A Simple Monitor For The VIC

atarimagazines.com/compute/issue20/199_1_VIC-20_Update.php

VIC-20 Update

TINYMON1: A Simple Monitor For The VIC

Jim Butterfield Toronto, Canada

One of the things you may miss on the VIC, is a Machine Language Monitor: it's not there.

Commodore will be releasing a very powerful MLM on a plug-in cartridge, and serious programmers will certainly want to use it. But for occasional use, a tape-loadable MLM might be very handy.

Here's an early version that may be of use. It should fit on any VIC, with or without extra memory added; and it honors all the commands from the built-in Monitors we know from PET/CBM usage. One minor syntax change: the two addresses of the Memory display command (.M) should be separated by a space rather than a comma.

It's not really practical to type TINYMON directly into a VIC. DATA statements in decimal would take up more room than is available in small VICs; and hex entry would need an MLM to be in place already. So I've prepared the program so that it can be entered on a PET and saved on tape. After it's been created once, the VIC can make its own copies. You'll need a PET with Upgrade ROM or 4.0 ROM to do the job, since the Original ROM PETs don't have a Machine Language Monitor and things would get too complicated.

TINYMON loads like a BASIC Program, and copies can be made with a simple LOAD and SAVE sequence as you would do with BASIC. When you load TINYMON and say RUN, however, some interesting things happen … the monitor system is repacked into the top of memory, and it will stay there until you turn the power off. You can say .X to return to BASIC and load and run BASIC programs, providing they are not too big. IINYMON grabs about 760 bytes of memory, so you lose a little space.

Find A Zero

Once you're back in BASIC, the question arises: how can you invoke TINYMON when desired? Not an easy trick, since memory is more mobile in the VIC than in the PET/CBM. The thing to do is to find a zero value in memory and SYS to that location. If you have a basic (5K) VIC, SYS 4096 is safe. The sure way is to PEEK first and ensure that there's a zero there (location 10 is often zero).

TINYMON 1 must be considered preliminary. It was designed with two major considerations: to use minimum space, and to automatically load into any VIC regardless of the memory fitted. The space consideration is fairly obvious: with 3500-odd bytes available on a small VIC, you want to use up as little as posible. The automatic load feature was tricky to implement; VIC may relocate programs as it loads. What's more, the screen area tends to move around as you add memory.

I scratched my head over the .S (Save) command. If VIC automatically relocates programs during loading, will a SAVEd Machine Language program be safe? As it turns out, VIC has a new tape format available — when a tape is written, it may be defined as "absolute" and will not relocate when it loads.

This seems the best compromise, but it has one drawback – the PET/CBM won't load this type of tape. Perhaps that's a design decision that will need to be revised…

Finding Space In Zero Page

VIC is desperately short of zero page space; machine language programmers will have to cope with the shortage as best they can. I have used the same locations that the big Commodore MLM is expected to use. There's a difference, however, the Commodore job will swap out selected parts of zero page and put them back later; I didn't want to give up the space for that kind of luxury. As a result, you may be annoyed by some locations that are disturbed by TINYMON 1.

For those unfamiliar with the PET/CBM Machine Language Monitor, the commands are:

> **.R** – display 6502 registers;
>
> Users can use screen editing to type over a display and change the registers;
>
> **.M FFFF TTTT** – display memory (from .. to);
>
> Users can use screen editing to type over a display and change memory;
>
> **.X–exit to BASIC;**
>
> It may be wise to type CLR in BASIC after exiting;
>
> **.G AAAA** – GOTO (execute) address;
>
> **.S "PPPP",01,FFFF,TTTT** – Save (program-name, device, from, to);
>
> **.L "PPPP"** – Load (program-name)

There's a delicate tradeoff between features and memory space. There will undoubtedly be other small monitors with a different balance. In any case, I wrote one because I had nothing … and others in the same position will undoubtedly greet TINYMON1 with glad cries.

Program 1: TINYMON1

Enter on a PET/CBM, using the Machine Language Monitor. Do not try to RUN, but follow your entry with the checksum program, Program 2.

First, make the following change:

```
: 0028 01 04 18 08 18 08 18 08
```

Now, enter TINYMON1:

Whew! TINYMON1 for the VIC is now entered. Check it with the following program:

Program 2: A Checking Program

Type the following direct line on the screen of yourPET/CBM:

```
forj = 1024to2071step8:t = 0:fork = jtoj + 7:t = t + peek (k):next:?t;:next
```

You should see the following numbers appear on the screen of your PET. Check them carefully. Each one represents one line of entry, starting at 0400 hexadecimal. If any of these totals is wrong, you've entered the line incorrectly.

The numbers in brackets appearing to the right won't appear on your screen; they are there to help you locate an incorrect line.

When you are satisfied that the program is entered correctly, SAVE it to cassette tape. It may now be loaded into your VIC.

```
462 255 506 399 575 541 592 511 (0400)
769 620 756 780 802 910 886 853
801 784 876 840 835 1383 753 0
1422 589 816 720 584 680 535 576
944 972 1130 845 876 1357 1010 1188 (0500)
1311 852 898 1109 1125 897 809 1021
1340 1078 1005 1212 905 902 770 1239
762 1133 1388 652 659 629 1072 803
748 150 617 413 1020 1030 1057 818 (0600)
944 844 705 831 939 1072 639 1033
943 824 1137 970 929 1149 1395 940
654 840 807 926 706 1146 1015 1146
1175 742 563 645 695 860 1064 1042 (0700)
1235 1202 1355 922 1445 1346 789 1068
1104 1204 975 1306 1339 1169 1168 1210
1340 1204 972 522 460 520 591 942
1010 1079 280 (0800)
```

Copyright ©1981

Jim Butterfield