
: POINT TO NEXT CHARACTER
: INC COUNT
: HAVE WE LOOKED AT 16 BYTES?
: YES
DE = PTR TO UNPRINTED STRNG, DE=PTR TO UNPRINTED+16
HL = PTR TO UNPRINTED STRNG, DE=PTR TO UNPRINTED+16
: WRITE COMPLETED WITH NO ERRORS
: IT A NAK?
: AN ERROR

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
F543	18F3	7014	JR	WRITE_COMMAND_AGAIN ; SEND ANOTHER WRITE COMMAND
F545	EB	7017	WRITE_DONE:	
F546	18DE	7018	EX	DE,HL ; DE = PTR TO UNPRINTED STRNG, HL=PTR TO UNPRINTED+16
		7019	JR	CHECK_FOR_ETX ; GO CHECK THE NEXT CHARACTER
F548	EB	7022	ITS_ETX:	
F549	AF	7023	EX	DE,HL ; ZERO OUT A
F54A	B9	7024	XOR	A ; WAS THE FIRST CHARACTER CHECKED
		7025	CP	C ; AN ETX?
F54B	280B	7026	JR	Z,END_PRINT_ROUTINE ; YUP, DON'T BOTHER PRINTING IT
F54D	3E02	7028	SEND_ANOTHER_WRITE:	
F54E	CDFB75	7029	LD	A,PRINTER_ID ; WILL WRITE TO THE PRINTER
		7030	CALL	__WR_CH_DEV ; WRITE CHARACTERS
F552	2804	7032	JR	Z,COMMAND_FINISHED_SUCCESSFULLY ; ALL DONE
F554	FE86	7034	CP	PR_NAK ; DO WE HAVE A NAK?
F556	28F5	7035	JR	Z,SEND_ANOTHER_WRITE ; YUP
		7036		
		7037		
		7038		
		7039		
F558		7040	COMMAND_FINISHED_SUCCESSFULLY:	
F558		7041	PRINT_CMND_FAILED:	
F558		7042	PRNT_CMND_FAILED:	
F558		7043	PRTR_DCB_ACTIVE:	
F558	FDE1	7044	END_PRINT_ROUTINE:	
F55A	E1	7045	POP	IY
F55B	D1	7046	POP	HL
F55C	C1	7047	POP	DE
		7048	POP	BC
		7049		
F55D	C9	7050	RET	
		7051		
		7052		

; ON ERROR, ZERO FLAG SET AND A
; CONTAINS THE ERROR CODE

LOCATION OBJECT CODE LINE SOURCE LINE

```

7054 *****
7055 *
7056 *   SETUP_PR_BUFF SETS UP PRINT_BUFFER AND THE HL REGISTER AND
7057 *   CALLS __START_PR_BUFF
7058 *   INPUT: A = ASCII CHARACTER
7059 *   OUTPUT: CONDITION FLAGS
7060 *   Z: NO ERROR
7061 *   NZ: ERROR OCCURED
7062 *   A = ERROR CODE
7063 *
7064 *****
7065 *GLB   __SETUP_PR_BUFF
7066
7067
7068 *   SETUP_PR_BUFF:
7069   LD   [PRINT_BUFFER],A
7070   LD   A,ETX
7071   LD   [PRINT_BUFFER+1],A
7072
7073   LD   HL,PRINT_BUFFER
7074
7075   CALL __START_PR_BUFF
7076
7077   RET
7078
7079
7080

```

```

F55E
F55E 32FD76
F561 3E03
F563 32FD77
F566 21FD76
F569 CDF580
F56C C9

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
7082 *****
7083 *
7084 * __START_PR_CH SETS UP PRINT_BUFFER AND THE HL REGISTER AND
7085 * CALLS __START_PR_BUFF
7086 * INPUT: A = ASCII CHARACTER
7087 * OUTPUT: CONDITION FLAGS
7088 * Z: NO ERROR
7089 * NZ: ERROR OCCURED
7090 * A = ERROR CODE
7091 *
7092 *****
7093
7094 :GLB __START_PR_CH
7095
7096 __START_PR_CH:
7097 LD [PRINT_BUFFER],A
7098 LD A,ETX
7099 LD [PRINT_BUFFER+1],A
7100
7101 LD HL,PRINT_BUFFER
7102
7103 CALL __START_PR_BUFF
7104
7105 RET
7106
7107
7108
F56D
F56D 32FD76
F570 3E03
F572 32FD77
F575 21FD76
F578 CDF580
F57B C9

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

7110 *****
7111 *
7112 *   END_PR_CH   CALLS_END_PR_BUFF TO CHECK THE DCB STATUS BYTE AND
7113 *   RETURNS THE RESULT TO THE CALLER
7114 *
7115 *   INPUT:  NONE
7116 *
7117 *   OUTPUT: CONDITION FLAGS
7118 *           C:   COMMAND COMPLETED
7119 *           NC:  COMMAND NOT COMPLETED
7120 *           Z:   NO ERRORS OCCURED
7121 *           NZ:  ERRORS
7122 *           A = ERROR CODE
7123 *
7124 *   ALL   REGISTERS ARE PRESERVED
7125 *
7126 * *****
7127 *
7128 *           :GLB   __END_PR_CH
7129 *           __END_PR_CH:
7130 *           __CALL  __END_PR_BUFF
7131 *
7132 *           RET
7133 *
7134 *

```

F57C
F57C CDF588

F57F C9

LOCATION OBJECT CODE LINE SOURCE LINE

```

7136 *****
7137 * __START_PR_BUFF ALLOWS THE CALLER TO PRINT AN ASCII ETX-TERMINATED STRING *****
7138 * INPUT: HL ==> ADDRESS OF ETX-TERMINATED STRING
7139 * OUTPUT: CONDITION FLAGS
7140 * Z : NO ERROR
7141 * NZ: ERROR OCCURED
7142 * A ==> ERROR CODE
7143 *
7144 *****
7145
7146 :GLB __START_PR_BUFF
7147
7148 :EXT __START_WR_CH_DEV
7149
7150 __START_PR_BUFF:
7151 PUSH BC
7152 PUSH DE
7153 PUSH HL
7154 PUSH IY
7155
7156 LD A,PRINTER_ID ; WANT THE PRINTERS DCB
7157 CALL __FIND_DCB ; GET THE DEVICE'S DCB
7158
7159 JR NZ,NO_DCB_FOUND1 ; THERE WAS A PROBLEM FINDING THE DCB
7160
7161 CALL CHK_IF_INACTIVE
7162 JR NZ,PRTR_DCB_ACTIVE1
7163
7164 LD BC,O ; CHARACTER COUNT
7165 LD E,L ; SAVE THE STRING
7166 LD D,H ; DE=HL= PTR TO UNPRINTED STRING
7167
7168 GET_NEXT_ASCII2:
7169 LD A,ETX
7170 CP [HL] ; IS IT AN ETX?
7171 JR Z,ITS_ETX2 ; YES
7172
7173
7174 INC HL ; NO
7175 INC C ; POINT TO NEXT CHARACTER
7176 LD A,C ; INC COUNT
7177 CP 16 ; HAVE WE LOOKED AT 16 BYTES?
7178 JR NZ,GET_NEXT_ASCII2 ; YES
7179 ; DE = PTR TO UNPRINTED STRNG, DE=PTR TO UNPRINTED+16
7180 EX DE,HL ; HL = PTR TO UNPRINTED STRNG, DE=PTR TO UNPRINTED+16
7181 LD A,PRINTER_ID
7182 CALL __START_WR_CH_DEV
7183
7184 JR WRITE_CMND_SENT
7185
7186 ITS_ETX2:
7187 EX DE,HL
7188 XOR A
7189 CP C ; ZERO OUT A
7190 ; WAS THE FIRST CHARACTER CHECKED
7191 ; AN ETX?
7192 LD A,PRINTER_ID ; WILL WRITE TO THE PRINTER
7193
7194
7195
7196
7197
7198
7199
7200
7201
7202
7203
7204
7205
7206
7207
7208
7209
7210
7211
7212
7213
7214
7215
7216
7217
7218
7219
7220
7221
7222
7223
7224
7225
7226
7227
7228
7229
7230
7231
7232
7233
7234
7235
7236
7237
7238
7239
7240
7241
7242
7243
7244
7245
7246
7247
7248
7249
7250
7251
7252
7253
7254
7255
7256
7257
7258
7259
7260
7261
7262
7263
7264
7265
7266
7267
7268
7269
7270
7271
7272
7273
7274
7275
7276
7277
7278
7279
7280
7281
7282
7283
7284
7285
7286
7287
7288
7289
7290
7291
7292
7293
7294
7295
7296
7297
7298
7299
7300
7301
7302
7303
7304
7305
7306
7307
7308
7309
7310
7311
7312
7313
7314
7315
7316
7317
7318
7319
7320
7321
7322
7323
7324
7325
7326
7327
7328
7329
7330
7331
7332
7333
7334
7335
7336
7337
7338
7339
7340
7341
7342
7343
7344
7345
7346
7347
7348
7349
7350
7351
7352
7353
7354
7355
7356
7357
7358
7359
7360
7361
7362
7363
7364
7365
7366
7367
7368
7369
7370
7371
7372
7373
7374
7375
7376
7377
7378
7379
7380
7381
7382
7383
7384
7385
7386
7387
7388
7389
7390
7391
7392
7393
7394
7395
7396
7397
7398
7399
7400
7401
7402
7403
7404
7405
7406
7407
7408
7409
7410
7411
7412
7413
7414
7415
7416
7417
7418
7419
7420
7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432
7433
7434
7435
7436
7437
7438
7439
7440
7441
7442
7443
7444
7445
7446
7447
7448
7449
7450
7451
7452
7453
7454
7455
7456
7457
7458
7459
7460
7461
7462
7463
7464
7465
7466
7467
7468
7469
7470
7471
7472
7473
7474
7475
7476
7477
7478
7479
7480
7481
7482
7483
7484
7485
7486
7487
7488
7489
7490
7491
7492
7493
7494
7495
7496
7497
7498
7499
7500
7501
7502
7503
7504
7505
7506
7507
7508
7509
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600
7601
7602
7603
7604
7605
7606
7607
7608
7609
7610
7611
7612
7613
7614
7615
7616
7617
7618
7619
7620
7621
7622
7623
7624
7625
7626
7627
7628
7629
7630
7631
7632
7633
7634
7635
7636
7637
7638
7639
7640
7641
7642
7643
7644
7645
7646
7647
7648
7649
7650
7651
7652
7653
7654
7655
7656
7657
7658
7659
7660
7661
7662
7663
7664
7665
7666
7667
7668
7669
7670
7671
7672
7673
7674
7675
7676
7677
7678
7679
7680
7681
7682
7683
7684
7685
7686
7687
7688
7689
7690
7691
7692
7693
7694
7695
7696
7697
7698
7699
7700
7701
7702
7703
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713
7714
7715
7716
7717
7718
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736
7737
7738
7739
7740
7741
7742
7743
7744
7745
7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757
7758
7759
7760
7761
7762
7763
7764
7765
7766
7767
7768
7769
7770
7771
7772
7773
7774
7775
7776
7777
7778
7779
7780
7781
7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793
7794
7795
7796
7797
7798
7799
7800
7801
7802
7803
7804
7805
7806
7807
7808
7809
7810
7811
7812
7813
7814
7815
7816
7817
7818
7819
7820
7821
7822
7823
7824
7825
7826
7827
7828
7829
7830
7831
7832
7833
7834
7835
7836
7837
7838
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7850
7851
7852
7853
7854
7855
7856
7857
7858
7859
7860
7861
7862
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7890
7891
7892
7893
7894
7895
7896
7897
7898
7899
7900
7901
7902
7903
7904
7905
7906
7907
7908
7909
7910
7911
7912
7913
7914
7915
7916
7917
7918
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7930
7931
7932
7933
7934
7935
7936
7937
7938
7939
7940
7941
7942
7943
7944
7945
7946
7947
7948
7949
7950
7951
7952
7953
7954
7955
7956
7957
7958
7959
7960
7961
7962
7963
7964
7965
7966
7967
7968
7969
7970
7971
7972
7973
7974
7975
7976
7977
7978
7979
7980
7981
7982
7983
7984
7985
7986
7987
7988
7989
7990
7991
7992
7993
7994
7995
7996
7997
7998
7999
8000

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
F5AF	C4FBC2	7193	CALL	NZ,___START_WR_CH_DEV
		7194		
F5B2		7195	NO_DCB_FOUND1:	
F5B2		7196	WRITE_CMND_SENT:	
		7197		
F5B2	FDE1	7198	POP	IY
F5B4	E1	7199	POP	HL
F5B5	D1	7200	POP	DE
F5B6	C1	7201	POP	BC
		7202		
F5B7	C9	7203	RET	
		7204		
		7205		

: NOPE, WRITE CHARACTER
: IS PASSED BACK TO CALLER

```

LOCATION OBJECT CODE LINE SOURCE LINE
7207
7208 *****
7209 * END_PR_BUFF CHECKS THE DCB COMM/STATUS BYTE AND RETURNS THE RESULT
7210 * OF THE CHECK
7211 *
7212 * INPUT: NONE
7213 *
7214 * OUTPUT: CONDITION FLAGS
7215 * C: COMMAND COMPLETED
7216 * NC: COMMAND NOT COMPLETED
7217 * Z: NO ERRORS OCCURED
7218 * NZ: ERRORS
7219 * A = ERROR CODE
7220 *
7221 * ALL REGISTERS ARE PRESERVED
7222 *
7223 * *****
7224 *
7225 * :GLB END_PR_BUFF
7226 * :EXT END_WR_CH_DEV
7227 *
7228 *
7229 * END_PR_BUFF:
7230 *
7231 * PUSH IY
7232 *
7233 * LD A,PRINTER_ID
7234 * CALL FIND_DCB
7235 * CALL END_WR_CH_DEV ; CHECK STATUS BYTE
7236 *
7237 * JR NC,CMND_NOT_PROCESSED ; COMMAND NOT COMPLETED
7238 *
7239 * JR Z,FINISHED_WO_ERROR ; COMMAND COMPLETED, NO ERROR
7240 *
7241 * CP PR_NAK ; WAS IT A NAK?
7242 *
7243 * JR NZ,NOT_PR_NAK ; NOPE, MUST BE AN ERROR
7244 *
7245 * LD [IY+D_COM_STAT],DCB_WR ; SEND ANOTHER WRITE COMMAND
7246 *
7247 * OR A ; CLEAR CARRY TO INDICATE COMMAND
7248 * ; HAS NOT BEEN FINISHED
7249 *
7250 * NOT_PR_NAK:
7251 * CMND_NOT_PROCESSED:
7252 * FINISHED_WO_ERROR:
7253 *
7254 * POP IY
7255 *
7256 * RET
7257 *
F588
F588 FDE5
F58A 3E02
F58C CDF446
F58F CDFBE1
F5C2 300B
F5C4 2809
F5C6 FEB6
F5C8 2005
F5CA FD360003
F5CE B7
F5CF
F5CF FDE1
F5D1 C9

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

7259 *****
7260 * THIS ROUTINE REQUESTS THE STATUS ON THE PRINTER
7261 * INPUT: NONE
7262 * OUTPUT:
7263 *
7264 *          NO ERROR
7265 *          ERROR OCCURED
7266 *          A ==> ERROR CODE
7267 *
7268 *****
7269 GLOBAL  __REQ_PR_STAT
7270
7271 :EXT  __REQUEST_STATUS
7272
7273 __REQ_PR_STAT:
7274 LD  A,PRINTER_ID
7275 JP  __REQUEST_STATUS
7276 :
7277 :
7278 :
7279 :
7280 :
7281

```

F502
F502 3E02
F504 C3F473

LOCATION OBJECT CODE LINE SOURCE LINE

```

7283
7284 :REV 0 (V/D 8-24-83)
7285
7286 GLOBAL __REQ_TAPE_STAT
7287
7288 :EXT __REQUEST_STATUS
7289
7290 *****
7291 * THIS ROUTINE REQUESTS THE STATUS ON THE TAPE DRIVE *****
7292 * INPUT: NONE
7293 * OUTPUT:
7294 * CONDITION FLAGS
7295 * Z : NO ERROR
7296 * NZ: ERROR OCCURED
7297 * A ==> ERROR CODE
7298
F5D7 __REQ_TAPE_STAT:
F5D7 3E08 LD A,TAPE_ID
F5D9 C3F473 JP __REQUEST_STATUS
7302 :
7303 :
7304 :
7305 :
7306 :

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

7355 ;
7356 ;Local Equates
7357 ;
<000A> 7358 LF EQU OAH
<000C> 7359 FF EQU OCH
<0008> 7360 BS EQU O8H
<000D> 7361 CR EQU ODH
<0016> 7362 CLR_TO_EOL EQU 16H
<0018> 7363 CLR_TO_EOS EQU 18H
<001C> 7364 GOTO_XY_CHAR EQU 1CH
<0080> 7365 HOME EQU 80H
<00A0> 7366 UP EQU OAOH
<00A1> 7367 RIGHT EQU OA1H
<00A2> 7368 DOWN EQU OA2H
<00A3> 7369 LEFT EQU OA3H
<005F> 7370 CURSOR_CHAR EQU "-"
<0020> 7371 BLANKING_CHAR EQU "-"
<FEA6> 7372 CURSOR_X EQU CURSOR+1
7373

```

;Char to emit as cursor
;Char used when scrolling to fill screen

```

LOCATION OBJECT CODE LINE SOURCE LINE
7375 : :
7376 : :__CONS_INIT
7377 : :
7378 : : Sets up the window sizes for the CONSOLE_OUT routine
7379 : :
7380 : : Input Parameters:
7381 : : B - number of columns in the window - 1, 0..31
7382 : : C - number of lines in the window - 1, 0..23
7383 : : D - Absolute (0 based) X coord of upper left corner of window, 0..31
7384 : : E - Absolute (0 based) Y coord of upper left corner of window, 0..23
7385 : : HL - Pattern Name Table (PNT) starting address
7386 : :
7387 : : NOTE- User must initialize the VDP
7388 : :
7389 : : GLB __CONS_INIT
7390 : :
7391 : : INC B : Adjust for old calling convention
7392 : : INC C
7393 : : LD [NUM_LINES],BC : Set num lines, cols in window
7394 : : LD [UPPER_LEFT],DE : Set upper left corner of window
7395 : : LD [CURSOR],DE
7396 : : LD [PTRN_NAME_TBL],HL : Save address of Pattern name table
7397 : :
7398 : : LD A,D : Calc min/max coords of window
7399 : : LD [X_MIN],A
7400 : : ADD A,B
7401 : : DEC A
7402 : : LD [X_MAX],A
7403 : : LD A,E
7404 : : LD [Y_MIN],A
7405 : : ADD A,C
7406 : : DEC A
7407 : : LD [Y_MAX],A
7408 : :
7409 : : LD A,BLANKING_CHAR
7410 : : LD [OLDCHAR_]_A
7411 : : LD A,CURSOR_CHAR
7412 : : CALL PUT_CHAR_ON_SCREEN
7413 : :
7414 : : RET
F5DC 04
F5DD 0C
F5DE ED43FE9F
F5E2 ED53FEA1
F5E6 ED53FEA5
F5EA 22FEA3
F5ED 7A
F5EE 32FE7A
F5F1 80
F5F2 3D
F5F3 32FE7B
F5F6 7B
F5F7 32FE7C
F5FA 81
F5FB 3D
F5FC 32FE7D
F5FF 3E20
F601 32FE79
F604 3E5F
F606 CDF7DA
F609 C9

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

7416 :
7417 :   __CONS_OUT - This entry point will check special chars such as
7418 :   Carriage Return and performs their normal functions
7419 :   all other, printable, chars are put onto the screen
7420 :
7421 :   __CONS_DISP - This entry point will put all chars onto the screen
7422 :   no matter what, no checking is done for control chars
7423 :
7424 :
7425 : Input Parameters:
7426 :   A - character to emit
7427 :
7428 :   optionally
7429 :   E - Y location to goto (must be in window)
7430 :   D - X location to goto (must be in window)
7431 :
7432 :
7433 : Saves all registers
7434 :
7435 :   :GLB      __CONS_OUT
7436 :   :GLB      __CONS_DISP
7437 :
7438 :   __CONS_OUT:
7439 :   PUSH AF
7440 :   PUSH BC
7441 :   PUSH HL
7442 :   PUSH IX
7443 :   PUSH IY
7444 :   PUSH DE
7445 :
7446 :   LD HL,SPECIAL_CHARS
7447 :   LD BC,NUM_SPC_CHARS
7448 :   CPIR
7449 :   JR NZ,NOT_SPECIAL
7450 :
7451 :   LD HL,SPCL_VECTOR_TBL
7452 :   ADD HL,BC
7453 :   ADD HL,BC
7454 :
7455 :   LD B,A ;Save char in B
7456 :
7457 :   LD A,[HL]
7458 :   INC HL
7459 :   LD H,[HL]
7460 :   LD L,A
7461 :   JP [HL] ;vector to special char handler

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
F627 F5 7462 ___CONS_DISP:
F627 AF 7463
F628 C5 7464
F629 E5 7465
F62A DDE5 7466
F62C FDE5 7467
F62E D5 7468
F62F 7469 NOT_SPECIAL:
F62F CDF7DA 7470
F632 2AFEA5 7471
F635 24 7472
F636 3AFE7B 7473
F639 BC 7474
F63A 3014 7475
F63C 3AFE7A 7476
F63F 67 7477
F640 2C 7478
F641 3AFE7D 7479
F644 8D 7480
F645 3009 7481
F647 2D 7482
F648 E5 7483
F649 CDF7ED 7484
F64C CDF794 7485
F64F E1 7486
F650 22FEA5 7487
F653 CDF7FO 7488 SAVE_CURSOR:
F656 7489 NEW_CURSOR:
F656 3E5F 7490 PUT_CURSOR_ON:
F658 CDF7DA 7492
F658 D1 7493
F65C FDE1 7494 EXIT:
F65E DDE1 7495
F660 E1 7496
F661 C1 7497
F662 F1 7498
F663 C9 7499
7500

PUSH AF
PUSH BC
PUSH HL
PUSH IX
PUSH IY
PUSH DE
CALL
LD
INC
LD
CP
JR
LD
LD
INC
LD
CP
JR
DEC
PUSH
CALL
CALL
POP
LD
CALL
POP
POP
POP
POP
POP
POP
RET

PUT_CHAR_ON_SCREEN ; Display char where cursor is at
HL,[CURSOR]
H ; Move cursor to right (+1)
; but check for going out of
A,[X_MAX] ; the window
NC,SAVE_CURSOR ; If taken then stayed in window
A,[X_MIN] ; Went out of window so set X
H,A ; to left edge
L ; and move down one line
A,[Y_MAX] ; but check to see if still
L ; in the window
NC,SAVE_CURSOR ; Put back into window
L
HL
GET_CHAR_UNDER_CURSOR
SCROLL_UP
HL
[CURSOR],HL
GET_CHAR_UNDER
LD
CALL
LD
CALL
POP
POP
POP
POP
POP
POP
RET
    
```

LOCATION OBJECT CODE LINE SOURCE LINE

```

7502 :
7503 :Special Character Handlers
7504 :
7505 :
7506 : RETURN_WITH_ERASE - moves cursor to right edge of window on current line
7507 : after clearing to end of line
7508 :
7509 :RETURN_WITH_ERASE:
7510 : CALL ERASE_THIS_LINE
7511 RETURN: CALL PUT_OLD_ON_SCREEN
7512 LD A,[X_MIN]
7513 LD [CURSOR X],A
7514 LD HL,[CURSOR]
7515 JP NEW_CURSOR

```

```

F664 CDF707
F667 3AFE7A
F66A 32FEA6
F66D 2AFEA5
F670 C3F653

```

:Display char cursor was hiding

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		7517	:	
		7518	:	MOVE_UP - Moves the cursor up 1f not at top of window
		7519	:	
		7520	:	MOVE_UP:
F673	2AFEAS	7521	LD	HL,[CURSOR]
F676	3AFE7C	7522	LD	A,[Y_MIN]
F679	BD	7523	CP	L
F67A	28DF	7524	JR	Z.EXIT ;Forget request
F67C	2D	7525	DEC	L ;Move up
		7526	:	
F67D	CD7D7	7527	CALL	PUT_OLD_ON_SCREEN ;Display char cursor was hiding
F680	C3F650	7528	JP	SAVE_CURSOR ;Display cursor in new spot

HEWLETT-PACKARD: CONS_OUT (c) Coleco 1... Confidential

LOCATION OBJECT CODE LINE SOURCE LINE

7530 ;
 7531 ; MOVE_DOWN - Moves the cursor down if will stay in window, except
 7532 ; if used by linefeed when the screen is scrolled
 7533 ;

F683 7534 MOVE_DOWN:

F683 2AFEAS HL,[CURSOR]
 F686 3AFE7D LD LD A,[Y_MAX] ;See if at bottom of window
 F689 BD L L
 F68A 2904 JR Z.SCROLL_POSSIBLE ;Move down
 F68C 2C INC L RESTORE_OLD
 F68D C3F67D JP

F690 3E0A LD A,LF ;See if linefeed, and if it is
 F692 B8 CP B ; then scroll up, else leave
 F693 20C6 JR NZ.EXIT ; cursor and screen as is
 F695 CDF794 CALL SCROLL_UP
 F698 2AFEAS LD HL,[CURSOR]
 F69B C3F653 JP NEW_CURSOR

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		7550		
		7551	MOVE_RIGHT	- Moves the cursor to the right, and if at window's edge will move down a line to the left edge, unless at the lower right corner of the window in which case nothing is down
		7552		
		7553		
		7554		
		7555		
F69E		7556	MOVE_RIGHT:	
F69E 2AFEAS	LD		HL,(CURSOR)	
F6A1 3AFE7B	LD		A,[X_MAX]	
F6A4 BC	CP		H	
F6A5 2804	JR		Z,TRY_MOVE_DOWN	
F6A7 24	INC		H	: Move to right
F6A8 C3F67D	JP		RESTORE_OLD	
		7563		
F6AB 3AFE7D	LD		A,[Y_MAX]	
F6AE BD	CP		L	
F6AF 28AA	JR		Z,EXIT	: At bottom of window so abort
F6B1 2C	INC		L	
F6B2 3AFE7A	LD		A,[X_MIN]	
F6B5 67	LD		H,A	
F6B6 C3F67D	JP		RESTORE_OLD	

LOCATION OBJECT CODE LINE SOURCE LINE

```

7572 :
7573 : MOVE_LEFT - Moves the cursor to the left and if at left edge of
7574 : window will move up a line an to right edge if not
7575 : at upper left corner of window
7576 :
7577 MOVE_LEFT: LD HL,[CURSOR]
7578 LD A,[X_MIN]
7579 CP H
7580 JR Z,TRY_SCROLL_UP
7581 DEC H
7582 JP RESTORE_OLD ;Move to left
7583 :
7584 TRY_SCROLL_UP: LD A,[Y_MIN]
7585 CP L
7586 JR Z,EXIT ;Abort if at upper left corner
7587 DEC L
7588 LD A,[X_MAX]
7589 LD H,A
7590 JP RESTORE_OLD

```

```

F6B9 2AFEAS
F6BC 3AFE7A
F6BF BC
F6C0 2804
F6C2 25
F6C3 C3F67D
F6C6 3AFE7C
F6C9 8D
F6CA 288F
F6CC 2D
F6CD 3AFE78
F6D0 67
F6D1 C3F67D

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

7592 ; MOVE_HOME - Moves cursor to the upper left corner of the window
7593 ;
7594 ;
7595 ;
7596 ; FORM_FEED - Clears the screen and puts the cursor at the upper left
7597 ; corner of the window
7598 ; FORM_FEED:
7599 ; LD A,[NUM_LINES]
7600 ; LD B,A
7601 ; LD HL,[UPPER_LEFT]
7602 ;
7603 ; CALL BLANK_SCREEN
7604 ;
7605 ; LD A,BLANKING_CHAR
7606 ; LD [OLDCHAR],A
7607 ;
7608 ; MOVE_HOME:
7609 ; CALL PUT_OLD_ON_SCREEN ;Needed only for MOVE_HOME not FORM_FEED
7610 ; LD HL,[UPPER_LEFT]
7611 ; JP SAVE_CURSOR
    
```

```

F6D4
F6D4 3AFE9F
F6D7 47
F6D8 2AFE A1
F6D8 CDF74C
F6DE 3E20
F6E0 32FE79
F6E3 CDF7D7
F6E6 2AFE A1
F6E9 C3F650
    
```

LOCATION OBJECT CODE LINE SOURCE LINE

7612 :				
7613 :	ERASE_TO_EOL	-	From cursor position fills rest of line	
7614 :			with BLANKING_CHAR. to end of window only	
7615 :			Will blank char under cursor always.	
7616 :				
7617 :	ERASE_TO_EOL:			
F6EC			CALL	ERASE_THIS_LINE
F6EC CDF766			JP	PUT_CURSOR_ON
F6EF				
F6EF C3F656				

LOCATION OBJECT CODE LINE SOURCE LINE

```

7621 ;
7622 ; ERASE_TO_EOS - Will erase the screen (window) from the current
7623 ; cursor position to the end of the window. The
7624 ; cursor position is also blanked.
7625 ;
7626 ;
7627 ERASE_TO_EOS:
7628 F6F2 CDF766 CALL ERASE_THIS_LINE ;Erase to end of current line
7629 F6F5 2AFEA5 HL.[CURSOR] HL.[CURSOR]
7630 F6F8 2C INC L ;Move down
7631 F6F9 3AFE7C LD A.[Y_MIN]
7632 F6FC 4F C,A
7633 F6FD 3AFE9F LD A.[NUM_LINES]
7634 F700 81 ADD A,C
7635 F701 95 SUB L ;Number of lines left to blank
7636 F702 CAF656 JP Z,PUT_CURSOR_ON
7637 F705 47 LD B,A
7638 F706 3AFE7A LD A,[X_MIN]
7639 F709 67 LD H,A
7640 F70A CDF74C CALL BLANK_SCREEN
7641 F70D C3F656 JP PUT_CURSOR_ON

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

7643 ;
7644 ; GOTO_XY - Puts cursor to desired X-Y coord if in window
7645 ;
7646 ; Input Parameters:
7647 ; D - X location
7648 ; E - Y location
7649 ;
7650 GOTO_XY:
7651 LD A,[X_MIN]
7652 CP D
7653 JR Z,CHK_X_WITH_MAX
7654 JP NC,EXIT
7655 LD A,[X_MAX]
7656 CP D
7657 JR Z,CHK_Y_COORD
7658 JP C,EXIT
7659 LD A,[Y_MIN]
7660 CP E
7661 JR Z,CHK_Y_WITH_MAX
7662 JP NC,EXIT
7663 LD A,[Y_MAX]
7664 CP E
7665 JR Z,SET_CURSOR
7666 JP C,EXIT
7667 EX DE,HL
7668 JP RESTORE_OLD

```

LOCATION OBJECT CODE LINE SOURCE LINE

7670 :
7671 : Internal Subroutines
7672 :
7673 :
7674 : Subroutine
7675 : BLANK_PORTION - Fills LINEBUFFER_ with number of chars in B
7676 : from where HL points
7677 :
7678 : Input Parameters:
7679 : HL - Pointer into LINEBUFFER_
7680 : B - Number of the BLANKING_CHAR's to stuff in
7681 :
7682 : A is destroyed
7683 :
7684 BLANK_PORTION:
7685 PUSH HL
7686 PUSH BC
7687 JR BLANK_REST

F738
F738 E5
F739 C5
F73A 1807

LOCATION OBJECT CODE LINE SOURCE LINE

```

7689 ;
7690 ; Subroutine
7691 ; BLANKS - Fills LINEBUFFER_ with 32 BLANKING_CHAR's
7692 ;
7693 ; A is destroyed
7694 ;
7695 BLANKS:
7696
7697
7698
7699
7700 BLANK_REST:
7701 BLANK_FILL:
7702
7703
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713
7714
7715
7716
7717
7718
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736
7737
7738
7739
7740
7741
7742
7743
7744
7745
7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757
7758
7759
7760
7761
7762
7763
7764
7765
7766
7767
7768
7769
7770
7771
7772
7773
7774
7775
7776
7777
7778
7779
7780
7781
7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793
7794
7795
7796
7797
7798
7799
7800
7801
7802
7803
7804
7805
7806
7807
7808
7809
7810
7811
7812
7813
7814
7815
7816
7817
7818
7819
7820
7821
7822
7823
7824
7825
7826
7827
7828
7829
7830
7831
7832
7833
7834
7835
7836
7837
7838
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7850
7851
7852
7853
7854
7855
7856
7857
7858
7859
7860
7861
7862
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7890
7891
7892
7893
7894
7895
7896
7897
7898
7899
7900
7901
7902
7903
7904
7905
7906
7907
7908
7909
7910
7911
7912
7913
7914
7915
7916
7917
7918
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7930
7931
7932
7933
7934
7935
7936
7937
7938
7939
7940
7941
7942
7943
7944
7945
7946
7947
7948
7949
7950
7951
7952
7953
7954
7955
7956
7957
7958
7959
7960
7961
7962
7963
7964
7965
7966
7967
7968
7969
7970
7971
7972
7973
7974
7975
7976
7977
7978
7979
7980
7981
7982
7983
7984
7985
7986
7987
7988
7989
7990
7991
7992
7993
7994
7995
7996
7997
7998
7999
8000

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		7708	:
		7709	: Subroutine
		7710	: BLANK_SCREEN - Will blank a given number of lines on the screen
		7711	:
		7712	: Input Parameters:
		7713	: B - Number of lines to blank
		7714	: HL - Absolute X-Y coords of start position of 1st line to blank
		7715	:
		7716	: Destroys C, A, DE
		7717	: User must reset cursor
		7718	:
F74C		7719	BLANK_SCREEN:
F74C	CD73C	7720	CALL BLANKS
F74F	3AFEAO	7721	LD A,[NUM_COLUMNS]
F752	4F	7722	LD C,A
		7723	:
F753	C5	7724	BLNK_SCRN_LOOP:
F754	E5	7725	PUSH BC
F755	CD7FC	7726	PUSH HL
F758	0600	7727	CALL CONVERT_XY
F75A	21FE7E	7728	LD B,00H
F75D	CD7D1A	7729	LD HL,LINEBUFFER_
F760	E1	7730	CALL WRITE_VRAM
F761	2C	7731	POP HL
F762	C1	7732	INC L
F763	10EE	7733	POP BC
F765	C9	7734	DJNZ BLNK_SCRN_LOOP
			RET
			:Fill buffer with BLANKING_CHAR
			:Save X-Y coord of line
			:C has number of bytes to read
			:To next line below

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		7736	:
		7737	: Subroutine
		7738	: ERASE_THIS_LINE - This routine will erase the current line
		7739	: from the cursor position to the right edge
		7740	: of the window, the char under the cursor is
		7741	: also blanked.
		7742	:
		7743	: Input Parameters:
		7744	: None
		7745	:
		7746	: Destroys all registers
		7747	: Sets OLDCHAR_ to BLANKING_CHAR
		7748	:
F766		7749	: ERASE_THIS_LINE:
F766	0DF779	7750	: CALL READ_FROM_CURSOR_TO_EOL
F769	E5	7751	: HL
F76A	C5	7752	: PUSH BC
F76B	41	7753	: LD B,C
F76C	0DF738	7754	: CALL BLANK_PORTION
F76F	C1	7755	: POP BC
F770	E1	7756	: POP HL
		7757	:
F771	3E20	7758	: LD A,BLANKING_CHAR
F773	32FE79	7759	: LD [OLDCHAR_],A
		7760	:
F776	C3FD1A	7761	: JP WRITE_VRAM ;Write line with blanks, return thru WRITE_VRAM

LOCATION OBJECT CODE LINE SOURCE LINE

```

7763 :
7764 : Subroutine
7765 : READ_FROM_CURSOR_TO_EDL - reads the line on screen from the
7766 : cursor position to the right edge of the
7767 : screen
7768 :
7769 : Input Parameters:
7770 : None
7771 :
7772 : Returns:
7773 : BC - number of chars read
7774 : HL - address of LINEBUFFER_
7775 : DE - VRAM address of place cursor is
7776 :
7777 READ_FROM_CURSOR_TO_EDL:
7778 LD HL,[CURSOR]
7779 LD A,[X_MAX]
7780 INC A
7781 SUB H
7782 LD C,A
7783 CALL CONVERT_XY
7784 LD HL,LINEBUFFER_
7785 LD B,00H
7786 PUSH BC
7787 PUSH HL
7788 PUSH DE
7789 CALL READ_VRAM
7790 POP DE
7791 POP HL
7792 POP BC
7793 RET

```

:Calc # chars to read

LOCATION OBJECT CODE LINE SOURCE LINE

```

7795 ; Subroutine
7796 ; SCROLL_UP - Scrolls the entire window up 1 line and puts a line
7797 ; of BLANKING_CHAR's on the last line
7798 ;
7799 ;
7800 ; Caller must restore cursor to screen
7801 ;
7802 SCROLL_UP:
7803     CALL PUT_OLD_ON_SCREEN
7804     LD HL,[UPPER_LEFT]
7805     HL
7806     CALL CONVERT_XY
7807     HL
7808     LD A,[NUM_LINES]
7809     LD B,A
7810     DEC B
7811
7812 SCROLL_LOOP:
7813     PUSH BC
7814     PUSH DE
7815     INC L
7816     PUSH HL
7817     CALL CONVERT_XY
7818     DE
7819     LD A,[NUM_COLUMNS]
7820     LD C,A
7821     LD B,OOH
7822     HL,LINEBUFFER_
7823     READ_VRAM
7824     BC
7825     POP DE
7826     POP HL
7827     EX [SP],HL
7828     DE,HL
7829     PUSH HL
7830     LD HL,LINEBUFFER_
7831     WRITE_VRAM
7832     DE
7833     POP HL
7834     POP BC
7835     DJNZ SCROLL_LOOP
7836
7837     CALL BLANKS
7838     LD HL,LINEBUFFER_
7839     LD A,[NUM_COLUMNS]
7840     LD C,A
7841     JP WRITE_VRAM

```

; Save number of remaining lines
; Save next write location
; Point to next line below
; Save address of this read
; Save number of chars in line
; Get number of bytes that were read
; Get address of last read
; Get X-Y coords
; Swap prev read address with X-Y coords
; Swap last read with previous address
; and save previous
; Write line one line up
; Get address of next write
; Get starting X-Y of last read
; B is 0 from DJNZ
; Return thru WRITE_VRAM

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		7843	:
		7844	: Subroutines
		7845	: PUT_OLD_ON_SCREEN - This entry point puts the char that the
		7846	cursor was on back onto the screen
		7847	:
		7848	: PUT_CHAR_ON_SCREEN - This entry point will put the char in A
		7849	onto the screen where the cursor currently is
		7850	:
		7851	:
F7D7		7852	PUT_OLD_ON_SCREEN: LD
F7D7	3AFE79	7853	A.[OLDCHAR_]
		7854	:
F7DA		7855	PUT_CHAR_ON_SCREEN: PUSH
F7DA	E5	7856	LD
F7DB	2AFEA5	7857	LD
F7DE	GDF7FC	7858	CALL
F7E1	21FE7E	7859	LD
F7E4	77	7860	LD
F7E5	010001	7861	LD
F7E8	GDFD1A	7862	CALL
F7EC	E1	7863	POP
F7EC	C9	7864	RET

```

HL,[CURSOR]           ;Get location to put char
CONVERT_XY           ;Convert to a VRAM address
HL,LINEBUFFER_
[HL],A
BC,0001H
WRITE_VRAM
HL

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		7866	:	Subroutine
		7867	:	GET_CHAR_UNDER_CURSOR - This entry loads HL from CURSOR before
		7868	:	falling into GET_CHAR_UNDER
		7869	:	
		7870	:	
		7871	:	GET_CHAR_UNDER - Reads the char where the cursor will go
		7872	:	and saves it in the variable OLDCHAR_
		7873	:	
		7874	:	Input Parameters:
		7875	:	HL - Cursor position
		7876	:	
		7877	:	Destroys - HL, BC, DE, A
		7878	:	
F7ED		7879	:	GET_CHAR_UNDER_CURSOR: HL,[CURSOR]
F7ED	2AFEAS	7880	:	LD
		7881	:	
F7F0		7882	:	GET_CHAR_UNDER: CALL CONVERT_XY
F7F0	CDF7FC	7883	:	LD BC,0001H
F7F3	010001	7884	:	LD HL,OLDCHAR_
F7F6	21FE79	7885	:	LD READ_VRAM_
F7F9	C3FD1D	7886	:	JP ;Return thru READ_VRAM_

LOCATION OBJECT CODE LINE SOURCE LINE

7920 :
 7921 : This is a table of chars that perform some function rather
 7922 : than being printed out to the screen as is. There is a
 7923 : jump table that is associated with this table, but NOTE that
 7924 : it has its entries in reverse order of this table. It would
 7925 : be best to add to the end of this table and to the beginning
 7926 : of the jump table.
 7927 :

F80E	08	SPECIAL_CHARS:	BS	
F80E	09		DEFB	
F80F	0D		DEFB	CR
F810	0A		DEFB	LF
F811	0C		DEFB	FF
F812	80		DEFB	HOME
F813	16		DEFB	CLR_TO_EOL
F814	18		DEFB	CLR_TO_EOS
F815	1C		DEFB	GOTO_XY_CHAR
F816	A0		DEFB	UP
F817	A2		DEFB	DOWN
F818	A3		DEFB	LEFT
F819	A1		DEFB	RIGHT
				\$-SPECIAL_CHARS
<000C>		7942	NUM_SPL_CHARS	EQU
		7943		

```

LOCATION OBJECT CODE LINE SOURCE LINE
7945 ;
7946 ;NOTE - This table is in the reverse order of the chars so
7947 ; any additions to the end of the chars goes at the top
7948 ; of this table
7949 ;
7950 SPCL_VECTOR_TBL:
7951 F81A F69E MOVE_RIGHT DEFW
7952 F81C F689 MOVE_LEFT DEFW
7953 F81E F683 MOVE_DOWN DEFW
7954 F820 F673 MOVE_UP DEFW
7955 F822 F710 GOTO_XY DEFW
7956 F824 F6F2 ERASE_TO_EOS DEFW
7957 F826 F6EC ERASE_TO_EOL DEFW
7958 F828 F6E3 MOVE_HOME DEFW
7959 F82A F6D4 FORM_FEED DEFW
7960 F82C F683 MOVE_DOWN DEFW
7961 F82E F664 RETURN DEFW
7962 F830 F689 MOVE_LEFT DEFW
7963
7964 ;
7965 ; CODE FROM HERE TO END OF CONS_OUT WAS INSERTED AFTER THE
7966 ; REV. 06 ROM WAS BURNED AND HAS BEEN COMMENTED OUT TO MAINTAIN
7967 ; COMPATIBILITY WITH THAT ROM.
7968 ;
7969 ; END
7970 ; SKIP
7971 ;
7972 ; RETURN_CURSOR - Will return the current cursor position
7973 ; to the caller. It is in absolute screen
7974 ; coordinates (0 based).
7975 ;
7976 ; Input Parameters:
7977 ; None
7978 ;
7979 ; Returns:
7980 ; D - X coord of cursor, absolute based on upper left of screen not window
7981 ; E - Y coord of cursor
7982 ;
7983 ;:GLB RETURN_CURSOR
7984 ;RETURN_CURSOR: LD DE,[CURSOR]
7985 ; RET
7986 ;
7987 ; SKIP
7988 ;
7989 ; RETURN_RELATIVE_CURSOR - Returns the current cursor position relative
7990 ; to the window. 0,0 will be returned for the
7991 ; upper left corner of the window.
7992 ;
7993 ; Input Parameters:
7994 ; NONE
7995 ;
7996 ; Returns:
7997 ; D - X cursor position relative to window
7998 ; E - Y cursor position relative to window
7999 ;
8000 ; Destroys A
8001 ;

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

8002 : : :GLB RETURN_RELATIVE_CURSOR
8003 : :
8004 : :
8005 : RETURN_RELATIVE_CURSOR:
8006 : LD DE,[CURSOR] :get absolute position
8007 : LD A,[X_MIN]
8008 : NEG
8009 : ADD
8010 : LD D,A
8011 : LD A,[Y_MIN]
8012 : NEG
8013 : ADD
8014 : LD E,A
8015 : RET
8016 : :
8017 : :
8018 : :
8019 : :
8020 : :
8021 : :

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

8023 NAME ^Rev 07 - rfj^
8024
8025 De_EDS_START MACRO :Header Rev. 4
8026 :GOTO Ede_EDS_START
8027
8028 Project: EDS, 83-101
8029
8030 |&k1S
8031 |&k1S
8032 |&k1S EDS_START DLS
8033 |&k1S
8034 |&k1S
8035
8036
8037 Rev History
8038 Rev. Date Name
8039 7 24oct1951 rfj
8040 6 13oct1045a RPD
8041 5 1oct1005a RPD
8042 4 25sept1000a RPD
8043 3 24sept1110a RPD
8044 2 23sept1255p RPD
8045 1 03SEP1445 DLS
8046 0 02SEP0630 DLS
8047 Ede_EDS_START MEND

```

Change
 Changed value for REV_NUM to 06
 added init of REV_NUM
 took out memory bank switch logic for GOTO_WP
 reworked for new boot from tape logic
 moved clear ram logic
 moved FCB's and added calls A_u0S routines
 MORE CODE
 CODE

|&kOS
 |&kOS
 |&kOS
 |&kOS
 |&kOS

LOCATION OBJECT CODE LINE SOURCE LINE

```

8049 * .....
8050 * .....
8051 * MODULE NAME:
8052 * .....
8053 * EOS_START
8054 * .....
8055 * INPUTS:
8056 * .....
8057 * NONE
8058 * .....
8059 * FUNCTION(S):
8060 * .....
8061 * 0. SET STACK.
8062 * 1. TO BANK SELECT OS7 AND 24K RAM.
8063 * 2. CALL INIT_EOS TO ALLOW THE 1) THE Z80/6801 SYNCH_UP.
8064 * POLL FOR NET DEVICES AND ESTABLISH DCB'S.
8065 * 3. CHECK FOR THE PRESENCE OF A TAPE. IF PRESENT THEN
8066 * LOAD IN A COLD_START_LOADER AND EXECUTE FROM THERE.
8067 * 4. IF NO TAPE, THEN BRING UP THE ROM RESIDENT WORD
8068 * PROCESSOR BY JUMPING TO OS7.
8069 * .....
8070 * OUTPUTS:
8071 * .....
8072 * DEVICE ADDR _ REG B
8073 * .....
8074 * CALLS:
8075 * .....
8076 * EOS_INIT
8077 * EOS_QUERY_DEVICE
8078 * EOS_READ_BLOCK
8079 * .....
8080 * CALLED BY:
8081 * .....
8082 * INVOKED BY EOS_800T
8083 * .....
8084 * NOTES:
8085 * .....
8086 * NONE.
8087 * .....

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

8090 :
8091 : external routines used
8092 :
8093 :EXT FIND_DCB
8094 :EXT REQUEST_STATUS
8095 :EXT HARD_INIT
8096 :EXT RD_1_BLOCK
8097 :EXT FMGR_INIT
8098 :EXT GOTO_WP
8099 :EXT FILL_VRAM
8100 :EXT TURN_OFF_SOUND
8101 :EXT SWITCH_MEM
8102 :EXT PORT_COLLECTION
8103 :
8104 : external data structures used
8105 :
8106 :EXT REV_NUM
8107 :EXT CLEAR_RAM_START
8108 :EXT CLEAR_RAM_SIZE
8109 :EXT EOS_STACK
8110 :EXT DEFAULT_BT_DEV
8111 :EXT MEM_CNFG01
8112 :EXT MEM_CNFG03
8113 :
8114 : local equates
8115 :
8116 EOS_REV EQU 005H
8117
8118
8119
8120
8121
8122 COLD_START_ADDR EQU OC800H
8123 DSK4 EQU 4
8124 DSK5 EQU 5
8125 ONE_BLOCK EQU 1
8126 BLOCK_ZERO EQU 0
8127

```

```

<0005>
:current EOS revision number
:---NOTE--- this is actually rev.
:06 but to match the production ROM
:which was labeled rev. 05 we fudge
:just a bit (two bits actually)

```

```

<C800>
<0004>
<0005>
<0001>
<0000>

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
8129 :GLB __EOS_START
F832 8130 __EOS_START: LD SP,EOS_STACK ; SET THE STACK
F832 8131 LD SP,EOS_STACK ;
F832 8132 LD BC,CLEAR_RAM_SIZE ; clear EOS ram
F835 010147 DE,CLEAR_RAM_START+1
F838 11FD61 HL,CLEAR_RAM_START
F83B 21FD60 XOR A
F83E AF [HL],A
F83F 77 LD LDIR
F840 E080 LD A,EOS_REV ; init REV_NUM
F842 3E05 [REV_NUM],A ;
F844 32FD60 CALL PORT_COLLECTION ; collect the port addresses from OS7
F847 CDFD11 CALL TURN_OFF_SOUND ; KILL SOUNDS
F84A CDFD53 LD A,O ; CLEAR VRAM
F84D 3E00 LD HL,O
F84F 210000 LD DE,16*1024
F852 114000 CALL FILL_VRAM
F855 CDFD26 LD A,[MEM_CNFG01] ; get the data for configuration 1 (64K RAM)
F858 3AFC18 CALL SWITCH_MEM ; select this configuration
F85B CDFD14 ;
F85E CDF8F6 CALL __HARD_INIT ;
F859 ;
F860 ; setup the FCB's
F861 ;
F861 11D400 LD DE,THREE1K_BLKS ; START ADDRESS OF THREE 1K BLOCKS
F864 21D390 LD HL,FCB_S ; START ADDRESS OF 3 FCB BLOCKS
F867 CDEEEA CALL __FMGR_INIT ; TELL THE FILE MANAGER

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		8166	;
		8167	;
		8168	;
F86A	3E08	8169	LD A,TAPE_ID
F86C	32FD6F	8170	LD [DEFAULT_BT_DEV],A ;set the default boot device to tape
F86F	3E04	8171	LD A,DSK4
F871	GD473	8172	CALL REQUEST_STATUS ;check for DISK 4 device active
F874	2010	8173	JR NZ,CHK_D5
		8174	
		8175	
F876	3E04	8176	LD A,DSK4
F878	GD446	8177	CALL FIND_DCB
F87B	FD7E14	8178	LD A,[IY+D_STATUS_FLAGS]
F87E	E60F	8179	AND OFH
F880	FE03	8180	CP 3
F882	3E04	8181	LD A,DSK4
F884	3842	8182	JR C,DEV1_OK
		8183	
F886	3E05	8184	LD A,DSK5 ;check for DISK 5 device active
F888	GD473	8185	CALL REQUEST_STATUS
F88B	2010	8186	JR NZ,CHK_TAPE
		8187	
F88D	3E05	8188	LD A,DSK5
F88F	GD446	8189	CALL FIND_DCB
F892	FD7E14	8190	LD A,[IY+D_STATUS_FLAGS]
F895	E60F	8191	AND OFH
F897	FE03	8192	CP 3
F899	3E05	8193	LD A,DSK5
F89B	3828	8194	JR C,DEV1_OK
F89D	3E08	8195	LD A,TAPE_ID ;check if TAPE device active
F89F	GD473	8196	CALL REQUEST_STATUS
F8A2	201E	8197	JR NZ,CALL_WP
		8198	
		8199	
F8A4	3E08	8200	LD A,TAPE_ID
F8A6	GD446	8201	CALL FIND_DCB
F8A9	FD7E14	8202	LD A,[IY+D_STATUS_FLAGS]
F8AC	F5	8203	PUSH AF
F8AD	E60F	8204	AND OFH
F8AF	FE03	8205	CP 3
F8B1	3812	8206	JR C,DEVO_OK
		8207	
F8B3	F1	8208	POP AF
F8B4	CB3F	8209	SRL A
F8B6	CB3F	8210	SRL A
F8B8	CB3F	8211	SRL A
F8BA	CB3F	8212	SRL A
F8BC	FE03	8213	CP 3
F8BE	3E18	8214	LD A,TAPE_ID+10H
F8C0	3806	8215	JR C,DEV1_OK

```

LOCATION OBJECT CODE LINE SOURCE LINE
F8C2 8217 CALL_WP:
8218 ; A.[MEM CNFG03]
8219 ; CALL
8220 ; JP
F8C5 8221 DEVO_OK:
8222 ; POP AF
8223 ; LD A,TAPE_ID
8224 ; [DEFAULT_BT_DEV],A
8225 ; LD HL,COLD_START_ADDR
8226 ; LD BC,0
8227 ; LD A,[DEFAULT_BT_DEV]
8228 ; DE,BLOCK_ZERO
8229 ; CALL RD_1_BLOCK
8230 ; Z,GO_TO_TAPECODE
8231 ; C,A
8232 ; A,[DEFAULT_BT_DEV]
8233 ; AND OFH
8234 ; LD CP,TAPE_ID
8235 ; JR NZ,READ_AGAIN
8236 ; LD A,C
8237 ; TIMEOUT
8238 ; JR Z,READ_AGAIN
8239 ; CP CALL_WP
8240 ; JP
8241 ; GO_TO_TAPECODE:
8242 ; LD A,[DEFAULT_BT_DEV]
8243 ; LD B,A
8244 ; LD C,COLD_START_ADDR
8245 ; JP
8246 ;
8247 ;
8248 ;
8249 ;
8250 ;
8251 ;
8252 ;

```

;devices disk 4, 5 and tape are not active
;get the data for configuration 3 (8K 057 and 56K RAM)
;select this configuration
;default to ALPHA

LOCATION OBJECT CODE LINE SOURCE LINE

```

8254 :
8255 :
8256 :
8257 :
8258 :
8260 :
8261 :
8262 :
8263 :
8264 :
8265 :
8266 :
8267 :
8268 :
8269 :
8270 :
8271 :
8272 :
8273 :
8274 :
8275 :
8276 :
8277 :
8278 :
8279 :
8280 :
8281 :
8282 :
8283 :
8284 :
8285 :
8286 :
8287 :
8288 :
8289 :
8290 :
8291 :
8292 :
8293 :
8294 :
8295 :
8296 :
8297 :
8298 :
8299 :
8300 :
8301 :
8302 :
8303 :
8304 :
8305 :
8306 :
8307 :
8308 :
8309 :
8310 :

Rev History      Name      Change
Rev.  Date      RPD      WP equate changed to 100H
3      5oct1135a    RPD      added initialization of reserved byte at OFFFFH to 0
2      2oct155p    RPD      original
0      aug/sept    VB

      __HARD_INIT
ENTRY :      NOTHING
NEEDS :      DEFAULT PCB ADDRESS SET UP FOR USE ([CURRENT_PCB] = PCB)
RETURNS :     HARD RESET APPLIED TO NETWORK
      NETWORK MASTER NODE IN SYNC WITH Z80
      1 DCB ALLOTTED FOR EACH DEVICE ANSWERING A STATUS REQUEST
      CORRECTLY
      MASTER NODE SCANNING EACH OF THESE DCB'S FOR WORK TO DO

      :GLB      __HARD_INIT
      :EXT      CURRENT_PCB
      :EXT      PCB

      :GLB      __HARD_INIT
      :EXT      BC          : SAVE BC
      :EXT      DE          : DE
      :EXT      HL          : HL
      :EXT      IV          : IV

      __HARD_INIT
ENTRY :      LD HL,PCB          : INIT OUR IDEA OF CURRENT_PCB
NEEDS :      LD [CURRENT_PCB],HL
RETURNS :     __HARD_RESET_NET : HARD RESET NETWORK
      __DLY_AFT_HRD_RES : GOOD DELAY AFTER HARD
      :EXT      NET_RESET : NET RESET

      LD HL,PCB          : ZERO OUT PCB/DCB'S
      LD DE,PCB+1
      LD BC,P_SIZE+(15*D_SIZE)-1
      LD BC,P_SIZE+(15*D_SIZE) : this byte count will zero out PCB, DCB's and the reserved byte
      LD [HL],0
      LDIR

REPEAT_SYNC1:
CALL      __SYNC
JR        NZ,REPEAT_SYNC1

CALL      __SCAN_ACTIVE      FIND ALL THE ACTIVE DEVICES ON THE NET

POP      IV          : RESTORE IV
POP      HL          : HL
POP      DE          : DE
POP      BC          : BC

RET
    
```


LOCATION OBJECT CODE LINE SOURCE LINE

```

8312 : ENTRY:          __SOFT_INIT
8313 : NEEDS:          HL = PCB ADDRESS SET UP FOR USE
8314 : RETURNS:        [CURRENT_PCB] = HL
8315 :                HARD RESET APPLIED TO NETWORK
8316 :                NETWORK MASTER NODE IN SYNC WITH Z80
8317 :                1 DCB ALLOTTED FOR EACH DEVICE ANSWERING A STATUS REQUEST
8318 :                CORRECTLY
8319 :                MASTER NODE SCANNING EACH OF THESE DCB'S FOR WORK TO DO
8320 :
8321 :                :GLB          __SOFT_INIT
8322 :
8323 :
8324 :                __SOFT_INIT
8325 :                PUSH          BC          : SAVE BC
8326 :                PUSH          DE          : DE
8327 :                PUSH          HL          : HL
8328 :                PUSH          IY          : IY
8329 :
8330 :                LD            [CURRENT_PCB],HL  : ESTABLISH CURRENT_PCB ADDR
8331 :
8332 :                CALL         __HARD_RESET_NET  : HARD RESET NETWORK
8333 :
8334 :                CALL         __DLY_AFT_HRD_RES :
8335 :
8336 :                LD            HL,[CURRENT_PCB]  : ZERO OUT PCB/DCB'S
8337 :                LD            E,L
8338 :                LD            D,H
8339 :                INC           DE
8340 :                LD            BC,P_SIZE+(15*D_SIZE)-1
8341 :                LD            [HL],0
8342 :                LDIR
8343 :
8344 :                REPEAT_SYNC2:
8345 :                __SYNC
8346 :                JR            NZ,REPEAT_SYNC2
8347 :
8348 :                CALL         __SCAN_ACTIVE
8349 :
8350 :                POP          IY
8351 :                POP          HL
8352 :                POP          DE
8353 :                POP          BC
8354 :
8355 :                RET
8356 :

```

FIND ALL THE ACTIVE DEVICES ON THE NET

: RESTORE IY
: HL
: DE
: BC

LOCATION OBJECT CODE LINE SOURCE LINE

```

8358
8359 ; ENTRY:      HARD_RESET_NET
8360 ; NEEDS:      NOTHING
8361 ; RETURNS:   HARD RESET APPLIED TO NETWORK
8362
8363 NET_DELAY_COUNT EQU 0
8364
8365 ; GLB      __HARD_RESET_NET
8366
8367 __HARD_RESET_NET:
8368 ;EXT      NET_RESET_PORT
8369 LD      A,[NET_RESET_PORT]
8370 LD      C,A
8371 LD      A,OFH
8372 OUT
8373
8374 LD      A,NET_DELAY_COUNT
8375
8376 NET_RES_DELAY:
8377 NOP
8378 NOP
8379 NOP
8380 DEC
8381 JR
8382
8383 XOR      A
8384 OUT
8385
8386 RET
8387
<0000>
F94B
F94B 3AFC28
F94E 4F
F94F 3E0F
F951 ED79
F953 3E00
F955
F955 00
F956 00
F957 00
F958 3D
F959 20FA
F95B AF
F95C ED79
F95E C9

```

; RESET NETWORK

; DELAY (***** PROBABLY TOO LONG)

; HELLLLLL000000 NETWORK

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		8389		
		8390		
		8391	:GLB	__DLY_AFT_HRD_RES
		8392	__DLY_AFT_HRD_RES:	
F95F	C5	8393	PUSH	BC
F960	D5	8394	PUSH	DE
		8395		
F961	0601	8396	LD	B.1
		8397		
		8398	LOOP8:	
F963		8399	LD	DE.0001H
F963	110001	8400		
		8401	LOOP7:	
F966		8402	DEC	DE
F967	7A	8403	LD	A,D
F968	B3	8404	OR	E
F969	20FB	8405	JR	NZ,LOOP7
		8406		
F96B	10F6	8407	DJNZ	LOOP8
		8408		
F96D	D1	8409	POP	DE
F96E	C1	8410	POP	BC
		8411		
F96F	C9	8412	RET	
		8413		

```

LOCATION OBJECT CODE LINE SOURCE LINE
8415 :
8416 : ENTRY: SYNC
8417 : NEEDS: [CURRENT_PCB] SET TO ADDRESS OF CURRENTLY OPERATING PCB
8418 : RETURNS: NETWORK MASTER NODE IN SYNC WITH Z80
8419 : NUMBER OF DCB'S = 0 (SO MASTER NODE IS NOT SCANNING
8420 : ANY DCB'S - SO A SCAN_ACTIVE NEEDS TO BE DONE
8421 : TO ESTABLISH DCB'S)
8422 :
<0000> 8423 SYNC_DELAY_DE EQU 0
<0002> 8424 SYNC_DELAY_B EQU 2
8425 :GLB SYNC
8426 :
8427 :
8428 : SYNC
8429 : PUSH IY ; SAVE IY
8430 : PUSH HL ;
8431 : PUSH BC
8432 : PUSH DE
8433 :
8434 : LD IY,[CURRENT_PCB] ; CURRENT PCB
8435 :
8436 : LD [IY+P_NUM_DCBS],0 ; 0 DCB'S
8437 :
8438 : LD [IY+P_COM_STAT],PCB_SYNC1 ; START SYNC WITH FIRST
8439 : ; BYTE
8440 :
8441 : LD DE,SYNC_DELAY_DE
8442 : LD B,SYNC_DELAY_B
8443 :
8444 : LOOP1:
8445 : DEC DE
8446 : LD A,D
8447 : OR E
8448 : JR NZ,NOT_TIME1
8449 :
8450 : LD DE,SYNC_DELAY_DE
8451 :
8452 : DJNZ NOT_TIME1
8453 :
8454 : LD A,CANT_SYNC1
8455 : OR A ; ERROR
8456 :
8457 : JR END_SYNC
8458 :
8459 : NOT_TIME1:
8460 : LD A,[IY+P_COM_STAT] ; DID MASTER NODE ACKNOWLEDGE ?
8461 : CP PCB_SYNC1_ACK
8462 : JR NZ,LOOP1 ; ***** NEED PROTECTION HERE
8463 :
8464 : LD [IY+P_COM_STAT],PCB_SYNC2 ; CONTINUE SYNC WITH
8465 : ; SECOND BYTE
8466 :
8467 : LD DE,SYNC_DELAY_DE
8468 : LD B,SYNC_DELAY_B
8469 :
8470 : LOOP2:
8471 : DEC DE

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

F9A6 7A                    8472                    LD                    A,D
F9A7 B3                    8473                    OR                    E
F9A8 200A                  8474                    JR                    NZ,NOT_TIME2
                          8475                                           DE,SYNC_DELAY_DE
F9AA 110000                8476                    LD                    NOT_TIME2
                          8477                    DJNZ                  A,CANT_SYNC2
F9AD 1005                    8478                                           A
F9AF 3E13                  8480                    LD                    A
F9B1 B7                    8481                    OR                    A
                          8482                    JR                    END_SYNC
F9B2 1811                  8483                                           A
                          8484                                           A,[[Y+P_COM_STAT]
F9B4                        8485                    NOT_TIME2:            PCB_SYNC2_ACK
F9B4 FD7E00                8486                    LD                    NZ,LOOP2
F9B7 FE82                  8487                    CP                    IY
F9B9 20EA                  8488                    JR                    HL
                          8489                    PUSH                  [Y+P_REL_ADDR_LO],L
F9BB FDE5                  8490                    POP                    [Y+P_REL_ADDR_HI],H
F9BD E1                    8491                                           A
                          8492                                           DE
F9BE FD7501                8493                    LD                    BC
F9C1 FD7402                8494                    LD                    HL
                          8495                                           IV
F9C4 AF                    8496                    XOR                    IY
                          8497                                           HL
                          8498                                           A
F9C5                        8499                    END_SYNC:            [Y+P_REL_ADDR_LO],L
F9C5 D1                    8500                    POP                    [Y+P_REL_ADDR_HI],H
F9C6 C1                    8501                    POP                    A
F9C7 E1                    8502                    POP                    DE
F9C8 FDE1                  8503                    POP                    BC
                          8504                                           HL
F9CA C9                    8505                    RET                    IV
                          8506                                           IV

```

; DID MASTER NODE ACKNOWLEDGE ?
; ***** NEED PROTECTION HERE
; GET PCB ADDR INTO HL
; IN CASE SOMEONE ACCIDENTALLY DOES
; A PCB RELOCATION COMMAND -
; IT WILL RELOCATE TO THIS SPOT

; NO ERROR
; RESTORE HL
; IV

LOCATION OBJECT CODE LINE SOURCE LINE

```

8508 :
8509 : ENTRY:          SCAN_ACTIVE
8510 : NEEDS:          [CURRENT_PCB] SET TO CURRENTLY OPERATING PCB
8511 : RETURNS:        1 DCB ALLOTTED FOR EACH DEVICE ANSWERING A STATUS REQUEST
8512 :
8513 :
8514 : MASTER NODE SCANNING EACH OF THESE DCB'S FOR WORK TO DO
8515 : GLB             __SCAN_ACTIVE
8516 :
8517 : SCAN_ACTIVE
8518 : PUSH           BC          : SAVE BC
8519 : PUSH           DE          : DE
8520 : PUSH           HL          : HL
8521 : PUSH           IV          : IV
8522 : PUSH           IX          : IX
8523 :
8524 : LD             HL,[CURRENT_PCB] : ZERO OUT DCB'S
8525 : LD             DE,P_SIZE        : POINT TO DCB'S
8526 : ADD            HL,DE
8527 : LD             E,L
8528 : LD             D,H
8529 : INC            DE
8530 : LD             BC,(15*D_SIZE)-1
8531 : LD             [HL],0
8532 : LDIR
8533 :
8534 : LD             IV,[CURRENT_PCB] : FIND START ADDR OF DCB'S
8535 : LD             DE,P_SIZE
8536 : ADD            IV,DE
8537 :
8538 : LD             IX,[CURRENT_PCB] : IX = [CURRENT_PCB]
8539 : LD             [IX+P_NUM_DCBS],1 : NEED AT LEAST 1 ACTIVE DCB
8540 :
8541 : LD             A,1              : START WITH DEV ADDR 1
8542 :
8543 :
8544 : LOOP3:
8545 : PUSH          AF              : SAVE CURRENT DEVICE ADDR
8546 :
8547 : LD             [IV+D_DEV_ADDR],A : SELECT A NEW ADDRESS
8548 :
8549 : LD             [IV+D_COM_STAT],DCB_STATUS : ISSUE A REQUEST FOR STATUS
8550 :
8551 : LOOP4:
8552 : BIT           CMND_COMPLETE_BIT,[IV+D_COM_STAT] : IO DONE ?
8553 : JR            Z,LOOP4         : ***** NEED PROTECTION HERE
8554 :
8555 : LD             A,[IV+D_COM_STAT] : CHECK IF VALID RESPONSE
8556 : CP            CMND_FIN_STATUS
8557 : JR            Z,RECVD_VALID_STAT : YES INDEED ...
8558 :
8559 : POP           AF              : TRY NEXT DEVICE ADDR
8560 :
8561 : INC            A
8562 : CP            MAX_DEV_ADDR+1
8563 : JR            NZ,LOOP3
8564 :

```

FILE: L3SABS:EOS_TF HEWLETT-PACKARD: EOS_UTIL (c) Coleco 1983 Confidential

LOCATION	OBJECT CODE	LINE	SOURCE	LINE	
FA11	DD3503	8565	DEC	[IX+P_NUM_DCBS]	: DON'T NEED LAST DCB WE USED
		8566			
FA14	1811	8567	JR	END_S_A	: ALL DONE
		8568			
FA16		8569	RECVD_VALID_STAT:		
FA16	DD3403	8570	INC	[IX+P_NUM_DCBS]	
		8571			
FA19	110015	8572	LD	DE.D_SIZE	: ADVANCE TO NEXT DCB
FA1C	FD19	8573	ADD	IY,DE	:
		8574			
FA1E	F1	8575	POP	AF	: ADVANCE TO NEXT NET ADDR
		8576			
FA1F	3C	8577	INC	A	:
FA20	FE10	8578	CP	MAX_DEV_ADDR+1	: NO MORE ?
FA22	2002	8579	JR	NZ,LOOP3	: MORE TO CHECK
		8580			
FA24	DD3503	8581	DEC	[IX+P_NUM_DCBS]	: DON'T NEED LAST DCB WE USED
		8582			
FA27	DDE1	8583	END_S_A:		
FA27	DDE1	8584	POP	IX	: RESTORE IX
FA29	FDE1	8585	POP	IY	: IY
FA2B	E1	8586	POP	HL	: HL
FA2C	D1	8587	POP	DE	: DE
FA2D	C1	8588	POP	BC	: BC
		8589			
FA2E	C9	8590	RET		
		8591			

LOCATION OBJECT CODE LINE SOURCE LINE

```

8593
8594 : ENTRY:          _RELOC_PCB
8595 : NEEDS:          [CURRENT_PCB] SET TO CURRENTLY OPERATING PCB
8596 :                HL = ADDRESS OF WHERE PCB SHOULD BE RELOCATED TO
8597 : RETURNS:        PCB RELOCATED AS REQUESTED
8598 :                [CURRENT_PCB] UPDATED TO REFLECT NEW ADDRESS
8599
8600 :GLB              _RELOC_PCB
8601
8602 _RELOC_PCB
8603  PUSH          IV
8604
8605 LD              IV,[CURRENT_PCB]      ; IV = ADDRESS OF CURRENT PCB
8606
8607 LD              [IV+P REL_ADDR_LO],L   ; PLACE ADDR OF NEW PCB IN PLACE
8608 LD              [IV+P_REL_ADDR_HI],H
8609
8610 LD              [IV+P_COM_STAT],PCB_SNA ; TELL MASTER TO SET NEW PCB ADDRESS
8611
8612 LOOPS:
8613 LD              A,[IV+P_COM_STAT]      ; CHECK FOR OPERATION COMPLETE
8614 CP              PCB_SNA_ACK          ; ***** NEED PROTECTION HERE
8615 JR              NZ,LOOPS             ; NOPE
8616
8617 LD              [CURRENT_PCB],HL      ; REMEMBER WHERE PCB IS (PLEASE...)
8618
8619 POP             IV
8620
8621 RET
8622
FA2F FD2AFD70
FA2F FDE5
FA31 FD2AFD70
FA35 FD7501
FA38 FD7402
FA3B FD360003
FA3F
FA3F FD7E00
FA42 FE83
FA44 20F9
FA46 22FD70
FA49 FDE1
FA4B C9

```


LOCATION	OBJECT CODE	LINE	SOURCE LINE
		8624	
		8625	: ENTRY: GET_PCB_ADDR
		8626	
		8627	: GLB GET_PCB_ADDR
		8628	
FA4C		8629	GET_PCB_ADDR:
FA4C	FD2AFD70	8630	LD 1Y.[CURRENT_PCB] ; ADDRESS OF PCB
		8631	
FA50	C9	8632	RET
		8633	

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		8635	:GLB	SOFT_RES_KBD, SOFT_RES_PR, SOFT_RES_TAPE
		8636		
		8637	:EXT	FIND_DCB_CHK_IF_INACTIVE
		8638		
		8639		
FA51		8640	SOFT_RES_KBD:	
FA51 3E01		8641	LD	A.KEYBOARD_ID ; RESET KBD
FA53 1808		8642	JR	SOFT_RES_DEV
		8643		
FA55		8644	SOFT_RES_PR:	
FA55 3E02		8645	LD	A.PRINTER_ID ; RESET PRINTER
FA57 1804		8646	JR	SOFT_RES_DEV ;
		8647		
FA59		8648	SOFT_RES_TAPE:	
FA59 3E08		8649	LD	A.TAPE_ID ; RESET TAPE
FA5B 1800		8650	JR	SOFT_RES_DEV ;
		8651		
		8652	:GLB	SOFT_RES_DEV
		8653		
FA5D		8654	SOFT_RES_DEV:	
FA5D FDE5		8655	PUSH	IY ; SAVE IY
		8656		
FA5F CDF446		8657	CALL	FIND_DCB ; FIND APPROPRIATE DEVICE
FA62 2014		8658	JR	NZ,NO_DCB_FOUND ; BAD DEVICE I.D.
		8659		
FA64 CDF496		8660	CALL	CHK_IF_INACTIVE ; IS DEVICE IDLE?
FA67 200F		8661	JR	NZ,ACTIVE_DCB ; NOPE ... OOPS ...
		8662		
FA69 FD360002		8663	LD	[IY+D_COM_STAT],DCB_RESET ; SOFT RESET DEVICE
		8664		
FA6D		8665	DCB_CMD_LOOP:	
FA6D FDCB007E		8666	BIT	CMND_COMPLETE_BIT,[IY+D_COM_STAT] ; WAS COMMAND COMPLETE?
FA71 28FA		8667	JR	Z,DCB_CMD_LOOP ; NOPE, CHECK AGAIN
		8668		
FA73 FD7E00		8669	LD	A,[IY+D_COM_STAT] ; YUP
FA76 FE80		8670	CP	CMND_FIN_STATUS ;
		8671		
FA78		8672	ACTIVE_DCB:	
FA78		8673	NO_DCB_FOUND:	
FA78 FDE1		8674	POP	IY ;
		8675		
FA7A C9		8676	RET	
		8677		

