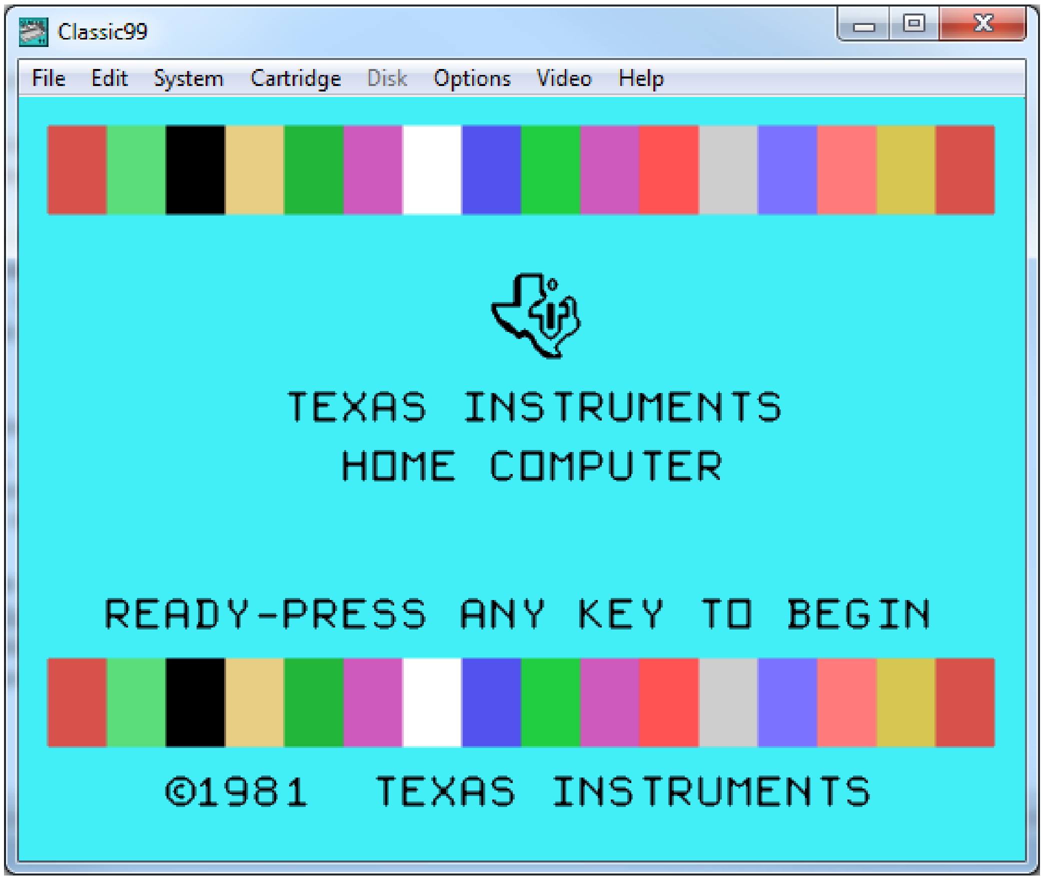
Classic99


Classic99
File Edit System Cartridge Disk Options Video Help
TEXAS INSTRUMENTS
HOME COMPUTER
READY-PRESS ANY KEY TO BEGIN
©1981 TEXAS INSTRUMENTS

A TI-99/4A and TI-99/4 Home Computer Emulator
for Windows

by M.Brent/Tursi
http://www.harmlesslion.com/software/classic99

last update, emulator version 399.061, 10 July 2022

1. Copyrights

The following programs are embedded within Classic99 or distributed along with it and are distributed under license from Texas Instruments (TI).

TI makes no warranty with respect to the programs and is under no obligation to provide any support or assistance with respect to the programs. TI is under no obligation to provide upgrades to the programs. No liability is accepted on the part of Texas Instruments or the author with respect to use, copying or distribution of the programs.

System ROMs and GROMs: TI-99/4	Extended BASIC	Car Wars	TI Invaders
System ROMs and GROMs: TI-99/4A	Home Finance	Chisholm Trail	Tombstone City
System ROMs and GROMs: TI-99/4A v2.2	Mini Memory	Football	Tunnels of Doom (with Pennies, Penny and Quest disk files)
P-Code ROMs, GROMs and disks	Terminal Emulator 2	Hustle	Video Chess
Demonstration	TI Logo][	Hunt the Wumpus	Line by Line Assembler for MiniMemory
Diagnostics	Alpiner	Mind Challengers	Lines demo for MiniMemory
Editor/Assembler (With Editor and Assembler disk files)	A-Maze-Ing	Munch Man	
	BlackJack & Poker	Parsec	

The following programs are embedded within Classic99 or distributed along with it and are owned by Mike Brent. No warranty or liability is offered with these programs.

Super Space Acer	EPSGMod Sample	Space Fighter	XB Demo
CarDemo	Julius Demo	Stranger	

The following programs and documentation are embedded within Classic99 or distributed along with it and are distributed with permission from Richard Lynn Gilbertson. No warranty or liability is offered with these programs.

RXB/REA 2022 package

The following software and documentation is included with Classicc99, and are distributed with permission from Harry Wilhelm. No warranty or liability is offered with these programs.

The Missing Link 2.0	Playground	JUWEL	XB 2.9 GEM

The following program is embedded within Classic99 or distributed along with it and is distributed with permission from Mark Wills. No warranty or liability is offered with these programs.

TurboForth 1.2.1

The following software and documentation is embedded within Classic99 or distributed along with it, and is distributed with permission from Lee Stewart. No warranty or liability is offered with these programs.

fbForth 2.0:13	fbForth Manual and Addendum	fbBlocks	fbFont

The following program is embedded within Classic99 or distributed along with it and is a derived work by Mike Brent of art and music created by Capcom. No warranty or liability is offered with these programs.

MegaMan 2 Music

The following program is included with Classic99 and is owned by Barry Boone. It is believed to be freely distributable, and no warranty or liability is offered with this program.

Archiver 3.03

The following program is included with Classic99 and are owned by Asgard and/or the SW99ers group. It is believed to be freely distributable, and no warranty or liability is offered with this program.

SAMS System Boot Version 2.0

The following program is included with Classic99, and is owned by Shawn Baron, B. Carmany and B. Harrison. It is believed to be freely distributable, and no warranty or liability is offered with this program.

SAMS Memory Tester v4.0

The following program is included with Classic99 and is owned by Jon Dyer and Joe Delekto. It is included with permission from Joe Delekto and no warranty or liability is offered with this program.

TI-NOPOLY Version 1.0

The following program is included with Classic99 and is owned by DataBioTics Ltd. It is included with permission of Edgar Dohmann and no warranty or liability is offered with this program.

TI Workshop

The following software is included in Classic99 and is owned by the Snes9x team. It is believed to be freely usable so long as credit is given, which this notice is doing.

2xsaiwin.cpp

The following software is included in Classic99 and is owned by John Butler. It is believed to be freely usable so long as credit is given, which this notice is doing.

9900dism.cpp

The following software is included with Classic99 and is a derived work based on code owned by Frank Palazzolo, Raphael Nabet, Jarek Burczynski, Aaron Giles, Jonathan Gevaryahu, Couriersud and Michael Zapf. It falls under the BSD-3-Clause license, which is described below.

SpeechDLL

The following software is included with Classic99 and is a derived work based on code owned by Maxim Stepin. It is believed freely distributable with its license.

Hq4x.dll

The following software is included with Classic99 and is a derived work based on code owned by Shay Green. It is believed freely distributable with its license.

Filter.dll

The following image is included in Classic99 with permission from Ron Reuter:

TI-99/4A Keyboard Map

The following software is included in Classic99 with permission from Joe Morris

Fred (aka Fred's Tower)

The following software is included in Classic99 with permission from Scott Adams, (Visit http://www.clopas.net, and check out his Adventureland XL)

Adventure (Cart)
Adventure
Count
Funhouse
Ghosttown
Mission
Odyssey
Pirate
Pyramid
Savage1
Savage2
Sorcerer
Voodoo
Voyage

1.1 Licenses

See above Copyrights section for a description of the licensing of each individual add-in component. For portions of the code not listed, all rights are reserved by Tursi aka Mike Brent, the developer of this code. Redistribution and use of code not explicitly licensed in another way is prohibited without written permission from Mike. If you modify the software, you are not permitted to distribute the modified software without permission. That means just ask first, buddy! 😉

So far as using the emulator goes, you are free to use, copy, display and exhibit the software, subject to the condition that you accept all responsibility for that use, display or exhibition. This software, like most software, is provided AS-IS and no express or implied warranty is made as to its reliability or suitability for any specific purpose. In short, I hope it’s useful, but if it’s not, that’s on you, even if someone warned me that some condition was possible. If the use of this software carries risk of damage, injury, loss or any other liability and you are unwilling to accept that risk, then you are not authorized to use this software.

The necessary texts for other licenses are copied below and apply only to the code that references them above.

BSD-3-Clause

2. Background

Welcome to Classic99! This program attempts to reproduce, more or less accurately, the operation of the TI-99/4A Home Computer. I created it because the TI was my favorite machine, and I wanted to be able to develop for it on my PC.

The TI-99/4A was not a calculator, as many people think when they hear the numbers. Indeed, it was a full-fledged home computer. At the time, the term “home computer” was slightly differentiated from “personal computer”. Usually the “home” computer was a less expensive machine, with less of a business aspect. Home computers usually connected to a television set rather than a dedicated monitor, as well.

Illustration 1: Not a calculator...

TI aggressively targeted education, and in part due to this, the TI-99/4A made inroads into elementary schools across the United States, which contributed to a high estimation of 35% of the US home computer market, with roughly 2.8 million consoles sold. TI competed with Atari, Commodore, Sinclair, and many others for this market.

TI first entered the home computer market in 1979 with the TI-99/4. It was a powerhouse of a home computer for the time, with 16k of RAM, a 16-bit CPU running at 3MHz, 15-color video with 32-sprites onscreen via TI's own TMS9918 video chip, 4-channel sound, BASIC in ROM; and the ability to expand the system with plug-in cartridge-based software, as well as peripherals for the side expansion port. One of the promised (and released) peripherals was the impressive Speech Synthesis unit, which brought computer speech home at a low price. The keyboard was a somewhat less impressive collection of plastic buttons, often called a 'chiclet' keyboard (as the buttons look like the small candy-coated gum). The machine did not sell well, or few were manufactured, and it was extremely expensive, selling for over $1000. Today it is somewhat rare, and most are slightly different, demonstrating early changes in the machine's target market and manufacture.

In 1981 TI released their upgraded machine, the TI-99/4A. It was a minor upgrade in version number, but a pleasant upgrade to the machine. Most notably the build was made more consistent, the keyboard was replaced with a real 48-key full-stroke keyboard, and the video chip was upgraded to the TMS9918A. The price was reduced to just over $500 for the console. The 'A' added only a single new internal feature, a bitmapped graphics mode, with no additional RAM or other upgrades needed. This machine was better priced and more comfortable to use than the 99/4, and began to catch on.

Peripherals became more available as well. Cartridges like Extended BASIC and Editor/Assembler allowed developers to create their own software, although TI was criticized for presenting a 'closed' business model. Games like Parsec and MunchMan were popular in their own right. The Speech Synthesizer landed on nearly every desk, while the power users invested in the Peripheral Expansion Box, which provided slots for expansion cards. Prior to this device, users were expected to “chain” devices from the expansion port on the right side of the machine, a concept many quickly found unreasonable. With the expansion box most users also invested in the 32k RAM expansion, floppy disk controller, and RS232/PIO (printer) card.

TI soon found themselves in a battle with the other home computer manufacturers, notably Commodore and Atari, whose machines were much cheaper. With Commodore's release of the vastly popular C64, it became very hard for the expensive 99/4A to compete. TI offered large rebates, rewards like free speech synthesizers, and eventually lowered the console's price to $150, reportedly selling for a loss, to keep them moving off shelves.

To slow the loss, TI developed the “QI” version of the 99/4A, quoted as standing for “Quality Improved”. This was a redesigned motherboard and cheaper, beige plastic case, designed to save costs. The QI machines quickly became notorious for another change, however, TI changed the operating system in order to prevent third party cartridges, such as those from AtariSoft, from booting. The QIs were not on the market for long, and because stock was matched into various cases, not all beige machines are QI. The best way to tell, if you can power it on, is to look for the copyright “1983” and “V2.2” on the title screen.

TI also released a 'baby brother' of sorts, the TI-99/2 was a small portable machine that ran TI BASIC.

In order to take back the market from the C64, seen as superior and vastly more popular than anything else on the market, TI engineers designed the TI-99/8. This powerhouse came with 64k RAM, upgradable to 15MB, both BASIC and Pascal languages built-in, built-in speech synthesizer, a 10MHz upgrade to the 99/4's processor, a 'hexbus' peripheral interface which was similar in concept to today's USB, and backwards compatibility with the original machine. In 1983 these specifications were almost unheard of. Unfortunately, it was not to be. In October 1983, citing losses of $330 million in a single quarter, TI management pulled the plug on the entire home computer division and shut down all projects current and future. Approximately 150 TI-99/8 prototypes were built, and today they are valuable collectors' items.

User groups continued for years after the machine was discontinued, in some cases 35 years later they are still continuing. Over the years many third party accessories have been created, including memory expansions, RAMdisks, hard drive controllers, keyboard replacements, and much more.

Classic99 focuses on the TI-99/4 machines, and is capable of emulating all three of them.

Illustration 2: TI-99/4

Illustration 3: TI-99/4A

Illustration 4: TI-99/4A 2.2

Classic99 itself was started in 1994 as Ami99, a TI emulator for the Commodore Amiga 2000. Although it functioned, it never reached a releasable stage, and some years later, was ported to the PC running under DOS. Later I ported to Windows, and later still I renamed it to Classic99, as most of the websites were still calling it an Amiga emulator.

Classic99 is written in C and C++ under Microsoft Visual C++, currently with Visual Studio 2017. It uses DirectX 8 interfaces and is capable of running on Windows 2000 through and including Windows 10. There are also reports that it works under Wine and Parallels, but these are not supported configurations.

3. Support

Classic99 emulates the following:

TI-99/4 – including lack of bitmap mode and different keyboard layout. No special hardware or tricks are emulated on this machine mode.

BUG: The 99/4 does not support the emulated keyboard. You may need to find the original key locations to use this. As an assist, there are only number and letter keys, plus shift.

TI-99/4A and TI-99/4A V 2.2 – as the hardware for these machines is functionally identical, Classic99 treats them the same. The ROMs differ and both ROM sets are available.

All console hardware, including keyboard, 9900 CPU with timing, 9901 interface, 9918A video chip (and known tricks/undocumented features except sprite ghosting), 9919 sound chip, and system 16 bit/8 bit bus multiplexer (for cycle counting only).

The VDP can run at both 50hz and 60hz. Limited F18A support is implemented including the 9938 compatible 80-column mode, palettes, and the GPU.

BUG: F18A support is preliminary and very incomplete. It should not yet be relied on.

BUG: 80 column mode does not support screenshots, AVI Recording, video filters, or any blit mode except GDI. It will automatically adjust the video mode, but screenshots and AVI images will be corrupt.

BUG: 80 column mode is independent of the F18A enable, it shouldn’t be.

32k memory expansion is emulated, as well as the 8k 'supercart', with proper 8-bit bus timing.

AMS up to 32MB – this is a third party memory expansion card.

Disk Controller – through a custom disk DSR, the file system interacts with the file system on your PC, as well as with floppy disk images. Floppy disk images are full read and write and support both standard (90k, 180k, 360k) and non-standard (400k) sizes.

Cartridges – both ROM and GROM cartridges are supported. 'Super Space' bank switching cartridges up to 64k, and '378/379' (aka non inverted/inverted) bank switching cartridges up to 32768k are supported. MBX Bank switching is supported.

P-Code card – a UCSD Pascal operating system.

Speech Synthesizer – implemented with the 5110 core. The speech is usually recognizable.

PS/2 keyboard interface – provides a natural keyboard layout to both 99/4A machines. This is based on my own (Mike Brent)'s PS/2 adapter code for AVR.

Two joysticks (or “Wired Remote Controllers”) are supported, either through the keyboard or PC joysticks. Keyboard joystick emulation is shared with the emulated keyboard.

There are also several emulator-only features in Classic99:

Paste Text – by selecting “Edit->Paste”, Classic99 will “type” any text data that you have copied to the Windows clipboard. During paste the emulation automatically goes into *CPU Overdrive* mode. A ‘Paste XB’ mode allows for longer lines to be pasted into Extended BASIC and also trims unneeded whitespace automatically. The clipboard is now Unicode-aware and should have better IE compatibility.

Ability to disable VDP layers including backdrop, sprites, and blanking, as well as to disable the 4-sprites-per-line limitation (flickering).

Video recording, and capturing of still frames both filtered and original. No audio is included.

High Quality Display – Classic99 includes numerous filters to improve the quality of the image on modern high resolution displays.

'TV' mode – Classic99 also includes a filter that simulates the artifacts and blurring of an original television.

‘CPU Overdrive’ mode – runs the 9900 CPU very quickly while preserving the timing of the VDP and VDP interrupts.

Debugger – Classic99 has an integrated debugger, as well as integrated support for “Bug99”, an external debugger by Thierry Nouspikel.

Heatmap – shows where in the memory map the system is accessing at any given time.

Direct Text File Access – Although TI files normally are stored in a native format, Classic99 can read and write Windows text files as though they were native files, allowing easier interface to the host system.

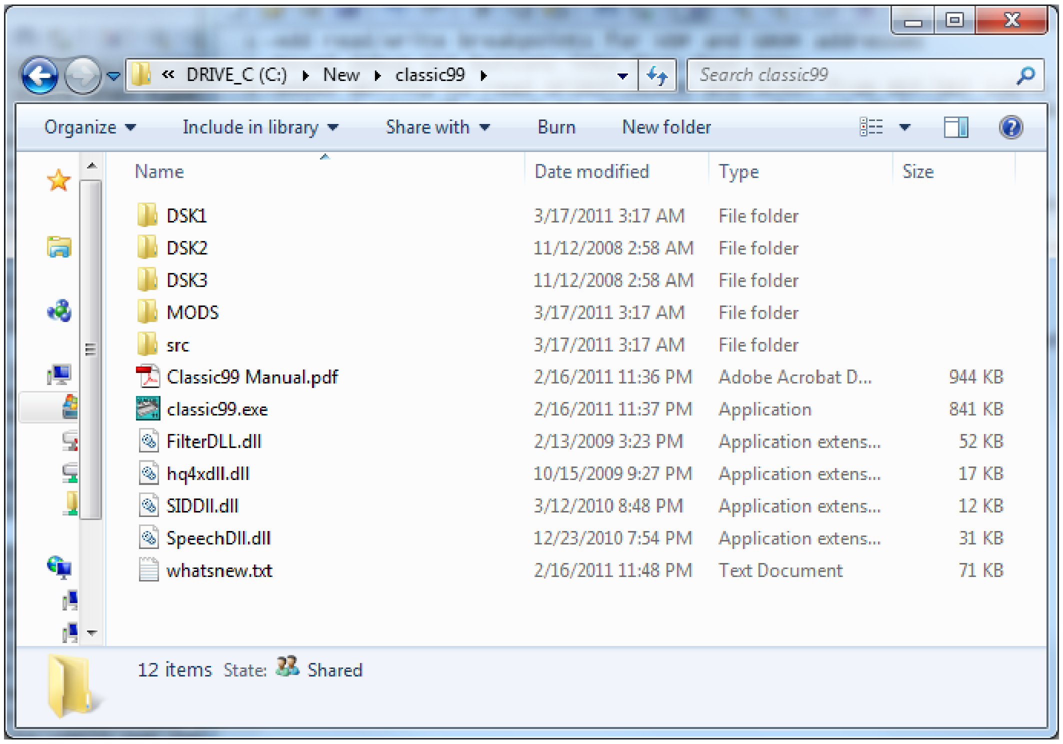
Clipboard file access – Classic99 can also use the Clipboard as a text-only file device, both reading and writing data.

