```
TTTTTTTT   IIIIIII            GGGG      UU    UU   IIIIIII    DDDDD     EEEEEE
   TT         II           GG   G     UU    UU      II      DD  DD    EE
   TT         II          GG          UU    UU      II      DD   DD   EE
   TT         II         GG           UU    UU      II      DD    D   EE
   TT         II         G            UU    UU      II      DD    D   EEEE
   TT         II         GG    GGGG   UU    UU      II      DD    D   EE
   TT         II          GG     GG   UU    UU      II      DD   DD   EE
   TT         II           GG   GG     UU  UU       II      DD  DD    EE
   TT  O   IIIIIII   O        GGGGG       UUUU     IIIIIII    DDDDD     EEEEEE
```

It is my intent that this guide be used AS a guide and is not intended to be used as a substitute for the original T.I. documentation that was supplied with home computer. This guide is intended for the new user on the T.I. 99/4(A) but information contined within may be useful to any user, new or experienced.
This guide will step thru 15 chapters, each dedicated to a particular part of the computer or to a particular peripheral.
The best advice I can give to an individual looking for help, or looking to get the most out of the computer is to JOIN A USERS GROUP. Many areas have a users group, and some have several. Check the area nearest you for a BBS and call if you have a modem.

THE CHAPTERS ARE :

If anyone has a better idea or more information that could be added to what is offered here, please let me know. As I give a lot of attention to the information, What is presented here is, to the best of my knowledge, complete and accurate, any errors found can be relayed back to me, and I will check and correct them. This is the second edition, and if it seems a little too short in context, the original first edition was 66 pages, I had to condense certain sections and the end result is this guide. Use it and let me know what you think. If you like the guide, you may obtain a copy of the 'original' version for a $ 4.00 donation for printing. The original gude has mini-tutorials for Assembly and Pascal. See the address on the back of this guide.

Thank You.

CHAPTER ONE - PUTTING IT TOGETHER!!!!

Ok, so you have just aquired a T.I. 99/4(A) home computer and all the accessories to go with it, or you have just picked up that P-Box you have always wanted. With more people selling off the T.I. for some other branc of computer, more and more people are just now purchasing either full or semi-full 99 systems. Unfortionatly, the previous owner cannot sell the information that they have gaind through use, and sometimes they even have lost or thrown away parts of the original documentation. This leaves it up to the new user to find a way to 'make the thing work' and that tends to frustrate a lot of the new owners. The T.I. is not the hardest thing to use, but without prior knowledge and the correct instruction manuals, the job is about ten times harder.
That is where I hope that this manual will help you.

Now, lets put that computer together and get going !

After unpacking all of the parts of the system, hook up the computer as shown in figure #1. If you have gotten a P-Box (Peripheral Expansion System box) then the large connector on the interface card goes to the connector on the RIGHT side of the console, see figure 1. At this point we will not go into the RS232 hookup for a printer or the disk drive connections, If you have purchased a P-Box with a disk drive in it, it is more likely to be hooked up already, if not, then see figure 2 as to how to hook it up, then come back here.
Now that all the components are hooked up together, it is time to plug them all in and see if the system works. NOTE : late in 1982 T.I. had a safety modification for the console in-line power supply. Two types of supplies are available, one is a self-contained unit that plugs into the wall and then a cord runs to the computer. This is not the one that needs the safety mod. The one that needs the safety mod has the transformer IN-LINE, in the center of the cord from the wall plug to the computer. If you have one of these, check the box that the console came in, on the transformer itself, or on the console. If you see a green 'saf*ety checked' sticker, then you are O.K, if not, look at the wall plug-end of the cord, and see if it has a 'bulge' on it. The safety mod is installed if you have what looks like about a two inch long fat plug. If you have a regular plug, you can obtain a safety adapter free from T.I. The address is in the back of this guide. The safety mod will not affect the operation of the computer and you do not need it, but T.I. was concerned about the power supply in the console shorting out. I have never seen one do anything but just go 'off' and quit working.
T.I. tells us to turn on the monitor (or TV) first, then the P-box and finaly the console. This is a good idea, but I have known several people including myself that use a power strip with a 'master' switch to turn on the entire system at once, and it works just as well, just leave all the power on and shut off at the main switch on the power strip.
If the computer is working right, the screen will display the TI logo and two stripes of color bars one each above and below the logo. A beep will be heard from the screen. If this does not occur, then please shut all the power down and check all the connections. If the computer still does not work, then proceed to the TROUBLESHOOTING chapter of this guide. (not in the short vers)
If all works, then you are ready to compute. I will not give a description of the BASIC language here, many books have been written on the subject, and the TI BASIC is close to any BASIC language there is. As with all computers, some of the conventions of languages are not used, some have a unique language
s
usage (such as the CALL CLEAR) of commands, but all have manuals written that can take you step-by-step to program POWER !

While I will not go over the BASIC language itself, I will go over some of the more common usages that the TI uses to load, edit and save programs.

Here is a list of the commands and what they do for you. The command is listed on the left in the syntax (wording) that the computer expects, and the result of that command is on the right.

OLD DSK1.LOAD	---	Loads a program called "LOAD" from disk Drive #1. Command must be in capitals.
OLD CS1	---	Loads a program from the cassette tape player designated as #1. If you have a single tape cable, this is all that you use. OLD CS1 must be in capital letters.
OLD CS2	---	Loads a program from second cassette player, cannot be used with some of the new grey case consoles. Note that cassettes do not require a program name to load.
SAVE CS1	---	Save a progr*am to cassette recorder. (no program name needed)
SAVE DSK1.LOAD	---	Save a program to disk in drive #1 with a program name of "LOAD". Command must be in capital letters only
CALL CLEAR	---	Clears the screen.
CALL LOAD(-22,Ø	---	This command can only be used if the Editor assembler, Mini Memory or Extended BASIC modules are in the game port and the memory expansion attached. This loads a 'Ø' in the memory spot '-22'. For more info, see the chapter on the memory expansion.
CALL LINK("START")	---	runs an assembly program that has been loaded with a 'CALL LOAD' or a subroutine. See above
EDIT 1ØØ	---	Displays line 1ØØ with the cursor ready to EDIT the line.
OPEN #1:"RS232.BA=48ØØ.DA=8"	---	This is the 99's way of handling a file, in this case, a device attached to the RS232 port #1 at 48ØØ baud using 8 data bits. See the section on the RS232 for more information
PRINT #1	---	Writes to a file opened as #1. See above.
CALL SOUND(1ØØ,4ØØ,Ø)	---	Produces a 1ØØ millisecond 4ØØ hz sound at the loudest possible volume. 1ØØ=length of the tone, 4ØØ=frequency and Ø=volume. The volume is variable from Ø (high) to 16 (off).

CHAPTER 2 - EXTENDED BASIC

In this part, I would like to discuss the advantages of Extended BASIC ove the console BASIC, and at the same time provide a small insight as to the reason that this software module is so popular. I will begin with the actual module itself.

Extended BASIC is a module that plugs into the GROM (or game) port on the computer. It is a language like BASIC but much more expanded in that Extended BASIC has 43 more commands over the regular console BASIC that comes with the computer. With it you can create moving images, called SPRITES, magnify the image, and even tell when two or more of the sprites are 'touching.' The sprite feature is probably the single biggest reason for getting the Extended BASIC module, but with the module comes another surprising change from the console, and that is SPEED ! The EX. BASIC module is about three times faster displaying data to the screen, and LISTs much faster than console BASIC.

Are sprites and speed the only advantage ? No, you also get error control, the feature of being able to choose what action is taken if an error occurs durring program execution. You also get 'chaining,' or having multiple statements on a single line of up to 14Ø characters per line, and having the ability to place comments at the end of the line. You can load and run another Ex. BASIC program from within your program, and even have subroutines written on disk that can be called from the main program.

Direct screen control is one feature TI should have put in the console, but Ex. BASIC has it. This is the ability to input or display data from any place on the screen, not just to input from the bottom line scrolling up. You can even check how much memory you have left !

In Ex. BASIC, if the P-Box is attached and the disk drives hooked up, when the module is inserted into the GROM port and Ex. BASIC is selected, the module will automaticaly look at disk drive #1 and if it finds a program stored under the name 'LOAD' then it will load and run this program immediately. It is a nice feature if you have a multiprogram series to run, and want a main title to select from, and then you can always return to the main title (LOAD program) when the program execution is complete. If you do not have the P-Box or disk drive, or do not have a disk in drive #1 that has a program name LOAD on it, the computer will respond with the >READY prompt, and you are ready to enter a command.

When programing in Ex. BASIC, you have 24K of space to program if the memory expansion is attached, and 13K if not. If you do have the memory expansion, you also have 8K of stack, or workspace, to use. this space is used by the DIM statement to allow you greater usage of program space.

Lets go over some of the T.I. Extended BASIC conventions, again with the commands on the left and the action of the command to the right. Please note that most of the commands that deal with files need to be in capital letters ONLY or an error message will result, even if the spelling is correct.

