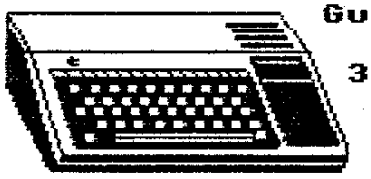
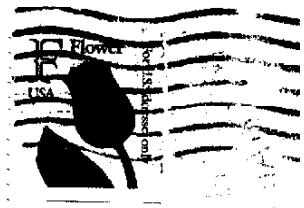
GUILFORD 99'ERS NEWSLETTER

SUPPORTING THE TEXAS INSTRUMENTS TI-99/4A COMPUTER



GUILFORD 99'ERS UG
3202 CANTERBURY DR
GREENSBORD NC
27408



TO:

VOLUME 8 NUMBER 4 APRIL 1991

George von Seth, Pres. (292-2035) Bob Carmany, Newsletter Ed (855-1538)
Tony Kleen, Sec/Treas (924-6344) Bill Woodruff,Pgm/Library (228-1892)
BBS: (919)621-2623 --ROS

The Guilford 99'er Users' Group Newsletter is free to dues paying members (One copy per family, please). Dues are $12.00 per family, per year. Send check to:Tony Kleen c/o 3202 Canterbury Dr., Greensboro, NC 27408. The Software Library is for dues paying members only. (Bob Carmany Ed)

OUR NEXT MEETING

DATE: April 2, 1991 Time: 7:30 PM. Place: Glenwood Recreation Center, 2010 S. Chapman Street.

Program for this meeting will be a demonstration of databases that could be used to catalog our library. Be sure to be there to cast your vote and give us your ideas about which of the databasees you would like to use for our library.

MINUTES

We welcomed Paul Holt and Andrew Small to our monthly meeting. Andrew is considering rejoining the club after an absense of a couple of years. Paul was looking for IBM expertise but did stick around for the entire meeting.

Treasurer's report: The club has $142.91 as of 03/05/91.

Old Business topic: Cataloging the club's disk library! Mac and Bob will choose a database format and bring this format to our next meeting. With each member cataloging a few disks each, we can complete this taks easily and quickly.

New Business topic: What about classes? In FORTH, XB, TI-BASE, Assembly!? Bob Carmany will introduce FORTH on the 19th at 7:30 at George von Seth's home.

This was our monthly meeting set aside for copying library disks. I had promised to bring my 2 drive system (but forgot!). Needless to say, my name was Mister Mud until Bill saved the day bu bringing a standalone drive. Thanks Bill!!

REMEMBER WHEN?

Antiques described by Charles Good Lima Ohio User Group

The "original" TI Home Computer system, released to the public in 1979 and 1980, consisted of the 99/4 computer (without the "A") and a series of stand alone peripherals that plug directly into the side of the 99/4 (or 99/4A) console, or into the side of the previous peripheral (hence the unoffical descriptive term "freight train peripheral"). Each of these freight train peripherals except the speech synthesizer has a base that measures 17x26cm (a bit larger than 6.5x10 inches), a separate power supply rated at 0.2A (23 watts) at 115 volts, and its own separate power cord. I recently purchased a "4" (just to play with) and was later given many of the freight train peripherals. After using these devices for a while I realize how fortunate we are to have the "4A"and its peripheral expansion box.

Components of the "original" TI home computer system are listed below, together with their official TI part numbers and some prices mostly quoted from an ad by CBM INC of Lexington KY published on page 12 of the first edition (May/June 1981) of 99er Magazine. These CBM INC prices are probably below TI's official list price. These peripherals are not the same as those designed to fit in the PE Box. PE box peripherals all have "PHA12xx" part numbers and are described in official TI publications as "cards".

--TI 99/4 console, (PHC0040); $499

--RF (TV) modulator; in 1980 this was an extra cost item, (PHA2100): $41
--Solid State Speech Synthesizer; the same one most of us still use, (PHP1500): $122
--32K RAM memory expanion, (PHP2200): $325
--RS232 Acessories Peripheral, (PHP1700): $183
--Solid State Thermal Printer, (PHP1900): $325
--Disk Drive Controller, (PHP1800): $243
--Disk Memory Drive, (PHP1850): $399
--P Code Peripheral, (PHP 2400): $399.95
--Video Controller, (PHP2300): $699.95

Prices of the last two items are official list prices quoted from TI's suggested retail price list dated June - December 1982 (1049703-1).

You can connect a maximum of three periphrals in series to the right side of the computer. If present, the speech synthesizer has to be first and the 32K second. A "typical" freight train minimum expansion system (99/4 with modulator, 32K, thermal printer, controller and one drive) would be almost four feet wide and cost $1832. Bringing the system up to the maximum of three SSSD drives and buying all the other freight train peripherals would bring the cost up to a total of $4035. Wow! And you can only have simultaneous use of 3 peripherals.

In this article I will describe what I know about these freight train peripherals. I have hands on experience with the Thermal Printer, 32K, and Disk Controller. I will not discuss the Speech Synthesizer since the 1979/80 device is the same one we are all familiar with. In a separate article I will describe my experiences with the 99/4.

--32K EXPANSION MEMORY: This functions exactly like the equivalent PE box card. These days you can, for about $10, buy a 32K RAM chip that measures about 1x3cm and draws very little current. It amazes me that TI's original 32K was so bulky and required a 23 watt power supply. A 12 inch black and white TV only draws 29 watts. But I guess if you compare a 1955 room sized UNIVAC computer in memory, watts of power consumption, and bulk, the vintage 1979 TI 32K looks pretty good.