FILE: JABS:EOS_TF HEWLETT-PACKARD: EOS_UTIL (c) Coleco Inc. Confidential

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		8679	
		8680	: READ DCB RETURN CODE (FIRST BYTE OF DCB)
		8681	
		8682	: GLB __RD_KBD_RET_CODE, __RD_PR_RET_CODE, __RD_TAPE_RET_CODE
		8683	
FA7B		8684	__RD_KBD_RET_CODE: ; RESET KBD
FA7B 3E01		LD	A, KEYBOARD_ID
FA7D 1808		JR	__RD_RET_CODE
		8687	
FA7F		8688	__RD_PR_RET_CODE: ; RESET PRINTER
FA7F 3E02		LD	A, PRINTER_ID
FA81 1804		JR	__RD_RET_CODE
		8691	
FA83		8692	__RD_TAPE_RET_CODE: ; RESET TAPE
FA83 3E08		LD	A, TAPE_ID
FA85 1800		JR	__RD_RET_CODE
		8695	: GLB __RD_RET_CODE
		8696	
FA87		8698	__RD_RET_CODE: ; SAVE IY
FA87 FDE5		PUSH	IY
		8700	
FA89 CDF446		CALL	__FIND_DCB
FA8C 2003		JR	NZ, NO_DCB_FOUND2
		8702	
		8703	
		8704	
		8705	
FA8E FD7E00		LD	A, [IY+D_COM_STAT]
		8706	
FA91		8708	NO_DCB_FOUND2: IY
FA91 FDE1		POP	IY
		8710	
FA93 C9		RET	
		8712	

: Z CONDITION CODE PRESENT
: COMMAND/STATUS BYTE FROM DCB

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
		8714	:WP	
		8715	:WP	
	<0100>			EQU 02000H
				EQU 00100H
		8716	:GLB	:GOTO WP
		8717	:EXT	:MEM_CNFGOO
		8718	:EXT	:SWITCH_MEM
FA94		8719	:GOTO	:WP:
FA94 3AFC17		8720	:LD	A,[MEM_CNFGOO]
FA97 CDFD14		8721	:CALL	SWITCH_MEM
		8722		
		8723	:WP_CODE_START	EQU 2000H
		8724	:WP_SIZE	EQU 3*8*1024
		8725	:WP_DEST	EQU 8000H
		8726	:LD	A,0
		8727	:OUT	[7FH].A
		8728		
		8729	:LD	HL,WCODE_START
		8730	:LD	DE,WCODE_DEST
		8731	:LD	BC,WCODE_SIZE
		8732	:LDIR	
		8733		
		8734	:LD	A,3
		8735	:OUT	[7FH].A
		8736		
FA9A C30100		8737	:JP	WP
		8738		

```

: WHERE TO GO TO START WP
: WHERE TO GO TO START WP

:get configuration byte for 32K ROM and 32 RAM

: BANK SWITCH IN THE BOOT ROMS
: ( I HOPE )

: WHERE DOES WP SIT IN BOOT ROMS
: WHERE WP GOES
: HOW BIG WP IS
: MOVE WP INTO PLACE

: BANK SWITCH IN DS_7
: ( I KNOW )

: GOTO WP
    
```

```

LOCATION OBJECT CODE LINE SOURCE LINE
FA9D
FA9D C9
8740 :GLB READ_EOS
8741 :GLB READ_EOS
8742 READ_EOS:
8743 RET
8744 :
8745 : CODE FROM HERE TO END OF EOS_UTIL WAS INSERTED AFTER THE
8746 : REV. 06 ROM WAS BURNED. IT HAS BEEN COMMENTED OUT TO MAINTAIN
8747 : COMPATIBILITY WITH THAT ROM
8748 :
8749 : END
8750 :
8751 :
8752 :EOS_CODE_START EQU 00800H
8753 :EOS_SIZE EQU 02000H-800H
8754 :EOS_DEST EQU 0E000H+800H
8755 :
8756 : LD A,0
8757 : OUT [7FH],A
8758 :
8759 : LD HL,EOS_CODE_START
8760 : LD DE,EOS_DEST
8761 : LD BC,EOS_SIZE
8762 : LDIR
8763 :
8764 : LD A,3
8765 : OUT [7FH],A
8766 :
8767 : RET
8768 :
8769 :=====
8770 :=====
8771 :=====
8772 :=====
: BANK SWITCH IN THE BOOT ROMS
: ( I HOPE )
: WHERE DOES EOS SIT IN BOOT ROMS
: WHERE EOS GOES
: HOW BIG EOS IS
: MOVE EOS INTO PLACE
: BANK SWITCH IN OS_7
: ( I KNOW )
: EOS IS IN

```

LOCATION OBJECT CODE LINE SOURCE LINE

8774 : REV 0 8/24/83 V/D
8775 *****
8776 *

8777 * RD_1_BLOCK WILL READ ONE BLOCK OF DATA FROM A BLOCK DEVICE
8778 * TO A LOCATION SPECIFIED BY REGISTER HL
8779 *

8780 * INPUT: A DEVICE ID
8781 * LOW NIBBLE - DEVICE ADDRESS
8782 * HI NIBBLE - SECONDARY DEVICE ID
8783 *

8784 * HL DESTINATION ADDRESS IN RAM
8785 *

8786 * REGISTER PAIR.
8787 * BC DE - SECTOR NUMBER ON DEVICE
8788 *

8789 * THE TRANSFER BYTE COUNT IS DEFINED BY "DCB_MAX_MESS_LEN" WHICH
8790 * IS FOUND IN THE DEVICE DCB.
8791 *

8792 * OUTPUT: CONDITION BITS:
8793 * Z: NO ERRORS
8794 * NZ: ERRORS OCCURED
8795 * A - ERROR CODE
8796 *

8797 * ALL REGISTERS ARE PRESERVED
8798 *

8799 * *****
8800 *****

8801 :GLB RD_1_BLOCK
8802 :EXT FIND_DCB
8803 :EXT CHK_IF_INACTIVE
8804 *

FA9E :RD_1_BLOCK: IY : SAVE DEVICE ID
FA9E FDES

FAA0 F5 PUSH AF

FAA1 CDFAC6 CALL START_RD_1_BLOCK

FAA4 2007 JR NZ.ERROR_WITH_DCB

FAA6 :RD_NOT_COMPLETED:
FAA6 F1 POP AF : GET DEVICE ID

FAA7 F5 PUSH AF : SAVE IT AGAIN
8818 *

FAA8 CDFAE2 CALL END_RD_1_BLOCK : LOOK FOR STATUS
8820 *

FAAB 30F9 JR NC.RD_NOT_COMPLETED : COMMAND NOT COMPLETED, CHECK STATUS AGAIN
8822 *

FAAD :ERROR_WITH_DCB:
FAAD FDE1 POP IY : REALLY IS AF BUT NEED TO SAVE IF AN ERROR
8825 *

FAAF FDE1 POP IY
8826 *

FAB1 C9 RET
8828 *

LOCATION OBJECT CODE LINE SOURCE LINE

```

8830 .....
8831 *
8832 *   ___WR_1_BLOCK WILL WRITE ONE BLOCK OF DATA TO A BLOCK DRIVER
8833 *   FORM A LOCATION SPECIFIED BY REGISTER HL
8834 *
8835 * INPUT:
8836 *   A   DEVICE ID
8837 *   LOW NIBBLE - DEVICE ADDRESS
8838 *   HI NIBBLE - SECONDARY DEVICE ID
8839 *
8840 *   HL  SOURCE ADDRESS IN RAM
8841 *
8842 * REGISTER PAIR:
8843 *   BC DE - SECTOR NUMBER ON DEVICE
8844 *
8845 * THE TRANSFER BYTE COUNT IS DEFINED BY "DCB_MAX_MESS_LEN"
8846 * WHICH IS FOUND IN THE DEVICE DCB
8847 *
8848 * OUTPUT:
8849 *   CONDITION BITS:
8850 *   Z: NO ERRORS
8851 *   NZ: ERRORS OCCURED
8852 *   A - ERROR CODE
8853 *
8854 * ALL REGISTERS ARE PRESERVED
8855 *
8856 .....
8857 *
8858 *   :GLB   ___WR_1_BLOCK
8859 *
8860 *   ___WR_1_BLOCK:
8861 *   ___PUSH  IY
8862 *
8863 *   PUSH  AF
8864 *
8865 *   CALL  ___START_WR_1_BLOCK
8866 *
8867 *   JR    NZ,ERROR_WITH_DCB2
8868 *
8869 *   WR_NOT_COMPLETED:
8870 *   ___POP  AF
8871 *
8872 *   PUSH  AF
8873 *
8874 *   CALL  ___END_WR_1_BLOCK
8875 *
8876 *   JR    NC,WR_NOT_COMPLETED
8877 *
8878 *   ERROR_WITH_DCB2:
8879 *   ___POP  IY
8880 *
8881 *   POP  IY
8882 *
8883 *   RET

```

; IS REALLY AF BUT NEED TO PRESERVE A REG IN CASE OF ERRORS

; COMMAND NOT COMPLETED,CHECK STATUS AGAIN

; DCB NOT FOUND OR NOT IDLE

; SAVE DEVICE ID

; GET DEVICE ID

```

LOCATION OBJECT CODE LINE SOURCE LINE
8885 *****
8886 *
8887 *
8888 *
8889 *
8890 *
8891 *
8892 *
8893 *
8894 *
8895 *
8896 *
8897 *
8898 *
8899 *
8900 *
8901 *
8902 *
8903 *
8904 *
8905 *
8906 *
8907 *
8908 *
8909 *
8910 *****
8911 *****
8912 :
8913 :NO_DCB_FOUND1 CHANGED TO BLK_NO_CB_FOUND1 TO AVOID CONFLICT
8914 :WITH IDENTICAL LABEL ELSEWHERE
8915 :GLB __START_RD_1_BLOCK
8916
8917 __START_RD_1_BLOCK:
8918 PUSH BC
8919 PUSH DE
8920 PUSH HL
8921 PUSH IY
8922
8923 CALL __FIND_DCB
8924
8925 JR NZ,B_NO_DCB_FOUND1 ; COULD NOT FIND DCB FOR THIS DEVICE
8926
8927 CALL CHK_IF_INACTIVE ; IS DEVICE IDLE?
8928 JR NZ,ACTIVE_DCB1 ; NOPE, GET OUT
8929
8930 CALL PREP_DCB ; SET UP THE DCB FOR READ
8931
8932 LD [IY+D_COM_STAT],DCB_RD ; SEND READ COMMAND
8933
8934 ACTIVE_DCB1:
8935 B_NO_DCB_FOUND1:
8936 POP IY
8937 POP HL
8938 POP DE
8939 POP BC
8940
8941 RET

```

__START_RD_1_BLOCK WILL SEND A READ COMMAND TO READ A BLOCK
OF DATA FROM A BLOCK DEVICE TO A BUFFER SPECIFIED BY REGISTER HL

INPUT: A DEVICE ID
LOW NIBBLE - DEVICE ADDRESS
HI NIBBLE - SECONDARY DEVICE ID

HL DESTINATION ADDRESS IN RAM

REGISTER PAIR:
BC DE - SECTOR NUMBER ON DEVICE

THE TRANSFER BYTE COUNT IS DEFINED BY "DCB_MAX_MESS_LEN" WHICH
IS FOUND IN THE DEVICE DCB.

OUTPUT: CONDITION BITS:
Z: NO ERRORS
NZ: ERRORS OCCURED
A - ERROR CODE

ALL REGISTERS ARE PRESERVED

FILE: L-3ABS:EOS_TF HEMLETT-PACKARD: EOS_BLK (c) Coleco 1500 Confidential Sat, 8 Sep 1984, 23:42 PAGE 229

LOCATION OBJECT CODE LINE SOURCE LINE

8942

LOCATION OBJECT CODE LINE SOURCE LINE

```

8944 * .....
8945 *
8946 *   END RD_1_BLOCK CHECKS THE STATUS AFTER A __START_RD_1_BLOCK
8947 *   IS_EXECUTED
8948 *
8949 *   INPUT:  A      DEVICE ID
8950 *          LOW NIBBLE - DEVICE ADDRESS
8951 *          HI NIBBLE - SECONDARY DEVICE ID
8952 *
8953 *   OUTPUT: CONDITION FLAGS
8954 *           C: OPERATION COMPLETED
8955 *           NC: OPERATION NOT COMPLETED
8956 *           Z: NO ERRORS
8957 *           NZ: ERRORS
8958 *           A = ERROR CODE
8959 *
8960 *   ALL REGISTERS ARE SAVED
8961 * .....
8962 * .....
8963 *
8964 *   :GLB   END_RD_1_BLOCK
8965 *   :EXT   CHK_IF_IDLE
8966 *
8967 *   END RD_1_BLOCK:
8968 *   PUSH  IV
8969 *
8970 *   CALL  __FIND_DCB
8971 *
8972 *   SCF
8973 *
8974 *   JR    NZ,NO_DCB_FOUND5
8975 *
8976 *   CALL  CHK_IF_IDLE
8977 *
8978 *   JR    NZ,DCB_IS_IDLE5
8979 *
8980 *   OR    A
8981 *   BIT  CMND_COMPLETE_BIT,[IV+D_COM_STAT] ; WAS COMMAND COMPLETE?
8982 *   JR    Z,COMMAND_NOT_COMPLETED_ ; NOPE.
8983 *
8984 *
8985 *   LD    A,[IV+D_COM_STAT]
8986 *
8987 *   CP    CMND_FIN_STATUS
8988 *   SCF
8989 *
8990 *
8991 *
8992 *   DCB_IS_IDLE5:
8993 *   NO_DCB_FOUND5:
8994 *   COMMAND_COMPLETED:
8995 *   COMMAND_NOT_COMPLETED:
8996 *   POP   IV
8997 *
8998 *   RET
8999 *

```

```

; SET CARRY FLAG TO INDICATE OPERATION COMPLETED
; NO DCB FOUND
; IS DEVICE IDLE?
; CLEAR CARRY FLAG
; CMND_COMPLETE_BIT,[IV+D_COM_STAT] ; WAS COMMAND COMPLETE?
; Z.COMMAND_NOT_COMPLETED_ ; NOPE.
; GET STATUS OF LAST OPERATION
; ANY ERRORS?
; INDICATE COMMAND COMPLETED
; CONDITION FLAGS: IF NO ERRORS: C,Z
; IF ERRORS: C,NZ A=ERROR CODE

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

9001 .....
9002 *
9003 *
9004 *
9005 *
9006 *
9007 *
9008 *
9009 *
9010 *
9011 *
9012 *
9013 *
9014 *
9015 *
9016 *
9017 *
9018 *
9019 *
9020 *
9021 *
9022 *
9023 *
9024 *
9025 *
9026 *
9027 *
9028 .....
9029
9030 ;NO_DCB_FOUND3 CHANGED TO B_NO_DCB_FOUND3 TO AVOID CONFLICT WITH
9031 ;IDENTICAL LABEL ELSEWHERE
9032
9033 ;GLB __START_WR_1_BLOCK
9034
9035 __START_WR_1_BLOCK:
9036 PUSH BC
9037 PUSH DE
9038 PUSH HL
9039 PUSH IY
9040
9041 CALL __FIND_DCB
9042
9043 JR NZ,B_NO_DCB_FOUND3 ; NO DCB FOUND
9044
9045 CALL CHK_IF_INACTIVE ; IS DEVICE ACTIVE?
9046 JR NZ,ACTIVE_DCB3 ; NOPE, GET OUT
9047
9048 CALL PREP_DCB ; SET UP DCB FOR WRITE
9049
9050 LD [IY+D_COM_STAT],DCB_WR ; SEND WRITE COMMAND
9051
9052 B_NO_DCB_FOUND3:
9053 ACTIVE_DCB3:
9054 POP IY
9055 POP HL
9056 POP DE
9057 POP BC

```

START WR 1 BLOCK WILL SEND A WRITE COMMAND TO WRITE A BLOCK OF DATA TO A BLOCK DEVICE FROM A LOCATION SPECIFIED BY REGISTER HL

INPUT:

A DEVICE ID
LOW NIBBLE - DEVICE ADDRESS
HI NIBBLE - SECONDARY DEVICE ID

HL SOURCE ADDRESS IN RAM

REGISTER PAIR:

BC DE - SECTOR NUMBER ON DEVICE

THE TRANSFER BYTE COUNT IS DEFINED BY "DCB_MAX_MESS_LEN" WHICH IS FOUND IN THE DEVICE DCB

OUTPUT:

CONDITION BITS:

Z: NO ERRORS
NZ: ERRORS OCCURRED
A - ERROR CODE

ALL REGISTERS ARE PRESERVED

LOCATION OBJECT CODE LINE SOURCE LINE

FB1A C9 9058 RET
 9059

```

LOCATION OBJECT CODE LINE SOURCE LINE
9061 *****
9062 *
9063 *
9064 *   END_WR_1_BLOCK WILL CHECK THE STATUS OF THE LAST OPERATION
9065 *   EXECUTED
9066 *
9067 *   INPUT:      A      DEVICE ID
9068 *             LOW NIBBLE - DEVICE ADDRESS
9069 *             HI NIBBLE - SECONDARY DEVICE ID
9070 *
9071 *
9072 *   OUTPUT:    C      CONDIOTN FLAGS
9073 *             NC     OPERATION COMPLETED
9074 *             Z      OPERATION NOT COMPLETED
9075 *             NZ     NO ERRORS
9076 *             NZ     ERRORS
9077 *             A      A = ERROR CODE
9078 *
9079 *
9080 *   ALL REGISTERS ARE PRESERVED
9081 *
9082 * *****
9083 *
9084 *   NO_DCB_FOUND4      B_NO_DCB_FOUND4
9085 *   DCB_IS_IDLE4     CHANGED TO BLK_DCB_IS_IDLE4 TO AVOID CONFLICT WITH
9086 *   IDENTICAL LABEL ELSEWHERE
9087 *
9088 *   :GLB   __END_WR_1_BLOCK
9089 *
9090 *   END_WR_1_BLOCK:
9091 *   PUSH   IV
9092 *
9093 *   CALL   __FIND_DCB
9094 *
9095 *   SCF
9096 *   JR     NZ,B_NO_DCB_FOUND4 ; CONDITION FLAGS: C,NZ A=ERROR CODE
9097 *
9098 *   CALL   CHK_IF_IDLE ; IS DEVICE IDLE?
9099 *   JR     NZ,BLK_DCB_IS_IDLE4 ; YUP
9100 *
9101 *   OR     A ; RESET CARRY FLAG
9102 *
9103 *   BIT   CMND_COMPLETE_BIT,[IY+D_COM_STAT] ; WAS COMMAND COMPLETE?
9104 *   JR     Z,CMND_NOT_COMPLETED ; NOPE.
9105 *
9106 *   LD    A,[IY+D_COM_STAT] ; YUP
9107 *
9108 *
9109 *   CP    CMND_FIN_STATUS ; ANY ERRORS?
9110 *   SCF   ; INDICATE COMMAND COMPLETED
9111 *   ; CONDITION FLAGS: IF NO ERRORS: C,Z
9112 *   ; IF ERRORS: C,NZ, A=ERROR CODE
9113 *
9114 *
9115 *   BLK_DCB_IS_IDLE4:
9116 *   B_NO_DCB_FOUND4:
9117 *   CMND_COMPLETED:

```

LOCATION OBJECT CODE LINE SOURCE LINE

FB35 9118 CMDND_NOT_COMPLETED:
FB35 FDE1 9119 POP IY
FB37 C9 9120 RET

```

LOCATION OBJECT CODE LINE SOURCE LINE
9122 *
9123 *
9124 * PREP DCB LOADS THE DCB BUFFER WITH THE PROPER VALUES FOR
9125 * A READ OR WRITE COMMAND
9126 *
9127 * INPUT:
9128 * IV START OF SPECIFIC DCB
9129 * HL ADDRESS OF RAM DESTINATION(READ)/SOURCE(WRITE)
9130 *
9131 * BC DE DEVICE SECTOR NUMBER
9132 *
9133 * OUTPUT: NONE
9134 *
9135 *
9136 *
9137 PREP_DCB:
9138 PUSH AF
9139
9140 SRL A
9141 SRL A ; EXTRACT SECONDARY DEVICE ID
9142 SRL A
9143 SRL A
9144
9145 LD [IY+D_SEC_DEV_ID],A ; ... AND PLACE IN DCB
9146 LD A,[IY+D_MAX_MSG_LEN_LO] ; GET BLOCK LENGTH AND PLACE
9147 LD [IY+D_BUF_LEN_LO],A ; ... IT IN BUFFER LENGTH
9148 LD A,[IY+D_MAX_MSG_LEN_HI] ; ... FOR READ
9149 LD [IY+D_BUF_LEN_HI],A
9150
9151 LD [IY+D_SECT_NUM],E ; LOAD UP A 4 BYTE SECTOR
9152 LD [IY+D_SECT_NUM+1],D ; ... NUMBER FROM REGISTERS
9153 LD [IY+D_SECT_NUM+2],C ; ... PAIR BC DE
9154 LD [IY+D_SECT_NUM+3],B
9155
9156 LD [IY+D_BUF_ADR_LO],L ; ADDRESS OF SOURCE/DESTINATION
9157 LD [IY+D_BUF_ADR_HI],H ; ... IN RAM
9158
9159 POP AF
9160 RET
9161 ;
9162 ;
9163 ;
9164 ;
9165 ;
9166 ;

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

9168
9169 :REV O (V/D 8-24-83)
9170
9171 :EXT DEVICE_ID
9172
9173 *****
9174 * THIS ROUTINE WILL INITIATE A READ COMMAND ON A CHARACTER DEVICE. THE
9175 * ROUTINE WILL WAIT FOR THE COMMAND TO BE COMPLETED. IF THE COMMAND
9176 * COMPLETED WITH NO ERRORS, THEN THE FLAG WILL BE ZERO. IF THERE
9177 * WERE ERRORS THEN THE FLAG WILL BE NON-ZERO.
9178 * INPUT: A ==> DEVICE ID
9179 * DE ==> DESTINATION ADDRESS
9180 * BC ==> NUMBER OF BYTES TO READ
9181 *
9182 * OUTPUT:
9183 *
9184 *
9185 *
9186 *
9187 *****
9188
9189 GLOBAL RD_CH_DEV
9190 :EXT FIND_DCB
9191
9192 RD_CH_DEV:
9193 LD [DEVICE_ID],A : SAVE DEVICE ID
9194
9195 CALL START_RD_CH_DEV
9196 JR NZ,GOOD_BYE
9197
9198 RD_DEV_LOOP:
9199 LD A,[DEVICE_ID]
9200 CALL END_RD_CH_DEV
9201 JR NC,RD_DEV_LOOP
9202
9203 GOOD_BYE:
9204 RET
9205

```

: RESTOR THE DEVICE ID

LOCATION OBJECT CODE LINE SOURCE LINE

```

9207 *
9208 * WRITE CHAR DEV ALLOWS THE CALLER TO SEND A PRINT COMMAND BY SPECIFYING
9209 * THE BUFFER OF THE STRING AND THE NUMBER OF BYTES TO PRINT.
9210 * INPUT: A ==> DEVICE ID
9211 * BC ==> NUMBER OF BYTES TO WRITE
9212 * HL ==> SOURCE OF DATA
9213 *
9214 * OUTPUT:
9215 * Z : NO ERRORS
9216 * NZ: ERROR OCCURED
9217 * A ==> ERROR CODE
9218 *
9219 *
9220 GLOBAL __WR_CH_DEV
9221
9222 __WR_CH_DEV:
9223 LD [DEVICE_ID],A
9224
9225 CALL __START_WR_CH_DEV
9226 JR NZ,SO_LONG
9227
9228 DCB_WRITE_LOOP:
9229 LD A,[DEVICE_ID]
9230 CALL __END_WR_CH_DEV
9231 JR NC,DCB_WRITE_LOOP
9232
9233 SO_LONG:
9234 RET
9235

```

: RESTORE THE ID FOR USE IN THE CALL

LOCATION OBJECT CODE LINE SOURCE LINE

```

9237 *****
9238 * THIS ROUTINE IS RESPONSIBLE FOR SETTING UP A DEVICE DCB TO DO A READ.
9239 * THE LAST THING DONE IN THIS ROUTINE IS TO ISSUE THE COMMAND TO READ.
9240 * A KEY FROM THE KEYBOARD.
9241 * INPUT:  A ==> CONTAINS THE DEVICE ID
9242 *         DE ==> DESTINATION ADDRESS
9243 *         BC ==> NUMBER OF BYTES TO READ
9244 *
9245 * OUTPUT: CONDITION FLAGS
9246 *         Z:  THE READ DATA FROM KEYBOARD COMMAND HAS BEEN SENT
9247 *         NZ: ERROR OCCURED
9248 *         A ==> ERROR CODE
9249 *****
9250
9251 ;GLB      START_RD_CH_DEV
9252 ;EXT      CHK_IF_INACTIVE
9253
9254 ___START_RD_CH_DEV:
9255   PUSH    IV
9256   CALL    ___FIND_DCB      ; FIND THE DCB
9257
9258   JR     NZ,NO_DCB_DEFINED
9259
9260   CALL   CHK_IF_INACTIVE  ; SEE IF THE KEY BOARD IS AVAILABLE
9261
9262   JR     NZ,DCB_IS_ACTIVE ; DCB IS CURRENTLY ACTIVE
9263
9264   LD     [IV+D_BUF_ADR_LO],E ; LOAD THE DESTINATION ADDRESS INTO DCB
9265   LD     [IV+D_BUF_ADR_HI],D
9266
9267   LD     [IV+D_BUF_LEN_LO],C ; LOAD THE BYTE COUNT
9268   LD     [IV+D_BUF_LEN_HI],B
9269
9270   LD     [IV+D_COM_STAT],DCB_RD ; INITIATE THE COMMAND TO READ
9271
9272
9273 DCB_IS_ACTIVE:
9274 NO_DCB_DEFINED:
9275   POP    IV
9276
9277   RET
9278

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

9280 *****
9281 * THIS ROUTINE WILL CHECK THE DEVICES DCB COM/STATUS BYTE AND RETURN *****
9282 * THE RESULT OF THE CHECK.
9283 * INPUT: A ==> DEVICE ID
9284 * OUTPUT: CONDITION FLAGS
9285 * C: COMMAND HAS FINISHED
9286 * NC: COMMAND HAS NOT BEEN PROCESSED
9287 * Z: NO ERROR OCCURED
9288 * NZ: ERROR OCCURED
9289 * A ==> ERROR CODE
9290 *****
9291 *****
9292 *****
9293 *****
9294 *****
9295 *****
9296 *****
9297 *****
9298 *****
9299 *****
9300 *****
9301 *****
9302 *****
9303 *****
9304 *****
9305 *****
9306 *****
9307 *****
9308 *****
9309 *****
9310 *****
9311 *****
9312 *****
9313 *****
9314 *****
9315 *****
9316 *****
9317 *****
9318 *****
9319 *****
9320 *****

          :GLB          _END_RD_CH_DEV
          :EXT          _CHK_IF_IDLE

          _END_RD_CH_DEV:
          _PUSH_IV
          CALL _FIND_DCB
          SCF

          JR          NZ,_NO_DCB          ; THERE WAS NO KEYBOARD DCB
          CALL       JR          _IS_DEVICE_IDL?      ; IS DEVICE IDL?
          NZ,_DCB_IS_IDL          ; YUP

          OR          A          ; CLEAR THE CARRY FLAG
          BIT        CMND_COMPLETE_BIT,[IY+D_COM_STAT] ; WAS THE COMMAND PROCESSED?
          JR          Z,_ROUTINE_FINISHED          ; NO, RETURN WITH THE CARRY FLAG CLEARED
          LD         A,[IY+D_COM_STAT]
          CP         CMND_FIN_STATUS
          SCF
          JR          NZ,_DCB_IS_IDL:
          NO_DCB:
          ROUTINE_FINISHED:
          POP IV
          RET
    
```

```

LOCATION OBJECT CODE LINE      SOURCE LINE
9322 * .....
9323 * START_WR_CH_DEV ALLOWS THE CALLER TO SEND A PRINT COMMAND BY SPECIFYING
9324 * THE BUFFER OF THE STRING AND THE NUMBER OF BYTES TO PRINT.
9325 * INPUT: A ==> DEVICE ID
9326 * BC => NUMBER OF BYTES TO WRITE
9327 * HL => SOURCE OF DATA
9328 * OUTPUT:
9329 *      CONDITION FLAGS
9330 *      Z : NO ERRORS
9331 *      NZ: ERROR OCCURED
9332 *      A ==> ERROR CODE
9333 * .....
9334
9335 :NO_DCB_FND3 CHANGED TO C_NO_DCB_FND3 TO AVOID CONFLICT WITH
9336 :IDENTICAL LABEL ELSEWHERE
9337
9338 :GLOBAL __START_WR_CH_DEV
9339
9340 __START_WR_CH_DEV:
9341 PUSH IV
9342 CALL __FIND_DCB ; FIND THE CORRECT DEVICE DCB
9343
9344 JR NZ,C_NO_DCB_FND3
9345
9346 CALL CHK_IF_INACTIVE ; IS DCB ACTIVE?
9347 JR NZ,DCB_IS_ACTIVE3 ; YUP
9348
9349 LD [IV+D_BUF_LEN_LO],C ; LENGTH OF THE BUFFER
9350 LD [IV+D_BUF_LEN_HI],B ;
9351
9352 LD [IV+D_BUF_ADR_LO],L ; ADDRESS OF THE STRING
9353 LD [IV+D_BUF_ADR_HI],H ;
9354
9355 LD [IV+D_COM_STAT],DCB_WR ; INITIATE THE WRITE DATA COMMAND
9356
9357 C_NO_DCB_FND3:
9358 DCB_IS_ACTIVE3: IV
9359 POP RET
9360
FBC2 FDE5
FBC4 CDF446
FBC7 2015
FBC9 CDF496
FBCC 2010
FBCE FD7103
FBD1 FD7004
FBD4 FD7501
FBD7 FD7402
FBDA FD360003
FBDE
FBDE FDE1
FBEO C9

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

9362 *****
9363 * __END_WR_CH_DEV WILL CHECK THE DEVICES DCB COM/STATUS BYTE AND RETURN *****
9364 * THE RESULT OF THE CHECK.
9365 * INPUT: A ==> DEVICE ID
9366 * OUTPUT: CONDITION FLAGS
9367 * C : COMMAND HAS FINISHED
9368 * NC: COMMAND HAS NOT BEEN PROCESSED
9369 * Z : NO ERROR OCCURED
9370 * NZ: ERROR OCCURED
9371 * A ==> ERROR CODE
9372 *****
9373 *****
9374 *****
9375 :NO_DCB_FND4 C_NO_DCB_FND4
9376 :DCB_IS_IDL4 CHANGED TO CHAR_DCB_IS_IDL4 TO AVOID CONFLICT WITH
9377 :IDENTICAL LABEL ELSEWHERE
9378 *****
9379 :GLB __END_WR_CH_DEV
9380 *****
9381 __END_WR_CH_DEV:
9382 __PUSH IY
9383 *****
9384 CALL __FIND_DCB
9385 SCF
9386 *****
9387 JR NZ,C_NO_DCB_FND4 ; THERE WAS NO PRINTER DCB
9388 *****
9389 CALL CHK_IF_IDLE ; IS DEVICE IDL?
9390 JR NZ,CHAR_DCB_IS_IDL4
9391 *****
9392 OR A ; CLEAR THE CARRY FLAG
9393 BIT CMND_COMPLETE_BIT,[IY+D_COM_STAT] ; WAS THE COMMAND PROCESSED?
9394 JR Z,NOT_COMPLETED ; NO, RETURN WITH THE CARRY FLAG CLEARED
9395 ; YES, THE COMMAND WAS PROCESSED
9396 LD A,[IY+D_COM_STAT] ; DID ANY ERRORS OCCUR?
9397 CP CMND_FIN_STATUS
9398 SCF
9399 *****
9400 CHAR_DCB_IS_IDL4:
9401 C_NO_DCB_FND4:
9402 NOT_COMPLETED: IY
9403 POP
9404 RET
9405 *****
9406 *****
9407 *****
9408 *****
9409 *****

```

LINE#	SYMBOL	TYPE	REFERENCES
8672	ACTIVE_DCB	A	8661
8934	ACTIVE_DCB1	A	8928
9053	ACTIVE_DCB3	A	9046
3819	ACTIVE_FILE	A	3792
322	ADD816	A	
1727	ADD_TO_BASE	A	1691, 1696
5695	ADJUST	A	5693
2499	APSV	A	2600
2838	AREA_SONG_IS	A	3200
2160	ARM_MASK	A	2240
5529	AROUND_POP	A	5525
2497	ASTEP	A	3280, 3361, 3383
2490	ATN	A	3277
2588	ATN_SWEEP	A	3217
5716	ATTRIB_TBL	A	5690
1052	ATTR_DELETED	A	5812
1053	ATTR_DEL_BIT	A	3791, 3921, 5367
1054	ATTR_EXECUTE	A	5720
1056	ATTR_HOLE	A	4086, 6522
1055	ATTR_HOLE_BIT	A	3785, 5365, 6192, 6232
1047	ATTR_PERMANENT	A	5719, 6505, 6513
1049	ATTR_READ_PROT	A	5717
1051	ATTR_SYSTEM	A	6505, 6513
1050	ATTR_USER	A	4190
1048	ATTR_WRITE_PROT	A	5718, 6513
2902	B1	A	2905
1074	BAD_FNJM_ERR	A	4478, 4591, 4849, 5079
5494	BASECMP	A	3823, 5375
1578	BASE_FACTORS	A	1560
7371	BLANKING_CHAR	A	7409, 7605, 7700, 7758
7695	BLANKS	A	7720, 7837
7701	BLANK_FILL	A	7703
7684	BLANK_PORTION	A	7754
7700	BLANK_REST	A	7687
7719	BLANK_SCREEN	A	7603, 7640
9115	BLK_DCB_IS_IDLE	A	9099
553	BLK_STRT_PTR	A	
7724	BLNK_SCRN_LOOP	A	7733
580	BLOCKS_REQ	A	3710, 3714, 3799, 3927, 3936, 3978, 3982, 3991, 4193
8126	BLOCK_ZERO	A	8230
7360	BS	A	7929
579	BUF_END	A	3736, 4662, 4675, 4930, 4949
578	BUF_START	A	3733, 3828, 3889, 4032, 4044, 4097, 4291, 4341, 4374, 4660, 4707, 4722, 4791, 4928, 5011, 5022, 5045
575	BYTES_REQ	A	4641, 4819, 4852, 4905
576	BYTES_TO_GO	A	4643, 4685, 4693, 4696, 4767, 4803, 4808, 4811, 4820, 4853, 4907, 4956, 4969, 4972, 5060
8935	B_NO_DCB_FOUND1	A	8925
9052	B_NO_DCB_FOUND3	A	9043
9116	B_NO_DCB_FOUND4	A	9096
304	CALC_OFFSET	A	
8217	CALL_WP	A	8198, 8241
1084	CANT_SYNC1	A	8454
1085	CANT_SYNC2	A	8480
1553	CASE_OF_CLR10	A	1550
1546	CASE_OF_COLOR	A	1527
1533	CASE_OF_GEN	A	1525
1540	CASE_OF_GEN10	A	1537
9400	CHAR_7_IS_IDL	A	9390

LINE#	SYMBOL	TYPE	REFERENCES
2401	CHECK	A	2394, 2396, 2398
3777	CHECK_ENT	A	3863
6986	CHECK_FDR_ETX	A	7019
3981	CHECK_HOLE_SIZE	A	3972
5448	CHECK_IF_DIRECT	A	3764, 5300
4985	CHECK_LAST_BLOC	A	4967
6098	CHECK_STATUS	A	6090
8184	CHK_D5	A	8174
6746	CHK_IF_IDLE	A	8976, 9098, 9303, 9389
6702	CHK_IF_INACTIVE	A	6983, 7161, 8660, 8927, 9045, 9260, 9346
1408	CHK_REG_1	A	1403
8195	CHK_TAPE	A	8186
7655	CHK_X_WITH_MAX	A	7653
7659	CHK_Y_COORDI	A	7657
7663	CHK_Y_WITH_MAX	A	7661
5020	CLEAR_DIRTY_BIT	A	
663	CLEAR_RAM_SIZE	A	8133
434	CLEAR_RAM_START	A	663, 8134, 8135
4432	CLOSE2	A	4437
4465	CLOSE3	A	4444
4457	CLOSE4	A	
7362	CLR_TO_EOL	A	7934
7363	CLR_TO_EDS	A	7935
9117	CMND_COMPLETED	A	
1201	CMND_COMPLETE_B	A	6654, 6710, 8552, 8666, 8981, 9103, 9307, 9393
1203	CMND_FIN_STATUS	A	6658, 8556, 8670, 8987, 9109, 9311, 9397
9118	CMND_NOT_COMPLE	A	9104
7251	CMND_NOT_PROCES	A	7237
8122	COLD_START_ADDR	A	8227, 8246
463	COLORTABLE	A	
8994	COMMAND_COMPLET	A	
7041	COMMAND_FINISH	A	7033
8995	COMMAND_NOT_COM	A	8982
2316	CONTROLLER_O_MA	A	2323
853	CONTROLLER_O_PO	A	2215, 2428
2317	CONTROLLER_1_BI	A	2327, 2349
856	CONTROLLER_1_PO	A	
7905	CONVERT_XY	A	7726, 7783, 7806, 7816, 7858, 7883
7361	CR	A	7930
2491	CTRL	A	3040
478	CURRENT_DEV	A	
483	CURRENT_PCB	A	6588, 8284, 8330, 8336, 8434, 8524, 8534, 8538, 8605, 8617, 8630
659	CUSOR	A	7372, 7395, 7471, 7488, 7514, 7521, 7536, 7547, 7557, 7577, 7629, 7778, 7857, 7880
7370	CUSOR_CHAR	A	7411, 7491
7372	CUSOR_X	A	7513
468	CUR_BANK	A	1897, 1957, 1966
4473	C_ERROR	A	4425, 4427, 4441
4481	C_ERROR2	A	4455
9357	C_NO_DCB_FND3	A	9344
9401	C_NO_DCB_FND4	A	9387
679	DCBS	A	
1066	DCB_BUSY	A	6721
8665	DCB_CMND_LOOP	A	8667
1185	DCB_IDLE	A	6707
1067	DCB_IDLE_ERR	A	6753
516	DCB_IMAGE	A	516
9273	DCB_IS_ACTIVE	A	9262

LINE#	SYMBOL	TYPE	REFERENCES
9358	DCB_IS_ACTIVE3	A	9347
9314	DCB_IS_IDL	A	9304
8992	DCB_IS_IDLE5	A	8978
6601	DCB_LOOP	A	6606
1065	DCB_NOT_FOUND	A	6616
6615	DCB_NO_FIND	A	6592,6608
1189	DCB_RD	A	8932,9270
1187	DCB_RESET	A	8663
1186	DCB_STATUS	A	6650,8549
1188	DCB_WR	A	7245,9050,9355
9228	DCB_WRITE_LOOP	A	9231
2393	DEBOUNCE	A	2338
316	DECLSN	A	
318	DECM5N	A	
2139	DEC_KBD_TBL	A	2246
477	DEFAULT_BT_DEV	A	8170,8225,8229,8234,8244
1829	DEL	A	1854
1081	DELETE_ERR	A	5826
8221	DEVO_OK	A	8206
8224	DEV1_OK	A	8182,8194,8215
6719	DEVICE_BUSY	A	6712
1089	DEVICE_DEPD_ERR	A	5965,6062
488	DEVICE_ID	A	6337,6485,9193,9199,9223,9229
3469	DE_TO_DEST	A	3357,3380,3410
6497	DIRECTORY_CHECK	A	5453
994	DIR_ATTR	A	3783,3921,4086,4191,5362,5698,6192,6232
546	DIR_BLOCK_NO	A	3715,3812,3885
1001	DIR_DAY	A	4209
1002	DIR_ENT_LENGTH	A	521,525,565,3567,3615,3772,3910,4021,5304,5384,6178,6348,6392,6396,6397
998	DIR_LAST_COUNT	A	4201,4202
2424	DIR_MASK	A	2433,2448
996	DIR_MAX_LENGTH	A	3796,3797,3924,3925,3968,3969,4074,4075,4194,4195,6245,6246,6251,6252,6396,6410,6411
1000	DIR_MONTH	A	4207
993	DIR_NAME	A	
995	DIR_START_BLOCK	A	3992,3996,4000,4004,4078,4080,4082,4084,5398,5399,5400,5401,6254,6255,6260,6261,6400
997	DIR_USED_LENGTH	A	4198,4199,6397
999	DIR_YEAR	A	4205
1565	DIVIDE	A	1569
6723	DONE	A	6717
3156	DONE_SNDMAN	A	3126
7368	DOWN	A	7938
3849	DO_READ_AGAIN	A	
8123	DSK4	A	8172,8176,8181
8124	DSK5	A	8184,8188,8193
2942	DUMAREA	A	2913,3115
1117	D_BUF_ADR	A	1118,1119
1119	D_BUF_ADR_HI	A	9157,9265,9353
1118	D_BUF_ADR_LO	A	9156,9264,9352
1121	D_BUF_LEN	A	1122,1123
1123	D_BUF_LEN_HI	A	9149,9268,9350
1122	D_BUF_LEN_LO	A	9147,9267,9349
1115	D_COM_STAT	A	6650,6654,6657,6706,6710,6748,7245,8549,8552,8555,8663,8666,8669,8706,8932,8981,8985,9050,9103,9106,9270
1134	D_DEV_ADDR	A	9307,9310,9355,9393,9396
1141	D_DEV_TYPE	A	6602,8547
3825	D_FILE	A	3789,3802,3806,3813
1136	D_MAX_LEN	A	1138,1139

LINE#	SYMBOL	TYPE	REFERENCES
1139	D_MAX_MSG_LN_HI	A	9148
1138	D_MAX_MSG_LN_LO	A	9146
1129	D_RET_COUNT	A	1131, 1132
1132	D_RET_COUNT_HI	A	
1131	D_RET_COUNT_LO	A	
1125	D_SECT_NUM	A	9151, 9152, 9153, 9154
1127	D_SEC_DEV_ID	A	9145
1145	D_SIZE	A	680, 6596, 8294, 8340, 8530, 8572
1143	D_STATUS_FLAGS	A	6084, 6678, 8178, 8190, 8202
332	EFFECT_OVER	A	
3221	EFXOVER	A	3188
1766	ELSE_11	A	1763
1781	ELSE_12	A	1778
3297	ENDMOREP	A	3289
2482	ENDSDATA	A	3086
5619	END_FCB	A	5597
1768	END_IF_11	A	1765
1783	END_IF_12	A	1780
7045	END_PRINT_ROUT1	A	6981, 7027
6680	END_R_D_D_S	A	6674
6662	END_R_S	A	6648
8499	END_SYNC	A	8457, 8483
8583	END_S_A	A	8567
4752	ENOUGH	A	4688
1004	ENT_PER_BLOCK	A	3767, 3862, 5313, 5353, 6166
1075	E0F_ERR	A	4827, 4998
5527	E0S	A	5507
97	E0S_CODE	A	1263
562	E0S_DAY	A	4208, 5130, 5158
100	E0S_GLB_RAM	A	428
98	E0S_GLB_TBL	A	694
99	E0S_JMP_TBL	A	118
561	E0S_MONTH	A	4206, 5132, 5160
101	E0S_PCB_DCB	A	670
8116	E0S_REV	A	8140
596	E0S_STACK	A	8131
560	E0S_YEAR	A	4204, 5134, 5162
7749	ERASE_THIS_LINE	A	7618, 7628
7617	ERASE_TO_EOL	A	7957
7627	ERASE_TO_E0S	A	7956
6286	ERROR	A	6143, 6212, 6233, 6273
5379	ERROR_CHECK	A	5376
6809	ERROR_OCCURRED	A	6797
1155	ERROR_RETRY	A	5875, 5897, 5925, 6010, 6034
8823	ERROR_WITH_DCB	A	8813
8878	ERROR_WITH_DCB2	A	8867
4161	ERR_GFN	A	4170
1209	ETX	A	5506, 5523, 5532, 5609, 5634, 6358, 6369, 6504, 6512, 6521, 6943, 6992, 7070, 7098, 7169
7494	EXIT	A	7524, 7545, 7566, 7586, 7654, 7658, 7662, 7666
88	FALSE	A	91
1011	FCB_ATTR	A	5809, 5813
1023	FCB_BLOCK	A	3621, 3622, 3623, 3624, 3719, 3720, 3721, 3722, 3744, 3745, 3746, 3747, 3809, 3810, 3831, 3837, 3840, 3843, 3846, 3854, 3855, 3856, 3857, 3883, 3895, 3896, 3948, 3951, 3954, 3957, 4033, 4034, 4035, 4036, 4048, 4054, 4055, 4056, 4057, 4098, 4099, 4100, 4101, 4346, 4348, 4350, 4352, 4375, 4376, 4377, 4378, 4449, 4450, 4451, 4452, 4530, 4531, 4532, 4533, 4543, 4545, 4547, 4549, 4569, 4570, 4571, 4572, 4711, 4713, 4715, 4717, 4723, 4724, 4725, 4726, 4734, 4737, 4740, 4743, 4754, 4757, 4760, 4763, 5012, 5013, 5014, 5015, 5029, 5031, 5033, 5035, 5046, 5047, 5048, 5049, 5264, 5265, 5266, 5267, 5279, 5280, 5281, 5282, 5317, 5324, 5327, 5330, 5333, 5339, 5340, 5341, 5342, 6267, 6268, 6269, 6270

LINE#	SYMBOL	TYPE	REFERENCES
524	FCB_BUFFER	A	6139, 6145, 6181, 6208, 6222
569	FCB_DATA_ADDR	A	3620, 3729, 3753, 4259, 4431, 4513, 4652, 4923, 5210, 5269, 5278, 5338, 5349, 6266, 6347, 6353, 6374, 6460, 6461, 6479
1021	FCB_DEVICE	A	3619, 3671, 3741, 3851, 3899, 4031, 4051, 4096, 4296, 4305, 4373, 4448, 4458, 4529, 4568, 4721, 5010, 5044, 5262, 5277, 5337, 6265
1013	FCB_FIRST_BLOCK	A	4313, 4315, 4318, 4321, 4345, 4347, 4349, 4351, 4542, 4544, 4546, 4548
568	FCB_HEAD_ADDR	A	3555, 3605, 3668, 4255, 4430, 4512, 4651, 4915, 5211, 5219, 5259, 5581, 6173, 6225
1024	FCB_LAST_BLOCK	A	3759, 3760, 3761, 3762, 3836, 3839, 3842, 3845, 3949, 3952, 3955, 3958, 4314, 4317, 4320, 4323, 4325, 4327, 4329, 4331, 4333, 4335, 4337, 4339, 4733, 4736, 4739, 4742, 4753, 4756, 4759, 4762, 5295, 5296, 5297, 5298, 5323, 5326, 5329, 5332
1016	FCB_LAST_COUNT	A	1018, 4559, 4560, 4792, 4793, 5026, 5027, 5061, 5062, 5064, 5065
1026	FCB_LENGTH	A	4256, 4269, 4433, 4515, 4654, 4916, 5218, 5587
1014	FCB_MAX_LENGTH	A	4941, 4944, 6150, 6151, 6162, 6163
1022	FCB_MODE	A	3669, 3681, 3788, 3971, 3979, 4265, 4294, 4354, 4369, 4396, 4439, 4466, 4521, 4539, 4541, 4554, 4672, 4745, 4788, 4947, 4983, 4986, 5021, 5038, 5073, 5222, 5593, 5641, 5680
1010	FCB_NAME	A	
1025	FCB_POINTER	A	3565, 3566, 3613, 3614, 3730, 3731, 3742, 3743, 3770, 3771, 3774, 3775, 3778, 3779, 3820, 3821, 3829, 3830, 3852, 3853, 3890, 3891, 3907, 3908, 3914, 3915, 4017, 4020, 4023, 4024, 4045, 4046, 4052, 4053, 4063, 4064, 4342, 4343, 4551, 4552, 4676, 4677, 4698, 4699, 4708, 4709, 4770, 4771, 4775, 4776, 4795, 4796, 4813, 4814, 4950, 4951, 4974, 4975, 5023, 5024, 5066, 5067, 5071, 5072, 5270, 5271, 5303, 5305, 5306, 5308, 5350, 5351, 5359, 5360, 5383, 5385, 5386, 6174, 6175, 6226, 6227
1195	FCB_S	A	8163
1012	FCB_START_BLOCK	A	1013, 3724, 3725, 3726, 3727
1018	FCB_STORED_BYTE	A	
1015	FCB_USED_LENGTH	A	4312, 4316, 4362, 4365, 4562, 4563, 4940, 4943, 5006, 5008, 6147, 6148
7399	FF	A	7932
542	FILENAME_CMPS	A	3549, 3576, 5372
528	FILE_COUNT	A	5274, 5310, 5356, 6165, 6199, 6235
5382	FILE_DELETED	A	5368
4123	FILE_EXISTS	A	3824
1071	FILE_EXISTS_ERR	A	4124
492	FILE_NAME_ADDR	A	4446, 4461
1079	FILE_NM_ERR	A	4162
538	FILE_NUMBER	A	5585, 5620, 5646
1469	FILL	A	1475
298	FILL_VRAM	A	8150
6619	FINISHED	A	6613
7252	FINISHED_WO_ERR	A	7239
2159	FIRE_MASK	A	2255
565	FMGR_DIR_ENT	A	
574	FMJM	A	4642, 4906
7598	FORM_FEED	A	7959
549	FOUND_AVAIL_ENT	A	3717, 3804, 3866, 3875
3659	FOUND_ENTRY	A	3805, 3808, 3867, 3876, 3880
3874	FOUND_HOLE	A	3786
6611	FOUND_IT	A	6603
2494	FPSV	A	2549
2489	FREQ	A	2688, 2692, 2695
2529	FREQ_SWEEP	A	3218
2495	FSTEP	A	2533, 2568, 3279, 3360
3865	FULL_DIR	A	3838, 3841, 3844, 3847
1077	FULL_DIR_ERR	A	4128
4127	FULL_DIR_EXIT	A	3868
5377	FULL_FILENAME	A	5374
1078	FULL_TAPE_ERR	A	4132
5286	GET_BUFF_ADDR	A	
5346	GET_BUFF_ADDR1	A	
7882	GET_CHAR_UNDER	A	7489
7879	GET_CHAR_UNDER_	A	7484
2441	GET_CO' _LLER_	A	2432

FILE: _SABS:E0S_TF CROSS REFERENCE TABLE

LINE#	SYMBOL	TYPE	REFERENCES
5171	GET_ERROR	A	5167
4148	GET_FILE_NM_LEN	A	4185
6991	GET_NEXT_ASCII	A	7001
7168	GET_NEXT_ASCII2	A	7178
5651	GET_OUT_ERR	A	5630
5468	GET_OUT_HERE	A	5477
6075	GET_STATUS_FLAG	A	5918, 6056
302	GET_VRAM	A	
9203	GOOD_BYE	A	9196
7650	GOTO_XY	A	7955
7364	GOTO_XY_CHAR	A	7936
3917	GOT_BLOCK	A	3893, 3903
2220	GOT_CORRECT_CON	A	2217
3689	GOT_FILE_SIZE	A	3676, 3680
4167	GOT_IT	A	4155
4290	GOT_ONE	A	4267
8242	GO_TO_TAPECODE	A	8232
6714	G_OK	A	6708
1711	HAVE_CNT	A	1706
1724	HAVE_CNT2	A	1721
6904	HAVE_THE_KEY	A	6886
762	HI_AUXILIARY_S	A	783, 786, 789, 792
764	HI_COLECOVISION	A	807, 810, 813, 816
763	HI_EXPANSION_80	A	795, 798, 801, 804
761	HI_INTRINSIC_80	A	771, 774, 777, 780
6520	HOLE_FILE_NAME	A	4091, 6217
7365	HOME	A	7933
2480	INACTIVE	A	2942, 3201, 3251, 3298, 3450
2438	INCREMENT_O_DIR	A	2434
2453	INCREMENT_1_DIR	A	2449
6494	INITIAL_DIRECTO	A	6346, 6529
6529	INIT_INFO_SIZE	A	
5221	INIT_LOOP	A	5224
1193	INIT_PCB_ADDR	A	
298	INIT_TABLE	A	
1559	INIT_TABLE80	A	1521, 1528
1571	INIT_TABLE90	A	1539, 1542, 1552, 1555
1358	INPUT_LOOP	A	1363, 1366
2425	INT_BIT	A	2431, 2445
1364	IN_DEC_HI_BYTE	A	1357
90	IN_EOS	A	
6372	ITS_AN_ETX	A	6360
7022	ITS_ETX	A	6994
5632	ITS_ETX1	A	5611
7186	ITS_ETX2	A	7171
6244	ITS_HOLE	A	6195
2161	JOY_MASK	A	2258
4746	JP_TO_READ3	A	4735, 4738, 4741, 4744
2956	JUKE_BOX	A	3292
2158	KBD_MASK	A	2243
1205	KBD_NAK	A	6889
2157	KBD_NULL	A	2140, 2155
497	KEYBOARD_BUFFER	A	6854, 6905
1151	KEYBOARD_ID	A	6831, 6853, 6881, 8641, 8685
3085	L1	A	3098
3216	L10	A	3207
3285	L13	A	3263

LINE#	SYMBOL	TYPE	REFERENCES
3305	L14	A	3287
3338	L15	A	3308
3366	L16	A	3343
3388	L17	A	3368
3402	L18	A	3397
3123	L2	A	3154
2546	L20	A	2535
3458	L20_LOAD_NEX	A	3455
2636	L23	A	2611
2670	L24	A	2662
3038	L5	A	3032
3077	L6	A	3034, 3044, 3048
3063	L7	A	3056
3067	L8	A	
3151	L9	A	3131
4787	LAST_BLOCK	A	4764
6198	LAST_ENTRY	A	6169
2237	LEAVE_THEM_OFF	A	2235
7369	LEFT	A	7939
4107	LETS_GET_OUT	A	
7358	LF	A	7543, 7931
654	LINEBUFFER	A	7698, 7728, 7784, 7822, 7830, 7838, 7859
308	LOAD_ASCII	A	
6356	LOAD_DIR	A	6367
4179	LOAD_NEW_ENTRY	A	3938, 4011
3244	LOAD_NEXT_NOTE	A	2978, 3224
2718	LOCL_DECLSN	A	2551, 2602, 2610
2739	LOCL_DECM5N	A	
2758	LOCL_MSNTOLSN	A	2557, 2608
2780	LOC_ADD816	A	2569
977	LOC_IN_ALPHA	A	1906
8444	LOOP1	A	8462
8470	LOOP2	A	8488
8544	LOOP3	A	8563, 8579
8551	LOOP4	A	8553
8612	LOOP5	A	8615
8401	LOOP7	A	8405
8398	LOOP8	A	8407
2076	LOOP EVERY BYTE	A	2080
2056	LOOP_EVERY_SPRI	A	2084
756	LD_800T_ROM_N_A	A	771, 783, 795, 807
758	LD_EXPANSION_O	A	777, 789, 801, 813
757	LD_INTRINSIC_RA	A	774, 786, 798, 810
759	LD_057_N_INTRIN	A	780, 792, 804, 816
4108	MAKE_ERROR	A	3750, 3765, 3859, 3902, 3940, 4015, 4040, 4060, 4104, 4120, 4125, 4129, 4133
4213	MAKE_ERR_1	A	4186
5534	MATCH	A	5515
5516	MATCHLESS	A	5510
1073	MATCH_ERR	A	5544
1157	MAX_DEV_ADDR	A	8562, 8578
770	MEM_CNFG00	A	1899, 8720
773	MEM_CNFG01	A	8152
776	MEM_CNFG02	A	
779	MEM_CNFG03	A	1969
782	MEM_CNFG04	A	
785	MEM_CNFG05	A	
788	MEM_C?_76	A	

LINE#	SYMBOL	TYPE	REFERENCES
791	MEM_CFG07	A	
794	MEM_CFG08	A	
797	MEM_CFG09	A	
800	MEM_CFG0A	A	
803	MEM_CFG0B	A	
806	MEM_CFG0C	A	
809	MEM_CFG0D	A	
812	MEM_CFG0E	A	
815	MEM_CFG0F	A	
841	MEM_SWITCH_PORT	A	1953
3445	MODBO	A	3281, 3300, 3330, 3362, 3384, 3419
5676	MODE_CHECK	A	4309, 4668, 4936
1041	MODE_DIRTY	A	4443, 4525
1040	MODE_DIRTY_BIT	A	4539, 4983, 5021, 5073
1036	MODE_EXEC	A	1037
1043	MODE_LAST	A	
1042	MODE_LAST_BLOCK	A	4369, 4541, 4672, 4745, 4788, 4947, 4986
1037	MODE_MAX	A	5687
1044	MODE_MODE	A	4355, 4555, 5039, 5594, 5642, 5684
1033	MODE_READ	A	1034
1039	MODE_REMAINDER	A	3681, 3788, 3971, 3979
1032	MODE_UNUSED	A	1033, 4396, 4466, 5222
1035	MODE_UPDATE	A	1036, 5040
1034	MODE_WRITE	A	1035, 4356, 4556
531	MOD_FILE_COUNT	A	6200, 6236
3967	MORE_BLOCKS	A	3950, 3953, 3956, 3959
5005	MORE_FILE	A	4987
5009	MORE_FILE1	A	5007
7534	MOVE_DOWN	A	7953, 7960
7608	MOVE_HOME	A	7958
7577	MOVE_LEFT	A	7952, 7962
7556	MOVE_RIGHT	A	7951
7520	MOVE_UP	A	7954
320	MSNTOLSN	A	
1059	NAME_LENGTH	A	5503, 5605
1821	NEGIV	A	1813
8363	NET_DELAY_COUNT	A	8374
844	NET_RESET_PORT	A	8369
8376	NET_RES_DELAY	A	8381
7489	NEW_CURSOR	A	7515, 7548
589	NEW_HOLE_SIZE	A	3986, 4073, 6161, 6248
586	NEW_HOLE_START	A	3995, 3999, 4003, 4007, 4077, 4079, 4081, 4083
5504	NEXT_CHAR	A	5513
5607	NEXT_CHAR1	A	5616
5457	NEXT_CHECK	A	5464
3905	NEXT_DIR	A	3922
3769	NEXT_ENT	A	3826
2345	NEXT_IX	A	2341
2492	NLEN	A	2536, 2539, 3278
1694	NOT_COLOR_TBL_A	A	1687
9402	NOT_COMPLETED	A	9394
2348	NOT_CONTROLLER	A	2324
5825	NOT_DELETED	A	5811
5475	NOT_DIR	A	5460
4066	NOT_END	A	4026
6263	NOT_HOLE	A	6193
6759	NOT_IDLE	A	6751

LINE#	SYMBOL	TYPE	REFERENCES
6172	NOT_LAST_ENTRY	A	
7250	NOT_PR_NAK	A	7243
5777	NOT_RENAMED	A	5746,5756,5770
2406	NOT_SAME	A	2403
4371	NOT_SIZE_1	A	4363,4366
7469	NOT_SPECIAL	A	7448
4732	NOT_TAPE1	A	
8459	NOT_TIME1	A	8448,8452
8485	NOT_TIME2	A	8474,8478
1069	NO_DATE_ERR	A	5172
9315	NO_DCB	A	9301
9274	NO_DCB_DEFINED	A	9258
8673	NO_DCB_FOUND	A	8658
7195	NO_DCB_FOUND1	A	7159
8708	NO_DCB_FOUND2	A	8702
8993	NO_DCB_FOUND5	A	8974
1091	NO_DIR_ERR	A	5476
5415	NO_ENT	A	5325,5328,5331,5334,5366
5951	NO_ERRORS	A	5939
4278	NO_FCBS	A	
1072	NO_FCB_ERR	A	4286
1070	NO_FILE_ERR	A	5419,5628
5540	NO_MATCH	A	5518,5524,5533
5625	NO_MATCH1	A	5636
6900	NO_NAK	A	6890
4578	NO_READ	A	4565
5896	NO_READ_ERRORS	A	5883,5887
6438	NO_SECTORS_TO_I	A	6425
3861	NO_TIMEOUT	A	
6275	NO_TRIM	A	6159,6241
6032	NO_WRITE_ERRORS	A	6018,6022
656	NUM_COLUMNS	A	7721,7818,7839
1062	NUM_FCBS	A	4276,4426,4508,4647,4911,5217,5621
655	NUM_LINES	A	7393,7599,7633,7808
7842	NUM_SPL_CHARS	A	7446
2476	OFF	A	2932,2934,2936,2938,3003,3011,3019,3027
649	OLDCHAR	A	7410,7606,7759,7853,7885
8125	ONE_BLOCK	A	
4383	OPENS	A	4357
4395	OP_ERR	A	4307,4310,4381
5828	OTHER_ERR	A	5808,5819
1307	OUTPUT_LOOP	A	1312,1315
1313	OUT_DEC_HI_BYTE	A	1306
3486	OUT_TO_SOUND_PO	A	2675,2691,2704,3033,3059
4264	O_NEXT_FCB	A	4277
3327	PASS1	A	3324
5082	PAST_WRITE_ERR	A	5077
461	PATRRGENTBL	A	1849
459	PATRRNAMETBL	A	
676	PCB	A	8283,8291,8292
1164	PCB_IDLE	A	
1175	PCB_RESET	A	1176
1176	PCB_RESET_ACK	A	
1172	PCB_SNA	A	1173,8610
1173	PCB_SNA_ACK	A	8614
1166	PCB_SYNC1	A	1167,8438
1167	PCB_S_ACK	A	8461

LINE#	SYMBOL	TYPE	REFERENCES
1169	PCB_SYNC2	A	1170,8464
1170	PCB_SYNC2_ACK	A	8487
1178	PCB_WAIT	A	1179
1179	PCB_WAIT_ACK	A	2321
619	PERSONAL_DEBOUN	A	
328	PLAY_IT	A	
312	POLLER	A	
2329	POLL_CONTROLLER	A	2355
277	PORT_COLLECTION	A	8143
839	PORT_TABLE	A	1971
2786	POS	A	2783
9137	PREP_DCB	A	8930,9048
1152	PRINTER_ID	A	6978,7007,7030,7156,7181,7192,7233,7274,8645,8689
503	PRINT_BUFFER	A	6942,6944,6945,7069,7071,7073,7097,7099,7101
7042	PRINT_CMD_FAIL	A	7013
7043	PRINT_CMD_FAIL	A	
3189	PROCESS_DATA_AR	A	3092
1090	PROG_NON_EXIST	A	6551
5711	PROTECT	A	5708
5706	PROTECT_FAULT	A	5700
7044	PRTR_DCB_ACTIVE	A	6984,7162
1086	PRT_ERR	A	5707
6977	PR_BUFF2	A	
1207	PR_NAK	A	
658	PRTR_NAME_TBL	A	7012,7035,7241
626	PTR_TO_LST_OF_S	A	7396,7915
627	PTR_TO_S_ON_0	A	2812,2890
628	PTR_TO_S_ON_1	A	2914,3029,3116,3139
629	PTR_TO_S_ON_2	A	2915,3006,3117
630	PTR_TO_S_ON_3	A	2916,3014,3118
2807	PT_IX_TO_SxDATA	A	2917,3022,3119
281	PUT_ASCII	A	2961,3083,3121
7855	PUT_CHAR_ON_SCR	A	7412,7470,7492
7490	PUT_CURSOR_ON	A	7619,7636,7641
7852	PUT_OLD_ON_SCORE	A	7511,7527,7608,7803
300	PUT_VRAM	A	
1805	PX_2_P_P_1	A	1808
306	PX_TO_PTRN_POS	A	
1100	P_COM_STAT	A	
1106	P_NUM_DCBS	A	8438,8460,8464,8486,8610,8613
1102	P_REL_ADDR	A	6589,8436,8539,8565,8570,8581
1104	P_REL_ADDR_HI	A	1103,1104
1103	P_REL_ADDR_LO	A	8494,8608
1108	P_SIZE	A	8493,8607
3539	QUEER	A	677,6594,8294,8340,8525,8535
520	QUERY_BUFFER	A	3534
3577	Q_ERROR	A	5744,5754,5757,5768,5806,5809,5813
1082	RANGE_ERR	A	3559
5709	RANGE_NONO	A	5710
9198	RD_DEV_LOOP	A	5686,5688
6793	RD_KBD_LOOP	A	9201
8815	RD_NOT_COMPLETE	A	
5273	RD_TIMEOUT	A	8821
5336	RD_TIMEOUT1	A	
4653	READ1	A	4658
4671	READ3	A	4747
4719	READ4	A	4712,4714,4716

LINE#	SYMBOL	TYPE	REFERENCES
8226	READ_AGAIN	A	8237, 8240
5867	READ_BLOCK	A	3748, 3858, 3901, 4058, 4379, 4574, 4728, 5050, 5283, 5343
5914	READ_COMMAND_CO	A	5905
4851	READ_ERR1	A	4669, 4730
4848	READ_ERR2	A	4646, 4648
5957	READ_ERROR	A	5893, 5912, 5923
7777	READ_FROM_CURSO	A	7750
4567	READ_IT	A	4557
4790	READ_LAST	A	4673
4766	READ_LAST1	A	4755, 4758, 4761, 4806
5877	READ_LOOP	A	5894
2331	READ_N_DEBOUNCE	A	2326
294	READ_REGISTER	A	
4359	READ_TIMEOUT	A	
290	READ_VRAM	A	1668, 7789, 7823, 7886
8569	RECVD_VALID_STA	A	8557
1080	RENAME_ERR	A	
8298	REPEAT_SYNC1	A	8300
8344	REPEAT_SYNC2	A	8346
6653	REQUEST_LOOP	A	6655
4514	RES1	A	4519
4540	RES2	A	4526
682	RESERVED_BYTE	A	
4538	RESET_BIT	A	
4043	RESET_THE_POINT	A	
7527	RESTORE_OLD	A	7541, 7562, 7570, 7582, 7590, 7668
4399	REST_REGS	A	4388
535	RETRY_COUNT	A	5876, 5890, 5926, 5941, 6011, 6025
5899	RETRY_FOR_ERROR	A	5908, 5910
6036	RETRY_WRITE	A	6046, 6048
7511	RETURN	A	7961
440	REV_NUM	A	8141
7367	RIGHT	A	7940
5075	ROOM1_OK	A	
5059	ROOM_OK	A	4959
6911	ROUTINE_DONE	A	6884, 6902, 6906
9316	ROUTINE_FINISHE	A	9308
4838	RQ_STAT_ERR	A	
1088	RQ_TP_STAT_ERR	A	4844
4586	R_ERROR	A	4507, 4509, 4523
4594	R_ERROR2	A	4536, 4576
4828	R_GET_OUT	A	4859
631	SAVE_CTRL	A	2924, 3042
7488	SAVE_CURSOR	A	7475, 7481, 7528, 7610
5316	SCAN1	A	5392
5355	SCAN2	A	5314, 5390
5394	SCAN3	A	5380
5409	SCAN_ERROR	A	5284, 5301, 5344
5252	SCAN_FOR_FILE	A	3558, 3609
7812	SCROLL_LOOP	A	7835
7543	SCROLL_POSSIBLE	A	7539
7802	SCROLL_UP	A	7485, 7546
508	SECTORS_TO_INIT	A	6340, 6376, 6420
511	SECTOR_NO	A	6414, 6430, 6480
6429	SECT_INIT_LOOP	A	6436
7029	SEND_ANOTHER_WR	A	7036
1683	SET_C	A	1625, 1665

LINE#	SYMBOL	TYPE	REFERENCES
7667	SET_CURSOR	A	7665
2112	SET_READ	A	1345
4948	SET_UP	A	4942, 4945
3548	SET_UP_A	A	3545
3712	SET_UP_DIR	A	3683
3752	SET_UP_FCB	A	
2121	SET_WRITE	A	1294, 1467, 2046
6908	SHOW_COMPLETE	A	6893
1217	SKIP	A	
3078	SND_MANAGER	A	
865	SOUNDPORT	A	2930, 3489
330	SOUNDS	A	
324	SOUND_INIT	A	
9233	SD_LONG	A	9226
7950	SPCL_VECTOR_TBL	A	7450
7928	SPECIAL_CHARS	A	7445, 7942
2318	SPINNER_ENABLE	A	2340
817	SPIN_SWO_CT	A	2214, 2430
618	SPIN_SW1_CT	A	
455	SPRITEATTRIBL	A	2045
457	SPRITEGENTBL	A	
2874	SR1ATN	A	2932, 3003, 3004
2873	SR1FRQ	A	3005
2876	SR2ATN	A	2934, 3011, 3012
2875	SR2FRQ	A	3013
2878	SR3ATN	A	2936, 3019, 3020
2877	SR3FRQ	A	3021
4152	SRCH_LOOP	A	4159
2883	SRNATN	A	2938, 3027, 3028
2882	SRNCTL	A	3046
592	STACK_START	A	
582	START_BLOCK	A	
5496	STRCMP	A	5378
862	STROBE_RESET_PO	A	
859	STROBE_SET_PORT	A	2211
6852	STRT_RD_KBD_LOO	A	
91	SUPERGAME	A	444, 485
279	SWITCH_MEM	A	1902, 1913, 1970, 2000, 8153, 8721
768	SWITCH_TABLE	A	
8424	SYNC_DELAY_B	A	8442, 8468
8423	SYNC_DELAY_DE	A	8441, 8450, 8467, 8476
3631	S_ERROR	A	3610
4131	TAPE_FULL	A	3965, 3976, 3985
1153	TAPE_ID	A	7300, 8169, 8196, 8200, 8214, 8223, 8236, 8649, 8693
3618	TAPE_TIMED_OUT	A	
621	TEMP_STACK	A	1894
1196	THREE1K_BLKs	A	8162
4528	TIMED_OUT	A	
1211	TIMEOUT	A	5885, 5907, 6020, 6044, 8239
4050	TIMEOUT_IN_READ	A	
4095	TIME_TO_WRITE	A	3942
3053	TOPE_OUT	A	3007, 3015, 3023
4118	T00_BIG	A	3705, 3709
1076	T00_BIG_ERR	A	4119
3906	T00_SMALL	A	3934
87	TRUE	A	90
6167	TRY_AGAIN	A	6170

LINE#	SYMBOL	TYPE	REFERENCES
7564	TRY_MOVE_DOWN	A	7560
7584	TRY_SCROLL_UP	A	7580
6012	TRY_WRITE_AGAIN	A	6030
326	TURN_OFF_SOUND	A	8145
3425	TYPE3	A	3390
6092	UNIT1	A	6086
7366	UP	A	7937
2657	UPATNCTRL	A	3039, 3047, 3064
314	UPDATE_SPINNER	A	
2687	UPFREQ	A	3065
657	UPPER_LEFT	A	7394, 7601, 7609, 7804
3113	UP_CH_DATA_PTRS	A	2979, 3230
577	USER_BUF	A	3553, 3564, 3603, 3612, 4640, 4700, 4704, 4772, 4815, 4904, 4976, 4980, 5068
581	USER_NAME	A	3672, 3822, 4183, 5257, 5370
3879	USE_ENTRY	A	3869
3947	USE_HOLE	A	3877
5648	U_GET_OUT	A	
5583	U_NEXT_FCB	A	5623
5701	U_OK	A	
847	VDP_CTRL_PORT	A	1393, 1435, 2124
850	VDP_DATA_PORT	A	
449	VDP_MODE_WORD	A	1404, 1411, 1519, 1689
451	VDP_STATUS_BYTE	A	1438
709	VECTOR_08H	A	
712	VECTOR_10H	A	
715	VECTOR_18H	A	
718	VECTOR_20H	A	
721	VECTOR_28H	A	
724	VECTOR_30H	A	
727	VECTOR_38H	A	
730	VECTOR_66H	A	
984	VOL_ATTR	A	
554	VOL_BLK_SZ	A	
989	VOL_DAY	A	
990	VOL_DES_LENGTH	A	6348, 6392
983	VOL_DIRSIZE	A	3755, 5291, 6378
985	VOL_DIR_CHECK	A	5459
988	VOL_MONTH	A	
982	VOL_NAME	A	
986	VOL_SIZE	A	6388, 6389
987	VOL_YEAR	A	
453	VRAM_ADDR_TABLE	A	1513, 1729
6801	WAITING	A	6806
6442	WE_GOT_ERRORS	A	6418, 6434
8715	WP	A	8737
4917	WRITE1	A	4919
4924	WRITE2	A	4926
4939	WRITE3	A	5041
5037	WRITE4	A	5030, 5032, 5034
6476	WRITE_1_BLOCK	A	6416, 6433
5043	WRITE_AGAIN	A	
6005	WRITE_BLOCK	A	3625, 4038, 4102, 4454, 4535, 5016, 6272, 6487
7196	WRITE_CMND_SENT	A	7184
7005	WRITE_COMMAND_A	A	7015
6052	WRITE_COMMAND_C	A	6042
7017	WRITE_DONE	A	7010
5080	WRITEF	A	4937, 5018, 5052