CALL SAY("HELLO")	-	'Speaks' the word hello if the speech synthesiser is attached to the computer. Gives and error if the speech unit is not atached.
CALL SPRITE(#1,A,X,Y,Ø,Ø)	-	Produces a sprite labled #1 (of 26 available) and uses the charcater code A, places the sprite at screen location X,Y and has Ø movement in the X and Y axis. X is row. Y is Column.
CALL COINC(ALL,C)	-	Checks for the coincidence of all sprites. If any two sprites are touching or overlaping, then the value placed in variable C will be a -1. You can also specify any two individual sprites, or any location on the screen to check for. A tolerance can be given to allow some 'slop' in the results

CALL CHARSET	-	This command resets the standard TI character set. This is useful if you use the run feature a lot, and do a lot of re-defining the characters.
RUN "DSK1.LOAD"	-	This command can be given in a program or in the immediate mode, and is used to load and run a program from disk. Note : if you have defined a new character in a previous program and then do a RUN for a new program, the old characters are still set and will show up in the new program being run. To avoid this, do a CALL CHARSET.
CALL DELSPRITE(#1)	-	This is to delete a sprite from a program. It also clears the sprite from the screen. After using this command, you can re-define that sprite number. The cammand can also be used to delete ALL sprites.
CALL PEEK(-22,XX)	-	This command allows you to look at any memory location, in this case location -22, and places the value in the variable XX. The value may then be read by the command Print XX. This is very useful if you are planning on doing a few Loads, and want to see what was there first, before you do the call load command.
CALL LOAD("DSK1.AA)	-	Loads a 9900 assembly program from disk drive #1. Note : You must do a CALL INIT sometime prior to the CALL LOAD command or a syntax error will result, even if the spelling is correct. You also need to have the Memory expansion connected to use this command. Only one CALL INIT need to be used prior to any CALL LOAD statements.
CALL INIT	-	Must be used before the CALL LOAD command. This command tells the computer to get ready to accept an assembly language program from Ex. BASIC. Also clears any assy. program from memory.
ON ERROR STOP	-	Instructs the computer to stop upon an error in the program. A line number could be specified instead of STOP if you wish to bring control to another part of the program.
SIZE	-	Returns the number of bytes left for program.
!	-	The exclamation mark (!) can be used instead of the REM statement to place a REMark at the end of a line or on a seperate line. You may place the remark at the end of a multiple statement line, such as : 100 GOSUB 10355 :: ! Branch and loop to control. The REM or ! statement is not executed, and can be any comments that you desire.

The CALL LOAD subprogram can be used to load an assembly language program directly into memory, such as the Nibbler disk copy program. This is a long way to load a program, but it works. The CALL LOAD can also be used to load values directly into the computer. In other computers this action is called 'poking' a value. The TI is a 16 bit computer, and any value from 0 to 255 can be loaded to any location. Be advised that indescriminate Poking of values into unknown locations can cause the computer to 'lock up' or to act very strange. Use the list of Peeks and Pokes at the back of this guide to be sure of what you are doing. While using this table, remember that most of the locations are accurate, but some, more notably the speech locations, are not the same from the 99/4 to the 99/4A. A little bit of experimentation will yield the best results. Just remember that at any time, the computer could lock up and you will lose any data that you have in the computer.

CHAPTER 3 - MODULES, GROM, ROM, RAM

This is a short chapter. many people have questions about the difference between these three terms, and why some of the grey (blond) consoles cannot run the ATARISOFT modules.

ROM - Stands for Read Only Memory. A ROM is a semiconductor chip that has been programmed at the factory and cannot be altered. A ROM can be in many different versions such as PROM, a ROM that is 'blank' when purchased, and can be programmend using a special machine, or an EPROM, which is about the same as a PROM but can be erased after being programmed and then re-programmed again.

GROM - Stands for Graphics Read Only Memory. A GROM is a specialy designed ROM that T.I. uses in most of the modules produced by T.I. The main difference is that the GROM has an internal PC regester, or counter, that the 99/4A recognizes. The reason some of the grey consoles do not work with ROM's is that T.I. checks for the internal PC to see if it is a GROM, and if not, the PC interrupt will not increment, 'locking up' the computer. The consoles that do not work display a 'COPYRIGHT 1983 Texas Instruments' title screen.
Those who have a 1983 console can get the ROM modules to work by getting another module-type device that plugs into the computer game port and then the module plugs into it. It is called a 'GROM BUSTER' and can be purchsed from several of the retail outlets listed in the back of this guide. Prices are usualy in the $2Ø-$25 range.

RAM - Random Access Memory. This is the memory the computer uses to store the program and program information. RAM is not permanent, and can be written to and read from. The T.I. has two types of RAM, it has console RAM and also VDP (Video Display Proccessor) RAM. Console RAM is the BASIC storage area for programs and data, and VDP RAM is where the computer stores certain vital conters, data and screen information. You need not be concerned with the difference unless you are planning to program in assembly language, and that discussion is far beyond the scope of this text.

CHAPTER 4 - RS232 INTERFACE PERIPHERAL

This is a rather long chapter, and will deal with the role of the RS232 interface and some tips on using it along with differences in the 'standard' RS232 and the T.I. RS232.

The RS232 interface, either in the stand-alone or the P-Box card form, has the job of 'interfacing' the computer with the outside world. The RS232 allows you to connect the computer to a MODEM, printer, another computer, or any other RS232 equipped device.

The T.I. interface comes with TWO serial (RS232C) ports and one parallel port. See the end of section for the pin numbers and names. Most printers on the market are ready for parallel connection, having a 'CENTRONICS' type-connection. All this means is that the printer connector is using a standard 36 pin connector, with (more or less) standard pin outs. The pin connections vary with each manufacturer, but all are pretty much the same. The parallel connection coming from the T.I. Peripheral is in the form of a 16 pin connector, again see the section on pin outs for the pin number and name.

The RS232 (serial) interface ports are on the same connector, a DB 25 female connector. Most accessories that can be hooked up to the RS232 ports require only three wires to operate, transmit data (out), receive data (in), and ground. Other applications such as auto-dial, auto-answer modems may require more connections. If you come up on one, you will have to refer to the instructions of the accessory you will be using.

When setting up a printer to the computer, the easiest way is to run the parallel set-up. Parallel is easy for two reasons, 1) Most printers are parallel from the factory, serial is generaly an extra charge, and 2) When programming, it is easier to type in "PIO" to open a port for parallel than to type in "RS232.BA=4800" as a typical serial port open statement. A word of caution for the "PIO" users, if the printer prints garbage, or misses a character now and then, check the position of the cable to the monitor, a drill, etc. The parallel configuration is very suseptable to noise, and the longer the cable, the more important is is to run a shielded cable. Most cases the parallel cable should not be longer than four feet. This keeps noise and errors due to line loss (signal delay) down.

TIP - To print only a few lines to a parallel printer, use the command :

```
LIST "PIO:120-150"       This will list the lines 120 thru 150 to the printer.
```

Modems and terminal emulation terminology :

Modems are devices that allow you to communicate with other computers over the telephone lines. Modem stands for MODulate, DEModulate, which is what a MODEM does, converts the computers serial RS232 output into an audible sound that the phone lines can handle. A modem has two modes, Originate and answer. If you are calling, you are Originating a call. If you are Receiving a call, yo are answering. Each mode uses a different tone, and the modem 'knows' the difference, and will not work if you have the wrong mode. See the section on dialing up the FLUG BBS for more info on Modem use.

STANDARD TERMS - WHAT THEY MEAN.

HALF DUPLEX	- In the half duplex mode, both communicating terminals can 'TALK,' or, transmit data, but only one at a time. If one computer wants to send data, it must wait for the other to finish first.
FULL DUPLEX	- In full duplex, both computers are permitted to 'talk' at the same time. It is not that simple, but the idea is the same.
STOP BIT	- The number of BIT's length to use to tell the other system that your transmission is complete. 1 BIT= 1 BIT length stop BIT, 2= 2 BIT length stop bit.
PARITY	- A method of sending an extra 'odd' bit. If you select eight bit data, 9 BITs are sent. The other system knows that a total of nine BITs are coming, and if it does not get all nine, then a 'parity error' occurs and the sending system is requested to repeat the transmission. Parity can be checked with either an odd or even BIT, or not be checked at all (ODD, EVEN, NONE)
BAUD RATE	- Actualy, BITs-per-second. The speed at which the data is sent or received.

STANDARD RS232 SIGNALS AND T.I. NAMES FOR THEM.

PIN #	RS232 name	T.I. name
1	Protective ground	Protective ground.
2	Transmitted data	Transmitted data (RS232 port 1)
3	Received data	Recieved data (port 1)
4	Request to send	Not used
5	Clear to send	Clear to send (handshake)
6	DSR (Data set ready)	Pull up to +12volts DC.
7	Signal ground (return)	Signal ground (return)
8	Received line signal detect	Data carrier detect (port 1)
9	DST (reserved)	Not used
10	DST	Not used
11	Not used	Not used
12	Port 2 line signal detect	Data carrier detect (port 2)
13	Secondary Clear to send	Clear to send (port 2)
14	Secondary transmitted data	Port 2 Transmitted data
15	DCE (TRANS TIMING)	Not used
16	Secondary received data	Port 2 recieved data
17	DCE (REC TIMIMG)	Not used
18	Not used	Not used
19	Secondary request to send	Data terminal ready (port 2)
20	Data terminal ready	Data terminal ready (port 1)
21	Signal quality	Not used
22	Ring detector	Not used
23	DSR Select (DTE/DCE)	Not used
24	TSET	Not used
25	Not used	Not used.

CHAPTER 5 - DISK DRIVES AND CONTROLLERS

This chapter deals with the types of disk drives and disk drive controllers on the market today.

The original disk drive that T.I. sold for the 99/4A was a Shugart 400L, a 5 1/4 inch single sided double density disk drive. The drive could only be operated in the single density mode due to the controller card. Later on, T.I. also sold MPI 51 disk drives, also single sided double density drives.

The original controller peripheral or card, was able to use double sided drives, but still in the single density mode. To make the card read and write in double density was not done because of the cost of the controller chips that would have to be used to do double density. Two controller cards are available on the market to make use of double density drives, they are made by MYARC and COR-COMP. Although both are double density-double sided controllers, they cannot read each others format. The MYARC card is coming out with an up-grade to allow it to read the COR-COMP format.