--RS232: This stand alone box offeres only one RS232 port and no parallel port. The PE Box RS232 card allows connection of TWO RS232 (serial) devices (with a special Y cable) AND one parallel device all to the same card. The PE Box card is obviously superior to the stand alone peripheral.

--DISK DRIVE CONTROLLER: This device used the original DISK MANAGER module (the DMI), and can control up to three SSSD stand alone drives. Double sided is not available with the freight train disk controller. The main difference between the DMI and DMII modules is that the "I" has no provision for double sided disk initialization. A TI stand alone drive plugs directly into the back of the freight train Disk Controller without the need for any special adapter cable other than the cable that comes with the stand alone drive. Other drives plug into the cable of DSK1 using a small adapter board. A special cable that comes with the PE box controller card is needed to plug a stand alone TI drive to the back of the controller card for use as DSK2 or DSK3. An interesting feature of the freight train controller in combination with the DMI module is that they do not recognize the "whole disk protected" byte >10 of sector zero. With the TI PE box controller and the DMII module, if this byte is set for a value of >50 you cannot copy the disk with the DMII.

--THERMAL PRINTER: This is printer device "TP", and was sold to TI users at a time when cheap dot matrix or daisy wheel printers cost $600+. The 1982 list price for the 99/4A's official dot matrix printer was $750. The TP uses 3.5 inch thermal paper, prints 32 characters per line, and like all thermal printers is both quiet and slow. 3.5 inch thermal paper rolls are a non standard size these days. TP users either have to purchase 10 year old official TI paper from one of the few TI dealers that stock this item, or use a paper cutter to trim 8.5 inch FAX paper rolls down to 3.5 inches. Such 8.5 inch wide FAX rolls are commonly available these days from many stores including SEARS, KMART, and WALMART. On the title page of the TP manual it says that the TP "prints a copy of a TI BASIC program or the screen displays from certain Command Modules." And that is about it! A few modules, such as MUSIC MAKER, allow screen dumps with the TP. You can specify output to the TP with the DMI, DMII, PRK, Statistics, LOGO2, and maybe a few other modules. You can't use the TP with TI Writer, Funnelweb, the EA module, or Microsoft Multiplan. From BASIC you can LIST a program to the TP, a common application. You can also OPEN a file to the TP using any of these file attributes: SEQUENTIAL or RELATIVE, DISPLAY or INTERNAL, OUTPUT or APPEND, FIXED or VARIABLE. I can't imagine what use RELATIVE, INTERNAL, or APPEND have in OPENing a printer file. When opened in INTERNAL, the printer prints a meaningless graphic of the internal representation of each ASCII character. The maximum length of a VARIABLE TP attribute is 32. All printed characters of the TP's built in character set are on a 5x7 dot grid. The TP has a unique graphic set for ASCII 0-31 and the usual alpha/numeric characters for ASCII 32-127. Each printed dot of a character is printed only once and individual dots can be seen with the naked eye. There is no way to make extra dense high quality characters. Emphasized, double strike, and "NLQ" is not available. The user can also, using an 8x8 dot grid, redefine ASCII chars 32-159 in BASIC

using CALL CHAR, and then directly print any of these redefined chars to the TP with the appropriate keyboard keypress as in PRINT #1:"&" where & has been redefined, or with PRINT #1:CHR$(xxx). This is a neat trick! It is much harder to print redefined characters with other kinds of dot matrix printers.

--VIDEO CONTROLLER: A photograph and brief description of this peripheral appears as part of an article on page 53 of Volume 1, No. 4 of 99er Magazine (Nov/Dec 1981). The photograph shows a box identical to that of the stand alone 32K or disk controller, with a cable coming out of the right side where the "pass through" expansion bus is found on other stand alone peripherals. The article describes the video controller as allowing "computer controlled interactive video with VCR's and Video Disk Players", whatever that means. As evidenced by the videos we created from the formal preentations at the 1990 Lima MUG Conference, it is possible without this device to mix human voice, computer audio and video outout, and video camera footage on the same video tape. Such mixing of various audio and video sources was done by us manually however, not under computer control. An extra cost cable ($99.95 for each of the three available cables in the June - December 1982 TI price list) is needed to interface the video controller to a Sony or Panasonic VCR or a Pioneer video disk player. I really don't understand the need for computer control of a video disk player. If I remember correctly, 1980 video disks resembled phonograph records in that you could only PLAY them from the beginning, not record onto them.

--P CODE PERIPHERAL: My June - December 1982 TI price list states that this device is "available only until replaced by peripheral card", with such a card "available in second quarter 1982." The freight train P Code peripheral is apparently exactly equivalent to the PE Box P Code card.

There you have it folks, the original TI Home Computer expansion "system". Now you know why the expansion port on the 99/4 and 99/4A is on the SIDE of the console, rather than on the back where it should have been placed. You can only use three of these freight train peripherals at once, and they take up huge amounts of desktop space. Arn't you glad we now have the PE Box!

I want to acknowledge the generous gift of Mr. E.T. Breer of the St. Louis Missouri User Group who gave me several of the freight train peripherals described in this review.

This article/item comes from the January 1991 issue of BITS, BYTES PIXELS (Charles Good, editor), the newsletter of the Lima OH 99/4A User Group, P.O. Box 647, Venedocia, OH 45894.

NOTES

Despite the discussion, it appears that there isn't enough interest to start formal classes in FORTH --or anything else for that matter! Unless there are some definite committments to attend at the next meeting, consider all of the classes cancelled until further notice. The fact is, you can't have classes unless somebody shows up!

==========

TI-Base Topic - VDPram **/I Paging.
by Tony Kleen, Guilford TI99er Users Grp

==========

==========

Last month, I discussed **/C paging, which I defined as a process of loading / unloading **/C files into/from the VDPram area. The definition for **/I paging is very similar: the process of loading/unloading **/I files into/from the VDPram area.

