Disk/Extramon 64

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
kernal 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 adr2adr1 ;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 adr1adr1 ;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 the program counter.

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

l * 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 * 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 kernel 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 kernel ROM to give RAM
bkin bank in the kernel ROM

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

\[ t \text{adr1} \text{adr2} \text{adr3} \]\n\[ \text{adr1} : \text{start address} \]
\[ \text{adr2} : \text{end address} \]
\[ \text{adr3} : \text{beginning address of transfer} \]

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

\[ f \text{adr1} \text{adr2} \text{xx} \]
\[ \text{adr1} : \text{start address} \]
\[ \text{adr2} : \text{end address} \]
\[ \text{xx} : \text{value to fill memory with} \]

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

\[ h \text{adr1} \text{adr2} \text{'string'} \text{adr1} \]
\[ \text{adr1} : \text{start address} \]
\[ \text{adr2} : \text{end address for hunt} \]
\[ \text{'string'} : \text{characters to be searched for} \]
\[ \text{xx} : \text{hex values to be searched for} \]
\[ (\text{max. length of string or bytes is 20}) \]

COMPARE: Compare two portions of memory to each other.

\[ c \text{adr1} \text{adr2} \text{adr3} \]
\[ \text{adr1} : \text{start address} \]
\[ \text{adr2} : \text{end address} \]
\[ \text{adr3} : \text{start address (second block)} \]

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

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

\[ i \text{adr1} \text{adr2} \]
\[ \text{adr1} : \text{start address} \]
\[ \text{adr2} : \text{end address (optional)} \]

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

\[ q \text{adr1} \]
\[ \text{adr1} : \text{address to begin execution} \]
\[ * \text{Pressing the 'n' key skips the trace} \]
\[ * \text{Pressing the 'm' key speeds it up} \]
\[ * \text{Pressing the space bar halts execution} \]

* 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.

\[ a \text{adr1} \text{lda} \#841 \]
\[ \text{adr1} : \text{beginning address for assembly} \]

* 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.

\[ d \text{adr1} \text{adr2} \]
\[ \text{adr1} : \text{start address} \]
\[ \text{adr2} : \text{end address (optional)} \]

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

ALTER DISASSEMBLY: Change the screen disassembly.

\[ ., 1000 \text{d2 ff jsr $f0d2} \]

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

NEW LOCATER: Relocate a machine language program.

\[ n \text{adr1} \text{adr2} \text{offset} \text{adr3} \text{adr4} \text{w} \]
\[ \text{adr1} : \text{beginning address of code to be relocated} \]
\[ \text{adr2} : \text{end address} \]
\[ \text{offset} : \text{value to be added to absolute indexed memory locations} \]
\[ \text{adr3} : \text{lower address limit of absolute addressed data which is to be changed} \]
\[ \text{adr4} : \text{upper limit} \]
\[ \text{w} : \text{relocating a word table – if included} \]

* 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.

\[ \text{*65535} : \text{decimal to hexadecimal} \]
\[ \text{*$ffff} : \text{hexadecimal to decimal} \]

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.

\[ k \]
\[ * \text{All the page three vectors used are restored.} \]

COLD: Do a power on reset sequence.

\[ p \]

Disk Monitor Commands

DIRECTORY: Do a screen list of the directory.

\[ / \]
\[ / \text{directory of disk} \]
\[ / \text{'m'} \]
\[ / \text{directory of files starting with the letter 'm'} \]
\[ / \text{'l'} \]
\[ / \text{directory of drive 1} \]

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

\[ \text{sr \text{dd \text{tt \text{ss \text{bb}}} \text{dd : drive}}} \]
\[ \text{tt : track} \]
\[ \text{ss : sector} \]
\[ \text{bb : buffer (optional, default is 01)} \]

WRITE: Write a disk buffer to the disk surface.

\[ \text{sw \text{dd \text{tt \text{ss \text{bb}}} \text{dd : drive}}} \]
\[ \text{tt : track} \]
\[ \text{ss : sector} \]
\[ \text{bb : buffer (optional)} \]

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

\[ \text{sg \text{adr1 \text{adr2 \text{adr3}}} \text{adr1 : start address of get}} \]
\[ \text{adr2 : end address} \]
\[ \text{adr3 : address to store at in C64} \]

PUT: Put 64's memory to disk memory.

\[ \text{sp \text{adr1 \text{adr2 \text{adr3}}} \text{adr1 : start address of put}} \]
\[ \text{adr2 : end address} \]
\[ \text{adr3 : address to store at in drive} \]
VIEW: View the disk drives memory.
$V \text{ adr1} \text{ adr2} \quad \text{adr1: start address}
\quad \text{adr2: end address (optional)}

ALTER: Alter the disk drives memory.
.8:0300 00 00 00 00 00 00 00

DIRECT: Send a direct command to the disk drive.
$> \ldots \quad \text{any basic 2.0 disk command}

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

TRACE: Trace a file's track and sector links and display them. (begin tracing at . . . )
$T \text{ dd tt ss bb} \quad \text{dd:drive}
\quad \text{tt:track}
\quad \text{ss:sector}
\quad \text{bb:buffer (optional)}

FETCH: Fetch a sector from the disk drive surface to the 64's memory.
$F \text{ adr1 dd tt ss bb} \quad \text{adr1: start address in C64}
\quad \text{dd:drive}
\quad \text{tt:track}
\quad \text{ss:sector}
\quad \text{bb:buffer (optional)}

