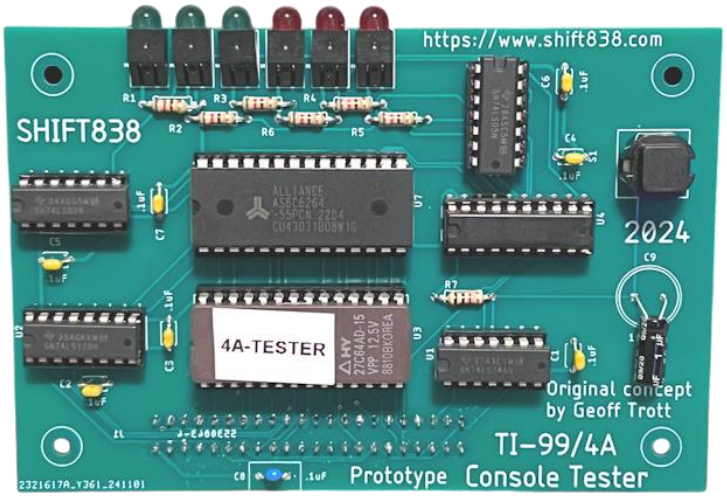


TI-99/4A Console Tester

Chris Schneider
[SHIFT838]
https://www.shift838.com
2024

TI-99/4A Console Tester

Table of Contents

Version History / Revision

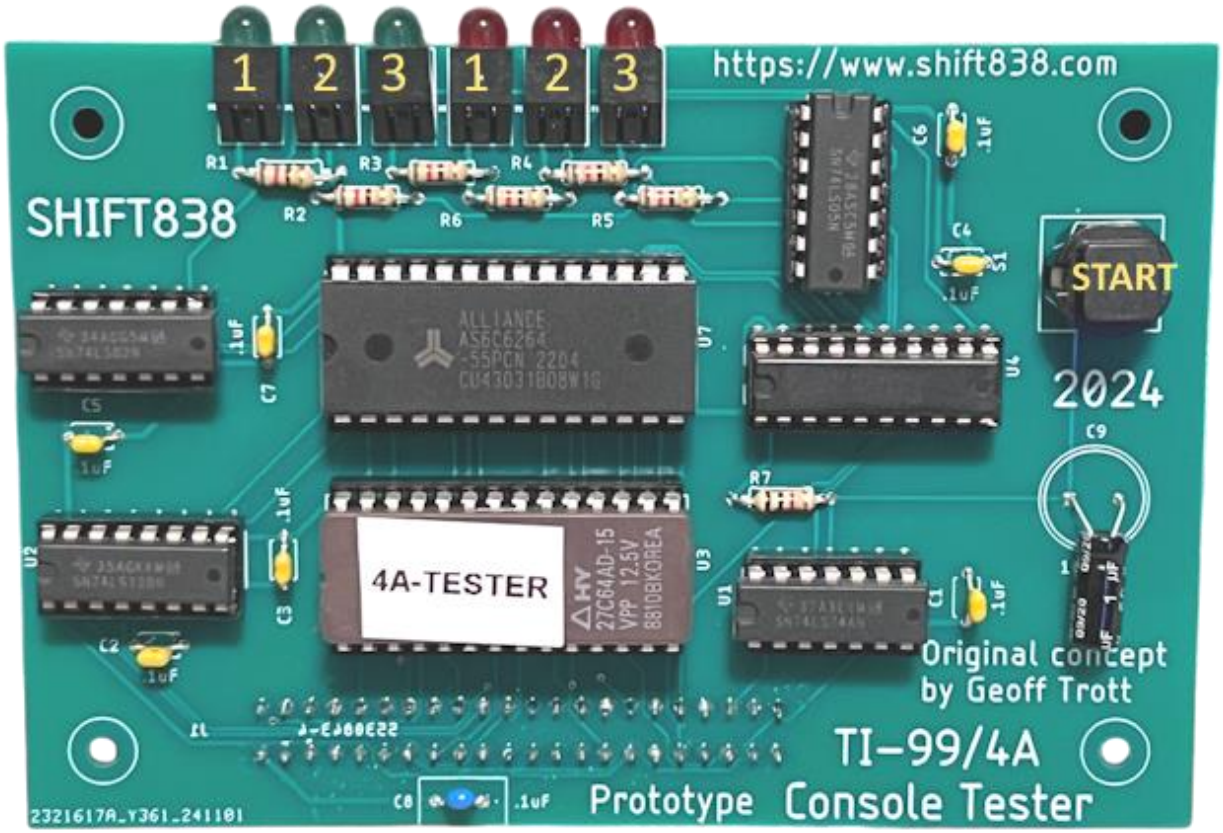
Version	Release Date	Updates
1.0	November 7, 2024	Initial Release
2.0	November 11, 2024	Updated to reference EPROM v2.0

Introduction

This device is a recreation of Geoff Trott's original design with a few corrections found comparing an original board to the released schematic. This board has a new version of the EPROM code to correct some errors and is now Version 2.0

Board Layout

To briefly go over how the board is designed for interpretation. There are 3 green and 3 red LEDs which will tell you if something is right or wrong. They are number 1 through 3 for both green and red LEDs. A single button the board will start the program stored on the EPROM.



