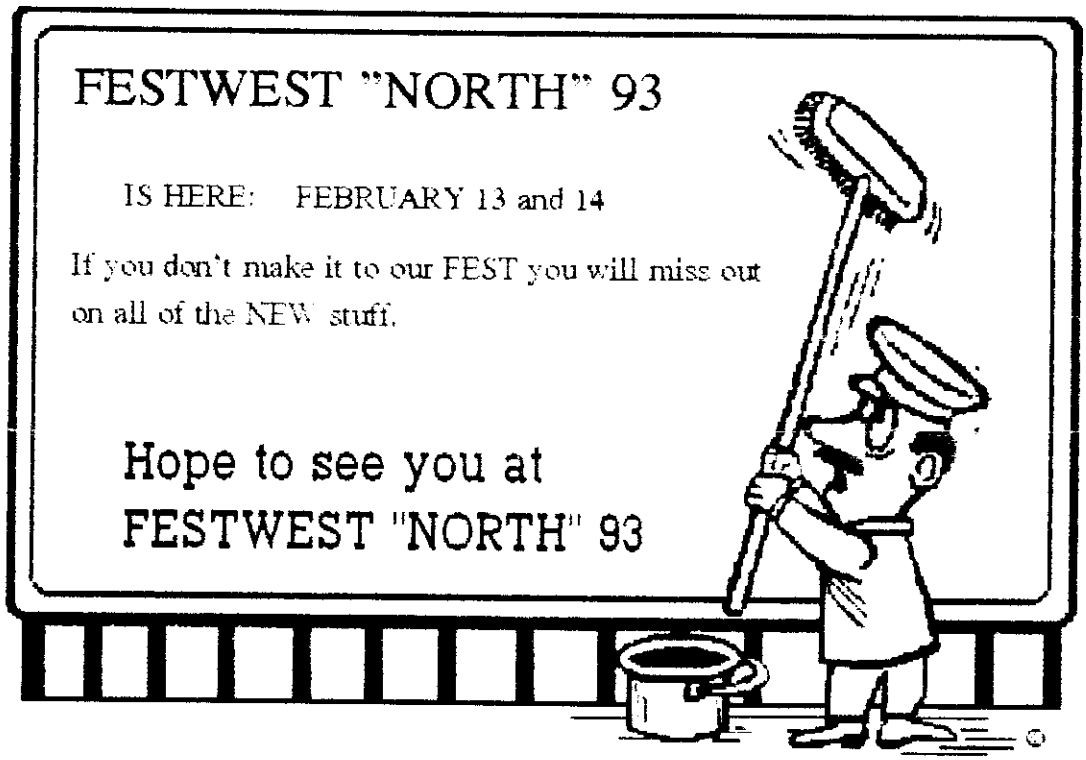
OTIUG and SLAVES JANUARY and FEBRUARY NEWSLETTER 1992/1993



Come to FESTWEST and see the NEW WATCHAMACALIT It is used on the MYARC HFDC Clock

Chip. What it does is let you have an accurate time. It is also Battery Backed. You can tune your clock with in

seconds. It comes with a clock program to use to set the clock.

You can pick one up at the Ogden Ti User Groups table.. They sell for $29.95 plus $3.00 shipping and handling.

BUT if you pick one up at FESTWEST "NORTH" 93, You will only have to pay $24.95 Plus Tax.

FESTWEST 'NORTH' 93

SUPPORTING: TEXAS INSTRUMENTS TI-99/4A AND THE 9640 GENEVE, AND COMPATIBLES

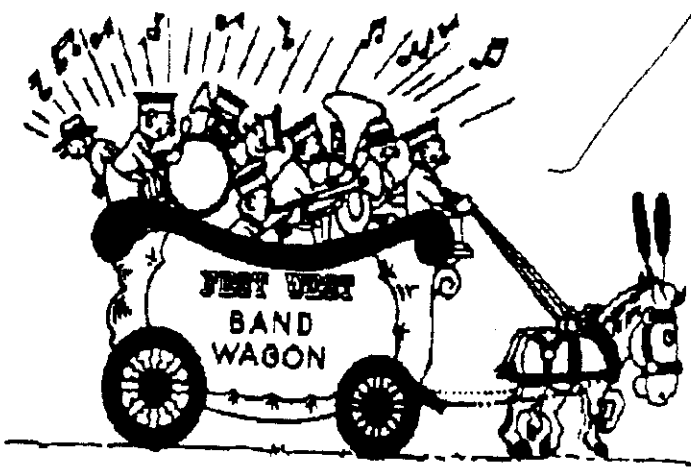
HOSTED BY:
OTTUG - OGDEN TI USERS GROUP
AND THE
SLAVES - SALT LAKE AND VALLEY
Users Group

LOCATION: HOWARD JOHNSON HOTEL
122 West South Temple
Salt Lake City, Utah 84101
(801) 521-0130
TOLL FREE (800) 366-3684
FAX (801) 322-5057

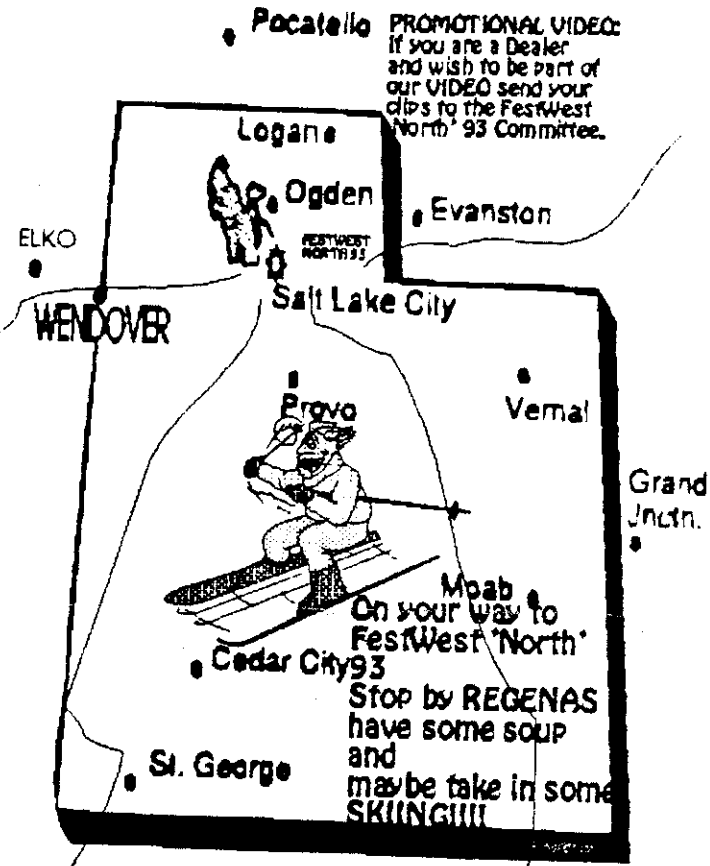
DATE/TIME:
SATURDAY - FEBRUARY 13 1993
9am TO 5pm
SUNDAY - FEBRUARY 14 1993
9am TO 3pm

RATES:
$55.00 for 2 persons
$62.50 for 3 or 4 persons

When calling for reservations,
Please state, you are calling for
reservations for FestWest 'North' 93



C/O FestWest 'North' 93 Committee
1396 Lincoln Apt. B
Ogden, Utah 84404



SALT FLATS BBS
SYSOP: DAVE DEHEER

(801 394-0064
BAUD 300-1200-2400
24HRS PER DAY
CALL OUR BBS FOR
FURTHER UPDATES ON
FestWest 'North' 93

FEST-WEST "NORTH" '93

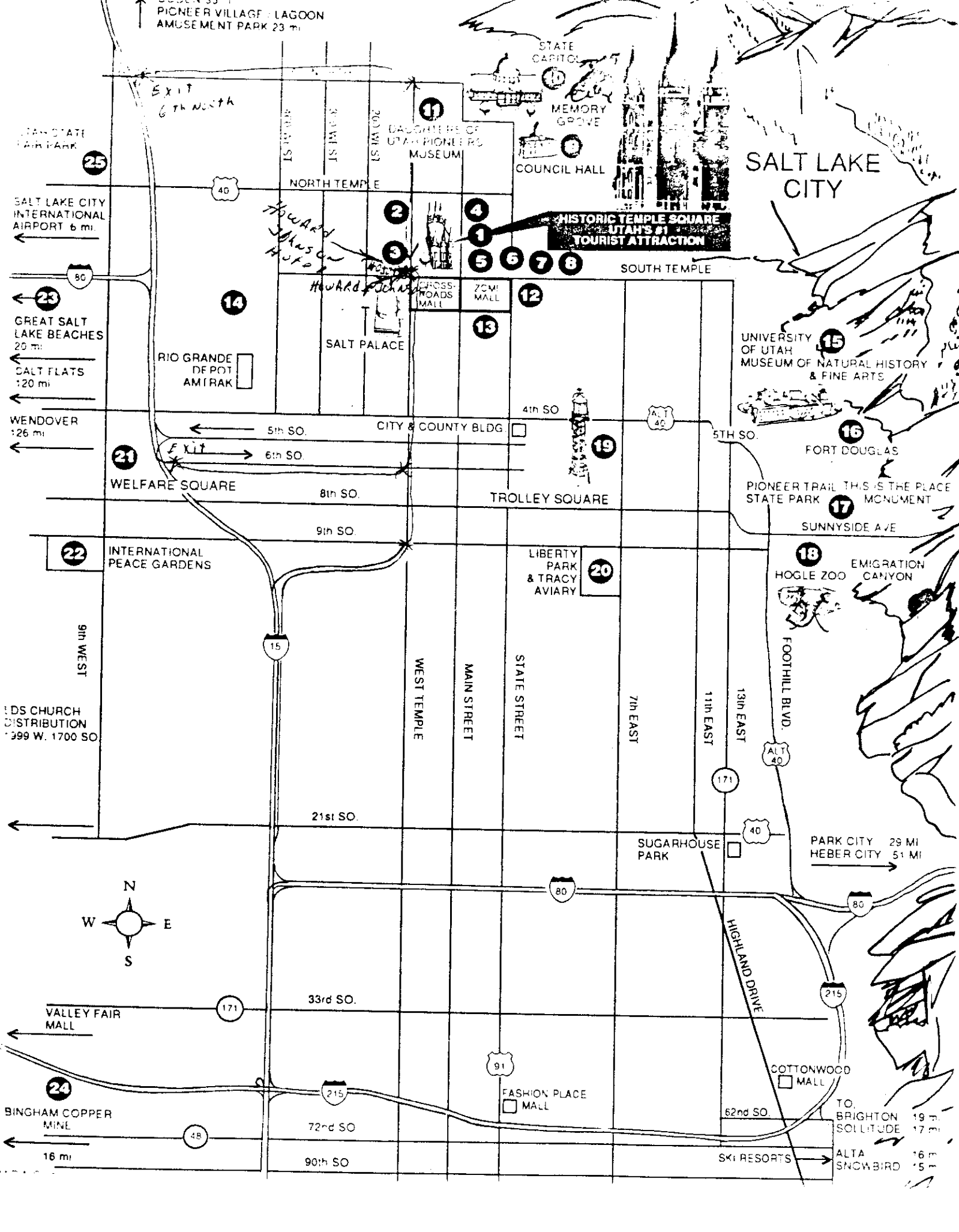
VACATION

Thursday, Feb. 11 --Visit REGENA
Friday, Feb. 12 --Ski Brian Head
Saturday, Feb. 13 --FEST-WEST
Sunday, Feb. 14 --FEST-WEST
Monday, Feb. 15 --Ski near SLC
Tuesday, Feb. 16 --Ski Brian Head

TI friends are invited to an open house at REGENA's home in Cedar City, 250 miles south of the FEST-WEST site in Salt Lake City. Stop in Thursday or Friday on your way to FEST-WEST. If it's cold, snowy weather, a crock pot of soup will be ready for you.

