

BALCHECKHR OVERVIEW

UPON RESET, AN UPGRADED VERSION OF BALCHECK (BALLYCHECK) IS EXECUTED AT 2000_H JUMPING TO BALCHECK AT 207C_H

BALCHECK INITIALIZES VARIOUS PARAMETERS AND THEN CHECKS FOR ANY OF 5 KEYS PRESSED DOWN IN THE COLUMN 0 KEYPAD. IF ONE OF THESE KEYS IS DOWN, A JUMP TO THE RESPECTIVE ROUTINE WILL OCCUR FOR EXECUTION. THE 5 JUMPS ARE LISTED BELOW

KEY	JUMP ADDRESS	ROUTINE / PROGRAM
8	NO JUMP	SETS BIT 7 OF B' (ALTERNATE Z80 REG) = FOUR 2K RAM CHECKSUM FLAG
C	314C	CUSTOM TROUBLESHOOTING READ LOOP (RDLP)
O	301A	CUSTOM TROUBLESHOOTING WRITE LOOP (WLP)
SX	33BC	SETSCREENHR (HIGH-RES)
S3	3202	SETSCREEN3 (LOW-RES)

IF NONE OF THESE KEYS IS DOWN AT RESET, BALCHECK THEN BEGINS EXECUTING VARIOUS TESTS. REFER TO THE BALCHECKHR USER MANUAL FOR DETAILS ON THESE TESTS.

IF THE FIRST RAM TEST PASSES, THEN BALCHECK WILL ATTEMPT TO SET UP SCREEN INTERRUPTS, SO THAT BALCHECK'S SCREEN INTERRUPT SUBROUTINE CAN BE EXECUTED DURING EACH INTERRUPT REQUEST.

THE INTERRUPT VECTOR IS AT 2014_H.

THE SCREEN INTERRUPT SUBROUTINE IS AT 30DA_H

THIS INTERRUPT ROUTINE UPDATES BALCHECK'S HOURS:MINUTES:SECS CLOCK AND ALSO SCANS THE KEYPAD TO EXECUTE AN OPTIONAL PROGRAM REQUEST BY THE USER.

THE LISTING BELOW INDICATES THE OPTIONAL ROUTINES THAT WILL BE EXECUTED WHEN THE RESPECTIVE KEY IS PRESSED DOWN.

KEY		JUMP ADDRESS	PROGRAM
1	BALCHECK OPTION	25D9 ^{DOG}	MEMORY READ LOOP
2		25EA	MEMORY WRITE LOOP
3		2611	INPUT PORT READ LOOP
4		2627	OUTPUT PORT WRITE LOOP
5		264F	MEMORY WRITE AND READ LOOP
6		2673	MONITOR DEVICE INPUT PORTS
7		3639	CRASH TEST (FISH DEMO)
9		378F	MCM DESIGN TITLE PAGE
A		3ED0 ^{DOG}	NEW CHECKMATE DEMO
B		3A58	TEST DUAL-DISPLAY, DISPLAY PATTERNS
D		2E13	TEST USER RAM 6000-7FFF
E		2E1E	8000-FFFF
TUR (L)		2800	WITH USER SET UP
	INTERNAL LABEL		

HIGH RESOLUTION

F	INTERNAL LABEL	241E	TEST HI-RES SCREEN RAM
HD1 (H)		2E34	HI-RES DEMO 1 MOVE CRITTER, FISH AND WAGON
HD2 (G)		3AF7	2 TEXTURED 10COLOR TEST PATTERN
HD3 (J)		37F7	3 THE PIXEL STRINGER
HD4 (K)		3155	4 LOW-RES ROM ON HI-RES SCREEN MAP

BALCHECK WILL ATTEMPT TO EXECUTE 4 PASSES OF TESTS OR RUN NONSTOP, IF THE USER PRESSES THE Q/S KEY. AN ERROR DETECTED WILL BE OUTPUT TO THE DUAL 7-SEGMENT DISPLAY USING AN ERROR REPORT SEQUENCE.

PAGE C

USING BALCHECKHR ON A "MODIFIED FOR HI-RES" BALLY/ASTROCADE

BALCHECKHR INCLUDES 3 DEMOS THAT RUN IN THE HIGH RESOLUTION MODE.
THE ORIGINAL "TEST HI-RES" DEMO
TEXTURED 10 COLOR TEST PATTERNS
THE PIXEL STRINGER

INCLUDED ALSO IS A HI-RES SCREEN RAM TEST, SIMILAR TO THE COMMERCIAL BALCHECK RAM TEST. SETSCREENHR IS A HI-RES VERSION OF SETSCREEN3.

BY CHANGING ONLY 14 BYTES IN THE BALCHECKHR CODING, THE ENTIRE 8KB BALCHECKHR WILL EXECUTE ON A HI-RES SCREEN MAP. LOW-RES DEMOS/ROUTINES WILL ONLY DISPLAY ON THE TOP 1/4 OF THE HI-RES SCREEN RAM AREA. BECAUSE OF THE WAY THAT THE HI-RES RAM IS MAPPED OUT ON THE SCREEN, THE LOW-RES GRAPHICS WILL APPEAR CHOPPED UP WITH 50% OF THE GRAPHICS APPEARING ON THE LEFT HALF AND THE OTHER 50% APPEARING ON THE RIGHT HALF OF THE SCREEN.

THE FOLLOWING CODE CHANGES ARE REQ'D TO RUN BALCHECKHR 100% IN HI-RES.

OLD CODE (LOW AND HI-RES)		NEW CODE (100% HI-RES)	
2000H	C3 7C 20	2000H	C3 35 2D JPA2D35H
2D35H	FF FF FF	2D35H	3E 01 LDA, 01H
2D38H	FF FF	2D37H	C3 DC 39 JPA, 39DCH
39DCH	FF FF FF	39DCH	D3 08 OUT(CONSM), A
39DFH	FF FF	39DEH	C3 7C 20 JP BCHK
2368H	3E 00 LDA, 00	2369H	01
236AH	D3 08 OUT(CONCM), A		

TO SET CUSTOM ADDRESS/DATA CHIPS IN HI-RES MODE, OUTPUT 01 TO PORT 8
↓ ↓
IN LOW-RES MODE, OUTPUT 00 TO PORT 8

14 BYTE CHANGE TO RUN 100% IN HI-RES OR RESTORE TO LOW/HI-RES OPERATION

BALCHECK HR

2000H C3 7C 20 JP BCHK JMP TO START PROGRAM

MULTPLY A BY 16_D; SHIFT A LEFT 4 PLACES

A2003	2003H	87	ADD A,A	x2	USED BY GET 2/4 HEX DIGIT. FROM KEYPAD @ 2571H
		87	ADD A,A	x4	
		87	ADD A,A	x8	
		87	ADD A,A	x16	
		C9	RET		

KEY MASK TABLE

USED BY HEX NIBBLE TO ASCII BYTE
ROUTINE @ 2546H

T2008	2008H	2F	0010 1111	← COL 3	PORT 17
		0F	0000 1111	2	16
		0F	0000 1111	1	15
		2F	0010 1111	0	14

BIT 5 ↗ ↖ BIT 0

EXPANDER TEST PATTERNS

200CH	00	USED IN EXPANDER TEST ROUTINE @ 225AH
	55	
	AA	
	FF	

INTERRUPT VECTORS

2010	28 27	DISPLAY ALL INPUT DEVICES
2012	8D 39	HI-RES PIXEL STRINGER DEMO
2014	DA 30	NORMAL INTERRUPT ROUTINE

↖
006

3

DISPLAY DIGIT IN REG A

DLCHR OUTPUT MSD TO LEFT DISPLAY
 DLCHR1 OUTPUT LSD TO LEFT DISPLAY
 DRCHR OUTPUT LSD TO RIGHT DISPLAY

REG A ^{D11} 7 6 5 4 3 2 1 0
 MSD LSD
 DIGIT = 0-F_H

DISPLAY LEFT DIGIT

DLCHR	2026	21 16 20	LD HL, CTBL	POINT HL AT CHAR TABLE
		E6 F0	AND 1111 0000	A = MOST SIGNIFICANT DIGIT
		0F	RRCA	7 → 0 → CF ROTATE LEFT NIBBLE RIGHT TO BITS 3-0
		0F	RRCA	
		0F	RRCA	
		0F	RRCA	
DLCHR1	202F	85	ADD A, L	POINT HL AT CHAR DRIVER TO DISPLAY
	2030	6F	LD L, A	
		7E	LD A, (HL)	GET SEGMENTS DRIVER
		D3 EF	OUT (LDSPY), A	OUTPUT DIGIT TO LEFT DISPLAY
		C9	RET	

DISPLAY RIGHT DIGIT

DRCHR	2035	21 16 20	LD HL, CTBL
		E6 0F	AND 0000 1111 A = LEAST SIGNIFICANT DIGIT
		85	ADD A, L
		6F	LD L, A
		7E	LD A, (HL)
		D3 FF	OUT (RDSPY), A
	203F	C9	RET

DISPLAY 2 DIGIT INPUT

(CALL INSTRUCTION PERMITTED
REGISTER HL IS Clobbered)

ENTER WITH DIGIT TO DISPLAY IN REG H

DSPLYI	2040H	7C	LDA, H	PUT 2 DIGIT INPUT IN A REG
		08	EX AF, AF'	SAVE A FOR LSD
		7C	LDA, H	
		CD 26 20	CALL DLCHR	DISPLAY MSD
		08	EX AF, AF'	
		CD 35 20	CALL DRCHR	DISPLAY LSD
	204AH	C9	RET	

204BH	42 41 4E 4B	"BANK SIZE"
	20 53 49 5A	
2053H	45 00	

2055H	28 4B 20 42	"(K BYTES)"
	59 54 45 53	
205DH	29 00	

205FH	52 41 4D 20	"RAM TEST"
2063H	54 45 53 54	
2067H	00	

DISPLAY AND INCREMENT RAM TEST PASS COUNTER

ENTER WITH: E' = TEST COUNTER (00-99 MAX)

ICNT	2068H	D9	EXX	E' = TEST COUNTER
		83	LDA, E	
		D9	EXX	
		DD 21 72 20	LD IX, ICNT1	
		C3 DD 27	JP DSPLY	
ICNT1	2072H	D9	EXX	
		7B	LDA, E	A = TEST PASS COUNTER
	2074H	C6 01	ADD A, 1	INCREMENT CTR

5

2076H 27

5F

D9

DAA

LDE, A

EXX

ADJUST CARRY OVER

SAVE UPDATE FOR E

2079H C3 BB 28

JP TRAM

BAL CHECK
START PROGRAM

BCHK	207CH	00	NOP	
		F3	DI	DISABLE INTERRUPTS
		AF	XOR A	
	207FH	011808	LDBC, 0818H	STOP ALL SOUND
	2082H	ED79	OUT (C), A	
	DOG	10FC	DJNZ-2	
		D9	EXX	INITIALIZE
		47	LD B, A	B' = 00 FOUR ZK ROM CHECKSUM FLAG (BIT 7)
		0E81	LDC, 81H	C' = 81H Q/S FLAG, REP TESTS CTR (SEE NOTE 1)
		57	LD D, A	D' = 00 1/60 SEC CTR
		5F	LDE, A	E' = 00 DECIMAL SECS
		67	LD H, A	H' = 00 DECIMAL MINUTES
	DOG	6F	LD L, A	L' = 00 HOURS
		D9	EXX	
	208FH	3E40	LD A, 0100 0000 DASH -	SET DISPLAY
	2091H	D3EF	OUT (LDSPLY), A	TO " - - "
		D3FF	OUT (RDSPLY), A	
		3E46	LD A, 99x2	SET VERTICAL BANK REG
		DOG	OUT (VERBL), A	TO 99x2 = 198D
	2099H	DB14	IN A, (COLD)	READ COLUMN 0
		FE08	CP 8	IF KEY WLP IS DOWN,
		CA1A3D	JPZ, WLOOP	JMP TO WRITE LOOP (WLP)
	20A0H	0000	NOP'S	
		FE02	CP 2	
		2004	JP NZ, STPGM1	IF KEY 4R/15 DOWN, SET
		D9	EXX	FOUR ZK ROM CHECKSUM FLAG
		CBF8	SET 7, B	(BIT 7 IN B' = 1)
		D9	EXX	
STPGM1		FE04	CP 4	IF KEY RDLP IS DOWN,
	20ACH	CA4C31	JPZ, RDLP	JMP TO READ LOOP (RDLP)
	20AFH	FE10	CP 16	IF KEY SX IS DOWN,
	20B1H	CABC33	JPZ, ST5rnHR	JMP TO Set Screen HR
			OVER	

7

20B4 _H	FE 20	CP 32 _D	} IF KEY 53 is DOWN, JMP TO SetScreen 3
20B6 _H	CA 02 32	JPZ, SetScreen	

TEST 1 DELETE "CHECK KEYPAD FOR KEY STUCK DOWN"

TEST 2 TEST RAM AT 4F00-4FFF_H

CHECKS IF STACK AREA / UPI DATA LOCATIONS / SCRATCH PAD IS WORKING.

A20B9	20B9 _H	FD 21 C0 20	LD IY, RTABLE	POINT IY AT RAM TABLE
	20BD _H	C3 64 21	JR A2164	JMP TO MEMORY TEST ROUTINE

TABLE FOR RAM TEST ROUTINE ABOVE

RTABLE	20C0 _H	18 04	JR CHKSI	JMP TO SCREEN INTERRUPTS CHECK / SET UP ROUTINE
		4E	DEFB 78 _D	LOWER HIGH ORDER BYTE OF LIMIT (4F00-1)
		50	DEFB 80 _D	↓ UPPER LIMIT (4FFF+1)
	20C4 _H	00 4F	DEFW 4F00 _H	FIRST RAM BYTE TO TEST

AND FALL INTO TEST 3

CHECK SCREEN INTERRUPTS

TEST 3 CHECK / SETUP SCREEN INTERRUPTS

CHKSI	20C6	31CE4F	LD SP, 4FCE _H	SET STACK POINTER TO 4FCE _H (SEE NOTE)
	20C9	3E08	LD A, 8	- SET INTERRUPT MODE
	20CB	D30E	OUT (INM0D), A	
	20CD	211420	LD HL, 2014 _H	
	20D0	7C ^{DOG}	LD A, H	INITIALIZE INTERRUPT ROUTINE
	20D1	ED47	LDI, A	INTERERRUPT VECTOR IS AT 2014 _H (INTERUPT ROUTINE IS AT 30DA _H)
	20D3	7D ^{DOG}	LD A, L	
	20D4	D30D	OUT (INFBK), A	
	20D6	AF	XOR A	INTERUPT ON LINE G (EVERY 65ns)
	20D7	D30F ^{DOG}	OUT (INLIN), A	
	20D9	ED5E	IM2	Z80 TO INTERRUPT IN MODE 2
	20DB	FB ^{BOY}	EI	ENABLE INTERRUPT NOW
	20DC	1601	LD D, 1	PURPOSE ?
	20DE	CD CF 25 ^{DOG}	CALL A25CF	CALL DELAY WHICH ACTS LIKE A MAIN PROGRAM DELAYING FURTHER TESTS
	20E1	F3	DI	DISABLE INTERRUPTS

CHECK IF NORMAL INTERRUPT ROUTINE WAS EXECUTED

20E2	D9 ^{DOG}	EXX	D' = 1/60 SEC CTR, E' = SECS CTR
	7A	LDA, D	IF 1/60 SECS CTR = 0, CONTINUE TO CHECK SECS
	A7	AND A	
20E5	200E	JR NZ, CHKSI1	IF CTR ≠ 0, JMP TO CHKSI1
20E7	7B ^{BOY}	LDA, E	CTR WAS INCREMENTED. SCREEN INTERRUPT WAS GENERATED
	A7	AND A	A = E = SECS
20E9	200A	JR NZ, CHKSI1	IF SECS = 0, CONTINUE TO REPORT ERROR
20EB	D9 ^{DOG}	EXX	IF SECONDS ≠ 0, JMP TO CHKSI1
20EC	1E02	LDE, 2	SECS WAS INCREMENTED. SCREEN INTERRUPT WAS GENERATED
20EE	0E00	LDC, 00 _H	E = ERROR CODE 02 = SCREEN INT ERROR
20F0	215227	LD HL, A2752	(= INFO BYTE = 00)
20F3	1852	JR A2147	HL = RETURN ADR, DISPLAY ERROR (PASS 0 ONLY)
			WAS 23EF FOR COMMERCIAL BACKGK

9

NOTE: SINCE PREVIOUS RAM TEST CHECKED OUT OK, THE PROGRAM CAN NOW
UTILIZE A STACK FOR CALL INSTRUCTIONS, ETC.

CHKSI1 20F5 0E80

20F7 D9 ← B0Y

20F8 FB

LDS, 80H

EXX

EI

SET REP TEST CTR IN C¹, BIT 2-0, TO
ZERO WHICH WILL NOT ALLOW REPORTING
OF ERRORS IN PASS 0

ENABLE INTERRUPT

FALL INTO CHECKSUM CHECK ROUTINE

CHECK ROM CHECKSUMS

CHKSUM		DOG	BIT 7	0 8K ROM 1 FOUR 2K ROM	
20F9	D9	BOY	EXX B = ROM FLAG		
20FA	CB 78		BIT 7, B	IF BIT 7 OF B	
20FC	D9		EXX	IS SET,	
20FD	20 20		JR NZ, A211F	JMP TO	
20FF	21 FF1F		LD HL, 1FFF _H	FOUR 2K ROM	
2102	AF		XOR A	CHECKSUM TEST	
SUM	86	BOY	ADD A, (HL)	8K BYTES	
	2B		DEC HL	A = CHECKSUM	CHECKSUM
	57	DOG	LD D, A	SAVE CHKSUM	FOR
	7D	BOY	LD A, L	H=L=0?	8KB ROM
	B4		OR H		
	7A		LD A, D	A = CHKSUM	
2109	20 F8		JR NZ, SUM	LOOP BACK TO	
210B	FE A4		CP A4 _H	CONTINUE SUMMATION	
210D	28 04		JR Z, CHKSUM1	COMPARE CHKSUM WITH 1978 CHKSUM A4 _H .	
210F	FE FC		CP FC _H	IF A4, JMP TO TEST RAM	
2111	20 03		JR NZ, CHKSUM2	COMPARE CHKSUM WITH 1977 CHKSUM FC _H .	
CHKSUM1	2113	C3 4A 21	JP A214A	IF NOT FC EITHER, JMP TO REPORT ERROR.	
CHKSUM2	2116	1E 03	LDE, 03	JMP TO TEST RAM	
	2118	4A	LDC, D	E = ERROR CODE	
	2119	21 4A 21	LD HL, A214A	C = CHECKSUM	
	211C	C3 E6 23	JP A23E6	HL = ADR OF RAM TEST ROUTINE	
				(FOR PASS0, NOT REPORTING ERRORS)	
				JMP TO ERROR ROUTINE	

211F 010003
 2122 210000
 2125 AF
 2126 86
 2127 57
 2128 23
 2129 0D
 212A 20FA
 212C 05
 212D 20F7
 212F 7A
 2130 FEFF
 2132 200A
 2134 010003
 2137 7C
 2138 FE 20
 213A 23 0E
 213C 18E7
 213E 1E03
 2140 7C
 2141 D608
 2143 4F
 2144 214A21
 2147 C3 E6 23
 214A F3
 214B FD215121
 214F 1813
 2151 1804
 2153 4E
 2154 50
 2155 004F
 2157 FB
 2158 FD215E21
 215C 1806
 215E 1868
 2160 3F
 2161 4F
 2162 0040
 2164 0E00
 2166 0601
 2168 FD6605
 216B FD6E04
 216E 70
 216F 7E
 2170 A8

0847 : Start repetitive tests
 0848 : Checksum check ROMs
 0849 A 211F LD BC, 2048
 0850 LD HL, SCREEN
 0851 A 2125 XOR A
 0852 A 2126 ADD A, (HL)
 0853 LD D, A
 0854 INC HL
 0855 DEC C
 0856 JR NZ, A2126
 0857 DEC B
 0858 JR NZ, A2126
 0859 LD A, D
 0860 CP 0FFH
 0861 JR NZ, 213E
 0862 LD BC, 800H
 0863 LD A, H
 0864 CP [FIRSTC]
 0865 JR Z, A214A
 0866 JR A2125
 0867 : REPORT CHECKSUM ERROR
 0868 A 213E LD E, 3
 0869 LD A, H
 0870 SUB 8
 0871 LD C, A
 0872
 0873
 0874 A2144 LD HL, A214A
 0875 A2147 JP A23E6
 0876 :
 0877 : Test RAM
 0878 A214A DI
 0879 LD IV, T2151
 0880 JR A2164
 0881 :
 0882 T2151 JR A2157
 0883 DB 78
 0884 DB 80
 0885 DW NORMEM+3840
 0886 :
 0887 A2157 EI
 0888 LD IV, T215E
 0889 JR A2164
 0890 :
 0891 T215E JR A21C8
 0892 DB 63
 0893 DB 79
 0894 DW NORMEM
 0895 :
 0896 : Memory test routine
 0897 A2164 LD C, 0
 0898 LD B, 1
 0899 A2168 LD H, (IV+5)
 0900 LD L, (IV+4)
 0901 A216E LD (HL), B
 0902 LD A, (HL)
 0903 XOR B

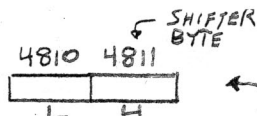
CHECK FOUR 2KB CHIPS
 0800H = 2048 BYTES
 C COUNTER = 00, B COUNTER = 08
 Start with low memory ADDRESS
 Initialize checksum TO ZERO
 Accumulate checksum IN A
 SAVE SUM IN D
 INCREMENT ROM ADDRESS
 LOOP BACK TO A2126 255 TIMES
 LOOP BACK TO A2126 7 TIMES
 PUT CHECKSUM IN A
 IS Checksum bad? IF NO, CONTINUE. IF YES, JUMP TO 213E
 NOT EQUAL TO FFH
 IF HIGH ORDER BYTE OF ROM MEMORY ADDRESS IS 20, THEN WE'RE Done with checksum test.
 Do next checksum (Loop BACK?)
 E = Error 03, ROM checksum error)
 C = ERROR DATA = 00 FOR 0000-07FF ERROR
 08 0800-0FFF
 10 1000-17FF
 18 1800-1FFF
 20 2000-27FF
 Where to continue testing (PASS ONLY)
 JUMP TO REPORT ERROR
 DISABLE INTERRUPTS
 Test 4F00 - 4FFF. IV POINTS TO DATA TABLE @ 2151. JUMP TO MEMORY TEST ROUTINE AT 2164
 JUMP TO TEST 4000-4EFF
 HIGH ORDER BYTE OF LOWER LIMIT (4F00-1)
 HIGH ORDER BYTE OF UPPER LIMIT (4FFF+1)
 FIRST BYTE OF RAM TO TEST
 4000H
 Test 4000 - 4EFF. IV POINTS TO DATA TABLE @ 215E. JUMP TO MEMORY TEST ROUTINE AT 2164.
 JUMP TO SHIFTER TEST
 HIGH ORDER BYTE OF LOWER LIMIT (4000-1)
 HIGH ORDER BYTE OF UPPER LIMIT (4F00-1)
 FIRST BYTE OF RAM TO TEST
 4000
 (REFERENCE FLOW CHART)
 C = ERROR DATA TO REPORT = 0
 B = 1ST TEST BYTE = 01 (BIT 0 IS SET)
 POINT HL AT RAM BYTE (START ADDRESS)
 LOAD RAM BYTE WITH TEST BYTE
 READ SAME RAM BYTE.
 INDICATE IN A WHICH BITS ARE IN ERROR.
 XOR BYTE WRITTEN WITH BYTE READ.
 THE XOR FUNCTION WILL REVEAL WHICH DATA BIT IS IN ERROR.

TABLE USED FOR RAM TEST ROUTINE @ 214B
 TABLE USED FOR RAM TEST ROUTINE @ 2158

12

2171	2806	0904	JR	Z, A2179	IF NO ERROR OCCURED, JUMP TO 2179 TO TEST NEXT BYTE. IF AN ERROR OCCURED, SET IX=2179 AND JUMP TO UPDATE ERROR DATA.
2173	DD217921	0905	LD	IX, A2179	
2177	184B	0906	JR	A21C4	
		0907 ;			JMP TO SAVE ERROR
2179	23	0908	A2179 INC	HL	INCREMENT RAM BYTE POINTER HL
217A	7C	0909	LD	A, H	IS HL POINTING TO UPPER LIMIT+1?
217B	FDBE03	0910	CP	(IX+3)	IF NO, LOOP BACK TO 216E TO TEST NEXT RAM BYTE
217E	20EE	0911	JR	NZ, A216E	IF YES, CONTINUE.
2180	2B	0912	A2180 DEC	HL	BACK UP RAM BYTE POINTER HL
2181	7C	0913	LD	A, H	IS HL POINTING TO LOWER LIMIT-1?
2182	FDBE02	0914	CP	(IX+2)	IF YES, JUMP TO 219D.
2185	2816	0915	JR	Z, A219D	IF NO, CONTINUE
2187	7E	0916	LD	A, (HL)	READ RAM BYTE
2188	A8	0917	XOR	B	INDICATE IN A WHICH BITS ARE IN ERROR.
2189	2806	0918	JR	Z, A2191	IF NO ERROR OCCURED, JUMP TO 2191 TO LOAD TB.
218B	DD219121	0919	LD	IX, A2191	IF AN ERROR OCCURED, SET IX=2191 AND
218F	1833	0920	JR	A21C4	JUMP TO UPDATE ERROR DATA.
		0921 ;			
2191	78	0922	A2191 LD	A, B	LOAD RAM BYTE WITH
2192	2F	0923	CPL		COMPLIMENT OF TEST
2193	77	0924	LD	(HL), A	BYTE (TB).
2194	AE	0925	XOR	(HL)	READ THE SAME RAM BYTE INDICATING IN A WHICH
2195	28E9	0926	JR	Z, A2180	BITS ARE IN ERROR.
2197	DD218021	0927	LD	IX, A2180	IF NO ERROR OCCURED, LOOP BACK TO 2180 TO TEST
2198	1827	0928	JR	A21C4	NEXT RAM BYTE.
		0929 ;			IF AN ERROR OCCURED, SET IX=2180 AND JUMP TO
					UPDATE ERROR DATA.
219D	23	0930	A219D INC	HL	INCREMENT RAM BYTE POINTER HL.
219E	7C	0931	LD	A, H	IS HL POINTING TO UPPER LIMIT+1?
219F	FDBE03	0932	CP	(IX+3)	IF YES, JUMP TO 21B3.
21A2	280F	0933	JR	Z, A21B3	IF NO, CONTINUE
21A4	78	0934	LD	A, B	READ RAM BYTE INDICATING IN A WHICH
21A5	2F	0935	CPL		BITS ARE IN ERROR.
21A6	AE	0936	XOR	(HL)	
21A7	2806	0937	JR	Z, A21AF	IF NO ERROR OCCURED, JUMP TO 21AF TO TEST
21A9	DD21AF21	0938	LD	IX, A21AF	NEXT BYTE.
21AD	1815	0939	JR	A21C4	IF AN ERROR OCCURED, SET IX=21AF AND
		0940 ;			JUMP TO UPDATE ERROR DATA.
21AF	AF	0941	A21AF XOR	A	A → A = 0
21B0	77	0942	LD	(HL), A	CLEAR RAM BYTE
21B1	18EA	0943	JR	A219D	LOOP BACK TO 219D
		0944 ;			
21B3	CB20	0945	A21B3 SLA	B	CHANGE TEST BYTE BY SHIFTING LEFT ARITHMETIC
21B5	30B1	0946	JR	NC, A2168	JUMP TO 2168 IF CARRY = 0. IF CARRY = 1, CONTINUE
21B7	79	0947	LD	A, C	
21B8	A7	0948	AND	A	IF ERROR DATA = 0, RAM IS OK: JUMP (IX)
21B9	2002	0949	JR	NZ, A21BD	2153 → 2154
21BB	FDE9	0950	JP	(IX)	IF ERROR DATA ≠ 0, RAM IS NOT OK: JUMP
		0951 ;			TO 21BD TO DISPLAY ERROR REPORT
21BD	1E04	0952	A21BD LD	E, 4	E ≠ Error 4 (RAM error)
21BF	21C821	0953	LD	HL, A21C8	HL POINTS AT "SHIFTER TEST" ROUTINE
21C2	184F	0954	JR	A2213	JUMP TO DISPLAY ERROR REPORT
		0955 ;			
21C4	B1	0956	A21C4 OR	C	OR OLD ERROR DATA WITH NEW ERROR DATA
21C5	4F	0957	LD	C, A	PUT UPDATED ERROR DATA IN C
21C6	DDE9	0958	JP	(IX)	UPDATE ERROR DATA
		0959 ;			
		0960 ;			Shifter test

↑
END OF
MEMORY TEST ROUTINE



13

SHIFTER TEST

21C8	FD211008	0961	A21C8	LD
21C0	DD211048	0962		LD
21D0	AF	0963		XOR
21D1	4F	0964		LD
21D2	0601	0965	A21D2	LD
21D4	79	0966	A21D4	LD
21D5	D30C	0967		OUT
21D7	AF	0968		XOR
21D8	67	0969		LD
21D9	6F	0970		LD
21DA	FD7700	0971		LD
21DD	78	0972		LD
21DE	68	0973		LD
21DF	FD7700	0974		LD
21E2	AF	0975		XOR
21E3	FD7701	0976		LD
21E6	79	0977		LD
21E7	A7	0978		AND
21E8	280B	0979		JR
21EA	CB1D	0980	A21EA	RR
21EC	CB1C	0981		RR
21EE	CB1D	0982		RR
21F0	CB1C	0983		RR
21F2	3D	0984		DEC
21F3	20F5	0985		JR
21F5	DD7E00	0986	A21F5	LD
21F8	BD	0987		CP
21F9	2013	0988		JR
21FB	DD7E01	0989		LD
21FE	BC	0990		CP
21FF	200D	0991		JR
2201	CB10	0992		RL
2203	30CF	0993		JR
2205	79	0994		LD
2206	3C	0995		INC
2207	4F	0996		LD
2208	FE04	0997		CP
220A	20C6	0998		JR
220C	1807	0999		JR
		1000		;
220E	1E05	1001	A220E	LD
2210	211522	1002		LD
2213	1833	1003	A2213	JR
		1004		;
		1005		;
		1006	A2215	LD
2215	3E40	1007		OUT
2217	D30C	1008		LD
2219	111048	1009		LD
221C	214A22	1010		LD
221F	0E01	1011	A2221	LD
2221	79	1012		LD
2222	FD7700	1013		LD
2225	1A	1014		CP
2226	BE	1015		JR
2227	201A	1016		INC
2229	23	1017		SLR
222A	CB21			

Flopper test

IV, 2064 → IY = MAGIC RAM ADDRESS = 0810
 IX, NORMEM+2064 → IX = SCREEN RAM ADDRESS = 4810

A
 C, A → C = Init shift AMOUNT = 00
 B, 1 → B = INITIAL PATTERN TO WRITE = 0000 0001

A, C
 (MAGIC), A → Set shifter TO MAGIC REGISTER
 A → ZERO A
 H, A → CLEAR Expected SECOND byte IN H
 L, A → CLEAR EXPECTED FIRST BYTE IN L
 (IY), A → CLEAR MAGIC RAM ADDRESS 0810
 A, B → A = PATTERN TO WRITE IN B
 L, B → L = PATTERN OF Expected first byte IN B
 (IY), A → WRITE PATTERN TO MAGIC RAM ADDRESS 0810.

A
 (IY+1), A → CLEAR THE SHIFTER BYTE TO ACCEPT SHIFT

A, C
 A → IS SHIFT AMOUNT = 0?
 Z, A21F5 → JF, if shift = 0, CONTINUE IF SHIFT ≠ 0.
 L → TO 21F5

H
 L
 H → DECREMENT SHIFTER
 A → ADJUST EXPECTED BYTES L AND H
 NZ, A21EA → Shift proper number of times.

A, (IX) → GET resulting RAM data FROM 4810H
 L → IS THERE A SHIFTER ERROR IN 1ST BYTE
 NZ, A220E → JUMP TO 220E IF A Shifter error.

A, (IX+1) → GET Second resulting byte FROM 4811H
 H → IS THERE A SHIFTER ERROR IN 2ND BYTE.
 NZ, A220E → JUMP TO 220E IF A Shifter error.

B → Shift Pattern left 1 bit FOR NEXT TEST
 NC, A21D4 → Keep trying (TRY 7 MORE PATTERNS)

A, C → Done 8 Patterns.
 A → INCREMENT SHIFT AMOUNT.
 C, A → KEEP SHIFT AMOUNT IN C
 4 → IS SHIFT AMOUNT = 4?
 NZ, A21D2 → IF YES, WERE DONE. IF NO, LOOP BACK TO 21D2.
 A2215 → JUMP TO FLOPPER TEST.

C = SHIFT AMOUNT
 E, 5 → E = Error 5 Shifter error
 HL, A2215 → HL = ADDRESS OF FLOPPER TEST ROUTINE
 A2248 → JUMP TO DISPLAY ERROR REPORT

A, 01000000B → Set flopper
 (MAGIC), A → DE, NORMEM+2064 → DE = RAM ADDRESS 4810
 HL, T224A → HL = 224A = POINTS TO EXPECTED FLOPPER PATTERN
 C, 1 → Write WITH PATTERN C.
 A, C → C = 0000 0001 INITIALLY.
 (IY), A → PUT CONTENTS OF RAM ADDRESS 4810 IN A
 A, (DE) → JUMP TO 2243 IF A Bad Flop.
 (HL) → NZ, A2243 → POINT TO NEXT EXPECTED FLOPPER PATTERN.
 HL → C → CHANGE PATTERN C

10 00 00 00

00 00 00 10
 00 00 01 00
 00 00 10 00
 00 01 00 00
 00 10 00 00
 01 00 00 00
 10 00 00 00

222C 30F3	1018 JR	NC, A2221	→ Do all 8 bits
222E CB19	1019 RR	C	→ Shift back to Hi bit; C=1000 0000
2230 79	1020 A2230 LD	A, C	→ Write 0s
2231 2F	1021 CPL		C = 0111 1111 INITIALLY
2232 FD7700	1022 LD	(IV), A	
2235 1A	1023 LD	A, (DE)	→ PUT CONTENTS OF RAM ADDRESS 4810 IN A
2236 BE	1024 CP	(HL)	
2237 2007	1025 JR	NZ, A2240	→ JUMP TO 2240 IF A Bad flopp
2239 23	1026 INC	HL	→ POINT TO NEXT EXPECTED FLOPPER PATTERN
223A CB39	1027 SRL	C	
223C 30F2	1028 JR	NC, A2230	→ Do all 8
223E 181A	1029 JR	A225A	→ JUMP TO EXPANDER TEST
	1030 ;		
2240 79	1031 A2240 LD	A, C	
2241 2F	1032 CPL		C = PATTERN WRITTEN TO MAGIC RAM
2242 4F	1033 LD	C, A	
2243 1E07	1034 A2243 LD	E, 7	→ E = Error 7 (Flopper error)
2245 215A22	1035 LD	HL, A225A	→ HL = ADDRESS OF EXPANDER TEST ROUTINE
2248 1847	1036 A2248 JR	A2291	→ JUMP TO DISPLAY ERROR REPORT
	1037 ;		

1038 ; Expected flopper patterns

224A 40	1039 T224A DB	01000000B	PATTERN WRITTEN TO MAGIC RAM
224B 80	1040 DB	10000000B	00 00 00 01
224C 10	1041 DB	00010000B	00 00 00 10
224D 20	1042 DB	00100000B	00 00 01 00
224E 04	1043 DB	00000100B	00 00 10 00
224F 08	1044 DB	00001000B	00 01 00 00
2250 01	1045 DB	00000001B	00 10 00 00
2251 02	1046 DB	00000010B	01 00 00 00
2252 FD	1047 DB	11111101B	10 00 00 00
2253 FE	1048 DB	11111100B	01 11 11 11
2254 F7	1049 DB	11110111B	10 11 11 11
2255 FB	1050 DB	11110110B	11 01 11 11
2256 DF	1051 DB	11011111B	11 10 11 11
2257 EF	1052 DB	11011110B	11 11 01 11
2258 7F	1053 DB	01111111B	11 11 10 11
2259 BF	1054 DB	10111111B	11 11 11 01
	1055 ;		11 11 11 10

1056 ; Expander test

225A 3E08	1057 A225A LD	A, 8	
225C D30C	1058 OUT	(MAGIC), A	→ Set Expander
225E AF	1059 XOR	A	
225F 4F	1060 LD	C, A	CLEAR C
2260 57	1061 LD	D, A	CLEAR D = EXPECTED EXPANDED RESULT
2261 210C20	1062 LD	HL, T200C	→ HL = ADDRESS OF EXPANDER TEST PATTERN TABLE
2264 D319	1063 A2264 OUT	(XPAND), A	→ 1 OR 0 IN DATA BUS WILL BE EXPANDED PER A.
2266 0604	1064 LD	B, 4	B = COUNTER = 4
2268 7E	1065 A2268 LD	A, (HL)	→ PUT PATTERN TO EXPAND IN A
2269 FD7700	1066 LD	(IV), A	→ Expand M. S. nybble (WRITE TO 0810)
226C FD7701	1067 LD	(IV+1), A	→ Expand L. S. nybble (WRITE TO 0811)
226F DD7E00	1068 LD	A, (IX)	→ PUT RESULTING DATA OF RAM ADDRESS 4810 INTO A.
2272 BA	1069 CP	D	
2273 2017	1070 JR	NZ, A228C	→ IF RESULT WASN'T ZERO, JUMP TO 228C; A Bad expand occurred.
2275 DD7E01	1071 LD	A, (IX+1)	→ PUT RESULTING DATA OF RAM ADDRESS 4811 INTO A.
2278 BA	1072 CP	D	
2279 2011	1073 JR	NZ, A228C	→ IF RESULT WASN'T ZERO, JUMP TO 228C; A Bad expand occurred.
227B 23	1074 INC	HL	→ POINT TO NEXT PATTERN TO EXPAND.

227C 10ER	1075	DJNZ	A2268	→ GO BACK AND EXPAND 3 MORE PATTERNS	EXPAND 3 MORE TIMES: DATA TO SEND TO OUTPUT PORT 19H (5 05, 0A AND OF.	
227E 7A	1076	LD	A,D	→ PUT EXPECTED EXPANDED RESULT IN A		
227F C655	1077	ADD	A,01010101B	→ A = NEXT EXPECTED EXPANDED RESULT		
2281 57	1078	LD	D,A	→ SAVE NEXT EXPECTED EXPANDED RESULT IN D		
2282 79	1079	LD	A,C	A = NEW VALUE TO SEND TO EXPAND REGISTER		
2283 C605	1080	ADD	A,0101B			
2285 4F	1081	LD	C,A		PUT NEW VALUE TO SEND TO EXPAND REGISTER IN C.	
2286 FE14	1082	CP	14H			
2288 20DA	1083	JR	NZ,A2264			
228A 1807	1084	JR	A2293			
	1085 ;			C = DATA SEND TO EXPAND REGISTER AT TIME OF ERROR (00,05,0A OR 0F).		
228C 1E12	1086 A228C	LD	E,12H	→ E = Error 12 (Expander error)		
228E 219322	1087	LD	HL,A2293	→ HL = ADDRESS OF OR/XOR WRITE ROUTINE		
2291 1838	1088 A2291	JR	A22CB	→ JUMP TO DISPLAY ERROR REPORT.		
	1089 ;					
	1090 ;	OR/XOR write, OR/XOR Intercept test				
2293 3E10	1091 A2293	LD	A,10H	→ Set for OR write		
2295 CDCE22	1092	CALL	A22CE			
2298 CDE322	1093	CALL	A22E3			
229B FE01	1094	CP	1	→ IF A = 1, THERE IS AN OR write error; JUMP TO 22BB		
229D 281C	1095	JR	Z,A22BB			
229F FE02	1096	CP	2			
22A1 2814	1097	JR	Z,A22B7	; OR intercept error		
22A3 3E20	1098 T22A3	LD	A,20H	; Set for XOR write		
22A5 CDCE22	1099	CALL	A22CE			
22A8 2C	1100	INC	L	→ L INDICATES AN XOR WRITE		
22A9 CDE322	1101	CALL	A22E3			
22AC FE01	1102	CP	1			
22AE 2812	1103	JR	Z,A22C2	; XOR Write error		
22B0 FE02	1104	CP	2			
22B2 2812	1105	JR	Z,A22C6	; XOR intercept error		
22B4 C33A23	1106	JP	A233A			
	1107 ;					
22B7 1E10	1108 A22B7	LD	E,10H	→ E = Error 10 (OR intercept error)		
22B9 1802	1109	JR	A22BD	→ JUMP TO 22BD		
	1110 ;					
22BB 1E08	1111 A22BB	LD	E,8	→ E = Error 8 OR (Write error)		
22BD 21A322	1112 A22BD	LD	HL,T22A3	→ HL = ADDRESS OF XOR WRITE TEST		
22C0 1809	1113	JR	A22CB	→ JUMP TO 22CB		
	1114 ;					
22C2 1E09	1115 A22C2	LD	E,9	→ E = Error 9 XOR (Write error)		
22C4 1802	1116	JR	A22C8	→ JUMP TO 22C8		
	1117 ;					
22C6 1E11	1118 A22C6	LD	E,11H	→ E = Error 11 XOR intercept err.		
22C8 213A23	1119 A22C8	LD	HL,A233A	→ HL = ADDRESS OF TRIGGER/JOYSTICK TEST		
22CB C3E623	1120 A22CB	JP	A23E6	→ JUMP TO DISPLAY ERROR REPORT		
	1121 ;					
22CE D30C	1122 A22CE	OUT	(MAGIC),A	→ SET MAGIC REGISTER		
22D0 DB08	1123	IN	A,(INTST)	→ Clear intercept status (INPUT PORT 08)		
22D2 210401	1124	LD	HL,104H	SET PATTERN D TO ZERO		
22D5 AF	1125	XOR	A	E = EXPECTED INTERCEPT STATUS = 00		
22D6 57	1126	LD	D,A	SET COUNTER B TO 04		
22D7 5F	1127	LD	E,A	SET PATTERN C TO 00		
22D8 4F	1128	LD	C,A	H = PATTERN CHANGER = 01		
22D9 6F	1129	LD	L,A	L = OR/XOR INDICATOR = 00		
22DA 0604	1130	LD	B,4	A = 00		
22DC C9	1131	RET				

16

220D 3E01	1132 ;		
22DF C9	1133 A220D LD	A.1 A=01 = INDICATES Write error	
	1134 RET		
	1135 ;		
22E0 3E02	1136 A22E0 LD	A.2 A=02 = INDICATES Intercept error	
22E2 C9	1137 RET		
	1138 ;		
22E3 CD3323	1139 A22E3 CALL	A2333	WRITE PATTERN D TO 4810
22E6 CD3323	1140 CALL	A2333	WRITE PATTERN C TO 0810
22E9 CD3323	1141 CALL	A2333	REPEAT 2 MORE TIMES
22EC AF	1142 XOR	A → ZERO A	
22ED BD	1143 CP	L	IF L=0, AN OR WRITE WAS PERFORMED, SO
22EE 2004	1144 JR	NZ, A22F4	CONTINUE. IF L≠0, AN XOR WRITE WAS
22F0 7R	1145 LD	A, D	PERFORMED, SO JUMP TO 22F4
22F1 B1	1146 OR	C	A = EXPECTED OR D. PATTERN
22F2 1802	1147 JR	A22F6 →	JUMP TO 22F6
	1148 ;		
22F4 7R	1149 A22F4 LD	A, D	A = EXPECTED XOR D PATTERN
22F5 A9	1150 XOR	C	
22F6 DBE000	1151 A22F6 CP	(IX)	COMPARE RESULTANT PATTERN WITH EXPECTED
22F9 20E2	1152 JR	NZ, A22DD	PATTERN. JUMP TO 22DD IF NOT THE SAME;
22FB 79	1153 LD	A, C	THERE IS A Write error. IF SAME, CONTINUE.
22FC A7	1154 AND	A	IF PATTERN C WRITTEN WAS 00, INTERCEPT=00;
22FD 2807	1155 JR	Z, A2306	JUMP TO 2306.
22FF DB08	1156 IN	A, (INTST)	→ Get intercept status
2301 BB	1157 CP	E	IF STATUS = EXPECTED STATUS, CONTINUE.
2302 20DC	1158 JR	NZ, A22E0	IF STATUS ≠ EXPECTED STATUS, JUMP TO 22E0;
2304 1805	1159 JR	A2306	THERE IS AN Intercept error.
	1160 ;		
2306 DB08	1161 A2306 IN	A, (INTST)	→ PUT INTERCEPT STATUS IN A
2308 B9	1162 CP	C	IF INTERCEPT STATUS = 0, CONTINUE.
2309 20D5	1163 JR	NZ, A22E0	IF IT IS ≠ 0, JUMP TO 22E0; THERE
230B 79	1164 A230B LD	A, C	IS AN Intercept error.
230C 84	1165 ADD	A, H	
230D 4F	1166 LD	C, A	CHANGE PATTERN C
230E 10D3	1167 DJNZ	A22E3	
2310 010004	1168 LD	BC, 400H	→ COUNTER B=04 PATTERN C=00
2313 CB3B	1169 SRL	E	→ CHANGE EXPECTED INTERCEPT STATUS.
2315 CB24	1170 SLA	H	CHANGE PATTERN CHANGER
2317 CB24	1171 SLA	H	
2319 30C8	1172 JR	NC, A22E3	→ LOOP BACK TO 22E3 IF CARRY=0
231B CB14	1173 RL	H	→ H=01 CARRY=0
231D 7B	1174 LD	A, E	
231E A7	1175 AND	A	
231F 2004	1176 JR	NZ, A2325	
2321 1E80 1E88	1177 LD	E, 89H	
2323 1802	1178 JR	A2327	
	1179 ;		
2325 CB3B	1180 A2325 SRL	E	
2327 7R	1181 A2327 LD	A, D	CHANGE PATTERN D
2328 C655	1182 C=1 ON ADD	A, 55H	
232A 57	1183 A CARRY	D, A	
232B 010004	1184 OUT OF LD	BC, 400H	COUNTER B=0 PATTERN C=00
232E 1E88 1E88	1185 A REGISTER LD	E, 89H	→ E = EXPECTED INTERCEPT STATUS = 1000 1000
2330 30B1	1186 JR	NC, A22E3	→ IF CARRY ≠ 0, LOOP BACK TO 22E3.
2332 C9	1187 RET		
	1188 ;		

```

01010101
10101010
11111111

```

17

2333 DD7200
2336 FD7100
2339 C9

1189 A2333 LD (IX),D WRITE PATTERN D TO RAM ADDRESS 4810
1190 LD (IV),C WRITE PATTERN C TO MAGIC RAM ADDRESS 0810
1191 RET

1192 ;

1193 ; Trigger/Joystick test, POT TEST

233A 06FF
233C DB8E
233E DB8F
2340 0E10
2342 ED78
2344 A7
2345 C2C823
2348 0C
2349 79
234A FE14
234C 20F4

1194 A233A LD B,255 B = COUNTER = 255
1195 A233C IN A,(VERAF) → Clear feedback registers
1196 IN A,(HORAF)
1197 LD C,500 C = INPUT PORT 10 = JOY/TRIG 1
1198 A2342 IN A,(C) IF JOY/TRIG INPUT = 0, CONTINUE.
1199 AND A IF JOY/TRIG INPUT ≠ 0, JUMP TO 23C8 TO
1200 JP NZ,A23C8 DISPLAY Error 13.
1201 INC C → POINT TO NEXT JOY/TRIG
1202 LD A,C LOOP BACK TO TEST 3 REMAINING
1203 CP SW3+1 JOY/TRIG, THEN CONTINUE.
1204 JR NZ,A2342

1205 ;

234E DB14
2350 DB15
2352 DB16
2354 DB17
2356 0E1C
2358 ED78
235A FEFF
235C 2071
235E 0C
235F 79
2360 FE20
2362 20F4
2364 10D6

1206 T234E IN A,(KEY0) EXERCISE
1207 IN A,(KEY1) THEM ONLY?
1208 IN A,(KEY2)
1209 IN A,(KEY3)
1210 LD C,POT0 C = INPUT PORT 1C = POT0
1211 A2358 IN A,(C) IF POT VALUE = 255, CONTINUE.
1212 CP 0FFH IF POT VALUE ≠ 255, JUMP TO 23CF TO
1213 JR NZ,A23CF DISPLAY Error 14.
1214 INC C → POINT TO NEXT POT
1215 LD A,C LOOP BACK TO TEST 3 REMAINING
1216 CP POT3+1 POT'S, THEN CONTINUE.
1217 JR NZ,A2358
1218 DJNZ A233C → LOOP BACK ANOTHER 254 TIMES AND PERFORM
SAME JOY/TRIG AND POT TESTS. THEN CONTINUE

1219 ; Exercise output ports

2366 06FF
2368 3E00
236A D308
236C 78
236D D300
236F D304
2371 D301
2373 D305
2375 D302
2377 D306
2379 D303
237B D307
237D 3E14
237F D309
2381 3E00
2383 D30A
2385 50
2386 21DE23
2389 0608
238B 0E0B
238D EDB3
238F 21D623
2392 0608
2394 0E18
2396 EDB3

1220 LD B,255 B = COUNTER = 255
1221 A2368 LD A,B
1222 OUT (CONOM),A → Set consumer mode
1223 LD A,B A = VALUE OF COUNTER
1224 OUT (COL0R),A
1225 OUT (COL0L),A
1226 OUT (COL1R),A
1227 OUT (COL1L),A
1228 OUT (COL2R),A
1229 OUT (COL2L),A
1230 OUT (COL3R),A
1231 OUT (COL3L),A
1232 LD A,20
1233 OUT (HORCB),A
1234 LD A,99+2
1235 OUT (VERBL),A
1236 LD D,B → SAVE COUNTER
1237 LD HL,T23DE
1238 LD B,8
1239 LD C,COLBX
1240 OTIR
1241 LD HL,T23DE
1242 LD B,8
1243 LD C,SNDBX
1244 OTIR
1245

SET ALL 8 COLORS TO VALUE OF COUNTER

SET HORIZ COLOR BOUNDARY/ BKGRD COLOR PORT TO 14. SPLIT SCREEN. BACKGROUND COLOR = VALUE OF COUNTER.

