megamon command summary (loads from $9ff6 to $be21)

```
a or *; mini assembler
a <addr><memonic><operand>

b ; set/display breakpoint
b <brkpt#><addr> (b=display)

c ; compare memory
c <start><end><target>

d ; dissassemble memory
d <start><end>

e ; memory display w/checksum
e <start><end>

f ; fill memory
f <start><end><byte>

g ; execute program 'go'
g <addr>

h ; hunt memory
h <start><end>[<number:string>]

i ; set/display irq vector
i <addr> (i alone to display)

j ; reset monitor

k ; kill breakpoint
k <breakpoint#>

l ; load a file
l <"name"><dev><addr>

m ; memory dump (hex or ascii)
m <start><end>

n ; dynamic memory scan
n <addr>

o ; output redirection
o <dev><secondary addr><"text">

p ; display disk dir
p <"mask"><dev> or (p <return>)

q ; quit (cold start)

r ; display/chg  registers

s ; save a file
s <"name"><dev><start><end>

t ; transfer block of memory
t <start><end><destination>

u ; user definable command
u <anything>

v ; set/display user vectors
v <addr> (alone to display)

w ; single step (walk)
w <addr> see walk command

x ; exit monitor (warm start)
x caution see eXit command

y ; set colors
y <text><bgnd><brdr> (0-
15)decimal

% ; comment 'null' line
% <anything>

# ; convert number
# 34f0 or # &49152

+ ; add numbers
+ ffd2 &32768

- ; subtract numbers
- ffd2 d020

? ; display help file
? "name" dev (optional)

@ ; i/o to disk (like wedge)
@ "n0:name,00"

: modify memory
: addr bytes

! ; text to memory
! c000 "text at c000"

; ; 'mass register change
; pc a x y sp sr  or r cmd

^ ; configuration mode
```

```
numbers may be input in these formats
hex $ is optional. $ffd2 or ffd2.
decimal & prefix e.g. &49152
ascii  " prefix e.g. "text"
binary % prefix e.g. %10101110
```

```
use run/stop to abort and the
space bar to toggle output
memory dumps etc.
```

megamon commands

command: mini-assembler
format: a [addr[mnemonic]]
* [addr[mnemonic]]
example: a c000 lda #$20
function: assembles <mnemonic> to memorylocation designated by <addr>. the next available memory location is then displayed. if <addr> or <mnemonic> is missing, the command is ignored, and the normal monitor prompt returns.

command: set/display breakpoints
format: b [bp#[addr]]
example: b 0 c000
function: allows the user to set up to 8 "break points" in a ml program by replacing the byte at <addr> with a brk instruction. the byte at <addr> is saved, and can be replaced by using the "k" (kill breakpoint) command. <bp#> specifies the breakpoint number, and can be between 0-7. if no parameters are given, the current breakpoints are displayed.

command: compare memory blocks
format: c start end target
example: c 1000 2000 c000
function: compares the memory between <start>-<end> with <target>, and prints any differences found.

command: disassemble memory
format: d [start[end]]
example: d a474 a500
function: displays a disassembly of the memory between <start>-<end>, if both parameters are specified. if only <start> is given, the single instruction at that location is displayed. if no parameters are specified, the instruction at the current pc is displayed.
command: memory display with checksum
format: e [start[end]]
example: e 5f60 6000
function: displays memory between <start> and <end> in hex format, 8 bytesto a line. at the end of each line a 1-byte line checksum is displayed. after the display is complete, a 2-byte total checksum is shown. if only 1 parameter is given, the 8 bytes following <start> are displayed. if no parameters are given, the 8 bytes following the current pc are shown.

command: fill memory block with byte
format: f start end byte
example: f c000 cfff 0
function: fills all memory between
<start> and <end> with <byte>. if byte
isn't specified it defaults to 00.

command: execute program
format: g [addr]
example: g c800
function: performs a jsr to <addr> if
given. if not specified, a jsr to the
current pc takes place. a user program
must end with an rts or brk instruction
to return to the monitor.

command: hunt memory for data
format: h start end [nbr/string]
example: h c000 c800 "txt" 0 "more" 20
function: searches memory between
<start> and <end> for the string and/or
number sequence given. If a match is
found, the all locations of the
occurance are displayed.

command: set/display irq vector
format: i [addr]
example: i cf00
function: allows the user to define a
routine to be serviced by the 1/60 sec.
processor interrupt. the routine should
(normally) end with a jmp $ea31
instruction. if <addr> is not given,
the current irq vector (normally $ea31)
is displayed.