I can help you with skiing plans at Brian Head. Also--let's plan a ski trip near SLC on Monday!!

Write for details: REGENA
918 Cedar Knolls West
Cedar City, Utah 84720


PIONEER VILLAGE - LAGOON AMUSEMENT PARK 23 mi
STATE CAPITOL
MEMORY GROVE
COUNCIL HALL
DAUGHTERS OF UTAH PIONEERS MUSEUM
1ST WEST
1ST EAST
200 WEST
NORTH TEMPLE
SALT LAKE CITY
HISTORIC TEMPLE SQUARE
UTAH'S #1 TOURIST ATTRACTION
SOUTH TEMPLE
SALT LAKE CITY INTERNATIONAL AIRPORT 6 mi.
GREAT SALT LAKE BEACHES 20 mi
SALT FLATS 120 mi
WENDOVER 126 mi
CROSSROADS MALL
ZCMI MALL
SALT PALACE
RIO GRANDE DEPOT AMTRAK
UNIVERSITY OF UTAH MUSEUM OF NATURAL HISTORY & FINE ARTS
4th SO
5th SO.
CITY & COUNTY BLDG
5TH SO.
6th SO.
FORT DOUGLAS
WELFARE SQUARE
8th SO.
TROLLEY SQUARE
PIONEER TRAIL STATE PARK
THIS IS THE PLACE MONUMENT
9th SO.
SUNNYSIDE AVE
INTERNATIONAL PEACE GARDENS
LIBERTY PARK & TRACY AVIARY
HOGLE ZOO
EMIGRATION CANYON
9th WEST
WEST TEMPLE
MAIN STREET
STATE STREET
7th EAST
11th EAST
13th EAST
FOOTHILL BLVD.
LDS CHURCH DISTRIBUTION 1399 W. 1700 SO.
21st SO.
SUGARHOUSE PARK
PARK CITY 29 MI
HEBER CITY 51 MI
N
W
E
S
HIGHLAND DRIVE
33rd SO.
VALLEY FAIR MALL
COTTONWOOD MALL
FASHION PLACE MALL
62nd SO.
TO BRIGHTON 19 mi
SOLITUDE 17 mi
BINGHAM COPPER MINE
16 mi
72nd SO
90th SO
SKI RESORTS
ALTA 16 mi
SNOWBIRD 15 mi
Exit 6th North
Howard Johnson Hotel
Howard Johnson

SALT LAKE CITY ATTRACTIONS

NC = No Charge

Salt Lake street grid originates at Temple Square. As you move South, so do the streets (First South, Second South, etc.). As you move East (now and the Wasatch Mountains) or West (toward the Great Salt Lake), the streets do not run First East, Second East or Third West, Fourth West, etc.

1 HISTORIC TEMPLE SQUARE NC – This ten acre block is the heart of the world-wide Mormon Church and Salt Lake City's #1 tourist attraction. The Square is enclosed by a fifteen-foot wall which surrounds the imposing granite Temple, domed Tabernacle with its great organ, Assembly Hall, Seagull Monument, two Visitor's Centers, Law and Liberty plaques, compass, flag pole and The Nauvoo Bell. Winter hours 9-9 daily, summer 8:00 a.m.-10:00 p.m. daily. Organ recitals Monday-Saturday, 12 noon, and Sunday, 2:00 p.m., additional recitals at 2:00 p.m., June-August. Choir national broadcast Sunday, 9:30 a.m., rehearsal Thursday 8-9:30 p.m. Concert series Friday-Saturday, 7:30 Assembly Hall. Mormon Youth Symphony and Chorus rehearsals and performances as scheduled. Tours begin every few minutes at designated points. NC for all events and services on Historic Temple Square.

2 MUSEUM OF CHURCH HISTORY AND ART NC – The museum has historical Mormon artifacts and art in interpretive exhibits. Open Monday-Friday, 9-9, Saturday, Sunday, most holidays 10-7 (closed Easter, Thanksgiving, Christmas, New Years). 45 North West Temple.

3 FAMILY HISTORY LIBRARY NC – Largest repository of records in the world and open to everyone. Hours: Monday, 7:30 a.m.-6:00 p.m., Tuesday-Friday, 7:30 a.m.-10:00 p.m., Saturday, 7:30 a.m.-5:00 p.m. (Closed Sundays). 35 North West Temple.

4 CHURCH OFFICE BUILDING NC – World headquarters for the Church of Jesus Christ of Latter-Day Saints. Guided public tours are conducted from main lobby and 26th floor weekdays from 9 a.m.-4:30 p.m. except holidays. Saturdays, only from April-September, 9:30 a.m.-4:30 p.m. Closed Sundays.

5 THE BEEHIVE HOUSE NC – This colonial-style house served as the official residence of Brigham Young from 1854-1877. Atop the house, the wooden beehive – a Utah pioneer symbol of industry – gave the house its name. The adobe-brick structure, including a widow's walk, reflects New England tradition. The house has been restored to furnishings of the period. 9:30 a.m.-4:30 p.m. Monday-Saturday; 10:00 a.m.-1:00 p.m. on Sunday. (Extended weekday summer hours to 6:30 p.m.)

6 EAGLE GATE – This 6,000 pound eagle spanning State Street at South Temple, known as Eagle Gate, guards the entrance to what was once the private estate of Brigham Young.

7 BRIGHAM YOUNG GRAVE NC – 1st Avenue between State Street and "A" Street on the south side.

8 CATHEDRAL OF THE MADELEINE – Completed in 1909 in Roman-Gothic style, its interior features a beautiful series of German stained glass windows. 331 East South Temple.

9 COUNCIL HALL NC – Home of the Utah Travel Council, this sandstone structure, built in 1866, is the restored former capitol building of Utah's territorial government and the former Salt Lake City Hall. Tourist information available here plus self-guided tour. Open Monday-Friday, 8 a.m.-5 p.m. Capitol Hill. Phone (801) 538-1030.

10 STATE CAPITOL NC – Overlooking Salt Lake City since its completion in 1915, its pure Corinthian style resembles the National Capitol. Open seven days a week. Winter 8 a.m.-6 p.m.; Summer 6 a.m.-8 p.m. Self-guided tours.

11 DAUGHTERS OF UTAH PIONEERS MUSEUM NC – A treasure house of frontier relics, the museum was designed as a replica of the famous old Salt Lake Theatre. Open 9-5 Monday-Saturday. Also open Sunday 1-5 June, July & August. Capitol Hill.

12 HANSEN PLANETARIUM – Offers a wide variety of programs. For information on current shows and admission prices call (801) 538-2098. 15 South State Street.

13 PROMISED VALLEY PLAYHOUSE – 132 South State Street. Restored 19th century playhouse serves as a center for the arts. For information on performances call (801) 364-5696.

14 UTAH STATE HISTORY MUSEUM NC – Offers a variety of unusual and [illegible]e exhibits on the history of Utah. Open Monday-Friday, 8 a.m.-5 p.m. Located at 300 Rio Grande (450 West 300 South). For information call (801) 533-7037.

15 UNIVERSITY OF UTAH – A complete co-educational university founded in 1850, three years after the Mormons entered the Salt Lake Valley. Utah Museum of Fine Arts located in Art & Architecture Building. Open weekdays 10-5; Saturday-Sunday, 2-5. NC. Utah Museum of Natural History located on University Circle, 1400 East 2nd South. Open Monday-Saturday, 9:30-5:30, Sunday 12-5. Adults $2.00, children $1.00.

16 FORT DOUGLAS NC – Founded in 1862, the Fort includes an impressive group of red sandstone buildings, parade ground, cannons, officer's quarters and a post cemetery. Military Museum open daily, Tues.-Sat. 10 a.m.-4 p.m. Located east of and adjacent to University of Utah campus. Phone (801) 524-4154.

17 PIONEER TRAIL STATE PARK, THIS IS THE PLACE MONUMENT, OLD DESERET VILLAGE NC – Located near the mouth of Emigration Canyon, where Mormon pioneers entered the valley in 1847. Audio-visual display explains pioneer trek from Nauvoo, Illinois to Salt Lake. Winter hours 8-5. Visitor Center 10-5. Summer hours 8-8 and Visitors Center 10-6. Guided tours of village buildings, 12-5 from April through September. 2601 East Sunnyside Avenue.

18 HOGLE ZOO – Near the mouth of Emigration Canyon. Birds and animals in a natural setting. Winter, gate open 9-4:30. Summer, gate open 9-6. Miniature train, food concessions, picnic area. Adults $4.00, ages 5-14 $2.00, under 5 free, senior citizens $2.00. 2600 Sunnyside Avenue.

19 TROLLEY SQUARE – Salt Lake City's nationally famous shopping and entertainment center situated within renovated 1908 trolley barns. Open day and night with tasteful old-time flair, all indoors. Shops, restaurants, theaters, open market, art displays. Free parking. At 5th South and 7th East.

20 LIBERTY PARK, TRACY AVIARY – 80 acres for family recreation. Over 240 species of birds in the Aviary. Between 9th and 13th South, 5th to 7th East. For hours call (801) 596-5034.

21 WELFARE SQUARE NC – Located at 751 West 7th South Street, this facility provides assistance to the poor and needy. Visitors are shown the Bishops' Storehouse, a Deseret Industries operation, the milk processing plant, and the cannery. A short explanatory video is shown. A free van is available on request for transportation to and from Temple Square. Ample parking is available to those visiting by car. The Visitors Center is open from 10:00 a.m. until 4:00 p.m., Monday through Friday.