VERTICAL BLANK REGISTER = 198

SET COLORS PER TABLE AT 2EDE. (USE Color multiple port 0B)

HL POINTS TO SOUND PORT VALUES TABLE. EXERCISE SOUND PORTS. USE Sound multiple port 18 AND ACCESS PORTS INDIVIDUALLY.

2398 7A
2399 00

LD A,D
NOP

ELIMINATE IRRITATING "BEEPING" SOUND. ADDRESS 2398H WAS 2398H 3EFF LD A,FFH

239A D317 1246
 239C D316 1247
 239E D314 1248
 23A0 7A 1249
 23A1 D310 1250
 23A3 D311 1251
 23A5 D312 1252
 23A7 D313 1253
 23A9 D315 1254
 23AB 42 1255
 23AC 10BA 1256
 23AE F3 1257
 23AF D9 1258
 23B0 79 1259
 23B1 CB7F 1260
 23B3 200A 1261
 23B5 0C 1262
 23B6 79 1263
 23B7 FE84 1264
 23B9 2000 1265
 23BB D9 1266
 23BC FB 1267
 23BD 18FE 1268
 23BF A7 1270
 23C0 2001 1271
 23C2 0C 1272
 23C3 D9 1273
 23C4 FB 1274
 23C5 C31B 1275
 23C8 1E13 1277
 23CA 214E23 1278
 23CD 1817 1279
 23CF 1E14 1281
 23D1 216823 1282
 23D4 1810 1283

1284 ;
 1285 ; Sound Port values

23D6 48 1286 T23D6 DB 48H
 23D7 44 1287 DB 44H
 23D8 34 1288 DB 34H
 23D9 00 1289 DB 0
 23DA FF 1290 DB 0FFH
 23DB FD 1291 DB 0FDH
 23DC F5 1292 DB 0F5H
 23DD F5 1293 DB 0F5H

1294 ;
 1295 ; Color Port values

23DE DB 1296 T23DE DB 0DBH
 23DF 92 1297 DB 92H
 23E0 49 1298 DB 49H
 23E1 00 1299 DB 0
 23E2 DE 1300 DB 0DEH
 23E3 92 1301 DB 92H
 23E4 49 1302 DB 49H

23E5 00

OUT (VOLN), A = 255 A = CTR VALUE
 OUT (VOLAB), A = 255
 OUT (UIBRA), A = 255
 LD A, D A = COUNTER VALUE
 OUT (TONMO), A = COUNTER VALUE
 OUT (TONEA), A =
 OUT (TONEB), A =
 OUT (TONEC), A =
 OUT (VOLC), A =

LD B, D → PUT COUNTER BACK IN B
 DJNZ A2368 → LOOP BACK ANOTHER 254 TIMES TO EXERCISE OUTPUT PORTS, THEN CONTINUE.
 DI → DISABLE INTERRUPT ROUTINE.

EXX → GET REPETATIVE TESTS LOOP COUNTER AND PUT IT IN A. SEE ABOVE NOTE.
 LD A, C
 BIT 7, A → IF BIT 7, Q/S FLAG, IS SET, THEN CONTINUE (4 PASSES ONLY).
 JR Z, A23BF → IF IT IS NOT SET (ZERO), JMP TO 23BFH (NON STOP TESTING).

INC C → Increment loop counter
 LD A, C
 CP 84H → IF LOOP COUNTER = 84 CONTINUE.
 JR NZ, A23C3 → IF LOOP COUNTER ≠ 84, JUMP TO 23C3 (Not done yet).
 → Done with tests, BUT ENABLE INTERRUPT ROUTINE.

EI → Done with tests, BUT ENABLE INTERRUPT ROUTINE.

JR A23BD → HAND (CONTINUOUS LOOP)

→ BIT 7 → REALLY BITS 2-0
 → IF C, 7=0 and 6-0 not 0, JUMP TO 23C3. NON STOP TESTING

→ INCREMENT LOOP COUNTER → IF IT IS ZERO TO KEEP THE TESTS GOING NON STOP
 → ENABLE INTERRUPT ROUTINE

→ Start tests over

CHKSUM → C = INPUT PORT NUMBER
 E, 13H → E = Error 13 (Trisder/Joystick Error)

HL, T234E → HL = ADDRESS OF POT TEST ROUTINE

A23E6 → JUMP TO DISPLAY ERROR REPORT.

→ C = INPUT PORT NUMBER
 E, 14H → E = Error 14 (Pot Error)

HL, A2368 → HL = ADDRESS OF OUTPUT PORT EXERCISE.

A23E6

PORT #

; 17H
 ; 16H
 ; 15H
 ; 14H
 ; 13H
 ; 12H
 ; 11H
 ; 10H

; 07
 ; 06
 ; 05
 ; 04
 ; 03
 ; 02
 ; 01

; 00

NOTE: C' IS INITIALIZED TO 8/H
 AT START OF PROGRAM.
 C' IS ALSO SET TO
 80H BY SCREEN
 INTERRUPT SET UP
 AT 20F5H

23E6_H C34827 JPERRORJMP TO DISPLAY ERROR ROUTINE
(LOW-RES SCREEN RAM)

SAVE RAM ERROR (USER OR HI-RES RAM)

ENTER WITH: A = ERROR TO SAVE (READ ERROR)

B = TEST BYTE TO WRITE

C = SINGLE BANK ERROR (USER RAM BITS IN ERROR)

HL = RAM ADDRESS WRITTEN/READ

IX = "CONTINUE" (RETURN) ADDRESS

D' = ERROR REPORT STATUS 00 = SINGLE BANK (STATIC RAM)
01 = HIGH-RES BANKS

C' = HIGH RESOLUTION BANK 1 ERROR

B' = BANK 2

L' = BANK 3

H' = BANK 4

ERROR 23E9_H 57

BOY → B1

4F

7A

DOG 08

→ D9

7A

23F0_H A7

D9

20 02

DDE9

7D ← DOG

E6 03

FE 03

20 06

DOG 08

BOY → D9

B4

2400_H 67

LD D, A

OR C

LD C, A

LDA, D

EX AF, AF'

EXX

LD A, D

AND A

EXX

JR NZ, ERROR 1

JP (FX) JMP TO RAM TEST

LD A, L

AND 3

CP 3

JR NZ, ERROR 2

EX AF, AF' A = READ ERROR

EXX

ORH

LD H, A

SAVE READ ERROR IN D

UPDATE "BITS IN ERROR"
FOR USER RAM BANK.
SAVE UPDATE IN C

SAVE READ ERROR IN A'

IS THIS ERROR IN HI-RES RAM?

IF YES, JMP TO ERROR 1
TO SAVE HI-RES ERRORIF NO, CONTINUE "TEST RAM"
ROUTINE AT (IX) (2801)LAST 2 BITS IN RAM ADDRESS
INDICATE HI-RES BANK
IN ERRORBITS 1 AND 0 = 3?
IF YES, CONTINUE THIS IS 4TH
BANK.
IF NO, JMP TO NEXT TESTSAVE BIT(S) IN ERROR
FOR H'

20

ERROR2	2401 _H	1818	JR ERROR5	JMP TO EXIT THIS ROUTINE
		FE02	CP 2	BITS 1 AND 0 = 2?
		2006	JR NZ, ERROR3	IF YES, CONTINUE. THIS IS 3RD BANK
		08	EX AF, AF'	IF NO, JMP TO NEXT TEST
		D9	EXX	
	BoY →	B5	OR L	SAVE BIT(S) IN ERROR FOR L'
		6F	LD L, A	
		180E	JR ERROR	JMP TO EXIT THIS ROUTINE
ERROR3		FE01	CP 1	BITS 1 AND 0 = 1?
		2006	JR NZ, ERROR4	IF YES, CONTINUE. THIS IS 2ND BANK.
	2411 _H	08	EX AF, AF'	IF NO, JMP TO NEXT TEST
		D9	EXX	
	BoY →	B0	OR B	SAVE BIT(S) IN ERROR FOR B'
		47	LD B, A	
		1804	JR ERROR5	JMP TO EXIT THIS ROUTINE
ERROR4		08	EX AF, AF'	
		D9	EXX	BIT(S) IN ERROR MUST BE IN 1ST BANK.
	BoY →	B1	OR C	SAVE BIT(S) IN ERROR IN C'
		4F	LD C, A	
ERROR5		D9	EXX	
	2414 _H	DDE9	JP (IX)	JMP TO CONTINUE "TEST RAM" AT TRAM3 @ 28D1 _H

TEST HIGH RESOLUTION VIDEO RAM

THRES	241E _H	3E01	LD A, 1	SET CUSTOM CHIPS TO OPERATE IN HI-RES MODE
		D308	OUT (08), A	
2422 _H	21DE 23	010B 08	LD HL, 23DE _H	HL = COLOR TABLE
		EDB3	LD BC, 080B _H	SET COLORS
		3ECC	OTIR	
		D30A	LD A, 204 _D	
		FD21 9C 29	OUT (VERBL), A	POINT IY AT TEST RAM TABLE 2
2432 _H	1601		LD IY, RAMTAB2	
2434 _H	C3B1 28		LD D, 1	SET ERROR STATUS TO HI-RES
			JP CHKRAM	CHECK THE VIDEO RAM

SETQS 2437_H D9
CB F9
D9

243B_H C3 52 27

243E_H FF FF FF

EXX
SET 7, C
EXX
JP A2752

UNUSED BYTES

SET Q/S FLAG IN C'
("RUN TIME" DISALLOWED ON DISPLAY)

21

22

BALCHECK PR

2441	454E5452 20342D44 49475420 48455800	1365 T2441	DB	'ENTR 4-DIGT HEX',0
2451	52454144 20414444 5200	1366 T2451	DB	'READ ADDR',0
245B	57524954 45204144 445200	1367 T245B	DB	'WRITE ADDR',0
2466	454E5452 20322D44 49475420 48455800	1368 T2466	DB	'ENTR 2-DIGT HEX',0
2476	52454144 20504F52 5400	1369 T2476	DB	'READ PORT',0
2480	57524954 4520504F 525400	1370 T2480	DB	'WRITE PORT',0
248B	53545254 20414444 5200	1371 T248B	DB	'STRT ADDR',0
2495	454E5452 20322D44 49475420 48455800	1372 T2495	DB	'ENTR 2-DIGT HEX',0
24A5	44415441 00	1373 T24A5	DB	'DATR',0
24AA	42595445 20544F20 57524954 4500	1374 T24AA	DB	'BYTE TO WRITE',0
24B3	2A2A00	1375 T24B3	DB	'**',0
24BB	2A2A2A2A 00	1376 T24BB	DB	'****',0
		1377 ;		
24C0	30453D	1378 T24C0	DB	'0E='
24C3	30463D	1379	DB	'0F='
24C6	202020	1380	DB	' '
24C9	202020	1381	DB	' '
24CC	31303D	1382	DB	'10='
24CF	31313D	1383	DB	'11='
24D2	31323D	1384	DB	'12='
24D5	31333D	1385	DB	'13='
24D8	31343D	1386 T24D8	DB	'14='
24DB	31353D	1387	DB	'15='
24DE	31363D	1388	DB	'16='
24E1	31373D	1389	DB	'17='
24E4	202020	1390	DB	' '
24E7	31433D	1391	DB	'1C='
24EA	31443D	1392	DB	'1D='
24ED	31453D	1393	DB	'1E='
24F0	31463D	1394	DB	'1F='
24F3	22474F22 20544F20 52554E00	1395 T24F3	DB	'GO" TO RUN',0

1396 ;
 1397 ; GENERAL USE SUBROUTINES
 1398 ;
 SET COLORS / CLEAR SCREEN

24FF 3E6A
 2501 D309
 FF
 19
 12 25
 2507 FF
 1B
 00 40
 B0 0F
 00
 250E C9

A24FF LDA, 6A
 OUT(HORCB), A
 RST 38H
 COLSET + 1
 COLOR TABLE
 A2507 RST 38H
 FILL + 1
 START ADR
 BYTES TO FILL
 FILL WITH
 RET

SET UP
 BACKGRD CLR
 HOR CLR BOUNDRY
 01 10 10 10
 PIXEL 42D
 01

SET
 COLORS

CLEAR
 SCREEN

UNUSED BYTES

CLR REG 7
 6
 5
 4
 PIXEL 11
 10
 01
 00
 YEL
 BLU
 RED
 BLK

2516
 2516 FF
 2517 35
 2518 04
 2519 28
 251A 0C
 251B 4124
 251D C9

1418 ;
 1419 A2516 SYSSUK STRDIS
 1420+ RST 38H
 1421+ DB STRDIS+1
 1425 DB 4
 1426 DB 40
 1427 DB 00001100B
 1428 DW T2441
 1429 RET

DISPLAY AT 4,40D
 'ENTR 4-DIGT HEX'

251E
 251E FF
 251F 35
 2520 04
 2521 32
 2522 0C
 2523 5124
 2525 C9

1430 ;
 1431 A251E SYSSUK STRDIS
 1432+ RST 38H
 1433+ DB STRDIS+1
 1437 DB 4
 1438 DB 50
 1439 DB 00001100B
 1440 DW T2451
 1441 RET

DISPLAY A
 'READ ADDR'

2526
 2526 FF
 2527 35
 2528 04
 2529 28
 252A 0C
 252B 6624
 252D C9

1442 ;
 1443 A2526 SYSSUK STRDIS
 1444+ RST 38H
 1445+ DB STRDIS+1
 1449 DB 4
 1450 DB 40
 1451 DB 00001100B
 1452 DW T2465
 1453 RET

DISPLAY AT 4,40D
 'ENTR 2-DIGT HEX'

252E
 252E FF
 252F 35
 2530 04
 2531 28
 2532 0C
 2533 9524
 2535
 2535 FF
 2536 35

1454 ;
 1455 A252E SYSSUK STRDIS
 1456+ RST 38H
 1457+ DB STRDIS+1
 1461 DB 4
 1462 DB 40
 1463 DB 00001100B
 1464 DW T2495
 1465 SYSSUK STRDIS
 1466+ RST 38H
 1467+ DB STRDIS+1

DISPLAY AT 4,40
 'ENTR 2-DIGT HEX'

24

2537 04	1471 DB 4	
2538 32	1472 DB 50	
2539 00	1473 DB 00001100B	DISPLAY AT 4,50
253A AA24	1474 DW T24AA	→ 'BYTE TO WRITE'
253C C9	1475 RET	
	1476 ;	
	1477 ; <u>Display character in A</u>	
253D 0E0C	1478 A253D LD C,00001100B	→ C = CHAR DISPLAY PARAMETERS
253F DD210D02	1479 A253F LD IX,FNTSML	→ Unnecessary instruction
2543	1480 SYSTEM CHRDIS	
2543 FF	1481+ RST 30H	
2544 32	1482+ DB CHRDIS	
2545 C9	1486 RET	
	1487 ;	
	1488 ; <u>Hex nybble to ASCII byte</u>	
2546 FE0A	1489 A2546 CP 10	
2548 FA4D25	1490 S=1 IF < 9 → JP M,A254D	; Could be JR C.
254B C607	1491 IF S=1 → ADD A,7	
254D C630	1492 A254D ADD A,'0'	
254F C9	1493 RET	
	1494 ;	
2550 D5	1495 A2550 PUSH DE	SAVE DE = Y, X COORDINATES OF LAST DIGIT
2551 E5	1496 PUSH HL	SAVE HL = 1-4 NIBBLE ENTRIES
2552 3EFF	1497 LD A,255	SET 4EEC TO FF SO SYSTEM DOESN'T SHUT DOWN
2554 32EC4F	1498 LD (TIMOUT),A	(4 min 15 sec timeout)
2557	1499 A2557 SYSSUK SENTRY	START SENTRY
2557 FF	1500+ RST 30H	
2558 43	1501+ DB SENTRY+1	
2559 0820	1505 DW T200B	← KEYBOARD MASK TABLE
255B FE13	1506 CP SKVD	
255D 20F8	1507 JR NZ,A2557	Wait for key to be pressed
255F 78	1508 LD A,B	PUT KEY STATUS IN A.
2560 FE10	1509 CP 160	
2562 2001	1510 JR NZ,A2565	IF "0" KEY WAS PRESSED, KEY CODE = ZERO
2564 AF	1511 S=0 IF ≥ 20 XOR A	
2565 FE14	1512 A2565 CP 20	JUMP IF KEY PRESSED WAS OF KEYS 20-24 (20,22,23 ARE MASKED)
2567 F26E25	1513 IF S=0 → JP P,A256E	
256A 47	1514 LD B,A	JUMP TO CONVERT KEYS 1-19 TO ASCII CODE (17-19 ARE MASKED)
256B CD4625	1515 CALL A2546	B = KEY CODE = 0-15
256E E1	1516 A256E POP HL	
256F D1	1517 POP DE	
2570 C9	1518 RET	
	1519 ;	
	1520 ; <u>Get 2 or 4 hex digits from keypad.</u>	
	1521 ; A=0=2 digits A=1=4 DIGITS	
2571 F5	1522 A2571 PUSH AF	SAVE # OF DIGITS
2572 A7	1523 AND A	IF A=0, CONTINUE (DISPLAY 4 ASTERISKS)
2573 2009	1524 JR NZ,A257E	
2575	1525 A2575 SYSSUK STRDIS	Display 2 asterisks AT 72,70D
2575 FF	1526+ RST 30H	
2576 35	1527+ DB STRDIS+1	
2577 48	1531 DB 72	
2578 46	1532 DB 70	
2579 00	1533 DB 00001100B	
257A B824	1534 DW T24B8	; '***'
257C 1807	1535 JR A2595	
	1536 ;	

CONVERT KEY CODE 0-F TO EQUIVALENT ASCII CODE.
A = ASCII CODE
= 30-39, 41-46
0 1 2 3 4 5 6 7 8 9 A B C D E F

GET ENTRY FROM KEYPAD

NOTE: YOU REVISED 256EH
256EH ← 31124 JPA2411

257E	FF	1537	R257E	SVSSUK	STRDIS	→ Display 4 asterisks AT 72,70D
257F	35	1538+		RST	39H	
2580	48	1539+		DB	STRDIS+1	
2581	46	1543		DB	72	
2582	8C	1544		DB	70	
2583	BB24	1545		DB	00001100B	
2585	210000	1546		DW	T24BB	; '****'
2588	CD5025	1547	R2585	LD	HL,0	→ HL=0000 B= KEY CODE 0-F
258B	FE18	1548		CALL	R2550	→ IF KEYS 0-F PRESSED, A= EQUIVALENT ASCII CODE
258D	283E	1549		CP	24	→ IF KEYS CE AND GO PRESSED, A=B= KEY STATUS (21,24D)
258F	FE15	1550		JR	Z,R25CD	→ IF "GO" KEY WAS PRESSED, JUMP TO 25XD
2591	28E2	1551		CP	21	
2593		1552		JR	Z,R2575	→ IF "CE" KEY WAS PRESSED, LOOP BACK TO 2575 TO GET A NEW DIGIT.
2593	114846	1553		XYRELL	DE,72,70	
2596	CD3D25	1554+		LD	DE,70 SHL 8+(72)	→ DE = Y,X COORDINATES OF DIGIT
2599	78	1555		CALL	R253D	→ DISPLAY FIRST DIGIT
259A	CD0320	1556		LD	A,B	→ PUT KEY CODE IN A
259D	67	1557		CALL	R2003	→ MUL A BY 16, ie, PUT CODE (0-15) IN MOST SIGNIFICANT NIBBLE OF A.
259E	CD5025	1558		LD	H,A	→ PUT First nibble IN H
25A1	FE15	1559		CALL	R2550	→ GET ANOTHER DIGIT FROM KEYPAD
25A3	28D0	1560		CP	21	
25A5	CD3D25	1561		JR	Z,R2575	→ IF "CE" KEY WAS PRESSED, LOOP BACK TO 2575 TO GET 2 NEW DIGITS.
25A8	7C	1562		CALL	R253D	→ DISPLAY SECOND DIGIT.
25A9	B0	1563		LD	A,H	
25AA	67	1564		OR	B	
25AB	F1	1565		LD	H,A	→ PUT Second nibble IN H
25AC	A7	1566		POP	AF	→ PUT # OF DIGITS BACK IN A
25AD	2001	1567		AND	A	→ IF A=0, YOUR DONE
25AF	C9	1568		JR	NZ,R25B0	→ IF A≠0, JUMP TO A25B0
		1569		RET		
		1570				
25B0	F5	1571	R25B0	PUSH	AF	→ SAVE NUMBER OF DIGITS
25B1	CD5025	1572		CALL	R2550	→ GET ENTRY FROM KEYBOARD
25B4	FE15	1573		CP	21	
25B6	2806	1574		JR	Z,R257E	→ IF "CE" KEY WAS PRESSED, LOOP BACK TO GET A NEW NUMBER.
25B8	CD3D25	1575		CALL	R253D	→ DISPLAY 3RD DIGIT
25B8	78	1576		LD	A,B	→ PUT KEY CODE IN A
25BC	CD0320	1577		CALL	R2003	→ MUL A BY 16, ie, PUT STATUS (0-15) IN MOST SIGNIFICANT NIBBLE OF A
25BF	6F	1578		LD	L,A	→ PUT THIRD NIBBLE IN L
25C0	CD5025	1579		CALL	R2550	→ GET ENTRY FROM KEYPAD.
25C3	FE15	1580		CP	21	
25C5	28B7	1581		JR	Z,R257E	→ IF "CE" KEY WAS PRESSED, LOOP BACK TO 257E TO GET A NEW NUMBER.
25C7	CD3D25	1582		CALL	R253D	→ DISPLAY 4TH DIGIT.
25CA	7D	1583		LD	A,L	
25CB	B0	1584		OR	B	→ PUT SECOND NYBBLE IN L
25CC	6F	1585		LD	L,A	
25CD	F1	1586	R25CD	POP	AF	→ RESTORE STACK.
25CE	C9	1587		RET		
		1588				
		1589				
25CF	3EFF	1590	R25CF	LD	A,255	
25D1	06FF	1591	R25D1	LD	B,255	
25D3	10FE	1592	R25D3	DJNZ	R25D3	
25D5	3D	1593		DEC	A	
25D6	28F9	1594		JR	NZ,R25D1	
25D8	C9	1595		RET		
		1596				

1570 ;

1588 ; ← 1/2 SEC?

1589 ; **Fixed delay routine**

THIS ROUTINE IS USED IN:

- ① TEST 3, CHECK SCREEN INTERRUPTS @ 20DEH.
- ② SPECIAL ROUTINES 2, 4 AND 5.
- ③ ENTER "TEST RAM" PARAMETERS @ 2818H, 2850H AND 2873H


```

1597 : Special routine 1      Memory Read Routine
2509 CDFF24 1598 R2509 CALL R24FF → Set colors / CLEAR SCREEN
250C CD1625 1599      CALL R2516 DISPLAY "ENTR 4-DIGT HEX"
250F CD1E25 1600      CALL R251E DISPLAY "READ ADDR"
25E2 3E01 1601      LD R,1 → Get 4 digits FROM KEYPAD
25E4 CD7125 1602      CALL R2571 HL = 4 DIGIT ENTRY
25E7 7E 1603 R25E7 ← LD R, (HL) READ FROM 4 DIGIT ADDR ENTERED
25E8 18FD 1604      JR R25E7 → Loop forever
1605 :
1606 : Special routine 2      Memory Write Routine
25EA CDFF24 1607 R25EA CALL R24FF → Set colors / CLEAR SCREEN
25ED CD2E25 1608      CALL R252E DISPLAY "ENTR 2-DIGT BYTE TO WRITE"
25F0 AF 1609      XOR A → A=0 → Get 2 disits
25F1 CD7125 1610      CALL R2571
25F4 E5 1611      PUSH HL → Save 2 DIGIT BYTE TO WRITE
25F5 CD 40 20 1612      CALL DSPLYI → Put disits on display
25F6 1613
25F8 CDCF25 1614      CALL R25CF → Delay
25FB CDFF24 1615      CALL R24FF → Clear screen, Set Colors
25FE CD1625 1616      CALL R251E DISPLAY "ENTR 4-DIGT HEX"
2601 1617      SYSSUK STRDIS
2601 FF 1618+      RST 39H
2602 35 1619+      DB STRDIS+1
2603 04 1623      DB 4
2604 32 1624      DB 50
2605 0C 1625      DB 00001100B
2606 5824 1626      DW T245B
2608 3E01 1627      LD R,1 → Get 4 disits ADDRESS FROM KEYPAD
260A CD7125 1628      CALL R2571 PUT IT IN HL
260D C1 1629      POP BC B = 2 DIGIT BYTE TO WRITE
260E 70 1630 R260E ← LD (HL), B WRITE INPUT BYTE TO INPUT ADDRESS
260F 18FD 1631      JR R260E ; Loop forever
1632 :
1633 : Special routine 3      Input Port Read Routine
2611 CDFF24 1634 R2611 CALL R24FF → Set colors / CLEAR SCREEN
2614 CD2625 1635      CALL R252E DISPLAY "ENTR 2-DIGT HEX"
2617 1636      SYSSUK STRDIS
2617 FF 1637+      RST 39H
2618 35 1638+      DB STRDIS+1
2619 04 1642      DB 4
261A 32 1643      DB 50
261B 0C 1644      DB 00001100B
261C 7624 1645      DW T2476
261E AF 1646      XOR A → GET 2 DIGIT INPUT FROM KEYPAD
261F CD7125 1647      CALL R2571 H = 2 DIGITS
2622 4C 1648      LD C, H
2623 ED78 1649 R2623 ← IN R, (C) INPUT FROM 2 DIGIT PORT
2625 18FC 1650      JR R2623 ; Loop forever
1651 :
1652 : Special Routine 4      Output Port Write Routine
2627 CDFF24 1653 R2627 CALL R24FF → Set colors / CLEAR SCREEN
262A CD2625 1654      CALL R252E DISPLAY "ENTR 2-DIGT HEX"
262D 1655      SYSSUK STRDIS
262D FF 1656+      RST 39H
262E 35 1657+      DB STRDIS+1
262F 04 1661      DB 4
2630 32 1662      DB 50

```

2631 00 1663 DB 00001100B → DISPLAY
 2632 8024 1664 DM T2480 → 'WRITE PORT' ← ie, OUTPUT PORT NUMBER
 2634 AF 1665 XOR R
 2635 CD7125 1666 CALL R2571 → ENTER 2 DIGITS FROM KEYPAD. PUT THEM IN REG H.
 2639 E5 1667 PUSH HL → SAVE OUTPUT PORT NUMBER
 2639 CD 40 20 1668 CALL DSPLYI → DISPLAY ENTRY ON TWO 7 SEGMENT LED'S
 263A 1669
 263C C0CF25 1670 CALL R25CF → WAIT A BIT
 263F C0FF24 1671 CALL R24FF → SET COLORS, CLEAR SCREEN
 2642 CD2E25 1672 CALL R252E → DISPLAY "ENTR 2-DIGT BYTE TO WRITE"
 2645 AF 1673 XOR R
 2646 CD7125 1674 CALL R2571 → ENTER 2 DIGITS. PUT "BYTE TO WRITE" IN REG H
 2649 C1 1675 POP BC → PUT OUTPUT PORT NUMBER IN REG B
 264A 48 1676 LD C,B
 264B ED61 1677 R264B OUT (C),H → OUTPUT 2ND ENTRY TO OUTPUT PORT OF 1ST ENTRY THE BYTE TO WRITE
 264D 18FC 1678 JR R264B → LOOP FOREVER
 1679 ;
 1680 ; Special Routine 5 Memory Read and Write Routine
 264F C0FF24 1681 R264F CALL R24FF → Set colors / CLEAR SCREEN
 2652 CD2E25 1682 CALL R252E → DISPLAY "ENTR 2-DIGT BYTE TO WRITE"
 2655 AF 1683 XOR R
 2656 CD7125 1684 CALL R2571 → Get 2 digits FROM KEYPAD. PUT THEM IN REG H
 2659 E5 1685 PUSH HL → Save "BYTE TO WRITE"
 265A CD 40 20 1686 CALL DSPLYI
 265B 1687 ; Send to display
 265D C0CF25 1688 CALL R25CF → Delay
 2660 C0FF24 1689 CALL R24FF → Clear screen
 2663 CD1625 1690 CALL R2516 → DISPLAY "ENTR 4-DIGT HEX"
 2666 CD1E25 1691 CALL R251E → DISPLAY "READ ADDR"
 2669 3E01 1692 LD R,1
 266B CD7125 1693 CALL R2571 → Get 4 digits ADDRESS FROM KEYPAD. PUT ADDRESS IN HL
 266E C1 1694 POP BC → B = BYTE TO WRITE
 266F 70 1695 R266F LD (HL),B → WRITE BYTE ENTERED AT ADDRESS ENTERED
 2670 7E 1696 LD R,(HL) → READ ADDRESS ENTERED
 2671 18FC 1697 JR R266F → Loop forever
 1698 ;
 1699 ; Special Routine 6 Display All Input Devices
 2673 C0FF24 1700 R2673 CALL R24FF → SET COLORS, CLEAR SCREEN
 2676 210024 1701 LD HL,T2400 → HL = Labels TABLE #1
 2679 1702 XYRELL DE,4,10
 2679 11040A 1703+ LD DE,10 SHL 8+(4) → DE = Y,X COORDINATE = 10,04
 267C C0FB26 1704 CALL R26FB → DISPLAY 1ST COLUMN OF 8 LABELS
 267F 21D824 1705 LD HL,T24D8 → HL = Labels TABLE #2 ADDRESS
 2682 1706 XYRELL DE,80,0
 2682 115000 1707+ LD DE,0 SHL 8+(80) → DE = Y,X COORDINATES = 00,00D
 2695 C0FB26 1708 CALL R26FB → DISPLAY 2ND COLUMN OF 9 LABELS.
 2698 00 1709 EX AF,AF'
 2699 AF 1710 XOR R
 269A 00 1711 EX AF,AF'
 269B 211020 1712 LD HL,T2010 → Set interrupt routine adrs. (VECTOR @ 2010)
 269E 7C 1713 LD R,H
 269F ED47 1714 LD I,R
 2691 7D 1715 LD R,L
 2692 D30D 1716 OUT (INFBK),R
 2694 3E03 1717 LD R,3
 2696 D30E 1718 OUT (INMD),R → SCREEN INTERRUPT MODE 0 LIGHT PEN INTERRUPT MODE 1
 2698 FB 1719 EI → ENABLE INTERRUPTS

BALCHECK ~~TR~~

```

2699 DB10      1720 A2699  IN      R, (SW0) → GET PLAYER 1 JOY/TRIG INPUT, PUT IT IN A
269B          1721      XYRELL  DE, 28, 50
269B 111032    1722+     LD      DE, 50 SHL 8+(28) → DE = Y, X COORDINATE
269E CD1127    1723      CALL    A2711 → DISPLAY INPUT
26A1 DB11      1724      IN      R, (SW1)
26A3          1725      XYRELL  DE, 28, 60
26A3 111030    1726+     LD      DE, 60 SHL 8+(28)
26A6 CD1127    1727      CALL    A2711
26A9 DB12      1728      IN      R, (SW2)
26AB          1729      XYRELL  DE, 28, 70
26AB 111046    1730+     LD      DE, 70 SHL 8+(28)
26AE CD1127    1731      CALL    A2711
26B1 DB13      1732      IN      R, (SW3)
26B3          1733      XYRELL  DE, 28, 80
26B3 111050    1734+     LD      DE, 80 SHL 8+(28)
26B6 CD1127    1735      CALL    A2711
26B9 DB14      1736      IN      R, (KEY0)
26BB          1737      XYRELL  DE, 104, 0
26BB 116800    1738+     LD      DE, 0 SHL 8+(104)
26BE CD1127    1739      CALL    A2711
26C1 DB15      1740      IN      R, (KEY1)
26C3          1741      XYRELL  DE, 104, 10
26C3 11680A    1742+     LD      DE, 10 SHL 8+(104)
26C6 CD1127    1743      CALL    A2711
26C9 DB16      1744      IN      R, (KEY2)
26CB          1745      XYRELL  DE, 104, 20
26CB 116814    1746+     LD      DE, 20 SHL 8+(104)
26CE CD1127    1747      CALL    A2711
26D1 DB17      1748      IN      R, (KEY3)
26D3          1749      XYRELL  DE, 104, 30
26D3 11681E    1750+     LD      DE, 30 SHL 8+(104)
26D6 CD1127    1751      CALL    A2711
26D9 DB10      1752      IN      R, (POT0)
26DB          1753      XYRELL  DE, 104, 50
26DB 116832    1754+     LD      DE, 50 SHL 8+(104)
26DE CD1127    1755      CALL    A2711
26E1 DB1D      1756      IN      R, (POT1)
26E3          1757      XYRELL  DE, 104, 60
26E3 116830    1758+     LD      DE, 60 SHL 8+(104)
26E6 CD1127    1759      CALL    A2711
26E9 DB1E      1760      IN      R, (POT2)
26EB          1761      XYRELL  DE, 104, 70
26EB 116846    1762+     LD      DE, 70 SHL 8+(104)
26EE CD1127    1763      CALL    A2711
26F1 DB1F      1764      IN      R, (POT3)
26F3          1765      XYRELL  DE, 104, 80
26F3 116850    1766+     LD      DE, 80 SHL 8+(104)
26F6 CD1127    1767      CALL    A2711
26F9 189E      1768      JR      A2699 ; LOOP forever
1769 ;
1770 ; Display input port tables
26FB 0603      1771 A26FB  LD      B, 3 → B=counter=3 char/line
26FD 7E        1772 A26FD LD      A, (HL) → Get character CODE, PUT IT IN A
26FE CD3025    1773      CALL    A253D → Display CHAR ON SCREEN
2701 23        1774      INC     HL → POINT TO NEXT CHAR IN TABLE
2702 10F9      1775      DJNZ    A26FD → Do all 3 CHAR'S
2704 7B        1776      LD      A, E → Get horizontal pos'n, PUT IT IN A

```

GET AND DISPLAY IN THE
CORRECT POSITION THE
FOLLOWING INPUTS.

PLAYER 2 JOY/TRIG STATUS

PLAYER 3

PLAYER 4

KEYPAD COLUMN 0 STATUS

1
2
3

PLAYER 1 POT STATUS

PLAYER 2

PLAYER 3

PLAYER 4

2705 D618	1777 SUB 18H	] → Reduce 3 char positions E = PREVIOUS Position
2707 5F	1778 LD E, R	
2708 7R	1779 LD R, D	] → Get vertical position, PUT IT IN A
2709 C60R	1780 ADD R, 10	
270B 57	1781 LD D, R	] → Increment by 1 line ; (10 pixels)
270C FE5R	1782 CP 90	
270E 20EB	1783 JR NZ, R26FB	] → IF Y=90, DO NOT GET ANY MORE LABELS (Line 80 is last line)
2710 C9	1784 RET	
	1785 ;	
	1786 ; Display R in ASCII	
2711 67	1787 R2711 LD H, R	→ Save input value in H
2712 E6F0	1788 AND 0F0H	→ GET M. S. nibble first
2714 0F	1789 RRCA	] → SHIFT IT INTO L.S. NYBBLE
2715 0F	1790 RRCA	
2716 0F	1791 RRCA	
2717 0F	1792 RRCA	
2718 CD4625	1793 CALL R2546	→ Convert to ASCII
271B CD3D25	1794 CALL R253D	→ Display IT
271E 7C	1795 LD R, H	→ GET L. S. nibble last
271F E60F	1796 AND 0FH	→ MASK M.S. NYBBLE
2721 CD4625	1797 CALL R2546	→ Convert to ASCII
2724 CD3D25	1798 CALL R253D	→ Display IT
2727 C9	1799 RET	
	1800 ;	↙ VECTOR @ 2010H
	1801 ; Interrupt routine for Display All Input Devices	A' IS INITIALLY TO ZERO
2728 F3	1802 R2728 DI	→ DISABLE INTERRUPTS
2729 08	1803 EX AF, AF'	] IF A' = 0, JUMP TO 2744. DO NOT GET LITE PEN INPUT. IF A' ≠ 0, CONTINUE
272A A7	1804 AND R	
272B 2817	1805 JR Z, R2744	
272D 08	1806 EX AF, AF'	
272E DB0E	1807 IN R, (VERAF)	→ Get vertical Lite Pen COORDINATE
2730 CB3F	1808 SRL R → IGNORE BIT 0	] DISPLAY ITS Y COORDINATE (0-127 _D)
2732	1809 XYRELL DE, 28, 10	
2732 111C0R	1810+ LD DE, 10 SHL 8+(28)	
2735 CD1127	1811 CALL R2711	
2738 DB0F	1812 IN R, (HORAF)	→ Get horizontal Lite Pen COORDINATE
273A D608	1813 SUB 8	] DISPLAY ITS X COORDINATE 0-159 _D
273C	1814 XYRELL DE, 28, 20	
273C 111C14	1815+ LD DE, 20 SHL 8+(28)	
273F CD1127	1816 CALL R2711	
2742 FB	1817 R2742 EI	→ ENABLE INTERRUPTS
2743 C9	1818 RET	
	1819 ;	
2744 3C	1820 R2744 INC R	] INC A'. GET LITE PEN INPUT NEXT TIME.
2745 08	1821 EX AF, AF'	
2746 18FA	1822 JR R2742	→ JUMP TO 2742

ERROR ROUTINE

(ERROR DATA IS SAVED IN A'
BC POINTS TO CHAR DRIVERS TABLE
IX IS USED TO JUMP BACK TO "CONTINUE" ADR)

ENTER WITH: E = ERROR CODE, C = ERROR DATA (INFO BYTE)

ERROR 2748 _H	F3 D9 79 D9 E6 OF A7 CA DB 27	DI EXX LD A, C EXX AND OF _H AND A JR Z, DELAY4	DISABLE INTERRUPTS A = C' = Q/S FLAG, REPETITIVE TESTS CTR SEE NOTE (BACK OF PAGE) IF REPETITIVE TESTS CTR = 0, JMP TO DELAY 4 (REPORT NO ERROR)
2752 _H	79 08 7B	LD A, C EX AF, AF'	SAVE ERROR DATA IN A'
ERROR1 2754 _H	DD 21 5C 27 C3 DD 27	LD A, E LD IX, ERROR2 JP DSPLY	PUT ERROR CODE IN A IX = RETURN (CONTINUE) ADR DISPLAY ERROR CODE
ERROR2 275C _H	21 61 27 18 6B	LD HL, ERROR3 JR DELAY	HL = RETURN (CONTINUE) ADR JMP TO DISPLAY DELAY ROUTINE
ERROR3 2761 _H	3E OF D3 EF 3E 39 D3 FF 21 6E 27 18 5E	LD A, 0000 1111 OUT (LDSPLY), A LD A, 0011 1001 OUT (RDSPLY), A LD HL, ERROR4 JR DELAY	OUTPUT 1111 TO DISPLAY (I = INFO NOTATION) HL = RETURN (CONTINUE) ADR JMP TO DISPLAY DELAY ROUTINE
ERROR4 276E	08 4F	EX AF, AF' LD C, A EX AF, AF' LD A, C	A = ERROR DATA (INFO BYTE) A = A' = ERROR DATA SAVE IT
2770 _H	08 79 DD 21 78 27 18 65 21 7D 27 18 4F	LD IX, ERROR5 JR DSPLY LD HL, ERROR6 JR DELAY	DISPLAY ERROR DATA (INFO BYTE) HL = RETURN (CONTINUE) ADDRESS JMP TO DISPLAY DELAY ROUTINE
ERROR5 2778 _H	21 7D 27 18 4F	LD A, 0111 0110 OUT (LDSPLY), A LD A, 0111 1001 OUT (RDSPLY), A LD HL, ERROR7 JR DELAY	DISPLAY 1111 DISPLAY 1111
ERROR6 277D _H	3E 76 D3 EF 3E 79 D3 FF 21 8A 27 18 42		

NOTE: C' IS INITIALIZED TO 81_H AT START UP AND SET TO 80_H AT THE END OF TEST 3 "CHECK SCREEN INTERRUPTS". THE REPETITIVE TESTS COUNTER IS INSPECTED BY THE ERROR ROUTINE TO PREVENT REPORTING OF AN ERROR DURING THE FIRST TESTS PASS ONCE THE NORMAL SCREEN INTERRUPT IS SET UP.

