

Disk/Extramon 64

Mike Forani
Burlington, Ontario

Disk/Extramon 64 is the all-in-one ultra-monitor for the Commodore 64. It has just about everything you could ask for, and then some. Just one problem though – it's over 7K long! And 8 pages of DATA statements not only consumes too much magazine space, but makes hand entry far too impractical. So why print instructions without the program? Mike uses standard commands and command syntax which makes the instructions practically universal for all other monitor utilities. Mike's program may have commands that the others don't, but odds are the others have none that Mike doesn't include. Two assembled versions of Disk/Extramon 64 are available on Transactor Disk #5. – M.Ed.

Embarking on an investigative journey through your 64? Yes? Then Disk/Extramon 64 is a travel companion you shouldn't be without. It's the monitor program to end all monitor programs.

A lot of you may be saying, "Why should I bother with this one?". Agreed, there are several monitor programs around, some for sale, others for free. Disk/Extramon 64 has features not found in other monitors which was going to be a strong "selling point" for the program. However, the program is now public domain so all those selling points make the fact that it's free even more attractive.

Disk/Extramon 64 has all the common machine language monitor commands such as Hunt, Assemble and Disassemble, Transfer, plus Newlocate, Interrogate, Compare, Quick Trace, and Bank Switching commands especially for the 64. Hex/Decimal conversions are in there too.

The Disk Monitor portion of the program has everything the budding young drive programmer needs for experimenting with the inner workings.

Note: The program has been tested with Commodore equipment only. I therefore cannot insure that it will work properly with non-Commodore printers, disk drives, or IEEE interfaces. (There has been some success with the 4040 dual drive and the Bus-Card II interface.)

The following is a list of the Disk/Extramon 64 commands. Some of the commands require special attention so please read on.

Affected Memory

This program is located at \$1000 or at \$8000 and uses 8K of memory. The page 3 vectors; IRQ, BRK, and ICRNCH are changed and the upper 5 bytes of page 0 are used by the monitor program. A number of kernel routines are also used and these will affect some zero page variables. CHRGET is used by the monitor program and this will affect the page 2 input buffer. On a break instruction before anything can change, zero page and page two are saved at \$9e00 to \$9fff with the \$8000 version or at \$2e00 to \$2fff with the \$1000 version of the monitor program. Therefore you will always be able to see what zero page or page two locations your program affected. Also on a break instruction the VIC chip's video and character generator registers are reset to their defaults as is the I/O port in page 0.

Note: While in the monitor all numeric input must be in hexadecimal numbers except when doing decimal to hexadecimal conversion.

Monitor Commands

DISPLAY REGISTERS: Display the current processor status.
r

DISPLAY MEMORY: Display contents of memory in hex.

m adr1 adr2 adr1 ;beginning address
 adr2 ;ending address (optional)

If adr2 is left out then one line of eight bytes will be displayed.

ALTER MEMORY: Alter contents of the 64's memory
.: 1000 00 00 00 00 00 00 00 alter memory

ALTER REGISTERS: Alter contents of 64's processor registers.

pc irq sr ac xr yr sp
.; 1000 ea31 b0 00 00 00 ff alter processor status

GO: Begin execution of a machine language program.

g adr1 adr1 ;beginning address of execution (optional)

* If the kernel is banked out the processor status is not restored at the 'go' command and the IRQ's are disabled. Also if the address is omitted or invalid, the 64 will jump to address in the program counter.

LOAD: Load a program into the 64's memory.

l "sdr:filename",dn,adr1

sdr ;source drive number (optional)
dn ;device number (08 – 0f)
adr1 ;load address (optional, defaults to the disk load address)

SAVE: Save a program from 64 to disk.

s "ddr:filename",dn,adr1,adr2

ddr ;destination drive # (optional)
dn ;device number (08 – 0f)
adr1 ;beginning address of save
adr2 ;end address of save (last byte is saved)

EXIT: Exit the monitor to BASIC.

x * All wedges are left intact so the monitor may be reentered.