22 INTERNATIONAL PEACE GARDENS (Jordan Park) NC – Floral architecture and displays representative of countries throughout the world. May-November. 1000 South 900 West.

23 GREAT SALT LAKE – America's inland sea. Water you can float on. Beaches for swimming and sun-bathing, marina for boating.

24 BINGHAM COPPER MINE – Largest open-pit copper mine in the world. Operated by the Kennecott Copper Corporation, the mine is two and one half miles across and one half mile deep. For information phone (801) 569-6000.

25 UTAH STATE FAIRPARK – Home of the Utah State Fair. The Fairpark provides space for seminars, trade exhibits, conventions, meetings, reunions, livestock shows, auctions and parties. 155 North 1000 West. Phone (801) 538-8440.

Chris' Corner

Why the TI, part 4

For the past few months I have shared with you reasons why I still use the 99/4A. I feel that just because a computer is not state of the art, unless you need state of the art features, use what you have. Unfortunately, in the TI community very few powerful applications were ever produced that exploited its features. I realize that the decline of its popularity (coupled with the withdrawal of Texas Instruments support in 1983) is responsible for the lack of power programs. I, also, realize that if I devoted all my efforts to developing the most powerful programs possible for the 99/4A, those efforts would pale in comparison to having programs developed by a team of programmers. My objective, therefore, is to provide simple tools and concepts to aid you in expanding the use of your computer by becoming a member of the TI power users team.

Most of you know that I selected forth as my language of choice. The particular version of forth I use is Wycove forth for which I now have product rights. I chose forth because it has no competition as far as programming productivity is concerned, especially on the 99/4A. It offers access to the all the computer's resources: the video display processor, including the bit-map mode, the sound processor, the speech processor, ROM and GROM routines, and simple access to the peripherals. All of this at speeds which approached or equal assembly language speed. Furthermore, accessing the machine's hidden capabilities can be done as easily in forth as in BASIC.

When I initially purchased forth, I ordered two versions of forth: Wycove and TI. While forth allows the programmer the ability to re configure the language to assume any form, the amount of effort required to make TI forth as powerful as Wycove forth was not worth the effort nor could I have done so without extensive knowledge of assembly language programming and the inner workings of forth. Had I had the knowledge to make such changes at that time I would have, because if I had marketed any programs usin g Wycove forth, I would have had to pay a royalty fee. Now, that is not the case provided certain guidelines are met.

So, why am I trying to sell you on forth. Simple. I believe that over the next few months I will be able to provide you with enough easily digestible information, which will allow you to take control of your computer. In preparation for this information, the following TI BASIC program is offered for you to study. If you can master this program, you are on the way to power computing. I hope that you realize that this is a very simple database program.

```
100 CALL CLEAR

110 INPUT " Enter Disk Drive
 Number: " :DRIVE

120 PRINT

130 INPUT " Enter data base
Filename: ":FILENAME$

140 INPUT " Enter number of
names: " :MAXNAMES

150 PRINT

160 PRINT " TO QUIT  type 'q
uit' or 'QUIT'"

170 PRINT " for a data name
entry."
```

— See CHRIS' CORNER on page 7

— CHRIS' CORNER, from page 6

```
180 REM

190 REM

200 OPEN #1:"DSK"&STR$(DRIVE
)&"."&FILENAME$

210 FOR ENTRY = 1 TO MAXNAME
S

220 INPUT "Enter name: ":DB$

230 IF DB$="QUIT" THEN 270

240 IF DB$="quit" THEN 270

250 PRINT #1:DB$

260 NEXT ENTRY

270 CLOSE #1

280 END
```

This program uses quite a few concepts, the most important of which is the concept of variables. Each time you are asked to "INPUT" something, you are putting that something in to a variable. A variable is a place (computer memory location) where values vary. Variables are alphabet and number combinations (alphanumeric) used to show that place. The reason we use variables is to enable us to change values within a program. The changes (INPUTs) may be made by a person or machine.

There are two types of variables "string" and "number" variables. The difference between the two is that string variables hold alphanumeric entries which cannot be used arithmetically and number variables only hold numbers which can. Since you can store 12345 in either a string or number variable, you need a way to tell one from the other. If the variable is a string then the value will be enclosed in quotes and the variable will end in the dollar sign ("$"). On the other hand if the variable is a number then its value has no quotes nor does it end in any special symbol.

e.g. string variable: 100 ADDR$="12345"
number variable: 110 UNITS=12345

The second concept is ease of use or the man machine interface. In the absence of the tools necessary to make a fancier display, we use what is available. For example the first thing we do is to clear the screen. We would, also, make any changes to the screen color or load any special character sets or symbols at this point. Then we must decide how to make the program invisible to the user as well as providing simple instructions. This is accomplished by deciding just what we want to say and using combinations of the "INPUT" and "PRINT" commands to format how information appears on the screen. The look and feel of the program (did I say Mac?) will determine if the user wants to spend time with the program. As an important side note, the use of REM (remark) statements helps to make our program easy for us to debug (correct) because it is easy for us to read.

The main program begins in line 200 and illustrates three additional concepts. First, it shows how to open a device. Second, it shows how to combine strings to create the proper syntax – the exact language structure that the computer understands – to open that device. And finally, it shows the concept of defaults, which are values that the command uses, invisibly (they ARE there), when you don't tell it otherwise. (See your TI BASIC reference manual for the full "OPEN" command.)

The sixth concept is looping which is in line 210. This is the core of all computing. By placing an instruction outside a selected group of instructions we can repeat that instruction(s) as many times as we desire. In this case we will repeat lines 220, 230, 240, and 250. As long as the "MAXNAMES" variable (we entered in line 140 to show the number of maximum number of times that we wish to enter names) is less that what we input,

— See CHRIS' CORNER on page 8

— CHRIS' CORNER, from page 7

every time the program reaches line 260 it will kick back to line 220 and repeat the instructions in those lines: 220, 230, 240, 250. When we reach the value in "MAXNAMES" the program will continue with line 270. Note that the "1" used in line 210, also, could have been a numeric variable. Also, note that looping uses a default. (See your BASIC reference manual.)

The seventh concept is conditional branching. This is the concept which allows the computer to make decisions. You tell it in an algebraic form which conditions or things you are comparing and then instruct the computer to branch (GOTO) another portion of the program if certain conditions are met.

The final concept of this article is the "PRINT" command. The "PRINT" command is extremely powerful and all it does is just print. But WHERE it prints is the question. The answer is anywhere you tell it: to disk drives, ramdisks, rs232, pio, ram, printers, etc. Using the syntax in line 250, the information you input in line 220 (your name data) is stored in the variable DB$. This is PRINTed to the device OPENed in line 200 (which in this case is a disk drive). Of course, you OPENed a printer in line 200 then DB$ would be printed to a printer which if set up properly would print the output. In the case of a disk drive, when it is printed to, the information is saved. It should be noted that the reverse of PRINTing to a device is INPUTting from that device. You should study your reference manual because there are many defaults associated with: OPEN, PRINT, and INPUT. Understanding how to use these commands will allow you to control the output of your computer. If there is enough interest I will explain the defaults in a future installment.

So there you have eight programming concepts: variables (string and numeric), look and feel interface, opening a device, combining strings, defaults, looping, branching and printing. I hope you will spend two to three hours studying the information presented.

Next month: An introduction to the most awesome language available for the TI 99/4A, Wycove forth with the t_extensions™. (t_dos™ is a subset of the t_system™. t_dos will be gradually introduced as we explore the t_extensions to Wycove forth and the t_system later in this series.)

PS. In case you did not know it Wycove forth runs on the 9640 and with the t_extensions allow access to all available memory and video resources.

Tech Comments:

Although I have not received my TIM back from Bud Mills which I sent it to him in February 1992 for repair, I feel obligated to express my enthusiasm for the new HRD 4000. If it functions as advertised in the latest edition of MICROpendium then it is the one piece of hardware that every active TI user should have if you don't already own a Horizon Ramdisk. Why? Because the Horizon protects your investment in the TI by providing ram and ram disk capabilities. While most of you may not know it, the vast majority of programs written for the IBM class of computers are written for only 64K code blocks with extended memory programs using 16K blocks of program code. A process called bank switching is used to allow larger programs to be written. This scheme is available on the TI using the HRD. In my last article I said that had two Megs of ram. One meg of this is for programs (the t_system) and the other meg is for ramdisk data. To understand how bank switching can expand the potential of the TI 99/4A, examine the following table.

Memory Address	Computer function	Size
>0000	console ROM: two 4K ROM chips used for the system monitor	8K bytes
>2000	Low Memory expansion: 8K of the 32K memory expansion	8K bytes

>4000	Peripheral ROMs: controls devices like disk drives	8K bytes
>6000	Cartridge space: used for TI cartriges like Extended BASIC	8K bytes
>8000	VDP, Sound and Speech input/output area	8K bytes
>A000	High Memory: 24K of the 32K memory expansion	24K bytes

The maximum amount of memory that the 99/4A can directly address is 64K. That memory is subdivided electrically into eight areas of 8k. The 6000 hex memory space reserves 8K for use with different cartridges. Because a lot of software for the 99/4A is cartridge based, programs can be changed by simply changing cartridges. We can combined the contents of several cartridges into one cartridge and add a selector switch to choose which cartridge program we wanted. Each cartridge represents a bank of programs. Every time we manually choose a different cartridge we are in effect selecting a different bank. Of course, the smarter route would be to develop a method which would allow us to accomplish the switching by way of software. Such software could contain pull-down menus with choices being selected possibly by a mouse.

Most of the cartridges for the TI contain ROM or GROM and, therefore, cannot be changed. If we used RAM, then we could have our choice programs available instead. Programs like a calendar, a drawing program, a word processor, etc., would be instantly available. The HRD offers such a capability!

When I first explained that to use the t_system™ you had to have an Horizon ramdisk many people complained. What they did not understand is that although companies such as Myarc, Foundation, Corcomp, and Rave created memory boards which, also, functioned as ramdisks and RAM, the most significant difference was that they forced the programmer to bank switch ALL four 8k banks the main 32k used for memory expansion in and out at a time whereas the HRD leaves the 32K area alone. This difference permitted me to use a stable (virtually bug-free) programming environment (using forth) without having to develop custom software for the very small market of users of these memory cards. That is, I could write programs using my t_system™ kernel for 32K as well as several megabytes. My code will work with the original HRD which only supported 2K banks. I used these 2K banks in a database in my military job. Imagine being able to search a thousand records in a matter of seconds on a TI. Think about it. Till next month. ct

---

NEW WATCHAMACALIT NEW

Do you own a MYARC HFDC?
Are you tired of always having to "Set Your Clock every time you turn your System on"?. Does yours run Fast?