LINE#	SYMBOL	TYPE	REFERENCES
5078	WRITE_ERR2	A	4910,4912
6061	WRITE_ERROR	A	6028,6050
3630	WRITE_ERRORS	A	3627
6064	WRITE_OK	A	6059
282	WRITE_REGISTER	A	1572
4447	WRITE_TIMEOUT	A	
288	WRITE_VRAM	A	1628,1910,7729,7761,7831,7841,7862
8869	WR_NOT_COMPLETE	A	8876
310	WR_SPR_ATTRIB	A	
651	X_MAX	A	7402,7473,7558,7588,7655,7779
650	X_MIN	A	7399,7476,7512,7568,7578,7638,7651
653	Y_MAX	A	7407,7479,7537,7564,7663
652	Y_MIN	A	7404,7522,7584,7631,7659
6454	ZERO_OUT_BUFFER	A	6344,6427
HP 64000 Linker			

Tue, 10 Jul 1984, 2:32 Page 1

FILE/PROG NAME	PROGRAM	DATA	COMMON	ABSOLUTE	DATE	TIME	COMMENTS
A_UOS_00:N_EOS	E000				Thu, 14 Jun 1984,	14:15	Rev 07 - jk1
next address	E618						
DIR_HANDL:N_EOS	E618				Thu, 14 Jun 1984,	14:16	
OPENS:N_EOS	E600				Thu, 14 Jun 1984,	14:18	
RD_WT:N_EOS	EC17				Thu, 14 Jun 1984,	14:19	
DATES:N_EOS	EEC5				Thu, 14 Jun 1984,	14:16	
UTILS:N_EOS	EEEE				Thu, 14 Jun 1984,	14:19	
EOS_MKDIR:N_EOS	F323				Thu, 14 Jun 1984,	14:17	
next address	F442						
EOS_RET:N_EOS	F442				Thu, 14 Jun 1984,	14:18	REV 1 - RPD
EOS_GEN:N_EOS	F446				Thu, 14 Jun 1984,	14:17	
EOS_KBD:N_EOS	F4BA				Thu, 14 Jun 1984,	14:17	
EOS_PRTN:N_EOS	F4FC				Thu, 14 Jun 1984,	14:18	
EOS_TAPE:N_EOS	F5D7				Thu, 14 Jun 1984,	14:18	
CONS_OUT:N_EOS	F5DC				Thu, 14 Jun 1984,	14:15	Rev 18 - RPD
next address	F832						
EOS_START:N_EOS	F832				Thu, 14 Jun 1984,	14:18	Rev 07 - rfj
EOS_UTIL:N_EOS	F8F6				Thu, 14 Jun 1984,	14:18	
EOS_BLK:N_EOS	F89E				Thu, 14 Jun 1984,	14:16	
EOS_CHAR:N_EOS	F864				Thu, 14 Jun 1984,	14:16	
next address	F8FE						
TABLES:N_EOS	F8FF				Thu, 14 Jun 1984,	14:19	Rev 00 - RPD
next address	FC30						
EOSCHN_06:N_EOS	FC30	FD60			Thu, 14 Jun 1984,	14:32	Rev 12 - rfj
next address	FD5F	FEA7		FECO-FFFF			

XFER address= 0000 Defined by DEFAULT
 absolute & link.com file name=N_EOS:N_EOS
 Total# of bytes loaded= 1FE5


```

MEM_CNFG04      A FC1B      E0SCMN_06:N_EOS
MEM_CNFG05      A FC1C      E0SCMN_06:N_EOS
MEM_CNFG06      A FC1D      E0SCMN_06:N_EOS
MEM_CNFG07      A FC1E      E0SCMN_06:N_EOS
MEM_CNFG08      A FC1F      E0SCMN_06:N_EOS
MEM_CNFG09      A FC20      E0SCMN_06:N_EOS
MEM_CNFG0A      A FC21      E0SCMN_06:N_EOS
MEM_CNFG0B      A FC22      E0SCMN_06:N_EOS
MEM_CNFG0C      A FC23      E0SCMN_06:N_EOS
MEM_CNFG0D      A FC24      E0SCMN_06:N_EOS
MEM_CNFG0E      A FC25      E0SCMN_06:N_EOS
MEM_CNFG0F      A FC26      E0SCMN_06:N_EOS
MEM_SWITCH_PORT A FC27      E0SCMN_06:N_EOS
MODE_CHECK      P F009      UTILS:N_EOS
MOD_FILE_COUNT  D FDD5      E0SCMN_06:N_EOS
MSNTOLSN        P FD4A      E0SCMN_06:N_EOS
NET_RESET_PORT  A FC28      E0SCMN_06:N_EOS
NEW_HOLE_SIZE   D FE1A      E0SCMN_06:N_EOS
NEW_HOLE_START  D FE16      E0SCMN_06:N_EOS
NUM_COLUMNS     D FEAO      E0SCMN_06:N_EOS
NUM_LINES       D FE9F      E0SCMN_08:N_EOS
OLDCHAR         D FE79      E0SCMN_06:N_EOS
PATTRNGENTBL    D FD6A      E0SCMN_06:N_EOS
PATTRNAMETBL    D FD68      E0SCMN_06:N_EOS
PCB              A FEFO      E0SCMN_06:N_EOS
PERSONAL_DEBOUN D FESA      E0SCMN_06:N_EOS
PLAY_IT         P FD56      E0SCMN_06:N_EOS
POLLER          P FD3E      E0SCMN_06:N_EOS
PORT_COLLECTION P FD11      E0SCMN_06:N_EOS
PORT_TABLE      A FC27      E0SCMN_06:N_EOS
PRINT_BUFFER    D FD76      E0SCMN_06:N_EOS
PTRM_NAME_TBL   D FE43      E0SCMN_06:N_EOS
PTR_TO_LST_OF_S D FE6E      E0SCMN_06:N_EOS
PTR_TO_S_ON_0   D FE70      E0SCMN_06:N_EOS
PTR_TO_S_ON_1   D FE72      E0SCMN_06:N_EOS
PTR_TO_S_ON_2   D FE74      E0SCMN_06:N_EOS
PTR_TO_S_ON_3   D FE76      E0SCMN_06:N_EOS
PUT_ASCII       P FD17      E0SCMN_06:N_EOS
PUT_VRAM        P FD2C      E0SCMN_06:N_EOS
PX_TO_PTRN_POS  P FD35      E0SCMN_06:N_EOS
QUERY_BUFFER    D FDAO      E0SCMN_06:N_EOS
READ_BLOCK      P F17B      UTILS:N_EOS
READ_REGISTER   P FD23      E0SCMN_06:N_EOS
READ_VRAM       P FD1D      E0SCMN_06:N_EOS
RETRY_COUNT     D FDD6      E0SCMN_06:N_EOS
REV_NUM         D FD60      E0SCMN_06:N_EOS
SAVE_CTRL       D FE78      E0SCMN_06:N_EOS
SCAN_FOR_FILE   P EF0B      UTILS:N_EOS
SECTORS_TO_INIT D FD86      E0SCMN_06:N_EOS
SECTOR_NO       D FD87      E0SCMN_06:N_EOS
SOUNDPORT      A FC2F      E0SCMN_06:N_EOS
SOUNDS          P FD59      E0SCMN_06:N_EOS
SOUND_INIT     P FD50      E0SCMN_06:N_EOS
SPIN_SWO_CT     D FE58      E0SCMN_06:N_EOS
SPIN_SW1_CT     D FE59      E0SCMN_06:N_EOS
SPRITEATTRBL   D FD64      E0SCMN_06:N_EOS
SPRITEGENTBL   D FD66      E0SCMN_06:N_EOS

```

```

A_U0S_00:N_EOS
RD_WT:N_EOS OPENS:N_EOS
UTILS:N_EOS

```

```

EOS UTIL:N_EOS
UTILS:N_EOS DIR_HANDL:N_EOS
DIR_HANDL:N_EOS
CONS_OUT:N_EOS
CONS_OUT:N_EOS
CONS_OUT:N_EOS
A_U0S_00:N_EOS

```

```

EOS_UTIL:N_EOS
A_U0S_00:N_EOS

```

```

EOS_START:N_EOS
A_U0S_00:N_EOS
EOS_PRTR:N_EOS
CONS_OUT:N_EOS
A_U0S_00:N_EOS
A_U0S_00:N_EOS
A_U0S_00:N_EOS
A_U0S_00:N_EOS

```

```

UTILS:N_EOS
RD_WT:N_EOS OPENS:N_EOS DIR_HANDL:N_EOS

```

```

CONS_OUT:N_EOS A_U0S_00:N_EOS
UTILS:N_EOS

```

```

EOS_START:N_EOS
A_U0S_00:N_EOS
DIR_HANDL:N_EOS
EOS_MKDIR:N_EOS
EOS_MKDIR:N_EOS
A_U0S_00:N_EOS

```

```

A_U0S_00:N_EOS
A_U0S_00:N_EOS

```

```

START_BLOCK      D FE12      E0SCMN_06:N_EOS
STRCMP           P F053      UTILS:N_EOS
STROBE_RESET_PO A C2E       E0SCMN_06:N_EOS
STROBE_SET_PORT A FC2D      E0SCMN_06:N_EOS
SWITCH_MEM      P FD14      E0SCMN_06:N_EOS
SWITCH_TABLE    A FC17      E0SCMN_06:N_EOS
TEMP_STACK      D FE6E       E0SCMN_06:N_EOS
TURN_OFF_SOUND  P FD53      E0SCMN_06:N_EOS
UPDATE_SPINNER  P FD41      E0SCMN_06:N_EOS
UPPER_LEFT     D FE41        E0SCMN_06:N_EOS
USER_BUF       D FE06        E0SCMN_06:N_EOS
USER_NAME      D FE10        E0SCMN_06:N_EOS
VDP_CTRL_PORT  A FC29       E0SCMN_06:N_EOS
VDP_DATA_PORT  A FC2A       E0SCMN_06:N_EOS
VDP_MODE_WORD  D FD61       E0SCMN_06:N_EOS
VDP_STATUS_BYTE D FD63       E0SCMN_06:N_EOS
VECTOR_08H     A FBFF        E0SCMN_06:N_EOS
VECTOR_10H     A FC02        E0SCMN_06:N_EOS
VECTOR_18H     A FC05        E0SCMN_06:N_EOS
VECTOR_20H     A FC08        E0SCMN_06:N_EOS
VECTOR_28H     A FC0B        E0SCMN_06:N_EOS
VECTOR_30H     A FC0E        E0SCMN_06:N_EOS
VECTOR_38H     A FC11        E0SCMN_06:N_EOS
VECTOR_66H     A FC14        E0SCMN_06:N_EOS
VOL_BLK_SZ     D FDDC        E0SCMN_06:N_EOS
VRAM_ADDR_TABLE D FD64       E0SCMN_06:N_EOS
WRITE_BLOCK    P F1E6        UTILS:N_EOS
WRITE_VRAM     P FD1A        E0SCMN_06:N_EOS
WR_SPR_ATTRIB  P FD3B        E0SCMN_06:N_EOS
X_MAX         D FE7B         E0SCMN_06:N_EOS
X_MIN         D FE7A         E0SCMN_06:N_EOS
Y_MAX         D FE7D         E0SCMN_06:N_EOS
Y_MIN         D FE7C         E0SCMN_06:N_EOS
_CHECK_FCB    P FCFO        E0SCMN_06:N_EOS
_CLOSE_FILE   P FCC3        E0SCMN_06:N_EOS
_CONS_DISP    P FC33        E0SCMN_06:N_EOS
_CONS_INIT    P FC36        E0SCMN_06:N_EOS
_CONS_OUT     P FC39        E0SCMN_06:N_EOS
_CV_A         P FDOE        E0SCMN_06:N_EOS
_DELETE_FILE  P FCE1        E0SCMN_06:N_EOS
_DLY_AFT_HRD_RE P FC3C       E0SCMN_06:N_EOS
_END_PR_BUFF  P FC3F        E0SCMN_06:N_EOS
_END_PR_CH    P FC42        E0SCMN_06:N_EOS
_END_RD_1_BLOCK P FC45       E0SCMN_06:N_EOS
_END_RD_CH_DEV P FC48       E0SCMN_06:N_EOS
_END_RD_KBD   P FC4B       E0SCMN_06:N_EOS
_END_WR_1_BLOCK P FC4E       E0SCMN_06:N_EOS
_END_WR_CH_DEV P FC51       E0SCMN_06:N_EOS
_EOS_1        P FDO5        E0SCMN_06:N_EOS
_EOS_2        P FD08        E0SCMN_06:N_EOS
_EOS_3        P FD0B        E0SCMN_06:N_EOS
_EOS_START    P FC30        E0SCMN_06:N_EOS
_FILE_QUERY   P FCFF        E0SCMN_06:N_EOS
_FIND_DCB     P FC54        E0SCMN_06:N_EOS
_FMGR_INIT    P FCBA        E0SCMN_06:N_EOS
_GET_DATE     P FCDB        E0SCMN_06:N_EOS

```

```

DIR_HANDL:N_EOS
A_U0S_00:N_EOS
EOS_UTIL:N_EOS  EOS_START:N_EOS  A_U0S_00:N_EOS
A_U0S_00:N_EOS
EOS_START:N_EOS
CONS_OUT:N_EOS
RD_WT:N_EOS   DIR_HANDL:N_EOS
UTILS:N_EOS  DIR_HANDL:N_EOS
A_U0S_00:N_EOS
A_U0S_00:N_EOS
DIR_HANDL:N_EOS
A_U0S_00:N_EOS
EOS_MKDIR:N_EOS  RD_WT:N_EOS  OPENS:N_EOS  DIR_HANDL:N_EOS
A_U0S_00:N_EOS
CONS_OUT:N_EOS
CONS_OUT:N_EOS
CONS_OUT:N_EOS
CONS_OUT:N_EOS

```

```

GET_DCB_ADDR      P FC57      EOSCMN_06:N_EOS
GET_PCB_ADDR     P FC5A      EOSCMN_06:N_EOS
GOTO_WP          P FCE7      EOSCMN_06:N_EOS
HARD_INIT        P FC5D      EOSCMN_06:N_EOS
HARD_RESET_NET   P FC60      EOSCMN_06:N_EOS
INIT_TAPE_DIR    P FCBD      EOSCMN_06:N_EOS
MAKE_FILE        P FCC9      EOSCMN_06:N_EOS
MODE_CHECK       P FCF9      EOSCMN_06:N_EOS
OPEN_FILE        P FCC0      EOSCMN_06:N_EOS
POSIT_FILE       P FDO2      EOSCMN_06:N_EOS
PR_BUFF          P FC63      EOSCMN_06:N_EOS
PR_CH            P FC66      EOSCMN_06:N_EOS
QUERY_FILE       P FCCC      EOSCMN_06:N_EOS
RD_1_BLOCK       P FC69      EOSCMN_06:N_EOS
RD_DEV_DEP_STA  P FCE4      EOSCMN_06:N_EOS
RD_KBD           P FC6C      EOSCMN_06:N_EOS
RD_KBD_RET_COD  P FC6F      EOSCMN_06:N_EOS
RD_PR_RET_CODE  P FC72      EOSCMN_06:N_EOS
RD_RET_CODE     P FC75      EOSCMN_06:N_EOS
RD_TAPE_RET_CD  P FC78      EOSCMN_06:N_EOS
READ_BLOCK      P FCF3      EOSCMN_06:N_EOS
READ_EOS        P FCEA      EOSCMN_06:N_EOS
READ_FILE       P FCD2      EOSCMN_06:N_EOS
RELOC_PCB       P FC7B      EOSCMN_06:N_EOS
RENAME_FILE     P FCDE      EOSCMN_06:N_EOS
REQUEST_STATUS  P FC7E      EOSCMN_06:N_EOS
REQ_KBD_STAT    P FC81      EOSCMN_06:N_EOS
REQ_PR_STAT     P FC84      EOSCMN_06:N_EOS
REQ_TAPE_STAT   P FC87      EOSCMN_06:N_EOS
RESET_FILE      P FCC6      EOSCMN_06:N_EOS
SCAN_ACTIVE     P FC8A      EOSCMN_06:N_EOS
SCAN_FOR_FILE   P FCFC      EOSCMN_06:N_EOS
SET_DATE        P FCD8      EOSCMN_06:N_EOS
SET_FILE        P FCCF      EOSCMN_06:N_EOS
SOFT_INIT       P FC8D      EOSCMN_06:N_EOS
SOFT_RES_DEV    P FC90      EOSCMN_06:N_EOS
SOFT_RES_KBD    P FC93      EOSCMN_06:N_EOS
SOFT_RES_PR     P FC96      EOSCMN_06:N_EOS
SOFT_RES_TAPE   P FC99      EOSCMN_06:N_EOS
START_PR_BUFF   P FC9C      EOSCMN_06:N_EOS
START_PR_CH     P FC9F      EOSCMN_06:N_EOS
START_RD_1_BLO P FCA2      EOSCMN_06:N_EOS
START_RD_CH_DE  P FCA5      EOSCMN_06:N_EOS
START_RD_KBD    P FCAB      EOSCMN_06:N_EOS
START_WR_1_BLO P FCAC      EOSCMN_06:N_EOS
START_WR_CH_DE  P FCB1      EOSCMN_06:N_EOS
SYNC            P FCE1      EOSCMN_06:N_EOS
TRIM_FILE       P FCE4      EOSCMN_06:N_EOS
WRITE_BLOCK     P FCF6      EOSCMN_06:N_EOS
WRITE_FILE      P FCD5      EOSCMN_06:N_EOS
WR_1_BLOCK      P FCB4      EOSCMN_06:N_EOS
WR_CH_DEV       P FCB7      EOSCMN_06:N_EOS
ADDR16          P E374      A_00S_00:N_EOS
CALC_OFFSET     P E10A      A_00S_00:N_EOS
CHECK_FCB       P F089      UTILS:N_EOS
CLOSE_FILE      P EB04      OPENS:N_EOS
CONS_DISP       P F627      CONS_OUT:N_EOS

```

EOS_START:N_EOS

```

EOSCMN_06:N_EOS
EOSCMN_06:N_EOS
EOSCMN_06:N_EOS
EOSCMN_06:N_EOS
EOSCMN_06:N_EOS

```




SECTION V

OS 7 ABSOLUTE LISTINGS



APPENDIX E
JUMP TABLE

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
26

PLAY_SONGS	1F61
ACTIVATEP	1F64
PUTOBJP	1F67
REFLECT_VERTICAL	1F6A
REFLECT_HORIZONTAL	1F6D
ROTATE_90	1F70
ENLARGE	1F73
CONTROLLER_SCAN	1F76
DECODER	1F79
GAME_OPT	1F7C
LOAD_ASCII	1F7F
FILL_VRAM	1F82
MODE_1	1F85
UPDATE_SPINNER	1F88
INIT_TABLEP	1F8B
GET_VRAM	1F8E
PUT_VRAM	1F91
INIT_SPR_ORDERP	1F94
NR_SPR_NR_TBLP	1F97
INIT_TIMERP	1F9A
FREE_SIGNALP	1F9D
REQUEST_SIGNALP	1FA0
TEST_SIGNALP	1FA3
WRITE_REGISTERP	1FA6
WRITE_VRAM	1FA9
READ_VRAM	1FAC
INIT_WRITERP	1FAF
SOUND_INITP	1FB2
PLAY_ITP	1FB5
INIT_TABLE	1FB8
GET_VRAM	1FBB
PUT_VRAM	1FBE
INIT_SPR_ORDER	1FC1
NR_SPR_NR_TBL	1FC4
INIT_TIMER	1FC7
FREE_SIGNAL	1FCA
REQUEST_SIGNAL	1FCB
TEST_SIGNAL	1FCD
TIME_MGR	1FD3
TURN_OFF_SOUND	1FD6
WRITE_REGISTER	1FD9
READ_REGISTER	1FDC
WRITE_VRAM	1FDF
READ_VRAM	1FE2
INIT_WRITER	1FE5
WRITER	1FE8
POLLER	1FEB
SOUND_INIT	1FEE
PLAY_IT	1FF1
SOUND_MGR	1FF4
ACTIVATE	1FF7
PUTOBJ	1FFA
RAND_GEN	1FFD



APPENDIX F

OS SYMBOLS

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

ACTIVATE	EQU 01FF7H	;	OS:OS
ACTIVATEP	EQU 01F64H	;	OS:OS
ADD816	EQU 001B1H	;	OS:OS
AMERICA	EQU 00069H	;	OS:OS
ASCII_TABLE	EQU 0006AH	;	OS:OS
ATN_SWEEP	EQU 0012FH	;	OS:OS
CARTRIDGE	EQU 08000H	;	OS:OS
CONTROLLER_MAP	EQU 08008H	;	OS:OS
CTRL_PORT_PTR	EQU 01D43H		
DATA_PORT_PTR	EQU 01D47H		
DECLSN	EQU 00190H	;	OS:OS
DECM5N	EQU 0019BH	;	OS:OS
DECODER	EQU 01F79H	;	OS:OS
DEFER_WRITES	EQU 073C6H	;	OS:OS
EFXOVER	EQU 002EEH	;	OS:OS
ENLARGE	EQU 01F73H	;	OS:OS
ENLRG	EQU 01D6CH	;	OS:OS
FILL_VRAM	EQU 01FB2H	;	OS:OS
FREE_SIGNAL	EQU 01FCAH	;	OS:OS
FREE_SIGNALP	EQU 01F9DH	;	OS:OS
FREQ_SWEEP	EQU 000FCH	;	OS:OS
GAME_NAME	EQU 08024H	;	OS:OS
GAME_OPT	EQU 01F7CH	;	OS:OS
GET_VRAM	EQU 01FBBH	;	OS:OS
GET_VRAMP	EQU 01FBEH	;	OS:OS
INIT_SPR_ORDER	EQU 01FC1H	;	OS:OS
INIT_SPR_ORDERP	EQU 01F94H	;	OS:OS
INIT_TABLE	EQU 01FB8H	;	OS:OS
INIT_TABLEP	EQU 01F8BH	;	OS:OS
INIT_TIMER	EQU 01FC7H	;	OS:OS
INIT_TIMERP	EQU 01F9AH	;	OS:OS
INIT_WRITER	EQU 01FE5H	;	OS:OS
INIT_WRITERP	EQU 01FAFH	;	OS:OS
IRQ_INT_VECT	EQU 0801EH	;	OS:OS
LEAVE_EFFECT	EQU 001D5H	;	OS:OS
LOAD_ASCII	EQU 01F7FH	;	OS:OS
LOCAL_SPR_TBL	EQU 08002H	;	OS:OS
MODE_1	EQU 01FB5H	;	OS:OS
MSNTOLSN	EQU 001A6H	;	OS:OS
MUX_SPRITES	EQU 073C7H	;	OS:OS
NMI_INT_VECT	EQU 08021H	;	OS:OS
NUMBER_TABLE	EQU 0006CH	;	OS:OS
PLAY_IT	EQU 01FF1H	;	OS:OS
PLAY_ITP	EQU 01FB5H	;	OS:OS
PLAY_SONGS	EQU 01F61H	;	OS:OS

COLECOVISION PROGRAMMERS' MANUAL

Rev. 5

©Coleco Industries, Inc. 1982

CONFIDENTIAL DOCUMENT - DO NOT COPY

Page 2

1	POLLER	EQU 01FEBH	}	OS:OS
2	PUTOBJ	EQU 01FFAH	}	OS:OS
3	PUTOBJP	EQU 01F67H	}	OS:OS
4	PUT_VRAM	EQU 01FBEH	}	OS:OS
5	PUT_VRAM	EQU 01F91H	}	OS:OS
6	RAND_GEN	EQU 01FFDH	}	OS:OS
7	RAND_NUM	EQU 073C8H	}	OS:OS
8	READ_REGISTER	EQU 01FDCH	}	OS:OS
9	READ_VRAM	EQU 01FE2H	}	OS:OS
10	READ_VRAM	EQU 01FACH	}	OS:OS
11	REFLECT_HORIZONTAL	EQU 01F6DH	}	OS:OS
12	REFLECT_VERTICAL	EQU 01F6AH	}	OS:OS
13	REQUEST_SIGNAL	EQU 01FCDH	}	OS:OS
14	REQUEST_SIGNALP	EQU 01FA0H	}	OS:OS
15	ROTATE_90	EQU 01F70H	}	OS:OS
16	RST_10H_RAM	EQU 0800FH	}	OS:OS
17	RST_18H_RAM	EQU 08012H	}	OS:OS
18	RST_20H_RAM	EQU 08015H	}	OS:OS
19	RST_28H_RAM	EQU 08018H	}	OS:OS
20	RST_30H_RAM	EQU 0801BH	}	OS:OS
21	RST_8H_RAM	EQU 0800CH	}	OS:OS
22	SOUND_INIT	EQU 01FEEH	}	OS:OS
23	SOUND_INITP	EQU 01FB2H	}	OS:OS
24	SOUND_MAN	EQU 01FF4H	}	OS:OS
25	SPRITE_ORDER	EQU 08004H	}	OS:OS
26	STACK	EQU 073B9H	}	OS:OS
27	START_GAME	EQU 0800AH	}	OS:OS
28	TEST_SIGNAL	EQU 01FD0H	}	OS:OS
29	TEST_SIGNALP	EQU 01FA3H	}	OS:OS
30	TIME_MGR	EQU 01FD3H	}	OS:OS
31	TURN_OFF_SOUND	EQU 01FD6H	}	OS:OS
32	UPDATE_SPINNER	EQU 01F88H	}	OS:OS
33	VDP_MODE_WORD	EQU 073C3H	}	OS:OS
34	VDP_STATUS_BYTE	EQU 073C5H	}	OS:OS
35	WORK_BUFFER	EQU 08006H	}	OS:OS
36	WRITER	EQU 01FEBH	}	OS:OS
37	WRITE_REGISTER	EQU 01FD9H	}	OS:OS
38	WRITE_REGISTERP	EQU 01FA6H	}	OS:OS
39	WRITE_VRAM	EQU 01FDFH	}	OS:OS
40	WRITE_VRAM	EQU 01FA9H	}	OS:OS
41	WR_SPR_NM_TBL	EQU 01FC4H	}	OS:OS
42	WR_SPR_NM_TBLP	EQU 01F97H	}	OS:OS

1	GLB ACTIVATE	OS:OS
2	GLB ACTIVATEP	OS:OS
	GLB ADD816	OS:OS
3	GLB AMERICA	OS:OS
	GLB ASCII_TABLE	OS:OS
4	GLB ATN_SWEEP	OS:OS
	GLB CARTRIDGE	OS:OS
5	GLB CONTROLLER_MAP	OS:OS
	GLB CTRL_PORT_PTR	
6	GLB DATA_PORT_PTR	
	GLB DECLSN	OS:OS
7	GLB DECM5N	OS:OS
	GLB DECODER	OS:OS
8	GLB DEFER_WRITES	OS:OS
	GLB EFXOVER	OS:OS
9	GLB ENLARGE	OS:OS
	GLB ENLRG	OS:OS
10	GLB FILL_VRAM	OS:OS
	GLB FREE_SIGNAL	OS:OS
11	GLB FREE_SIGNALP	OS:OS
	GLB FREQ_SWEEP	OS:OS
12	GLB GAME_NAME	OS:OS
	GLB GAME_OPT	OS:OS
13	GLB GET_VRAM	OS:OS
	GLB GET_VRAMP	OS:OS
14	GLB INIT_SPR_ORDER	OS:OS
	GLB INIT_SPR_ORDERP	OS:OS
15	GLB INIT_TABLE	OS:OS
	GLB INIT_TABLEP	OS:OS
16	GLB INIT_TIMER	OS:OS
	GLB INIT_TIMERP	OS:OS
17	GLB INIT_WRITER	OS:OS
	GLB INIT_WRITERP	OS:OS
18	GLB IRQ_INT_VECT	OS:OS
	GLB LEAVE_EFFECT	OS:OS
19	GLB LOAD_ASCII	OS:OS
	GLB LOCAL_SPR_TBL	OS:OS
20	GLB MSNTOLSN	OS:OS
	GLB MODE_1	OS:OS
21	GLB MUX_SPRITES	OS:OS
	GLB NMI_INT_VECT	OS:OS
22	GLB NUMBER_TABLE	OS:OS
	GLB PLAY_IT	OS:OS
23	GLB PLAY_ITP	OS:OS
	GLB PLAY_SONGS	OS:OS
24	GLB POLLER	OS:OS
	GLB PUTOBJ	OS:OS
25	GLB PUTOBJP	OS:OS
26		

1		
2	GLB PUT_VRAM	OS:OS
	GLB PUT_VRAMP	OS:OS
3	GLB RAND_GEN	OS:OS
	GLB RAND_NUM	OS:OS
4	GLB READ_REGISTER	OS:OS
	GLB READ_VRAM	OS:OS
5	GLB READ_VRAMP	OS:OS
	GLB REFLECT_HORIZON	OS:OS
6	GLB REFLECT_VERTICAL	OS:OS
	GLB REQUEST_SIGNAL	OS:OS
7	GLB REQUEST_SIGNALP	OS:OS
	GLB ROTATE_90	OS:OS
8	GLB RST_10H_RAM	OS:OS
	GLB RST_18H_RAM	OS:OS
9	GLB RST_20H_RAM	OS:OS
	GLB RST_28H_RAM	OS:OS
10	GLB RST_30H_RAM	OS:OS
	GLB RST_8H_RAM	OS:OS
11	GLB SOUND_INIT	OS:OS
	GLB SOUND_INITP	OS:OS
12	GLB SOUND_MAN	OS:OS
	GLB SPRITE_ORDER	OS:OS
13	GLB STACK	OS:OS
	GLB START_GAME	OS:OS
14	GLB TEST_SIGNAL	OS:OS
	GLB TEST_SIGNALP	OS:OS
15	GLB TIME_MGR	OS:OS
	GLB TURN_OFF_SOUND	OS:OS
16	GLB UPDATE_SPINNER	OS:OS
	GLB VDP_MODE_WORD	OS:OS
17	GLB VDP_STATUS_BYTE	OS:OS
	GLB WORK_BUFFER	OS:OS
18	GLB WRITER	OS:OS
	GLB WRITE_REGISTER	OS:OS
19	GLB WRITE_REGISTERP	OS:OS
	GLB WRITE_VRAM	OS:OS
20	GLB WRITE_VRAMP	OS:OS
	GLB WR_SPR_NH_TBL	OS:OS
21	GLB WR_SPR_NH_TBLP	OS:OS
22		
23		
24		
25		
26		

LOCATION OBJECT CODE LINE SOURCE LINE

```

1 "Z80"
3 NAME ^OS_7PRIME^
4
5 DESCRIPTION MACRO
6 .GOTO ENDESCRIPTION
7
8 Author: Coleco Industries Inc.
9 Advanced Research & Development - Software Engineering
10 OS
11 Starting date: A long long time ago in a galaxy far far away . . .
12
13 Prom release Date: 24 Nov 1982. for internal use only
14 Prom release Rev: 7B
15
16 Prom release Date: December 28, 1982
17 Prom release Rev: 7PRIME
18
19 Header Rev: 2
20
21 *****
22 *
23 * ColecoVision Operating System
24 * Absolute Listing ( Rev 7PRIME )
25 * (c) Coleco Industries 1982
26 *
27 * *** Confidential ***
28 *
29 *****

```

This listing has the actual addresses of the start of OS routines
 Rev History (one line note indicating the change)

Rev.	Date	Change
4	14feb1983	Filler locations changed to OFFH to reflect OS_7PRIME. Prom release date changed to December 28, 1982 from May 1982. Name change to OS_7PRIME to reflect majority of versions in the field at this date.
3	24nov1982	Timing change to shorten LOGO delay Title changes to JMPTABLES and OSSR EQU
2	6oct1982	Minor comment modifications
1	23sept1982	OS_7 as one absolute file
0	May 1982	OS_7 listing by module

50 ENDESCRIPTION:
 51 MEND
 52 PROG

```

LOCATION OBJECT CODE LINE SOURCE LINE
55 ; Operating system sound routine EQUATES
56 ; FILE NAME: OSSR.EQU
57 ; *** Equates ***
58 ; Dedicated Cartridge RAM locations
<7020> 59 DEDAREA EQU 7020H ;the start of the RAM area dedicated to sound routines
<7020> 60 PTR_TO_LIST_OF_SND_ADDRS EQU DEDAREA+0
<7024> 61 PTR_TO_S_ON_0 EQU DEDAREA+2
<7024> 62 PTR_TO_S_ON_1 EQU DEDAREA+4
<7026> 63 PTR_TO_S_ON_2 EQU DEDAREA+6
<7028> 64 PTR_TO_S_ON_3 EQU DEDAREA+8
<702A> 65 SAVE_CTRL EQU DEDAREA+10
66 ; Attenuation level codes
<000F> 67 OFF EQU 0FH ;OFF [NO SOUND]
68 ; Sound output port
<00FF> 69 SOUND_PORT EQU 0FFH ;data to sound chip thru this port
70 ; Special byte 0 codes
<00FF> 71 INACTIVE EQU 0FFH
<003E> 72 SEFFECT EQU 62
<0000> 73 ENDSDATA EQU 0
74 ; Offsets within an SxDATA song data area
<0000> 75 CH EQU 0
<0000> 76 SONGNO EQU 0
<0001> 77 MEXTNOTEPTR EQU 1
<0003> 78 FREQ EQU 3
<0004> 79 ATN EQU 4
<0004> 80 CTRL EQU 4
<0005> 81 MLEN EQU 5
<0006> 82 FPS EQU 6
<0006> 83 FPSV EQU 6
<0007> 84 FSTEP EQU 7
<0008> 85 ALEN EQU 8
<0008> 86 ASTEP EQU 8
<0009> 87 APS EQU 9
<0009> 88 APSV EQU 9
89 ; song end codes
<0010> 90 CHOEND EQU 00010000H
<0050> 91 CH2END EQU 01010000H
<0090> 92 CH2END EQU 10010000H
<0000> 93 CH3END EQU 11010000H
<0018> 94 CHOREP EQU 00011000H
<0058> 95 CH1REP EQU 01011000H
<0098> 96 CH2REP EQU 10011000H
<0008> 97 CH3REP EQU 11011000H
98 ; channel numbers, B7 -B6
<0000> 99 CHO EQU 0
<0040> 100 CH1 EQU 01000000H
<0080> 101 CH2 EQU 10000000H
<00C0> 102 CH3 EQU 11000000H
103 ; [page]
104 PROG
105

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

107
108 ;
109 ;
110 ;
111 ;
112 ***** EXTERNAL SYMBOLS *****
113
114 * EXTERNAL ROUTINES LINKED INTO OS
115
116 ;EXT REG_WRITE
117 ;EXT REG_READ
118 ;EXT VRAM_WRITE
119 ;EXT VRAM_READ
120 ;EXT INIT_QUEUE
121 ;EXT WRITER
122 ;EXT REG_WRITEQ
123 ;EXT VRAM_WRITEQ
124 ;EXT VRAM_READQ
125 ;EXT INIT_QUEUEQ
126
127 ;EXT POLLER
128 ;EXT UPDATE_SPINNER_
129 ;EXT CONT_SCAN
130 ;EXT DECODER_
131
132 ;EXT INIT_SOUND
133 ;EXT ALL_OFF
134 ;EXT JUKE_BOX
135 ;EXT SMD_MANAGER
136 ;EXT PLAY_SONGS
137 ;EXT INIT_SOUNDQ
138 ;EXT JUKE_BOXQ
139
140 ;EXT INIT_TIMER
141 ;EXT FREE_SIGNAL
142 ;EXT REQUEST_SIGNAL
143 ;EXT TEST_SIGNAL_
144 ;EXT TIME_MGR
145 ;EXT INIT_TIMERQ
146 ;EXT FREE_SIGNALQ
147 ;EXT REQUEST_SIGNALQ
148 ;EXT TEST_SIGNALQ
149
150 ;EXT INIT_TABLE_
151 ;EXT GET_VRAM
152 ;EXT PUT_VRAM
153 ;EXT INIT_SPR_ORDER_
154 ;EXT WR_SPR_NH_TBL_
155 ;EXT INIT_TABLEQ
156 ;EXT GET_VRAMQ
157 ;EXT PUT_VRAMQ
158 ;EXT INIT_SPR_ORDERQ
159 ;EXT WR_SPR_NH_TBLQ
160
161 ;EXT ACTIVATE
162 ;EXT PUTOBJ_
163 ;EXT REFLCT_VERT

```

;VIDEO DRIVERS

;PASCAL CALLS

; CONTROLLER ROUTINE

; SOUND ROUTINES

; PASCAL CALLS

; TIME MGMT ROUTINES

; PASCAL CALLS

;TABLE MA

;PASCAL CALLS

; GRAPHICS ROUTINES

.LOCATION OBJECT CODE LINE SOURCE LINE

```

164 ;EXT RFLCT_HOR
165 ;EXT ROT_90
166 ;EXT ENLGR
167 ;EXT PUTOBJO
168 ;EXT ACTIVATEQ
169
170 ;EXT GAME_OPT
171 ;EXT LOAD_ASCII
172 ;EXT FILL_VRAM
173 ;EXT MODE_1
174
175
176 * "HIDDEN EXTERNALS"
177
178 ;EXT DISPLAY_LOGO
179 ;EXT CONTROLLER_INIT
180 ;EXT ASCII_TBL
181 ;EXT NUMBER_TBL
182
183
184 ***** EXPORTS *****
185
186 * ENTRY POINTS TO OS ROUTINES
187
188 GLB INIT_TABLE
189 GLB GET_VRAM
190 GLB PUT_VRAM
191 GLB INIT_SPR_ORDER
192 GLB WR_SPR_MM_TBL
193 GLB INIT_TABLEP
194 GLB GET_VRAM
195 GLB PUT_VRAM
196 GLB INIT_SPR_ORDERP
197 GLB WR_SPR_MM_TBLP
198
199 GLB WRITE_REGISTER
200 GLB READ_REGISTER
201 GLB WRITE_VRAM
202 GLB READ_VRAM
203 GLB INIT_WRITER
204 GLB WRITER
205 GLB WRITE_REGISTERP
206 GLB WRITE_VRAMP
207 GLB READ_VRAMP
208 GLB INIT_WRITERP
209
210 GLB POLLER
211 GLB UPDATE_SPINNER
212 GLB CONTROLLER_SCAN
213 GLB DECODER
214
215 GLB SOUND_INIT
216 GLB TURN_OFF_SOUND
217 GLB PLAY_IT
218 GLB SOUND_MAN
219 GLB PLAY_SONGS
220 GLB SOUND_INITP

```

; PASCAL CALLS

;DISPLAYS THE GAME OPTION SCREEN
;LOADS ASCII CHARACTER GENERATORS
;FILLS DESIGNATED AREA OF VRAM WITH VALUE
;SETS UP A DEFAULT GRAPHICS MODE 1

***** EXPORTS *****

* ENTRY POINTS TO OS ROUTINES

; TABLE MA

; PASCAL CALLS

; VIDEO DRIVERS

; PASCAL CALLS

; CONTROLLER ROUTINES

; SOUND ROUTINES

; PASCAL CALLS

```

LOCATION OBJECT CODE LINE SOURCE LINE
221 GLB PLAY_ITP
222 GLB INIT_TIMER
223 GLB FREE_SIGNAL
224 GLB REQUEST_SIGNAL
225 GLB TEST_SIGNAL
226 GLB TIME_MGR
227 GLB INIT_TIMERP
228 GLB FREE_SIGNALP
229 GLB REQUEST_SIGNALP
230 GLB TEST_SIGNALP
231
232
233 GLB STACK
234 GLB VDP_STATUS_BYTE
235 GLB VDP_MODE_WORD
236 GLB AMERICA
237 GLB MIX_SPRITES
238 GLB DEFER_WRITES
239 GLB RAND_GEN
240
241 GLB PUTOBJ
242 GLB ACTIVATE
243 GLB REFLECT_VERTICAL
244 GLB REFLECT_HORIZONTAL
245 GLB ROTATE_90
246 GLB ENLARGE
247 GLB PUTOBJP
248 GLB ACTIVATEP
249
250 GLB GAME_OPT
251 GLB LOAD_ASCII
252 GLB FILL_VRAM
253 GLB MODE_1
254 GLB ASCII_TABLE
255 GLB NUMBER_TABLE
256

; TIME MGMT ROUTINES

; PASCAL CALLS

; MISC GLOBALS

;Can be called from Pascal
; or assembly language
; GRAPHICS ROUTINES

; PASCAL CALLS

;GAME OPTIONS DISPLAY
;LOADS ASCII CHARACTER GENERATORS
;FILLS DESIGNATED AREA OF VRAM WITH VALUE
;SETS UP A DEFAULT GRAPHICS MODE 1
;POINTER TO TABLE OF ASCII GENERATORS
;POINTER TO TABLE OF 0-9 PATTERN GENERATORS

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
258 ***** CARTRIDGE ROM DATA AREA *****
259
260 GLB CARTRIDGE
    EQU 8000H
<8000>
261 * THIS IS THE MEMORY LOCATION TESTED TO SEE IF A CARTRIDGE IS PLUGGED
262 * IN. IF IT CONTAINS THE PATTERN A455H THE OS ASSUMES THAT A GAME
263 * CARTRIDGE IS PRESENT. IF IT CONTAINS THE PATTERN 55AAH, THE OS
264 * ASSUMES THAT A TEST CARTRIDGE IS PRESENT.
265
266 LOCAL_SPR_TBL LOCAL_SPR_TBL
    EQU 8002H
<8002>
267 * THIS IS A POINTER TO THE CPU RAM COPY OF THE SPRITE NAME TABLE. THE
268 * TABLE COPY IS USED WHENEVER ONE LEVEL OF INDIRECTION IS DESIRED IN
269 * ADDRESSING THE VRAM TABLE. FOR EXAMPLE WHEN USING THE OS SPRITE
270 * MULTIPLEXING SOFTWARE.
271
272 SPRITE_ORDER SPRITE_ORDER
    EQU 8004H
<8004>
273 * THIS IS A POINTER TO THE CPU RAM SPRITE ORDER TABLE. THIS TABLE IS
274 * USED TO ORDER THE LOCAL SPRITE NAME TABLE.
275
276 WORK_BUFFER WORK_BUFFER
    EQU 8006H
<8006>
277 * THIS IS A POINTER TO A FREE BUFFER SPACE IN RAM. THE OBJECT ORIENTED
278 * GRAPHICS ROUTINES USED THIS BUFFER FOR TEMPORARY STORAGE.
279
280 CONTROLLER_MAP CONTROLLER_MAP
    EQU 8008H
<8008>
281 * THIS IS A POINTER TO THE CONTROLLER MEMORY MAP THAT IS MAINTAINED BY
282 * THE HIGH-LEVEL CONTROLLER SCANNING AND DEBOUNCE SOFTWARE.
283
284 START_GAME START_GAME
    EQU 800AH
<800A>
285 * THIS IS A POINTER TO THE START OF THE GAME.
286
287 ***** RESTART AND INTERRUPT VECTORS *****
288 * THESE ARE ADDRESSES IN CARTRIDGE ROM OF VECTORS WHICH MUST BE PLACED
289 * THERE BY THE CARTRIDGE PROGRAMMER. WHEN AN INTERRUPT OR RESTART
290 * OCCURS, THE OS VECTORS IT THROUGH THIS AREA. THE CARTRIDGE PROGRAMMER
291 * SHOULD PLACE A JUMP TO HIS OWN INTERRUPT HANDLER IN THE APPROPRIATE
292 * LOCATION.
293
294 RST_8H_RAM RST_8H_RAM
    EQU 800CH
<800C>
295 * THIS IS THE RESTART 8 SOFT VECTOR.
296
297 RST_10H_RAM RST_10H_RAM
    EQU 800FH
<800F>
298 * THIS IS THE RESTART 10 SOFT VECTOR.
299
300 RST_18H_RAM RST_18H_RAM
    EQU 8012H
<8012>
301 * THIS IS THE RESTART 18 SOFT VECTOR.
302
303 RST_20H_RAM RST_20H_RAM
    EQU 8015H
<8015>
304 * THIS IS THE RESTART 20 SOFT VECTOR.

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
315
316 GLB RST_28H_RAM
317 RST_28H_RAM EQU 8018H
318 * THIS IS THE RESTART 28 SOFT VECTOR.
319
320 GLB RST_30H_RAM
321 RST_30H_RAM EQU 801BH
322 * THIS IS THE RESTART 30 SOFT VECTOR.
323
324 GLB IRQ_INT_VECT
325 IRQ_INT_VECT EQU 801EH
326 * THIS IS THE MASKABLE INTERRUPT SOFT VECTOR
327
328 GLB NMI_INT_VECT
329 NMI_INT_VECT EQU 8021H
330 * THIS IS THE NMI SOFT VECTOR.
331
332 GLB GAME_NAME
333 GAME_NAME EQU 8024H
334 * FROM HERE TO START GAME THERE SHOULD BE A STRING OF ASCII CHARACTERS
335 * NAMES THAT HAS THE FOLLOWING FORM:
336 *
337 * NAME_OF_THIS_GAME/MAKER_OF_THIS_GAME/COPYWRITE_YEAR.
338 *
339 * FOR EXAMPLE:
340 *
341 * "DONKEY KONG/NINTENDO/1982"
342
343 * IMPORTANT NOTE *****
344
345 * ***** IT IS THE RESPONSIBILITY OF THE *****
346 * ***** CARTRIDGE PROGRAMMER TO PLACE *****
347 * ***** THESE CODES IN CARTRIDGE ROM *****
348

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

350 *****
351 *****
352 * OPERATING SYSTEM ROM CODE *
353 * *
354 * *
355 *****
356 *****
357 ***** PAGE ZERO *****
358 * PAGE ZERO CONTAINS THE RESTART VECTORS, INTERRUPT VECTORS, AND
359 * THE INTERRUPT VECTORING SOFTWARE, AS WELL AS THE DEFAULT HANDLERS
360 * FOR INTERRUPTS AND RESTARTS.
361 * BOOT-UP ROUTINE
362 *
363
364 * THE BOOT-UP ROUTINE HANDLES POWER ON RESETS AND RESTARTS TO 0. IT
365 * INITIALIZES THE STACK AND JUMPS TO THE POWER_UP ROUTINE.
366
367 * BEGIN BOOT-UP
368 BOOT_UP PROG
369
370 * KICK STACK LD SP,STACK
371
372
373 * JUMP TO POWER_UP
374
375 JP POWER_UP
376 END_BOOTUP
377 * END BOOT-UP

```

0000 317389

0003 C3006E
0006

LOCATION OBJECT CODE LINE SOURCE LINE

```

379
380 * RESTART VECTORS
381
382 * THE FOLLOWING ARE THE 8 PROGRAMMABLE RESTARTS. FOR EACH OF THE
383 * RESTART LOCATIONS BELOW THERE IS A VECTOR IN CARTRIDGE ROM.
384 * TO USE A RESTART, THE PROGRAMMER MUST PLACE THE ADDRESS OF THE
385 * ROUTINE WHICH HE/SHE WISHES TO ACCESS THROUGH THE RESTART AT THE
386 * CORRESPONDING VECTOR. THEREAFTER EVERY TIME THAT RESTART IS
387 * EXECUTED, THE CARTRIDGE PROGRAMMER'S ROUTINE WILL BE CALLED.
388
0006 FFFF          HEX          FF,FF          ;filler
0008 C3800C        JP RST_8H_RAM
391
0008 FFFFFFFFFF    HEX          FF,FF,FF,FF,FF,FF,FF,FF ;filler
0010 C3800F        JP RST_10H_RAM
394
0013 FFFFFFFFFF    HEX          FF,FF,FF,FF,FF,FF,FF,FF ;filler
0018 C38012        JP RST_18H_RAM
397
0018 FFFFFFFFFF    HEX          FF,FF,FF,FF,FF,FF,FF,FF ;filler
0020 C38015        JP RST_20H_RAM
400
0023 FFFFFFFFFF    HEX          FF,FF,FF,FF,FF,FF,FF,FF ;filler
0028 C38018        JP RST_28H_RAM
403
0028 FFFFFFFFFF    HEX          FF,FF,FF,FF,FF,FF,FF,FF ;filler
0030 C3801B        JP RST_30H_RAM
406

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
408 * MASKABLE INTERRUPT VECTORING SOFTWARE
409 *
410
411 * A MASKABLE INTERRUPT OCCURRING IN THE SYSTEM IS EQUIVALENT TO A
412 * RESTART TO 38H. THUS, THE MASKABLE INTERRUPT IS VECTORED IN EXACTLY
413 * THE SAME WAY AS THE VARIOUS RESTARTS GIVEN ABOVE. IN ORDER TO USE
414 * THE INTERRUPT, THE CARTRIDGE MUST PLACE THE ADDRESS OF HIS/HER
415 * INTERRUPT HANDLER IN THE IRQ_INT_VECT LOCATION IN CARTRIDGE ROM.
416
417 * THE CARTRIDGE PROGRAMMER IS RESPONSIBLE FOR SAVING ANY REGISTERS
418 * HIS/HER OWN INTERRUPT HANDLERS MAY USE, AND FOR RE-ENABLING
419 * INTERRUPTS IF HE/SHE NEEDS THEM TO BE RE-ENABLED.
420
421 * MASKABLE INTERRUPT
422 HEX FF,FF,FF,FF,FF ;Filler
423 IRQ_INTERRUPT JP ;38H
424
425 ***** RANDOM NUMBER GENERATOR *****
426
427 * (PLACED HERE FOR PURPOSES OF CODE COMPACTION)
428
429 * Random number generator (psuedo) for a 16 bit value
430 * This routine 'exclusive or's the 15th and 8th bit
431 * together. It then rotates the entire quantity to the
432 * left and inserts the 'exclusive or'ed bit into the rightmost
433 * bit. Upon leaving it stores the random # in a specified
434 * memory location.
435
436
437 * The random number can be accessed from the global location
438 * RAND_NUM or the HL pair or the Accumulator.
439
440 RAND_GEN: LD HL,[RAND_NUM]
441 BIT 7,H
442 JR Z,NOT_ON ;15th bit is on
443
444 BIT 0,H
445 JR Z,SET ;for 10 quantity then set
446 JR RESET ;for 11 reset
447
448 NOT_ON: BIT 0,H
449 JR Z,RESET ;For 00 reset
450
451
452 SET: SCF
453 JR CARRY_READY ;For 01 fall through to set
454
455 RESET: OR A
456
457 CARRY_READY: RL L
458 RL H
459 LD [RAND_NUM],HL
460 LD A,L
461 RET
462
463 AFTER_RANDOM
464
0033 FFFFFFFF
0038
0038 C3801E
0038 2A73C8
003C C87C
0040 2806
0042 C844
0044 2806
0046 1807
0048
0048 C844
004A 2803
004C
004C 37
004D 1801
004F B7
0050
0050 C815
0052 C814
0054 2273C8
0057 7D
0058 C9
0059

```

LOCATION OBJECT CODE LINE SOURCE LINE

465
466

LOCATION OBJECT CODE LINE SOURCE LINE

```

468
469 * THE NMI VECTORIZING SOFTWARE AND DEFAULT HANDLER
470
471 * WHEN AN NMI IS RAISED BY THE VDP IN THE COLECOVISION SYSTEM, IT
472 * CAUSES THE CPU TO RESTART TO 66H. THE VECTORIZING SOFTWARE FOR THE
473 * NMI IS IDENTICAL TO THAT FOR THE MASKABLE INTERRUPT EXCEPT THAT
474 * IT GETS ITS VECTOR FROM NMI_INT_VECT INSTEAD OF IRQ_INT_VECT.
475
476 * AGAIN THE CARTRIDGE PROGRAMMER IS RESPONSIBLE, IN HIS/HER OWN
477 * INTERRUPT HANDLERS FOR SAVING AND RESTORING THE PROCESSOR STATE
478 * WHEN NECESSARY, AND FOR CLEARING THE VDP CONDITION BY READING THE
479 * VDP STATUS REGISTER.
480
0059 FFFFFFFF HEX FF,FF,FF,FF,FF ;Filler
005E FFFFFFFF HEX FF,FF,FF,FF,FF ;Filler
0063 FFFFFF HEX FF,FF,FF ;Filler
484 * NON-MASKABLE INTERRUPT
485 NMI_INTERRUPT JP (NMI_INT_VECT)
486
0066 C38021
487

```

LOCATION OBJECT CODE LINE SOURCE LINE

```
489 ***** OS ROM DATA AREA *****
490
491
0069 3C 492 AMERICA DEF8 60
493 * THIS BYTE SHOULD BE USED WHENEVER THE CARTRIDGE PROGRAMMER WANTS TO
494 * SET UP REAL-TIME COUNTERS. IT HAS A VALUE OF 60 FOR COLECOVISIONS
495 * MARKETED IN THE USA AND 50 FOR EUROPEAN UNITS. USE OF THIS BYTE
496 * ENSURES CARTRIDGE COMPATIBILITY AT LEAST WHERE REAL-TIME COUNTING
497 * IS CONCERNED.
498
006A 16AB 499 ASCII TABLE DEF8 ASCII.TBL
500 * THIS IS THE ADDRESS OF THE ROM PATTERN GENERATORS FOR UPPERCASE
501 * ASCII WHICH ARE CONTAINED WITHIN THE OPERATING SYSTEM.
502
006C 1623 503 NUMBER TABLE DEF8 NUMBER.TBL
504 * THIS IS THE ADDRESS OF THE ROM PATTERN GENERATORS FOR THE NUMBERS
505 * 0-9 WHICH ARE CONTAINED WITHIN THE OPERATING SYSTEM.
506
```

LOCATION OBJECT CODE LINE SOURCE LINE

```

508 ***** POWER ON BOOT SOFTWARE *****
509 *****
510 ;
511 ; BOOT_UP
512 ;
513 ; SINCE THE VIDEO GAME SYSTEM MAY BE STARTED UP WITH A
514 ; GAME CARTRIDGE, KEYBOARD MODULE, OR BOTH (OR NOTHING)
515 ; INSTALLED AT BOOT UP, THE SOFTWARE MUST PERFORM THE
516 ; FOLLOWING:
517 ;
518 ; A. INITIALIZE THE INTERRUPT VECTORS.
519 ;
520 ; B. INITIALIZE RESTART VECTORS
521 ;
522 ; C. TURN OFF THE SOUND CHIP.
523 ;
524 ; D. DETERMINE IF A CARTRIDGE IS PLUGGED IN.
525 ; IF SO, BRANCH TO THE CARTRIDGE PROGRAM
526 ; ELSE, WAIT FOR CARTRIDGE.
526 FALSE EQU 0
527 TRUE EQU 1
528 * VALUES FOR BOOLEAN FLAGS
529
530 * BEGIN POWER_UP EQU $
531 POWER_UP EQU
532
533 * IF CARTRIDGE = 55AAH THEN EXIT TO START GAME (TEST)
534 LD HL,(CARTRIDGE)
535 A,L
536 CP 55H
537 JP NZ,NO_TEST
538 LD A,H
539 CP 0AAH
540 JP NZ,NO_TEST
541 LD HL,(START_GAME)
542 JP [HL]
543
544 * ELSE
545 NO_TEST_
546
547 * TURN OFF SOUND CHIP
548 CALL TURN_OFF_SOUND
549
550 * INITIALIZE RANDOM NUMBER GENERATOR
551 LD HL,33H
552 LD [RAND_NUM],HL
553
554 * CLEAR CONTROLLER BUFFER AREAS
555 CALL CONTROLLER_INIT
556
557 * DEFER_WRITES := FALSE
558 LD A,FALSE
559 LD [DEFER_WRITES],A
560
561 * MUX_SPRITES := FALSE
562 LD [MUX_SPRITES],A
563
564

```


FILE: OS_7PRIME:POS

HEWLETT-PACKARD: OPERATING SYSTEM (c) Coleco, 1982 CONFIDENTIAL

Fri, 18 May 1984, 16:19

PAGE 15

LOCATION OBJECT CODE LINE

SOURCE LINE

565 * EXIT TO DISPLAY LOGO AND TEST FOR CARTRIDGE
566 JP
567 DISPLAY_LOGO
568 * END BOOT-UP

0095 C31319

```

LOCATION OBJECT CODE LINE SOURCE LINE
0000
<738A>
570 ***** SYSTEM RAM AREA *****
571 DATA
572 DEFS
573 SYSTEM RAM AREA EQU 738AH ;Added to offset to first location
574 * THIS IS THE RAM AREA DEDICATED TO THE BASIC OS NEEDS. IT INCLUDES THE
575 * STACK, VARIOUS STATUS VARIABLES, AND ALL THE VARIABLES USED BY OS
576 * ROUTINES.
577
578 STACK EQU SYSTEM_RAM_AREA-1
579 * THIS IS THE TOP OF THE STACK
580
581 ; COM1
582 ; DEFS 9
583 ; DATA
584 PARAM AREA:
585 INIT_SOUND_DATA:
586 PRM AREA:
587 INIT_TIME_DATA:
588 TEMP1: DEFS 1
589 DEFS 1
590 DEFS 1
591 TEMP2: DEFS 1
592 DEFS 1
593 DEFS 1
594 SIGNAL_NUM: DEFS 1
595 DEFS 1
596 REPEAT_SIG_CODE: DEFS 1
597 DEFS 1
598 TIMER_LENGTH: DEFS 1
599 DEFS 1
600 DEFS 1
601 TEST_SIG_NUM: DEFS 1
602 DEFS 1
603
604
605 * THIS IS THE COMMON PARAMETER PASSING AREA AND THE HOLE IN THE DATA
606 * AREA THAT IS PROVIDED TO MAKE ROOM FOR IT.
607
608 VDP MODE WORD DEFS 2
609 * THE VDP MODE WORD CONTAINS A COPY OF THE DATA IN THE FIRST TWO VDP
610 * REGISTERS. BY EXAMINING THIS DATA, THE OS AND CARTRIDGE PROGRAMS
611 * CAN MAKE MODE-DEPENDENT DECISIONS ABOUT THE SPRITE SIZE OR VRAM
612 * TABLE ARRANGEMENT. THIS WORD IS MAINTAINED BY THE WRITE_REGISTER
613 * ROUTINE WHENEVER THE CONTENTS OF REGISTERS 0 OR 1 ARE CHANGED.
614
615 * IMPORTANT NOTE *****
616
617 * ***** IT IS THE RESPONSIBILITY OF THE *****
618 * ***** CARTRIDGE PROGRAMMER TO MAKE *****
619 * ***** SURE THAT NON-STANDARD USE OF *****
620 * ***** THE VDP REGISTERS DOES NOT MAKE *****
621 * ***** THE DATA IN THIS WORD INVALID *****
622
623 VDP_STATUS_BYTE DEFS 1
624 * THE DEFAULT HANDLER FOR THE HMI, WHICH MUST READ THE VDP STATUS
625 * REGISTER TO CLEAR THE INTERRUPT CONDITION, PLACES ITS CONTENTS
626 * HERE. THIS BYTE IS THE MOST ACCURATE REPRESENTATION OF THE ACTUAL

```

738A
738A
738A
738A
738A
738B
738C
738C
738D
738E
738E
738F
738F
738G
738G
738H
738H
738I
738I
738J
738J

73C3

73C5

LOCATION	OBJECT CODE LINE	SOURCE LINE
	627	* VDP STATUS THAT IS AVAILABLE TO THE CARTRIDGE PROGRAMMER PROVIDED
	628	* THAT THE VDP INTERRUPT IS ENABLED ON-CHIP
73C6	629	DEFER WRITES DEFS 1
	631	* DEFER WRITES IS A BOOLEAN FLAG WHICH IS SET TO FALSE AT POWER UP
	632	* TIME, SHOULD BE SET TO TRUE ONLY IF THE CARTRIDGE PROGRAMMER WISHES
	633	* TO DEFER WRITES TO VRAM. IF THIS FLAG IS TRUE THEN THE WRITER
	634	* ROUTINE MUST BE CALLED REGULARLY TO PERFORM DEFERRED WRITES.
	635	
73C7	636	MUX SPRITES DEFS 1
	637	* THIS BOOLEAN FLAG WITH DEFAULT FALSE VALUE SHOULD BE SET TO TRUE IF
	638	* THE CARTRIDGE PROGRAMMER WISHES ONE LEVEL OF INDIRECTIOIN TO BE
	639	* INSERTED INTO SPRITE PROCESSING BY HAVING ALL SPRITES WRITTEN TO
	640	* A LOCAL SPRITE NAME TABLE BEFORE BEING WRITTEN TO VRAM. THIS AIDS
	641	* SPRITE MULTIPLEXING SOLUTIONS TO THE FIFTH SPRITE PROBLEM.
	642	
73C8	643	GLB RAND_NUM RAND_NUM
	644	DEFMS 2
	645	* THIS IS THE SHIFT REGISTER USED BY THE RANDOM NUMBER GENERATOR.
	646	* IT IS INITIALIZED AT POWER-UP.
	647	GLB PARAM
	648	PROG
	649	PARAM
0098	650	HEX E1,E3,E5,0A,6F,03,0A,03,67,E3,D5,5E,23,56,23,E5
0098	E1E3E50A6F	
0090	030A0367E3	
00A2	D5E235623	
00A7	E5	
00A8	78B2C2B700	
00AD	E15E235623	
00B2	E5E8E2356	
00B7	03	
00B8	0A07D2A00	
00B0	03E1E37323	
00C2	722301E328	
00C7	AF	
00C8	BCC2D00080	
00CD	CAD600E3E5	
00D2	EBC3A300E1	
00D7	EB	
00D8	E3E9E1E3E5	
00D0	0F67080A6F	
00E2	E303031A77	
00E7	23	
00E8	13E32BAFB0	
00ED	C2F400BCCA	
00F2	F800E3C3E5	
00F7	00	
00F8	E1C3C400	
	656	HEX E1,C3,C4,00
	657	PROG
	652	HEX 0A,07,D2,DA,00,03,E1,E3,73,23,72,23,D1,E3,28,AF
	651	HEX 7B,B2,C2,87,00,E1,5E,23,56,23,E5,EB,5E,23,56,03
	653	HEX BC,C2,D0,00,8D,CA,D6,00,E3,E5,EB,C3,A3,00,E1,EB
	654	HEX E3,E9,E1,E3,E5,0F,67,08,0A,6F,E3,03,03,1A,77,23
	655	HEX 13,E3,28,AF,8D,C2,F4,00,BC,CA,F8,00,E3,C3,E5,00

LOCATION OBJECT CODE LINE SOURCE LINE

```

659 ; .IDENT FREQSWE ;includes FREQ_SWEEP
660 ;*****
661 ; FREQ_SWEEP *
662 ;*****
663 ;.COMMENT )
664 ;See Users' Manual for description
665 ;RETS Z SET: if note over
666 ;RETS Z RESET: if sweep in progress or note not over
667 ;)
668
669 GLB FREQ_SWEEP
670 ;EXT DECLSN,MSNTOLSN,DECSM,ADD816
671 ;INCLUDE OSSR_EQU:OS:0 ;equates
672 FREQ_SWEEP
673 ; * if freq not swept, dec MLEN and RET [setting Z flag]
674 LD A,(IX+FSTEP) ;check for no sweep code
675 CP 0 ;SET Z flag if FSTEP = 0
676 IF [PSW,IS,ZERO] ;note not to be swept
677 JR NZ,L20
678 LD A,(IX+MLEN) ;dec MLEN and
679 DEC A ;SET Z flag if MLEN = 0
680 RET Z ;leave if note over with Z flag SET
681 LD (IX+MLEN),A ;store decremented MLEN
682 RET ;RET with Z flag RESET [note not over]
683 ENDF
684 ; * sweep going, so dec FPSV
685 PUSH IX ;point HL to FPSV
686 POP HL
687 LD E,FPSV
688 LD D,0
689 ADD HL,DE
690 CALL DECLSN
691 IF [PSW,IS,ZERO] ;dec FPSV
692 JR NZ,L21 ;FPSV has timed out
693 * dec MLEN and leave if sweep is over
694 CALL MSNTOLSN ;reload FPSV from FPS
695 DEC HL ;point to MLEN [# steps in the sweep]
696 LD A,(HL) ;dec MLEN and
697 DEC A ;SET Z flag if MLEN = 0
698 RET Z ;leave if sweep over with Z flag SET
699 * sweep not over, so add FSTEP to FREQ
700 LD (HL),A ;store decremented MLEN
701 DEC HL ;point HL
702 DEC HL ;to FREQ
703 LD A,(IX+FSTEP) ;A = FSTEP [two's complement step size]
704 CALL ADD816 ;FREQ = FREQ + FSTEP
705 INC HL ;point HL to hi FREQ
706 RES 2,(HL) ;RESET B2 in hi FREQ in case add caused > 10 bit FREQ
707 OR OFFH ;RESET Z flag, sweep not over yet
708 ENDF
709 RET
710 END ;FREQSWE

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

712 ; .IDENT ATMSWEE
713 ; ***** ;Includes ATM_SWEEP
714 ; *
715 ; * ATM_SWEEP
716 ; *****
717 ; * COMMENT )
718 ; * See User's Manual for description
719 ; * RETs Z SET: if byte 8 is 0 [means sweep is over, or note was never swept]
720 ; * RETs Z RESET: if sweep in progress
721 ;
722 ; GL8 ATM_SWEEP
723 ; EXT DECLSM,DECSM,MSNTOLSM
724 ; INCLUDE OSSR_EQU:OS:0 ;equates
725 ;
726 ; * RET with Z SET if byte 8 = 00
727 ; LD A, (IX+8) ;check byte 8 for no sweep code
728 ; CP 0 ;Z is set if byte 8 = 0
729 ; RET Z ;leave if Z set, sweep not going
730 ; * sweep going, so dec APSV ;point HL to APSV
731 ; PUSH IX
732 ; POP HL
733 ; LD D,0
734 ; LD E,APSV
735 ; ADD HL,DE
736 ; CALL DECLSM ;dec APSV [LSM of byte 9]
737 ; IF (PSW,IS,ZERO) ;APSV has timed out
738 ; JR NZ,L22
739 ; * dec ALEN to see if sweep over
740 ; CALL MSNTOLSM ;reload APSV from APS
741 ; DEC HL ;point to ALEN [# of steps in the sweep]
742 ; CALL DECLSM ;dec ALEN [LSM byte 8]
743 ; IF (PSW,IS,NZERO) ;sweep not over yet
744 ; JR Z,L23
745 ; * add ASTEP to ATM
746 ; LD A,(HL)
747 ; AND OFOH
748 ; LD E,A
749 ; DEC HL
750 ; DEC HL
751 ; DEC HL
752 ; LD A,(HL)
753 ; AND OFOH
754 ; ADD A,E
755 ; LD E,A
756 ; LD A,(HL)
757 ; AND OFH
758 ; OR E
759 ; LD (HL),A
760 ; OR OFFH
761 ; JR L22
762 ; ELSE
763 ; LD (HL),0
764 ;
765 ; ENDF
766 ; ENDF
767 ; RET
768 ; END ;ATMSWEE

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

770 ; .IDENT UTIL
771 ; Includes UPATCTRL,UPFREQ,
772 ;DECLSN,DECSM,MSNTOLSM,ADD816,PT_IX_TO_SKDATA,
773 ;LEAVE_EFFECT,AREA_SONG_1S
774 ;*****
775 ; UPATCTRL *
776 ;.COMMENT )
777 ;Perform single byte update of the snd chip noise control register or any
778 ;attenuation register. IX is passed pointing to byte 0 of a song data area, MSN
779 ;register C = formatted channel attenuation code.
780 ;)
781 GLB UPATCTRL
782 ;INCLUDE OSSR_EQU:OS:0 ;equates
783 UPATCTRL
784 LD A, [IX+4] ;MSN A = ATM, LSM may be CTRL data
785 BIT 4,C ;test for ATM
786 ; IF (PSM,IS,NZERO) ;ATM is to be sent, move it to the LSM
787 JR Z,L24 ;swap nibbles
788 RRCA
789 RRCA
790 RRCA
791 RRCA ;LSN A = ATM
792 ; ENDIF
793 L24 AND OFH ;mask MSN
794 OR C ;A = formatted register# | ATM or CTRL
795 OUT [SOUND_PORT],A ;output ATM or CTRL data
796 RET
797 ;*****
798 ; UPFREQ *
799 ;*****
800 ;.COMMENT )
801 ;Perform double byte update of a sound chip frequency register. IX is passed
802 ;pointing to byte0 of a song data area, MSN register D = formatted channel
803 ;frequency code.
804 ;)
805 GLB UPFREQ
806 UPFREQ
807 LD A, [IX+FREQ]
808 AND OFH
809 OR D
810 OUT [SOUND_PORT],A
811 LD A, [IX+FREQ]
812 AND OFH
813 LD D,A
814 LD A, [IX+FREQ+1]
815 AND OFH
816 OR D
817 RRCA
818 RRCA
819 RRCA
820 RRCA
821 OUT [SOUND_PORT],A
822 RET
823 ;*****
824 ; DECLSN *
825 ;*****
826 ;.COMMENT )

```

```

0164
0164 D07E04
0167 CB61

```

```

0169 2804
0168 0F
016C 0F
0160 0F
016E 0F
016F E60F
0171 B1
0172 D3FF
0174 C9

```

```

0175 D07E03
0178 E60F
017A B2
0178 D3FF
0170 D07E03
0180 E6F0
0182 57
0183 D07E04
0186 E60F
0188 B2
0189 0F
0188 0F
018C 0F
018D D3FF
018F C9