The only difference in the above two definitions is in what type of file is used, either a **/C or a **/I file. BUT .. this difference has ramifications. When you load a **/C file, you are just partially loading the VDPram. When you load a **/I file, the ENTIRE VDPram is loaded. With our **/C paging, the loading was controlled by a 'master' **/C file that permanently resided in VDPram. This 'master' command file INSTALL ADDed and INSTALL REMOVEd other **/C files. Every time we page in a **/I file, the entire contents of VDPram is overlaid with the new **/I file that is paged. As such, we can no longer have a 'master' command file permanently resident in VDPram to control our paging. SO .. Not only are the files different (**/C vs **/I), but now our process for loading and unloading must be different.

Let's review some of what we already know. Keep in mind that I am referring to the TIBase V3.0, or newer software.

The **/I files are created by using TIBase's 'install capability'. When one previously ADDs **/C files to the VDPram, and then SAVEs this VDPram area to disk, a **/I file is written to disk; the file being an image of VDPram. An image (or page) of VDPram previously SAVEd to disk can be reLOADed to VDPram.

What have we determined? Two directives ie. INSTALL LOAD and INSTALL SAVE, can be used to fulfill our 'paging' process's requirement; that is, the loading and unloading of 'pages'. How we do the paging, using these two directives has yet to be determined, though!

I figure the best way may simply be to start where I did initially and keep building on our base knowledge. I'll demonstrate all the requirements and explain their reasons for existence. Here goes..

..... The basics .....

First off, here's the three command files that we'll be working with.

```
* ---------------------------------- *
* PROC1/C v01-Basic                  *
* ---------------------------------- *
SET TALK ON
SET RECNUM OFF
DISPLAY "DIRECTIVE 3 OF 4 FOR PROC1."
DISPLAY "DIRECTIVE 4 OF 4 FOR PROC1."
* ---------------------------------- *


* ---------------------------------- *
* PROC2/C v01-Basic                  *
* ---------------------------------- *
DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2."
DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2."
* ---------------------------------- *


* ---------------------------------- *
* PROC3/C v01-Basic                  *
* ---------------------------------- *
DISPLAY "DIRECTIVE 1 OF 2 FOR PROC3."
DISPLAY "DIRECTIVE 2 OF 2 FOR PROC3."
* ---------------------------------- *
```

Now, I'm going to create the **/C command files that will add our **/C files to VDPram, and then save the VDPram image to a **/I file (what I am considering as our VDPram page).

```
* ---------------------------------- *
* PROC4/C v01-Create PAGE1/I file.   *
* ---------------------------------- *
INSTALL CLEAR
INSTALL ADD PROC1
INSTALL SAVE PAGE1
* ---------------------------------- *


* ---------------------------------- *
* PROC5/C v01-Create PAGE2/I file.   *
* ---------------------------------- *
INSTALL CLEAR
INSTALL ADD PROC1
INSTALL SAVE PAGE2
* ---------------------------------- *


* ---------------------------------- *
* PROC6/C v01-Create PAGE3/I file.   *
* ---------------------------------- *
INSTALL CLEAR
INSTALL ADD PROC1
INSTALL SAVE PAGE3
* ---------------------------------- *
```

Let's execute the PROC4. At the .DOT prompt, enter the DO PROC4. Your screen should look like the following:

```
.DO PROC4                                   008
001 * ----------------------------------
002 * PROC4/C v01-Create PAGE1/I file.
003 * ----------------------------------
004 INSTALL CLEAR
install area clear 2546 bytes available
005 INSTALL ADD PROC1
install complete, 2427 bytes available
006 INSTALL SAVE PAGE1
007 * ----------------------------------
```

Likewise, executing PROC5 and PROC6 would produce similar results. So, after entering the six **/C files, and executing PROC4, PROC5, and PROC6; we now have created three **/I files (PAGE1/I, PAGE2/I, and PAGE3/I).

Keep in mind that PAGE1/I is nothing more than an image of PROC1/C as it resides in VDPram. Likewise, the same similarities exist between PROC2/C and PAGE2/I; and between PROC3/C and PAGE3/I.

Now then, it's easy enough to get the first PAGE1 into VDPram. Simply enter the following at the .DOT prompt:

```
.INSTALL LOAD PAGE1
.DO PROC1
```

The question is: How are we going to get PROC1 to 'page' in the next PAGE2 and execute PROC2? Well, let's make a modification to PROC1, as follows:

```
* ---------------------------------- *
* PROC1/C v01-Basic                  *
*         v02-LOAD & DO              *
* ---------------------------------- *
SET TALK ON
SET RECNUM OFF
DISPLAY "DIRECTIVE 3 OF 4 FOR PROC1."
DISPLAY "DIRECTIVE 4 OF 4 FOR PROC1."
INSTALL LOAD PAGE2
DO PROC2
* ---------------------------------- *
```

Looks like this ought to do it. WRONG. The problem is: When we INSTALL LOAD PAGE2, we overlay what previously exist-

ed in VDPram, namely PROC1, and therefore the last directive 'DO PROC2'. We never reach the command telling TIBase to go DO PROC2! AND, we can't place the DO PROC2 prior to the INSTALL LOAD PAGE2 because PROC2 hasn't yet been loaded to VDPram!