THEN THE WATCHAMACALIT IS FOR YOU!

The Watchamacalit provides you with a means of "tuning" your Clock Speed. Even if your Clock is Fast or Slow, it can be corrected with the Watchamacalit.

Just think of it. Accurate Time/Date of your HardDisk files each and every time you turn your system on. With the Battery back-up, the time will be there each time you Power-up.

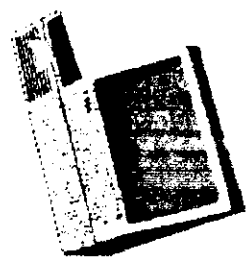
The Watchamacalit comes with instructions on a 5.25 disk (DS/SD) along with a Clock program you can use to adjust your Clock Speed. It comes with a one year full warranty (except the battery).

KEEP THE CORRECT TIME FOR ONLY:
$29.95 PLUS $3.00 S+H

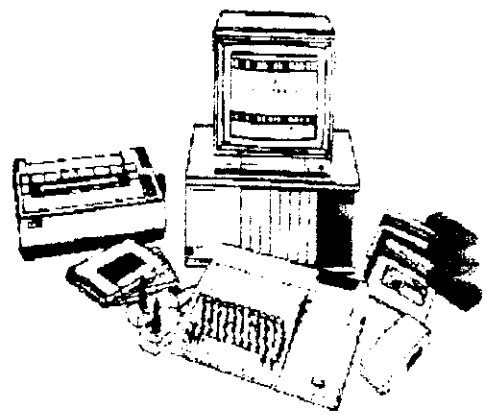
SEND ORDER TO: OGDEN TI USERS GROUP
1775 22ND ST
OGDEN, UT 84404

The Watchamacalit will be on sale at FestWest "NORTH" 93.
For a reduced price of only $24.95

FESTWEST "NORTH" 93



SO FAR THIS IS WHO IS COMING TO THE FEST.



SouthWest 99ers
Regena
Don Shorock
Bob Webb
Asgard represented by Mel Bragg
Rave 99 Will be represented
Mike Wright
Barry or Gary Harmsen. Holland
Ogden TI Users Group With the NEW WATCHAMACALIT
9640 NEWS
Rich Gilbertson C.A.D.D SOFTWARE
Delbert Wright D. WRIGHT STUFF
Slaves User Group
COMPETITION Computer
Gerrd Weissenman GERMANY

TIsHUG NEWS DIGEST September 1992

Writing in Machine Code

The Video Processor, part I by J.E. Banfield

The TI99/4A relies on the TMS9918A series of video processor chips, which differ mainly in their video output specifications, which need not concern us. This processor is a very complicated integrated circuit, probably much more complicated than the TMS9900 CPU chip, at least its data manual is three times as thick as the TMS9900 manual.

The video processor controls a 16K dynamic RAM which it uses for screen display data, the colour table and sprite definitions. There is sufficient dynamic RAM address space left over to be a very significant addition to other TI99/4A RAM; the BASIC interpreter uses it extensively. I use it as a buffer for disk data. For example, a full disk track can be set in VDP RAM prior to transfer to a DMA memory in my disk controller.

The control levels required to interface the video processor are defined in Table 6-1, extracted from the data manual. We can now face the job of screen control.

Screen Border Control

The border screen colour is controlled by the rightmost 4 bits of register 7 in the VDP chip. To change this colour code, we need to write to that register, 1 for black, F for white, etcetera.

This write to VDP register can be done in various ways. I chose here a long method which is less complicated than the alternative described later. The method of writing a program using the MiniMemory Easybug option was detailed in the last article and will not be repeated here. Enter the following program starting at M7FC0.

```
M7FC0   02 01    MOVEI 1,
        01 00    "black"
        02 02    MOVEI 2,
        87 00    VDP register 7
        D8 01    MOVB @.+2, 1
        8C 02    VDP write address
        10 00    SKIPA nil (delay)
        D8 02    MOVB @.+2, 2
        8C 02    vdp write address
        10 FF    SKIPA minus 1 (STOP)
```

Note that the first byte written contains the data, the second byte specifies a write to register 7 (see Table 6-1). Now press "." and type "E7FC0<enter>". The border colour goes black and we enter an infinite loop. Turn off the console and back on again (or reset any way you can).

Let us examine the program.

```
M7FC0   02 01
0 0 0 0   0 0 1 0   0 0 0 0   0 0 0 1
\_____020_____/ \_____/
   op-code        S address
   MOVEI          Ac 1
```

See Figure 1 in the last article. The instruction moves the immediate data, to be found in the next word (01 00) to Ac 1.

```
M7FC4   02 02
0 0 0 0   0 0 1 0   0 0 0 0   0 0 1 0
\_____020_____/ \_____/
   op-code        S address
   MOVEI          Ac 2
```

The data in the next word (87 00) is placed in Ac 2.

```
M7FC8   D8 01
1 1 0 1   1 0 0 0   0 0 0 0   0 0 0 1
\___/     \_____/   \_____/   \_____/
op-code   D address  S address
MOVB      @.+2       Ac 1
```

Move the contents of the left hand byte in Ac 1 (that is 01) to the address given in the next word, which is 8C 02.

```
1 0 0 0   1 1 0 0   0 0 0 0   0 0 1 0
```

This has the address line of A14 high which enables the VDP chip for writing to as can be seen from figure 6-1 and Table 6-1, that is U100 pin 14 (CSW(L)) will be low and pin 13 (MODE) will be high.

The next instruction is a delay which is required by the Video Processor chip. The following instruction is similar to 7FC8 but specifying the source as the left most byte of Ac 2, so the second byte written is 87, defining VDP register 7. The final instruction, SKIPA minus 1, as explained in the second last article, enters an infinite loop, in effect causing a stop.

OK, if all goes well we will make a change to avoid the inconvenience of the infinite loop. Change the instruction at 7FD2. In Easybug type:

```
M7FD2   04 60    JUMPA @.+2
        7F E0    address 7FE0
```

This instruction is made from:

```
M7FD2   04 60
0 0 0 0   0 1 0 0   0 1 1 0   0 0 0 1
\_____044_____/ \_____/
   op-code        S address
   JUMPA          @.+2
```

Then enter a short program at 7FE0.

```
M7FE0   02 06    MOVEI 6,
        70 E5    GROM address
        04 60    JUMPA @.+2
        00 60    ROM address
```

This loads Ac 6 with a GROM address (miniMemory GROM entry?) and the next jumps to the GPL interpreter in the console ROM. Check this by executing 7FC0.

There is no change to the border colour but "?" is displayed. Now change the colour data by:

```
M7FC2   09
```

and repeat the execution of 7FC0.

Now examine the contents of M83FE.

```
M83FE = 8C
M83FF = D2
```

Incredible, the VDP write data address is in console RAM and we did not put it there!

Sacred Sites

TI, in its wisdom (??), have sequestrated certain addresses for specific purposes and you may only change their contents AT YOUR PERIL. In particular:

```
83FA    (Ac D or R13 in GPL interpreter)
83FC    (Ac E or R14 in GPL interpreter)
83FE    (Ac F or R15 in GPL interpreter)
```

Although we must not change the contents of these registers, there is no bar to using them as data as described for Ac F later.

Multiple VDP Byte Write

Time and space is running out fast so I will leave detailed explanation of these routines to a later article. However, you might like to try them out and analyse the code. This routine writes a selected byte a number of times as defined in the data to incrementing VDP addresses. I use it to set up track data for formatting disks.

```
M7C60   C0 FB    MOVE 3, @ B+
        C1 03    MOVE 4, 3
        02 43    ANDI 3,
        FF 00
        02 44    ANDI 4,
        00 FF    count
        D8 03    MOVEB @.+2, 3
        8C 00    VDP write data
        06 04    SOS 4
        16 FC    SKIPNE minus 4
        04 5B    JUMPA @ B (return)
```

This is called by:

```
06 A0     JSP @.+2
7C 60
WX YZ     data: WX is the ASCII byte to be
            written YZ times.
```

Before executing the 7FC0 subroutine, it is necessary to set the VDP address which can be done in the following subroutine.

```
M7A00   C0 7B     MOVE 1, @ B+
        D7 E0     MOVEB @ F, @.+2
        83 E3     Ac 1, right byte
        10 DO     delay
        D7 C1     MOVEB @ F, 1
        04 5B     JUMPA @ B
```

In fact, we can use the 7A00 subroutin to change the border colour as as alternative to the 7FC0 routine.

Now enter the program:

```
M7A40   06 A0     JSP @.+2
        7A 00     set VDP address
        40 E3     address with write flag
        06 A0     JSP @.+2
        7C 60     multiple byte write
        41 10     write "A" 16 times
        04 60     JUMPA @.+2
        7F E0     to ? in Easybug
```

Now execute 7A40 several times. Try changes to address, data and count.

From Table 6-1, you will see that the only difference in VDP register write from VDP address write, is in the second byte written. So try:

```
M7A20   06 A0     JSP @.+2
        7A 00     set VDP address
        87 01     or 87 09 or 87 0X
        04 60     JUMPA @.+2
        7F E0     to ? in Easybug
```

The next article in this series will explain the above and show how to write the colour table in VDP RAM.

Correction to the article in Volume 11 Number 6, page 20. "Not valid for TMS9900" should read "Not valid for the TI99/4A". The TI99/4A lacks address decoding needed to implement these TMS9900 instructions.