31

ERROR 7	278A _H	3E 38	LD A, 0011 1000	1	DISPLAY 1 1
		D3 EF	OUT (LSPLY), A		
		3E 73	LD A, 0111 0011	1	
		D3 FF	OUT (LSPLY), A		
	2790 _H	21 97 27	LD HL, ERROR 8		
		DOG 18 35	JR DELAY		
ERROR 8	2797 _H	DOG → D9 _{BOY}	EXX C' = Q/S FLAG IN BIT 7		IS Q/S FLAG RESET? IF NO, JMP TO BLANK DISPLAY IF YES, CONTINUE TO DISPLAY "RUN TIME"
		CB 79	BIT 7, C		
		D9	EXX		
		DOG → 20 22	JR NZ, ERROR B		
		D9	EXX L' = DECIMAL HOURS, 00-09 MAX		
		7D ← DOG	LDA, L		
		D9	EXX		
	27A0 _H	01 16 20	LD BC, CTBL		DISPLAY HOURS (0-9 MAX) TO LEFT DISPLAY
		81	ADD A, C		
		4F	LD C, A		
		0A	LD A, (BC)		
		D3 EF	OUT (LDSPLY), A		
		3E 74	LD A, 0111 0100	1	SEND IT TO RIGHT DISPLAY
		D3 FF	OUT (RDSPLY), A		
		21 B1 27	LD HL, ERROR 9		
		DOG → 18 1B _{BOY}	JR DELAY		
ERROR 9	27B1 _H	D9	EXX H' = DECIMAL MINUTES		DISPLAY MINUTES
		7C	LDA, H		
		D9	EXX		
		DD 21 BA 27	LD IX, ERROR A		
		18 23 _{BOY}	JR DSPLY		
ERROR A	27BA _H	21 BF 27	LD HL, ERROR B		
		18 0D _{DOG}	JR DELAY		
ERROR B	27BF _H	AF	XOR A 0000 0000 BLANK		BLANK DISPLAY
	27C0 _H	D3 EF	OUT (LDSPLY), A		
	27C2 _H	D3 FF	OUT (RDSPLY), A		

SAVED IN A' REG

27C4 _H	21C9 27	LD HL, ERRORC	
	1803	JR DELAY	
ERRORC 27C9 _H	2154 27	LD HL, ERROR 1	→ LOOP BACK TO REPEAT ERROR REPORT. E=ERROR CODE, A=ERROR DATA (INFO BYTE)

FALL INTO DELAY ROUTINE (TOTAL BLANK DELAY 1.8 SEC)

DISPLAY 1 SEC DELAY ROUTINE

33

DELAY 27CC_H 16 02
 DELAY 1 27CE_H 3E FF
 DELAY 2 27D0_H 06 FF
 DELAY 3 1 10 FE
 DOG 3D DOG
 20 F9
 15
 20 F4
 E9

LD D, 2 D COUNTER = 2
 LD A, 255_D A COUNTER = 255_D
 LD B, 255_D B COUNTER = 255_D
 DJNZ DELAY 3 LOOP BACK TO DELAY 3 255 TIMES
 DEC A } LOOP BACK TO DELAY 2 255 TIMES
 JR NZ, DELAY 2 }
 DEC D } LOOP BACK TO DELAY 1 2 TIMES
 JR NZ, DELAY 1 }
 JP (HL) LOOP BACK TO RETURN ADDRESS TO
 CONTINUE ERROR ROUTINE

DELAY 4 27DB_H FB
 E9 ← BOY

EI ENABLE INTERRUPTS
 JP (HL) DO NOT REPORT ERROR FOR FIRST PASS (C = 80_H)
 JMP TO NEXT TEST

OUTPUT 2 HEX DIGITS TO DISPLAY (CALL INSTRUCTION NOT PERMITTED)
 ENTER WITH: A = 2 HEX DIGITS (NIBBLES), IX = ROUTINE RETURN (CONTINUE) ADDRESS

NOTE: REGISTERS B, C AND D ARE CLOBBERED

DSPLY - 27DD_H 01 16 20
 27E0_H 57
 E6 F0
 0F
 0F
 0F
 0F
 81
 4F
 DOG 0A
 D3 EF

LD BC, CTBL POINT BC AT CHAR TABLE DRIVERS
 LD D, A SAVE 2 HEX DIGITS IN D
 AND 1111 0000 A = MSD

RRCA 47 → 0 → CF ROTATE MSD
 RRCA RIGHT TO BITS 3-0
 RRCA
 RRCA
 RRCA

ADD A, C ← TABLE LOW ORDER ADR }
 LD C, A } POINT BC AT
 LD A, (BC) } DIGIT DRIVER TO DISPLAY
 OUT (LDSPY), A } GET SEGMENTS DRIVER.
 SEND IT TO LEFT DISPLAY

DSPLY 1 0E 16

LD C, 16_H POINT BC AT CHAR TABLE AGAIN.
 LD A, D (SET LOW ORDER BYTE ONLY REQ'D)
 AND 0000 1111 A = 2 HEX DIGITS AGAIN
 A = LSD

DSPLY 2 E6 0F

DSPLY 3 27F4_H 81

ADD A, C
 LD C, A
 LD A, (BC)
 OUT (RDSPY), A
 JP (IX)

SEND LSD SEGMENTS DRIVER
 TO RIGHT DISPLAY.

27F8_H 42 41 4E 4B
 20 23 00

"BANK #"

27FF_H FF UNUSED BYTE

TEST USER RAM ENTER "TEST RAM" PARAMETERS

ENTER	2800 _H	CD FF 24 CD 16 25 FF 35 04 32 0C 8B 24 3E 01 CD 71 25	CALL A24FF CALL A2516 RST 38 _H STRDIS+1 X=4 Y=50 _D CHAR DISP=00001100 STR ADDR=248B _H LD A, 1 CALL A2571	SET COLORS, CLEAR SCREEN DISPLAY "ENTR 4-DIGIT HEX"
	2812 _H	7C FE 50 38 E9 E5	LD A, H CP 50 _H JRC, ENTER PUSH HL	DISPLAY "STRT ADDR"
ENTER 1	2818 _H	CD CF 25 CD 07 25 CD 26 25	CALL A25CF CALL A2507 CALL A2526	ENTER/DISPLAY 4 DIGIT ADDRESS PUT IT IN HL
	2821 _H	FF 35 04 32 0C 4B 20 ^{Boy}	RST 38 _H STRDIS+1 X=4 Y=50 _D CHAR DISP=00001100 STR ADDR=204B _H	GET ANOTHER ENTRY IF < 5000 _H
		FF 35 04 3C 0C	RST 38 _H STRDIS+1 X=4 Y=60 _D CHAR DISP=06001100	SAVE START ADDRESS WAIT 1/2 SEC CLEAR SCREEN
	282D _H	55 20 AF	STR ADDR=2055 _H XOR A	DISPLAY "ENTER 2-DIGIT HEX"
	2830 _H	CD 71 25 06 07 3E 01 BC	CALL A2571 LD B, 7 LD A, 1 CP H	DISPLAY "BANK SIZE"
ENTER 3				DISPLAY "(K BYTES)"
				ENTER/DISPLAY 2 DIGIT BANK SIZE PUT IT IN H

35

	28 05	JR Z, ENTER 4
	87	ADD A, A
	10 FA	DJNZ ENTER 3
	18 DC	JR ENTER 1
ENTER 4	FE 40	CP 64 _D
2841 _H	20 02	JR NZ, ENTER 5
	D6 14	SUB 20 _D
ENTER 5	47	LD B, A
	21 00 00	LD HL, 0
	11 00 04	LD DE, 1024 _D
ENTER 6	19	ADD HL, DE
	10 FD	DJNZ ENTER 6
	E5	PUSH HL
2850 _H	CD CF 25	CALL A25CF
ENTER 7	CD 07 25	CALL A2507
	CD 26 25	CALL A2526
	FF	RST 38 _H
	35	STRDIS+1
	04	X=4
	32	Y=50 _D
	DC	CHAR DISP=00001100
	F8 27	STR ADDR=27F8 _H
2860 _H	AF	XOR A
	CD 71 25	CALL A2571
	7C	LD A, H
	A7	AND A
	28 EB	JR Z, ENTER 7
	44	LD B, H
	D1	POP DE
	E1	POP HL
	E5	PUSH HL
	D5	PUSH DE
	2B	DEC HL
ENTER 8	19	ADD HL, DE
	38 E2	JR C, ENTER 7
2871 _H	10 FB	DJNZ ENTER 8

LEGAL ENTRIES ARE
1, 2, 4, 8, 16, 32 OR 64
K BYTES

IF 64K BYTES WAS
ENTERED, ONLY 44K
IS ADDRESSABLE

HL = BANK SIZE (IN BYTES)

SAVE BANK SIZE
WAIT 1/2 SEC
CLEAR SCREEN

DISPLAY "ENTR 2-DIGIT HEX"

DISPLAY
"BANK #"

ENTER/DISPLAY 2 DIGIT BANK#
PUT IT IN H

ZERO ENTRY
IS NOT LEGAL

DE = BANK SIZE
HL = START ADDRESS
SAVE START ADDRESS
SAVE BANK SIZE
BACK UP START ADDRESS
DETERMINE BANK END ADDRESS
GET ANOTHER ENTRY IF BANK#
IS TOO LARGE (END ADDRESS > FFFF)

36

2873_H CD CF 25
CD 07 25

DI
E5
23
22 FB 4F
AF

ED 52

2882_H 22 FD 4F

2B
22 F9 4F
23

288A 11 2C 28

7C
CD 12 27

2891_H 7D
CD 11 27

3E 2D
CD 3D 25
E1

7C
CD 12 27
7D

28A0_H CD 11 27

FF
35
2C
32
0C

5F 20

DI
FD 21 F9 4F

28AF_H 16 00

CALL A25CF

CALL A2507

POP DE

PUSH HL

INC HL

LD (4FFB_H), HL

XOR A

SBC HL, DE

LD (4FFD_H), HL

DEC HL

LD (4FF9_H), HL

INC HL

LD DE, 282C

LD A, H

CALL A2712

LD A, L

CALL A2711

LD A, 2D

CALL A253D

POP HL

LD A, H

CALL A2712

LD A, L

CALL A2711

RST 38_H

STRDIS+1

X=44_D

Y=50_D

CHAR DISP=00001100

STR ADDR=205F_H

POP DE

LD IY, 4FF9_H

LD D, 0

WAIT 1/2 SEC

CLEAR SCREEN

DE= BANK SIZE (IN BYTES)

SAVE END ADDRESS

HL= BANK UPPER LIMIT

SAVE BANK UPPER LIMIT

CLEAR CARRY

HL= BANK START ADDRESS

SAVE BANK START ADDRESS

SAVE BANK LOWER LIMIT

HL= BANK START ADDRESS

DE= Y, X COORDINATES

DISPLAY 1ST
2 DIGITS

DISPLAY
BANK

DISPLAY LAST
2 DIGITS

START ADDRESS

DISPLAY " - "

HL= BANK END ADDRESS

DISPLAY

BANK
END ADDRESS

DISPLAY
"RAM TEST"

RESTORE STACK (TOSS OUT)

POINT IY AT TEST RAM TABLE

ERROR STATUS=0

AND FALL INTO

37

NO
CALL
INSTR
USED
HERE ON



CHECK RAM

TABLES: HI-RES @ 299C
USER RAM PARAMETER INPUT @ 4FF9H

ENTER WITH: IY = ADDRESS OF TEST RAM TABLE

D = ERROR STATUS 00 = SINGLE BANK
01 = HIGH RES BANKS (SAVED IN D')

WHERE RAM TEST TABLE IS DEFINED AS:

RAMTAB ?? LOWER LIMIT (BANK START ADDRESS - 1)
?? UPPER LIMIT (BANK END ADDRESS + 1)
?? BANK START ADDRESS

CHKRAM 28B1H AF XOR A
5F LDE, A
4F LDC, A
47 LDB, A
6F LD L, A
67 LDH, A
D9 EXX
28B8H C3 68 20 JP ICNT

ZERO
TEST COUNTER = E'
HIGH RES BANK 1 ERROR = C'
BANK 2 = B'
BANK 3 = L'
BANK 4 = H'
ERROR STATUS = D'
DISPLAY/INCREMENT TEST COUNTER
↑
PASS NUMBER

TEST RAM

ENTER WITH: SAME AS ABOVE CHKRAM

C' = B' = L' = H' = HIGH RES BANK 1-4 ERROR = 0

TRAM 28BBH 0E 00 LDC, 0
06 01 LDB, 1
TRAM1 28C2H FD 66 05 LDH, (IY+5)
FD 6E 04 LD L, (IY+4)
TRAM2 70 LD (HL), B
7E LD A, (HL)
A8 XOR B
28 07 JR Z, TRAM3
DD 21 D1 28 LD IX, TRAM3
C3 E9 23 JP ERROR
TRAM3 28D1H 23 INC HL

C = SINGLE BANK ERROR
B = TEST BYTE = 00000001
HL = BANK START ADDRESS
LOAD TEST BYTE INTO RAM
READ SAME RAM BYTE
IF ERROR OCCURED,
JUMP TO SAVE ERROR
INCREMENT RAM ADDRESS

38

	28D2 _H	7C FD BE 03 20 ED 7D FD BE 02 20 E7	LD A, H CP (IY+3) JR NZ, TRAM2 LD A, L CP (IY+2) JR NZ, TRAM2	LOOP BACK TO TRAM2 TO TEST NEXT BYTE IF RAM ADDR $\neq$ UPPER LIMIT
TRAM 4	28DE _H	2B 7C 28E0 _H	DEC HL LD A, H CP (IY+1)	BACK UP RAM ADDRESS
		FD BE 01 20 06 7D FD BE 00 28 18	JR NZ, TRAM5 LD A, L CP (IY) JR Z, TRAM7	JUMP TO TRAM7 IF RAM ADDR = LOWER LIMIT
TRAM 5		7E A8 28 07 DD 21 F6 28	LD A, (HL) XOR B JR Z, TRAM6 LD IX, TRAM6	READ RAM BYTE
	28F3 _H	C3 E9 23	JP ERROR	IF ERROR OCCURED, JUMP TO SAVE ERROR
TRAM 6	28F6 _H	78 2F 77 AE 28 E2 DD 21 DE 28	LD A, B CPL LD (HL), A XOR (HL) JR Z, TRAM4 LD IX, TRAM4	LOAD RAM BYTE WITH COMPLIMENT OF TEST BYTE
	2900 _H	C3 E9 23	JP ERROR	IF ERROR OCCURED, JUMP TO SAVE ERROR. LOOP BACK TO TRAM4 TO TEST NEXT BYTE.
TRAM 7		23 7C FD BE 03 20 06 7D FD BE 02 28 10	INC HL LD A, H CP (IY+3) JR NZ, TRAM8 LD A, L CP (IY+2) JR Z, TRAM A	INCREMENT RAM ADDRESS
		78 2F AE 28 07	LD A, B CPL XOR (HL) JR Z, TRAM9	JUMP TO TRAM A IF RAM ADDR = UPPER LIMIT
TRAM 8	2910 _H			READ RAM BYTE

39

	DD 21 IC 29	LD IX, TRAM 9
	C3 E9 23	JP ERROR
TRAM 9 291CH	AF	XOR A
	77	LD (HL), A
	18 E3	JR TRAM 7
TRAM A 2920H	CB 20	SLA B
	30 9B	JR NC, TRAM 1
	79	LD A, C
	A7	AND A
	20 03	JR NZ, TRAM B
	C3 68 20	JP ICNT
TRAM B	D9	EXX
	7A	LD A, D
	D9	EXX
	1E 04	LD E, 04
2930H	A7	AND A
	CA 37 24	JP Z, SET QS

IF ERROR OCCURED,
JUMP TO SAVE ERROR

CLEAR RAM BYTE

CHANGE TEST BYTE
IF CARRY SET, TEST BANK AGAIN
WITH NEW TEST BYTE.

IF ANY ERROR OCCURED,
REPORT IT.

IF NO ERROR OCCURED,
JUMP TO ICNT

D=ERROR STATUS
00 SINGLE BANK
01 HI-RES BANKS

E=ERROR CODE

SINGLE BANK
REPORT

FALL INTO REPORT HIGH RESOLUTION ERROR

NOTE: ERROR CODE 04 IS NOT INCLUDED IN
HI-RES ERROR REPORT SEQUENCE.

REPORT HI-RES SCREEN RAM ERROR

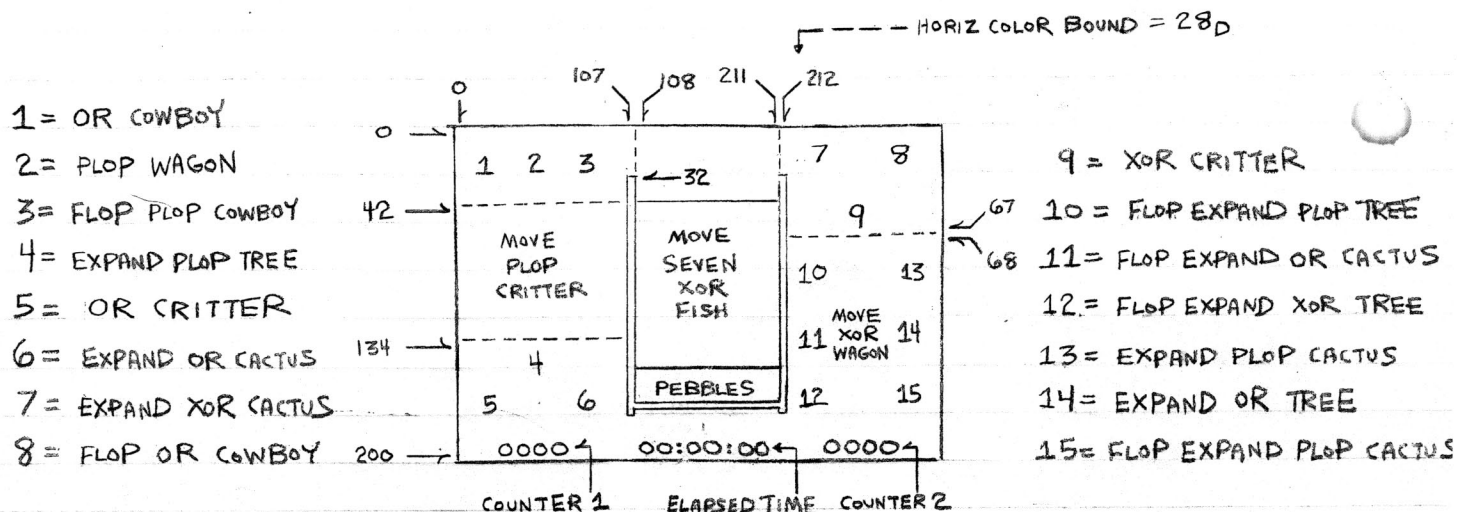
REPORT	2934 _H	3E 0F D3 EF 3E 39 D3 FF	LDA, 0000 1111 I OUT (LDSPLY), A LDA, 0011 1001 I OUT (RDSPLY), A	OUTPUT II TO DISPLAY (I = INFO NOTATION)
		21 42 29 C3 CC 27	LD HL, REPORT 1 JP DELAY	HL = RETURN (CONTINUE) ADR JMP TO DISPLAY DELAY ROUTINE
REPORT 1	2942 _H	3E 3F D3 EF 3E 06 D3 FF 21 50 29 C3 CC 27	LDA, 3F 0011 1111 I OUT (LDSPLY), A LDA, 06 0000 0110 I OUT (RDSPLY), A LD HL, REPORT 2 JP DELAY	HL = CONTINUE ADR DISPLAY = 01 (HI-RES BANK 1)
REPORT 2	2950 _H	D9 79 D9 DD 21 5A 29 C3 DD 27	EXX C = BANK 1 ERROR BYTE LDA, C EXX LD IX, REPORT 3 JP DSPLY	IX = CONTINUE ADR DISPLAY ERROR BYTE FOR HI-RES BANK 1
REPORT 3	295A _H	21 60 29 C3 CC 27	LD HL, REPORT 4 JP DELAY	HL = CONTINUE ADR
REPORT 4	2960 _H	3E 3F D3 EF 3E 5B D3 FF 21 6E 29 C3 CC 27	LDA, 3F OUT (LDSPLY), A LDA, 5B 0101 1011 I OUT (RDSPLY), A LD HL, REPORT 5 JP DELAY	DISPLAY = 02 (HI-RES BANK 2)
REPORT 5	296E _H	D9 78 D9 DD 21 78 29 C3 DD 27	EXX B = BANK 2 ERROR BYTE LDA, B EXX LD IX, REPORT 6 JP DSPLY	IX = CONTINUE ADR DISPLAY ERROR BYTE FOR HI-RES BANK 2
REPORT 6	2978 _H	21 7E 29	LD HL, REPORT 7	HL = CONTINUE ADR

41

	297B _H	C3 CC 27	JP DELAY	
REPORT 7	297E _H	3E 3F	LDA, 3F	
	2980 _H	D3 EF	OUT (LDSPLY), A	
		3E 4F	LDA, 4F 0100 1111 $\bar{1}$	DISPLAY = 03
		D3 FF	OUT (RDSPLY), A	(HI-RES BANK 3)
		21 8C 29	LD HL, REPORT 8 $\overline{HL=CONTINUE}$	
		C3 CC 27	JP DELAY	
REPORT 8	298C _H	D9	EXX $L' = BANK 3$	
		7D $\leftarrow DOG$	EXX $L' = ERROR BYTE$	
		D9	LDA, L	
		DD 21 96 29	EXX $\overline{IX=CONTINUE}$	
			LD IX, REPORT 9 $\leftarrow$	DISPLAY ERROR BYTE
				FOR HI-RES BANK 3
	2993 _H	C3 DD 27 $\leftarrow DOG$	JP DSPLY	
REPORT 9	2996 _H	21 D1 31	LD HL, REPORT B $\overline{HL=CONTINUE}$	
	2999 _H	C3 CE 31	JP REPORT A $\overline{CONTINUE}$	
			@ 31CE _H $\leftarrow$	

42

HIGH RESOLUTION DEMO



DATA BASES

↑
STACK BEGINS HERE

7FA7	RANSHT (4 BYTES)	
7FAB	REG Y	
7FAC	COUNTER 1 (2 BYTES)	
7FAE	COUNTER 2 (2 BYTES)	
7FB0	SECONDS	
7FBI	MINUTES	
7FB2	HOURS	
7FB3	FISH PATTERN ADDRESS (2 BYTES)	
7FB5	$X_H, Y_H, \Delta X_L, \Delta Y_L, \Delta \text{TIMER FOR FISH \#1}$	
7FBA		2
7FBF		3
7FC4		4
7FC9		5
7FCE		6
7FD3		7
7FD8	MR	
	VECTOR STATUS	
7FDA	TIME BASE	

43

7FDB	ΔX_L	FISH VECTOR PACKET
7FDC	ΔX_H	
	X_L	
7FDE	X_H	
	X CHECKS MASK	
7FE0	ΔY_L	CRITTER VECTOR PACKET
7FE1	ΔY_H	
	Y_L	
7FE3	Y_H	
	Y CHECKS MASK	
7FE5	MR	
	VECTOR STATUS	
7FE7	TIME BASE	
	ΔX_L	
	ΔX_H	
	X_L	WAGON VECTOR PACKET
	X_H	
	X CHECKS MASK	
	ΔY_L	
	ΔY_H	
	Y_L	
	Y_H	
7FF1	Y CHECKS MASK	
7FF2	MR	
	VECTOR STATUS	
	TIME BASE	
7FF5	TMR 60	WAGON VECTOR PACKET
7FF6	BIT 7 = 1PPS SET BY INTERRUPT	
	X_L	
	X_H	
	X CHECKS MASK	
	ΔY_L	
	ΔY_H	
	Y_L	
	Y_H	
7FFE	Y CHECKS MASK	

44

HI-RES 4000-7FFF_H SCREEN RAM TABLE

RAMTAB1 2994_H FF 3F LOWER LIMIT (START ADDRESS - 1)
 00 80 UPPER LIMIT (END ADDRESS + 1)
 29A0_H 00 40 START ADDRESS

WAGON Y LIMITS TABLE

29A2_H 44 WAGON Y LOWER LIMIT
 93 WAGON Y UPPER LIMIT

DEMO COLOR TABLE 2 (HI-RES DEMO
MOVE FISH/CRITTER/WAGON)

COLOR1	29A4 _H	07	WHITE	PIXEL 11	LEFT SCREEN
		A3	GREEN	10	
		FC	LIGHT BLUE	01	
		00	BLACK	00	
		7D	MAIZE	PIXEL 11	RIGHT SCREEN
		A3	GREEN	10	
		FC	LIGHT BLUE	01	
		00	BLACK	00	

SIMILAR TO
ON-BOARD

SUB#56

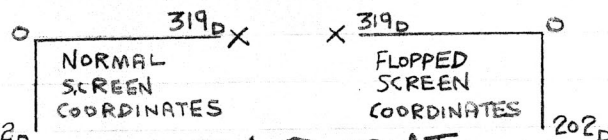
SUB#58

DETERMINE MAGIC ADDRESS @ 29AC_H
 DETERMINE SCREEN ADDRESS @ 29B2_H

ENTER WITH: REG Y = (7FAB_H) = Y COORDINATE SEE P.42

DE = X COORDINATE

A = MAGIC REGISTER VALUE



MAGIC ENTRY POINT

MADDR 29AC_H CD B8 29 CALL RELTA1
 D3 0C OUT (MAGIC), A
 29B1_H C9 RET

45

NON MAGIC ENTRY POINT

SADDR 29B2_H CD B8 29 CALL RELTA1
 CBF2 SET 6,D
 C9 RET

NON MAGIC THE ADDRESS

SIMILAR TO ON-BOARD RELTA1

RELTA1 29B8_H E5 PUSH HL
 E6 78 AND 78_H
 6F LD L,A
 7B LD A,E
 EG 03 AND 03_H
 B5 OR L
 29C0_H F5 PUSH AF
 6B LD L,E
 62 LD H,D
 EG 40 AND 40_H
 28 0B JR Z, RELTA2
 7B LD A,E
 2F CPL
 5F LD E,A
 7A LD A,D
 2F CPL
 57 LD D,A
 13 INC DE
 21 3F 01 LD HL, 319_D
 19 ADD HL, DE
 RELTA2 29D1_H E5 PUSH HL
 3A AB 7F LD A, (REG Y)
 6F LD L, A
 26 00 LD H, 0
 29 ADD HL, HL
 29
 29
 29
 54 LD D, H
 5D LD E, L
 29 ADD HL, HL
 29 ADD HL, HL
 29E0_H 29

SAVE PATTERN ADDRESS
 CLEAR SHIFT AMOUNT IN
 MAGIC REGISTER VALUE

"OR" SHIFT AMOUNT WITH
 MAGIC REGISTER VALUE

SAVE MAGIC REGISTER VALUE

HL = X COORDINATE

JUMP TO RELTA2 IF
 FLOP BIT IS NOT SET.

FLOP Y AXIS

 $X_{FLOP} = 319 - X$

SAVE THIS X

HL = Y

HL = 2Y

HL = 4Y

HL = 8Y

HL = 16Y

DE = 16Y

HL = 32Y

HL = 64Y

HL = MAGIC ADDRESS

46

29E1 _H	19	ADD HL,DE	HL=80Y
	D1	POP DE	DE=X
	CB1A	RRD	DE=X'
	CB1B	RRE	
	CB3B	SRL E	
	16 00	LDD,0	
	19	ADD HL,DE	HL=80Y+X'
	EB	EX DE,HL	
	F1	POP AF	
	E1	POP HL	
	C9	RET	

DE = MAGIC ADDRESS
 A = UPDATED MAGIC REGISTER VALUE
 HL = PATTERN ADDRESS

FILL/BLANK AREA

SIMILAR TO ON-BOARD SUB #42

ENTER WITH: E = XSIZE (# OF BYTES WIDE TO FILL/BLANK)
 D = YSIZE (# OF LINES HIGH TO FILL/BLANK)
 B = DATA TO FILL/BLANK WITH
 HL = SCREEN ADDRESS TO BEGIN FILLING/BLANKING

FILL 29F0 _H	3E 50	LD A,50 _H	HI-RES = 80 _D BYTES PER LINE
	93	SUB E	
	4F	LD C,A	
	78	LD A,B	
FILL1	43	LD B,E	
FILL2	77	LD (HL),A	
	23	INC HL	
	10 FC	DJNZ FILL2	
	09	ADD HL,BC	
	15	DEC D	
	20 F7	JR NZ, FILL1	
	C9	RET	

47

RESTORE AREA

SIMILAR TO ON-BOARD SUB#46

ENTER WITH: DE = SAVE AREA ADDRESS

HL = ADDRESS TO RESTORE TO

SIZE OF AREA IS LOADED FROM THE SAVE AREA

RESTOR	29FF _H	EB	EX DE, HL	DE = ADDRESS TO RESTORE
	2A00 _H	4E	LD C, (HL)	HL = SAVE AREA ADDRESS
		23	INC HL	C = X SIZE
		46	LD B, (HL)	B = Y SIZE
		23	INC HL	
RESTOR 1	2A04 _H	AF	XOR A	ZERO A
RESTOR 2		C5	PUSH BC	
		D5	PUSH DE	
		47	LD B, A	ZERO B
		ED B0	LDIR	WRITE A LINE
		EB	EX DE, HL	
		E1	POP HL	
		0E 50	LD C, 50 _H	DE = ADDRESS OF NEXT LINE TO WRITE TO
		09	ADD HL, BC	
		EB	EX DE, HL	
	2A10 _H	C1	POP BC	
		10 F2	DJNZ RESTOR2	WRITE ANOTHER LINE?
		C9	RET	

WRITE FROM VECTOR

13 BYTE VECTOR PACKET CALL @2A14_H15 BYTE VECTOR PACKET CALL @2A1E_H

SIMILAR TO ON-BOARD SUB#30

ENTER WITH: IX = VECTOR PACKET ADDRESS

HL = PATTERN ADDRESS - 4 (POINTING AT RELATIVE X)

VWRIT1	2A14 _H	DD 46 0B	LD B, (IX+0B _H)	B = Y _H
		DD 5E 06	LD E, (IX+06 _H)	E = X _H
		16 00	LD D, 0	
		18 09	JR VWRIT3	

48

VWRIT2	2A1E _H	DD 46 0D	LD B, (IX+0D _H)	} B = Y _H DE = X _H
	2A21 _H	DD 5E 07	LD E, (IX+07)	
		DD 56 08	LD D, (IX+08)	
VWRIT3		DD 7E 00	LD A, (IX)	A = MR VALUE SET BLANK BIT
		DD CB 01 F6	SET G, (IX+1)	

WRITE RELATIVE

SIMILAR TO ON-BOARD SUB#34

ENTER WITH: DE = X COORDINATE

B = Y COORDINATE

A = MAGIC REGISTER VALUE

HL = PATTERN ADDRESS - 4 (POINTING AT RELATIVE X

WRITR	2A2E _H	F5	PUSH AF	
		7E	LD A, (HL)	
	2A30 _H	23	INC HL	
		83	ADD A, E	
		5F	LD E, A	
		7A	LD A, D	
		CE 00	ADC A, 0	
		57	LD D, A	
		7E	LD A, (HL)	
		23	INC HL	
		80	ADD A, B	
		32 AB 7F	LD (REGY), A	SEE P. 42
		F1	POP AF	

WRITE WITH PATTERN SIZE

SIMILAR TO ON-BOARD SUB#36

ENTER WITH: REGY = (7FAB_H) = Y COORDINATE SEE P. 42

DE = X COORDINATE

A = MAGIC REGISTER VALUE

HL = PATTERN ADDRESS - 2 (POINTING AT X SIZE)

WRITP	2A3E _H	4E	LD C, (HL)
		23	INC HL

49

2A40_H 46
23LD B, (HL)
INC HL

WRITE PATTERN

SIMILAR TO ON-BOARD SUB #36

ENTER WITH: REGY = (7FAB_H) = Y COORDINATE SEE P.42

DE = X COORDINATE

C = X SIZE

B = Y SIZE

A = MAGIC REGISTER VALUE

HL = PATTERN ADDRESS

WRIT 2A42_H CD AC 29

CALL MADDR

WRITE ABSOLUTE

SAME AS ON-BOARD SUB #38, EXCEPT AS SHOWN

ENTER WITH: DE = SCREEN ADDRESS

C = X SIZE

B = Y SIZE

A = MAGIC REGISTER VALUE

HL = PATTERN ADDRESS

WRITA 2A45_H CB 77
20 2C

BIT MRFL0P, A

CHECK FOR FLOP

CB 5F

JR NZ, MWRTFL

20 11

BIT MRXPND, A

JR NZ, MWX

WPLOP 2A4D_H
MWRT

AF

XOR A

NORMAL PLOP

C5

PUSH BC

D5

PUSH DE

2A50_H

47

LD B, A

ED B0

LDIR

12

LD (DE), A

D1

POP DE

EB

EX DE, HL

OE 50

LD C, BYTEPL

HI-RES 80_D BYTES PER LINE

09

ADD HL, BC

50

MWX, WEXPD
MWX1

2A60H

MWX2

EB
C1
10 F1
C9EB
C5
E5
411A
13
7723
77
2310 F8
70
23
70

E1

0E 50

09
C110 EB
C9CB 5F
20 16

AF

C5

D5

47

ED A0

1B

1B

EA 7D 2A

12

D1

EX DE, HL
POP BC
DJNZ MWRT
RETEX DE, HL
PUSH BC
PUSH HL
LD B, C
LD A, (DE)INC DE
LD (HL), AINC HL
LD (HL), A
INC HLDJNZ MWX2
LD (HL), B
INC HL
LD (HL), B
POP HLLD C, BYTEPL
ADD HL, BC
POP BC
DJNZ MWX1
RETBIT MRXPND, A
JRNZ, MWXF
XOR APUSH BC
PUSH DE
LD B, ALDI
DEC DE
DEC DEJP PE, WFLOP1
LD (DE), A
POP DE

WRITE EXPANDED

HI-RES 80D BYTE PER LINE

WRITE FLOPPED

MWRTF

WFLOP
WRFL1WRFL2
WFLOP1 2A7DH

2A80H

51

	2A86 _H	EB	EX DE, HL	
		0E 50	LD C, BYTEPL	HI-RES 80, BYTES PER LINE
		09	ADD HL, BC	
		EB	EX DE, HL	
		C1	POP BC	
		10 EC	DJNZ WRFL1	
		C9	RET	
MWXF		EB	EX DE, HL	WRITE WITH EXPANDED FLOP
WXFLOP		C5	PUSH BC	
MWXF1	2A90 _H	E5	PUSH HL	
		41	LD B, C	
		1A	LD A, (DE)	
		13	INC DE	
MWXF2		77	LD (HL), A	
		2B	DEC HL	
		77	LD (HL), A	
		2B	DEC HL	
		10 F8	DJNZ MWXF2	
		70	LD (HL), B	
		2B	DEC HL	
		70	LD (HL), B	
		E1	POP HL	
		0E 50	LD C, BYTEPL	HI-RES 80, BYTES PER LINE
	2AA1 _H	09	ADD HL, BC	
		C1	POP BC	
		10 EB	DJNZ MWXF1	
		C9	RET	

COWBOY PATTERN

2AAG _H	03 X SIZE
	14 Y SIZE
	00 44 00
	11 55 10
	15 55 50
2ABI _H	02 AA 00

52

02 A2 00
 02 AA 80
 00 AA 00
 00 A8 00
 2AC0_H 15 55 00
 55 55 50
 51 55 50
 41 55 00
 41 55 00
 45 55 00
 2AD2_H 01 55 00
 01 55 00
 05 45 40
 15 01 40
 50 01 40
 2AE1_H 15 00 54

COWBOY'S ARM PATTERN

2AE4_H 0A RELATIVE X
 07 ↓ Y
 02 X SIZE
 04 Y SIZE
 10 00
 05 40
 54 00
 50 00

WAGON PATTERN @ 2AEF_H

2AF1_H 00 RELATIVE Y
 04 X SIZE
 16 Y SIZE
 00 05 50 00
 00 55 55 00
 2AFB_H 01 55 55 40

53

2AFF _H	05 55 55 50
2B03 _H	15 54 15 54
	15 50 05 54
	15 40 01 54
	15 40 01 54
2B13 _H	15 50 05 54
	05 54 15 50
	01 55 55 40
	00 55 55 00
2B23 _H	00 15 54 00
	02 AA AA 80
	00 AA AA 00
	12 AA AA 84
2B33 _H	10 A8 2A 04
	10 20 08 04
	52 AA AA 85
	10 20 08 04
2B43 _H	10 00 00 04
	10 00 00 04

TREE PATTERN

2B4B _H	01 X SIZE
	11 Y SIZE
	08
	1C
	3E
2B50 _H	6B
	08
	08
	3C
	7E
	A9
	08
	3C
	7E

54

EB
89
08
1C
AE

CRITTER PATTERN

2B5E_H 00 RELATIVE X
00 ↓ Y

2B60_H 04 X SIZE
12 Y SIZE
00 00 00 00

00 14 14 00
01 55 55 40
05 41 41 50

2B72_H 05 41 41 50
05 55 55 50
01 51 45 40
00 54 15 00

2B82_H 04 05 50 10
16 AA AA 94
04 AA AA 10
00 BB EE 00

2B92_H 02 AA AA 80
05 55 55 50
05 40 01 50
05 00 00 50

2BA2_H 3F 00 00 FC
00 00 00 00

CACTUS PATTERN

2BAA_H 01 X SIZE
0C Y SIZE
20

2BAD_H 30
 38
 30
 2BB0_H B2
 F2
 F6
 3C
 3C
 30
 30
 30

7x9 PIXEL DECIMAL TABLE

2BB8 _H	7C 82 86 8A 92 A2 C2 82 7C	0
2BC1 _H	10 30 10 10 10 10 10 10 7C	1
	7C 82 02 02 7C 80 80 80 FE	2
2BD3 _H	7C 82 02 02 1E 02 02 82 7C	3
	84 84 84 84 FE 04 04 04 04	4
2BE5 _H	FE 80 80 FC 02 02 02 82 7C	5
	7C 82 80 80 FC 82 82 82 7C	6
2BF7 _H	FE 02 04 08 10 20 40 40 40	7
2C00 _H	7C 82 82 82 7C 82 82 82 7C	8
	7C 82 82 82 7E 02 02 82 7C	9
2C12 _H	00 00 00 10 00 10 00 00 00	:

WRITE 7x9 PIXEL DECIMAL

ENTER WITH: OUTPUT PORT 19_H DEFINEDREGY = (7FAB_H) = Y COORDINATE SEE P.42

DE = X COORDINATE

A = DECIMAL TO WRITE (IN BITS 3-0)

WDEC 2C1B_H 21 AF 2B LD HL, 2BAF_H
 01 09 00 LD BC, 9
 2C21_H 3C INC A

POINT HL AT
 DECIMAL TABLE-9
 (DECIMAL TABLE @ 2BB8_H)

BUMP COUNTER

56

WDEC1	2C22 _H	09	ADD HL, BC	POINT HL AT RESPECTIVE DECIMAL PATTERN
		3D	DEC A	
		20 FC	JR NZ, WDEC1	
WDEC2	2C26 _H	01 01 09	LD BC, 0901	WRITE DECIMAL
		3E 08	LD A, 08 _H	
		D5	PUSH DE	
		CD 42 2A	CALL WRIT	
		E1	POP HL	POINT DE AT X COORDINATE OF NEXT DECIMAL FRAME
2C30 _H		01 08 00	LD BC, 8	
		09	ADD HL, BC	
		EB	EX HL, DE	
		C9	RET	

$Y_{SIZE} = 9$
 $X_{SIZE} = 1$
 EXPAND PLOP
 PATTERN
 SAVE
 $X_{COORDINATE}$
 $HL = X_{COORDINATE}$
 $DE = X_{COORD} + 8$

WRITE DECIMAL NUMBER

ENTER WITH: OUTPUT PORT 19_H DEFINED

$REG Y = (7FAB_H) = Y_{COORDINATE}$ SEE P.42

DE = X COORDINATE

HL = ADDRESS OF NUMBER

MSD
(DIGIT 3)

A = NUMBER OF DIGITS TO DISPLAY

NOTE: DECIMAL DIGITS ARE 7x9 PIXELS IN SIZE

WNUM	2C36 _H	3D	DEC A	QUIT IF NO MORE DIGITS TO WRITE
		F8	RET M	
		4F	LDC, A	C = DIGIT NUMBER
		FF	RST 38 _H	
		56	INDEX N	A = DECIMAL INDEXED BITS 3-0
		E5	PUSH HL	
		C5	PUSH BC	SAVE ADDRESS OF NUMBER
		CD 1B 2C	CALL WDEC	SAVE DIGIT NUMBER
2C40 _H		C1	POP BC	WRITE DECIMAL C = DIGIT NUMBER
		79	LD A, C	PUT DIGIT NUMBER BACK IN A
		E1	POP HL	HL = ADDRESS OF NUMBER
		18 F1	JR WNUM	LOOP BACK TO DISPLAY NEXT DECIMAL.

