

PROGRAMS  
FOR THE  
COSMAC ELF  
GRAPHICS

PAUL C. MOENS

for noncommercial use only

Oct. 20, 2010

PCM

# PROGRAMS FOR THE COSMAC ELF GRAPHICS

Paul C. Moews

## List of Sections

1. Introduction	3
2. Registers and Expanded Memory	4
3. Graphic Interrupt Routines	4
4. Variable Code	5

## List of Programs

1. Display Routine (128 Byte)	5
2. Display Routine (64 Byte)	7
3. Display Routine (32 Byte)	10
4. Variable Resolution Display Routine (48 Bytes)	12
5. Horse Race	15
6. Pattern Generator	16
7. Stop Watch	19
8. Twelve Hour Clock	22

Copyright © 1978 by Paul C. Moews

All rights reserved

Published September 1978 by Paul C. Moews

Printed by Parousia Press, Storrs, Connecticut

## Introduction

The programs in this booklet were written for the basic COSMAC 1802 “Elf” with Pixie graphics as described in the July, 1977 issue of *Popular Electronics*. All programs can be easily modified to work in expanded memory systems by initializing a few registers with high order memory addresses (see Registers and Expanded Memory). The programs are documented to aid understanding, but knowledge of the 1802 instruction set is presumed, and assembly language mnemonics are not provided because of space limitations.

The first four programs introduce new graphic interrupt routines which use 128 bytes or less for display purposes. This is necessary because the basic “Elf” has only 256 bytes of memory and some has to be reserved for programming. While these routines either produce a display with fewer spots or a reduced area with the same resolution as that given in the Pixie examples, they avoid showing active code. A discussion of graphic interrupt routines is provided as an aid to understanding these new routines (see Graphic Interrupt Routines).

As the basic “Elf” has so little memory a programming “trick” is used in programs 5, 7 and 8. This trick allows one to have an instruction that is varied each time a set of code is obeyed by changing the instruction after each pass through the code. See the section entitled “Variable Code” for a discussion of the method used.

## Registers and Expanded Memory

The 1802 requires that register 0 be used for the direct memory access (DMA) address, and register 1 for the interrupt address. Register 2 is used as an X register in all the programs and in some programs additional X registers are employed. Register 3 is used as the main program counter in all programs and subroutines are called by the "SEP register technique" (see RCA's User Manual for the 1802 Cosmac Microprocessor, pp. 54-58). All programs make the high order DMA address the same as the high order interrupt address. Basic "Elf" owners may run the programs just as they are written. In order to work in an expanded memory system R(1), R(2), and R(3) together with any other memory addressing registers must have their high order bytes preset. For example if the first program was to be run in page 4 (block 4) of an expanded system it would be loaded just as written but to page 4 instead of page 0. The following "patch" would be inserted on page 0:

Addr.	Code	Notes
00 00	F8 04	load page number to D
	02 B1 B2 B3	initialize high order bytes
	05 B4 B5	of registers used by program 1
	07 C0 04 00	transfer control to page 4 to run program

This patch when executed would initialize the necessary high order bytes and enter the program on page 4. Each program includes a list of registers which must be set to work in expanded systems.

## Graphic Interrupt Routines

In the "Elf" the generation of the interrupt and direct memory access (DMA) signals are all done under the control of the 1861 chip and its operation must be understood when writing or modifying interrupt routines. When enabled by an I/O signal from the microprocessor the 1861 chip generates interrupt requests at intervals of 1/61 seconds (3.579545/2 MHz clock). Twenty-nine machine cycles after each interrupt the first direct memory access (DMA) request occurs. Therefore when the 1861 generates an interrupt request we must use exactly 29 machine cycles to do the housekeeping required to get ready for the direct memory access requests. Things that must be done include saving the current designations of the X and P registers and saving the con-

tents of the D register so that we can return from the interrupt routine. We can also use these instructions to set register 0 to the address where we wish the DMA to start. If we put a counter in these preliminary instructions it will be incremented 61 times a second and can serve as a convenient timer or clock.

After these 29 preliminary machine cycles we get the first "burst" of eight DMA's. Memory bytes with the addresses designated by the contents of register 0, R(0) incremented once, R(0) incremented twice, . . . to R(0) incremented seven times, are mapped to the top line of the TV screen; R(0) is left pointing + 8 memory locations from where it started (i.e. R(0) has been incremented 8 times). We are now given 6 machine cycles, which can be used as we wish, these are normally used to reset register 0. After 6 machine cycles another burst of 8 DMA's occur, followed by 6 machine cycles, and so on until a total of 128 DMA bursts have filled the TV screen. During the last four DMA bursts the EF1 flag is held low as an aid to returning from the interrupt routine.

In the one page interrupt used by the Pixie graphics the starting address of register 0 is held constant for 4 DMA bursts giving a display in which each bit represents a square and we have the display divided into a field of 64 x 32 squares.

If we were to hold the starting address of register 0 constant for 8 DMA bursts one set bit would form a rectangle twice as high as it is wide, and we would form an array of such rectangles 64 across by 16 down. An interrupt routine which does just this is given in program 1. If we chose to set 2 adjacent bits at a time we would then have an array of squares 32 across and 16 down.

In the same manner holding the starting address of register 0 constant for 16 DMA bursts would form rectangles 4 times as high as they are wide and if we set 4 bits at a time we would produce an array of squares 16 across and 8 down. Program 2 contains such an interrupt routine. Holding register 0 constant for 32 DMA bursts and setting one word at a time would give an array of squares 8 across and 4 down as in Program 3. Note that each doubling of the number of DMA bursts halves the number of store locations required for display; 256 locations for 4 DMA bursts, 128 for 8 DMA bursts, 64 for 16 DMA bursts, and 32 for 32 DMA bursts.

It isn't necessary to keep the number of DMA bursts constant at each starting address. There is nothing to prevent changing the resolutions across

the TV screen if we so desire and we can save display locations in this way. Program 4 introduces such an interrupt routine, the top part of the screen is blanked by doing 108 DMA bursts on the same 8 locations, while the last 20 DMA bursts are used to produce an array of 64 x 5 squares with the resolution of a single page interrupt routine.

### Variable Code

A simple way to set up code which can be varied is to save an instruction in a register and then modify the instruction by changing the contents of the register. The register contents must, of course, be transferred to memory before the instruction can be executed. The following example is taken from program 8 where it is used to load the low order part of seven registers with seven consecutive switch bytes.