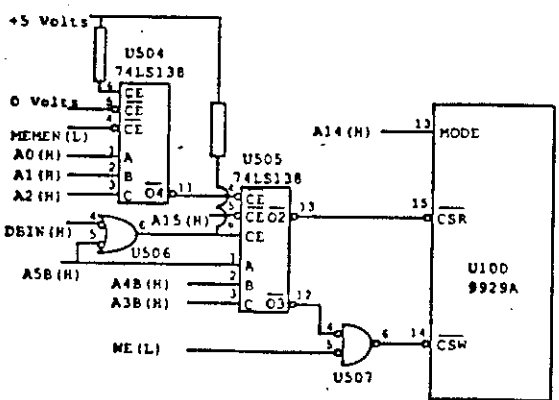


Figure 6-1
From TI99/4A Schematics Diagrams, sheets 1 and 4

$O4(U504)(L) = MEMEN \cdot A0 \cdot \overline{A1} \cdot \overline{A2}$

$O2(U505)(L) = O4(U504) \cdot \overline{A15} \cdot (\overline{DBIN} + \overline{A5B}) \cdot \overline{A5B} \cdot A4B \cdot \overline{A3B}$

$CSR(L) = MEMEN \cdot A0 \cdot \overline{A1} \cdot \overline{A2} \cdot \overline{A3B} \cdot A4B \cdot \overline{A5B} \cdot \overline{A15}$

x=don't care 1000 10xx xxxx xxx0
8 8 even

$O3(U505)(L) = O4(U504) \cdot \overline{A15} \cdot (\overline{DBIN} + \overline{A5B}) \cdot A5B \cdot A4B \cdot \overline{A3B}$

$CSW(L) = ME \cdot O3(U505)$
$= ME \cdot \overline{DBIN} \cdot MEMEN \cdot A0 \cdot \overline{A1} \cdot \overline{A2} \cdot \overline{A3B} \cdot A4B \cdot A5B \cdot \overline{A15}$

1000 11xx xxxx xxx0
8 C even

OPERATION	D7	D6	D5	D4	D3	D2	D1	D0	R/W	CD	MODE
WRITE TO VDP REGISTER									0	1	1
BYTE 1 DATA WRITE	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	0	1	0
BYTE 2 REGISTER SELECT	1	0	0	0	0	R_2	R_1	R_0	0	1	1
WRITE TO VRAM											
BYTE 1 ADDRESS 8 S/UP	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	0	1	1
BYTE 2 ADDRESS 14 S/UP	0	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_{13}	0	1	1
BYTE 3 DATA WRITE	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	0	1	0
READ FROM VDP REGISTER											
BYTE 1 DATA READ	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	1	0	1
READ FROM VRAM											
BYTE 1 ADDRESS 8 S/UP	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	0	1	1
BYTE 2 ADDRESS 14 S/UP	0	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_{13}	0	1	1
BYTE 3 DATA READ	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	1	0	0

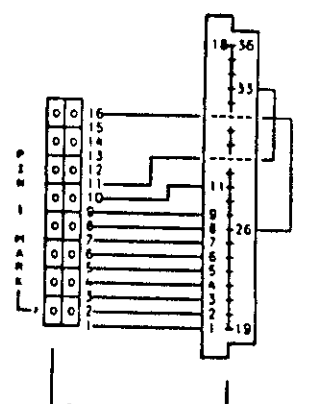
Table 6-1 - CPU/VDP Data Transfers. From the TMS9918A Manual

CIN-DRY

TI 99/4A PARALLEL PRINTER CABLE

Last week I helped one of our members make up a parallel printer cable for a new printer, and it turned out to be so easy that I decided to share it so you can make up a spare for your system. The easiest way is to use some insulation-displacement connectors and ribbon cable you don't have to do any soldering. But you can use hooded connectors and 12-conductor cable to make a fancier hookup. If you use sixteen-conductor ribbon cable, go for the multi-colored type since it's a lot easier to keep track of the lines. Six feet of it cost about $3. The 36-pin female IDC connector is about $2 and the 36-pin "Centronics" male connector is about $4. All you have to do is clamp the 36-conductor ribbon in the 36-pin connector on one end and split the wires on the other end so you can seperate the 3 that don't go in order. Notice that the numbering scheme is different on the opposite end of the cable, but the #1 pin is marked with an arrow on the computer end and the printer end is marked on pins 1 & 19 and pins 18 & 36. This means that on the printer end with an IDC connector you use every other wire slot for 1-6. Be sure the wire is firmly inserted in each terminal pin so the insulation is displaced (hence the name - insulation displacement connector). Or you can use 36-connector cable and use every other wire on the 36-pin plug end for 1-9. If you have any problems, give me a call at 714/628-6886.

Gene Bohot



GPL (Graphics Programming Language)
by Richard Lynn Gilbertson

There are many things that can only be done in GPL. And any Assembly Language programmer knows many of the subroutines built into the console lead right back to the GPL interpreter. Making them useless for pure Assembly approaches that would like to use them.

My approach is to imbed Assembly into the GPL code and branch to it only when speed is required. GPL is great for what Texas Instruments designed it for, which is set-up, menus, and for storage. Programing wise GPL always leads back to GPL, while Assembly has to exit to the original start up screen, or to other Assembly Language programs. When creating programs GPL takes less memory space to do the same thing as Assembly. That is becuase GPL is a BYTE orientainted language and Assembly is a WORD (two bytes) orientainted language. I use both languages, but for speed I need the Assembly. For control I use GPL.

So Assembly is good for speed, and GPL is good for menus, storage, set-up, and controlling the whole thing in a orderly fashion. You need gobbs memory to get that out of Assembly. Looking at some news letters I found TI-99/4A MEMORY ARCHITECTURE by John F. Willforth. It shows where all MEMORY MAPPED PORTS are, including FastRAM, Sound, VDP, Speech, and last but not least GROM/GRAM.

History: Texas Instruments decided that if they sacrificed 8 bytes they could gain 40K of GROM/GRAM. Setting up 16 banks with 8 bytes each used up 128 bytes of address lines. But now there are 16 banks so that is: 40K times 16 banks equals 640K. It requires little thought to see that if each of the 16 bank lines went through a PAL chip that made 16 more per normal bank that you would get:

```
PAL   BANKS          1BANK        TOTAL
       016 * 00016 * 0040K =      10240K
```

or 1 Meg, or better yet:

```
PAL   BANKS          1BANK        TOTAL
       256 * 00016 * 0040K =     163840K
```

or 16 Meg or up to:

```
PAL   BANKS          1BANK        TOTAL
     65535 * 00016 * 0040K =   41942400K
```

or 4095MEG

I'm not an electronics wiz, I am a programmer. But I've had a few discussions with those who do know it can be done. And I've read the comments of the engineers that designed the TI-99/4A. This seemed exactly where they were going. Consider Assembly and GPL all being run from the original operating system that is already in the TI-99/4A !!!!

THAT IS TOTAL COMPATIBILITY!

The TECHNICAL TRAINING COURSE OUTLINE which is in my library of books to have. This book is mostly the development outline of the TI-99/4A, 4B (yes, 4B), and the TI-99/4X (99/8). Now having read it several hundred times with no electronics talent, I have found that the only difference between the 4A and 4B is three (3) jumper wires and one (1) chip. W. Germany is suppose to send us the data (5 months ago) of how it is done.

The 4B has twice the load/save speed of the 4A. The 4B does'nt need the VDP to transfer disk to memory, it goes straight to CPU memory. Also it's 100% compatiable with the 99/4A. The 4B software sets up a PAB like the 4A, but the 4A uses the VDP to transfer the data, so the 4B can run the 4A software.

I just wanted to mention that from the average users point of veiw, he just wants to load and go. GPL combined with Assembly and hardware modifications like the 99/4B are the types of approches that will never create a bottle neck. Most of the hardware made for the TI is an attempt toward that. But it seems there is a real lack of knoweldge of GPL and what it is best at. Also as the console has most of it's memory devoted to it, trouble only occurs when you are trying to avoid GPL in Assembly.

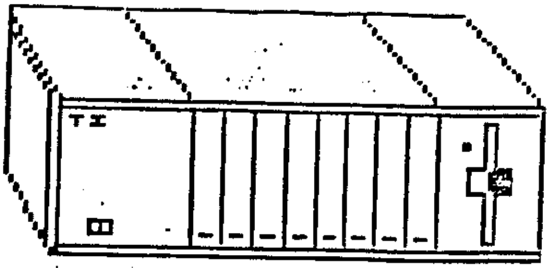
Programs written in Assembly can be re-written to run from GPL and little change is needed. Imbedded Assembly run from GPL saves the area it runs from, it runs, restores the area and continues. You can call it, use it, and return the to GPL or Extended Basic program just where it left off. Now I will admit that this method would slow down the original Assembly program. But we want convenience, speed, and compatibility! Not just speed. Besides this is exactly the kind of stunt we can do that the rest of Computers around can't do.

WE DON'T LOAD A DIFFERENT OPERATING SYSTEM!

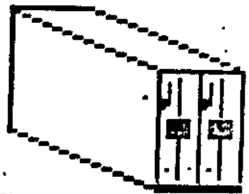
OUR OPERATING SYSTEM IS BUILT IN!

NUFF SAID! RICH

PBOX



by Bill Keller



DUAL DRIVE FULL POWER MODIFICATION

Have you ever found a good deal on a disk drive, and then found out that it was a full power drive and that you could not run it with your present P box. Well this little mod in your P box will enable you to run two (2) full power drives (either full heights - if you build a case for it to stand alone - , or two half height drives that will mount inside your P box).

If you attempt this modification you do so at your own risk. The mod is working in at least two (2) P boxes. This mod is brought to you with special thanks to Tony and Joe.

PARTS REQUIRED:
1 - +5volt REGULATOR 7805
1 - +12volt REGULATOR 7812
2 - 47uf 50volt CAPACITOR
2 - 3uf 25volt CAPACITOR
1 - CIRCUIT BOARD
1 - DISK DRIVE MALE SOCKET CONN.
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

Looking at the P box from the rear, select the pins on the Right Hand side of the motherboard socket. Pins one(1) and thirty(30) show a nominal +12 volts and +24 volts, again when you are looking from the rear of the P box. The grounds (and there must be two of them), are located at pins four(4) and twenty-four(24). If you find this hard to follow, get a digital multimeter and measure for +12v, +24v, and the two grounds and mark them as such. Next fashion your circuit board so that your pins will line up with the slots you have determinded are your hot and ground pins. The circuit board with completed runs should look something like figure #1. &&NOT DRAWN TO SIZE.&&

By drilling little holes in your board the lead of your componets may be fastened easily and neatly to the board

The next thing to do is to soldier your two(2), 47uf capacitors to the board. The capacitors should be connected: one lead to ground, and the other attached to one of your input slots. This should be done for both your +12v and +24v input slots. See Figure #3.

Next mount your regulators, remembering the 5v regulator goes with the +12v supply and 12v regualtor goes with the +24v supply. The regulators should have heat sinks, to keep them running cool. The pin out for the regulators are discribed in figure #2. Connect the input lead of your regulator to the same common point as your 47uf capacitor(Hot side). Connect output lead to your output slot. Soldier the ground to the respective ground on your board. See Figure #3.

The 3uf capicators, are connected from ground to the output slot of your board. See Figure #3.

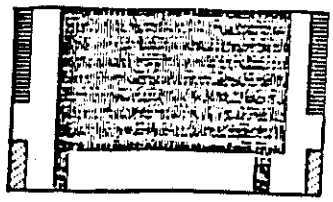
A light can be rigged up to the front of your P box to tell you that in fact you have a circuit board running in that slot. I picked off the 12v line to run my lamp. See Figure #3.