Disk drives for the computer are 40 track 48 TPI (tracks per inch) 5 1/4" diskette drives. For a disk drive to be QUAD density, it must be an 80 track drive (96 TPI), or in other words, a DOUBLE-DOUBLE density drive. These high density drives are available at a reasonable cost, but beware, if you purchase them, you cannot use them on the origianal T.I. controller card, but can use them on a COR-COMP or a MYARC controller. You must tell the card that you are using an 80 track drive, so that it can set up the stepping to the higher rate and steps. If you plan on getting the newer Double Sided, Double Density (DSDD) disk drives and have the original T.I. controller, you can still use the double sided feature, just not the double density. I like the double sided disk format because most of the folks I know have it, and it makes it a lot easier to keep track of. A double sided disk will work in a single side drive if the programs are on the first side (which, incedentaly, when you look at the 'front' side of the disk with the lable, the program side is the BACK of the disk.) of the disk, and are not 'fractured,' or spread onto the second side. (More on fractured files in the longer version of this guide)

3 1/2" Drives :

The 3 1/2" drives CAN be used with a a few modifications, but savings are lost in the cost of the mods and of the diskette itself. The 3 1/2" drives are 132 track drives and are 4X density, so the formated disk has (get this) - 9,520 sectors. It takes about 50 minutes to format.
The modifications necessary to run these drives are beyond the scope of this guide, but they can be performed only on the MYARC card.

WINCHESTER (HARD) DRIVES :

The hard drives (5-40 MEGABYTES) are a special drive with a sealed chamber that the disk spins in. The disk is not removable. To run a hard drive requires two cards, a personality card and a controller card. The personality card simply inerfaces to the computer to tell it that the hard drives are in place and introduces the software instructions into the operating system to allow the computer to accept command intended for the hard drive.
The hard drive controller is NOT the same as the disk drive controllers mentioned before. The hard drive controller is a seperate entity that will allow up to four hard disk drives to be used. Both cards are needed to run a hard drive system. The personality card is not hard to make or to get, you can get one from MYARC or other makers. The controller is expensive, and will run about $300.00 or more. A COMPLETE hard drive system for the computer will run about $ 700.00 if you have to buy all of the parts new. A good list of suppliers can be found in the Computer Shopper magazine or look around at some of the surplus stores, Lolir Electronics is a good place to start.

CHAPTER 6 - DISK PROGRAMS AND FORMATS

*** WARNING ***

*** WARNING ***

********** THE INFORMATION CONTAINED HERE HAS THE POTENTIAL TO DESTROY ANY DATA ON THE DISK BEING USED. IF YOU PLAN TO USE ANY OF THE PROGRAMS DESCRIBED HERE, MAKE SURE THAT IT IS DONE ON A BACK-UP DISK ONLY !! DO NOT ATTEMPT TO CHANGE DATA ON ANY DISK THAT YOU CANNOT AFFORD TO RE-INITIALIZE LATER.

This chapter will be a lengthy segment on how the 99 uses the disk drives to store information and how disk sector editor programs such as the public domain DISKO, and copyright programs such as DISK FIXER, DISK+AID and others may be used to recover lost data on a disk.

The first part of our discussion will be on the diskette media itself. The T.I. uses standard 5 1/4 inch 48 TPI disks. The disks are formatted into 4Ø tracks for a regular disk drive and 8Ø tracks for a quad drive. The tracks are seperated into 9 sectors each for a total of 36Ø for normal SSSD and 144Ø sectors for SSQD. Each sector holds about 256 characters. A character is the same as a letter or number on the screen, so looking at a printout of a sector dump, you can count directly across or down. While using the disk sector editor programs DISKO and DISK FIXER (what I use in the examples later,) you have a row (vertical count) and column (horizontal count) of 19X28 Hex display for DISKO and a 16X32 display for DISK FIXER in Hex
When the disk is formatted the report comes back to the user that you have 358 sectors available, not the 36Ø that I said earlier. The reason for the 'lost' two sectors is that the computer automaticaly uses up the first two sectors on the disk for the disk directory and the disk bit-map. The directory is at sector Ø and the bit-map is at sector 1. With computers, most of the internals start at zero and increment up, hence the sector 'Ø' is the first.
The disk directory holds the information such as the disk name, number of sectors available and the number of sectors used. If you have ever encountered an error of 'DISK NOT INITIALIZED' when trying to read a disk catalog, this is the sector that does it. If sector Ø is damaged in anyway, the disk may be imposible to catalog, but all is not lost. later I will go over the way to re-buld a disk with a blown directory and bit-map. (original version only)
The bit-map sector (1) is the 'roadmap' to the data on the disk. As programs are written to the disk, a record must be kept as to the spot (sector) that the program is placed, and the alphabetical order that the program falls into. This is the job of the bit-map. The bit-map 'tells' the disk controller what sector to send the read/write head to find the program header. The bit-map sector also keeps track of the order the programs are in. If a new program is written to the disk, the bit-map is updated to reflect the order of loading. I will give an example of this later.
The sectors 2-22 are reserved for the program header. Just as the bit-map tells the disk read/write head where to find the program header, the program header tells the read/write head what sector the actual program is located. In addition to the program location, the type of program, the program name, if the program is write-protected (cannot be accidentaly written over), and the length of the program are recorded on this sector. The program length is in sectors, and one is added to this number durring the disk catalog to show the space used by the program header itself. All that is read durring a disk catalog is the first few sectors, no reading to any of the programs or other data is done at that time.

The following is a screen dump of the data on the TI-WRITER disk that I am using to write this. I will give the information that the two most popular programs used show on the screen. The DISK FIXER version I am using is the module version marketed by Navarone industries. The DISKO program is a T.I. public domain program that has been around for a long time. DISKO has the feature of searching for a specific program name, and if found will tell you all the information about that file. Although the information is not changed from one program to the next, you will see the different ways that the data is displayed to the user to get an clearer idea of how to count the bytes. In each case, I will point out what most of the numbers mean in the sector, and later will explain how to manipulate those numbers to gain the results that you need to re-build a 'blown' directory. (NOTE : a directory can be blown by not closing a file you were writing to, as in BASIC or when writing to the disk while using a Terminal emulator, and forgetting to 'log off' the disk. This will usualy result in the disk directory showing a 'DIS/VAR Ø' file instead of a 'DIS/VAR 8Ø' (TI WRITER file) like it should.)

The disk sector Ø (Disk directory) screen dump using DISKO :

```
54492D57524954455220Ø1680944
534B2028Ø1Ø1ØØØØØØØØØØØØØØØØ
ØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
ØØØØØØØØØØØØFFFFF8DØ3ØFCFFFF
ØØØØØØ7FFFØØØØØØØF7FF7ØØØØØØ
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFØFFFFFFØFFFFFF7FFØ
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF
```

The disk sector Ø screen dump using DISK FIXER by navarone :

```
NAVARONE IND.  ** DISK FIXER V2Ø. ** SECTOR DUMP     SECTOR ADDRESS  ØØØØ
ADDR =  Ø 1  2 3  4 5  6 7  8 9  A B  C D  E F  INTERPRETED
------------------------------------------------------------------------
ØØØØ = 5449 2D57 5249 5445 522Ø Ø168 Ø944 534B  TI-WRITER *h*DSK
ØØ1Ø = 2028 Ø1Ø1 ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ   (**************
ØØ2Ø = ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ  ****************
ØØ3Ø = ØØØØ ØØØØ ØØØØ ØØØØ FF8D Ø3ØØ FCFF FFFF  ****************
ØØ4Ø = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØ5Ø = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØ6Ø = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØ7Ø = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØ8Ø = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØ9Ø = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØAØ = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØBØ = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØCØ = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØDØ = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØEØ = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
ØØFØ = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
```

The disk sector 1 (BIT MAP) screen dump using DISKO (NOTE : ALL THE 'ZERO's IN BYTES >0040 AND BELOW ARE NOT SHOWN, OTHERWISE THE NEXT FOUR SCREEN DUMPS ARE IDENTICAL TO THE SCREEN WHILE BEING VIEWED. - THE ZEROS ARE EXTRANEOUS AND ARE BEING DELETED FROM THIS TEXT ONLY FOR SPACE SAVINGS.)

```
            ØØØ3ØØØ2ØØØ4ØØØ5ØØØ6ØØØ7ØØØA
            ØØØBØØØFØØ11ØØ1ØØØØ8ØØØØØØØØ
            ØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
            ØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
            ØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
```

The disk sector 1 using the DISK FIXER :

```
NAVARONE IND. *** DISK FIXER V2.Ø ** SECTOR DUMP    SECTOR ADDRESS  ØØØ1
ADDR =  Ø 1  2 3  4 5  6 7  8 9  A B  C D  E F  INTERPRETED

-----------------------------------------------------------------------
ØØØØ = ØØØ3 ØØØ3 ØØØ4 ØØØ5 ØØØ6 ØØØ7 ØØØA ØØØB  ****************
ØØ1Ø = ØØØF ØØ11 ØØ1Ø ØØØ8 ØØØØ ØØØØ ØØØØ ØØØØ  ****************
ØØ2Ø = ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ  ****************
ØØ3Ø = ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ  ****************
ØØ4Ø = ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ  ****************
```

The disk sector 2 (First program on disk) screen dump using DISKO :

```
            44454D4F46494C4520200000008003
            ØØØ2AE5ØØ2ØØØØØØØØØØØØØØØØØØ
            221ØØØØØØØØØØØØØØØØØØØØØØØØØ
            ØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
            ØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
```