57

WRITE ELAPSED TIME

ENTER WITH: OUTPUT PORT 19_H DEFINEDREGY = (7FAB_H) = Y COORDINATE SEE P.42

DE = X COORDINATE

NOTE: DECIMALS ARE 7x9 PIXELS IN SIZE

WTIME	2C45 _H	3E 02	LD A, 2		] WRITE HOURS
		21 B2 7F	LD HL, 7FB2 _H	SEE P.42	
		CD 36 2C	CALL WNUM		
		21 12 2C	LD HL, 2C12 _H		] WRITE ": "
	2C50 _H	CD 26 2C	CALL WDEC2		
		3E 02	LD A, 2		
		21 B1 7F	LD HL, 7FB1 _H	SEE P.42	] WRITE MINUTES
		CD 36 2C	CALL WNUM		
		21 12 2C	LD HL, 2C12 _H		
		CD 26 2C	CALL WDEC2		] WRITE ": "
	2C61 _H	3E 02	LD A, 2		
		21 B0 7F	LD HL, 7FB0 _H	SEE P.42	
		CD 36 2C	CALL WNUM		] WRITE SECONDS
		C9	RET		

INITIAL FISH VECTOR PACKET

2C6A _H	20	MAGIC REGISTER
	80	VECTOR STATUS
	03	TIME BASE
	00	ΔX _L
	00	ΔX _H
	00	X _L
2C70 _H	00	X _H
	03	X CHECKS MASK
	00	ΔY _L
	00	ΔY _H
	00	Y _L

58

2C75_H 00 Y_H
 03 Y CHECKS MASK

INITIAL CRITTER VECTOR PACKET

2C77_H 00 MAGIC REGISTER
 80 VECTOR STATUS
 02 TIME BASE
 30 ΔX_L
 00 ΔX_H
 00 X_L
 00 X_H
 03 X CHECKS MASK
 30 ΔY_L
 2C80_H 00 ΔY_H
 00 Y_L
 2A Y_H
 03 Y CHECKS MASK

INITIAL WAGON VECTOR PACKET

2C84_H 20 MAGIC REGISTER
 00 VECTOR STATUS
 00 TIME BASE
 3B TMR 60
 00 BIT 7 = 1PPS
 00 X_L
 FF X_H
 00 X CHECKS MASK
 A0 ΔY_L
 00 ΔY_H
 00 Y_L
 44 Y_H
 2C90_H 03 Y CHECKS MASK

59

CRITTER LIMITS TABLE

2C91 _H	00	X LOWER LIMIT
	57	X UPPER
	2A	Y LOWER
	74	Y UPPER

INCREMENT COUNTER

ENTER WITH: HL = ADDRESS OF COUNTER (7FAC OR 7FAE, ^{SEE} PAGE 42)

ICNT	2C95 _H	06 02	LD B, 2	B = LOOP COUNTER
ICNT1		7E	LD A, (HL)	A = COUNTER BYTE
		C6 01	ADD A, 1	INCREMENT THE BYTE
		27	DAA	ADJUST BCD
		77	LD (HL), A	PUT NEW COUNTER BACK
		C0	RET NZ	INCREMENT MOST SIGNIFICANT BYTE ¹
		23	INC HL	POINT HL TO MS BCD
		10 F7	DJNZ ICNT1	
2CA0 _H		(9	RET	

DO CRITTER

CRITTER	2CA1 _H	DD 21 E5 7F	LD IX, 7FE5 _H	SEE P. 43	WRITE CRITTER
		21 5E 2B	LD HL, 2B5E _H		
		CD 14 2A	CALL VWRT1		
	2CAB _H	3E 02	LD A, 2		SET TIME BASE
		32 E7 7F	LD (7FE7 _H), A	SEE P. 43	
	2CB0 _H	FF	RST 38 _H		UPDATE CRITTER VECTOR PACKET
		3F	VECT+1		
		E5 7F	VECTOR PACKET ADDRESS	SEE P. 43	
		91 2C	LIMITS TABLE ADDRESS		
		21 F1 7F	LD HL, 7FF1 _H	POINT HL AT Y CHECKS MASK	IF LIMIT ATTAINED BIT IS SET, INCREMENT COUNTER1.
		CB 5E	BIT 3 (HL)	SEE P. 43	
		C8	RET Z		
	2CBC _H	21 AC 7F	LD HL, 7FAC	SEE P. 42	

60

2CBF CD 95 2C CALL ICNT
2CC2H 18 09 NR DCT1

I

USER RAM 6000-7FFF_H TABLE
RAMTAB2 2CC4_H FF 5F LOWER LIMIT (START ADDRESS-1)
00 80 UPPER LIMIT (END ADDRESS+1)
00 60 START ADDRESS
2CCA_H FF FF FF UNUSED BYTES

DISPLAY COUNTER 1 (LEFT "CRITTER LIMITS" COUNTER)

DCT1 2CCD_H 11 24 00 LD DE, 36_D DE = X COORDINATE
2CDO_H 21 AC 7F LD HL, 7FAC_H SEE P.42 POINT HL TO COUNTER 1
18 06 JR DCT2A

DISPLAY COUNTER 2 (RIGHT "WAGON LIMITS" COUNTER)

DCT2 2CD5_H 11 F8 00 LD DE, 248_D DE = X COORDINATE
21 AE 7F LD HL, 7FAE_H SEE P.42 POINT HL TO COUNTER 2
DCT2A 3E BE LD A, 190_D SEE P.42 PUT Y COORDINATE IN REG Y
32 AB 7F LD (REGY), A
2CE0_H 3E 04 LD A, 4 A = DISPLAY 4 BCD'S
CD 36 2C CALL WNUM WRITE THE DECIMAL NUMBER
C9 RET

DISPLAY TIME

DTIM 2CE6_H 3E BE LD A, 190_D SEE P.42 PUT Y COORDINATE IN REG Y
32 AB 7F LD (REGY), A
11 80 00 LD DE, 128_D DE = X COORDINATE
CD 45 2C CALL WTIME WRITE THE TIME
2CF1_H C9 RET

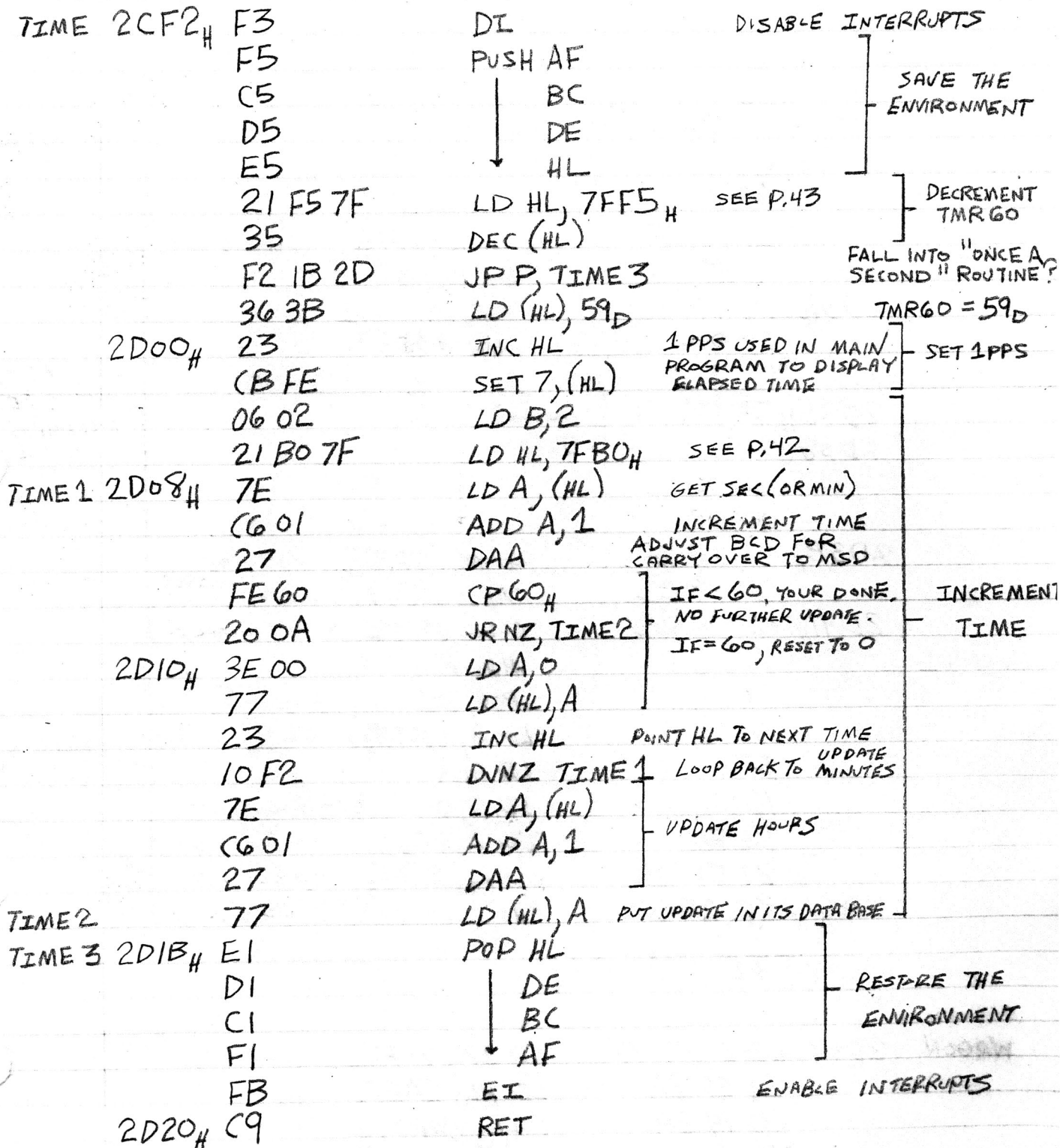
BALCHECK #R

7FB0 SECONDS
 7FB1 MINUTES
 7FB2 HOURS
 7FF5 TMR60
 7FF6 BIT 7=1PPS

TIME 0051

61

DEMO INTERRUPT ROUTINE (VECTOR 2E00H) UPDATE ELAPSED TIME



62

MOVE WAGON

MWAGON 2D21_H 3A FD 7F

F5

FF

3D

FA 7F

05

A2 29

21 FE 7F

CB 5E

2D31_H 28 11

18 06

LD A, (7FFD_H) SEE P.43

PUSH AF

RST 38_H

VECTC+1

ΔYL ADDRESS SEE P.43

TIME BASE

LIMITS TABLE ADDRESS SEE P.43

POINT HL AT
Y CHECKS MASK

LD HL, 7FFE

BIT 3, (HL)

JRZ, MWAGON1

JR A2D3B

SAVE
OLD Y_HUPDATE
WAGON
VECTOR PACKET2D35_H FF FF FF] UNUSED
2D38_H FF FF FF] BYTESIF LIMIT ATTAINED
BIT IS SET,
INCREMENT
COUNTER 22D3B_H 21 AE 7F

CD 95 2C

2D41_H CD D5 2C

76

C1

3A F8 7F

5F

1600

3E 20

21 EF 2A

2D51_H CD 2E 2ALD HL, 7FAE_H SEE P.43

CALL ICNT INCREMENT COUNTER

CALL DCT2 DISPLAY COUNTER 2

HALT

POP BC B=OLD Y_HLD A, (7FF8_H) A=X_H SEE P.43

LDE, A

LDD, 0 DE=X COORDINATE

LD A, 20_H A=MAGIC REGISTERLD HL, 2AEF_H HL=PAT ADR

CALL WRITR BLANK WAGON

BLANK
WAGON

WRITE WAGON

WAGON 2D54 DD 21 F2 7F

21 EF 2A

CD 14 2A

C9

LD IX, 7FF2_HLD HL, 2AEF_H

CALL VWRT1

RET

SEE P.43 POINT IX AT
VECTOR PACKET
PATTERN

HL=WAGON ADDRESS

WRITE WAGON

63

RANGED (FOR HI-RES DEMO 1)
 SAME AS ON-BOARD SUB#118, EXCEPT AS SHOWN
 ENTER WITH: A=MAXIMUM NUMBER TO GENERATE+1

RANGED 2D5F_H F5
 2D60_H 2A A7 7F
 CD 8A 2D
 01 17 00
 09
 8A

22 A7 7F
 2A A9 7F
 2D71_H 5F
 CD 8A 2D
 19

22 A9 7F
 2D79_H 5A
 EB
 F1
 A7
 4F
 7A
 28 08

2D81_H AF
 R1 19
 30 01
 3C

R2 0D
 20 F9
 C9

SHIFTR 2D8A_H 44
 4D
 AF
 16 07

SH1 29
 2D90_H 17

PUSH AF
 LD HL, 7FA7_H SEE P.42
 CALL SHIFTR
 LD BC, 23_D
 ADD HL, BC
 ADC A, D
 LD (7FA7_H), HL SEE P.42
 LD HL, 7FA9_H SEE P.42
 LD E, A

CALL SHIFTR
 ADD HL, DE
 LD (7FA9_H), HL
 LD E, D
 EX DE, HL
 POP AF
 AND A
 LD C, A
 LD A, D
 JR, R3
 XOR A

ADD HL, DE
 JR NC, R2
 INC A
 DEC C
 JRNZ, R1
 RET

LD B, H
 LD C, L
 XOR A
 LDD, 7
 ADD HL, HL
 RLA

64

2D91_H 15
 20 FB
 09
 8A
 C9

DEC D
 JR NZ, SH1
 ADD HL, BC
 ADC A, D
 RET

FISH PATTERN

2D97_H 00
 00
 02
 07
 80 A0
 51 54
 AA AA
 2DA1_H 15 52
 AA 2A
 52 82
 80 A8

FLOPPED FISH PATTERN

2DA9_H 00
 00
 02
 07
 0A 02
 15 45
 2DB1_H AA AA
 85 54
 A8 AA
 82 85
 2DB9_H 2A 02

65

FISH LIMITS TABLE

2DBB _H	6C	X LOWER	108 _D
	CC	X UPPER	204 _D
	2A	Y LOWER	42 _D
	95	Y UPPER	149 _D

RANDOMIZE A DELTA

RANDELT	2DBF _H	3E 7F	LD A, 127 _D	(A-1 = MAX# TO GENERATE) } RANDOMIZE A VALUE FROM 0-126_D IF RANDOM VALUE IS EVEN, USE THIS VALUE AS A POSITIVE Δ ($+\Delta = 0, 2, 4 \dots 126$) IF RANDOM VALUE IS ODD, CHANGE THIS VALUE SO AS TO APPEAR AS A NEGATIVE Δ ($-\Delta = -2, -4, -6 \dots -126$)
	2DCI _H	CD 5F 2D	CALL RANGED	
		4F	LDC, A	
		1F	RRA	
		79	LD A, C	
		DO	RET NC	
		2F	CPL	
		C9	RET	

FISH VECTOR PACKET INFO INDEX TABLE

SEE PAGE 42

2DCA _H	B5 7F
	BA 7F
	BF 7F
2DDO _H	C4 7F
	C9 7F
	CE 7F
	D3 7F

RANDOMIZE ΔX_L AND ΔY_L

DELTX Y	2DD8 _H	CD BF 2D	CALL RANDELT	GET A RANDOM $\pm \Delta X_L$. PUT IT IN VECTOR PACKET.
		32 DB 7F	LD (7FDB _H), A	
2DEI _H		CD BF 2D	CALL RANDELT	GET A RANDOM $\pm \Delta Y_L$. PUT IT IN VECTOR PACKET.
		32 EO 7F	LD (7FEO _H), A	
		C9	RET	

66

TIMER CHECK

TIMERCK	2DE5 _H	35	DEC (HL)	} (HL POINTS TO ATIMER) DEC ATIMER. IF ATIMER=0, CONTINUE. IF ATIMER≠0, RETURN
		C0	RET NZ	
		E5	PUSH HL	SAVE ATIMER POINTER
		3E 50	LDA, 80D	} RANDOMIZE A NEW ATIMER (0-79D)
		CD 5F 2D	CALL RANGED (A-1 = MAX# TO GENERATE)	
		C3 D4 30	JP 30D4 _H	JUMP TO FINISH (OOPS!)
	2DFO _H	FF	UNUSED BYTE	

CHECK FOR FLOP

CKFLOP	2DF1 _H	E5	PUSH HL	GET AXL FROM VECTOR PACKET. IF POSITIVE, SET XH=00, PUT FISH PATTERN ADDRESS IN 7FB3 _H . IF NEGATIVE, SET XH=FF, PUT FLOPPED FISH PATTERN ADDRESS IN 7FB3 _H
		3A DB 7F	LDA, (7FDB _H)	SEE GET AXL FROM P.43 VECTOR PACKET
		CB 7F	BIT 7, A	TEST SIGN BIT
		3E 00	LDA, 00	ADJUST FOR AXH
		21 97 2D	LD HL, 2D97 _H	POINT HL AT FISH PATTERN
	2DFC _H	C3 22 30	JP 3022 _H	JUMP TO FINISH ROUTINE (ANOTHER OOPS!)
	2DFF _H	FF	UNUSED BYTE	

HI-RES DEMO IM2 SET UP

2E00_H F2 2C DEMO IM2 VECTOR

2E02 _H	F3	DI	DISABLE INTERRUPT, 2E	} OUTPUT PAGE OF VECTOR	USE VECTOR 2E00H
	3E 2E	LDA, 2E			
	ED 47	LDI, A		} OUTPUT LINE OF VECTOR ← 00	
	3E 00	LDA, 00			
	D3 0D	OUT(0D _H), A			
	ED 5E	IM2		} INTERRUPT AFTER 200 SCREEN LINES	
	3E C8	LDA, C8 _H			
	D3 0F	OUT(0F _H), A			
2E11 _H	FB	EI	BEGIN INTERRUPT LINE COUNT		
	C9	RET			

TEST USER RAM 6000-7FFF_H (10 SEC PER PASS)

URM8K	2E13 _H	CD FF 24	CALL A24FF	SET COLORS, CLEAR SCREEN
		11 00 60	LD DE, 6000 _H	SET LOWER LIMIT IN DE
		21 00 80	LD HL, 8000 _H	HL = UPPER LIMIT
		18 09	JR A2E27	

TEST USER RAM 8000-FFFF_H (40 SEC PER PASS)

URM32K	2E1E _H	CD FF 24	CALL A24FF	
	2E21	11 00 80	LD DE, 8000 _H	
		21 00 00	LD HL, ZERO	
	2E27 _H	22 FB 4F	LD (4FFB _H), HL	PLACE UPPER LIMIT IN TABLE
		2B ← BOT	DECHL	HL = BANK END ADR
		E5	PUSH HL	} SET UP TOP OF STACK FOR ROUTINE @ 2882 _H
		E5	PUSH HL	
		EB ← BOT	EX DE, HL	HL = LOWER LIMIT
	2E2E _H	C3 82 28	JP A2882	JMP TO TEST THIS USER RAM

2E31_H FF FF FF

UNUSED BYTES

BEGIN HIGH RES DEMO (MOVE FISH, CRITTER AND WAGON)

HRDEM1	2E34 _H	3E 01	LDA, 1	} SET CUSTOM CHIPS TO OPERATE IN HIGH RESOLUTION MODE.
		D3 08	OUT (08 _H), A	
		31 A7 7F	LD SP, 7FA7 _H	INITIALIZE STACK POINTER
		FF	RST 38 _H	
	2E3C _H	00	INTPC	BEGIN MULTIPLE CALLS

68

	14	EMUSIC	STOP ALL SOUND	
	19	COLSET +1	SET COLORS	
2E3F _H	A4 29	COLOR TABLE		
2E41 _H	17	SET OUT +1	SET	
	CB	VERT BLK REG	VERTICAL BLANK REGISTER = 203 _D	
	9C	HOR CLR BNDRY	HORIZONTAL COLOR BOUNDARY = 1001100	
	08	INT MODE	INTERUPT MODE / BKGND PIXEL 28 _D	
	1B	FILL +1		
	00 40	START ADDRESS	CLEAR SCREEN	
	70 3F	# OF BYTES TO FILL		
	00	FILL WITH 00 _H		
	02	XINTC	END MULTIPLE CALLS	
	3E 20	LD A, 32 _D	REGY = 32 _D	
2E51 _H	32 AB 7F	LD (REGY), A		
	11 6B 00	LD DE, 107 _D	X = 107 _D	
	AF	XOR A	MR = 0	
	CD B2 29	CALL SADDR	DETERMINE SCREEN ADDRESS	PAINT LEFT SIDE OF FISH TANK
	EB	EX DE, HL	HL = SCREEN ADDRESS	
	11 01 89	LD DE, 8901 _H	E = XSIZE = 1	
	06 0F	LD B, 0F _H	D = YSIZE = 137 _D	
	CD F0 29	CALL FILL	FILL DATA = 00001111	
2E61 _H	11 D4 00	LD DE, 212 _D		
	AF	XOR A		
	CD B2 29	CALL SADDR		PAINT RIGHT SIDE OF FISH TANK
	EB	EX DE, HL		
	11 01 89	LD DE, 8901 _H		
	06 F0	LD B, F0 _H		
	CD F0 29	CALL FILL		
2E71 _H	3E A6	LD A, 166 _D		
	32 AB 7F	LD (REGY), A		
	11 6C 00	LD DE, 108 _D	X = 108 _D	
	AF	XOR A		
	CD B2 29	CALL SADDR		PAINT BOTTOM OF FISH TANK
	EB	EX DE, HL		
	11 1A 01	LD DE, 011A _H	E = XSIZE = 26 _D	
2E81 _H	06 FF	LD B, FF _H	D = YSIZE = 1	

69

2E83 _H	CD FO 29	CALL FILL			
	3E 9C	LD A, 156 _D			
	32 AB 7F	LD (REGY), A			
	11 6C 00	LD DE, 108 _D	X = 108 _D		PLACE PEBBLE
	AF	XOR A			IN BOTTOM OF
	CD B2 29	CALL SADDR			FISH TANK
2E92 _H	21 85 04	LD HL, 0485 _H	SAVE AREA		
	01 1A 0A	LD BC, 0A 1A _H	HL = ADDRESS		
	CD 04 2A	CALL RESTOR 1	C = XSIZE = 26 _D		
	3E 2A	LD A, 42 _D	B = YSIZE = 10 _D		
	32 AB 7F	LD (REGY), A	WRITE THE PEBBLES		
2EA0 _H	11 6C 00	LD DE, 108 _D			
	AF	XOR A			FILL
	CD B2 29	CALL SADDR			FISH TANK
	EB	EX DE, HL			WITH WATER
	11 1A 72	LD DE, 72 1A _H	E = XSIZE = 26 _D		
	06 55	LD B, 55 _H	D = YSIZE = 114 _D		
	CD FO 29	CALL FILL			
2EB0 _H	3E 0A	LD A, 10 _D			
	32 AB 7F	LD (REGY), A	Y = 10 _D		WRITE
	11 12 00	LD DE, 18 _D	X = 18 _D		WITH OR
	3E 10	LD A, 10 _H	OR IT		COWBOY
	21 A6 2A	LD HL, 2AA6 _H			(PATTERN 1)
	D5	PUSH DE			
	CD 3E 2A	CALL WRITP			
2EC1 _H	D1	POP DE	X = 18 _D		
	06 0A	LD B, 10 _D	B = Y = 10 _D		WRITE
	3E 10	LD A, 10 _H	OR IT		WITH OR
	21 E4 2A	LD HL, 2AE4 _H			COWBOY'S ARM
	CD 2E 2A	CALL WRITR			(PATTERN 1)
	11 2E 00	LD DE, 46 _D	X = 46 _D		
	06 09	LD B, 9 _D	Y = 9 _D		
2ED1 _H	AF	XOR A	PLOP IT		WRITE
	21 EF 2A	LD HL, 2AEF _H			WITH PLOP
	CD 2E 2A	CALL WRITR			WAGON
	3E 0A	LD A, 10 _D			(PATTERN 2)

70

2EEO _H	32 AB 7F 11 E6 00 3E 40 21 A6 2A D5 CD 3E 2A D1 06 0A 3E 50 21 E4 2A	LD (REGY), A LD DE, 230 _D LD A, 40 LD HL, 2AA6 _H PUSH DE CALL WRITP POP DE LD B, 10 _D LD A, 50 _H LD HL, 2AE4 _H	Y = 10 _D X = 230 _D FLOP PLOP IT	WRITE WITH FLOP PLOP COWBOY (PATTERN 3)
2EF1 _H	CD 2E 2A 3E 08 D3 19 3E 86 32 AB 7F 11 32 00 3E 08 21 4B 2B CD 3E 2A 11 1C 00 06 99 3E 10 21 5E 2B	CALL WRITR LD A, 08 _H OUT (19 _H), A LD A, 134 _D LD (REGY), A LD DE, 50 _D LD A, 08 _H LD HL, 2B4B _H	X = 230 _D Y = 10 _D FLOP OR IT COLOR IT GREEN Y = 134 _D X = 50 _D EXPAND PLOP IT	WRITE WITH FLOP OR COWBOY'S ARM (PATTERN 3) WRITE WITH EXPAND PLOP TREE (PATTERN 4)
2F00 _H	3E 08 21 4B 2B CD 3E 2A 11 1C 00 06 99 3E 10 21 5E 2B	CALL WRITP LD DE, 28 _D LD B, 153 _D LD A, 10 _H LD HL, 2B5E _H		WRITE WITH OR CRITTER (PATTERN 5)
2F12 _H	CD 2E 2A 3E 0C D3 19 3E 9C 32 AB 7F 11 44 00 3E 18 21 AA 2B CD 3E 2A 3E 0C D3 19 3E 13 32 AB 7F	CALL WRITR LD A, 0C OUT (19 _H), A LD A, 156 _D LD (REGY), A LD DE, 68 _D LD A, 18 _H LD HL, 2BAA _H	COLOR IT WHITE	WRITE WITH EXPAND OR CACTUS (PATTERN 6)
2F21 _H	3E 18 21 AA 2B CD 3E 2A 3E 0C D3 19 3E 13 32 AB 7F	CALL WRITP LD A, 0C _H OUT (19 _H), A LD A, 19 _D LD (REGY), A	COLOR IT MAIZE Y = 19 _D	WRITE WITH EXPAND XOR CACTUS (PATTERN 7)

71

2F32_H 11 EB 00
 3E 28
 21 AA 2B
 CD 3E 2A
 3E 0F

2F42_H 11 14 00
 3E 50
 21 A6 2A
 D5
 CD 3E 2A
 D1

2F51_H 06 0F
 3E 50
 21 E4 2A
 CD 2E 2A
 11 01 01

2F60_H 06 28
 3E 20
 21 5E 2B
 CD 2E 2A
 3E 04
 D3 19
 3E 4E

2F72_H 32 AB 7F
 11 4E 00
 3E 48
 21 4B 2B
 CD 3E 2A
 3E 0C

2F80_H D3 19
 3E 70
 32 AB 7F
 11 4F 00
 3E 58
 21 AA 2B

LD DE, 235_D X=235_D
 LD A, 28_H EXPAND XOR IT
 LD HL, 2BAA_H
 CALL WRITP

LD A, 15_D Y=15_D
 LD (REG Y), A X=20_D
 LD DE, 20_D FLOP OR IT
 LD A, 50_H
 LD HL, 2AA6_H
 PUSH DE

CALL WRITP
 POP DE X=29_D
 LD B, 0F_D Y=15_D
 LD A, 50_H FLOP OR IT

LD HL, 2AE4_H
 CALL WRITR
 LD DE, 257_D X=257_D
 LD B, 40_D Y=40_D
 LD A, 20_H XOR IT

LD HL, 2B5E_H
 CALL WRITR
 LD A, 04_H COLOR IT
 OUT (19_H), A LIGHT BLUE

LD A, 78_D
 LD (REG Y), A Y=78_D
 LD DE, 78_D X=78_D
 LD A, 48_H FLOP EXPAND
 LD HL, 2B4B_H FLOP IT

CALL WRITP
 LD A, 0C
 OUT (19_H), A COLOR IT MAIZE

LD A, 112_D
 LD (REG Y), A Y=112_D
 LD DE, 79_D X=79_D
 LD A, 58_H FLOP EXPAND
 LD HL, 2BAA_H OR IT

WRITE WITH
 FLOP OR
 COWBOY
 (PATTERN 8)

WRITE WITH
 FLOP OR
 COWBOY'S ARM
 (PATTERN 8)

WRITE WITH
 XOR
 CRITTER
 (PATTERN 9)

WRITE WITH
 FLOP EXPAND FLOP
 TREE
 (PATTERN 10)

WRITE WITH
 FLOP EXPAND OR
 CACTUS
 (PATTERN 11)

72

	CD 3E 2A	CALL WRITP		
	3E 08	LD A, 08	COLOR IT GREEN	
2F90 _H	D3 19	OUT (19 _H), A		
	3E 8E	LD A, 142 _D		
	32 AB 7F	LD (REG _D), A	Y = 142 _D	WRITE WITH FLOP EXPAND XOR TREE (PATTERN 12)
	11 4E 00	LD DE, 78 _D	X = 78 _D	
	3E 68	LD A, 68 _H	FLOP EXPAND XOR IT	
	21 4B 2B	LD HL, 2B4B _H		
	CD 3E 2A	CALL WRITP		
2FA2 _H	3E 08	LD A, 08	COLOR IT GREEN	
	D3 19	OUT (19 _H), A		
	3E 50	LD A, 80 _D		
	32 AB 7F	LD (REG _D), A	Y = 80 _D	WRITE WITH EXPAND FLOP CACTUS (PATTERN 13)
	11 22 01	LD DE, 290 _D	X = 290 _D	
	3E 08	LD A, 08	EXPAND FLOP IT	
2FB0 _H	21 AA 2B	LD HL, 2BAA _H		
	CD 3E 2A	CALL WRITP		
	3E 04	LD A, 04	COLOR IT LIGHT BLUE	
	D3 19	OUT (19 _H), A		
	3E 6E	LD A, 110 _D		
	32 AB 7F	LD (REG _D), A	Y = 110 _D	WRITE WITH EXPAND OR TREE (PATTERN 14)
	11 21 01	LD DE, 289 _D	X = 289 _D	
2FC2 _H	3E 18	LD A, 18 _H	EXPAND OR IT	
	21 4B 2B	LD HL, 2B4B _H		
	CD 3E 2A	CALL WRITP		
	3E 0C	LD A, 0C	COLOR IT MAIZE	
	D3 19	OUT (19 _H), A		
	3E 9D	LD A, 144 _D		
2FD0 _H	32 AB 7F	LD (REG _D), A	Y = 144 _D	WRITE WITH FLOP EXPAND FLOP CACTUS (PATTERN 15)
	11 19 00	LD DE, 25 _D	X = 25 _D	
	3E 48	LD A, 48 _H	FLOP EXPAND FLOP IT	
	21 AA 2B	LD HL, 2BAA _H		
	CD 3E 2A	CALL WRITP		
	FF	RST 38 _H		
	1B	FILL + 1		
2FEO _H	AB 7F	ADDRESS TO BEGIN FILLING		ZERO DATA BASES (EXCEPT RANSHT)

73

2FE2 _H	55 00	# OF BYTES TO FILL	
	00	FILL WITH ZERO	
	3E 04	LD A, 4	BLUE TEXT ON BLACK BACK- GROUND
	D3 19	OUT (19 _H), A	
	CD CD 2C	CALL DCT1	DISPLAY COUNTER 1
	CD E6 2C	CALL DTIM	DISPLAY TIME
	CD D5 2C	CALL DCT2	DISPLAY COUNTER 2
2FF2 _H	FF	RST 38 _H	INITIALIZE FISH, CRITTER AND WAGON VECTOR PACKETS
	5F	MOVE + 1	
	D8 7F	DESTINATION ADDRESS	
	27 00	# OF BYTES TO MOVE	
	6A 2C	SOURCE ADDRESS	
	CD 02 2E	CALL DIM2	START INTERRUPT ROUTINE
	CD 54 2D	CALL WAGON	INITIALIZE WAGON
	3000 _H	CALL MFISH	INITIALIZE FISH
RUN 3003 _H	CD 21 2D	CALL MWAGON	MOVE/WRITE WAGON
	21 F6 7F	LD HL, 7FF6 _H SEE P. 43	
	CB 7E	BIT 7, (HL)	IF 1PPS IS SET DISPLAY INCRE- MENTED TIME
	28 05	JR Z, RUN1	
	CB BE	RES 7, (HL)	
	CD E6 2C	CALL DTIM	
RUN1 3012 _H	CD 88 30	CALL GO	MOVE FISH
	CD A1 2C	CALL CRITTER	DO CRITTER
3018 _H	18 E9	JR RUN	LOOP BACK TO CONTINUE PROGRAM

WRITE LOOP (FOR TROUBLESHOOTING)

301A _H	3E 66	LD A, 0110 0110	WRITE WITH 66 _H
	21 96 46	LD HL, 0100 0110 1001 0110	
301F _H	77	LD (HL), A	LOOP WRITE @ 4696 _H NON STOP
3020 _H	18 FD	JRA 301F	

CHECK FOR FLOP (CONTD) (CKFLOP @ 2DF1_H)

3022 _H	28 05	JR Z, CK2	IF ΔXL IS NEGATIVE, USE FLOPPED FISH PATTERN
	3E FF	LD A, FF _H	ADJUST FOR ΔX _H
	21 A9 2D	LD HL, 2DA9 _H	POINT HL AT FLOPPED PATTERN
	32 DC 7F	LD (7FDC _H), A	SET ΔX _H P. 43
	22 B3 7F	LD (7FB3 _H), HL	SET UP FISH PAT ADR

-3 0000 0011
1111 1100
#1
1101

CK2

74

EI
3030_H C9POP HL
RET

WRITE ONE FISH

WRITE1 3031_H C5
CD E5 2D
CD F1 2D
2A B3 7F
DD 21 D8 7F
CD 14 2A3042_H C1
78
FF
5B
C8 2D
EB3A DE 7F
77
23
3A E3 7F
77
233051_H 3A DB 7F
77
23
3A E0 7F
77
C9PUSH BC
CALL TIMERCK
CALL CKFLOP
LD HL, 7FB3_H
LD IX, 7FD8_H
CALL VWRIT1POP BC
LD A, B
RST 38H
INDEW + 1
L, H REG (INDEX TABLE)
ADDRESS - 2EX DE, HL
LD A, (7FDE)
LD (HL), AINC HL
LD A, (7FE3_H)
LD (HL), AINC HL
LD A, (7FDB_H)
LD (HL), AINC HL
LD A, (7FE0_H)
LD (HL), A

RET

SAVE FISH # IN B
CHECK ΔTIMER TO SEE
IF A NEW ΔXL, ΔYL IS REQ'D,
GET CORRECT FISH ORIENTATION
SET ΔXH ACCORDINGLYSEE P.42
HL = FISH PAT ADR
VECT
IX = PACKET ADR
WRITE THE FISHPUT FISH # BACK IN B
PUT INDEX #
IN A

GET

"5 BYTE VECTOR PACKET INFO"
ADDRESS FOR THIS FISHSEE P.42
7FB5 - 7FD3

PUT THAT ADDRESS IN HL

GET X_H FROM VECTOR PACKET
PUT IT IN "5 BYTE PACKET"GET Y_H

GET ΔXL

GET ΔYL

DO THE FISH

FOR EACH FISH [FIRST, INITIALIZE X_H, Y_H, ΔXL, ΔYL AND PUT THEM IN FISH VECTOR PACKET
INITIALIZE ALSO ΔTIMERMFISH 305D_H 06 07
MFISH1 C5
3060_H 3E 61
CD 5F 2DLDB, 7 B = FISH # TO WRITE
PUSH BC SAVE FISH #
LD A, 97D
CALL RANGED
RANDOMIZE X_H
FROM 0 - 96D

75

3065 _H	C6 6C	ADD A, 108 _D	ADD RELATIVE (FISH TANK) SEE P. 42
	32 DE 7F	LD (7FDE), A	PUT X _H IN VECTOR PACKET SEE P. 43
	3E 6C	LD A, 108 _D	RANDOMIZE Y _H FROM 0-107 _D
	CD 5F 2D	CALL RANGED	PUT Y _H IN VECTOR PACKET
	C6 2A	ADD A, 42 _D	
3071 _H	32 E3 7F	LD (7FE3 _H), A	GET A ± ΔXL AND ± ΔYL.
	CD D8 2D	CALL DELTXY	PUT THEM IN VECTOR PACKET
	C1	POP BC	B = FISH # (INDEX)
	78	LD A, B	GET
	FF	RST 38 _H	"5 BYTE VP INFO" ADDRESS
	5B	INDEX W + 1	PUT IT IN DE
	C8 2D	L, H REG (INDEX TABLE ADDRESS - 2)	
	21 04 00	LD HL, 0004	POINT HL AT ΔTIMER FOR THIS FISH
3080 _H	19	ADD HL, DE	
	36 80	LD (HL), 80	INITIALIZE DELTA TIMER = 80 _H
	CD 31 30	CALL WRITE1	PUT THIS FISH ON THE SCREEN
	10 D7	DJNZ MFISH1	LOOP BACK AND DO NEXT FISH
GO 3088 _H	06 07	LD B, 07	B = FISH # AGAIN
G1	78	LD A, B	
	FF	RST 38 _H	GET
	5B	INDEX W + 1	"5 BYTE VECTOR PACKET INFO" ADDRESS FOR THIS FISH SEE P. 42 7FB5 - 7FD3
	C8 2D	L, H REG (INDEX TABLE ADDRESS - 2)	
	EB	EX DE, HL	DE NOW = THIS ADDRESS
3090 _H	7E	LD A, (HL)	GET X _H , PUT IT IN VECTOR PACKET AND REGISTER E SEE P. 43
	32 DE 7F	LD (7FDE), A	
	5F	LD E, A	GET Y _H , PUT IT IN VP AND REG D
	23	INC HL	
	7E	LD A, (HL)	
	32 E3 7F	LD (7FE3 _H), A	
	57	LD D, A	
	23	INC HL	GET ΔXL, PUT IT IN VP SET ΔXH AND "FISH PATTERN ADDR" TO USE ACCORDINGLY.
	7E	LD A, (HL)	
	32 DB 7F	LD (7FDB _H), A	
30A0 _H	CD F1 2D	CALL CKFLOP	GET ΔYL, PUT IT IN VP
	23	INC HL	
	7E	LD A, (HL)	

76

	30A5 _H	32 E0 7F	LD (7FE0 _H), A	IF ΔY_L IS POSITIVE, SET $\Delta Y_H = 0$
		CB 7F	BIT 7, A	
		3E FF	LD A, FF _H	IF ΔY_L IS NEGATIVE, SET $\Delta Y_H = FF$
		20 01	JRNZ, G2	
		AF	XOR A	
G2		32 E1 7F	LD (7FE1 _H), A	
	30B2 _H	23	INC HL	SAVE Δ TIMER POINTER (ADDRESS)
		E5	PUSH HL	
		3E 03	LD A, 03	RESTORE TIME BASE IN VP, SEE P. 42
		32 DA 7F	LD (7FDA _H), A	
		FF	RST 38 _H	
		3F	VECT + 1	UPDATE VECTOR PACKET FOR NEXT FISH WRITE
		D8 7F	VECTOR PACKET ADDR	
		BB 2D	LIMITS TABLE ADDRESS	
		C5	PUSH BC	SAVE FISH #
	30C0 _H	76	HALT	WAIT UNTIL IM2 ROUTINE IS DONE (LENGTHEN FISH WRITE "ON TIME")
		42	LDB, D	
		16 00	LDD, 00	
		3E 20	LD A, 20	A = MAGIC REG XOR IT
		2A B3 7F	LD HL, (7FB3 _H)	HL = PATTERN ADDRESS
		CD 2E 2A	CALL WRITR	BLANK THE FISH
		C1	POP BC	B = FISH #
		E1	POP HL	RETRIEVE Δ TIMER ADDRESS
		CD 31 30	CALL WRITE1	WRITE NEW FISH ON SCREEN
	30D1 _H	10 B7	DJNZ G1	GO BACK TO PROCESS BLANK/WRITE NEXT FISH
		C9	RET	
OOPS!			FINISH FOR TIMERCK ROUTINE AT 2DE5 _H	
	30D4 _H	E1	POP HL	HL POINTS TO A TIMER
		77	LD (HL), A	LOAD NEW VECTOR Δ TIMER IN PACKET
		CD D8 2D	CALL DELTXY	GET A NEW $\pm \Delta X_L$ AND $\pm \Delta Y_L$. PUT THEM IN VECTOR PACKET
	30D9 _H	C9	RET	

NOTE 1 SAVE THE ENVIRONMENT. REG HL IS Clobbered BY
DISPLAY CHAR ROUTINES DLCHR(2026H) AND DRCHR(2035H)

77

COMMERCIAL

BALCHER INTERRUPT ROUTINE (VECTOR @ 2014H)

NINT	30DAH F3	DI	17161514 ← INPUT PORT
	E5	PUSH HL	BIT 0 1 2 3 4
	08	EX AF, AF'	1 5 6 7 8
	CHECK FOR PRESSED KEY (USES ON REG A)		2 9 A B C KEY PAD LAY OUT
	DB15	IN A, (COL1)	3 D E F O
	A7	AND A	4 Q R S G H I
	30E0H 2031	JR NZ, COL1P	5 J K L M
	DB16	IN A, (COL2)	3 2 1 0 ← COLUMN
	A7	AND A	WAS A KEY PRESSED IN COLUMN 2?
	2048	JR NZ, COL2P	IF YES, JMP TO "COL2 PRESSED" (COL2P).
	DB17	IN A, (COL3)	WAS A KEY PRESSED IN COLUMN 3?
	A7	AND A	IF YES, JMP TO "COL3 PRESSED" (COL3P)
	200C	JR NZ, COL3P	
	DB14	IN A, (COL0)	WAS A KEY PRESSED IN COLUMN 0?
	A7	AND A	IF NO, JMP TO "UPDATE RUN TESTS CLOCK" (UCLK)
	2871	JR Z, UCLK	(NO KEYS AT ALL WERE PRESSED). IF YES, CONTINUE. CHECK COL0 (COL0P).
COL0P	30F1H FE01	CP 1	WAS KEY 4 PRESSED?
	CA2726	JP Z, A2627	IF YES, JMP TO "OUTPUT PORT WRITE LOOP"
	186A	JR UCLK	KEY 4 IN COLUMN 0 WAS NOT PRESSED. JMP TO "UPDATE RUN TESTS CLOCK" (UCLK)
COL3P	FE01	CP 1	WAS KEY 1 PRESSED?
	CA D925	JP Z, A25D9	IF YES, JMP TO MEMORY READ LOOP
	FE02	CP 2	WAS KEY 5 PRESSED?
	CA 4F26	JP Z, A264F	IF YES, JMP TO MEMORY WRITE/READ LOOP
	3102H FE04	CP 4	WAS KEY 9 PRESSED?
	CA 8F37	JP Z, TITLE	IF YES, JMP TO LOW-RES DEMO 1 (TITLE PAGE)
	FE08	CP 8	WAS KEY D PRESSED?
	CA 132E	JP Z, VRM8K	IF YES, JMP TO TEST USER RAM 6000-7FFF
	FE10	CP 16	WAS Q'S KEY PRESSED?
	284C	JR Z, RST45	IF YES, JMP TO ZERO Q'S FLAG
	3110H C3 F7 37	JP PSTRT	(NONSTOP BALCHER TESTS CE KEY MUST HAVE BEEN PRESSED. JMP TO PIXEL STRINGER)
COL1P	FE01	CP 1	WAS KEY 3 PRESSED?
	3115H CA 11 26	JP Z, A2611	IF YES, JMP TO INPUT PORT READ LOOP

78

3118H	FE 02	CP 2	WAS KEY 7 PRESSED?	
	CA 39 36	JPZ, A3639	IF YES, JMP TO CRASH TEST DEMO	
	FE 04	CP 4	WAS KEY B PRESSED?	JMP IF KEYS
	CA 58 3A	JPZ, TDUAL	IF YES, JMP TO TEST DUAL DISPLAY	3, 7, B, F, H OR L
			DISPLAY TEST PATTERNS	
3122H	FE 08	CP 8	WAS KEY F PRESSED?	
	CA 1E 24	JPZ, THRES	IF YES, JMP TO TEST HI-RES SCREEN RAM (THR)	IN COLUMN 1
	FE 10	CP 16D	WAS KEY H PRESSED?	
	CA 34 2E	JPZ, HRDEM1	IF YES, JMP TO HI-RES DEMO 1 (FISH IN TANK)	WERE PRESSED
	C3 00 28	JP ENTER	KEY L MUST HAVE BEEN PRESSED. JMP TO TEST USER RAM (TUR).	