Mouse Based Menu select – double click the display to “type” the character under the mouse for menus.

Ability to enable banks of GROM for writing (turn them into GRAM).

4. Setup

Unzip the archive with folders - in theory, you've already done that. The folder layout should look something like this:



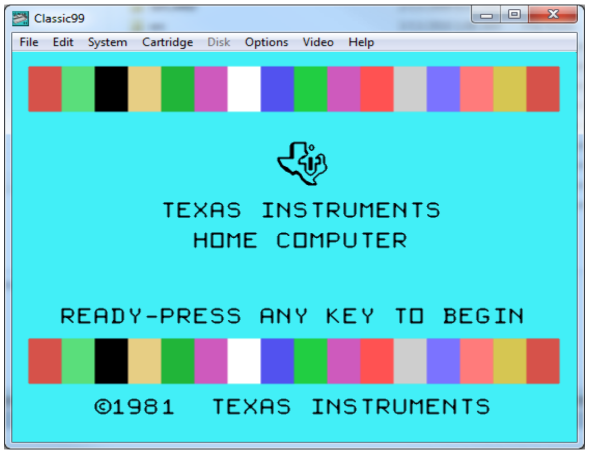
Name	Date modified	Type	Size
DSK1	3/17/2011 3:17 AM	File folder	
DSK2	11/12/2008 2:58 AM	File folder	
DSK3	11/12/2008 2:58 AM	File folder	
MODS	3/17/2011 3:17 AM	File folder	
src	3/17/2011 3:17 AM	File folder	
Classic99 Manual.pdf	2/16/2011 11:36 PM	Adobe Acrobat D...	944 KB
classic99.exe	2/16/2011 11:37 PM	Application	841 KB
FilterDLL.dll	2/13/2009 3:23 PM	Application extens...	52 KB
hq4xdll.dll	10/15/2009 9:27 PM	Application extens...	17 KB
SIDDll.dll	3/12/2010 8:48 PM	Application extens...	12 KB
SpeechDll.dll	12/23/2010 7:54 PM	Application extens...	31 KB
whatsnew.txt	2/16/2011 11:48 PM	Text Document	71 KB

12 items State: Shared

To start the emulator, double-click the Classic99.exe.

No installer is available in the current version of the program. You may place the program anywhere that you have access to read and write folder data. By default a new installation will be configured with default settings, and have access to the three disk folders shown.

On startup, you will be presented with the TI master title page:



```
TEXAS INSTRUMENTS
HOME COMPUTER

READY-PRESS ANY KEY TO BEGIN

©1981   TEXAS INSTRUMENTS
```

5. Using the TI

It is not the intent of this section to teach you how to use a TI Home Computer, but rather to summarize what you have available to you in Classic99. If you want help loading software, refer to the 'Loading Files' or 'Files included on DSK1' sections of this document.

By default, the machine starts up as an NTSC model TI-99/4A running at 3MHz, using the 2xSAI filter to double the screen resolution. Three disk devices are available; DSK1, DSK2, and DSK3. These correspond to the like-named folders from the ZIP archive. No cartridge is selected, speech is enabled, AMS is enabled at 1MB, and video and audio are set to defaults. Finally, the keyboard slow-down hack is enabled, PS/2 keyboard mode is enabled, joystick 1 is set to the keyboard, and joystick 2 is set to the PC's joystick 1, if it has one.

5.1 Keyboard

By default, the keyboard runs in PS/2 mode (except for the 99/4), which is a port of the physical PS/2 adapter created by Mike Brent for the machine (http://harmlesslion.com/software/adapter). There are a few special tweaks for Classic99. Most keys should work pretty much like you would expect, including Num Lock and the numeric keypad.

Caps Lock works like you would expect, but it is *deliberately* reversed. This is so you don't have to switch back and forth between PC and TI. Most TI applications expect uppercase typing, while the PC normally defaults to lowercase. This option may be disabled in the Options menu.

The function (F) keys map to FCTN+number, unless the debugger window is open. When the debugger window is open, the function keys will control debug functions of the emulator (F1-F12). All F keys stop being passed to the emulator. See the Debugger section for more information.

The arrow keys normally return FCTN+E/S/D/X, except if joystick emulation is set to keyboard, and in use. In that case, the emulator multiplexes their functionality. See the joystick notes below.

Should the PS2 emulation cause any problems, exit Classic99, and edit Classic99.ini. There is a setting ‘PS2Keyboard’ - set it to equal 0 and the original TI layout will be used. See the Classic99.INI section for details.

Many of the control keys map to FCTN keys as well, reflecting the key uses from the Editor/Assembler template.

PC Key	**TI Keys**
Up Arrow	FCTN-E
Down Arrow	FCTN-X
Left Arrow	FCTN-S
Right Arrow	FCTN-D
Tab	FCTN-7
F1 through F9	FCTN-1 through FCTN 9
F10	FCTN-0
Insert	FCTN-2
Delete	FCTN-1
Page down	FCTN-4
Page up	FCTN-6
Esc	FCTN-9

5.2 Joystick

The TI supports two joysticks, or “wired remote controllers”. In Classic99, they may be configured as any combination of keyboard, or the first and second PC joysticks (as per the Gaming Controller Control Panel).

If using real joysticks, Classic99 will respond to the first four buttons – any will respond as the TI's sole button. The analog X and Y axes are automatically converted to digital responses for the console.

In keyboard mode, the joystick uses the arrow keys for movement, and the Tab key for fire. These keys are normally mapped to keyboard functions, however, if Classic99 detects that the program is scanning the joystick, and the joystick is configured to the keyboard, then these keys (and Tab) respond to the joystick request only. After about 3 seconds if the joystick is not scanned again, they return to being keyboard keys.

5.3 Disk

Classic99 supports up to 10 disk directories. These are configured under the ‘Disk’ menu option. By default as it ships, DSK1, DSK2 and DSK3 are configured as like-named folders in the Classic99 folder. Supported disk indexes are from 0-9. See the configure section for details on extending this.

Although the Classic99 disk system does not require VDP buffers like the TI Controller does, it **does** reserve the same amount of memory and populate the DSR buffer header in VDP memory like the TI disk controller. For the purpose of BASIC and Extended BASIC, you can use CALL FILES(x) followed by NEW to adjust this pointer. Unlike the true disk system, you can use CALL FILES(0) to disable all VDP buffer space – in this situation you will have the same amount of free VDP RAM as on a cassette system.

Classic99 supports both FIAD (Files In A Directory) and DSK images (DOAD or Sector dump, as well as PC99 or Track Dump images). The extension of the file is not important but the browsers expect the extension to be DSK.

5.3.1 FIAD

Using the FIAD method of file storage, each TI file is a separate file on your PC hard drive (or any other file device). Classic99 can be configured to access any file that your system normally can access. By default it can read both TIFILES and V9T9 files, as well as Windows text files if they have the appropriate extension (as DF80 or DV80). It can also import non-TI files without adding a header, automatically using DF128 format, which is the same format that would have been used if you had downloaded the file to the TI via XModem. Configuration options let you disable any of these options if they are problematic.

FIAD-TIFILES
TIFILES format is also known as XMODEM format. This is the native format for a file transferred from a TI to another machine via XMODEM, and includes a 128 byte header which details what kind of file it was, its size, and so forth. Some extensions include the filename, but Classic99 deliberately does not support this, in order to avoid dependencies between the internal filename and the filename on the host disk. This is the native format of Classic99 to maximize interoperability with a real TI-99/4A. Files may have any legal Windows filename, including with extensions, but note that many applications limit you to 10 characters. All legal file types are supported and all standard file access modes are supported.

FIAD-V9T9
V9T9 format was created for the V9T9 emulator by Ed Swartz, and is similar to TIFILES. It includes the original filename inside the file header. Under modern Windows, most (but not all) of the original limitations no longer apply; however, some applications still support this format. Because this format includes the original filename in the header, it must match the file on the disk. The original V9T9 format used extended ASCII characters to match against characters not legal in DOS, but Classic99 does not recognize these. Thus, filenames with non-alphanumeric characters may have trouble loading. To support these files with a minimum of issue, Classic99 recognizes the tilde ('~') as a replacement character for the high ASCII characters of the original format. If you have trouble loading such a file, try renaming it as such. Characters that should be replaced are '?', '/', '>', '<', ':', '"', '*' or '|'. Classic99 will automatically handle the replacement if it creates the file. Note that the backslash (\) is legal in V9T9 files but Classic99 will not replace it, because Classic99 uses the backslash to support subdirectories. Likewise, the colon character (‘:’) is also not remapped. Programs that require these characters in filenames may need to be modified to work. Classic99 considers the V9T9 file format to be deprecated.

FIAD-Windows Text
Windows text files may be accessed as well, so long as the open mode is Display, and the record length is 80 characters. (Additional options allow flexibility on the record length, as well as whether this option is enabled at all – see the Classic99.ini section for details.) Windows text files are recognized by extension, in order to work the extension must be one of: .TXT, .OBJ, or .COB. By default, Classic99 will only **read** Windows text files, but it may also be configured to **write** them.

FIAD-Files Without Headers
By convention, files transferred to the TI without a TIFILES header were stored as DF128. Classic99 will automatically read such files as DF128 if they have no extension. Files written as DF128 will also be raw without a header by default as of 399.055. Note that if text files are configured to be read without extensions (this was a default in old versions), this option won't work. You can correct this in Classic99.ini (see the section on Classic99.ini).

FIAD-Subdirectories
In all FIAD modes, Classic99 supports accessing subdirectories inside the DSK directories by using a standard Windows backslash ('\'). Note that some TI applications may limit the length of a path to as little as 10 characters, but this is still useful in many cases. Subdirectories are not included when listing a disk from the TI.

FIAD-TI Artist+ Files
Classic99 also supports a naming extension for TI Artist+ files. There is a plugin for Photoshop that recognizes TI Artist+ pictures so long as they have the extensions .TIAP and .TIAC. There is also an image converter that uses these extensions. However, TI Artist+ on the real machine uses _P and _C respectively. Therefore, Classic99 can automatically map these – if you enter _C or _P, it will automatically try .TIAC or .TIAP respectively.

FIAD-Disk Sector Access
FIAD does not support true disk access, however, it does emulate some sector level services. The first 128 sectors on the disk are emulated for read only:

0 - This contains the disk header. This will contain the first 10 characters of the path, and show the allocation bitmap as full. The rest of the data will reflect an unprotected DSSD disk, regardless of the data in the folder.

1 – The File Descriptor Records index will have one entry for each file in the folder, up to a maximum of 127 files, and will sequentially reference incrementing sectors for each. If there are more than 127 files in the folder, the additional files are not indexed (but may still be accessed by software).

2-128 – Each sector contains an FDR for one of the files in the folder. This will reflect the file type information, but the cluster map is not filled in. If there are fewer than 127 files, then not all sectors are available. If there are more, the additional files are not listed here (but may still be accessed by software).

FIAD-File Sector Access
Classic99 fully supports reading and writing files via sector access, using the SBRLNK functions.

FIAD-Other SBRLNK Low Level Access
Classic99 does not support Write Sector, Format Disk, Protect File, Unprotect File, or Rename File on files in the file folders.

FIAD-Options
When opening or saving a FIAD, you may specify a file type override by prefixing the filename with the appropriate 2 character prefix – note the period separators. ie: DSK1.?W.WINFILE

?W.	Handle file as Windows text (Display format only, saves as text or reads text without extension)
?T.	Handle file as TIFILES – most useful for forcing a TIFILES format save if you have changed the default setting to V9T9.
?V.	Handle file as V9T9 – most useful for forcing a V9T9 format save, as TIFILES is the default.
?X	niX the file header entirely – no header will be written. Classic99 might not be able to read the file back in, so use with caution! When used with reads, header detection is bypassed and the file is treated as DF128.

5.3.2 Disk Images

Classic99 has nearly complete support for DSK images. Both sector dump images (V9T9/MAME style) and track dump images (PC99) are supported, including support for the reverse sector numbering on side two of a DSSD diskette (as the DOS V9T9 was reported to sometimes write - this should be very rarely needed today).

Disk images are a dump or reproduction of actual TI diskettes, and as such are subject to all of their limitations. Classic99 supports reading and writing disk images using the conventions set up on TI’s disk controller, but will work with the larger disk formats of Corcomp and Myarc as well.

No disk images are configured by default, use the configuration menu to attach a disk image to a particular disk number.

Classic99 will also allow you to create a new, blank diskette - simply specify a new filename when selecting your disk image and you will be prompted.

The following operations are not supported in the disk image emulation:

- Scratch Record (not supported on original TI DSR either)
- Write file sectors (sbrlnk)
- Format disk (sbrlnk)
- Protect file (sbrlnk)
- Unprotect file (sbrlnk)

BUG: PC99 disk images are auto-detected, but this auto-detection cannot be disabled. Writing to a PC99 disk does not update the intra-sector information (but all available data suggests this information was not captured correctly in original PC99 diskettes and is thus unusable anyway…)

5.3.2 CLIP Device

In addition to the DSK devices, Classic99 also supports DISPLAY mode access to the Windows clipboard. The devicename is "CLIP". Full access is available, including read, write, update and append. Note, however, for update and append, the clipboard is read and cached on the open call, then replaced when the file is flushed or closed. Intermediate external changes won't be reflected.

The CLIP device does not support non-text mode access. Also, the opcodes LOAD, SAVE, DELETE, SCRATCH, and STATUS are not supported. No SBRLNK opcodes are supported to the CLIP device.

Normally the CLIP device will append an end of line after each record. If you open it as "CLIP.NOCR" or "CLIP.NOLF" (as per your preference), line endings will not be appended, and you must do so in your own software. You can use CR (13) or LF (10) at will. Both options function identically and only one may be provided.

Note, the CLIP device does not support filenames, so unlike the disk system, there is no need for there to be a period in the device name. Some programs, like Editor/Assembler, will report error 0 if you attempt to load from any device without the period. Appending a period is enough to work around this error, so E/A can load from or print to "CLIP." If a filename is specified, the CLIP device will ignore it.

The CLIP device supports fixed and variable length records up to 254 bytes.

5.3.3 CLOCK device

Classic99 supports read-only access to the host PC’s real-time clock using the CLOCK device. The syntax of the returned data is compatible with the Corcomp clock, and is formatted as a single record, with three comma-separated entries:

d	The first entry is the day of week, from 0-6, with 0 being Sunday
mm/DD/yy	The second entry is the date. Each value is two digits, first month (01-12), then day (01-31), then 2-digit year (00-99)
hh:MM:SS	The third entry is the time. Each value is two digits, first the hour (24-hour clock, 00-23), then the minute (00-59), and the second (00-61). Note that unlike older clocks, modern systems support the concept of leap seconds, thus the second may be larger than 59! It

is not expected that this will break any older software.

The clock is implemented as a DSR only, register access to the clock is not available, nor are the alarm features. The clock may be opened in INPUT or UPDATE mode only (writing is forbidden even in UPDATE mode, however). The following TI BASIC application shows how to read the clock:

```
10 OPEN #1:”CLOCK”
20 INPUT #1:A$,B$,C$
30 PRINT A$:B$:C$
```

5.3.4 Cassette (CS1 and CS2)

Classic99 supports reading files and data from CS1. Classic99 does not support writing to cassette, neither CS1 nor CS2. Furthermore, the CS2 motor control is not supported.

To load a cassette program, you must have a WAV file recorded at 16khz or higher. Select Disk->Load/Rewind Tape. Select the WAV file from the file browser - this will 'rewind' and prepare the file to be played.

Because Classic99 does not provide a "play" button, the emulator will wait for the console to stop, and then start CS1, and then start to playback automatically. This is sufficient for most operations, but you must load the tape before or during the "REWIND TAPE" prompt.

During loading, you will be able to hear the audio. If the audio falls out of sync with the emulation this is not a cause for concern, as the emulated TI is reading the data directly and the audio feedback is just for human consumption. This often happens if you load during CPU Overdrive (which is recommended!)

If an error occurs, you may hear the tape wave file continuing to play, you can stop/eject the tape by selecting 'Disk->Load/Rewind Tape' again, then simply cancelling to abort loading a new tape.

Classic99 will appear to go through the motions if you attempt to save, but no data is recorded anywhere. You will not be able to hear the tones.

6. Using the Emulator

The Classic99 emulator usually keeps itself in the background, allowing you to focus on the experience of using the machine it emulates instead. Most functionality can be accessed via the standard Windows user experience. For instance, the Window may be re-sized by dragging the borders, or moved by dragging the title bar. The application can be exited by clicking the close button. Most other functions are available through the menu bar.

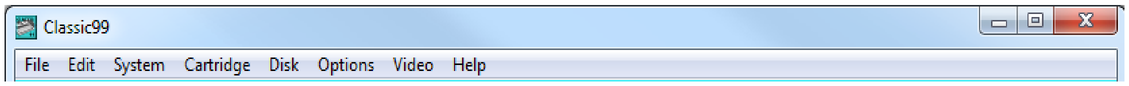


Illustration 5: Classic99 Menu Bar

6.1 File

File → Cold Reset – Erase RAM

Resets the emulator as if it had been powered off, and then on again. Brings all memory and systems to a clean state and restarts. Does not change configuration state or loaded cartridges.

File → Warm Reset – Leave RAM

Resets the emulator as if the reset line was toggled, leaving all memory as is. Note that the TI GROMs erase video RAM on startup.

File → Debug Reset – Scramble RAM

Resets the emulator but fills all RAM with random values. This can be used by programmers to try and find bugs with uninitialized memory. Note that due to bugs in the TI's own ROMs that sometimes the system can not start with this option, and you will have to select it again.

File → Erase UberGROM

If you are using UberGROM emulation, this will erase the current contents of the Flash, EEPROM and RAM portions of the device, providing a blank device to work with. You may need this before trying to load new software on it.

File → Quit

Exits the application. Classic99 will ask for confirmation before exitting.

6.2 Edit

Edit → Paste

If there is text on the Windows clipboard, Classic99 will enter the text as if a user was typing it. Classic99 will do this by intercepting the calls to KSCAN and inserting the appropriate return values, so will work in any application that uses KSCAN. Control-F1 will also start a paste.