```

```

;A = F2 F3 F4 F5 F6 F7 F8 F9
;A = 0 0 0 0 F6 F7 F8 F9
;A = formatted reg# | F6 F7 F8 F9
;output first freq byte
;A = F2 F3 F4 F5 F6 F7 F8 F9 again
;A = F2 F3 F4 F5 0 0 0 0
;LSN A = 0 0 F0 F1
;A = 0 0 0 0 0 F0 F1
;A = F2 F3 F4 F5 0 0 F0 F1
;swap nibbles
;A = 0 0 F0 F1 F2 F3 F4 F5
;output 2nd [most significant] freq byte

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
827 ;Without affecting the MSN, decrement the LSM of the byte pointed to by HL. HL
828 ;remains the same.
829 ;RET with Z flag set if dec LSM results in 0, reset otherwise.
830 ;RET with C flag set if dec LSM results in -1, reset otherwise.
831 ;)
832 GLB DECLSM
833 DECLSM LD A,0
834 RRD
835 SUB 1
836 PUSH AF
837 RLD
838 POP AF
839 RET
840 ;*****
841 ;* DECSM *
842 ;*****
843 ;.COMMENT )
844 ;Without affecting the LSM, decrement the MSN of the byte pointed to by HL. HL
845 ;remains the same.
846 ;RET with Z flag set if dec MSN results in 0, reset otherwise.
847 ;RET with C flag set if dec MSN results in -1, reset otherwise.
848 ;)
849 GLB DECSM
850 DECSM LD A,0
851 RLD
852 SUB 1
853 PUSH AF
854 RRD
855 POP AF
856 RET
857 ;*****
858 ;* MSNTOLSM *
859 ;*****
860 ;.COMMENT )
861 ;Copy MSN of the byte pointed to by HL to the LSM of that byte. HL remains
862 ;the same.
863 ;)
864 GLB MSNTOLSM
865 MSNTOLSM
866 LD A,[HL]
867 AND 0F0H
868 LD B,A
869 RRCA
870 RRCA
871 RRCA
872 RRCA
873 OR B
874 LD [HL],A
875 RET
876 ;*****
877 ;* ADD816 *
878 ;*****
879 ;.COMMENT )
880 ;Adds 8 bit two's complement signed value passed in A to the 16 bit location
881 ;pointed to by HL.
882 ;)
883 GLB ADD816

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
01B1 0600 854 LD B,0
01B3 C87F 885 BIT 7,A
01B5 2802 886 JR Z,POS
01B7 06FF 887 LD B,OFFH
01B9 86 888 ADD A,[HL]
01BA 77 889 LD [HL],A
01BB 23 890 INC HL
01BC 7E 891 LD A,[HL]
01BD 88 892 ADC A,B
01BE 77 893 LD [HL],A
01BF 28 894 DEC HL
01C0 C9 895 RET
*****
897 ; * PT IX TO SxDATA *
898 ; *****
899 ; .COMMENT )
900 ;SONGMO passed in B.
901 ;Point IX to byte 0 in SONGMO's song data area.
902 ;RET with both DE and IX pointing to SxDATA.
903 ;HL pointing to MSB SxDATA entry in LST_OF_SMD_ADDRS.
904 ; )
*****
01C1 2A7020 905 GLB PT IX TO SxDATA
01C4 28 906 PT IX TO SxDATA
01C5 28 907 ; * IX & DE := addr of byte 0 in SONGMO's song data area,
01C6 48 908 ; HL pointing to MSB SxDATA entry in LST_OF_SMD_ADDRS.
01C7 0600 909 LD HL,[PTR_TO_LST_OF_SMD_ADDRS] ;point HL to start LST_OF_SMD_ADDRS
01C9 C801 910 DEC HL
01CB C801 911 DEC HL
01CD 09 912 LD C,B
01CE 5E 913 LD B,0
01CF 23 914 RLC C
01D0 05 915 RLC C
01D1 05 916 ADD HL,BC
01D2 D0E1 917 LD E,[HL]
01D4 C9 918 INC HL
919 LD D,[HL]
920 PUSH DE
921 POP IX
922 RET
923 ;
924 ; LEAVE EFFECT
925 ; 4/19/82
926 ;
927 ; *****
928 ; * LEAVE EFFECT *
929 ; *****
930 ; .COMMENT )
931 ;LEAVE_EFFECT, called by a special sound effect routine when it's finished,
932 ;restores the SONGMO of the song to which the effect note belongs to B5 - 80 of
933 ;byte 0 in the effect's data area, and loads bytes 1 and 2 with the address of
934 ;the next note in the song. The address of the 1 byte SONGMO (saved by the
935 ;effect when first called) is passed in DE. The 2 byte address of the next note
936 ;in the song, also saved by the effect, is passed in HL. IX is assumed to be
937 ;pointing to byte 0 of the data area to which the song number is to be restored.
938 ;bits 7 and 6 of the saved SONGMO are ignored, and therefore may be used by the
939 ;effect to store flag information during the course of the note.
940 ; )

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
101D5          GLB LEAVE_EFFECT
101D5 007501 942 LEAVE_EFFECT
101D8 007402 943 LD [IX+1],L
101B 1A 944 LD [IX+2],H
101C E63F 945 LD A,[DE]
101E 47 946 AND 03FH
101F 007E00 947 LD B,A
101E2 E6C0 948 LD A,[IX+0]
101E4 80 949 AND 0C0H
101E5 007700 950 OR B
101E8 C9 951 LD [IX+0],A
952 RET
953
954 ;*****
955 ;* AREA SONG_IS *
956 ;*****
957 ;.COMMENT )
958 ;The address of byte 0 of a song data area is passed in IX. The song number of
959 ;the song using that area is returned in A [0FFH if inactive]. If a special
960 ;effect was using that area, 62 is returned in A and HL is returned with the
961 ;address of the special sound effect routine.
962 ;)
963
963          GLB AREA_SONG_IS
964 AREA_SONG_IS
965 LD A,[IX+0]
966 CP 0FFH
967 RET Z
968 AND 00111111B
969 CP 62
970 RET NZ
971 ; special effect, so set HL to addr effect, stored in bytes 1&2
972 PUSH IX
973 POP HL
974 INC HL
975 LD E,[HL]
976 INC HL
977 LD D,[HL]
978 EX DE,HL
979 RET
980 ;
981 PROG
981          END ;UTIL

101E9          GLB AREA_SONG_IS
101E9 007E00 966 AREA_SONG_IS
101EC FEFF 965 LD A,[IX+0]
101EE CB 966 CP 0FFH
101EF E63F 967 RET Z
101F1 FE3E 968 AND 00111111B
101F3 C0 969 CP 62
101F4 D0E5 970 RET NZ
101F6 E1 971 ; special effect, so set HL to addr effect, stored in bytes 1&2
101F7 23 972 PUSH IX
101F8 5E 973 POP HL
101F9 23 974 INC HL
101FA 56 975 LD E,[HL]
101FB EB 976 INC HL
101FC C9 977 LD D,[HL]
978 EX DE,HL
979 RET
980 ;
981          END ;UTIL

;LSB NEXT NOTE PTR := LSB addr next note in song
;MSB NEXT_NOTE_PTR := MSB addr next note in song
;A := x x SONGMO (i.e., the saved, original SONGMO)
;A := 0 0 SONGMO
;B := 0 0 SONGMO (B now = original SONGMO)
;A := CH# | 62 (all effect notes have SONGMO = 62)
;A := CH# | 0 0 0 0
;A := CH# | SONGMO
;restore the song number

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
983 ; .IDENT INIT_SOU ;includes INIT_SOUND,ALL_OFF
984 ; *****
985 ; * INIT_SOUND *
986 ; *****
987 ; .COMMENT )
988 ; See Users' Manual for description; includes ENTRY POINT ALL_OFF
989 ; addr LST OF_SND_ADDRS passed in HL
990 ; n = # of song data areas to init, passed in B
991 ; )
992 GLB INIT_SOUND,ALL_OFF,DUMAREA
993 ;INCLUDE OSSR EQU:OS ;equates
994 ; *** Sound chip register code EQUATES
995 ; Tone generator frequency and attenuation formatted register codes
996 <0080> EQU SR1FRQ EQU 100000008 ;BIT7 = 1, BIT6-4 = TONE GEN 1 FREQ CODE
997 <0090> EQU SR1ATN EQU 100100008 ;BIT7 = 1, BIT6-4 = TONE GEN 1 ATTN CODE
998 <00A0> EQU SR2FRQ EQU 101000008 ;BIT7 = 1, BIT6-4 = TONE GEN 2 FREQ CODE
999 <00B0> EQU SR2ATN EQU 101100008 ;BIT7 = 1, BIT6-4 = TONE GEN 2 ATTN CODE
1000 <00C0> EQU SR3FRQ EQU 110000008 ;BIT7 = 1, BIT6-4 = TONE GEN 3 FREQ CODE
1001 <00D0> EQU SR3ATN EQU 110100008 ;BIT7 = 1, BIT6-4 = TONE GEN 3 ATTN CODE
1002 ; Noise generator control and attenuation formatted register codes
1003 <00E0> EQU SRMCTL EQU 111000008 ;BIT7 = 1, BIT6-4 = NOISE GEN CONTROL CODE
1004 <00F0> EQU SRMATN EQU 111100008 ;BIT7 = 1, BIT6-4 = NOISE GEN ATTN CODE
1005
1006 * PROCEDURE INIT_SOUNDQ (AREA_COUNT:BYTE;LST_OF_ADDR:INTEGER)
1007
1008 * THIS IS THE PASCAL ENTRY POINT TO INIT_SOUND
1009
1010 ;EXT PARAM
1011 GLB INIT_SOUNDQ
1012 INIT_SOUND_PAR: DEFM 2,1,2
1013
1014 ; COMM
1015 ;INIT_SOUND_DATA: DEFS 3 ;Moved to OS
1016
1017 PROG
1018 INIT_SOUNDQ:
1019 LD BC,INIT_SOUND_PAR
1020 LD DE,INIT_SOUND_DATA
1021 CALL PARAM
1022 LD A,[INIT_SOUND_DATA]
1023 LD B,A
1024 LD HL,[INIT_SOUND_DATA+1]
1025
1026 INIT_SOUND
1027 ; * initialize PTR TO LST OF_SND_ADDRS with value passed in HL
1028 LD (PTR TO LST OF_SND_ADDRS),HL
1029 ; * store inactive code at byte 0 of each of the n data areas (B = n)
1030 INC HL ;pt HL to song 1 data area entry in LST_OF_SND_ADDRS
1031 INC HL ;pt DE to byte 0 in first song data area
1032 LD E,[HL] ;pt HL to byte 0 in first song data area
1033 INC HL ;set DE for 10 byte increment
1034 LD D,[HL]
1035 EX DE,HL
1036 LD E,10
1037 LD D,0
1038 B1: LD (HL),0FFH ;deactivate area

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0222 19 ADD HL,DE
0223 10FB DJNZ B1
1041 ; * store end of data area code (0) at first byte after last song data area
0225 3600 LD (HL),0
1043 ; * set the 4 channel data area pointers to a dummy, inactive data area
0227 21024C LD HL,DUMAREA
022A 227022 LD (PTR TO S_ON_0),HL
022D 227024 LD (PTR TO S_ON_1),HL
0230 227026 LD (PTR TO S_ON_2),HL
0233 227028 LD (PTR TO S_ON_3),HL
1049 ; * initialize SAVE_CTRL
0236 3EFF LD A,OFFH
0238 32702A LD (SAVE_CTRL),A
0238 ALL_OFF
1053 ; * turn off all 4 sound generators
0238 3E9F LD A,OFF+SR1ATM
0230 D3FF OUT [SOUND_PORT],A
023F 3EBF LD A,OFF+SR2ATM
0241 D3FF OUT [SOUND_PORT],A
0243 3EDF LD A,OFF+SR3ATM
0245 D3FF OUT [SOUND_PORT],A
0247 3EFF LD A,OFF+SRMATM
0249 D3FF OUT [SOUND_PORT],A
0248 C9 RET
024C FF
1063 DUMAREA DEFB INACTIVE
1064 ; END ;INITSOU
1065 PROG
;pt HL to byte 0 next area (10 bytes away)
;do this for the n (passed in B) data areas
;store end of data area code (0) at first byte after last song data area
;store end of data area code in byte 0 data area n + 1
;set the 4 channel data area pointers to a dummy, inactive data area
;point HL to inactive byte below (after the RET)
;store addr DUMAREA at PTR TO S_ON_0
;store addr DUMAREA at PTR TO S_ON_1
;store addr DUMAREA at PTR TO S_ON_2
;store addr DUMAREA at PTR TO S_ON_3
;note: this is only time MSN SAVE_CTRL will be non zero,
;thus ensuring PLAY_SONGS will output 1st real CTRL data

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
1067 ; .IDENT JUKEBOX ;includes JUKE_BOX
1068 ;*****
1069 ;* JUKE_BOX
1070 ;*****
1071 ;.COMMENT )
1072 ;see Users' Manual for description
1073 ;SONGNO passed in B
1074 ;)
1075 GLB JUKE_BOX
1076 ;EXT PT IX TO SxDATA,LOAD_NEXT_NOTE,UP_CH_DATA_PTRS
1077 ;INCLUDE OSSR_EQU:OS ;equates
1078
1079 * PROCEDURE JUKE_BOXQ (SONG_NUM:BYTE)
1080
1081 * THIS IS THE PASCAL ENTRY POINT FOR JUKE_BOX
1082
1083 JUKE_BOX PAR DEFN 1,1
1084 * THE PARAMETER DESCRIPTOR FOR JUKE_BOXQ
1085
1086 ;EXT PARAM_
1087 * THE PARAMETER PASSING ROUTINE
1088
1089 ;
1090 ;PARAM AREA DEFS 1
1091 * THIS IS WHERE THE SONG NUMBER IS ACTUALLY PASSED
1092
1093
1094
1095 JUKE_BOXQ JUKE_BOXQ
1096
1097 LD BC,JUKE_BOX PAR
1098 LD DE,PARAM_AREA
1099 CALL PARAM
1100 LD A,[PARAM_AREA]
1101 LD B,A
1102
1103 ;* RET if song already in progress
1104 PUSH BC ;save SONGNO on stack
1105 CALL PT_IX_TO_SxDATA ;point IX to SONGNO's song data area
1106 LD A,[IX+0] ;A := CH# [if any] | SONGNO [if any]
1107 AND 3FH ;A := 0 SONGNO
1108 POP BC ;B := SONGNO
1109 CP B ;test if already in progress
1110 RET Z ;if so, leave
1111 ;* Load first note and set NEXT NOTE PTR (thru LOAD_NEXT_NOTE)
1112 LD [IX+0],B ;store SONGNO in byte 0
1113 DEC HL ;HL left by PT_IX TO SxDATA pointing to MSB SxDATA
1114 DEC HL ;entry in LST_OF_SMD_ADDR; point HL to note list
1115 LD D,[HL] ;-starting addr entry in LST_OF_SMD_ADDR and save this
1116 DEC HL ;-addr in DE
1117 LD E,[HL] ;DE now has the initial value for NEXT_NOTE_PTR
1118 LD [IX+1],E ;set NEXT_NOTE_PTR for first note in song
1119 LD [IX+2],D
1120 CALL LOAD_NEXT_NOTE ;load note, byte 0 := CH#SONGNO, set new NEXT_NOTE_PTR
1121 CALL UP_CH_DATA_PTRS ;new song, so update channel data ptrs
1122 RET
1123 ;
END ;JUKEBOX

```

CONFIDENTIAL

HEMLETT-PACKARD: JUKEBOX (c) Coleco, 1982

FILE: OS_7PRIME:POS

LOCATION OBJECT CODE LINE SOURCE LINE

1124 ; [page]

1125 PROG

```

LOCATION OBJECT CODE LINE SOURCE LINE
1127 ; .IDENT SMDMAN ;includes SMD_MANAGER,PROCESS_DATA_AREA,
1128 ;***** ;UP_CH_DATA_PTRS
1129 ; SMD_MANAGER *
1130 ; SMD_MANAGER *
1131 ;*****
1132 ;.COMMENT )
1133 ;See Users' Manual for description
1134 ;)
1135 GLB SMD_MANAGER
1136 ;EXT PT IX TO SxDATA,AREA_SONG_IS
1137 ;INCLUDE OSSR_EQU:OS ;equates
027F 1138 SMD_MANAGER
1139 ; * IX := addr of song #1 data area [S1DATA]
1140 LD B,1 ;pt IX to byte 0 song data area for song # 1
1141 CALL PT IX TO SxDATA
1142 ; LOOP until end of song data areas
1143 LD A,ENDSDATA ;check for end of song data areas
1144 CP [IX+0] ;set Z flag if inactive
1145 RET Z ;leave [Z set], if all data areas have been processed
1146 ; * process active song data areas
1147 CALL PROCESS_DATA_AREA ;update counters or call effect; get next note
1148 ; * point IX to byte 0 next song data area
1149 LD E,10
1150 LD D,0
1151 ADD IX,DE
1152 JR L1 ;REPEAT LOOP
1153 ;*****
1154 ; UP_CH_DATA_PTRS *
1155 ;*****
1156 ;.COMMENT )
1157 ;for each active data area, starting with S1DATA and proceeding in order, load
1158 ;the associated channel data area pointer [PTR_TO_S_ON_X] with the address of
1159 ;byte 0. This routine is called by JUKE_BOX, when a song starts and
1160 ;PROCESS_DATA_AREA when the channel using a data area has changed as a result of
1161 ;calling LOAD_NEXT_NOTE [this happens when a song finishes and when it switches
1162 ;back and forth between noise and tone notes].
1163 ;)
1164 GLB UP_CH_DATA_PTRS
1165 ;EXT DUMAREA
0295 1166 UP_CH_DATA_PTRS
0295 D0E5 ;save current IX
0297 21024C ;set all 4 ch data ptrs to dummy inactive area
029A 227022 LD HL,DUMAREA
0290 227024 LD [PTR_TO_S_ON_0],HL
02A0 227026 LD [PTR_TO_S_ON_1],HL
02A3 227028 LD [PTR_TO_S_ON_2],HL
02A6 0601 LD [PTR_TO_S_ON_3],HL
02A8 C001C1 LD B,1
1174 CALL PT IX TO SxDATA ;set IX to byte 0 S1DATA
1175 ; LOOP until end of song data areas ;RETS with IX addr byte 0 song 1
1176 LD A,[IX+0] ;test for end of song data areas
1177 CP ENDSDATA
1178 JR Z,DONE_SMDMAN ;leave loop if all data areas checked
1179 ; * if area active, set appropriate channel data area pointer
1180 CP INACTIVE ;check for inactive data area: don't up date ptr if so
1181 IF [PSW,IS, NZERO] ;area is active: update channel data ptrs
1182 JR Z,L9
1183 LD A, [IX+0] ;get CH# in A

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0289 E6C0 1184 AND 0C0H
028B 07 1185 RLCA
028C 07 1186 RLCA
028D 07 1187 RLCA
028E 5F 1188 LD E,A
028F 1600 1189 LD D,0
02C1 217022 1190 LD HL, PTR_TO_S_ON_0
02C4 19 1191 ADD HL,DE
02C5 00E5 1192 PUSH IX
02C7 01 1193 POP DE
02C8 73 1194 LD [HL],E
02C9 23 1195 INC HL
02CA 72 1196 LD [HL],D
1197 ;
1198 ; * point IX to byte 0 next song data area
1199 L9 LD E,10
1200 LD D,0
1201 ADD IX,DE
1202 JR L2 ;REPEAT LOOP
1203 DONE_SNDMAN JR L2 ;restore IX
1204 POP IX
1205 RET
1206 ;
1207 ; * PROCESS DATA AREA *
1208 ;
1209 ;.COMMENT )
1210 ;See Users' Manual for description
1211 ;Terminology: SFX = address of sound effect routine
1212 ;)
1213 GLB PROCESS DATA AREA,EFXOVER
1214 ;EXT LOAD_NEXT_NOTE,ATN_SWEEP,FREQ_SWEEP
1215 PROCESS DATA AREA
1216 _CALL_AREA_SONG_IS ;return area's SONGNO in A [and addr SFX in HL]
1217 CP INACTIVE ;test for inactive code
1218 RET Z ;RET, no processing if area inactive
1219 ; * if special effect, call it to process the data area
1220 CP 62 ;test for special sound effect
1221 ; IF [PSW,IS,ZERO] ;data area used by sound effect
1222 JR NZ,L10 ;
1223 LD E,7 ;
1224 LD D,0 ;
1225 ADD HL,DE ;
1226 JP [HL] ;do 1 pass thru effect, RET from effect
1227 ;
1228 ; * else process a non-effect note
1229 L10 CALL ATN_SWEEP ;process atn sweep data, if any
1230 CALL FREQ_SWEEP ;proc frq sweep data, if any, & note dura timer s
1231 ; IF [PSW,IS,ZERO] ;note is over
1232 JR NZ,L12 ;
1233 EFXOVER LD A,[IX+0] ;A := CH# | SONGNO this note
1234 PUSH AF ;save on stack
1235 CALL LOAD_NEXT_NOTE ;load data for next note
1236 POP BC ;B := CH# | SONGNO previous note
1237 LD A,[IX+0] ;A := CH# | SONGNO new note [may be inactive]
1238 CP B ;check against new note's CH# | SONGNO
1239 ; IF [PSW,IS,NZERO] ;change to/from tone/efx/noise
1240 JR Z,L12 ;

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
02FC	CD0295	1241	CALL UP_CH_DATA_PTRS	;to maintain data area priority system
		1242	ENDIF	
02FF	C9	1243	ENDIF	
		1244	RET	
		1245	END	;SNDMAN
		1246	PROG	


```

LOCATION OBJECT CODE LINE SOURCE LINE
1248 ; .IDENT PLAYSON ; Includes TONE_OUT
1249 ; *****
1250 ; PLAY SONGS *****
1251 ; *****
1252 ; .COMMENT )
1253 ; see Users' Manual for description
1254 ; )
1255 ; GLB PLAY SONGS, TONE_OUT
1256 ; .EXT UPATCTRL_UPFREQ
1257 ; INCLUDE OSSR EQU:OS equates
1258 ; *** sound chip register code EQUATES
1259 ; Tone generator frequency and attenuation formatted register codes
1260 ; SR1FRQ EQU 100000008 ;BIT7 = 1, BIT6-4 = TONE GEN 1 FREQ CODE
1261 ; SR1ATM EQU 100100008 ;BIT7 = 1, BIT6-4 = TONE GEN 1 ATTN CODE
1262 ; SR2FRQ EQU 101000008 ;BIT7 = 1, BIT6-4 = TONE GEN 2 FREQ CODE
1263 ; SR2ATM EQU 101100008 ;BIT7 = 1, BIT6-4 = TONE GEN 2 ATTN CODE
1264 ; SR3FRQ EQU 110000008 ;BIT7 = 1, BIT6-4 = TONE GEN 3 FREQ CODE
1265 ; SR3ATM EQU 110100008 ;BIT7 = 1, BIT6-4 = TONE GEN 3 ATTN CODE
1266 ; Noise generator control and attenuation formatted register codes
1267 ; SRNCTL EQU 111000008 ;BIT7 = 1, BIT6-4 = NOISE GEN CONTROL CODE
1268 ; SRNATM EQU 111100008 ;BIT7 = 1, BIT6-4 = NOISE GEN ATTN CODE
1269 ; Noise generator formatted control codes
1270 ; WHITE EQU 000001008 ;BIT2 = 1, WHITE NOISE CODE
1271 ; PER100 EQU 000000008 ;BIT2 = 0, PERIODIC NOISE CODE
1272 ; MSKHI EQU 000000008 ;BIT0-1 SET FOR HIGHEST NOISE SHIFT RATE (M/512)
1273 ; MSKMD EQU 000000018 ;BIT0-1 SET FOR MEDIUM NOISE SHIFT RATE (M/1024)
1274 ; MSKLO EQU 000000018 ;BIT0-1 SET FOR LOWEST NOISE SHIFT RATE (M/2048)
1275 ; MSRTG3 EQU 00000001B ;BIT0-1 SET FOR SHIFT FROM TONE GEN 3 OUTPUT
1276 ; PLAY_SONGS
1277 ; * output CH1 attenuation and frequency
1278 ; LD A,OFF+SR1ATM ;format CH1 OFF byte into A
1279 ; LD C,SR1ATM ;format MSN C for CH1 attenuation
1280 ; LD D,SR1FRQ ;format MSN D for CH1 frequency
1281 ; LD IX,(PTR TO S_OM_1) ;point IX to byte 0 data area of song for CH1
1282 ; CALL TONE_OUT
1283 ; * output CH2 attenuation and frequency
1284 ; LD A,OFF+SR2ATM ;format CH2 OFF byte into A
1285 ; LD C,SR2ATM ;format MSN C for CH2 attenuation
1286 ; LD D,SR2FRQ ;format MSN D for CH2 frequency
1287 ; LD IX,(PTR TO S_OM_2) ;point IX to byte 0 data area of song for CH2
1288 ; CALL TONE_OUT
1289 ; * output CH3 attenuation and frequency
1290 ; LD A,OFF+SR3ATM ;format CH3 OFF byte into A
1291 ; LD C,SR3ATM ;format MSN C for CH3 attenuation
1292 ; LD D,SR3FRQ ;format MSN D for CH3 frequency
1293 ; LD IX,(PTR TO S_OM_3) ;point IX to byte 0 data area of song for CH3
1294 ; CALL TONE_OUT
1295 ; * output CH0 [noise] ATM [and CTRL, if different from last time]
1296 ; LD A,OFF+SRNATM ;format CH0 OFF byte into A
1297 ; LD C,SRNATM ;format MSN C for CH0 attenuation
1298 ; LD IX,(PTR TO S_OM_0) ;point IX to byte 0 data area of song for CH0
1299 ; LD E,(IX+0) ;look for inactive code, OFFH
1300 ; INC E ;this sets Z flag if E = OFFH
1301 ; IF (PSW,IS,ZERO) ; song data area is inactive
1302 ; JR NZ,L5 ; turn off CH0
1303 ; OUT (SOUND_PORT),A
1304 ; JR L6

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0339 CD0164 1305 ; ELSE
033C DD7E04 1306 L5 CALL UPATNCTRL ;send out current ATM
033F E60F 1307 LD A, [IX+CTRL] ;LSN A = current CTRL data
0341 21702A 1308 AND OFH ;mask MSN
0344 BE 1309 LD HL,SAVE_CTRL ;point to last CTRL data sent
0345 2806 1310 CP [HL] ;compare
0347 77 1311 IF [PSW,IS, NZERO] ;CTRL has changed
0348 0EE0 1312 JR Z,L6 ;SAVE_CTRL = new CTRL data
034A CD0164 1313 LD [HL],A ;send new CTRL data
1314 LD C,SRNCTL
1315 CALL UPATNCTRL
1316 ;
1317 ; ENDF
1318 L6 RET
034E 034E 1319 TONE_OUT
034E DD5E00 1320 LD E, [IX+0] ;look for inactive code, OFFH
0351 1C 1321 INC E ;this sets Z flag if E = OFFH
0352 2004 1322 ; IF [PSW,IS,ZERO] ;song data area is inactive
0354 03FF 1323 JR NZ,L7
0356 1806 1324 OUT [SOUND_PORT],A
1325 JR L8
1326 ; ELSE
0358 CD0164 1327 L7 CALL UPATNCTRL ;send out current ATM and FREQ
035B CD0175 1328 CALL UPFREQ ;send out attenuation
1329 ; ENDF ;send out frequency
1330 L8 RET
1331 ; END :PLAYSON
1332 PROG

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
03A1 C30461 1391 JP MOD80 ;to load byte 0
1392 ;
1393 ; ENDF
03A4 E63C 1394 L14 - test for special sound effect
03A6 FE04 1395 AMD 001111008 ;mask irrelevant bits
1396 ; CP 000001008 ;test for B5 - B2 = 0001
03A8 2028 1397 IF [PSW,IS,ZERO] ;note is a special effect
1398 ; JR NZ,L15
1399 EFFECT ;--CASE-- special effect
03AA FDE1 1400 POP IY ;IY := SONGNO
03AC FDE5 1401 PUSH IY ;put SONGNO back on stack
03AE C5 1402 PUSH BC ;save header on stack; NEXT_NOTE_PTR := SFX, DE := SFX
03AF 23 1403 INC HL ;-pt HL to next byte [LSB addr SFX]
03B0 5E 1404 LD E,[HL] ;E := LSB SFX
03B2 23 1405 LD [IX+1],E ;-put LSB of SFX in byte 1 of SxDATA [NEXT_NOTE_PTR]
03B4 56 1406 INC HL ;-D := MSB SFX
03B6 D07202 1407 LD D,[HL] ;-put MSB SFX in byte 2 of SxDATA
03B8 23 1408 INC HL ;point HL to next note [after this new note]
03BA FDE5 1409 PUSH IY ;A := SONGNO
03BC F1 1410 POP AF
03BD 05 1411 PUSH DE
03BE FDE1 1412 POP IY
03C0 1103C6 1413 LD DE,PASS1
03C2 05 1414 PUSH DE
03C4 FDE9 1415 JP [IY] ;create "CALL [IY]" with RET to PASS1 by storing
03C6 1600 1416 PASS1 ;PASS1 on the stack
03C8 1E07 1417 LD E,7 ;1st 7 bytes SFX will save addr next note & SONGNO
03CA FD19 1418 ADD IY,DE ;in same fashion, create a "CALL (IY+7)"
03CC 110461 1419 LD DE,MOD80 ;to allow SFX to load initial values
03CF 05 1420 PUSH DE ;RET to MOD80
03D0 FDE9 1421 JP [IY]
1422 ; ENDF
1423 ; - if here, note is type 0 - 3
03D2 C5 1424 L15 ;save header on stack
03D3 78 1425 LD A,B ;A := fresh copy header
03D4 E603 1426 AMD 000000118 ;mask all but type number
03D6 FE00 1427 CP 0 ;test for type 0
1428 ; IF [PSW,IS,ZERO] ;note is type 0: fixed freq and atn
1429 ; JR NZ,L16
1430 ; --CASE-- note type 0
1431 ; * set up NEXT_NOTE_PTR
03D8 23 1432 TYPE0 INC HL ;next note [after this new note] is 4 bytes away,
03D9 23 1433 INC HL ;point HL to it
03DA 23 1434 INC HL
03DB 23 1435 INC HL
03DD 007501 1436 LD [IX+1],L ;put addr in NEXT_NOTE_PTR
03DE D07402 1437 LD [IX+2],H ;* move new note data and fill in bytes where necessary
1438 ; DEC HL ;point HL back to 1st ROM data to move, MLEN
1439 2B 1440 LD DE,05 ;point DE to destination: bytes 5, 4, and 3
03E5 110005 1441 CALL DE,TO_DEST ;move 3 bytes
03E8 CD0478 1442 LD BC,3
03EA 010003 1443 LDDR ;set for no freq sweep
03EE ED88 1444 LD [IX+FASTEP],0 ;set for no atn sweep
03F0 DD360700 1445 LD [IX+ASTEP],0
03F4 DD360800 1446 JR MOD80
03F8 1867 1447 ; ENDF

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
03FA FE01 1448 L16 CP 1
1449 ; ;test for type 1
03FC 2018 1450 ; IF (PSM,IS,ZERO)
; JR NZ,L17
;note is type 1: swept freq, fixed attenuation
1451 ; --CASE-- note type 1
1452 ; * set up NEXT_NOTE_PTR
03FE 1E06 1453 TYPE1 LD E,6
;note after this note is 6 bytes away,
0400 1600 1454 LD D,0 ;pt HL to it
0402 19 1455 ADD HL,DE
;store in NEXT_NOTE_PTR
0403 D07501 1456 LD [IX+1],L
0406 D07402 1457 LD [IX+2],H
* move new note data and fill in bytes where necessary
0409 28 1458 ; DEC HL
;point HL back to 1st ROM data to move, FSTEP
040A 1C 1459 INC E ;E := 7; point DE to destination: bytes 7 - 3
0408 CD0478 1461 CALL DE,TO_DEST
;move 5 bytes
040E 010005 1462 LD BC,5
0411 ED88 1463 LDDR
;set for no atn sweep
0413 D0360800 1464 LD [IX+ASTEP],0
0417 1848 1465 JR MOD80
ENDIF
0419 FE02 1466 ;
041B 2028 1467 L17 CP 2
;test for type 2
;note is type 2: fixed freq, swept attenuation
1468 ;
1469 ;
1470 ; --CASE-- note type 2
041D 1E06 1471 ; LD E,6
;pt HL to note after this note; since it's 6 bytes away,
041F 1600 1472 TYPE2 LD D,0 ;pt HL to it by adding 6
0421 19 1473 ADD HL,DE
0422 F1 1474 POP AF
;A := header this note [CH# | SONGNO]
0423 F5 1475 PUSH AF ;put back on stack
0424 E6C0 1476 AND 11000000H ;mask SONGNO, leaving CH#
;this is a noise note, which is only 5 ROM bytes long
0426 2001 1477 IF (PSM,IS,ZERO)
0428 28 1478 ; JR NZ,L18
;so move HL back 1 byte
1480 ; DEC HL
ENDIF
042C D07501 1481 ;
042C D07402 1482 L18 LD [IX+1],L
;put addr in NEXT_NOTE_PTR
1483 ; LD [IX+2],H
* move new note data and fill in bytes where necessary
042F 28 1484 ; DEC HL
;point HL back to 1st ROM data to move, APS
0430 1E09 1485 LD E,9 ;point DE to destination: bytes 9,8,5 - 3
0432 CD0478 1486 CALL DE,TO_DEST
;move 2 bytes
0435 010002 1487 LDDR
;when done, DE points to FSTEP, HL to ROM ILEN
0436 ED88 1488 LD A,0
043A 3E00 1489 LD [DE],A
;FSTEP := 0 for no freq sweep
043C 12 1491 DEC DE
;pt DE to RAM ILEN
0430 18 1492 DEC DE
043E 18 1493 LD C,3
;move last 3 ROM bytes; if this is a noise note, garbage
043F 0E03 1494 LDDR
;will be loaded into byte 3, but's that's OK
0441 ED88 1495 JR MOD80
0443 181C 1496 ;
ENDIF
1497 ;
1498 ; If here, note is type 3: swept freq, swept attenuation
1499 ; --CASE-- note type 3
1500 ; * set up NEXT_NOTE_PTR
0445 1E08 1501 TYPE3 LD E,8
;note after this note is 8 bytes away,
0447 1600 1502 LD D,0
0449 19 1503 ADD HL,DE
;pt HL to it
044A D07501 1504 LD [IX+1],L
;put addr in NEXT_NOTE_PTR

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0440 DD7402 1505 LD [IX+2],H
0450 Z8 1506 ; * move new note data and fill in bytes where necessary
0451 DDE5 1507 DEC HL ;point HL back to 1st ROM data to move, APS
0453 FDE1 1508 PUSH IX ;point DE to destination: bytes 9 - 3
0455 1E09 1509 POP IX ;IX := addr byte 0 (and DE = 6)
0457 FD19 1510 LD E,9 ;DE := 9
0459 FDE5 1511 ADD IX,DE ;IX := addr byte 9 (APS)
045B D1 1512 PUSH IX ;DE := addr APS
045C 010007 1513 POP DE ;move 7 bytes
045F EDB8 1514 LD BC,7
1515 LDDR
1516 ; * modify byte 0 basis header new note
1517 MODB0 ;pt HL to byte 0
1518 POP HL
1519 POP AF ;A := header new note
1520 POP BC ;B := SONGNO
1521 CP INACTIVE ;test for inactive (song over, as detected above)
1522 RET Z
1523 LD D,A ;save header in D
1524 AND 3FH ;rid channel bits
1525 CP 04 ;Special effect
1526 JR NZ,L20_LOAD_NEX
1527 LD B,62
1528 L20_LOAD_NEX:
1529 LD A,D ;restore A to header
1530 AND 0C0H ;A := CH# 0 0 0 0
1531 OR B ;A := new CH# | SONGNO
1532 LD [HL],A ;store back in byte 0
1533 ; ENDF
1534 L19 RET
1535 DE_TO_DEST
1536 PUSH IX
1537 POP IX
1538 ADD IX,DE ;IX := addr byte 0 (and DE = offset)
1539 PUSH IX ;IX := addr byte 0 + offset
1540 POP DE ;DE := addr of destination byte in SADATA
1541 RET
1542 ; ENDF ;LOADNEX
1543 PROG

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
1545 ; .IDENT ACTIVATE
1546 ; .ZOP
1547 ; .EPOP
1548 ; .COMMENT )
1549 ; ***** ACTIVATE *****
1550 ;
1551 ;
1552 ;
1553 ;
1554 ; THE FOLLOWING CHANGES/REVISIONS WERE MADE:
1555 ;
1556 ;
1557 ; 1. ELIMINATE CODE PLACING OLD.SCREEN ADDRESS IN STATUS AREA
1558 ; 2. INIT X.PAT.POS IN OLD.SCREEN WHEN IN VRAM AS WELL AS WHEN IN CRAM
1559 ; 3. USE VDP.MODE.WORD TO TEST GRAPHICS MODE
1560 ; 4. ADD CODE TO EXPAND ONE COLOR GENERATOR BYTE TO 8
1561 ; 5. ADDED C.BUFF DEFS 8 FOR COLOR EXPANDING CODE
1562 ; 6. FIX COLOR GEN MOVE IN MODE I
1563 ; 7. USE CONTROLLER_MAP FOR BUFFER AREA
1564 ;
1565 ; ACTIVATE is used to initialize the RAM status area for the passed
1566 ; object and move its pattern and color generators to the PATTERN and
1567 ; COLOR GENERATOR tables in VRAM. The second function is enabled or
1568 ; disabled by setting or resetting the carry flag in the PSU. This is
1569 ; necessary to prevent sending the same graphics data to VRAM more than
1570 ; once when creating identical objects. The calling sequence for act.
1571 ; activating an object is as follows:
1572 * LD HL,OBJ_n ;->OBJ TO ACTIVATE
1573 * SCF ;SIGNAL MV TO VRAM
1574 * CALL ACTIVATE
1575 *
1576 *
1577 *OR
1578 *
1579 * LD HL,OBJ_n ;->OBJ TO ACTIVATE
1580 * OR A ;DON'T MV TO VRAM
1581 * CALL ACTIVATE
1582 *
1583 *)
1584 ;EXT PUT VRAM_VRAM_WRITE,VDP_MODE_WORD
1585 ;EXT WORK_BUFFER
1586 ; GLB ACTIVATE_
1587 ;
1588 ;
1589 ;REGISTER USAGE: FOLLOWING WILL BE CHANGED BY ACTIVATE, ADDITIONAL
1590 ;MAY BE CHANGED BY CALLED SUBR
1591 ; AF,HL,DE,BC,IX
1592 ;
1593 ;
1594 ;
1595 ; PROCEDURE ACTIVATE(IVAR OBJ:OBJECT;MOVE:BOOLEAN);
1596 ;
1597 ; ACTIVATE IS THE PASCAL ENTRY POINT TO ACTIVATE
1598 ;
1599 ;EXT PARAM
1600 ; THE PASCAL PARAMETER PASSING PROCEDURE
1601 ; COMM

```

4/22/82
13:50:00

LOCATION OBJECT CODE LINE SOURCE LINE

```

1602
1603 ;PRM AREA: DEFS 3 ;Moved to OS
1604 ; THIS IS THE COMMON PARAMETER PASSING AREA
1605
1606
1607 ACTIVATE_P: DEFV 2,-2,1

```

```

0482 0002FFFE
0486 0001

```

```

1608
1609
1610 ACTIVATEQ
1611 EQU $
1612 BC,ACTIVATE_P
1613 DE,PRM_AREA
1614 CALL PARAM
1615 HL,[PRM_AREA]
1616 E,[HL]
1617 HL
1618 D,[HL]
1619 DE,HL
1620 A,[PRM_AREA+2]
1621 O
1622 Z,NTZZZ_
1623 TZZZ_
1624 NTZZZ_ OR A
1625 TZZZ_
1626

```

<04A3>

```

1627 ACTIVATE EQU $
1628 ;SUP POINTERS ETC COMMON TO ALL SUBCASES
1629 HL->OBJ DEF CROM
1630 C FLG=SUP VRAM FLG
1631 LD E,[HL]
1632 INC HL
1633 LD D,[HL]
1634 INC HL
1635 LD C,[HL]
1636 INC HL
1637 LD B,[HL]
1638 INC HL
1639 LD A,0
1640 [BC],A
1641 LD A,[DE]
1642 PUSH AF
1643 AND OFH
1644 JND Z,ACT_SEMI
1645 DEC A
1646 JND Z,ACT_MOBILE
1647 DEC A
1648 JND Z,ACT_DSPRT
1649 DEC A
1650 JND Z,ACT_1SPRT
1651 DEC A
1652 JND Z,ACT_CMLPX
1653 POP AF
1654 RET
1655 ;ON ENTRY TO SUBCASES:
1656 ; STACK=OBJ_TYPE & SUP VRAM FLG
1657 ; HL->OBJ_n+4

```

```

;--OBJ GEN CROM
;--OBJ CROM
;ZERO FRAME
;GET OBJ_TYPE
;SV OBJ_TYPE & FLG
;GET OBJ_TYPE NUM
;TYPE=0
;TYPE=1
;TYPE=2
;TYPE=3
;TYPE=4
;SUBCASE ELSE

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
1658 ; DE->OBJ GRAPHICS+0
1659 ; BC->OBJ STATUS+0
1660 ; A=0
1661 ;
<04C6> 1662 ACT_CPLX EQU $
1663 ; SUBCASE Complex
04C6 1A LD A,[DEI] ;GET COMP_CNT
04C7 1F RRA
04C8 1F RRA
04C9 1F RRA
04CA 1F RRA
04CB E60F AND
04CD 47 LD
04CE 5E LD
04CF 23 INC
04D0 56 LD
04D1 23 INC
04D2 B7 OR
04D3 2B10 JR Z,CPLX9
<04D5> 1677 CPLX4 $
04D5 F1 AF ;SUP CALL, COMP OBJ
04D6 F5 AF
04D7 E5 PUSH HL
04D8 C5 PUSH BC
04D9 EB EX DE,HL
04DA CD04A3 CALL ACTIVATE_
04D0 C1 POP BC
04DE E1 POP HL
04DF 5E LD E,[HL]
04E0 23 INC HL
04E1 56 LD D,[HL]
04E2 23 INC HL
04E3 10F0 DJNZ CPLX4
04E5 F1 POP AF
04E6 C9 RET ;TECHNICALLY SHOULD JMP TO RTM
1693 ;
<04E7> 1694 ACT_SEMI EQU $
1695 ; SUBCASE Semi_Mobile
04E7 CD0572 INIT_XP_OS
04EA 1A LD A,[DEI] ;X PAT POS := 80H
04EB 6F LD L,A ;A := FIRST_GEN_NAME
04EC 13 INC DE ;A := NUMGEN
04ED 1A LD A,[DEI]
04EE B5 ADD A,L ;NEXT GEN := FIRST_GEN_NAME + NUMGEN
04EF FD7705 [IY+5],A ;HL=FIRST_GEN_NAME
04F2 2600 LD H,0
1704 ;AT THIS POINT:
1705 ; STACK=OBJ TYPE & SUP VRAM FLG
1706 ; HL=FIRST_GEN_NAME
1707 ; DE->NUMGEN
1708 ; BC:FREE
1709 ; SUP FOR VRAM INIT
04F4 F1 POP AF
04F5 3038 JR NC,SEMI_EXIT
04F7 F5 PUSH AF
04F8 3A73C3 A,[VDP_MODE_WORD] ;SEE WHICH GRAPHICS MODE
04FB CB4F LD BIT 1,A ;IF GR II MODE

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
04FD 2831 JR Z,SEMI_GRI
04FF EB DE,HL
0500 44 LD B,H
0501 40 LD C,L
0502 6E LD L,(HL)
0503 2600 LD H,0
0505 E5 PUSH HL
0506 29 ADD HL,HL
0507 29 ADD HL,HL
0508 29 ADD HL,HL
0509 E5 PUSH HL
050A 03 TMC BC
050B 0A LD A,(BC)
050C 6F LD L,A
050D 03 TMC BC
050E 0A LD A,(BC)
050F 67 LD H,A
0510 C1 POP BC
0511 FDE1 TY IV
0513 F1 POP AF

0514 C87F BIT 7,A
0516 2803 JR Z,SEMI_MID
0518 CD0594 SUP_GEN_CLR
<051B> EQU $
051E C877 CALL SUP_UPDATE
0520 2803 JR Z,SEMI_BOT
0522 CD0594 SUP_GEN_CLR
<0525> EQU $
0525 CD05E8 SUP_UPDATE
0528 C86F JR Z,SEMI_EXIT
052A 2803 CALL SUP_GEN_CLR
052C CD0594 SUP_GEN_CLR
<052F> EQU $
052F C9 RET

1757 ; Handle GRAPHICS MODE 1
1758 ; SEMI_GRI
1759 EQU
1760 EX
1761 LD
1762 LD
1763 PUSH
1764 POP
1765 TMC
1766 LD
1767 INC
1768 LD
1769 LD
1770 PUSH
1771 POP

; AT THIS POINT:
HL->SOURCE BUFFER, PTRN_GNRTRS
DE-INDEX TO START OF VRAM ENTRIES
IY=NUMBER OF ITEMS TO READ FROM VRAM
BC=OFFSET TO COLOR SOURCE BUFFER @
AF=OBJ TYPE (& SUP VRAM FLG, UNNEEDED)
1741 ; FILL AS NEEDED TOP, MID, AND BOT PTRN_GNRTRS & DITTO FOR COLOR_GNRTRS
; IF BIT 7 OBJ_TYPE ON (TOP)
7,A
; GO HNDL MID
7,A
; IF BIT 6 OBJ_TYPE ON (MID)
; IF BIT 5 OBJ_TYPE ON (BOT)
; HL->NUMGEN
; IY=NUMGEN
; HL->PTRN_GNRTRS
; SAVE FOR RESTORE
BC

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
053E D5		1772	
053F F0E5		1773	
0541 3E03		1774	
0543 CD1C27		1775	
0546 C1		1776	
0547 E1		1777	
0548 50		1778	
0549 54		1779	
054A 09		1780	
054B 28		1781	
054C C83C		1782	
054E C81D		1783	
0550 C83C		1784	
0552 C81D		1785	
0554 C83C		1786	
0558 C828		1788	
055A C828		1789	
055C C828		1790	
055E B7		1791	
055F ED52		1792	
0561 23		1793	
0562 E5		1794	
0563 F0E1		1795	
0565 E1		1796	
0566 29		1797	
0567 29		1798	
0568 29		1799	
0569 C1		1800	
056A 09		1801	
056B 3E04		1802	
056D CD1C27		1803	
0570 F1		1804	
0571 C9		1805	
0572 C5		1806	
0573 F0E1		1807	
0575 D5		1808	
0576 5E		1809	
0577 23		1810	
0578 56		1811	
0579 C87A		1812	
057B 2014		1813	
057D 7A		1814	
057E FE70		1815	
0580 3806		1816	
0582 3E80		1817	
0584 12		1818	
0585 180A		1819	
0587 80		1820	
0588 210587		1821	
0588 010001		1822	
058E CD1D01		1823	
		1824	
		1825	
		1826	
		1827	
		1828	

```

DE
; SIGNAL PTRN GEN FILL
; BC := NUMGEN
; HL := FIRST_GEN_NAME
LD E,L
LD D,H ; DE := FIRST_GEN_NAME
ADD HL,BC ; HL := FIRST_GEN_NAME + NUMGEN
DEC HL
SRL H
RR L
SRL H
RR L
SRL H
RR L
SRL H
RR L
SRA E
SRA E
OR A
SBC HL,DE
INC HL
PUSH HL
POP IY
POP
ADD
ADD
ADD
POP
ADD
LD
CALL
POP
RET
; Internal routine to initialize X_Pat_Pos in Old_Screen
INIT_XP_OS:
POP
PUSH
POP
LD
INC
LD
BIT
JR
LD
CP
JR
LD
LD
LD
LD
LD
DEFB
OS_IN_VRAM:
LD
LD
CALL
EQU
POP
INC
RET
; HL := (FIRST_GEN_NAME + NUMGEN - 1)/8
; DE := FIRST_GEN_NAME/8
; CLEAR CARRY
; HL := (F_G_M + NUMGN - 1)/8 - F_G_M/8 + 1 = NUMBER COLR GEN$
; RESTORE REG
; STEP OVER PTRN_GNRTS
; HL -> COLOR_GNRTR_SOURCE
; SIGNAL PTRN COLOR_TBL
; FIX_STACK
; IY -> STATUS
; SAVE -> GRAPHICS
; DE := OLD_SCREEN_ADDRESS
; ? OLD_SCRN_IN_CROM
; OLD_SCREEN_IN_VRAM?
; INIT_X_PAT_POS = 80H
; ONE BYTE TO MOVE TO VRAM
; DE -> GRAPHICS
; DE -> FIRST_GEN_NAME

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

1829 ;
1830 ;Internal rout to setup Ptrn Gen VRAM & Color Gen VRAM
1831 SUP_GEN_CLR EQU $ ;SAVE FOR RESTORE
1832 PUSH AF
1833 PUSH BC
1834 PUSH IY
1835 PUSH DE
1836 PUSH HL
1837 LD A,3 ;SIGNAL PTRN GEN FILL
1838 CALL PUT_VRAM_ ;RESTORE
1839 POP HL
1840 POP DE
1841 POP IY
1842 POP BC
1843 POP AF
1844 PUSH AF
1845 PUSH BC
1846 PUSH IY
1847 PUSH DE
1848 PUSH HL
1849 BIT 4,A ;HOW MANY COLOR GEN BYTES?
1850 JR NZ,ONE_BYTE
1851 ADD HL,BC ;HL->COLOR GEN SOURCE
1852 LD A,4 ;SIGNAL PTRN COLOR FILL
1853 CALL PUT_VRAM_
1854 O.B_RET: HL
1855 POP DE
1856 POP IY
1857 POP BC
1858 POP AF
1859 RET

1860 ; For each item to send, duplicate the color byte 8 times (in C_BUFF)
1861 ; then send this generator to VRAM color table indexed by DE
1862 ONE_BYTE:
1863 ADD HL,BC ;HL -> COLOR BYTE
1864 LD C,L ;BC -> COLOR BYTE
1865 LD B,H
1866 PUSH IY
1867 POP HL ;HL = ITEM COUNT
1868 NEXT_COLOR:
1869 PUSH HL ;SAVE COUNTER
1870 LD A,(BC) ;GET COLOR BYTE
1871 PUSH BC ;SAVE POINTER TO COLOR
1872 LD BC,8 ;CREATE 8 DUPLICATES
1873 LD HL,(WORK_BUFFER)
1874 ADD HL,BC ;PLACE THEM HERE, STARTING AT END OF BUFFER
1875 LD B,8
1876 DUPLI: DEC HL
1877 LD (HL),A
1878 DJNZ DUPLI
1879 PUSH DE ;SAVE INDEX INTO TABLES
1880 LD IY,1 ;1 ITEM TO SEND
1881 LD A,4 ;COLOR TABLE CODE
1882 CALL PUT_VRAM_ ;GET INDEX BACK
1883 POP DE ;POINTER TO COLOR BYTE
1884 POP BC ;INCREMENT INDEX
1885 INC DE

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

05DF 03      1886      IMC BC      ; INCREMENT COLOR POINTER
05E0 E1      1887      POP HL      ; GET ITEM COUNTER
05E1 28      1888      DEC HL
05E2 7C      1889      LD A,H
05E3 85      1890      OR L
05E4 200C    1891      JR NZ,NEXT_COLOR
05E6 18CD    1892      JR O,B_RET
<05EB>      1893      ; Internal rout to update to next VRAM index screen area $
1894 SUP_UPDATE EQU $
1895      1894      PUSH BC
1896      1895      LD BC,100H
1897      1896      EX DE,HL
1898      1897      ADD HL,BC
1899      1898      EX DE,HL
1900      1899      POP BC
1901      1900      RET
1902 ;
<05F1>      1903      ACT MOBILE EQU $
1904 ; SUBCASE Mobile
1905      1903      CALL INIT_XP_OS ; X_PAT_POS := 80H
1906 ; INSERT NEW_GENERATOR ADDRESS IN OBJECT_CRAM
1907      1906      IMC DE
1908      1907      LD A,(DE)
1909      1908      LD (IY+5),A
1910      1909      IMC DE
1911      1910      LD A,(DE)
1912      1911      LD (IY+6),A
1913      1912      POP AF
1914      1913      RET
<0600>      1915      ACT_OSPRT EQU $
1916 ; SUBCASE Sprite size 0
1917 ACT_1SPRT EQU $
<0600>      1918 ; SUBCASE Sprite size 1
1919      1917      IMC BC
1920      1918      IMC BC
1921      1919      IMC BC
1922      1920      IMC BC
1923      1921      IMC BC
1924      1922      EX DE,HL
1925      1923      IMC HL
1926      1924      LD A,(HL)
1927      1925      LD E,A
1928      1926      LD D,O
1929      1927      DE
1930      1928      PUSH DE
1931      1929      IMC HL
1932      1930      LD E,(HL)
1933      1931      IMC HL
1934      1932      LD D,(HL)
1935      1933      IMC HL
1936      1934      ADD A,(HL)
1937      1935      LD (BC),A
1938      1936      LD C,(HL)
1939      1937      LD B,O
1940      1938      LD BC
1941      1939      PUSH IY
1942      1940      POP IY
1943      1941      POP DE,HL
1944      1942      POP DE

```

;HL->SOURCE PTRN GEN
;DE=INDEX TO PTRN GEN VRAM

;CALC & SET NEXT_GEN_CRAM

;DE=PTRN_PTR

;SV INDEX TO VRAM

;HL->FIRST_GEN_NAME

;->NEXT_GEN_IN_CRAM

;INIT_NEW_GEN_IN_STATUS

LOCATION OBJECT CODE LINE SOURCE LINE

0618 F1	1943	POP	AF
061C D0	1944	RET	MC
061D 3E01	1945	LD	A,1
061F CD1C27	1946	CALL	PUT_VRAM
0622 C9	1947	RET	
	1948		
	1949	PROG	

;SIGNAL SPRITE PRIN GEN FILL

LOCATION OBJECT CODE LINE SOURCE LINE

```

1951 ***** PUTOBJ *****
1952 ***** PUTOBJ VECTORS TO ONE OF 5 SPECIFIC ROUTINES FOR PLACING THE
1953 ;DESCRIPTION: DIFFERENT OBJECT TYPES ON THE DISPLAY
1954 ;
1955 ; IX = ADDRESS OF OBJECT TO BE PROCESSED
1956 ; B = PARAMETER TO BE PASSED SPECIFIC PUT ROUTINES
1957
1958 * IN ADDITION, THIS MODULE CONTAINS ROUTINES WHICH ALLOW VRAM OPERATIONS
1959 * TO BE DEFERRED, TYPICALLY UNTIL AN INTERRUPT OCCURS, AND PERFORMED
1960 * IN A BLOCK BY A CENTRAL WRITER ROUTINE.
1961 *****
1962
1963 DATA
1964 QUEUE_SIZE DEFS 1
1965 * THIS IS THE SIZE OF THE DEFERRED WRITE QUEUE. IT IS SET BY THE
1966 * CARTRIDGE PROGRAMMER. IT HAS RANGE 0 - 255.
1967
1968 QUEUE_HEAD DEFS 1
1969 QUEUE_TAIL DEFS 1
1970 * THESE ARE THE INDICES OF THE HEAD AND TAIL OF THE WRITE QUEUE.
1971
1972 HEAD_ADDRESS DEFS 2
1973 TAIL_ADDRESS DEFS 2
1974 * THESE ARE THE ADDRESSES OF THE QUEUE HEAD AND TAIL
1975
1976 ;TRUE EQU 1
1977 ;FALSE EQU 0
1978 * VALUES FOR BOOLEAN DEFERAL_FLAG
1979
1980 BUFFER DEFS 2
1981 * THIS IS A POINTER TO THE BEGINNING OF THE DEFERRED WRITE QUEUE. THE
1982 * CARTRIDGE PROGRAMMER IS RESPONSIBLE FOR PROVIDING A RAM AREA TO HOLD
1983 * THE QUEUE, AND PASSING ITS LOCATION AND SIZE TO INIT_QUEUE.
1984
1985 ;
1986 ;PARAM_AREA DEFS 3
1987 * PARAM_AREA IS THE COMMON PARAMETER PASSING AREA FOR PASCAL ENTRY PTS
1988
1989
1990 PROG EQU $
1991 SET_UP_WRITE EQU $
1992
1993 * SET_UP_WRITE SETS UP A DEFERRED VRAM OPERATION.
1994
1995 * PUT DATA AT QUEUE_HEAD
1996 PUSH
1997 LD HL, (HEAD_ADDRESS)
1998 POP DE
1999 LD [HL], E ; PUT DATA POINTER
2000 INC HL
2001 LD [HL], D
2002 INC HL
2003 LD [HL], B ; STORE PUTOBJ PARAMETER
2004 INC HL
2005 EX DE, HL ; HEAD ADDRESS IN DE
2006
2007 * INCREMENT QUEUE_HEAD

```

<0623>

0623 DDE5
0625 2A73CD
0628 D1
0629 73
062A 23
0628 72
062C 23
062D 70
062E 23
062F EB

```

LOCATION OBJECT CODE LINE SOURCE LINE
0630 3A73C8 LD LD A, [QUEUE_HEAD]
0633 3C INC A ; NEW HEAD IN A
2010
2011 * IF QUEUE_HEAD = QUEUE_SIZE THEN
2012 LD HL, QUEUE_SIZE
2013 CP [HL]
2014 JR NZ, NOT_TOO_BIG
2015
2016 * QUEUE_HEAD := 0
2017 LD A, 0
2018 LD [QUEUE_HEAD], A
2019
2020 * HEAD_ADDRESS := BUFFER
2021 LD HL, [BUFFER]
2022 LD [HEAD_ADDRESS], HL
2023
2024 JR SET_UP_ENDIF
2025 * ELSE
2026 NOT_TOO_BIG EQU $
2027
2028 * STORE NEW QUEUE_HEAD
2029 LD [QUEUE_HEAD], A
2030
2031 * STORE HEAD_ADDRESS
2032 LD [HEAD_ADDRESS], DE
2033
2034 * END_IF
2035 SET_UP_ENDIF
2036
2037 * END SET_UP_WRITE
2038 RET
2039
2040 * PROCEDURE INIT_QUEUE (SIZE:BYTE; VAR A_QUEUE:QUEUE)
2041
2042 * SIZE PASSED IN A, LOCATION PASSED IN HL
2043 * DESTROYS: A
2044
2045 INIT_QUEUE_P DEFV 2, 1, -2
2046 * THIS IS THE PARAMETER DESCRIPTOR FOR INIT_QUEUE
2047
2048 * BEGIN INIT_QUEUE
2049 GLB INIT_QUEUE
2050 EQU $
2051 BC, INIT_QUEUE_P
2052 LD DE, PARAM_AREA
2053 LD PARAM
2054 LD A, [PARAM_AREA]
2055 LD HL, [PARAM_AREA+1]
2056
2057 GLB INIT_QUEUE
2058 EQU $
2059
2060 * QUEUE_SIZE := SIZE
2061 LD [QUEUE_SIZE], A
2062
2063 * QUEUE_HEAD := QUEUE_TAIL := 0

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
0667 3E00 LD A,0
0669 3273C8 LD [QUEUE_HEAD],A
066C 3273CC LD [QUEUE_TAIL],A
2067 * BUFFER := TAIL_ADDRESS := HEAD_ADDRESS := LOCATION
2068 LD [BUFFER],HL
2069 LD [HEAD_ADDRESS],HL
2070 LD [TAIL_ADDRESS],HL
2071 LD
2072 * END INIT_QUEUE
2073 RET
2074
2075 * PROCEDURE WRITER_
2076
2077 * TAKES NO PARAMETERS
2078 * DESTROYS: ALL
2079
2080 * BEGIN WRITER_
2081 WRITER_ GLB
2082 WRITER_ EQU
2083
2084 * SAVE DEFERAL FLAG
2085 LD
2086 PUSH
2087
2088 * DEFER_WRITES := FALSE
2089 LD
2090 LD
2091
2092 * WHILE QUEUE_TAIL <> QUEUE_HEAD DO
2093 WRTR_WHITE EQU
2094 LD A, [QUEUE_TAIL]
2095 LD HL, QUEUE_HEAD
2096 CP [HL]
2097 JR Z, WRTR_END_WHITE
2098
2099 * WRITE DATA AT QUEUE_TAIL TO VRAM
2100 LD HL, [TAIL_ADDRESS]
2101 E, [HL] ; GET OBJECT POINTER
2102 HL
2103 D, [HL]
2104 HL
2105 B, [HL] ; GET PARAMETER
2106 HL
2107
2108 * PROCESS OBJECT IN QUEUE
2109 PUSH DE
2110 POP IX
2111 PUSH HL
2112 CALL DO_PUTOBJ ;SAVE QUEUE TAIL ADDRESS
2113
2114 * INCREMENT QUEUE_TAIL
2115 LD A, [QUEUE_TAIL]
2116 INC A
2117
2118 * IF QUEUE_TAIL = QUEUE_SIZE THEN
2119 LD HL, QUEUE_SIZE
2120

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
06A2 BE 2121 CP (HL)
06A3 200E 2122 JR NZ,WRTR_ELSE
2123 *
2124 * QUEUE_TAIL := 0
2125 LD A,0
2126 LD (QUEUE_TAIL),A
2127 *
2128 * TAIL_ADDRESS := BUFFER
2129 LD HL,(BUFFER)
2130 LD (TAIL_ADDRESS),HL
2131 POP HL ;RESTORE STACK POINTER
2132 *
2133 JR WRTR_END_IF
2134 * ELSE
2135 WRTR_ELSE EQU $
2136 *
2137 * STORE NEW QUEUE_TAIL
2138 LD (QUEUE_TAIL),A
2139 *
2140 * TAIL_ADDRESS := TAIL_ADDRESS + 3
2141 POP HL
2142 LD (TAIL_ADDRESS),HL
2143 *
2144 * END_IF
2145 WRTR_END_IF EQU $
2146 *
2147 JR WRTR_WHILE
2148 * END WHILE
2149 WRTR_END_WHILE EQU $
2150 *
2151 * RESTORE DEFERAL FLAG
2152 POP AF
2153 LD (DEFER_WRITES),A
2154 *
2155 * END WRITER_ RET
2156 *
2157 *
2158 *
2159 *
2160 *
2161 * ;EXT PUTSENT,PUT_MOBILE,PUTOSPRITE,PUT1SPRITE,PUTCOMPLEX
2162 *
2163 * ;EXT DEFER WRITES
2164 * ;EXT PARAM_
2165 * GLB PUTOBJO
2166 PUTOBJ_PAR: DEFW 2,2,1
2167 *
2168 * PROCEDURE PUT_OBJ (VAR DATA:BUFFER;PARAM:BYTE);
2169 *
2170 * THIS IS THE PASCAL ENTRY POINT TO THE PUTOBJ ROUTINE
2171 *
2172 * PROG
2173 PUTOBJO:
2174 LD BC,PUTOBJ_PAR
2175 LD DE,PARAM_AREA
2176 CALL PARAM_
06C1 00020002
06C5 0001
06C7
06C7 0106C1
06CA 11738A
06CD CD0098

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0600 D02A738A 2177 LD IX,[PARAM_AREA]
0604 3A738C 2178 LD A,[PARAM_AREA+2]
0607 47 2179 LD B,A
2180
2181 DEFER EQU 1
2182 PUTOBJ_
0608 3A73C6 2183 LD A,[DEFER_WRITES] ;CHECK IF DEFERRED WRITE IS DESIRED
060B FE01 2184 CP DEFER
0600 2004 2185 JR NZ,DO_PUTOBJ ;IF NOT, PROCESS OBJECT
060F C00623 2186 CALL SET_UP_WRITE ;IF SO, SET UP FOR DEFERRED WRITE
06E2 C9 2187 RET
06E3 D06601 2188 DO_PUTOBJ LD H,[IX+1] ;GET ADDRESS OF GRAPHICS FOR OBJ_n
06E6 D06E00 2189 LD L,[IX+0]
06E9 7E 2190 LD A,[HL]
06EA 4F 2191 LD C,A
06EB E60F 2192 AND OFH
06ED CA06FF 2193 JP Z,PUTSEMI
06F0 30 2194 DEC A
06F1 CA0A07 2195 JP Z,PUT_MOBILE ;1 = MOBILE
06F4 30 2196 DEC A
06F5 CA080F 2197 JP Z,PUTOSPRITE ;2 = SPRITE
06F8 30 2198 DEC A
06F9 CA0955 2199 JP Z,PUTSPRITE ;3 = SPRITE1
06FC C30EA2 2200 JP PUTCOMPLEX ;>3 = COMPLEX
2201 ;
2202 ; END ;prname
2203 ; PROG

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
2205 ***** PUT_SEMI *****
2206 ;DESCRIPTION: PUTS SEMI_MOBILE OBJECTS ON SCREEN
2207 ;
2208 ;
2209 ;INPUT: IX = ADDRESS OF OBJECT TO BE PROCESSED
2210 ; HL = ADDRESS OF OBJECT'S GRAPHICS TABLES IN ROM
2211 ;*****
2212 GLB PUTSEMI
2213 ;
2214 ;
2215 ;
2216 PUTSEMI: LD D, [IX+3] ;GET ADDRESS OF STATUS
2217 LD E, [IX+2] ;
2218 PUSH DE ; AND PUT INTO IX
2219 POP IX ;
2220 LD D, [IX+2] ;GET X_LOCATION
2221 LD E, [IX+1] ;
2222 CALL PX_TO_PTRN_POS
2223 ;
2224 LD C,E ;C := PATTERN PLANE COL.
2225 LD D, [IX+4] ;GET Y_LOCATION
2226 LD E, [IX+3] ;
2227 CALL PX_TO_PTRN_POS
2228 ;
2229 LD B,E ;B := PATTERN PLANE ROW
2230 LD E, [IX+0] ;GET FRAME NUMBER
2231 ;
2232 ; HL = GRAPHICS_n, IX = OBJ_n, IV = STATUS_n, C = COL., B = ROW, E = FRAME
2233 ;
2234 LD D,0 ;DE HAS FRAME NUMBER
2235 ADD HL,DE ;2*FRAME NUMBER + ADDR OF GRAPHICS_n
2236 ADD HL,DE ;FRAME POINTER OFFSET
2237 LD E,5 ;HL NOW POINTS TO LOCATION HOLDING ADDRESS
2238 ADD HL,DE ;OF FRAME
2239 ;
2240 LD E, [HL] ;HL := ADDRESS OF FRAME
2241 INC HL ;
2242 LD D, [HL] ;DE := Y PAT POS & X PAT_POS
2243 EX DE, HL ;C := X_EXTENT
2244 PUSH BC ;
2245 POP DE ;
2246 LD C, [HL] ;B := Y_EXTENT
2247 INC HL ;HL POINTS TO FIRST NAME IN LIST
2248 LD B, [HL] ;
2249 INC HL ;
2250 ;
2251 ; TEST TO SEE IF OLD_SCREEN IS TO BE SAVED
2252 ;
2253 LD A, [IX+5] ;GET HIGH BYTE OF OLD_SCREEN ADDRESS
2254 BIT 7, A ;TEST BIT 15 OF OLD_SCREEN ADDRESS
2255 JR Z, S_OLD_SCRN
2256 ;
2257 CALL PUTFRAME
2258 RET
2259 ;
2260 S_OLD_SCRN
2261 ;
06FF 005603
0702 005E02
0705 05
0706 00E1
0708 005602
0708 005E01
070E 0007E8
0711 48
0712 005604
0715 005E03
0718 0007E8
0718 43
071C 005E00
071F 1600
0721 19
0722 19
0723 1E05
0725 19
0726 5E
0727 23
0728 56
0729 EB
072A C5
072B D1
072C 4E
072D 23
072E 46
072F 23
0730 007E05
0733 007F
0735 2804
0737 000808
073A C9
073B

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0738 C5 2262 PUSH BC ;SAVE REGS
073C D5 2263 PUSH DE
073D E5 2264 PUSH HL
073E FE70 2265 CP
0740 2802 2266 JR
0742 3807 2267 JR
2268
2269
2270 ; IF (.A,GE,70H)
2271 EQUAL TO
2272 LD H,A
2273 LD L,(IX+4)
2274 LD A,(HL)
2275
2276
2277 ELSE_1
2278
2279 ;
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307 ;
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318 ;
0748 2A8006 2279 ;
074E D05605 2280 ;OLD SCREEN IN VRAM
0751 D05E04 2281 ;GET ADDRESS OF FREE BUFFER SPACE
0754 E5 2282 LD D,(IX+5)
0755 D5 2283 LD E,(IX+4)
0756 E5 2284 PUSH HL
0757 010004 2285 PUSH DE
075A CD103E 2286 LD BC,4
075D E1 2287 CALL VRAM_READ
075E 7E 2288 POP HL
075F FE80 2289 LD A,(HL)
0761 2003 2290 CP 80H
0763 D1 2291 JR NZ,GET_OLD
0764 1819 2292 POP DE
0766 23 2293 JR SKIP_OLD
0767 23 2294 INC HL
0768 46 2295 INC HL
076A 5E 2296 LD B,(HL)
076B 5E 2297 INC HL
076D 23 2298 LD E,(HL)
076E EB 2299 LD D,0
2300 INC HL
2301 EX DE,HL
2302
2303 JR
2304 M_XY: ADD
2305 DJNZ
2306
2307 ;
2308 DJNZ M_XY
2309 PUSH HL
2310 POP BC
2311 EX DE,HL
2312 POP DE
2313 INC DE
2314 INC DE
2315 INC DE
2316 CALL VRAM_READ
2317 SKIP_OLD POP HL
2318 ; EMDIF
0744 2744 2271 ;
0744 67 2272 ;HL := OLD_SCREEN ADDRESS
0745 D06E04 2273 ;
0748 7E 2274 ;
0749 1835 2275 EMD_IF_1
0748 0748 2277 ;
0748 2A8006 2279 ;
074E D05605 2280 ;OLD SCREEN IN VRAM
0751 D05E04 2281 ;GET ADDRESS OF FREE BUFFER SPACE
0754 E5 2282 LD D,(IX+5)
0755 D5 2283 LD E,(IX+4)
0756 E5 2284 PUSH HL
0757 010004 2285 PUSH DE
075A CD103E 2286 LD BC,4
075D E1 2287 CALL VRAM_READ
075E 7E 2288 POP HL
075F FE80 2289 LD A,(HL)
0761 2003 2290 CP 80H
0763 D1 2291 JR NZ,GET_OLD
0764 1819 2292 POP DE
0766 23 2293 JR SKIP_OLD
0767 23 2294 INC HL
0768 46 2295 INC HL
076A 5E 2296 LD B,(HL)
076B 5E 2297 INC HL
076D 23 2298 LD E,(HL)
076E EB 2299 LD D,0
2300 INC HL
2301 EX DE,HL
2302
2303 JR
2304 M_XY: ADD
2305 DJNZ
2306
2307 ;
2308 DJNZ M_XY
2309 PUSH HL
2310 POP BC
2311 EX DE,HL
2312 POP DE
2313 INC DE
2314 INC DE
2315 INC DE
2316 CALL VRAM_READ
2317 SKIP_OLD POP HL
2318 ; EMDIF
0774 E5 2307 ;
0775 C1 2308 ;BC := NUMBER OF BYTES TO READ
0776 EB 2309 ;HL := FREE BUFF ADDR + 4
0777 D1 2310 ;DE := OLD_SCREEN ADDR.
0778 13 2311
0779 13 2312
077A 13 2313
077B 13 2314
077C CD103E 2315 ;READ SAVED NAMES FOR BACKGROUND
077F E1 2316 ;HL := FREE BUFF ADDR.

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0780 2319 END_IF_1
0780 7E 2320 LD A,[HL] ;A := X_PAT_POS
2321
2322
2323
2324
0781 FE80 CP
0783 280F JR ;THEN THERE IS AN OLD_SCREEN
2327 ;E := X_PAT_POS
0785 5E LD E,[HL] ;D := Y_PAT_POS
0786 23 INC HL ;C := X_EXTENT
0787 56 LD D,[HL] ;B := Y_EXTENT
0788 23 INC HL ;HL POINTS TO FIRST NAME IN LIST
0789 4E LD C,[HL]
078A 23 INC HL
078B 46 LD B,[HL]
078C 23 INC HL
2337
078D D0E5 PUSH
2338
2339 CALL PUTFRAME ;RESTORE OLD_SCREEN TO DISPLAY
078F C0808 POP
2340
0792 D0E1 POP
2341
0794 2342
2343
2344 END_IF_2
2345
0794 E1 ENDIF
0795 D1 POP HL ;HL := ADDR OF FIRST NAME IN FRAME
0796 C1 POP DE ;DE := Y,X_PAT_POS
0797 C5 POP BC ;BC := Y,X_EXTENTS
0798 D5 PUSH DE
0799 E5 PUSH HL
079A D06605 LD H,[(IX+5)
0790 D06E04 LD L,[(IX+4)
2354
2355
07A0 3E70 LD
07A2 BC CP
07A3 3803 JR
2358
2359
07A5 2A8006 IF (.H,LT,70H)
2360 ; LD HL,(WORK_BUFFER)
2361
2362
07A8 2363 END_IF_3
2364
2365 ;
07A8 73 LD (HL),E ;OLD_SCREEN + 0 := X_PAT_POS
07A9 23 INC HL ;
07AA 72 LD (HL),D ; " 1 := Y_PAT_POS
07AB 23 INC HL ;
07AC 71 LD (HL),C ; " 2 := X_EXTENT
07AD 23 INC HL ; " 3 := Y_EXTENT
07AE 70 LD (HL),B ;HL := ADDRESS TO STORE NAMES
07AF 23 INC HL
2374
0780 DDES PUSH
2375

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
2376 CALL GET_BKGRND
2377
2378
2379 POP
2380
2381 POP HL
2382 POP DE
2383 POP BC
2384
2385 PUSH
2386
2387 CALL PUTFRAME
2388
2389 POP
2390
2391 LD D, [IX+5]
2392
2393 SV2: LD
2394 CP
2395 JR
2396 JR
2397
2398 ;SV2:: IF [D,LT,70H]
2399 LD E, [IX+4]
2400 EXX
2401 LD HL, [WORK_BUFFER]
2402 PUSH HL
2403 INC HL
2404 INC HL
2405 LD E, [HL]
2406 LD D, 0
2407 INC HL
2408 LD B, [HL]
2409 EX DE, HL
2410 JR M, XY2+1
2411 M_XY2: ADD HL, HL
2412 DJNZ M, XY2
2413 PUSH HL
2414 EXX
2415 POP BC
2416 POP HL
2417 CALL VRAM_WRITE
2418
2419 END_IF_4
2420
2421 ;
2422 ENDIF
2423 RET
2424
2425 ***** PX.TO.PTRN.POS *****
2426 ;DESCRIPTION: DIVIDES REG DE BY B, IF SIGNED RESULT > 127 THEN E := MAX SIGNED
2427 ; POSITIVE NUMBER. IF RESULT < -128, THEN E := MIN NEGATIVE NUM
2428 DE = 16 BIT SIGNED NUMBER
2429 DE/8 < -128 E = -128
2430 -128 <= DE/8 <= +127 E = DE/8
2431 +127 < DE/8 E = +127
2432 *****

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2433
2434
2435 PX_TO_PTRN_POS GLB PX_TO_PTRN_POS
2436
2437 PUSH HL
2438 SRA D ;HL USED TO TEST MAGNITUDE
2439 RR E ;16 BIT SHIFT LEFT
2440 SRA D
2441 RR E
2442 SRA D
2443 RR E
2444 BIT 7,D
2445 JR NZ,MEGTV
2446 LD HL,OFF80H
2447 ADD HL,DE
2448 POP HL
2449 RET NC
2450 LD E,7FH
2451 RET
2452 ;XXXXXX
2453 MEGTV: LD HL,080H
2454 ;XXXXXX
2455 ADD HL,DE
2456 POP HL
2457 RET C
2458 LD E,80H
2459 RET
2460
2461 ***** PUT FRAME *****
2462 ;DESCRIPTION: THE NAMES WHICH CONSTITUTE A FRAME ARE MOVED TO THE NAME TABLE
2463 ; IN VRAM. THE UPPER LEFT HAND CORNER OF THE FRAME IS POSITIONED
2464 ; AT X_PAT_POS, Y_PAT_POS.
2465 ;INPUT: HL = ADDRESS OF LIST OF NAMES (IN CPU RAM)
2466 ; E = X_PAT_POS
2467 ; D = Y_PAT_POS
2468 ; C = X_EXTENT
2469 ; B = Y_EXTENT
2470 *****
2471 ;
2472 GLB
2473 PUTFRAME: PUSH BC
2474 PUSH DE
2475 PUSH HL
2476 EXX
2477 POP HL
2478 POP DE
2479 POP BC
2480 CALL CALC_OFFSET
2481 EXX
2482 ;XXXXXX
2483 ; TEST FOR THE FOLLOWING CONDITION: (X_PAT_POS SLE 32) AND (X_PAT_POS + X_EXTENT
2484 ; SGT 0)
2485 PF1: LD A,E
2486 BIT 7,A
2487 JR NZ,XP_NEG
2488 CP 32
2489 RET NC

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
081E 81 2490 XP_NEG: ADD A,C
081F CB7F 2491 BIT 7,A
0821 C0 2492 RET NZ
0822 B7 2493 OR A
0823 CB 2494 RET Z
2495 ;XXXXXX
2496 ;X.IN_BOUNDS::
2497
2498 X_IN_BOUNDS
2499
2500 ; IF [.,IS,MINUS]
2501 BIT
2502 JR
2503
2504
2505 LD A,C
2506 ADD A,E
2507 PUSH DE
2508 ;XXXXXX
2509 CP 33
2510 JR C,LT33
2511 LD A,32
2512 LT33:
2513 ;XXXXXX
2514
2515 LD D,0
2516 PUSH DE
2517 POP IY
2518 POP DE
2519 LD A,E
2520 EXX
2521 PUSH BC
2522 NEG
2523 LD C,A
2524 LD B,0
2525 ADD HL,BC
2526 EX DE,HL
2527 ADD HL,BC
2528 EX DE,HL
2529 POP BC
2530 EXX
2531 JR
2532
2533 ; ELSE
2534 ;
2535
2536 ELSE_8
2537
2538 PF2:
2539 LD A,E
2540 ADD A,C
2541 IF [.,GT,31]
2542 CP
2543 JR
2544 JR
2545
2546 LD A,32
2547 ;SUBTRACT X_PAT_POS FROM 31
2548
2549 81
2550 ;IS X_PAT_POS + X_EXTENT > 31
2551 Z,ELSE_9
2552 C,ELSE_9
2553
2554 ;SUBTRACT X_PAT_POS FROM 31
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0852 93 2547 SUB E
0853 05 2548 PUSH DE
0854 5F 2549 LD E,A
0855 1600 2550 LD D,0
0857 05 2551 PUSH DE
0858 FDE1 2552 POP IY
085A 01 2553 POP DE
085B 1807 2554
085B 1807 2555 JR
085B 1807 2556 END_IF_9
2557 ;
2558 ELSE
2559 ELSE_9
2560
0850 2561 PF3:
0850 C5 2562 PUSH BC
085E 0600 2563 LD B,0
0860 C5 2564 PUSH BC
0861 FDE1 2565 POP IY
0863 C1 2566 POP BC
2567 ;
2568 END_IF_9
0864 2569
2570 ;
2571 ENDIF
0864 2572 END_IF_8
2573
0864 1E00 2574 LD E,0
2575 ;
2576 REPEAT
0866 2577 RPT_1
2578
0866 7A 2579 PF4:
0867 83 2580 ADD A,E
2581 ;
2582 IF [.A,IS,PLUS]
0868 087F 2583 BIT
086A 2019 2584 JR
2585
2586 ;
2587 IF [.A,LE,23]
086C FE16 2588 CP
086E 3015 2589 JR
2590
0870 C5 2591 PUSH BC
0871 05 2592 PUSH DE
0872 09 2593 EXX
0873 C5 2594 PUSH BC
0874 05 2595 PUSH DE
0875 E5 2596 PUSH HL
0876 FDE5 2597 PUSH IY
0878 3E02 2598 LD A,2
087A CD1C27 2599 CALL PUT_VRAM
0870 FDE1 2600 POP IY
0880 01 2601 POP HL
2602 POP DE
0881 C1 2603 POP BC
;GET THIS NUMBER INTO IY
;BOTH ENDS OF FRAME WITHIN PATTERN PLANE
; Y_EXTENT-1 TIMES
;GET Y_PAT_POS
;ADD Y
7,A
NZ,END_IF_10
; IS 0<=Y_PAT_POS + Y <=23
24
NC,END_IF_10
;CODE FOR PATTERN NAME TABLE ADDED 4/20

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0882 D9 2604 EXX
0883 D1 2605 POP DE
0884 C1 2606 POP BC
2607 ; EMDIF
2608 ; EMDIF
2609
0885 2610 EMD_IF_10
2611
0885 D9 2611 EXX
0886 C5 2612 PUSH BC
0887 0600 2613 LD B,0
0889 09 2614 ADD HL,BC
088A EB 2615 EX DE,HL
088B 010020 2616 LD BC,32
088E 09 2617 ADD HL,BC
088F EB 2618 EX DE,HL
0890 C1 2619 POP BC
0891 D9 2620 EXX
0892 1C 2621 INC E
2622 ; UNTIL [E,EQ,-B]
2623 ;
2624 LD A,E
2625 CP B
2626 JR NZ,RPT_1
2627
2628
0897 C9 2629 RET
2630 ;
2631 ; .COMMENT )
2632 ; ***** GET_BKGRND *****
2633 ; DESCRIPTION: THIS ROUTINE GETS THE NAMES FROM THE NAME TABLE WHICH CONSTITUTE
2634 ; THE BACKGROUND ON WHICH AN OBJECT IS TO BE MOVED
2635 ; INPUT: HL = LOCATION IN CPU RAM TO WHICH THE NAMES ARE MOVED
2636 ; D = Y_PAT_POS (TOP ROW OF PATTERN)
2637 ; E = X_PAT_POS (LEFT HAND COLUMN)
2638 ; B = Y_EXTENT OF PATTERN
2639 ; C = X_EXTENT OF PATTERN
2640 ; *****
2641 ;
2642 ; GLB GET_BKGRND
0898 CD08C0 2643 GET_BKGRND: ;OFFSET INTO NAME TABLE OF POSITION OF UPPER LEFT
0898 C5 2644 CALL CALC_OFFSET ;
089C 0600 2645 PUSH BC ;GET X_EXTENT INTO IY
089F FDE1 2646 LD B,0 ; NUMBER OF NAMES PER ROW
08A1 C1 2647 PUSH BC
2648 POP IY
2649 POP BC
2650 ; REPEAT ;Y_EXTENT-1 TIMES
2651
08A2 2652 RPT_2
2653
08A2 C5 2654 PUSH BC
08A3 D5 2655 PUSH DE
08A4 E5 2656 PUSH HL
08A5 FDE5 2657 PUSH IY
08A7 3E02 2658 LD A,2
08A9 CD1BA3 2659 CALL GET_VRAM_
08AC FDE1 2660 POP IY

