Addr10	ORG16	Addr16	Data16	Addr8	Data	Label	Instr	Operands	Comment

Altair Programs in 2K EPROM

consCPort = 10
consDPort = 11

~~~~~~

VG Monitor

1st 1K of EPROM occupied by VG-Monitor

EPROM: 0x800::0xBF4
Memory: 0xF800::0xFBF4

~~~~~~

~~~~~~

Blinky Rotate
Simple rotating bit
Display walking bit pattern on data lights.
Uses front panel data lights port 377

~~~~~~

	Page								
	FC								
Addr10	**ORG16**	**Addr16**	**Data16**	**Addr8**	**Data8**	**Label**	**Instr**	**Operands**	**Comment**
0	0	000	3E	000	076		MVI A 1		
1		001	01	001	001			pattern	
2		002	D3	002	323	next	OUT 377		
3		003	FF	003	377			fp oport	
4		004	01	004	001		LXI BC 4096		
5		005	10	005	020			delayCnt L	
6		006	00	006	000			delayCnt H	
7		007	09	007	011	delayLoop	DAD BC		
8		008	D2	010	322		JP NC delayLoop		
9		009	07	011	007			al	

10		00A	FC	012	374		ah	
11		00B	07	013	007	RLC		17 for RRC
12		00C	C3	014	303	JP next		
13		00D	02	015	002		al	
14		00E	FC	016	374		ah	

~~~~~~

Blinky AhWooAh

Row of lights expanding and shrinking in and out from both sides

~~~~~~

	Page FC								
Addr10	**ORG16**	**Addr16**	**Data16**	**Addr8**	**Data8**	**Label**	**Instr**	**Operands**	**Comment**
0	10	010	3E	020	76		MVI	A, 0x0F	
1		011	0F	021	17			initPattern	
2		012	5F	022	137	flipDirLoop	MOV	E,A	
3		013	47	023	107		MOV	B,A	
4		014	2F	024	57		CMA		
5		015	4F	025	117		MOV	C,A	
6		016	78	026	170	blinkyLoop	MOV	A,B	rotate B right
7		017	0F	027	17		RRC		
8		018	47	030	107		MOV	B,A	
9		019	79	031	171		MOV	A,C	rotate C left
10		01A	07	032	7		RLC		
11		01B	4F	033	117		MOV	C,A	
12		01C	A3	034	243	(compose)	ANA	E	compose display
13		01D	57	035	127		MOV	D,A	
14		01E	7B	036	173		MOV	A,E	
15		01F	2F	037	57		CMA		
16		020	A0	040	240		ANA	B	
17		021	B2	041	262		ORA	D	
18		022	D3	042	323	(display)	OUT	PORTLEDS	display on front panel
19		023	FF	043	377			portLEDs	
20		024	21	044	41	(delay)	LXI	HL, DELAYCNT	
21		025	06	045	6			delayCnt L	
22		026	00	046	0			delayCnt H	

23		027	F9	047	371		SPHL	
24		028	39	050	71	delayLoop	DAD	SP
25		029	D2	051	322		JNC	delayLoop
26		02A	28	052	050			al
27		02B	FC	053	374			ah
28		02C	C2	054	302	(checkFor0)	JNZ	blinkyLoop	if display was blank, flip direction
29		02D	16	055	026			al
30		02E	FC	056	374			ah
31		02F	7B	057	173	(flipDir)	MOV	A,E
32		030	2F	060	57		CMA	
33		031	C3	061	303		JP	flipDirLoop
34		032	12	062	022			al
35		033	FC	063	374			ah

=40 :: <70 occupied by faulty pgm attempt

~~~~~~

Blinky Address

FIll RAM memory with a delay routine, then randomly jump to those delay routines.
At the end of the delay, return to the main code, calculate a new random delay address, and jump there.
Each delay period will also be random, within limits set by front panel Sense Switches
Also throw random value into front panel Data LEDs.
Random is generated as 15-bit LFSR in DE, 15-bit just because the terms are easy.