The disk sector 2 screen dump using DISK FIXER :

```
NAVARONE IND. *** DISK FIXER V2.Ø ** SECTOR DUMP    SECTOR ADDRESS  ØØØ2
ADDR =  Ø 1  2 3  4 5  6 7  8 9  A B  C D  E F  INTERPRETED

-----------------------------------------------------------------------
ØØØØ = 4445 4D4F 4649 4C45 2020 ØØØØ 8ØØ3 ØØØ2  DEMOFILE  ******
ØØ1Ø = AE5Ø Ø2ØØ ØØØØ ØØØØ ØØØØ ØØØØ 221Ø ØØØØ  .P**********"***
ØØ2Ø = ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ  ****************
ØØ3Ø = ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ  ****************
ØØ4Ø = ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ  ****************
```

Now that you have seen what the disk sectors look like and the different types of data storage, lets use that information to change the disk a bit. First get a good disk and copy it to anot*her disk. DO NOT USE THE ORIGINAL TO DO THIS. After you have made the copy, catalog the disk and keep the catalog for the next part. Using the disk sector program that you have, place the disk in drive #1 and look at sector 1, the bit map. See the information that is stored on the disk? Now move the numbers around a little, like taking the first two and switching them. What you have just done is tell the disk controller that the alphabetical order has changed. Save the new configuration, and then use the disk manager or other program to catalog the disk again. What has happened ? If it worked ok, the first two file names should now be reversed, if not, look at the catalog list and see what HAS changed. If nothing has changed, you may not have saved the changes made while

To load the DISKO program, plug in the Editor Assembler module and select 2 for the Editor Assembler. At the E/A menu select option #3, LOAD AND RUN, the enter in the program name that it is saved under. The program as released by T.I. was DISKO/1 and DISKO/2. If you have the V1.5 program, all you need is to load the DISKO/1 file at the prompt. If you are not sure of the program version, try loading just the first program and then press "ENTER." At the next prompt, "PROGRAM NAME," enter the name "START" in capital letters. The program should start running, if not, shut the computer off and then on again an start loading again, but this time enter BOTH file names.

After the program is loaded and running, it places a menu on the screen with sveral options. Only option #1 and #2 are available. To run the disk sector editor, press 1 and then follow the prompts. Enter the drive number, then the sector number, press enter and the program should access the disk and display th data on the screen. To toggle from the HEX mode to the ASCII mode, press "FUNCTION 2."

NOTE : The DISKO program will accept sector inputs for double sided or double density disks, but will not page "forward" past sector >168 (360), it will go "backward" only.

FUNCTION KEY ---	ACTION CAUSED
1 ---	Toggles (turns on) Hex display mode. All data is displayed in Hex format.
2 ---	Toggles the ASCII display mode. Screen displays normal characters.
3 ---	Return to Editor Assembler screen.
4 ---	Read one sector backwards from present sector.
5 ---	Return to Sector editor title screen. (change disk or sector number)
6 ---	Read forward one sector.
7 ---	Not used.
8 ---	Write to displayed sector. A confirmation will be asked prior to the write operation, and you must press 'Y' to write to the disk.
9 ---	Return to DISKO main title screen.

I hope that this helps. For a further explanation I would suggest that you get the HIDDEN POWERS OF DISK FIXER from Navarone. The book explains a lot of things that I can not get into here due to space limitaions.

CHAPTER 7 - LOADERS - XBASIC, E/A, MINI MEM.

This section deals with the difference between the 'big three' modules, the Editor Assembler, Extended BASIC, and the Mini Memory module. When we say loader, I mean the way that the module handles the loading of an assembly language file into memory. There are some differences, and the easiest way to understand them is to read the Editor Assembler book. If you do not have the E/A manual, then the following is a brief description of the loaders.

Please note that the number presented are in hexadecimal (base 16) and are represented by the > symbol. >2000 means "Hex location 2000," and that all the loaders are what are called 'tagged object code' loaders. Object code is the program produced in DISPLAY FIXED 80 format after being written and assembled from the SOURCE code.

The loader is called when the option #3, LOAD AND RUN is selected from the E/A main menu, if a CALL INIT is done in BASIC with the E/A in place, or the first time a CALL LOAD is invoked from BASIC. Please note that the CALL subroutines in BASIC will only work if the E/A module is in place, and the memory expansion turned on. The reason the memory expansion must be on is that the CALL INIT and CALL LOAD loads the E/A loader from the module to the lower memory block (>2000) of the expansion.

There are basicaly two types of code that can be written for the 99, they are absolute and relocatable. The differences are not important here, but for simplicity, absolute code is code that is written to allow it to load into and/or run in a specific place in memory, and relocatable code is an assembly program (code) that can be manipulated by the memory expansion to 'load' anywhere the expansion has room for it. This is an over-simplified explanation but it will do for now. If you load an absolute code program, then load a relocatable program, it is possible for the two to overwrite each other.

The E/A module uses the area >A000 to >FFD7 to attempt to load. Address >A000 is the first address in the memory expansion. If the area is used, the computer will look to >2024 for the first free address to start the load. These numbers do not mean much to a non-assembly programmer, but they are the locations used.

The REF/DEF table is located in low memory >3FFF to >3000. The REF/DEF table holds the program name and the memory location of the first instruction of the program. This is the location that the computer goes to when a CALL LINK is entred from BASIC or the program name is entered from the E/A option # 3.

If the operator tries to load a program twice or another program with the same name in the REF/DEF table, then the E/A loader will give an error of a 'DUPLICATE DEFINITION' and loading will stop. The Extended BASIC loader does not give this error when a duplicate is found, it just loads the new program OVER the first, leaving the second program.

The Extended BASIC loader is in the utility section of the module at location >2000 also. The main difference is the Speed of loading, with the Editor Assembler being much faster, and the type of object code that the loaders can load. The E/A loader will load either Compressed object or uncompressed object code, while the Ex BASIC will only load the uncompressed code. Compressed object code is the code that is produced when the program is assembled with the 'C' option on the assembler. The EX. BASIC module also requires a CALL INIT command to clear memory prior to the first CALL LOAD command to load a program from disk (or a value into memory), and BASIC with the E/A module does not require a CALL INIT first.

The MINI MEMORY module also has a linking loader that loads at >2000. The main difference between the MM module and the E/A is the use of the memory space, and the fact that the MM module dose not have to have the memory expansion to operate. The MM can be use to load assembly routines or programs from cassette and from disk, and comes with a 'line by line' assembler on cassette. For those wanting to learn 9900 assembly, the MINI MEMORY module is a good choice, along with the COMPUTE! book, Assembly language for the TI 99/4A. A program written for the MM module and saved to disk will not immediately run on the E/A module due to the memory usage differences, but with a small amount of effort, and a good amount of knowledge, the program can be made to run. Changing the REF/DEF table is one the things you need to know how to do, and that is beyond this text, but be advised that it can be done. Do a little experimentation (and frustration) and see what happens.
The MM REF/DEF table is at locations >7FFF down to >7119. The location >7118 is the first free memory location for the module.

LOADING PROGRAMS INTO MEMORY :

Sometimes you get a disk in the mail or from a friend that does not come with any instructions on loading it, and the program is not running, and you do not know how to get it going. Here are some tips on loading.

The Program image loader and save routine (option #5 on the Editor Assembler) only understands files in 8K 'blocks' which translate to a maximum of 33 sectors on a disk. If a file saved in program image format is longer than this, the program is stored under the original name, and then another program is created that is one ASCII character lower than the original, and so on. What does this mean ? If you have a disk listing like the one below, The Program name listed as 33 sectors, and the next LOWER program are in PROGRAM IMAGE format, and can be loaded with the E/A option #5. In this example, I am using the files for DM1000, a freeware utility most of the 99 comunity is familiar with. Although the DM1000 files have an Ex. BASIC loader program that loads these files, they can be loaded with option #5 of the E/A also. Please note that the PROGRAM IMAGE FORMAT is NOT a BASIC or Extended BASIC loadable format, even though they have the PROGRAM listing on a disk catalog. The easiest way to distinguish the two are the sectors, 33, on an assembly language program image format. Trying to load a program into BASIC or Ex BASIC will not harm the computer, but will result in an I/O error 50.

TIP - If you have a file in program image and re-name the file UTIL1, and each subsequent file UTIL2, UTIL3, etc, then you will be able to load the file just by selection option #5 on the E/A menu, and then press 'enter' at the filename prompt, and the program will load. The bad point about this is that the program cannot be identified by the program name (what does UTIL1 mean?) in a catalog, and only one program can be stored with this name per disk.

If you have a DISPLAY VARIABLE 80 file, that is a file that is readable from the TI WRITER module. You can load and edit the file using the 'LF' command on the TI WRITER, or you can print out the file using the E/A module and a printer. Just select option #1, EDIT from the main menu, and then select #4, print. Next enter the file name to print (DSK1.MGRDOC in the previous disk catalog) and press enter. Next enter the printer ID, such as RS232.BA=4800 or PIO depending on your printer. The file should start to print.
The display variable 80 file is also the format used by the E/A module to store source code for the assembly language programmer. Source code is the un-assembled assembly language program.

[illegible]

A DISPLAY FIXED 80 file is the object code storage format used by the Editor Assembler after assembly. It is loadable by the option 3, LOAD AND RUN command from the main menu. A program name may have to be entered to make the program run. Most program names are a relation to the file name, or are a command like START, RUN, GO, LOAD, Etc. The Fast-Term terminal emulator uses the program name TERM to start the program, so you can see that most programmers use some type of recognition with the type of program or program name. if you cannot find the program name by experimenting, use DISKO or some other disk sector editor to locate the LAST sector of the program and page forward one sector. The program name should appear in the ASCII view of the sector in the lower right hand corner, but it could be in the last sector also, see the "PROGRAM NAME" screen dump of DISKO at the end of this chapter.