```

; TABLE CODE FOR PATTERN NAME TABLE

```

LOCATION OBJECT CODE LINE SOURCE LINE
08AE E1 2661 POP HL
08AF D1 2662 POP DE
0880 C1 2663 POP BC
0881 C5 2664 PUSH BC
0882 0600 2665 LD B,0
0884 09 2666 ADD HL,BC
0885 010020 2667 LD BC,32
0888 EB 2668 EX DE,HL
0889 09 2669 ADD HL,BC
088A EB 2670 EX DE,HL
088B C1 2671 POP BC
088C 05 2672 DEC B
2673 ; UNTIL (.B,EQ,0)
2674
2675 JR
2676
2677 RET
2678 ;
2679 ;
2680 ***** CALC.OFFSET *****
2681 ;DESCRIPTION: THIS ROUTINE CALCULATES THE PROPER OFFSET INTO THE NAME TABLE
2682 ; FOR THE PATTERN POSITION GIVEN BY X_PAT_POS, Y_PAT_POS. THE
2683 ; FORMULA USED IS: OFFSET = 32*Y_PAT_POS + X_PAT_POS
2684 ; INPUT:
2685 ; D = Y_PAT_POS
2686 ; OUTPUT:
2687 ; DE = OFFSET
2688 ;
2689 CALC_OFFSET:
2690 PUSH HL
2691 ; IF (.D,IS,MINUS) ;EXTEND SIGN
2692
2693 BIT
2694 JR
2695
2696 LD H,OFFH
2697
2698 JR
2699
2700 ; ELSE
2701
2702 ELSE_11
2703
2704 LD H,0
2705 ; ENDTF
2706
2707 END_IF_11
2708
2709 LD L,D
2710 ADD HL,HL
2711 ADD HL,HL
2712 ADD HL,HL
2713 ADD HL,HL
2714 ADD HL,HL
2715 ; IF (.E,IS,MINUS)
2716 ;EXTEND SIGN
2717 BIT
2718
2719
2720 ; OFFSET = 32*Y_PAT_POS + X_PAT_POS
2721 ;HL=2*Y_PAT_POS
2722 ; 4* "
2723 ; 8* "
2724 ; 16* "
2725 ; 32* "
2726 ;EXTEND SIGN
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0803 2804 2718 JR Z,ELSE_12
2719
0805 16FF 2720 LD D,OFFH
2721
0807 1802 2722 JR END_IF_12
2723
2724 ; ELSE
2725
0809 2726 ELSE_12
2727
0809 1600 2728 LD D,0
2729 ; ENDIF
2730
0808 2731 END_IF_12
2732
0808 19 2733 ADD HL,DE
080C EB 2734 EK DE,HL
0800 E1 2735 POP HL
080E C9 2736 RET
2737
***** EXTERNALS *****
2738
2739
2740 ;EXT GET VRAM
2741 ;EXT PUT_VRAM_
2742 ;EXT VRAM_READ
2743 ;EXT VRAM_WRITE
2744 ;EXT WORK_BUFFER
2745 PROG
;HL=32*Y_PAT_POS + X_PAT_POS

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
2747 * THIS MODULE CONTAINS CODE FOR THE PUT1SPRITE AND PUTOSPRITE
2748 * ROUTINES. THESE ROUTINES TURN OUT TO BE ESSENTIALLY THE SAME CODE
2749 * WITH TWO SLIGHTLY DIFFERENT ENTRY POINTS
2750 * WITH TWO SLIGHTLY DIFFERENT ENTRY POINTS
2751
2752 * IT IS CALLED WITH THE ADDRESS OF THE SPRITE OBJECT IN THE IX REGISTER.
2753 * THE FORMAT FOR SPRITE OBJECTS IS
2754 *
2755 * SPRITE OBJECT = RECORD
2756 * GRAPHICS: ^SPRITE_GRAPHICS
2757 * STATUS: ^SPRITE_STATUS
2758 * PTRM_POINTER: ^PATTERN_GENERATOR
2759 * NUMGEN: BYTE
2760 * FRAME_TABLE_PTR: ^ARRAY(0..70) OF FRAME (TABLE OF ANIMATION FRAMES)
2761 * END SPRITE_OBJECT
2762
2763 * SPRITE_GRAPHICS = RECORD
2764 * OBJECT_TYPE: BYTE
2765 * FIRST_GEN_NAME: BYTE
2766 * PTRM_POINTER: ^PATTERN_GENERATOR
2767 * NUMGEN: BYTE
2768 * FRAME_TABLE_PTR: ^ARRAY(0..70) OF FRAME (TABLE OF ANIMATION FRAMES)
2769 * END SPRITE_ROM_GRAPHICS
2770
2771 * SPRITE_STATUS = RECORD
2772 * FRAME: BYTE
2773 * X_LOCATION: INTEGER
2774 * Y_LOCATION: INTEGER
2775 * NEXT_GEN: BYTE
2776 * END SPRITE_STATUS
2777
2778 * FRAME = RECORD
2779 * COLOR: BYTE
2780 * SHAPE: BYTE
2781 * END FRAME
2782
2783 * SPRITE = RECORD
2784 * Y: BYTE
2785 * X: BYTE
2786 * NAME: BYTE
2787 * COLOR_AND_TAG: BYTE
2788 * END SPRITE
2789
2790 *****
2791 ***** DICTIONARY *****
2792 *****
2793 EXT WORK_BUFFER
2794 * WORK_BUFFER IS A POINTER IN CARTRIDGE ROM, LOCATED AT 8006H, TO THE
2795 * FREE BUFFER AREA TO BE USED BY THE GRAPHICS ROUTINES.
2796 *
2797 <000F> 2798 SPRITE_PTR EQU IX
2799 * SPRITE_PTR IS A POINTER TO THE NEW SPRITE NAME TABLE ENTRY BEING
2800 * BUILT BY THIS ROUTINE.
2801
2802 <000E> 2802 THIS_SPRITE EQU IX
2803 * THIS_SPRITE IS A POINTER TO THE SPRITE OBJECT BEING PUT

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

2804
<0000> 2805 GRAPHICS EQU 0
<0002> 2806 STATUS EQU 2
<0004> 2807 SPRITE_INDEX EQU 4
2808 * FIELD OFFSETS FOR SPRITE_OBJECT RECORDS
2809
<0000> 2810 OBJECT_TYPE EQU 0
<0001> 2811 FIRST_GEN_NAME EQU 1
<0002> 2812 PTRN_POINTER EQU 2
<0004> 2813 NUMGEN EQU 4
<0005> 2814 FRAME_TABLE_PTR EQU 5
2815 * FIELD OFFSETS FOR SPRITE_GRAPHICS RECORDS
2816
<0000> 2817 FRAME EQU 0
<0001> 2818 X_LOCATION EQU 1
<0003> 2819 Y_LOCATION EQU 3
<0005> 2820 NEXT_GEN EQU 5
2821 * FIELD OFFSETS FOR SPRITE_STATUS RECORDS
2822
<0000> 2823 COLOR EQU 0
<0001> 2824 SHAPE EQU 1
2825 * FIELD OFFSETS FOR FRAME RECORDS
2826
<0000> 2827 Y EQU 0
<0001> 2828 X EQU 1
<0002> 2829 NAME EQU 2
<0003> 2830 COLOR_AND_TAG EQU 3
2831 * FIELD OFFSETS FOR SPRITE RECORDS
2832 ***** EXTERNAL PROCEDURES *****
2833
2834 ; EXT PUT_VRAM,GET_VRAM
2835 * EXTERNAL PROCEDURE PUT_VRAM (TABLE CODE:BYTE; START_INDEX:SLICE:BYTE;
2836 * VAR DATA:BUFFER;ITEM_COUNT:INTEGER);
2837
2838 * EXTERNAL PROCEDURE GET_VRAM (TABLE CODE:BYTE; START_INDEX:SLICE:BYTE;
2839 * VAR DATA:BUFFER;ITEM_COUNT:INTEGER);
2840
2841 * PUT_VRAM SENDS A BLOCK OF DATA TO THE TABLE SPECIFIED BY TABLE CODE.
2842 * THE SLICE, START_INDEX, AND ITEM_COUNT ARE TABLE DEPENDANT. GET_VRAM
2843 * DOES THE INVERSE OPERATION.
2844
2845 * - TABLE CODE IS PASSED IN A
2846 * - START_INDEX,SLICE IN DE
2847 * - DATA_BUFFER ADDRESS IN HL
2848 * - BYTE COUNT PASSED IN IV
2849
2850 ***** PROCEDURE BODY *****
2851
2852
2853 GLB PUTOSPRITE,PUT1SPRITE
2854
2855 * BEGIN PUTOSPRITE EQU $
2856 PUTOSPRITE EQU $
2857
2858 * SPRITE_PTR := WORK_BUFFER LD SPRITE_PTR,[WORK_BUFFER]
2859
2860
080F FD2A8006

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
2861 * WITH THIS_SPRITE^,SPRITE_PTR^ DO
2862
2863 * IF (STATUS^X_LOCATION > -8) AND (STATUS^X_LOCATION < 256) AND
2864 * (STATUS^Y_LOCATION > -8) AND (STATUS^Y_LOCATION < 192) THEN
2865 LD L,[THIS_SPRITE+STATUS]
2866 LD H,[THIS_SPRITE+STATUS+1]
2867 ADD DE,X_LOCATION ; [HL] = X_LOCATION
2868 LD C,[HL]
2869 HL
2870 INC B,[HL] ; BC = X_LOCATION
2871 LD A,B ; COMPARE BC WITH -8
2872 CP 0
2873 JR Z,OK_1
2874 CP -1
2875 CP NZ,DONT_PUT
2876 LD A,C
2877 CP -7
2878 JP M,DONT_PUT
2879
2880 OK_1
2881 INC HL ; [HL] = Y_LOCATION
2882 LD C,[HL]
2883 INC HL
2884 LD B,[HL] ; BC = Y_LOCATION
2885 LD A,B ; COMPARE BC WITH -8
2886 CP 0
2887 JR Z,OK_2
2888 CP -1
2889 CP NZ,DONT_PUT
2890 LD A,C
2891 CP -7
2892 JP M,DONT_PUT
2893 OK_2
2894
2895 * IF STATUS^X_LOCATION < 0 THEN
2896 DEC HL
2897 DEC HL ; [HL] = HI(X_LOCATION)
2898 LD A,[HL] ; COMPARE WITH 0
2899 CP 0
2900 JP Z,CONTINUE
2901
2902 * X := BYTE(STATUS^X_LOCATION) + 8
2903 DEC HL ; [HL] = ^X_LOCATION
2904 LD C,[HL]
2905 INC HL
2906 LD B,[HL]
2907 LD HL,8
2908 ADD HL,BC
2909 LD A,L
2910 LD [SPRITE_PTR+X],A
2911
2912 * COLOR AND TAG := GRAPHICS^.FRAME_TABLE[STATUS^.FRAME].COLOR OR 80H
2913 LD L,[THIS_SPRITE+GRAPHICS]
2914 LD H,[THIS_SPRITE+GRAPHICS+1]
2915 DE,FRAME_TABLE_PTR
2916 ADD HL,DE ; [HL] = FRAME_TABLE_PTR
2917 EX DE,HL
2918
2919 08E3 DD6E02
2920 08E6 DD6603
2921 08E9 110001
2922 08EC 19
2923 08ED 4E
2924 08EE 23
2925 08EF 46
2926 08F0 78
2927 08F1 FE00
2928 08F3 2808
2929 08F5 FEFF
2930 08F7 C20A54
2931 08FA 79
2932 08FB FEF9
2933 08FD FA0A54
2934 0900
2935 0900 23
2936 0901 4E
2937 0902 23
2938 0903 46
2939 0904 78
2940 0905 FE00
2941 0907 2808
2942 0909 FEFF
2943 090B C20A54
2944 090E 79
2945 090F FEF9
2946 0911 FA0A54
2947 0914
2948
2949 0914 28
2950 0915 28
2951 0916 7E
2952 0917 FE00
2953 0919 CA09CA
2954
2955 091C 28
2956 091D 4E
2957 091E 23
2958 091F 46
2959 0920 210008
2960 0923 09
2961 0924 70
2962 0925 FD7701
2963
2964 0928 DD6E00
2965 092B DD6601
2966 092E 110005
2967 0931 19
2968 0932 EB

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

0933 1A 2918 LD A,[DE]
0934 6F 2919 LD L,A
0935 13 2920 INC DE
0936 1A 2921 LD A,[DE]
0937 67 2922 LD H,A
0938 E5 2923 HL ;[HL] = FRAME_TABLE_PTR^
0939 D06E02 2924 LD L,[THIS_SPRITE+STATUS]
093C D06603 2925 H,[THIS_SPRITE+STATUS+1]
093F 110000 2926 DE,FRAME
0942 19 2927 ADD HL,DE
0943 7E 2928 A,[HL] ;[HL] = FRAME
0944 C827 2929 SLA A ;CALCULATE OFFSET OF
0946 010000 2930 LD BC,0 ;COLOR ENTRY
0949 4F 2931 LD C,A
094A E1 2932 HL
0948 09 2933 HL,BC ;[HL] = COLOR
094C 7E 2934 LD A,[HL] ;OR IN 80H
094D F680 2935 OR 80H
094F FD7703 2936 LD (SPRITE_PTR+COLOR_AND_TAG),A
0952 C30A00 2937 JP PUT_Y_AND_NAME
2938 *
2939 * ELSE
2940 * ***** CONTINUE BELOW
2941
2942
2943 * BEGIN PUT1SPRITE EQU $
2944 PUT1SPRITE EQU $
2945
2946 * SPRITE_PTR := WORK_BUFFER
2947 LD SPRITE_PTR,[WORK_BUFFER]
2948
2949 * WITH THIS_SPRITE^,SPRITE_PTR^ DO
2950
2951 * IF (STATUS^.X_LOCATION > -32) AND (STATUS^.X_LOCATION < 256) AND
2952 * (STATUS^.Y_LOCATION > -32) AND (STATUS^.Y_LOCATION < 192) THEN
2953 LD L,[THIS_SPRITE+STATUS]
2954 LD H,[THIS_SPRITE+STATUS+1]
2955 LD DE,X_LOCATION
2956 ADD HL,DE ; [HL] = X_LOCATION
2957 LD C,[HL]
2958 INC HL
2959 LD B,[HL]
2960 LD A,B
2961 CP 0
2962 JR 2,OK_3
2963 CP -1
2964 JP NZ,DONT_PUT
2965 LD A,C
2966 CP -31
2967 JP M,DONT_PUT
2968 OK_3
2969 INC HL
2970 LD C,[HL]
2971 INC HL
2972 LD B,[HL]
2973 LD A,B
2974 CP 0
; [HL] = Y_LOCATION
; BC = Y_LOCATION
; COMPARE BC WITH -32
; COMPARE BC WITH -32

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
097D 2808 JR Z,OK_4
097E FEFF CP -1
0981 C20A54 JP WZ,DONT_PUT
0984 79 LD A,C
0985 FEE1 CP -31
0987 FA0A54 JP M,DONT_PUT
098A 2981 OK_4
2982
2983 * IF STATUS^X_LOCATION < 0 THEN
2984 HL
2985 HL ; [HL] = H(X_LOCATION)
2986 LD A,[HL] ; COMPARE WITH 0
2987 CP 0
2988 JP Z,CONTINUE
2989
2990 * X := BYTE(STATUS^X_LOCATION) + 32
2991 DEC HL ; [HL] = ^X_LOCATION
2992 LD C,[HL]
2993 INC HL
2994 LD B,[HL]
2995 LD HL,32
2996 ADD HL,BC
2997 LD A,L
2998 LD [SPRITE_PTR+X],A
2999
3000 * COLOR_AND_TAG := GRAPHICS^.FRAME_TABLE[STATUS^.FRAME].COLOR OR 80H
3001 LD L,[THIS_SPRITE+GRAPHICS]
3002 LD H,[THIS_SPRITE+GRAPHICS+1]
3003 DE,FRAME_TABLE_PTR
3004 ADD HL,DE ; [HL] = FRAME_TABLE_PTR
3005 EX DE,HL
3006 LD A,[DE]
3007 LD L,A
3008 INC DE
3009 LD A,[DE]
3010 LD H,A
3011 PUSH HL ; [HL] = FRAME_TABLE_PTR^
3012 LD L,[THIS_SPRITE+STATUS]
3013 LD H,[THIS_SPRITE+STATUS+1]
3014 DE,FRAME
3015 ADD HL,DE ; [HL] = FRAME
3016 LD A,[HL] ; CALCULATE OFFSET OF
3017 SLA A ; COLOR ENTRY
3018 LD BC,0
3019 LD C,A
3020 POP HL
3021 ADD HL,BC ; [HL] = COLOR
3022 LD A,[HL] ; OR IN 80H
3023 OR 80H
3024 LD [SPRITE_PTR+COLOR_AND_TAG],A
3025
3026 JR PUT_Y_AND_NAME
3027 * ELSE
3028 ***** CONTINUE FROM HERE
3029 CONTINUE
3030
3031

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

3032 *
09CA D06E02 X := BYTE(STATUS^.X_LOCATION)
3033 * LD L,[THIS_SPRITE+STATUS]
3034 LD H,[THIS_SPRITE+STATUS+1]
09CD D06603 LD DE,X_LOCATION
3035 * LD DE,X_LOCATION
0900 110001 ADD HL,DE ; [HL] = X_LOCATION
3037 * LD A,[HL]
0904 7E LD [SPRITE_PTR+X],A
3039 * LD
3040 *
0908 D06E00 COLOR_AND_TAG := GRAPHICS^.FRAME_TABLE[STATUS^.FRAME].COLOR
3041 * LD L,[THIS_SPRITE+GRAPHICS]
3042 LD H,[THIS_SPRITE+GRAPHICS+1]
090B D06601 LD DE,FRAME_TABLE_PTR
3043 * LD DE,FRAME_TABLE_PTR
090E 110005 ADD HL,DE ; [HL] = FRAME_TABLE_PTR
3045 * LD A,[DE]
09E2 E8 EX DE,HL
3046 * LD L,A
09E3 1A LD A,[DE]
3047 * INC DE
09E4 6F LD A,[DE]
3048 * LD H,A
09E5 13 LD H,A
3049 * LD H,A
09E6 1A LD H,A
3050 * LD H,A
09E7 67 LD H,A
3051 * LD H,A
09E8 E5 LD H,A
3052 * LD H,A
09E9 D06E02 PUSH
3053 * LD L,[THIS_SPRITE+STATUS]
09EC D06603 LD H,[THIS_SPRITE+STATUS+1]
3054 * LD DE,FRAME
09EF 110000 ADD HL,DE ; [HL] = FRAME
3056 * LD A,[HL]
09F2 19 LD A,[HL]
3057 * LD A,[HL]
09F3 7E LD A,[HL]
3058 * LD A,[HL]
09F4 CB27 SLA A
3059 * LD BC,0
09F5 4F LD C,A
3060 * LD C,A
09FA E1 POP HL,BC
3061 * LD A,[HL]
09FB 09 ADD [SPRITE_PTR+COLOR_AND_TAG],A
3062 * LD A,[HL]
09FC 7E LD A,[HL]
3063 * LD A,[HL]
09FD F0703 LD
3064 * LD
3065 *
3066 * END_IF
3067 PUT_Y_AND_NAME
3068 *
3069 * Y := BYTE(STATUS^.Y_LOCATION)
3070 * LD L,[THIS_SPRITE+STATUS]
3071 * LD H,[THIS_SPRITE+STATUS+1]
0A00 D06E02 LD DE,Y_LOCATION
3072 * LD DE,Y_LOCATION
0A06 110003 ADD HL,DE ; [HL] = Y_LOCATION
3073 * LD A,[HL]
0A0A 7E LD [SPRITE_PTR+Y],A
3074 * LD
3075 * LD
3076 *
3077 * NAME := GRAPHICS^.FRAME_TABLE[STATUS^.FRAME].SHAPE
3078 * + GRAPHICS^.FIRST_GEN_NAME
0A0E D06E00 LD L,[THIS_SPRITE+GRAPHICS]
3079 * LD H,[THIS_SPRITE+GRAPHICS+1]
0A11 D06601 LD DE,FRAME_TABLE_PTR
3080 * LD DE,FRAME_TABLE_PTR
0A14 110005 ADD HL,DE ; [HL] = FRAME_TABLE_PTR
3081 * LD A,[DE]
0A17 19 EX DE,HL
3082 * LD L,A
0A18 EB LD A,[DE]
3083 * LD L,A
0A19 1A LD A,[DE]
3084 * LD L,A
0A1A 6F LD A,[DE]
3085 * LD L,A
0A1B 13 LD A,[DE]
3086 * LD A,[DE]
0A1C 1A LD A,[DE]
3087 * LD A,[DE]
0A1D 67 LD H,A

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0A1E E5 3089
0A1F D06E02 3090
0A22 D06603 3091
0A25 110000 3092
0A28 19 3093
0A29 7E 3094
0A2A CB27 3095
0A2C 010000 3096
0A2F 4F 3097
0A30 E1 3098
0A31 09 3099
0A32 23 3100
0A33 7E 3101
0A34 D06E00 3102
0A37 D06601 3103
0A3A 110001 3104
0A3D 19 3105
0A3E 86 3106
0A3F FD7702 3107
3108
3109 * PUT_VRAM (0,THIS_SPRITE^.SPRITE_INDEX,SPRITE_PTR,1)
0A42 AF 3110 XOR A
0A43 1600 3111 LD D,0
0A45 D05E04 3112 LD E,[THIS_SPRITE+SPRITE_INDEX]
0A48 FDE5 3113 PUSH SPRITE_PTR
0A4A E1 3114 POP HL
0A4B FD210001 3115 LD IY,1
0A4F CD1F8E 3116 CALL PUT_VRAM
3117
0A52 1832 3118 JR EXIT_PUT_SPR
3119 * ELSE
0A54 3120 DONT_PUT ; PUT SPRITE OFF THE SCREEN BY SETTING ITS X AND EARLY CLOCK
3121
3122 * GET_VRAM (0,THIS_SPRITE^.SPRITE_INDEX,SPRITE_PTR,1)
3123 PUSH SPRITE_PTR
0A56 D0E5 3124 PUSH THIS_SPRITE
0A58 FDE5 3125 PUSH SPRITE_PTR
0A5A FDE5 3126 PUSH SPRITE_PTR
0A5C AF 3127 XOR A
0A5D 1600 3128 LD D,0
0A5F D05E04 3129 LD E,[THIS_SPRITE+SPRITE_INDEX]
0A62 E1 3130 POP HL
0A63 FD210001 3131 LD IY,1
0A67 CD1F8B 3132 CALL GET_VRAM
3133
3134 * SPRITE_PTR.X := 0
3135 LD A,0
0A6A 3E00 3136 POP SPRITE_PTR
0A6C FDE1 3137 LD [SPRITE_PTR+X],A
0A6E FD7701 3138
3139 * SPRITE_PTR.COLOR_AND_TAG := 80H
3140 LD A,80H
0A71 3E80 3141 LD [SPRITE_PTR+COLOR_AND_TAG],A
0A73 FD7703 3142
3143 * PUT_VRAM (0,THIS_SPRITE^.SPRITE_INDEX,SPRITE_PTR,1)
3144 XOR A
0A76 AF 3145 LD D,0

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
0A79	D0E1	3146	POP	THIS_SPRITE
0A7B	D05E04	3147	LD	E, (THIS_SPRITE+SPRITE_INDEX)
0A7E	E1	3148	POP	HL
0A7F	FD210001	3149	LD	1Y, 1 ;COUNT OF ONE ITEM
0A83	CD1F8E	3150	CALL	PUT_VRAM
		3151		
		3152 *	END_IF	
		3153		
		3154 *	END_PUTOSPRITE, PUT1SPRITE	
0A86		3155	EXIT_PUT_SPR	
0A86	C9	3156	RET	
		3157	PROG	

LOCATION OBJECT CODE LINE SOURCE LINE

```

3159 ; ***** MODIFIED VERSION TO RUN ON HP ASSEMBLER *****
3160 ;
3161 ;
3162 ;
3163 ;
3164 ; ***** PUT_MOBILE *****
3165 ;
3166 ;
3167 ;
3168 ;
3169 ;
3170 ;
3171 ;
3172 ;
3173 ;
3174 ;
3175 ;
3176 ;
3177 ;
3178 ;
3179 ;
3180 ;
3181 ;
3182 ;
3183 ;
3184 ;
3185 ;
3186 ;
3187 ;
3188 ;
3189 ;
3190 ;
3191 ;
3192 ;
3193 ;
3194 ;
3195 ;
3196 ;
3197 ;
3198 ;
3199 ;
3200 ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ;
3206 ;
3207 ;
3208 ;
3209 ;
3210 ;
3211 ;
3212 ;
3213 ;
3214 ;
3215 ;

```

```

4/16/82
13:50:00

```

```

3165 ; ***** PUT_MOBILE *****
3166 ;
3167 ;
3168 ;
3169 ;
3170 ;
3171 ;
3172 ;
3173 ;
3174 ;
3175 ;
3176 ;
3177 ;
3178 ;
3179 ;
3180 ;
3181 ;
3182 ;
3183 ;
3184 ;
3185 ;
3186 ;
3187 ;
3188 ;
3189 ;
3190 ;
3191 ;
3192 ;
3193 ;
3194 ;
3195 ;
3196 ;
3197 ;
3198 ;
3199 ;
3200 ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ;
3206 ;
3207 ;
3208 ;
3209 ;
3210 ;
3211 ;
3212 ;
3213 ;
3214 ;
3215 ;

```

```

3165 ; ***** PUT_MOBILE *****
3166 ;
3167 ;
3168 ;
3169 ;
3170 ;
3171 ;
3172 ;
3173 ;
3174 ;
3175 ;
3176 ;
3177 ;
3178 ;
3179 ;
3180 ;
3181 ;
3182 ;
3183 ;
3184 ;
3185 ;
3186 ;
3187 ;
3188 ;
3189 ;
3190 ;
3191 ;
3192 ;
3193 ;
3194 ;
3195 ;
3196 ;
3197 ;
3198 ;
3199 ;
3200 ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ;
3206 ;
3207 ;
3208 ;
3209 ;
3210 ;
3211 ;
3212 ;
3213 ;
3214 ;
3215 ;

```

```

3165 ; ***** PUT_MOBILE *****
3166 ;
3167 ;
3168 ;
3169 ;
3170 ;
3171 ;
3172 ;
3173 ;
3174 ;
3175 ;
3176 ;
3177 ;
3178 ;
3179 ;
3180 ;
3181 ;
3182 ;
3183 ;
3184 ;
3185 ;
3186 ;
3187 ;
3188 ;
3189 ;
3190 ;
3191 ;
3192 ;
3193 ;
3194 ;
3195 ;
3196 ;
3197 ;
3198 ;
3199 ;
3200 ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ;
3206 ;
3207 ;
3208 ;
3209 ;
3210 ;
3211 ;
3212 ;
3213 ;
3214 ;
3215 ;

```

```

3165 ; ***** PUT_MOBILE *****
3166 ;
3167 ;
3168 ;
3169 ;
3170 ;
3171 ;
3172 ;
3173 ;
3174 ;
3175 ;
3176 ;
3177 ;
3178 ;
3179 ;
3180 ;
3181 ;
3182 ;
3183 ;
3184 ;
3185 ;
3186 ;
3187 ;
3188 ;
3189 ;
3190 ;
3191 ;
3192 ;
3193 ;
3194 ;
3195 ;
3196 ;
3197 ;
3198 ;
3199 ;
3200 ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ;
3206 ;
3207 ;
3208 ;
3209 ;
3210 ;
3211 ;
3212 ;
3213 ;
3214 ;
3215 ;

```

```

3165 ; ***** PUT_MOBILE *****
3166 ;
3167 ;
3168 ;
3169 ;
3170 ;
3171 ;
3172 ;
3173 ;
3174 ;
3175 ;
3176 ;
3177 ;
3178 ;
3179 ;
3180 ;
3181 ;
3182 ;
3183 ;
3184 ;
3185 ;
3186 ;
3187 ;
3188 ;
3189 ;
3190 ;
3191 ;
3192 ;
3193 ;
3194 ;
3195 ;
3196 ;
3197 ;
3198 ;
3199 ;
3200 ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ;
3206 ;
3207 ;
3208 ;
3209 ;
3210 ;
3211 ;
3212 ;
3213 ;
3214 ;
3215 ;

```

```

3165 ; ***** PUT_MOBILE *****
3166 ;
3167 ;
3168 ;
3169 ;
3170 ;
3171 ;
3172 ;
3173 ;
3174 ;
3175 ;
3176 ;
3177 ;
3178 ;
3179 ;
3180 ;
3181 ;
3182 ;
3183 ;
3184 ;
3185 ;
3186 ;
3187 ;
3188 ;
3189 ;
3190 ;
3191 ;
3192 ;
3193 ;
3194 ;
3195 ;
3196 ;
3197 ;
3198 ;
3199 ;
3200 ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ;
3206 ;
3207 ;
3208 ;
3209 ;
3210 ;
3211 ;
3212 ;
3213 ;
3214 ;
3215 ;

```

```

3165 ; ***** PUT_MOBILE *****
3166 ;
3167 ;
3168 ;
3169 ;
3170 ;
3171 ;
3172 ;
3173 ;
3174 ;
3175 ;
3176 ;
3177 ;
3178 ;
3179 ;
3180 ;
3181 ;
3182 ;
3183 ;
3184 ;
3185 ;
3186 ;
3187 ;
3188 ;
3189 ;
3190 ;
3191 ;
3192 ;
3193 ;
3194 ;
3195 ;
3196 ;
3197 ;
3198 ;
3199 ;
3200 ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ;
3206 ;
3207 ;
3208 ;
3209 ;
3210 ;
3211 ;
3212 ;
3213 ;
3214 ;
3215 ;

```

```

3165 ; ***** PUT_MOBILE *****
3166 ;
3167 ;
3168 ;
3169 ;
3170 ;
3171 ;
3172 ;
3173 ;
3174 ;
3175 ;
3176 ;
3177 ;
3178 ;
3179 ;
3180 ;
3181 ;
3182 ;
3183 ;
3184 ;
3185 ;
3186 ;
3187 ;
3188 ;
3189 ;
3190 ;
3191 ;
3192 ;
3193 ;
3194 ;
3195 ;
3196 ;
3197 ;
3198 ;
3199 ;
3200 ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ;
3206 ;
3207 ;
3208 ;
3209 ;
3210 ;
3211 ;
3212 ;
3213 ;
3214 ;
3215 ;

```

0A87

0A87 FD2A8006

```

LOCATION OBJECT CODE LINE      SOURCE LINE
0A98 3A73C3      3216      LD A, (VDP_MODE_WORD)      ;FIND OUT WHICH GRAPHICS MODE WE ARE IN
0A9E CB4F        3217      BIT 1,A
0A90 2004        3219      IF (PSW,IS,ZERO)
0A92 CB88        3220      JR NZ,ELSE1
0A94 1802        3221      RES 7,B
0A96 CBFB        3222      JR END1
0A98 FD7003      3223      ELSE
0A9B E5         3224      SET 7,B
0A9C DD6603      3225      ENDIF
0A9E DD6603      3226      LD (Y+FLAGS),B
0A9F DD6602      3227      PUSH HL
0AA2 7E         3228      LD H, (IX+3)
0AA3 FD7704      3229      LD L, (IX+2)
0AA6 EE80        3230      LD A, (HL)
0AA8 77         3231      LD (Y+FRM),A
0AA9 23         3232      XOR 80H
0AAA 5E         3233      LD (HL),A
0AAB 78         3234      INC HL
0AAC E607        3235      LD E, (HL)
0AAE ED44        3236      LD A,E
0AB0 C608        3237      AND 7
0AB2 FD7701      3238      NEG
0AB5 23         3239      ADD A,8
0AB6 56         3240      LD (Y+XDISP),A
0AB7 CD07E8      3241      INC HL
0ABA FD7311      3242      CALL PX TO PTRN_POS
0AB8 23         3243      LD (Y+XP_BK),E
0ABE 5E         3244      INC HL
0ABF 78         3245      LD E, (HL)
0AC0 E607        3246      LD A,E
0AC2 FD7700      3247      AND 7
0AC3 23         3248      LD (Y+YDISP),A
0AC6 56         3249      INC HL
0AC7 CD07E8      3250      LD D, (HL)
0ACA FD7312      3251      CALL PX TO PTRN_POS
0ACB 23         3252      LD (Y+YP_BK),E
0ACD 2A8006      3253      LD (Y+YP_BK),E
0AD0 110013      3254      ; NOW GET THE NINE NAMES THAT CONSTITUTE THE BACKGROUND ON WHICH THE MOBILE OBJECT
0AD3 19         3255      ; WILL BE SUPERIMPOSED
0AD4 F05612      3256      LD HL, (WORK_BUFFER)
0AD7 F05E11      3257      LD DE, YP_BK+1
0ADA 010303      3258      ADD HL,DE
0AD0 CD0898      3259      LD D, (Y+YP_BK)
0AD1 F05E11      3260      LD E, (Y+XP_BK)
0AD2 F05E11      3261      LD BC, 303H
0AD3 F05E11      3262      CALL GET_BKGRND
0AD4 F05E11      3263      ; READ OLD_SCREEN INTO BUFFER AND GET COLOR AND FIRST_GEN_NAME
0AD5 F05E11      3264      LD D, (IX+5)
0AD6 F05E11      3265      LD E, (IX+4)
0AD7 F05E11      3266      LD A, (IX+6)
0AD8 F05E11      3267      POP IX
0AD9 F05E11      3268      LD Y, (WORK_BUFFER)
0ADA F05E11      3269      LD (Y+GEN),A
0ADB F05E11      3270      PUSH DE
0ADC F05E11      3271      LD HL, (WORK_BUFFER)
0ADE F05E11      3272      LD BC, XP_OS
0AED D05605      3273      ;POINT TO SPACE FOR BACKGROUND NAMES
0AE0 D05605      3274      ;D := Y PAT_POS
0AE3 D05E04      3275      ;E := X PAT_POS
0AE6 D07E06      3276      ;B := Y_EXTENT, C := X_EXTENT
0AE9 D0E1        3277      ;GET BACKGROUND NAMES
0AEB FD2A8006    3278      ;DE := OLD_SCREEN_ADDRESS
0AEE FD7705      3279      ;GET FIRST GEN NAME
0AF0 D07E06      3280      ;IX := ADDRESS OF GRAPHICS
0AF3 2A8006      3281      ;SAVE IN BUFFER
0AF6 010006      3282      ;SAVE OLD_SCREEN_ADDRESS
0AF7 010006      3283      ;HL := ADDR OF START OF BUFFER
0AF8 010006      3284      ;SPACE TO MOVE OLD_SCREEN TO

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0AF9 09 3273 ADD HL,BC
0AFA 010008 3274 LD BC,11
3275 ; ;GET 9 NAMES FROM VRAM
0AFD 7A 3276 IF [D,LT,70H] ;THEN OLD_SCREEN IS IN VRAM
0AFE FE70 3277 LD A,D
0A00 3005 3278 CP 70H
0A02 C01FE2 3279 JR NC,ELSE2
0A05 1803 3280 CALL READ_VRAM
3281 JR END2
0807 EB 3282 ELSE
0808 ED80 3283 EX DE,HL
080A 3284 LDIR
3284 END2 ;ENDIF
3285

; AT THIS POINT, IX = GRAPHICS, [SP] = OLD_SCREEN
3286 ; BACKGROUND PATTERN POSITION AND NAMES STARTING AT YP BK
3287 ; OLD SCREEN PATTERN POSITION AND NAMES STARTING AT YP_OS
3288 ; FIND ALL NAMES IN BACKGROUND WHICH BELONG TO THIS OBJECT'S PATTERN GENERATORS
3289 ; AND REPLACE WITH NAME FROM OLD_SCREEN WHICH CORRESPONDS TO THAT PATTERN POSITION
3290 ; HL := BUFFER BASE
3291 PH3
3292 LD HL,[WORK_BUFFER]
3293 LD DE,YP,BK+1
3294 ADD HL,DE
3295 EXX
3296 LD DE,[WORK_BUFFER]
3297 LD HL,YP_OS+1
3298 ADD HL,DE
3299 EX DE,HL
3300 EXX
3301 LD Y,[WORK_BUFFER]
3302 LD C,[Y+Y_GEN]
3303 DO B,9
3304 LD B,9
3305 LD A,[HL]
3306 SUB C
3307 IF [A,LT,18]
3308 JR NC,END3
3309 IF [A,GE,9]
3310 CP 9
3311 JR C,END4
3312 SUB 9
3313 ;ENDIF
3314 EXX
3315 LD L,A
3316 LD H,0
3317 ADD HL,DE
3318 LD A,[HL]
3319 EXX
3320 LD [HL],A
3321 ;ENDIF
3322 IMC HL
3323 EMD00
3324 DJNZ DLP1
3325

3326 ; NOW NEW VERSION OF BACKGROUND NAMES WILL NOT CONTAIN ANY NAMES OF THIS OBJECT
3327 ; REPLACE PREVIOUS VERSION OF OLD_SCREEN WITH THIS NEW BACKGROUND
3328 PH4
3329 LD HL,[WORK_BUFFER]
3329

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
0840 010011 3330 LD BC,XP,BK
0843 09 3331 ADD HL,BC
0844 010008 3332 LD BC,11
3333 ;
3334 LD A,D
0847 7A 3335 CP 70H
0848 FE70 3336 JR NC,ELSE5
084A 3005 3337 CALL WRITE_VRAM
084C CD1DF 3338 JR END5
084F 1802 3339 ;ELSE
0851 0851 LDIR
0851 ED80 3340 ;ENDIF
0853 0853 3341 END5
3342 ; GET THE PATTERN AND COLOR GENERATORS SPECIFIED BY THE NINE BACKGROUND NAMES
3343 ; IX = GRAPHICS
3344
0853 D0E5 3345 PUSH IX
3346
0855 ED58006 3347 PMS LD DE,(WORK_BUFFER)
0859 210013 3348 LD HL,YP_BK+1
085C 19 3349 ADD HL,DE
085D EB 3350 EX DE,HL
085E 010014 3351 LD BC,BK_PTM-8
0861 09 3352 ADD HL,BC
3353 ;
3354 LD B,9
3355 DLP2 LD A,(DE)
3356 INC DE
3357 PUSH DE
3358 LD DE,B
3359 ADD HL,DE
3360 PUSH HL
3361 LD E,A
3362 LD D,0
3363 LD C,A
3364 PUSH BC
3365 LD A,9
3366 SUB B
3367 LD B,0
3368 PMS2 SUB 3
3369 JR C,PMS1
3370 INC B
3371 JR PMS2
3372 PMS1 LD A,B
3373 LD IY,(WORK_BUFFER)
3374 ADD A,[IY+YP_BK]
3375 BIT 7,[IY+FLAGS]
3376 LD IY,1
3377 PMS6 ;IF (PSW.IS,ZERO)
3378 JR NZ,ELSE6
3379 LD A,3
3380 CALL GET_VRAM
3381 POP BC
3382 LD HL,(WORK_BUFFER)
3383 PUSH BC
3384 LD DE,BK_CLR
3385 ADD HL,DE
3386 LD E,C

```

```

;HL POINTS TO BACKGROUND POSITION AND NAMES
;NUMBER OF BYTES TO MOVE
;THEN MOVE DATA TO VRAM

```

```

;MOVE TO CPU RAM

```

```

;GET THE PATTERN AND COLOR GENERATORS SPECIFIED BY THE NINE BACKGROUND NAMES
; IX = GRAPHICS

```

```

IX ;SAVE GRAPHICS POINTER

```

```

;DE := BUFFER BASE
;DISPLACEMENT TO BEGINNING OF BKGD NAMES
;DE POINT TO FIRST BACKGROUND NAME
;HL POINTS TO BACKGROUND PAT GEN STORAGE AREA -8
; WILL BE INCREMENTED BEFORE FIRST USE

```

```

;GET NAME
;POINT TO NEXT NAME
;SAVE NAME POINTER
;INCREMENT BUFFER POINTER
;SAVE BUFFER POINTER
;INDEX INTO PATTERN TABLE
;SAVE PATTERN NAME IN C
;SAVE COUNTER
;ADD (B-9)/3 TO YP BK TO FIND
; CUT WHICH THIRD OF TABLES GENS ARE IN

```

```

;TEST GRAPHICS MODE
;NUMBER OF ELEMENTS TO READ
;THEN MODE 1

```

```

;CODE FOR PATTERN GENERATOR TABLE
;GET COUNT AND NAME

```

```

;DISPLACEMENT TO COLOR GEN AREA
;POINT TO IT
;PATTERN NAME

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
089E CB38 3387 SRL E
08A0 CB38 3388 SRL E
08A2 CB38 3389 SRL E
08A4 1600 3390 LD D,0
08A6 3E09 3391 LD A,9
08A8 90 3392 SUB B
08A9 4F 3393 LD C,A
08AA 0600 3394 LD B,0
08AC 09 3395 ADD HL,BC
08AD FD210001 3396 LD IY,1
08B1 3E04 3397 LD A,4
08B3 CD1F8B 3398 CALL GET_VRAM
08B6 1821 3399 JR END6
08B8 ELSE6 3400 ELSE6
08B8 CB2F 3401 SRA A
08BA CB2F 3402 SRA A
08BC CB2F 3403 SRA A
08BE FE03 3404 PM7
08BE FE03 3405 CP 3
08C0 3017 3406 JR NC,END7
08C2 57 3407 LD D,A
08C3 D5 3408 PUSH DE
08C4 E5 3409 PUSH HL
08C5 3E03 3410 LD A,3
08C7 CD1F8B 3411 CALL GET_VRAM
08CA E1 3412 POP HL
08CB 110068 3413 LD DE,BK CLR-BK_PTM
08CE 19 3414 ADD HL,DE
08CF D1 3415 POP DE
08D0 FD210001 3416 LD IY,1
08D4 3E04 3417 LD A,4
08D6 CD1F8B 3418 CALL GET_VRAM
08D9 3419 END7
08D9 3420 END6
08D9 C1 3421 ;ENDIF
08DA E1 3422 POP BC
08DB D1 3423 POP HL
08DC 1086 3424 ; ENDDO
3425 DJNZ D1P2
3426
3427 ; NOW THE PATTERN AND COLOR GENERATORS ARE IN THEIR RESPECTIVE BUFFERS
3428 ; SO GET THE FOUR GENERATORS FOR THIS FRAME OF THE OBJECT
3429
08DE DDE1 3430 POP IX
08DE D9 3431 ;RESTORE GRAPHICS POINTER
08E1 D05603 3432 PMS EXX
08E4 D05E02 3433 LD D, [IX+3]
08E7 D04605 3434 LD E, [IX+2]
08EA D04E04 3435 LD B, [IX+5]
08ED D9 3436 LD C, [IX+4]
08EE D0E5 3437 EXX
08F0 E1 3438 PUSH IX
08F1 FD2A8006 3439 POP HL
08F5 FD7E04 3440 LD IY, (WORK_BUFFER)
08F8 87 3441 LD A, [IY+FRH]
08F9 4F 3442 ADD A,A
3443 LD C,A
;DIVIDE NAME BY 8
;CALC POSITION IN BUFFER TO MOVE GEN TO
;COLOR GENERATOR TABLE CODE
;MUST BE MODE 11
;DIVIDE Y_POS BY 8
;A := THIRD OF TABLE, 0=1ST, 1=2ND, 2=3RD
;IF A > 2, THEN Y_POS > 23 AND THEREFORE OFF SCREEN
;DE := 256*A + NAME
;PATTERN GENERATOR TABLE CODE
;HL := PATTERN BUFFER ADDRESS
;DISPLACEMENT BETWEEN PATTERN AND COLOR BUFFERS
;HL := POINTER TO COLOR BUFFER
;CODE FOR COLOR TABLE
;RESTORE REGISTERS
;RESTORE GRAPHICS POINTER
;DE' := NEW_GEN
;BC' := PTRN_POINTER
;HL := ADDRESS OF GRAPHICS
;A := FRM #
;X2

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
08FA 0600 3444 LD B,0
08FC 110006 3445 LD DE,6
08FF 19 3446 ADD HL,DE
0C00 09 3447 ADD HL,BC
0C01 5E 3448 LD E,(HL)
0C02 23 3449 INC HL
0C03 56 3450 LD D,(HL)
0C04 2A8006 3451 LD HL,(WORK_BUFFER)
0C07 01007C 3452 LD BC,OBJ_PTN+24
0C0A 09 3453 ADD HL,BC
0C0B E5 3454 PUSH HL
0C0C C5 3455 PUSH BC
0C0D 010005 3456 LD BC,5
0C10 7A 3457 IF [D,LT,70H]
0C11 FE70 3458 LD A,D
0C13 3005 3459 CP 70H
0C15 CD1FE2 3460 JR NC,ELSE8
0C18 1803 3461 CALL READ_VRAM
0C1A 3462 JR END8
0C1A EB 3463 ELSE8
0C1A EB 3464 EX DE,HL
0C1B ED80 3465 LDIR
0C1D FD2A8006 3466 END8
0C21 C1 3467 LD IY,(WORK_BUFFER)
0C22 FD09 3468 POP BC
0C24 FD7E04 3470 LD A,[IY+4]
0C27 FD2A8006 3471 LD IY,(WORK_BUFFER)
0C28 FD7702 3472 LD [IY+COLR],A
0C2E D1 3473 PH9
0C2F 2A8006 3474 POP DE
0C32 010064 3475 LD HL,(WORK_BUFFER)
0C35 09 3476 LD BC,OBJ_PTN
0C36 0604 3477 ADD HL,BC
0C38 1A 3478 DO B,4
0C39 0DBE01 3479 DLP4
0C3C D5 3480 CP [IX+1]
0C30 3016 3481 IF (PSM,IS,CARRY)
0C3F D9 3482 JR NC,ELSE9
0C40 87 3483 EXX
0C41 87 3484 ADD A,A
0C42 87 3486 ADD A,A
0C43 6F 3487 ADD A,A
0C44 2600 3488 LD L,A
0C46 09 3489 LD H,0
0C47 E5 3490 ADD HL,BC
0C48 D9 3491 PUSH HL
0C49 D1 3492 EXX
0C4A EB 3493 POP DE
0C4B C5 3494 EX DE,HL
0C4C 010008 3495 PUSH BC
0C4F ED80 3496 LD BC,8
0C51 C1 3497 LDIR
0C52 EB 3498 POP BC
0C53 1830 3499 EX DE,HL
3500 JR END9
;FRAME POINTERS START AT THIS OFFSET
;HL := POINTS TO FRAME_POINTER FOR FRAME
;DE := ADDRESS OF FRAME NAMES
;HL := BUFFER BASE ADDRESS
;USE LOCATION FOR LAST GEN TO STORE NAMES
;SAVE FOR LATER USE
;SAVE OFFSET
;THEN NAMES ARE IN VRAM
;GET THE 4 NAMES
;NAMES IN CPU MEMORY SPACE
;GET COLOR BYTE
;OFFSET TO FIRST NAME
;A := COLOR BYTE
;COLR := COLOR BYTE
;DE := ADDRESS OF FIRST NAME IN BUFFER
;HL := BUFFER BASE ADDRESS
;START OF OBJECT'S PATTERN BUFFER
;GET 4 PATTERNS CORRESPONDING TO THE FOUR NAMES
;GET NAME
;COMPARE TO NUMGEN
;SAVE POINTER TO NAMES
;THEN NAME < NUMGEN, THEREFORE
; PATTERN PART OF GRAPHICS TABLES
;A := 8*NAME
;HL := POINTER TO PATTERN
;DE := " "
;NUMBER OF BYTES TO MOVE

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0C55 ELSE9 ;ELSE
0C55 D09601 SUB [IX+1]
0C58 D9 EXX
0C59 B7 ADD A,A
0C5A B7 ADD A,A
0C5B B7 ADD A,A
0C5C 6F LD L,A
0C5D 2600 LD H,0
0C5F 19 ADD HL,DE
0C60 E5 PUSH HL
0C61 D9 EXX
0C62 D1 POP DE
0C63 PM10 ;IF (D,LT,70H)
0C63 7A LD A,D
0C64 FE70 CP 70H
0C66 3014 JR NC,ELSE10
0C68 C5 PUSH BC
0C69 E5 PUSH HL
0C6A D5 PUSH DE
0C6B 010008 LD BC,8
0C6C D1FE2 CALL READ_VRAM
0C6D 010008 LD BC,8
0C6E E1 POP HL
0C6F 09 ADD HL,BC
0C70 C5 EX DE,HL
0C71 09 POP HL
0C72 09 ADD HL,BC
0C73 C1 POP BC
0C74 1809 JR END10
0C7C EB ;ELSE
0C7C EB EX DE,HL
0C7D C5 PUSH BC
0C7E 010008 LD BC,8
0C81 ED80 LDIR
0C83 C1 POP BC
0C84 EB EX DE,HL
0C85 ;ENDIF
0C85 END10 ;ENDIF
0C85 END9 POP DE
0C86 13 INC DE
0C87 10AF ENDDO
DJNZ DLP4
3541 ;
3542 ;
3543 ;
3544 ; COMBINE OBJECT PATTERN GENERATORS WITH BACKGROUND
3545 PM11 LD IY,(WORK_BUFFER)
3546 LD DE,(WORK_BUFFER)
3547 LD HL,BK_PTH
3548 ADD HL,DE
3549 LD C,(IY+YDISP)
3550 LD B,0
3551 ADD HL,BC
3552 PUSH HL
3553 POP IX
3554 LD HL,OBJ_PTH
3555 ADD HL,DE
3556 PUSH HL
3557 LD A,16

;NAME => NUMGEN, THEREFORE NOT PART OF ROMED GENs
;SUBTRACT NUMGEN FROM NAME

;A := 8*NAME

;HL := POINTER TO PATTERN

;THEN PATTERN IN VRAM

;SAVE BUFFER ADDRESS
;SAVE GENERATOR ADDRESS
;NUMBER BYTES TO MOVE

;INCREMENT POINTERS BY 8

;GEN ADDR := GEN ADDR + 8

;BUF ADDR := BUF ADDR + 8

;PATTERN IN CPU RAM

;DE = BUFFER BASE ADDR
;HL POINTS TO START OF BACKGROUND GENERATORS
;BC=
; DISPLACEMENT OF OBJECT BELOW PATTERN BOUNDARY
;HL POINTS TO FIRST ROW IN BACKGROUND TO BE OVERLAYED
;IX := POINTER TO ROW IN PATTERN BUFFER
;HL POINTS TO FIRST BYTE IN OBJECT'S PATTERN GENs
;USE A' AS LOOP COUNTER

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
0CA5	08	3558	EX AF,AF'
0CA6		3559	COMBINE_LOOP
0CA6	E1	3560	POP HL
0CA7	56	3561	LD D,[HL]
0CA8	23	3562	INC HL
0CA9	E5	3563	PUSH HL
0CAA	01000F	3564	LD BC,15
0CAD	09	3565	ADD HL,BC
0CAE	5E	3566	LD E,[HL]
0CAF	E8	3567	EX DE,HL
0CB0	FD4601	3568	LD B,[[Y+XDISP]
0CB3	AF	3569	XOR A
0CB4	05	3570	DEC B
0CB5	FA0CB8	3571	JP M,SHFEX
0CB8	29	3572	ADD HL,HL
0CB9	17	3573	RLA
0CBA	18F8	3574	JR SHFLP
0CB8	5F	3575	LD E,A
0CB0	CD0E2F	3576	CALL COM PAT COL
0CC0	FD7E00	3577	LD A,[[Y+YDISP]
0CC3	3C	3578	INC A
0CC4	FD7700	3579	LD [[Y+YDISP],A
0CC7	FE08	3580	IF [A,EQ,8],OR,[A,EQ,16]
0CC9	2804	3581	CP 8
0CC9	2804	3582	JR Z,IF11
0CC8	FE10	3583	CP 16
0CCD	2005	3584	JR NZ,END11
0CCF	010010	3585	LD BC,16
0CD2	DD09	3586	ADD IX,BC
0CD4	DD23	3587	:EMDIF
0CD6	08	3588	INC IX
0CD7	30	3589	EX AF,AF'
0CD8	2803	3590	DEC A
0CD8	08	3591	JR Z,C_LP_EXIT
0CD8	18C9	3592	EX AF,AF'
0CD0	E1	3593	JR COMBINE_LOOP
0CDE	FDCB037E	3594	C_LP_EXIT
0CE2	201F	3595	POP HL
0CE4	2A8006	3596	BIT 7,[[Y+FLAGS]
0CE7	010084	3597	IF [PSW,IS,ZERO]
0CEA	09	3598	JR NZ,END12
0CE8	FD5602	3599	LD HL,(WORK_BUFFER)
0CEE	FDCB034E	3600	LD BC,BK CLR
0CF2	2004	3601	ADD HL,BC
0CF4	0E0F	3602	LD D,[[Y+COLR]
0CF6	1802	3603	BIT 1,[[Y+FLAGS]
0CF8	0E00	3604	IF [PSW,IS,ZERO]
0CFA		3605	JR NZ,ELSE13
0CF8	0E00	3606	LD C,OFH
0CF8	0E00	3607	JR END13
0CFA		3608	:ELSE
0CFA	0609	3609	LD C,0
0CFC	7E	3610	:EMDIF
0CFD	A1	3611	DO B,9
		3612	LD B,9
		3613	LD A,[HL]
		3614	AND C

;POINT TO OBJECT PATTERN BYTE
;GET PATTERN BYTE

;POINT TO ADJACENT PATTERN BYTE

;HL HAS 16 BIT ROW OF OBJECT'S PATTERN
;SHIFT LEFT PATTERN BY THIS AMOUNT
;CLEAR A

;SAVE LEFT BYTE IN E

;INCREMENT YDISP

;BEGINNING OF NEXT ROW

;POINT TO NEXT GEN BYTE

;DECREMENT LOOP COUNTER

;POP POINTER OFF STACK
;TEST FOR MODE 1
;THEN UPDATE MODE 1 COLOR GEN

;OFFSET FOR COLOR BUFFER
;HL POINTS TO START OF COLOR BUFFER
;GET OBJECT COLOR
;COLOR = BACKGROUND OR TRANSPARENT ?
;THEN USE BACKGROUND COLOR

;MASK FOR COLOR OF BACKGROUND

;MASK REPLACE COLOR WITH TRANSPARENT

;CHANGE ALL 9 COLOR BYTES

;GET BACKGROUND COLOR GEN
;MASK OUT COLOR

```

LOCATION OBJECT CODE LINE SOURCE LINE
0C7E B2 3615 OR D ;ADD OBJECT COLOR1
0C7F 77 3616 LD HL,A ;UPDATE COLOR GENERATOR
0000 23 3617 INC HL ;POINT TO NEXT COLOR GEN
0001 10F9 3618 ; ENDDO
0003 3619 DJNZ DLP5
0005 3620 END12 ;ENDIF
0006 3621
0003 FD7E05 3622 ; DECIDE WHICH PART OF OBJECT'S PATTERN AND COLOR TABLES TO USE
0006 FDCB047E 3623 LD A,[(Y+F GEN) ;GET NAME OF FIRST GENERATOR IN OBJECT'S TABLE
3624 BIT 7,[(Y+FRM) ;TEST WHICH PART OF TABLE TO USE
3625 ; IF (PSW,IS,NZERO) ;THEN USE UPPER HALF OF TABLES
000A 2802 3626 JR Z,END14
000C C609 3627 ADD A,9
000E 3628 END14 ;ENDIF
3629 ; CHANGE NAMES IN BACKGROUND BUFFER TO THOSE OF THE OBJECT'S PATTERNS
3630 ; THIS WILL THEN CONSTITUTE A NEW FRAME DISPLAYING THE OBJECT AT PATTERN PLANE
3631 ; LOCATION [YP_BK], [XP_BK]
3632 LD C,A ;SAVE INDEX
3633 LD HL,[WORK_BUFFER] ;HL := BUFFER BASE
3634 LD DE,YP_BK+1 ;POINT TO FIRST BACKGROUND NAME
3635 ADD HL,DE
3636 ; DO B,9
3637 LD B,9
3638 DLP6
3639 INC A
3640 INC HL
3641 ; ENDDO
001B 10FB 3642 DJNZ DLP6
3643 ; MOVE NEWLY FORMED GENERATORS TO OBJECT'S PATTERN AND COLOR GEN TABLES
001D FDCB037E 3644 PM12 BIT 7,[(Y+FLAGS) ;TEST WHICH MODE
3645 ; IF (PSW,IS,ZERO) ;THEN MODE 1
3646 JR NZ,ELSE04
3647 LD E,C
3648 LD D,0
3649 LD HL,[WORK_BUFFER]
3650 LD BC,BK_PTN
3651 ADD HL,BC
3652 LD IX,9
3653 LD A,3
3654 CALL PUT_VRAM
3655 ; SET UP POINTERS TO OBJECT'S PATTERN NAMES AND TO COLOR GEN BYTES
3656 LD IX,[WORK_BUFFER]
3657 LD HL,[WORK_BUFFER]
3658 LD BC,BK_CLR
3659 ADD HL,BC
3660 LD IX,[WORK_BUFFER]
3661 LD BC,YP_BK+1
3662 ADD IX,BC
3663 LD B,9
3664 RPT1 ;REPEAT
004C D07E00 3664 LD A,[(IX+0)
004F D023 3665 INC IX
0051 CB3F 3666 SRL A
0053 CB3F 3667 SRL A
0055 CB3F 3668 SRL A
0057 5F 3669 LD E,A
0058 1600 3670 LD D,0
;DE := OFFSET INTO COLOR GEN TABLE

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
005A C5 3672 PUSH BC
005B 3E09 3673 LD A,9
0050 90 3674 SUB B
005E 0600 3675 LD B,0
0060 FE03 3676 DVLP
0062 3805 3677 SUB 3
0064 D603 3678 JR C,DVEX
0066 04 3679 IMC B
0067 18F7 3680 JR DVLP
0069 FD8611 3681 ADD A,[(Y+XP_BK)]
006C FE20 3682 CP 32
006E 3017 3683 JR NC,MOTOS
0070 78 3684 LD A,B
0071 FD8612 3685 ADD A,[(Y+YP_BK)]
0074 FE18 3686 CP 24
0076 300F 3687 JR NC,MOTOS
0078 D0E5 3688 PUSH HL
007A E5 3689 LD A,4
007B 3E04 3690 LD IY,1
007D FD210001 3691 CALL PUT_VRAM
0081 CD1FBE 3692 POP HL
0085 D0E1 3693 POP IX
0087 C1 3694 POP BC
0088 23 3695 IMC HL
0089 13 3696 IMC DE
008A 05 3697 DEC B
008B 78 3698 LD A,B
008C FE00 3699 CP 0
008E 208C 3700 JR NZ,RPT1
3702 ; UNTIL [B,EQ,0]
3703 ; LD IY,[WORK_BUFFER] ;RESTORE BUFFER ADDR
3704 JR END04
3705 ELSE04
3706 PH13
3707 ;REPEAT
3708 RPT2
0098 C5 3709 PUSH BC
0099 79 3710 LD A,C
009A 80 3711 ADD A,B
0098 80 3712 ADD A,B
009C 80 3713 ADD A,B
0090 4F 3714 LD C,A
009E 210000 3715 LD HL,0
00A1 110018 3716 LD DE,24
00A4 78 3717 LD A,B
00A5 50 3718 AD_LP
00A6 FA0DAC 3719 JP M,AD_EXIT
00A9 19 3720 ADD HL,DE
00AA 18F9 3721 JR AD_LP
3722
3723 AD_EXIT
00AC FD2A8006 3724 LD IY,[WORK_BUFFER]
0080 FD7E12 3725 LD A,[(Y+YP_BK)]
0083 80 3726 ADD A,B
3726 ; IF [A,I,24]
3727 CP 24
0084 FE18 3727 JR NC,END15
0086 302B 3728

```

;SAVE COUNTER
;TEST WHETHER THIS PATTERN POSITION IS
; ON THE SCREEN OR NOT (I.E. IS YP.BK
; + (B-9)/3 BETWEEN 0 AND 23, AND IS
; XP.BAK + (B-9)/3 MOD 3 BETWEEN 0 AND 31)

;A := X_POS + (B-9)/3 MOD 3

;A := Y_POS + (B-9)/3
;IS THIS POSITION ON SCREEN ?

; AND POINTERS

;CODE FOR COLOR GENERATOR TABLE
;NUMBER OF ITEMS TO SEND

;RESTORE POINTERS

; AND COUNTER
;INCREMENT POINTERS

;DECREMENT COUNTER

;RESTORE BUFFER ADDR

;SAVE COUNTER AND INDEX
;ADD 3xCOUNTER TO INDEX, WHERE NEXT GENS TO BE MOVED

;C := INDEX TO TABLE LOCATION FOR NEXT 3 GENS
;CALCULATE OFFSETS FROM START OF PATTERN
; AND COLOR GENERATOR TABLES

;HL := OFFSET FROM START OF PTRN AND COLOR BUFFERS
;GET Y_PAT_POS TO SEE WHICH THIRD OF TABLES TO USE
;EACH GROUP OF THREE IN NEXT PATTERN PLANE ROW
;'A' MUST CONTAIN VALID Y_PAT_POS [0-23]

```

LOCATION OBJECT CODE LINE SOURCE LINE
0088 CB3F 3729 SRL A
008A CB3F 3730 SRL A
008C CB3F 3731 SRL A
008E 57 3732 LD D,A
008F 59 3733 LD E,C
00C0 D5 3734 PUSH DE
00C1 01001C 3735 LD BC,BK_PTN
00C4 09 3736 ADD HL,BC
00C5 ED488006 3737 LD BC,[WORK_BUFFER] ;GET BUFFER BASE ADDR
00C9 09 3738 ADD HL,BC
00CA E5 3739 PUSH HL
00CB FD210003 3740 LD IY,3
00CF 3E03 3741 LD A,3
00D1 CD1FBE 3742 CALL PUT_VRAM
00D4 E1 3743 POP HL
00D5 110068 3744 LD DE,BK CLR-BK_PTN ;OFFSET BETWEEN BUFFERS
00D8 19 3745 ADD HL,DE
00D9 D1 3746 POP DE
00DA FD210003 3747 LD IY,3
00DE 3E04 3748 LD A,4
00E0 CD1FBE 3749 CALL PUT_VRAM
00E3 3750 END15 ;ENDIF
00E3 C1 3751 POP BC
00E4 04 3752 INC B
00E5 78 3753 LD A,B
00E6 FE03 3754 CP 3
00E8 20AE 3755 JR NZ,RPT2
00EA 3756 ; UNTIL (B,EQ,3) ; REPEAT 3 TIMES
00EA 3757 END04 ;ENDIF
3758
3759 ; RESTORE OLD SCREEN IF IT'S Y_PAT_POS AND X_PAT_POS DIFFERS FROM THE
3760 ; Y_PAT_POS AND X_PAT_POS FOR THE OBJECT
3761 PM14 LD IY,[WORK_BUFFER]
3762 LD B,[IY+XP_OS]
3763 ; IF (B,NE,80H)
3764 LD A,B
3765 CP 80H
3766 JR Z,END16
3767 LD C,[IY+YP_OS]
3768 LD H,[IY+XP_BK]
3769 LD L,[IY+YP_BK]
3770 OR A
3771 SBC HL,BC
3772 ; IF (PSW,IS,WZERO)
3773 JR Z,END17
3774 LD HL,[WORK_BUFFER]
3775 LD DE,YP_OS+1
3776 ADD HL,DE
3777 LD E,[IY+XP_OS]
3778 LD D,[IY+YP_OS]
3779 LD BC,0303H
3780 CALL PUTFRAME
3781 END17 ;ENDIF
3782 END16 ;ENDIF
3783 ; PLACE OBJECT ON SCREEN
3784 LD IY,[WORK_BUFFER]
3785 LD HL,[WORK_BUFFER] ;HL := BUFFER BASE ADDR
;THIS NUMBER / 8 INDICATES WHICH 1/3 OF
; TABLES TO USE
;DE := INDEX INTO PATTERN AND COLOR TABLES
;SAVE INDEX
;FORM POINTER TO GENERATORS IN HL
;GET BUFFER BASE ADDR
;SAVE THIS POINTER
;NUMBER OF ELEMENTS TO MOVE
;PATTERN GENERATOR TABLE CODE
;GET POINTER BACK
;OFFSET BETWEEN BUFFERS
;HL POINTS TO START OF NEXT 3 COLOR GENERATORS
;GET INDEX INTO GEN TABLES
;CODE FOR COLOR GENERATOR TABLE
;RESTORE COUNTER AND INDEX
;TEST FOR VALID OLD_SCREEN DATA
;THEN THERE IS VALID DATA
;TEST IF OS POSITION SAME AS CURRENT POSITION
;CLEAR THE CARRY
;IS THERE ANY DIFFERENCE?
;THEN POSITION HAS CHANGED
;GET BUFFER BASE
;POINT TO OLD_SCREEN NAMES
;DE := X AND Y PAT_POS
;BC := X AND Y EXTENT

```



```

LOCATION OBJECT CODE LINE      SOURCE LINE
0E1E 110013                    LD DE,YP BK+1
0E21 19                        ADD HL,DE
0E22 FD5E11                    LD E,(IY+XP BK)
0E25 FD5612                    LD D,(IY+YP BK)
0E28 010303                    LD BC,0303H
0E2B CD0808                    CALL PUTFRAME
0E2E C9                        RET
                              ***** END OF PUT_MOBILE *****
0E2F FDC80346                    ;OR' GENS OR REPLACE
                              ;THEN 'OR'
0E33 2016                        ;OR' LEFT BYTE WITH BACKGROUND
0E35 D0B600                    ;AND SUBSTITUTE FOR THAT GENERATOR BYTE
0E38 D07700                    ;NOW DO MIDDLE BYTE
0E3B 7C                        ;RIGHT HAND BYTE
0E3C D0B608                    ;REPLACE BACKGROUND WITH NON-ZERO BYTES
0E3F D07708                    ;IS BYTE NON-ZERO
0E42 7D                        ;YES, THEN REPLACE BACKGROUND WITH OBJECT
0E43 D0B610                    ;SAME FOR MIDDLE BYTE
0E46 D07710                    ;SAME FOR RIGHT HAND BYTE
0E49 1814                        ;FIND OUT WHICH GRAPHICS MODE USED
0E4B B7                        ;THEN MODE 2 (MODE 1 COLORS DONE AFTER COMBINELOOP)
                              ;SAVE BACKGROUND POINTER
                              ;CHANGE IX TO POINT TO COLOR GENERATORS
0E4C 2803                        ;GET OBJECT COLOR
0E4E D07700                    ;COLOR = BACKGROUND OR TRANSPARENT ?
0E51 7C                        ;THEN USE BACKGROUND COLOR
0E52 B7                        ;MASK FOR COLOR OF BACKGROUND
0E53 2803                        ;MASK REPLACE COLOR WITH TRANSPARENT
0E55 D07708                    ;REGS A, H AND L CONTAIN 24 BIT PATTERN TO BE COMBINED WITH BACKGROUND GENERATORS
0E58 7D                        ;IX POINTS TO THE FIRST OF THREE GENERATOR BYTES TO BE COMBINED WITH A, H AND L
0E59 B7                        ;OR' A
                              ;IF (PSW,IS,NZERO)
0E63 283C                        JR Z,END19
0E65 D0E5                        LD [IX+0],A
0E67 010068                    OR [IX+0]
0E6A D009                        LD A,H
0E6C FD4602                    OR [IX+8]
0E6F FDC8034E                    LD [IX+8],A
                              LD A,L
                              OR [IX+16]
                              LD [IX+16],A
                              JR END18
                              ;ELSE
                              OR A
                              IF (PSW,IS,NZERO)
0E73 2004                        JR Z,END19
0E75 0E0F                        LD [IX+0],A
0E77 1802                        ;ENDIF
0E79 0E00                        LD A,H
                              OR A
                              IF (PSW,IS,NZERO)
0E83 2838                        JR Z,END20
0E85 D0E5                        LD [IX+8],A
0E87 010068                    ;ENDIF
0E89 0E0F                        LD A,L
0E8B 0E00                        OR A
0E8D 0E00                        IF (PSW,IS,NZERO)
0E8F 0E00                        JR Z,END21
0E91 0E00                        LD [IX+16],A
0E93 0E00                        ;ENDIF
0E95 0E00                        LD A,L
0E97 0E00                        OR A
0E99 0E00                        IF (PSW,IS,NZERO)
0E9B 0E00                        JR Z,END22
0E9D 0E00                        LD BC,BK CLR-BK_PTN
0E9F 0E00                        ADD IX,BC
0EA1 0E00                        LD B,(IY+COLR)
0EA3 0E00                        BIT 1,(IY+FLAGS)
0EA5 0E00                        IF (PSW,IS,ZERO)
0EA7 0E00                        JR NZ,ELSE23
0EA9 0E00                        LD C,OFH
0EAB 0E00                        JR END23
0EAD 0E00                        ;ELSE
0EAF 0E00                        LD C,0
0EB1 0E00                        ;ELSE23
0EB3 0E00                        LD C,0

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0E78 3843 END23 ;ENDIF
0E7B 78 3844 LD A,E
0E7C B7 3845 OR A
0E7D 2808 3846 ;
0E7E 78 3847 JR Z,END24
0E7F D07E00 3848 LD A,[IX+0]
0E82 A1 3849 AND C
0E83 B0 3850 OR B
0E84 D07700 3851 LD [IX+0],A
0E87 7C 3852 END24 ;ENDIF
0E88 B7 3853 LD A,H
0E89 2808 3854 OR A
0E8B D07E08 3855 ; IF [PSW,IS,NZERO]
0E8E A1 3856 JR Z,END25
0E8F B0 3857 LD A,[IX+8]
0E90 D07708 3858 AND C
0E93 7D 3859 OR B
0E94 B7 3860 LD [IX+8],A
0E95 2808 3861 END25 ;ENDIF
0E97 D07E10 3862 LD A,L
0E9A A1 3863 OR A
0E9B B0 3864 ; IF [PSW,IS,NZERO]
0E9C D07710 3865 JR Z,END26
0E9F D0E1 3866 LD A,[IX+16]
0EA1 C9 3867 AND C
0EA2 3868 OR B
0EA3 3869 LD [IX+16],A
0EA4 3870 END26 ;ENDIF
0EA5 3871 POP IX
0EA6 3872 END22 ;ENDIF
0EA7 3873 RET
0EA8 3874 ; END
0EA9 3875 ; PUT_MOBILE
0EAA 3875 ; PROG

```

```

;GET FIRST OBJECT'S PATTERN BYTE
;ARE THERE ANY '1' BITS ?

