Downloads

Download the whole site!

If you like this site, but suspect that you may not like your next phone bill, there is an easy way to get the best of two worlds: download the whole site as zipped html files. Then unzip them and view the pages directly from your hard drive.

Caution: there may be updates in the pages that have not been zipped yet...

webpages.zip 1.1 Mbyte. Contains all pages (*.htm) and disassembly listings (*.txt). Last update 12/3/02.
softs.zip 660 Kbytes. Contains all the free softwares below, as such or zipped (*.zip). Last update 12/3/02.
pics.zip 3.4 Mbytes. Contains all pictures (*.jpg, *.gif) plus a few pages that contain a big picture. Last update 12/3/02.

Disassembly

I disassembled and commented the ROMs on several peripheral cards and cartridges. (And if this is considered as a copyright violation, tough luck). Just one thing: as any programmer, I have my very own style for comments and listing. For instance, I never use a R for registers. If your style is different, you'll have to bear with mine...

These files are in .txt format, so that they are directly readable with a text editor (such as notepad) or a web browser. If you want to transfer them back to the TI-99/4A, you should use the TI99PIO.EXE program provided below: set the file type as Dis/Var 80, check the "Strip LF" box and select "CR+LF" as a separator. Then click the [Export] button.

On the TI-99/4A side, you should run a tiny Extended Basic program, such as:

```
100 OPEN #1: "DSK1.FILE1",DISPLAY, VARIABLE 80, OUTPUT
110 OPEN #2: "PIO.EC"
120 LINPUT #2:A$
130 PRINT #1:A$
140 GOTO 120
150 END
```

This will copy the incoming file to a DV80 file called FILE1. Once the PC is done with exporting the file, you'll need to press Fctn-4 to stop the Extended Basic program. These listing are splitted in several files so that they can be edited with a text editor such a Funnelweb's.

Console ROMs and GROMs
Heiner Martin's books, "TI-99/4A Intern" contains commented disassembly listings of the console ROMs and GROMs. It can be found in PDF format on Rich Polivka's site (tiintern.pdf 555 Kbytes).

The disk controller card ROM
dc1.txt ROM header
dc2.txt Power-up routine. Low level subs dealing with the controler (sector R/W, format track). Subroutines for VDP access.
dc3.txt Various file handling routines (mid-level)
dc4.txt Delete opcode. More mid-level routines
dc5.txt DSR entry points, Open, Close, Read, Write, Rewind opcodes.
dc6.txt Load, Save, Status opcodes. Disk directory: Open, Close, and Read opcodes.
dc7.txt Subprograms >10 to >16 (and FILES). Related subroutines.

The RS232/PIO interface card ROM
rs1.txt Power_up routine. Interrupt routine. DSR entry points, Open, Read, Write, Old and Save opcodes.
rs2.txt Close opcode, common DSR exit proc. Subroutines used by the above.

The german Gram-Karte ROM
gc1.txt ROM routines running at >4000-4B00: Power-up routine, call EDITMEM, call MODULE.
gc2.txt GPL program: launches a "modulified" basic program.
gc3.txt GPL program: Gram-Karte main menu.
gc4.txt ROM routines to be run in the low-mem expansion. Displaying Gram-Karte loader menu.
gc5.txt Ditto. Loader menu handling, calling the loaders.
gc6.txt Ditto. General subroutines: VDP access, DSR link, GPLLNK, keyboard scanning. GPL loader, 9900 loader.
gc0.txt A text file describing the arrangement of the above in the card ROM.

The Horizon ramdisk ROS v8.14B
The Ramdisk Operating System was written by John Jonston, Michael Ballmann and Bud Mills.
It's one of the best pieces of assembly programming I ever saw. I strongly recommend that you have a look at it, if only for the pleasure of it. Then compare it with the ugly ROMs in the original TI disk controller (you can tell these guys learned assembly on a PC).

ros1.txt ROM header, DSR entry points, power-up routines (2), Ramdisk + drive info.
ros2.txt Subroutines >10 (sector R/W), >11 (format), >12 (un/protect), >13 (rename), >14 (input), >15 (output).
ros3.txt DSR opcodes 0-9 Open, Close, Read, Write, Restore, Load, Save, Delete, Scratch rec (yes!), Status.
ros4.txt Extra opcodes: >A (Assembly load), >B (XBasic load), >C (Cartridge load).
ros5.txt Private subroutines (disk/file handling). User DSR and ISR entry points. Ends at >57FF.
ros6.txt Starts at >5800, RAMBO off. Subroutines DN, WO, WF, AO, AF, LD, Xbasic launcher, private routines, char defs
ros7.txt Starts at >5800, RAMBO on. Subroutine >B0: RAMBO page selection, P-Gram+ compatibility.
ros0.txt A text file that describes the overall structure of the ROS.