We could try a different approach. Instead of trying to execute a command file with a different name, why not use the same names for the various PAGEs? That way, when we 'page' in the next page, we don't have to issue a DO PROC1; we'll just continue DOing a COMMON procedure. This would require changes to the three command files that create the **/I files, as follows:

```
* ---------------------------------- *
* PROC4/C v01-Create PAGE1/I file.   *
*         v02-Common file name       *
* ---------------------------------- *
INSTALL CLEAR
COPY PROC1/C COMMON/C GO
INSTALL ADD COMMON
INSTALL SAVE PAGE1
* ---------------------------------- *


* ---------------------------------- *
* PROC5/C v01-Create PAGE2/I file.   *
*         v02-Common file name       *
* ---------------------------------- *
INSTALL CLEAR
COPY PROC2/C COMMON/C GO
INSTALL ADD COMMON
INSTALL SAVE PAGE2
* ---------------------------------- *


* ---------------------------------- *
* PROC6/C v01-Create PAGE3/I file.   *
*         v02-Common file name       *
* ---------------------------------- *
INSTALL CLEAR
COPY PROC3/C COMMON/C GO
INSTALL ADD COMMON
INSTALL SAVE PAGE3
* ---------------------------------- *
```

While we're at it, let's revise those first three **/C files to INSALL LOAD the next PAGE (**/I) file. PAGE1 will load PAGE2, PAGE2 will load PAGE3, and then we'll loop back to PAGE1, because PAGE3 will load PAGE1. Here's what they will look like.

```
* ---------------------------------- *
* PROC1/C v01-Basic                  *
*         v02-LOAD & DO              *
*         v03-LOAD PAGE2, only.      *
* ---------------------------------- *
SET TALK ON
SET RECNUM OFF
DISPLAY "DIRECTIVE 3 OF 4 FOR PROC1."
DISPLAY "DIRECTIVE 4 OF 4 FOR PROC1."
INSTALL LOAD PAGE2
* ---------------------------------- *


* ---------------------------------- *
* PROC2/C v01-Basic                  *
*         v02-LOAD PAGE3.            *
* ---------------------------------- *
DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2."
DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2."
INSTALL LOAD PAGE3
* ---------------------------------- *


* ---------------------------------- *
* PROC3/C v01-Basic                  *
*         v02-LOAD PAGE1             *
* ---------------------------------- *
DISPLAY "DIRECTIVE 1 OF 2 FOR PROC3."
DISPLAY "DIRECTIVE 2 OF 2 FOR PROC3."
INSTALL LOAD PAGE1
* ---------------------------------- *
```

At this point, we again need to DO PROC4,DO PROC5, and DO PROC6 to recreate and reSAVE our PAGEs, ie the **/I files.

Now then, let's set TRACE ON and execute our first PAGE. At the .DOT prompt, enter the following two lines:

```
.INSTALL LOAD PAGE1
.DO COMMON
001 SET TALK ON
002 SET RECNUM OFF
003 DISPLAY "DIRECTIVE 3 OF 4 FOR PROC1.
004 DISPLAY "DIRECTIVE 4 OF 4 FOR PROC1.
005 INSTALL LOAD PAGE2
006 DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2.
007 DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2.
008 INSTALL LOAD PAGE3
009 DISPLAY "DIRECTIVE 1 OF 2 FOR PROC3.
010 DISPLAY "DIRECTIVE 2 OF 2 FOR PROC3.
011 INSTALL LOAD PAGE1
012 SET TALK ON
013 SET RECNUM OFF
014 DISPLAY "DIRECTIVE 3 OF 4 FOR PROC1.
015 DISPLAY "DIRECTIVE 4 OF 4 FOR PROC1.
016 INSTALL LOAD PAGE2
017 DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2.
018 DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2.
019 INSTALL LOAD PAGE3
020 DISPLAY "DIRECTIVE 1 OF 2 FOR PROC3.
021 DISPLAY "DIRECTIVE 2 OF 2 FOR PROC3.
022 INSTALL LOAD PAGE1
023 SET TALK ON
024 SET RECNUM OFF
025 DISPLAY "DIRECTIVE 3 OF 4 FOR PR
.
.
```

Notice that the first five trace lines are from PROC1 (PAGE1), the next three trace lines are from PROC2 (PAGE2), the next three lines are from PROC3 (PAGE3), the next five lines are from PROC1 (PAGE1), etc.., adinfinitim.. literally! Notice that the trace lines keep incrementing, without regard to which PROC*/C (or PAGE*/I) is in VDPram. When the trace numbering reaches 999, it simply rolls over to 000.

We've got one minor problem with a paging scenario such as this; how the dickens do we stop it? You have to use fctn-9! Not very clean, is it?

Let's make sure we know what we've got here! We have determined a way in which we can get the current **/C file that is in VDPram to load the VDPram with the next **/C, and begin executing that next **/C file. I'll reiterate the steps that had to be completed to get us this far.

First, create your **/C files with the last directive INSTALL LOADing your next VDPram page, ie., the **/I file.

Second, you must use a COMMON name for the command file being paged. I used COMMON/C! Your **/I files (pages) must correspond to how your command files reference the **/I files. Look at my above correlation between PROC1/C and PAGE1/I; between PROC2/C and PAGE2/I; and between PROC3/C and PAGE3/I.

Third, execute the **/C files that save your **/I, ie. the **/C files you created in step two.

Fourth, at the .DOT prompt, INSTALL LOAD your first **/I (the one corresponding to your first **/C file) and DO COMMON.

Fifth, hit function-9 when you want to stop the processing!

Now we're done with the basics, and are now able to 'page' **/I files. I find

it somewhat unacceptable using fctn-9 to stop the process, though. Also, this scenario of an uncontrolled, infinite loop is somewhat distasteful to me; I like the menu-driven systems (see ART004 in February's newsletter). SO .. let's go to the next step, that of using a 'menu driver' as our first **/C file.

.....Menu-driver paging.....