command: reset monitor
format: j
example: j
function: re-initializes the monitor
and displays the boot-up screen. some
of the functions performed: irq is
reset to $ea31, all breakpoints are
cleared, the pseudo-registers are set
to their default values and the
screen/text colors are returned to
their default settings.

command: "kill" (reset) a breakpoint
format: k [bp#]
example: k 0
function: resets the breakpoint
specified by <bp#>. specifically, it
removes the brk instruction placed by
the b (breakpoint) command, and
replaces it with it's former contents.
if <bp#> is not given, the status of
all 8 available breakpoints is shown.

command: load a file from disk or tape
format: l "filename" [device[addr]]
example: l "a file"
function: loads a file from disk or tape into memory. if <device> is not given, the current default is used (see the "^" command for info on the default device). if <addr> is not given, the file is loaded at it's internal "load address". if given, the file will be force-loaded starting at <addr>.

command: memory dump in hex & ascii
format: m [start[end]]
example m a000 a400
function: displays the contents of memory between <start> and <end> in hex and ascii, 8 bytes to a line. if the display is going to the printer, numbers0-31 and 128-159 (decimal) are converted to "."'s before being displayed. if going to screen, only the carriage return (13 and 141) is converted - all other bytes are displayed as if in "quote mode".
as with the "e" command, all parameters are optional.

command: dynamic memory scan
format: n addr
example: n a0
function: displays the memory contents between <addr> and <addr>+$1f at the top of the screen, and updates this display continuously. this is useful for examining hardware locations and storage areas for interrupt routines (the example shows the memory used by the kernal real-time clock). the space bar will "freeze" the display and pressing run/stop terminates the scan.

command: output redirection
format: o [device[s.a.["text"]]]
example o 4 4
function: opens a channel to the device specified, and redirects all monitor output to this device. all parameters are optional, and if none are given the output is directed back to the screen. note that the n,w,p,@, and ^ commands reset output redirection.

command: display disk directory
format: p ["dir mask"[device]]
example: p "mon*"
function: displays the disk directory
on <device> using <"dir mask"> as a
directory display mask (see your disk
manual for details on this). if
<"dir mask"> is not given, the entire
directory is displayed, if <device> is
omitted, the current default is used.
this command can only be used on
devices 8 and above, otherwise the
command is ignored.

command: exit monitor (cold start)
format: q [addr]
example: q
function: returns control back to basic
after performing a "power on" reset.
<addr> is usually not given, but if it
is, the monitor will simply jump to the
location given. (the cold start vector
can be changed - see the "^" command).

command: display/change registers
format: r [regname value]
example: r a 20
function: displays the current monitor
"pseudo registers" that are restored
before executing the g,q, or x
commands.this command can also be used
to change single registers by
specifying a register name
(a,x,y,s,p or *) and a new value.
the register display consistsof a
banner with the register values
displayed underneath it. the status
register is further broken down into
it's respective flag values, with a
reversed letter representing a 'set'
condition.

command: save memory to device
format: s "filename" device start end
example: s "testfile" 8 c200 c3f0
function: saves the block of memory
between <start> and <end> to <device>.
all parameters must be given.

command: move (transfer) memory block
format: t start end destination
example: t a000 a400 c000
function: performs a non-destructive
move of the memory between <start> and
<end> to <destination>. if the blocks
overlap, the command will automatically
perform the transfer in the nessesary
direction to preserve the data.

command: user-definable command
format: u [anything]
example: u
function: this command is open-ended, waiting for you to impliment. it's vector may be defined by using the 'v' command or in the configuration mode. for directions on implimentation, refer to the article.

command: define user vector
format: v [addr]
example: v c000
function: sets the user (u) command execution vector to <addr>. (this can also be done in configuration mode). if no <addr> is given, the current user vector is displayed.

command: single-step (walk) program
format: w [addr[ar[xr[yr[sp[sr]]]]]]
example: w c000
function: this is perhaps one of the most useful monitor commands. it allows you to single-step a machine language program. when this command is executed, the monitor goes into "walk mode", displaying the registers, the instruction to be executed, and some directions. for more info, refer to the article.

command: exit monitor (warm start)
format: x [addr]
example: x
function: basically the same as the "q" command, execpt the monitor terminates to the basic warm start ("ready") vector. the warm start can be changed in the configuration mode. caution many ml operations change the basic pointers and using x to exit the monitor can lock up the computer.