Add.	Code	Notes
0B	F8 1A A4	load address of variable code to R(4)
0E	F8 AF A5	AF is the instruction to be executed in location 1A, 1st time it will be AF, 2nd time AE, then AD, etc. and finally A9
11	85 44	get the instruction and store it in location 1A
13	3F 13 37 15	wait for in on, off
17	6C 64 22	read and display switch byte
1A	XX	the variable instruction, first AF, etc.
1B	85 FB A9	check, are we done?
1E	32 23	yes-go on
20	25 30 11	no-change instruction and go back through code

A similar method is used to increment one of 4 registers in program 5 as follows:

Add.	Code	Notes
48	8E	R(E) contains the instruction to be executed, in program 5 it is one of 1A, 1B, 1C or 1D. The instruction is here loaded to the D register.
49	53	R(3) is the program counter and a 53 instruction writes 1A, 1C, or 1D to the next location (4A)

4A XX This location becomes 1A, 1B, 1C, or 1D depending upon the contents of R(E).0 and is executed.

Variable code has the drawback that it cannot be used in read only memory. However random access memory is available in the Elf and variable code can be used to get more work out of the limited memory available. Care must be taken to maintain control over the locations changed and the instructions allowed when using this method.

### Program 1—Display Routine (128 Byte)

This program introduces an interrupt routine which displays 128 memory locations on the screen. Each bit is displayed as a rectangle twice as tall as it is wide; in order to get a square, two adjacent bits must be set. Such an interrupt routine could be written by simply extending RCA's single page interrupt, however a method is used in which a counter is incremented between DMA bursts. This method is more versatile and the number of similar DMA bursts can be made equal to  $5 + 3N$  where N is any integer, in this case  $n = 1$  to give 8 identical sets of DMA bursts. Program 3 employs a similar routine with  $N = 9$  to produce 32 identical DMA bursts. Note that the routine must not go across a page boundary and that the locations displayed are actually 7F through FE. The memory locations are, however, addressed individually as 00 through 7F in hexadecimal.

The program has three modes depending on what is set in the switches on turning to run. If 01 is set, memory is examined, enter the address desired, (remember 00 to 7F) push the in button and the contents of that location will be displayed; successive pushes of the in button step through memory. If 02 is set memory locations are changed, enter the address desired, push the in button, then enter the new contents and push in again, successive pushes of the in button change successive memory locations. If CC is set the display area is cleared and the program automatically enters the input mode.

A picture which illustrates the use of this interrupt routine is listed at the end of the program. Each line of the listing contains 64 bits (8 words) and forms one of the lines of the TV display, there are 16 lines in all. If this picture is entered by means of the program it will be seen to form as memory is filled.

Registers 1, 2, 3, 4 and 5 must be set if this pro-

gram is to be run in an expanded memory system.

FF we've run over displayed locations

### Use of Registers

	High	Low
R(0)		DMA address
R(1)		interrupt address
R(2)		X register
R(3)		program counter
R(4)		memory pointer
R(5)		memory pointer
R(C)	counter for interrupt routine	
R(F)	work in interrupt	work in interrupt

### Program Listing

Addr.	Code	Notes
00	F8 4F A1	interrupt address
03	F8 7E A2	stack address
06	F8 09 A3	program counter
09	D3 E2 69	program counter 3, TV on
OC	F8 02 BC	counter for DMA cycles, see interrupt routine
OF	F8 7F A5	starting address of displayed locations
12	6C 64 22	read and display switch byte
15	FD CC	is it CC?
17	3A 24	go to 24 unless switch byte is CC
19	85 A4	otherwise clear store locations
1B	F8 00 54	and turn on Q line when
1E	14 84	through
20	3A 1B	
22	24 7B	
24	3F 24 37 26	wait for in on, off
28	FO	bring back original switch byte
29	F6 33 2D	skip next instruction if switch byte odd
2C	7B	turn Q on (memory to be changed)
2D	6C	read switch byte
2E	FA 7F	and it against 7F (prevents illegal addresses)
30	FC 7F	add starting address of displayed locations
32	A4	load to R(4).0—memory pointer
33	3F 33 37 35	wait for in on, off
37	84 FF FF	check R(4).0 if contents are

3A	3A 3E	go to 3E if address OK
3C	85 A4	else reset to 7F
3E	04	load memory contents
3F	31 47	go to 47 if Q on—we have to change memory
41	52 64 22	else display memory
44	14	increment R(4)
45	30 33	go to wait for in on, off
47	6C 64 22	here if memory to be changed read and display switch byte
4A	54 30 44	store it and go to 44
4D	72 70	return from interrupt
4F	C4	entry to interrupt
50	22 78 22 52	save X, P and D
54	9C AF	load counter for DMA bursts to R(F).0
56	F8 7F BF	load starting address of displayed locations to R(F).1
59	91 BO	load high order DMA address
5B	E2 9F AO	load low order DMA address
--	--	DMA 1 occurs here (1st burst only)
5E	E2 9F AO	restore DMA address
--	--	DMA 2, 5 here
61	2F AO 8F	decrement counter, restore DMA address, load counter
--	--	DMA 3, 6 here
64	32 6A 9F AO	to 6A if counted out, else restore DMA address
--	--	DMA 4
68	30 5F	and go back to 5F
6A	9F AO	here if counted out, restore DMA address
--	--	DMA 7
6C	AO 9C 34 4D	restore DMA address, and reload counter, leave routine if done
--	--	DMA 8 (except last one)
70	AF 80 BF	restore counter and set new DMA address
--	--	new DMA 1
73	30 5F	back to 5F for DMA 2, etc.
74-7B		unused
7C-7E		stack
7F-FE		displayed locations

Add.	Code	Picture						Use of Registers	
								High	Low
00	98 00 00	60 00	60 00	00 00			R(0)	DMA address	
08	88 00 00	20 00	10 01	80			R(1)	interrupt address	
10	F8 00 07	27 F8	18 01	80			R(2)	X register, stack	
18	00 00 1F	FE DF	98 06	01			R(3)	program counter	
20	F8 00 07	FF FF	FO 18	06			R(4)	memory pointer	
28	08 00 00	74 FF	F7 F9	98			R(5)	memory pointer	
30	F8 00 1F	80 3F	9F FF	EO			R(9)	counter for timer in interrupt routine	
38	00 00 F5	FF CO	7E 1E	60					
40	F8 00 1F	80 3F	9F FF	EO			R(C)	counter for interrupt routine	
48	A8 00 00	7F FF	F7 F9	98					
50	F8 00 07	FF FF	FO 18	06					
58	A8 00 1F	FE DF	98 06	01			R(F)	work in interrupt	
60	00 00 07	27 F8	18 01	80					
68	88 00 00	20 00	10 01	80					
70	88 00 00	60 00	60 00	00					
78	F8 00 00	00 00	00 00	00					

### Program 2 – Display Routine (64 Byte)

This program contains an interrupt routine which displays 64 memory locations on the screen. Each location is displayed as two adjacent squares, the screen is divided into 128 squares (16 wide by 8 high). Two sets of 64 locations are displayed so there are two displays which appear alternately, each for about 0.5 seconds.