An INTERNAL VARIABLE 254 file is the way Ex. BASIC saves a file to disk that is longer than 12K in length. The loading of this program may require you do a CALL INIT and then a CALL FILES(1) command to load the program, depending on the total lenght. Internal variable 254 files are generaly over 47 sectors in length and can be loaded using the regular OLD DSK1.XXXXX command.

A DISPLAY VARIABLE 163 file is an Ex. BASIC MERGE format file, and can be loaded with the "MERGE DSK1.XXXXX" command.

Please note that any type of file, with any number, including those listed above, can be used by a program to store data or text, and may not be an executable file, so a little experimentation mey be in order. Remember to try any of your experiments with a backup disk to avoid "permanent" loss of data due to an error. Reading from a disk will not cause any damage to the data, but when writing a file or data to disk, if the disk sector program goes off into never-never land, you can say good-by to the data on the disk.

C H A P T E R 8 - L A N G U A G E S F O R T H E T I 9 9

This chapter deals with the available languages for the TI 99/4(A). This is not a review or tutorial of the languages, just a presentation of the languages that are available and a few pro's and con's of each. As with all programming languages, the preference is entirely up to the programmer/user that is doing the work.

A language is classified as either high or low level. A low level langage is close to the actual binary (the lowest) language that the microprocessor understands, and a high level language is one that requires an 'interpreter,' a program that 'converts' that level into a form the microprocessor can understand (binary). The 'HIGHER' the level, the more interpretation is required. BASIC is a HIGH level language, and actual goes through TWO interpretations per instruction, one through the BASIC interpreter, and the other in the GPL (graphics programming language) interpreter. This dual-translaton is one reason the console BASIC is as slow as it is.

The languages so far available are :

BASIC	(Comes with the console)
Extended BASIC	(module)
Assembly	(E/A module and disk)
FORTH	(Three versions, one cassette, two on disk)
PASCAL	(P-Box card or stand alon peripheral and disk)
PILOT 99	(disk)
LOGO	(module)
LOGO II	(module)
C	(disk)
BASIC COMPILER	(disk)

Most of the languages require either the E/A or Ex BASIC module to load, plus the memory expansion and disk drive.

BASIC : The language that comes with the computer console, is slow and has quite a few limitations, such as no direct access to the internal program counters, clock, or interrupts. The TI BASIC is very easy to learn and use, and is a good language to learn on.

Extended BASIC : This module offers the advantage of greater speed and versatility through more commands, and has the advantage of more disk I/O available. The same limitations are in effect for the Ex BASIC module as BASIC, with some execptions. Please see the section on Ex. BASIC.

Assembly language : Assembly is very fast and efficient while running, but very difficult to learn and de-bug once written. The assembly programmer has many variables to take care of and keep track of, but the speed of the language is the greatest advantage. All of the functions of the 9900 microprocessor are available to the assembly language programmer to control the program.

FORTH : The three versions of FORTH are Pretty much the same, with some samll differences in the syntax and disk storage. FORTH is a difficult language to learn at first, but has the advantage of having more SIG's (special interest groups) than most other languages. This interest in FORTH is due to the speed of the program execution, which is almost as fast as an assembly program. Many good books have been written on FORTH and are available at most book stores.

PASCAL : The T.I. version is UCSD PASCAL. The P-Code card allows you to run the P-System without using up much memory space. The P-System language is easy to learn, and is a compiled language so it runs very fast. The dissadvantage of PASCAL is the extra cost of the P-Code card, and the time involved to compile a large file.

PILOT 99 : A FREEWARE language that is easy to learn and very usefull. Requires E/A to load and run, and comes with a 68 page manual. PILOT 99 is a simple language to learn and use, and is good for a begining programmer. Speed is slow, but not as slow as BASIC.

LOGO   : Designed as a 'childrens' language, LOGO is a high level language with many features, and can be used even by most adults who enjoy programming. TI LOGO requres a module and cassette player to save programs and procedures.

LOGO II : A higher-developed version of LOGO. Uses cassette or disk to load the programs or procedures.

C   : A new programming language, offered as FREEWARE. I do not know much about the T.I. version, but c in general is a difficult language to learn and use, but yields exellent results in speed and accuracy.

BASIC COMPILERS : While not a language, compilers are a part of the packages offered for the 99. most are simple, use either the Mini Memory or E/A modules, and have quite a few restrictions as to the use of variables, disk I/O and higher functions. In most cases, a complicated program can be written in a lower-level language easier than trying to write for a compiled version.

chapter 9 - Peeks & Pokes --

10/2/85

==

The following is a list of Peeks and
Pokes that was downloaded from the
Source. It is in 80 column format.
Compiled by Scott Darling.

==

24K OF DATA STORAGE

If you need to work with quite a bit of data or would like to change programs, but save the data after you press CALL QUIT then you can set up the 24K of High-Memory in the PEB as a single data file called "EXPMEM2", you open this file just as you would a disk file with one exception - you must PRECEED th OPEN statement with a CALL LOAD to the location -24574 as follows:

For INT/VAR files - 24
For DIS/VAR files - 16
For INIT/FIX files - 8
For DIS/FIX files - 0

Heres and example:
If you want to open up the Expansion Memory for Display,Variable 80 files this is what you'd do:

```
100 CALL INIT
110 CALL LOAD(-24574,16)
120 OPEN #1:"EXPMEM2",RELATIVE,UPDATE,DISPLAY,VARIABLE 80
```

Then continue on as you normally would.

If you want to store both data and assembly language routines at the same time do this:

```
100 CALL INIT
110 CALL LOAD(-24574,-16)
120 OPEN #1:"EXPMEM2"
130 CALL LOAD ("DSK1.ASSM1")
140 CALL LOAD ("DSK2.ASSM2")
150 CALL LINK ("START")
160 REM CONTINUE REST OF PROGRAM
```

In the above example the 24 K of high-memory was saved for use as a DATA file (DIS/VAR 80 format) then the assembly routines were loaded. The computer will look for the best place to put the routines and will adjust the pointer accordingly. After the routines are loaded, a LINK statement starts the first routine and off we go.

If that's not enough for you, you can also use the MINI-MEMORY for 4K more of storage of assembly routines! Now that's 16K of program space, 12K of assembly routine space!

****

THESE ARE ALL OF THE PEEKS & POKES I HAVE COME ACROSS FOR USE WITH X-BASIC AND 32K MEMORY EXPANSION (BE SURE TO DO A "CALL INIT"). THE P & Q VARIABLES ARE USED FOR "PEEK" - THE NUMBERS ARE FOR."POKE" OR "LOAD". IF YOU KNOW OF ANY OTHERS PLEASE LET ME KNOW AND I WILL ADD THEM IN.

****

ADDRESS	, VALUE(S)	MEANING IN EXTENDED BASIC
		CALL VERSION(X) IF X=100 100= NEWEST VERSION OF X/B CART
8192	, P	USE (PEEK,P) IF P<> 70 OR <>121 THEN DO A CALL INIT
8194	,	FIRST FREE ADDRESS IN LOW MEMORY
8196	,	LAST FREE ADDRESS IN LOW MEMORY
-28672	, P	P=0 SPEECH NOT ATTACHED P=96 OR P=255 SPEECH IS ATTACHED
-31572	, 0 TO 255	VARY KEYBOARD RESPONSE
31740	, P , Q	PUT IN DIFFERENT TO CHANGE BEEPS,WARNINGS, ETC
31744	, 0 TO 15	CONTINUATION OF LAST SOUND (0=LOUD AND 15=SOFT)
-31748	, 0 TO 255	CHANGE THE CURSOR FLASHING AND RESPONSE TONE RATES
-31788	, 160	BLANK OUT THE SCREEN (MUST PUSH A KEY TO ACTIVATE)
	, 224	NORMAL OPERATION
	, 226	DOUBLE SIZE SPRITES
	, 227	MAGNIFIED & DOUBLE SIZED SPRITES
	, 232	MULTICOLOR MODE (48 BY 64 SQUARES)
-31794	, P	TIMER FOR CALL SOUND (COUNTS FROM 255 TO 0)
-31804	, X , Y	RETURN TO THE TITLE SCREEN (USE "PEEK (2,X,Y)")