command: set text & screen colors
format: z text [backgnd[border]]
example: z 1 0
function: allows you to set the text and screen colors to your liking. the <text> color must be given, and <backgnd> and <border> are optional. colors range from 0-15 (decimal).

command: comment line
format: % [anything]
example   % this is a comment
function: none. the text after the "%"
is ignored. (useful for nullifying
lines when using the screen editor to
move from place to place)

command: convert number to other forms
format: # [number]
example: # 34f0
function: allows you to convert a
number in one form to the four standard
monitor forms - hex, decimal, ascii
and binary. this command will display a
line containing these equivilents for
the number given.

command: add numbers
format: + nbr1 nbr2
example: + 400 &120
function: adds the two numbers together
using unsigned 16-bit arithmetic, and
displays the result in hex, decimal,
ascii and binary
(as with the "#" command).

command: subtract numbers
format: - nbr1 nbr2
example   - 130 c82f
function: subracts <nbr1> from <nbr2>
using unsigned 16-bit arithmetic, and
prints the result in hex, decimal,
ascii and binary (eg: # and + commands)
both the "+" and "-" commands will not
display any carry resulting from their
operation.

command: display help file
format: ? ["filename"[device]]
example: ?
function: normally prints the "default"
help file which you are reading right
now in paginated (22 lines/pg) format.
if <"filename"> is specified, that file
will be printed, assuming it exists and
that it is a seqential file (type seq).
if <device> is not given, the default
is assumed. if you do not wish to view
the entire help file, pressing run/stop
will abort this command. note that if
parameters are given, the file
specified will become the "default"
help file (this can also be set in
configuration mode).

command: i/o to disk command channel
format: @ ["diskcmd"[device]]
example: @ "n:new disk,00"
function: acts much like the commodore disk wedge, allowing you to either check the status of the default disk channel (no parameters), or send a disk command to a disk <device>. if <device> is not given, the default is assumed. if the <device> number is <8, the command is ignored. note that you can check the status of a drive other than the default by specifiying a "null string"; e.g. @ "" 9

command: modify memory
format: : [addr[bytes]]
example: : c800 "x" $50 &120 %11010
function: allows you to change the contents of memory to any desired value (0-ff hex). note that the "e" and "m" commands prefix their displays with a ":", allowing you to use the screen editor to edit a memory display. if <addr> or <bytes> are omitted, the command is ignored and the normal monitor prompt returns, otherwise the next available address is displayed as a prompt.

command: put text in memory
format: ! [addr["text"]]
example: ! c000 "text at $c000"
function: similar in ways to the ":" command, but allows an ascii string to be put directly into memory (instead of numeric data). parameter handling is the same as the ":" command.

command: "mass" register change
format: ; [pc[ar[xr[yr[sp[sr]]]]]]
example: ; c000 10 0 0 e0 0
function: used to set the monitor's internal "pseudo registers" that are restored before the g,q,w, or x commands are executed. usually, you would use the "r" command to obtain a register value display (which is prefixed by a ";" for your convinience) and then use the screen editor to modify the desired registers.

command: enter configuration mode
format: ^
example: ^
function: this command is used to enter the monitor's configuration screen, where several miscellaneous parameters such as the default i/o device, help file name, warm,cold and user vectors, help file name, et cetra can be set and permanantly saved to disk or tape. for more information, refer to the article.

other useful information:

on specifiying parameters: string parameters (such as filenames) must be entered between double quotes ("). addresses and byte parameters can be entered in any one of four forms:

hexdecimal: optional $ prefix; $c000
decimal: & prefix: e.g. &49152
ascii: " prefix; e.g. "a"
binary: % prefix; e.g. %1010010111

output control: for commands that could potentially produce voluminous output, the space bar can be used to halt the command output, and the run/stop key can be used to abort it. the commands that this works for are c,d,e,h, and m.

notes on output redirection: most commands will work normally when the "o" command is used to redirect output, but some commands object to this, namely the n,p,w,@ and ^ commands. these commands will usually reset output redirection.

"next address" prompting: the a,:, and ! commands will display the next available memory location prefixed with their respective command symbols as a prompt after they are done executing. simply press <return> after this prompt to return to the normal monitor prompt if you do not wish to enter more data. the w and ^ are more complex commands, and have their own special prompts.

the end !!!!!