The interrupt routine employs a counter similar to that used in program 1 except that the number of identical DMA bursts is  $7 + 3N$ , where N is any integer. N is set to 3 to give 16 identical DMA cycles. Again the routine must not go across a page boundary and the locations displayed are actually 7F through FE. However in this program the individual squares are the things that are addressed and they are numbered 00 through FF as shown in the diagram. Set one is numbered 00 to 7F and set 2, 80 to FF, again in hexadecimal.

The program has only two modes of operation. If CC is set in the switch register when run is turned on the displayed locations are erased, otherwise they are unaltered. Once the program is running the numbers entered are square identifiers. Enter a square identifier and push in, that square will be changed to its opposite value, that is if it is off it will be turned on, if on it will be turned off. As a trial clear the display area and enter the 74 square identifiers listed after the program.

Registers 1, 2, 3, 4 and 5 must be set if this program is to be run in an expanded memory system.

Program Listing		
Add.	Code	Notes
00	F8 54 A1	interrupt address
03	F8 7E A2	stack address
05	F8 09 A3	program counter
09	D3 E2	establish program counter and X register
0B	F8 04 BC	Counter for DMA bursts, see interrupt routine
0E	F8 7F A5	starting address of displayed memory locations
11	6C 64 22	read and display switch byte
14	FD CC	is the switch byte CC?
16	3A 22	unless it is go to 22
18	85 A4	if it is CC clear memory
1A	F8 00 54	locations
1D	14 84	
1F	3A 1A 24	
22	69	turn on TV
23	3F 3E	wait for in on (part of waiting loop)
25	37 25	wait for in off if pushed
27	6C 64 22	read and display switch byte
2A	F6 FC 7F	shift (i.e. divide by 2) and add 7F the starting address of displayed memory
2D	A4	save result in R(4).0
2E	F0 F6	bring back switch byte, shift
30	3B 38	to 38 if switch byte was even, else go on
32	04 FB OF 54	here if switch byte odd, x'or memory against OF and replace

36	30 23	return to waiting loop	--		DMA 2, 5, 8, 11 and 14 here
38	04 FB FO 54	here if switch byte even, x'or memory against FO and replace	69	32 6F 9F AO	to 6F if counted out, else restore DMA address
3C	30 23	return to waiting loop	--		DMA 3, 6, 9 and 12 here
3E	89 FD 40	waiting loop, examine timer	6D	30 64	and go back to 64
41	33 23	continue wait unless time up	6F	9F AO	here if counted out, restore DMA address
43	F8 00 A9	here if time up, reset timer	--		DMA 15
46	85	get starting address of currently displayed locations	71	AO 9C 34 52	restore DMA address, and reload counter, leave routine if done
47	FB BF	is it set 2?	--		DMA 16 (except last one)
49	32 4D	go to set 1 if set 2 is now being displayed	75	AF 80 BF	restore counter and set new DMA address
4B	FF 80	else set to display set 2	--		new DMA 1
4D	FC 7F A5	store correct starting address	78	30 67	back to 67 for new DMA 2, etc.
50	30 23	back to waiting loop	7A-7B		unused
52	72 70	return from interrupt	7C-7E		stack
54	C4	entry to interrupt	7F-FE		displayed locations
55	22 78 22 52	save X, P and D			
59	9C AF	load counter for DMA bursts to R(F).0			
5B	85 BF	load starting address of displayed locations to R(F).1			
5D	91 BO	load high order DMA address			
5F	9F AO 19	load low order DMA address, increment timer			
--		DMA 1 occurs here (1st burst only)			
62	30 67 9F AO	skip to 67, reenters at 64 for 2nd and subsequent bursts			
--		DMA 4, 7, 10 and 13 here			
66	2F AO 8F	decrement counter, restore DMA address, load counter			

Picture  
(Note that bytes given are square identifiers.)

00	03	04	08	09	OD	OE	10
12	15	17	1A	1C	1F	20	22
25	27	2A	2F	30	33	34	37
3A	3D	3E	40	42	45	47	4A
4C	50	52	55	57	5A	5C	60
63	64	68	69	6C	6D	6E	6F
80	81	82	83	84	86	8B	8C
8D	8E	8F	90	96	9B	AO	A6
AB	B0	B1	B2	B3	B6	BB	BC
BD	BE	CO	C6	CB	DO	D6	DB
E0	E1	E2	E3	E4	E6	E7	E8
E9	EB						



## Diagram – Program 2

### Set No. 1

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

### Set No. 2

The points in the above diagram represent squares arranged in arrays 16 wide by 8 high. An easy way to create and encode pictures is to use graph paper.

### Program 3 – Display Routine (32 Byte)

The third interrupt routine displays 32 store locations on the screen. Each location is displayed as a single square and the screen is divided into 32 squares (8 wide by 4 high). Four sets of 32 locations are displayed so there are four displays which appear one after the other for about 0.5 seconds each.

This program uses an interrupt routine like that of program 1 except that 32 identical DMA bursts are produced. Again the routine must not go across a page boundary and the locations displayed are actually 7F through FE. The individual squares are the things that are addressed and they are numbered 00 through 7F as shown in the diagram.

The program has the same modes of operation as program 2. If CC is set in the switch register when run is turned on the displayed locations are erased, otherwise they are unaltered. Once the program is running the numbers entered are square identifiers. Enter a square identifier and push in, that square will be changed to its opposite value, that is if it is off it will be turned on, if on it will be turned off. As a trial the program should be entered so as to clear the display and the 78 square identifiers given at the end of the program entered.

Registers 1, 2, 3, 4 and 5 must be set if this program is to be run in an expanded memory system.

#### Use of Registers

	High	Low
R(0)		DMA address
R(1)		interrupt address
R(2)		X register, stack
R(3)		program counter
R(4)		memory pointer
R(5)		memory pointer
R(9)		counter for timer in interrupt routine
R(C)	counter for interrupt routine	
R(F)	work in interrupt	work in interrupt

#### Program Listing

Add.	Code	Notes
00	F8 4F A1	interrupt address
03	F8 7E A2	stack address
06	F8 09 A3	program counter
09	D3 E2	establish program counter and X register

