Machine Language Monitor

ABOUT THE COMMODORE PET

MACHINE LANGUAGE MONITOR...

Commodore Business Machines, Inc.
901 California Avenue Palo Alto,
California 94304, USA

Commodore/MOS
Valley Forge Corporate Center
950 Rittenhouse Road
Norristown, Pennsylvania 19401, USA

Commodore Business Machines **Limited**
3370 Pharmacy Avenue Agincourt,
Ontario, Canada M1W2K4

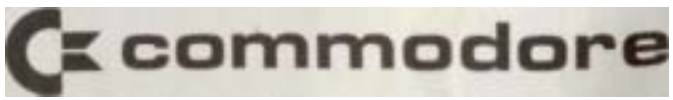
Commodore Business Machines **Limited**
Eaglescliffe Industrial Estate
Eaglescliffe, Stockton on Tees Teeside
TS 160 PN, England

Commodore Business Machines Limited
360 Euston Road
London NW1 3BL, England

Commodore Buromaschinen GmbH
Frankfurter Strasse 171-175 6078
Neu Isenburg West Germany

Commodore Japan Limited
Taisei-Denshi Building 8-14
Ikue 1-Chome Asahi-Ku,
Osaka 535, Japan

Commodore Electronics (Hong Kong) Ltd.
Watsons Estates
Block C, 1 1 th floor
Hong Kong, Hong Kong



Part No 323002

Machine language monitor

TIM is the Terminal Interface Monitor program for MOS Technology's 65XX microprocessors. It has been expanded and adapted to function on the Commodore PET. Later production models have the TIM in ROM but your early model PET must use a cassette tape version of this monitor. In either case execution is transferred from the PET BASIC interpreter to TIM by the SYS command.

To LOAD Your MONITOR, take the cassette with SQUIGGLE, MONITOR and BIGTIME, and put it on the tape unit with the MONITOR side up. Then type: LOAD "MONITOR" and when ready, RUN.

Commands typed on the PET keyboard can direct TIM to start executing a program, display or later registers and memory locations, set breakpoints, and load or save binary data. Oil modifying memory, TIM performs automatic read after write verification to insure that addressed memory exists, is R/W type, and is responding correctly.

TIM also provides several subroutines which may be called by user programs. These include reading and writing characters on the video display, typing a byte in hexadecimal and typing a CRLF sequence.

TIM Commands

M	display memory
R	display register
G	begin execution
X	exit to BASIC
L	load
S	save

EXAMPLES

M DISPLAY MEMORY

M C000,C010

```
C000  1 D C7 48 C6 35 CC E F C7
C008 C5 CA DF CA 70 CF 23 CB CO
10 9C C8 9C C7 74 C7 1 F C8
```

In a Display Memory command, the start and ending addresses must be completely specified as 4 digit hex numbers. To modify a memory location, move the cursor tip in the display, type the correction and press RETURN to enter the change. When you move the cursor to a line to do a screen edit, and press RETURN, the colon tells the monitor that you are re-entering data.

R DISPLAY REGISTERS

```
R PC SR AC XR YR SP C6ED 00
   20 00 20 F5
```

Registers are saved and restored upon each entry or exit from TIM. They may be modified or preloaded as in the display memory example above. The semicolon tells the monitor you are modifying registers.

- BEGIN EXECUTION

```
. G C38B
```

The GO command may have an optional address for the target. It none is specified, the PC from the R command is taken as the target.

- EXIT TO BASIC

```
.X
READY
```

Causes a warm start of BASIC. In a warm start memory is not altered in any way and BASIC resumes operation the way it was before a monitor call was made.

- LOAD

```
. L 01,MONITOR

PRESS PLAY ON TAPE :T1

OK

FOUND MONITOR

LOADING
```

No defaults on a LOAD command. The device number and the file name must be completely specified. Operating system prompts for operator intervention are the same as for BASIC. Memory addresses are loaded as specified in the file header which is set up by the SAVE command. Machine language subroutines may be loaded from BASIC but care must he taken not to use BASIC variables as the variable pointer is set to the last byte loaded + I .

S SAVE

```
. S 01,MONITOR,04000,076D

PRESS PLAY ON TAPE _-1

OK

WRITING MONITOR
```

076 is ending address +1 so that last byte of data is at 076C. You must specify I greater than last address used.

Likewise, no defaults on the SAVE command. Any start and ending address may be specified.

To cancel a command either type RETURN or press STOP to cancel a Display Memory, LOAD or SAVE.

Interrupt and breakpoint action

BRK is a software interrupt instruction which causes the CPU to interrupt execution, save PC and P registers on the stack and then branch through a vector at locations $02113and $021C. TIM initializes this vector to point at itself on entry by CALL. Unless the user modifies this vector, TIM will gain control when a BRK instruction is executed, print B* indicating entry via breakpoint (instead of C * entry via call) and the registers (as in the R command), and wait for user commands. Note that after a BRK which vectors to TIM, the user's PC points to the byte following the BRK; however, users who choose to handle BRK instructions themselves should note that BRK acts as a two-byte instruction, leaving the PC (on return via RTI) *two* bytes past the BRK instruction

IRO is vectored normally in PET to an ISR which updates the clock and scans the keyboard every *60th* of a second. If the vector is altered and the machine language subroutine does not restore it, a power-on reset must be performed.