ROS 8.14 is hard to improve, but I nevertheless wrote two short patches for it:
gk.txt This one makes RAMBO mode compatible with the german 128K Gram-Karte, instead of the P-Gram+ card.
vnb.txt This one causes the ROS to insert and automatically update version numbers for each file in the disk directory.

The Editor/Assembler cartridge
ea1.txt GPL program in GROM memory. Handles the cartridge menus, load programs (editor, assembler,

user's).
ea2.txt Ditto. Subroutines to be called from TI-Basic (INIT, LOAD, LINK, PEEK, PEEKV, POKEV, CHARPAT).
ea3.txt Assembly routines stored in GROM, but running in the low mem (LOADER, VDP access, DSRLNK, KSCAN, etc).

The TI-Writer cartridge
wr1.txt GPL program in GROM memory. Handles the cartridge menu, load programs, list disk directory.
wr2.txt Prompts, char patterns, etc, in various languages.

The Mini-memory cartridge
mmg.txt GPL programs in GROM: main menu, DSRs and subprograms (except CHARPAT).
mmg2.txt Ditto: CHARPAT, options 1-3 of the main menu, subroutines for internal use.
mmg3.txt Ditto: Easybug.
mmr.txt Assembly routines in ROM: loader and other assembly subprograms (XMLLNK, VDP access, etc).
mmr2.txt Ditto: subprograms dealing with Basic.

Free software

Being a hobbyist, my policy is not to ask any money for the programs I wrote. I still retain the copyrights though, but you can freely dowload and distribute any of the softwares below. Just make sure you include the whole package, with documentation and all.

Downloader programs

TI99PIO.EXE is a PC program, written in C++, that runs under Win95 and handles communication with the TI-99/4A via the PC-parallel port and the RS232 card's PIO port. You'll need to build a connection cable, but that's very easy to do. No special software is required on the TI-99/4A side as long as you are not trying to transfer "program" files.Required to transfer any file to the TI-99/4A.

FILE-PC/O is an assembly program to be run on the TI-99/4A in Extended basic or in Basic with the Editor/Assembler cartridge plugged in. It allows to transfer any kind of file to or from the PC, much faster than by using the RS232 card's ROM routines. It needs TI99PIO.EXE running on the PC.

So, is this a catch 22? You need FILE-PC/O on your TI-99/4A to download program files from the PC, but how are you going to dowload FILE-PC/O itself? Well, this program is a DF80 file and can therefore be

downloaded without any special software on the TI-99/4A side. It can then be used to download all the other files. See below. Required to download the following two files.

DSK-PC/O is an assembly program to be run on the TI-99/4A in Extended basic or in Basic with the Editor/Assembler cartridge plugged in. It allows to save a floppy disk to a PC file, or to restore it from the PC. Usefull for safety backups and required to download any of the packages below. It needs TI99PIO.EXE running on the PC.

TI2PC is a short Extended Basic program that demonstrates how to transfer non-program files to and from the PC by just using the routines in the RS232 cards.

download.zip, 15 KBytes

Gram-resident disassembler

You need a GRAM device for that one. It's a 9900 disassembler written entirely in GPL, so it does not use any space in CPU memory. It disassembles assembly programs in the whole range of CPU memory, including card ROMs, cartridges ROM banks, Gram-Karte banks, and Horizon Ramdisk RAMBO banks.

It is an interactive disassembler so default output is to the screen. But you can also send it to a printer or a disk file. You can also automatically disassemble a range of memory addresses (it's surprisingly fast for a GPL program).

There as several output formats available that can be customized the way you like (using R for registers, hex or decimal numbers, relative/absolute jumps, colors, etc). Last but not least, it has an extensive online help.

disass.zip 23 Kbytes

Module Explorer

Flabbergasted by Miller's Graphics outstanding Explorer, I wrote this program to do something similar with GPL. It contains a doctored copy of the console GPL interpreter that gives you total control over the GPL language. You can execute it step by step, in low speed, or in (almost) normal speed.