Edit → Paste XB

This functions the same as Edit->Paste, except that the data is assumed to be pasting into Extended BASIC. Spaces will be removed from strategic locations, and the XB input buffer will be artificially enlarged during the paste. This should allow most listings to be pasted without long lines being truncated. Never use this function if Extended BASIC is not running, it may crash the application you are running, and corrupt the text you are attempting to paste. It may not work in other variants of XB besides TI's original. No shortcut key will be provided for this due to accidental presses being risky.

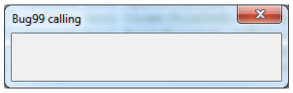
Edit → Debugger

Launches the Classic99 Debugger window. See the Debugger section for more details. Control-Home will also launch this.

Edit → Copy Screen

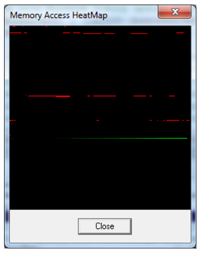
Copies the screen to the clipboard. This function will not work in multicolor or bitmap modes, but it will work in 32-column, 40-column, and 80-column modes. If you are working in BASIC or XB, hold SHIFT to apply the BASIC character offset for correct text. Control-F2 will also trigger this function (and Control-Shift-F2 will trigger with the BASIC offset).

Edit → Bug99 Window



Launches the Bug99 Calling Window for Bug99 to communicate with. This window is not user interactive but must be present for the Bug99 application to communicate with Classic99. Note that you must also load the Bug99 software inside Classic99 to use this tool. See http://www.nouspikel.com/ti99/bug99.htm

Edit → Heatmap



Launches the heatmap window. Part debug tool, part visualization, the heatmap shows you memory accesses inside Classic99. Red shows CPU (RAM and ROM) access, Green shows GROM access, and Blue shows VDP access. Address >0000 is at the top left, and address >FFFF is at the bottom right. Each pixel lights up when the address is hit, and fades out slowly. The update of the display may impact performance on some PCs. Click the Close button to exit.

6.3 System

System → TI-99/4

Selects the TI-99/4 emulation system. This will reset the emulator into the 1979 model of the TI-99/4.

System → TI-99/4A

Selects the TI-99/4A emulation system. This will reset the emulator into the 1981 model of the TI-99/4A.

System → TI-99/4A V2.2

Selects the V2.2 QI TI-99/4A emulation system. This will reset the emulator into the 1983 model of the TI-99/4A.

6.4 Cartridge

Cartridge → Apps → ????

Allows you to select one of many application cartridges included with Classic99. This will cause the emulator to reset and the cartridge to appear on the master title page. Note that TI Workshop (379) will not show up on the V2.2 TI-99/4A due to the Third Party ROM lockout.

Cartridge → Games → ????

Allows you to select one of many game cartridges included with Classic99. This will cause the emulator to reset and the cartridge to appear on the master title page. Note that Super Space Acer will not show up on the V2.2 TI-99/4A due to the Third Party ROM lockout.

Cartridge → User → ????

Allows you to select a cartridge that you have added to the Classic99.ini file. If you have not added any files this list will be empty. You may add custom submenus under this one, as well.

Cartridge → User → Open...

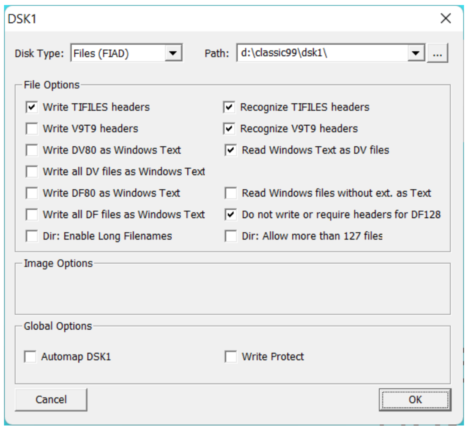
Brings up a file dialog that allows you to select a cartridge image to load. Currently the V9T9 .BIN format is supported – these cartridges are usually found as multiple files with the same filename, except for the last letter, which varies as C, D, or G, and a .BIN extension (ie: xxxC.BIN). Classic99 extends this and recognizes “3.BIN” as a Jon Guidry/379 (Inverted) or Superspace format cartridge ROM. Classic99 will also recognize .C, .D, .G and .3 as extensions, and can parse filenames that include the element “*(Part x of y)*”, automatically updating the part index. Simply selecting one file in the group is enough, Classic99 will automatically check for the other files in the group.

Cartridge → Eject

This will eject the currently loaded cartridge and reset the emulator.

6.5 Disk

Disk contains a list from DSK0 through DSK9, each representing one of the disk drives available to be configured.



DSK1

Disk Type: Files (FIAD)   Path: d:\classic99\dsk1\

File Options

- [✓] Write TIFILES headers
- [ ] Write V9T9 headers
- [ ] Write DV80 as Windows Text
- [ ] Write all DV files as Windows Text
- [ ] Write DF80 as Windows Text
- [ ] Write all DF files as Windows Text
- [ ] Dir: Enable Long Filenames
- [✓] Recognize TIFILES headers
- [✓] Recognize V9T9 headers
- [✓] Read Windows Text as DV files
- [ ] Read Windows files without ext. as Text
- [✓] Do not write or require headers for DF128
- [ ] Dir: Allow more than 127 files

Image Options

Global Options

- [ ] Automap DSK1
- [ ] Write Protect

Cancel   OK

Disk → DSK*x* → *Disk Path* or *Set DSK*x

Loads the disk configuration dialog which has the following options:

*Disk Type:*Select ‘None’ to disable the disk, ‘Files (FIAD)’ to select Files on a Disk format, or ‘Image (DSK)’ to select a disk image file. The appropriate options will be highlighted.

Path: Type the path to the folder or disk image as appropriate or click the ‘…’ button to browse for it. If selecting a FIAD path, select any file in the folder (if the folder is empty, you can invent a filename). For images, if you enter a non-existent file, Classic99 will offer to create it as a new blank 180k disk image.

File Options: Valid only when Files (FIAD) is selected as the disk type.

Write TIFILES headers/Write V9T9 headers – When Classic99 saves a file, it can use either TIFILES or V9T9 format to write. Classic99 prefers TIFILES format and considers V9T9 format deprecated, but you may select this format for compatibility with older tools. Only one of these may be selected at a time.

Write DV80 as Windows Text – Makes Classic99 save Display/Variable 80 files, traditionally used on the TI as text files, as native Windows text files instead. (So long as they have an appropriate extension such as .TXT, Classic99 can read them back again too.)

Write all DV files as Windows Text – When this is set, any Display/Variable file written by Classic99 will be written in Windows text format.

Write DF80 as Windows Text / Write all DF files as Windows Text – same as the previous two options, except these apply to Display/Fixed format files. Note that when reading back Windows text, Classic99 does not worry about whether the Windows file is really fixed or variable length.

Recognize TIFILES headers – when opening a file, attempt to detect a TIFILES header. If this is disabled, Classic99 will not recognize TIFILES headers on this disk.

Recognize V9T9 headers – when opening a file, attempt to detect a V9T9 header. If this is disabled, Classic99 will not recognize V9T9 headers on this disk. It is okay to have both header formats enabled for reading, Classic99 can tell them apart.

Read Windows Text as DV files – if a file with a .TXT or .OBJ extension is opened as Display/Variable, and it is not a TI file, treat it as a Windows text file and read it.

Read Windows Text as DF files – the same as the previous operation, but allow it if the file is opened as Display/Fixed.

Read Windows files without ext. as Text – allow even files without a .TXT or .OBJ extension to be read as text files. This option is deprecated now.

Do not write or require headers for DF128 – traditionally, non-TI files transferred to a TI system were saved as Display/Fixed 128 files. This option enables treating all Windows files not detected as text as DF128 files. In addition, written DF128 files will not have a header added. This option is incompatible with ‘Read Windows files with ext. as Text’.

Image Options: (empty)

Global Options: Valid for all disk types.

AutomapDSK1 – when enabled, all records read from the disk are examined for the exact uppercase string “DSK1”. If found, the index is changed to the index of the disk being accessed (for instance, if reading a file from DSK3, all references to DSK1 are updated). This facility recognizes binary files, as well as Editor/Assembler compressed and uncompressed object files. However, because it works per record, programs which have the DSK1 string split between two records cannot be updated. For programs it works on, which should be most, the upshot is to automatically run a program meant for DSK1 on another drive number.

BUG: Strings split across files or records cannot be detected or updated.

Write Protect – prevents any writes to the disk device. (**Note:** This setting is attached to the DRIVE, not the disk image! Pay attention when changing disk images or paths.)

Disk → DSK*x* → Open DSK*x*

This option simply attempts to open the configured folder or disk image using Windows Explorer. In the case of a FIAD disk, the folder should be opened. For an Image disk (DSK or TIDISK), the associated program should be opened, if there is one.

Disk → Corrupt DSK RAM

This is a debugging option for developers working on code that uses the disk system. It overwrites memory locations used by the TI disk system with >4A bytes, to debug problems with using memory that a real TI system may corrupt. (Classic99’s disk system does not require any TI memory to operate, so new software may use memory that is not actually safe to modify.) Note that this is not perfect, in particular it is known that it may be unsafe to use addresses >8310->831F, but Classic99 cannot corrupt these addresses without breaking software like Editor/Assembler. You may turn this off if you suspect your software needs data that is being overwritten.

Disk → Tape → Load/Rewind Tape

This option cues up a WAV file for cassette support. When it is selected it first ejects any previously loaded WAV file (this can be used to stop a file playing in the background, just Cancel). Next the file is loaded and converted to an internal format, ready to play.

The system has an extra catch not in the hardware - it will wait for the console to turn the tape motor off, and then back on again, and then automatically play out the WAVE file. This normally happens during the "PRESS CASSETTE PLAY" prompt, so you must cue up your file before that. (Before or during the REWIND prompt is sufficient.)

Disk → Tape → Stop Tape

This command will stop any currently playing tape, as if you pressed STOP on the player.

Disk → Tape → Play Tape

This command will cause the tape to start playing, even if a remote transition is not detected, as if you pressed PLAY on the player. Unlike real hardware, this will start playback even if the remote jack is still turned off, and is intended to work around issues with Classic99's handling of the remote.

6.6 Options

Options → Pause When Window Inactive
When this item is checked, selecting a different PC window for focus will cause Classic99 to automatically pause. That is, it will run only when it has focus.

Options → CPU Throttling → Normal
Makes the emulator run at the 'normal' speed, which is approximately the speed a real TI would run.

Options → CPU Throttling → CPU Overdrive
Runs the CPU at a much faster rate – as fast as a simulated 150MHz, but does not speed up the rest of the system. This can overcome issues such as slow performance in a program without interfering with video updates or sprite movement.

Options → CPU Throttling → System Maximum
This speeds up the entire emulator, including video and sound updates. However, it currently causes performance problems and on some systems may actually run slower than normal.

Options → CPU Throttling → CPU Slow
This slows down the CPU to a single clock per frame, meaning it runs at 60Hz when in NTSC mode. All other systems are affected. This can be useful when debugging but is generally too slow to watch updates on the screen.

Options → Ctrl+Alt+= to Reset
When checked, the QUIT key sequence will be modified in the keyboard driver to Ctrl+Alt+Equals, rather than just Alt+Equals. This only works if the PS/2 keyboard is active.

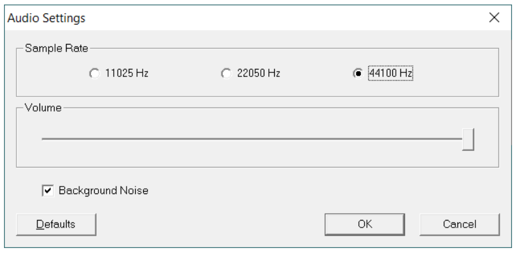
Options → Invert Caps Lock
When checked, the state of the Caps Lock key is read inverted. This is normally on to allow the TI default state to be uppercase without inconveniencing the normally-lowercase PC.

Options → Speech Enabled
When checked, the Speech Synthesizer is emulated. When cleared, the Speech Synthesizer will not be available to the emulator. Speech is provided by an external DLL (speechdll.dll) that may not be available.

Options → Audio...
Loads the Audio configuration dialog, which has the following options:

Sample Rate: Select 11025Hz, 22050Hz, or 44100Hz. Higher rates sound better but use more CPU time.

Volume: Push the slider right for louder sounds, or left for softer sounds.


Audio Settings

Sample Rate: 11025 Hz 22050 Hz [✓ 44100 Hz]

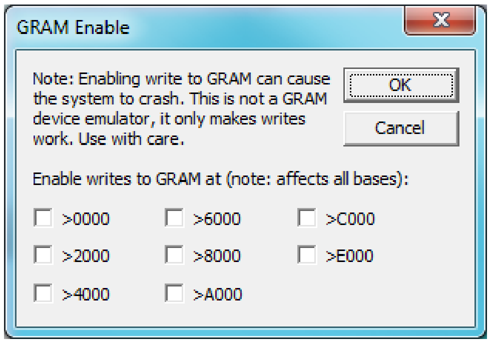
Volume

[✓] Background Noise

Background Noise: Reproduce some of the internal bus noise that was often heard on the audio output of the 4A.

Defaults: Click this to reset settings to the default.

Options → GRAM...


GRAM Enable

Note: Enabling write to GRAM can cause the system to crash. This is not a GRAM device emulator, it only makes writes work. Use with care.

Enable writes to GRAM at (note: affects all bases):

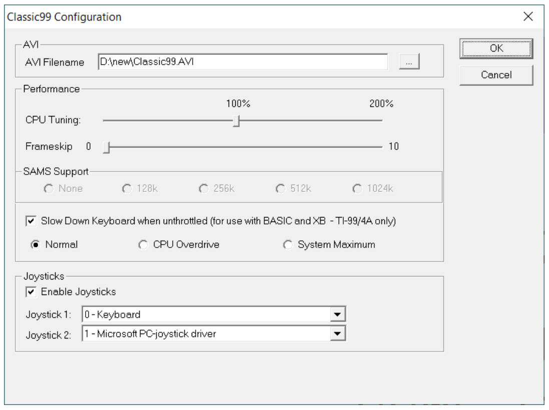
☐ >0000 ☐ >6000 ☐ >C000
☐ >2000 ☐ >8000 ☐ >E000
☐ >4000 ☐ >A000

Loads the GRAM Enable dialog, which allows you to specify which GROM chips are actually GRAM, which permits writing to them. Simply check the chips you are interested in. GRAM is enabled over the entire 8k memory range (note that real GROMs are only 6k).

All GROM bases are affected by this selection.

GRAM in Classic99 is volatile – if you close the emulator, or use File->Reset, the ROMs are reloaded and changes are lost.

Options → Options...


Classic99 Configuration

AVI
AVI Filename: D:\new\Classic99.AVI

Performance
CPU Tuning: 100% (slider to 200%)
Frameskip: 0 (slider 0 to 10)

SAMS Support: None, 128k, 256k, 512k, 1024k

☑ Slow Down Keyboard when unthrottled (for use with BASIC and XB - TI-99/4A only)
[✓ Normal] CPU Overdrive System Maximum

Joysticks
☑ Enable Joysticks
Joystick 1: 0 - Keyboard
Joystick 2: 1 - Microsoft PC-joystick driver

Loads the General configuration dialog, which has the following options:

AVI Filename: Enter a path and filename to save a recorded AVI at, or click the '…' button to browse for a path. The AVI will not be recorded until you select Video->Start Recording Video.

CPU Tuning: Drag the slider left for a slower 'normal' CPU, or right for faster. 100% represents a standard 3MHz CPU.

Frameskip: Set the slider right to skip drawing video frames for performance. This option may have little effect in the current build and may cause audio issues.

SAMS Support: Set the size of the emulated SAMS card. The valid options today are None (no SAMS card installed), or 1024 (1MB).

Slow Down Keyboard when unthrottled: This option manipulates the console operating system to slow down key repeat when the system is configured for CPU Overdrive or System Maximum. Select *Cpu Overdrive* to take effect only in overdrive, or *System Maximum* to take effect only in System Maximum.

Enable Joysticks: When checked, joysticks will be emulated using the settings below. *PC Keyboard* emulates the joysticks using the arrow keys and the Tab key. Otherwise, select the system joystick from the dropdown list. (Close options and re-open if you need to attach the joystick).

Options → Screen Reader → Continuous

If your system supports the Microsoft Speech API, this will enter a mode where Classic99 will attempt to read aloud any new text that appears on the screen – automatically! This setting is saved in the INI.

Options → Screen Reader → Read Once

If your system supports the Microsoft Speech API, this will attempt to read aloud the current text on the screen.

Options → Screen Reader → Stop Talking

If the screen reader is going off on a tangent, reading graphics characters and the like, this will cause it to stop speaking.

6.7 Video

Video → Maintain Aspect Ratio on Window

When checked, this option forces the window aspect to remain constant as the window is sized. It does not affect full screen mode, or if the window is maximized. If unchecked, the window may be distorted into incorrect aspects.

Video → Flicker

When checked, Classic99 will emulate the sprite limitation of 4 sprites displayed per scanline. Additional sprites are not drawn. If cleared, Classic99 ignores this limitation and draws all sprites regardless of position.

Video → Layers → Disable Blanking
When checked, Classic99 will ignore the blanking bit in the VDP, and always draw the display.

Video → Layers → Disable Sprites
When checked, Classic99 will not draw sprites.

Video → Layers → Disable Background
When checked, Classic99 will not draw the background graphics.

Video → 50Hz (60Hz if off)
When checked, Classic99 will run with a PAL refresh rate of 50Hz, instead of the default NTSC 60Hz. This may make software appear to run more slowly.

Video → Show FPS
When checked, the current frames per second being drawn are displayed at the bottom left.

Video → Enable 80 Column Hack
When checked, Classic99 modify the video emulation to allow a very limited subset of the 9938 80-column mode to run. This mode does not support any register settings greater than register 7, supports only 16k of video RAM, and does not support blink or additional hardware. This is intended primarily for TurboForth support.

BUG: When the emulator is in 80 column mode, most video bonuses are disabled. In particular, video recording and screenshots will be corrupted, video filters are disabled, and DIB stretch mode will be used.

Video → Enable F18A GPU
When checked, Classic99 will enable the F18A functionality.

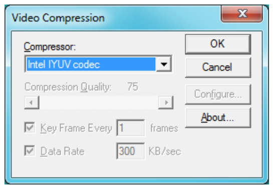
BUG: F18A support is VERY incomplete and is supposed to include the 80 column mode below, but that is currently separate. Most video functions DO NOT work.

Video → Interleave GPU
This should be on by default. When disabled, code executing on the F18A GPU will pause the 9900 CPU. This is primarily intended to make debugging GPU code easier, as it can then be stepped through in the debugger.

Video → Enable 128k Hack
When checked, the VDP will enable 128k of VRAM in a manner compatible with the 9938 VDP.

BUG: Although the RAM is available and functional, there is no real-world equivalent to this configuration. Most software that detects the RAM will assume the enhanced graphics of the 9938 are also available and may fail to operate correctly. There is no workaround for this.

Video → Start Recording Video
When this option is selected, a compression selection dialog is displayed. Select a video compressor to start recording video from Classic99 to the file configured in the Options dialog. No audio is recorded with this option, and the video uses the original unfiltered image.