OB	F8 OA BC	counter for DMA bursts, see interrupt routine
OE	F8 7F A5	starting address of displayed memory locations
11	6C 64 22	read and display switch byte
14	FD CC	is it CC
16	3A 22	no-go to 22
18	85 A4	if switch byte is CC clear display locations
1A	F8 00 54	
1D	14 84	
1F	3A 1A 24	
22	69	turn on TV
23	3F 36	wait for in on (part of waiting loop)
25	37 25	wait for in off if pushed
27	6C FA 7F	read switch byte, and against 7F to get legal square identifier
2A	52 64 22	display square to be changed
2D	FC 7F A4	add starting address of locations
30	04 FB FF 54	pick up location, x'or against FF, put it back
34	30 23	return to waiting loop
36	89 FD 20	waiting loop, examine timer
39	33 23	continue wait unless time up
3B	F8 00 A9	here if time up, reset timer
3E	85	get starting address of currently displayed locations
3F	FB DF	are we on the last set?
41	32 48	if we are go to 48 to start over
43	85 FC 20	if not add 20 to starting address of displayed bytes
46	30 4A	and return to in loop
48	F8 7F	here if starting over
4A	A5 30 23	restore R(5) and back to waiting loop
4D	72 70	return from interrupt
4F	C4	entry to interrupt
50	22 78 22 52	save X, P and D
54	9C AF	load counter for DMA bursts to R(F).0
56	19	increment timer
57	85 BF	load starting address of displayed locations
59	91 BO	load high order DMA address
5B	9F AO	load low order DMA address
--		DMA 1 here (1st burst only)

5D	E2 9F AO	restore DMA address	6F	AF 80 BF	restore counter and set new DMA address
--		DMA 2, 5, 8, 11, 14, 17, 20, 23, 26 and 29 here	--		new DMA 1 here
60	2F AO 8F	decrement counter, restore DMA address, load counter	72	30 5E	back to 5E for DMA 2, etc.
--		DMA 3, 6, 9, 12, 15, 18, 21, 24, 27 and 30 here	74-7B		unused
63	32 69 9F AO	to 69 if counted out, else restore DMA address	7C-7E		stack
--		DMA 4, 7, 10, 13, 16, 19, 22, 25 and 28 here	7F-FE		displayed locations
67	30 5E	and go back to 5E			Picture
69	9F AO	here if counted out, restore DMA address			(Note that bytes given are square identifiers.)
--		DMA 31 here	01 02 03 04 05 08 09 0B		
6B	AO 9C 34 4D	restore DMA address, and re-load counter, leave routine if done	OD OE 11 12 14 15 19 1A		
--		DMA 32 here	1B 1C 1D 21 22 23 24 25		
			28 29 2A 2B 2D 2E 31 32		
			34 35 39 3A 3B 3C 3D 41		
			42 43 44 45 48 49 4B 4D		
			4E 51 52 54 55 59 5A 5B		
			5C 5D 61 62 63 64 65 68		
			69 6B 6C 6D 6E 71 72 74		
			75 79 7A 7B 7C 7D		

Diagram – Program 3

Set No. 1

Set No. 2

00	01	02	03	04	05	06	07
08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F

20	21	22	23	24	25	26	27
28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37
38	39	3A	3B	3C	3D	3E	3F

40	41	42	43	44	45	46	47
48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57
58	59	5A	5B	5C	5D	5E	5F

60	61	62	63	64	65	66	67
68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77
78	79	7A	7B	7C	7D	7E	7F

Set No. 3

Set No. 4

The points in the above diagram represent squares arranged in arrays 8 wide by 4 high. An easy way to create and encode pictures is to use graph paper.

**Program 4 – Variable Resolution Display Routine  
(48 Bytes)**

Program 4 introduces a more complex interrupt routine. This interrupt routine displays 48 memory locations with a variable number of DMA bursts. The first 8 locations are used in producing 108 identical DMA bursts which fill most of the television screen. The last 40 store locations are used in 5 groups of 8, each group of which is used to produce 4 DMA bursts. In these last 40 locations, therefore, each bit produces a square of the same size that it would produce if a one page interrupt routine were used. That is the last 40 locations produces a display 64 squares wide and 5 squares high. The first 8 locations produce 64 bars 27 times as high as they are wide. If the first 8 locations are set to 00 the top 27/32 of the screen will be blank while the bottom 5/32 becomes the display area.

The program that uses this interrupt routine produces a “tickertape” display. The material to be displayed is coded in 11 sets of 5 bytes as shown in the diagram. After loading the program switch to run and enter bytes 5 at a time through the switches. When 5 bytes have been entered they are shifted into the tickertape display area and when all 11 sets have been entered the display area moves continuously.

Registers 1, 2, 3, 4, 5, and 6 must be set if this program is to be run in an expanded memory system.

Use of Registers	
High	Low
R(0)	DMA address
R(1)	interrupt address
R(2)	X register, stack
R(3)	program counter
R(4)	points to locations where data to be moved to display area is stored
R(5)	memory pointer to display area
R(6)	holds address of display sub-routine
R(7)	counter, 8 bits/word
R(8)	counter, 5 words vertically
R(9)	counter for timer in interrupt routine
R(A)	counter, 8 words/line
R(B)	another 5 counter

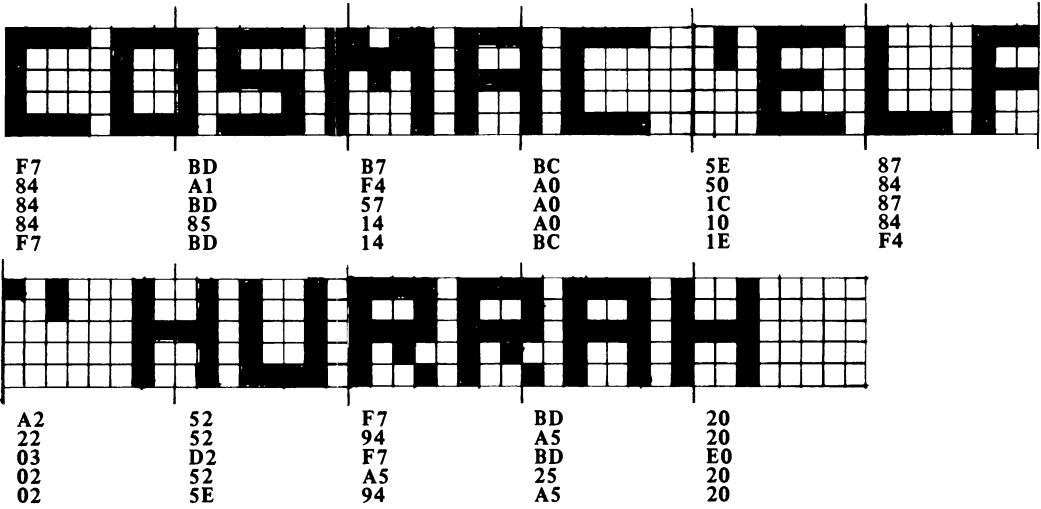
R(F) work in interrupt (stores counter) work in interrupt

