8080 Apple Monitor

Apple V1.0 ECT

Electronic Control Technology

763 Ramsey Ave.

Hillside, NJ 07205

(201) 686-8080

Electronic Control Technology

8080 Apple Monitor

The Apple Monitor is a program for the 8080 or Z80 microprocessors with executive commands and I/O handling routines. The author of the Apple Monitor is Roger Amidon of Applezap Corp. who also authored the Zapple Monitor <Z80 only version of the Apple Monitor> for TDL/Xitan. NOTE: The Apple Monitor has nothing to do with the Apple Computer and early versions of the Apple Monitor probably existed before the Apple Computer.

The Apple Monitor can be utilized as a software equivalent of a front panel. Memory, registers and I/O can be displayed and substituted from the system terminal. Debugging of both hardware and software is possible by use of the memory test or verifying blocks of memory or use of breakpoints.

The Apple Monitor is expandable. The user may add special routines for special I/O devices and/or additional commands. All software programs may utilize the I/O routines of the Apple Monitor through the vectors at the beginning of the Apple Monitor and thereby take advantage of the dynamic assigning of the I/O ports and routines. The Apple Monitor also includes many useful subroutines that may be used by user written programs.

USER WRITTEN COMMAND ROUTINES

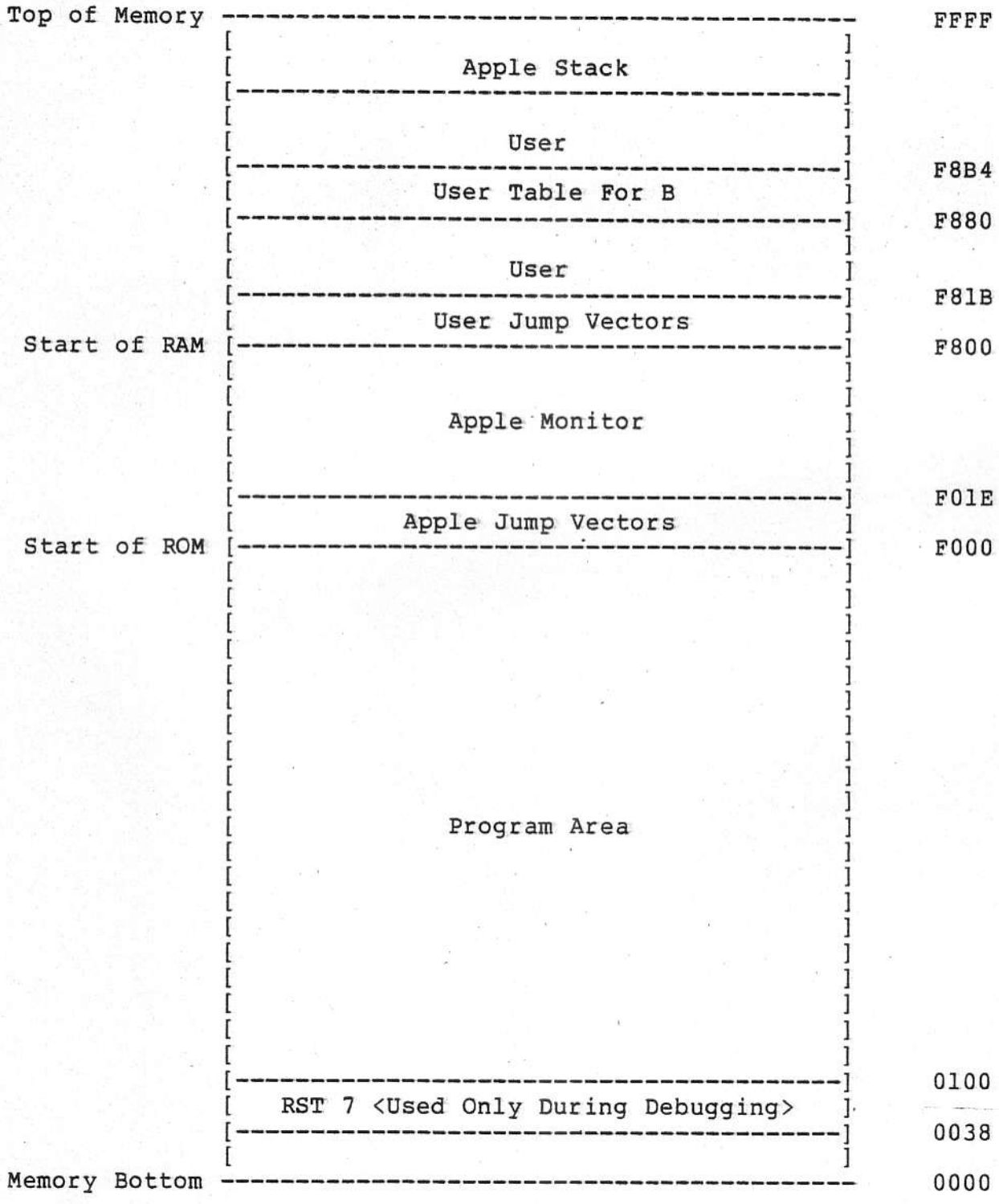
Three command letters are available for user written command routines - 'I', 'K' & 'O'. Apple vectors to the user jump vectors for these commands; 'I' to F812, 'K' to F815 & 'O' to F818. JMP's to the actual user written routines should be placed at these locations. A RET instruction at the end of the user written routine will return control back to the monitor displaying the prompt '>'.