```

```

;GET BACKGROUND COLOR GEN
;MASK OUT COLOR1
;ADD OBJECT COLOR1
;UPDATE COLOR GENERATOR
;SAME FOR MIDDLE BYTE

```

```

;RIGHT HAND BYTE

```

```

;RESTORE BACKGROUND POINTER

```

```

LOCATION OBJECT CODE LINE      SOURCE LINE
3877      .IDENT PUTCOMP
3878      .ZOP
3879      .EPOP
3880      .IF1 ,.INSERT B:SPZ80.ASM
3881      .COMMENT )
3882
3883
3884      ***** PUT_COMPLEX *****
3885
3886      ;DESCRIPTION:    THE POSITION AND FRAME NUMBER OF EACH OF A COMPLEX OBJECT'S
3887                         COMPONENT OBJECTS IS UPDATED. THEN PUT_OBJECT IS CALLED FOR
3888                         EACH OF THE COMPONENT OBJECTS.
3889
3890      ;INPUT:            IX = ADDRESS OF OBJECT TO BE PROCESSED
3891                         HL = ADDRESS OF OBJECT'S GRAPHICS TABLES IN ROM
3892                         B = SELECTOR FOR METHODE OF COMBINING OBJECT GENERATORS
3893                         WITH BACKGROUND GENERATORS
3894
3895                         1 = OBJECT PATTERN GENS ORed WITH BACKGROUND PATTERN GENS
3896                         COLOR1 OF BACKGROUND CHANGED TO MOBILE OBJECT'S COLOR
3897                         IF CORRESPONDING PATTERN BYTE NOT ZERO
3898
3899                         2 = REPLACE BACKGROUND PATTERN GENS WITH OBJECT PATTERN GENS
3900                         TREAT COLOR SAME AS #1
3901
3902                         3 = SAME AS #1 EXCEPT COLORO CHANGED TO TRANSPARENT
3903
3904                         4 = SAME AS #2 EXCEPT COLORO CHANGED TO TRANSPARENT
3905
3906                         C = OBJECT TYPE, AND NUMBER OF COMPONENTS
3907
3908      *****
3909      ;)
3910      EXT    PUTOBJ
3911      GLB    PUTCOMPLEX
3912
3913      PUTCOMPLEX
3914      ; UPDATE THE FRAME NUMBER AND THE X AND Y LOCATION IN EACH OF THE COMPONENT
3915      ; OBJECT'S STATUS AREAS
3916      PUSH BC
3917      EXX
3918      LD H, [IX+3]
3919      LD L, [IX+2]
3920      LD A, [HL]
3921      INC HL
3922      LD C, [HL]
3923      INC HL
3924      LD B, [HL]
3925      INC HL
3926      LD E, [HL]
3927      INC HL
3928      LD D, [HL]
3929      EXX
3930      ADD A,A
3931      ADD A,A
3932      LD E,A
3933      LD D,0
3933
DEA2
DEA2 C5                    ;SAVE SELECTOR AND COMPONENT COUNT (COMP_CNT)
DEA3 D9                    ;USE PRIMED REGS FOR X_LOC AND Y_LOC
DEA4 DD6603                ;HIGH BYTE OF STATUS
DEA7 DD6602                ;LOW BYTE OF STATUS
DEA8 7E                    ;A := FRAME
DEA9 23                    ;BC' := X_LOCATION
DEAC 4E                    ;BC' := Y_LOCATION
DEAD 23                    ;DE' := Y_LOCATION
DEAE 46                    ;FRAME := 4*FRAME
DEAF 23                    ;FORM POINTER TO FRAME AND OFFSET POINTERS
DEB0 5E                    ;FORM POINTER TO FRAME AND OFFSET POINTERS
DEB1 23                    ;FORM POINTER TO FRAME AND OFFSET POINTERS
DEB2 56                    ;FORM POINTER TO FRAME AND OFFSET POINTERS
DEB3 D9                    ;FORM POINTER TO FRAME AND OFFSET POINTERS
DEB4 87                    ;FORM POINTER TO FRAME AND OFFSET POINTERS
DEB5 B7                    ;FORM POINTER TO FRAME AND OFFSET POINTERS
DEB6 5F                    ;FORM POINTER TO FRAME AND OFFSET POINTERS
DEB7 1600                 ;FORM POINTER TO FRAME AND OFFSET POINTERS

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
0EB9 23 INC HL
0EBA 19 ADD HL,DE
0EBB 4E LD C,(HL)
0EBC 23 INC HL
0EBD 46 LD B,(HL)
0EBE 23 INC HL
0EBF 5E LD E,(HL)
0EC0 23 INC HL
0EC1 56 LD D,(HL)
0EC2 60 LD H,B
0EC3 69 LD L,C
3945 ;
3946 ; DE' = Y LOC, BC' = X LOC, HL = PNTR TO FRAME LIST, DE = PNTR TO OFFSET LIST
3947 ; IX = ADDR OF OBJ, (SP) = COMP CNT & SELECTOR
3948 ; FOR N=0 TO COMP_CNT-1: COMP_OBJ(IN) FRAME := FRAME#(IN) FROM FRAME LIST
3949 ; COMP_OBJ(IN) X_LOCATION := CNPLX_OBJ X_LOCATION + X_OFFSET(IN)
3950 ; COMP_OBJ(IN) Y_LOCATION := CNPLX_OBJ Y_LOCATION + Y_OFFSET(IN)
3951 ;
POP BC
LD A,C
LD C,B
SRL A
SRL A
SRL A
SRL A
SRL A
LD B,A
PUSH BC
PUSH IX
PUSH HL
PUSH DE
LD L,([IX+4])
LD H,([IX+5])
INC IX
INC IX
INC HL
LD E,(HL)
INC HL
LD D,(HL)
PUSH DE
POP IX
POP DE
POP HL
LD A,(HL)
BIT 7,([IX+0])
JR Z,TBLO
SET 7,A
3980
3981 TBLO:
LD ([IX+0],A
INC HL
LD A,(DE)
EXX
LD L,A
LD H,O
ADD HL,BC
LD ([IX+1],L
LD ([IX+2],H
3990
0EC4 C1
0EC5 79
0EC6 48
0EC7 CB3F
0EC9 CB3F
0ECB CB3F
0ECD CB3F
0ECF 47
0ED0 C5
0ED1 D0E5
0ED3 E5
0ED4 05
0ED5 D06E04
0ED8 D06605
0EDB D023
0EDD D023
0EDF 23
0EE0 23
0EE1 5E
0EE2 23
0EE3 56
0EE4 05
0EE5 FDE1
0EE7 D1
0EE8 E1
0EE9 7E
0EEA FDCB007E
0EEE 2802
0EF0 CBFF
0EF2 FD7700
0EF5 23
0EF6 1A
0EF7 09
0EF8 6F
0EF9 2600
0EFB 09
0EFC FD7501
0EFF FD7402
;POINT TO FIRST OF FRA OFFSET_PNTR PAIRS
;POINT TO FRAME POINTER
;BC := FRAME POINTER (POINTER TO LIST OF FRAME #'S)
;DE := OFFSET POINTER (PNTR TO LIST OF OFFSETS)
;HL := FRAME POINTER
;GET COMPONENT COUNT INTO B
;SAVE SELECTOR IN C
;GET COUNT INTO LOW NIBBLE
;B := component count
;SAVE COUNTER AND SELECTOR ON STACK
;SAVE ADDR OF OBJ
;SAVE PNTR TO FRAME LIST
;SAVE PNTR TO OFFSET LIST
;HL := ADDR OF COMPONENT OBJ
;Point to next object pointer
;HL POINTS TO STATUS POINTER
;DE := ADDR OF STATUS FOR COMPONENT OBJECT
;IX := ADDR OF STATUS FOR COMPONENT OBJECT
;DE := PNTR TO OFFSET LIST
;HL := PNTR TO FRAME LIST
;GET FRAME NUMBER
;Preserve bit 7 of frame
;used by mobile objects to indicate which
;VRAM tables in use.
;MOVE TO COMPONENTS STATUS AREA
;POINT TO NEXT FRAME NUMBER
;GET X_OFFSET
;HL' := X_OFFSET
;HL' := X_OFFSET + X_LOCATION
;COMPONENT'S X_LOCATION := X_OFFSET + X_LOCATION

```

```

LOCATION OBJECT CODE LINE      SOURCE LINE
OF02 D9                      EXX
OF03 13                      INC DE
OF04 1A                      LD A,(DE)
OF05 D9                      EXX
OF06 6F                      LD L,A
OF07 2600                    LD H,0
OF09 19                      ADD HL,DE
OF0A FD7503                   LD ([Y+3]),L
OF00 FD7404                   LD ([Y+4]),H
OF10 D9                      EXX
OF11 13                      INC DE
OF12 108F                    DJNZ LP1
                              4003 ;
                              4004 ; CALL PUT OBJECT FOR EACH OF THE COMPONENT OBJECTS, PASS SELECTOR IN B
                              4005 ; GET OBJECT ADDRESS BACK
OF14 FDE1                    POP IY
OF16 010004                   LD BC,4
OF19 FD09                    ADD IY,BC
OF1B D1                      POP DE
OF1C FD6E00                   LD L,([Y+0])
OF1F FD6601                   LD H,([Y+1])
OF22 FD23                    INC IY
OF24 FD23                    INC IY
OF26 E5                      PUSH HL
OF27 D0E1                    POP IX
OF29 FDE5                    PUSH IY
OF2B D5                      PUSH DE
OF2C 43                      LD B,E
OF20 CD1FFA                   CALL PUT08J
OF30 D1                      POP DE
OF31 FDE1                    POP IY
OF33 15                      DEC D
OF34 20E6                    JR NZ,LP2
OF36 C9                      RET
                              4023
                              4024      PROG

```

;POINT TO Y_OFFSET

;HL := Y_OFFSET
;HL := Y_OFFSET + Y_LOCATION

;COMPONENT'S Y_LOCATION := Y_OFFSET + Y_LOCATION
;POINT TO NEXT OFFSET PAIR

;CALL PUT OBJECT FOR EACH OF THE COMPONENT OBJECTS, PASS SELECTOR IN B
;GET OBJECT ADDRESS BACK

;IY POINTS TO POINTER TO FIRST COMPONENT OBJECT
;DE := COUNTER AND SELECTOR

;HL := ADDRESS OF COMPONENT OBJECT
;IY POINTS TO NEXT COMPONENT OBJECT POINTER

;IX := ADDRESS OF COMPONENT OBJECT
;SAVE POINTER
;SAVE COUNTER AND SELECTOR
;B := SELECTOR

;GET COUNTER AND SELECTOR
;GET ADDRESS OF NEXT COMPONENT OBJECT POINTER

LOCATION OBJECT CODE LINE SOURCE LINE

```

4026
4027
4028 ;
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048 DOME
4049 REPEAT
4050 FREE
4051 EOT
4052 LONG
4053
4054
4055
4056 ;
4057 ;
4058
4059
4060 ; DOME ; REPEAT ; FREE ; EOT ; LONG ;
4061 ; 7 ; 6 ; 5 ; 4 ; 3 ; 2 ; 1 ; 0 ;
4062
4063
4064
4065
4066
4067 TIME MGRQ
4068 TIME_MGR_
4069
4070 NEXT_TIMERO
4071
4072
4073
4074
4075
4076
4077
4078
4079 SCRAM
4080
4081
4082

```

```

Ken Lagace and Rob Jepson 3/82

TIMER_TABLE_BASE ;IS THIS NECESSARY ??
NEXT_TIMER_DATA_BYTE ; OR THIS???

PARAM ;Parameter passing routine needed
;for setting up pascal interfaces

;Global routine labels are the routines to call
;from pascal
INIT_TIMER
INIT_TIMERQ
FREE_SIGNAL
FREE_SIGNALQ
REQUEST_SIGNAL
REQUEST_SIGNALQ
TEST_SIGNAL
TEST_SIGNALQ
TIME_MGR
TIME_MGRQ
7
6
5
4
3

```

```

<0007>
<0006>
<0005>
<0004>
<0003>

```

```

..... MODE BIT .....
TO CHECK BIT ===== BIT ? JR Z = JUMP IF BIT IS 0 I

```

```

PROG
LD HL,[TIMER_TABLE_BASE];Current timer addr.
BIT FREE,[HL] ;Free?
CALL Z,DCR_TIMER ;if not, decr.
BIT EOT,[HL] ;End of table?
JR NZ,SCRAM ; If it is, we're done.
HL HL ;Otherwise get next timer
INC HL ; and start over.
JR NEXT_TIMERO
RET

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

4083
4084 .....
4085 .....
4086 DCR_TIMER
4087 .....
4088 .....
4089 .....
4090 .....
4091 .....
4092 DCR_L_MODE_TBL
4093 .....
4094 .....
4095 .....
4096 .....
4097 .....
4098 .....
4099 .....
4100 .....
4101 .....
4102 .....
4103 .....
4104 DCR_L_RPT_TBL
4105 .....
4106 .....
4107 .....
4108 .....
4109 .....
4110 .....
4111 .....
4112 .....
4113 .....
4114 .....
4115 .....
4116 .....
4117 .....
4118 .....
4119 .....
4120 .....
4121 .....
4122 .....
4123 .....
4124 .....
4125 .....
4126 .....
4127 .....
4128 .....
4129 DCR_S_MODE_TBL
4130 .....
4131 .....
4132 .....
4133 .....
4134 .....
4135 .....
4136 .....
4137 .....
4138 .....
4139 .....
4087 HL ;Save current timer.
4088 LONG,[HL] ;Long?
4089 Z,DCR_S_MODE_TBL;Short,non-repeating
4090 REPEAT,[HL] ;Repeat?
4091 NZ,DCR_L_RPT_TBL;Long, repeating
4092 ;Long non-repeating
4093 HL ;Move counter to DE
4094 E,[HL]
4095 HL
4096 D,[HL]
4097 DE ;Decrement.
4098 A,E
4099 D ;Check if 0.
4100 NZ,SAVE_2_BYTES ;if not, save'm.
4101 HL ;Otherwise, get mode byte
4102 HL ; and set it's done bit.
4103 SET_DOME_BIT
4104 HL ;Long-repeating timer.
4105 E,[HL] ;Load addr. into DE.
4106 HL
4107 D,[HL]
4108 D,[HL]
4109 DE,HL ;Exchange and
4110 E,[HL] ; load counter into DE.
4111 HL
4112 D,[HL]
4113 DE ;Decrement.
4114 A,E
4115 D ;Check for 0.
4116 NZ,SAVE_2_BYTES ;Save if not.
4117 HL ;Otherwise, reload
4118 E,[HL] ; original counter #.
4119 HL
4120 D,[HL] ;Jockey all over to
4121 HL ; perform said task!
4122 HL
4123 [HL],D
4124 HL
4125 [HL],E
4126 HL
4127 HL
4128 SET_DOME_BIT ;Then set done bit.
4129 HL
4130 [HL]
4131 NZ,TIMER_EXIT
4132 HL
4133 POP
4134 HL
4135 REPEAT,[HL] ;Repeat?
4136 Z,SET_DOME_BIT ;if not, leave.
4137 HL ;Otherwise, jockey
4138 HL ; around again and
4139 A,[HL] ; reload original #.

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
0F86	ZB	4140	DEC	HL
0F87	77	4141	LD	(HL),A
0F88	ZB	4142	DEC	HL
0F89	E1	4143	POP	HL
0F8A	E5	4144	PUSH	HL
0F8B		4145	SET_DOME_BIT	
0F8C	CBFE	4146	SET	7, (HL)
0F8D		4147	TIMER_EXIT	
0F8E	E1	4148	POP	HL
0F8F	C9	4149	RET	
		4150		
		4151		
		4152		
0F8F		4153	SAVE_2_BYTES	
0F8F	72	4154	LD	(HL),D
0F90	ZB	4155	DEC	HL
0F91	73	4156	LD	(HL),E
0F92	18F9	4157	JR	TIMER_EXIT
		4158		
		4159	Procedure Init Timer	
		4160	;HL has address of Timer Table	
		4161	;DE has address of Timer Data Table	
		4162		
		4163	COMN	
		4164	;INIT_TIME_DATA:	
		4165	;TEMP1:	
		4166	DEFS 2	
		4167	;TEMP2:	
		4168	DEFS 2	
		4169		
		4170	PROG	
0F94		4171	INIT_TIME_PAR:	
0F94	00020002	4172	DEFW 2,2,2	
0F98	0002			
		4173		
0F9A		4174	INIT_TIMERQ:	
0F9A	010F94	4175	LD BC,INIT_TIME_PAR	
0F9B	11738A	4176	LD DE,INIT_TIME_DATA	
0FA0	C00098	4177	CALL PARAM	
0FA3	2A738A	4178	LD HL,(TEMP1)	
0FA6	ED58738C	4179	LD DE,(TEMP2)	
0FAA		4180	INIT_TIMER :	
0FAA	227303	4181	LD (TIMER_TABLE_BASE),HL	
0FAD	3630	4182	(HL),30H	
0FAF	EB	4183	EX DE,HL	
0FB0	227305	4184	LD (NEXT_TIMER_DATA_BYTE),HL	
0FB3	C9	4185	RET	
		4186		
		4187	Procedure Free Signal	
		4188	;Acc has signal number to be Freed	
		4189	;No output is generated	
		4190	COMN	
		4191	;SIGNAL_NUM:	
		4192	DEFS 1	
		4193		
		4194		
		4195	PROG	

;Store given base address for timer table
;Set first byte in timer table to free and last timer
;Store given base address for data block

LOCATION OBJECT CODE LINE SOURCE LINE

```

4196
4197 FREE_SIG_PAR:
4198 OFB4 00010001 DEFV 1,1
4199
4200 FREE_SIGNALO:
4201 OFB8 010FB4 LD BC,FREE SIG_PAR
4202 OFB8 1173BE LD DE,SIGNAL_NUM
4203 OFBE C0098 CALL PARAM
4204 OFC1 3A73BE LD A,[SIGNAL_NUM]
4205
4206 FREE_SIGNAL:
4207 OFC4 4F C,A
4208 OFC5 2A7303 HL,[TIMER_TABLE_BASE]
4209 OFC8 47 B,A
4210 OFC9 110003 LD DE,3
4211 OFCC B7 OR A
4212 OFCD 2808 JR Z,FREE_MATCH
4213
4214 FREE1:
4215 OFCF CB66 BIT EOT,[HL]
4216 OFD1 206A JR NZ,FREE_EXIT
4217 OFD3 19 ADD HL,DE
4218 OFD4 00 DEC C
4219 OFD5 20F8 JR NZ,FREE1
4220
4221 FREE_MATCH:
4222 OFD7 CB6E BIT FREE,[HL]
4223 OFD9 2062 JR NZ,FREE_SET
4224 OFD8 CBEE SET FREE,[HL]
4225 OFD0 CB76 BIT REPEAT,[HL]
4226 OFDF 205C JR Z,FREE_SET
4227 OFE1 CB5E BIT LONG,[HL]
4228 OFE3 2858 JR Z,FREE_SET
4229 ; CALL FREE_COUNTER
4230
4231 ;
4232 ; FREE (DELETE) COUNTER
4233 ; KENL 3/82
4234 ;
4235 ; NEEDS "TO-DELETE":COUNTER ADDR.IN DE
4236
4237 FREE_COUNTER
4238 OFE5 23 INC ;HL should contain timer to delete addr.
4239 OFE6 5E LD E,[HL] ;Get next after mode byte
4240 OFE7 23 INC HL ; into DE.
4241 OFE8 56 LD D,[HL]
4242 OFE9 D5 DE ;Save them for later.
4243 OFEA 2A7303 HL,[TIMER_TABLE_BASE]
4244 OFEE E5 PUSH ;Save beginning of table.
4245 NEXT
4246 OFEE CB66 BIT ;End of table?
4247 OFF0 202E JR ;Free?
4248 OFF2 CB6E BIT ;If so we don't want it.
4249 OFF4 2023 JR NZ,GET_NEXT
4250 OFF6 7E LD A,[HL]
4251 OFF7 E648 AND 48H
4252 OFF9 FE48 CP

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
0FFB 201C	JR	4253	NZ,GET_NEXT	
0FFD 23	INC	4254	HL	;If NOT we don't want it.
OFFE 23	INC	4255	HL	
OFFF 7E	LD	4256	A,[HL]	
1000 BA	CP	4257	D	
1001 3816	JR	4258	C,GET_NEXT	;If so we don't want it.
1003 2008	JR	4259	NZ,SUBTRACT_4	;However, if larger, change it.
1005 28	DEC	4260	HL	
1006 7E	LD	4261	A,[HL]	
1007 8B	CP	4262	E	;Smaller?
1008 380F	JR	4263	C,GET_NEXT	;If so we don't want it.
100A 2031	JR	4264	Z,EXIT	;Error if equal
100C 23	INC	4265	HL	;Set up HL for SUBTRACT_4
100D		4266	SUBTRACT_4	
100E 56	LD	4267	D,[HL]	
100E 28	DEC	4268	HL	
100F 5E	LD	4269	E,[HL]	
1010 1B	DEC	4270	DE	;Reduce this addr. by 4.
1011 1B	DEC	4271	DE	
1012 1B	DEC	4272	DE	
1013 1B	DEC	4273	DE	
1014 73	LD	4274	(HL),E	;Replace reduced addr.
1015 23	INC	4275	HL	; back where we got it.
1016 72	LD	4276	(HL),D	
1017 1800	JR	4277	GET_NEXT	
1019		4278	GET_NEXT	
1019 E1	POP	4279	HL	;Now we can get next timer.
101A 23	INC	4280	HL	
101B 23	INC	4281	HL	
101C 23	INC	4282	HL	
101D E5	PUSH	4283	HL	
101E 18CE	JR	4284	NEXT	
1020		4285	MOVE_IT	
1020 0600	LD	4286	B,D	
1022 B7	OR	4287	A	;CLEAR CARRY
1023 E1	POP	4288	HL	
1024 D1	POP	4289	DE	;Get addr. of timer to delete.
1025 E5		4290	HL	
1026 2A7305	PUSH	4291	HL,(NEXT_TIMER_DATA_BYTE)	;Find # of bytes
1029 ED52	SBC	4292	HL,DE	;to move by subtraction.
102B 40	LD	4293	C,L	;Save in counter reg.
102C 68	LD	4294	L,E	;Copy into HL.
102D 62	LD	4295	H,D	
102E 23	INC	4296	HL	;Find source addr.
102F 23	INC	4297	HL	
1030 23	INC	4298	HL	
1031 23	INC	4299	HL	
1032 ED80	LDIR	4300		;Move it!
1034 010008	LD	4301	BC,8	;Adjust Next available byte by -4 from LDIR dest.
1037 ED42	SBC	4302	HL,BC	; (or -8 from source of LDIR! [saves instrs.]).
1039 227305	LD	4303	(NEXT_TIMER_DATA_BYTE),HL	
103C E1	POP	4304	HL	
1030		4305	EXIT	
		4306	;	
		4307	;	
		4308	;	
		4309	FREE_SET	

LOCATION	OBJECT CODE	LINE	SOURCE LINE
1030	;	4310	RES LONG, [HL]
1030 C9	FREE_EXIT:	4311	RET
		4312	;
		4313	;
		4314	;
		4315	;Procedure Request Signal
		4316	;HL pair has length of timer
		4317	;Acc has zero for repeating timer any other value for a non_repeating type
		4318	;Signal number is returned in the Accumulator
		4319	;
		4320	;REPEAT_SIG_CODE:
		4321	;
		4322	;TIMER_LENGTH:
		4323	;
		4324	;
		4325	PROG
		4326	;
103E	00020001	4327	REQUEST_SIG_PARAM:
1042	0002	4328	DEFM 2,1,2
1044		4329	REQUEST_SIGNALQ:
1047	01103E	4331	LD BC,REQUEST_SIG_PARAM
104A	000098	4333	CALL PARAM
104D	2A73C0	4334	LD HL,(TIMER_LENGTH)
1050	3A738F	4335	LD A,(REPEAT_SIG_CODE)
1053		4337	REQUEST_SIGNAL_:
1053 4F		4338	LD C,A
1054	E8	4339	EX DE,HL
1055	2A73D3	4340	LD HL,(TIMER_TABLE_BASE)
1058	AF	4341	XOR A
1059	47	4342	LD B,A
105A		4343	TIMER1:
105A	CB6E	4344	BIT FREE,[HL]
105C	283E	4345	JR Z,NEXT_TIMER1
105E	E5	4346	PUSH HL
		4347	PUSH AF
105F	7E	4348	LD A,[HL]
1060	E610	4349	AND 10H
1062	F620	4350	OR 20H
1064	77	4351	LD [HL],A
1065	AF	4352	XOR A
		4353	POP AF
1066	B2	4354	OR D
1067	2008	4355	JR NZ,LONG_TIMER
		4356	RES FREE,[HL]
		4357	OR D
		4358	JR NZ,LONG_TIMER
		4359	RES REPEAT,[HL]
		4360	RES LONG,[HL]
		4361	LD A,C
		4362	OR C
106A	2802	4363	JR Z,NOT_A_REPEAT_TIMER
106C	CBF6	4364	SET REPEAT,[HL]
106E		4365	NOT A_REPEAT_TIMER:

;Reset repeat bit just in case
;Return

;Put Repeat Code into C register
;Get length of timer into DE
;Get Timer Base Address
;Init offset to First Table value

;See if current timer free
;If not go get the next timer

;Reset Free bit
;Check for zero
;If non_zero then its a long timer
;Set for a NON_Repeating timer

;Check for a short repeating timer
;Don't reset repeat bit in mode byte if non_repeating
;Set repeat bit

```

LOCATION OBJECT CODE LINE SOURCE LINE
106E 23 INC HL
106F 73 LD [HL],E
1070 23 INC HL
1071 73 LD [HL],E
1072 1842 JR INIT_TIMER_EXIT
1074 SET LONG_TIMER:
1074 CBDE SET LONG, [HL]
1076 79 LD A,C
1077 B7 OR A
1078 2818 JR Z, NOT_A_LONG_REPEAT
107A D5 PUSH DE
107B E8 EX DE, HL
107C 2A7305 LD HL, [NEXT_TIMER_DATA_BYTE]
107F E8 EX DE, HL
1080 CBF6 SET REPEAT, [HL]
1082 23 INC HL
1083 73 LD [HL],E
1084 23 INC HL
1085 72 LD [HL],D
1086 EB EX DE, HL
1087 D1 POP DE
1088 73 LD [HL],E
1089 23 INC HL
108A 72 LD [HL],D
108B 23 INC HL
108C 73 LD [HL],E
108D 23 INC HL
108E 72 LD [HL],D
108F 23 INC HL
1090 227305 LD [NEXT_TIMER_DATA_BYTE], HL
1093 1821 JR INIT_TIMER_EXIT
1095 23 INC HL
1096 4399 NOT_A_LONG_REPEAT:
1096 73 INC HL
1097 23 INC HL
1098 72 LD [HL],D
1099 23 INC HL
109A 181A JR INIT_TIMER_EXIT
109C NEXT_TIMER1:
109C CB66 BIT EOT, [HL]
109E 2006 JR NZ, MAKE_NEW_TIMER
10A0 23 INC HL
10A1 23 INC HL
10A2 23 INC HL
10A3 04 INC B
10A4 1884 JR TIMER1
10A6 MAKE_NEW_TIMER:
10A6 D5 PUSH DE
10A7 E5 PUSH HL
10A8 23 INC HL
10A9 23 INC HL
10AA 23 INC HL
10AB 04 INC B
10AC 3630 LD [HL], 30H

```

;Go to next table location
;Store timer length
;Store timer length again in case of repeat
;All done so let's exit
;Set long timer bit
;Check for a long repeat timer
;if zero then go to section for non_repeating timer
;Store timer length temporarily
;Swap registers
;To get free space in long timer table
;Then swap back
;Set mode byte to repeating
;Store low byte of timer address into the value word
;Store high byte of timer address
;Move address of data area into HL
;Get back the length of timer
;Store that in the data table
;Store it again
;Store the next available data area for future use
;Store it again
;Go to next mode byte
;Count to next offset
;Go back up to init. timer
;Maximum of 255 signals allowed
;Save DE for a work register
;Save current timer address
;Go to next available memory location in the Timer Table
;Increment the signal count
;Set to free and last timer

```

LOCATION OBJECT CODE LINE SOURCE LINE
10AE E8 4423 EX DE,HL ;Save momentarily
10AF E1 4424 POP HL ;Get back original timer
1080 C8A6 4425 RES EOT,(HL) ;Reset previous last timer
1082 E8 4426 EX DE,HL ;Get back current last timer
1083 D1 4427 POP DE ;Restore DE register
1084 18A4 4428 JR TIMER1 ;Go back up and initialize counter for use
4429
1086 4430 INIT_TIMER_EXIT:
1087 C8AE 4431 POP HL
1089 78 4432 RES FREE,(HL) ;Put the offset into the Accumulator for the user of routine
108A C9 4433 LD A,B
4434 RET
4435
4436 ;Procedure Test Signal
4437 ;Acc has the Signal number to be tested
4439 ;A value of True(1) or False(0) is returned in the Accumulator for the
4440 ; Signal given.
4441 ; COMM
4442 ;TEST_SIG_NUM:
4443 ; DEFS 1
4444 ;
4445 PROG
4446 TEST_SIG_PARAM:
4447 DEFU 1,1
4448
108F 4449 TEST_SIGNALQ
108F 011088 LD BC,TEST_SIG_PARAM
10C2 1173C2 LD DE,TEST_SIG_NUM
10C5 CD0098 CALL PARAM
10C8 3A73C2 LD A,(TEST_SIG_NUM)
4454
10C8 4455 TEST_SIGNAL LD C,A
10C8 4F 4456 LD HL,(TIMER_TABLE_BASE)
10CC 2A73D3 LD B,A
10CF 47 4458 LD DE,3
1000 110003 OR A
1003 B7 4460 JR Z,SIGNAL_MATCH
1004 2808 4461
4462
1006 4463 TEST1:
1006 C866 BIT EOT,(HL)
1008 200C JR NZ,SIGNAL_FALSE
100A 19 ADD HL,DE
1008 00 DEC C
100C 20F8 JR NZ,TEST1
4469
100E 4470 SIGNAL_MATCH:
100E C86E BIT FREE,(HL)
10E0 2004 JR NZ,SIGNAL_FALSE
10E2 C87E BIT DONE,(HL)
10E4 2003 JR NZ,SIGNAL_TRUE
10E6 4475 SIGNAL_FALSE:
4476 XOR A
10E6 AF 4477 JR TEST_EXIT
10E7 180A 4478
4479
;Put Signal Code into C register
;Get Timer Base Address
;Save Signal
;Set up offset for next timer
;See if first timer is a match
;If so go check it
;Loop to match timer table to desired timer
;Check for end of table
;If so then return a not done
;Now index to next timer
;Decrement to the timer desired
;If not a timer desired timer then go back
;Here with a timer match
;Check for timer done
;If so then go return a True
;Here to return a false for either
;a not done or non_existent timer
;Put a false in Acc
;Go to the exit

```

```

LOCATION OBJECT CODE LINE      SOURCE LINE

10E9                              4480 SIGNAL_TRUE:
10E9 CB76                        4481            BIT      REPEAT, [HL]
10EB 2002                        4482            JR      NZ, SIGNAL_TRUE1
10ED CBEE                        4483            SET      FREE, [HL]
10EF                              4484 SIGNAL_TRUE1:
10EF CB8E                        4485            RES      DOME, [HL]
10F1 3E01                        4486            LD      A, 1
10F3                              4487            LD      A, 1
10F3 B7                          4488 TEST_EXIT:
10F4 C9                          4489            OR      A
                                 4490            RET
                                 4491            RET
                                 4492            DATA
                                 4493            DATA
7303                              4494            DEFBS 2
7305                              4495 TIMER_TABLE_BASE:
                                 4496 NEXT_TIMER_DATA_BYTE: DEFBS 2
                                 4497
                                 4498 PROG

;Here when timer is finished
;Check for repeating timer
;If so then just return True
;Free current timer since not repeating
;***** Start add 4/30/82****
;Reset current timer to not done
;***** End add 4/30/82****
;Put a True in the Acc
;Return

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
4500 ;CONTROLLER SOFTWARE
<0000> 4501 FIRE EQU
<0001> 4502 JOY EQU
<0002> 4503 SPIN EQU
<0003> 4504 ARM EQU
<0004> 4505 KBD EQU
<0007> 4506 PLYR_0 EQU
<0007> 4507 PLYR_1 EQU
<0007> 4508 SEG_0 EQU
<0018> 4509 SEG_1 EQU
<0000> 4510 FIRE_OLD EQU
<0001> 4511 FIRE_STATE EQU
<0002> 4512 JOY_OLD EQU
<0003> 4513 JOY_STATE EQU
<0004> 4514 SPIN_OLD EQU
<0005> 4515 SPIN_STATE EQU
<0006> 4516 ARM_OLD EQU
<0007> 4517 ARM_STATE EQU
<0008> 4518 KBD_OLD EQU
<0009> 4519 KBD_STATE EQU
<000F> 4520 KBD_MASK EQU
<0040> 4521 FIRE_MASK EQU
<000F> 4523 JOY_MASK EQU
<0080> 4524 SPIN_MASK EQU
<0005> 4525 NUM_DEV EQU
4526 ;STACK EQU
4527 ;MODE_0_PORT EQU
4528 ;MODE_1_PORT EQU
<000F> 4529 KBD_NULL EQU
<00FF> 4530 CTRL_1_PORT EQU
<00FC> 4531 CTRL_0_PORT EQU
<00C0> 4532 STRB_RST_PORT EQU
<0080> 4533 STRB_SET_PORT EQU
<0000> 4534 CONTROLLER_0 EQU
<0001> 4535 CONTROLLER_1 EQU
<0000> 4536 STROBE_RESET EQU
<0001> 4537 STROBE_SET EQU
4538 ***** MACRO *****
4539
4540
4541 DELAY_10 MACRO
4542 CALL DELAY
4543 MEMD
4544
4545 *****DATA*****
4546 *
4547 * DECODER TABLE FOR THE KEYBOARD
4548 *
4549 DEC_KBD_TBL DEFNB KBD_NULL ; NULL ENTRY
4550 DEFNB 6 ; '6'
4551 DEFNB 1 ; '1'
4552 DEFNB 3 ; '3'
4553 DEFNB 9 ; '9'
4554 DEFNB 0 ; '0'
4555 DEFNB 10 ; '10'
4556
;BITS IN STATUS WORD TO CHECK
;WHETHER DEVICE IS ACTIVE
0
1
2
3
4
5
6
7
07H
18H
;OFFSET TO CONTROLLER MEMORY
;BIT MASK IN PLAYER STATUS
;FOR SEGMENT 0 OR 1 DEVICES
;OFFSETS TO DEBOUNCE STATUS
;BUFFER: (OLD,STATE)
;MASK FOR INPUT DATA BYTE
;NUMBER OF POSSIBLE DEVICES
10110000B
73FFH
08FH ;D8
08FH ;D9
0FH
0FH
0FCH
0C0H
080H
;STROBE RESET PORT
;STROBE SET PORT
*****
MACRO *****
4539
4540
4541 DELAY_10 MACRO
4542 CALL DELAY
4543 MEMD
4544
4545 *****DATA*****
4546 *
4547 * DECODER TABLE FOR THE KEYBOARD
4548 *
4549 DEC_KBD_TBL DEFNB KBD_NULL ; NULL ENTRY
4550 DEFNB 6 ; '6'
4551 DEFNB 1 ; '1'
4552 DEFNB 3 ; '3'
4553 DEFNB 9 ; '9'
4554 DEFNB 0 ; '0'
4555 DEFNB 10 ; '10'
4556
10F5
10F6 0F
10F7 01
10F8 03
10F9 09
10FA 00
10FB 0A

```

```

OCCATION OBJECT CODE LINE      SOURCE LINE
10FC OF      4557      DEFB KBD_NULL      ; NULL ENTRY
10FD 02      4558      DEFB 2      ; '2'
10FE 08      4559      DEFB 11      ; RESET
10FF 07      4560      DEFB 7      ; '7'
1100 0F      4561      DEFB KBD_NULL      ; NULL ENTRY
1101 05      4562      DEFB 5      ; '5'
1102 04      4563      DEFB 4      ; '4'
1103 00      4564      DEFB 8      ; '8'
1104 0F      4565      DEFB KBD_NULL      ; NULL ENTRY
4566      *****SUBROUTINES*****
4567      *****
4568      GLB      CONTROLLER_INIT
4569      CONTROLLER_LIMIT      ; INITIALIZE CONTROLLER TO STROBE RESET
4570      CONTROLLER_OUT [STRB_RST_PORT],A
4571      XOR A
4572      LD IX, [CONTROLLER_MAP]
4573      INC IX
4574      INC IX
4575      INC IX
4576      LD IY, DBNCE_BUFF
4577      LD B, NUM_DEV*2
4578      * CLEAR CONTROLLER MEMORY AND DEBOUNCE STATUS BUFFER
4579      CINIT1
4580      LD [IX+0],A
4581      INC IX
4582      LD [IY+0],A
4583      INC IY
4584      LD [IY+0],A
4585      INC IY
4586      DEC B
4587      JR NZ,CINIT1
4588      * CLEAR REMAINING VARIABLES
4589      LD [SPIN_SW0_CT],A
4590      LD [SPIN_SW1_CT],A
4591      LD [S0_C0],A
4592      LD [S0_C1],A
4593      LD [S1_C0],A
4594      LD [S1_C1],A
4595      RET
4596     
4597     
4598     
4599      NOP      ; DELAY AFTER STROBE, BEFORE READ
4600      RET
4601      * CONTROLLER READ ROUTINE
4602      INPUT:
4603      *      H - CONTROLLER NUMBER
4604      *      OUTPUT:
4605      *      A - RAW DATA
4606      *
4607      *
4608     
4609      CONT_READ
4610      LD A,H
4611      CP CONTROLLER_0      ; IF CONTROLLER<0
4612      JR NZ,CONT_READ1      ; THEN READ PLAYER 1
4613      IN A, [CTRL_0_PORT]      ; ELSE READ PLAYER 0

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
1144 1802 4614 JR CONT_READX
1146 1146 4615 CONT_READ1
1146 DBFF 4616 IN A,[CTRL_1_PORT]
1148 4617 CONT_READX
1148 2F 4618 CPL
1149 C9 4619 RET
4620
4621
4622 *
4623 * CONTROLLER SCANNER ROUTINE
4624 *
4625 GLB CONT_SCAN
4626 CONT_SCAN
4627 IN A,[CTRL_0_PORT] ;READ SEGMENT 0, BOTH PLAYERS
4628 CPL
4628 2F 4629 LD [S0 C0],A
4629 3EE 4629 IN A,[CTRL_1_PORT]
4630 DBFF 4630 CPL
4631 2F 4631 LD [S0 C1],A
4632 3EF 4632 LD [STRB_SET_PORT],A ;STROBE SEGMENT 1
4633 0380 4633 OUT [STRB_SET_PORT],A ;WAIT 10 MICROSECS
4634
4635 CD1138 +
4635 DBFF 4635 IN A,[CTRL_0_PORT] ;READ SEGMENT 1, BOTH PLAYERS
4636 2F 4636 CPL
4637 3F0 4637 LD [S1 C0],A
4638 DBFF 4638 IN A,[CTRL_1_PORT]
4639 2F 4639 CPL
4640 3F3F1 4640 LD [S1 C1],A
4641 03C0 4641 OUT [STRB_RST_PORT],A ;RESET TO SEGMENT 0
4642 C9 4642 RET
4643
4644
4645 *
4646 *
4647 *
4648 *
4649
4650
4651
4652
4653 UPDATE_SPINNER_
4654 116A DBFC IN A,[CTRL_0_PORT] ;GET DATA
4655 2173EB 4655 LD HL,SPIN_SWO_CT ;ADDRESS OF SPINNER 0 COUNT
4656 CB67 4656 BIT 4,A ;IF INT BIT SET
4657 2008 4657 JR NZ,UPDATE_S1 ;THEN SPINNER 1
4658 * 4658 CHECK DIRECTION ;ELSE SPINNER 0
4659 CB6F 4659 BIT 5,A ;IF BIT 5 IS SET
4660 2003 4660 JR NZ,UPDATE_RO ;THEN GOING RIGHT
4661 ;ELSE LEFT
4662 35 4662 DEC [HL] ;DECREMENT SPINNER COUNTER
4663 1801 4663 JR UPDATE_S1 ;GO CHECK SPINNER 1
4664 *** 4664 RIGHT SPINNER SWITCH
4665 UPDATE_RO INC [HL] ;RIGHT, INCREMENT COUNTER
4666 * 4666 CHECK SPINNER 1
4667 1801 4667 * CHECK SPINNER 1
4668 DBFF 4668 UPDATE_S1 IN A,[CTRL_1_PORT] ;LOOK AT SPINNER 1 DATA
4669 CB67 4669 BIT 4,A ;IF INT BIT SET

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
117F 2009 4670 JR NZ,UPDATE_SPINX ;THEN NOT SPINNER 1
1181 23 4671 IMC HL ;ELSE SPINNER 1, BUMP HL
1182 C86F 4672 * CHECK DIRECTION ; IF BIT 5 IS SET
1184 2003 4673 BIT 5,A ; THEN GOING RIGHT
4674 JR NZ,UPDATE_R1 ; ELSE LEFT
4675 ; DECREMENT SPINNER COUNTER
1186 35 4676 DEC [HL]
1187 1801 4677 JR UPDATE_SPINX
4678 *** RIGHT SPINNER SWITCH
1189 4679 UPDATE_R1
1189 34 4680 IMC [HL] ;RIGHT, INCREMENT COUNTER
118A 4681 UPDATE_SPINX
118A C9 4682 RET
4683
4684
4685
4686 ***** DECODER ROUTINE *****
4687 * THIS ROUTINE RETURNS DECODED RAW, UNDEBOUNCED DATA *
4688 * AND MAY OR MAY NOT BE REQUIRED BY O/S *
4689 *
4690 * INPUT:
4691 * H - CONTROLLER NUMBER
4692 * L - SEGMENT NUMBER
4693 *
4694 * OUTPUT:
4695 * SEGMENT 0 SEGMENT 1
4696 * FIRE ARH
4697 * JOYSTK KBD
4698 * SPINNER
4699 *
4700 GLB DECODER_
4701 DECODER_
4702 LD A,L
4703 CP STROBE SET ;IF L=1 THEN DECODE SEGMENT 1
4704 JR Z,DEC_SEG1
4705 *
4706 * SEGMENT 0 (FIRE BUTTON, JOYSTICK)
4707 * RETURN H=FIRE BUTTON, L=JOYSTICK, E=SPINNER
4708 *
4709 * DO SPINNER FIRST
4710 LD BC,SPIN_SMO_CT
4711 LD A,H
4712 CP CONTROLLER 0 ;IF PLAYER=0 THEN GO DECODE
4713 JR Z,DEC_PLYR ;ELSE INCREMENT BC TO SPINNER 1
4714 INC BC ;GET SPINNER SWITCH COUNT
4715 DEC_PLYR LD A,[BC] ;RETURN IT IN E
4716 LD E,A
4717 XOR A ;CLEAR OUT SPINNER SWITCH COUNT
4718 LD [BC],A
4719
4720 CALL COMT_READ ;GET OTHER DEVICE DATA FOR PLAYER
4721 LD D,A ;SAVE IT
4722 AND JOY_MASK ;MASK OUT JOYSTICK DATA
4723 LD L,A ;RETURN IT IN L
4724
4725 LD A,D ;RESTORE DATA
4726 AND FIRE_MASK ;MASK OUT FIRE BUTTON DATA

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
11A7 67 4727 LD H,A ;RETURN IT IN H
11A8 1816 4728 JR DECODERX
4729 *
4730 * SEGMENT 1 (ARM BUTTON, KEYBOARD)
4731 * RETURN H=ARM BUTTON, L=KEYBOARD
4732 *
4733 DEC_SEG1
11AA 4734 OUT [STRB SET_PORT],A ;STROBE SEGMENT 1
11AA D380 4735 CALL COMT_READ ;READ SEGMENT 1 PLAYER DATA
11AC CD1130 4736 LD D,A ;SAVE IT
11AF 57 4737 OUT [STRB RST_PORT],A ;RESET BACK TO SEGMENT 0
1180 D3C0 4738 AND KBD_MASK ;MASK OUT KBD DATA
1182 E60F 4739 LD HL,DEC_KBD_TBL ;GET DECODER TABLE ADDRESS
1184 2110F5 4740 LD B,0
1187 0600 4741 LD C,A
1189 4F 4742 ADD HL,BC
118A 09 4743 LD L,[HL]
1188 6E 4744
118C 7A 4745 LD A,D
118D E640 4746 AND ARM_MASK ;RESTORE DATA
118F 67 4747 LD H,A ;MASK OUT ARM BUTTON DATA
11C0 4748 DECODERX ;RETURN IT IN H
11C0 C9 4749 RET
4750
4751
4752
4753
4754 * POLLING ROUTINE FOR ALL DEVICES IN CONTROLLER *
4755 *
4756 *
4757
4758 GLB POLLER_
4759 POLLER_
4760 CALL COMT_SCAN ;GO SCAN ALL THE DATA FIRST
4761 LD IX,DBNCE_BUFF ;DEBOUNCE BUFFER POINTER
4762 LD IX,[CONTROLLER_MAP] ;CONTROLLER MEMORY POINTER
4763 PUSH IX
4764 LD A,[IX+0] ;GET PLAYER 0 STATUS
4765 BIT 7,A ;IF PLAYER 0 NOT ACTIVE
4766 JR Z,CHK_PLYR_1 ;THEN CHECK PLAYER 1
4767 * PLAYER 0 IS ACTIVE ;ELSE
4768 LD B,A ;SAVE STATUS
4769 LD DE,PLYR_0 ;COMPUTE ADDRESS OF PLAYER_0
4770 ADD IX,DE ;CONTROLLER MEMORY
4771 AND SEG_0 ;IF SEGMENT 0 IS NOT ACTIVE
4772 JR Z,CHK_SEG_01 ;THEN CHECK SEGMENT 1
4773 * SEGMENT 0 ACTIVE ;ELSE
4774 LD A,[SO C0]
4775 LD HL,SPIN_SMO_CT
4776 CALL DECODE_0 ;DECODE DATA FOR SEGMENT 0
4777 CHK_SEG_01
4778 LD A,B ;RESTORE PLAYER 0 STATUS
4779 AND SEG_1 ;IF SEGMENT 1 IS NOT ACTIVE
4780 JR Z,CHK_PLYR_1 ;THEN CHECK PLAYER 1
4781 * SEGMENT 1 IS ACTIVE ;ELSE
4782 LD A,[S1 C0]
4783 CALL DECODE_1 ;DECODE DATA FOR SEGMENT 1

```

```

CATION OBJECT CODE LINE SOURCE LINE
11F3 4784 CHK_PLYR_1 POP IX
11F3 D0E1 4785 LD A,[IX+1]
11F5 D07E01 4786 BIT 7,A
11F8 CB7F 4787 JR Z,POLLER_X
11FA 2823 4788 * PLAYER 1 IS ACTIVE
4789 LD B,A
4790 LD DE,2*NUM_DEV
4791 ADD IY,DE
1200 FD19 4792 LD DE,PLVR_1
1202 110007 4793 ADD IX,DE
1205 D019 4794 AND SEG 0
1207 E607 4795 JR Z,CHK_SEG_11
1209 2809 4796 * SEGMENT 0 IS ACTIVE
4797 LD A,[S0 C11]
1208 3A73EF 4798 LD HL,SPIN_SW1_CT
120E 2173EC 4799 CALL DECODE_0
1211 CD1220 4800 CHK_SEG_11
1214 78 4802 LD A,B
1215 E618 4803 AND SEG 1
1217 2806 4804 JR Z,POLLER_X
4805 * SEGMENT 1 IS ACTIVE
4806 LD A,[S1 C11]
1219 3A73F1 4807 CALL DECODE_1
121C CD123F 4808 POLLER_X
121F C9 4809 RET
4810 * DECODER ROUTINE FOR SEGMENT 0
4811 LD A,[S0 C11]
4812 LD A,[S1 C11]
4813 * INPUT:
4814 A - DATA
4815 B - DEVICE STATUS BYTE FOR CURRENT PLAYER
4816 HL - ADDRESS OF SPINNER DATA
4817 IX - POINTER TO CONTROLLER MEMORY
4818 IY - POINTER TO DEBOUNCE STATUS BUFFER
4819 *
4820 DECODE_0
4821 LD C,A
1221 CB48 4822 BIT JOY,B
1223 2804 4823 JR Z,DEC_FIRE
4824 * JOYSTICK ACTIVE
4825 CALL JOY_DBNCE
1228 79 4826 LD A,C
4827 DEC_FIRE
4828 JR Z,DEC_SPMR
1229 CB40 4829 * FIRE BUTTON ACTIVE
1228 2804 4830 CALL FIRE_DBNCE
1220 CD1289 4831 LD A,C
1230 79 4832 DEC_SPMR
1231 4833 BIT SPIN,B
1231 CB50 4834 JR Z,DECODE_OX
1233 2809 4835 * SPINNER ACTIVE
4836 LD A,[HL]
1235 7E 4837 ADD A,[IX+SPIN]
1236 D06602 4838 LD [IX+SPIN],A
1239 D07702 4839 XOR A
123C AF 4840
;GET PLAYER 1 STATUS
;IF PLAYER 1 IS NOT ACTIVE
;THEN EXIT, ALL DONE
;SAVE PLAYER 1 STATUS
;COMPUTE ADDRESS OF DEBOUNCE BUFFER
;FOR PLAYER 1
;COMPUTE ADDRESS OF CONTROLLER_MEMORY
;FOR PLAYER 1
;IF SEGMENT 0 IS NOT ACTIVE
;THEN CHECK SEGMENT 1
;ELSE
;DECODE DATA FOR SEGMENT 0
;RESTORE STATUS FOR PLAYER 1
;IF SEGMENT 1 IS NOT ACTIVE
;THEN EXIT, ALL DONE
;ELSE
;DECODE DATA FOR SEGMENT 1
;SAVE DATA
;IF JOYSTICK NOT ACTIVE
;THEN CHECK FIRE BUTTON
;DEBOUNCE JOYSTICK DATA
;IF FIRE BUTTON NOT ACTIVE
;THEN CHECK SPINNER
;ELSE
;DEBOUNCE FIRE BUTTON
;IF SPINNER NOT ACTIVE
;THEN EXIT DECODER
;SAVE SPINNER COUNT
;IN CONTROLLER MEMORY

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
1230 77 4841 LD [HL],A ;CLEAR COUNTER
123E 123E C9 4842 DECODE_0X
4843 RET
4844
4845
4846 * DECODER ROUTINE FOR SEGMENT 1
4847 *
4848 *
4849 * INPUT:
4850 * A - DATA
4851 * B - DEVICE STATUS BYTE FOR CURRENT PLAYER
4852 * IX - POINTER TO CONTROLLER MEMORY
4853 * IY - POINTER TO DEBOUNCE STATUS BUFFER
4854 DECODE_1
4855 LD C,A ;SAVE DATA
4856 BIT ARM,B ;IF ARM BUTTON NOT ACTIVE
4857 JR Z,DEC_KBD ;THEN CHECK KEYBOARD
4858 * ARM BUTTON ACTIVE ;ELSE
4859 CALL ARM_DBNCE ;DEBOUNCE ARM BUTTON
4860 LD A,C
4861 DEC_KBD
4862 BIT KBD,B ;IF KEYBOARD NOT ACTIVE
4863 JR Z,DECODE_1X ;THEN EXIT DECODER
4864 * KBD ACTIVE ;DEBOUNCE KEYBOARD
4865 CALL KBD_DBNCE
4866 DECODE_1X
4867 RET
4868
4869
4870
4871 * KEYBOARD DEBOUNCE ROUTINE *
4872 *
4873 * INPUT:
4874 * A - RAW DATA
4875 * IX - CONTROLLER MEMORY POINTER
4876 * IY - DEBOUNCE STATE BUFFER
4877 *
4878 KBD_DBNCE
4879 PUSH BC
4880 PUSH DE
4881 PUSH HL
4882 AND KBD_MASK ;MASK OUT VALID DATA
4883 LD E,A ;SAVE IT
4884 LD B,[IY+KBD_OLD] ;GET OLD DATA AND CURREN STATE
4885 LD A,[IY+KBD_STATE]
4886 CP 0
4887 JR NZ,KBD_ST1 ;IF STATE <> 0
4888 * STATE = 0 ;THEN MUST BE STATE 1
4889 LD A,E ;GET CURRENT DATA
4890 CP B ;IF OLD=NEW
4891 JR Z,KBD_REG ;THEN SAW DATA TWICE IN SEQUENCE
4892 LD LD [IY+KBD_OLD],E ;ELSE FIRST TIME, SAVE CURRENT DATA
4893 JR KBD_EXIT
4894 * SAW DATA TWICE IN SEQUENCE
4895 KBD_REG
4896 LD A,1 ;SET STATE=1
4897 LD [IY+KBD_STATE],A
1250
1250 C5 4878 KBD_DBNCE
1251 D5 4879
1252 E5 4880
1253 E60F 4881
1254 F008 4882
1255 5F 4883
1256 FD4608 4884
1257 FD7E09 4885
1258 FE00 4886
1259 201A 4887
1260 78 4888 * STATE = 0
1261 B8 4889 LD A,E ;GET CURRENT DATA
1262 2805 4890 CP B ;IF OLD=NEW
1264 FD7308 4891 JR Z,KBD_REG ;THEN SAW DATA TWICE IN SEQUENCE
1267 181C 4892 LD LD [IY+KBD_OLD],E ;ELSE FIRST TIME, SAVE CURRENT DATA
4893 JR KBD_EXIT
4894 * SAW DATA TWICE IN SEQUENCE
4895 KBD_REG
4896 LD A,1 ;SET STATE=1
4897 LD [IY+KBD_STATE],A
1269 3E01 4896
1268 FD7709 4897

```

```

XCATION OBJECT CODE LINE SOURCE LINE
4898 * DECODE KEYBOARD DATA
4899 LD HL,DEC_KBD_TBL ;DECODE TABLE ADDRESS
4900 LD D,0 ;D/E RAW DATA
4901 ADD HL,DE ;COMPUTE ADDRESS INTO TABLE
4902 LD A,(HL) ;DO TABLE LOOKUP
4903 LD [(IX+KBD),A ;SAVE IN CONTROLLER MEMORY @KBD
4904 JR KBD_EXIT
4905 * STATE = 1
127A 4906 KBD_ST1
127A 78 LD A,E ;GET CURRENT DATA
127B 88 CP B ;IF OLD=NEW
127C 8007 JR Z,KBD_EXIT ;NO CHANGE IN STATE
127E FD7308 LD [(IX+KBD_OLD),E ;ELSE SAVE CURRENT DATA
XOR A ;SET STATE=0
LD [(IX+KBD_STATE),A
4913 KBD_EXIT
POP HL
POP DE
POP BC
RET
4918 * FIRE BUTTON DEBOUNCE ROUTINE *
4919
4920 *
4921 *
4922 * INPUT:
4923 * A - RAW DATA
4924 * IX - CONTROLLER MEMORY POINTER
4925 * IY - DEBOUNCE STATE BUFFER
4926 FIRE_DBNCE
1289 C5 PUSH BC
128A D5 PUSH DE
128B E640 AND FIRE_MASK ;MASK OUT VALID DATA
128D 5F LD E,A ;SAVE IT
128E F04600 LD B,[(IY+FIRE_OLD) ;GET OLD DATA AND CURRENT STATE
1291 FD7E01 LD A,[(IY+FIRE_STATE)
CP 0 ;IF STATE <> 0
JR NZ,FIRE_ST1 ;THEN MUST BE STATE 1
* STATE = 0
1298 78 LD A,E ;GET CURRENT DATA
1299 88 CP B ;IF OLD=NEW
129A 2805 JR Z,FIRE_REG ;THEN SAVE DATA TWICE IN SEQUENCE
129C FD7300 LD [(IY+FIRE_OLD),E ;ELSE FIRST TIME, SAVE CURRENT DATA
129F 1815 JR FIRE_EXIT
4941 * SAVE DATA TWICE IN SEQUENCE
4942 FIRE_REG
12A1 3E01 LD A,1 ;SET STATE=1
12A3 FD7701 LD [(IY+FIRE_STATE),A
12A6 D07300 LD [(IX+FIRE),E ;SAVE IN CONTROLLER MEMORY @FIRE
12A9 1808 JR FIRE_EXIT
4947 * STATE = 1
12AB 78 LD A,E ;GET CURRENT DATA
12AC 88 CP B ;IF OLD=NEW
12AD 2807 JR Z,FIRE_EXIT ;NO CHANGE IN STATE
12AF FD7300 LD [(IY+FIRE_OLD),E ;ELSE SAVE CURRENT DATA
XOR A ;SET STATE=0
LD [(IY+FIRE_STATE),A
4954

```

```

LOCATION OBJECT CODE LINE      SOURCE LINE

1286                            4955 FIRE_EXIT
1286 D1                        POP DE
1287 C1                        POP BC
1288 C9                        RET

4959 * JOYSTICK DEBOUNCE ROUTINE *
4960 *
4961 *
4962 *      INPUT:
4963 *      A - RAW DATA
4964 *      IX - CONTROLLER MEMORY POINTER
4965 *      IY - DEBOUNCE STATE BUFFER
4966 *

4967 JOY_DBNCE
4968      PUSH BC
4969      PUSH DE
4970      AND JOY_MASK            ;MASK OUT VALID DATA
4971      LD E,A                 ;SAVE IT
4972      LD B, [(IY+JOY_OLD)]   ;GET OLD DATA AND CURRENT STATE
4973      LD A, [(IY+JOY_STATE)]
4974      CP 0                    ;IF STATE <> 0
4975      JR NZ,JOY_ST1         ;THEN MUST BE STATE 1
4976      * STATE = 0
4977      LD A,E
4978      CP B
4979      JR Z,JOY_REG            ;GET CURRENT DATA
4980      LD [(IY+JOY_OLD),E]     ;IF OLD=NEW
4981      JR JOY_EXIT            ;THEN SAVE DATA TWICE IN SEQUENCE
4982 *      SAV DATA TWICE IN SEQUENCE
4983 JOY_REG
4984      LD A,1                    ;SET STATE=1
4985      LD [(IY+JOY_STATE),A]
4986      LD [(IX+JOY),E]         ;SAVE IN CONTROLLER MEMORY @JOY
4987      JR JOY_EXIT

4988 * STATE = 1
4989 JOY_ST1
4990      LD A,E                    ;GET CURRENT DATA
4991      CP B                      ;IF OLD=NEW
4992      JR Z,JOY_EXIT            ;NO CHANGE IN STATE
4993      LD [(IY+JOY_OLD),E]     ;ELSE SAVE CURRENT DATA
4994      XOR A
4995      LD [(IY+JOY_STATE),A]   ;SET STATE=0
4996 JOY_EXIT
4997      POP DE
4998      POP BC
4999      RET

5000 * ARM BUTTON DEBOUNCE ROUTINE *
5001 *
5002 *      INPUT:
5003 *      A - RAW DATA
5004 *      IX - CONTROLLER MEMORY POINTER
5005 *      IY - DEBOUNCE STATE BUFFER
5006 *
5007 *

5008 ARM_DBNCE
5009      PUSH BC
5010      PUSH DE
5011      AND ARM_MASK            ;MASK OUT VALID DATA

```

```

OCATION OBJECT CODE LINE SOURCE LINE
12ED 5F 5012 LD E,A ;SAVE IT
12EE F04606 5013 LD B,[(Y+ARM_OLD)] ;GET OLD DATA AND CURRENT STATE
12F1 FD7E07 5014 LD A,[(Y+ARM_STATE)] ;GET OLD DATA AND CURRENT STATE
12F4 FE00 5015 CP 0 ;IF STATE <> 0
12F6 2013 5016 JR NZ,ARM_ST1 ;THEN MUST BE STATE 1
5017 * STATE = 0 ;ELSE
5018 LD A,E ;GET CURRENT DATA
12F9 88 5019 CP B ;IF OLD=NEW
12FA 2805 5020 JR Z,ARM_REG ;THEN SAVE DATA TWICE IN SEQUENCE
12FC FD7306 5021 LD [(Y+ARM_OLD)],E ;ELSE FIRST TIME, SAVE CURRENT DATA
12FF 1815 5022 JR ARM_EXIT
5023 * SAVE DATA TWICE IN SEQUENCE
5024 ARM_REG
5025 LD A,1 ;SET STATE=1
1301 3E01 5026 LD [(Y+ARM_STATE)],A
1303 FD7707 5027 LD [(X+ARM)],E ;SAVE IN CONTROLLER MEMORY
1306 D07303 5028 JR ARM_EXIT
1309 1808 5029 * STATE = 1
1308 7B 5030 ARM_ST1
1308 7B 5031 LD A,E ;GET CURRENT DATA
130C B8 5032 CP B ;IF OLD=NEW
1300 2807 5033 JR Z,ARM_EXIT ;NO CHANGE IN STATE
130F FD7306 5034 LD [(Y+ARM_OLD)],E ;ELSE SAVE CURRENT DATA
1312 AF 5035 XOR A ;SET STATE=0
1313 FD7707 5036 LD [(Y+ARM_STATE)],A
1316 5037 ARM_EXIT
1316 D1 5038 POP DE
1317 C1 5039 POP BC
1318 C9 5040 RET
5041
5042
5043 ; EXT CONTROLLER MAP
5044 * THIS IS AN EXTERNAL POINTER ( DEFINED IN THE CARTRIDGE) TO THE
5045 * CARTRIDGE PROGRAMMER'S CONTROLLER MAP AREA.
5046
5047 * THE CARTRIDGE PROGRAMMER IS RESPONSIBLE FOR MAINTAINING THIS AREA.
5048
5049
DATA
5050 DBNCE_BUFF DEFS NUM_DEV*4
5051 SPIN_SW0_CT DEFS 1
73E8 5052 SPIN_SW1_CT DEFS 1
73EC 5053 STROBE_FLG DEFS 1
73ED 5054 S0_C0 DEFS 1
73EE 5055 S0_C1 DEFS 1
73EF 5056 S1_C0 DEFS 1
73FO 5057 S1_C1 DEFS 1
73F1 5058 PROG

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

5060 ***** EXTERNAL SYMBOLS *****
5061
5062
5063 * EXTERNAL ROUTINES FROM OS
5064
5065 ;EXT INIT TABLE
5066 ;EXT PUT_VRAM
5067 ;EXT GAME_NAME
5068 ;EXT WRITE_REGISTER
5069 ;EXT READ_REGISTER
5070 ;EXT WRITE_VRAM
5071 ;EXT START_GAME
5072
5073 ***** DEFINITIONS *****
5074
5075 MODE_0_PORT EQU 0BEH
5076 MODE_1_PORT EQU 0BFH
5077
5078 ***** EXPORTS *****
5079
5080 GLB ASCII_TBL ;POINTER TO UPPERCASE ASCII GENERATORS
5081 GLB NUMBER_TBL ;POINTER TO 0-9 GENERATORS
5082 GLB DISPLAY_LOGO ;DISPLAY COLECOVISION LOGO
5083 GLB LOAD_ASCII ;LOAD PATTERN GEN TABLE WITH FULL ASCII SET
5084 GLB FILL_VRAM_ ;FILL VRAM WITH A VALUE
5085 GLB MODE_1_ ;SET UP MODE_1 GRAPHICS
5086
5087 ***** DESCRIPTION *****
5088 *
5089 * DISPLAY_LOGO DISPLAYS THE COLECO LOGO SCREEN WITH COLECOVISION
5090 * ON A BLACK BACKGROUND. THE GAME TITLE, MANUFACTURER,
5091 * AND COPYRIGHT YEAR ARE OBTAINED FROM THE CARTRIDGE
5092 * AND OVERLAYED ONTO THE LOGO SCREEN. THE LOGO IS THEN
5093 * DISPLAYED FOR 10 SECONDS AFTER WHICH TIME A JUMP TO
5094 * THE GAME START ADDRESS IS EXECUTED.
5095 *
5096 * IF NO CARTRIDGE IS PRESENT A DEFAULT MESSAGE IS
5097 * DISPLAYED, INSTRUCTING THE OPERATOR TO:
5098 *
5099 * "TURN GAME OFF"
5100 * "BEFORE INSERTING CARTRIDGE"
5101 * "OR EXPANSION MODULE."
5102 * "[COPYRIGHT SYMBOL] 1982 COLECO"
5103 * THIS MESSAGE IS DISPLAYED FOR 60 SECONDS, THE SCREEN
5104 * IS THEN BLANKED AND FINALLY A SOFT HALT (JP $) IS
5105 * EXECUTED LOCKING UP THE PROGRAM UNTIL THE UNIT IS
5106 * RESET.
5107 *
5108 * DISPLAY LOGO EXITS WITH THE VDP IN MODE 1, THE SCREEN
5109 * BLANKED, AND THE ASCII CHARACTER SET IN VRAM.
5110 * THE MEMORY MAP IS AS FOLLOWS:
5111 *
5112 * VDP MEMORY MAP
5113 * 3800H-3FFFH SPRITE GENERATOR TABLE
5114 * 2000H-37FFFH PATTERN COLOR TABLE
5115 * 1800H-1B7FH SPRITE ATTRIBUTE TABLE
5116 * 1800H-1AFFH PATTERN NAME TABLE

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
5117 * 0000H-17FFH PATTERN GENERATOR TABLE
5118 *
5119 *****
5120 *****
5121 ***** DISPLAY LOGO *****
5122 *****
5123
5124 PROG
5125 * FILL VRAM WITH 0'S
5126 DISPLAY_LOGO LD HL,0
5127 LD DE,16384
5128 LD A,0
5129 CALL FILL_VRAM_
5130
5131 * SET UP VDP WITH MODE 1
5132 CALL MODE_1_
5133
5134 ***** WRITE OUT PATTERN GEN TABLE *****
5135
5136 * WRITE OUT ASCII GENERATOR TABLES
5137 CALL LOAD_ASCII_
5138
5139 * WRITE OUT GRAPHICS GENERATORS
5140 LD HL,OBJ_TABLE
5141 LD DE,60H
5142 WRITE_LOOP PUSH HL
5143 PUSH DE
5144 * CALCULATE GENERATOR LOCATION
5145 LD A,[HL]
5146 CP OFFH
5147 JR Z,DOME_LOGO
5148 LD B,A
5149 INC B
5150 LD HL,LOGO_GEN
5151 LD DE,8
5152 ADDR_ADJ DJNZ ADD_8
5153
5154 * DONE ADJUSTING ROM GENERATOR ADDRESS
5155
5156 POP DE
5157 PUSH DE
5158 LD IY,1
5159 LD A,3
5160 CALL PUT_VRAM
5161 POP DE
5162 POP HL
5163 INC DE
5164 INC HL
5165 JR WRITE_LOOP
5166
5167 DOME_LOGO POP DE
5168 POP HL
5169 JR WRITE_NAMES
5170
5171 ADD_8 ADD HL,DE
5172 JR ADDR_ADJ
5173
;POINT TO TABLE OF PTN GEN NUMBERS
;ITEM LOCATION IN VRAM PATTERN GEN TABLE
;SAVE LOCATION OF CURRENT CONSTRUCTION
;SAVE VRAM ITEM #
;HAVE WE PROCESSED ALL GENERATORS?
;YES. WE'RE ALL DONE
;NO. B=NUMBER FROM OBJ_TABLES
;POINT TO ROM GENERATOR TABLE
;WE'RE GOING TO ADD 8 FOR EVERY
;GENERATOR INTO THE ROM GEN TBL
;RESTORE ITEM # IN VRAM
;SAVE IT
;NUMBER OF GENERATORS
;PATTERN GEN TABLE CODE
;HL=ROM ADDRESS
;RESTORE ITEM # IN VRAM
;RESTORE CONSTRUCTION ADDRESS
;SET UP FOR NEXT ITEM
;KEEP GOING UNTIL DONE
;GOT TO POP FOR EVERY PUSH
;HOP AROUND ADD_8
;POINT TO NEXT GENERATOR

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

5174 ***** WRITE OUT PATTERN_NAME_TABLE *****
5175
1359 211440 5176 * WRITE OUT PATTERN_NAME_TABLE
135C 110085 5177 WRITE_NAMES LD HL,LOGO_NAMES ;WRITE OUT TOP HALF
135F FD210016 5178 LD DE,133 ;OF COLCOVISION
1363 3E02 5179 LD 1Y,22
1365 CD1FBE 5180 LD A,2
5181 CALL PUT_VRAM
5182
1368 211463 5181 LD HL,LOGO_NAMES+22 ;WRITE OUT BOTTOM HALF
1368 1100A5 5183 LD DE,165 ;OF COLECOVISION
136E FD210016 5185 LD 1Y,22
1372 3E02 5186 LD A,2
1374 CD1FBE 5187 CALL PUT_VRAM
5188
1377 2114C1 5188 LD HL,TRADEMARK ;WRITE OUT TM
137A 110098 5190 LD DE,155
5191 LD 1Y,2
137D FD210002 5192 LD 1Y,13
1381 3E02 5193 LD A,2
1383 CD1FBE 5194 CALL PUT_VRAM
5195
1386 2114B4 5195 * SET UP DEFAULT COPYRIGHT MESSAGE
1389 1102AA 5197 LD HL,LOGO_NAMES+103 ;WRITE OUT c 1982 COLECO
138C FD210040 5198 LD DE,682
5199 LD 1Y,13
1390 3E02 5200 LD A,2
1392 CD1FBE 5201 CALL PUT_VRAM
5202
5203 ***** WRITE OUT COLOR_NAME_TABLE *****
5204
1395 211438 5204 LD HL,LOGO_COLORS
1398 110080 5205 LD DE,0
1398 3E04 5206 LD A,4
139D FD210012 5207 LD 1Y,18
13A1 CD1FBE 5208 CALL PUT_VRAM
5209
5210 ***** ENABLE DISPLAY *****
5211
13A4 0601 5211 * ENABLE DISPLAY
13A6 DECO 5212 LD B,1
13A8 CD1FD9 5213 CALL WRITE_REGISTER
5214
13AB 218000 5214 LD HL,8000H ;IF A CARTRIDGE IS PRESENT
13AF 7E 5220 LD A,(HL) ;LOCATION 8000H WILL = 0AAH
13B1 204C 5221 CP 0AAH ;AND 8000H WILL = 55H
13B3 23 5222 JR NZ,NO_CARTRIDGE ; NOT PRESENT
13B4 7E 5223 INC HL
13B5 FE55 5224 LD A,(HL)
13B7 2046 5225 CP 55H
5226 JR NZ,NO_CARTRIDGE
5227 ; NOT PRESENT
5228 * CARTRIDGE PRESENT
5229 * DISPLAY GAME TITLE
5230 LD HL,GAME_NAME