**Program Listing**

Addr.	Code	Notes
00	F8 A5 A1	interrupt address
03	F8 3A A2	X register, stack
06	F8 3C A6	address of subroutine that shifts 5 bytes into display area
09	F8 A3 A4	A3 is one more than address to write first data point to
0C	F8 0F A3	program counter
0F	D3 E2	establish program counter and X register
11	69	turn on TV
12	7B	Q light on
13	F8 05 AB	set counter to read 5 bytes
16	24	decrement R(4)—points to place to store byte
17	3F 17 37 19	wait for in on, off
1B	6C 64 22	read and display byte
1E	54	store it in M(R(4))
1F	2B 8B	decrement 5 counter and load it
21	3A 16	if not counted out back to 16 for another
23	7A D6	else turn Q off and call routine to shift these 5 bytes to display
25	84 FB 6C	load R(4).0 if it equals 6C all data has been entered
28	3A 12	else go back for 5 more bytes
2A	F8 9E A4	here if all data loaded, restore R(4) address for display
2D	D6	this section of code repeatedly
2E	84 FF 05	calls display subroutine to
31	A4	shift the message into the display area
32	FB 67 3A 2D	play area
36	30 2A	
38	xx xx xx	stack
3B	D3	return from subroutine, this subroutine shifts the bytes in M(R(4)), M(R(4)+1), M(R(4)+2) M(R(4)+3) and M(R(4)+4) into the display area - R(4) and the memory locations are restored on return
3C	F8 08 A7	entry to routine, establish counter 8 bits/word

3F	F8 FF A5	display location where entry starts at	B1	91 BO	load high order DMA address
42	F8 05 A8	5 words to shift in—another counter	B3	9F AO	load low order DMA address
45	F8 00 A9	routine to give a brief delay	---		DMA 1 here (only once)
48	89 FB 06		B5	E2 9F AO	restore DMA address
4B	3A 48		---		DMA 2, 5, 8, 11, 14, . . . 104, and 107
4D	F8 08 AA	counter—8 words/line of display	B8	2F AO 8F	decrement counter, restore DMA address, load counter
50	04 FE	load D, shift left to fill DF	---		DMA 3, 6, 9, 12, 15, . . . 105, and 108
52	04 7E 54	load, circular shift left and restore	BB	32 C1 9F AO	to C1 if counted out, else restore DMA address
55	05 7E 55	load, circular shift left and restore	---		DMA 4, 7, 10, 13, 16, . . . and 106
58	25 2A	repeat 8 times till bit shifted in to line	BF	30 B6	and go back to B6
5A	8A 3A 55		C1	80 E2	here if counted out, new DMA address and change to 4 bursts with same DMA address
5D	28 88 14	repeat till a bit is shifted into all five lines	---		new DMA 1 here
60	3A 4D		C3	E2 20 AO	restore DMA address
62	84 FF 05 A4	reset R(4)	---		DMA 2 here
66	27 87	check if all bits shifted in if so return	C6	E2 20 AO	restore DMA address
68	32 3B		---		DMA 3 here
6A	30 3F	else go and shift another set in, this is end of subroutine	C9	E2 20 AO	restore DMA address
6C-A2	---	portion of memory used to store data for display	---		DMA 4 here
A3	72 70	return from interrupt	CC	3C C1 30 A3	back for another line or exit if done
A5	C4	entry to interrupt	DO-FF		the displayed locations, DO-D7 are the locations displayed with 108 identical DMA bursts and should be cleared to blank upper part of display
A6	22 78 22 52	save X, P and D			
AA	F8 24 AF	counter for 108 DMA bursts			
AD	19	increment R(9), timer			
AE	F8 DO BF	load starting address for DMA			

Diagram — Program 4



The above message was written and encoded on a piece of graph paper 88 squares wide and 5 squares high. Your own messages can be done in a similar way.

## Program 5 – Horse Race

Four horses, represented by bars, race across the screen. The first to reach the finish line blinks to indicate the winner. The idea is to bet on the outcome.

This program uses the interrupt routine which displays 64 store locations divided into 128 squares (see program 2). The squares are addressed individually in hex (00 through 7F) as they are in set 1 of program 2. Horse 1 lights squares 10 through 1F, horse 2 lights 30 through 3F, horse 3 lights 50 - 5F, and horse 4 lights squares 70 - 7F. The current position of each horse is held in registers A through D and each horse's position is incremented randomly to produce a fair race. To run the program simply switch to run; the horses appear at the starting gate and then race across the screen.

Registers 1, 2, 3, 4, 5, 6, and 7 must be set if this program is to be run in an expanded memory system.

Use of Registers	
High	Low
R(0)	DMA address
R(1)	interrupt address
R(2)	stack
R(3)	program counter
R(4)	memory pointer and work
R(5)	address of delay subroutine
R(6)	address of display subroutine
R(7)	points to table of winning positions
R(8)	used to generate random number
R(9)	used to generate random number
R(A)	counter for timer in interrupt routine
R(B)	position of horse 1
R(C)	position of horse 2
R(D)	position of horse 3
R(E)	holds counter for interrupt routine
R(F)	position of horse 4
	used to hold variable instruction
	work in interrupt
	work in interrupt

## Program Listing

Add.	Code	Notes
00	F8 93 A1	interrupt address
03	F8 BE A2	stack address
06	F8 09 A3	assign program counter
09	D3	program counter now R(3)
0A	F8 6C A5	address of delay subroutine
0D	F8 77 A6	address of display subroutine
10	F8 04 BC	counter for interrupt routine
13	F8 BF A4	clear the display locations
16	F8 00 54	
19	14 84	
1B	3A 16 24	
1E	E2 69	R(2) is X, turn on TV
20	F8 10 AA D6	set horses at the starting gate
24	F8 30 AB D6	and call the display routine to
28	F8 50 AC D6	show them
2C	F8 70 AD D6	
30	D5	call delay routine
31	F8 62 A7	point R(7) is address of 1st entry of table of winning positions
34	F8 1A AE	load R(E).0 with instruction 1A
37	88 F6 A4	this code together with instructions 8C through 90 generates a random byte in D
3A	E4 98 A4	
3D	B8 F6 A4	
40	98 F4	
42	E7	make R(7) the X register
43	FA 01	is random no. odd or even?
45	32 59	if even go to 59 do not increment horses position, if odd go on
47	D5	call delay routine
48	8E 53 XX	location 4A becomes 1A, 1B, 1C, or 1D and increments a horses position
4B	8E FC 70	change instruction to 8A, 8B, 8C, or 8D
4E	53 XX	location 4F becomes 8A, 8B, 8C, or 8D and loads a horses position
50	F3	check to see if horse has won by x'or against table of winning positions
51	32 66	if it has won go to 66 and blink
53	8E FC 70	else load horses position to D
56	53 XX D6	and call display routine
59	8E FB 1D	is instruction 1D?