I'm going to redo the PROC1/C to DO PROC2 if a '2' is entered, and to DO PROC3 if a '3' is entered. PROC2/C will be revised to return to PROC1 upon completion. PROC3 doesn't have to be revised, as it already returns to PROC1 upon completion.

```
* ---------------------------------- *
* PROC1/C v01-Basic                  *
*         v02-LOAD & DO              *
*         v03-LOAD PAGE2, only.      *
*         v04-MENU DRIVER, PAGE2,3.  *
* ---------------------------------- *
SET TALK OFF
SET RECNUM OFF
LOCAL ? C 1
WHILE ?<>"E"
  CLEAR
  WRITE 3,1 " 2 - PROC2 PROCESS"
  WRITE 5,1 " 3 - PROC3 PROCESS"
  WRITE 7,1 " E - EXIT  PROCESS"
  WRITE 9,1 "   - selection"
  READCHAR 9,2 ?
  DOCASE
    CASE ?="2"
      INSTALL LOAD PAGE2
    CASE ?="3"
      INSTALL LOAD PAGE3
  ENDCASE
ENDWHILE
* ---------------------------------- *

* ---------------------------------- *
* PROC2/C v01-Basic                  *
*         v02-LOAD PAGE3.            *
*         v03-LOAD PAGE1.            *
* ---------------------------------- *
DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2."
DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2."
INSTALL LOAD PAGE1
* ---------------------------------- *
```

Next, DO PROC4 (at the .DOT prompt) to reSAVE the PAGE1/I file; and DO PROC5 to reSAVE the PAGE2/I file. Now, let's test our work. At the .DOT prompt:

```
.INSTALL LOAD PAGE1
.DO COMMON
```

The following screen displays:

```
:------------------------------------:
: 2 - PROC2 PROCESS                  :
:                                    :
: 3 - PROC3 PROCESS                  :
:                                    :
: E - EXIT  PROCESS                  :
:                                    :
:   - selection                      :
:------------------------------------:
```

Enter an 'E' for your selection; and we're returned to the .DOT prompt. That works fine. Now, if you test just the '2' entry, or the '3' entry, you'll find that they work fine, too.

Now, if you remember back to the 'basic example'; where PAGE1 install loaded PAGE2, which loaded PAGE3, which reloaded PAGE1 to start all over again; we were able to cycle through 'ad infinitum'. With our new 'menu driver' PROC1, we'll need to be able to cycle through PAGE2 and PAGE3 any number of times, also. Let's test that scenario, by continueing to enter a value '2' every time we're presented the 'menu'. Enter at the .DOT prompt:

```
.INSTALL LOAD PAGE1
.DO COMMON
```

Our trace results are shortened, as follows:

```
001 SET TALK OFF
002 SET RECNUM OFF
003 LOCAL ? C 1
004 WHILE ?<>"E"
005 CLEAR
006 WRITE 3,1 " 2 - PROC2 PROCESS"
007 WRITE 5,1 " 3 - PROC3 PROCESS"
008 WRITE 7,1 " E - EXIT  PROCESS"
009 WRITE 9,1 "   - selection"
010 READCHAR 9,2 ?
011 DOCASE
013 INSTALL LOAD PAGE2
014 DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2.
015 DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2.
016 INSTALL LOAD PAGE1
 .
 .
144 INSTALL LOAD PAGE1
145 SET TALK OFF
146 SET RECNUM OFF
147 LOCAL ? C 1
148 WHILE ?<>"E"
149 CLEAR
150 WRITE 3,1 " 2 - PROC2 PROCESS"
151 WRITE 5,1 " 3 - PROC3 PROCESS"
152 WRITE 7,1 " E - EXIT  PROCESS"
153 WRITE 9,1 "   - selection"
154 READCHAR 9,2 ?
155 DOCASE
too many if, while, or case levels
157 INSTALL LOAD PAGE2
158 DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2.
159 DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2.
160 INSTALL LOAD PAGE1
161 SET TALK OFF
162 SET RECNUM OFF
163 LOCAL ? C 1
164 WHILE ?<>"E"
too many if, while, or case levels
165 CLEAR
166 WRITE 3,1 " 2 - PROC2 PROCESS"
167 WRITE 5,1 " 3 - PROC3 PROCESS"
168 WRITE 7,1 " E - EXIT  PROCESS"
169 WRITE 9,1 "   - selection"
170 READCHAR 9,2 ?
171 DOCASE
too many if, while, or case levels
172 CASE ?="2"
176 ENDCASE
177 ENDWHILE
```

In the above example, I show the first time through PAGE1 and Page2. Then I ignore the trace until the tenth time through. Notice the UT-OH's at trace lines 155, 164, and 171. We've got too many levels! How can that be? We've constructed our DOCASE/ENDCASE properly. WELL .. notice the trace never shows and ENDCASE being referenced. Once again, we overlay VDPram with the next page prior to ever executing the subsequent directive, in this case an ENDCASE.