~~~~~~

Page FC

Addr10	**ORG16**	**Addr16**	**Data16**	**Addr8**	**Data**	**Label**	**Instr**	**Operands**	**Comment**
0	70	70	31	160	61	BlinkyAddress	LXI	SP,STACK	init stack
1		71	00	161	0			al	
2		72	C0	162	300			ah	
3		73	21	163	41		LXI	HL,DSubsMem	copy delay subroutine to fill memory
4		74	00	164	0			al	
5		75	00	165	0			ah	
6		76	11	166	21	cpyDSubsLoop	LXI	DE,ProtoDSub	
7		77	C4	167	304			al	

8	78	FC	170	374			ah	
9	79	06	171	6		MVI	B,ProtoDSubEnd-ProtoDSub+1	
10	7A	04	172	004			4	
11	7B	1A	173	32	cpyBytesLoop	LDAX	DE	
12	7C	77	174	167		MOV	M,A	
13	7D	13	175	23		INX	DE	
14	7E	23	176	43		INX	HL	
15	7F	05	177	5		DCR	B	
16	80	C2	200	302		JNZ	cpyBytesLoop	loop for bytes of each DSub
17	81	7B	201	173			al	
18	82	FC	202	374			ah	
19	83	7C	203	174		MOV	A,H	
20	84	FE	204	376		CPI	DSubsMemEndH	loop till memory filled
21	85	10	205	020			d	
22	86	C2	206	302		JNZ	cpyDSubsLoop	
23	87	76	207	166			al	
24	88	FC	210	374			ah	
25	89	06	211	6	blinkyLoop	MVI	B,15	Main display loop. First gen random bits,
26	8A	0F	212	017			15	B=bit loop counter
27	8B	7A	213	172	randShiftLoop	MOV	A,D	DE contains random
28	8C	E6	214	346		ANI	0xC0	AND modifier bits to set parity flag
29	8D	60	215	140			96	
30	8E	37	216	67		STC		copy parity to carry
31	8F	EA	217	352		JPE	rand1	
32	90	93	220	223			al	
33	91	FC	221	374			ah	
34	92	3F	222	77		CMC		
35	93	7B	223	173	rand1	MOV	A,E	rotate into LSB of DE
36	94	17	224	27		RAL		
37	95	5F	225	137		MOV	E,A	
38	96	7A	226	172		MOV	A,D	
39	97	17	227	27		RAL		
40	98	57	230	127		MOV	D,A	
41	99	05	231	5		DCR	B	--loopCounter
42	9A	C2	232	302		JNZ	randShiftLoop	gen more bits till 0
43	9B	8B	233	213			al	
44	9C	FC	234	374			ah	
45	9D	7A	235	172	(randToDSubAddr)	MOV	A,D	Convert random in DE into a DSub address

46	9E	D3	236	323		OUT	DATALEDSPORT	set data LEDs to random too
47	9F	FF	237	377			0xFF	
48	A0	E6	240	346		ANI	DSubMemEndH-1	mask for mem size
49	A1	0F	241	17			d	
50	A2	67	242	147		MOV	H,A	
51	A3	7B	243	173		MOV	A,E	
52	A4	E6	244	346		ANI	~(ProtoDSubEnd-F	mask off LSBs to align to DSub size
53	A5	FC	245	374			252	
54	A6	6F	246	157		MOV	L,A	
55	A7	01	247	1	(prepDSubReturn)	LXI	BC,blinkyLoop	Push addr of main loop onto stack
56	A8	89	250	211			al	so DSub will return there
57	A9	FC	251	374			ah	
58	AA	C5	252	305		PUSH	BC	
59	AB	DB	253	333	(calcDCount)	IN	SENSEPORT	Calculate delayCount, switches are rangeMask
60	AC	FF	254	377			0xFF	
61	AD	A7	255	247		ANA	A	test A for 0
62	AE	C2	256	302		JNZ	maskOK	if not 0 then good
63	AF	B2	257	262			al	
64	B0	FC	260	374			ah	
65	B1	2F	261	57		CMA		make rangeMask FF if 0 (default)
66	B2	4F	262	117	maskOK	MOV	C,A	save rangeMask
67	B3	06	263	6		MVI	B,0	derive minDCount into B
68	B4	00	264	0			0	
69	B5	37	265	67		STC		
70	B6	78	266	170	calcDCountLoop	MOV	A,B	make all-1 up till lowest bit in rangeMask
71	B7	17	267	27		RAL		shift 1 up
72	B8	47	270	107		MOV	B,A	save minDCount
73	B9	A9	271	251		XRA	C	XOR & AND to find LS 1 in rangeMask
74	BA	A0	272	240		ANA	B	
75	BB	C2	273	302		JNZ	calcDCountLoop	if XOR took out bit, we found LSB, else go again
76	BC	B6	274	266			al	
77	BD	FC	275	374			ah	
78	BE	7A	276	172		MOV	A,D	get random bits
79	BF	B0	277	260		ORA	B	OR in the minDCount
80	C0	A1	300	241		ANA	C	mask off with rangeMask to set max
81	C1	47	301	107		MOV	B,A	BC is delayCount, C not 0 doesn't matter
82	C2	97	302	227		SUB	A	A=0 constant for end-of-delay test
83	C3	E9	303	351	(goDSub)	PCHL		goto DSub