5C 32 31 yes—go to 31 to start over  
5E 1E 17 30 37 no—increment R(E) and R(7) and go back to generate another random number  
62 1F 3F 5F 7F table of winning positions  
66 07 D6 D5 comes here when horse has  
69 30 66 won and blinks by calling delay and display  
6B D3 return from delay subroutine,  
6C F8 00 A9 R(9) is incremented in the  
6F 89 FB 08 interrupt routine  
72 3A 6F  
74 30 6B  
76 D3 return from display routine, this display routine works like program 2, the square identifier is passed to the subroutine in D  
77 BA F6 enter subroutine, save D in R(A).1 and shift right  
79 FC BF A4 add starting address of displayed locations, save in R(4)  
7C 9A F6 bring back passed byte, shift  
7E 3B 86 to 86 if passed byte even  
80 04 FB OF 54 byte odd, load x'or against OF and restore  
84 30 76 return from subroutine  
86 04 FB FO 54 byte even, load, x'or against FO and restore  
8A 30 76 return from subroutine, end of subroutine  
8C 18 98 part of interrupt routine, here  
8E FF 01 B8 just before returning, add 1 to R(8).0 and subtract 1 from R(8).1 as part of random number generator  
91 72 70 return from interrupt  
93 C4 entry to interrupt—this interrupt routine is very similar to that given in program 2 but is located in a different position in memory  
94 22 78 22 52  
98 9C AF  
9A F8 BF BF  
9D 91 BO  
9F 9F AO 19  
A2 30 A7 9F AO  
A6 2F AO 8F  
A9 32 AF 9F AO  
AD 30 A4  
AF 9F AO  
B1 AO 9C 34 8C  
B5 AF 80 BF  
B8 30 A7 end of interrupt routine

BA-BB unused  
BC-BD stack  
BF-FE displayed locations

### Program 6 — Pattern Generator

Program 6, like program 5, uses the 64 byte interrupt routine to display 8 rows of 16 squares. The area is divided into 4 quarters and used to generate patterns (see diagram).

An additive shift byte is used together with the identification numbers of up to 15 points in the upper left quarter of the display area. The logical exclusive or operation is carried out with each point and the display field and also with 3 related points in the 3 other quarters. The relationship between the points is such that the dotted lines in the diagram are “mirrors”. After all of the entered points are exclusively ored the additive shift is applied to each point and the process repeats indefinitely.

To run the program put the additive shift in the switch byte and turn to run, the shift byte will be stored and displayed. Then enter the points to be displayed. Terminate entry with EE (entry is otherwise automatically terminated when 15 points have been entered) and pattern generation begins. It is suggested that a trial run be made with 11 as the shift byte and 00, 01, 12, 23, and EE as the data points.

Registers 1, 2, 3, 4, 5, 6, and 7 must be set if this program is to be run in an expanded memory system.

Note: The following changes will generate a pattern in which the points in the upper left quarter are exactly repeated in the other 3 quarters. Change memory location 65 to F4 and the table entries at locations 80 through 83 to 00, 08, 40 and 48.

Use of Registers	
High	Low
R(0)	DMA address
R(1)	interrupt address
R(2)	stack
R(3)	program counter
R(4)	memory pointer to display locations
R(5)	address of display subroutine
R(6)	points to locations of entered points
R(7)	points to a table of 4 constants
R(8)	counter



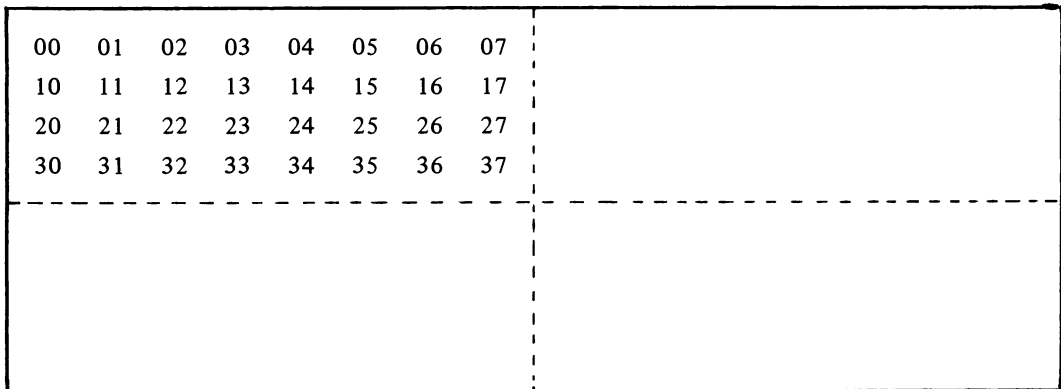
R(9)	counter for timer in interrupt routine			through 83 is display subroutine
R(B)	work for display subroutine	counter—4 bytes displayed at a time	3B F8 B0 A6	reload starting address of table of entered points
R(C)	holds counter for interrupt routine		3E 8E A8	load no. of points to R(8).0
R(D)	holds shift byte		40 06 FA 37	load point, and against 37 to get into upper left quarter of display
R(E)	counter, number of bytes entered		43 52 64 22	save it on stack, display it
R(F)	work in interrupt	work in interrupt	46 F8 00 A9	brief delay between calls to subroutine
			49 89 FB 20	
			4C 3A 49	
			4E D5	call display subroutine, display point is passed on stack

Program Listing

Addr.	Code	Notes		
00	F8 86 A1	interrupt address	4F 8D	get shift byte
03	F8 AF A2	stack	50 E6	make pointer to table of entered bytes R(X)
06	F8 09 A3 D3	program counter	51 F4 56	add shift byte to table entry and write new entry
0A	F8 04 BC	constant for interrupt routine	53 E2	restore R(X) as 2
0D	F8 5D A5	address of display routine	54 16 28	shift pointer to next location and decrement counter
10	F8 B0 A6	starting address of table of entered points	56 88	check, done with all points?
13	F8 BF A4	starting address of displayed locations	57 3A 40	no—go and do next point
16	F8 00 AE	clear display locations and leave R(E).0 equal to 00,	59 30 3B	yes—start over
19	54 14 84	R(E).0 will later contain number of points entered	--	start of display subroutine, this display routine works like program 2 except that that 4 squares are changed, the one passed on the stack and 3 related ones
1C	3A 16			
1E	24	R(X) is 2, turn on TV		
1F	E2 69	read and display shift byte		
21	6C 64 22	save shift byte in R(D).0	5B E2 D3	restore R(X) as 2 and return to main
24	AD	wait for in on, off		
25	3F 25 37 27	read and display entered point	5D F8 04 AB	entry point, load counter to display 4 points
29	6C 64 22	check, is it EE?		
2C	FB EE	yes—all points entered, go to 3B	60 F8 80 A7	load starting address of table of four constants
2E	32 3B	no—save in table of entered points	63 E7	make table pointer R(X)
30	F0 56	and increment table pointer and counter for number of points	64 02 F3 BB	load saved point and x'or against one of the table entries
32	16 1E	check counter, 15 points entered?	67 F6	shift result right
34	8E FB 0F	yes—table full, go to 3B	68 FC BF A4	add starting address of displayed locations and save in R(4)
37	32 3B	no—go back for more	6B 9B F6	get result back, shift
39	30 25	all points entered, code from 3B through 5A controls display while code from 5B	6D 3B 75	result even—go to 75
--			6F 04 FB 0F 54	result odd—x'or location against 0F and replace
			73 30 79	go to 79