Video Compression
Compressor: Intel IYUV codec
Compression Quality: 75
☑ Key Frame Every 1 frames
☑ Data Rate 300 KB/sec

Video → Start Recording Video+Audio
This option also starts recording video, but includes audio at the current sampling rate. The audio may not be compressed.

BUG: Audio is not currently working correctly in saved videos.

Video → Stop Recording Video
If video is being recorded to an AVI file, this option will stop it.

Video → Screenshot (Basic)
Allows you to save a screenshot using the raw, unfiltered image.

Video → Screenshot (Filtered)
Allows you to save a screenshot using the currently displayed image.

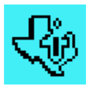
Video → Change Size → 1x / 2x / 3x / 4x
Changes the size of the window to be exactly 1 times, 2 times, 3 times, or 4 times the actual resolution of the current filtered mode image. 1x will produce 1:1 pixels, 2x will produce 2:1 pixels, etc.

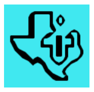
Video → Stretch Mode → None
Sets the video stretch mode to none. Classic99 will use GDI to draw the image on the screen, and will not scale it to fit the window. This is the simplest mode if problems are occurring.

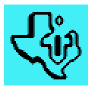
Video → Stretch Mode → DIB
Sets the video stretch mode to DIB. Classic99 will use GDI Device Independent Bitmaps to stretch the image to fit the Window. This is usually slower but on some older cards it can be faster or more stable than DirectX.

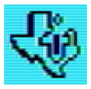
Video → Stretch Mode → DX
Sets the video stretch mode to DirectX. Classic99 will use the card hardware to stretch the image to fit the window, if available. This is usually the fastest windowed mode.

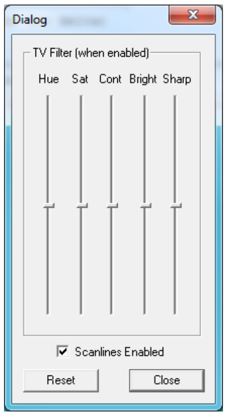
Video → Stretch Mode → DX Fullscreen
Switches the display to full screen mode using DirectX at the desktop resolution. To exit full screen mode, press Alt+Enter, or right-click the mouse.

Video → Filter Mode → None

Displays the image without filtering. It is sized to match a true TI display, then stretched to fill the window (if the stretch mode is set). The effective resolution without borders is 256x192.

Video → Filter Mode → hq4x

This implements the hq4x filter by Maxim Stepin to scale the image up four times each way and smooth pixels and lines. The effective resolution without borders is 1024x768. This filter is provided by a DLL and may not be available.

Video → Filter Mode → 2xSAI / Super2xSAI / SuperEagle

These options implement the 2xSAI filters by Derek Liauw Kie Fa. Each uses a slightly different technique to scale the image up twice each way, and smooth pixels and lines. The effective resolution without borders is 512x384.

Video → Filter Mode → TV Mode

This option implements the NTSC TV Mode filter by Shay Green. It uses several techniques to reproduce artifacts, ghosting, blur, and other effects of NTSC and RF modulation. It also responds to the TV Controls configuration dialog.



✓ Scanlines Enabled

Video → TV Filter Controls

These provide controls analogous to what was available on older NTSC television sets. *Hue* changes the colors displayed. *Sat* adjusts the color saturation. *Cont* adjusts the contrast of the image. *Bright* adjusts the brightness. *Sharp* adjusts the sharpness.

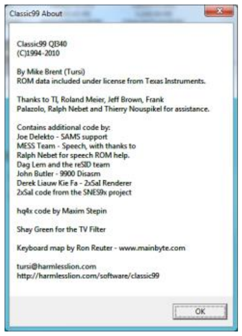
Scanlines Enabled turns the horizontal scanline effect on and off.

Reset restores all the sliders to their default, centered positions.

6.8 Help

Help → About

Reports the version of the emulator, and all the people to thank!



Classic99 QI340
(C)1994-2010

By Mike Brent (Tursi)
ROM data included under license from Texas Instruments.

Thanks to TI, Roland Meier, Jeff Brown, Frank
Palazolo, Ralph Nebet and Thierry Nouspikel for assistance.

Contains additional code by:
Joe Delekto - SAMS support
MESS Team - Speech, with thanks to
Ralph Nebet for speech ROM help.
Dag Lem and the reSID team
John Butler - 9900 Disasm
Derek Liauw Kie Fa - 2xSaI Renderer
2xSaI code from the SNES9x project

hq4x code by Maxim Stepin

Shay Green for the TV Filter

Keyboard map by Ron Reuter - www.mainbyte.com

tursi@harmlesslion.com
http://harmlesslion.com/software/classic99

Help → Keyboard Map

Displays an image of the TI-99/4A keyboard map, showing the FCTN keys.

Help → Known issues with current cartridges

If the currently selected cartridge has any known issues, this will display them.

Help → Open Manual

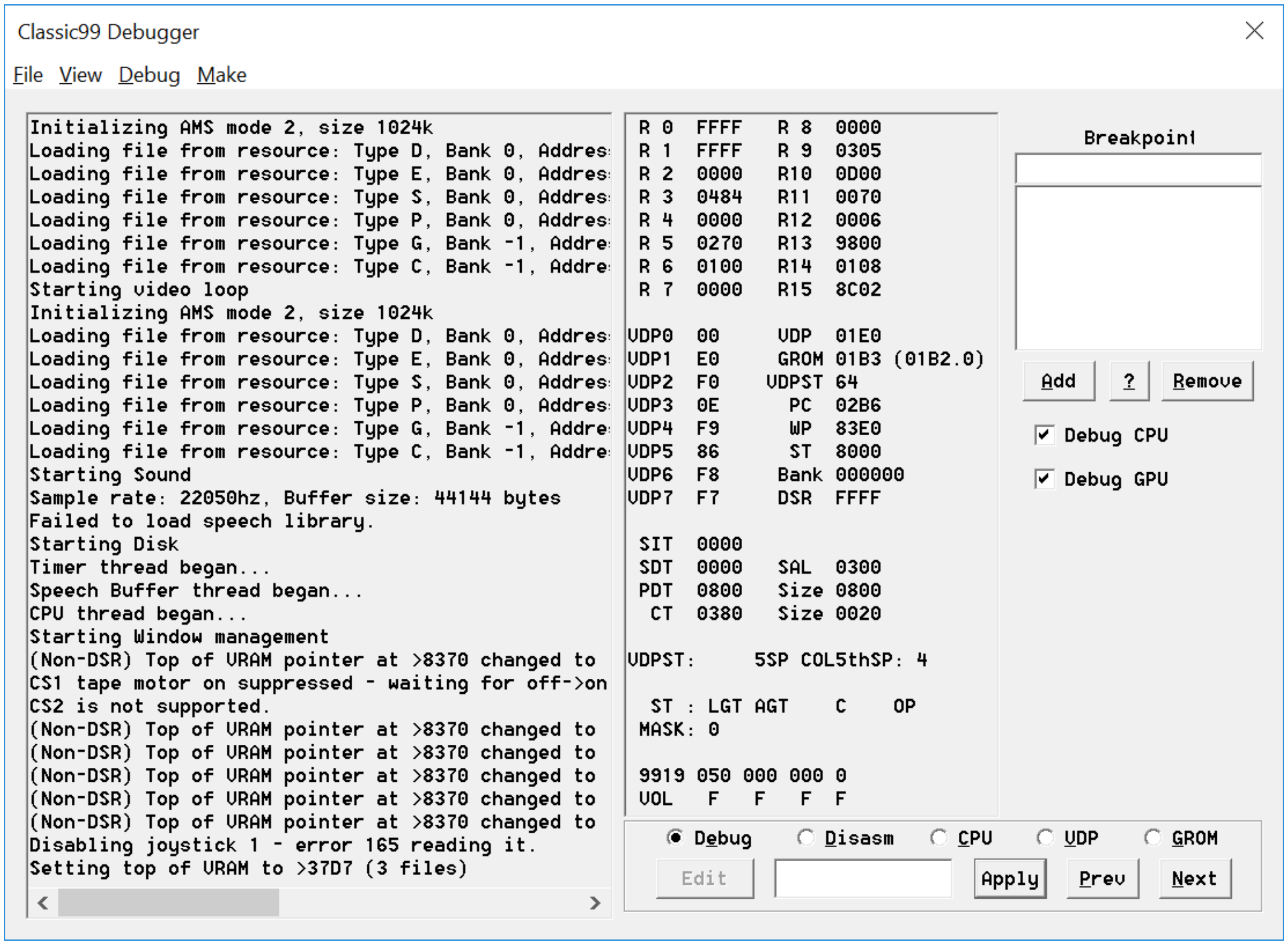
Opens this very PDF document, if it's located in the Classic99 folder. You must have a working PDF reader on your system for this to work.

7. Debugger

Classic99 includes an integrated visual debugger to aid in troubleshooting and debugging programs.

The most important note to the debugger is that keyboard shortcuts, and all breakpoints (including 'step over' breakpoints) require that the debugger window be open. When the debugger window is open, the 'F' keys (*F1-F12*) stop working like TI FCTN+<number> keys and instead map to the debug console.

To display the debugger Window, select Edit->Debugger from the window menu, or press Control+Home.



```
Initializing AMS mode 2, size 1024k
Loading file from resource: Type D, Bank 0, Addres
Loading file from resource: Type E, Bank 0, Addres
Loading file from resource: Type S, Bank 0, Addres
Loading file from resource: Type P, Bank 0, Addres
Loading file from resource: Type G, Bank -1, Addre
Loading file from resource: Type C, Bank -1, Addre
Starting video loop
Initializing AMS mode 2, size 1024k
Loading file from resource: Type D, Bank 0, Addres
Loading file from resource: Type E, Bank 0, Addres
Loading file from resource: Type S, Bank 0, Addres
Loading file from resource: Type P, Bank 0, Addres
Loading file from resource: Type G, Bank -1, Addre
Loading file from resource: Type C, Bank -1, Addre
Starting Sound
Sample rate: 22050hz, Buffer size: 44144 bytes
Failed to load speech library.
Starting Disk
Timer thread began...
Speech Buffer thread began...
CPU thread began...
Starting Window management
(Non-DSR) Top of VRAM pointer at >8370 changed to
CS1 tape motor on suppressed - waiting for off->on
CS2 is not supported.
(Non-DSR) Top of VRAM pointer at >8370 changed to
(Non-DSR) Top of VRAM pointer at >8370 changed to
(Non-DSR) Top of VRAM pointer at >8370 changed to
(Non-DSR) Top of VRAM pointer at >8370 changed to
(Non-DSR) Top of VRAM pointer at >8370 changed to
Disabling joystick 1 - error 165 reading it.
Setting top of VRAM to >37D7 (3 files)
```

```
 R 0  FFFF   R 8  0000
 R 1  FFFF   R 9  0305
 R 2  0000   R10  0D00
 R 3  0484   R11  0070
 R 4  0000   R12  0006
 R 5  0270   R13  9800
 R 6  0100   R14  0108
 R 7  0000   R15  8C02

VDP0  00     VDP  01E0
VDP1  E0     GROM 01B3 (01B2.0)
VDP2  F0    VDPST 64
VDP3  0E      PC  02B6
VDP4  F9      WP  83E0
VDP5  86      ST  8000
VDP6  F8     Bank 000000
VDP7  F7     DSR  FFFF

 SIT  0000
 SDT  0000   SAL  0300
 PDT  0800   Size 0800
  CT  0380   Size 0020

VDPST:     5SP COL5thSP: 4

  ST : LGT AGT   C   OP
 MASK: 0

 9919 050 000 000 0
 VOL   F   F   F  F
```

The initial debugger window looks as above. The exact messages may vary depending on your configuration.

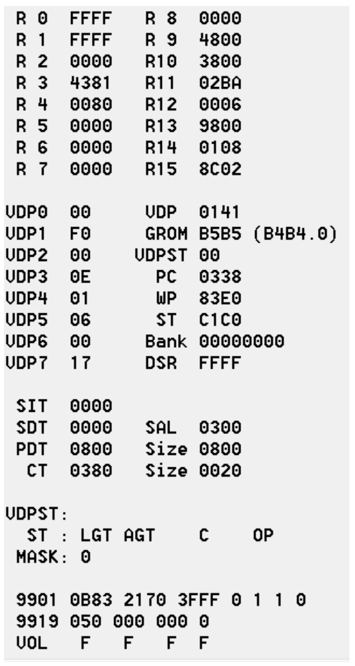
7.1 Information Pane

The *Leftmost* Information pane contains the current text display, as selected by the radio buttons in the bottom right. (Read on for details on the content).

The main user interface to the information dialog is located at the bottom right of the dialog, and contains a five-selection set of Radio Buttons, which control the information displayed in the leftmost information pane, as well as a text entry box and several buttons.

BUG*: the 'Edit' button is permanently disabled as that functionality has not yet been implemented*.

7.2 Register Dump



```
 R 0  FFFF   R 8  0000
 R 1  FFFF   R 9  4800
 R 2  0000   R10  3800
 R 3  4381   R11  02BA
 R 4  0080   R12  0006
 R 5  0000   R13  9800
 R 6  0000   R14  0108
 R 7  0000   R15  8C02

VDP0  00     VDP  0141
VDP1  F0     GROM B5B5 (B4B4.0)
VDP2  00    VDPST 00
VDP3  0E      PC  0338
VDP4  01      WP  83E0
VDP5  06      ST  C1C0
VDP6  00     Bank 00000000
VDP7  17     DSR  FFFF

 SIT  0000
 SDT  0000   SAL  0300
 PDT  0800   Size 0800
  CT  0380   Size 0020

VDPST:
  ST : LGT AGT    C    OP
 MASK: 0

 9901 0B83 2170 3FFF 0 1 1 0
 9919 050 000 000 0
 VOL   F   F   F  F
```

The Register Dump pane contains information about the current state of the system registers. The first section summarizes the 16 CPU Registers from R0-R15. (While you could obtain this information from the CPU memory dump, this is a convenience).

The 9918A VDP's Write-only registers are displayed as VDP0 through VDP7.

The next column contains some generic information:

VDP has the current value of the 9918A VDP's address counter
GROM has the current value of the GROM address counter, and in parentheses, the last GROM address read. The point value is the GROM base it was read from, normally 0.
VDPST is the current value of the 9918A VDP's status register
PC is the 9900 CPU's Program Counter
WP is the 9900 CPU's Workspace Pointer
ST is the 9900 CPU's Status register
Bank is the current cartridge bank-switch value. The first thee nibbles indicate the banking mask, which is determined by the size of the ROM, and the second three are the current bank.
DSR indicates the currently paged DSR. FFFF means no DSR.

The next group calculates the addresses of the VDP tables as a shortcut.

SIT gives the address of the Screen Image Table, calculated from VDP Register 2.
SDT gives the address of the Sprite Descriptor Table, calculated from VDP Register 6.
SAL gives the address of the Sprite Attribute List, calculated from VDP Register 5.
PDT gives the address of the Pattern Descriptor Table, calculated from VDP Register 4. Its size is also shown. In Bitmap mode, the mask is displayed.
CT gives the address of the Color Table, calculated from VDP Register 3. Its size is also shown. In Bitmap mode, the mask is displayed.

VDPST gives a visual representation of which VDP Status bits are currently set. They are:

INT	VDP Interrupt Request is set
5SP	5 sprites on a line bit is set (the 5th Sprite number is shown at the end of the line)
COL	Sprite collision bit is set

ST then breaks down the CPU register. The following bits are shown if they are set:

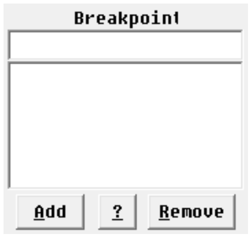
LGT	Logical Greater Than
AGT	Arithmetic Greater Than
EQ	Equal
C	Carry
OV	Overflow
OP	Odd Parity
XOP	Extended Operation

MASK shows the current value of the CPU interrupt mask (which is part of the Status Register).

9901 shows the status of the 9901 timer (current value, readback value, reset value) and the first four CRU bits, which respectively control: timer mode, peripheral interrupt mask, VDP interrupt mask, and timer interrupt mask.

Finally, there is a breakdown of the sound chip, labelled as *9919* and *Vol*. The first row is the current value of the countdown reset register for each channel (ie: for the tone channels, the frequency count, and for the noise channel, the noise type). The second row shows the current attenuation value for each channel.

7.3 Breakpoints



The Breakpoints area allows you to control automatic breakpoint functionality in the emulator. You type your Breakpoint description in the top line, and then click *Add* to add it to the list below. You can also select an existing Breakpoint and click *Remove* to remove it from the list. The *?* button can be used to get a popup summary of the various breakpoint syntaxes.

When Classic99 detects that a Breakpoint condition is met, the emulation is stopped, and the title bar of the emulator will reflect that a Breakpoint has occurred.

Important Note: Breakpoints will only halt the emulation (or execute timing) when the debug window is open!

A Breakpoint Description consists of a single character to indicate the type of the test, an address or address range, and an optional match value.

7.3.1 Match Values

A match value is specified after an address or address range by adding an equals sign (=) and either 2 hex characters (for a byte) or 4 hex characters (for a word). The type of match value required is dependent upon the type of the breakpoint. As a special note, it is the value that is written, not what ends up in memory, that is compared. Therefore you can use match values to detect specific writes to hardware devices (such as cartridge bank switching), even if the value is never stored.

7.3.2 Addresses and Address Ranges

An address is simply specified as 4 hexadecimal digits, and represented an exact address to be matched. Addresses may also be substituted by an address range, which is specified as an open parenthesis '(', the first address in the range, a dash '-', the last address in the range, and a close parenthesis ')'. For example: (8300-83FF) is an address range that specifies all of scratchpad memory. Addresses, whether a single value or a range, may specify a specific cartridge ROM bank by appending a colon ':' and the bank number. The first bank is 0, and the highest possible bank in Classic99 is 'FFF', for a 32MB cartridge. When not specified, all banks are tested.

7.3.3 Masking Data

If you are only interested in certain bits of the data, you may specify a mask after the address inside braces. The data will be verified to be entirely contained in the mask before it may be added.

Example: M837C{20}=20	*- break when GPL status bit is set, ignore all others*

7.3.4 Types of Breakpoints

(none) Break when the CPU PC equals the address specified
Classic99 will break only when the address or address range specified is matched by the CPU program counter.

Example1: A000	*- break when PC is equal to A000*
Example2: (7000-7100)	*- break when PC is any value from 7000 to 7100 inclusive*
Example3: 7000:1	*- break when PC is equal to 7000 in ROM bank 1 only*

***** Break when the address specified is accessed (by any read or write)
Classic99 will break when the CPU address or address range is accessed, whether read or write, whether program or data access.

*Example1: *2000*	*- break when address 2000 is accessed*
*Example2: *(8300-83FF)*	*- break when any address from 8300 to 83FF is accessed*
*Example3: *(6300-63FF):0*	*- break when an address from 6300 to 63FF is accessed in ROM bank 0*

> Break when the address specified is written to. VDP and GROM can also be selected.
Classic99 will break when the CPU, VDP or GROM address or address range is written to only.

Example1: >2000	*- break when address 2000 is written to*
Example2: >V(1300-13FF)	*- break when any VDP address from 1300 to 13FF is written to*

< Break when the address specified is read from. VDP and GROM can also be selected.
Classic99 will break when the CPU address or address range is read from only.