COL2P	FE 01	CP 1	WAS KEY 2 PRESSED?	
3131H	CA EA 25	JPZ, A25EA	IF YES, JMP TO MEMORY WRITE LOOP	
	FE 02	CP 2	WAS KEY 6 PRESSED?	JMP IF KEYS
	CA 73 26	JPZ, A2673	IF YES, JMP TO MONITOR DEVICE INPUT PORTS	2, 6, A, E, G OR H
	FE 04	CP 4	WAS KEY A PRESSED?	
	CA D0 3E	JPZ, A3ED0	IF YES, JMP TO NEW (CHECKMATE (LD2))	IN COLUMN 2
	FE 08	CP 8	WAS KEY E PRESSED?	WERE
			IF YES, JMP TO TEST USER RAM 8000-FFFFH	PRESSED.
3140H	CA 1E 2E	JPZ, URM32K	WAS KEY G PRESSED?	
	FE 10	CP 16D	IF YES, JMP TO HI-RES TEXTURED TEST PATTERN (HD2)	
	CA F7 3A	JPZ, H RTP	KEY K MUST HAVE BEEN PRESSED. JMP TO HI-RES DEMO 4 (HD4)	
3148H	C3 55 31	JPA3155		
314BH	FF			

READ LOOP (FOR TROUBLE SHOOTING)

314CH	3E 66	LD A, 0110 0110	WRITE 66H
	21 96 46	LD HL, 0100 0110 1001 0110	AT 4696H
3151H	77	LD (HL), A	
3152H	7E	LD A, (HL)	LOOP READ AT 4696H
	18 FD	JRA 3152	NONSTOP

RUN LOW-RES MENU ON HI-RES SCREEN MAP

3155H	3E 01	LD A, 01
	D3 08	OUT (CONSM), A
3159H	C3 69 0C	JP MENU

RESET Q/S FLAG (KEY WAS PRESSED)

RSTQS	315C _H	D9	EXX	C' = Q/S FLAG (BIT 7), REP TESTS CTR (BITS 2-0)
		79	LD A, C	Q/S KEY WAS PRESSED
		E6 0F	AND 0FH	RESET BIT 7 IN C'
	3160 _H	4F	LD C, A	BIT 7 = 1 QUIT REPETITIVE TESTS AFTER 4 PASSES
		D9	EXX	BIT 7 = 0 NON STOP REPETITIVE TESTS
			SAVE ALTERNATE REGISTERS	

UPDATE "RUN TESTS" CLOCK

UPDATE 1/60 SEC COUNTER

UCLK	3162 _H	D9	EXX	C' = Q/S FLAG D' = REP TEST CTR, D' = 1/60 SEC CTR, E' = DECIMAL SECONDS H' = DECIMAL MINUTES, L' = HOURS = 0-9 MAX
		14	INC D	INCREMENT 1/60 SEC CTR
		7A	LDA, D	IF 1/60 SEC CTR < 60, CLOCK UPDATE IS FINISHED. JMP TO EXIT INTERRUPT ROUTINE.
		FE 3C	CP 60 _D	
		20 60	JRNZ, ENVIRO	IF = 60, CONTINUE
		16 00	LD D, 0	RESET 1/60 SEC CTR = 0

CHECK Q/S FLAG

CHKQ5	316B _H	79	LDA, C	A = Q/S FLAG REP TESTS CTR	CHECK Q/S FLAG AND REP TESTS CTR
		FE 84	CP 84 _H		IF FLAG IS SET AND CTR = 4, TESTING IS FINISHED.
		20 0A	JR NZ, CHKQ51		DISPLAY E1 = TESTING PASSED OK (WILL CONTINUE TO OUTPUT E1) EVERY SEC
	3170 _H	3E 79	LD A, 0111 1001	E	
		D3 EF	OUT (LDSPLY), A		OTHERWISE, JMP TO CHECK Q/S FLAG AGAIN.
		3E 06	LDA, 0000 0110	I	
		D3 FF	OUT (RDSPLY), A		
	BOY	18 31	JR USEC		JMP TO CONTINUE "RUN TESTS" CLOCK UPDATE. ← WHY BOTHER?
CHKQ51		CB 79	BIT 7, C		IS Q/S FLAG RESET = 0?
		21 7C	JR Z, HRMIN		IF YES, JMP TO DISPLAY HR/MIN FOR NON STOP TESTING.
		28 0E			IF NO, DISPLAY SECS ONLY FOR 4 PASS TESTING

80

DISPLAY SECONDS ONLY FOR 4 PASS TEST

DSEC	317EH	7B ^{←BOY}	LD A, E	A = SECS
		D9	EXX	SAVE ALTERNATE REGISTERS
	3180H	CD 26 20	CALL DLCHAR	DISPLAY LEFT (MSD) DIGIT
		D9	EXX	GET ALTERNATE REGISTERS
		7B ^{←BOY}	LD A, E	A = SECS AGAIN
		D9	EXX	SAVE ALTERNATE REGISTERS
		CD 35 20	CALL DRCHAR	DISPLAY RIGHT (LSD) DIGIT
		D9	EXX	GET ALTERNATE REGISTERS
		18 1F	JR USEC	JMP TO CONTINUE "RUN TESTS" CLOCK

ALTERNATE HRS/MINUTES FOR NONSTOP TESTING

HRMIN	318CH	CB 43	BIT 0, E	IF SECS IS ODD, JMP TO DISPLAY DECIMAL MINUTES
		20 0F	JR NZ, DMIN	

DISPLAY DECIMAL HOURS

DHR	3190H	7D ^{←DOG}	LD A, L	A = HOURS (0-9h MAX)
		D9	EXX	SAVE ALTERNATE REGISTERS
		21 16 20	LD HL, CTBL	NOT DLCHR
		CD 2F 20	CALL DLCHR	OUTPUT HRS TO LEFT DISPLAY
		3E 74	LD A, 0111 0100	OUTPUT "HOURS" NOTATION TO
		D3 FF	OUT (RDSPLY), A	RIGHT DISPLAY
		D9	EXX	GET ALTERNATE REGISTERS
	319DH	18 0C	JR USEC	JMP TO CONTINUE "RUN TESTS" CLOCK

DISPLAY DECIMAL MINUTES

DMIN	319F	7C	LDA, H	A = DECIMAL MINUTES (0-59 _H)
	31A0 _H	D9	EXX	SAVE ALTERNATE REGISTERS
		CD 26 20	CALL DLCHAR	DISPLAY LEFT (MSD) DIGIT
		D9	EXX	GET ALTERNATE REGISTERS
		7C	LDA, H	A = DECIMAL MINUTES AGAIN
		D9	EXX	
		CD 35 20	CALL DRCHAR	DISPLAY RIGHT (LSD) DIGIT
		D9	EXX	

UPDATE DECIMAL SECONDS

USEC	31AB _H	7B	LDA, E	A = SECONDS
		CG 01	ADD A, 1	INCREMENT SECS
		27	DAA	ADJUST CARRY OVER TO MSD
		5F	LDE, A	SAVE UPDATE FOR E'
	31B0 _H	FE 60	CP 60 _H	IS SEC < 60?
		20 15	JR NZ, ENVIRO	IF YES, YOU'RE DONE WITH UPDATE.
		1E 00	LDE, 0	JMP TO EXIT INTERRUPT ROUTINE
				IF NO, ZERO SECS AND CONTINUE UPDATE

UPDATE DECIMAL MINUTES

UMIN	31B6 _H	7C	LDA, H	A = MINUTES
		CG 01	ADD A, 1	INCREMENT MINUTES
		27	DAA	ADJUST CARRY OVER TO MSD
		67	LDH, A	SAVE UPDATE FOR H'
		FE 60	CP 60 _H	IS MINUTES < 60?
		20 0A	JR NZ, ENVIRO	IF YES, YOU'RE DONE WITH UPDATE.
		26 00	LDH, 0	JMP TO EXIT INTERRUPT ROUTINE.
				IF NO, ZERO MINUTES AND CONTINUE TO UPDATE HOURS.

82

UPDATE HOURS

31C1H	2C	INCL	INCREMENT HOURS HOURS = 0-9 MAX IF HOURS = 10, ZERO IT
	7D ← DOG	LDA, L	
	FE0A	CP10D	
	20 02	JRNZ, ENVIRO	
	2E 00	LDL, 0	

EXIT INTERRUPT

ENVIRO	31C9H	D9	EXX	SAVE ALTERNATE REG	RESTORE THE ENVIRONMENT
		08	EXAF, AF		
		EI	POP HL		
		FB ← BOY	EI		
	31CDH	C9	RET		ENABLE INTERRUPTS

NOTE = MAXIMUM "RUN TESTS" CLOCK TIME IS 9:59:59

REPORT HI-RES SCREEN RAM ERROR (CONTINUED FROM 299H)

REPORT A	31CEH	C3CC27	JP DELAY	
REPORT B	31D1H	3E3F	LDA, 3F	
		D3EF	OUT (LDSPY), A	
		3E66	LD A, 0110 0110	4
		D3FF	OUT (RDSPLY), A	
		21DF31	LD HL, REPORT C	HL = CONTINUE ADDRESS
		C3CC27	JP DELAY	
REPORT C	31DFH	D9	EXX	HL = BANK 4 ERROR BYTE
	31E0H	7C	LDA, H	
		D9	EXX	
		DD 21E9 31	LD IX, REPORT D	IX = CONTINUE ADDRESS
		C3DD27	JP DSPY	
REPORT D	31E9H	21EF31	LD HL, REPORT E	HL = CONTINUE ADR
		C3CC27	JP DELAY	
REPORT E	31EFH	AF	XOR A	
	31F0H	D3EE	OUT (LDSPY), A	
		D3FF	OUT (RDSPLY), A	
REP	31F4H	215332	LD HL, REPORT E	HL = CONTINUE ADDRESS
	31F7H	C3CC27	JP DELAY	JMP TO DELAY

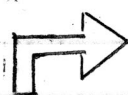
DISPLAY = 04 (HI-RES BANK 4)

DISPLAY ERROR BYTE FOR HI-RES BANK 4

BLANK DISPLAY

0.9 SEC DELAY

NOTE: THIS ROUTINE EXTENSION IS AT 3253H

CODING AT
3253H
(PAGE 85)213429
C3CC27LD HL, REPORT
JP DELAY

HL = LOOP BACK ADDRESS

ANOTHER
0.9 SEC DELAY
AND LOOP BACK
TO REPEAT DISPLAY
ERROR SEQUENCE

Set Screen 3 COLOR TABLE

CLRTBL 31FA _H	AC	GREEN	PIXEL 11	COLOR REG 7
86		YELLOW	10	LEFT
07		WHITE	01	COLORS
00		BLACK	00	
CD ^{DOG}		CYAN (BRIGHT BLUE)	11	
60		MAROON	10	RIGHT
3200 _H 63		ORANGE	101	COLORS
F9		BLUE	00	

Set Screen 3

SetScreen 3202 _H	AF	XOR A	A=0	
	01 18 08	LD BC, 0818 _H		STOP ALL SOUND
	^{DOG} ED 79	OUT(C), A		
	10 FC	DJNZ-2		
	3E CC	LDA, 204 _D		SET VERTICAL BLANK REGISTER To 204 _D
	^{DOG} D3 0A	OUT(0A _H), A		SET HORIZONTAL COLOR BOUNDARY To 20 _D SPLIT SCREEN
	3E 14	LDA, 20 _D		
3210 _H	D3 09	OUT(09), A		
	21 FA 31	LD HL, CLRTBL		SET ALL COLORS VIA OUTPUT PORT 0B _H
	01 0B 08	LD BC, 080B _H		
	^{DOG} ED B3	OTIR		
MFILL	21 00 40	LD HL, 4000 _H		102 LINES = 34 LINES
	01 50 05	LD BC, 0550 _H		FILL 1/3 OF SCREEN
	3220 _H 3E AA	LDA, 1010 1010		34 x 40 = 1360 _D = 0550 _H BYTES WITH COLOR AA _H
MFILL 1	3222 _H 77	LD (HL), A		COLORS L YEL R MAROON
	ED A1	CPI	A ← (HL) HL ← HL+1 BC ← BC-1	
	EA 22 32	JP PE, MFILL 1		
	01 A8 02	LD BC, 02A8 _H		FILL 17 LINES WITH 00 _H BELOW TOP FILL
	3E 00	LDA, 00 _H (IN CASE OF CHANGE)	NO XOR A	17 x 40 = 680 _D = 02A8 _H BYTES
MFILL 2	322D _H 77	LD (HL), A		COLORS L BLK R BLW

NOTE: FOR USE WITH HI-RES SCREEN RAM ERROR SEQUENCE ROUTINE
SEE PAGE 83.

85

DOG

322EH EDA1 DOG CPI
3230H EA 2D 32 JP PE, Mfill2
3E 55 LDA, 55H
3200 40 LD(4000H), A
3227 40 LD(4027H), A
32C8 4F LD(4FC8H), A
32EF 4F LD(4FEFH), A

WRITE (FILL) TO 4 CORNER BYTES

4000H 4027H

4FC8H 4FEFH

L WHT COLORS R MAGENTA

3241H DB 15 IN A, (COL 1)
A7 AND A

IF KEY IN COLUMN 1 IS
PRESSED, JUMP TO WRITE
AND READ RAM BYTES.

C2 14 33 JR NZ, CHKCOL 1

DB 17 IN A, (COL 3)

SCAN COLUMN 3

CB 6F BIT 5, A

IF CE KEY IS PRESSED,
HALT SET SCREEN 3
(DON'T WANT TIGHT READ LOOP)

28 01

JR Z, A324E

76

HALT

324EH A7 AND A

IF ANY OF TOP 5 KEYS
IN COL 3 ARE PRESSED,
JMP TO PIXEL WRITE ROUTINES

20 24 JR NZ, CHKCOL 3

USE TIGHT LOOP TO SCAN COL 1 AND COL 3

3251H 18 EE JRA 3241H LOOP BACK TO 3241H

BLANK DISPLAY EXTENSION TO 1.9 SEC (SEE NOTE ABOVE)

REPORT 3253H 21 34 29 LD HL, REPORT HL = LOOP BACK ADR

0.9 SEC DELAY
AND LOOP BACK TO
REPEAT DISPLAY
ERROR SEQUENCE

3256H C3 CC 27 JP DELAY

PIXEL BYTES TO WRITE (PIXEL TABLES)

PIXEL 1 TABLE

3259H 00 00 00 00 00 PASS 1

04 00 00 01 00 PASS 2

08 00 00 10 00 PASS 3

0C 00 00 11 00 PASS 4

PIXEL 3 TABLE

325DH 00 00 00 00 00 PASS 1

40 01 00 00 00 2

80 10 00 00 00 3

3260H C0 11 00 00 00 4

PIXEL 0 TABLE

3261 _H	00	00 00 00 00	PASS 1
	01	00 00 00 01	2
	02	00 00 00 10	3
	03	00 00 00 11	4

PIXEL 2 TABLE

3265 _H	00	00 00 00 00	PASS 1
	10	00 01 00 00	2
	20	00 10 00 00	3
	30	00 11 00 00	4

FILL SCREEN TABLE

3269 _H	00		PASS 1
	55	0101 0101	2
	AA	1010 1010	3
HL → 326C _H	FF	1111 1111	4

SELECT COLUMN 3 PIXEL TABLE ENTER WITH A = COL3 STATUS

326D _H	59	← DOG	PIXEL 1 BYTE TABLE (LOW ORDER)
	5D		PIXEL 3
	FF		UNUSED BYTE

3270 _H	61		PIXEL 0 BYTE TABLE (LO)
	FF FF FF		UNUSED BYTES
	65		PIXEL 2 BYTE TABLE (LO)

CHKCOL3	3275 _H	E6 IF	AND 0001 1111	MASK OUT COL3 BIT5 (MAY NOT NEED THIS INSTRUCTION)
		21 6C 32	LD HL, A326C	HL = LOW ORDER BYTE TABLE - 1
		18 01	JR A327D	JUMP TO NEXT INSTRUCTION
		69		PIXEL FILL BYTE TABLE (LO)

327D _H	85	ADD A, L	] SET L AT LOW ORDER ADDRESS OF PIXEL COLOR (PASS) TABLE
	6F	LD L, A	
327F _H	6E	LD L, (HL)	

AND FALL INTO WRITE PIXEL STRIPES

ENTER WITH L = LOW ORDER BYTE OF PIXEL BYTE COLOR TABLE

87

	3280 _H	16 60	LDD, 60 _H	D = INITIAL DELAY 1st LOOP (COUNTER VARIES 1 TO 4)
WSTRIPE		06 04	LDB, 4	B = INITIAL PASS LOOP COUNTER
		0E 01	LDC, 1	C = INITIAL PASS DISPLAY NUMBER
		79	LDA, C	A = PASS DISPLAY NUMBER (READ FOR DISPLAY ROUTINE)
	DOG →	D9	EXX	SAVE L' = PIXEL TABLE LO BYTE B' = PASS LOOP CTR C' = PASS DISPLAY NUMBER D' = DELAY 1st LOOP CTR
		DD 21 8F 32	LDIX, WSTR1PE 1	IX = RETURN (CONTINUE) ADR
		C3 DD 27	JP DSPLY	OUTPUT PASS NUMBER TO DISPLAY
WSTR1PE 1	328F	D9	EXX	RESTORE L = PIXEL TABLE LO BYTE B = PASS LOOP CTR C = PASS DISPLAY NUMBER D = DELAY 1st LOOP CTR
	3290 _H	7D ← DOG	LDA, L	A = PIXEL TABLE LO BYTE
		D9	EXX	SAVE L' = PIXEL TABLE LO BYTE B' = PASS LOOP CTR C' = PASS DISPLAY NUMBER D' = DELAY 1st LOOP CTR
		6F	LDL, A	L = PIXEL TABLE LO BYTE *
		26 32	LDH, 32 _H	H = ↓ HO BYTE *
WSTR1PE 2		7E	LDA, (HL)	A = PIXEL BYTE TO WRITE
		11 00 40	LD DE, 4000 _H	DE = SCREEN ADR
		0E 66	LDC, 102 _D	C = "LINES TO WRITE" COUNTER *
WLINE 1		06 28	LDB, 40 _D	B = "BYTES PER LINE" CTR
WLINE 2		12	LD (DE), A	WRITE A BYTE
		13	INC DE	BUMP SCR ADR
		10 FC	DJNZ, WLINE 2	WRITE ONE LINE *
	32A1 _H	47	LDB, A	SAVE "PIXEL BYTE TO WRITE" IN B *

* THESE PARAMETERS HL = TABLE ADDRESS FOR "PIXEL BYTE TO WRITE"
DE = LAST SCREEN ADDRESS
B = SAVED "PIXEL BYTE TO WRITE" FOR NEXT LINE
C = LINES LEFT TO WRITE WRITTEN

ARE OUTSIDE THE PARAMETERS SAVED IN THE ALTERNATE REGISTERS
L' = PIXEL TABLE LO BYTE
B' = PASS LOOP CTR
C' = PASS DISPLAY NUMBER
D' = DELAY 1st LOOP CTR

BEGIN DELAY BEFORE WRITING NEXT LINE

DELAY	32A2 _H	D9 ^{← DOG}	EXX	PRIOR TO EXX D' = DELAY 1ST LOOP CTR	A = 1ST LOOP CTR
		7A	LDA, D	PUT IT IN A	
		D9	EXX		
DELAY 1		08	EX AF, AF'		SAVE IT IN A' COL 2 KEY PRESSED? IF NO, SKIP USER SET NEW WRITE SPEED
		DB 16	IN A (COL 2)		
		E6 3F	AND 0011 1111	ISOLATE COL BITS 0-5	
		28 13 ^{← DOG}	JR Z, DELAY 5		
		DD 21 DE 32	LD IX, SPEED T		POINT IX AT SPEED TABLE
DELAY 2	32B0 _H	CB IF ^{← BOY}	RR A	7 → 0 → CF	ROTATE COL STATUS RIGHT
		38 04	JRC, DELAY 3		ROTATE COL STATUS TO POINT IX AT RESPECTIVE SPEED
		2 DOGS → DD 23	INC IX	POINT IX TO NEXT SPEED	
		18 F8	JR DELAY 2		
DELAY 3		DD 7E 00	LDA, (IX)		A = SELECTED WRITE SPEED
		D9	EXX		SAVE NEW SPEED (DELAY 1ST LOOP CTR) IN D'
		57	LDD, A		
		D9	EXX		
DELAY 4		08	EX AF, AF'		SAVE 1ST LOOP CTR IN A'
DELAY 5		3E FF	LDA, FF _H		A = 2ND LOOP CTR
DELAY 6	32C1 _H	3D ^{← DOG}	DEC A		2ND DELAY LOOP
		20 FD	JR NZ, DELAY 6		
		08	EX AF, AF'		A = 1ST LOOP CTR AGAIN
		3D ^{← DOG}	DEC A		DECREMENT 1ST LOOP CTR
		20 FG	JR NZ, DELAY 4		LOOP BACK IF CTR ≠ 0

CONTINUE WITH A=0

END DELAY BEFORE WRITING NEXT LINE

	0D ^{← DOG}	DEC C	DECREMENT " LINES TO WRITE " CTR
	78	LDA, B	A = " PIXEL BYTE TO WRITE "
	20 CF	JR NZ, WLINE 1	WRITE ANOTHER LINE? LOOP BACK IF SO
	D9	EXX	RESTORE
			L = PIXEL TABLE LO BYTE B = PASS LOOP CTR C = PASS DISPLAY NUMBER D = DELAY 1ST LOOP CTR
32CD _H	0C	INCC	INCREMENT PASS NUMBER

89

32CE_H 10 02
32D0_H 18 B0 ← B0₄

DJNZ, NXPASS
JR WSTRIP2

FINISHED WITH 4 PASSES?
IF NO, JMP TO NXPASS TO WRITE
NEXT PASS.
IF YES, REPEAT 4 PASSES AGAIN.

NXPASS

79
D9
DD 21 DB 32
C3 DD 27

LDA, C
EXX
LD IX, NXPASS 1
JP DSPLY

A = PASS NUMBER
L' = PIXEL TABLE LO BYTE, B' = PASS LOOP CTR
C' = PASS DISPLAY NUMBER, D' = DELAY 1ST LOOP CTR
IX = CONTINUE (RETURN) ADDRESS
HL POINTS TO "PIXEL BYTE TO WRITE"
AND IS NOT Clobbered BY ROUTINE DSPLY

NXPASS1 32DB 23
32DC_H 18 B7

INC HL POINT HL AT NEXT "PIXEL BYTE TO WRITE"
JR WSTRIP2 LOOP BACK TO WRITE NEXT PASS

SPEED (DELAY LOOP CTR) TABLE.

NOTE: TABLE CONTAINS TIME DELAY LOOP COUNTERS.

THE LOWER THE COUNTER, THE FASTER THE BYTE LINES WILL
BE WRITTEN.

LOOP COUNTER COL 2
KEY

SPEEDT 32DE_H 10 ↑
32E0_H 20 MS
32E0_H 35 8
47 5
60 2
32E3_H A0 0

↑ WRITE
SPEED
INCREASES

LOOP COUNTER IS
INITIALIZED TO 60_H
AT WRITE PIXEL STRIPES
(32E0_H)

CONTINUE

PROGRAM TEST DUAL DISPLAY / DISPLAY TEST PATTERNS

TDUALC	32E4H	C5	PUSH BC	SAVE DISPLAY COUNTER
		79	LD A,C	
		E6 0F	AND 00001111	IF LO NIBBLE IN
		20 05	JRNZ, TDUAL1	DISPLAY COUNTER=0
		CDA13B	CALL PATT1.	DISPLAY PATTERN 1
		18 10	JR TDUAL3	
TDUAL1		FE 05	CP 5	IF LO NIBBLE IN
	32F1H	20 05	JRNZ, TDUAL2	DISPLAY COUNTER=5,
		CD AA 3B	CALL PATT2	DISPLAY PATTERN 2
		18 07	JR TDUAL3	
TDUAL2		FE 0A	CP 0A	IF LO NIBBLE IN
		20 03	JRNZ, TDUAL3	DISPLAY COUNTER=A,
		CD B3 3B	CALL PATT3	DISPLAY PATTERN 3
TDUAL3		C1	POP BC	C= DISPLAY COUNTER
	3300H	79	LD A,C	
		CD 26 20	CALL DLCHR	DISPLAY L DIGIT
		79	LD A,C	
		CD 35 20	CALL DRCHR	DISPLAY R DIGIT
	3308H	21 D8 39	LD HL, 3908	HL= CONTINUE ADR
	330BH	CD 5 39	JP TDUALC	

WRITE/READ 6 RAM BYTES USING KEYPAD COL1

PROGRAM AT 3314H

LOW ORDER RAM ADDRESS TABLE

LORMAD	330EH	00
		14
	3310H	27
		C8
		DC
	3313H	EF

91

EXAMINE
6 RAM BYTES

4000 4014 4027

4FC8 4FDC 4FEF

ENTER WITH A=COL1 STATUS

CHKCOL1	3314 _H	21 0D 33	LD HL, LORMAD-1	POINT HL AT LO RAM ADR TABLE-1
CHKCL1A		CB IF ^{DOGS}	RRA ^{POINT TO NEXT LO RAM ADR} $\rightarrow 0 \rightarrow CF$	CHECK COL1 STATUS POINT HL AT SELECTED LO RAM ADR
		23	INC HL	
		30 FC	JRNC, CHKCL1A	START WITH H0 RAM ADR 40H.
		16 4D	LD D, 40H	E = LO RAM ADR
		5E	LDE, (HL)	
		3E 27	LD A, 27H	
3321 _H		BB ^{2 BYTES}	CPE C=1 (BORROW) IF E=CB, DC OR EF	D = SELECTED H0 RAM ADR
		30 02	JRNC, CHKCL1B ^{JMP IF C=1}	
		16 4F	LD D, 4F	
CHKCL1B	3326 _H	7A	LD A, D	A = H0 RAM ADR
		08	EX AF, AF'	SAVE H0 RAM ADR IN A'
		7A	LD A, D	A = H0 RAM ADR AGAIN
		DD 21 30 33	LD IX, CHKCL1C	DISPLAY H0 RAM ADR FOR 1 SEC
		C3 DD 27	JP DSPLY	
CHKCL1C	3330 _H	21 36 33	LD HL, CHKCL1D	
		C3 CC 27	JP DELAY	
CHKCL1D	3336 _H	7B ^{BOY}	LD A, E	A = LO RAM ADR
		DD 21 3E 33	LD IX, CHKCL1E	DISPLAY LO RAM ADR FOR 1 SEC
		C3 DD 27	JP DSPLY	
CHKCL1E	333E _H	21 44 33	LD HL, CHKCL1F	
	3341 _H	C3 CC 27	JP DELAY	
CHKCL1F	3344 _H	AF	XOR AF	BLANK DISPLAY FOR 1 SEC
		D3 EF	OUT (LDSPLY), A	
		D3 FF	OUT (RDSPLY), A	
		21 4F 33	LD HL, CHKCL1G	
		C3 CC 27	JP DELAY	
CHKCL1G	334F _H	08	EX AF, AF'	D = H0 RAM ADR
	3350 _H	57	LD D, A	
		08	EX AF, AF'	
	3352 _H	AF	XOR A	

	3353 _H	12 1A	LD (DE), A LD A, (DE)	ZERO RAM ADR, THEN READ IT
		DD 21 5C 33	LD IX, CHKCL1H	DISPLAY 1ST
		C3 DD 27	JP DSPLY	WRITE AND READ
CHKCL1H	335C _H	21 62 33	LD HL, CHKCL1I	FOR 1 SEC
		C3 CC 27	JP DELAY	
CHKCL1I	3362 _H	08	EX AF, AF'	
		57	LD D, A	D = H0 RAM ADR
		08	EX AF, AF'	
		3E 55	LD A, 0101 0101	WRITE 55 _H TO RAM ADR
		12	LD (DE), A	THEN READ IT
		1A	LD A, (DE)	
		DD 21 70 33	LD IX, CHKCL1J	DISPLAY 2ND
		C3 DD 27	JP DISPLAY	WRITE AND READ
CHKCL1J	3370 _H	21 76 33	LD HL, CHKCL1K	FOR 1 SEC
		C3 CC 27	JP DELAY	
CHKCL1K	3376 _H	08	EX AF, AF'	
		57	LD D, A	D = H6 RAM ADR
		08	EX AF, AF'	
		3E AA	LD A, 1010 1010	WRITE AA _H TO RAM ADR
		12	LD (DE), A	THEN READ IT.
		1A	LD A, (DE)	
		DD 21 84 33	LD IX, CHKCL1L	DISPLAY 3RD
	3381 _H	C3 DD 27	JP DISPLAY	WRITE AND READ
CHKCL1L	3384 _H	21 8A 33	LD HL, CHKCL1M	FOR 1 SEC
		C3 CC 27	JP DELAY	
CHKCL1M	338A _H	08	EX AF, AF'	
		57	LD D, A	D = H0 RAM ADR
		08	EX AF, AF'	
		3E FF	LD A, FF	WRITE FF _H TO RAM ADR,
		12	LD (DE), A	THEN READ IT.
	3390 _H	1A	LD A, (DE)	
		DD 21 98 33	LD IX, CHKCL1N	DISPLAY 4TH
	3395 _H	C3 DD 27	JP DISPLAY	WRITE AND READ
				FOR 1 SEC

93

CHKCL1N	3398 _H	21 9E33	LD HL, CHKCOL10	} BLANK DISPLAY FOR 2 SEC
		C3 CC27	JP DELAY	
CHKCL10	339E _H	AF	XOR A	
		D3 EF	OUT (LDSPY), A	
	33A1 _H	D3 FF	OUT (RDSPY), A	
		21 A933	LD HL, CHKCOL1P	
		C3 CC27	JP DELAY	
CHKCOL1P	33A9 _H	21 AF33	LD HL, CHKCOL1Q	
		C3 CC27	JP DELAY	
CHKCOL1Q	33AF _H	08	EX AF, AF	
	33B0 _H	57	LD D, A	D = HD RAM ADR
	33B1 _H	C3 2633	JP CHKCL1B	LOOP BACK AND REPEAT WRITE/READ/DISPLAY SEQUENCE

NOTES: SETSCREEN3 31FA-33B3_H (441 BYTES)

JUMPS ARE MADE OUTSIDE THIS AREA TO UTILIZE EXISTING ROUTINES IN BALCHECK HR.

ENTIRE LOW-RES SCREEN RAM (102 LINES) IS UTILIZED.

NO STACK IS USED. NO SUBROUTINE CALLS ARE MADE.

NO UPI ON-BOARD SUBROUTINES ARE UTILIZED.

8 ON-BOARD ROM INSTRUCTIONS ARE USED FROM RESET TO JP 2000_H

↑
0000_H

COLOR TABLE (LOW-RES TEST PATTERN)

COLTBL	33B4 _H	71	LIGHT BROWN	PIXEL 11	
		09	LIGHT BLUE	10	LEFT COLORS
		2B	PINK	01	
		10	DARK BLUE	00	
		6B	ORANGE	PIXEL 11	
		60	MAROON	10	RIGHT COLORS
		86	YELLOW	01	
		C0	DARK GREEN	00	

SetScreenHR (SAME AS SetScreen3 EXCEPT AS NOTED BELOW)

StScrHR	33BC _H	3E01	LDA, 01 _H	SET CUSTOM CHIPS TO OPERATE IN HIGH RESOLUTION MODE.
		D308	OUT(08 _H), A	
	33C0 _H	AF	XOR A A=0	
		011808	LD BC, 0818 _H	STOP ALL SOUND
		ED79 ^{DOG}	OUT(C), A	
		10FC	DJNZ-2	
		3ECC	LDA, 204 _D	SET VERTICAL BLANK REGISTER TO 204 _D
	^{DOG}	D30A	OUT(0A _H), A	
		3E14	LDA, 20 _D	SET HORIZONTAL COLOR BOUNDARY TO 20 _D , SPLIT SCREEN
		D309	OUT(09), A	
	33D0 _H	21FA31	LD HL, COLTBL	
		010B08 ^{BOY}	LD BC, 080B _H	$\frac{204}{3} = 68 \text{ LINES}$
	^{DOG}	EDB3	OTIR	FILL 1/3 OF SCREEN
FILL		210040	LD HL, 4000 _H	$68 \times 80 = 5440D = 1540H BOY$
		014015	LD BC, 1540 _H	WITH COLOR AA _H
		3EAA	LDA, 10101010	COLORS L TEL
				R MAROON
FILL1	33E0 _H	77 ^{DOG}	LD(HL), A	
		EDAI	CPI	A ← (HL) HL ← HL + 1 BC ← BC - 1
	33E3 _H	EAE033	JP PE, FILL1	

95

	33E6 _H	01 A0 0A	LD BC, 0AA0	FILL 34 LINES WITH 00 _H BELOW TO FILL 34x80=2720 _D =0AA0 _H BYTES COLORS L BLK R BLU WRITE (FILL) TO 4 CORNER BYTES 4000 _H 404F _H 7F70 _H 7FBF _H
		3E 00	LDA, 00 (NO $\parallel$ x0RA (IN CASE OF CHANGE))	
FILL 2	33EB _H	77	LD (HL), A	
		ED A1 ← DOG	CPI	
		EA EB 33 ← BOY	JP PE, FILL 2	
	33F1 _H	3E 55	LDA, 55 _H	
		32 00 40	LD (4000 _H), A	
		32 4F 40	LD (404F _H), A	
		32 70 7F	LD (7F70 _H), A	
		32 BF 7F	LD (7FBF _H), A	
		DB 17	IN A, (COL 3)	KEY PRESSED IN LEFT MOST COLUMN 3? IF SO, JMP TO CHECK COL 3
	3401 _H	A7	AND A	
		20 0F	JRNZ, CKCL 3	
		DB 15	IN A, (COL 1)	KEY PRESSED IN COLUMN 1? IF SO, JMP TO CHECK COL 1
		A7	AND A	
		C2 88 34	JP NZ, CKCL 1	
HL →	340A _H	76	HALT	HALT 280 NO KEYS PRESSED. EXECUTE NOPs

PIXEL BYTES TO WRITE (USE SAME PIXEL TABLES AS
SetScreen 3 @ 3259_H)

SELECT COLUMN 3 PIXEL TABLE

ENTER WITH A = COL 3 STATUS

	340B _H	59	PIXEL 1 BYTE TABLE (LOW ORDER)
		5D ← DOG	PIXEL 3 ↓
		FF	UNUSED BYTE
		61	PIXEL 0 BYTE TABLE (LOW ORDER)
		FF FF FF	UNUSED BYTES
	3412 _H	65	PIXEL 2 BYTE TABLE (LOW ORDER)
CKCL 3	3413 _H	E6 1F	MASK OUT COL 3 BITS (MAY NOT NEED THIS INSTRUCTION)
		21 0A 34	LD HL, 340A _H HL = LOW ORDER BYTE TABLE - 1
		18 01	JR A 341B _H JUMP TO NEXT INSTRUCTION
	341A _H	69	PIXEL FILL BYTE (LO)

* THESE PARAMETERS ARE OUTSIDE THE PARAMETERS SAVED IN THE Z80 ALTERNATE REGISTERS

HL = TABLE ADDRESS FOR "PIXEL BYTE TO WRITE"
 DE = LAST SCREEN ADDRESS
 B = SAVED "PIXEL BYTE TO WRITE" FOR NEXT LINE WRITTEN
 C = LINES LEFT TO WRITE
 L' = PIXEL TABLE LO BYTE
 B' = PASS LOOP CTR
 C' = PASS DISPLAY NUMBER
 D' = DELAY 1ST LOOP CTR

96

341BH 85
 6F
 6E

ADD A, L
 LD L, A
 LD L, (HL)

SET L AT LOW ORDER ADDRESS OF
 PIXEL COLOR (PASS) TABLE

AND FALL INTO WRITE PIXEL STRIPES
 ENTER WITH L = LOW ORDER BYTE OF PIXEL BYTE COLOR TABLE

HR STRIPE 341EH 1660 LDD, 60H D = INITIAL DELAY "1ST LOOP" COUNTER
 3420H 0604 LDB, 4 B = INITIAL "PASS LOOP COUNTER" (VARIES 1 TO 4)
 0E01 LDC, 1 C = INITIAL "PASS DISPLAY NUMBER"
 79 LDA, C A = PASS DISPLAY NUMBER (REQ'D FOR
 D9 EXX SAVE L' = PIXEL TABLE LO BYTE C' = PASS DISPLAY NUMBER
 DD 21 2D34 LD IX, HR STRIPE 1 IX = RETURN (CONTINUE) ADR
 C3 DD 27 JP DPLY RESTORE L = PIXEL TABLE LO BYTE C = PASS DISPLAY NUMBER
 D9 EXX B = PASS LOOP CTR D = DELAY 1ST LOOP CTR
 7D LDA, L A = PIXEL TABLE LO BYTE
 D9 EXX SAVE AS SHOWN ABOVE
 3430H 6F LD L, A L = PIXEL TABLE LO BYTE *
 2632 LD H, 32H H = HD BYTE *
 7E LD A, (HL) A = PIXEL BYTE TO WRITE
 3434H 110040 LD DE, 4000H DE = SCREEN ADDRESS
 0ECC LDC, 204H C = "LINES TO WRITE" COUNTER *
 HR LINE 1 0650 LD B, 80H B = "BYTES PER LINE" CTR
 HR LINE 2 12 LD (DE), A WRITE A BYTE
 13 INC DE BUMP SCRN ADR } WRITE ONE LINE *
 10FC DJNZ, HR LINE 2
 47 LDB, A SAVE "PIXEL BYTE TO WRITE" IN B *

BEGIN DELAY BEFORE WRITING NEXT LINE

HR DELAY 3440H D9 EXX PRIOR TO EXX D' = DELAY 1ST LOOP CTR
 7A LDA, D PUT IT IN A
 3442H D9 EXX A = 1st LOOP CTR

97

HRDELAY1 3443H 08
 DOG → DB 16
 E6 3F
 28 13

EX AF, AF'
 IN A(COL2)
 AND 0011 1111 ISOLATE COL
 BITS 0-5
 JRZ, HRDELAY5

SAVE IT IN A'
 COL 2 KEY PRESSED?
 IF NO, SKIP USER
 SET NEW WRITE SPEED

HRDELAY2 3450H 38 04
 20065 → DD 23
 18 F8

LD IX, HRSPEEDT POINT IX AT SPEED TABLE
 RRA ROTATE COL STATUS RIGHT
 7 → 0 → CF
 JRC, HRDELAY3
 INC IX POINT IX TO NEXT SPEED
 JR HRDELAY2

ROTATE COL STATUS
 TO POINT IX AT
 RESPECTIVE SPEED

HRDELAY3 DD 7E 00
 D9
 57
 D9

LD A, (IX)
 EXX
 LD D, A
 EXX

A = SELECTED WRITE SPEED

SAVE NEW SPEED
 (DELAY 1ST LOOP CTR)
 IN D'

HRDELAY4 08
 HRDELAY5 3E FF
 HRDELAY6 3D ← DOG
 3460H 20 FD
 08
 3D ← DOG
 20 F6

EX AF, AF'
 LD A, FF
 DEC A
 JRNZ, HRDELAY6
 EX AF, AF'
 DEC A
 JRNZ, HRDELAY4
 CONTINUE WITH A=0

SAVE 1ST LOOP CTR IN A'

A = 2ND LOOP CTR
 2ND DELAY LOOP

A = 1ST LOOP CTR AGAIN
 DECREMENT 1ST LOOP CTR
 LOOP BACK IF CTR ≠ 0

END DELAY BEFORE WRITING NEXT LINE

DOG → 0D
 78
 20 CF
 DOG → D9
 0C
 10 02
 18 B0 ← BOY

DEC C DECREMENT "LINES TO WRITE" CTR
 LD A, B
 JRNZ, HRLINE1
 EXX RESTORE B = PASS LOOP CTR
 INC
 DJNZ HRNXPASS
 JR HRSTRIPE
 INCREMENT PASS NUMBER
 FINISHED WITH 4 PASSES?
 IF NO, JUMP TO HRNXPASS TO
 WRITE NEXT PASS
 IF YES, REPEAT 4 PASSES
 AGAIN

HRNXPASS 3470H 79
 D9

LD A, C
 EXX SAVE C = PASS DISPLAY NUMBER D = DELAY 1ST LOOP

3472H DD 21 79 34 LD IX, HRNXPASS1 IX = CONTINUE (RETURN) ADDRESS

3476H C3 DD27 JP D5PLY HL POINTS TO "PIXEL BYTE TO WRITE" AND IS NOT Clobbered BY ROUTINE D5PLY
 HRNXPASS 3479H 23 INCHL POINT HL AT NEXT "PIXEL BYTE TO WRITE"

18 B7 JR HRSTRIP2 LOOP BACK TO WRITE NEXT PASS
 SPEED (DELAY LOOPCTR) TABLE TABLE CONTAINS TIME DELAY LOOP COUNTER THE LOWER THE COUNTER, THE FASTER THE BYTE LINES WILL BE WRITTEN.

HR SPEED	LOOP COUNTER	COL2 KEY
347CH	07	↑
	10	MS
	15	8
	25	5
	45	2
3481H	A0	0

↑ WRITE SPEED INCREASES LOOP COUNTER IS INITIALIZED TO 60H AT WRITE PIXEL STRIPES (341EH)

WRITE/READ 6 HR RAM BYTES USING KEYPAD COL1
 PROGRAM AT 3488H

LOW ORDER RAM ADDRESS TABLE

HR LORM 3482H 00
 28
 4F
 70

Boy 98
 3487H BF

4000 4028 404F

EXAMINE 6 BYTES 7F70 7F98 7FBF

ENTER WITH A = COL1 STATUS

SAME AS LOW-RES CHKCOL1 @ 3314H EXCEPT AT NOTED BELOW
 CKCL1 3488H 218134 LD HL, HR LORM-1 POINT HL AT LO RAM ADDR TABLE-1 LOW ORDER

CKCL1A CBIF RRA 7 → 0 → CF CHECK COL1 STATUS
 23 INCHL POINT TO NEXT LO RAM ADDR POINT HL AT SELECTED LO RAM ADDRESS
 30 FC JR NC, CKCL1A HIGH ORDER

3490H 16 40 LD D, 40H START WITH HO RAM ADDR 40H
 5E LDE, (HL) E = LO RAM ADDR

3495H BB ← 2B 45 LDA, 4F CPE IF E = 70, 98 OR BF D = SELECTED HO RAM ADDR

99

3496 _H	30 02	JRNC, CKCL1B	]
	16 7F	LD D, 7F	
CKCL1B 349A _H	C3 2633	JP CHKCL1B	JUMP TO FINISH ROUTINE SAME AS SETSCREEN3 COLUMN 1 ROUTINE

NOTES: SETSCREENHR AT 33BC-349C_H AND 3326-33B3_H

JUMPS ARE MADE OUTSIDE OF THE AREA TO UTILIZE EXISTING
ROUTINES IN BALCHECKHR.

ENTIRE HI-RES SCREEN RAM (204 LINES) IS UTILIZED.

NO STACK IS USED. NO SUBROUTINE CALLS ARE MADE.

NO UPI ON-BOARD SUBROUTINES ARE UTILIZED.