Extra Monitor Commands

MONITOR: Enter monitor from BASIC.

mon

BANK: The kernal and/or basic may be banked out of memory so that the RAM memory sitting behind it may be modified, assembled, saved, executed or traced.

bbout bank out the basic ROM to give RAM
bbin bank in the basic ROM
bkout bank out the kernal ROM to give RAM
bkin bank in the kernal ROM

TRANSFER: A portion of memory may be transferred from one memory location to another.

t adr1 adr2 adr3 adr1 ;start address
adr2 ;end address
adr3 ;beginning address of transfer

FILL: Fill a portion of memory with a given value.

f adr1 adr2 xx adr1 ;start address
adr2 ;end address
xx ;value to fill memory with

HUNT: Hunt for a string of values in a specified portion of memory.

h adr1 adr2 'string' adr1 ;start address
h adr1 adr2 xx xx xx adr2 ;end address for hunt
'string' ;characters to be searched for
xx ;hex values to be searched for
(max. length of string or bytes is 20)

COMPARE: Compare two portions of memory to each other.

c adr1 adr2 adr3 adr1 ;start address
adr2 ;end address
adr3 ;start address (second block)

* memory locations that do not compare equal will be displayed.

INTERROGATE: Display the screen printable characters along with the memory locations values.

i adr1 adr2 adr1 ;start address
adr2 ;end address (optional)

QTRACE: Trace a machine language routine and display the processor status after each instruction is executed.

q adr1 adr1 ;address to begin execution
* Pressing the 'n' key skips the trace
* Pressing the 'm' key speeds it up
* Pressing the space bar halts execution

* No separate interrupt control, unless non maskable, is allowed during the trace routine and any i/o routines may be affected. The qtrace works on an interrupting system. Interrupts occur after each instruction is executed, therefore IRQ control in the program being traced may crash the system. (CIA #1 - Timer A is used for interrupt timing.)

ASSEMBLE: Assemble a machine language program in memory.
(this is a simple assembler)

a adr1 lda #\$41 adr1 ;beginning address for assembly

* To end assembling a return (blank line) must be entered before doing any other operations such as altering the assembled code.

DISASSEMBLE: Disassemble hexadecimal memory location values into mnemonic op-codes with operands.

d adr1 adr2 adr1 ;start address
adr2 ;end address (optional)

* If adr2 is left out then only one op-code and its operand will be displayed.

ALTER DISASSEMBLY: Change the screen disassembly.
., 1000 20 d2 ff jsr \$ffd2

* The hex values are to be changed not the mnemonics.

NEW LOCATER: Relocate a machine language program.

n adr1 adr2 offset adr3 adr4 w

adr1 ;beginning address of code to be relocated
adr2 ;end address
offset ;value to be added to absolute indexed memory locations
adr3 ;lower address limit of absolute addressed data which is to be changed
adr4 ;upper limit
w ;relocating a word table - if included

* The code to be relocated must first be transferred, this is a two step command.

DEC/HEX CONVERSION: Convert a hexadecimal number to a decimal number or a decimal number to a hexadecimal number.

***65535** ;decimal to hexadecimal
***\$ffff** ;hexadecimal to decimal

KILL: The disk/extra monitor wedges are destroyed and normal basic operations may be done - the monitor may not be reentered unless you jump to the start of the program. ie. \$1000 or \$8000.

k * All the page three vectors used are restored.

COLD: Do a power on reset sequence.

P

Disk Monitor Commands

DIRECTORY: Do a screen list of the directory.

/ ;directory of disk
/ "m" ;directory of files starting with the letter 'm'
/ "1:" ;directory of drive 1

READ: Read a sector from the disk to a disk buffer.

\$r dd tt ss bb dd ;drive
tt ;track
ss ;sector
bb ;buffer (optional, default is 01)

WRITE: Write a disk buffer to the disk surface.

\$w dd tt ss bb dd ;drive
tt ;track
ss ;sector
bb ;buffer (optional)

GET: Get disk memory to the 64's memory.

\$g adr1 adr2 adr3 adr1 ;start address of get
adr2 ;end address
adr3 ;address to store at in C64

PUT: Put 64's memory to disk memory.

\$p adr1 adr2 adr3 adr1 ;start address of put
adr2 ;end address
adr3 ;address to store at in drive

VIEW: View the disk drives memory.

\$v adr1 adr2 adr1 ;start address
 adr2 ;end address (optional)

ALTER: Alter the disk drives memory.