Example1: <2000	*- break when address 2000 is read from*
Example2: <G6300	*- break when GROM address 6300 is read from*
Example3: <(6300-63FF):2	*- break when an address from 6300 to 63FF is read from ROM bank 2*

WP Break when the Workspace Pointer is set to a specific value.
ST Break when the Status Register is set to a specific value. (This is most useful when used with the data masks to match specific bits).

Example: ST{F}=2	*- break when the status register interrupt mask is set to 2*

M Break when the BYTE address specified is written a matching value
Classic99 will break when the CPU address or address range is written a byte matching the one specified. A byte match argument must be included.

Example1: M2005=FF	*- break when address 2005 is set to FF*
Example2: M(8300-83FF)=55	*- break when any address from 8300 to 83FF is set to 55*

W Break when the WORD address specified is written a matching value
Classic99 will break when the CPU address or address range is written a word matching the one specified. A word match argument must be included. Note that the address specified must be even.

Example1: W2004=FFFF	*- break when address 2004 is set to FFFF*
Example2: W(8300-83FE)=55AA	*- break when any address from 8300 to 83FE is set to 55AA*

V Break when the VDP memory byte specified is written a matching value
Classic99 will break when the VDP is written a byte to the VDP address or address range specified matching the byte match specified. A byte match argument must be included.

Example1: V2005=FF	*- break when VDP address 2005 is set to FF*
Example2: V(0000-03FF)=40	*- break when a VDP address from 0000 to 03FF is set to 40*

U Break when the VDP Register specified (0-7) is written a matching value
Classic99 will break when the specified VDP register is written the matching byte value. The VDP register must be from 0-7 (ranges are not permitted), and a byte match argument must be included.

Example1: U7=02	*- break when VDP register 7 is set to 02*

R Break when the CPU Register specified (0-15) is written a matching word value
Classic99 will break when the specified CPU Register is written the matching byte value. The CPU register must be from 0-15 (ranges are not permitted), and a word match argument must be included.

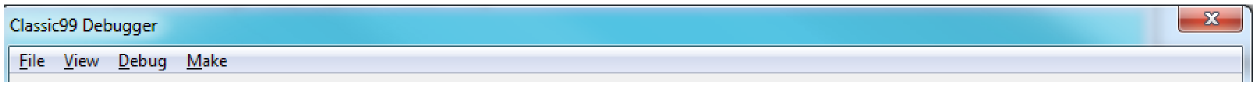
Example1: R14=83E0	*- break when CPU register 14 is set to 83E0*

T Perform cycle counting between a range of addresses (address range required)
Classic99 will count cycles from the first address match until the second, and emit the result each time the second address is reached into the debug log. Addresses must match exactly, but need not be in any particular order. (For example, the first address does not need to be lower than the second). The debug log output will show the word 'Timer', followed by the number of cycles counted. Minimum and Maximum counts are reported, as well as the overall average. The number in parenthesis after 'Average' is the number of times the timer has been hit.

Example1: T(0060-0074)	*- Count cycles between PC 0060 and 0074*

Debug log shows:
```
Timer: 102 CPU cycles - Min: 102  Max: 722  Average(5326): 106
Timer: 722 CPU cycles - Min: 102  Max: 722  Average(5327): 106
```

7.4 Debug Menu



File → Load Breakpoints

Loads a previously saved set of breakpoints from a disk file.

File → Save Breakpoints

Saves the current set of breakpoints to a disk file.

File → Close

Closes the debugger window and returns the emulator to normal operation. (Note: keys do not return to normal unless Scroll Lock is also off).

View → Freeze

The display tends to be continuously updated when the emulator is running. You can select this option to stop all updates, to enable copy and paste from the display.

View → Redraw

If for any reason parts of the display does not seem to be properly drawing everything, select Redraw to force a full update.

View → Clear

This option only affects the Debug view, and clears all messages from the log.

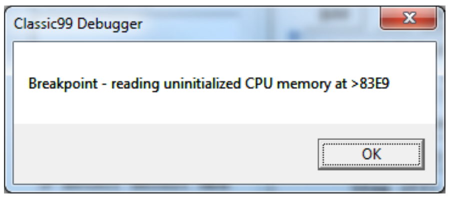
View → ASCII Add Screen Offset for BASIC

When selected, on the VDP view, all characters have 96 subtracted from their values so they read correctly in BASIC and XB.

Debug → Detect Uninitialized Memory

Classic99 has the ability to trigger a Breakpoint on read access to RAM that has never been written to. This can help to find uninitialized memory bugs.

To use this functionality, simply select this option before launching your program. When uninitialized memory is read, a popup box will display while the emulator breakpoints in the background.



Breakpoint - reading uninitialized CPU memory at >83E9

Classic99 tracks writes to memory from the moment that emulation is started. At any time you can select *Debug->Reset Uninitialized Memory Tracking* to clear this tracking. From the moment you select it, ALL RAM is considered uninitialized. If a program is running at the time, including the TI operating system, it is very likely that you will immediately breakpoint for reading uninitialized memory (when in fact, it was initialized before you pressed the button.)

The best way to use this feature is to leave it off while you load your program, and to set a breakpoint on the first instruction of your code. When your breakpoint is hit, then select the Reset option, and then continue the program.

NOTE: if you use any console ROM or GROM functions, including the console interrupt routine, those functions expect that the operating system has set up certain addresses in scratchpad. This will almost certainly cause your program to throw an uninitialized memory breakpoint when those functions are called. Thus you must use some judgment as to whether or not you should reset the memory tracking before starting your program.

Debug → Normal Speed
This will set the emulator running normally. Use this when stopped for a breakpoint, and you want to resume operation.

Debug → High Speed (F11)
This activates *System Maximum* speed. Use the *Normal Speed* button to return to normal operation.

BUG: *System Maximum may consume too many resources and actually run slower on some systems.*

Debug → Pause (F1)
This will cause an immediate breakpoint of the emulator, stopping all operation.

Debug → Step (F2)
Step is used when the emulator is paused for a breakpoint already, and will allow the CPU to execute exactly one instruction. This may or may not allow other hardware to run depending on the exact CPU timing.

Debug → Step Over (F3)
Step Over is a specialized version of Step that will skip displaying all code executed by a BL or BLWP instruction, and stop upon return to the same area of code. Note that this function can detect the return only if the code returns to the next instruction address or 2 bytes after that (to allow for BLWP calls with DATA arguments).

Debug → VDP Chars (F9)
VDP Chars disables the normal screen draw function, and instead displays the entire character set. The effect is as if the Screen Image Table were filled with the values 0-255 repeatedly.

Debug → Dump RAM (F10)
Dump RAM saves the contents of CPU RAM and VDP RAM to files in the current working directory. The CPU RAM is saved as the file MEMDUMP.BIN and is 64k in size, and the VDP RAM is saved as the file VDPDUMP.BIN and is 16k + 8 bytes in size. The last 8 bytes are the 8 VDP Read-Only Registers.

BUG: *The CPU memory saved is a snapshot of the currently visible 64k of CPU RAM. Paged cartridges, DSRs, and AMS memory are not taken into account.*

Debug → Trigger Load Interrupt (F12)
This option triggers a non-maskable LOAD interrupt to the 9900 CPU. Before doing this, the emulator checks whether the LOAD vector at >FFFC has been initialized with non-zero values and displays an error if not. Note that the real 9900 does not perform any such check.

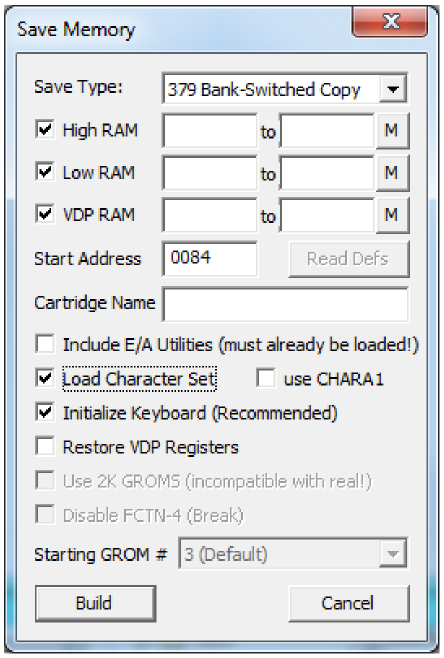
Debug → Break on Illegal Opcode
This option makes the debugger trigger a breakpoint when any illegal opcode is executed.

Debug → Break on Disk Corrupt
This option will breakpoint if the TI Disk Controller's buffer headers in VDP RAM are altered by anything except the disk controller itself.

Make → Save Memory as Program
This option permits you to save the current contents of memory in various formats, see next section.

7.5 Save Memory as Program

This option is only available when the emulator is currently paused for a Breakpoint. When selected, the following dialog is made available:



Various options are enabled and disabled dynamically for each of the various save types. These save types and their options are documented below.

In all formats, three memory types are available.

HIGH RAM – This is the RAM expansion from >A000 to >FFFF.

LOW RAM – This is the RAM expansion from >2000 to >3FFF.

VDP RAM – This is the 16k of RAM attached to the 9918, and runs from >0000 to >3FFF

Each of these memory types, when enabled, has an ‘M’ button that automatically fills in the range of memory modified since the last time uninitialized memory tracking was reset. For HIGH RAM and LOW RAM, this is usually acceptable to use from reset, but the console enumerates all VDP RAM on startup, so if you wish to use this option, you should select *Debug->Reset Uninitialized Memory Tracking* before starting your program.

All formats also have a START ADDRESS, which is the address at which the loaded program should begin execution. By default, this is set to the current Program Counter that the emulator is breakpointed at, but if this is incorrect (and it is definitely okay for your program to do so), you may change it. The start address must reside within either the HIGH RAM or LOW RAM range provided.

E/A#5

This saves RAM in the standard Editor/Assembler PROGRAM image format for later loading via Editor/Assembler or other PROGRAM image loaders. It provides equivalent behavior to the TI provided ‘SAVE’ utility, except that it does not need to be loaded into the TI’s memory and does not require any labels to be set up in your program.

If you DO use the SLOAD, SFIRST, and SLAST labels as recommended by TI, you may press the READ DEFS button to pull the addresses out of the Ref/Def table. Unlike the SAVE utility, you do not need to have SLOAD and SFIRST equal (however, if they are different, you will get one extra output file).

In addition to the memory ranges and start address, you can select “Include E/A Utilities”. These are the

Editor/Assembler utility functions such as VSBW and KSCAN that your program may have REF'd. A PROGRAM image file does not have access to these unless you explicitly provide them, so if you know you need them, select this option to include them. If you have not loaded them or are not sure, it will copy the data into LOW MEMORY for you and ensure your range includes it.

Load Character Set will add the VDP range currently containing the standard characters from 30-127, inclusive. It will also enable "use CHARA1". This is a commonly used character set that included true lower-case letters. Checking this box as well will update the in-memory character set with CHARA1. Note that if the program has already changed the character definitions in this range, the change will be overwritten.

When you click Build, the options are validated and anything questionable or wrong will be called out for your attention. Special patches for c99 programs are available and will be offered if detected. You will then be able to enter a filename – enter a standard TI filename.

The memory ranges will then be saved as separate files, no more than 8k each, with TIFILES and 6-byte PROGRAM image headers on them. The filename will have the last character incremented on each file as is convention, and the first file will start at the Start Address.

Note that if you save VDP memory, that Editor/Assembler loaders will not load it into VDP for you. It is saved as a convenience, but your program will have to load it into place itself.

379 Bank-Switched Copy

This saves RAM as a standard cartridge image, copying all data from bank-switched ROM up to 64k in size.

The Cartridge Name must be filled in, and may be up to 20 characters long. All letters will be made uppercase. Punctuation is not recommended.

In addition to the memory ranges and start addresses, you can select "Include E/A Utilities". These are the Editor/Assembler utility functions such as VSBW and KSCAN that your program may have REF'd. A PROGRAM image file does not have access to these unless you explicitly provide them, so if you know you need them, select this option to include them. If you have not loaded them or are not sure, it will copy the data into LOW MEMORY for you and ensure your range includes it.

Load Character Set will add the VDP range currently containing the standard characters from 30-127, inclusive, and the necessary code to copy it into VDP on startup. It will also enable "use CHARA1". This is a commonly used character set that included true lower-case letters. Checking this box as well will update the in-memory character set with CHARA1. Note that if the program has already changed the character definitions in this range, the change will be overwritten.

Initialize Keyboard adds a call to the ROM keyboard scan function, setting the default unit to 5. This is occasionally needed by software which assumes the mode is already set.

Restore VDP Registers takes a snapshot of the VDP registers, and restores them at cartridge startup as well. This ensures that all tables are located in the same place as at the current breakpoint.

When you click Build, the options are validated and anything questionable or wrong will be called out for your attention. You will then be able to enter a filename – enter any filename. If the filename does not end with "3.BIN", this string is appended to the end of it (so don't enter an extension).

The cartridge image is then built and saved. This bank-switched cartridge includes appropriate code in every bank to guarantee correct startup. The startup code checks the keyboard, restores the VDP registers (if selected), then proceeds to copy the data into the appropriate locations, starting with the VDP memory. At the end of the copy, it jumps to the Start Address, which may be anywhere in the copied CPU RAM range.

GROM Copy

This functions much the same as the 379 Bank-Switched Copy, please check that section for details.

It is limited to only 30k, however, using standard 6k GROMs. By selecting "Use 8k GROMS", this can be increased to 40k. 8k GROMs are not standard, however.

The other GROM-specific option is "Starting GROM #". GROM 3, the default, is based at >6000 and is the beginning of the cartridge space, which runs to the top of the memory space. However, if you wish, you can build the cartridge for GROMs 1 or 2 (which would replace TI BASIC if you removed it), or even GROM 0.

TI BASIC Copy

This makes a special cartridge that copies all 16k of VRAM (as well as the scratchpad and VDP registers), enabling a full resume function of the TI BASIC program currently running. The ideal place to trigger this is on the program's title page, saving all the time of prescan and initialization.

Normally this is a full save, however, it is possible in cases of CALL FILES(1) that there is not enough space in the cartridge to save all important VRAM. In this case, you will be warned, and the program will not include the current contents of the display (ie: it will start with a blank screen). If you run into this, the recommended course of action is to breakpoint right BEFORE the title screen is displayed by the program, so that you still save the time of prescan and initialization.

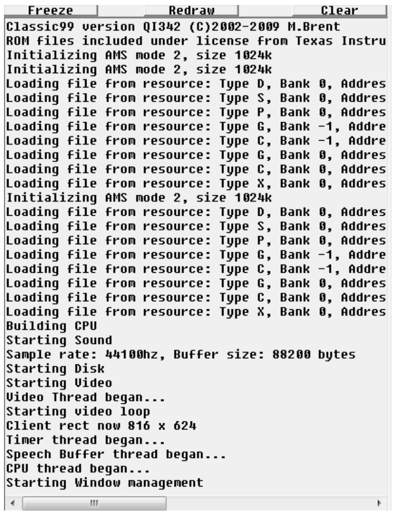
This mode will NOT save Extended BASIC successfully.

ROM Cart Dump

This is just a very simple option with no additional features that will save the contents of cartridge space from >6000 to >7FFF as a binary file. It can be used to make a simple dump of a cartridge image.

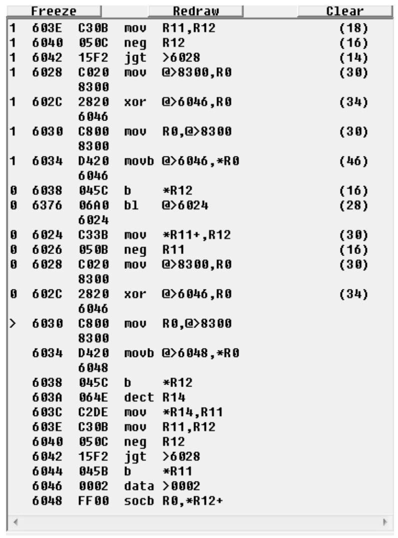
7.6 Radio Buttons

The purpose of the buttons and the text-entry box is modified by the radio buttons:



```
Classic99 version QI342 (C)2002-2009 M.Brent
ROM files included under license from Texas Instru
Initializing AMS mode 2, size 1024k
Initializing AMS mode 2, size 1024k
Loading file from resource: Type D, Bank 0, Addres
Loading file from resource: Type S, Bank 0, Addres
Loading file from resource: Type P, Bank 0, Addres
Loading file from resource: Type G, Bank -1, Addre
Loading file from resource: Type C, Bank -1, Addre
Loading file from resource: Type G, Bank 0, Addres
Loading file from resource: Type C, Bank 0, Addres
Loading file from resource: Type X, Bank 0, Addres
Initializing AMS mode 2, size 1024k
Loading file from resource: Type D, Bank 0, Addres
Loading file from resource: Type S, Bank 0, Addres
Loading file from resource: Type P, Bank 0, Addres
Loading file from resource: Type G, Bank -1, Addre
Loading file from resource: Type C, Bank -1, Addre
Loading file from resource: Type G, Bank 0, Addres
Loading file from resource: Type C, Bank 0, Addres
Loading file from resource: Type X, Bank 0, Addres
Building CPU
Starting Sound
Sample rate: 44100hz, Buffer size: 88200 bytes
Starting Disk
Starting Video
Video Thread began...
Starting video loop
Client rect now 816 x 624
Timer thread began...
Speech Buffer thread began...
CPU thread began...
Starting Window management
```

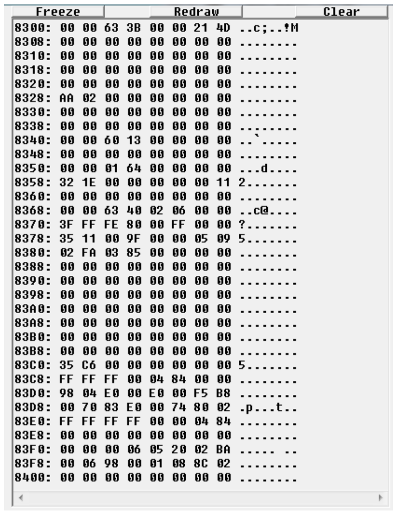
Debug contains a log of debugging information from the emulator itself. This can be helpful when something does not appear to be working correctly, or troubleshooting issues like a file that fails to load. This is just a raw dump of debug text without a specific format.



```
1   603E  C30B  mov   R11,R12                 (18)
1   6040  050C  neg   R12                     (16)
1   6042  15F2  jgt   >6028                   (14)
1   6028  C020  mov   @>8300,R0               (30)
          8300
1   602C  2820  xor   @>6046,R0               (34)
          6046
1   6030  C800  mov   R0,@>8300               (30)
          8300
1   6034  D420  movb  @>6046,*R0              (46)
          6046
0   6038  045C  b     *R12                    (16)
0   6376  06A0  bl    @>6024                  (28)
          6024
0   6024  C33B  mov   *R11+,R12               (30)
0   6026  050B  neg   R11                     (16)
0   6028  C020  mov   @>8300,R0               (30)
          8300
0   602C  2820  xor   @>6046,R0               (34)
          6046
>   6030  C800  mov   R0,@>8300
          8300
    6034  D420  movb  @>6048,*R0
          6048
    6038  045C  b     *R12
    603A  064E  dect  R14
    603C  C2DE  mov   *R14,R11
    603E  C30B  mov   R11,R12
    6040  050C  neg   R12
    6042  15F2  jgt   >6028
    6044  045B  b     *R11
    6046  0002  data  >0002
    6048  FF00  socb  R0,*R12+
```