75	04 FB F0 54	result even-x'or location against F0 and replace	8D	F8 BF BF	and program 5
79	2B 17 8B	decrement counter, increment table pointer, examine counter	90	91 B0	
7C	3A 64	not done-go to 64	92	9F A0 19	
7E	30 5B	done-return from subroutine	95	30 9A 9F A0	
80	00 0F 70 7F	table of constants, end of display subroutine	99	2F A0 8F	
84	72 70	return from interrupt	9C	32 A2 9F A0	
86	C4	entry to interrupt-this interrupt routine is very similar to the one used in program 2	A0	30 97	
87	22 78 22 52		A2	9F A0	
8B	9C AF		A4	A0 9C 34 84	
			A8	AF 80 BF	
			AB	30 9A	end of interrupt routine
			AD-AF		stack
			BO-BE		entered bytes
			BF-BE		displayed locations

Diagram -- Program 6



## Program 7 — Stop Watch

This stop watch program displays hours, minutes, and seconds and uses the interrupt routine of program 4. A character table is used to generate the numbers. The operation of the program is simple; R(9) is incremented 61 times a second in the interrupt routine, and this count is propagated through registers A through E to generate seconds, minutes, and hours. Whenever the contents of registers A through E are changed a display routine is called to indicate the new time. Both the timing routine and the display routine make use of variable code and one should read the section on variable code if one has difficulty following the program. Note that with the interrupt routine of program 4 the time is displayed in the last five lines of the display while the top 27/32 of the display is blanked.

On entry to the program the time is reset to zero (however the display is not cleared) and on pushing the in button the timer is started, a second push of the in button stops it, and a third push starts it again from where it left off. To reset the stop watch turn the run switch off and then on again. The maximum time that can be shown is 9 hours, 59 minutes, and 59 seconds.

Registers 1, 2, 3, 4, 5, and 6 must be set if this program is to be run in an expanded memory system.

### Use of Registers

	High	Low
R(0)		DMA address
R(1)		interrupt address
R(2)		stack
R(3)		program counter
R(4)		memory pointer
R(5)		memory pointer
R(6)		memory pointer, also used to store variable instruction
R(7)		holds variable instruction
R(8)		counter in display routine
R(9)		counts 61 times a second, incremented in interrupt routine
R(A)		seconds counter, units
R(B)		seconds counter, tens
R(C)		minute counter, units
R(D)		minute counter, tens
R(E)		hours counter, units
R(F)	work in interrupt	work in interrupt

## Program Listing

Add.	Code	Notes
00	F8 7E A1	interrupt address
03	F8 49 A2	stack
06	F8 09 A3	program counter
09	D3 E2	establish program counter, X register
0B	F8 00 AE AD	clear registers for hours,
0F	AC AB AA	minutes and seconds
12	69	turn on TV
13	37 13	wait for in off—comes here from 21
15	3F 15 37 17	wait for in on, off
19	F8 00 A9	set interrupt counter to 0
1C	30 4F	go to display routine—note display routine always returns to 1E
1E	89 FB 3D	here we look for 61 interrupt cycles
21	37 13	or did someone push the in button
23	3A 1E A9	to 1E unless 1 second has passed—else goes on first clear R(9).0
26	F8 3D A5	load address of variable code to R(5)
29	F8 4A A4	load starting address of table of 5 entries to R(4)
2C	E4	make R(4) the X register
2D	F8 1A A6	load instruction (1A) to R(6).0, first obeyed as 1A, then 1B, . . . and finally as 1E, increments registers
30	86 53 XX	carry out above instruction
33	FC 90 55	write AA to M(3D), first pass, second time write AB, . . . and finally write AE
36	FF 20 53	create and obey 8A first time, then 8B, . . . and finally 8E
39	XX	contents of one of the counting registers loaded here
3A	F3	x'or against table entry, maximum count allow in register
3B	3A 4F	if register contents are not equal to maximum go to display routine
3D	XX 86	3D becomes one of AA - AE, register which has reached maximum value is here zeroed

		out, load instruction in R(6)	72	30 68	go and write it
		to see where we are	74	D9 DB DC	table of 5 locations to write
3F	FB 1E	if instruction is 1E we're done	78	DE DF	characters to
41	32 4F	and should go to display	79	XX XX XX	unused
		routine	7C	72 70	return from interrupt
43	16 60	else change instruction, incre-	7E	C4	entry to interrupt—this inter-
		ment X to point to new table	7F	22 78 22 52	rupt routine is identical to
		entry	83	F8 24 AF	that used in program 4 except
45	30 30	and go back to 30 for another	86	19	that it is located in a different
		pass	87	F8 D0 BF	place in memory
47	XX XX XX	stack	8A	91 B0 9F A0	
4A	0A 06 0A	table of maximum values	8E	E2 9F A0	
4D	06 0A	allowed in the counting	91	2F A0 8F	
		registers	94	32 9A 9F A0	
--		the display routine starts here,	98	30 8F	
		it uses a character table to	9A	80 E2	
		show the contents of registers	9C	E2 20 A0	
		A through E in the display	9F	E2 20 A0	
		locations	A2	E2 20 A0	
4F	F8 74 A6	load starting address of table	A5	3C 9A	
		of memory locations to	A7	30 7C	end of interrupt routine
		write to	--		locations A9 through CF are
52	F8 8F A7	load variable instruction (8F)	A9	B3	the character table
		to R(7), it will be obeyed as	AA	C6	starting address of 0
		one of 8E, 8D, 8C, 8B, or 8A	AB	BB	starting address of 1
55	87 FB 8A	check variable instruction, if	AC	B9	starting address of 2
		it's 8A we're done and should	AD	CB	starting address of 3
		return	AE	B7	starting address of 4
58	32 1E	all done—return	AF	BD	starting address of 5
5A	27	more to do—decrement vari-	BO	C5	starting address of 6
		able instruction	B1	BF	starting address of 7
5B	46 A4	load display location to R(4)	B2	C1	starting address of 8
5D	87 53 XX	obey variable instruction,	B3	F0	starting address of 9
		loads contents of one of the	B4	90	character 0, (5 words/charac-
		counting registers to D	B5	90	ter
60	FC A9	add the starting address of	B6	90	
		the character table	B7	F0	character 5
62	A5 05 A5	and pick up the address of	B8	80	
		the character to be displayed	B9	F0	character 3
65	F8 05 A8	establish counter, 5 words per	BA	10	
		character	BB	F0	character 2
68	45 54	load and write word of	BC	10	
		character	BD	F0	character 6
6A	28 88	decrement no. words/charac-	BE	80	
		ter	BF	F0	character 8
6C	32 55	to 55 for next register con-	C0	90	
		tents if done	C1	F0	character 9
6E	84	else load R(4).0	C2	90	
6F	FC 08 A4	and set address for next word	C3	F0	
		to write	C4	10	
			C5	F0	character 7
			C6	10	character 1