Background

The TI99/4A is quite a sophisticated computer which relies on a number of parts for its correct operation. All microprocessors require a program in ROM to be accessible upon power up. This is called the monitor program and for most computers is the only program which runs the computer until something like BASIC is started.

This monitor program is written in assembler and is run directly by the processor. In the TI99/4A there is a monitor program in ROM but its main function is to provide an interpreter for another language called GPL (Graphics Programming Language), which is the language in which the operating system of the computer is written.

This GPL interpreter expects to find the GPL commands in another sort of read only memory called GROM. The contents of GROM can only be read in such a way that a GROM cannot be used to store assembler language which is to be executed directly by the processor from that GROM. Thus, for the correct operation of the computer, both ROM and GROM must be operating correctly.

The 9900 processor requires some RAM for its registers and other system constants. There are 256 bytes of RAM provided in the console called system RAM. For most of the other storage requirements of the computer, VDP RAM is used. This is 16K of dynamic RAM which also is used by the Video Processor chip to contain the information required for the screen display. Once again, this memory cannot be used to store assembler language programs to be executed directly by the processor, whereas the System RAM can be.

If there is a problem with one of the major parts of the console, it is quite difficult to determine which one is at fault because of all the interactions

between them. If the problem is only in VDP RAM the computer will usually start up and produce a recognizable title screen. This is because the VDP RAM is made up of 8 ICs, one for each bit in every byte. If one IC is faulty it only affects 1 bit of each byte, so there are 7 bits correct and a recognizable screen results. If there is a black screen on startup however, the problem could be due to any one of the following being faulty:

- Video Modulator
- Video Processor
- CPU - TMS9900
- System ROM
- System RAM
- GROM

A large number of other components and ICs, for these reasons it was considered necessary to have a way of determining which parts of the computer were working, and even to pinpoint the actual faulty part. The easiest way to do this is by using the computer itself, but if it was not even giving a title screen this would seem to be impossible.

The LOAD Interrupt

The solution to this dilemma lies in using the LOAD interrupt to start another program running in hardware external to the console, but using the processor in the console, to check out the System RAM, VDP RAM and hence VDP processor, System ROM and GROM. This must be done without relying on the screen display, but using it if it is working to give more information than would be otherwise be possible.

What is this LOAD interrupt?

All microprocessors have a RESET input for power up or panic restarts. The TMS9900 has a RESET which is used for this purpose on power up and whenever a cartridge is plugged into the cartridge port. RESET causes the processor to do an interrupt sequence through addresses 0 to 3 and thus to enter the System ROM and produce the title screen. The

TMS9900 has another interrupt input like RESET called LOAD, not normally used in the 99/4A, which causes the processor to do an interrupt sequence through addresses 65532 to 65535, at the top of memory expansion. This LOAD signal is very like the non-maskable interrupt of other processors. LOAD is not very useful normally as one cannot rely on a program and its vectors to be present in these locations of expansion RAM. However, if a diagnostic program is put into EPROM with the vectors at these addresses, and some RAM was made available also, then the LOAD signal could be used to start this program executing regardless of the state of all but the processor in the console. All that would then be necessary would be some indicators to show any errors found, in case the screen display does not work.

The Hardware

The hardware is quite simple, consisting of an EPROM containing the program and the vectors and occupying the last 8K of the expansion memory address space, a RAM chip in the next to last 8K of memory (up to 8K bytes in size), a push button and circuitry for the LOAD signal, address decoding for the EPROM and RAM, and an 8-bit latch which is enabled by a write to any EPROM address. The output of six of the bits of the latch are connected to 3 red LEDs and 3 green LEDs.

There is a 44-way edge connector on the printed circuit board and this plugs into the I/O port on the console, and uses the +5-volt supply from the console for power to the board. If a console was in trouble, any internal memory expansion would need to be removed before this device is attached to ensure no address conflicts.

Operational Procedure

The diagnostic board is plugged into the I/O port of a "dead" console and the power turned on. The START/LOAD button is pressed. This starts the diagnostic program, the screen if visible will go blank, and if the processor

is working the three red LEDs turn on and the three green LEDs are off. The first red LED starts blinking to show that the System RAM is being tested. This test is done by writing a pattern into the entire memory and then reading the entire memory checking for any errors, swapping the bits in the pattern, and doing it again. After doing this 100 times, if there are no errors the first green LED turns on, and the blinking red LED is off. If there are errors, the green LED stays off and the formerly blinking red LED remains on.