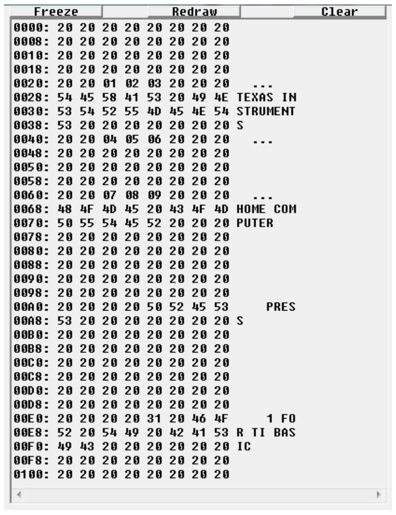
Disasm contains a running disassembly of the emulator. The disassembly shows each instruction as it is executed, with the current instruction indicated by a greater-than symbol (>). Instructions above this symbol were previously executed, and provide a short history. If these instructions were in bank-switched cartridge space, they will be prefixed by a number 0-F, indicating the bank the code came from. Executed instructions include, in parentheses, the number of CPU cycles calculated for this instruction's execution, including memory access time. Instructions below this symbol show the next instructions in memory. *Note that if the F18A GPU is executing, its instructions show instead with a G prefix. These instructions may interleave.*

BUG: *the disassembly is taken from memory at the time that the display is drawn, therefore if AMS or DSR memory pages, the history may show incorrect instructions at addresses executed before the page changed.*



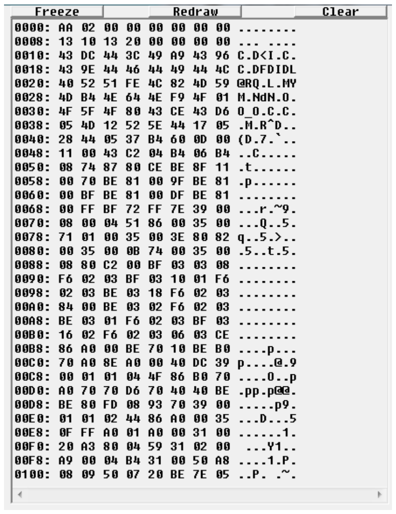
```
8300: 00 00 63 3B 00 00 21 4D ..c;..!M
8308: 00 00 00 00 00 00 00 00 ........
8310: 00 00 00 00 00 00 00 00 ........
8318: 00 00 00 00 00 00 00 00 ........
8320: 00 00 00 00 00 00 00 00 ........
8328: AA 02 00 00 00 00 00 00 ........
8330: 00 00 00 00 00 00 00 00 ........
8338: 00 00 00 00 00 00 00 00 ........
8340: 00 00 60 13 00 00 00 00 ..`.....
8348: 00 00 00 00 00 00 00 00 ........
8350: 00 00 01 64 00 00 00 00 ...d....
8358: 32 1E 00 00 00 00 00 11 2.......
8360: 00 00 00 00 00 00 00 00 ........
8368: 00 00 63 40 02 06 00 00 ..c@....
8370: 3F FF FE 80 00 FF 00 00 ?.......
8378: 35 11 00 9F 00 00 05 09 5.......
8380: 02 FA 03 85 00 00 00 00 ........
8388: 00 00 00 00 00 00 00 00 ........
8390: 00 00 00 00 00 00 00 00 ........
8398: 00 00 00 00 00 00 00 00 ........
83A0: 00 00 00 00 00 00 00 00 ........
83A8: 00 00 00 00 00 00 00 00 ........
83B0: 00 00 00 00 00 00 00 00 ........
83B8: 00 00 00 00 00 00 00 00 ........
83C0: 35 C6 00 00 00 00 00 00 5.......
83C8: FF FF FF 00 04 84 00 00 ........
83D0: 98 04 E0 00 E0 00 F5 B8 ........
83D8: 00 70 83 E0 00 74 80 02 .p...t..
83E0: FF FF FF FF 00 00 04 84 ........
83E8: 00 00 00 00 00 00 00 00 ........
83F0: 00 00 00 06 05 20 02 BA ..... ..
83F8: 00 06 98 00 01 08 8C 02 ........
8400: 00 00 00 00 00 00 00 00 ........
```

CPU shows a dump of 256 bytes of CPU memory starting at the address in the text box. For each line, first the memory address is displayed, followed by eight bytes in hexadecimal, followed by the same eight bytes in ASCII text. Characters not valid for printing as 7-bit ASCII are represented as a period. A 4-character hex address may be entered here, or a CPU register (R0-R15) may be entered, and the memory display will track the address pointed to by that register. Press *Apply* when changing the value in the text box. In addition, the *Prev* and *Next* buttons may be used to advance 256 bytes forwards or backwards in memory. This display normally shows the memory currently visible to the CPU, so applications with pages of memory will only show the current one. For cartridge ROM, you can specify a different bank by entering the address as XXXX:Y, where Y is the bank number.



```
0000: 20 20 20 20 20 20 20 20
0008: 20 20 20 20 20 20 20 20
0010: 20 20 20 20 20 20 20 20
0018: 20 20 20 20 20 20 20 20
0020: 20 20 01 02 03 20 20 20   ...
0028: 54 45 58 41 53 20 49 4E TEXAS IN
0030: 53 54 52 55 4D 45 4E 54 STRUMENT
0038: 53 20 20 20 20 20 20 20 S
0040: 20 20 04 05 06 20 20 20   ...
0048: 20 20 20 20 20 20 20 20
0050: 20 20 20 20 20 20 20 20
0058: 20 20 20 20 20 20 20 20
0060: 20 20 07 08 09 20 20 20   ...
0068: 48 4F 4D 45 20 43 4F 4D HOME COM
0070: 50 55 54 45 52 20 20 20 PUTER
0078: 20 20 20 20 20 20 20 20
0080: 20 20 20 20 20 20 20 20
0088: 20 20 20 20 20 20 20 20
0090: 20 20 20 20 20 20 20 20
0098: 20 20 20 20 20 20 20 20
00A0: 20 20 20 20 50 52 45 53     PRES
00A8: 53 20 20 20 20 20 20 20 S
00B0: 20 20 20 20 20 20 20 20
00B8: 20 20 20 20 20 20 20 20
00C0: 20 20 20 20 20 20 20 20
00C8: 20 20 20 20 20 20 20 20
00D0: 20 20 20 20 20 20 20 20
00D8: 20 20 20 20 20 20 20 20
00E0: 20 20 20 20 31 20 46 4F     1 FO
00E8: 52 20 54 49 20 42 41 53 R TI BAS
00F0: 49 43 20 20 20 20 20 20 IC
00F8: 20 20 20 20 20 20 20 20
0100: 20 20 20 20 20 20 20 20
```

VDP shows a dump of 256 bytes of VDP memory starting at the address in the text box. For each line, first the memory address is displayed, followed by eight bytes in hexadecimal, followed by the same eight bytes in ASCII text. Characters not valid for printing as 7-bit ASCII are represented as a period. A 4-character hex address may be entered here, or a CPU register (R0-R15) may be entered, and the memory display will track the address pointed to by that register. Press *Apply* when changing the value in the text box. In addition, the *Prev* and *Next* buttons may be used to advance 1k forwards or backwards in memory.



```
0000: AA 02 00 00 00 00 00 00 ........
0008: 13 10 13 20 00 00 00 00 ... ....
0010: 43 DC 44 3C 49 A9 43 96 C.D<I.C.
0018: 43 9E 44 46 44 49 44 4C C.DFDIDL
0020: 40 52 51 FE 4C 82 4D 59 @RQ.L.MY
0028: 4D B4 4E 64 4E F9 4F 01 M.NdN.O.
0030: 4F 5F 4F 80 43 CE 43 D6 O_O.C.C.
0038: 05 4D 12 52 5E 44 17 05 .M.R^D..
0040: 28 44 05 37 B4 60 0D 00 (D.7.`..
0048: 11 00 43 C2 04 B4 06 B4 ..C.....
0050: 08 74 87 80 CE BE 8F 11 .t......
0058: 00 70 BE 81 00 9F BE 81 .p......
0060: 00 BF BE 81 00 DF BE 81 ........
0068: 00 FF BF 72 FF 7E 39 00 ...r.~9.
0070: 08 00 04 51 86 00 35 00 ...Q..5.
0078: 71 01 00 35 00 3E 80 82 q..5.>..
0080: 00 35 00 0B 74 00 35 00 .5..t.5.
0088: 08 80 C2 00 BF 03 03 08 ........
0090: F6 02 03 BF 03 10 01 F6 ........
0098: 02 03 BE 03 18 F6 02 03 ........
00A0: 84 00 BE 03 02 F6 02 03 ........
00A8: BE 03 01 F6 02 03 BF 03 ........
00B0: 16 02 F6 02 03 06 03 CE ........
00B8: 86 A0 00 BE 70 10 BE B0 ....p...
00C0: 70 A0 8E A0 00 40 DC 39 p....@.9
00C8: 00 01 01 04 4F 86 B0 70 ....O..p
00D0: A0 70 70 D6 70 40 40 BE .pp.p@@.
00D8: BE 80 FD 08 93 70 39 00 .....p9.
00E0: 01 01 02 44 86 A0 00 35 ...D...5
00E8: 0F FF A0 01 A0 00 31 00 ......1.
00F0: 20 A3 80 04 59 31 02 00  ...Y1..
00F8: A9 00 04 B4 31 00 50 A8 ....1.P.
0100: 08 09 50 07 20 BE 7E 05 ..P. .~.
```

GROM shows a dump of 256 bytes of GROM memory starting at the address in the text box. For each line, first the memory address is displayed, followed by eight bytes in hexadecimal, followed by the same eight bytes in ASCII text. Characters not valid for printing as 7-bit ASCII are represented as a period. A 4-character hex address may be entered here, or a CPU register (R0-R15) may be entered, and the memory display will track the address pointed to by that register. Press *Apply* when changing the value in the text box. In addition, the *Prev* and *Next* buttons may be used to advance 1k forwards or backwards in memory.

7.7 Changing Memory

In addition to controlling the memory address to view in the control input box, you can also change memory and registers by entering an address or register followed by an equals sign '=', and the value that you wish to set. It is recommended, but not required, that the emulator be breakpointed while changing values, to avoid unexpected side effects.

You can type “help” here (and view the debug output) for more information.

The following values may be entered. You must click *Apply* for the change to take effect.

PC=xxxx	Where xxxx is a 4-digit hex value, this changes the Program Counter to the specified value.
WP=xxxx	Where xxxx is a 4-digit hex value, this changes the Workspace Pointer to the specified value.
Cxxxx=yy	Sets the CPU byte at address xxxx to be the byte yy, where both values are hexadecimal. Note that if you attempt to change any type of ROM, the emulator will verify the change before it does it. If you agree to the ROM change, the ROM in memory is modified with no side-effects, and the change will be lost when the emulator is reset. If you decline the ROM change, then the write is performed as if the CPU attempted it. This can be used to trigger hardware and may invoke side effects in the hardware, such as bank switching. You may also specify additional bytes to be written sequentially.
Vxxxx=yy	Sets the byte of video memory at address xxxx to be the byte yy, where both values are hexadecimal. You may also specify additional bytes to be written sequentially.
Gxxxx=yy	Sets the byte of GROM at address xxxx to be the byte yy, where both values are hexadecimal. The emulator will confirm that you wish to modify ROM, and if you accept, the memory is changed. You may also specify additional bytes to be written sequentially. This change is not permanent and will be lost when the emulator is reset.
CRx=yy	Sets the CPU register numbered 'x' to the hexadecimal byte value yy. This modifies only the high byte of a register. Note this is actually a memory access and the address is based on the WP register. CPU registers are numbered from 0-15, and the register number is entered in decimal.
CRx=yyyy	Sets the CPU register numbered 'x' to the hexadecimal word value yyyy. Note this is actually a memory access and the address is based on the WP register. CPU registers are numbered from 0-15, and the register number is entered in decimal.
VRx=yy	Sets the VDP register numbered 'x' to the hexadecimal byte value yy. VDP registers are numbered from 0-7.
ARx=yy	Sets the AMS register numbered 'x' to the hexadecimal byte value yy. AMS registers are numbered from 0-15
DISASM=xxxx,yyyy	Dumps a disassembly from address xxxx to address yyyy to the debug panel
AMS	dumps the AMS registers to the debug panel.

7.8 Keyboard Shortcuts

Normally, the F keys will be mapped to standard TI FCTN keypresses. However, when the debugger is open, all F keys become debugger control keys. Furthermore, the Control key can trigger some behaviour without the debugger being open. And finally, if you set ‘enableSpeedKeys’ to ‘1’ in the INI file, then four of the F keys become emulator speed control keys instead of TI FCTN keys. The following table summarizes. Note that some keys have changed as of version 399.44:

Key	Normal	Control	enableSpeedKeys=1	Debugger Open
F1	Fctn-1	Paste		Breakpoint
F2	Fctn-2	Copy		Step
F3	Fctn-3			Step over
F4	Fctn-4	Read Screen		
F5	Fctn-5	Screenshot unfiltered*		CPU slow**
F6	Fctn-6	Screenshot filtered*	CPU Normal	CPU Normal**
F7	Fctn-7	Toggle Sprites*	CPU Overdrive	CPU Overdrive**
F8	Fctn-8	Toggle Background*	System Maximum	System Maximum**
F9	Fctn-9	Toggle Screen Reader		Show character set
F10	Fctn-0	Stop Talking		Dump RAM
F11	Ctrl-1		Toggle System Max	Toggle System Max
F12	Ctrl-2	Reset*		LOAD interrupt
HOME	Ctrl-U	Open Debugger		

* - key requires debug to be open, even with control pressed.
** - key is different as of 399.44.

7.9 Additional debug opcodes

Classic99 now implements some additional CPU opcodes for debug purposes. To use these, you must add debug/enableDebugOpcodes and set it to ‘1’ before starting Classic99. They will display in the debugger when enabled.

These instructions should be safe to leave in finished code and should have no effect on hardware. Instruction timing in Classic99 should be pretty much the same as hardware. (Follow instructions below for c99_dbg to make these statements true!)

Hex	**Mneumonic**	**Purpose**
0110	c99_norm	CPU normal
0111	c99_ovrd	CPU overdrive
0112	c99_smax	System Maximum
0113	c99_brk	Breakpoint
0114	c99_quit	Quit emulator
012r	c99_dbg	debug printf

Debug printf needs a little explanation. The instruction itself is THREE words long.

In the first word, there are 16 possible hex opcodes – the ‘r’ in the opcode is where you put the register number you want to print, from 0-F. If you don’t want to print a register, use any one of them.

In the second word, and this is **very important**, place a dummy JMP instruction after the hex in order to branch over the argument. This should almost always be >1001. Note that Classic99 DOES NOT PARSE this opcode, but ***real hardware will***.

The third word is the address of the format string. You can look up printf format strings to see how to format it, but note that only types %d, %u, %c, %X and %x will be allowed (you should be able to use any format prefixes, though). The string must be NUL terminated (that is, the last byte must be 00), and it must be less than 128 bytes.

So for instance, if you have this string at >A000: The value of my register is: >%X\0

Then you encode your instruction to print R1 like so: 0121 1001 A000

The debugger will show: c99_dbg(1) >A000,R1

The (1) is the offset specified in your JMP. (It does NOT verify that it really is a JMP though!) It’s just a reminder that you should see ‘1’ there.

>A000 is the address of the format string in CPU RAM.

R1 is the register passed to the format string.

The debug will emit in the debug panel prefixed with **CODE:**

8. Loading Files

While it is not the intention of this document to provide extensive training on how to use a TI, the loading of files takes a combination of emulator understanding and original TI usage understanding. As such, this section provides a brief summary as to how to get files loaded into Classic99 for use.

8.1 TI Console ROMs

It is not necessary to provide console ROMs - they are built in! The system will automatically start with Classic99, and you may choose another under the System menu. However, note that only the TI-99/4A is fully supported.

If you wish to include a separate ROM set, you can override the built-in ROMs by loading them in Classic99.ini, as described under the cartridge section. Classic99 does not provide native support for a custom console ROM set, so you will need to create a cartridge that includes both the custom ROMs and any cartridge ROM images that you wish to use with it. See the INI section for how to create a cartridge configuration.

8.2 Cartridge ROMs

A number of cartridges are now built into Classic99, however, you may want to add more yourself.

Currently, you must modify the Classic99.ini (it will be created after the first time you run and close Classic99). See the description of the INI file below for details on the file and how a cartridge ROM is specified therein.

You may also select Cartridge->User->Open. This will pop up a dialog that asks for the cartridge to load. It only recognizes V9T9 style cartridges, with the naming convention of <name><T>.bin, where 'T' is a letter indicating the type of ROM (such as C, D, 8, 9, G, etc). You need select only one file of the group, and Classic99 will automatically find and load the rest of the files in the same folder. (If you have trouble, you can look at the debug log to see what it attempted to do). Cartridge ROMs may not be zipped or packed into one file.

Note: ROM images distributed for the FinalROM99 and FinalGROM99 often do not contain any tagging information, which Classic99 would need to load them. These files are always "non-inverted 378" style and you can simply change the ROM file's extension to <name>8.BIN to permit this option to work, or create an entry in Classic99.ini

8.3 Disk Images (.DSK)*

These are images of diskettes made popular by PC99 and V9T9. These images come in various sizes, with the most common sizes being 90k, 180k, and 360k. There is also a 400k variant used with the "CF7" or "MiniPeb" device. The TI file system allows 127 files on a single disk, with no subdirectories.

Classic99 allows you to mount V9T9 (DSK) or Win994A (TIDISK) disk images by selecting Image for the type, and the selecting the disk file. It also supports PC99 files, which are sometimes confusingly called DSK and sometimes called TRK, and will attempt to auto-detect them by file size.

8.4 Files on a Disk

Once you have the files on the disk, they may be in the older V9T9 format, especially if you have used TI99Dir to extract them from a disk image. V9T9 was one of the original TI Emulators for DOS - and the first free one. To work around differences in the TI filesystem to DOS, it does a couple of unusual things.

First, certain characters are replaced with high ASCII replacements in order to work in the file system. For instance, a forward slash becomes an overscore line. Classic99 doesn't support this renaming scheme. Instead, the following TI characters should be replaced with '~' on the disk: ?, /, >, <, :, ", *, |

(Note: in a pinch, Classic99 will attempt to work with the extended filenames, but this is not guaranteed).

Classic99 will deal with the internal name. Also, the length of these files was limited to the old DOS 8.3 naming format, meaning any filename longer than 8 characters (TI disk controllers supported at least 10) has an extension. For instance: MYFILENA.ME -- rename it to remove the '.' (MYFILENAME) for Classic99.