ADDRESS	VALUE(S)	MEANING IN EXTENDED BASIC
	32	DISABLE SOUND (USE NEG DUR FOR CONTINOUS SOUND)
	48	DISABLE SOUND & QUIT KEY
	64	DISABLE AUTO SPRITE MOTION
	80	DISABLE SPRITES & QUIT KEY
	96	DISABLE SPRITES AND SOUND
	128	DISABLE ALL THREE
-31808	P , Q	DOUBLE RANDOM NUMBERS (0 TO 255) NEED "RANDOMIZE"
-31860	4	GO FROM EX-BASIC TO CONSOLE BASIC (NEED "NEW")
-31866	P , Q	END OF CPU PROGRAM ADDRESS (P*256+Q)
-31868	0	NO "RUN" OR "LIST" AFTER "BREAK" IS USED
	0 , 0	TURNS OFF THE 32K MEMORY EXPANSION
	255 , 231	TURNS ON THE 32K MEMORY EXPANSION
-31873	3 TO 30	SCREEN COLUMN TO START AT WITH A "PRINT"
-31877	P	P&32 = SPRITE COINCIDENCE P&64 = 5 SPRITES ON A LINE
-31878	P	HIGHEST NUMBER SPRITE IN MOTION (0 STOPS ALL)
-31880	P	RANDOM NUMBER (0 TO 99) NEED "RANDOMIZE"
-31884	0 TO 5	CHANGE KEYBOARD MODE (LIKE "CALL KEY(K,...)")
-31888	63 , 255	DISABLE ALL DISK DRIVES (USE "NEW" TO FREE MEMORY)
	55 , 215	ENABLE ALL DISK DRIVES (USE "NEW" TO FREE DRIVES)
-31931	0	UNPROTECT X-B PROTECTION
	2	SET "ON WARNING NEXT" COMMAND
	4	SET "ON WARNING STOP" COMMAND
	14	SET "UNTRACE" COMMAND
	15	SET "UNTRACE" COMMAND & "NUM" COMMAND
	16	SET "TRACE" COMMAND
	64	SET "ON BREAK NEXT" COMMAND
	128	PROTECT X/B PROGRAM
-31952	P	PEEK P=55 THEN 32K EXPANSION MEMORY IS OFF <>55 MEANS ON
-31962	32	RETURN TO THE TITLE SCREEN
	255	RESTART X/B W/DSK1.LOAD
-31974	P , Q	END OF VDP STACK ADDRESS (P*256+Q)
-32112	8	SEARCHES DISK FOR ?
-32114	2	RANDOM GARBAGE
	13	SCREEN GOES WILD
	119	PRODUCE LINES
-32116	2	RANDOM CHARACTERS ON SCREEN
	4	GO FROM X/BASIC TO BASIC
-32187	0	UNPROTECT XB PROGRAM
	2	SET "ON WARNING NEXT" COMMAND
	4	SET "ON WARNING STOP" COMMAND
	9	SET 0 LINE NUMBER
	14	SET "UNTRACE" COMMAND
	15	SET "UNTRACE" COMMAND & "NUM" COMMAND
	16	SET "TRACE" COMMAND
	64	SET "ON BREAK NEXT" COMMAND
	128	PROTECT XB PROGRAM
-32188	1	CHANGE COLOR AND RECEIVE SYNTAX ERROR
	127	CHANGE COLOR AND RECEIVE BREAKPOINT
-32630	128	RESET TO TITLE SCREEN
-32699	0	UNPROTECT XB PROGRAM
	2	SET "ON WARNING NEXT" COMMAND
	4	SET "ON WARNING STOP" COMMAND
	14	SET "UNTRACE" COMMAND
	15	SET "UNTRACE" & "NUM" COMMAND
	16	SET "TRACE" COMMAND
	64	SET "ON BREAK NEXT"
	128	PROTECT XB PROGRAM
-32700	0	CLEARS CREEN FOR AN INSTANT
-32729	0	RUN "DSK1.LOAD"
-32730	32	RESET TO TITLE SCREEN
-32961	51	RESET TO TITLE SCREEN
	149	SETS "ON BREAK GOTO" LOCKS SYSTEM

THE FOLLOWING LOADS REQUIRE E/A OR MM

ADDRESS	VALUE(S)	MEANING
784	P	USE POKEV(784,P) (WHERE P IS 16 TO 31) CHANGES BACKGROUND COLOR OF CURSOR
-24574	8	I THINK THIS ALLOWS THE MINI-MEM TO USE THE 24K FOR STORAGE
-30945	0	WHITE EDGES
-32272	0 , "" ,	-30945 , 0) WILL PUT YOU IN TEXT MODE
-32766	0	BIT MAP MODE
-32768	0	GRAPHICS (NORMAL MODE)
-32280	0	MULTI-COLOR MODE
-32352	107	WILL BLANK THE SCREEN, ANY KEY PRESS WILL RESTORE

* PASCAL LOADS

14586	0 , 0	THIS ALLOWS YOU TO DO A "RUN-TIME WARM START" FROM PASCAL TO BASIC.

End of file

CITY/STATE	SYSOP	A/C Phone #	Hours	TYPE OF PROGRAM
Toms River NJ	Jeanete Shader	201 929 8161 -	24 hours	A home-brew BBS prog
New Haven CT	Matt Sinclair	203 777 8588 -	24 hours	TIBBS
Seattte WA	J.R.-XCHANGE	206 542-8529 -	24 hours	TIBBS
Portland ME	Mark Rideout	207 797-5690 -	24 hours	TIBBS
Veazie, ME	Eric Benton	*207 945-5709 -	24 hours	TIBBS
Dallas TX	Bill Kauth	*214 353-0502 -		
Mesquite TX	Keith Hughey	214 681-5729 -	24 hours	Dungeon Keep TIBBS
Dallas TX	R.A.Fleetwood	214 995-3054 -		FLUG TIBBS (TI)
Phil. PA	Philly TIBBS	215 927-6432 -	24 hours	
Reading PA	Gene Deisher	215 929-5348 -		TICOMM BBS
Cleveland Oh		216 289-7311 -	24 hours	TIBBS
Washington DC	Phil	301 434-0117 -	24 hours	TIBBS
Newark DE	Dlwre Vly U.G.	302 322-3999 -		TIBBS
Colo Spg CO	John Williams	*303 574-5762 -	24 hours	TIBBS 300/1200
Ocoee FL	Dennis Neubauer	305 877-6546 -	24 hours	TIBBS
Ft Laud. FL	Ed O'Shaunessy	305 583-4343 -	24 hours	
West Palm FL	Dave Sholdar	305 793-8050 -		
Orlando FL.	Brian Delany	305 831-5990 -	24 hours	TIBBS
Hazel Pk MI	Thom Thibodeau	313 544-0714 -		TIBBS
Detroit MI	Computer corner	313 544-7788 -	24 hours	TIBBS
Clawson MI	Craig Barton	313 751-1119 -		TIBBS
Taylor, MI	Jeff Thrush	313 292-6147 -		TIBBS
St Louis MO	Ron Courtois	314 878-4289 -		TIBBS
Wichita KS	Jerry McClusky	*316 681-3167 -		TIBBS
Lake Chas LA	Bayou TIBBS	318 474-6144 -		TIBBS
Atlanta #1	Ralph Fowler	*404 425-5254 -		TIBBS
Marietta, GA	Mreta TIBBS	404 955-2731 -		TIBBS
Atl GA	Ham Radio TIBBS	*404 363-1640 -	24 hours	TIBBS
Atl GA	Atl 99/4A U.G.	404 366-1914 -	24 hours	TIBBS
Pittsburgh PA.	Computer Bug	412 882-0717 -	24 hours	TIBBS
Appleton WI	Marc Schmidt	414 739-5380 -	24 hours	TIBBS
Milwaukee WI	Dan Gunia	414 649-TEAM -	24 hours	TIBBS
Greenbay WI	D. Pfotenhauer	414 437-6930 -		TIBBS
Redwood City	Bay Cities	415 364-8517 -		TIBBS
Gresham OR	Mike Werstlen	503 661-0408 -	24 hours	TIBBS
Knoxvlle IA	Keith Jamison	515 842-2104 -		
Albany NY	Dick Ferrigan	518 765-4993 -	24 hours	TIBBS
Biloxi, MS	Larry Levy	601 392-8717 -		
BC Canada 3P-7A	WD/24	*604 531-6423 -	Weekends	TIBBS
Mound, MN	Mark Ziesmer	612 472-3490 -	24 hours	TIBBS
Columbus OH	Spirit of 99	614 451-0880 -	24 hours	Spirtit of 99 TIBBS
Knoxville TN	Dick Tracey	*615 691-9558 -		
Chatt. TN	Mines of Moria	615 267-1721 -	24 hours	TIBBS
Boston MA	Elite TIBBS	617 367-6341 -	24 hours	ELITE TIBBS
San Diego CA	Irish Input	619 276-3173 -		
San Diego CA	SCCG TIBBS	619 282-3525 -	24 hours	SCCG TIBBS
Las Vegaas NA	John Martin	702 648-1247 -	24 hours	TICOMM
Wash. DC	PHIL	*703 631-8772 -	24 hours	TIBBS
Charlotte NC	Bits N Chips	704 376-8124 -		
Char. NC	Queen City TIBBS	704 541-3776 -		
Houston TX	H.U.G.TIBBS	*713 487-5530 -	changes numbers frequently	
Houston TX	Phoenix TIBBS	*713 537-0741 -	24 hours	Good TIBBS
Fontana CA	Peter Covert	714 350-8583 -		
Tonawanda NY	Peter Testa	716 837-6635 -		TIBBS
Va Bch. VA	T.U.G TIBBS	804 486-1484 -		TIBBS
Bradenton FL	Action-Link	*813-747-2081 -	24 hours	ACTION TIBBS
	10 meg hard disk			

```
Tampa FL         Mike Carroll      813 677-0718 - 24 hours   TIBBS
Sheldon,IL       Wayne Burgess     815 429-3533 -            TIBBS
Pasadena CA                        818 578-0678 - Nights only
Sylmar, CA       <WCN>            *818 361-9294 -
Memphis, TN      Memphis U.G.      901 357-5425 - 24 hours   TIBBS
Dartmouth        Terry Atkinson    902 434-3121 -
Nova Scotia
Daytona FL       Dave Taylor       904 255-0326 - 24 hours   TICOMM
Savanna GA.      David Smith       912 354-0508 -            TIBBS
Middletwn NY                       914 343-5076 - Weekends   TIBBS
Sacramento Ca    SAC TIBBS        *916 927-3012 - 24 hours   TIBBS
Durham NC        Bull City TIBBS   919 383-8707 -            TIBBS
Wnstn-Slm NC                       919 723-2415 - 24 hours   TIBBS
Raleigh NC       Amnon Nissan     *919 851-8460 -
Orphanage(994a & PCjr)            *214 276-7832 - 24 Hours   TURBO BBS
                 RANDY BAXTER                   - BBS Runs on IBM PCjr and has
Nice BBS !                                      - X-MODEM TI file transfers.
```

CHAPTER 11 - TI 99 PIN OUTS

The following pin connections are in reference to the figures on the next page. In most cases, I will tell you the top, bottom or side that the view is of.