8 ON-BOARD ROM INSTRUCTIONS ARE USED FROM RESET TO
JP 2000_H ↑ 0000_H

NOTE: THIS DEMO DOES JUMP OUTSIDE ITS ROM AREA AT
3569_H AND 3578_H

100

(CRASH TEST (FISH) DEMO

MOVE 10 FISH AROUND ENTIRE SCREEN WITH HR:MIN:SEC CLOCK
COUNTING UP AT BOTTOM OF SCREEN



STACK BEGINS HERE

4F91 _H	"FISH PATTERN ADDRESS" TO WRITE (2 BYTES)
4F93	SECONDS
4F94	MINUTES
4F95	HOURS
4F96	MR (XOR IT)
4F97	VECTOR STATUS
4F98	TIME BASE
4F99	ΔX_L } ΔX
4F9A	ΔX_H }
4F9B	X_L } X
4F9C	X_H }
4F9D	X CHECKS MASK (LIMIT CHECK REVERSE Δ)
4F9E	ΔY_L } ΔY
4F9F	ΔY_H }
4FA0	Y_L } Y
4FA1	Y_H }
4FA2	Y CHECKS MASK (LIMIT CHECK REVERSE Δ)

$X_H, Y_H, \Delta X_L, \Delta Y_L, \Delta$ TIMER FOR SMALL FISH #

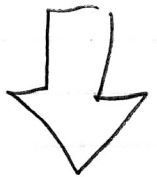
5 BYTE
VECTOR
PACKET
WITH
FISH
VARIABLES

4FA3
4FA8
4FAD
4FB2
4FB7
4FBC
4FC1
4FC6
4FCB
4FDD_H

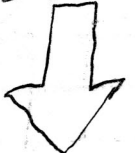
LARGE FISH #

1
2
3
4
5
6
7
1
2
3

RAM
DATA
BASES



RAM
DATA
BASES



101

4FE3 1PPS, BIT 7 ONLY
 4FEB TMR60
 4FED SECONDS } FOR
 4FEE MINUTES } DISTIM
 SUB#82

RAM
 DATA
 BASES



CRASH TEST COLOR TABLE

COLOR2	349D _H	F9	BLUE	PIXEL 11
		62	RED	10
		07	WHITE	01
	34AD _H	F8	DARK BLUE	00

INITIAL FISH VECTOR PACKET

34A1_H 20 MR (XOR IT)
 80 VECTOR STATUS
 02 TIME BASE
 00 ΔXL } ΔX
 00 ΔXH }
 00 XL } X
 00 XH }
 03 X CHECKS MASK (LIMIT CHECK
 REVERSE Δ)
 00 ΔYL } ΔY
 00 ΔYH }
 00 YL }
 00 YH }
 03 Y CHECKS MASK (LIMIT CHECK
 REVERSE Δ)

LIMITS TABLE 1 (8x7 PIXEL FISH)

34AE_H 00 X LOWER LIMIT
 98 X UPPER
 34B0_H 00 Y LOWER
 55 Y UPPER

$$X_{UPPER} = 159_D - 7 = 152_D = 98_H$$

$$Y_{COORD \text{ FOR TIME CLOCK}} = 92_D$$

$$Y_{UPPER} = 91 - 6 = 85_D = 55_H$$

LIMITS TABLE 2 (12x12 PIXEL FISH)

34B2_H 00 X LOWER LIMIT
 94 X UPPER
 00 Y LOWER
 34B5_H 50 Y UPPER

$$X_{UPPER} = 159 - 11 = 148_D = 94_H$$

$$Y_{UPPER} = 91 - 11 = 80_D = 50_H$$

"5 BYTE VECTOR PACKET VARIABLES" INDEX TABLES

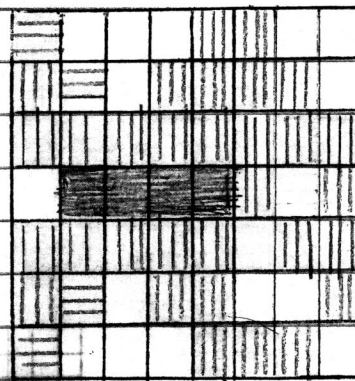
INDEX 1	34B6H	A3 4F	SMALL FISH #	1
		A8		2
		AD		3
		B2		4
		B7		5
	34C0H	BC		6
		C1		7
INDEX 2	34C4H	C6	LARGE FISH #	1
		CB		2
	34C8H	DO		3

FISH PATTERN (8x7 PIXELS)

34CAH 00 RELATIVE X
 00 ↓ Y
 02 X SIZE
 07 Y SIZE

C0 50
 34DOH 71 54
 55 55 ← DOG
 2A 9D
 55 15
 71 41

34DAH C0 54



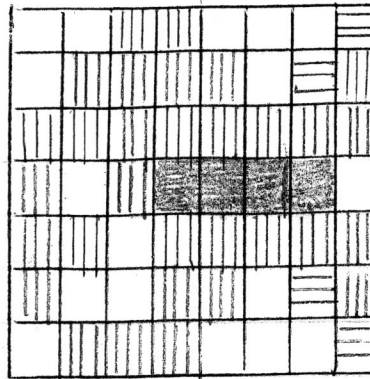
11 BLUE
 10 RED
 01 WHT
 00 DARK BLUE

FLOPPED FISH PATTERN (8x7 PIXELS)

34DCH 00 RELATIVE X
 00 ↓ Y
 02 X SIZE
 34DEH 07 Y SIZE

103

34E0H 05 03
 15 4D ← DOG
 55 55
 76 A8
 54 55 ← DOG
 41 4D
 34EC_H 15 03

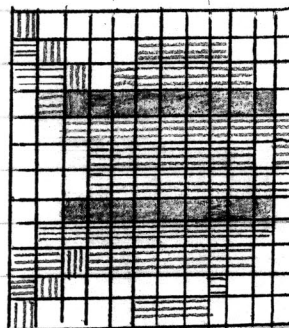


FISH PATTERN (12x12 PIXELS)

34EE_H 00 RELATIVE X
 00 ↓ Y

34FO_H 03 X SIZE
 0C Y SIZE

DOG → 40 00 00
 D0 3F C0
 F4 FF F0
 3A AA A8
 OF FF FF
 3501_H 03 FF F3
 03 FF FF
 0A AA A8
 3F FF FC
 3510_H → D0 00 C0
 40 3F 00



NEW

11 BLUE
 10 RED
 01 WHT
 DARK
 00 BLUE

3515_H RELATIVE X

FLOPPED FISH PATTERN (12x12 PIXELS)

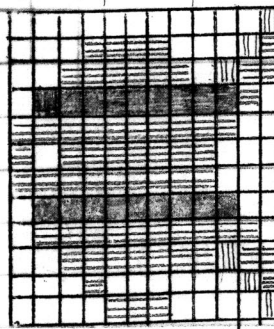
3515 00 RELATIVE X

3516_H 00 ↓ Y

03 X SIZE

0C Y SIZE

00 00 01
 03 FC 07
 OF FF IF
 3522_H 2A AA AC
 FF FF FO
 CF FF CO
 FF FF CO
 2A AA AO
 3531_H 3F FF FC
 OF FF DE
 03 00 07
 353A_H 00 FC 01



4F93 SECONDS
4F94 MINUTES
4F95_H HOURS

WRITE ELAPSED TIME

USE 3x5 PIXEL NUMBER
WITH: 1 PIXEL HORIZ SPACING
1 PIXEL VERTICAL
FRAME SIZE, 4x6 PIXELS
HR:MN:SC, 8 FRAMES LONG = 32 FRAMES
CENTER $(160-32) \div 2 = 64$
USE X=63D
Y=92D

104

DSTIM 353D_H 21934F
3540_H 7E _{← DOG}
32 ED4F
23
7E
32 EE4F _{← DOG}
DD 210D02
FF
00
37 _{← DOG}
3550_H 3D
5C
0C
C2
95 4F
0D _{← DOG}
20
BA
32 _{← DOG}
0D
04
8C
52
02
355F C9

LD HL, 4F93
LDA, (HL)
LD (4FED_H), A
INC HL
LDA, (HL)
LD (4FEE_H), A
LD IX, 020D _{IX = FONT DESCRIPTOR}
RST 38_H

INTPC _{BEGIN MULTIPLE CALLS}
DISNUM+1 (SUB#54_D)

REG E	X COORD = 61D	PLOP WRITE 5 CHAR PIXEL CLR 0000 1100 BCKGRD COLOR → NORMAL SIZE 11000100 # OF DIGITS → ALT FONT TO DISPLAY → BLANK LEADING ZERO	DISPLAY HOURS
D	Y COORD = 92D		
C	CHAR DISPLAY OPTIONS		
B	NUMBER DISPLAY OPTIONS		
L, H	NUMBER ADDRESS		
SUCK+1 (SUB#12)	← A REG		SUCK COLON CHAR INTO UPI (CONTEXT BLANK)
REG B	0010 0000		
↓ A	CHAR IS COLON (':')		

CHRDIS (SUB#50_D, NO SUCK) _{DISPLAY COLON}
SUCK+1 (SUB#12) _{← C}
REG B 0000 0100 _{CHAR PIXEL}
↓ C _{DISPLAY → STD SIZE} 1000 1100 _{(SUB#82 HAS DIFFERENT) DISPLAY OPTIONS}
MIN:SEC _{← PLOP → BCKGRD COLOR}
DISTIM (SUB#82, NO SUCK) _{DISPLAY MIN:SEC}
XINTC _{END MULTIPLE CALLS}
RETURN

VECTOR FOR CRASH DEMO INTERRUPT ROUTINE
IS AT 3702_H

105

4F93 SEC
4F94 MIN
4F95 HRS

4FE3 1PPS BIT 7 = 1PPS MARKER
4FEB_H TMR60

(CRASH DEMO INTERRUPT ROUTINE (UPDATE ELAPSED TIME))

UTIME	3560 _H	F3	DI	DISABLE INTERRUPTS
		F5	PUSH AF	
		C5	BC	SAVE THE ENVIRONMENT
		D5	DE	
		E5	HL	
		21 EB 4F	LD HL, TMR60	DECREMENT TMR60
		35	DEC (HL)	
3569 _H		F2 1B 2D	JP P, TIME 3	IS SECONDS NEGATIVE (FF _H)? IF NO, YOUR DONE. JMP TO EXIT ROUTINE. IF YES, CONTINUE WITH UPDATE
		36 3B	LD (HL), 59 _D	TMR60 = 59 _D SET 1PPS
		21 E3 4F	LD HL, 4FE3 _H	
3571 _H		CB FE	SET 7, (HL)	
		06 02	LD B, 2 B = LOOP CTR = 2	
		21 93 4F	LD HL, 4F93 _H	POINT HL AT SECONDS
3578 _H		C3 08 2D	JP TIME 1	JMP TO UPDATE TIME AND EXIT INTERRUPT ROUTINE

RANDOMIZE A DELTA (X OR Y)

RDELTA	357B _H	FF	RST 38 _H	RANDOMIZE A VALUE FROM 0-126 _D
		77	RANGED + 1 (SUB# 118)	
		7F	REG A (A-1 = MAX# TO GENERATE)	
		4F	LDC, A	IF RANDOM VALUE IS "EVEN", USE THIS VALUE AS A POSITIVE DELTA Δ
		1F	RRA	
3580 _H		79	LD A, C	
		D0	RET NC	IF RANDOM VALUE IS "ODD", CHANGE IT SO IT APPEARS AS A NEGATIVE DELTA Δ (-Δ = -2, -4, -6 ... -126)
		2F	CPL	
3583 _H		C9	RET	

CHECK DELTA TIMER

ENTER WITH: HL POINTING AT FISH DELTA Δ TIMER

CKTIMER	3584 _H	35	DEC(HL)	DECREMENT THIS Δ TIMER.
		C0	RET NZ	IF Δ TIMER $\neq 0$, RETURN (NO RESET REQ'D)
		FF	RST 38H	} Δ TIMER REACHED ZERO. RESET AND RANDOMIZE NEW Δ TIMER FROM 0-79 _D
		77	RANGED+1 (SUB#118)	
		50	A REG (A-1 = MAX # TO GENERATE)	
		77	LD (HL), A	SAVE NEW Δ TIMER FOR THIS FISH.

AND LOAD NEW ΔX , ΔY INTO VECTOR PACKET

LOAD RANDOM DELTAS INTO VECTOR PACKET

DELXY	358A _H	CD 7B 35	CALL RDELTA	RANDOMIZE $\pm \Delta X_L$
		32 99 4F	LD (4F99 _H), A	PUT IT IN VECTOR PACKET
	3590 _H	CD 7B 35	CALL RDELTA	RANDOMIZE $\pm \Delta Y_L$
		32 9E 4F	LD (4F9E _H), A	PUT IT IN VECTOR PACKET.
		C9	RET	

CHECK FOR FLOP - SMALL FISH

SFLOPCK	3597 _H	E5	PUSH HL	SAVE POINTER
		3A 99 4F	LDA, (4F99 _H)	A = ΔX_L
		CB 7F	BIT 7, A	} IF ΔX_L IS POSITIVE, $\Delta X_H = 0$ HL $\rightarrow$ NORMAL FISH PATTERN
		3E 00	LD A, 0	
		21 CA 34	LD HL, 34CA _H	
	35A2 _H	28 05	JR Z, SFLOPCK1	} IF ΔX_L IS NEG, $\Delta X_H = FFH$ HL $\rightarrow$ FLOPPED FISH PATTERN
		3E FF	LDA, FFH	
		21 DC 34	LD HL, 34DC _H	
SFLOPCK1	35A9 _H	32 9A 4F	LD (4F9A _H), A	PUT ΔX_H IN VECTOR PACKET PUT FISH PATTERN ADR TO WRITE @ 4F91 _H
		22 91 4F	LD (4F91 _H), HL	
		E1	POP HL	RESTORE POINTER
	35B0 _H	C9	RET	

107

WRITE SMALL FISH AND SAVE VP VARIABLES

ENTER WITH: HL POINTING TO Δ TIMER FOR THIS FISH

WRITE F 35B1^{←D6}H CD 8435 CALL CKTIMER CHECK Δ TIMER TO SEE IF A NEW Δ XL, Δ YL IS REQ'D
 CD 9735 CALL SFLOPCK SET UP CORRECT FISH ORIENTATION FOR THIS FISH, i.e. CHECK FOR A FLOPPED FISH. THEN, SET Δ XH ACCORDINGLY.

2A 914F LD HL, (4F91_H) HL = PATTERN ADDRESS
²00G5 DD 21964F LD IX, 4F96_H IX = VECTOR PACKET ADDRESS
 FF RST 38H
 1E VWRITER (SUB#30 - NO REG - SUCK) } WRITE THE FISH

35C0_H 78 LDA, B PUT FISH # IN REG A AGAIN
 FF RST 38H PUT
 5B^{←BOY} INDEXW+1 (SUB#90) "5 BYTE VECTOR PACKET VARIABLES ADDRESS IN DE."
 B4 34 L, H REG (INDEX TABLE)
 WRITE F 1 35C5_H EB^{←BOY} EX DE, HL HL NOW POINTS TO "5 BYTE VECTOR PACKET VARIABLES" FOR THIS FISH

3A 9C4F LDA, (4F9C_H) GET X_H FROM VP AND SAVE IT IN "5 BYTE FISH VARIABLES PACKET"
 77 LD (HL), A
 23 INC HL
 3A A14F LDA, (4FA1_H) GET Y_H
 77 LD (HL), A
 23 INC HL
 35D0_H 3A 994F LDA, (4F99_H) GET Δ XL
 77 LD (HL), A
 23 INC HL
 3A 9E4F LDA, (4F9E_H) GET Δ YL
 77 LD (HL), A
 35D9_H C9 RET

INITIALIZE VP VARIABLES X_H , Y_H , ΔX_L AND ΔY_L LOAD VARIABLES IN VP

IVPKT	35DA _H FF	RST 38 _H	RANDOMIZE X_H FROM 0-148D
	77	RANGED+1 (SUB#118)	
	95	A REG (A-1 = MAX# TO GENERATE)	
	329C4F	LD (4F9C _H), A	PUT X_H IN VECTOR PACKET
35E0 _H	FF	RST 38 _H	RANDOMIZE Y_H FROM 0-80D
	77	RANGED+1	
	51	A REG	
	32A14F	LD (4FA1 _H), A	PUT Y_H IN VECTOR PACKET
	CD8A35	CALL DELXY	RANDOMIZE ΔX_L AND ΔY_L . PUT THEM IN VECTOR PACKET.
35E9 _H	C9	RET	

(CHECK FOR FLOP - LARGE FISH (SIMILAR TO SMALL FISH FLOP, SFLOPCK @ 3597_H))

LFLPCK	35EA _H E5	PUSH HL	NEEDED?	
	3A994F	LD A, (4F99 _H)	A = ΔX_L	
	CB7F	BIT 7, A		IF ΔX_L IS POSITIVE $\Delta X_H = 0$
35F0 _H	3E00	LDA, 0		- HL → NORMAL FISH PATTERN
	21EE34	LD HL, 34EE _H	POINT HL AT LARGE FISH PATTERN	
	2805	JRZ, LFLPCK1		
	3EFF	LD A, FF _H		IF ΔX_L IS NEG, $\Delta X_H = FF_H$
	211535	LD HL, 3515 _H		HL → FLOPPED FISH PATTERN
LFLPCK1	35FC _H C3A935	JP SFLOPCK1	JUMP TO FINISH ROUTINE	

109

WRITE LARGE FISH AND SAVE VP VARIABLES

(SIMILAR TO WRITE SMALL FISH, WRITE F @ 35B_H)

ENTER WITH: HL POINTING TO A TIMER FOR THIS FISH

WRITE F	35FF _H	CD 84 35	CALL CKTIMER	CHECK A TIMER TO SEE IF A NEW ΔX_L , ΔY_L IS REQ'D.
	3602 _H	CDEA 35	CALL LFLOPCK	SETUP CORRECT FISH ORIENTATION FOR THIS FISH, IE, CHECK FOR A FLOPPEN FISH. THEN SET X_H ACCORDINGLY
		2A 91 4F	LD HL, (4F91 _H)	HL = ADDRESS VECTOR PACKET
	2005	DD 21 96 4F	LD IX, 4F96 _H	IX = ADDRESS
		FF	RST 38 _H	
		1E	VWRITER (SUB#30 - NO REG)	WRITE THE FISH
		78	LD A, B	PUT FISH # IN REG A AGAIN
		FF	RST 38 _H	
	3610 _H	5B ← BOY	INDEXW+1 (SUB#90)	PUT "5 BYTE VECTOR PACKET VARIABLES" ADDRESS IN DE
		C2 34	L, H REG (INDEX TABLE)	
	3613 _H	C3 C5 35	JP WRITE F1	JMP TO SAVE VP VARIABLES AND FINISH ROUTINE

SET UP X_H , Y_H AND ΔX_L IN VECTOR PACKET FOR FISH VECTOR ROUTINE
 ENTER WITH: DE POINTING TO X_H IN "5 BYTE VP VARIABLES" FOR THIS FISH

3616 _H	EB	EX DE, HL	POINT HL AT X_H
	7E	LD A, (HL)	- LOAD X_H INTO VP
	32 9C 4F	LD (4F9C _H), A	
	5F	LD E, A	E = X_H FOR VECTOR ROUTINE
	23	INC HL	POINT HL AT Y_H
	7E	LD A, (HL)	- LOAD Y_H INTO VP
	32 A1 4F	LD (4FA1 _H), A	
3621 _H	57	LD D, A	D = Y_H FOR VECTOR ROUTINE
	23	INC HL	POINT HL AT ΔX_L
	7E	LD A, (HL)	- LOAD ΔX_L INTO VP
	32 99 4F	LD (4F99 _H), A	
3627 _H	C9	RET	

SET UP ΔY_L AND ΔY_H IN ^{VECTOR} PACKET FOR FISH VECTOR ROUTINE
 ENTER WITH: HL POINTING TO ΔX_L IN "5 BYTE VP VARIABLES FOR THIS FISH."

3628 _H	23	INC HL	POINT HL AT ΔY_L
	7E	LDA, (HL)	LOAD ΔY_L INTO VP
	32 9F 4F	LD(4F9E _H), A	
	^{← BODY} CB 7F	BIT 7, A	IF ΔY_L IS POSITIVE, SET ΔY_H
	3E FF	LDA, FF _H	IN VP TO 00 _H .
3631 _H	20 01	JRNZ, A ?	IF ΔY_L IS NEGATIVE, SET ΔY_H
	AF	XOR A	IN VP TO FF _H
	32 9F 4F	LD(4F9F _H)	
	23	INC HL	POINT HL AT Δ TIMER
3638 _H	C9	RET	

CRASH TEST (FISH) PROGRAM

CT 3639H	3191 4F	LD SP, 4F91H	INITIALIZE STACK POINTER
	FF	RST 38H	
	00	INTPC	ENTER INTERPRETIVE CALLS (BEGIN MULTIPLE CALLS)
	14	EMUSIC (#20 _D)	STOP ALL SOUND
	19 DOG	COLSET+1 (#24 _D)	SET COLORS
3640H	9D 34	COLOR TABE	
	17	SETOUT+1 (#22 _D)	SET
	C4	VERT BLK REG	VERTICAL BLANK REGISTER = 196 _D
	2A	HOR CLR BNDAY	HOR COLOR BOUNDARY 00 42 _D BKSRD PIXEL
	08	INT MODE	
	1B	FILL+1 (#26 _D)	CLEAR SCREEN
	00 40	START ADR	CLEAR 99 LINES
	78 0F	# OF BYTES TO FILL	99x40 = 3960 _D 0F78H
	00	FILL WITH 00H	4000 - 4F77H
	1B	FILL+1 (#26 _D)	CLEAR DATA BASES
	91 4F	ADDRESS TO BEGIN FILLING	4F91 - 4F9F = 15
	6F 00	# OF BYTES TO FILL	4FA0 - 4FEF = 80
			4FF0 - 4FFF = 16
3651H	00	FILL WITH ZERO	111 _D = 6FH
	5F	MOVE+1 (#94 _D)	INITIALIZE
	96 4F DOG	DESTINATION ADDRESS	FISH VECTOR PACKET
	0D 00	# OF BYTES TO MOVE = 13 _D	@ 4F96H
	A1 34	SOURCE ADDRESS	
	02 DOG	XINTC	EXIT INTERPRETIVE CALLS (END MULTIPLE CALLS)
	CD 3D 35	CALL DSTNV	DISPLAY HR:MM:SC, CLOCK AT BOTTOM OF SCREEN
SET UP SCREEN INTERRUPT			
365DH	3E 37	LDA, 37	OUTPUT PAGE OF IM2 VECTOR
	ED 47	LDI, A	
3661H	3E 02 DOG	LDA, 02	OUTPUT LINE OF IM2 VECTOR
	D3 0D DOG	OUT (0D _H), A	
	ED 5E	IM2	
	3E C8	LDA, C8H	GENERATE INTERRUPT AT 200 SCAN LINES
3669H	D3 0F DOG	OUT (0F _H), A	

112

366B_H FB ← BOY

EI

ENABLE INTERRUPTS

INITIALIZE VECTOR PACKET VARIABLES FOR EACH OF THE 7 SMALL FISH, THEN WRITE FISH ON SCREEN.

IFISH

IFISH1

06 07_{DOG}
CD DA 353671_H 78

FF

5B ← BOY

B4 34

21 04 00

19

36 80 ← BOY

CD BI 35

10 ED ← DOG

LD B, 7 B = BEGIN WITH FISH # 7

CALL IVPKT INITIALIZE VP VARIABLES AND LOAD THEM INTO VP.

LDA, B

A = FISH # TO LOOK UP

RST 38_H

INDEXW+1 (SUB #90)

L, H REG (INDEX TABLE)

LD HL, 4D

ADD HL, DE

LD (HL), 80_H

CALL WRITEF

DJNZ IFISH1

GET
"5 BYTE VP VARIABLES
ADDRESS
FOR THIS FISH.
PUT ADR IN DECB"

INITIALIZE

ΔTIMER

FOR THIS FISH

POINT HL AT ΔTIMER ADDRESS FOR THIS FISH

SET ΔTIMER = 148_D FOR THIS FISH

WRITE A FISH ON SCREEN

LOOP BACK TO WRITE NEXT FISH?

-19 168421
00010011
11101100
+1
11101101

INITIALIZE THE 3 LARGE FISH, THEN WRITE FISH ON SCREEN

ILFISH

ILFISH1

3681_H 06 03_{DOG}
CD DA 35

78

FF

5B ← BOY

C2 34

21 04 00

19

36 80

3691_H CD FF 353694_H 10 ED ← DOG

LD B, 3

B = BEGIN WITH FISH #3

CALL IVPKT

SIMILAR

LDA, B

TO

RST 38_H

ABOVE

INDEXW+1

L, H REG (INDEX TABLE)

LD HL, 4D

ADD HL, DE

LD (HL), 80

CALL WRITEF

DJNZ ILFISH1

-19 168421
00010011
11101100
+1
11101101

SET UP VECTOR PACKET FOR VECTOR ROUTINE (SMALL FISH)

3696H 0607	LDB, 7	B = FISH #
3698H 78	LDA, B	A = FISH # INDEX
FF	RST 38H	GET "5 BYTE VP VARIABLES" ADDRESS FOR THIS FISH PUT ADDRESS IN DECB
5B ← B ← Y	INDEX W + 1	
B ← Y → B434	L, H REG (INDEX TABLE ADDRESS - 2)	
CD 16 36	CALL A3616	SET UP IN VP X _H , Y _H AND OX _L DE = Y AND X COORDINATES
CD 97 35	CALL 5 FLOPCK	CHECK FOR PATTERN FLOP
36A3H CD 28 36	CALL A3628	SET UP IN VP ΔYL AND ΔYH
E5	PUSH HL	SAVE Δ TIMER POINTER (ADDRESS)
3E 03	LDA, 3	RESTORE TIME BASE IN VECTOR PACKET
32 98 4F	LD (4F98H), A	

MOVE (VECTOR) SMALL FISH

FF	RST 38H	
3F	VECT + 1	SUB # 62, REG LOAD REQ'D
96 4F	IX _L , IX _H REG	VECTOR PACKET ADDRESS
36B0H AE 34	L, H REG	LIMITS TABLE ADDRESS

BLANK (ERASE) SMALL FISH

ENTER WITH: DE = X, Y COORDINATES OF FISH

WAIT UNTIL IM2 ROUTINE IS FINISHED
(MINIMIZE FISH "OFF TIME")MAY NOT
NEED THIS
INSTRUCTION

76	HALT	
3E 20	LDA, 20H	A = MAGIC REG, XOR WRITE
2A 91 4F	LD HL, (4F91H)	HL = FISH PATTERN ADDRESS
FF	RST 38H	
20	WRITR (NO SUCK)	BLANK THE FISH

WRITE VECTORED SMALL FISH

E1	POP HL	RETRIEVE Δ TIMER POINTER (ADR) FOR THIS FISH
CD B1 35	CALL WRITEF	WRITE VECTORED FISH
36BEH 10 D8	DJNZ A3698	LOOP BACK FOR NEXT FISH

-40 3216 8421
00101000
11010111
+1
11011000

114

SET UP VECTOR PACKET FOR VECTOR ROUTINE (LARGE FISH)

SIMILAR TO SMALL FISH @ 3696H

36C0H 06 03	LDB, 3	B = FISH #
36C2H 78	LDA, B	A = FISH # INDEX
FF	RST 38H	GET "5 BYTE VP VARIABLES" ADDRESS FOR THIS FISH. PUT ADDRESS IN DE CB.
5B ← BOT	INDEXW+1	
02 34	L, H REG (INDEX TABLE ADDRESS-2)	SET UP VP X _H , Y _H AND ΔX. DE = Y AND X COORDINATES
CD 16 36	CALL A3616	CHECK FOR PATTERN FLOP.
CD EA 35	CALL LFL0PCK	SET UP VP ΔYL AND ΔYH
CD 28 36	CALL A3628	SAVE Δ TIMER POINTER (ADDRESS)
36D0H E5	PUSH HL	RESTORE TIME BASE IN VECTOR PACKET
3E 03	LDA, 3	
32 98 4F	LD (4F98H), A	

MOVE (VECTOR) LARGE FISH

FF	RST 38H	
3F	VECT+1	SUB#02, REG LOAD REQ'D
96 4F	IX _L , IX _H REG	VECTOR PACKET ADDRESS
BOT → B2 34	L, H REG	LIMITS TABLE ADDRESS

BLANK (ERASE) LARGE FISH

ENTER WITH: DE = X, Y COORDINATES

76	HALT	EXTEND FISH WRITE "ON TIME" HALT Z80, EXECUTE NOP'S FOR AWHILE.
3E 20	LDA, 20	A = MAGIC REGISTER, XOR IT
2A 91 4F	LD HL, (4F91H)	HL = "FISH PATTERN ADDRESS" TO WRITE
36E2H FF	RST 38H	
20	WRITE SUB#32	

WRITE VECTORED LARGE FISH

E1	POP HL	
CD FF 35	CALL WRITELF	
36E8H 10 D8	DJNZ A36C2	LOOPBACK FOR NEXT FISH

36EAH 21E34F	LD HL, 4FE3H	IF 1PPS MARKER IS SET, RESET IT AND DISPLAY NEW TIME. NOTE: 1PPS IS SET BY SCREEN INTERRUPT @ 3560H IF KEY IN RIGHT MOST COLUMN IS PRESSED, CHANGE BACKGROUND COLOR TO LIGHT BLUE (COLOR REGISTER 4)
CB7E ^{Bot}	BIT 7, (HL)	
2805	JRZ, A36F6	
36F1H CBBE ^{2 Bot's}	RES 7, (HL)	
CD3D35 ^{Doc}	CALL DSTIM	
36F6H DB14 ^{Bot}	IN A, (COLOR)	
A7	AND A	
2804	JRZ, A36FF	
3E FB ^{Bot}	LD A, FC ^{LIGHT BLUE}	
D304	OUT (04H), A	
36FF C39636	JPA3696	LOOP BACK TO (CONTINUE MOVING FISH)

INTERRUPT VECTOR

SET UP AT 365DH
 INTERRUPT ROUTINE AT 3560H

3702H 6035

COLOR TABLE

3704	FA	LIGHT BLUE	PIXEL 11	REG 7
	FC	IRRELAVENT	10	6
	87	YELLOW	01	5
	00	BLK	00	4

LOW RES DEMO 1
TITLE PAGE
116

CUSTOM FONT DESCRIPTOR TABLE

3708H	A0	BASE CHAR	HEX CHAR CODE OF 1ST CHAR IN CHAR TABLE
	06	X FRAME SIZE	NUMBER OF PIXELS WIDE (INCLUDING X SPACING)
	07	Y FRAME SIZE	NUMBER OF PIXELS HIGH (NO Y SPACING)
	01	X PATTERN SIZE	PATTERN BYTES WIDE IN TABLE
	07	Y PATTERN SIZE	PATTERN LINES HIGH ↓
	OF 37	CHAR TABLE ADR	

CUSTOM UPPER/LOWER CASE 5x7 PIXEL CHAR (WITH NONSTANDARD CHAR CODES)

	CHAR	NON STD CODE
370F _H	F0 88 88 F0 88 88 F0	B A0
3716 _H	00 3C 44 44 4C 34 00	a A1
	20 20 20 20 20 20 20	l A2
3724 _H	00 00 78 80 80 80 78	c A3
	80 80 F0 88 88 88 88	h A4
3732 _H	00 00 F8 88 88 80 F8	e A5
	00 00 78 80 80 80 78	c A6
3740 _H	40 40 48 50 60 50 48	k A7
	00 00 00 00 00 00 00	SPACE A8
	88 88 88 F8 88 88 88	H A9
3755 _H	F0 88 88 F0 A0 90 88	R AA

4D 43 4D 20 44 45 53 49 47 4E 05 32 30 31 38 0C
375CH M C M D E S I G N ↑ 2 0 1 8
TAB

INTERRUPT VECTORS

376C _H	7037	INTERRUPT ROUTINE 1
376E _H	8137	↓ 2

117

INTERRUPT ROUTINE 1

INT1	3770H	F3	DI		
		F5	PUSH AF		
		3E 20	LDA, 20		USE
		D3 0F	OUT (0FH), A		INTERRUPT ROUTINE 2
		3E 6E	LDA, 6EH		NEXT
		D3 0D	OUT (0DH), A	OUTPUT LINE OF IMZ VECTOR 2	AFTER 32 SCAN LINES
		3E F9	LDA, F9H		
INT1A		D3 04	OUT (04), A		SET COLOR REG 4 = BLUE
		FI	POP AF		
		FB	EI		
	3780H	C9	RET		

INTERRUPT ROUTINE 2

INT2	3781H	F3	DI		
		F5	PUSH AF		
		3E A0	LDA, A0		USE
		D3 0F	OUT (0FH), A		INTERRUPT ROUTINE 1
		3E 6C	LDA, 6C		NEXT
		D3 0D	OUT (0DH), A	OUTPUT LINE OF IMZ VECTOR 1	AFTER 160 SCAN LINES
		3E 00	LDA, 00		
		18 ED	JR INT1A		JMP TO SET 2ND COLOR TO BLACK
-19	168421	378DH			
	00010011				
	11101100				
	+1				
	11101101				

TITLE PAGE (LOW-RES DEMO 1)

PROGRAM

TITLE			SCREEN DISABLE INTERRUPTS
378F _H	F3	DI	
3790 _H	31CE4F	LD SP, 4FCE _H	SET RAM STACK POINTER
	FF	RST 35 _H	
	00	INTPC	BEGIN MULTIPLE CALLS
	14	EMUSIC (SUB#26)	STOP ALL SOUND
	19	COLSET+1 (SUB#24)	SET COLORS
	04 37	COLOR TABLE	
	17	SETOUT+1 (SUB#22)	SET
	C6	VERT BLK REG	VERT BLK REG = 196 _D
	2A	HOR CLR BNDRT	HOR CLR BNDRT $\frac{00\ 10\ 1010}{1}$
	08	INT MODE	INT MODE BKGRD 42 _D COLOR
	1B	FILL+1 (SUB#26)	CLEAR SCREEN
	00 40	START ADR # OF BYTES TO FILL	CLEAR 99 LINES
37A0 _H	78 0F		$99 \times 40 = 3960_{D} = 0F78_{H}$
	00	FILL WITH 00 _H	4000 - 4F7 _H
	1B	FILL+1 (SUB#26)	CLEAR DATA BASES
	CE 4F	START ADR # OF BYTES TO FILL	4FCE - 4FFF _H
	32 00		50 _D = 32 _H
	00	FILL WITH 00 _H	
	02	XINTC	END MULTIPLE CALLS

SET UP INTERRUPT VECTOR 1

3E 37	LD A, 37	] OUTPUT PAGE OF IM2 VECTOR 1
37AC _H	ED 47	
	OVER	

119

37AEH 3E 6C LDA, 6C
37B0H D3 0D OUT (0DH), A
 DOG ED 5E IM2
37B4H FB EI

OUTPUT LINE OF
IM2 VECTOR 1

ENABLE INTERRUPT NOW

PROGRAM CONTINUED NEXT PAGE.

WRITE "Balcheck HR" ROUTINE

120

WITLE 37B5H DD 21 08 37 LD IX, A3708
 01 54 0B LD BC, 0B54H
 11 0F 39 LD DE, 350F
 3E A0 LD A, A0
 WITLE1 37C1H FF RST 38H
 32 CHRDIS (SUB#50, NOSUCK) WRITE CHAR
 15 DEC D
 15 DEC D
 15 DEC D
 3C INCA
 10 F8 DJNZ WITLE1
 IX = FONT DESCRIPTOR TABLE ADDRESS
 B = # OF CHAR TO WRITE (INCLUDING SPACE) = 11D
 C = CHAR DISP 5 OR 5 OFF CLR
 OPTIONS 01 01 01 00 ON CLR
 D = Y COORD = 57D = 39H
 E = X ↓ = 15D = 0FH
 A = 1st CHAR CODE IN CUSTOM STRING
 SUBTRACT 3 FROM NEXT FRAME Y COORD (RAISE NEXT CHAR UP) 3 PIXELS
 A = NEXT CUSTOM CHAR CODE
 WRITE NEXT CHAR

-8 00001000
 11110111
 +1
 11111000

WRITE DESIGNER NAME

FF RST 38H
 35 STRDIS+1
 06 X COORD
 59 Y ↓
 0C CHAR DISPLAY
 5C 37 OPTION
 VARY TITLE COLOR CHAR STRING
 ADDR
 WRITE
 MCM DESIGN 2018
 AT SCREEN BOTTOM

VARY TITLE COLOR

CHGCLR 37D0H 01 05 87 LD BC, 8705H
 ED 41 OUT(C), B
 C5 006 PUSH BC
 21 DC 37 LD HL, A37DC
 C3 CC 27 JP DELAY
 37DC 21 E2 37 LD HL, A37E2
 C3 CC 27 JP DELAY
 37E2H C1 POP BC
 3E 08 LD A, 8
 INITIAL YELLOW
 COLOR REG 5
 OUTPUT COLOR REG 5 TO VALUE IN B
 SAVE COLOR / REG 5
 HL = RET ADDR
 2 SEC DELAY
 HL = RET ADDR
 COLOR / REG NOW BACK IN BC

121

-1

-22 168421
00010116
11101001
+1
11101010

80

47

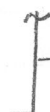
37E7H

18EA

ADD A, B

LD B, A

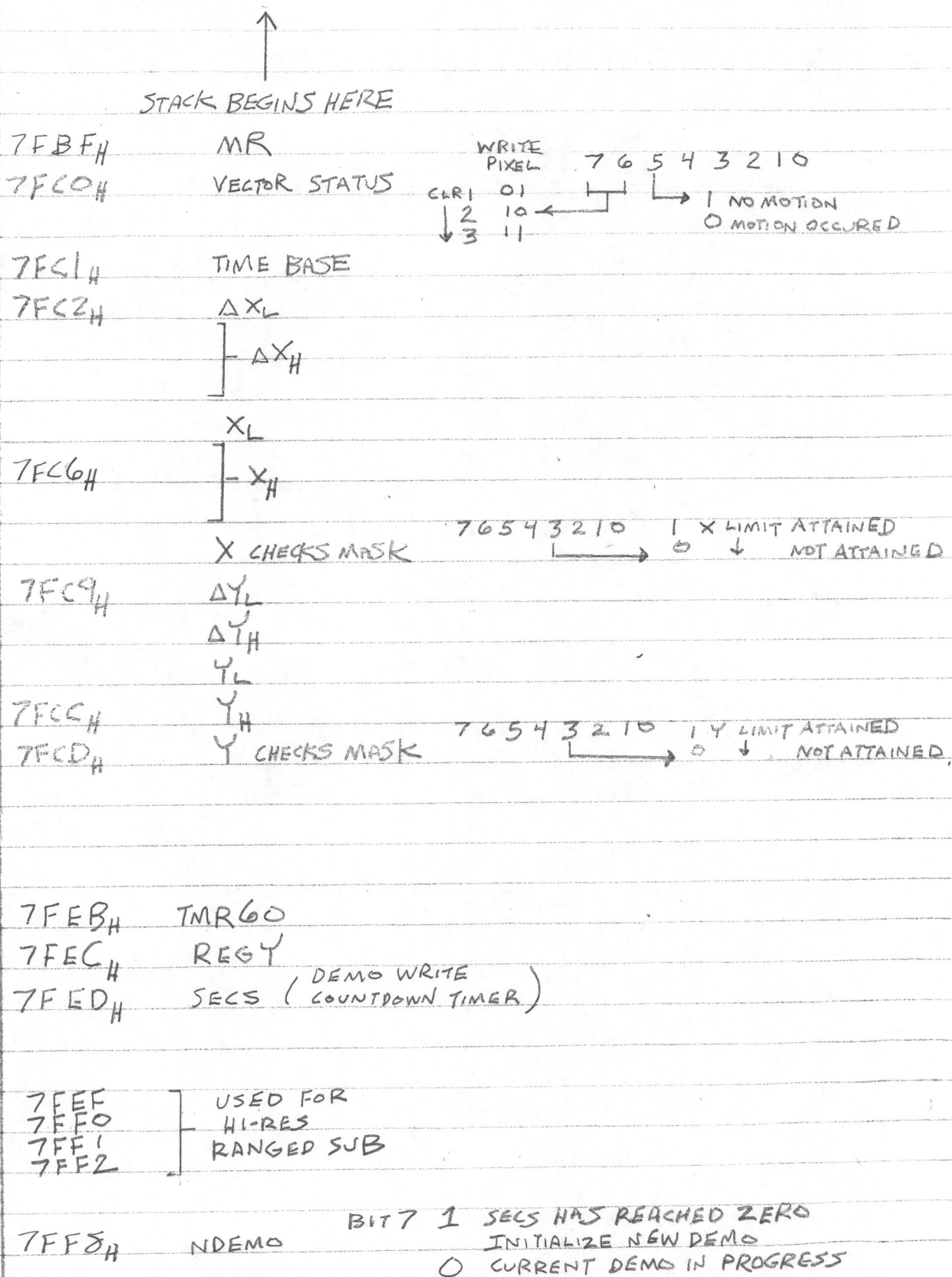
JR CHGCLR



CHANGE
COLOR

LOOP BACK TO OUTPUT COLOR
CONTINUOUS LOOP

HI-RES PIXEL STRINGER



123

1

PIXEL LIMITS TABLE

37E9 _H	00 00	X LOWER LIMIT	
	3F 01	X UPPER	319 _D = 013F _H
37ED _H	00	Y LOWER	
	CA	Y UPPER	202 _D = CA _H

LIMIT CHECK

LCHK	37EF _H	E5	PUSH HL	SAVE LIMITS POINTER	
	37F0 _H	6B ← BOY	LD L, E	] HL = UPDATED X _H (NEW)	
		62	LD H, D		
		A7 ← DOG	AND A	CARRY = 0	
		ED 42	SBC HL, BC	HL = NEW X _H - LIMIT - CARRY	] COMPARE NEW X _H WITH LIMIT
		E1	POP HL		
	37F6 _H	C9	RET		

* COLOR TABLE IS OUTSIDE RAM AREA OF THIS
HI-RES PIXEL STRINGER DEMO

2

124

START PROGRAM

(MISSING

DI)

PSTRT 37F7H 3E01

D308

LDA,01

OUT(08H),A

SET CUSTOM CHIPS TO OPERATE IN
HIGH RESOLUTION MODE

37FBH 31BF7F

FF

LD SP,7FBFH

RST 38H

INITIALIZE STACK POINTER

00

INTPC

BEGIN MULTIPLE CALLS

3800H

14

EMUSIC

TURN OFF SOUND

19

COLSET+1

SET COLORS

CC 3E

COLOR TABLE *

17

SETOUT+1

SET

CB

VERT BLK REG

BACKGRD
COLOR
01 10 10 10
420

VERT BLANK REG

6A

HOR CLR BNDRY

BACKGRD COLOR
HORIZONTAL CLR BNDRY
INTERRUPT MODE

3807H

08

INTMODE

02

XINTC

END MULTIPLE CALLS

3E59, B.Y

LDA,59D

32EB7F, D.O.G

LD(TMR60),A

SET TMR60=59D

D.O.G

32ED7F

LD(SECS),A

SET SECS=59D

3811H

CD 7C 39

CALL SUINT

SET UP / ACTIVATE
SCREEN INTERRUPT

INITIALIZE DEMO RE-ENTRY HERE

3814H

AF

XORA

CLEAR
NEW DEMO FLAG

32F87F

LD(NDEMO),A

FF

RST 38H

3817H

1B, B.Y

FILL+1

CLEAR
VECTOR PACKET

BF7F

START ADR

0F00

OF BYTES TO FILL

00

FILL WITH ZERO

21 C0 7F

LD HL,7FC0H

3822H

36 40

LD(HL),40

VECTOR STATUS = 40H
BITS 7,6 = 01 PIXEL CLR OT
BIT 5 = 0 MOTION OCCURED
TIME (FOR 1ST WRITE)

23

INC HL

POINT HL AT BASE

3825H

CD E1 39

CALL CINIT

CONTINUE
INITIALIZATION

125

WRITE PIXEL BYTE ROUTINE

3

WPIX 3828_H DD 21 BF 7F ^{← BOY} LD IX, 7FBF
 DD CB 01 6E BIT 5, (IX+1)
 3830_H 20 57 JR NZ, USTRVP ^{DOG}
 DD 7E 0D ^{← DOG} LD A, (IX+0D)
 32 EC 7F LD (REGY), A
 DD 5E 07 LD E, (IX+7)
 DD 56 08 LD D, (IX+8)
 DD CB 09 5E BIT 3, (IX+9)
 3842_H 20 06 JR NZ, WPAT1
 DD CB 0E 5E BIT 3, (IX+0E)
 28 0E JR Z, WPAT3
 WPAT1 3A C0 7F LD A, (VECTOR STATUS)
 E6 C0 AND 1100 0000
 C6 40 ADD A, 0100 0000
 3851_H 30 02 JR NC, WPAT2
 3E 40 LD A, 0100 0000
 32 C0 7F LD (7FC0_H), A
 21 C0 7F LD HL, 7FC0_H
 385B_H 3E 20 LD A, ^{MAGIC} VALUE
 WPAT2
 WPAT3

POINT IX AT PACKET
 CHECK MOTION BIT 5 IN VECTOR STATUS. 1 NO MOTION
 IF NO MOTION OCCURED, 0 MOTION OCCURS
 SKIP PIXEL WRITE
 REGY = Y COORDINATE (Y_H)
 DE = X COORDINATE = X_H
 IF X LIMIT ATTAINED,
 JMP TO WPAT1
 IF Y LIMIT IS NOT ATTAINED,
 JMP TO WPAT3
 A = VECTOR STATUS
 ISOLATED BITS 7 AND 6
 IN VECTOR STATUS
 CHANGE COLOR OF
 WRITE PIXEL TO
 NEXT COLOR
 (01, 10 OR 11)
 SAVE NEW CLR IN VECTOR STATUS
 POINT HL AT PIXEL BYTE TO WRITE
 SET UP MAGIC VALUE
 FOR "XOR" WRITE

ASSUME
 BITS 5
 IS
 ALREADY
 ZERO → WPAT2

 DETERMINE MAGIC ADDRESS
 (SIMILAR TO RELTA1 @ 29B8_H)

E5 PUSH HL
 E6 78 AND 78_H
 3860_H 6F ^{← BOY} LD L, A
 7B LDA, E
 E6 03 AND 03
 BOY → B5 ORL
 F5 PUSH AF
 6B LD L, E
 62 LD H, D
 E5 PUSH HL
 3869_H 3A EC 7F LD A, (REGY)

SAVE PIXEL BYTE ADR
 CLEAR SHIFT AMOUNT IN
 MAGIC REGISTER VALUE
 "OR" SHIFT AMOUNT WITH
 MAGIC REGISTER VALUE
 SAVE MAGIC REG VALUE
 HL = X COORDINATE

386CH	6F	LD L, A	HL=Y
	26 00	LD H, 0	
	29	ADD HL, HL	HL=2Y
3870H	29		HL=4Y
	29		HL=8Y
	29		HL=16Y
	54 ← DOG	LD D, H	DE=16Y
	5D	LDE, L	HL=MAGIC ADDRESS
	29	ADD HL, HL	HL=32Y
	29	ADD HL, HL	HL=64Y
	19	ADD HL, DE	HL=80Y
DOG →	D1 ← BOY	POP DE	DE=X
	CB 1A	RRD	
	CB 1B	RRE	DE=X'
	CB 3B	SRLE	
	16 00	LD D, 0	
3881H	19	ADD HL, DE	HL=80Y+X'
	EB ← BOY	EX DE, HL	DE=MAGIC ADR UPDATED
	F1	POP AF	A=MAG REG VALUE
	E1	POP HL	HL=PIXEL ADR=7FC0H

WRITE THE PIXEL (BYTE)

	D3 0C	OUT (MAGIC), A	OUTPUT MAGIC VALUE (XOR IT PX SHIFT)
	7E	LDA, (HL)	GET PIXEL BYTE
3888H	12	LD (DE), A	WRITE IT TO MAGIC RAM

PROGRAM CONTINUED NEXT PAGE

UPDATE COORDINATES IN VECTOR PACKET

(SIMILAR TO SUB#62, REVISED FOR HI-RES "PIXEL STRINGER" DEMO.)