84		C4	0B	304	13	ProtoDSub	DCX	BC	--delayCounter
85		C5	B8	305	270		CPA	B	if B=0
86		C6	C8	306	310		RZ		return to blinkyLoop
87		C7	E9	307	351	ProtoDSubEnd	PCHL		else go delay more
						STACK	dEQU	C000	top of 48K
						DSubsMem	dEQU	0	
			10		020	DSubsMemEndH	dEQU		8K=0x20, 4K=0x10, addr hi-byte of last DSub+1

~~~~~~

Load BASIC Loader

This is a preloader for the initial loader for Altair BASIC 4.0 / 4K

The initial loader is copied from here to base-address 0, from where it is to be executed

~~~~~~

	Page								
	FC								
Addr10	**ORG16**	**Addr16**	**Data16**	**Addr8**	**Data**	**Label**	**Instr**	**Operands**	**Comment**
0	D0	D0	11	320	21		LXI	DE,LdrEnd	DE = last src address
1		D1	F9	321	371			al	
2		D2	FC	322	374			ah	
3		D3	21	323	41		LXI	HL,LdrEnd-Ldr+1	HL = last dst address + 1
4		D4	1C	324	034			28	
5		D5	00	325	0				
6		D6	1A	326	32	cpyLoop	LDAX	DE	A = src byte
7		D7	1B	327	33		DCX	DE	--srcAddr
8		D8	2D	330	55		DCR	L	--dstAddr
9		D9	77	331	167		MOV	M,A	dst byte = A
10		DA	C2	332	302		JNZ	cpyLoop	get out after L goes 0
11		DB	D6	333	326			al	
12		DC	FC	334	374			ah	
13		DD	76	335	166		HLT		done, halt. PCHL=351: HL=0 now, jump there
14		DE	3E	336	3E	Ldr			
15		DF	03	337	3				
16		E0	D3	340	D3				
17		E1	10	341	10				
18		E2	3E	342	3E				

19	E3	15	343	15		
20	E4	D3	344	D3		
21	E5	10	345	10		
22	E6	21	346	21		
23	E7	C2	347	C2		
24	E8	0F	350	0F		4K=0F, 8K=1F, etc.
25	E9	31	351	31		
26	EA	1A	352	1A		
27	EB	00	353	0		
28	EC	DB	354	DB		
29	ED	10	355	10		
30	EE	0F	356	0F		
31	EF	D0	357	D0		
32	F0	DB	360	DB		
33	F1	11	361	11		
34	F2	BD	362	BD		
35	F3	C8	363	C8		
36	F4	2D	364	2D		
37	F5	77	365	77		
38	F6	C0	366	C0		
39	F7	E9	367	E9		
40	F8	0B	370	B		
41	F9	00	371	0	LdrEnd	

EPROM END