GROM PORT CONNECTOR (36 pin)

PIN NUMBER	SIGNAL
1	RESET
2	GROUND
3	DATA LINE 7
4	CRU CLOCK
5	DATA LINE 6
6	CRU INPUT
7	DATA LINE 5
8	CRU OUT & ADDRESS LINE 15
9	DATA LINE 11
1Ø	ADDRESS LINE 13
11	DATA LINE 3
12	ADDRESS LINE 12
13	DATA LINE 2
14	ADDRESS LINE 11
15	DATA LINE 1
16	ADDRESS LINE 1Ø
17	DATA LINE Ø
18	ADDRESS LINE 9
19	+ 5 VOLTS
2Ø	ADDRESS LINE 8
21	GROM SELECT SIGNAL
22	ADDRESS LINE 7
23	ADDRESS LINE 14
24	ADDRESS LINE 3
25	DATA BUS IN
26	ADDRESS LINE 6
27	GROM CLOCK
28	ADDRESS LINE 5
29	-5 VOLTS
3Ø	ADDRESS LINE 4
31	GROM READY
32	WRITE ENABLE LOW
33	GROM VSS
34	ROMG LOW
35	DATA LINE 4
36	GROUND

PERIPHERAL PORT (44 PIN)

PIN NUMBER	SIGNAL
1	+5 VOLTS
2	SBE
3	RESET
4	EXT INTERRUPT
5	ADDRESS LINE 5
6	ADDRESS LINE 1Ø
7	ADDRESS LINE 4
8	ADDRESS LINE 11
9	DB INPUT
1Ø	ADDRESS LINE 13
11	ADDRESS LINE 12
12	READY/HOLD LINE
13	LOAD
14	ADDRESS LINE 8
15	ADDRESS LINE 3
16	ADDRESS LINE 14
17	ADDRESS LINE 7
18	ADDRESS LINE 9
19	ADD. 15/CRU OUT
2Ø	ADDRESS LINE 2
21	GROUND REFERENCE
22	CRU CLOCK
23	GROUND REFERENCE
24	CPU CLOCK PHASE 3
25	GROUND
26	WRITE ENABLE
27	GROUND
28	MEMORY BLOCK EN.
29	ADDRESS LINE 6
3Ø	ADDRESS LINE 1
31	ADDRESS LINE Ø
32	MEMORY ENABLE
33	CRU INPUT
34	DATA LINE 7
35	DATA LINE 4
36	DATA LINE 6
37	DATA LINE Ø (MSB)
38	DATA LINE 5
39	DATA LINE 2
4Ø	DATA LINE 1
41	INST. AQUAS.(IAQ)
42	DATA LINE 3
43	-5 VOLTS
44	AUDIO IN

JOYSTICK PORT CONNECTOR

PIN NUMBER	SIGNAL
1	LEFT
2	FIRE BUTTON
3	UP
4	JOYSTICK 1 GND
5	NO CONNECTION
6	RIGHT
7	DOWN
8	JOYSTICK 2 GND
9	NO CONNECTION

CASSETTE PORT CONNECTOR

PIN NUMBER	SIGNAL
1	MOTOR CONTROL 1
2	MOTOR CONTROL 1
3	RECORD OUT GROUND
4	AUDIO INPUT
5	RECORD OUT
6	MOTOR CONTROL 2
7	MOTOR CONTROL 2
8	MAG IN
9	MAG OUT

VIDEO CONNECTOR

PIN NUMBER	SIGNAL
1	+12 VOLTS
2	GROUND
3	AUDIO OUT
4	COMPOSITE VIDEO
5	GROUND

RS232 PIN PARALLEL CONNECTOR

PIN NUMBER	SIGNAL
1	HANDSHAKE OUT
2	DATA LINE 15
3	DATA LINE 14
4	DATA LINE 13
5	DATA LINE 12
6	DATA LINE 11
7	DATA LINE 1Ø
8	DATA LINE 9
9	DATA LINE 8
1Ø	HANDSHAKE IN
11	SIGNAL GROUND
12	1Ø OHM PULLUP TO +5 VOLTS
13	SPARE INPUT BIT
14	SPARE OUTPUT BIT
15	1 K PULL UP TO +5 VOLTS
16	LOGIC (SIGNAL) GROUND

RS232 SERIAL CONNECTOR

PIN NUMBER	SIGNAL
1	PROTECTIVE GND.
2	TRANSMIT DATA
3	RECEIVED DATA
4	NOT USED
5	CLEAR TO SEND (1)
6	+12V PULL UP
7	SIGNAL GROUND
8	CARRIER DET (1)
9	NOT USED
1Ø	NOT USED
11	NOT USED
12	CARRIER DET (2)
13	CLEAR TO SEND (2)
14	TRANSMIT DATA (2)
15	NOT USED
16	RECEIVED DATA (2)
17	NOT USED
18	NOT USED
19	DTR PORT 2
2Ø	DTR PORT 1
21-25	NOT USED

CHAPTER 12 - ASCII CHARACTERS AND VALUES

COLOR	VALUE
TRANSPARENT	1
BLACK	2
MED. GREEN	3
LT. GREEN	4
DARK BLUE	5
LT. BLUE	6
DARK RED	7
CYAN	8
MED. RED	9
LT. RED	1Ø
DK. YELLOW	11
LT. YELLOW	12
DK. GREEN	13
MAGENTA	14
GRAY	15
WHITE	16

CHARACTER SETS

SET	ASCII CODES	
Ø	3Ø-31	
1	32-39	
2	4Ø-47	
3	48-55	
4	56-63	
5	64-71	
6	72-79	
7	8Ø-87	
8	88-95	
9	96-1Ø3	
1Ø	1Ø4-111	
11	112-119	
12	12Ø-127	
13	128-131	
14	132-143	
15	144-151	-\ USE IN
16	152-159	-/ BASIC

ASCII CHARACTER CODES

The defined characters on the TI-99/4A Computer are the standard ASCII characters for codes 32 through 127. The following chart lists these characters and their codes.

ASCII CODE	CHARACTER	ASCII CODE	CHARACTER	ASCII CODE	CHARACTER
32	(space)	65	A	97	A
33	! (exclamation point)	66	B	98	B
34	" (quote)	67	C	99	C
35	# (number or pound sign)	68	D	100	D
36	$ (dollar)	69	E	101	E
37	% (percent)	70	F	102	F
38	& (ampersand)	71	G	103	G
39	' (apostrophe)	72	H	104	H
40	((open parenthesis)	73	I	105	I
41	) (close parenthesis)	74	J	106	J
42	* (asterisk)	75	K	107	K
43	+ (plus)	76	L	108	L
44	, (comma)	77	M	109	M
45	− (minus)	78	N	110	N
46	. (period)	79	O	111	O
47	/ (slant)	80	P	112	P
48	0	81	Q	113	Q
49	1	82	R	114	R
50	2	83	S	115	S
51	3	84	T	116	T
52	4	85	U	117	U
53	5	86	V	118	V
54	6	87	W	119	W
55	7	88	X	120	X
56	8	89	Y	121	Y
57	9	90	Z	122	Z
58	: (colon)	91	[(open bracket)	123	{ (left brace)
59	; (semicolon)	92	\ (reverse slant)	124	¦
60	< (less than)	93	] (close bracket)	125	} (right brace)
61	= (equals)	94	∧ (exponentiation)	126	~ (tilde)
62	> (greater than)	95	_ (line)	127	DEL (appears on screen as a blank.)
63	? (question mark)	96	(grave)		
64	@ (at sign)				

Function Key Codes

Codes TI-99/4 & BASIC Modes	Pascal Mode	Function Name	Function Key
1	129	**AID**	**FCTN 7**
2	130	**CLEAR**	**FCTN 4**
3	131	**DEL**ete	**FCTN 1**
4	132	**INS**ert	**FCTN 2**
5	133	**QUIT**	**FCTN =**
6	134	**REDO**	**FCTN 8**
7	135	**ERASE**	**FCTN 3**
8	136	**LEFT** arrow	**FCTN S**
9	137	**RIGHT** arrow	**FCTN D**
10	138	**DOWN** arrow	**FCTN X**
11	139	**UP** arrow	**FCTN E**
12	140	**PROD'D**	**FCTN 6**
13	141	**ENTER**	**ENTER**
14	142	**BEGIN**	**FCTN 5**
15	143	**BACK**	**FCTN 9**

Control Key Codes

Codes BASIC Mode	Pascal Mode	Mnemonic Code	Press	Comments
129	1	SOH	CONTROL A	Start of heading
130	2	STX	CONTROL B	Start of text
131	3	ETX	CONTROL C	End of text
132	4	EOT	CONTROL D	End of transmission
133	5	ENQ	CONTROL E	Enquiry
134	6	ACK	CONTROL F	Acknowledge
135	7	BEL	CONTROL G	Bell
136	8	BS	CONTROL H	Backspace
137	9	HT	CONTROL I	Horizontal tabulation
138	10	LF	CONTROL J	Line feed
139	11	VT	CONTROL K	Vertical tabulation
140	12	FF	CONTROL L	Form feed
141	13	CR	CONTROL M	Carriage return
142	14	SO	CONTROL N	Shift out
143	15	SI	CONTROL O	Shift in
144	16	DLE	CONTROL P	Data link escape
145	17	DC1	CONTROL Q	Device control 1 (X-ON)
146	18	DC2	CONTROL R	Device control 2
147	19	DC3	CONTROL S	Device control 3 (X-OFF)
148	20	DC4	CONTROL T	Device control 4
149	21	NAK	CONTROL U	Negative acknowledge
150	22	SYN	CONTROL V	Synchronous idle
151	23	ETB	CONTROL W	End of transmission block
152	24	CAN	CONTROL X	Cancel
153	25	EM	CONTROL Y	End of medium
154	26	SUB	CONTROL Z	Substitute
155	27	ESC	CONTROL .	Escape
156	28	FS	CONTROL ;	File separator
157	29	GS	CONTROL =	Group separator
158	30	RS	CONTROL 8	Record separator
159	31	US	CONTROL 9	Unit separator