The last thing to connect is your disk drive socket connector. By measuring your other drive's power line socket, you can tell which pins should be your +12v, +5v, and Ground leads for your new socket. The socket only fits one way. Once you have the lines figured out then connect your wire to the corresponding power output slots from your board.

&&&&&DO NOT GET YOUR 5V AND 12V LINES CROSSED OR YOU WILL DESTORY YOUR DISK DRIVE.&&&&&

This little mod has eliminated some of my daul drive lockouts. I was running two(2) full power drives with just the single P box power supply. When I tried making back-up copies using both drives I would get read/write errors. Sometimes it would just initialize wrong, and sometimes the keyboard would lock me out. I now can run double density on the both drives(still have the old TI Controller) with no error.

Inside every small problem is a
larger problem struggling to get out!



GROUND PLATE

INPUT RUNS

OUTPUT RUNS

FIGURE #1

REGULATORS (Top View)

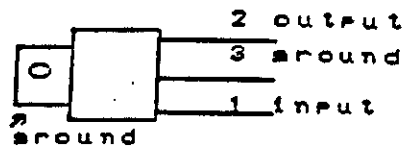


FIGURE #2

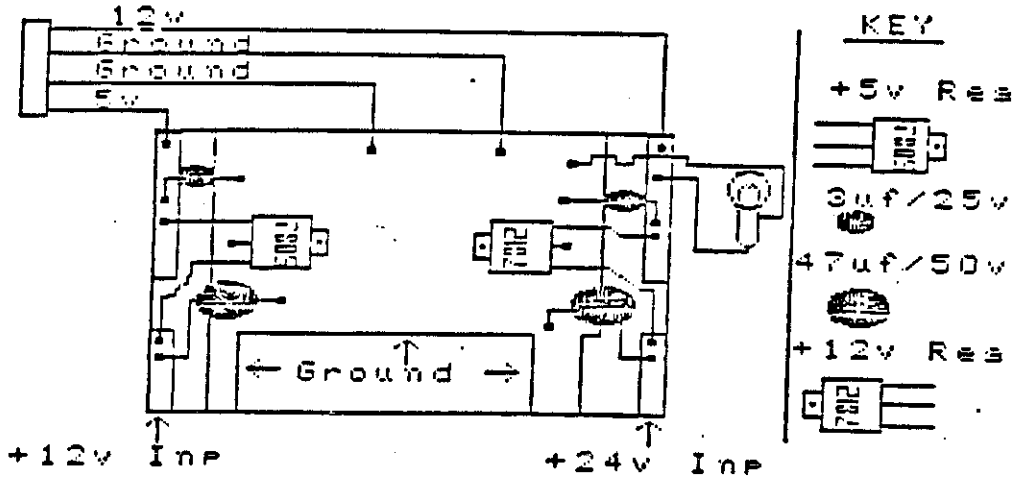


FIGURE #3

Erase All -- Day of the Week

We have two short programs for you this month that you may enjoy. The first program is called "Erase All". What it does is compare the result of using CALL CLEAR or ERASE ALL to clear the screen. Some say that ERASE ALL is quicker than CALL CLEAR but with this program there doesn't appear to be much difference.

Anyway try it out. It's simple to type in and you may improve your programming skills by studying the logic of the program.

The second program is entitled DAY/WEEK. With this program you can type in any date and the program will reveal to you what day of the week any event occurred on. To understand some of the logic of the program you might want to refer to our calendar program that we published in the January 1990 issue of Word Play. Adrian Johnson of the Orange County User Group provided an interesting history of how our present day calendar came about and the mathematics needed to research when a particular date occurred in the past.

Erase All

```
10 R=11
20 CALL CLEAR
30 DISPLAY AT(14,2):"(Press
(ANY) key to start)"
40 CALL KEY(0,K,S):: IF S=0
THEN 40
50 FOR I=1 TO R
60 CALL HCHAR(1,1,I+32,768)
70 CALL CLEAR
80 NEXT I
90 DISPLAY AT(14,10)BEEP:"TE
ST #1"
100 DISPLAY AT(14,2):"Press
(ANY) key to start)"
110 CALL KEY(0,K,S):: IF S=0
 THEN 110
120 FOR I=1 TO R
130 CALL HCHAR(1,1,I+32,768)
140 DISPLAY ERASE ALL
150 NEXT I
160 DISPLAY AT(14,10)BEEP:"T
EST #2"
170 GOTO 40
```

Day of the Week

```
100 INPUT "ENTER MM,DD,YYYY:
":M,D,Y
110 A=Y-(INT(Y/28)*28):: B=A
/4 :: B=A-INT(B)*7
120 C$="311462403513" :: IF
M=0 THEN IF M<3 THEN C$="40"
130 E=VAL(SEG$(C$,M,1)):: IF
 Y<1900 THEN A=A+12
140 G=A+INT(B)+D+E :: F=G-(I
NT(G/7)*7)
150 DATA SUN,MON,TUES,WEDNES
,THURS,FRI,SATUR
160 RESTORE :: FOR R=0 TO F
:: READ C$ :: NEXT R
170 PRINT "THE DAY IS ";C$;"
DAY"
180 PRINT
190 INPUT "DO ANOTHER? (Y/N)
: ":YN$
200 IF YN$="Y" OR YN$="y" TH
EN GOTO 100
```

INCOME TAX HELPER
modified by
Bob DeVilbiss

Shortly after I purchased my TI PEB and printer I received this program. Who gave it to me? I have no idea, but I find this program very useful when time comes to file my income tax returns.I have no idea who the author is,so I cannot give due credit.

My knowledge of programming remains a lot to be desired, but I was able to modify certain categories to fit my personal use.

The program is divided into two sections. The first section relates to all income that is received and the second section covers all expense items

When the program is printed, subtotals and totals are povided for all categories.

I find if I enter my income and expense items each month I am able to keep up with the paper work.

The program is written in BASIC and the input is entered in DATA statements.Instructions on how to enter the data are included in the program and they can either be viewed on the screen or sent to a printer.

The following is a list of the symbols and descriptions in case you want to modify the program:

SYMBOL	DESCRIPTION
M	Maximum number of categories.
MO	Maximum number of data reads
M1	Number of income categories
C1$()	Master category code array
D1$()	Master category description array
T$	Income/Deduction code
C4	Transaction category code
D	Transaction amount
S$	Transaction description
T1	Subtotal Income - Deduction
T2	Total Income/Deductions

```
100 CALL CLEAR
110 OPEN #1:"PIO"
120 PRINT "INCOME TAX RECORD
ING PROGRAM"
130 PRINT
140 PRINT "DATA STATEMENTS S
TART WITH"
150 PRINT "LINE NUMBER 1990"
,::
160 PRINT
170 PRINT "DO YOU WANT TO SE
E THE"
180 PRINT "INSTRUCTIONS? (Y
OR N)"
190 INPUT A$
200 CALL CLEAR
210 IF A$="N" THEN 840
220 PRINT
230 PRINT "THIS PROGRAM INIT
IALIZES"
240 PRINT "THE VARIOUS INCOM
E/DEDUCTION"
250 PRINT "CATEGORIES. OUTPU
T IS"
260 PRINT "PRODUCED IN SEPAR
ATE"
270 PRINT "SECTIONS FOR INCO
ME AND"
280 PRINT "DEDUCTIONS. SUBTO
TALS AND"
290 PRINT "TOTALS ARE PRODUC
ED FOR ALL"
300 PRINT "CATEGORIES."
310 PRINT
320 PRINT "ALL DATA IS ENTER
ED USING"
330 PRINT "-DATA-STATEMENTS.
"
340 PRINT "EXAMPLE:"
350 PRINT "DATA I,W,13.45,EM
PLOYER 1.  (INCOME,WAGES,AMO
UNT,SOURCE)"
360 PRINT
370 PRINT "PRESS ENTER TO CO
NTINUE"
380 INPUT G$
390 PRINT "INCOME ITEMS ARE:
"
400 PRINT "   W,WAGES"
410 PRINT "   P,PENSION"
420 PRINT "   TR,TAX RETURN"
430 PRINT "   I,INTEREST"
440 PRINT "   D,DIVIDENDS"
450 PRINT "   R,RENT/ROYALTY
"
460 PRINT "   O,OTHER"
470 PRINT
480 PRINT "DEDUCTION ITEMS A
RE:"
490 PRINT "   C,CONTRIBUTION
S"
500 PRINT "   I,INTEREST"
510 PRINT "   T,TAXES PAID"
520 PRINT "   MD,MEDICAL/DEN
TICAL"
530 PRINT "   CT,CASUALTY TH
EFT"
540 PRINT "   M,MISC EXPENSE
"
550 PRINT "   O,OTHER EXPENS
E"
560 PRINT
570 INPUT "DATA STARTS WITH
LINE 1990 PRESS ENTER":A$
590 PRINT "DO YOU WANT A PRI
NTOUT OF"
600 PRINT "THESE INSTRUCTION
S? (Y OR N)"
610 INPUT A$
620 IF A$="N" THEN 840
630 PRINT #1:"THIS PROGRAM I
NITIALIZES THE VARIOUS INCOM
E/DEDUCTION CATEGORIES"
640 PRINT #1:"OUTPUT IS PROD
UCED IN SEPARATE SECTIONS"
650 PRINT #1:"FOR INCOME AND
 DEDUCTIONS. SUBTOTALS AND"
660 PRINT #1:"TOTALS ARE PRO
DUCED FOR ALL CATEGORIES."
670 PRINT #1:
680 PRINT #1:"ALL DATA IS EN
TERED USING -DATA-STATEMENTS
."690 PRINT #1:"EXAMPLE:"
700 PRINT #1:"DATA I,W,13.45
,EMPLOYER 1"
710 PR      :
720 PR      #1:"INCOME ITE.  A
RE:"
730 PRINT #1:"   W,WAGES B,B
USINESS F,FARM I,INTEREST D,
DIVIDENS R,RENT/ROYALTYD,OTH
ER"
740 PRINT #1:
750 PRINT #1:"DEDUCTION ITEM
S ARE:"
760 PRINT #1:"   C,CONTRIBUT
IONSI,INTEREST T,TAXES PAID"
770 PRINT #1:"   MD,MEDICAL/
DENTAL CT,CASUALTY THEFT M,M
ISC EXPENSE"
780 PRINT #1:"   O,OTHER EXP
ENSE"
790 PRINT #1:
800 PRINT #1:"DATA ENTRIES S
TART AT LINE #1990 . DATA ST
ATEMENT (DATA END) MUST FOLL
OW"
810 PRINT #1:"LAST DATA ENTR
Y."
820 PRINT #1: : : : :
830 INPUT G$
840 CALL CLEAR
850 REM INCOME TAX RECORDING
 PROGRAM
860 PRINT "INCOME TAX RECORD
ING PROGRAM "
870 REM  ##DATA INITIALIZATI
ON##
880 M=15
890 MO=10000
900 M1=8
910 DIM C1$(15)
920 DIM D1$(15)
930 C1$(1)="W"
940 D1$(1)="WAGES (1040 LINE
 7)"
950 C1$(2)="I"
960 D1$(2)="INTEREST INCOME
(LINE 8) & (SCHEDULE B)"
970 C1$(3)="D"
980 D1$(3)="DIVIDEND INCOME
(LINE 10) & (SCHEDULE B)"
990 C1$(4)="TR"
1000 D1$(4)="TAX REFUND (LIN
E 11)"
1010 C1$(5)="P"
1020 D1$(5)="PENSION BENEFIT
S (LINE 16a)"
1030 C1$(6)="R"
1040 D1$(6)="RENT/ROYALTY IN
COME (LINE 17) & SCHEDULE E)
"
1050 C1$(7)="S"
1060 D1$(7)="SOCIAL SECURITY
 BENEFITS)"
1070 C1$(9)="C"
1080 D1$(9)="CONTRIBUTIONS (
SCHEDULE A)"
1090 C1$(10)="I"
1100 D1$(10)="INTEREST EXPEN
SES (SCHEDULE A)"
1110 C1$(11)="T"
1120 D1$(11)="TAXES PAID (SC
HEDULE A)"
1130 C1$(12)="MD"
1140 D1$(12)="MEDICAL/DENTAL
 (SCHEDULE A)"
```