NO NEED TO CHECK ACTIVE BIT 7 IN VECTOR STATUS, TIME BASE IS ALREADY INITIALIZED

USTRVP 3889H DD21BF7F LD IX, 7FBFH POINT IX AT VECTOR PACKET
 21E937 LD HL, 37E9H HL AT LIMITS TABLE
 3890H DD4E02 LDC, (IX+2) C = TIME BASE
 DD CB01EE SET 5, (IX+01) USE BIT 5 IN VECTOR STATUS AS MOTION BIT 1 = NO MOTION
 SET BIT = NO MOTION 0 = MOTION OCCURRED

110300 LD DE, 3
 DD19 ADD IX, DE

POINT IX AT ΔXL

UPDATE COORDINATE X

VECTX E5 PUSH HL SAVE LIMITS POINTER

DD5602 LDD, (IX+2)
 38A0H DD5E01 LDE, (IX+1)
 DD6605 LDH, (IX+5)
 DD6E04 LDL, (IX+4)

DE = ΔX_H HL = OLD X_H

E5 PUSH HL SAVE OLD X_H

41 LDB, C B = TIME BASE = LOOP CTR
 DD4E00 LDC, (IX) IX $\rightarrow$ ΔXL
 DD7E03 LDA, (IX+3) C = ΔXL
 A = X_L
 38B1H 81 ADD A, C A = $X_L + \Delta X_L$
 ED5A ADC HL, DE HL = OLD $X_H + \Delta X_H + \text{CARRY}$
 10FB DJNZ VCTX1

ADD DELTA TO COORDINATE "TIME BASE" TIMES

LOAD UPDATED (NEW) X_L IN PACKET NOW

DD7703 LD (IX+3), A
 C1 POP BC BC = OLD X_H
 5D $\leftarrow$ DOG LDE, L DE = UPDATED (NEW) X_H
 54 LDD, H

A7 $\leftarrow$ DOG AND A
 ED42 SBC HL, BC HL = NEW $X_H - \text{OLD } X_H - \text{CARRY}$

IF MOTION HAS OCCURRED, RESET BIT 5 OF VECTOR STATUS

2804 JR Z, VCTX1A
 38C1H DD CBFEAE RES 5, (IX-2)

-5 0000 0101
 1111 1010
 41

-2 0000 0010
 1111 1101
 76

128

DE = UPDATED (NEW) X_H

6

VTX1A 38C5_H E1

POP HL

POINT HL AT LOWER LIMIT

PROCEED TO CHECK IF LIMIT WAS ATTAINED

4E	LD C, (HL)	] BC = LOWER LIMIT POINT HL AT UPPER LIMIT
23	INC HL	
46	LD B, (HL)	
23	INC HL	
C5	PUSH BC	SAVE LOWER LIMIT

CHECK LOWER / UPPER LIMITS

-48 0 → 319 +48 (SIMILAR TO LOW-RES VECT ROUTINE)

38CB _H 01 6F 01	LD BC, 367 _D	] COMPARE NEW X_H WITH LIMIT BC = LOWER LIMIT	HANDLE < 0 CASE. ANY $X > 367_D$ IS CONSIDERED NEGATIVE.
CD EF 37	CALL LCHK		
38D1 _H C1	POP BC		
30 0E	JRNC, VCTX2		
CD EF 37	CALL LCHK	] IF NEW X_H < LOWER LIMIT JMP TO VCTX2	
38 09	JRC, VCTX2		
4E	LD C, (HL)	] BC = UPPER LIMIT	IF NEW X_H < UPPER LIMIT JMP TO VCTX3
23	INC HL		
46	LD B, (HL)		
CD EF 37	CALL LCHK		
38 2E	JRC, VCTX3		
38E1 _H 2B ← 80Y	DEC HL		
VCTX2 23	INC HL		
23	INC HL	POINT HL AT Y LIMITS	
DD 71 04	LD (IX+4), C	] X_H = LIMIT	
DD 70 05	LD (IX+5), B		
DD 36 03 00	LD (IX+3), 0		$X_L = 0$
DD CB 06 DE	SET 3, (IX+6)	SET X LIMIT ATTAINED BIT	

PROCEED WITH REVERSE THE DELTA

38F2 E5

PUSH HL

SAVE LIMITS POINTER

(POINTS AT Y LOWER LIMIT)

REVERSE ΔX (2'S COMPLEMENT)

$\swarrow$ 2 PAGES
 DDE5 PUSH IX
 DT⁰⁰⁶ POP DE] POINT DE AT ΔX_L
 1A LD A, (DE) A = ΔX_L
 2F CPL] 2'S COMPLEMENT ΔX_L
 38F8H 601 ADD A, 1
 12 LD (DE), A
 13 INC DE
 1A LD A, (DE)
 2F CPL
 6F LD L, A] HL = COMPLEMENT ΔX_H
 13 INC DE
 3960H 1A LD A, (DE)
 2F CPL
 67 LD H, A
 010000 LD BC, 0
 ED4A ADC HL, BC HL = HL + BC + CARRY
 EB^{B0Y} EX DE, HL] ADD CARRY
 72 LD (HL), D] TO $\overline{\Delta X_H}$
 2B^{B0Y} DEC HL
 73 LD (HL), E
 E1^{B0Y} POP HL POINT HL AT Y LOWER LIMIT
 180B JR VECTY JMP TO UPDATE Y COORDINATE

VCTX3 390FH 23 INCHL POINT HL AT Y LIMITS
 3910H DD 73 04 LD (IX+4), E] SET X_H IN PACKET
 DD 72 05 LD (IX+5), D] TO UPDATED (NEW) X_H
 DD CB 06 9E RES 3, (IX+6) CLEAR X LIMIT ATTAINED BIT

UPDATE COORDINATE Y

ENTER WITH: HL POINTING TO Y LIMITS

IX POINTING TO ΔX_L IN VECTOR PACKET

VECTY	391A DD4EFF	LD C, (IX-1)	$\leftarrow$ TIME BASE	} POINT IX AT ΔY_L
	11 0700	LD DE, 7		
	3920 DD19	ADD IX, DE		
	200GS			
	E5	PUSH HL	SAVE Y LIMITS POINTER	
	DD5601	LD D, (IX+1)	} DE = ΔY	
	DD5E00	LD E, (IX+0)		
	DD6603	LD H, (IX+3)	} HL = $Y_H Y_L$	
	DD6E02	LD L, (IX+2)		
	7C	LDA, H	A = Y_H	
	3930 41	LDB, C	B = TIME BASE = $\leftarrow$ TR	SAVE Y_H
	19	ADD HL, DE		ADD DELTA TO COORDINATE "TIME BASE" TIMES
	10 FD	DJNZ VECT1		
	BC	CP H		IF NO CHANGE IN Y, JMP TO VECT1A
	2804	JRZ, VECT1A		
	DDCBF7AE	RES 5, (IX-9)		MOTION OCCURED RESET MOTION BITS IN VECTOR STATUS
	7C	LDA, H	A = Y_H	
	E3	EX (SP), HL	TOP OF STACK = Y COORDINATE	
	46	LD B, (HL)	HL = Y LIMITS POINTER	
	23	INC HL	B = Y LOWER LIMIT	
	FE FA	CP 250D	POINT HL AT Y UPPER LIMIT	
	3941 3007	JR NC, VECT2		-48 0-203D +48
	B8	CP B		IF $Y_H < Y_{UPPER LIMIT}$, JMP VECT2
	3804	JR C, VECT2		
	46	LD B, (HL)	B = UPPER LIMIT	
	B8	CP B		
	381B	JR C, VECT3		
	DD7003	LD (IX+3), B		VECTOR Y_H IN PACKET = NEW Y_H
VECT2	394D DD360200	LD (IX+2), 0	$Y_L \downarrow = 0$	

-9 00001001
11110110PROCEED WITH
LIMIT CHECK
VECT1A

BOY

BOY

VECT2

* IM2 VECTOR IS OUTSIDE ROM CODING AREA FOR THIS DEMO

131 9

3951H DD CB04 DE SET 3, (IX+4) SET Y LIMIT ATTAINED BIT 3
FI POP AF CLEAN UP STACK

PROCEED WITH
REVERSE THE DELTA

7A LD A, D

2F CPL

57 LDD, A

7B LD A, E

2F CPL

5F LDE, A

13 INC DE

DD 7300 LD (IX+0), E

3960H DD 7201 LD (IX+1), D

180C JR, A3971

REVERSE DELTA
(2'S COMPLEMENT)

LOAD REVERSE DELTA
IN VECTOR PACKET

JMP TO CONTINUE PROGRAM

VECT3

E3 EX (SP), HL

HL = YH YL

DD 7502 LD (IX+2), L

DD 7403 LD (IX+3), H

E1 POP HL

CLEAN UP STACK

DD CB04 9E RES 3, (IX+4) RESET LIMIT ATTAINED BIT

STORE COORDINATE
IN VECTOR PACKET

3971H 21 F8 7F LD HL, 7FF8H

80Y 7B 7E BIT 7, (HL)

C2 14 38 JPNZ, INDEM

3979H C3 28 38 JP WPAT

INITIALIZE NEW DEMO
WHEN SECS COUNTER
REACHES ZERO

BIT 7 @ 7FF8H = 1

LOOP BACK TO
WRITE NEXT PIXEL BYTE

NOTE:

ADD VECTOR

AT 2012H

HI-RES PIXEL STRINGER DEMO

IM2 VECTOR SET UP

VECTOR @ 2012H * (2012H 8D 39)

DOG

ADDRESS OF
INTERRUPT ROUTINE

SUINT 397CH F3

3E 20

ED 47

DI

LD A, 20H

LD I, A

OUTPUT PAGE OF VECTOR

132

10

3981H 3E12

D30D

ED5E

D06 3EC8

D30F

FB

398CH C9

LD A, 12H
OUT (0DH), A

IM2

LDA, C8H
OUT (0FH), A

EI

RET

] OUTPUT
LINE OF VECTOR] GENERATE INTERRUPT
AT 200 LINE SCANS

ENABLE INTERRUPTS NOW

HI-RES PIXEL STRINGER INTERRUPT ROUTINE
 VECTOR @ 2012H (2012H 8D39)
 DECREMENT SECONDS COUNTER (NO BCD AT SECONDS)
 REQUIRED

SINT 398DH F3
F5DI
PUSH AFDISABLE INTERRUPTS
SAVE A AND HL ONLY

E5

21EB7F ^{BOY}

3993H 35

F2 A539

363B ^{BOY}21ED7F ^{D06}

35

PUSH HL

LD HL, TMR60

DEC (HL)

JP P, SINT2

LD (HL), 59D

LD HL, SECS

DEC (HL)

] DECREMENT *
TMR60YOUR DONE IF NOT ZERO,
OTHERWISE, EXECUTE "ONCE A SEC" ROUTINE

SET TMR60 = 59D

] DECREMENT SECONDS
COUNTDOWN TIMER *

F2 A539

39A0H 21F87F

CBFE ^{BOY}

JPP, SINT2

LD HL, NDEMO

SET 7, (NDEMO)

] IF SECS = 0
SET NEW DEMO BIT 7 FLAG
(TO INDICATE TIME TO
INITIALIZE NEW DEMO)SINT2 39A5 EI
FI

POP HL

POP AF

RESTORE HL AND A

FB

39A8H C9

EI

RET

ENABLE INTERRUPTS

* TMR60 IS INITIALIZED TO 59D BY MAIN PROGRAM
 SECS

RANGED (FOR HI-RES PIXEL STRINGER DEMO)

ENTER WITH : A = MAXIMUM NUMBER TO GENERATE + 1
SIMILAR TO LOW-RES SUB#118

PSRANG	39A9 _H	F5	PUSH AF	
		2A EF 7F	LD HL, 7FEF	
		CD 8A 2D	CALL SHIFTR	
	39B0 _H	01 17 00	LD BC 23 _D	
		09	ADD HL, BC	
		8A	ADC A, D	
		22 EF 7F	LD (7FEF _H), HL	
		2A F1 7F	LD HL, 7FF1	
		5F	LDE, A	
		CD 8A 2D	CALL SHIFTR	
		19	ADD HL, DE	
	39C0 _H	22 F1 7F	LD (7FF1 _H), HL	
	39C3 _H	C3 79 2D	JPA 2D.79	CONTINUE PSRANGE AT ANOTHER LOCATION

FACTORY TEST (SEE "FACTORY TEST" INSTRUCTIONS)

VARY ROUNDS INPUT FOR NEW CHECKMATE DEMO

SEE NOTE BELOW

39C6 _H	FF	RST 38 _H	ACTIVATE INTERRUPT
	0E	ACTINT	
	11 C0 3B	DE, DEMO	
	FF	RST 38 _H	
	4D ← DOG	GETPAR+1 (SUB#78)	GET # OF ROUNDS
	C5 3B	INPUT STRING	INPUT FROM
	82	MESSAGE GET ADR	KEYPAD. STASH
	DC 4F	CHAR DISPLAY OPTIONS	INPUT AT 4FDC _H
		STASH ADDRESS	
39D2	C3 D0 3E	JPA 3ED0	JMP TO PROGRAM NEW CHECKMATE DEMO

NOTE: FOR ABOVE FACTORY TEST, THE 2 CHARACTER STRINGS
ARE AT: 3BC0_H DEMO AND 3BC5_H ROUND

134

12

CONTINUE

PROGRAM TEST DUAL DISPLAY / DISPLAY TEST PATTERNS

TDUOC 39D5H C3 CC 27

JP DELAY 1 SEC DELAY

39D8H 0C

INC C INCREMENT DISPLAY CONTINUE

39D9H C3 E4 32

JP TDUALC LOOP BACK TO UPDATE COUNTER

39DC H FF FF FF

UNUSED BYTES

39DE H FF FF

↓

CONTINUE HI-RES PIXEL STRINGER DEMO INITIALIZATION

CINIT

39E1H 36 03

LD (HL), 03

SET TIME BASE = 03

FF

RST 38H

1B ← BYT

FILL + 1

CLEAR SCREEN

00 40

START ADR

203 LINES

70 3F

OF BYTES
TO FILL

203 x 80 =

00

FILL WITH 00H

16,240 BYTES

39EAH 3E 0B

LD A, 0BH

RANDOMIZE A REG
FROM 0-10D

CD A9 39

CALL PSRANG

21 B0 7F

LD HL, 7FB0H

HL = STACK ADR
(TOP OF INDEX)

39F2H

85

ADD A, L

INDEX 0-10D

6F

LD L, A

FROM A BYTE

39F4H

7E

LD A, (HL)

IN STACK

CB 47

BIT 0, A

0 = VARY X
1 = + Y

28 0A

JR Z, VARYX

VARY X OR Y

VARY

3E C6

LD A, 198D

INITIALIZE
VARYED

CD A9 39

CALL PSRANG

Y COORDINATE

32 CC 7F

LD (7FCC H), A

IN
VECTOR PACKET

3A01H

18 08

JR VDELX

OR

VARX

3E FF

LD A, FF

INITIALIZE
VARYED

CD A9 39

CALL PSRANG

X COORDINATE

3A08H

32 C6 7F

LD (7FC6 H), A

IN
VECTOR PACKET

13

135

VARY ΔXL AND ΔYL

VDELX	3A0B _H	3E FA	LD A, FA	] INITIALIZE VARIED ΔXL IN VECTOR PACKET MIN 6 MAX FF
		CD A939	CALL PSRANG	
	3A10 _H	06 06	LD B, 6	
		80	ADD A, B	
		32 C27F	LD (7FC2 _H), A	] IF ΔXL IS HIGH, JMP TO MAKE ΔYL LOW. MAX OUT ΔYL
		FE 30	CP 30 _H	
		38 04	JR C, VDELY1	
VDELY		3E FA	LD A, FA	
		18 02	JR VDELY2	] KEEP ΔYL LOW INITIALIZE VARIED ΔYL IN VECTOR PACKET MIN 6 MAX FF
VDELY1		3E 30	LD A, 30 _H	
VDELY2	3A20 _H	CD A939	CALL PSRANG	
		06 06	LD B, 6	
		80	ADD A, B	] SET WRITE TIME TO 16 SEC
		32 C97F	LD (7FC9 _H), A	
		3E 10	LD A, 16 _D	
	3A2B _H	32 ED7F	LD (7FED _H), A	

VARY COLORS 01, 02 AND 03

REFERENCE BALLY/ASTROCADE COLOR TABLE AT BallyAlley.com

VCOL3	3A2E _H	CD 4C3A	CALL RCOL0	DO COLOR RANGE
	3A31 _H	C6 05	ADD A, 0000 0101	A = COLUMN 05 COLOR
		4F	LD C, A	] C = COL3
		C5	PUSH BC	
VCOL2		CD 4C3A	CALL RCOL0	] A = COLUMN 04 COLOR B = COL2
		C6 04	ADD A, 0000 0100	
		C1	POP BC	
		47	LD B, A	
		C5	PUSH BC	] C = COL3
VCOL1	3A3D _H	CD 4C3A	CALL RCOL0	

136

3A40H	C6 02	ADD A, 0000 0010	A = COLUMN 1 COLOR = COL 1
	C1	POP BC	B = COL 2, C = COL 3
	D3 05	OUT (05), A	OUTPUT CLR TO REG 5
	78	LD A, B	
	D3 06	OUT (06), A	REG 6
	79	LD A, C	
	D3 07	OUT (07), A	REG 7
	C9	RET	

RANGE COLOR COLUMN C

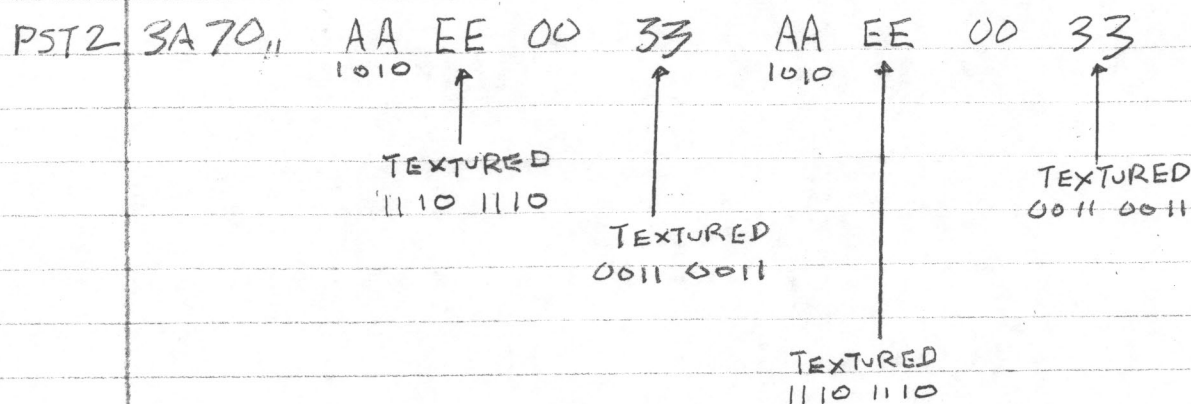
Rcolo	3A4CH	3EFF	LDA, FF
		CD A9 39	CALL PSRANG
	3A51H	E6 F8	AND 1111 1000
		20 02	JR NZ, A3A5T
		3E 08	LD A, 08
	3A57H	C9	RET

TEST DUAL 7-SEGMENT DISPLAY DISPLAY TEST PATTERNS

TDUAL	3A58H	31CE 4F	LD SP, 4FCEH	
		FF	RST 38H	
		00	INT PC	BEGIN MULTIPLE CALLS
		14	EMUSIC (SUB#20)	STOP SOUND
		17	SETOUT+1 (SUB#22)	SET
		C6	VERT BLK REG	VERTICAL BLANK REG = 198
	3A60H	14	OUTPUT PORT 9	HOR CLR BNDY = 200, SPLIT SCREEN
		08	INT MODE	BKGRD CLR = PIXEL 00
		19	COLSET+1	
		80 3A	COLOR TABLE	SET COLORS
		02	XINIC	END MULTIPLE CALLS
		3E 55	LD A, 0101 0101	INITIALIZE BORDER CLR TO COLOR 01
	3A68H	CD 84 3B	CALL FBDR	FILL BORDER
		0E 00	LDC, 0	DISPLAY
				(= INITIALIZE COUNT TO ZERO
	3A6DH	C3 E4 32	JP TDUALC	JMP TO CONTINUE PROGRAM

WRITE HI-RES TEXTURED TEST PATTERN

PIXEL SET TABLE 2 (TEXTURE SWAP)



PIXEL SET TABLE 1

PST1 3A78_H 33 00 AA EE 33 00 AA EE

TEXTURED TEXTURED TEXTURED TEXTURED

INITIAL COLOR TABLE (FOR LOW AND HI-RES TEST PATTERNS)

ICT	3A80 _H	AC	GREEN	PIXEL 11	
		86	YELLOW	10	LEFT COLORS
		07	WHITE	01	
		00	BLACK	00	
		CD	CYAN	PIXEL 11	
	3A85 _H	5A	RED	10	RIGHT COLORS
		2B	MAGENTA	01	
		F9	BLUE	00	

WRITE TEST PATTERN

ENTER WITH: DE = PIXEL SET TABLE ADR

WPAT 3A88_H 214441 LD HL, 4144_H HL = PATTERN START ADDRESS

3A8B_H 0608 LD B, 8 B = # OF BARS = LOOP COUNTER

NOTE: SUBROUTINE "FILL" @ 29F0H IS OUTSIDE THIS HI-RES TEST PATTERN ROM AREA.

2

138

BAR 3A8D_H E5
C5
1A
3A90_H 47
DOG → 13
D5
1109C1

PUSH HL SAVE BAR SCREEN ADR
PUSH BC SAVE LOOP CTR
LDA, (DE) A = PIXEL BAR BYTE TO WRITE
LDB, A B =
INC DE POINT DE AT NEXT PIXEL BAR BYTE
PUSH DE SAVE PIXEL BAR BYTE POINTER
LD DE, YSIZE/XSIZE D = # OF LINES = 193_D
E = # OF BYTES/LINE = 9

NOTE
SUB "FILL"
IS @ 29F0H →

DOG → CD F0 29
D1
C1
E1 ← DOG
7D
C6 09
6F ← BOY
3AA0_H 10EB
C9

CALL FILL (SEE NOTE ABOVE)
POP DE DE = NEXT PIXEL BAR BYTE POINTER
POP BC B = LOOP CTR
POP HL HL = LAST BAR SCREEN ADR
LDA, L
ADD A, 9_D
LDL, A
DJNZ BAR
RET

POINT HL
AT NEXT
BAR SCREEN START ADR

7 SEC DELAY (SIMILAR TO DELAY @ 27CC_H)

TPDLY 3AA3_H 16 10
TPDLY1 3EFF
TPDLY2 06 FF
TPDLY3 10 FE
3D ← DOG
20 F9
15
20 F4
3AB1_H C9

LDD, 1A_H
LD A, 255_D
LDB, 255_D
DJNZ TPDLY3
DECA
JRNZ, TPDLY2
DEC D
JRNZ, TPDLY1
RET

NOTE: SUBROUTINE "PSRANGE" IS OUTSIDE THIS HI-RES TEST PATTERN ROM AREA. 3

"DARK COLORS" VARIATION TABLE

139

DKCLR 3AB2_H 00 BLACK
10 DARK BLUE
48 LIGHTER RED
79 BROWN
70 DARK BROWN
C0 DARK GREEN

"OTHER COLORS" VARIATION TABLE

OTCLR 3AB8_H 02 LIGHT GRAY
07 WHITE
09 LIGHTER BLUE
2B^{80Y} PINK
5A RED
63 MELON
6B NEAR ORANGE
71 LIGHT BROWN
3AC0_H 72 NEAR TAN
73 ORANGE/BROWN
7E SHADE OF YELLOW
86 YELLOW
92 MEDIUM GREEN
E4 SHADE OF LIGHT BLUE

SELECT WITH VARIATION DARK COLOR

EXIT WITH DARK COLOR IN A REG

SDARK	3AC6 _H	21 B23A	LD HL, DKCLR	POINT HL AT "DARK COLORS" TABLE
SDARK1	E5		PUSH HL	SAVE THIS TABLE ADDR
	DOG D5		PUSH DE	SAVE RAM COLOR TABLE POINTER
	3E06		LDA, C0	RANGE COLOR INDEX 0 THRU 5
NOT PSRANGE IS AT 39A9 _H	→ 3ACD _H	CD A939	CALL PSRANGE	EXIT A = COLOR INDEX SEE NOTE ABOVE

140

3AD0^{DOG}H

DI

EI

4F

7D^{DOG}

8I

6F

7E

C9

POP DE

POP HL

LDC, A

LDA, L

ADD A, C

LD L, A

LDA, (HL)

RET

DE POINTS AT RAM COLOR TABLE

HL = COLORS TABLE ADR

INDEX

SELECTED COLOR.

PUT IT IN A REG

SELECT WITH VARIATION OTHER COLOR

EXIT WITH "OTHER COLOR" IN A REG

SOTHR 3AD8^H 21 B8 3A

LD HL, 0TCR

POINT HL AT "OTHER CLRS" TABLE

3E 0E

LDA, 14_DRANGE INDEX 0 THRU 13_D

18 EA

JR SPARK 1

PUT SELECTED COLOR IN A REG

-22 0001 0110
 1110 1001
 +1
 1010

SET UP DARK AND "OTHER" COLORS IN RAM COLOR TABLE

ENTER WITH DE = RAM COLORTABLE ADR + 4 (POINTING AT ^{RIGHT} PIXEL 11 COLOR)

SRAMT

3ADF^H

1B

DEC DE

POINT DE AT LEFT DARK COLOR

3AE0^H

CD C6 3A

CALL SDARK

SELECT DARK COLOR

12

^{DOG}

LD (DE), A

PUT IT IN RAM ^{COLOR} TABLE AS ^{LEFT} PIXEL 00 COLOR

1E DC

LDE, DC^H

POINT DE AT RIGHT DARK COLOR

CD C6 3A

CALL SDARK

SELECT DARK COLOR

12

LD (DE), A

PUT IT IN RAM ^{COLOR} TABLE AS ^{RIGHT} PIXEL 00 COLOR

1E D6

LDE, D6^H

POINT DE AT LEFT "OTHERS" COLOR

CD D8 3A

CALL SOTHR

SELECT OTHER COLOR

12

LD (DE), A

PUT IT IN RAM ^{COLOR} TABLE AS ^{LEFT} PIXEL 10 COLOR3AF0^H

1E DA

LDE, DA^H

POINT DE AT RIGHT "OTHERS" COLOR

CD D8 3A

CALL SOTHR

SELECT OTHER COLOR

12

LD (DE), A

PUT IT IN RAM ^{COLOR} TABLE AS ^{RIGHT} PIXEL 10 COLOR

C9

RET

* COLOR COLUMNS DEFINED FOR 256 ASTROCADE PALETTE. SEE FULL COLOR
 PALETTE TABLE AT BALLY ALLEY / ML PROGRAMMING / ML DOCUMENTATION

141 5

HI-RES TEST PATTERN

SCREEN RAM COLOR TABLE

RCTB	7FD5 _H	COLOR REG 7	PIXEL 11	GREEN A _C _H	
		6	10	"OTHER COLORS" VARIATION	LEFT
		5	01	NOT USED	COLORS
		4	00	"DARK COLORS" VARIATION	
		3	PIXEL 11	CYAN C _D _H	
		2	10	"OTHER COLORS" VARIATION	RIGHT
		1	01	NOT USED	COLORS
	7FDC _H	0	00	"DARK COLORS" VARIATION	

HI-RES TEST PATTERN PROGRAM

HRTB	3AF7 _H	F3	PI	DISABLE INTERRUPTS
	3E 01		LDA, 01 _H	MAP SCREEN IN HI-RES
D0G	D3 08		OUT (08 _H), A	
	31 CE 7F		LD SP, 7FCE	INITIALIZE Z80 STACK
	AF		XOR A	A=0
3B00 _H	01 18 08		LD BC, 0818 _H	STOP ALL SOUND
D0G	ED 79		OUT (<), A	(SIMILAR TO SUB#20)
	10 FC		DJNZ-2	
	3E C9		LD A, 20 _D	SET
D0G	D3 0A		OUT (0A _H), A	VERTICAL BLANK REG
	3E 14		LDA, 20 _D	To 20 _D
D0G	D3 09		OUT (09 _H), A	00010100 HOR CLR BOUNDARY
	21 80 3A		LD HL, 3A80 _H	To 20 _D (SPLIT SCREEN)
3B12 _H	01 0B 08	BOY	LD BC, 080B _H	BACKGRD CLR=PIXEL 00
D0G	ED B3	BOY	OTIR	SET COLOR FOR INITIAL
				(COLOR NOT RANGED)
				TEST PATTERN
(CLEAR SCREEN - ONCE ONLY)				
	11 00 40		LD DE, 4000 _H	START
	01 D0 3E		LD BC, 3ED0 _H	DE = ADR
				BC = # OF BYTES
				TO CLEAR

NOTE: SUBROUTINE "FILL" @ 29F0H IS OUTSIDE THIS HI-RES TEST PATTERN ROM AREA.

142

6

3B1D_H AF _{BOY} X0RA FILL WITH ZERO
MALL EB EX DE, HL
M.FILL 1 3B1F_H 77 _{DOG} LD (HL), A
3B20_H EDAI _{BOY} CPI
EA 1F3B JP DE, M.FILL 1

CLEAR SCREEN
201 LINES X 80 BYTES/LINE
TOTAL 16,080 BYTES
(SIMILAR TO
FILL AREA, SUB#26)

FILL UNDERLYING BORDER AROUND TEST PATTERN
ONCE ONLY FILL

114AC9

LD DE, YSIZE/XSIZE

0622

LD B, 22_H

FILL WITH
0010 0010

FILL WITH TEXTURED
BORDER

210340

LD HL, 4003_H

BEGIN BORDER
FILL AT 4003_H

CD F029

CALL FILL (SEE ABOVE NOTE)

WRITE INITIAL (COLOR NOT RANGED) TEST PATTERN

3B30_H 11783A

LD DE, PST1

DE = PIXEL SET TABLE 1 ADR

CD 883A

CALL WPAT

WRITE TEST PATTERN

7 SECONDS DELAY

3B36_H CD A33A

CALL TPDLY

NOTE
SUB "FILL"
IS AT
29F0_H

REPEAT →
HERE

WRITE TEST PATTERN WITH RANGED COLORS

3B39 _H	11 D5 7F	LD DE, 7ED5 _H	POINT HL AT RAM COLOR TABLE
	3E AC	LD A, GREEN	LOAD GREEN IN TABLE
	12	LD (DE), A	FOR LEFT PIXEL 11
	1E D9	LDE, D9 _H	LOAD CYAN IN TABLE
3B41 _H	3E CD	LD A, CYAN	FOR RIGHT PIXEL 11
	12	LD (DE), A	
	CD DF 3A	CALL SRAMT	SET UP DARK AND "OTHER" COLORS IN RAM COLOR TABLE
	21 D5 7F	LD HL, RCTB	OUTPUT
	01 0B 08	LD BC, 080B	RAM COLORS IN TABLE
	ED B3	OTIR	CHANGE COLORS IN 1ST TEST PATTERN
	CD A3 3A	CALL TPDLY	7 SEC DELAY
3B52 _H	11 D5 7F	LD DE, 7ED5 _H	
	3E CD	LD A, CYAN	
	12	LD (DE), A	SIMILAR TO ABOVE.
	1E D9	LD E, D9 _H	SWAP TEXTURE COLORS
	3E AC	LD A, GREEN	GREEN AND CYAN IN
	12	LD (DE), A	RAM COLOR TABLE.
	CD DF 3A	CALL SRAMT	
3B60 _H	21 D5 7F	LD HL, RCTB	OUTPUT
	01 0B 08	LD BC, 080B _H	RAM COLORS IN TABLE
	ED B3	OTIR	
	11 70 3A	LD DE, PST2	DISPLAY
	CD 88 3A	CALL WPAT	3RD TEST PATTERN
	CD A3 3A	CALL TPDLY	7 SEC DELAY
3B71 _H	21 80 3A	LD HL, ICT	HL = INITIAL COLOR TABLE ADDR
	01 0B 08	LD BC, 080B _H	
	ED B3	OTIR	
3B79 _H	C3 30 3B	JP REPEAT	

WRITE LOW-RES TEST PATTERN

COLOR TABLE

ICT	AC	GREEN	LEFT COLORS
	86	YELLOW	
	07	WHITE	
	00	BLACK	
	CD ^{←DOG}	CYAN	RIGHT COLORS
	5A	RED	
	2B ^{←BOT}	MAGENTA	
	F9	BLUE	

NOTE

THIS COLOR TABLE
IS AT 3A80H

PIXEL SET TABLE 3 (SET VERTICAL BAR COLOR)

PST 3 3B7C_H 00 55 AA FF 00 55 AA FF
BLK WHT YEL GRN BLU MAG RED CYN

FILL PATTERN BORDER

ENTER WITH A = FILL COLOR

FBDR	3B84 _H 11 00 40	LD DE, 4000 _H	DE = ^{START} ADR	FILL 99 LINES 99 × 40 = 3960 BYTES = 0F78 _H
	01 78 0F	LD BC, 0F78 _H	BC = ^{# OF BYTES} TO FILL	
	FF	RST 38 _H		
	1A	FILL (SUB#26, ^{NO} SUCK)		
	C9	RET		

WRITE 8 VERTICAL "TEST" BARS

ENTER WITH: DE = COORDINATES OF LEFT MOST BAR ^{D = Y COORD}
E = X ^{E = X}

HL = PIXEL SET TABLE ADDRESS

W8BR	3B8D _H 06 08	LD B, 8 _D	B = COUNTER (# OF VERTICAL BARS)
W8BR1	C5	PUSH BC	SAVE LOOP CTR
	7E	LD A, (HL)	A = PIXEL BYTE TO WRITE
	3B91 _H 23	INC HL	POINT HL AT NEXT "PIXEL BYTE" TO WRITE

145

NOTE: HL =
"NEXT BYTE TO WRITE"
IS SAVED IN
UPI CONTEXT BLOCK

-17 0001 0001
1110 1110
+1
1111

3B92 01 13 5F

D5

FF

1C

DOG → DI

7B ← BOT

C6 13

5F

C1

10 EF

3BA0H C9

LD BC, 5F13H

PUSH DE

RST 38H

RECTAN (SUB#28, NO SUCK)

POP DE

LDA, E

ADDA, 19D

LDE, A

POP BC

DJNZ W8BR1

RET

B = Y SIZE 99-4 = 95D
C = PIXEL X SIZE = 19D

SAVE Y, X COORD OF BAR

WRITE BAR

DE = Y, X COORD OF LAST BAR

POINT XCOORD AT
NEXT BAR TO WRITE

B = BAR (LOOP) CTR

LOOP BACK TO WRITE ANOTHER BAR

WRITE LOW-RES TEST PATTERN 1

TDSP	3BA1 _H	FF	RST 38 _H	} RESET INITIAL COLORS
		19	COLSET+1	
		80 3A	COLOR TABLE	
		21 7C 3B	LD HL, PST 3	
		18 0C	JR, TDSP 3	

WRITE LOW-RES TEST PATTERN 2

TDSP 1	3BAA _H	FF	RST 38 _H	} SET NEW COLOR
		19	COLSET+1	
		B4 33	COLOR TABLE	
		21 7C 3B	LD HL, PST 3	
	3BB1 _H	18 03	JR TDSP 3	

WRITE LOW-RES TEST PATTERN 3

TDSP 2	3BB3 _H	21 70 3A	LD HL, PST 2	
TDSP 3		11 04 02	LD DE, 02 04	DE = COORD OF
		CD 8D 3B	CALL W8BR	LEFT MOST BAR
	3BBL _H	C9	RET	D=Y=2 E=X=4

3BBD_H FF FF FF UNUSED BYTES

3BC0_H D E M O
44 45 4D 4F 00

3BC5_H R O U N D
52 4F 55 4E 44 00

R

CHECKMATE (REVISED DEMO) 147

BEGIN PROGRAM @ 3ED0H

NON STOP, NON REPETITIVE FAST ACTION WITH 100 ROUNDS.

NO "PIZZA BREAK" PAUSE.

NEW VIBRANT COLORS,

DISTINCT BACKGROUND FLASH DURING A COLLISION, BETTER THAN THE ORIGINAL.

"GAME OVER" CHANGED TO "DEMO OVER".

THIS IS A DEMO ONLY, WITH NO JOYSTICK CONTROL.

RELOCATED TO 3BCB-3EFFH WITH PROGRAM START AT 3ED0H.

EXECUTABLE VIA C3D03E AT 2000H OR MENU LINKED LIST.

USEABLE TO TEST A MOTHERBOARD FOR GRAPHIC ACTION PERFECTION.

A PLEASANT AUDIO/VIDEO EXPERIENCE FROM A BALLY/ASTROCADE.

DATA
BLOCK?

↓

INCREMENT THE CURRENT PLAYER INDEX (PLIX) BY 1 UNTIL AN ACTIVE PLAYER IS FOUND. THEN UPDATE THIS PLAYER

ACTION	3BCB _H	CD CE 3F	CALL TICKIT	
INXIX		3A AB 4F	LD A(PLIX)	A = CURRENT TO UPDATE 0,1,2 OR 3
	3BD1 _H	3C	INC A	INC PLIX
		E6 03	AND 03	MAX PLIX = 3
		32 AB 4F	LD (PLIX), A	INC PLIX AND SAVE IT
		CD A1 3E	CALL LDPLIX	POINT IX AT PLAYER RAM DATA PACKET
		DD CB 06 7E	BIT ACTBIT, (IX+PSTATE)	IX ↓ ROM ↓
		28 EE ^{Boy}	JR Z, INCIX	IF THIS PLAYER IS STILL IN GAME, CONTINUE. OTHERWISE, CHECK NEXT PLAYER.
				SWITCH AS IN "ROTATE"

CHECK MOVE, WRITE PATTERN AND PLAY NOTE

MOVEIT	3BE0 _H	DD 4E 00	LD C, (IX+LASTSW) C = CURRENT SWITCH
		DD 46 02	LD B, (IX+CURSW B =
		DD CB 06 76	BIT HUMB1T, (IX+PSTAT) ^{BIT 6}
	3BEA _H	20 03	JR NZ, NOCUR [←] JUMP IF NO HUMAN PRESENT

CHECK HUMAN BIT IN PLAYER RAM DATA PACKET

148

9

ZSW	3BEC _H	AF	XOR AF	
		47	LD B, A	CLEAR CURRENT SWITCH
		4F	LD C, A	↓ LAST SWITCH
NOCUR	80Y 78		LD A, B	A = CURRENT SWITCH = 0
	3BFO _H	B7	OR A	
		20 01	JR NZ, RANTST	IF SWITCH MADE, SKIP NEXT INSTRUCTION ?
		41	LD B, C	
RANTST		DD 70 00	LD (IX+LASTSW), B	SAVE SWITCH AS LAST
		78	LD A, B	
	80Y →	B7	ORA	
		20 05	JR NZ, GOTSW	
		0E 00	LD C, 00	
		CD 36 3D	CALL RANMOV	GET RANDOM MOVE
GOTSW	3C00 _H	DD 7E 01	LD A, (IX+LASTMV)	
		CD 63 3D	CALL MOVST	
		28 13	JR Z, GOTMOV	
				SWITCH DIRECTION ?
			ANY MOVE AND CURRENT SW MUST CHECK 3 DIRECTIONS ?	
		CD 61 3D	CALL MOVANY	
		28 0E	JR Z, GOTMOV	
32		41	LD B, C	TRY LAST SWITCH
			ANY MOVE	
		CD 61 3D	CALL MOVANY	
	3C11 _H	28 08	JR Z, GOTMOV	
		DD 46 01	LD B, (IX+LASTMV)	TRY LAST MOVE
			ANY MOVE	
		CD 61 3D	CALL MOVANY	
	3C19 _H	20 2E	JR NZ, CRASH	

A LEGITIMATE MOVE HAS BEEN FOUND UPDATE/WRITE NEXT PATTERN

GOTMOV	3C1B _H	DD 77 01	LD (IX+LASTMV), A	SAVE ACTUAL MOVE FOR LATER
		DD 77 03	LD (IX+AROT), A	THIS IS ALSO ARROW ROTATION INDICATOR
	3C21 _H	DD 56 05	LD D, (IX+ARRY)	D = ARROW Y COORD
		DD 56 04	LD E, (IX+ARRX)	E = X COORD
		CD 0A 3D ← DOG	CALL ERASE	BLANK THIS SCREEN SQUARE
		FD 66 05	LD H, (IX+PPATH)	HL = PLAYER PATTERN ADR (TRAIL PATTERN)
	DOG	FD 6E 04	LD L, (IX+PPAT L)	
	3C30 _H	01 01 04	LD BC, YSIZE/XSIZE	
		3E 10	LD A, WRITOK	MAGIC VALUE, OR IT
		FF	RST 38 _H	WRITE PLAYER PATTERN (OVER ARROW SQUARE)
		24	WRIT (SUB#36)	
		3A A5 4F	LD A, (TARRX) ← TEMP ARROW X	SAVE NEW ARROW X
		DD 77 04	LD (IX, ARRX), A ← ARROW X	SAVE NEW ARROW Y
		3A A6 4F	LD A, (TARRY)	
	3C40 _H	DD 77 05	LD (IX+ARRY), A	ARROW Y
		CD 18 3D ← DOG	CALL ANIARR	WRITE ARROW
		C3 29 3E	JP UPMUZK	PLAY NOTE

PLAYER HAS CRASHED

EXPLODE HIS ARROW,

ELIMINATE HIS TRAIL FROM THE GAME

5 EXP PATTERNS
↓
COLORS
↓
SOUNDS

CRASH	3C49 _H	01 FC 3D ← DOG	LD BC, EXPATS	EXPLOSION PATTERNS	BC = BASE ADR OF PATTERNS
		11 10 3E	LD DE, EXCOLS	EXPLOSION COLORS	DE = ADR OF EXP COLORS SEQUENCE
		3E 05	LD A, 05	A = LOOP CTR	
	3C51 _H	21 D8 3F	LD HL, EXPSND	EXPLOSION SOUNDS	HL = ADR OF EXP SOUNDS
EXLOOP		F5	PUSH AF	PUSH	LOOP CTR
↑ EXPLODE		C5	PUSH BC		EXP PATTERN ADR
		D5	PUSH DE		EXP COLORS TABLE ADR
		E5	PUSH HL		EXP SOUNDS ADDRESS
		1A	LDA, (DE)		A = EXP COLOR
		D3 05	OUT (COL1)		OUTPUT COL TO BACKGROUND NOW.
	3C5B _H	C5	PUSH BC		PUSH AGAIN EXP PAT ADR

WRITE EXPLOSION

ARROW Y COORD

3C5CH	DD 56 05	LD D, (IX+ARRY)	
	DD 5E 04	LDE, (IX+ARRY)	
3C62H	CD 0A 3D ← DOG	CALL ERASE	BLANK ARROW
	EI	POPHL	HL = EXPLOSION PAT ADR
	3E 10	LD A, 10H	MAG REG VALUE, OR WRITE
	01 01 04	LD BC, YSIZE/XSIZE	
	FF	RST 38H	WRITE EXPLOSION PATTERN
	24	WRIT (SUB#36)	
	FF	RST 38H	PAUSE A WHILE
	51	PAWS+1	
	07	PAUSE 7 INTERRUPTS	
3C70H	EI	POPHL	HL = EXP SOUND ADDRESS
	01 18 08	LD BC, 0818H	B=08 = LOOP CTR C=18H = (10-17H) OUTPUT 8 BYTE
	ED B3	OTIR	B-B-1, HL←HL+1
	DOG → DI	POP DE	
	CI	POP BC	
	FI	POPAF	
	3D ← DOG	DECA	DECREMENT CTR
	28 07	JR Z, EXPFIN	JMP TO EXPFIN IF EXPLOSION IS FINISHED
	13	INC DE	POINT DE AT NEXT EXP COLOR
	03	INC BC	POINT BC AT NEXT EXP PATTERN ADR
	03	INC BC	
	03	INC BC	
	03	INC BC	
EXPLOSION FINISHED ↓	3C80H	18 DI	LOOP BACK TO CONTINUE EXPLOSION NEXT SOUND SEQ POINTER UPDATED BY ABOVE OTIR
	EXPFIN	FD 56 05	LD D, (IY+PPATH) ← PLAYE PAT
		FD 5E 04	LDE, (IY+PPATH) ← DE = PLAYER TRAIL PAT ADR
		FD 21 00 00	LD IY, 0000
		FD 19	ADD IY, DE
	3C8FH	11 00 04	LD DE, 0400
	DOG →	FD 7E 00	LD A, (IY+00)
STOMP		21 B8 41	LD HL, STARTS

D = LOOP CTR E = BLANK BYTE OF
A = PLAYER TRAIL PAT
HL = START ADR OF PLAYING FIELD

151 N

3C98H

01 20 0D

LD BC, ALLBYT

BC = TOTAL # OF BYTES
IN PLAYING FIELDSCAN SCREEN RAM FOR MATCH OF PLAYER
TRAIL PATTERN AND BLANK ITSTATUS
STLOOP

DOGS

ED BT BOY

CPIR

A ← (HL), HL+1, BC ← BC-1 COMPARE PAT
WITH SCREEN ADR

20 05

JR NZ, RESTOM

03

INC BC

3CA0H

2B ← BOY

DEC HL

REST OF
MEMORY

73

LD (HL), E BLANK THAT
SCREEN ADR

18 F7

JR STLOOP

RESTOM

FD 23

INC IX

15

DEC D

HUMAN
PRESENT?