You can view the screen as it would be if Module Explorer was not runnning, or choose among several information screens. These include a memory editor (VDP, CPU, GRAM/GROM and speech synthesizer memory), a GROM / ROM header analyser, a hexadecimal/decimal/binary calculator, etc.

It comes with a very detailed user manual, including a tutorial that takes you through some easy explorations.

It's a usefull debugging tool if you are writing GPL programs. It can also be used to explore the console GROMs or cartridges. Finally, it could be used to cheat when playing game cartridges (slowing down the speed, increasing your score, etc).

modexp.zip 72 Kbytes

GPL assembler package

This package is meant to run with Funnelweb. It consists in:

A **GPL assembler**, wich allows you to write your own GPL programs. It can produce relocatable code, reference other GPL or assembly files, expand macros, etc. It comes with a very **extensive documentation**, including a description of GPL language and a list of usefull routines in console GROMs and ROMs.

A **"Universal" loader**, which loads GPL files produced by a GPL assembler, or assembly files produced by a 9900 assembler and links them together. It can load assembly language in gramcard banks or Horizon Ramdisk banks, either as normal programs or as segments that will be automatically selected by a switching routine upon execution. It can also use library files that follow to the standard defined by RA Green. Finally, it can perform batch operations. The loader comes in two versions: one for the german Gram Karte, one for the P-Gram+ card.

A **batch compiler**. Sometimes, loading a program in memory can become a complex operation. Especially if you load many GPL and assembly files, segments, etc. The batch compiler allow you to write the loader commands once and for all in a DV80 text file. This file will then be interpreted and sent to the universal loader that will performed the required operations. You can also save the compiled version of the batch file in DF80 format, to save the compilation time: the loader recognizes it automatically.

MISS This utility allows you to save GPL and assembly programs as memory image "program" files (see below)

MILD This utility loads "program" files that contain either GPL or assembly language (see below)

A whole bunch of **loader utilities** to be used with the loader: a symbol table editor, a utility table editor, a segment table editor (segments are what you load in a pagable memory), assembly language utilities, etc.

Fweb utilities to be used with Funnelweb: a list editor that allows you to use another loader than Funnelweb's to load the programs on the list. This is required for GPL files since F'web doen't know about them. And a pseudo-list that displays the contents of the GRAM/GROMs just as if it was a program list.

All of the above come with **user manuals**, generally fairly detailed.

gpl_asm.zip 174 Kbytes

MILD and MISS package

MISS
Memory Image Super-Saver. This utility allows you to save assembly programs and/or GPL programs in memory-image "program" files. It also allows you to save a title screen, with music and sprites. It comes as a small assembly-language utility (MISS/O) and with a user-friendly (but memory-hungry) semi-graphic interface.

MILD
Memory image loader. This utility loads memory-image "program" files containing assembly language (in EA5 standard format) or GPL programs (in Gram Kracker format). It also understands some extra formats produced by MISS: RAMBO banks, title screens, GRAMs in alternate bases, etc. Furthermore, MILD can shift itself dynamically if the memory area it sits in is needed for loading. It can even commit suicide and load a program that needs the whole range of CPU memory! Meant to work with Funnelweb, it currently comes in two versions: one for the german Gram-Karte, on for the P-Gram+.

mildmiss.zip, 23 Kbytes.

RIP debugger

Version I
RIP is a postmortem debugger, which means you run it after your program has crashed and use it to find out what happened. A nice feature is that you can load it before your program and have it sitting quietly anywhere in memory. Should your program crash you can activate RIP with the non-maskable LOAD interrupt. You can then see what your program was up to, view its registers, view the three type of memory and the symbol table.

rip.zip 16 Kbytes

Version II
RIP is now a real debugger: it inserts breakpoints in your programs (XOP instructions) that allow you to trace execution or to enter RIP at a given point. The same viewing features than in version I are still available, plus some new ones. You can also execute instructions step-by-step. Additionally, RIP can run directly inside an Horizon Ramdisk, so all you need is 24 bytes in the high memory expansion to activate it!
This program has now become Bug99 (see below). It still works as a remote debugger, but also incorporates an emulator for on-PC debugging.