```

OCATION	OBJECT CODE	LINE	SOURCE LINE
13BC	CD1946	5231	CALL_PARSE
13BF	118024	5232	LD DE, GAME_NAME
13C2	210201	5233	LD HL, 513
13C5	CD1951	5234	CALL CENTER_PRT
		5235	
		5236	* DISPLAY COMPANY NAME
13CB	218024	5237	LD HL, GAME_NAME
13CB	CD1946	5238	CALL_PARSE
13CE	23	5239	INC HL
13CF	54	5240	LD D, H
1300	50	5241	LD E, L
1301	CD1946	5242	CALL_PARSE
1304	2101C1	5243	LD HL, 449
1307	CD1951	5244	CALL_CENTER_PRT
		5245	
		5246	* CHANGE DATE
130A	218024	5247	LD HL, GAME_NAME
1300	CD1946	5248	CALL_PARSE
13E0	23	5249	INC HL
13E1	CD1946	5250	CALL_PARSE
13E4	23	5251	INC HL
13E5	1102AC	5252	LD DE, 684
13E8	FD210004	5253	LD IV, 4
13EC	3E02	5254	LD A, 2
13EE	CD1F8E	5255	CALL_PUT_VRAM
		5256	
		5257	* DISPLAY 10 SECONDS
13F1	CD1968	5258	CALL_DELAY_10
		5259	
		5260	* TURN OFF DISPLAY
13F4	0601	5261	LD B, 1
13F6	0E80	5262	LD C, 10000000B
13F8	CD1FD9	5263	CALL_WRITE_REGISTER
		5264	
		5265	* EXIT LOGO
13F8	2A800A	5266	LD HL, [START_GAME]
13FE	E9	5267	JP [HL]
		5268	
		5269	* WRITE OUT PATTERN NAME TABLE
13FF	211479	5270	NO_CARTRIDGE
1402	1101A4	5271	LD HL, LOGO_NAMES+44
1405	FD210000	5272	LD DE, 426
1409	3E02	5273	LD IV, 13
140B	CD1F8E	5274	LD A, 2
		5275	CALL_PUT_VRAM
		5276	
140E	211486	5276	LD HL, LOGO_NAMES+57
1411	1101E4	5277	LD DE, 484
1414	FD21001A	5278	LD IV, 26
1418	3E02	5279	LD A, 2
141A	CD1F8E	5280	CALL_PUT_VRAM
		5281	
141D	2114A0	5281	LD HL, LOGO_NAMES+83
1420	110227	5282	LD DE, 551
1423	FD210014	5284	LD IV, 20
1427	3E02	5285	LD A, 2
1429	CD1F8E	5286	CALL_PUT_VRAM
		5287	

```

;GET LENGTH OF STRING
;STARTING LOCATION OF 1ST STRING
;LOCATION (0,767) TO START PRINTING
;PRINT IT

```

```

;GET PAST 1ST STRING
;STARTING LOCATION OF 2ND STRING
;SAVED IN DE
;GET LENGTH OF STRING
;LOCATION (0,767) TO START PRINTING
;PRINT IT

```

```

;USE PARSE TO ADVANCE HL TO
;COPYRIGHT YEAR

```

```

;SCREEN LOCATION
;# OF DIGITS

```

```

;A CARTRIDGE WAS PRESENT
;SO JUMP TO IT

```

```

;NO CARTRIDGE PRESENT
;DISPLAY DEFAULT
;MESSAGE

```

```

;"TURN GAME OFF"
;"TO INSERT CARTRIDGE"
;"OR EXPANSION MODULE"

```

LOCATION	OBJECT CODE LINE	SOURCE LINE
142C 218A00	5288	LD HL,8A00H
142F CD1968	5289	CALL TIMER_1
1432 0601	5290	* TURN OFF DISPLAY ;DISPLAY 60 SECONDS
1434 0E80	5291	LD B,1 ;BLANK SCREEN
1436 CD1FD9	5292	LD C,1000000B
1439 18FE	5293	CALL WRITE_REGISTER
	5294	JR \$;SOFT HALT
	5295	
	5296	*****
	5297	* DATA TABLES
	5298	*
	5299	*
	5300	*****
	5301	***** COLOR_NAME_TABLE *****
	5302	*****
143B 000000F0F0	5303	5304 LOGO_COLORS HEX 00,00,00,FD,FD,FD,FD,FD,FD,FD,FD,FD,FD,FD,FD,FD
1440 F0F0F0F0F0	5305	HEX D0,80,90,80,30,40
1445 F0F0	5306	
1447 0080908030	5307	***** PATTERN_NAME_TABLE *****
144C 40	5308	
	5309	LOGO_NAMES EQU \$
	5310	HEX 60,61,68,69,70,71,78,79,80,81,88,89
144D 6061686970	5311	HEX 64,65,6C,74,75,7C,84,85,8C,8D
1452 7178798081	5312	HEX 62,63,6A,68,72,73,7A,7B,82,83,8A
1457 8889	5313	HEX 88,66,67,6D,76,77,7D,86,87,8E,8F
1459 64656C7475	5314	
145E 7C84858C8D	5315	
1463 62636A6872	5316	DEFB "TURN GAME OFF"
1468 737A788283	5317	DEFB "BEFORE INSERTING CARTRIDGE"
146D 8A	5318	DEFB "OR EXPANSION MODULE."
146E 8866676D76	5319	
1473 777D86878E	5320	DEFB 1DH, " 1982 COLECO"
1478 8F	5321	
1479 5455524E20	5322	TRADEMARK HEX 1E,1F
147E 47414D4520		
1483 4F6666		
1486 4245464F52		
1488 4520494E53		
1490 455254494E		
1495 4720434152		
149A 5452494447		
149F 45		
14A0 4F52204558		
14A5 50414E5349		
14AA 4F4E204D4F		
14AF 44554C452E		
14B4 1D20313938		
14B9 3220434F4C		
14BE 45434F		

LOCATION OBJECT CODE LINE SOURCE LINE

LOCATION	OBJECT CODE	LINE	SOURCE LINE	EQU \$
5323				
5324				
5325				
5326				
5327				
5328				
5329	LOGO_GEN			
5330				
5331	*GR0			00,00,00,00,00,00,00
5332				
5333	*GR1			3F,7F,FF,FF,F3,F3,F0,F0
5334				
5335	*GR2			00,80,C0,C0,C0,C0,00,00
5336				
5337	*GR3			3F,7F,FF,FF,F3,F3,F3,F3
5338				
5339	*GR4			00,80,C0,C0,C0,C0,C0,C0
5340				
5341	*GR5			F0,F0,F0,F0,F0,F0,F0,F0
5342				
5343	*GR6			FF,FF,FF,F0,F0,FF,FF,FF
5344				
5345	*GR7			C0,C0,C0,00,00,00,00,00
5346				
5347	*GR8			F1,F1,F1,7B,7B,7B,3F,3F
5348				
5349	*GR9			E0,E0,E0,C0,C0,C0,80,80
5350				
5351	*GR10			1F,3F,7F,79,7B,7F,7F,3F
5352				
5353	*GR11			80,C0,E0,E0,00,80,C0,E0
5354				
5355	*GR12			F3,F3,FB,FB,FF,FF,FF,FF
5356				
5357	*GR13			C0,C0,C0,C0,C0,C0,C0,C0
5358				
5359	*GR14			F3,F3,FF,FF,7F,3F,00,00
5360				
5361	*GR15			C0,C0,C0,C0,80,00,00,00
5362				
5363	*GR16			

<14C3> ***** PATTERN GENERATOR TABLES *****

LOCATION	OBJECT CODE LINE	SOURCE LINE	
1543	F0F0FFFFF	5364	HEX F0,F0,FF,FF,FF,FF,00,00
1548	FF0000		
	5365 *GR17		
1548	0000C0C0C0	5366	HEX 00,00,C0,C0,C0,C0,00,00
1550	C00000		
	5367 *GR18		
1553	3F1F1F0E	5368	HEX 3F,1F,1F,0E,0E,00,00
1558	0E0000		
	5369 *GR19		
1558	8000000000	5370	HEX 80,00,00,00,00,00,00,00
1560	000000		
	5371 *GR20		
1563	F0F0F0F0	5372	HEX F0,F0,F0,F0,F0,F0,00,00
1568	F00000		
	5373 *GR21		
1568	1F01797F3F	5374	HEX 1F,01,79,7F,3F,1F,00,00
1570	1F0000		
	5375 *GR22		
1573	E0E0E0C0	5376	HEX E0,E0,E0,E0,C0,80,00,00
1578	800000		
	5377 *GR23		
1578	FF7F7F7F3F	5378	HEX FF,F7,F7,F7,F3,F3,00,00
1580	F30000		
	5379 *GR24		
1583	C0C0C0C0C0	5380	HEX C0,C0,C0,C0,C0,C0,00,00
1588	C00000		
	5381		
	<1588>		
	5382 ASC TABLE		
	5383 * C=1D		
1588	7E8180A1A1	5384	HEX 7E,81,8D,A1,A1,8D,81,7E ;COPYRIGHT
1590	80817E		
	5385 * t=1E		
1593	1F04040400	5386	HEX 1F,4,4,4,0,0,0,0 ;TRADE
1598	000000		
	5387 * #=1F		
1598	44C545400	5388	HEX 44,6C,54,54,0,0,0,0 ;MARK
15A0	000000		
	5389 * =20		
15A3	0000000000	5390	DEFB 0,0,0,0,0,0,0,0
15A8	000000		
	5391 * l=21		
15AB	2020202020	5392	DEFB 20H,20H,20H,20H,20H,20H,0,20H,00H
1580	002000		
	5393 * "#=22		
1583	5050500000	5394	DEFB 50H,50H,50H,0,0,0,0,0
1588	000000		
	5395 * #=23		
1588	5050F850F8	5396	DEFB 50H,50H,0F8H,50H,0F8H,50H,50H,0
15C0	505000		
	5397 * \$=24		
15C3	2078A07028	5398	DEFB 20H,78H,0A0H,70H,28H,0F0H,020H,0
15C8	F02000		
	5399 * %=25		
15CB	C0C8102040	5400	DEFB 0C0H,0C8H,10H,20H,40H,98H,18H,0
15D0	981800		
	5401 * &=26		
15D3	40A0A040A8	5402	DEFB 40H,0A0H,0A0H,40H,0A8H,90H,68H,0

OBJECT CODE LINE SOURCE LINE

1508	906800						
		5403 * 1=27					
1508	2020200000	5404	DEFB	20H,20H,20H,0,0,0,0,0			
15E0	0000000						
		5405 * (=28					
15E3	2040808080	5406	DEFB	20H,40H,80H,80H,80H,40H,20H,0			
15E8	402000						
		5407 *)=29					
15E8	2010080808	5408	DEFB	20H,10H,08H,08H,08H,10H,20H,0			
15F0	102000						
		5409 * *=2A					
15F3	20A8702070	5410	DEFB	20H,0A8H,70H,20H,70H,0A8H,20H,0			
15F8	A82000						
		5411 * +=2B					
15F8	002020F820	5412	DEFB	0,20H,20H,0F8H,20H,20H,0,0			
1600	200000						
		5413 * ,=2C					
1603	0000000020	5414	DEFB	0,0,0,0,20H,20H,40H,0			
1608	204000						
		5415 * -=2D					
1608	000000F800	5416	DEFB	0,0,0,0,0F8H,0,0,0,0			
1610	000000						
		5417 * .=2E					
1613	0000000000	5418	DEFB	0,0,0,0,0,0,20H,0			
1618	002000						
		5419 * /=2F					
1618	0008102040	5420	DEFB	0,8,10H,20H,40H,80H,0,0			
1620	800000						
		5421					
	<1623>	5422 NUMBER_TBL	EQJ \$				
		5423 * 0=30					
1623	708898A8C8	5424	DEFB	70H,88H,98H,0A8H,0C8H,088H,70H,0			
1628	887000						
		5425 * 1=31					
1628	2060202020	5426	DEFB	20H,60H,20H,20H,20H,20H,70H,0			
1630	207000						
		5427 * 2=32					
1633	7088083040	5428	DEFB	70H,88H,08,30H,40H,80H,0F8H,0			
1638	80F800						
		5429 * 3=33					
1638	F808103008	5430	DEFB	0F8H,08,10H,30H,08,88H,70H,0			
1640	887000						
		5431 * 4=34					
1643	10305090F8	5432	DEFB	10H,30H,50H,90H,0F8H,10H,10H,0			
1648	101000						
		5433 * 5=35					
1648	F880F00808	5434	DEFB	0F8H,80H,0F0H,08H,08H,88H,70H,0			
1650	887000						
		5435 * 6=36					
1653	384080F088	5436	DEFB	38H,40H,80H,0F0H,88H,88H,70H,0			
1658	887000						
		5437 * 7=37					
1658	F808102040	5438	DEFB	0F8H,08H,10H,20H,40H,40H,40H,0			
1660	404000						
		5439 * 8=38					
1663	7088887088	5440	DEFB	70H,88H,88H,70H,88H,88H,70H,0			
1668	887000						

LOCATION OBJECT CODE LINE SOURCE LINE

1656	7088887808	5441 * 9=39	DEFB	70H,88H,88H,78H,08H,10H,0E0H,0
1670	10E000	5442		
1673	0000200020	5443 * :=3A	DEFB	0,0,20H,0,20H,0,0
1678	000000	5444		
1678	0000200020	5445 * :=3B	DEFB	0,0,20H,0,20H,20H,40H,0
1680	204000	5446		
1683	1020408040	5447 * <=3C	DEFB	10H,20H,40H,80H,40H,20H,10H,0
1688	201000	5448		
1688	0000F800F8	5449 * ==3D	DEFB	0,0,0F8H,0,0F8H,0,0,0
1690	000000	5450		
1693	4020100810	5451 * >=3E	DEFB	40H,20H,10H,08H,10H,20H,40H,0
1698	204000	5452		
1698	7088102020	5453 * ?=3F	DEFB	70H,88H,10H,20H,20H,0,20H,0
16A0	002000	5454		
16A3	7088A88880	5455 * @=40	DEFB	70H,88H,0A8H,088H,080H,080H,78H,0
16A8	807800	5456		
16AB	<16AB>	5457		
16AB	20508888F8	5458 ASCII_TBL	EQU \$	
1680	888800	5459 * A=41	DEFB	20H,50H,88H,88H,0F8H,88H,88H,0
1683	F08888F088	5460		
1688	88F000	5461 * B=42	DEFB	0F0H,88H,88H,88H,0F0H,88H,88H,0F0H,0
1688	7088808080	5462		
16C0	887000	5463 * C=43	DEFB	70H,88H,80H,80H,80H,88H,70H,0
16C3	F088888888	5464		
16C8	88F000	5465 * D=44	DEFB	0F0H,88H,88H,88H,88H,88H,0F0H,0
16CB	F8080F080	5466		
16D0	80F800	5467 * E=45	DEFB	0F8H,80H,80H,0F0H,080H,80H,0F8H,0
16D3	F8080F080	5469 * F=46	DEFB	0F8H,80H,80H,0F0H,80H,80H,80H,0
16D8	808000	5470		
16DB	7880808098	5471 * G=47	DEFB	78H,80H,80H,80H,80H,98H,78H,0
16E0	887800	5472		
16E3	888888F888	5473 * H=48	DEFB	88H,88H,88H,0F8H,88H,88H,88H,0
16E8	888800	5474		
16EB	7020202020	5475 * I=49	DEFB	70H,20H,20H,20H,20H,20H,70H,0
16F0	207000	5476		
16F3	0808080808	5477 * J=4A	DEFB	8,8,8,8,8,88H,70H,0
16F8	887000	5478		
		5479 * K=4B		

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
16FB	8890A0C0A0	5480	DEFB	88H,90H,0A0H,0C0H,0A0H,90H,88H,0
1700	908800	5481 * L=C	DEFB	80H,80H,80H,80H,80H,80H,0F8H,0
1703	8080808080	5482	DEFB	88H,0D8H,0A8H,0A8H,88H,88H,88H,0
1708	80F800	5483 * M=D	DEFB	88H,80H,80H,80H,80H,80H,80H,88H,0
1708	8808A8A888	5484	DEFB	88H,80H,80H,80H,80H,80H,80H,88H,0
1710	888800	5485 * N=E	DEFB	70H,88H,88H,88H,88H,88H,70H,0
1713	8888C8A898	5486	DEFB	0F0H,88H,88H,88H,0F0H,80H,80H,80H,00
1718	888800	5487 * O=F	DEFB	70H,88H,88H,88H,88H,88H,70H,0
1718	7088888888	5488	DEFB	0F0H,88H,88H,88H,0F0H,0A0H,90H,88H,0
1720	887000	5489 * P=50	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1723	F08888F080	5490	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1728	808000	5491 * Q=51	DEFB	70H,88H,88H,88H,88H,88H,88H,88H,0
1728	70888888A8	5492	DEFB	0F0H,88H,88H,88H,0F0H,0A0H,90H,88H,0
1730	906800	5493 * R=52	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1733	F08888F0A0	5494	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1738	908800	5495 * S=53	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1738	70888807008	5496	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1740	887000	5497 * T=54	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1743	F820202020	5498	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1748	202000	5499 * U=55	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1748	8888888888	5500	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1750	887000	5501 * V=56	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1753	8888888888	5502	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1758	502000	5503 * W=57	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1758	888888A8A8	5504	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1760	D88800	5505 * X=58	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1763	8888502050	5506	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1768	888800	5507 * Y=59	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1768	8888502020	5508	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1770	202000	5509 * Z=5A	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1773	F808102040	5510	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1778	80F800	5511 * [=58	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1778	F8C0C0C0C0	5512	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1780	C0F800	5513 * \=5C	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1783	0880402010	5514	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1788	080000	5515 *]=50	DEFB	70H,88H,80H,70H,08H,88H,70H,0
1788	F818181818	5516	DEFB	0F0H,88H,88H,88H,88H,88H,88H,88H,0
1790	18F800	5517 * ^=5E	DEFB	70H,88H,80H,70H,08H,88H,70H,0

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
1793	0000205088	5518	DEFB	0,0,20H,50H,88H,0,0,0
1798	00000000	5519 * =5F	DEFB	0,0,0,0,0,0,0,0,0F8H
1798	0000000000	5520	DEFB	0,0,0,0,0,0,0,0,0,0F8H
17A0	00000F8	5521 * =60	DEFB	40H,20H,10H,0,0,0,0,0,0
17A3	4020100000	5522	DEFB	0,0,70H,88H,0F8H,88H,88H,0
17A8	0000000	5523 * a=61	DEFB	0,0,0F0H,48H,70H,48H,0F0H,0
17AB	00007088F8	5524	DEFB	0,0,78H,80H,80H,80H,78H,0
17B0	888800	5525 * b=62	DEFB	0,0,0F0H,048H,048H,0F0H,0
17B3	0000F04870	5526	DEFB	0,0,0F0H,080H,0E0H,80H,0F0H,0
17B8	48F000	5527 * c=63	DEFB	0,0,0F0H,080H,0E0H,080H,80H,0
17B8	0000788080	5528	DEFB	0,0,78H,80H,80H,80H,78H,0
17C0	807800	5529 * d=64	DEFB	0,0,0F0H,048H,048H,0F0H,0
17C3	0000F04848	5530	DEFB	0,0,0F0H,080H,0E0H,80H,0F0H,0
17C8	48F000	5531 * e=65	DEFB	0,0,0F0H,080H,0E0H,080H,80H,0
17CB	0000F080E0	5532	DEFB	0,0,78H,80H,80H,80H,78H,0
17D0	80F000	5533 * f=66	DEFB	0,0,0F0H,080H,0E0H,080H,80H,0
17D3	0000F080E0	5534	DEFB	0,0,0F0H,080H,0E0H,080H,80H,0
17D8	808000	5535 * g=67	DEFB	0,0,78H,80H,80H,80H,78H,0
17DB	0000788088	5536	DEFB	0,0,88H,88H,0F8H,88H,88H,0
17E0	887000	5537 * h=68	DEFB	0,0,0F8H,20H,20H,20H,0F8H,0
17E3	00008888F8	5538	DEFB	0,0,70H,20H,20H,0A0H,0E0H,0
17E8	888800	5539 * i=69	DEFB	0,0,90H,0A0H,0C0H,0A0H,90H,0
17EB	0000F82020	5540	DEFB	0,0,80H,80H,80H,80H,0F8H,0
17F0	20F800	5541 * j=6A	DEFB	0,0,88H,008H,0A8H,88H,88H,0
17F3	0000702020	5542	DEFB	0,0,88H,0C8H,0A8H,98H,88H,0
17FB	A0E000	5543 * k=68	DEFB	0,0,0F8H,088H,88H,0F8H,0
17FB	000090A0C0	5544	DEFB	0,0,80H,80H,80H,80H,0F8H,0
1800	A09000	5545 * l=6C	DEFB	0,0,88H,008H,0A8H,88H,88H,0
1803	0000808080	5546	DEFB	0,0,88H,0C8H,0A8H,98H,88H,0
1808	80F800	5547 * m=60	DEFB	0,0,88H,0C8H,0A8H,98H,88H,0
1808	00008888A8	5548	DEFB	0,0,0F8H,088H,88H,0F8H,0
1810	888800	5549 * n=6E	DEFB	0,0,0F0H,88H,0F0H,80H,80H,0
1813	00008888A8	5550	DEFB	0,0,0F0H,88H,0F0H,80H,80H,0
1818	988800	5551 * o=6F	DEFB	0,0,0F0H,88H,0F0H,80H,80H,0
1818	0000F88888	5552	DEFB	0,0,0F0H,88H,0F0H,80H,80H,0
1820	88F800	5553 * p=70	DEFB	0,0,0F0H,88H,0F0H,80H,80H,0
1823	0000F088F0	5554	DEFB	0,0,0F0H,88H,0F0H,80H,80H,0
1828	808000	5555 * q=71	DEFB	0,0,0F0H,88H,0F0H,80H,80H,0

LOCATION	OBJECT CODE LINE	SOURCE LINE
1828	0000F888A8	5556
1830	90E000	
		DEFB
		0,0,0F8H,88H,0A8H,90H,0E0H,0
1833	0000F888F8	5557 * I=72
1838	A09000	5558
		DEFB
		0,0,0F8H,88H,0F8H,0A0H,90H,0
1838	0000788070	5559 * S=73
1840	08F000	5560
		DEFB
		0,0,78H,80H,70H,08H,0F0H,0
1843	0000F82020	5561 * t=74
1848	202000	5562
		DEFB
		0,0,0F8H,20H,20H,20H,20H,0
1848	0008888888	5563 * u=75
1850	887000	5564
		DEFB
		0,0,88H,88H,88H,88H,70H,00
1853	0008888890	5565 * v=76
1858	A04000	5566
		DEFB
		0,0,88H,88H,90H,0A0H,40H,0
1858	00088888A8	5567 * w=77
1860	D88800	5568
		DEFB
		0,0,88H,88H,0A8H,0D8H,88H,00
1863	0008886020	5569 * x=78
1868	608800	5570
		DEFB
		0,0,88H,60H,20H,60H,88H,0
1868	0008885020	5571 * y=79
1870	202000	5572
		DEFB
		0,0,88H,50H,20H,20H,20H,0
1873	0000F81020	5573 * z=7A
1878	40F800	5574
		DEFB
		0,0,0F8H,10H,20H,40H,0F8H,0
1878	384020C020	5575 * (=7B
1880	403800	5576
		DEFB
		38H,40H,20H,0C0H,20H,40H,38H,0
1883	4020100810	5577 * >=7C
1888	204000	5578
		DEFB
		40H,20H,10H,08H,10H,20H,40H,0
1888	E010201820	5579 *)=7D
1890	10E000	5580
		DEFB
		0E0H,10H,20H,18H,20H,10H,0E0H,0
1893	40A0100000	5581 * -=7E
1898	000000	5582
		DEFB
		40H,0A8H,10H,0,0,0,0,0
1898	A850A850A8	5583 * #=7F
18A0	50A800	5584
		DEFB
		0A8H,50H,0A8H,50H,0A8H,50H,0A8H,0
		EQU \$
		DEFB 1,2,14,15,8,9,18,19
		DEFB 3,4,14,15,5,20,0,0
		DEFB 5,0,16,17,10,11,21,22
		DEFB 6,7,16,17,5,20,0,0
		DEFB 1,2,14,15,3,4,14,15
		DEFB 3,4,14,15,12,13,23,24
		DEFB 1,2,14,15,8,9,18,19
		DEFB 3,4,14,15,5,20,0,0
		DEFB 5,0,16,17,10,11,21,22
		DEFB 6,7,16,17,5,20,0,0
		DEFB 1,2,14,15,3,4,14,15
		DEFB 3,4,14,15,12,13,23,24
		DEFB 1,2,14,15,8,9,18,19
		DEFB 3,4,14,15,5,20,0,0
		DEFB 5,0,16,17,10,11,21,22
		DEFB 6,7,16,17,5,20,0,0
		DEFB 1,2,14,15,3,4,14,15
		DEFB 3,4,14,15,12,13,23,24

LOCATION OBJECT CODE LINE SOURCE LINE

```

1800 001718
1803 FF
5593          HEX FF          ;END OF TABLE INDICATOR
5594 *****
5595 *****
5596 *          *****
5597 *          SUBROUTINES *****
5598 *          *****
5599 *****
5600 *****
5601 ***** OS SUBROUTINES *****
5602 *****
5603 * FILL_VRAM_ WRITES TO VRAM ADDRESS POINTED TO BY HL THE VALUE IN A
5604 * DE TIMES.
5605 *****
5606 *          VRAM STARTING ADDRESS IN HL
5607 *          NO OF BYTES IN DE
5608 *          VALUE TO BE WRITTEN IN A
5609 *****
5610 FILL_VRAM_ LD C,A
5611          LD A,L
5612          OUT (MODE_1_PORT),A
5613          LD A,H
5614          OR 40H
5615          OUT (MODE_1_PORT),A
5616          LD A,C
5617          OUT (MODE_0_PORT),A
5618          DEC DE
5619          LD A,D
5620          OR E
5621          JR NZ,FILL
5622          CALL READ_REGISTER
5623          RET
5624 *****
5625 *****
5626 * MODE_1_ SETS UP GRAPHICS MODE 1 WITH VRAM ADDRESSES AS IN THE TABLE
5627 * BELOW AND EXITS WITH THE VIDEO BLANKED AND A BLACK BACKGROUND.
5628 *****
5629 *          VDP MEMORY MAP
5630 *          3800H-3FFFH SPRITE GENERATOR TABLE
5631 *          2000H-37FFH PATTERN COLOR TABLE
5632 *          1800H-187FH SPRITE ATTRIBUTE TABLE
5633 *          1800H-1AFFH PATTERN NAME TABLE
5634 *          0000H-17FFH PATTERN GENERATOR TABLE
5635 *****
5636 MODE_1_
5637          LD B,0
5638          LD C,0
5639          CALL WRITE_REGISTER
5640 *****
5641          LD B,1
5642          LD C,10000000B
5643          CALL WRITE_REGISTER
5644 * SET UP TABLE ADDRESSES IN VRAM
5645 *****
5646 *          PATTERN NAME TABLE
5647          LD A,2
5648          LD HL,1800H

```

```

ACTION OBJECT CODE LINE SOURCE LINE
18FC CD1F88 5649 CALL INIT_TABLE
5650
5651 * PATTERN COLOR TABLE
5652 LD A,4
5653 LD HL,2000H
5654 CALL INIT_TABLE
5655
5656 * PATTERN GENERATOR TABLE
5657 LD A,3
5658 LD HL,0
5659 CALL INIT_TABLE
5660
5661 * SPRITE ATTRIBUTE TABLE
5662 LD A,0
5663 LD HL,1800H
5664 CALL INIT_TABLE
5665
5666 * SPRITE GENERATOR TABLE
5667 LD A,1
5668 LD HL,3800H
5669 CALL INIT_TABLE
5670
5671 * SET UP BLACK BACKGROUND
5672 LD B,7
5673 LD C,0
5674 CALL WRITE_REGISTER
5675 RET
5676
5677 * LOAD_ASCII_WRITES OUT ASCII CHARACTER GENERATORS TO THE PATTERN
5678 * GENERATOR TABLE. INIT_TABLE MUST BE USED TO SET UP
5679 * THE TABLE ADDRESS.
5680
5681 LOAD_ASCII_ LD HL,ASC_TABLE ;LOCATION OF GENERATORS
5682 LD DE,10H ;OFFSET TO PLACE ASC_TABLE
5683 LD IV,96 ;IN THE CORRECT LOCATION
5684 LD A,3
5685 CALL PUT_VRAM
5686
5687 * WRITE OUT A BLANK PATTERN FOR ASCII_0
5688 LD HL,SPACE
5689 LD DE,0
5690 LD IV,1
5691 LD A,3
5692 CALL PUT_VRAM
5693 RET
5694
5695 ***** LOCAL SUBROUTINES *****
5696
5697 PARSE LD BC,0 ;FROM HL INCREMENT BC UNTIL
5698 P_LOOP LD A,[HL] ;[HL] = "/*
5699 CP */*
5700 RET Z
5701 INC HL
5702 INC BC
5703 JR P_LOOP
5704
5705 CENTER_PRT PUSH BC ;BC=LENS
1951 C5

```

```

LOCATION OBJECT CODE LINE      SOURCE LINE
1952 FDE1                    5706      POP IV
1954 3E20                    5707      LD A,32
1956 99                     5708      SBC A,C
1957 1F                     5709      RRA
1958 0600                   5710      LD B,0
195A 4F                     5711      LD C,A
195B 09                     5712      ADD HL,BC
195C 44                     5713      LD B,H
195D 40                     5714      LD C,L
195E 62                     5715      LD H,D
195F 68                     5716      LD L,E
1960 50                     5717      LD D,B
1961 59                     5718      LD E,C
1962 3E02                   5719      LD A,2
1964 CD1FBE                 5720      CALL PUT_VRAM
1967 C9                     5721      RET
                            5722
1968 211700                 5723      LD HL,1700H
1968 1100FF                 5724      LD DE,255
196E 18                     5725      DEC DE
196F 7A                     5726      LD A,D
1970 83                     5727      OR E
1971 20FB                   5728      JR NZ,TIMER_2
1973 28                     5729      DEC HL
1974 7C                     5730      LD A,H
1975 85                     5731      OR L
1976 20F3                   5732      JR NZ,TIMER_1
1978 C9                     5733      RET
                            5734      PROG
;IY= #ITEMS TO BE TRANSFERED IN PUT_VRAM
;DE= LOCATION OF START OF STRING
; A=32-C
; DIV 2

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

5736 ***** EXTERNAL SYMBOLS *****
5737 *****
5738 *****
5739 *****
5740 *****
5741 * EXTERNAL ROUTINES FROM OS
5742
5743 ;EXT INIT TABLE
5744 ;EXT PUT_VRAM
5745 ;EXT WRITE_REGISTER
5746 ;EXT WRITE_VRAM
5747 ;EXT VRAM_ADDR_TABLE
5748 ;EXT SPRITEMAME_TBL
5749 ;EXT SPRITEGEN_TBL
5750 ;EXT PATRNNAME_TBL
5751 ;EXT PATRNGEN_TBL
5752 ;EXT COLORTABLE
5753 ;EXT LOAD_ASCII
5754 ;EXT FILL_VRAM
5755 ;EXT MODE_1
5756 ***** DEFINITIONS *****
5757 *****
5758 ***** EXPORTS *****
5759 *****
5760
5761 GLB GAME_OPT_
5762 ***** DISPLAY GAME OPTION SCREEN *****
5763 *****
5764 * GAME_OPT_ DISPLAYS THE GAME OPTION SCREEN WITH WHITE LETTERS ON A
5765 * BLUE BACKGROUND. VDP IS LEFT IN MODE 1 WITH THE VRAM
5766 * MEMORY MAP AS FOLLOWS.
5767 *
5768 * VDP MEMORY MAP
5769 *
5770 * 3800H-3FFFH SPRITE GENERATOR TABLE
5771 * 2000H-37FFFH PATTERN COLOR TABLE
5772 * 1800H-1B7FFH SPRITE ATTRIBUTE TABLE
5773 * 1800H-1AFFH PATTERN NAME TABLE
5774 * 0000H-17FFFH PATTERN GENERATOR TABLE
5775
5776 PROG
5777 GAME_OPT_ LD HL,0 ;ZERO VRAM
5778 LD DE,16384
5779 LD A,0
5780 CALL FILL_VRAM
5781
5782 * SET UP VDP WITH MODE 1
5783 CALL MODE_1
5784
5785 * SET UP BACKGROUND COLOR
5786 LD B,0FH
5787 LD C,4
5788 CALL WRITE_REGISTER
5789 ***** WRITE OUT PATTERN GEN TABLE *****
5790 *****
5791 CALL LOAD_ASCII
5792
1979 210000
197C 114000
197F 3E00
1981 CD1F82
1984 CD1F85
1987 060F
1989 0E04
198B CD1FD9
198E CD1FF7

```


LOCATION	OBJECT CODE LINE	SOURCE LINE
	5793	***** WRITE OUT PATTERN_NAME_TABLE *****
	5794	
	5795	* WRITE OUT PATTERN_NAME_TABLE
	5796	LD HL,LINE_1
1991	211A7C	LD DE,37
1994	110025	LD IV,22
1997	FD210016	LD A,2
1998	3E02	CALL PUT_VRAM
1999	CD1FBE	
	5801	
	5802	
19A0	211A92	LD HL,LINE_2
19A3	110065	LD DE,101
19A6	FD210017	LD IV,23
19AA	3E02	LD A,2
19AC	CD1FBE	CALL PUT_VRAM
	5807	
	5808	
19AF	1100C5	LD DE,197
19B2	CD1ACA	CALL WRITE_L3
	5810	
	5811	
19B5	110105	LD DE,261
19B8	CD1ACA	CALL WRITE_L3
	5812	
	5813	
	5814	
19BB	110145	LD DE,325
19BE	CD1ACA	CALL WRITE_L3
	5815	
	5816	
	5817	
19C1	110185	LD DE,389
19C4	CD1ACA	CALL WRITE_L3
	5818	
	5819	
	5820	
19C7	1101E5	LD DE,485
19CA	CD1ACA	CALL WRITE_L3
	5821	
	5822	
	5823	
19CD	110225	LD DE,549
19D0	CD1ACA	CALL WRITE_L3
	5824	
	5825	
	5826	
19D3	110265	LD DE,613
19D6	CD1ACA	CALL WRITE_L3
	5827	
	5828	
	5829	
19D9	1102A5	LD DE,677
19DC	CD1ACA	CALL WRITE_L3
	5830	
	5831	
	5832	
19DF	110105	LD DE,261
19E2	CD1AD7	CALL WRITE_L4
	5833	
	5834	
	5835	
19E5	110145	LD DE,325
19E8	CD1ADC	CALL WRITE_L5
	5836	
	5837	
	5838	
19EB	110185	LD DE,389
19EE	CD1AE1	CALL WRITE_L6
	5839	
	5840	
	5841	
19F1	211AC2	LD HL,LINE_7
19F4	1101E5	LD DE,485
19F7	CD1AE4	CALL WRITE_CHAR
	5842	
	5843	
	5844	
	5845	
19FA	211AC3	LD HL,LINE_8
19FD	110225	LD DE,549
1A00	CD1AE4	CALL WRITE_CHAR
	5846	
	5847	
	5848	
	5849	

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
1A03	211AC4	5850	LD HL,LINE_9	
1A06	110265	5851	LD DE,613	
1A09	CD1AE4	5852	CALL WRITE_CHAR	
		5853		
1A0C	211AC5	5854	LD HL,LINE_10	
1A0F	1102A5	5855	LD DE,677	
1A12	CD1AE4	5856	CALL WRITE_CHAR	
		5857		
1A15	11010F	5858	LD DE,271	
1A18	CD1AD7	5859	CALL WRITE_L4	
		5860		
1A1B	11014F	5861	LD DE,335	
1A1E	CD1ADC	5862	CALL WRITE_L5	
		5863		
1A21	11018F	5864	LD DE,399	
1A24	CD1AE1	5865	CALL WRITE_L6	
		5866		
1A27	1101F1	5867	LD DE,497	
1A2A	CD1AEE	5868	CALL WRITE_L11	
		5869		
1A2D	110231	5870	LD DE,561	
1A30	CD1AEE	5871	CALL WRITE_L11	
		5872		
1A33	110271	5873	LD DE,625	
1A36	CD1AEE	5874	CALL WRITE_L11	
		5875		
1A39	1102B1	5876	LD DE,689	
1A3C	CD1AEE	5877	CALL WRITE_L11	
		5878		
1A3F	11022F	5879	LD DE,559	
1A42	CD1AD7	5880	CALL WRITE_L4	
		5881		
1A45	11026F	5882	LD DE,623	
1A48	CD1ADC	5883	CALL WRITE_L5	
		5884		
1A4B	1102AF	5885	LD DE,687	
1A4E	CD1AE1	5886	CALL WRITE_L6	
		5887		
1A51	1101FB	5888	LD DE,507	
1A54	CD1AFB	5889	CALL WRITE_L12	
		5890		
1A57	11023B	5891	LD DE,571	
1A5A	CD1AFB	5892	CALL WRITE_L12	
		5893		
1A5D	11027B	5894	LD DE,635	
1A60	CD1AFB	5895	CALL WRITE_L12	
		5896		
1A63	11028B	5897	LD DE,699	
1A66	CD1AFB	5898	CALL WRITE_L12	
		5899		
5900			***** WRITE OUT COLOR_NAME_TABLE *****	
5901				
1A69	2A73FA	5902	LD HL,[COLORTABLE]	
1A6C	110020	5903	LD DE,32	
1A6F	3EF4	5904	LD A,OF4H	
1A71	CD1FB2	5905	CALL_FILL_VRAM	
		c006		

```

LOCATION OBJECT CODE LINE SOURCE LINE
5907 ***** ENABLE DISPLAY *****
5908
5909 * ENABLE DISPLAY
5910 LD B,1
5911 LD C,11000000B
5912 CALL WRITE_REGISTER
5913 RET
5914
5915 *****
5916 * DATA TABLES
5917 *
5918 *
5919 *****
5920
5921 ***** PATTERN_NAME_TABLE *****
5922 LINE_1 DEFB "TO SELECT GAME OPTION,"
5923 LINE_2 DEFB "PRESS BUTTON ON KEYPAD."
5924 LINE_3 DEFB "1 = SKILL 1/ONE PLAYER"
5925 LINE_4 DEFB "2"
5926 LINE_5 DEFB "3"
5927 LINE_6 DEFB "4"
5928 LINE_7 DEFB "5"
5929 LINE_8 DEFB "6"
5930 LINE_9 DEFB "7"
5931 LINE_10 DEFB "8"
5932 LINE_11 DEFB "TWO"
5933 LINE_12 DEFB "S"
5934
5935 ***** LOCAL SUBROUTINES *****
5936 *
5937 *
5938 *
5939 *****
5940
5941 WRITE_L3 LD HL,LINE_3
5942 LD LY,22
5943 LD A,2
5944 CALL PUT_VRAM
5945 RET
5946
5947 WRITE_L4 LD HL,LINE_4
5948 JR WRITE_CHAR
5949
5950 WRITE_L5 LD HL,LINE_5
5951 JR WRITE_CHAR
1A7C 544F205345
1A81 4C45435420
1A86 4741404520
1A88 4F5054494F
1A90 4E2C
1A92 5052455353
1A97 2042555454
1A9C 4F4E204F4E
1AA1 2048455950
1AA6 41442E
1AA9 3120302053
1AAE 48494C4C20
1AB3 312F4F4E45
1AB8 20504C4159
1ABD 6552
1ABF 32
1AC0 33
1AC1 34
1AC2 35
1AC3 36
1AC4 37
1AC5 38
1AC6 54574F
1AC9 53
1ACA 211AA9
1ACD F0210016
1AD1 3E02
1AD3 CD1F8E
1AD6 C9
1AD7 211ABF
1ADA 1808
1ADC 211AC0
1ADF 1803

```

LOCATION OBJECT CODE LINE SOURCE LINE

1AE1 211AC1	5952		
1AE4 FD210001	5953	WRITE_L6	LD HL,LINE_6
1AE8 3E02	5954	WRITE_CHAR	LD IV,1
1AEA CD1FBE	5955		LD A,2
1AED C9	5956		CALL PUT_VRAM
	5957		RET
	5958		
1AEE 211AC6	5959	WRITE_L11	LD HL,LINE_11
1AF1 FD210003	5960		LD IV,3
1AF5 3E02	5961		LD A,2
1AF7 CD1FBE	5962		CALL PUT_VRAM
1AFA C9	5963		RET
	5964		
1AFB 211AC9	5965	WRITE_L12	LD HL,LINE_12
1AFE FD210001	5966		LD IV,1
1B02 3E02	5967		LD A,2
1B04 CD1FBE	5968		CALL PUT_VRAM
1B07 C9	5969		RET
	5970		
	5971	PROG	

LOCATION OBJECT CODE LINE SOURCE LINE

```

5973 ;TRUE EQU 1
5974 ;EXT VRAM_WRITE_REG_WRITE,VRAM_READ
5975 ;EXT VDP_MODE_WORD
5976 ;EXT MUX_SPRITES
5977 ;EXT PARAM
5978 ;EXT LOCAL_SPR_TBL_SPRITE_ORDER
5979 GLB INIT_TABLE_GET_VRAM_PUT_VRAM_INIT_SPR_ORDER_MR_SPR_MH_TBL
5980 GLB INIT_TABLEQ_GET_VRAMQ_PUT_VRAMQ_INIT_SPR_ORDERQ_MR_SPR_MH_TBLQ
5981
5982 PROG
5983 * PROCEDURE INIT_TABLEQ (TABLE_CODE:BYTE;TABLE_ADDRESS:INTEGER)
5984
5985 * THIS IS THE PASCAL ENTRY POINT TO INIT_TABLE_
5986
5987 INIT_TABLE_P DEFW 2,1,2
5988 * THIS IS THE PARAMETER DESCRIPTOR FOR INIT_TABLEQ
5989
5990 INIT_TABLEQ LD BC,INIT_TABLE_P
5991 LD DE,PARAM_AREA
5992 CALL PARAM
5993 LD A,[PARAM_AREA]
5994 LD HL,[PARAM_AREA+1]
5995
5996
5997 INIT_TABLE_ ;INIT_TABLE_ INITIALIZES THE TABLE ADDRESSES
5998 ;FOR VRAM TABLES. IT ALSO WRITES THE APPROPRIATE
5999 ;BASE ADDRESS INTO THE RESPECTIVE VDP REGISTER.
6000 ;
6001 ;IN: VRAM ADDRESS IN HL
6002 ;TABLE CODE IN 'A' : 0=SPRITE NAME TABLE
6003 ; 1=SPRITE GENERATOR TABLE, 2=PATTERN NAME
6004 ; 3= PATTERN GENERATOR TABLE, 4=
6005 ; COLOR TABLE
6006
6007 LD C,A
6008 LD B,0 ;USE TABLE CODE AS INDEX
6009 LD IX,VRAM_ADDR_TABLE
6010 ADD IX,BC
6011 LD [IX+0],L
6012 LD [IX+1],H
6013
6014 LD A,[VDP_MODE_WORD]
6015 BIT 1,A
6016 JR 2,INIT_TABLE80
6017 ;** GRAPHICS MODE 2
6018 LD A,C
6019 CP 3
6020 JR 2,CASE_OF_GEN
6021 CP 4
6022 JR 2,CASE_OF_COLOR
6023 JR INIT_TABLE80
6024 CASE_OF_GEN
6025 LD B,4
6026 LD A,L
6027 OR H
6028
1808 00020001
180C 0002
180E 011808
1811 11738A
1814 CD0098
1817 3A738A
181A 2A738B
181D
181E 0600
1820 002173F2
1824 DD09
1826 DD09
1828 DD7500
182B DD7401
182E 3A73C3
1831 CB4F
1833 2827
1835 79
1836 FE03
1838 2806
183A FE04
183C 2810
183E 181C
1840
1840 0604
1842 7D
1843 84

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
1844 2004 JR NZ,CASE_OF_GEN10
1846 0E03 LD C,3 ;VALUE TO WRITE FOR VRAM ADDRESS OF 00H
1848 1828 JR INIT_TABLE90
184A LD CASE_OF_GEN10
184C 0E07 LD 6032 ;VALUE TO WRITE FOR VRAM ADDRESS OF 2000H
184E 1824 JR INIT_TABLE90
184E 0603 LD B,3 ;REGISTER TO WRITE
1850 7D LD A,L
1851 B4 OR H
1852 2004 JR NZ,CASE_OF_CLR10
1854 0E7F LD C,7FH ;VALUE TO WRITE FOR VRAM ADDRESS OF 00H
1856 181A JR INIT_TABLE90
1858 LD CASE_OF_CLR10
1858 0E7F LD C,OFFH ;VALUE TO WRITE FOR VRAM ADDRESS OF 2000H
185A 1816 JR INIT_TABLE90
185C LD CASE_OF_CLR10
185C 0E7F LD C,OFFH ;VALUE TO WRITE FOR VRAM ADDRESS OF 2000H
185C 6045
185C 6046 INIT_TABLE80 ;**
185C 6047 ;**
185C 6048 ;**
185C FD211876 LD IY,BASE_FACTORS
1860 FD09 LD IY,BC
1862 FD09 ADD IY,BC
1864 FD7E00 LD A,[[IY+0] ;SHIFT COUNT NOW IN 'A'
1867 FD4601 LD B,[[IY+1] ;REGISTER NUMBER TO WRITE IN 'B'
186A DIVIDE ;**
186A 0654 ;**
186A 0655 SRL H ;SHIFT HI BYTE
186A 0656 RRR L ;SHIFT LO BYTE
186E 3D DEC A ;DECREMENT SHIFT COUNT
186F 20F9 JR NZ,DIVIDE
186F 20F9 ;**
1871 4D LD C,L ;VALUE TO WRITE IN 'C'
1872 CD1CCA CALL REG_WRITE
1875 C9 RET
1876 LD CASE_OF_CLR10
1876 0655 ;BASE_FACTOR,REGISTER_NUMBER
1876 6066
1876 6067 DEFEB 7,5,11,6,10,2,11,4,6,3
1876 6068
1876 070508060A * PROCEDURE GET_VRAMO (TABLE CODE:BYTE;START_INDEX:BYTE;SLICE:BYTE;
1878 0208040603 VAR DATA:BUFFER;ITEM_COUNT:INTEGER);
1880 00050001
1884 00010001
1886 FFFE0002
188C LD BC,GET_VRAM_P
188C 011880 LD DE,PARAM_AREA
188F 11738A CALL PARAM
1892 CD0098 LD A,[PARAM_AREA]
1895 3A738A LD DE,[PARAM_AREA+1]
1898 ED587388 LD

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
18DE 2808 6140 JR Z,END_ADJ_COUNT
18E0 18E0 ADJUST_COUNT ;MULTIPLY ITEM_COUNT TO GET BYTE COUNT
18E1 6142 SLA C
18E2 CB10 6143 RL B
18E4 30 6144 DEC A
18E5 20F9 6145 JR NZ,ADJUST_COUNT
18E7 ED4373FE 6146 LD (SAVED_COUNT),BC ;SAVE ADJUSTED COUNT
18E8 6147 END_ADJ_COUNT
6148
18EB C1 6149 POP BC ;RESTORE TABLE_CODE/INDEX
18EC 6150 SET_COUNT20
18ED E5 6151 PUSH HL
18ED D009 6152 ADD IX,BC ;GET TABLE ADDRESS IN VRAM
18EF D009 6153 ADD IX,BC
18F1 D06E00 6154 LD L,[(IX+0)] ;LOW ORDER OF VRAM ADDRESS
18F4 D06601 6155 LD H,[(IX+1)] ;HIGH ORDER OF VRAM ADDRESS
18F7 19 6156 ADD HL,DE ;ADD BYTE INDEX TO GET VRAM START ADDRESS
18F8 EB 6157 EX DE,HL ;VRAM DESTINATION NOW IN DE
18F9 E1 6158 POP HL ;RESTORE DATA POINTER
18FA ED4873FE 6159 LD BC,(SAVED_COUNT) ;RESTORE ADJUSTED COUNT
18FE 6160 SET_COUNTX
18FE C9 6161 RET
6162
6163
18FF 6164 SHIFT_CT DEFB 2,3,0,3,3
18FF 0203000303 6165
6166
6167
6168 * PROCEDURE PUT_VRAMQ (TABLE_CODE:BYTE;START_INDEX:BYTE;SLICE:BYTE;
6169 * VAR DATA:BUFFER;ITEM_COUNT:INTEGER);
6170
6171 * THIS IS THE PASCAL ENTRY POINT TO INIT_TABLE_
6172 6173 PUT_VRAM_P DEFH 5,1,1,1,-2,2
6174 * THIS IS THE PARAMETER DESCRIPTOR FOR INIT_TABLEQ
6175
6176 PUT_VRAMQ
6177 LD BC,PUT_VRAM_P
6178 LD DE,PARAM_AREA
6179 CALL PARAM
6180 LD A,(PARAM_AREA)
6181 DE,(PARAM_AREA+1)
6182 LD IY,(PARAM_AREA+5)
6183 LD HL,(PARAM_AREA+3)
6184
6185 PUT_VRAM_ ;WRITES A CERTAIN NUMBER OF BYTES TO VRAM
6186 ;FROM A BUFFER.
6187 ;IN: TABLE CODE IN A
6188 ; 0=SPRITE NAME TABLE
6189 ; 1=SPRITE GENERATOR TABLE, 2=PATTERN NAME
6190 ; TABLE, 3= PATTERN GENERATOR TABLE, 4=
6191 ; COLOR TABLE
6192 ; START INDEX IN DE
6193 ; DATA BUFFER IN HL, AND COUNT IN IY.
6194
1C04 00050001
1C08 00010001
1C0C FFE0002
6195
1C10
1C10 011C04
1C13 11738A
1C16 C00098
1C19 3A738A
1C1C ED5B7388
1C20 FD2A738F
1C24 2A738D
6196
1C27

```



```

LOCATION OBJECT CODE LINE SOURCE LINE
1C27 F5 6195 PUSH AF
6196 * IF (TABLE CODE = SPRITE_NAME_TABLE) AND (MUX_SPRITES = TRUE) THEN
6197 CP_0
1C28 FE00 JR NZ,ELSEZZ
1C2A 2022 LD A,(MUX_SPRITES)
1C2C 3A73C7 CP 1
1C2F FE01 JR NZ,ELSEZZ
1C31 2018
6202
6203 * WRITE ENTRY TO LOCAL TABLE
1C33 F1 POP AF ; CLEAR STACK
1C34 E5 PUSH HL ; [SP] = DATA BUFFER
6206
1C35 2A8002 LD HL,(LOCAL_SPR_TBL) ; CALCULATE ADDRESS IN TABLE
1C38 78 LD A,E
1C39 CB27 SLA A
1C3B CB27 SLA A
1C3D 5F LD E,A
1C3E 19 ADD HL,DE
1C3F EB EX DE,HL
6214
1C40 FDE5 PUSH IV ; CALCULATE BYTE COUNT
1C42 C1 POP BC
1C43 79 LD A,C
1C44 CB27 SLA A
1C46 CB27 SLA A
1C48 4F LD C,A
6221
1C49 E1 POP HL ; STACK CLEAR
1C4A ED80 LDTR ; PERFORM WRITE FROM BUFFER
6224
1C4C 1807 JR END_IFZZ
6225 * ELSE
6227 ELSEZZ
1C4E POP AF
1C4F CD18AA CALL SET_COUNT
1C52 CD1D01 CALL VRAM_WRITE
6231 * END_IF
6232 END_IFZZ
1C55 RET
1C55 C9

1C56 00010001 6240 INIT_SPR_P DEFW 1,1
6241
<1C5A>
1C5A 011C56 6242 INIT_SPR_ORDER EQU $ BC,INIT_SPR_P
1C5D 11738A 6243 LD DE,PARAM_AREA
1C60 CD0098 6244 LD CALL PARAM
1C63 3A738A 6245 LD A,(PARAM_AREA)
6247
1C66 6248 INIT_SPR_ORDER ;INITIALIZES THE SPRITE DISPLAY ORDER
6249 ;LIST IN RAM TO DEFAULT ORDER (0...31)
6250 ;IN: NUMBER OF SPRITES TO ORDER IN 'A'
6251

```

```

OCATION OBJECT CODE LINE SOURCE LINE
1C66 47 6252 LD B,A ;SAVE SPRITE COUNT
1C67 AF 6253 XOR A
1C68 2A8004 6254 LD HL,[SPRITE_ORDER]
1C68 77 6255 INIT_SPR10
1C68 77 6256 LD [HL],A
1C6C 23 6257 INC HL
1C6D 3C 6258 INC A
1C6E 88 6259 CP B
1C6F 20FA 6260 JR NZ,INIT_SPR10
1C71 C9 6261 RET
6262
6263 * PROCEDURE WR_SPR_MM_TBLQ (SPRITE_COUNT:BYTE);
6264
6265 * THIS IS THE PASCAL ENTRY POINT TO THE WR_SPR_MM_TBL_ ROUTINE.
6266
1C72 00010001 6267 WR_SPR_P DEFW 1,1
6268
        <1C76>
1C76 011C72 6269 WR_SPR_MM_TBLQ EQU $
1C79 11738A 6270 LD BC,WR_SPR_P
1C7C C00098 6271 LD DE,PARAM_AREA
1C7F 3A738A 6272 CALL PARAM
        A,[PARAM_AREA]
1C82 6274 6275 WR_SPR_MM_TBL_ ;WRITES SPRITE_NAME_TABLE TO VRAM
        ;USING THE SPRITE ORDER LIST.
        ;
        ; NUMBER OF SPRITES TO WRITE IN 'A'
        OBEH
        OBFH
        EQU
        EQU
        LD IX,[SPRITE_ORDER] ;LIST OF DISPLAY ORDERS
        LD [SPRITE_CT],A ;SAVE COUNT
        PUSH AF ;SAVE COUNT
        LD IY,VRAM_ADDR_TABLE
        LD E,[IY+0]
        LD D,[IY+1] ;VRAM SPRITE_NAME_TABLE ADDRESS NOW IN DE.
        ;** SET UP VDP TO RECEIVE DATA
        LD A,E
        OUT (MODE_1_PORT),A
        LD A,D
        OR 40H
        OUT (MODE_1_PORT),A
        POP AF ;RESTORE COUNT
        OUTPUT_LOOP_TABLE_MA
1C9A 2A8002 6299 LD HL,(LOCAL_SPR_TBL) ;RESTORE RAM SPRITE_NAME_TABLE POINTER
1C9D D04E00 6301 LD C,[IX+0] ;DISPLAY ORDER
1CA0 D023 6302 INC IX ;ADVANCE DISPLAY ORDER POINTER
1CA2 0600 6303 LD B,0
1CA4 09 6304 ADD HL,BC
1CA5 09 6305 ADD HL,BC
1CA6 09 6306 ADD HL,BC
1CA7 09 6307 ADD HL,BC
        6308
    
```

LOCATION	OBJECT CODE LINE	SOURCE LINE
	6309	*,** OUTPUT TO VRAM THROUGH VDP
1C88 0604	6310	LD B,4 ;ELEMENT COUNT FOR ONE SPRITE
1CAA 0E0E	6311	LD C,MODE_0_PORT ;OUTPUT PORT IN 'C'
1CAC	6312	LD C,MODE_0_PORT ;OUTPUT PORT IN 'C'
1CAC EDA3	6313	OUTPUT_LOOP10
1CAE 00	6314	OUTI
1CAF 00	6315	NOP ;DELAY
1CB0 20FA	6316	NOP ;DELAY
	6317	JR NZ,OUTPUT_LOOP10
	6318 ;	LD A,(SPRITE_CT)
	6319 ;	DEC A
	6320 ;	LD (SPRITE_CT),A
1CB2 30	6321	DEC A
1CB3 20E5	6322	JR NZ,OUTPUT_LOOP_TABLE_MA
1CB5 C9	6323	RET
	6324	
	6325	GLB VRAM ADDR TABLE
	6326	GLB SPRITENAMETBL
	6327	GLB SPRITEGENTBL
	6328	GLB PATTRNAMETBL
	6329	GLB PATTRNGENTBL
	6330	GLB COLORTABLE
	6331	DATA
73F2	6332	VRAM ADDR TABLE
73F2	6333	SPRITENAMETBL DEFS 2
73F4	6334	SPRITEGENTBL DEFS 2
73F6	6335	PATTRNAMETBL DEFS 2
73F8	6336	PATTRNGENTBL DEFS 2
73FA	6337	COLORTABLE DEFS 2
	6338	* THIS TABLE HOLDS THE BASE ADDRESSES OF ALL THE VRAM TABLES.
	6339	
	6340	
73FC	6341	SAVE_TEMP DEFS 2
73FE	6342	SAVED_COUNT DEFS 2
	6343	
	6344 ;	COMM
	6345	
	6346 ;	PARAM_AREA DEFS 6
	6347	* THIS IS THE PARAMETER PASSING AREA FOR THE PASCAL ENTRY POINTS TO
	6348	* ROUTINES IN THIS MODULE. IT IS HELD IN COMMON WITH OTHER SUCH ENTRY
	6349	* POINTS FOR OTHER MODULES.
	6350	PROG

.OCATION OBJECT CODE LINE SOURCE LINE

```

6352 * The video drivers provide a standard protocol for low-level communication
6353 * with the 9918/28 VDP. There are four basic driver routines which between
6354 * them allow the programmer to write a value to a VDP register, read the
6355 * VDP status register, write a RAM or ROM buffer to a specified address
6356 * in VRAM, and read a RAM buffer from a specified address in VRAM.
6357 *
6358 * The four routines outlined above are:
6359 *
6360 * Procedure Reg_Write
6361 *
6362 * Reg_Write takes a VDP register number (0..7) in the B register
6363 * and a byte value to be written to it in the C register. It writes
6364 * the value to the given VDP register and returns.
6365 *
6366 * If the specified register is one of the VDP mode control registers,
6367 * ie. 0 or 1, then Reg_Write also writes the given value to the
6368 * corresponding half of the VDP_Mode_Word in RAM. All mode dependant
6369 * decisions made by the operating system are made by referencing the
6370 * contents of this word. Thus it is important for the cartridge
6371 * programmer to maintain it should he/she choose not to use
6372 * Reg_Write in accessing the VDP registers.
6373 *
6374 * In addition to the BC pair, Reg_Write also makes use of AF.
6375 *
6376 * Function Reg_Read
6377 *
6378 * Reg_Read reads the VDP status register and returns its contents
6379 * in the A register.
6380 *
6381 * It uses no other registers.
6382 *
6383 * NOTE ***** While this routine has no side effects with respect to
6384 * the CPU, it should be used with caution since reads
6385 * to the status register have the effect of resetting the
6386 * VDP interrupt flag and may cause field interrupts to be
6387 * missed.
6388 *
6389 * Procedure Vram_Write
6390 *
6391 * Vram_Write takes a pointer to the beginning of a data buffer in the
6392 * HL pair, the VRAM destination address in the DE pair, and a byte
6393 * count in the BC pair.
6394 *
6395 * It writes the specified number of bytes from the buffer to VRAM
6396 * starting at the destination address.
6397 *
6398 * The AF,BC,DE, and HL register pairs are all affected.
6399 *
6400 * NOTE ***** This procedure is not re-entrant.
6401 *
6402 * Procedure Vram_Read
6403 *
6404 * Vram_Read takes a pointer to the beginning of a data buffer in the
6405 * HL pair, the VRAM source address in the DE pair, and a byte count
6406 * count in the BC pair.
6407 *
6408 *

```

LOCATION OBJECT CODE LINE SOURCE LINE

6409 * It reads the specified number of bytes into the buffer from VRAM
 6410 * starting at the source address.

6411 *
 6412 * The AF,BC,DE, and HL register pairs are all affected.
 6413 *

6414 * NOTE ***** This procedure is not re-entrant.
 6415 *
 6416 *
 6417 *

6418 * For each of the routines listed above, there is an additional entry
 6419 * point which allows the routine to be called using the standard pascal
 6420 * 64000 parameter passing protocol and passing the parameters through
 6421 * a common data area into the registers. It should be noted that use of
 6422 * these routines in this fashion may cause problems in an interrupt
 6423 * driven environment. They should therefore be used with care. If the
 6424 * name of a given routine is Name, the name of the additional entry point
 6425 * is NameQ for the actual routine and NameP when called through the
 6426 * jump table in OS ROM. *Q* entry points destroy all registers.
 6427 * *****

6428 *****
 6429 ***** DICTIONARY *****
 6430 *****

<00BE>
 <00BF>

6431 DATA PORT EQU 0BEH
 6432 CTRL_PORT EQU 0BFH
 6433 GLB DATA_PORT_CTRL_PORT
 6434 * DATA PORT AND CTRL_PORT ARE THE PORTS THROUGH WHICH THE CPU
 6435 * COMMUNICATES WITH THE 9928.
 6436
 6437

6438 ; COMM

6439 ;PARAM AREA DEFS 6

6440 * THIS THE PARAMETER PASSING AREA THAT IS HELD IN COMMON WITH ALL OTHER
 6441 * PASCAL ENTRY POINTS TO OS ROUTINES.
 6442 *****

6443 *****

6444 ***** PROCEDURES AND FUNCTIONS *****

6445 PROG

6446
 6447

6448 ;EXT PARAM

6449 * PARAM_ IS THE PASCAL PARAMETER PASSING ROUTINE.

6450
 6451 * PROCEDURE REG_WRITE (REGISTER:BYTE;VALUE:BYTE)

6452

6453 * REGISTER IS PASSED IN THE B REGISTER.

6454 * VALUE IS PASSED IN THE C REGISTER.