20 E9

JR NZ, STOMP

DD (B 06 76)

BIT HUMBIT, (IX+PSTAT) HUMAN BIT 6 PLAYER STATUS
CHECK PLAYER
HUMAN BIT.
IF ZERO,
SKIP NEXT 2 INSTR

28 04

JR Z, KILLST

KILL
STATUS

21 A4 4F

LD HL, (NO HUM POINT HL AT
CURRENT # OF HUMANS

3CB2H

35

DEC (HL)

DEC CUR # OF HUMANS

KILL ST

DD (B 06 BE BOY

RES ACTBIT, (IX+PSTAT) KILL ANIMATE ARROW
BITLOOP AND
INCREMENT
PLAYER SCORE(S)

INCREMENT ALL ACTIVE PLAYERS SCORES

0E 04

LD C, 04H

0D

DEC C C = PLAYER # 3, 2, 1 OR 0

BUMPEN

79

LD A, C

CD A13E BOY

CALL LDPLIX

2 DOGS

DD (B 06 7E

BIT ACTBIT, (IX+PSTAT)

3CC2H

28 18

JR Z, BUMPCK

06 00

LD B, 0

C5

PUSH BC

79

LD A, C

CURRENT SCR

21 96 4F

LD HL, CURSER

HL POINTS AT BASE ADR
FOR 3 BYTES PLT SCORES

09

ADD HL, BC

POINT HL AT

09

ADD HL, BC

PLAYER SCORE

09

ADD HL, BC

37

SCF

3CCFH

CD 4C 3E

CALL DISPSC

3CD2H

C1

POP BC

3CD3H	FF	RST 38H	] INCREMENT SCORE
	54	INC SCR (SUB#84)	
	79	LD A, C	
	C5	PUSH BC	
	B7	OR A	
	CD 4C 3E	CALL DISP SC	
	C1	POP BC	
	FF	RST 38H	] WAIT A WHILE (BEFORE NEXT PLAYER SCORE INCREMENT)
	51	PAWS+1 (SUB#80)	
	1E	WAIT 30 INTERRUPTS	
Boy	79	LD A, C	
3CE0H	B7	OR A	
	20 D6	JR NZ, BUMPEN	CURRENT NUMBER OF PLAYERS
	21 A2 4F	LD HL, CNOPL	
	35	DEC (HL)	] IF ONLY 2 PLAYERS LEFT AND 1 JUST CRASHED, DEC CNOPL TO GET ZERO
	35	DEC (HL)	
	28 02	JR Z, ENDCHK	ONLY 1 PLAYER LEFT ROUND HAS ENDED
	34	INC (HL)	
	C9	RET	

CHECK IF END OF ROUNDS

ENDCHK	3CECH	3A DC 4F	LD A, (CT7)	A = # OF ROUNDS (LEFT)
		3D	DEC A	
	3CF0H	27	DAA	
		32 DC 4F	LD (CT7), A	
		C2 DA 3E	JP NZ, FIRE IT	
ENDMO	3CF7	FF	RST 38H	
		35	STRDIS+1 (SUB#52)	
		30	X COORD	
		22	Y ↓	
	3CFBH	44	DISPLAY OPTIONS	2X IN COLOR 1 ON COLOR 0

0100 0100

3CEH 00 3D ← DOG STRING
F3 ADR
3CFH 76 DI
HALT

44 45 4D 4F 06 4F 56 45 52 00
3D00H D E M O 6 SPACES O V E R

ERASE PATTERN (BLANK 1x4 BYTE "SQUARE" IN SCREEN RAM) MAGIC RAM NOT USED

ERASE	3D0AH	D5	PUSH DE	SAVE Y/X COORDINATES
		FF	RST 38H	] CONVERT COORDINATES TO RESPECTIVE SCREEN ADR IN DE
		3B ← BOY	RELAB 1+1 (SUB#58)	
		00	MAGIC VALUE	
		EB ← BOY	EX DE, HL	
		06 00	LD B, 0	B = BLANK BYTE (WRITTEN TO SCREEN RAM)
	3D11H	11 01 04	LD DE, YSIZE/XSIZE	
		FF	RST 38H	
		2A	BLANK (SUB#42)	
		D1	POP DE	
		C9	RET	

ANIMATE (WRITE) THE ARROW

A = BIT 3 2 1 0
R L DWN UP

IX IS NOT ClobberED

ALL OTHER REGISTERS ARE ClobberED
IF "ACTIVE" BIT IS SET,

ANIARR	3D18H	DD CB 06 7E	BIT ACTBIT, (IX+PSTAT)	BIT ← PLAYER STILL IN GAME	CHECK "ACTIVE" BIT IN RAM PLAYER DATA PACKET (IX+06).
		C8	RET Z		
		DD 7E 03	LD A, (IX+AROT)	A = ARROW ROTATE MASK	
	3D20H	CD E6 3D	CALL GETROT	HL = ARROW PATTERN ADR	
		DD 56 05	LD D, (IX+ARRY)	D = Y COORD	
		DD 5E 04	LD E, (IX+ARRX)	E = X ↓	
		E5	PUSH HL	SAVE PATTERN ADR	
	3D2AH	CD 0A 3D ← DOG	CALL ERASE		

3D2D _H	E1	POP HL	
	01 01 04	LDBC, YSIZE/XSIZE	
3D34 _H	3E 10	LDA, WRIT _R	MAGIC REG = OR WRITE
	FF	RST 38 _H	] WRITE ARROW
	24	WRIT (SUB#36)	
	C9	RET	

WHICH DIRECTION?
 GENERATE A RANDOM MOVE FOR THE PLAYER PACKET POINTER
 ANY SENTRY (SEE DO IT TABLE)

INPUT AND OUTPUT: B = CURRENT SWITCH (S...)
 C = LAST SWITCH

RANMOV	3D36 _H	FF	RST 38 _H	] EXIT A < 32 _D A = RANDOM 8 BIT NUMBER
		77	RANGED+1 (SUB# 118)	
		20	RANGE < 32 _D	

BUY → B7

OR A TIME TO CHANGE DIRECTION?

2 DOGS → DD 46 01
 78

JR Z, NEWMOV
 LD B, (IX+LASTMV) USE LAST MOVE

3D40_H CD 63 3D ← DOG
 C8

LD A, B STATUS?
 CALL MOVST

LAST MOVE IS GOOD ENOUGH

NEWMOV

FF
 77
 04
 47
 04

RETZ
 RST 38_H
 RANGED+1
 RANGE < 4
 LD B, A
 INC B

SHFIT

3E 80
 07
 10 FD

LD A, 80_H
 RLCA
 DJNZ SHFIT

RANFIN

47
 3E 08 ← DOG
 3D54_H CD 63 3D
 3D54_H 20 02

LD B, A A MOVE? SEE 3D61_H
 LD A, 08_H
 CALL MOVST
 JR NZ, ANYMOV

3D56_H 47 LD B, A
C9 RET

ANY MOV

06 OF LD B, 0FH TRY ALL MOVES
3E 08 DOG LD A, 08H A MOVE? SEE 3D61_H
CD 63 3D CALL MOVTEST

47 LD B, A
3D60_H C9 RET

MOVANY 3D61_H 3E 08 LDA, A MOVE

CONFIRM

TEST THE NEW MOVE FOR VALIDITY

A BITS 3 2 1 0
R L DOWN UP

B =

C = UNCHANGED

A = FIRST MOVE TO TEST = VALUE OF GOOD MOVE AT EXIT

DE, HL AT ENTRY/EXIT UNCHANGED

D = # OF ROTATES (FOR RRCA)

Z FLAG = ZERO IF A GOOD MOVE FOUND (REG A CONTAINS FIRST GOOD MOV.)
= NZ? IF BAD (DON'T WANT) MOVE FOUND (IN REG B)

STATUS?

MOVTEST 3D63_H D5 PUSH DE
ROTATE MASK? 16 08 LD D, 08H INITIATE # OF ROTATES
ROTMSK 0F RRCA 7 → 0 → CF

5F LDE, A
A0 DOG AND B
CD 77 3D CHKMOV

7B LD A, E

28 06 JR Z, MOVEXIT EXIT WITH GOOD MOVE?

15 DEC D DEC # OF ROTATES

3D70_H 20 FH JR NZ, ROTMSK

37 SCF NO GOOD MOVES

CB 12 BOY RLD SET Z FLAG = NZ

MOVEXIT

DOG DI POP DE
3D76_H C9 RET

156

CHECK THE MOVE "INTO" FOR BEING UNOCCUPIED I

A = BITS 3 2 1 0
R L DOWN UP

ROTATE DIRECTION → A = UP, DOWN, RIGHT OR LEFT BIT (RETURN UNCHANGED)
ZFLAG = ZERO (Z) IF MOVE IN REG A IS TO AN "EMPTY" POSITION
NZ IF MOVE IN REG A IS "BAD" ← NOT WANTED (INVALID)
BC, DE, HL RETURNED UNTOUCHED
IX = POINTER TO CURRENT PLAYER DATA PACKET
LOCAL TO THIS ROUTINE: D = TEMP X COORD OF ARROW
E = TEMP Y

	CHKMOV 3D77H	C5 D5 E5 F5 DD 56 04 DD 5E 05 CB 57 ^{Boy} 28 0A 7A FE 00 28 2F DOG → D6 04 57 18 30 CB 5F ^{Boy} 28 0A 7A FE 98 30 21 C6 04 57 18 22 CB 47 ^{Boy} 28 0A 7B ^{Boy} FE 0B ^{Boy} 28 13 DOG → D6 04 3DA8H 5F	PUSH BC PUSH DE PUSH HL PUSH AF LD D, (IX + ARRX) D = ARROW X COORD LD E, (IX + ARRY) E = ↓ Y ↓ BIT CHLEFT, A ^{BIT 2} JR Z, TRIGHT LD A, D CP LOWX ^{X=0} JR Z, BADMOV SUB WIDTH LD D, A JR LOOKSQ BIT CHRIGH, A ^{BIT 3} JR Z, TUP LD A, D CP MAX JR NC, BADMOV ADD A, WIDTH LD D, A JR LOOKSQ BIT CHUP, A ^{BIT 0} JR Z, TDOWN LD A, E CP LOWY JR Z, BADMOV SUB HEIGHT LD E, A	JMP IF L BIT NOT SET IF X COORD IS ALREADY 0, DON'T WANT TO TURN LEFT DEC TEMPX BY 1 POSITION TURN LEFT IS POSSIBLE D = TEMPX JMP TO CHECK IF NEW SQUARE IS "OCCUPIED" JMP IF R BIT IS NOT SET IF X COORD IS ALREADY 152, DON'T WANT TO TURN RIGHT TURN RIGHT IS POSSIBLE ADJUST TEMPX JMP TO CHECK IF NEW SQUARE IS "OCCUPIED" GOT AN UP MOVE JMP TO BADMOV IF Y COORD IS ALREADY AT LOWEST Y DEC TEMPY BY 1 POSITION TURN UP IS POSSIBLE E = TEMPY
TEST MOVE LEFT	TLEFT 3D81H			
TEST MOVE RIGHT	TRIGHT 3D91H			
TEST MOVE UP	TUP 3DA1H			

TEST
MOVE T DOWN
DOWN

3DA9H

18 14

CB 4F

28 0A

7B ← Boy

3DB0H

FE 5B ← Boy

28 05

C6 04

5F

18 06

F1

37

CB 12 ← Boy

18 23

D5

3DC0H

D5

C1

51

58

FE

3B ← Boy

00

E1

EB ← Boy

7E

Boy → B7

20 EC

01 28 00

3DD0H

09

7E

Boy → B7

20 E4

3DD5H

7A

JR LOOKSQ

BIT CHDOWN, A

JR Z, BADMOV

LD A, E

CP Y MAX

JR Z, BADMOV

ADD A, HEIGHT

LD E, A

JR LOOKSQ

POP AF

SCF

RLD

JR MOVEND

PUSH DE

PUSH DE

POP BC

LD D, C

LD E, B

RST 38H

RELAB1+1 (SUB#58)

MAG REG

POP HL

EX DE, HL

LD A, (HL)

OR A

JR NZ, BADMOV

LD BC, BYTE PL

ADD HL, BC

LD A, (HL)

OR A

JR NZ, BADMOV

LD A, D

JMP TO CHECK IF
NEW SQUARE IS "OCCUPIED"
BIT 1

GOT A DOWN MOVE

IF Y COORD IS ALREADY 9/D
DON'T WANT TO TURN DOWN

DON'T
WANT
TO MAKE BADMOV
THIS
MOVE

LOOKSQ
LOOK TO SEE IF
NEW SQUARE
IS "OCCUPIED".

SWAP X, Y IN DE
FOR
RELAB1 SUBROUTINE

TEST SQUARE

JMP TO BADMOV
IF SQUARE IS "OCCUPIED"

← TEMP ARROW X COORD

3DD6 _H	32 A5 4F	LD (TARRX), A	SAVE TEMPX
	7B ^{← BOY}	LD A, E	
	32 A6 4F	LD (TARRY), A	SAVE TEMPT
	FI	POP AF	
	16 00	LD D, 0	
3DE0 _H	CB 3A	SRL D	SET Z FLAG = ZERO
	EI	POP HL	
	DI ^{DOG}	POP DE	
	C1	POP BC	
	C9	RET	

GET ROTATED ARROW PATTERN ADDRESS

A = ROTATE DIRECTION BITS ^{3 2 1 0} R L DWN UP

IF A HAS MORE THAN 1 BIT SET, THEN ONLY ONE IS USED

HL → ROTATED PATTERN ADDRESS

GETROT	3DE6 _H	21 19 3E	LD HL, AUP	
		CB 47	BIT CHUP, A	← BIT 0
		C0	RET NZ	
		21 21 3E	LD HL, ADOWN	
	3DEF _H	CB 4F	BIT CHDOWN, A	← BIT 1
	3DF1 _H	C0	RET NZ	
		21 1D 3E	LD HL, ARIGHT	
		CB 5F	BIT CHRIGH, A	← BIT 3
		C0	RET NZ	
		21 25 3E	LD HL, ALEFT	MUST BE BIT 2
		C9	RET	

EXPLOSION PATTERN 1

EXPAT1	3DFC _H	00
		14
		14
	3DFF _H	00

EXPLOSION PATTERN 2

EXPAT2 3E00H 00
45
51
00

EXPLOSION PATTERN 3

EXPAT3 3E04H 05
40
01
50

EXPLOSION PATTERN 4

EXPAT4 3E08H 11
40
01
44

EXPLOSION PATTERN 5

EXPAT5 3E0CH 00
00
00
3E0FH 00

SCREEN EXPLOSION COLORS SEQUENCE

EXCOLS	3E10H	FC	LIGHT BLUE	] BACKGROUND FLASH
		00	BLACK	
		FC	LIGHT BLUE	
		00	BLACK	

3E14H FA RESET BACK TO NORMAL BLUE

COUNT DOWN DISPLAY PACKET ?

CDCLR 3E15H 04 00

TIMER DISPLAY PACKET ?

TDPACK 3E17H 01 80

160

ARROW ANIMATION PATTERNS FOR EACH ROTATION

E

ARROW UP

AUP 3E19_H 14
55
41
00

ARROW RIGHT

ARIGHT 3E1D_H 14
05
05
3E20_H 14

ARROW DOWN

ADOWN 3E21_H 00
41
55
14

ARROW LEFT

ALEFT 3E25_H 14
50
50
14

PLAY NOTE

PLAYER
UPDATING

UPMUZK 3E29_H 3A A3 4F
CD A1 3E
20GS DD 7E 03

3E32_H 06 03
TSTBIT 0F

38 02
10 FB

GOTBIT 48
06 00

3E3C_H FD 09

LDA, (PLIX) A=WHO IS CURRENT PLAYER?
CALL LDPLIY IX POINTS TO RAM DATA PACKET
IX ↓ ROM ↓

LDA, (IX+AROT)

LDB, 3

RRCA

JRC, GOTBIT

DJNZ TSTBIT

LD C, B

LDB, 0

ADD IY, BC

146
BYTES

```

3E3EH  FD7E00  LDA, (IY+00H)
3E41H  D313      OUT (TONEC), A
        3E09      LDA, MUSVOL
        D315      OUT (VOLC), A
        3E11      LDA, 0AH
        D310      OUT (TONMO), A
        C9        RET

```

DISPLAY PLAYER SCORE (BCD SCORE IN 2 BYTES)

ENTER WITH: A = PLAYER# / LEAST SIGNIFICANT

HL → LAST BYTE OF SCORE / PLAYER SCORE DISPLAY OPTIONS

```

DISPSC 3E4CH  FD4E09  LDC, (IY+PSDOP)
        3004      JR NC, NOTXOR ← NOT XOR
        3E51H  CBA1      RES MROR, C ← MAGIC REG OR
        CBE9      SET MRXOR, C
        FD5E07  LDE, (IY+PSP0SX) ← PLAYER SCORE POSITION X
        FD5608  LDD, (IY+PSP0SX)  E=XCOORD
        3E0C      LDA, 0CH
        83        ADD A, E
        5F        LDE, A
        14        INC D
        3E60H  0643  DSG  LDB, 01000011 3BCD
        DD210D02  LDIX, FNTSML
        FF        RST 38H
        36        DISNUM (SUB#56)  DISPLAY SCORE
        C9        RET

```

PLAYER 0 PATTERN (TRAIL PATTERN)

```

3E69H  08
        A8
        2A
3E6CH  20

```

162

C

PLAYER 1 PATTERN (TRAIL PATTERN)

3E6D_H FF

C3

C3

3E70_H FF

PLAYER 2 PATTERN (TRAIL PATTERN)

3E71_H 0C

FC

3F

30

PLAYER 3 PATTERN (TRAIL PATTERN)

3E75_H AA

82

82

AA

PLAYER 0 ROM DATA

IY → 3E79^{DOG}_H FD EE E1 D4^{DOG}

69 3E

18

OR, COLOR 2

3E80_H 0C

01

18

OR, COLOR 2

4 PLAYER NOTES

PLAYER PATTERN ADR

CHAR DISPLAY OPTIONS

SCORE X COORD

Y COORD

DISPLAY OPTIONS

PLAYER 1 ROM DATA

3E83_H C8 BD B2 A86D^{DOG} 3E

1C

OR, COLOR 3

7D

01

3E8C_H 1C

OR, COLOR 3

PLAYER 2 ROM DATA

3E8D_H 9F96 8D^{DAG}853E91_H 713E

1C OR, COLOR 3

2D

01

1C OR, COLOR 3

PLAYER 3 ROM DATA

3E97_H 7E77 706A

753E

18 OR, COLOR 2

59

01

3EAD_H 18 OR, COLOR 1

POINT IX AT PLAYER RAM DATA PACKET AND

POINT IY AT PLAYER ROM ↓

ENTER WITH A = PLAYER NUMBER 0, 1, 2 OR 3

LDPLIX(IY) 3EA_H

D5

PUSH DE

E5

PUSH HL

FF

RST 38_H

5B

INDEX W+1 (SUB#90)

A = PLAYER#

POINT DE AT
PLAYER
ROM DATA PACKET

DAG

B43E

INDEX
TABLE

D5

PUSH DE

POINT IY AT PLAYER
ROM DATA PACKET

FDEI

POP IY

FF

RST 38_H

5B

INDEX W+1 (SUB#90)

A = PLAYER#

POINT DE AT
PLAYER
RAM DATA PACKET

BC3E

INDEX
TABLE

D5

PUSH DE

IX POINTS TO PLAYER
RAM DATA PACKET3EAF_H

DDEI

POP IX

3EB4H EI POP HL
DI POP DE
C9 RET

INDEX
PLAYER # ROM DATA PACKETS TABLE

ROMTBL 3EB4H 79 3E PLAYER 0
DOG 83 3E 1
8D 3E 2
3EBAH 97 3E 3

INDEX
PLAYER # RAM DATA PACKETS TABLE

RAMTBL 3EBC4H A8 4F PLAYER 0
AF 4F 1
3ECO4H B6 4F 2
BD 4F 3

CLEAR PLAYING FIELD

CLEARF 3EC4H FF RST 38H
1B FILL+1
BOY 88 41 START ADR
20 0D DOG # OF BYTES TO CLEAR
00 CLEAR WITH ZERO (000)
C9 RET

COLOR TABLE

3ECCH 85 YELLOW PIXEL 11 COL 3
D4 CYAN 10 2
FA BLUE 01 1
00 BLACK 00 0

CHECKMATE (REVISED DEMO)

ONE TIME ONLY HOUSEKEEPING
BEGIN PROGRAM HERE

3ED0 _H	31 96 4F	LD SP, 4F96 _H	
	FF	RST 38 _H	
	1B ← BOY	FILL+1 (SUB#26)	[CLEAR 12D RAMBYTES STARTING @ 4F96
	96 4F	FILL START ADR	
	0C 00	# OF BYTES TO FILL	
	00	FILL WITH ZERO	

REENTRY POINT FOR END OF GAME (ROUND)

FIREIT	3EDA _H	F3	DI	
		31 96 4F	LD SP, 4F96 _H	
		FF	RST 38 _H	
		00	INTPC	BEGIN MULTIPLE CALLS
	3EE0 _H	19	COLSET+1 (SUB#24)	[SET COLORS
		CC 3E	TABLE ADR	
		14	EMUSIC (SUB#20)	STOP SOUND
		1B	FILL+1 (SUB#26)	[CLEAR 34 BYTES OF RAM STARTING @ 4FA2
		A2 4F	FILL START ADR	
		22 00	# OF BYTES TO FILL	
		00	FILL WITH ZERO	
		17	SETOUT+1 (SUB#22)	[SET VERT BLNK REG HORIZ CLR BNDRY BKGRND COLOR
		BE	VERTICAL BLANK = 190D HORIZONTAL CLR BNDRY BACKGROUND	
	3EEC _H	6A	IMOD	01101010 BKGRND 42D
		08 ← DOG	RECTAN+1 (SUB#28)	[INT MODE FILL TOP OF SCREEN WITH BLUE AREA
		1D	X COORD	
		00	Y COORD	
	3EFO _H	00	X SIZE	
		A0		

166

Z

3EF2	0B		Y SIZE	
	55		FILL COLOR	01010101
	1D	← DOB	RECT+1 (SUB#28)	BLUE
	7A		X COORD	
	00		Y ↓	
	1E		X SIZE	
	09		Y SIZE	
	00		FILL COLOR	00 00 00 00
	1D	← DOB	RECT+1 (SUB#28)	
	57		X COORD	
	00		Y ↓	
	1D	← DOB	X SIZE	
	09		Y SIZE	
	00		FILL COLOR	00 00 00 00
3F00H	1D	← DOB	RECT+1 (SUB#28)	
	2B	← DOB	X COORD	
	00		Y ↓	
	1D	← DOB	X SIZE	
	09		Y SIZE	
	00		FILL COLOR	00 00 00 00
	1D	← DOB	RECT+1	
	04		X COORD	
	00		Y ↓	
	21		X SIZE	
	09		Y SIZE	
	00		FILL COLOR	00 00 00 00
	0E		ACTINT	ACTIVATE INTERRUPTS
	02		XINTC	END MULTIPLE CALLS

FILL IN

4 PLAYER

SCORE FIELDS

USING

4 RECTAN

(SUB#28)

SIMILAR TO ABOVE

INITIALIZE STARTING ^{X, Y} COORD OF ARROWS (PLAYER ARROWS)

		X	Y	
	212833			LDHL, PLAYO ARROW
3F11H	22AC4F			LD(4FAC _H), HL

X Y

3F14 _H	21 78 33		
	22 B3 4F ^{Boy}	LD(4FB3 _H), HL	
	21 50 1F		
	22 BA 4F	LD(4FBA _H), HL	
	21 50 47		
3F23 _H	22 C1 4F	LD(4FC1 _H), HL	
	CD C4 3E	DOGCALL CLEARF	CLEAR PLAYING FIELD

INITIALIZE
PLAYER (ARROW
PATTERN)

START AOR'S @
PLYO 4FAC
1 4FB3
2 4FBA
3 4FC1

20065 → DD 21 0D 02 LD IX, FNTSML

FF	RST 38 _H
37	DISNUM+1 (SUB#54)
4C	X COORDINATE
3F30 _H	Y ↓
24	COLOR OPTIONS
42	DISPLAY OPTIONS
DC 4F	BOD ADR
3E FF	LDA, FF
32 EC 4F	LD (TIMOUT), A

DISPLAY
OF GAMES (ROUNDS)
LEFT

KEEP RESETING TIMOUT
TO FFH EVERY ROUND.
NO PIZZA BREAK FOR THIS
DEMO

THIS DEMO USES "O PLAYERS INPUT" SELECTION

DEMO IS EXECUTED USING COMPUTER ONLY

NO HUMANS USING JOYSTICKS

3F3A _H	3E 04	LDA, 4	SET COMPUTER # OF PLAYS = 4
	32 A2 4F	LDA, (CNOPL), A	CNOPL = 4

INITIALIZE THE PLAYER PACKETS (VARIATION OF ORIGINAL CHECKMATE)

	4F	LDC, A	C = 4 = CNOPL
3F40 _H	AF	XOR AF	A = 0
	42	LD B, A	B = 0 = CNOHUM
	57	LD D, A	D = 0 = PLAYER#
GTPLIX	3F43 _H	7A	LDA, D
	3F44 _H	CD A13E	CALL LDPLIX

POINT
IX AT PLAYER RAM DATA PACKET
IY ↓ ROM ↓

3F47H C5

PUSH BC

SAVE

B = CURRENT # OF HUMANS (USING JOYSTICK) = 0

C = ↓ PLAYERS

D5

PUSH DE

D = THIS PLAYER #

DISPLAY PLAYER # IN PLAYING FIELD

7A LD A, D A = THIS PLAYER #

C631 ADD A, 31H CONVERT TO ASCII

DD5E04 LDE, (IX+04) E = X COORD

DD5605 LDD, (IX+05) D = Y ↓

3F52H 1D

DEC E

1D

DEC E

PLAYER CHAK DISP OPTIONS

FD4E06 LDC, (IY+PCDOP)

FF RST 38

32 CHDIS (SUB#50)

DISPLAY
PLAYER #
IN PLAYING FIELD

DISPLAY PLAYER # IN SCORE FIELD

↓ DOG

FD5E07 LDE, (IY+PSPSX) E = X COORD

FD5608 LDD, (IY+PSPSY) D = Y COORD

D5 PUSH DE

3F60H FF

RST 38H

32 CHDIS (SUB#50)

DISPLAY
PLAYER #
IN SCORE FIELD

WRITE PLAYER PATTERN IN SCORE FIELD

D1

POP DE

DE = Y, X COORD

7B

LDA, E

C606

ADD A, 6

5F

LDE, A

14

INC D

3F68H 14

INC D

ADJUST
COORDINATES

3F69_H 01 01 04

D06

FD 66 05

FD 6E 04

3F72_H 3E 10

FF

24

LD BC, YSIZE XSIZE ✓

PLAYER PATTERN ADR H

LD H, (IX+PPATH)

LD L, (IX+PPATL)

LD A, 0001 0000^{OR} A = MAGIC REGRST 38_H

WRIT (SUB#36)

D1

POP DE

D = THIS PLAYER #

D5

PUSH DE

SAVE IT AGAIN

DDE5

PUSH IX

SAVE PLAYER RAM DATA PACKET POINTER

7A

LD A, D

06 00

LD B, 0

4A

LDC, D

21 96 4F

LD HL, CURSCR

HL = CURRENT SCORE ADR

3F81_H 09

ADD HL, BC

09

ADD HL, BC

09

ADD HL, BC

CD 4C 3E

CALL DISPSC

DDE1

POP IX

IX = PLAYER RAM DATA PACKET POINTER

D1

POP DE

D = THIS PLAYER #

C1

POP BC

B = CURRENT # OF HUMANS (USING JOYSTICK) = C

C = ↓ PLAYERS

DELETED
7 INSTRUCTIONS
FROM ORIGINAL
CHECKMATE. NO
HUMANS PRESENT.
C NO HUMANS = 0

+
CKSUM 3

NO THUM

3E 80

LD A, ACTIVE

PLAYER PLAYER
BIT 7 = 1 ACTIVE 0 DEAD

DD 77 06

LD (IX+PSTAT), A

PLAYER STATUS
THIS PLAYER IS STILL
ON SCREEN

CK NO PL

3F90_H 14

0D

AF

3F93_H B1

INC D

D = NEXT PLAYER #

DEC C

C = CTR? = DEC CURRENT # OF PLAYERS

XOR A

OR C

170

6

-83 6432168421
01010011
10101100
10101101

COUNTDOWN
Loop

↓
CDOWNL

3F94H 20 AD ← DOG

JR NZ, GTPLIX

LOOP BACK TO CONTINUE
INITIALIZATION WITH
OTHER PLAYERS

3E 03

F5

FF

51

05

32 A34F

LD A, 3

PUSH AF

RST 38H

PAWS+1 SUB#80

WAIT 5 INTERRUPTS

LD (PLIX), A

SET COUNTDOWN TO 3

SAVE IT

TIME DELAY

INDICATE AT PLIX
WHO IS CURRENT PLAYER

TIME TO BEGIN COUNTDOWN

CD 29 3E

3FA2H FI

F5

C6 30

11 4C 2F

0E 44

FF

32

FF

51

28

3FB0H FF

14

FI

3D ← DOG

20 E2

CD C4 3E

3FB9H CD CE 3F

AF

3FBDH 32 DD 4F

CALL UPMUZ/K

POP AF

A = COUNT DOWN #

PUSH AF

SAVE IT

ADD A, 30H CONVERT IT TO ASCII

LD DE, YCOORD / XCOORD Y=2F X=4C

LD C, CDOPT ← COUNTDOWN OPTIONS

RST 38H

CHRDIS (SUB#5a)

DISPLAY
COUNTDOWN #

RST 38H

PLAY NOTE
DELAY

PAWS+1

WAIT 40 INTERRUPTS

RST 38H

END NOTE

EMUSIC

POP AF

A = COUNT DOWN #

DECA

DEC COUNTDOWN

JR NZ, CDOWNL

CALL CLEARF

CALL TICKET

XOR A

LD (CNT), A

RESET CTD BIT REPORT
(CHECKED BY SENTRY)

-30 168421
00011110
11100001
11100010

171 7

LOOPY	3FC0 _H	FF	RST 38 _H	
		43	SENTRY + 1	
		14 02	KEYMASK	← DON'T NEED. PRESS KEY TO TURN ON/OFF
		FF	RST 38 _H	PIZZA BREAK. NO BREAK FOR THIS DEMO
		45	DO IT + 1 (SUB #68)	
		CA 3F	DO IT TABLE ADR	
		18 F6	JR LOOPY	

DO IT TABLE

WHEN CTO @ 4FD5 REACHES ZERO
CTO FLAG (BIT0) IN 4FDD IS SET

4FDD 76543210

3FCA_H 41
CB 3B

(SET TO
DEVICE 01)

SUB
"ACTION"

JMP TO "SUB
"ACTION" WHEN CTO FLAG IS SET
@ 3BCB_H

3FCD_H C0

DO IT TABLE
TERMINATOR

TICK COUNT (CALLS @ 3BCB AND 3FB9_H)

VARIAION OF ORIGINAL CHECKMATE. NO HUMANS PRESENT FOR THIS DEMO.

TICKIT 3FCE_H 3E 02 LDA, 02_H

3FDO_H 32 D5 4F LD(CTO), A SET CTO FOR "DO IT" SUB "ACTION"
C9 RET

3FD4_H FF FF FF FF UNUSED BYTES

EXPLOSION SOUNDS

EXPSND 3FD8 EF FF 3F 00 FF FD F5 F5
3FE0 8F EE 3E 00 FF FD F5 F5
3FE8 4E 88 38 00 FF FD F5 F5
3FF0_H 48 44 34 00 FF FD F5 F5
3FF8_H 00 00 00 00 00 00 00 00

REVISED CHECKMATE DEMO

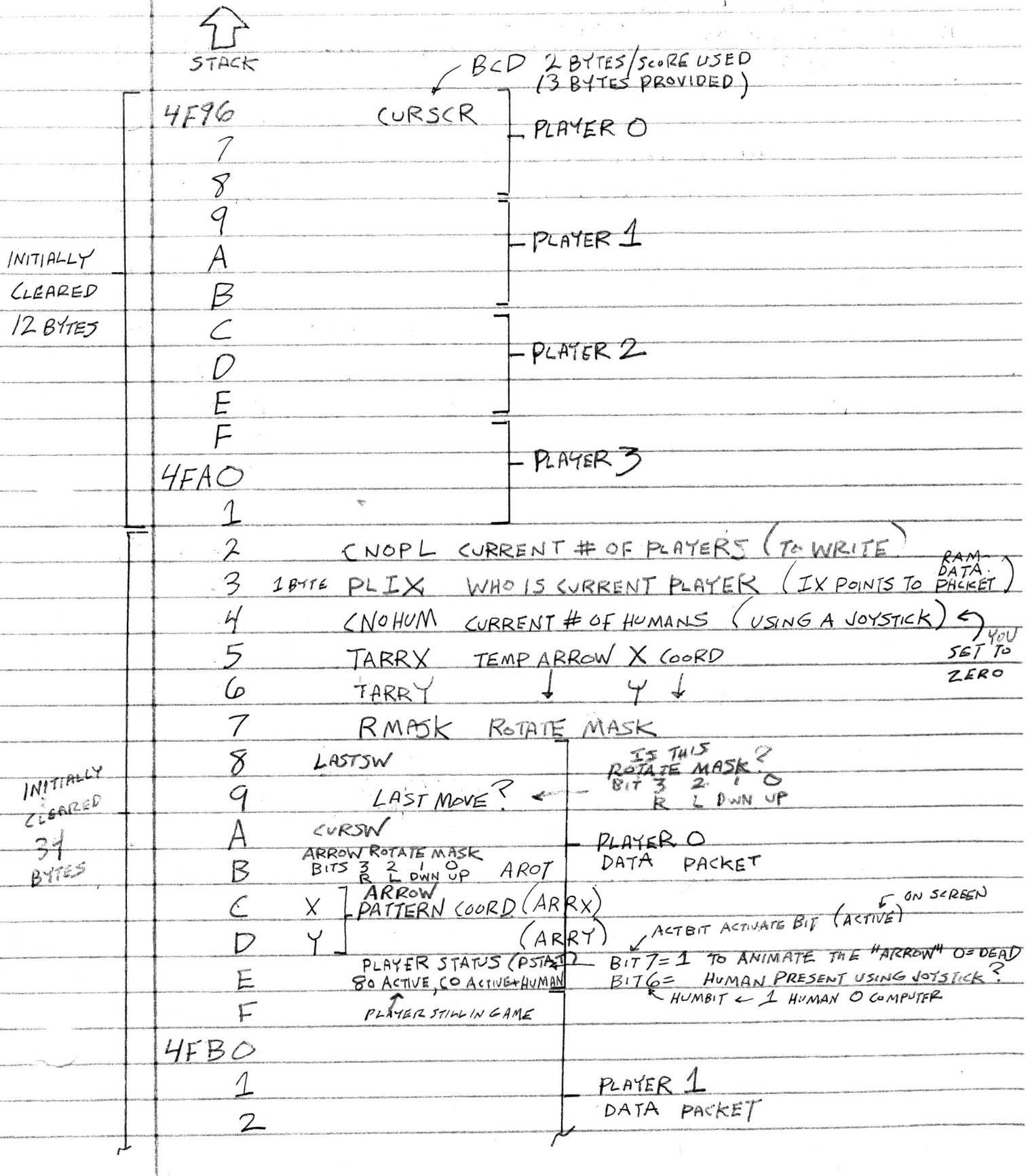
SUBROUTINES

TICKIT 3FCE
 ACTION 3BCB
 MOVEIT 3BEO
 GOTMOV 3CIB
 CRASH 3C49
 ENDCHK 3CEC
 ERASE 3DOA
 ANIARR 3D18
 RANMOV 3D36
 MOVANY 3D61
 MOVTST 3D63
 CHKMOV 3D77
 GETROT 3DE6
 UPMUZZK 3E29
 DISPCS 3E4C
 LDPLIX(IV) 3EA1
 CLEARF 3EC4

TABLES / DATA BLOCKS

EXPATS 3DFC ← EXCOLS 3E10
 AUP 3E19
 ARIGHT 3E1D
 A DOWN 3E21
 ALEFT 3E25
 PLYRO PATTERN 3E69
 ↓ 1 ↓ 3E6D
 ↓ 2 ↓ 3E71
 ↓ 3 ↓ 3E75
 PLYRO RAM DATA 3E79
 ↓ 1 ↓ 3E83
 ↓ 2 ↓ 3E8D
 ↓ 3 ↓ 3E97
 PLY ROM INDEX TABLE 3EB4
 PLY RAM ↓ 3EBC
 COLOR TABLE 3ECC
 DO IT TABLE 3FCA
 EXPJND 3FD8

DATA BASES



174

INITIALLY CLEARED 34 BYTES	4FB3	X	PATTERN COORD
	4	Y	
	5		
	6		
	7		
	8		
	9		PLAYER 2 DATA PACKET
	A	X	PATTERN COORD
	B	Y	
	C		
	D		
	E		
	F		PLAYER 3 DATA PACKET
	4FC0		
	1	X	PATTERN COORD
	2	Y	
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	A		
	B		
	C		
	D		
	E		
	F		
	4FDO		
	1		
	2		
	3		

ORIGINAL BALCHECK

4FD4

5 CTO TICK CTR FLAG

2 NO HUMANS
8 HUMANS

TICKET
SEE SUB @ 1649H

@ 3FCEH

↑ YOUR DEMO

6

7

8

9

A

B

C

OF GAMES (ROUNDS LEFT)

D

E

F

176

4FEO

4FF5

1

6

2

7

3

8

4

9

5

A

6

B

7

C

8

D

9

E

A

F

B

C

TIME OUT SET TO FF BY FIREIT PROGRAM LOOP
SO SENTRY WON'T TAKE "PIZZA BREAK" PAUSE

D

E

F

4FFO

1

2

3

OF PLAYERS

NOT USED

4

176

From: Michael Matte
Sent: Monday, April 8, 2019
To: Adam Trionfo
Subject: RE: Missing Page 176 of BalCheckHR Source Code

Page 176 is just a continuation of page 175 and reads:

176

4FF5

6
7
8
9
A
B
C
D
E
F

I guess I decided this page was a useless page and did not submit it to you. You can change the page numbers 177 and 178 to read 176 and 177. Or you can add this useless listing to page 177 and re-label that page as 176/177. Whatever is easy for you. I didn't bother to deal with this useless page 176 myself. Sorry about that. Another option would be to not include pages 176 and 177 and re-label page 178 as page 176. This would save some memory. You could also not even include pages 176-178.

YELLOW FIELD
RED GRAPHICS
SCORE

GRN GRAPHICS

WHT 2 DIGIT ROUND

RED GRAPHICS

GRN GRAPHICS

177

1+

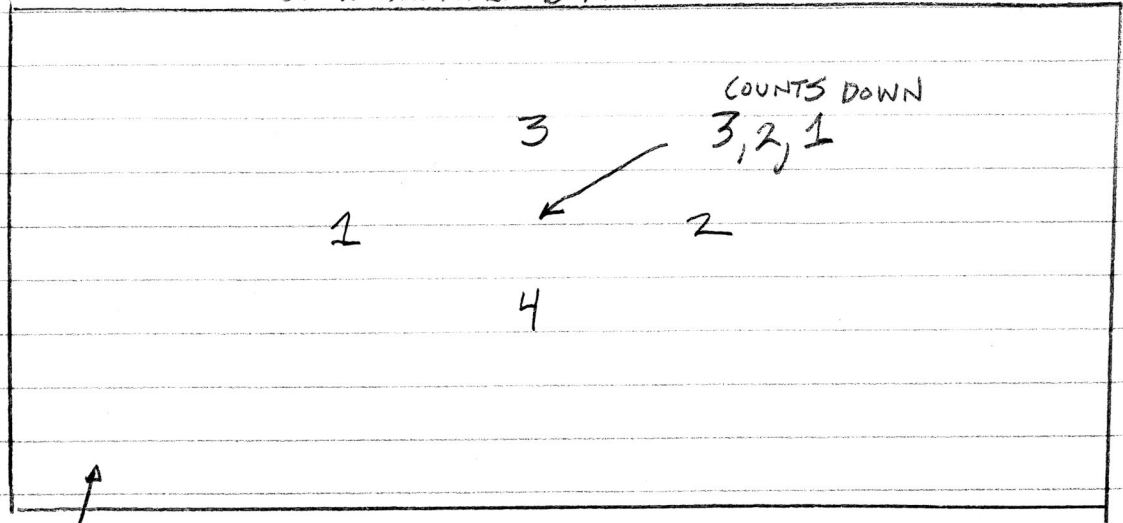
3+

4 □

1 2 □

BLUE MATCHES BACKGROUND COLOR

ORIGINAL
CHECKMATE



YELLOW FIELD

UNUSED BYTES

243E - 2440 _H	3
250F - 2511	3
27FF	1
2CC A - 2CC C	3
^{DOG} 2D35 - 2D3A	6
2DF0	1
2DFF	1
2E31 - 2E33	3
314B ← B0Y	1
326F	1
3271 - 3273	3
340D ← DOG	1
340F - 3411	3
39DC - 39E0	5
3BBD - 3BBF	3
3FD4 - 3FD7 _H	4