NEXT PAGE

```
1150 C1$(13)="CT"1160 D1$(13
)="CASULTY/THEFT (SCHEDULE A
)"
1170 C1$(14)="MI"1180 D1$(14
)="MISC EXPENSE (SCHEDULE A)
"
1190 C1$(15)="O"
1200 D1$(15)="OTHER EXPENSES
"
1210 REM  INCOME CATEGORIES
ARE FIRST 8 POSTIONS OF THE
ARRAY
1220 REM END OF CATEGORY ARR
AY INPUTS
1230 REM PRINT OF INCOME ITE
MS - BY CATEGORIES
1240 PRINT "ALIGN TO TOP OF
PAGE "
1250 PRINT
1260 PRINT "PRESS ENTER TO C
ONTINUE"
1270 INPUT G$
1280 PRINT #1:CHR$(14);"
    INCOME TAX HELPER"
1290 PRINT #1: :
#1"
1310 PRINT #1: :
1320 PRINT #1:"!!!!!!!!!!!!!
!!!!!!! INCOME !!!!!!!!!!!!!
!!!!!!!"
1330 PRINT #1:
1340 FOR J=1 TO M1
1360 PRINT #1:D1$(J)
1370 FOR I=1 TO M0
1380 READ T$
1390 IF T$="END" THEN 1470
1400 READ C$,D,S$
1410 IF T$<>"I" THEN 1460
1420 IF C$<>C1$(J)THEN 1460
;D
1440 PRINT #1:TAB(5);S$;TAB(
50);D
1450 T1=T1+D
1460 NEXT I
AB(50);T1
1480 PRINT #1:TAB(51);"-----
---"
1490 PRINT #1:TAB(42);"TOTAL
";TAB(50);T1
1500 T2=T2+T1
1510 T1=0
1530 PRINT #1:"-------------
--------------------------"
1540 RESTORE
1550 NEXT J
1560 RESTORE
COME";TAB(50);T2
1580 PRINT #1:TAB(36);"TOTAL
 INCOME";TAB(50);T2
1590 T2=0
1600 T1=0
1610 J0=J
1620 REM !!!!!!!! END OF INC
OME-START DEDUCTION PRINT !!
!!!!!!
1630 REM PRINT "ALIGN TO TOP
 OF NEXT PAGE AND PRESS ENTE
R KEY OR "
1640 REM INPUT Z$
1650 PRINT "!!!!!!DEDUCTIONS
!!!!!"
1660 PRINT "ALIGN TO NEXT PA
GE AND PRESS ENTER";X$
1670 INPUT X$
1680 PRINT #1:"!!!!!!!!!!!!!
!!!!!!!!!! DEDUCTIONS !!!!!!
!!!!!!!!!!!!!!!!!!"
1690 PRINT #1:
1700 FOR J=J0 TO M
1710 PRINT D1$(J)
1720 PRINT #1:D1$(J)
1730 FOR I=1 TO M0
1740 READ T$
1750 IF T$="END" THEN 1840
1760 READ C$,D,S$
1770 IF T$<>"D" THEN 1820
1780 IF C$<>C1$(J)THEN 1820
1790 PRINT TAB(5);S$;TAB(50)
;D
1800 PRINT #1:TAB(5);S$;TAB(
50);D
1810 T1=T1+D
1820 NEXT I
1830 PRINT TAB(42);"TOTAL";T
AB(50);T1
1840 PRINT #1:TAB(51);"-----
---"
1850 PRINT #1:TAB(42);"TOTAL
";TAB(50);T1
1860 PRINT #1:"-------------
--------------------------"
1870 PRINT "-----------------
---"
1880 T2=T2+T1
1890 T1=0
1900 RESTORE
1910 NEXT J
1920 PRINT TAB(36);"TOTAL DE
DUCTIONS";TAB(50);T2
1930 PRINT #1:TAB(36);"TOTAL
 DEDUCTIONS";TAB(50);T2
1940 T2=0
1950 T1=0
1960 PRINT #1:"=============
==========================
==============="
1970 REM !!!!!!!!!!!!!!!! DAT
A ENTRIES FOR INITIALIZATION
 !!!!!!!!!!!!!!!!
1980 REM DATA ENTRIES FOLLOW
1990 DATA END
                          END
```

****

KEYBOARD READER
by Bob Webb
Reprinted from
TIsHUG NEWS DIGEST
April 1992

This small program is one of my most used programs. I can never remember the number associated with a key press or ASCII symbol, so I threw this thing together. Let me caution you before I continue- DO NOT run this program until you have saved it, as once you start it the only way to stop it is to turn your computer off. Once this program is running,press any key-it's associated number will be displayed. If an ASCII symbol is associated with the particular key press it will be displayed just to the left of the number.

This program does not break any new ground, however you might find a part of it to be of use. I have added one of my favorite little details to it. If no key is is pressed for a given amount of time, it jumps to a screen saver type of subprogram.

This BLANK variable is a counter. This clock ticks away and if a key is pressed it is reset to zero and begins again. If no key is pressed it jumps down to line 410 and stays there until a key is pressed.

```
100 ! KEY TO NUMBER PROGRAM
110 ! EXTENDED BASIC AND 32K
120 ! BY BOB WEBB,6/91
130 ! CAUTION: YOU WILL HAVE
TO
140 ! TURN OFF COMPUTER TO E
ND
150 !
160 ! CALL LOAD DISABLES QUI
T
170 CALL INIT :: CALL LOAD(-
31,806,16)
180 !
190 ON BREAK NEXT
200 !
210 CALL CLEAR
220 BLANK=0
230 DISPLAY AT(5,5):"KEY TES
T PROGRAM"
240 DISPLAY AT(7,5):"PRESS A
NY KEY"
250 DISPLAY AT(9,5):"IT'S NU
MBER WILL"
260 DISPLAY AT(10,5):"BE DIS
PLAYED"
270 DISPLAY AT(11,5):"ASCII"
:: DISPLAY AT(11,10):" KEY"
280 !
290 !
300 CALL KEY(0,K,S)
310 BLANK=BLANK+1
320 IF BLANK>1000 THEN 140
330 IF S=0 THEN 300
340 DISPLAY AT(12,4):K
360 BLANK=0
370 GOTO 300
380 !
390 !
400 !
410 CALL CLEAR
420 CALL KEY(0,K,S)
430 IF S=0 THEN 420
440 GOTO 410
```

Root Finder

Besides having the ability to play great games and run many other complex programs, your TI-99/4A is also a superior number crunching machine. The following 2 programs demonstrate its ability to find the square root to 9th root of any number up to ten digits.

The first program will find the root and display its calculations on the screen as it solves the problem. The second program is much faster, reveals an almost instantaneous answer, but does not show the calculations on the screen.

I checked the answers on several test numbers with Macmillan's Logarithmic and Trigonometric Tables and the answers are absolutely correct. This book, by the way, is what I had to use when I was a college math student many years ago. What a difference today!

--Charles Ball, Editor

Root Finder with display

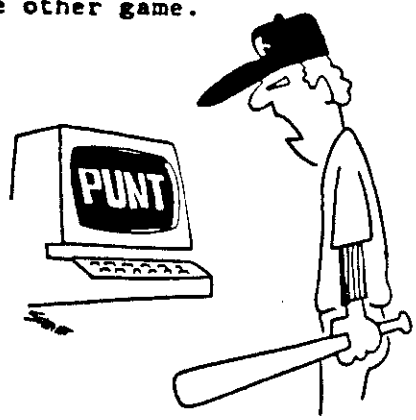
```
90 CALL CHARPAT(121,CB$):: C
ALL CHAR(33,CB$)
100 DISPLAY AT(1,1)ERASE ALL
:" ROOTS b! Lucie Borais":"
To find an! root from cube
root to 9th root"
110 !
120 ON WARNING NEXT :: PR$="
P10"
130 L$=RPT$("_",28):: E$=RPT
$(" ",168):: S$=RPT$(" ",8)
140 CALL CHAR(120,"000000000
0020502",121,"1F1020202040408
0",122,"018182C2C4646830",1
23,"080101",125,"FF")
150 DISPLAY AT(5,9):"x3y"&RP
T$(")",10):S$&"(z" :: GOSUB
280
160 ACCEPT AT(6,12)VALIDATE(
NUMERIC)BEEP:N :: IF N>2 THE
N 180
170 IF N=1 THEN AY=N :: GOTO
 220 ELSE AY=SQR(N):: GOTO 2
20
180 LO=0 :: HI=SQR(N)
190 AY=(LO+HI)/2 :: T=AY^N
200 IF OAY=AY THEN 220 ELSE
OAY=AY :: DISPLAY AT(12,8):A
Y
210 IF N<T THEN HI=AY :: GOT
O 190 ELSE IF N>T THEN LO=AY
 :: GOTO 190
220 AN=AY :: DISPLAY AT(12,8
)BEEP:"=";AN
230 DISPLAY AT(22,1):L$:" |A
|nother |C|hange root |P
|rint |Q|uit
240 CALL KEY(0,K,S):: IF S=0
 THEN 240 ELSE K=POS("ACPQ",
CHR$(K),1)
250 IF K=0 THEN 240 ELSE ON
K GOTO 260,260,270,290
260 DISPLAY AT(7,12):E$:E$:E
$ :: IF K=2 THEN GOSUB 280 :
: GOTO 160 ELSE 160
270 OPEN #1:PR$ :: PRINT #1:
S$&"_____":S$&"&ST
R$(R)&"/" :: PRINT #1:S$&"\/
";N;TAB(26);"=";AN:"" :: CL
OSE #1 :: GOTO 240
280 ACCEPT AT(5,10)VALIDATE(
"123456789")SIZE(-1)BEEP:R :
: RETURN
```