The Branch 'B' command also allows additional user written command routines with the use of a letter A - Z. Control is passed to the routine at the address found in the user table at F880 to F8B3.

USER WRITTEN I/O ROUTINES

Occasionally I/O devices require special routines. The user I/O jump vectors should be located at F800 through F811. Be careful not to modify any register except those called for and do not upset the stack pointer. PUSH and match with POP's to restore registers. Use a RET at the end of the routine to return control back to the monitor.

Memory Map


Top of Memory
FFFF
Apple Stack
User
F8B4
User Table For B
F880
User
F81B
User Jump Vectors
Start of RAM
F800
Apple Monitor
F01E
Apple Jump Vectors
Start of ROM
F000
Program Area
0100
RST 7 <Used Only During Debugging>
0038
Memory Bottom
0000

Apple Jump Vectors		User Jump Vectors	
F000	Begin Apple	F800	Console Input
F003	Console Input	F803	Console Output
F006	Reader Input	F806	Console Input Status
F009	Console Output	F809	User Defined Storage <Input>
F00C	Punch Output	F80C	User Defined Storage <Output>
F00F	List Output	F80F	User Defined Printer <List>
F012	Console Status	F812	I
F015	I/O Assignment Check	F815	K
F018	I/O Set	F818	O
F01B	Memory Limit Check	F880	User Table For B
		F8B3	

Assign

C	Console	P	Punch
C	CRT	D	Data Transfer Device
P	Printer	P	Printer
B	Batch	A	Alternate <Parallel>
U	User	U	User
R	Reader	L	List
D	Data Transfer Device	C	CRT
P	Printer	P	Printer
A	Alternate <Parallel>	D	Data Transfer Device
U	User	U	User

Electronic Control Technology

Apple V1.0 ECT

A - Assign I/O
B - Branch to user routine A-Z
C - Undefined
D - Display memory on console in Hex
E - End of file tag for Hex dumps
F - Fill memory with a constant
G - GOTO an address with breakpoints
H - Hex math sum & difference
I * User defined
J - Non-destructive memory test
K * User defined
L - Load a binary format file
M - Move memory area to another address
N - Nulls leader/trailer
O * User defined
P - Put ASCII into memory
Q - Query I/O ports QI<N>=read I/O; QO<N,V>=send I/O
R - Read a Hex file with checksum
S - Substitute/examine memory in Hex
T - Types the contents of memory in ASCII equivalent
U - Unload memory in Binary format
V - Verify memory block against another memory block
W - Write a checksummed Hex file
X - Examine/modify CPU registers
Y - 'Yes there' find 'N' Bytes in memory
Z - 'Z END' address of last R/W memory location

A >A[d]=[u]

Assigns a device to be a particular unit.
First letter of specifier is all that's required.

device:= Console, Reader, Punch, List
unit:=

if Console:	**CRT, Printer, Batch Mode, User**
if Reader:	**Data transfer, Printer, Alternate (parallel), User**
if Punch:	**Data transfer, Printer, Alternate (parallel), User**
if List:	**CRT, Printer, Data transfer, User**

EXAMPLE: **AC=P Assign Console = Printer**
AP=P Assign Punch = Printer

B - >B.[a-z]

Branches into address table based on letter a-z.
If no command implemented, address will
contain OFFFFH, which aborts command.

EXAMPLE: B.A

C - unused

D - >D[addr1],[addr2]<,byte>

Dumps memory from addr1 thru addr2, where <byte>
is optional depending on line width desired.
NOTE- Defaults to 16 bytes per line.

EXAMPLE: D0,1F

```
0000  C3 07 F7 C3 24 F7 C3 32 F5 C3 84 F5 C3 53 F5 C3
0010  65 F6 DB 76 C9 C3 CD F1 C3 DC F0 C3 38 F0 C3 38
```

E - >E<addr>

End of file is generated to assigned punch device.
<addr> is optional.

EXAMPLE: E
E1234

F - >F[addr1],[addr2],[byte]

Fills from addr1 thru addr2 with byte.

EXAMPLE: F0,17FF,0

G - >G[addr1]<,addr2><,addr3>

Goes to addr1, and optionally set breakpoints at addr2 & addr3. If continuing from a breakpoint, the first parameter may be ommitted. This will cause execution of whatever addr. is contained in the "P" register.

EXAMPLE: G1600,163E

H - >H[val1],[val2]

Hex math of: val1+val2 & val1-val2 is displayed.

EXAMPLE: H2000,102A
302A 0FD6

I - unused

J - >J[addr1],[addr2]

Justifies memory from addr1 thru addr2. Any errors are displayed as:

addr 00100000

where the "1" indicates a bad bit, in this case, bit 5, and addr is the location in memory the error occurred.

EXAMPLE: J800,17FF
0F3D 00000010
0F88 00000010
16FD 10000000

K - unused

L - >L[addr]

Loads a binary file, starting at addr. The address following the last byte loaded will then be displayed on the console.

EXAMPLE: L800

12A0

M - >M[addr1],[addr2],[addr3]

Moves a block of memory starting at addr1, ending at addr2, to the block starting at addr3.

EXAMPLE: M0,7FF,1000