REMEDY? Place an ENDCASE in PROC1 where we know it'll get processed immediately after we reLOAD with PAGE1. Below is our revised version of PROC1/C. Remember to DO PROC4 to reSAVE PAGE1 to disk.

```
* ---------------------------------- *
* PROC1/C v01-Basic                  *
*         v02-LOAD & DO              *
*         v03-LOAD PAGE2, only.      *
*         v04-MENU DRIVER            *
*         v05-ENDCASE repositioned.  *
```

```
* ---------------------------------- *
ENDCASE
SET TALK OFF
SET RECNUM OFF
LOCAL ? C 1
WHILE ?<>"E"
  CLEAR
  WRITE 3,1 " 2 - PROC2 PROCESS"
  WRITE 5,1 " 3 - PROC3 PROCESS"
  WRITE 7,1 " E - EXIT  PROCESS"
  WRITE 9,1 "   - selection"
  READCHAR 9,2 ?
  DOCASE
    CASE ?="2"
      INSTALL LOAD PAGE2
    CASE ?="3"
      INSTALL LOAD PAGE3
  ENDCASE
ENDWHILE
* ---------------------------------- *
```

Now, the question is; what will TIBase do with the ENDCASE the first time into the PROC1. We've got an improper DOCASE /ENDCASE construct the first time thru, but with our paging, it's proper for the second through 'n' times. Again, here's the trace after entering the INSTALL LOAD PAGE1, and DO COMMON at the .DOT prompt.

```
001 ENDCASE
002 SET TALK OFF
003 SET RECNUM OFF
004 LOCAL ? C 1
005 WHILE ?<>"E"
006 CLEAR
007 WRITE 3,1 " 2 - PROC2 PROCESS"
008 WRITE 5,1 " 3 - PROC3 PROCESS"
009 WRITE 7,1 " E - EXIT  PROCESS"
010 WRITE 9,1 "   - selection"
011 READCHAR 9,2 ?
012 DOCASE
014 INSTALL LOAD PAGE2
015 DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2.
016 DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2.
017 INSTALL LOAD PAGE1
018 ENDCASE
019 SET TALK OFF
020 SET RECNUM OFF
021 LOCAL ? C 1
022 WHILE ?<>"E"
023 CLEAR
024 WRITE 3,1 " 2 - PROC2 PROCESS"
025 WRITE 5,1 " 3 - PROC3 PROCESS"
026 WRITE 7,1 " E - EXIT  PROCESS"
027 WRITE 9,1 "   - selection"
028 READCHAR 9,2 ?
029 DOCASE
031 INSTALL LOAD PAGE2
032 DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2.
033 DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2.
034 INSTALL LOAD PAGE1
035 ENDCASE
036 SET TALK OFF
037 SET RECNUM OFF
038 LOCAL ? C 1
039 WHILE ?<>"E"
040 CLEAR
041 WRITE 3,1 " 2 - PROC2 PROCESS"
042 WRITE 5,1 " 3 - PROC3 PROCESS"
043 WRITE 7,1 " E - EXIT  PROCESS"
044 WRITE 9,1 "   - selection"
045 READCHAR 9,2 ?
046 DOCASE
052 ENDWHILE
```

Hmmm. We've still got a problem. Look at trace line 018. Even though we have an ENDCASE, we still never reach the ENDWHILE! After 10 repetitions through, we'll get the UT-OH mentioned earlier. REMEDY .. Again, place the ending construct (ENDWHILE in this case) at the front of PROC1 to assure its being executed the second time through paging. Here are the results; after modifying PROC1 and reSAVEing PAGE1).

```
* ---------------------------------- *
* PROC1/C v01-Basic                  *
*         v02-LOAD & DO              *
*         v03-LOAD PAGE2, only.      *
*         v04-MENU DRIVER            *
*         v05-ENDCASE repositioned.  *
*         v06-ENDWHILE repositioned. *
* ---------------------------------- *
ENDCASE
ENDWHILE
SET TALK OFF
SET RECNUM OFF
LOCAL ? C 1
WHILE ?<>"E"
  CLEAR
  WRITE 3,1 " 2 - PROC2 PROCESS"
  WRITE 5,1 " 3 - PROC3 PROCESS"
  WRITE 7,1 " E - EXIT  PROCESS"
  WRITE 9,1 "   - selection"
  READCHAR 9,2 ?
  DOCASE
    CASE ?="2"
      INSTALL LOAD PAGE2
    CASE ?="3"
      INSTALL LOAD PAGE3
  ENDCASE
ENDWHILE
```

```
* ----------------------------------

.INSTALL LOAD PAGE1
.DO COMMON
001 ENDCASE
002 ENDWHILE
003 SET TALK OFF
004 SET RECNUM OFF
005 LOCAL ? C 1
006 WHILE ?<>"E"
007 CLEAR
008 WRITE 3,1 " 2 - PROC2 PROCESS"
009 WRITE 5,1 " 3 - PROC3 PROCESS"
010 WRITE 7,1 " E - EXIT  PROCESS"
011 WRITE 9,1 "   - selection"
012 READCHAR 9,2 ?
013 DOCASE
015 INSTALL LOAD PAGE2
016 DISPLAY "DIRECTIVE 1 OF 2 FOR PROC2.
017 DISPLAY "DIRECTIVE 2 OF 2 FOR PROC2.
018 INSTALL LOAD PAGE1
019 ENDCASE
020 ENDWHILE
006 WHILE ?<>"E"
007 CLEAR
008 WRITE 3,1 " 2 - PROC2 PROCESS"
009 WRITE 5,1 " 3 - PROC3 PROCESS"
010 WRITE 7,1 " E - EXIT  PROCESS"
011 WRITE 9,1 "   - selection"
012 READCHAR 9,2 ?
013 DOCASE
019 ENDWHILE
006 WHILE ?<>"E"
```

Here's a bonus! Look at line 019 and 020 where we ENDCASE and ENDWHILE. The next line of PROC1 to be executed is line 006; the beginning of the WHILE consruct. The statements between the ENDWHILE and WHILE directives of PROC1 are only executed the first time thru. If we've got any initialization to be done; such as USEing an input database or setting up our printer; we could place those initialization directives after the ENDWHILE and before the WHILE!

.....What have we done?.....