PC joysticks adapter

The standard TI joysticks can only return binary information: up/down, left/right. By contrast, PC joysticks tell you how far up or down the stick was pushed, which is of course much more convenient. It's quite easy to build a 1-chip adapter board that allow for connecting PC joysticks in the TI-99/4A joystick port.

This package contains instructions to build such an adapter, a TI-Artist schematic, and several associated programs:

- A driver that can be used from both Extended Basic and assembly.
- A program that automatically links one or more sprite to a joystick.
- A calibration program.
- Drivers for TI-Artist version I and TI-Artist Plus!

pcjoy.zip 26 Kbytes

IDEAL: IDE Access Layer

IDEAL version 1.0 is the DSR for my IDE interface card. It's actually a bit more than a DSR, but let's not cut hairs... The package is described in details in a dedicated page.

ideal.zip 44 Kbytes

Multitasking package

This package consists in 4 schedulers that let you multitask your programs in:
- TI-Basic (with the Editor/Assembler or Mini-memory cartridge plugged in)
- Extended Basic
- Assembly language
- GPL

It comes with two series of demo files arranged as tutorials, one for (Extended) Basic, one for Assembly & GPL, and with a detailed instruction manual in Winword format. The manual can also be accessed online.

tasks.zip 140 Kbytes

Bug99 remote debugger & emulator

This Windows program can be used as a remote debugger, controlling the TI-99/4A via a parallel cable. But it also comprises a TI-99/4A emulation, so you can do your debugging on your PC.

See the user manual and download link therein.

How to dowload files to your TI-99/4A

Getting started

First you need to build a connection cable. Follow the instructions below.

Then you need a PC program to handle the connection. Dowload TI99PIO.EXE, optionally with its online help files TI99PIO.HLP and TI99PIO.CNT. Also download the TI-99/4A programs: PIO.EC, DSK-PC.DF80, FILE-PC.DF80 and TI2PC.XB.

Alternatively, just download DWONLOAD.ZIP and unzip it: it contains all the above files.

Step 1:Load and run the downloader

On the TI-99/4A, either enter Extended Basic, or enter TI-Basic with the Editor/Assembler cartridge plugged in (the latter is faster). Then type:
CALL INIT
CALL LOAD("PIO.EC")

On the PC, launch TI99PIO.EXE.
Use the [Browse] button to select the file PIO.EC.
Set the file type as Dis/Fix 80. Make sure all the checkboxes are unchecked and the "separator" field is empty.
Click the [Export] button. This will soon complete the CALL LOAD on the TI-99/4A, (although it may take quite a time with Extended Basic).

Step 2: Download the downloader

Now we want to download the downloading program and save it on disk. This way we won't have to load it from the PC every time we need it.

On the TI-99/4A, type:
CALL LINK("FROMPC","DSK1.FILE-PC/O","DF80")

On the PC, change the file type as "File dump"
Use the [Browse] button and select FILE-PC.DF80 (it is the same program as PIO.EC, only in a different format)
Click the [Export] button once more. This will copy the downloader to DSK1.

Step 3: Download the other utility programs

While you are at it, also download the floppy disk backup program DSK-PC.DF80. Type:
CALL LINK("FROMPC","DSK1.DSK-PC/O","DF80")

On the PC, select DSK-PC.DF80 and click [Export].

You may also want to download the Extended Basic program TI2PC.XB that demonstrates how to transfer files to/from the PC using only the RS232 card routines. Type:
CALL LINK("FROMPC","DSK1.TI2PC","PROG")

Select TI2PC.XB on the PC and click [Export].

Using FILE-PC/O

This program transfers any file in a sector-wise manner to or from the PC. The program TI99PIO.EXE should be running on the PC, with the file type set as "File dump". FILE-PC/O should be loaded from Extended Basic, or from TI-Basic with the Editor/Assembler cartridge plugged in. Just type:
CALL INIT
CALL LOAD("DSKx.FILE-PC/O")

To copy files from the TI-99/4A to the PC

Type any of the following:

```
CALL LINK("TOPC")
CALL LINK("TOPC","DSKx.FILENAME")
A$="DSKx.FILENAME
CALL LINK("TOPC",A$)
```

Where x is the drive number. It MUST be a drive number: you cannot use the syntax DSK.DISKNAME.FILENAME. On the other hand, only the last character before the dot is checked, so you could type "1.MYFILE" instead of "DSK1.MYFILE".