C7 10  
 C8 10  
 C9 10  
 CA 10  
 CB A0  
 CC A0  
 CD F0  
 CE 20  
 CF 20

character 4

display locations are DO through FF, XX means doesn't matter, other locations must be set to give

legible display

Addr.	Code
D0	00 00 00 00 00 00 00 00
D8	00 XX 00 XX XX 00 XX XX
E0	00 XX 20 XX XX 20 XX XX
E8	00 XX 00 XX XX 00 XX XX
F0	00 XX 20 XX XX 20 XX XX
F8	00 XX 00 XX XX 00 XX XX

Note: It is probably easiest to clear memory before loading this program; then set locations E2, E5, F2 and F5 to 20.

### Program 8—Twelve Hour Clock

This clock program is similar to the stop watch program but displays a settable twelve hour clock instead of a stop watch. It was not possible to use the interrupt routine of program 4 and a one page interrupt is used instead. Register 9 is incremented within the interrupt routine to act as the timer. The count in register 9 is propagated through registers A to F to generate seconds, minutes and hours.

The clock is a 12 hour one and changes from 12:59:59 to 01:00:00 at 1:00 AM and 1:00 PM. To set the clock each digit is set individually. For example say we wish to set the time 12:03:48. First turn to run, then:

set tens digit of hours (01) to switch byte, push in,  
 set units digit of hours (02) to switch byte, push in,  
 set tens digit of minutes (00) to switch byte, push in,  
 set units digit of minutes (03) to switch byte, push in,  
 set tens digit of seconds (04) to switch byte, push in,  
 set units digit of seconds (08) to switch byte, push in,  
 and finally to start clock,  
 set 00 on switch byte, push in.

Registers 1, 2, 3, 4, 5, and 6 must be set if this program is to be run in an expanded memory system.

#### Use of Registers

Register use is the same as in program 7 except that register F is used as the tens of hours counter and is not employed in the interrupt routine.

#### Program Listing

Add.	Code	Notes
00	F8 8D A1	interrupt
03	F8 4D A2	stack
06	F8 09 A3	program counter
09	D3 E2	establish program counter, X register
0B	F8 1A A4	address of location whose contents will be variable code
0E	F8 AF A5	variable instruction to R(5).0 first time will be AF
11	85 54	write instruction to 1A, first time AF
13	3F 13 37 15	wait for in on, off
17	6C 64 22	read and display switch byte
1A	XX	variable instruction, saves switch byte in a register to set clock, first AF, then AE, etc. and finally A9

1B	85 FB A9	check is instruction A9
1E	32 23	yes—go to 23 and start clock
20	25 30 11	no—change variable instruction and go back to read another switch byte.
23	69	ready to start clock—turn on TV
24	89 FB 3D	get timer from interrupt— is it 61
27	3A 24	no—go and check again
29	A9	yes—reset interrupt timer and go on
--		This portion of code—2A through 53 is almost identical to that of program 7 (26 through 4E) it propagates the second count through the registers
2A	F8 41 A5	load address of variable code to R(5)
2D	F8 4E A4	load starting address of table of 6 entries to R(4)
30	E4	make R(4) the X register
31	F8 1A A6	load instruction (1A) to R(6).0 first obeyed as 1A, then 1B, and finally 1F, increments counting registers
34	86 53 XX	carry out above instruction
37	FC 90 55	write AA to location 41, first pass, then AB, . . . and finally write AF
3A	FF 20 53	create and obey 8A first time then 8B, . . . and finally 8F
3D	XX	8A through 8F obeyed here, loads one of counting registers to D
3E	F3	x'or against table entry, maximum count allowed in register
3F	3A 54	if register contents are not equal to maximum go to check hours and then to display routine
41	XX 86	41 becomes one of AA—AF, register which has reached maximum value is here zeroed out, load instruction in R(6) to see where we are
43	FB 1F	if instruction is 1F we're done
45	32 54	and go to check hours and then to display routine

47	16 60	else change instruction, increment X to point to new table entry	73	A5 05 A5	and pick up the address of the character to be displayed
49	30 34	and go back to 34 for another pass	76	F8 05 A8	establish counter, 5 words per character
4B	XX XX XX	stack	79	45 54	load and write word of character
4E	0A 06 0A	table of maximum values allowed in the counting registers	7B	28 88	decrement no. words/character
51	06 0A 0A	comes here to check hours, following code resets 13:00:00 to 01:00:00, else does nothing	7D	32 66	to 66 for next register contents if done
--			7F	84	else load R(4).0
54	8F F6	load hours counter, tens, shift right	80	FC 08 A4	and set address for next word to write
56	3B 60	to display if DF is zero, else go on	83	30 79	go and write it
58	8E FB 03	loads hours counter, units, is it 3?	85	D8 D9 DB	table of 6 locations to write characters to
5B	3A 60	if not go to display	88	DC DE DF	end of display routine—begin one page interrupt
5D	AF AE 1E	if R(E).0 is 3 reset hours to 01	--		return from interrupt
--		this display routine is almost identical to the one in program 7, it returns to location 24	8B	72 70	entry to interrupt
60	F8 85 A6	entry point, load starting address of table of memory locations to write to	8D	C4	save X, P and D
63	F8 90 A7	load variable instruction (90) to R(7), it be obeyed as one of 8F, 8E, 8D, 8C, 8B, or 8A	8E	22 78 22 52	increment R(9)
66	87 FB 8A	check variable instruction, if it's 8A we're done and should return	92	E2 E2 19	load high order DMA address
69	32 24	all done—return	95	91 B0	load low order DMA address
6B	27	more to do—decrement variable instruction	97	F8 00 A0	DMA address to D
6C	46 A4	load display location to R(4)	9A	80 E2	DMA 1, here
6E	87 53 XX	obey variable instruction, loads contents of one of the counting registers to D	--		restore DMA address
71	FC A9	add the starting address of the character table	9C	E2 20 A0	DMA 2, here
			--		restore DMA address
			9F	E2 20 A0	DMA 3, here
			--		restore DMA address
			A2	E2 20 A0	DMA 4, here
			--		not done, back to set new DMA address
			A5	3C 9A	done, return
			A7	30 8B	The character table and display locations are the same as for program 7.
			A9-FF		