Well folks, we have successfully used **/I paging. BUT.. not very effectively. All that I've shown you this session is how to set up one 'master' page that calls one of two subordinate pages. Next month, I'll conclude this entire 'paging' topic by showing you how to page any number of command files.

```
100 !//////////////////
110 !/ AUTO SPRITEDEF /
120 !//////////////////
130 !
140 !<<MERGE DATA BEFORE>>
150 DATA 0005020701316BBFDFB
70F1B0202050000000800018ACF
AF6C2E0B080804000

160 !<<OR ON LINE 150>>
170 !///INITIALIZE///
180 OPTION BASE 1 :: DIM SP$
(16),B(8,8),SP(16,16):: T=20
 :: K,C=100 :: LX,LY=5 :: CA
LL INIT :: CALL LOAD(-31878,
1)
190 CALL CHAR(100,RPT$("0",1
6)&RPT$("F",16)&"007E4242424
27E00FF81BDBDBDBDB81FF")
200 HEX$="0123456789ABCDEF"
:: CALL COLOR(9,2,16)

210 !///SCREEN///
220 CALL MAGNIFY(4):: CALL S
CREEN(12):: CALL CLEAR :: CA
LL SPRITE(#1,96,9,10,10,0,4)
230 DISPLAY AT(1,4):"AUTO SP
RITE DEFINITION": : :"  1234
567812345678"
240 FOR I=0 TO 1 :: FOR I1=1
 TO 8 :: CALL VCHAR(4+I1+8*I
,4,48+I1):: NEXT I1 :: NEXT
I
250 FOR I=5 TO 20 :: CALL HC
HAR(I,5,K,16):: NEXT I :: C=
K :: CALL VCHAR(LX,LY,C+2)
260 CALL SCREEN(12):: DISPLA
Y AT(5,2):"PRESS " :: DISPL
AY AT(7,T):"1 WHITE" :: DISP
LAY AT(9,T):"2 BLACK" :: DIS
PLAY AT(11,T):"3 OPTIONS"
265 DISPLAY AT(13,T):"4 STOP
" :: DISPLAY AT(15,T):"5 DAT
A" :: DISPLAY AT(17,T):"R SA
VE"

270 !//KEY INPUT//
280 CALL KEY(1,K,S):: IF S=0
 THEN 280 ELSE X,Y=0 :: IF K
=5 THEN X=-1 :: GOTO 620 ELS
E IF K+1=1 THEN X=1 :: GOTO
620 ELSE IF K=2 THEN Y=-1 ::
 GOTO 620
290 IF K=3 THEN Y=1 :: GOTO
620 ELSE IF K=4 THEN X,Y=-1
:: GOTO 620 ELSE IF K=14 THE
N X,Y=1 :: GOTO 620 ELSE IF
K=15 THEN X=1 :: Y=-1 :: GOT
O 620
300 IF K=6 THEN X=-1 :: Y=1
:: GOTO 620 ELSE IF K<>18 TH
EN 320 ELSE CALL SOUND(100,1
10,0,-6,0):: IF C=101 THEN C
=100 ELSE C=101
310 CALL VCHAR(LX,LY,C+2)::
GOTO 280
320 CALL SOUND(100,660,0)::
IF K=19 THEN K=100 :: GOTO 2
50 ELSE IF K=16 THEN 660 ELS
E IF K=7 THEN K=101 :: GOTO
250
330 IF K=9 THEN 1050 ELSE IF
 K=8 THEN 350 ELSE IF K<>10
THEN 280 ELSE GOSUB 380 :: G
OTO 570

340 !///OPTIONS///
350 CALL SCREEN(10):: DISPLA
Y AT(7,T):"1 VMIRROR" :: DIS
PLAY AT(9,T):"2 HMIRROR" ::
DISPLAY AT(11,T):"3 REVERSE"
 :: DISPLAY AT(13,T):"4 ROTA
TE" :: DISPLAY AT(15,T):"5 S
COOT" :: DISPLAY AT(17,T):"&
 RETURN"
360 CALL KEY(0,K,S):: IF S=0
 OR(K<49 OR K>54)THEN 360 EL
SE CALL SOUND(100,330,0):: H
=K-48 :: CALL VCHAR(LX,LY,C)
:: CALL VCHAR(H#2+5,23,42)
370 IF H=6 THEN 570 ELSE IF
H=3 THEN 590 ELSE IF H<>5 TH
EN 500

380 !//SCOOT//
390 CALL SCREEN(6):: DISPLAY
 AT(7,T):"1 UP" :: DISPLAY A
T(9,T):"2 DOWN" :: DISPLAY A
T(11,T):"3 LEFT" :: DISPLAY
AT(13,T):"4 RIGHT" :: DISPLA
Y AT(15,T): :: DISPLAY AT(17
,T):
400 CALL KEY(0,K,S):: IF S=0
 OR(K<49 OR K>52)THEN 400 EL
SE CALL VCHAR((K-49)*2+7,23,
42)
410 !/UP,DOWN/
420 CALL SOUND(100,110,0)::
IF K>50 THEN 460 ELSE IF K=5
0 THEN K=20 :: S=-1 ELSE K=5
 :: S=1
430 FOR I=K TO S*15+K STEP S
 :: FOR J=5 TO 20 :: CALL GC
HAR(I+S,J,H):: IF H<>100 AND
 H<>101 THEN H=100
440 CALL VCHAR(I,J,H):: NEXT