.\$:0300 00 00 00 00 00 00 00

DIRECT: Send a direct command to the disk drive.

\$>... ;any basic 2.0 disk command

* The disk status is displayed after the command is executed.

TRACE: Trace a files track and sector links and display them. (begin tracing at. .)

\$t dd tt ss bb dd ;drive
 tt ;track
 ss ;sector
 bb ;buffer (optional)

FETCH: Fetch a sector from the disk drive surface to the 64's memory.

\$f adr1 dd tt ss bb adr1 ;start address in C64
 dd ;drive
 tt ;track
 ss ;sector
 bb ;buffer (optional)

DUMP: Dump a block of the 64's memory to the disk surface.

\$d adr1 dd tt ss bb adr1 ;start address in C64
 dd ;drive
 tt ;track
 ss ;sector
 bb ;buffer (optional)

CHANGE: Change the device number of the disk drive. (send to drive or just for program defaults.)

\$c do dn* do ;old device number
 dn ;new device number

* If the asterisk is included the change is only done in the 64's memory so that a device 09-0f may be used as a default if hard wired.

ALLOCATE: Allocate a sector as being used in the BAM.

\$a dd tt ss dd ;drive
 tt ;track
 ss ;sector

* To de-allocate sectors use the basic 2.0 command Validate (v0)

EXECUTE: Execute disk memory.

\$e adr1 adr1 ;beginning of execution

BLOCK EXECUTE: Load a sector off the disk surface into a disk buffer and execute it.

\$b dd tt ss bb dd ;drive
 tt ;track
 ss ;sector
 bb ;buffer (optional)

STATUS: Check the disk status.

\$s

INTERROGATE: Display screen printable characters while displaying the memory of the disk drive.

\$i adr1 adr2 **adr1 ;start address**
 adr2 ;end address (optional)

Note 1 After doing any disk memory commands the drive should be initialized to avoid any unfriendly errors. (\$>i0)

Note 2 An automatic scroll up and down is built into the memory display routines and the disassemble.

Note 3 Pressing the run/stop and restore keys will reset the computers page 3 vectors, this will result in the monitor not working on the scroll routines therefore the monitor must be exited and reentered at its starting point to reset the vectors once more.

Disk/Extramon 64 Quick Reference Chart

MONITOR COMMANDS

a adr1	simple assembler
bbin	bank basic in
bbout	bank basic out
bkin	bank kernal in
bkout	bank kernal out
c adr1 adr2 adr3	compare memory
d adr1 [adr2]	disassemble memory
f adr1 adr2 xx	fill memory
g [adr1]	begin execution of program
h adr1 adr2 'string'	search memory for string
h adr1 adr2 xx xx xx ...	search memory for bytes
i adr1 [adr2]	interrogate memory
k	kill monitor wedges and exit
l "sdr:filename",dn,[adr1]	load a file from disk
m adr1 [adr2]	display memory bytes
mon	enter monitor from basic
n adr1 adr2 offset adr3 adr4 [w]	relocate program code
p	do a power on reset sequence
q [adr1]	quick trace of program code
r	display processor registers
s "sdr:filename",dn,adr1,adr2	save a file to disk
t adr1 adr2 adr3	transfer memory
x	exit the monitor to basic
* xxxxx	decimal to hex conversion
*\$ xxxx	hex to decimal conversion

DISK MONITOR COMMANDS

\$a dd tt ss	allocate a sector in BAM
\$b dd tt ss [bb]	block execute
\$c do dn	change disk device number
\$c do dn*	change disk default device #
\$d adr1 dd tt ss [bb]	dump memory to disk
\$e adr1	execute disk memory
\$f adr1 dd tt ss [bb]	fetch sector from floppy
\$g adr1 adr2 adr3	get disk memory
\$i adr1 [adr2]	interrogate disk memory
\$p adr1 adr2 adr3	put memory to disk memory
\$r dd tt ss [bb]	read a sector to disk buffer
\$s	check disk status
\$t dd tt ss [bb]	trace file link pointers
\$v adr1 [adr2]	view disk memory
\$w dd tt ss [bb]	write a buffer to disk
\$> 'string'	send disk command
/	directory