Root Finder

```
90 CALL CHARPAT(121,CB$):: C
ALL CHAR(33,CB$)
100 DISPLAY AT(1,1)ERASE ALL
:" ROOTS b! Lucie Borais":"
To find an! root from cube
root to 9th root"
110 !REVISED VERSION USING T
HE FORMULA AN=N^(1/R)
120 ON WARNING NEXT :: PR$="
P10"
130 L$=RPT$("_",28):: E$=RPT
$(" ",168):: S$=RPT$(" ",8)
140 CALL CHAR(120,"000000000
0020502",121,"1F1020202040408
0",122,"018182C2C4646830",1
23,"080101",125,"FF")
150 DISPLAY AT(5,9):"x3y"&RP
T$(")",10):S$&"(z" :: GOSUB
280
160 ACCEPT AT(6,12)VALIDATE(
NUMERIC)BEEP:N :: IF N>2 THE
N 180
170 IF N=1 THEN AY=N :: GOTO
 220 ELSE AY=SQR(N):: GOTO 2
20
220 DISPLAY AT(12,8)BEEP:"="
;AN
230 DISPLAY AT(22,1):L$:" |A
|nother |C|hange root |P
|rint |Q|uit
240 CALL KEY(0,K,S):: IF S=0
 THEN 240 ELSE K=POS("ACPQ",
CHR$(K),1)
250 IF K=0 THEN 240 ELSE ON
K GOTO 260,260,270,290
260 DISPLAY AT(7,12):E$:E$:E
$ :: IF K=2 THEN GOSUB 280 :
: GOTO 160 ELSE 160
270 OPEN #1:PR$ :: PRINT #1:
S$&"_____":S$&"&ST
R$(R)&"/" :: PRINT #1:S$&"\/
";N;TAB(26);"=";AN:"" :: CL
OSE #1 :: GOTO 240
280 ACCEPT AT(5,10)VALIDATE(
"123456789")SIZE(-1)BEEP:R :
: RETURN
```

The Wave

Stand up and cheer for your team. The "Wave" emulates the crowd at some atletic event. When you run the program you'll see that fans in the bleachers stand and then sit. This is repeated over and over again and will remind you of the spectators at the Blazer or some other game.



. . . Wrong Season!

```
90 !THE WAVE by David Renken
berger/mod by Jim Peterson
100 CALL CLEAR :: CALL SCREE
N(4)
110 A$="<<the wave>>"
120 DISPLAY AT(4,14-LEN(A$)/
2):A$
130 B$="press any key to sto
p"
140 DISPLAY AT(22,14-LEN(B$)
/2):B$
150 B$="005A3C3C3C3C2466"
160 A$="000018187EBD3C3C"
170 FOR CH=91 TO 118 :: CALL
 CHAR(CH,A$):: M$=M$&CHR$(CH
):: NEXT CH :: FOR R=8 TO 12
 :: DISPLAY AT(R,1):M$ :: NE
XT R
175 FOR T=1 TO 26 STEP 5 ::
DISPLAY AT(22,T):SEG$(M$,T,1
):: NEXT T
180 FOR CH=91 TO 123 :: CALL
 CHAR(CH,B$):: CALL CHAR(CH-
5,A$):: CALL SOUND(-999,-7,5
#RND):: CALL KEY(3,K,ST):: I
F ST<>0 THEN CALL MCH :: STO
P
190 NEXT CH :: GOTO 180
200 SUB MCH :: CALL MCHAR(8,
1,31,160):: CALL MCHAR(22,1,
32,32):: SUBEND :: END
```

GENEVE INFORMATION COLUMN
By Tom Arnold

I realize that there are only a few Geneve owners in our group but I have a little information about the Geneve that I would like to pass on. Actually there are 7 owners that I know of. They are Neville Blair, Wayne Anderson, Clint Pulley, Dick Broad (Colgan, Ont), John Van Wheelie, Eric Wicklund and myself. If you buy a Geneve then I'm sure all these people would be willing to help you out. Gary Bowser of the Toronto Users Group usually comes to our meetings and he too is a Geneve owner.

First I would like to point out that the new version (2.1) of TI-Base works quite well with the Geneve. The speed difference between the TI and the Geneve really makes loading this program quite quick. I'll talk about this great program another time.

Myword version 1.Ø2 has a bug which I have not had time to work out. When you format to the screen it does not work. If you do this don't panic, I think the formatter is working, it just does not show on the screen. Try a previous version of the formatter file FORMAT, this might correct the problem. Normally it is not wise to mix files from different versions of programs but this is worth trying.

A rather unusual feature of MyWord that I dicovered is really quite neat. Our Myarc programmers are at it again, hiding little features in the programs for us to find. I wouldn't be surprised if Peter Hoddie is behind this one. "What the hell is he talking about?" you ask. Well do the following: Load Myword, then type "H", select one of the options, anyone will do. Actually you could just type "EK" for example and you would accomplish the same thing. Now press enter and your help screen appears. Now type "Control 3". Your computer will start to play a tune!!! I think it is the Little Fugue by Beethoven that appeared for the TI a few years ago. It was written in Forth. Anyway it is a nice tune. You can control the speed of the tune by the number you choose. Control #1 is the fastest and Control #7 is the slowest. Why someone has put it in I have no idea but it makes for an amusing break.

I just bought a Horizon 2000+ Ramdisk. I spent a weekend trying to get this thing to work on the Geneve. I thought I might relay a few things so you can avoid the grief I went through because the documents are useless.

First the Gen[illegible] not compatible with the John Johnson R[illegible]erating System (ROS). This allows you configure the Ram Disk, format it, etc. This only works on the TI. There is a package out that allows you to use the Horizon as a boot disk. This package, sold by Bud Mills is called the Pheonix Modification. I was given this software but it is useless unless you buy the hardware to go with it. Anyway I was not interested in useing mine as a boot disk as it is only 192K. I just wanted a battery backed ram disk! Anyway here is how to get a Horizon Ram Disk operating on a Geneve.

1) Set your DIP switch on the card to #5 closed if you want to call the ram disk DSK6 or to #7 if you want to call the ram disk DSK7. The DIP switch is set in the "closed" position.

2) Use MDOS 1.Ø1 only. Boot use to DOS.

3) Run the program CONFIG-16 which is on the Pheonix disk. Most people will not give you this disk when you buy the Ram Disk so you will have to get a copy from someone such as me.

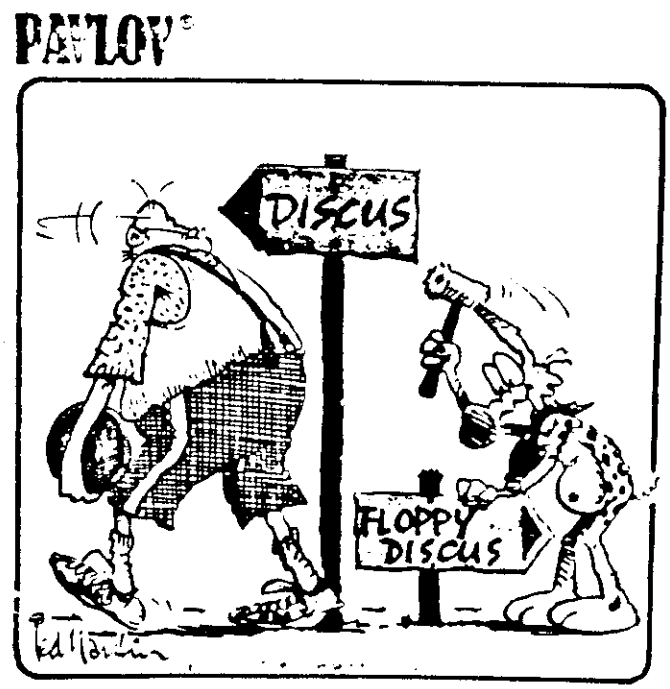
4) Put the statment "ASSIGN F=DSK6:" or "ASSIGN F=DSK7:" in your AUTOEXEC files.

5) Load up DM1000 and initalize the ram disk to the size you want.

6) Copy your files into the ram disk. You can now use the ram disk with any MDOS version you wish.

7) These instructions are for the Horizon 2000+ series Ram Disk and may not work with other versions.

If you need help feel free to call. Thanks to Eric Wicklund, Neville Blair and Gary Bowser for their help. Without them the Ram Disk would be on it's way back to the vendor.



TI SLAVES AND OGDEN TI USERS GROUPS OFFICERS

TI SLAVES		
PRESIDENT	ALEX SCHAFF	487-3704
VICE PRESIDENT	JACK BAXTER	265-2053
SEC/TREAS	STEVE RICHARDSON	250-1573
LIBRAIAN	DAVE BERRY	966-440
ASST.LIB	MIKE BOSEN	250-9240
NEWSLETTER EDITOR FOR BOTH GROUPS		

OGDEN TI USERS GROUP	
LORRAINE WADMAN	825-7690
DAVE MISCHLER	782-1004
HELEN HILBURN	773-0622
LINDA ROBINSON	621-8975
MEL BRAGG	393-9605
MEL BRAGG	393-9605

JANUARY AND FEBRUARY 1993 NEWSLETTER

TI SLAVES

OUR NEXT MEETING IS FEB. 20TH. 1993 AT 9:AM WE MEET AT THE DISABLED AMERICAN VETERANS HALL AT 273 E. 800 SO. PLEASE BE THERE PROMPTLY.!!

OGDEN TI USERS GROUP

OUR NEXT MEETING IS FEB. 6TH AT 9 AM AND FEB. 22ND AT 7 PM. WE MEET AT THE OGDEN MUNICIPALE AIRPORT IN THE FIRST BUILDING JUST EAST OF THE NEW TOWER.

FESTWEST "NORTH" 93 IS HERE. WE NEED EVERY MEMBERS SUPPORT. PLEASE HELP OUT WHERE YOU CAN.

SLAVES AND OTIUG
1396 LINCOLN APT.B
OGDEN, UTAH 84404