Except for the changing of certain characters to '~' and removing the period, do not rename V9T9 files on the disk. Doing so will break them, because the real filename is embedded in the header and must match. *If you rename a V9T9 file on the PC, Classic99 will NOT be able to identify it!*

Note that these changes hurt interaction back to V9T9 and other tools that recognize this format. If you need to interwork with these tools, I recommend keeping filenames to 8 characters or less, and avoid all punctuation except for ';' and '_'.

Classic99's preferred file format is called 'TIFILES' or sometimes 'XMODEM'. It is the format that was globally used by file transfer programs to upload and download files to BBSs, which were likely running on PCs, while preserving the TI standard information. The filename is not included in the header, meaning that the file on disk is the only filename. The same limitation of punctuation above must be followed, however, you are free to rename or move these files on your disks at will. They can also be sent directly back and forth with real TIs using a terminal program and a serial connection, while V9T9 files must be converted to do so.

It may be most convenient to convert V9T9 files to TIFILES format in order to use them with Classic99. To do this, you can run a copy program inside Classic99, or use TI99Dir.

When opening or saving a FIAD, you may specify a file type override by prefixing the filename with the appropriate 3 character prefix:

?W.	Handle file as Windows text (Display format only, saves as text or reads text without extension)
?T.	Handle file as TIFILES – most useful for forcing a TIFILES format save if you have changed the default setting to V9T9.
?V.	Handle file as V9T9 – most useful for forcing a V9T9 format save, as TIFILES is the default.

Classic99's implementation of the file system has none of the restrictions of the TI disk system. Folders may contain an unlimited number of files, filenames may be of any length, and subdirectories may be used. However, the original TI disk system imposed a limit of 127 files in a folder, and 10 character filenames. You may configure any particular disk folder to return filenames longer than 10 characters, or more than 127 files in a directory listing. In addition, you may always use filenames longer than 10 characters. However, exceeding these limits may cause some software, such as disk management software, to malfunction or not see all the files. It is recommended that file management be performed in Windows.

8.5 Archiver 3.03 Files

Archiver 3.03, often abbreviated as Arc303, is the TI's equivalent of ZIP files - they are compressed files that usually contain more than one other files inside them.

Usually these end with an '@' character to indicate they are archived, although some PC distributions may use a .ARC or .ARK extension instead. To extract the files, you will need a copy of Archiver 3.03, which is included in the Classic99 distribution on DSK1 as *ARC303G.*

Copy the archived file (@) into the DSK1\ folders (or any other if you set up more) using Windows. The example assumes you are putting everything into DSK1. If not, just substitute the correct drive number.

Start CLASSIC99.EXE. Select *Editor/Assembler* under Carts->Apps. Press a key then select *EDITOR/ASSEMBLER* from the main menu. Next select *5 RUN PROGRAM FILE* from the Editor/Assembler menu. *(NOTE: Is is from this sequence that 'program' image files on the TI are often called "EA#5" files.)* Enter **DSK1.ARC303G**

Press a key to pass the Shareware notice, then select *2) Extract Files*. Enter your source drive *(1)*, and the name of the '@' file for Source Filename. Select your output drive (may still be 1), and for *Extract all files?* you will usually say *Y*. If you select the same source and output drive, it will also ask *Swap Disks?*, of course with Classic99 you may say *N*.

You'll see *BACK / REDO / Any Key to Begin*, just hit the space bar and the files will be extracted. You can then quit the archive program and move on to running the files themselves.

8.6 Cassette Files

Cassette files are generally distributed in Microsoft WAV format. Classic99 does not support other formats nor zipped files at this time. To mount a WAV file, select Disk->Load/Rewind Tape. Classic99 will wait for the console to cue up the tape by stopping the motor, and then starting it again (normally during the tape load instructions), and then will automatically play the file out to the machine. WAV files should be recorded at least 16khz, 8-bit mono format. Classic99 will automatically resample any higher quality files.

To work with a cassette file, the same commands as for disk are used, but instead of "DSK1", enter "CS1". Cassette operations do not support filenames.

Classic99 does not current support saving to cassette files. Save operations will go through the motions but generate no audio nor files.

Cassette reads are compatible with CPU Overdrive.

8.6 TI BASIC Files

No cartridges are required to be enabled.

Select *TI BASIC* from the main menu. The system will report *TI BASIC READY* and provide a cursor.

You will almost certainly need CPU THROTTLING *on* - it's under Options. If you get rapid key repeat, make sure it is set to Normal.

Enter **OLD DSK1.FILENAME**, where DSK1 is the disk folder you have the file in, and FILENAME is the name of the file. Note that the TI likes everything in uppercase, but it often doesn't matter with Classic99 because the Windows file system is not usually case-sensitive. Classic99 will also support DSK being uppercase or lowercase (although the real TI will not.)

If you have a cassette wave file, enter **OLD CS1** instead. Go to the Disk->Load/Rewind Tape option and select the wave file you wish to load. Press ENTER on the "REWIND TAPE" prompt, then press ENTER again on the "PRESS PLAY" prompt. Classic99 will automatically play the wave file into the system. Note that if you load an Extended BASIC tape program into TI BASIC, it will appear to load but then be corrupted.

Once it loads successfully, type **RUN** will start it up. (For programs that you don't have to type during, turning CPU throttling to *CPU Overdrive* at this point makes slow programs more bearable.)

8.7 Extended BASIC (XB) Files

You must have Extended BASIC selected as the cartridge under Carts->Apps.

Select Extended BASIC from the main menu. The system will report ** READY ** and provide a cursor.

You will almost certainly need CPU THROTTLING *on* - it's under Options. If you get rapid key repeat, make sure it is set to Normal.

Enter **OLD DSK1.FILENAME**, where DSK1 is the disk folder you have the file in, and FILENAME is the name of the file. Note that the TI likes everything in uppercase, but it often doesn't matter with Classic99 because the Windows file system is not usually case-sensitive. Classic99 will also support DSK being uppercase or lowercase (although the real TI will not.)

If you have a cassette wave file, enter **OLD CS1** instead. Go to the Disk->Load/Rewind Tape option and select the wave file you wish to load. Press ENTER on the "REWIND TAPE" prompt, then press ENTER again on the "PRESS PLAY" prompt. Classic99 will automatically play the wave file into the system.

Once it loads successfully, type **RUN** will start it up. ((For programs that you don't have to type during, turning CPU throttling to *CPU Overdrive* at this point makes slow programs more bearable. In XB it usually will not affect the speed of sprites, but it will improve the responsiveness!)

8.8 PROGRAM IMAGE files (E/A#5)

You must have Editor/Assembler selected as the cartridge under Carts->Apps.

Select *EDITOR/ASSEMBLER* from the main menu. Select *5 RUN PROGRAM FILE* from the menu. For *File Name?*, enter **DSK1.FILENAME**, where DSK1 is the disk you have the file(s) on, and FILENAME is the first program file to load. Program images were limited to 8k, so for programs larger than 8k, multiple files were created by adding 1 to the last character of each name. Some programs numbered the files (ie: POPEYE1, POPEYE2, POPEYE3), but many just allowed the last letter to increment (ie: LASSO, LASSP, LASSQ).

Program files will autostart after being loaded.

8.9 Object Files (E/A#3)

It's unusual to come across these as Program Images were faster, smaller, and easier to load, however, just in case you do need to load one, here are the steps. NOTE: Some object files are intended for Extended BASIC, they are not discussed here as these are usually programmer utilities and not standalone programs.

You must have Editor/Assembler selected as the cartridge under Carts->Apps.

Select *EDITOR/ASSEMBLER* from the main menu. Select *3 LOAD AND RUN* from the menu. For *File Name?*, enter **DSK1.FILENAME**, where DSK1 is the disk you have the file(s) on, and FILENAME is the file you are currently loading. It's possible, though even more rare, to have to load more than one file. In that case, keep entering filenames until all files have been loaded. You may need additional details from the program's author.

If the program does not autostart after loading the file(s), hit Enter on the *File Name?* prompt to enter a blank line. You will be prompted with *PROGRAM NAME?,* which unfortunately you need to know. However, most E/A#3 files use **START** or **MAIN**.

9. Using the TI Keyboard

Many programs refer to 'REDO', 'BACK', etc, keys that are not immediately obvious. The TI used FCTN and the number keys for these special terms, so on Classic99 it's Alt and the number key. If Scroll Lock is OFF, and the extended keyboard is active (this is the default), you can also use your PC's 'F' keys F1 through F10.

BUG: This entire discussion covers the TI-99/4A and 2.2 TI99/4A. The TI-99/4 had a smaller keyboard which is not extended with PS/2 support in Classic99. Most 99/4 functions are handled with Shift plus a letter key, as it did not have a FCTN key. These keys are not currently documented!

The TI key names are laid out as follows:

FCTN+	1	DELETE	Deletes the character under the cursor.
	2	INSERT	Toggles Insert mode.
	3	ERASE	Erases the current line
	4	CLEAR	Stops a BASIC or XB program.
	5	BEGIN	Depends on the program being used.
	6	PROCEED	Depends on the program being used.
	7	AID	Depends on the program being used.
	8	REDO	In XB, recalls the last entered line.
	9	BACK	Depends on the program being used.
	=	QUIT	Restarts the TI (soft reset).

The following keys are not necessary to know in Classic99 as the PS/2 keyboard maps them to the keys you would expect to press, but they are useful to know for standard mode or just reference in documentation.

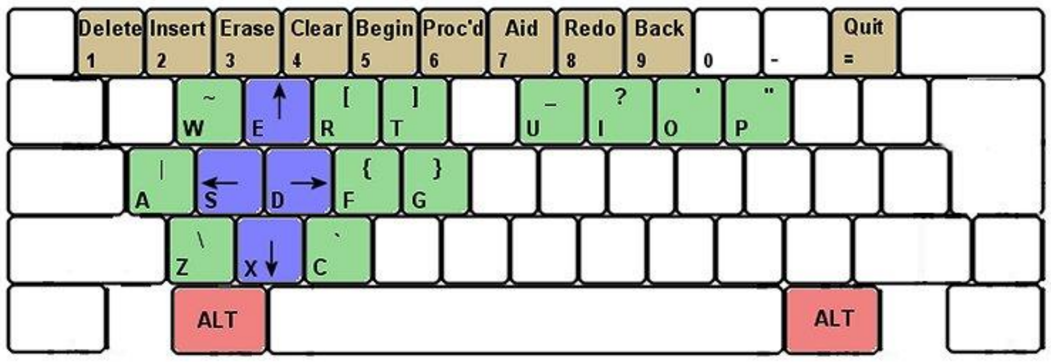
Arrow Keys:

FCTN+	E - Up
	S - Left
	D - Right
	X - Down

Extra characters:

FCTN+	W - ~	I - ?	F - {
	R - [	O - '	G - }
	T -]	P - "	Z - \
	U - _	A - |	C - `

You can reference this graphic in Classic99 as Help->Keyboard Map



10. Classic99.ini

10.1 Description of Settings

This is the main configuration file for Classic99. It contains all settings needed to operate and configure the emulator, including many settings that are not available through the GUI. Since these functions may be useful to some people, the settings are documented here.

Use care with changing the values – invalid settings may cause the emulator to crash. If you have trouble, you can delete this file and Classic99 will create a new one with default settings after being started and closed normally.

[audio]

max_volume

scaler for maximum volume from (percentage) 0-100 (all audio is scaled to this)

samplerate

Sample rate in HZ for the audio system – valid rates are 11025, 22050, 44100

sid_blaster

0	SID Blaster is disabled
1	SID Blaster is enabled *Note: SID Blaster support is deprecated and requires the SIDDll.dll file*

backgroundNoise

0	No background noise
1	Background noise caused by keyboard and interrupt is simulated

continuousReader

0	Speech API will not attempt to read the screen continuously
1	Speech API will attempt to read the screen continuously

[Disk0]

<u>Note: this can be Disk0 through Disk9, representing DSK0-DSK9 in the emulator</u>

Type

0	this disk is not configured
1	this disk uses FIAD (files in a directory) access
2	this disk uses sector-based DSK image files.
3	this disk uses the TICC DSR code *(not supported!)*

Path

string	Specifies the path to the folder for FIAD drives. May be a full path or a relative path. For sector-based DSK images, specify the path and filename to the DSK file.

Entries prefixed with 'FIAD' are applicable only to FIAD disks.

FIAD_WriteV9T9

0	write files using TIFILES headers (default)
1	write files using V9T9 headers (not recommended, V9T9 filenames are limited to 10 characters and have mapping problems with non-alphanumeric characters)

FIAD_ReadTIFILES

0	Reading TIFILES files is disabled, files are never detected as TIFILES
1	Reading TIFILES files is enabled (default)

FIAD_ReadV9T9

0	Reading V9T9 files is disabled, files are never detected as V9T9
1	Reading V9T9 files is enabled (default)

FIAD_ReadTextAsDV

0	Do not read Windows text files as Display/Variable
1	Read Windows text files as Display/Variable (extensions must be TXT, COB or OBJ)

FIAD_ReadTextAsDF

0	Do not read Windows text files as Display/Fixed
1	Read Windows text files as Display/Fixed (extensions must be TXT, COB or OBJ)

FIAD_ReadTextWithoutExt

0	Do not treat files without extension as Windows text (must be this value to read as DF128)
1	Treat files without extension as Windows text (if they are not detected as V9T9 or TIFILES)

FIAD_ReadImgAsTIAP

0	Do not read images as TI Artist files
1	Read images as TI Artist files <u>(NOT IMPLEMENTED YET)</u>

FIAD_WriteDV80AsText

0	Write DV80 files in TI format (default)
1	Write DV80 files as Windows text files

FIAD_WriteAllDVAsText

0	Write all DV files in TI format (unless WriteDV80AsText is set)
1	Write all DV files as Windows text files (regardless of the setting of WriteDV80AsText)

FIAD_WriteDF80AsText

0	Write DF80 files in TI format (default)
1	Write DF80 files as Windows text files

FIAD_WriteAllDFAsText

0	Write all DF files in TI format (unless WriteDF80AsText is set)
1	Write all DF files as Windows text files (regardless of the setting of WriteDF80AsText)

FIAD_AllowNoHeaderAsDF128

0	Do not recognize Windows files without a header
1	Read Windows files without a header and no extension as DF128 (default)

FIAD_EnableLongFilenames

0	Filenames are limited to the usual TI 10 character standard in directories
1	Filenames may be up to 127 characters long in directories. This has no effect on normal file operations which may always use long filenames, and has no effect on emulated sector access which always uses 10 character filenames.

FIAD_AllowMore127Files

0	Directories return no more than the TI standard of 127 files
1	Directories return as many files as are present, without limit. This has no effect on emulated sector access, which always returns a maximum of 127 files.

Entries prefixed with 'IMAGE' are applicable only to Image (sector) disks.

IMAGE_UseV9T9DSSD

0	Disk image files on this drive are numbered from 0-1439 normally (default)
1	Disk images on this drive reverse the order of sectors 360-719 (V9T9)

Entries prefixed with ‘DISK’ are applicable to all disk types.

DISK_AutoMapDSK1

0	Records read from this disk will not have DSK1 strings modified
1	Records read from this disk will modify DSK1 strings to the drive index loaded from

DISK_WriteProtect

0	Disk device is not write protected
1	Writes are prohibited to this disk device

[emulation]

AVIFilename

string	Filename that will be used with 'Start Recording' to write an AVI file. Defaults to C:\Classic99.AVI

cputhrottle *Deprecated, delete it.*

ctrlaltreset

0	Allow normal quit key sequence (Alt+=)
1	Require Ctrl for quit sequence (Alt+Crtl+=)

enableAltF4

0	Disable Alt+F4 key sequence to exit emulator, use as TI keys (Default)
1	Enable Alt+F4, will exit emulator without any warning.

enableINIWrite

0	Disable writing INI – settings will never be saved
1	Enable writing INI (default)

enableSpeedKeys

0	Disable speed control keys in normal use (F keys are all FCTN)
1	Enable speed control keys in normal use (F6,F7,F8,F11 are non-TI)

enableEscape

0	Disable Escape key
1	Enable Escape key as Fctn-9 (BACK) *default*

enableF10Menu

0	F10 will function as TI FCTN-0
1	F10 will activate the program menu

invertcaps

0	Do not invert caps lock
1	Invert caps lock (default)

maxcpf

48387	Maximum cycles per frame (either 50hz or 60hz). Assumes a 3MHz clock.

overdrive

50	CPU multiplier in Overdrive mode. It’s okay if it can’t reach this value

pauseinactive

0	Do not pause emulation when window is not focused

1	pause emulation when window is not focused

ps2keyboard

0	Disable PS/2 keyboard (always set for 99/4)
1	Enable PS/2 keyboard (99/4A only)

sams_enabled

0	Disable SAMS memory expansion card
1	Enable SAMS memory expansion card

sams_size *(temporarily non-functional, SAMS is 32MB by default)*

0	128k SAMS card
1	256k SAMS card
2	512k SAMS card
3	1MB SAMS card (default)

slowdown_keyboard

0	Do not slow down keyboard auto-repeat
1	Slow down keyboard auto-repeat (99/4A only!) (during GPL reads only)

speechenabled

0	Disable speech synthesis
1	Enable speech synthesis

system

0	TI-99/4
1	TI-99/4A
2	TI-99/4A v2.2

systemthrottle *Deprecated, delete it.*

throttlemode

-1	Slow mode (not recommended!)
0	Normal mode
1	Overdrive
2	System Maximum

[joysticks]
active

0	Do not map joysticks
1	Map joysticks using joy1mode and joy2mode

joy1mode *(and joy2mode for joystick 2)*

0	Keyboard (arrow keys and tab)
1-16	PC Joystick 1 through 16

joy1xaxis *(and joy2xaxis for joystick 2)*

0	Controller X axis *(default)*
1	Controller Y axis
2	Controller Z axis
3	Controller R axis
4	Controller U axis
5	Controller V axis
6	X component of controller POV hat
7	Y component of controller POV hat

joy1yaxis	*(and joy2yaxis for joystick 2)*
0	Controller X axis
1	Controller Y axis *(default)*
2	Controller Z axis
3	Controller R axis
4	Controller U axis
5	Controller V axis
6	X component of controller POV hat
7	Y component of controller POV hat

joy1btns	*(and joy2btns for joystick 2)*
-1	32-bit bitmask of buttons to use for fire, -1 for all

joy1minX	*(and joy2minX for joystick 2)*
16384	Minimum value for dead zone on X axis

joy1minY	*(and joy2minY for joystick 2)*
16384	Minimum value for dead zone on Y axis

joy1maxX	*(and joy2maxX for joystick 2)*
49152	Maximum value for dead zone on X axis

joy1maxY	*(and joy2maxY for joystick 2)*
49152	Maximum value for dead zone on Y axis

[debug]

scrambleRAM

0	initialize RAM to 0 on power-on reset
1	initialize RAM to random values on power-on reset

corruptDSKRAM

0	do not corrupt memory used by the TI disk controller after a disk operation
1	corrupt memory used by the TI disk controller after a disk operation

enableDebugOpcodes

0	do not enable Classic99-specific debug opcodes in the 9900
1	enable Classic99-specific debug opcodes in the 9900

[roms]

cartgroup

0	Apps
1	Games
2	User

cartidx

-1	No cartridge selected
??	index of the cartridge from the appropriate menu

[CartGroups]

Group0 (and so on up to 99)

string Name of the group. A number will be appended when searching for cartridges. For instance, if you name a group "**Testing**", then a submenu named "Testing" will appear on the User menu, and you can load cartridges into it by defining sections such as [**Testing0]**, [**Testing1]**, etc. These sections have the same format as the UserCart sections defined below. Note that UserCart sections are still supported. Note too that the total number of cartridges allowed in a single group is 100, and the total number of user cartridges in the system is approximately 9000.