CHAPTER 13 - THIRD PARTY ADDRESSES

A semi-complete listing of magazines, newsletters, and mail-order companies devoted to the Texas Instruments 99/4a Home Computer.

name	address	description	cost
Micropendium	MICROPENDIUM P.O. BOX 1343 ROUNDROCK, TEXAS 78680	A magazine TOTALLY devoted to the T.I. 99/4a. Every issue gets better. Includes such things as reviews, articles, how-tos, editorial pages, and the best collection of advertising you'll find anywhere.	$18.50 first class $15.00 third class 12 ISSUES
Super 99 Monthly	BYTEMASTER COMP. SERVICES 171 MUSTANG ST. SULPHUR, LA 70633	Another magazine for 99/4a's only. Really worth checking out.	$16.00 first class $12.00 third class 12 ISSUES
Computer Shopper	COMPUTER SHOPPER 407 WASHINGTON AVE P.O. BOX F TITUSVILLE, FL 32781	While not a 99/4a only magazine, Randy Holcombs T.I. column, plus all the ads, classifieds, BBS numbers, etc. make this one a must for any library. (a monthly with ~300 pages!)	$18.00 A YEAR 12 ISSUES
Millers Graphics Newsletter	MILLERS GRAPHICS 1475 W.CYPRESS AVE SAN DIMAS, CA 91773	MILLERS GRAPHICS has become known as one of the best sources of quality programs for the 99/4a around. Their newsletter cannot be beat for hardcore tech. and prog.	$12.50 A YEAR
mail order companies that specialize in the T.I. 99/4A			
TEX-COMP	P.O. BOX 33084 GRANADA HILLS,CA 91344	BIGGEST SELECTION OF /4A SOFTWARE AND HARDWARE.	(818) 366-6631 ORDERS ONLY
TENEX COMPUTER EXPRESS	P.O. BOX 6578 SOUTHBEND, IN 46660	NICE SELECTION OF ALMOST EVERYTHING FOR THE /4A -FREE CATALOG-	(800) 348-2278 ORDERS (219) 259-7051 QUESTIONS

TRITON	P.O. BOX 8123 SAN FRANCISCO, CA 94128	FAIR SELECTION OF STUFF FOR THE /4A. LOTS OF EDUCATIONAL SOFTWARE.-FREE CATALOG-	(8ØØ) 227-69ØØ
TRITON	P.O. BOX 8123 SAN FRANCISCO, CA 94128	FAIR SELECTION OF STUFF FOR THE /4A. LOTS OF EDUCATIONAL SOFTWARE.-FREE CATALOG-	(8ØØ) 227-69ØØ
TIGERCUB SOFTWARE	156 COLLINGWOOD AVE. COLUMBUS, OHIO 43213	JIM PETERSON IS THE OWNER OF THIS COMPANY AND PROVIDES A GREAT QUANTITY OF SOFTWARE ALL OF WHICH HE HAS WRITTEN HIMSELF. HE ALSO WRITES A NEWSLETTER FOR USERS GROUPS AT NO CHARGE. HE IS A DRIVING FORCE BEHIND THE /4A -CATALOG $3 REFUNDABLE-	(614) 235-3545

FAST TERM Paul Charlton 1110 Pinehurst Court Charlottesville, VA 22901
Simply, THE BEST TERMINAL EMULATOR IN THE WORLD!

SPRITE BUILDER John Taylor 2170 Estaline Drive Florence, AL. 35630 XB graphics generating program with assembly language routines for speed at crucial places. Includes a full disk of preformed graphics.

PILOT99 Thomas Weithofer 1000 Harbury Drive Cincinnati, OH. 45220 An ENTIRE language for the TI that is the simplest programming language known to us (or anyone else!) **********TWO SSSD DISKS REQUIRED**********

MASTER CATALOG Mack McCormick 215 A Yorktown, Ft. Lee, Virginia 23801 A 100% assembly language disk catalog program that is super fast; handles up to 2000 different disk files.

TI SORT David R. Romer 213 Earl St. Walbridg, Oh. 43465 An exellent program that sorts a display variable 80 file in either a quick sort or a shell sort.

EASYSPRITE Tom Freeman 515 Alma Real Dr.,Pacific Palisades, CA 90272 An extremely fast XB program with assembly routines to create graphics & sprites with easy cursor control & saving for program insertion.

DISASSEMBLER Marty Kroll 218 Kaplan Avenue Pittsburg, P. 15227 Super-fast disassembler, 100% assembly and full featured.

XB_UTILITY Silver Wolf Software P.O. Box 4242 Santa Rosa, Ca. 95402. This is a great utilities program that lets you save any screen for instant recall, and gives an on-screen ascii instant referance. Many more features.

TECHIE BBS Monty Schmidt 121 N. Blair, Madison, WI. 53703 Freeware BBS system for the 99/4A. After 8/15/85: 525 Wingra St., Madison, Wi 53714

COMPACTOR Monty Schmidt (see above) Assembly langauge program that takes an uncompressed D/F80 AL program and will compress to about 2/3 the disk space and yield faster load times.

UNCOMPACTOR Monty Schmidt (see above) Opposite of above.

PRO 99er BBS Mark Hoogendoorne 21 Long Street, Burlington, MA. 01803 TI BBS system with TRUE TE2 transfer capabilites.

DISK MANAGER Todd Kaplan, 5802 N. Western Apt. 3S, Chicago, IL. 60659 INCREDIBLE Disk ae on disk; forget TI'S DM2

TRIVIA 99er Robert L. Wessler 4300 Frazier Fort Worth, Texas 76115. A Trivia game in XBASIC that has a nice data base and exellent doc's. Send a blank initialized disk or $ 7.00 and he will provide the disk, mailer and postage. Extra disk lables are available for users groups for the asking.

TRIVIA and GRAPHICS DEMO programs by Danny Cox 1861 The Elms Avenue, Memphis, TN. 38127. Six graphics demos in Ex BASIC and an up-to four player trivia game. Both require Ex. BASIC.

TOMB OF DEATH John Behnke, 5755 W. Grace, Chicago, Ill. 60634. Requires Tunnels of Doom cartridge, some new graphics.

ASSAULT THE CITY, John Behnke (see above).

FAST FORTH, Tim Curran, 4153 Four Pole Road, Huntington, WV 25701. XBasic Loader, fast editor, fast editor locator, 40 column auto-repeat.

A HANDY DANDY TI-WRITER USERS REFERENCE GUIDE

SUBMITTED BY BOB STEPHENS

The following handy TI-WRITER commands are reprinted for the June issue of the 99'er News published by the TI Users Group of Will County, Romeoville, Il. This puts the most used commands on one page for handy access at your computer.

EDITOR COMMAND	FCTN	CTRL	EDITOR COMMAND	FCTN	CRTL	EDITOR COMMAND	FCTN	CTRL
Back tab		T	Ins. Blank line	8	O	Quit	=	
Beginning/line		V	Insert character	2	G	Reformat		2orR
Command/escape	9	C	Last paragrapph		6orH	Right arrow	D	D
Delete character	1	F	Left arrow	S	S	Roll down	4	A
Del. end of line		K	Left margin rel.		Y	Roll up	6	B
Delete line	3	N	New page		9orP	Screen color		3
Line #'s(on/off)	0		New paragraph		8orM	Tab	7	I
Down arrow	X	A	Next paragraph		4OrJ	Up arrow	E	E
Duplicate line		5	Next window	5		Word tab		7orW
Home cursor		L	Oops!		1orZ	Word wrap/fixed		0

```
Load files: LF (enter) DSK1.FILENAME  (load entire file)
            LF (enter) 3 DSK1.FILENAME  (merges filename with data in memory
                                         after line 3)
            LF (enter) 3 1 10 DSK1.FILENAME  (lines 1 thru 10 of filename are
                                              merged after line 3 in memory)
            LF (enter) 1 10 DSK1.FILENAME  (loads lines 1 thru 10 of filename)

Save files: SF (enter) DSK1.FILENAME (save entire file)
            SF (enter) 1 10 DSK1.FILENAME (save lines 1 thru 10)

Print Files:PF (enter) PIO (prints control characters and line numbers)
            PF (enter) C PIO (prints with no control characters)
            PF (enter) L PIO (prints 74 characters with line numbers)
            PF (enter) F PIO (prints fixed 80 format)
            PF (enter) 1 10 PIO (prints lines 1 thru 10)
NOTE: The above assumes PIO. DSK1.FILENAME, and RS232 are also valid!
      To cancel the print command press FCTN 4.

Delete file:DF (enter) DSK1.FILENAME

Setting Margins and Tabs: (16 tabs maximum)
      L - Left margin     R - Right margin     I - Indent       T - Tab
         Use ENTER to execute or COMMAND/ESCAPE to terminate command.

Recover Edit: RE (enter) Y or N

Line move:  M (enter) 2 6 10  (moves lines 2 thru 6 after line 10)
            M (enter) 2 2 10  (moves line 2 after line 10)

Copy:       same as move except use C instead of M.

Find String: FS (enter) /string/  (will look for string in entire file)
             FS (enter) 1 15 /string/  (will look for string in lines 2 thru 15)

Delete:      D (enter) 10 15 (deletes lines 10 thru 15 in memory)
```