NMI is not provided for in PET. The processor line corresponding to this interrupt is permanently pulled UP.

RESET vectors to a cold-start of BASIC. Memory is cleared. Reload and re-enter TIM via SYS command.

TIM monitor calls and special locations

```
JSR  WRT    $FFD2
JSR  RDT    $FFCF
JSR  GET    $FFE4
JSR  CRLF   S 04F2
JSR  SPACE  S 063A
JSR  WROB   $0613
JSR  RDOB   $ 065E
JSR  HEXIT  $0685
```

type a character
input a character
Get a character
type a CR type a space type a byte
read a byte Ascii to hex in A

Memory usage

$0A-S22 zero page $400-$76A absolute RAM

$23-$5A are zero page locations in the BASIC input buffer which may be used when BASIC is not using these locations. The second cassette buffer $33A-$3F F is a well protected location if that device is not used. Other memory locations may be used with considerable risk, depending on which piece of PET software wants to use it also.

Monitor checkout procedure

1) Power up your PET normally into BASIC command mode. Insert the cassette containing a monitor and use the SHIFT-RUN sequence to initiate a program load. You should see a display something like:

```
C*  PC  SR  AC  XR  YR  SP
    C6ED 29  00  88  89  FE
```

Exact values may vary, although the first and last values should he as shown.

2) The display of registers is the standard entry display message. It consists of C'* to identify entry by call, followed by the CPU register contents: program counter, processor status, accumulator, X index, Y index, and stack pointer. Note that all TIM inputs and outputs are in base 16 which is referred to as hexadecimal, or just hex. In hexadecimal the digits are 0,1,2,3,4,5,6,7,8.9,A,B,C,D,E,F. After printing the CPU registers, TIM is ready to receive commands from you. TIM indicates this "ready" status by' typing the prompting character " . " on a new line.

3) The user's CPU register may also be displayed with the R command. Type an R and press RETURN. The monitor should respond as above, but without the asterisk.

4) Displayed values may be modified by screen edit and re-entry of the line via return. Remember to type spaces to delimit fields and type 4 digit hex numbers for addresses and 2 digits for byte contents.

5) Memory may be displayed and modified using the M command. Type:

```
.M 0100,0107
```

You will see a display something like:

```
            0  1  3  4  5  6  7 0100
   20  00  30  30  30  30  30
```

Now use screen edit to modify in place on the screen, type RETURN and display again.

6) Use M and : to enter the following test program called CHSET because it prints the ASCII-64 character-set on the terminal. The M command is used to display memory locations on the PET screen and it is then possible to use screen edit on each line and type RETURN to alter memory.

```
                        * = $33A
                        CRLF = $4F2
                        WRT = $
                        FFD2

33A 20 F2 04    CHSET JSR CRLF
33D A2 20             LDX #$20
33F 8A          LOOP  TXA
340 20 D2 FF          JSR WRT
343 E8                INX
344 E0 60             CPX y$60
346 DO F7             BNE LOOP
348 00                BRK
349 4C 3A 03          JMP CHSET
```

```
M 033A,034B
  033A 20 F2 04 A2 20 8A
  20 D2 0342 F F E8 E0
  60 DO F7 00 4C 034A 3A
  03
```

7) CHSET was assembled to reside in the *2nd* cassette buffer. Type:

```
.G 033A
```

to execute the program.

The listing should look like this:

```
 !"#$%'()*, - . / 0 1 2 3 4 5 6 7 8 9 : ; _ ?@ A B
C D E F G H I J K L M N O P Q R S T U V  W X Y Z [I]
B* PC SR AC XR YR SP
 .; 0349 3B 5F 60 8D FE
```

Note the address contained in the PC. It is possible to type G execute the program again without specifying an address.

8) Next we will link CHSET with BASIC. First replace the BRK instruction in location $348 with an RTS (return subroutine) 1 change $348 from 00 to 60).

9) ('hinge the USR function vector in locations 1 to point at the subroutine at $33A.

```
.: 0000 4C 3A 03
```

10) Exit from the monitor and re-enter BASIC.

.x

READY

1 1) Prow that the linkage is established by using both **SYS** and **USR.**

A = **USR (0)**

(Enter these as direct base commands)

SYS (3 * 256 + 3 * 16 + 10)

PET MEMORY MAP (IN 4K BLOCKS)

F	1/0, Diagnostics, Monitor RUM	
E	$E80O- I/O Ports and Expansion I/O $E000-E7FF Screen Editor ROM	
D C	Basic ROM	
B A 9	Expansion ROM	
8	$8000-$83E7 TV display	RAM
7 6 5 4 3	Expansion RAM	
I	Basic Text RAM (8k Version)	
Page 0	$A-$5A	Basic Input Buffer
Page I	Stack	
Pages 2-3	S-00 S-)1() $2113 $27A $33A	3 byte clock register Interrupt vector Break inst. vector Buffer for cassette = 1 Buffer for cassette = 2
Pages 4-8	$400-$FFF	Basic Text RAM

Notes