[UserCart0] (and so on up to 99)

name

string Name of the cartridge. If blank or missing, the cartridge is ignored.

message

string Compatibility notes, if desired (optional – will show as 'known issues' under Help).

rom0 string (0-31 for a total of 32 ROMs per cartridge)
Contains notes for loading the cartridge. This is a pipe-limited row of data, formatted like so:
T|AAAA|LLLL|filename

T is the single character ROM type. Most carts only use G, C and X types.

A	AMS memory, RLE encoded (NOT WORKING CORRECTLY)
C	CPU ROM
D	DSR ROM
E	DSR ROM bank 2 (used for p-code)
G	GROM*
K	Paste string for the keyboard after boot – AAAA and LLLL are ignored. The filename is entered into the paste buffer as if Edit->Paste were used. Note that usually a dummy keypress is needed to get paste the master title page.
M	MPD GROM code (activates MPD emulation)
O	'Other' – loads ROMs from another entry, the order is the same as listed in the cartridge menu. AAAA is the group (0-2), LLLL is the index (0-??). filename is ignored and can be used as a comment. *(ex: XB is group 0, index 4)*
P	P-Code GROM
R	CPU RAM - this is loaded like CPU ROM but is not flagged as read-only!
S	Speech ROM
U	UberGROM GROM space (120k) - this activates the UberGROM emulation
T	UberGROM EEPROM space (4k)
V	VDP RAM
X	XB Bank 2 ROM (called 'D' in V9T9 notation)
9	Packed banks with a 74LS379 style decoder. Similar to XB style but more banks and all in one file. Note that this assumes a load at >6000, so you are specifying the offset in the banking space, with bank 0 at >0000, bank 1 at >2000, etc. This is the modern tag, some older ROMs may use '3'. SuperSpace carts (CRU >400) can be loaded this way for now, too, but will likely split them up later. Max size is 32MB, although SuperSpace is limited to accessing 64k.
8	Same as above, but using a 378 style decoder. The net effect is that 379 packed files tend to be 'inverted' while 378 packed files are 'not inverted'.
!	Packed banks with MBX style decoder. Specify files the same way as for 379, as if packed into the same memory area.
*	Autodetect. The type is detected by filename the same as Cartridge->User->Load. The length is the length of the file on the disk. The length and load values are ignored. This is only intended for use by front-ends, since no special detection takes place. If the ROM doesn't work with drag-and-drop or User->Open, then it won't work here.

(GROM is special - to support multiple GROM bases, you can now load GROMs into different memory bases - up to 16 of them! By default, GROMs load into memory base 0. Append a hex number 1-F after the G to specify a different base (like: G1). The console will detect these and add them to the start menu! Note that not all GROMs will work at alternate bases. Also note, in the real console, the console GROMs appear at all bases, so you can't load alternate bases at less than >6000. Classic99 does not support using this with cartridges that include CPU ROM. Each GROM base is offset 4 from the previous, ie: GROM Read is >9800, >9804, >9808, etc.)*

AAAA	load address in hexadecimal (except for K and O and *, see above)
LLLL	length of data in hexadecimal (except for K and O and *, see above) (will override actual size of file)

filename - filename on disk to load (except for K and O, see above)

This system supports both raw ROM files, and ROM files with a 6 byte GRAM Kracker style header. The header is detected by checking if the first byte is >00 or >FF, and if the 5th and 6th bytes represent the load address. The header will be ignored if detected - the data in this INI file is considered authoritative.

[video]

Enable80Col

0	80 column hack is disabled – VDP behaves like a 9918A
1	80 column hack is enabled – VDP honors the 9938 Text2 bit, and ignores writes to registers 8-F

Enable128k

0	128k hack is disabled – VDP has 16k of addressable space
1	128k hack is enable – VDP enables a banking register at VDP R14 (see VDP9938 manual)

EnableF18A

0	F18A is disabled
1	F18A is enabled

FilterMode

0	None
1	2xSAI
2	Super 2xSAI
3	Super Eagle
4	NTSC TV Filter
5	HQ4x

flicker

0	disable sprite flicker (32 sprites per scanline)
1	enable sprite flicker (4 sprites per scanline) (default)

frameskip

0	Number of frames to skip drawing. This may affect interrupts as well. Default 0.

fullscreenmode Note: not all modes are supported on all cards. Classic99 does 32-bit rendering internally.

1	320x240x8
2	640x480x8
3	640x480x16 (default)
4	640x480x32
5	800x600x16
6	800x600x32

7	1024x768x16
8	1024x768x32

heatmapfadespeed

25	Number of pixels updated in heat map per frame. Higher number is faster fade but more CPU is needed to draw it. Default is 25.

hzRate

50	50hz interrupt (PAL)
60	60hz interrupt (NTSC)

InterleaveGPU

0	F18A GPU (if enabled) halts the main 9900 CPU while running
1	GPU instructions are intermixed with the main 9900 CPU instructions (default)

LockFullScreen

0	Normal – full screen may be entered and exited freely
1	Always full screen (overrides StretchMode)

MaintainAspect

0	Do not maintain aspect ratio (allow any size window)
1	Force aspect ratio (fix proportions of window)

StretchMode

0	Do not stretch (fastest)
1	DIB - use GDI to stretch
2	DX - use Direct-X to stretch
3	DX Full - use a full-screen Direct-X mode (set by fullscreenmode)

topX/topY

value	Left (topX) and top (topY) coordinates of the window. -1 for both if not set. *Note: if this position is off screen, the values will be ignored.*

ScreenScale

-1	Custom – use ScreenX/ScreenY
1	1X normal TI resolution
2	2X normal TI resolution
3	3X normal TI resolution
4	4X normal TI resolution

ScreenX/ScreenY

Value	Size of the window if ScreenScale is set to -1

[tvfilter] These settings apply only to the NTSC TV filter

scanlines

0	Don't draw scanlines
1	Draw scanlines

hue

100	Value from 0-200, with 100 being the default setting

saturation

100	Value from 0-200, with 100 being the default setting

contrast

100	Value from 0-200, with 100 being the default setting

brightness	
100	Value from 0-200, with 100 being the default setting
sharpness	
100	Value from 0-200, with 100 being the default setting

10.2 Example of adding a User Cartridge

These lines will add AtariSoft's Pole Position. The assumption is that you have the files in V9T9 format, named POLEPOSC.BIN, and POLEPOSD.BIN. Store the files into the Classic99\MODS folder. Make sure Classic99 is not running, then edit the Classic99.ini file, adding these lines:

```
[UserCart0]
name="Pole Position"
ROM0=C|6000|2000|MODS\POLEPOSC.BIN
ROM1=X|6000|2000|MODS\POLEPOSD.BIN
```

Pole Position is not included - this is just an example! :) Read the [UserCart0] section above for detailed on what the fields mean!

Note that you may have to change the number after 'UserCart' to the first free number not already in the file. The first line sets the name that will appear under Cartridge->User. The second line tells Classic99 that the first ROM to load is POLEPOSC.BIN, indicates that it is type 'C' (cartridge ROM), specifies the load address and size, and the path to the file. It does the same with POLEPOSD.BIN, except that the type is 'X' (bank 2 XB style cartridge).

If you do not know the load address of a ROM file, 6000 is almost always a good guess as this is the base address of the cartridge port for both ROM and GROM. You can get the size from the file if you need to, and the type from the last letter of the filename before BIN: C is cartridge ROM, D is XB bank 2, and G is GROM.

10.3 Example of adding User Groups

This demonstrates adding two new user cartridge groups, called Atari and Tursi. It reuses the Pole Position example above, but note that the cartridge layout is exactly the same as for [UserGroup].

```
[CartGroups]
Group0=Atari
Group1=Tursi

[Atari0]
name="Pole Position"
ROM0=C|6000|2000|MODS\POLEPOSC.BIN
ROM1=X|6000|2000|MODS\POLEPOSD.BIN

[Tursi0]
name=EPSGMod Example
rom0=G|6000|60C8|MODS\EPSGMODG.BIN
```

11. Files Included on DSK1 with Classic99

ARC303G - Barry Boone's archiver - use this to extract '@' or .ARK files. Select cartridge Editor/Assembler, select option 5 Run Program File, and enter 'DSK1.ARC303G'

BLOCKS – Mark Wills' default BLOCKS file containing useful utilities and demos for TurboForth.

CARDEMO - Simple TI BASIC demonstration by Mike Brent. No cartridge required. Start TI BASIC and enter 'OLD DSK1.CARDEMO'. When it is finished, enter 'RUN'

DEMO - Simple assembly demonstration by Mike Brent. Select cartridge Editor/Assembler. Select Option 5 Run Program File, and enter 'DSK1.DEMO'

DEMP and **DEMQ** - part of DEMO. Do not run directly, DEMO will load them.

SPACEFIGHT - Simple TI BASIC game by Mike Brent - no cartridge required. Start TI BASIC and enter 'OLD DSK1.SPACEFIGHT'. When it is finished, enter 'RUN'. The object is to patrol Earth's orbit and destroy only hostile craft. Keys A, Z, S, and X control speed and shields. Speed increases how often you find other craft in orbit. Shields reduce how much damage you take if attacked. Both increase your energy usage. The alarm will sound to warn of a detected craft, and an image of it will appear on the scanner. You must quickly decide whether to fire your torpedos with the space bar (hint: most hostile craft have weapons). If you destroy a friendly craft, you will lose points. If you do not destroy a hostile craft, it will attack you. You will also occasionally reach a starbase, it will refuel your energy and torpedos (if you don't destroy it!).

STRANGER - Simple TI BASIC text adventure by Mike Brent - no cartridge required. Start TI BASIC and enter 'OLD DSK1.STRANGER'. When it is finished, enter 'RUN'. You will be presented with an on-screen story - type simple two and three word commands (ie: *get rope*, *go north*, etc) and try to survive!

XBDEMO - Simple Extended BASIC demo by Mike Brent. Select cartridge Extended BASIC. Enter 'OLD DSK1.XBDEMO'. When it is finished, enter 'RUN'.

FRED – An Extended BASIC game called 'Fred's Tower' by Joe Morris. "The game is aimed at kids, it's basic mathematics for six or seven year olds to use, it's called "Fred's Tower" and it focusses on a steeplejack named Fred who climbs further up his tower the more correct answers you give. One wrong answer and he comically falls from his tower". Select cartridge Extended BASIC. Enter 'OLD DSK1.FRED'. When it is finished, enter 'RUN'.

ABOOT - AMS card Boot menu by the SW 99ers. This lets you launch the preconfigured AMS program **-** TI-Nopoly. Select cartridge Editor/Assembler, select option 5 Run Program File, and enter 'DSK1.ABOOT'. Note that AMS must be enabled in the configuration for this to work.

AEMSSYS, **ABOOTMOD**, **ABOOTMODS**, **AEMSRES1**, **AEMSRES2 -** part of ABOOT. Do not run directly, ABOOT will load them.

ASHOE - advanced boot loader for ABOOT - used to retain menu items. If you don't already know it, it probably won't be too useful today. Someday!

AMSTEST4 - AMS card test program. Does not work properly from ABOOT! Select cartridge Editor/Assembler, select option 5 Run Program File, and enter 'DSK1.AMSTEST4'. As this test is slow you may want to set CPU Overdrive to watch it. Note that AMS must be enabled in the configuration for this to work.

TI-NOPOLY - AMS-based Monopoly. This program must be booted from the AMS Boot menu, as described above. Select cartridge Editor/Assembler, select option 5 Run Program File, and enter 'DSK1.ABOOT'. Note that AMS must be enabled in the configuration for this to work. Enter the menu and simply select TI-Nopoly to start it.

MM_LBLA.OBJ – Line-by-Line Assembler for Mini-Memory. Select cartridge Mini-Memory. Select ‘3’ for Mini Memory, then ‘1’ to Load and Run. Type “DSK1.MM_LBLA.OBJ”. After it returns, press Enter for the Program Name prompt. Type ‘NEW’ for a new program or ‘OLD’ for an old one. Type a space, then ‘END’ to quit. Refer to the Line-by-Line Assembler manual from TI for more details. Note this file is distributes as Windows Text, your disk configuration must support reading Windows Text as DF.

MM_LINES.OBJ – Lines bitmap graphic example for Mini-Memory. Select cartridge Mini-Memory. Select ‘3’ for Mini Memory, then ‘1’ to Load and Run. Type “DSK1.MM_LINES.OBJ”. After it returns, press Enter for the Program Name prompt. Type ‘LINES’ to start it. Note this file is distributes as Windows Text, your disk configuration must support reading Windows Text as DF.

You may drop other files into this folder, either from http://www.harmlesslion.com/cgi-bin/showprog.cgi?TI-99/4A or any other site offering files (sometimes called FIAD or TIFILES) for Classic99 or V9T9. Subfolders are also possible with Classic99, the backslash is accessed by pressing FCTN(ALT) – Z (or pressing backslash with the PS2 keyboard emulation).

12. Contributor Files included with Classic99

Harry_Wilhelm: Harry has permitted the distribution of his four system programming enhancements. These powerful enhancements are intended for programmers wanting to push their system harder. Three are aimed at Extended BASIC programmers, while Playground opens the potential of TI BASIC on a stock, unexpanded console and allows full assembly programs to run.

> **The Missing Link 2.0** : The Missing Link is an add-on environment for accessing full bitmap graphics from within an Extended BASIC program. Harry has updated and re-released this new version in 2015 for all to use. Mount Harry_Wilhelm\TML2.0 as DSK1, then start TI EXTENDED BASIC. Enter **RUN "DSK1.TML"** to start it! Full documentation is available in this folder.
>
> **ISABELLA**: Isabella replaces XB256 and Compiler256 to provide a full Extended BASIC game development system. Isabella includes assembly subroutines for debugging, and then can compile many Extended BASIC programs into high-speed assembly programs. Full documentation is available in the folder - please see the documentation for usage.
>
> **Playground** : Playground is a breakthrough toolset for writing assembly language programs that will ultimately run on a stock, non-expanded TI-99/4A. Programs load into VDP memory and are started by exploiting a bug in the OPEN command, then execute by swapping through the 256 bytes of scratchpad memory. Performance of this technique is impressive - sometimes beating the stock 32k memory expansion! This is a series of tools and functions to help create software that can run in this mode. Full documentation is included, please review the documentation for use.

Lee Stewart: Lee has contributed fbForth, and a collection of documentation and addition disk data.

Richard Lynn Gilbertson: Rich has contributed RXB, which is built in as a cartridge, but he has also contributed a wealth of documentation, sample programs, source code and binaries for emulation and real hardware both, which is included in this folder. Please review the folders and documentation for use.

Tony Knerr: Tony's XB27 suite and documentation is included here.

UCSD pCode: Disk images and documentation for the pCode card are included here.

13. Application Mode ("AppMode")

Classic99 supports an Application Mode which will allow you to distribute your own TI software as a standalone Windows application. In Application Mode, the configuration menu is disabled, and your title's own name appears in the window bar. In addition, you can skip the startup screens and auto-start your title.

This is intended for cartridge-based software (with disk support), but with some effort it is also possible to distribute disk-based software in this method (using, for instance, Editor/Assembler and a keyboard entry in Classic99.ini. This is beyond the scope of this manual however.)

Before starting, please note the following conditions:

- Your software **MUST** be distributed for free. If you wish to sell it or receive income related to it via *any* means, you **MUST** reach an agreement with me (Mike Brent) for the emulator distribution.
- You **MUST NOT** include any of the third party software. This means you are obligated to delete the **DSK1** folder contents, **Contributors** folder, the **MODS** folder, and the **CartPack.dll**. *(If you require TI modules such as Extended BASIC, provide them separately. Classic99 is licensed to include these TI packages and I believe this would be okay - but this is not legal advice. If you have doubt, you must find out on your own. If you require any other modules in the pack you must negotiate with the owner yourself.)*
- **FilterDLL**, **hq4xdll**, **SIDDll** and **SpeechDll** have their own licenses. The simplest answer is to delete them as well. If your software requires them, you **MUST** comply with those licenses on your own, the Classic99 usage does NOT cover your distribution.
- You **MAY** delete the **Classic99 Manual.pdf**, **source.txt** and **whatsnew.txt**, since they are not relevant to your end user.
- You **MAY** rename the Classic99.exe to the name of your choice.
- You **MUST** include a note in the software startup that states either "Powered by Classic99" or "Built with Classic99", and the URL "http://harmlesslion.com" (pure uppercase is also okay). See notes below. You **MAY** also include it in any documentation, but it is mandatory in the software itself.
- You **MUST NOT** indicate approval by HarmlessLion nor use the HarmlessLion logo, except as provided by Classic99 itself as described below.

If that all seems okay to you, here's how to set it up!

- First, copy Classic99.exe into a new folder.
- Then, copy in your software - any cartridge files you need and any disk files and/or folders you need. Set it up as you would like to distribute it.
- If you have a cartridge file, configure it in Classic99.exe as a user cart. This must still be done by hand and is documented earlier in the manual.
- Start Classic99, and configure your disk setup, if needed, any preferred options, and select your cartridge. Test that everything is working, and then close the emulator.
- Start the emulator again, and verify that the cartridge and disk settings were saved properly. You should be able to start your title without changing any configuration.
- Close the emulator, and open Classic99.ini in a text editor. Add the following section (replace 'My Cool App Name' with the name you would like to appear in the title bar.

  ```
  [AppMode]
  EnableAppMode=1
  AppName=My Cool App Name
  AutoStartCart=0
  SkipTitle=0
  ```

- When you start the emulator now, you will notice a few differences. The menu bar will be gone, and your app name will show in the window title bar. In addition, the master title page will be modified to show the HarmlessLion logo, the text "POWERED BY CLASSIC99", and the URL "HTTP://HARMLESSLION.COM". Click through and make sure your software works.

- Most of the time you will want to skip the menu screen. To do this, exit the emulator and edit Classic99.ini. Change AutoStartCart to the number of the entry you want to start. (For most cartridges, this will be 2 since TI BASIC is 1). If you are using startup keyboard presses, you may be able to remove these presses from the cartridge config (test it).
- Now you have a choice. You are required to display the Powered by Classic99 logo and the harmlesslion.com URL. You can leave the default Classic99 screen in place, but it is kind of ugly. You can also put this text inside the startup of your own title. (Please do NOT include the HarmlessLion logo inside your title). If you have done this, you can bypass the title page as well. Just change SkipTitle in Classic99.ini to 1. (Remember that you can only change it when the emulator is closed.)
- If you wish, rename Classic99.exe to WhateverYouLike.exe. You can not change the name of Classic99.ini, however.
- Enjoy! Let me know if you use it!

Here's your excitement: The modem was calling from inside the house! (People found the last on in July 2022)