6455 * DESTROYS: A

6456 ***** NOTE

6457 *

6458 * ***** THIS ROUTINE MAY SIDE EFFECT THE *****

6459 * ***** VDP MODE WORD *****

6460
 6461 ;EXT VDP_MODE_WORD

6462

6463 GLB REG_WRITE_P

6464 REG_WRITE_P DEFW 2,1,1

```

LOCATION OBJECT CODE LINE SOURCE LINE
6465 * THIS IS THE PARAMETER DESCRIPTOR FOR REG_WRITEQ
6466
6467 * BEGIN REG_WRITE REG_WRITEQ
6468 EQU $
6469 REG_WRITEQ BC,REG_WRITE_P
6470 LD DE,PARAM_AREA
6471 LD PARAM
6472 CALL HL,(PARAM_AREA)
6473 LD C,H
6474 LD B,L
6475 LD REG_WRITE
6476 EQU $
6477
6478 REG_WRITE EQU $
6479
6480 * VALUE =; CTRL_PORT
6481 LD CTRL_PORT
6482 OUT A,C
6483 [CTRL_PORT],A
6484 * REGISTER + 80H =; CTRL_PORT
6485 LD A,B
6486 ADD A,80H
6487 OUT [CTRL_PORT],A
6488
6489 * IF REGISTER = 0 THEN VDP_MODE_WORD[0] := VALUE
6490 LD A,B
6491 CP 0
6492 JR NZ,NOT_REG_0
6493 LD A,C
6494 LD [VDP_MODE_WORD],A
6495 NOT_REG_0 EQU $
6496
6497 * IF REGISTER = 1 THEN VDP_MODE_WORD[1] := VALUE
6498 LD A,B
6499 CP 1
6500 JR NZ,NOT_REG_1
6501 LD A,C
6502 LD [VDP_MODE_WORD+1],A
6503 NOT_REG_1 EQU $
6504
6505 * END REG_WRITE RET
6506
6507
6508 * PROCEDURE VRAM_WRITE (VAR DATA:BUFFER;DEST:INTEGER;COUNT:INTEGER)
6509
6510 * VAR DATA (POINTER TO DATA BUFFER) IS PASSED IN HL
6511 * DEST IS PASSED IN DE
6512 * COUNT IS PASSED IN BC
6513 * DESTROYS: ALL
6514
6515 VRAM_WRITE_P DEFW 3,-2,2,2
6516 * THIS IS THE PARAMETER DESCRIPTOR FOR VRAM_WRITEQ
6517 EQU $
6518 * BEGIN VRAM_WRITE GLB VRAM_WRITEQ
6519 EQU $
6520 VRAM_WRITEQ EQU $
6521
6522
6523
6524
6525
6526
6527
6528
6529
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6550
6551
6552
6553
6554
6555
6556
6557
6558
6559
6560
6561
6562
6563
6564
6565
6566
6567
6568
6569
6570
6571
6572
6573
6574
6575
6576
6577
6578
6579
6580
6581
6582
6583
6584
6585
6586
6587
6588
6589
6590
6591
6592
6593
6594
6595
6596
6597
6598
6599
6600
6601
6602
6603
6604
6605
6606
6607
6608
6609
6610
6611
6612
6613
6614
6615
6616
6617
6618
6619
6620
6621
6622
6623
6624
6625
6626
6627
6628
6629
6630
6631
6632
6633
6634
6635
6636
6637
6638
6639
6640
6641
6642
6643
6644
6645
6646
6647
6648
6649
6650
6651
6652
6653
6654
6655
6656
6657
6658
6659
6660
6661
6662
6663
6664
6665
6666
6667
6668
6669
6670
6671
6672
6673
6674
6675
6676
6677
6678
6679
6680
6681
6682
6683
6684
6685
6686
6687
6688
6689
6690
6691
6692
6693
6694
6695
6696
6697
6698
6699
6700
6701
6702
6703
6704
6705
6706
6707
6708
6709
6710
6711
6712
6713
6714
6715
6716
6717
6718
6719
6720
6721
6722
6723
6724
6725
6726
6727
6728
6729
6730
6731
6732
6733
6734
6735
6736
6737
6738
6739
6740
6741
6742
6743
6744
6745
6746
6747
6748
6749
6750
6751
6752
6753
6754
6755
6756
6757
6758
6759
6760
6761
6762
6763
6764
6765
6766
6767
6768
6769
6770
6771
6772
6773
6774
6775
6776
6777
6778
6779
6780
6781
6782
6783
6784
6785
6786
6787
6788
6789
6790
6791
6792
6793
6794
6795
6796
6797
6798
6799
6800
6801
6802
6803
6804
6805
6806
6807
6808
6809
6810
6811
6812
6813
6814
6815
6816
6817
6818
6819
6820
6821
6822
6823
6824
6825
6826
6827
6828
6829
6830
6831
6832
6833
6834
6835
6836
6837
6838
6839
6840
6841
6842
6843
6844
6845
6846
6847
6848
6849
6850
6851
6852
6853
6854
6855
6856
6857
6858
6859
6860
6861
6862
6863
6864
6865
6866
6867
6868
6869
6870
6871
6872
6873
6874
6875
6876
6877
6878
6879
6880
6881
6882
6883
6884
6885
6886
6887
6888
6889
6890
6891
6892
6893
6894
6895
6896
6897
6898
6899
6900
6901
6902
6903
6904
6905
6906
6907
6908
6909
6910
6911
6912
6913
6914
6915
6916
6917
6918
6919
6920
6921
6922
6923
6924
6925
6926
6927
6928
6929
6930
6931
6932
6933
6934
6935
6936
6937
6938
6939
6940
6941
6942
6943
6944
6945
6946
6947
6948
6949
6950
6951
6952
6953
6954
6955
6956
6957
6958
6959
6960
6961
6962
6963
6964
6965
6966
6967
6968
6969
6970
6971
6972
6973
6974
6975
6976
6977
6978
6979
6980
6981
6982
6983
6984
6985
6986
6987
6988
6989
6990
6991
6992
6993
6994
6995
6996
6997
6998
6999
7000
7001
7002
7003
7004
7005
7006
7007
7008
7009
7010
7011
7012
7013
7014
7015
7016
7017
7018
7019
7020
7021
7022
7023
7024
7025
7026
7027
7028
7029
7030
7031
7032
7033
7034
7035
7036
7037
7038
7039
7040
7041
7042
7043
7044
7045
7046
7047
7048
7049
7050
7051
7052
7053
7054
7055
7056
7057
7058
7059
7060
7061
7062
7063
7064
7065
7066
7067
7068
7069
7070
7071
7072
7073
7074
7075
7076
7077
7078
7079
7080
7081
7082
7083
7084
7085
7086
7087
7088
7089
7090
7091
7092
7093
7094
7095
7096
7097
7098
7099
7100
7101
7102
7103
7104
7105
7106
7107
7108
7109
7110
7111
7112
7113
7114
7115
7116
7117
7118
7119
7120
7121
7122
7123
7124
7125
7126
7127
7128
7129
7130
7131
7132
7133
7134
7135
7136
7137
7138
7139
7140
7141
7142
7143
7144
7145
7146
7147
7148
7149
7150
7151
7152
7153
7154
7155
7156
7157
7158
7159
7160
7161
7162
7163
7164
7165
7166
7167
7168
7169
7170
7171
7172
7173
7174
7175
7176
7177
7178
7179
7180
7181
7182
7183
7184
7185
7186
7187
7188
7189
7190
7191
7192
7193
7194
7195
7196
7197
7198
7199
7200
7201
7202
7203
7204
7205
7206
7207
7208
7209
7210
7211
7212
7213
7214
7215
7216
7217
7218
7219
7220
7221
7222
7223
7224
7225
7226
7227
7228
7229
7230
7231
7232
7233
7234
7235
7236
7237
7238
7239
7240
7241
7242
7243
7244
7245
7246
7247
7248
7249
7250
7251
7252
7253
7254
7255
7256
7257
7258
7259
7260
7261
7262
7263
7264
7265
7266
7267
7268
7269
7270
7271
7272
7273
7274
7275
7276
7277
7278
7279
7280
7281
7282
7283
7284
7285
7286
7287
7288
7289
7290
7291
7292
7293
7294
7295
7296
7297
7298
7299
7300
7301
7302
7303
7304
7305
7306
7307
7308
7309
7310
7311
7312
7313
7314
7315
7316
7317
7318
7319
7320
7321
7322
7323
7324
7325
7326
7327
7328
7329
7330
7331
7332
7333
7334
7335
7336
7337
7338
7339
7340
7341
7342
7343
7344
7345
7346
7347
7348
7349
7350
7351
7352
7353
7354
7355
7356
7357
7358
7359
7360
7361
7362
7363
7364
7365
7366
7367
7368
7369
7370
7371
7372
7373
7374
7375
7376
7377
7378
7379
7380
7381
7382
7383
7384
7385
7386
7387
7388
7389
7390
7391
7392
7393
7394
7395
7396
7397
7398
7399
7400
7401
7402
7403
7404
7405
7406
7407
7408
7409
7410
7411
7412
7413
7414
7415
7416
7417
7418
7419
7420
7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432
7433
7434
7435
7436
7437
7438
7439
7440
7441
7442
7443
7444
7445
7446
7447
7448
7449
7450
7451
7452
7453
7454
7455
7456
7457
7458
7459
7460
7461
7462
7463
7464
7465
7466
7467
7468
7469
7470
7471
7472
7473
7474
7475
7476
7477
7478
7479
7480
7481
7482
7483
7484
7485
7486
7487
7488
7489
7490
7491
7492
7493
7494
7495
7496
7497
7498
7499
7500
7501
7502
7503
7504
7505
7506
7507
7508
7509
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600
7601
7602
7603
7604
7605
7606
7607
7608
7609
7610
7611
7612
7613
7614
7615
7616
7617
7618
7619
7620
7621
7622
7623
7624
7625
7626
7627
7628
7629
7630
7631
7632
7633
7634
7635
7636
7637
7638
7639
7640
7641
7642
7643
7644
7645
7646
7647
7648
7649
7650
7651
7652
7653
7654
7655
7656
7657
7658
7659
7660
7661
7662
7663
7664
7665
7666
7667
7668
7669
7670
7671
7672
7673
7674
7675
7676
7677
7678
7679
7680
7681
7682
7683
7684
7685
7686
7687
7688
7689
7690
7691
7692
7693
7694
7695
7696
7697
7698
7699
7700
7701
7702
7703
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713
7714
7715
7716
7717
7718
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736
7737
7738
7739
7740
7741
7742
7743
7744
7745
7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757
7758
7759
7760
7761
7762
7763
7764
7765
7766
7767
7768
7769
7770
7771
7772
7773
7774
7775
7776
7777
7778
7779
7780
7781
7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793
7794
7795
7796
7797
7798
7799
7800
7801
7802
7803
7804
7805
7806
7807
7808
7809
7810
7811
7812
7813
7814
7815
7816
7817
7818
7819
7820
7821
7822
7823
7824
7825
7826
7827
7828
7829
7830
7831
7832
7833
7834
7835
7836
7837
7838
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7850
7851
7852
7853
7854
7855
7856
7857
7858
7859
7860
7861
7862
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7890
7891
7892
7893
7894
7895
7896
7897
7898
7899
7900
7901
7902
7903
7904
7905
7906
7907
7908
7909
7910
7911
7912
7913
7914
7915
7916
7917
7918
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7930
7931
7932
7933
7934
7935
7936
7937
7938
7939
7940
7941
7942
7943
7944
7945
7946
7947
7948
7949
7950
7951
7952
7953
7954
7955
7956
7957
7958
7959
7960
7961
7962
7963
7964
7965
7966
7967
7968
7969
7970
7971
7972
7973
7974
7975
7976
7977
7978
7979
7980
7981
7982
7983
7984
7985
7986
7987
7988
7989
7990
7991
7992
7993
7994
7995
7996
7997
7998
7999
8000

```

LOCATION	OBJECT CODE LINE	SOURCE LINE
1CED 011CE5	6521	LD
1CF0 11738A	6522	LD
1CF3 CD0098	6523	CALL
1CF6 2A738A	6524	LD
1CF9 ED58738C	6525	LD
1CFD ED48738E	6526	LD
	6527	
	6528	GLB
<1001>	6529	VRAM_WRITE EQU \$
	6530	
1001 E5	6531	* DEST := DEST + 4000H
1002 D5	6532	PUSH
1003 E1	6533	PUSH
1004 114000	6534	POP
1007 19	6535	LD
	6536	ADD
	6537	
1008 7D	6538	* LOW BYTE OF DEST =; CTRL_PORT
1009 D38F	6539	LD
	6540	OUT
	6541	
1008 7C	6542	* HIGH BYTE OF DEST =; CTRL_PORT
100C D38F	6543	LD
	6544	OUT
	6545	
100E C5	6546	* DATA =; DATA_PORT
100F D1	6547	PUSH
1010 E1	6548	POP
1011 0EBE	6549	POP
1013 43	6550	LD
	6551	LD
	6552	OUTPUT_LOOP
1014 EDA3	6553	EQU
1016 00	6554	OUTI
1017 00	6555	NOP
1018 C21D14	6556	NOP
101B 15	6557	JP
101C FA1D21	6558	DEC
101F 20F3	6559	JP
	6560	JR
	6561	* END VRAM_WRITE
<1021>	6562	END_OUTPUT EQU \$
1021 C9	6563	RET
	6564	
	6565	* PROCEDURE VRAM_READ (VAR DATA:BUFFER;SRCE:INTEGER;COUNT:INTEGER)
	6566	
	6567	* VAR DATA (POINTER TO DATA BUFFER) IS PASSED IN HL
	6568	* SRCE IS PASSED IN DE
	6569	* COUNT IS PASSED IN BC
	6570	* DESTROYS: ALL
	6571	
1022 0003FFE	6572	VRAM_READ_P DEFW 3,-2,2,2
1026 00020002	6573	* THIS IS THE PARAMETER DESCRIPTOR FOR VRAM_READQ
	6574	
	6575	* BEGIN VRAM_READ
	6576	GLB VRAM_READQ

```

.OBJECT OBJECT CODE LINE SOURCE LINE
<102A> 6577 VRAM_READQ EQU $
102A 011D22 6578 LD BC,VRAM_READ_P
102D 1173BA 6579 LD LD DE,PARAM_AREA
1030 C00098 6580 CALL PARAM
1033 2A73BA 6581 LD HL,(PARAM_AREA)
1036 ED5B73BC 6582 LD DE,(PARAM_AREA+2)
103A ED4873BE 6583 LD BC,(PARAM_AREA+4)
6584 VRAM_READ
6585 GLB
6586 VRAM_READ EQU $
<103E> 6587 * LOW BYTE OF SRCE =; CTRL_PORT A,E
103E 7B 6588 LD [CTRL_PORT],A
103F D3BF 6589 OUT
6590
6591 * HIGH BYTE OF SRCE =; CTRL_PORT A,D
1041 7A 6592 LD [CTRL_PORT],A
1042 D3BF 6593 OUT
6594
6595 * DATA =; DATA_PUSH
6596 6597 LD POP
1044 C5 6598 LD LD DE
1045 D1 6599 LD LD B,E
1046 0EBE 6600 LD $
1048 43 <1049> 6601 EQU NZ,INPUT_LOOP
1049 ED42 6602 INI D
1048 00 6603 MOP M,END_INPUT
104C 00 6604 JP NZ,INPUT_LOOP
104D C21049 6605 DEC
1050 15 6606 JP
1051 FA1056 6607 JR
1054 20F3 6608
6609
6610 * END VRAM_READ EQU $
6611 END_INPUT EQU
1056 C9 6612 RET
6613
6614 * FUNCTION REG_READ:BYTE
6615
6616 * FUNCTION OUTPUT RETURNED IN A
6617 * DESTROYS A ONLY
6618
6619 * BEGIN REG_READ REG_READ
6620 GLB
6621 REG_READ EQU $
<1057> 6622 * REG_READ =; CTRL_PORT A,[CTRL_PORT]
6623 IN
6624
6625 * END REG_READ RET
1057 D8BF
6626
6627
6628
1059 C9 6629 PROG

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

6631 * THIS IS A PACKAGE OF ROUTINES THAT ALLOW APPLICATIONS PROGRAMMERS TO
6632 * OPERATE ON SHAPE GENERATORS. EACH OF THEM TAKES, AS INPUTS, AN AREA
6633 * IN ONE OF THE GENERATOR TABLES IN WHICH THE GENERATORS TO BE OPERATED
6634 * UPON RESIDE, A COUNT OF THE GENERATORS TO BE USED, AND AN AREA OF THE
6635 * SAME TABLE INTO WHICH THE RESULTS ARE TO BE PUT. THE ONLY RAM AREA THEY
6636 * IS IN THE WORK_BUFFER A POINTER TO WHICH IS DECLARED AS AN EXTERNAL
6637 * AND DEFINED IN THE CARTRIDGE.
6638 *****
6639 ***** NOTE: *****
6640 ***** THESE ROUTINES WRITE TO AND READ *****
6641 ***** WITHOUT POSSIBILITY OF DEFERAL *****
6642 ***** AND SHOULD NOT BE USED IN ANY *****
6643 ***** SITUATION WHERE THEY MAY BE *****
6644 ***** INTERRUPTED. *****
6645 ***** *****
6646 *****
6647 *****
6648 ; EXT WORK_BUFFER
6649 ; POINTER TO THE WORK_BUFFER DEFINED BY THE CARTRIDGE PROGRAMMER
6650 *****
6651 ; EXT VOP_MODE_WORD
6652 ; THE WORD IN OS RAM THAT DESCRIBES THE CURRENT GRAPHICS MODE.
6653 *****
6654 ; EXT GET_VRAM
6655 ; EXT PUT_VRAM
6656 ; EXTAL OS ROUTINES IN TABLE_MANAGER_MODULE
6657 *****
6658 ; EXT MIRROR_LR
6659 ; EXT MIRROR_UD
6660 ; EXT ROTATE
6661 ; EXT MAGNIFY
6662 ; EXT QUADRUPL
6663 * EXTERNAL ROUTINES THAT PERFORM BLOCK OPERATIONS
6664 *****
6665 ;TRUE EQU 1
6666 ;FALSE EQU 0
6667 * VALUES FOR BOOLEAN FLAGS
6668 *****
6669 PATTERN_GEN EQU 3
6670 COLOR_TABLE EQU 4
6671 * VALUES FOR TABLE CODE
6672 *****
6673 * PROCEDURE REFLECT_VERTICAL (TABLE_CODE(A),SOURCE(DE),DESTINATION(HL),COUNT(BC))
6674 *****
6675 * REFLECT REFLECTS EACH OF A BLOCK OF GENERATORS FROM VRAM AROUND
6676 * THE VERTICAL AXIS. IF THE GENERATORS ARE FROM THE PATTERN PLANE
6677 * AND THE GRAPHICS MODE IS 2, THEN THE ROUTINE ALSO COPIES THE
6678 * CORRESPONDING COLOR GENERATORS., OTHERWISE IS ASSUMES THAT THE COLOR
6679 * DATA HAS ALREADY BEEN SET UP.
6680 *****
6681 * BEGIN REFLECT_VERTICAL
6682 ; GLB
6683 RFLCT_VERT
6684 ; ACTUAL ROUTINE NAME EXISTS IN OS
6685 ; JUMP TABLE ONLY
6686 * SET OPERATION CODE
6687 LD IX,RFLCT_VERT_

```

105A

105A D0211096

```

LOCATION OBJECT CODE LINE SOURCE LINE
105E 1810
6688 * CONTINUE BELOW
6689 * JR CONTINUE_GRAPHICS
6690
6691
6692
6693 * PROCEDURE REFLECT_HORIZONTAL (TABLE_CODE(A),SOURCE(DE),DESTINATION(HL),COUNT(BC))
6694
6695 * REFLECT HORIZONTAL REFLECTS EACH OF A BLOCK OF GENERATORS FROM VRAM
6696 * AROUND THE HORIZONTAL AXIS. IF THE GENERATORS ARE FROM THE PATTERN
6697 * PLANE AND THE GRAPHICS MODE IS 2 THEN IT REFLECTS THE CORRESPONDING
6698 * COLOR GENERATORS AS WELL.
6699
6700 * BEGIN REFLECT_HORIZONTAL
6701 * RFLCT_HOR
6702 * ; ACTUAL ROUTINE NAME EXISTS IN OS
6703 * ; JUMP TABLE ONLY
6704
6705 * SET OPERATION CODE
6706 * LD IX,RFLCT_HOR_
6707
6708 * CONTINUE BELOW
6709 * JR CONTINUE_GRAPHICS
6710
6711
6712
6713
6714 * PROCEDURE ROTATE_90 (TABLE_CODE(A),SOURCE(DE),DESTINATION(HL),COUNT(BC))
6715
6716 * ROTATE_90 ROTATES EACH OF A BLOCK OF GENERATORS FROM VRAM 90 DEGREES
6717 * CLOCKWISE. IF THE GENERATORS ARE FROM THE PATTERN PLANE AND THE
6718 * GRAPHICS MODE IS 2 THEN THE ROUTINE COPIES THE CORRESPONDING COLOR
6719 * ENTRIES AS WELL.
6720
6721 * BEGIN ROTATE_90
6722 * RFLCT_HOR_
6723 * ; ACTUAL ROUTINE NAME EXISTS IN OS
6724 * ; JUMP TABLE ONLY
6725
6726 * SET OPERATION CODE
6727 * LD IX,ROT_90_
6728
6729 * CONTINUE BELOW
6730 * JR CONTINUE_GRAPHICS
6731
6732
6733
6734
6735 * PROCEDURE ENLARGE (TABLE_CODE(A),SOURCE(DE),DESTINATION(HL),COUNT(BC))
6736
6737 * ENLARGE TAKES EACH OF A BLOCK OF GENERATORS AND ENLARGES IT INTO
6738 * A BLOCK OF FOUR GENERATORS WHEREIN EACH PIXEL OF THE ORIGINAL
6739 * GENERATOR IS EXPANDED TO FOUR PIXELS IN THE NEW ONES. IF THE
6740 * GENERATORS ARE FROM THE PATTERN PLANE AND THE GRAPHICS MODE IS 2
6741 * THE ROUTINE ALSO QUADRUPLES EACH OF THE CORRESPONDING COLOR
6742 * GENERATORS AS WELL.
6743
6744 * BEGIN ENLARGE

```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
106C		6745	GLB	ENLRG
		6746	ENLRG	; ACTUAL ROUTINE NAME EXISTS IN OS
		6747		; JUMP TABLE ONLY
		6748		
		6749		
106C D0211E07		6750 *	SET OPERATION CODE	IX,ENLRG_
		6751	LD	
		6752		
		6753		
1070		6754 *	CONTINUE EXECUTION HERE	
		6755	CONTINUE_GRAPHICS	
		6756		
		6757 *	SAVE ALL ENTRY PARAMETERS	
		6758	EXX	
1070 D9		6759	EX	AF,AF'
1071 08		6760	PUSH	IX
1072 D0E5		6761		; [SP] = OPERATION CODE
		6762 *	REPEAT	
1074		6763	MAIN_LOOP	
		6764		
		6765 *	GET_VRAM_ (TABLE_CODE,SOURCE,WORK_BUFFER(0..7),1)	
1074 08		6766	EX	AF,AF'
1075 F5		6767	PUSH	AF
1076 08		6768	EX	AF,AF'
1077 F1		6769	POP	AF
1078 D9		6770	EXX	
1079 05		6771	PUSH	DE
107A D9		6772	EXX	
107B D1		6773	POP	DE
107C FD210001		6774	LD	IV,1
1080 2A8006		6775	LD	HL,(WORK_BUFFER)
1083 CD18A3		6776	CALL	GET_VRAM_
		6777		
		6778 *	EXECUTE ENCODED OPERATION BELOW	
1086 D0E1		6779	POP	IX
1088 D0E5		6780	PUSH	IX
108A D0E9		6781	JP	[IX]
		6782	RETURN_HERE	\$
		6783		
		6784 *	SOURCE : SUCC (SOURCE)	DE
108C 13		6785	INC	
		6786		
108D 08		6787 *	COUNT := PRED (COUNT)	BC
		6788	DEC	
		6789		
		6790 *	UNTIL COUNT = 0	
108E 78		6791	LD	A,B
108F B1		6792	OR	C
1090 D9		6793	EXX	
1091 20E1		6794	JR	NZ,MAIN_LOOP
		6795		
		6796 *	END (ALL)	
1093		6797	ALL_X	
1093 D0E1		6798	POP	IX ; CLEAR STACK
1095 C9		6799	RET	
		6800		
1096		6801	RFLCT_VERT	

OBJECT CODE LINE SOURCE LINE

```

6802 * OPERATIONS SPECIFIC TO REFLECT_VERTICAL_ROUTINE
6803
1096 2A8006      LD      MIRROR_L_R(WORK_BUFFER[0..7],WORK_BUFFER[8..15])
1099 010008      LD      HL,(WORK_BUFFER)
109C E5         LD      BC,8
109D D1         PUSH   HL
109E 09         POP    DE
109F EB         ADD   HL,BC
10A0 CD1F00     CALL  MIRROR_L_R
6812
6813 * PUT_VRAM (TABLE_CODE,DESTINATION,WORK_BUFFER[8..15],1)
6814      CALL  PUT_TABLE
6815
6816 * IF COLOR_TEST THEN
6817      CALL  COLOR_TEST
6818      CP   TRUE
6819      JR   NZ,END_IF_1_GRAPHICS
6820
6821 * GET_VRAM(COLOR_TABLE,SOURCE,WORK_BUFFER[0..7],1)
6822      CALL  GET_COLOR
6823
6824 * PUT_VRAM(COLOR_TABLE,DESTINATION,WORK_BUFFER[0..7],1)
6825      CALL  PUT_COLOR
6826
6827      END IF
6828 END_IF_1_GRAPHICS
6829
6830 * DESTINATION := SUCC (DESTINATION)
6831      EXX
6832      INC  IMC
6833
6834 * END
6835
6836      JR   RETURN_HERE
6837
6838
1087          RFLCT HOR
6839
6840 * OPERATIONS SPECIFIC TO REFLECT_HORIZONTAL_ROUTINE
6841
6842 * MIRROR_U_D(WORK_BUFFER[0..7],WORK_BUFFER[8..15])
6843      LD      HL,(WORK_BUFFER)
6844      LD      BC,8
6845      PUSH  HL
6846      POP   DE
6847      ADD  HL,BC
6848      EX   DE,HL
6849      CALL  MIRROR_U_D
6850
6851 * PUT_VRAM (TABLE_CODE,DESTINATION,WORK_BUFFER[8..15],1)
6852      CALL  PUT_TABLE
6853
6854 * IF COLOR_TEST THEN
6855      CALL  COLOR_TEST
6856      CP   TRUE
6857      JR   NZ,END_IF_2_GRAPHICS
6858

```

LOCATION	OBJECT CODE LINE	SOURCE LINE
1DCE	CD1E89	6859 * GET_VRAM(COLOR_TABLE,SOURCE,WORK_BUFFER[0..7],1) CALL GET_COLOR
		6860
		6861
1DD1	2A8006	6862 * MIRROR_U_D(WORK_BUFFER[0..7],WORK_BUFFER[8..15]) LD HL,(WORK_BUFFER)
1DD4	010008	LD BC,8
1DD7	E5	PUSH HL
1DD8	D1	POP DE
1DD9	09	ADD HL,BC
1DDA	EB	DE,HL
1DDB	CD1F4E	CALL MIRROR_U_D
		6870
1DDE	CD1E9A	6871 * PUT_VRAM(COLOR_TABLE,DESTINATION,WORK_BUFFER[8..15],1) CALL PUT_COLOR
		6872
		6873
		6874 * END_IF
1DE1		6875 END_IF_2_GRAPHICS
		6876
1DE1	D9	6877 * DESTINATION := SUCC (DESTINATION)
1DE2	23	EXX
		6879 INC HL
		6880
1DE3	18A7	6881 * EMD JR RETURN_HERE
		6882
		6883
		6884
		6885
1DE5		6886 ROT 90
		6887 * OPERATIONS SPECIFIC TO THE ROTATE_90 ROUTINE
		6888
		6889
1DE5	2A8006	6890 * ROTATE(WORK_BUFFER[0..7],WORK_BUFFER[8..15]) LD HL,(WORK_BUFFER)
1DE8	010008	LD BC,8
1DEB	E5	PUSH HL
1DEC	D1	POP DE
1DED	09	ADD HL,BC
1DEE	EB	DE,HL
1DEF	CD1F12	CALL ROTATE
		6898
1DF2	CD1E72	6899 * PUT_VRAM (TABLE_CODE,DESTINATION,WORK_BUFFER[8..15],1) CALL PUT_TABLE
		6900
		6901
1DF5	CD1E5D	6902 * IF COLOR_TEST THEN CALL COLOR_TEST
1DF8	FE01	CP TRUE
1DFA	2006	6904 CP NZ,END_IF_3_GRAPHICS
		6905
		6906
1DFC	CD1E89	6907 * GET_VRAM(COLOR_TABLE,SOURCE,WORK_BUFFER[0..7],1) CALL GET_COLOR
		6908
		6909
1DFF	CD1E9A	6910 * PUT_VRAM(COLOR_TABLE,DESTINATION,WORK_BUFFER[0..7],1) CALL PUT_COLOR
		6911
		6912
		6913 * END_IF
1E02		6914 END_IF_3_GRAPHICS
		6915

OBJECT CODE	LINE	SOURCE LINE
1E02 D9	6916 *	DESTINATION := SUCC (DESTINATION)
1E03 23	6917	EXX
	6918	INC HL
	6919	
1E04 C31D8C	6920 *	END
	6921	JP RETURN_HERE
	6922	
	6923	
	6924	
1E07	6925 ENLRC	
	6926 *	OPERATIONS SPECIFIC TO THE ENLARGE ROUTINE
	6927	
	6928 *	MAGNIFY(WORK_BUFFER(0..7),WORK_BUFFER(8..39))
1E07 2A8006	6929	LD HL,(WORK_BUFFER)
1E0A 010008	6930	LD RC,8
1E0D E5	6931	PUSH HL
1E0E D1	6932	POP DE
1E0F 09	6933	ADD HL,BC
1E10 EB	6934	EX DE,HL
1E11 CD1EAB	6935	CALL MAGNIFY
	6936	
	6937 *	PUT_VRAM (TABLE_CODE,DESTINATION,WORK_BUFFER(8..39),4)
1E14 08	6938	EX AF,AF'
1E15 F5	6939	PUSH AF
1E16 08	6940	EX AF,AF'
1E17 F1	6941	POP AF
1E18 D9	6942	EXX
1E19 E5	6943	PUSH HL
1E1A D9	6944	EXX
1E1B D1	6945	POP DE
1E1C 2A8006	6946	LD HL,(WORK_BUFFER)
1E1F 010008	6947	LD BC,8
1E22 09	6948	ADD HL,BC
1E23 FD210004	6949	LD IY,4
1E27 CD1C27	6950	CALL PUT_VRAM
	6951	
	6952 *	IF COLOR_TEST THEN
1E2A CD1E5D	6953	CALL COLOR_TEST
1E2D FE01	6954	CP TRUE
1E2F 2024	6955	JR NZ,END_IF_4_GRAPHICS
	6956	
	6957 *	GET_VRAM(COLOR_TABLE,SOURCE,WORK_BUFFER(0..7),1)
1E31 CD1E89	6958	CALL GET_COLOR
	6959	
	6960 *	QUADRUPLE(WORK_BUFFER(0..7),WORK_BUFFER(8..39))
1E34 2A8006	6961	LD HL,(WORK_BUFFER)
1E37 010008	6962	LD BC,8
1E3A E5	6963	PUSH HL
1E3B D1	6964	POP DE
1E3C 09	6965	ADD HL,BC
1E3D EB	6966	EX DE,HL
1E3E CD1EEA	6967	CALL QUADRUPLE
	6968	
	6969 *	PUT_VRAM(COLOR_TABLE,DESTINATION,WORK_BUFFER(8..39),4)
1E41 3E04	6970	LD A,COLOR_TABLE
1E43 D9	6971	EXX
1E44 E5	4972	PUSH HL

```

LOCATION OBJECT CODE LINE SOURCE LINE
1E45 D9 6973 EXX
1E46 D1 6974 POP
1E47 2A8006 6975 HL, [WORK_BUFFER]
1E4A 010008 6976 LD BC,8
1E4D 09 6977 ADD HL,BC
1E4E FD210004 6978 LD IY,4
1E52 CD1C27 6979 CALL PUT_VRAM_
6980
1E55 6981 * END IF
6982 END_IF_4_GRAPHICS
6983
6984 * DESTINATION := DESTINATION + 4
6985 EXX
1E55 D9 6986 HL
1E56 23 6986 INC HL
1E57 23 6987 INC HL
1E58 23 6988 INC HL
1E59 23 6989 INC HL
6990
1E5A C31D8C 6991 * END RETURN_HERE
6992 JP
6993
6994
6995
6996
6997 COLOR_TEST
6998
7000
7001
7002
7003
7004
7005
7006
7007
7008
7009
7010 * BEGIN COLOR_TEST
7011
7012 * CHECK TABLE CODE IN A'
7013 EX AF,AF'
7014 PUSH AF
7015 EX AF,AF'
7016 POP AF
7017 CP PATTERN_GEN
7018 JR NZ,EXIT_FALSE
7019
7020 * CHECK MODE
7021 LD HL,VDP_MODE_WORD
7022 BIT 1,[HL]
7023 JR Z,EXIT_FALSE
7024
7025 * EXIT HERE IF TRUE
7026 LD A,TRUE
7027 RET
7028
7029 * EXIT HERE IF FALSE

```

```

; TESTS WHETHER PATTERN GENERATORS ARE
; BEING MANIPULATED AND WHETHER THE
; GRAPHICS MODE IS 2. IF SO THE ABOVE
; ROUTINES NEED TO DEAL WITH THE COLOR
; GENERATORS THAT CORRESPOND TO THE
; PATTERN GENERATORS THEY ARE OPERATING
; ON.

```

```

; NOT INPUTS, RETURNS WITH TRUE (1) IN
; A IF CONDITION IS TRUE, FALSE (0) IF
; NOT.

```

```

AF,AF'
AF
AF,AF'
AF
PATTERN_GEN
NZ,EXIT_FALSE

```

```

HL,VDP_MODE_WORD
1,[HL]
Z,EXIT_FALSE

```

```

A,TRUE

```


LOCATION OBJECT CODE LINE SOURCE LINE

```

7142 *
7143 * WRITE IT TO DESTINATION
7144 LD (DESTINATION+0),H
7145 LD (DESTINATION+16),L
7146 INC DESTINATION
7147 LD (DESTINATION+0),H
7148 LD (DESTINATION+16),L
7149 INC DESTINATION
7150
7151 * DECREMENT BYTE_COUNT
7152 DEC BYTE_COUNT
7153
7154 * UNTIL BYTE_COUNT = 0
7155 LD A,C
7156 OR B
7157 JR NZ,MAG_LOOP
7158
7159 * END
7160 RET
7161
7162 QUADRUPL
7163
7164
7165
7166
7167
7168
7169
7170 * BEGIN
7171
7172 * BYTE_COUNT := 16
7173 LD BYTE_COUNT,16
7174
7175 * SAVE SOURCE
7176 PUSH HL
7177
7178 * REPEAT
7179 QUAD_LOOP
7180
7181 * GET A BYTE FROM SOURCE
7182 LD A,[HL]
7183 INC HL
7184
7185 * WRITE IT TWICE TO DESTINATION
7186 LD (DE),A
7187 INC DE
7188 LD (DE),A
7189 INC DE
7190
7191 * DECREMENT BYTE_COUNT
7192 DEC BYTE_COUNT
7193
7194 * IF BYTE_COUNT = 8 THEN RESTORE SOURCE
7195 LD A,C
7196 CP 8
7197 JR NZ,SKIPZZ
7198 POP HL
    
```

LOCATION	OBJECT CODE	LINE	SOURCE	LINE
1EFB		7199	SKIPZZ	
		7200		
		7201	* UNTIL BYTE_COUNT = 0	
1EFB 79		7202	LD	A,C
1EFC B0		7203	OR	B
1EFD 20EF		7204	JR	NZ,QUAD_LOOP
		7205		
		7206	* END	
1EFF C9		7207	RET	
		7208	PROG	

CATION OBJECT CODE LINE SOURCE LINE

```

7210
7211 * THE ROUTINES IN THIS FILE TAKE A SINGLE 8-BYTE BLOCK AS INPUT
7212 * AND OPERATE ON IT PRODUCING A SINGLE 8-BYTE BLOCK AS OUTPUT.
7213 * THEY PERFORM MIRRORING AROUND THE VERTICAL AXIS, MIRRORING
7214 * AROUND THE HORIZONTAL AXIS, AND 90 DEGREE ROTATION.
7215
7216 GLB MIRROR_L_R
7217 GLB ROTATE
7218 GLB MIRROR_U_D
7219
7220
7221 MIRROR_L_R
7222 ; REFLECTS AN 8X8 PIXEL DATA BLOCK
7223 ; AROUND THE VERTICAL AXIS.
7224
7225 ; SOURCE IN HL, DEST IN DE
7226 ; DESTROYS AF,BC,DE,HL
7227
7228 LD BC,8 ;SET BLOCK BYTE COUNT
7229 LD B,[HL] ;GET SOURCE BYTE
7230 LD A,80H ;SET WORK REGISTER MASK
7231
7232 RL B ;PUT SOURCE IN CARRY
7233 RRA ;PUT CARRY INTO WORK REGISTER
7234 JR NC,MIR_L_R20 ;CONTINUE UNTIL MASK BIT IS IN CARRY
7235 LD [DE],A ;WRITE MIRRORED BYTE TO DESTINATION
7236 INC HL ;ADVANCE POINTERS
7237 INC DE
7238 DEC C
7239 JR NZ,MIR_L_R10 ;CHECK COUNT
7240 MIR_L_RX
7241
7242
7243
7244
7245 ROTATE
7246 ; ROTATE OBJECT 90 DEGREES
7247
7248 ; SOURCE IN HL DESTINATION IN DE.
7249 ; DESTROYS AF,BC,DE,HL
7250
7251
7252
7253
7254 TRANSP_10
7255
7256 RL [IX+0] ;PUT HI BIT OF FIRST SOURCE BYTE IN CARRY
7257 RR [HL] ;PUT CARRY IN DESTINATION BYTE
7258 RL [IX+1]
7259 RR [HL]
7260 RL [IX+2]
7261 RR [HL]
7262 RL [IX+3]
7263 RR [HL]
7264 RL [IX+4]
7265 RR [HL]
7266 RL [IX+5]
7267 RR [HL]
7268
7269
7270
7271
7272
7273
7274
7275
7276
7277
7278
7279
7280
7281
7282
7283
7284
7285
7286
7287
7288
7289
7290
7291
7292
7293
7294
7295
7296
7297
7298
7299
7300
7301
7302
7303
7304
7305
7306
7307
7308
7309
7310
7311
7312
7313
7314
7315
7316
7317
7318
7319
7320
7321
7322
7323
7324
7325
7326
7327
7328
7329
7330
7331
7332
7333
7334
7335
7336
7337
7338
7339
7340
7341
7342
7343
7344
7345
7346
7347
7348
7349
7350
7351
7352
7353
7354
7355
7356
7357
7358
7359
7360
7361
7362
7363
7364
7365
7366
7367
7368
7369
7370
7371
7372
7373
7374
7375
7376
7377
7378
7379
7380
7381
7382
7383
7384
7385
7386
7387
7388
7389
7390
7391
7392
7393
7394
7395
7396
7397
7398
7399
7400
7401
7402
7403
7404
7405
7406
7407
7408
7409
7410
7411
7412
7413
7414
7415
7416
7417
7418
7419
7420
7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432
7433
7434
7435
7436
7437
7438
7439
7440
7441
7442
7443
7444
7445
7446
7447
7448
7449
7450
7451
7452
7453
7454
7455
7456
7457
7458
7459
7460
7461
7462
7463
7464
7465
7466
7467
7468
7469
7470
7471
7472
7473
7474
7475
7476
7477
7478
7479
7480
7481
7482
7483
7484
7485
7486
7487
7488
7489
7490
7491
7492
7493
7494
7495
7496
7497
7498
7499
7500
7501
7502
7503
7504
7505
7506
7507
7508
7509
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600
7601
7602
7603
7604
7605
7606
7607
7608
7609
7610
7611
7612
7613
7614
7615
7616
7617
7618
7619
7620
7621
7622
7623
7624
7625
7626
7627
7628
7629
7630
7631
7632
7633
7634
7635
7636
7637
7638
7639
7640
7641
7642
7643
7644
7645
7646
7647
7648
7649
7650
7651
7652
7653
7654
7655
7656
7657
7658
7659
7660
7661
7662
7663
7664
7665
7666
7667
7668
7669
7670
7671
7672
7673
7674
7675
7676
7677
7678
7679
7680
7681
7682
7683
7684
7685
7686
7687
7688
7689
7690
7691
7692
7693
7694
7695
7696
7697
7698
7699
7700
7701
7702
7703
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713
7714
7715
7716
7717
7718
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736
7737
7738
7739
7740
7741
7742
7743
7744
7745
7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757
7758
7759
7760
7761
7762
7763
7764
7765
7766
7767
7768
7769
7770
7771
7772
7773
7774
7775
7776
7777
7778
7779
7780
7781
7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793
7794
7795
7796
7797
7798
7799
7800
7801
7802
7803
7804
7805
7806
7807
7808
7809
7810
7811
7812
7813
7814
7815
7816
7817
7818
7819
7820
7821
7822
7823
7824
7825
7826
7827
7828
7829
7830
7831
7832
7833
7834
7835
7836
7837
7838
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7850
7851
7852
7853
7854
7855
7856
7857
7858
7859
7860
7861
7862
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7890
7891
7892
7893
7894
7895
7896
7897
7898
7899
7900
7901
7902
7903
7904
7905
7906
7907
7908
7909
7910
7911
7912
7913
7914
7915
7916
7917
7918
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7930
7931
7932
7933
7934
7935
7936
7937
7938
7939
7940
7941
7942
7943
7944
7945
7946
7947
7948
7949
7950
7951
7952
7953
7954
7955
7956
7957
7958
7959
7960
7961
7962
7963
7964
7965
7966
7967
7968
7969
7970
7971
7972
7973
7974
7975
7976
7977
7978
7979
7980
7981
7982
7983
7984
7985
7986
7987
7988
7989
7990
7991
7992
7993
7994
7995
7996
7997
7998
7999
8000

```

```

LOCATION OBJECT CODE LINE SOURCE LINE
1F30 DDCB0616 7267 RL (IX+6)
1F41 CB1E 7268 RR (HL)
1F43 DDCB0716 7269 RL (IX+7)
1F47 CB1E 7270 RR (HL)
1F49 23 7271 INC HL
1F4A 00 7272 DEC C
1F4B 20CC 7273 JR NZ,TRANSP_10
1F4D 1F40 7274 TRANSP_X
1F4D C9 7275 RET
7276
7277
7278
1F4E 7279 MIRROR_U_D ;REFLECT 8X8 PIXEL BLOCK AROUND THE
; HORIZONTAL AXIS
7280
7281
7282
7283 ; SOURCE IN HL, DESTINATION IN DE
7284 ; DESTROYS AF,BC,DE,HL
7285 * SOURCE := SOURCE + 7
7286 LD BC,7
7287 ADD HL,BC
7288
7289 * BYTE COUNT := 8 BC
7290 INC BC
7291
7292 * REPEAT $
7293 REFLECT_LOOP EQU $
7294
7295 * [DESTINATION] := [SOURCE] A, [HL]
7296 LD [DE],A
7297 LD
7298
7299 * DESTINATION := SUCC (DESTINATION) DE
7300 INC DE
7301
7302 * SOURCE := PRED (SOURCE) HL
7303 DEC HL
7304
7305 * BYTE COUNT := PRED BYTE COUNT BC
7306 DEC BC
7307
7308 * UNTIL BYTE COUNT = 0
7309 LD A,B
7310 OR C
7311 JR NZ,REFLECT_LOOP
7312 * END
7313 RET
7314
7315 ;Modified February 14, 1983. Filler locations were
7316 ;changed to OFFH to reflect OS_7PRIME.
7317
7318 HEX FF,FF,FF,FF ;Filler
7319
7320
7321

```

```

XCATION OBJECT CODE LINE      SOURCE LINE
7324 ***** ROM_JUMP_TABLE *****
7325 ;
7326 ; JUMP_TABLE THIS IS THE JUMP TABLE TO BE USED IN ACCESSING CODE
7327 ; RESIDING IN THE O.S. ROM. THIS TABLE MUST HAVE ITS
7328 ; ORIGIN REDEFINED TO ACCOUNT FOR GROWTH. PILE NEW ROUTINES
7329 ; AT THE BEGINNING OF THE TABLE MAKING SURE TO INCREMENT
7330 ; THE NO_OF_ROUTINES VALUE.
7331
7332 * NOTE ****
7333
7334 * ***** NO DELETIONS SHOULD BE MADE FROM *****
7335 * ***** THIS TABLE *****
7336
<2000> 7337 ROM_END EQU 2000H
7338 * THIS IS THE END OF OS ROM
7339
<0035> 7340 NO OF ROUTINES EQU 53
7341 * THIS NUMBER KEEPS COUNT OF THE NUMBER OF ROUTINES ACCESSED THROUGH
7342 * THE JUMP TABLE.
7343
7344 JUMP_TABLE ORG ROM_END-(NO_OF_ROUTINES*3)
7345
1F61 C30300 7346 PLAY_SONGS JP PLAY_SONGS
1F64 C30488 7347 ACTIVATED JP ACTIVATED
1F67 C306C7 7348 PUT0BJP JP PUT0BJP
1F6A C3105A 7349 REFLECT_VERTICAL JP RFLCT_VERT
1F6D C31060 7350 REFLECT_HORIZONTAL JP RFLCT_HOR
1F70 C31066 7351 ROTATE_90 JP ROT_90
1F73 C3106C 7352 ENLARGE JP ENLGR
1F76 C3114A 7353 CONTROLLER_SCAN JP CONT_SCAN
1F79 C31188 7354 DECODER JP DECODER
1F7C C31979 7355 GAME_OPT JP GAME_OPT
1F7F C31927 7356 LOAD_ASCII JP LOAD_ASCII
1F82 C31804 7357 FILL_VRAM JP FILL_VRAM
1F85 C318E9 7358 MODE_1 JP MODE_1
1F88 C3116A 7359 UPDATE_SPINNER JP UPDATE_SPINNER
1F8B C3180E 7360 INIT_TABLE JP INIT_TABLE
1F8E C3188C 7361 GET_VRAM JP GET_VRAM
1F91 C31C10 7362 PUT_VRAM JP PUT_VRAM
1F94 C31C5A 7363 INIT_SPR_ORDERP JP INIT_SPR_ORDERP
1F97 C31C76 7364 WR_SPR_MM_TBL JP WR_SPR_MM_TBL
1F9A C30F9A 7365 INIT_TIMER JP INIT_TIMER
1F9D C30F88 7366 FREE_SIGNALP JP FREE_SIGNALP
1FA0 C31044 7367 REQUEST_SIGNALP JP REQUEST_SIGNALP
1FA3 C3108F 7368 TEST_SIGNALP JP TEST_SIGNALP
1FA6 C31C8C 7369 WRITE_REGISTERP JP REG_WRITEQ
1FA9 C31CED 7370 WRITE_VRAM JP VRAM_WRITEQ
1FAC C31D2A 7371 READ_VRAM JP VRAM_READQ
1FAF C30655 7372 INIT_WRITEP JP INIT_QUEUEQ
1FB2 C30203 7373 SOUND_INITP JP INIT_SOUNDQ
1FB5 C30251 7374 PLAY_TIP JP JUKE_BOXQ
1FB8 C31B1D 7375 INIT_TABLE JP INIT_TABLE
1FBB C31BA3 7376 GET_VRAM JP GET_VRAM
1FC1 C31C66 7377 PUT_VRAM JP PUT_VRAM
1FC4 C31C82 7378 INIT_SPR_ORDER JP INIT_SPR_ORDER
1FC7 C30FAA 7379 WR_SPR_MM_TBL JP WR_SPR_MM_TBL
7380 INIT_TIMER JP INIT_TIMER

```

LOCATION OBJECT CODE LINE SOURCE LINE

1FCA C30FC4	7381 FREE_SIGNAL	JP FREE_SIGNAL
1FCD C31053	7382 REQUEST_SIGNAL	JP REQUEST_SIGNAL
1FD0 C310C8	7383 TEST_SIGNAL	JP TEST_SIGNAL
1FD3 C30F37	7384 TIME_MGR	JP TIME_MGR
1FD6 C30238	7385 TURN_OFF_SOUND	JP ALL_OFF
1FD9 C31CCA	7386 WRITE_REGISTER	JP REG_WRITE
1FDC C31057	7387 READ_REGISTER	JP REG_READ
1FDF C31001	7388 WRITE_VRAM	JP VRAM_WRITE
1FE2 C31D3E	7389 READ_VRAM	JP VRAM_READ
1FE5 C30664	7390 INIT_WRITER	JP INIT_QUEUE
1FEB C30679	7391 WRITER	JP WRITER
1FEC C311C1	7392 POLLER	JP POLLER
1FEE C30213	7393 SOUND_INIT	JP INIT_SOUND
1FF1 C3025E	7394 PLAY_IT	JP JUKE_BOX
1FF4 C3027F	7395 SOUND_MAN	JP SMD_MANAGER
1FF7 C304A3	7396 ACTIVATE	JP ACTIVATE
1FFA C306D8	7397 PUTOBJ	JP PUTOBJ
1FFD C30038	7398 RAND_GEN	JP RAND_GEN
	7399	

Errors= 0

.LINE#	SYMBOL	TYPE	REFERENCES
7396	ACTIVATE	A	242
7347	ACTIVATEP	A	248
1610	ACTIVATEQ	P	1609, 7347
1627	ACTIVATE_P	P	1587, 1683, 7396
1607	ACTIVATE_P	P	1611
1915	ACT_OSPRT	P	1648
1917	ACT_TSPRT	P	1650
1662	ACT_CMLX	P	1652
1903	ACT_MOBILE	P	1646
1694	ACT_SEMI	P	1644
884	ADD816	P	703, 883
5152	ADDR_ADJ	P	5172
5171	ADD_8	P	5152
6141	ADJUST_COUNT	P	6145
6130	ADJUST_INDEX	P	6134
3723	AD_EXIT	P	3719
3718	AD_LP	P	3721
463	AFTER_RANDOM	P	
85	ALEN	A	
1052	ALL_OFF	P	992, 7385
6797	ALL_X	P	
492	AMERICA	P	236
87	APS	A	
88	APSV	A	733
964	AREA_SONG_IS	P	963, 1216
4504	ARM_DBNCE	A	4856, 5027
5008	ARM_EXIT	P	4859
5037	ARM_EXIT	P	5022, 5028, 5033
4522	ARM_MASK	A	4746, 5011
4516	ARM_OLD	A	5013, 5021, 5034
5024	ARM_REG	P	5020
5030	ARM_ST1	P	5016
4517	ARM_STATE	A	5014, 5026, 5036
499	ASCII_TABLE	P	254
5458	ASCII_TBL	P	499, 5080
5382	ASC_TABLE	P	5681
86	ASTEP	A	1373, 1445, 1464
79	ATN	A	1370
724	ATN_SWEEP	P	721, 1229
1038	B1	P	1040
6065	BASE_FACTORS	P	6049
3211	BK_CLR	A	3384, 3413, 3600, 3658, 3744, 3833
3209	BK_PTN	A	3351, 3413, 3547, 3650, 3735, 3744, 3833
368	BOOT_UP	P	
1980	BUFFER	D	2021, 2069, 2129
7107	BYTE_COUNT	S	
2689	CALC_OFFSET	P	2480, 2644
457	CARRY_READY	P	454
261	CARTRIDGE	A	260, 534
6042	CASE_OF_CLR10	P	6039
6035	CASE_OF_COLOR	P	6023
6025	CASE_OF_GEN	P	6021
6032	CASE_OF_GEN10	P	6029
5705	CENTER_PRT	P	5234, 5244
75	CH	A	
99	CHO	A	
90	CHOEND	A	

LINE#	SYMBOL	TYPE	REFERENCES
94	CHOREP	A	
100	CH1	A	
91	CH1END	A	
95	CH1REP	A	
101	CH2	A	
92	CH2END	A	
96	CH2REP	A	
102	CH3	A	
93	CH3END	A	
97	CH3REP	A	
4784	CHK_PLYR_1	P	4766,4780
4777	CHK_SEG_01	P	4772
4801	CHK_SEG_11	P	4796
4579	CIMIT1	P	4587
1677	CMPXK4	P	1690
1691	CMPXK9	P	1676
2823	COLOR	A	
6337	COLORTABLE	D	5902,6330
2830	COLOR_AND_TAG	A	2936,3024,3064,3141
6670	COLOR_TABLE	A	6970,7059,7074
6997	COLOR_TEST	P	6817,6855,6903,6953
3201	COLR	A	3472,3602,3835
3559	COMBINE_LOOP	P	3593
3797	COM_PAT_COL	P	3576
3029	CONTINUE	P	2900,2988
6755	CONTINUE_GRAPHI	P	6690,6709,6730
4534	CONTROLLER_0	A	4611,4712
4535	CONTROLLER_1	A	
4570	CONTROLLER_INIT	P	555,4569
285	CONTROLLER_MAP	A	284,4573,4762
7353	CONTROLLER_SCAN	A	212
4609	CONT_READ	P	4720,4735
4615	CONT_READ1	P	4612
4617	CONT_READX	P	4614
4626	CONT_SCAN	P	4625,4760,7353
80	CTRL	A	1307
4531	CTRL_0_PORT	A	4613,4627,4635,4654
4530	CTRL_1_PORT	A	4616,4630,4638,4668
6432	CTRL_PORT	A	6433,6482,6487,6540,6544,6590,6594,6624
3594	C_LP_EXIT	P	3591
6431	DATA_PORT	A	6433,6550,6599
5050	DBNCE_BUFF	D	4576,4761
4092	DCR_L_MODE_TBL	P	
4104	DCR_L_RPT_TBL	P	4091
4129	DCR_S_MODE_TBL	P	4089
4086	DCR_TIMER	P	4072
833	DECLSM	P	689,735,741,832
850	DECSM	P	849
7354	DECODER	A	213
4748	DECODERX	P	4728
4701	DECODER	P	4700,7354
4820	DECODE_0	P	4776,4800
4842	DECODE_OX	P	4835
4854	DECODE_1	P	4783,4807
4866	DECODE_1X	P	4863
4827	DEC_FIRE	P	4823
4861	DEC_K80	P	4857

LINE#	SYMBOL	TYPE	REFERENCES
4549	DEC_KBD_TBL	P	4739,4899
4715	DEC_PLYR	P	4713
4733	DEC_SEG1	P	4704
4833	DEC_SPHR	P	4829
59	DEDAREA	A	60,61,62,63,64,65
2181	DEFER	A	2184
630	DEFER_WRITES	D	238,559,2086,2091,2153,2183
4599	DELAY	P	4542
5723	DELAY_10	P	5258
7109	DESTINATION	S	
1535	DE_TO_DEST	P	1343,1441,1461,1487
5126	DISPLAY_LOGO	P	566,5082
5054	DIVIDE	P	6058
3304	DLP1	P	3324
3355	DLP2	P	3425
3479	DLP4	P	3542
3613	DLP5	P	3619
3638	DLP6	P	3642
4048	DONE	A	4473,4486
3167	DONE_LOGO	P	5147
1203	DONE_SHDMMAN	P	1178
1120	DONT_PUT	P	2876,2879,2889,2892,2964,2967,2977,2980
2188	DO_PUT08J	P	2113,2185
1063	DUMAREA	P	992,1044,1168
1876	DUP1	P	1878
4681	DVEX	P	3677
4676	DVLP	P	3680
1399	EFFECT	P	1342
1233	EFXOVER	P	1213
1705	ELSE04	P	3646
1223	ELSE1	P	3219
1530	ELSE10	P	3516
1608	ELSE13	P	3605
1810	ELSE18	P	3800
1282	ELSE2	P	3278
1041	ELSE23	P	3838
1339	ELSE5	P	3336
1400	ELSE6	P	3378
1463	ELSE8	P	3460
1501	ELSE9	P	3483
1227	ELSEZ2	P	6198,6201
1277	ELSE_1	P	2268
1702	ELSE_11	P	2694
1726	ELSE_12	P	2718
1536	ELSE_8	P	2503
1559	ELSE_9	P	2543,2544
1757	END04	P	3704
1225	END1	P	3221
1537	END10	P	3529
1587	END11	P	3584
1620	END12	P	3598
1610	END13	P	3607
1628	END14	P	3626
1750	END15	P	3728
1782	END16	P	3766
1781	END17	P	3773
1828	END18	P	3809

LINE#	SYMBOL	TYPE	REFERENCES
3815	END19	P	3813
3284	END2	P	3260
3821	END20	P	3819
3827	END21	P	3825
3872	END22	P	3831
3843	END23	P	3840
3852	END24	P	3847
3861	END25	P	3856
3870	END26	P	3865
3321	END3	P	3308
3313	END4	P	3311
3361	END5	P	3338
3420	END6	P	3399
3419	END7	P	3406
3466	END8	P	3462
3538	END9	P	3500
1389	ENDOREP	P	1342, 1382
1384	ENDREP	P	1342
73	ENDSDATA	A	1143, 1177
6147	END_ADJ_COUNT	P	6140
6135	END_ADJ_INDEX	P	
376	END_BOOTUP	P	
6232	END_IF2	P	6225
2320	END_IF_1	P	2276
2610	END_IF_10	P	2584, 2589
2707	END_IF_11	P	2698
2731	END_IF_12	P	2722
6828	END_IF_1_GRAPH1	P	6819
2344	END_IF_2	P	2326
6875	END_IF_2_GRAPH1	P	6857
2363	END_IF_3	P	2358
6914	END_IF_3_GRAPH1	P	6905
2419	END_IF_4	P	2395, 2396
6982	END_IF_4_GRAPH1	P	6955
2572	END_IF_8	P	2532
2568	END_IF_9	P	2555
6611	END_INPUT	P	6607
6562	END_OUTPUT	P	6558
7352	ENLARGE	A	246
6746	ENLRG	P	6745, 7352
6925	ENLRG	P	6751
4051	EOT	A	4073, 4215, 4246, 4407, 4425, 4464
2271	EQUAL_TO	P	2267
4305	EXIT	P	4264
7030	EXIT_FALSE	P	7018, 7023
3155	EXIT_PUT_SPR	P	3118
7129	EXP_1	P	7134
7136	EXP_2	P	7141
526	FALSE	A	558, 2090, 7030
5616	FILL	P	5621
7357	FILL_VRAM	A	252, 5780, 5905
5610	FILL_VRAM	P	5084, 5129, 7357
4501	FIRE	A	4828, 4945
4926	FIRE_DBNCE	P	4831
4955	FIRE_EXIT	P	4940, 4946, 4951
4521	FIRE_MASK	A	4726, 4929
4510	FIRE_OLD	A	4931, 4939, 4952

NE#	SYMBOL	TYPE	REFERENCES
942	FIRE_REG	P	4938
948	FIRE_ST1	P	4934
511	FIRE_STATE	A	4932, 4944, 4954
811	FIRST_GEN_NAME	A	3104
202	FLAGS	A	3225, 3375, 3596, 3603, 3644, 3798, 3829, 3836
82	FPS	A	686
83	FPSV	A	2926, 3014, 3055, 3092
817	FRAME	A	2915, 3003, 3044, 3081
814	FRAME_TABLE_PTR	A	4071, 4222, 4224, 4248, 4344, 4432, 4471, 4483
050	FREE	P	4219
214	FREE1	P	4216
237	FREE_COUNTER	P	4212
311	FREE_EXIT	P	4223, 4226, 4228
221	FREE_MATCH	P	224
309	FREE_SET	A	229
381	FREE_SIGNAL	A	4041, 7366
366	FREE_SIGNALP	P	4040, 7381
200	FREE_SIGNALQ	P	4201
206	FREE_SIGNAL	P	807, 811, 814
197	FREE_SIG_PAR	A	668, 1230
78	FREQ	P	3230, 3441, 3624
671	FREQ_SWEEP	A	673, 702, 1372, 1444
203	FRM	A	3269, 3301, 3623
84	FSTEP	A	332, 5230, 5232, 5237, 5247
204	F_GEN	A	250
333	GAME_NAME	P	5761, 7355
355	GAME_OPT	P	2377, 2642, 3262
777	GAME_OPT	P	6822, 6860, 6908, 6958
643	GET_BKGRND	P	4249, 4253, 4258, 4263, 4277
055	GET_COLOR	P	2291
278	GET_NEXT	P	189, 3132, 3380, 3398, 3411, 3418
294	GET_OLD	A	194
376	GET_VRAM	P	5980, 7361
361	GET_VRAM	P	2659, 5979, 6776, 7066, 7376
377	GET_VRAM	P	6078
387	GET_VRAM	P	2913, 2914, 3001, 3002, 3042, 3043, 3079, 3080, 3102, 3103
174	GET_VRAM_P	A	1997, 2022, 2032, 2070
305	GRAPHICS	D	3582
772	HEAD_ADDRESS	P	1063, 1180, 1217, 1351, 1389, 1521
585	IF11	P	1822
71	INACTIVE	A	2057, 7390
321	INIT_80	P	2049, 7372
358	INIT_QUEUE	P	2051
350	INIT_QUEUE	P	992, 7393
345	INIT_QUEUE_P	P	1011, 7373
326	INIT_SOUNDQ	P	1020, 1022, 1024
318	INIT_SOUNDQ	P	1019
385	INIT_SOUND_DATA	D	6260
312	INIT_SOUND_PAR	P	191
355	INIT_SPR10	P	196
378	INIT_SPR_ORDER	A	5980, 7363
163	INIT_SPR_ORDERP	A	5979, 7378
342	INIT_SPR_ORDERQ	P	6243
348	INIT_SPR_ORDER	P	188, 5649, 5654, 5659, 5664, 5669
340	INIT_SPR_P	P	6017, 6024
175	INIT_TABLE	A	
346	INIT_TABLE80	P	

LINE#	SYMBOL	TYPE	REFERENCES
6061	INIT_TABLE90	P	6031,6034,6041,6044
7360	INIT_TABLEP	A	193
5990	INIT_TABLEQ	P	5980,7360
5997	INIT_TABLE_P	P	5979,7375
5967	INIT_TABLE_P	P	5991
7380	INIT_TIMER	A	223
7365	INIT_TIMERP	A	228
4174	INIT_TIMERQ	P	4039,7365
4180	INIT_TIMER	P	4038,7380
4430	INIT_TIMER_EXI	P	4370,4396,4404
587	INIT_TIME_DATA	D	4176
4171	INIT_TIME_PAR	P	4175
7390	INIT_WRITER	A	203
7372	INIT_WRITERP	A	208
1807	INIT_XP_OS	P	1696,1905
6601	INPUT_LOOP	P	6605,6608
423	IRQ_INTERRUPT	P	
325	IRQ_INT_VECT	A	324,424
4502	JOY	A	4822,4986
4967	JOY_DBNCE	P	4825
4996	JOY_EXIT	P	4981,4987,4992
4523	JOY_MASK	A	4722,4970
4512	JOY_OLD	A	4972,4980,4993
4983	JOY_REG	P	4979
4989	JOY_ST1	P	4975
4513	JOY_STATE	A	4973,4985,4995
1102	JUKE_BOX	P	1075,1385,7394
1095	JUKE_BOXQ	P	1094,7374
1083	JUKE_BOX_PAR	P	1096
7344	JUMP_TABLE	A	
4505	KBD	A	4862,4903
4878	KBD_DBNCE	P	4865
4913	KBD_EXIT	P	4893,4904,4909
4520	KBD_MASK	A	4738,4882
4529	KBD_NULL	A	4550,4557,4561,4565
4518	KBD_OLD	A	4884,4892,4910
4895	KBD_REG	P	4891
4906	KBD_ST1	P	4887
4519	KBD_STATE	A	4885,4897,4912
1143	L1	P	1152
1229	L10	P	1222
1244	L12	P	1232,1240
1377	L13	P	1361
1394	L14	P	1379
1424	L15	P	1397
1448	L16	P	1429
1467	L17	P	1450
1482	L18	P	1479
1534	L19	P	
1176	L2	P	1202
684	L20	P	676
1528	L20_LOAD_MEX	P	1526
708	L21	P	691
766	L22	P	737,761
763	L23	P	743
793	L24	P	787
1306	L5	P	1302

LINE#	SYMBOL	TYPE	REFERENCES
1318	L6	P	1304, 1312
1327	L7	P	1323
1330	L8	P	1325
1199	L9	P	1182
	942 LEAVE EFFECT	P	941
5922	LINE_1	P	5797
5931	LINE_10	P	5854
5932	LINE_11	P	5959
5933	LINE_12	P	5965
5923	LINE_2	P	5803
5924	LINE_3	P	5941
5925	LINE_4	P	5947
5926	LINE_5	P	5950
5927	LINE_6	P	5953
5928	LINE_7	P	5842
5929	LINE_8	P	5846
5930	LINE_9	P	5850
7356	LOAD_ASCII	A	251, 5792
5681	LOAD_ASCII	P	5083, 5137, 7356
1346	LOAD_NEXT_NOTE	P	1120, 1235, 1342
268	LOCAL_SPR_TBL	A	267, 6207, 6300
5304	LOGO_COLORS	P	5205
5329	LOGO_GEN	P	5150
5309	LOGO_NAMES	P	5177, 5183, 5197, 5270, 5276, 5282
4052	LONG	A	4088, 4227, 4372
4371	LONG_TIMER	P	4355
3961	LP1	P	4002
4009	LP2	P	4022
2512	LT33	P	2510
7099	MAGNIFY	P	6935, 7092
7122	MAG_LOOP	P	7157
6763	MAIN_LOOP	P	6794
4415	MAKE_NEW_TIMER	P	4408
7221	MIRROR_LR	P	6811, 7216
7279	MIRROR_UD	P	6849, 6869, 7218
7228	MIR_LR10	P	7239
7231	MIR_LR20	P	7234
7240	MIR_LR_RX	P	
1517	MOD80	P	1343, 1374, 1391, 1419, 1446, 1465, 1496
5075	MODE_0_PORT	A	5617, 6312
7358	MODE_1	A	253, 5783
5636	MODE_1	P	5085, 5132, 7358
5076	MODE_1_PORT	A	5612, 5615, 6293, 6296
4285	MOVE_IT	P	4247
865	MSNTOLSN	P	693, 739, 864
636	MUX_SPRITES	D	237, 562, 6199
2304	M_XY	P	2303, 2305
2411	M_XY2	P	2410, 2412
2829	NAME	A	3107
2453	NEGTV	P	2445
4245	NEXT	P	4284
77	NEXTMOTEPTR	A	
1868	NEXT_COLOR	P	1891
2820	NEXT_GEN	A	
4070	NEXT_TIMER0	P	4078
4406	NEXT_TIMER1	P	4345
4496	NEXT_TIMEP	A D	4031, 4184, 4291, 4303, 4378, 4395

LINE#	SYMBOL	TYPE	REFERENCES
81	MLEN	A	677,680,1371
485	MMI_INTERRUPT	P	
329	MMI_INT_VECT	A	328,486
3695	MOTOS	P	3683,3687
4397	NOT_A_LONG_REPE	P	4375
4365	NOT_A_REPEAT_T1	P	4363
448	NOT_ON	P	443
6495	NOT_REG_0	P	6492
6503	NOT_REG_1	P	6500
2026	NOT_TOO_BIG	P	2014
5270	NO_CARTRIDGE	P	5222,5226
7340	NO_OF_ROUTINES	A	7344
545	NO_TEST	P	537,540
1272	NSRHI	A	
1274	NSRLOW	A	
1273	NSRNEED	A	
1275	NSRTG3	A	
1624	NTZZZ	P	1621
503	NUMBER_TABLE	P	255
5422	NUMBER_TBL	P	503,5081
2813	NUMGEN	A	
4525	NUM_DEV	A	4577,4791,5050
2810	OBJECT_TYPE	A	
3210	OBJ_PTN	A	3452,3475,3554
5586	OBJ_TABLE	P	5140
67	OFF	A	1054,1056,1058,1060,1278,1284,1290,1296
2880	OK_1	P	2874
2893	OK_2	P	2887
2968	OK_3	P	2962
2981	OK_4	P	2975
1862	ONE_BYTE	P	1850
1822	OS_IN_VRAM	P	1817
6552	OUTPUT_LOOP	P	6556,6559
6313	OUTPUT_LOOP10	P	6317
6299	OUTPUT_LOOP_TAB	P	6322
1854	O_B_RET	P	1892
649	PARAM	P	647,1021,1098,1613,2053,2176,4177,4203,4333,4452,5993,6080,6179,6245,6272,6472,6523,6580
584	PARAM_AREA	D	1097,1099,2052,2054,2055,2175,2177,2178,5992,5994,5995,6079,6081,6082,6083,6084,6178,6180,6181,6182,6183
5697	PARSE	P	6244,6246,6271,6273,6471,6473,6522,6524,6525,6526,6579,6581,6582,6583
1416	PASST	P	5231,5238,5242,5248,5250
6669	PATTERN_GEN	A	1343,1413
6336	PATTRNGENTBL	D	7017
6335	PATTRNAMETBL	D	6329
1271	PERIOD	A	6328
2485	PF1	P	
2538	PF2	P	
2561	PF3	P	
2579	PF4	P	
7394	PLAY_IT	A	217
7374	PLAY_TTP	A	221
7346	PLAY_SONGS	A	219
1276	PLAY_SONGS_	P	1255,7346
4506	PLYR_0	A	4769
4507	PLYR_1	A	4793
3256	PM1	P	
3513	PM10	P	

LINE#	SYMBOL	TYPE	REFERENCES
3545	PM11	P	
3644	PM12	P	
3706	PM13	P	
3761	PM14	P	
3829	PM15	P	
3264	PM2	P	
3291	PM3	P	
3328	PM4	P	
3347	PM5	P	
3372	PM51	P	3369
3368	PM52	P	3371
3377	PM6	P	
3404	PM7	P	
3432	PM8	P	
3473	PM9	P	
7392	POLLER	A	210
7759	POLLER	P	4758,7392
8008	POLLER_X	P	4788,4804
888	POS	P	886
531	POWER UP	P	375
586	PRM AREA	D	1612,1614,1619
1215	PROCESS DATA AR	P	1147,1213
2812	PTRN POINTER	A	
60	PTR TO LST OF S	A	909,1028
61	PTR TO S ON 0	A	1045,1169,1190,1298
62	PTR TO S ON 1	A	1046,1170,1281
63	PTR TO S ON 2	A	1047,1171,1287
64	PTR TO S ON 3	A	1048,1172,1293
906	PT IX TO S DATA	P	905,1105,1141,1174
856	PUTOSPRITE	P	2197,2853
944	PUT1SPRITE	P	2199,2853
1913	PUTCOMPLEX	P	2200,3911
473	PUTFRAME	P	2257,2340,2387,2472,3780,3791
397	PUTOBJ	A	241,4018
348	PUTOBJP	A	247
173	PUTOBJQ	P	2165,7348
182	PUTOBJJ	P	2158,7397
166	PUTOBJ PAR	P	2174
216	PUTSEMI	P	2193,2213
7070	PUT COLOR	P	6825,6872,6911
212	PUT MOBILE	P	2195,3196
035	PUT TABLE	P	6814,6852,6900
377	PUT VRAM	A	190,3116,3150,3654,3692,3742,3749,5160,5181,5187,5194,5201,5209,5255,5274,5280,5286,5685,5692,5720,5801
7362	PUT VRAMP	A	5807,5944,5956,5962,5968
176	PUT VRANG	P	195
185	PUT VRAM	P	5980,7362
173	PUT VRAM P	P	1775,1803,1838,1853,1882,1946,2599,5979,6950,6979,7051,7081,7377
067	PUT_Y_AND_NAME	P	6177
435	PX TO PTRN_POS	P	2938,3026
698	P LOOP	P	2222,2227,2434,3242,3251
162	QUADRUPLE	P	5703
179	QUAD LOOP	P	6967,7093
968	QUEUE HEAD	D	7204
964	QUEUE SIZE	D	2008,2018,2049,2065,2096
969	QUEUE TAIL	D	2012,2061,2120
398	RAND_GEN	A	2066,2095,2116,2126,2138

LINE#	SYMBOL	TYPE	REFERENCES
440	RAND_GEN	P	7398
644	RAND_NUM	D	441,460,552,643
7387	READ_REGISTER	A	200,5622
7389	READ_VRAM	A	202,3279,3461,3521
7371	READ_VRAMP	A	207
7350	REFLECT_HORIZONTAL	A	244
7293	REFLECT_LOOP	P	7311
7349	REFLECT_VERTICAL	A	243
6621	REG_READ	P	6620,7387
6478	REG_WRITE	P	6062,6477,7386
6469	REG_WRITEQ	P	6468,7369
6464	REG_WRITE_P	P	6463,6470
4049	REPEAT	A	4090,4135,4225,4364,4380,4481
596	REPEAT_SIG_CODE	D	4332,4335
7382	REQUEST_SIGNAL	A	225
7367	REQUEST_SIGNALP	A	230
4330	REQUEST_SIGNALQ	P	4043,7367
4337	REQUEST_SIGNAL	P	4042,7382
4327	REQUEST_SIG_PAR	P	4331
455	RESET	P	447,450
1363	REST	P	1342
6782	RETURN_HERE	P	6835,6882,6921,6992
6702	RFLCT_HOR	P	6701,7350
6839	RFLCT_HOR	P	6706
6683	RFLCT_VERT	P	6682,7349
6801	RFLCT_VERT	P	6687
7337	ROM_END	A	7344
7245	ROTATE	P	6897,7217
7351	ROTATE_90	A	245
6723	ROT_90	P	6722,7351
6886	ROT_90	P	6727
3664	RPT1	P	3701
3708	RPT2	P	3755
2577	RPT_1	P	2627
2652	RPT_2	P	2675
393	RST_10H	P	
305	RST_10H_RAM	A	304,393
396	RST_18H	P	
309	RST_18H_RAM	A	308,396
399	RST_20H	P	
313	RST_20H_RAM	A	312,399
402	RST_28H	P	
317	RST_28H_RAM	A	316,402
405	RST_30H	P	
321	RST_30H_RAM	A	320,405
390	RST_8H	P	
301	RST_8H_RAM	A	300,390
5054	S0_C0	D	4591,4629,4774
5055	S0_C1	D	4592,4632,4798
5056	S1_C0	D	4593,4637,4782
5057	S1_C1	D	4594,4640,4806
6342	SAVED_COUNT	D	6113,6137,6146,6159
4153	SAVE_2_BYTES	P	4100,4116
65	SAVE_CTRL	A	1051,1309
6341	SAVE_TEMP	D	
4079	SCRAM	P	4074
72	SEFFECT	A	

.LINE# SYMBOL TYPE REFERENCES

4508	SEG_0	A	4771,4795
4509	SEG_1	A	4779,4803
1750	SEMI_BOT	P	1748
1755	SEMI_EXIT	P	1711,1753
1759	SEMI_GRI	P	1715
1745	SEMI_MID	P	1743
452	SET	P	446
6102	SET_COUNT	P	6097,6229
6124	SET_COUNT10	P	6118
6150	SET_COUNT20	P	6122,6129
6160	SET_COUNTX	P	
4145	SET_DONE_BIT	P	4103,4128,4136
2035	SET_UP_ENDIF	P	2024
1991	SET_UP_WRITE	P	2186
2824	SHAPE	A	
3575	SHFEX	P	3571
3570	SHFPL	P	3574
6164	SHIFT_CT	P	6125
4475	SIGNAL_FALSE	P	4465,4472
4470	SIGNAL_MATCH	P	4461
594	SIGNAL_NUM	D	4202,4204
4480	SIGNAL_TRUE	P	4474
4484	SIGNAL_TRUE1	P	4482
7199	SKIPZZ	P	7197
2317	SKIP_OLD	P	2293
1825	SM_BY_OLD	P	1814,1820
1138	SND_MANAGER	P	1135,7395
76	SONGNO	A	
7393	SOUND_INIT	A	215
7373	SOUND_INITP	A	220
7395	SOUND_MAN	A	218
69	SOUND_PORT	A	795,810,821,1055,1057,1059,1061,1303,1324
7108	SOURCE	S	
5390	SPACE	P	5688
6503	SPIN	A	4834,4838,4839
6514	SPIN_OLD	A	
4515	SPIN_STATE	A	
5051	SPIN_SWO_CT	D	4589,4655,4710,4775
5052	SPIN_SWI_CT	D	4590,4799
6524	SPMR_MASK	A	
5334	SPRITEGENTBL	D	6327
5333	SPRITEINDEX	D	6326
2807	SPRITE_INDEX	A	3112,3129,3147
275	SPRITE_ORDER	A	274,6254,6283
2798	SPRITE_PTR	S	
997	SR1ATN	A	1054,1278,1279
996	SR1FRQ	A	1280
999	SR2ATN	A	1056,1284,1285
998	SR2FRQ	A	1286
1001	SR3ATN	A	1058,1290,1291
1000	SR3FRQ	A	1292
1004	SRMATN	A	1060,1296,1297
1003	SRMCTL	A	1314
578	STACK	D	233,371
290	START_GAME	A	289,541,5266
2806	STATUS	A	2865,2866,2924,2925,2953,2954,3012,3013,3034,3035,3053,3054,3070,3071,3090,3091
532	STRB_RST_PORT	A	4571,4641,4737

LINE#	SYMBOL	TYPE	REFERENCES
4533	STRB SET PORT	A	4633, 4734
5053	SIROBE_FLG	D	
4534	SIROBE RESET	A	
4537	SIROBE SET	A	4703
4266	SUBTRACT 4	P	4259
1831	SLP_GEN_CLR	P	1744, 1749, 1754
1894	SLP_UPDATE	P	1746, 1751
2347	SV1	P	
2393	SV2	P	
573	SYSTEM_RAM_AREA	D	578
2260	S_OLD_SCRN	P	2255
1973	TAIL_ADDRESS	D	2071, 2101, 2130, 2142
3981	TALO	P	3978
588	TEMP1	D	4178
591	TEMP2	D	4179
4463	TEST1	P	4468
4489	TEST_EXIT	P	4478
7383	TEST_SIGNAL	A	226
7368	TEST_SIGNALP	A	231
4449	TEST_SIGNALO	P	4045, 7368
4455	TEST_SIGNAL	P	4044, 7383
601	TEST_SIG_INUM	D	4451, 4453
4446	TEST_SIG_PARAM	D	4450
2802	THIS_SPRITE	S	
4343	TIMERT	P	4413, 4428
4399	TIMER2	P	
5724	TIMER_1	P	5289, 5732
5725	TIMER_2	P	5728
4147	TIMER_EXIT	P	4132, 4157
598	TIMER_LENGTH	D	4334
4495	TIMER_TABLE_BAS	D	4030, 4069, 4181, 4208, 4243, 4340, 4457
7384	TIME_MGR	A	227
4067	TIME_MGRQ	P	4047
4068	TIME_MGR	P	4046, 7384
1319	TONE_OUT	P	1255, 1282, 1288, 1294
5322	TRADEMARK	P	5190
7254	TRANSP_10	P	7273
7274	TRANSP_X	P	
527	TRUE	A	6818, 6856, 6904, 6954, 7026
7385	TURN_OFF_SOUND	A	216, 548
1432	TYPE0	P	1342
1453	TYPE1	P	1342
1472	TYPE2	P	1342
1501	TYPE3	P	1342, 1469
1625	TYPE4	P	1623
783	UPATMCTRL	P	781, 1306, 1315, 1327
4665	UPDATE_R0	P	4660
4679	UPDATE_R1	P	4674
4668	UPDATE_S1	P	4657, 4663
7359	UPDATE_SPINNER	A	211
4653	UPDATE_SPINNER	P	4652, 7359
4681	UPDATE_SPINX	P	4670, 4677
806	UPFREQ	P	805, 1328
1166	UP_CN_DATA_PIRS	P	1121, 1164, 1241
608	VDP_MODE_WORD	D	235, 1713, 3216, 6015, 6120, 6494, 6502, 7021
623	VDP_STATUS_WTIE	D	234
6332	VRAM_ADDR_TABLE	D	6009, 6114, 6286, 6325

LINE#	SYMBOL	TYPE	REFERENCES
6506	VRAM_READ	P	2287, 2316, 6098, 6585, 7389
6577	VRAM_READQ	P	6576, 7371
6572	VRAM_READ_P	P	6578
6529	VRAM_WRITE	P	1824, 2417, 6230, 6528, 7388
6520	VRAM_WRITEQ	P	6519, 7370
6515	VRAM_WRITE_P	P	6521
1270	WRITE	A	
280	WORK_BUFFER	A	279, 1873, 2280, 2361, 2401, 2859, 2947, 3215, 3256, 3268, 3271, 3291, 3295, 3300, 3329, 3347, 3373, 3382, 3440, 3451, 3467, 3471, 3474, 3545, 3546, 3599, 3633, 3649, 3656, 3657, 3660, 3703, 3723, 3737, 3761, 3774, 3784, 3785, 6775, 6805, 6843, 6863, 6891, 6929, 6946, 6981, 6975, 7047, 7064, 7079
7391	WRITER	A	204
2083	WRITER	P	2082, 7391
5954	WRITE_CHAR	P	5044, 5048, 5852, 5856, 5940, 5951
5959	WRITE_L11	P	5868, 5871, 5874, 5877
5965	WRITE_L12	P	5809, 5892, 5895, 5898
5941	WRITE_L3	P	5810, 5813, 5816, 5819, 5822, 5825, 5828, 5831
5947	WRITE_L4	P	5834, 5859, 5860
5950	WRITE_L5	P	5837, 5862, 5883
5953	WRITE_L6	P	5840, 5865, 5886
5142	WRITE_LOOP	P	5165
5177	WRITE_NAMES	P	5169
7306	WRITE_REGISTER	A	199, 5216, 5263, 5293, 5638, 5642, 5674, 5788, 5912
7369	WRITE_REGISTERP	A	205
7388	WRITE_VRAM	A	201, 3337
7370	WRITE_VRAM_P	A	206
2335	WRITE_ELSE	P	2122
2145	WRITE_END_IF	P	2133
2149	WRITE_END_WHILE	P	2098
2094	WRITE_WHILE	P	2147
7379	WR_SPR_IMM_TBL	A	192
7364	WR_SPR_IMM_TBLP	A	197
6269	WR_SPR_IMM_TBLQ	P	5980, 7364
6275	WR_SPR_IMM_TBL	P	5979, 7379
6267	WR_SPR_P	P	6270
2828	X	A	2910, 2998, 3039, 3137
3200	XDISP	A	3239, 3568
3208	XP_BK	A	3243, 3260, 3330, 3681, 3768, 3788
2490	XP_MEG	P	2487
3206	XP_OS	A	3272, 3762, 3777
2498	X_IN_BOUNDS	P	
2818	X_LOCATION	A	2867, 2955, 3036
2827	Y	A	3075
3199	YDISP	A	3248, 3549, 3577, 3579
3207	YP_BK	A	3252, 3257, 3259, 3292, 3348, 3374, 3634, 3661, 3685, 3724, 3769, 3786, 3789
3205	YP_OS	A	3296, 3767, 3775, 3778
2819	Y_LOCATION	A	3072