DUMP: Dump a block of the 64's memory to the disk surface.
$D \text{ adr1 dd tt ss bb} \quad \text{adr1: start address in C64}
\quad \text{dd:drive}
\quad \text{tt:track}
\quad \text{ss:sector}
\quad \text{bb:buffer (optional)}

CHANGE: Change the device number of the disk drive. (send to drive
or just for program defaults.)
$C \text{ do dn*} \quad \text{do: old device number}
\quad \text{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 \text{ dd tt ss} \quad \text{dd:drive}
\quad \text{tt:track}
\quad \text{ss:sector}

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

EXECUTE: Execute disk memory.
$E \text{ adr1} \quad \text{adr1: beginning of execution}

BLOCK EXECUTE: Load a sector off the disk surface into a disk buffer and execute it.
$B \text{ dd tt ss bb} \quad \text{dd:drive}
\quad \text{tt:track}
\quad \text{ss:sector}
\quad \text{bb:buffer (optional)}

STATUS: Check the disk status.
$S

INTERROGATE: Display screen printable characters while displaying the memory of the disk drive.
$I \text{ adr1 adr2} \quad \text{adr1: start address}
\quad \text{adr2: end address (optional)}

Note 1 After doing any disk memory commands the drive should be initialized to avoid any unfriendly errors. (S>0)

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

Note 3 Pressing the run/stop and restore keys will reset the computer's 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a adr1</td>
<td>simple assembler</td>
</tr>
<tr>
<td>bbin</td>
<td>bank basic in</td>
</tr>
<tr>
<td>bbout</td>
<td>bank basic out</td>
</tr>
<tr>
<td>bkin</td>
<td>bank kernal in</td>
</tr>
<tr>
<td>bcout</td>
<td>bank kernal out</td>
</tr>
<tr>
<td>c adr1 adadr2 adadr3</td>
<td>compare memory</td>
</tr>
<tr>
<td>d adadr1 [adr2]</td>
<td>disassemble memory</td>
</tr>
<tr>
<td>f adadr1 adadr2 xx</td>
<td>fill memory</td>
</tr>
<tr>
<td>g [adr1]</td>
<td>begin execution of program</td>
</tr>
<tr>
<td>h adadr1 adadr2 string</td>
<td>search memory for string</td>
</tr>
<tr>
<td>h adadr1 adadr2 xx xx xx . . .</td>
<td>search memory for bytes</td>
</tr>
<tr>
<td>l adadr1 [adr2]</td>
<td>interrogate memory</td>
</tr>
<tr>
<td>k</td>
<td>kill monitor wedges and exit</td>
</tr>
<tr>
<td>l &quot;sdr:filename &quot;,dn,adr1</td>
<td>load a file from disk</td>
</tr>
<tr>
<td>m adadr1 [adr2]</td>
<td>display memory bytes</td>
</tr>
<tr>
<td>mon</td>
<td>enter monitor from basic</td>
</tr>
<tr>
<td>n adadr1 adadr2 offset adadr3 adadr4 [w]</td>
<td>relocate program code</td>
</tr>
<tr>
<td>p</td>
<td>do a power on reset sequence</td>
</tr>
<tr>
<td>q [adr1]</td>
<td>quick trace of program code</td>
</tr>
<tr>
<td>r</td>
<td>display processor registers</td>
</tr>
<tr>
<td>s &quot;sdr:filename &quot;,dn,adr1,adr2</td>
<td>save a file to disk</td>
</tr>
<tr>
<td>t adadr1 adadr2 adadr3</td>
<td>transfer memory</td>
</tr>
<tr>
<td>x</td>
<td>exit the monitor to basic</td>
</tr>
<tr>
<td>+xxxxx</td>
<td>decimal to hex conversion</td>
</tr>
<tr>
<td>-$xx</td>
<td>hex to decimal conversion</td>
</tr>
</tbody>
</table>

DISK MONITOR COMMANDS

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a dd tt ss</td>
<td>allocate a sector in BAM</td>
</tr>
<tr>
<td>$b dd tt ss [bb]</td>
<td>block execute</td>
</tr>
<tr>
<td>$c do dn</td>
<td>change disk device number</td>
</tr>
<tr>
<td>$c do dn*</td>
<td>change disk default device</td>
</tr>
<tr>
<td>$d adadr1 dd tt ss [bb]</td>
<td>dump memory to disk</td>
</tr>
<tr>
<td>$e adadr1</td>
<td>execute disk memory</td>
</tr>
<tr>
<td>$f adadr1 dd tt ss [bb]</td>
<td>fetch sector from floppy</td>
</tr>
<tr>
<td>$g adadr1 adadr2 adadr3</td>
<td>get disk memory</td>
</tr>
<tr>
<td>$l adadr1 [adr2]</td>
<td>interrogate disk memory</td>
</tr>
<tr>
<td>$p adadr1 adadr2 adadr3</td>
<td>put memory to disk memory</td>
</tr>
<tr>
<td>$r dd tt ss [bb]</td>
<td>read a sector to disk buffer</td>
</tr>
<tr>
<td>$s</td>
<td>check disk status</td>
</tr>
<tr>
<td>$t dd tt ss [bb]</td>
<td>trace file link pointers</td>
</tr>
<tr>
<td>$v adadr1 [adr2]</td>
<td>view disk memory</td>
</tr>
<tr>
<td>$w dd tt ss [bb]</td>
<td>write a buffer to disk</td>
</tr>
<tr>
<td>$&gt; &quot;string&quot;</td>
<td>send disk command</td>
</tr>
<tr>
<td>/</td>
<td>directory</td>
</tr>
</tbody>
</table>