A similar test is then run on the VDP RAM 21 times while the middle red LED blinks. Since the VDP RAM is attached to the Video processor, some of its functions are also checked. At the end of this test the middle green LED will come on if there are no errors while the red one will remain on if there are errors.

If all is OK so far, the program sets up the VDP RAM and processor with a green background and black foreground and a character set of 256 characters, using the 128 TI-Writer characters repeated once with a red background color for those codes between 128 and 255.

All the characters appear on the bottom of the screen in reverse numerical order, taking up the last 8 lines of the screen. Then the rest of the screen appears with a heading in the first two lines followed by diagnostic information. There is a message about the System RAM and another about the VDP RAM, which will mirror the state of the first 2 pairs of LEDs. Then the checksum of the System ROMs and the 3 GROMs in the console and 2 GROMs which may be in a cartridge in the cartridge port are calculated and displayed as they are calculated. They are displayed using the full character set and so the actual value can be determined if required. The program loops around calculating the checksums indefinitely, with the third red LED blinking as it does so. Interrupts are enabled for one instruction at the end of the loop. If the System ROM is faulty the diagnostic may not loop, as the interrupt service routine is in System ROM.

The Checksums and Error Codes

The checksums are calculated by adding the bytes of whichever ROM is being checked. In the case of the System ROMs there are two of them, one for the High or even address byte, and the other for the Low or odd address byte. The checksum is calculated for each of these ROMs and displayed side by side, using 3 characters for each. For the GROMs the bytes of each are added together to produce the checksum. Each checksum requires 3 bytes and is displayed as 3 characters using the character set displayed on the bottom of the screen. The values of the checksums of some ROMs and GROMs are:

- System ROM High byte >048181
- System ROM Low byte >05FC8C
- System GROM
    - 0 >075574 (1981) >0731D7 (V2.2)
- System GROM 1 >0911399 (BASIC)
- System GROM 2 >089ADC (BASIC)

The codes for the checksums and RAM errors can be worked out from the characters on the bottom of the screen. if a character has a green background, its hexadecimal value is less than >80. If it has a red background, its hexadecimal value is greater than >80. If it is a recognizable character, its value can be determined using the ASCII code, or by counting characters on the screen. For example, the characters in the last column starting from the top would have the values >E0, >C0, >A0, >80, >60, >40, >20, >00. The character "U" with a green background would have the value of >55, while with a red background it would have the value >D5.

Interpretation of Results

First, if there are any errors reported it is best to press the START/LOAD button again to verify as known issues sometimes cause a false reading.

If the System RAM has errors, then two error code characters will appear on the screen alongside the message. If these characters are decoded into hexadecimal and then into binary, they show in which of the 16 bits errors have been found. Since this RAM uses two chips, one for the most significant 8 bits, and the other for the least significant 8 bits, the chip which should be changed can be determined. The most significant bits are in the left character and a 1 indicates an error, while a 0 indicates no error in the bit.

If the VDP RAM has errors, a single character is output. This can be decoded as before to determine which bit of the memory has errors. For this memory, each bit is contained in a single chip so the code allows the chip in error to be determined and replaced. The checksums given are the only ones found so far, but there have been other versions of things like system ROMs, or so we believe. The most frequent problem encountered is the failure of a system ROM, the low byte ROM. A guide could be that if one of a pair of checksums is as expected, but the other is not, then there is probably an error.

If there are errors in VDP RAM and GROMs, then check the -5-volt supply. If the tester does not start flashing when activated, it may be an error in the processor, power supply, address buffers between the processor and the I/O port or the data buffers in the 8-to-16-line interface area. Looking at various signals around these areas with an oscilloscope is the only way to determine which of these is the problem.

Prior to this development of this tester, troublesome consoles could take many hours to diagnose the fault.

Known Issues

This occurs both on the original and on the recreated board.

On rare occasions after the tester has been plugged into the side of the TI-99/4A and powered up the tester will start the program automatically without pressing the START button. This occasionally happens if the console was powered on recently and not all power has been drained. Ensuring the console has been powered off for a minute or so usually resolves that issue. When powering up the console all the LED lights should be off, they may blink for a microsecond when powered up, but that is normal.

Resolved Issues in Version 2.0

- Version 1 does not read the VDP status register to reset the VDP address latch
- Version 1 did not set the VDP into 16KB mode before testing VRAM (this could report working memory when it's bad)
- Version 1 does not do a dummy GROM read to reset the GROM address latch

The above has been addressed in version 2.0 of the EPROM code.

Recognitions

- This unit was originally designed by Geoff Trott and it is an amazing little device that runs and completely separate program to test various components in the console. This proved to be very helpful when it was originally released and kudos to Geoff for dreaming this up!
- Thanks to Dan Eicher for loaning me a real unit to reverse engineer as there were a few errors on the released schematic. This allowed me to get the 2nd prototype working with no issues. Thanks Dan, you are a great asset to the TI community.