If you do not specify any filename, as in the first exemple, the program defaults to "DSK1.PC".

On the PC side, enter a filename in the edit box. I suggest that you use an extension that reflects the file type: this way you will know what to specify when transfering them back to the TI-99/4A. Then click the [Import] button. As the transfer proceeds you'll see a lot of giberrish on the upper part of the TI-99/4A screen. Once done, the PC will tell you how many bytes were transfered.

To abort transmission, you can press Fctn-4 at any time. The PC automatically aborts if the TI-99/4A does not answer after a few seconds.

To copy files from the PC to the TI-99/4A

Type any of the following

```
CALL LINK("FROMPC")
CALL LINK("FROMPC","DSKx.FILENAME")
CALL LINK("FROMPC","DSKx.FILENAME","DV80")
FORMAT$="DF128"
CALL LINK("FROMPC","DSKx.FILENAME",FORMAT$)
```

Here also, DSKx. must contain the drive number. If you do not include a filename, the program defaults to DSK1.PC

If you do not include any format specification, the current file format is retained if the file already exists. If not, a new "Program" file will be created. If you specify a format for a file that already exists, the new format will replace the old one. Valid format strings are DFxx, DVxx, IFxx, IVxx and PROG, where xx is the record length.

On the PC side, enter a filename in the edit box. Then click the [Export] button. As the transfer proceeds you'll see a lot of giberrish on the upper part of the TI-99/4A screen. Once done, the PC will tell you how many bytes were transfered.

Using DSK-PC/O

This programs transfers the content of a floppy disk to or from the PC. The program TI99PIO.EXE should be running on the PC, with the file type set as "Disk dump". TI-PC/O should be loaded from Extended Basic, or from TI-Basic with the Editor/Assembler cartridge plugged in. Just type:
CALL INIT
CALL LOAD("DSKx.DSK-PC/O")

To save a floppy as a PC file

Simply type:
CALL LINK("TOPC","DSKx")

Where x is the drive number (it cannot be a disk name). Only the last character of the string is considered, so you could omit the "DSK" part. You can also use a string variable:
CALL LINK("TOPC",A$)

The program reads the first sector from the disk to determine the disk size. Then it waits for the PC to answer.

On the PC side, enter a filename in the edit box. I suggest that you use a .DSK extension to identify files containing a floppy disk. Then click the [Import] button. As the transfer proceeds you'll see a lot of giberrish on the upper part of the TI-99/4A screen. Once done, the PC will tell you how many bytes were transfered.

To abort transmission, you can press Fctn-4 at any time. The PC automatically aborts if the TI-99/4A does not answer after a few seconds.

To restore a floppy from a PC file

Type:
CALL LINK("FROMPC","DSKx")

Where x is the drive number. Be aware that all data currently on the floppy will be mercilessly overwritten without further warning!

On the PC side, select the appropriate file and click the [Export] button. Again you'll see a funny display on the TI-99/4A as the transfer is proceeding, then the PC tells you how many bytes were transfered.

How to make a connection cable

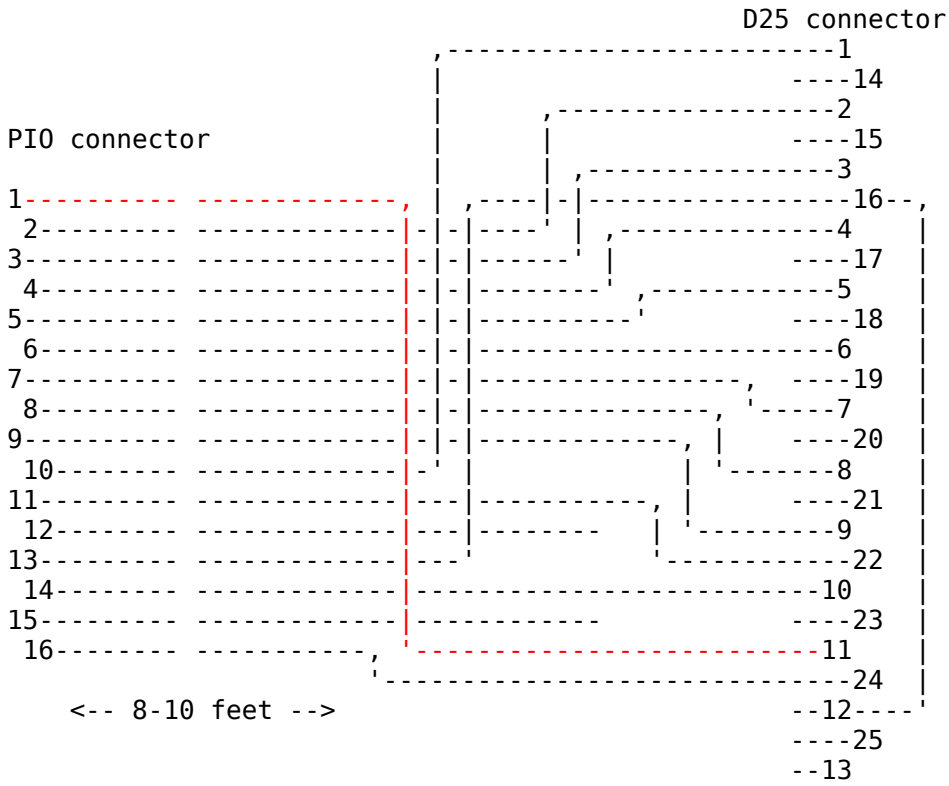
You will need:

- A length of 16-wire ribbon cable. Don't make it too long as this may result in poor transmission. 10 feet is ok.
- A 16-pin socket connector for the PIO plug. Preferably a "clip-on" model, that requires no soldering.
- A male D-25 connector for the PC port (25 pins). Again, try to find a "clip-on" model.
- Pliers, scissors, scotch tape, etc.
- About 30 minutes of your time.

How to proceed:

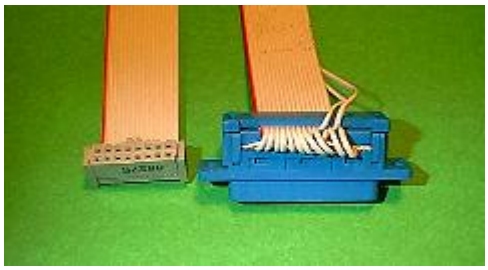
- Insert the ribbon cable into the 16-pin socket. Clip the lid on, which may require using pliers. The goal is that the sharp inner edge of the connectors inside the plug will dent the cable and make contact with the cupper inside. You may have to remove the lid and make sure each wire is fully down, pushing it with your fingernail. Trim the cable close to the connector once done.
- Cut about 2 inches from the other end of the cable. Split apart the individual wires: we'll use them as spacers.
- Split the individual wires from each other, over about 3 inches at the free and of the cable. This is so that we can insert spacers between them and cross some wires over the others (see wiring diagram).
- Make a loop of scotch tape and paste is on the table so that you have a sticky surface to lay the wires on.
- Arrange the wires according to the diagram below. Make sure which side of the cable is wire #1 and which is #16! Leave enough room upstream of the tape to clip the connector on. For a better appearance, the spacers should be evenly aligned on the cable side of the connector. Constantly cross-check the spacing by placing the D25 plug (or a spare piece of cable) near your assembly.
- Once done, use another piece of scotch tape to consolidate the whole. Then clip on the D25 connector as described above for the 16-pin connector. Trim the wires past the connector.
- That's it. Your cable is ready to be used.

Wiring diagram



Remarks

- Wires #12 and #15 are not connected. (They are only pull-up resistors to +5V). I just folded them back on the cable, so that they could be connected to a pin on the D25 connector if it ever became necessary.
- Wires #11 and #16 are grounded. They could be connected to any ground pin on the D25 connector: these are pins # 18 to #25, pick up the most convenient for you.
- The back connection between pin #12 and pin #16 on the D25 connector is optional. It is meant to be used with an ECP port, to fake an answer from the TI-99/4A (there are not enough outputs to provide a real answer). It's not visible on the picture below, as it's on the other side of the plug.
- Wires #13 and #14 (SPAREIN and SPAREOUT) are not strictly required. They are meant for use with an ECP port.



Preliminary. 3/25/99. Not for release.
Revision I. 5/23/99. OK to release.
Revision 2. 5/2/99. Added picture.
Revision 3. 6/17/00. New version of TI99PIO.EXE
Revision 4. 8/25/01. Added IDEAL 1.0
Revision 5. 11/3/10. Added Bug99

Back to the TI-99/4A Tech Pages