 J :: NEXT I :: GOTO 570

450 !/RIGHT,LEFT/
460 IF K=51 THEN K=5 :: S=1
ELSE K=20 :: S=-1
470 FOR J=K TO S*15+K STEP S
 :: FOR I=5 TO 20 :: CALL GC
HAR(I,J+S,H):: IF H<>100 AND
 H<>101 THEN H=100
480 CALL VCHAR(I,J,H):: NEXT
 I :: NEXT J :: GOTO 570

490 !//GET PICTURE//
500 FOR I=1 TO 16 :: FOR J=1
 TO 16 :: CALL GCHAR(I+4,J+4
,SP(I,J)):: NEXT J :: CALL V
CHAR(I+4,21,42):: NEXT I ::
CALL VCHAR(5,21,32,16)

510 !//MIRRORS//
520 IF H=1 THEN K=4 :: S=-21
 ELSE IF H=2 THEN K=-21 :: S
=4 ELSE 550
530 FOR I=1 TO 16 :: FOR J=1
 TO 16 :: CALL VCHAR(ABS(K+I
),ABS(S+J),SP(I,J)):: NEXT J
 :: NEXT I :: GOTO 570

540 !//ROTATE//
550 FOR J=20 TO 5 STEP -1 ::
 FOR I=5 TO 20 :: CALL VCHAR
(I,J,SP(21-J,I-4)):: NEXT I
:: NEXT J

560 !//PUT BACK CURSOR//
570 CALL GCHAR(LX,LY,C):: CA
LL VCHAR(LX,LY,C+2):: GOTO 2
60

580 !//REVERSE//
590 FOR I=1 TO 16 :: FOR J=1
 TO 16 :: CALL GCHAR(I+4,J+4
,H):: IF H=100 THEN H=101 EL
SE H=100
```

```
600 CALL VCHAR(I+4,J+4,H)::
NEXT J :: NEXT I :: GOTO 570

610 !///MOVE CURSOR//
620 IF LX+X=21 THEN X=-15 EL
SE IF LX+X=4 THEN X=15
630 IF LY+Y=21 THEN Y=-15 EL
SE IF LY+Y=4 THEN Y=15
640 CALL VCHAR(LX,LY,C):: LX
=LX+X :: LY=LY+Y :: CALL VCH
AR(LX,LY,C+2):: GOTO 280

650 !///GET SPRITEDEF///
660 CALL VCHAR(LX,LY,C):: CA
LL SOUND(100,128,0):: CALL S
OUND(300,912,0)
670 FOR Q=5 TO 13 STEP 8 ::
FOR W=5 TO 13 STEP 8 :: FOR
E=0 TO 7 :: FOR R=0 TO 7 ::
CALL GCHAR(W+E,Q+R,CH):: B(E
+1,R+1)=CH-100
680 NEXT R :: NEXT E :: GOSU
B 800 :: DISPLAY AT(22,1):M$
 :: NEXT W :: NEXT Q

690 !//RECORD TO DISK?//
700 CALL CHAR(96,M$):: DISPL
AY AT(7,T)BEEP:"R RECORD" ::
 FOR I=9 TO 17 STEP 2 :: DIS
PLAY AT(I,T): :: NEXT I
710 A$=""
720 CALL KEY(0,KEY,ST):: IF
ST=0 THEN 720 ELSE IF KEY<>8
2 THEN 770
730 DISPLAY AT(17,T):" NAME?
" :: DISPLAY AT(19,T):"LINE"
 :: ACCEPT AT(18,T)BEEP:NA$
:: ACCEPT AT(19,25)BEEP VALI
DATE(DIGIT)SIZE(3):LI :: IF
LI>254 THEN 730

740 !//SAVE AS MERGE//
750 OPEN #1:"DSK1."&NA$,VARI
ABLE 163 :: PRINT #1:CHR$(0)
&CHR$(LI)&CHR$(147)&CHR$(200
)&CHR$(LEN(M$))&M$&CHR$(0)::
 PRINT #1:CHR$(255)&CHR$(255
):: CLOSE #1

760 !///IMPACT PRINTER?//
770 DISPLAY AT(7,T):"PRINT O
N" :: DISPLAY AT(9,T):"PRINT
ER?N" :: ACCEPT AT(9,T+8)BEE
P SIZE(-1):A$ :: IF A$="N" T
HEN M$="" :: GOTO 570
780 GOSUB 950 :: M$="" :: GO
TO 570

790 !///HEX SUBR//
800 FOR R=1 TO 8
810 LOW=B(R,5)*8+B(R,6)*4+B(
R,7)*2+B(R,8)+1
820 HIGH=B(R,1)*8+B(R,2)*4+B
(R,3)*2+B(R,4)+1
830 M$=M$&SEG$(HEX$,HIGH,1)&
SEG$(HEX$,LOW,1)
840 NEXT R :: RETURN

850 !///FIGURE FROM DATA///
860 DATA 0,0,0,0,0,0,0,1,0,0
,1,0,0,0,1,1,0,1,0,0,0,1,0,1
,0,1,1,0,0,1,1,1,1,0,0,0
870 DATA 1,0,0,1,1,0,1,0,1,0
,1,1,1,1,0,0,1,1,0,1,1,1,1,0
,1,1,1,1
880 RESTORE 860 :: FOR Q=1 T
O 16 :: READ W,E,R,K :: SP$(
Q)=CHR$(W+100)&CHR$(E+100)&C
HR$(R+100)&CHR$(K+100):: NEX
T Q
890 RESTORE :: READ M$ :: DI
SPLAY AT(22,1):M$
900 FOR R=3 TO 11 STEP 8 ::
FOR Q=5 TO 20 :: FOR W=0 TO
7 STEP 4
910 A$=SP$(POS(HEX$,SEG$(M$,
1,1),1)):: M$=SEG$(M$,2,LEN(
M$)-1):: DISPLAY AT(Q,R+W)SI
ZE(4):A$
920 NEXT W :: NEXT Q :: NEXT
 R :: RETURN

930 !///PRINT TO PRINTER//
940 !***USE "RS232.CR" IF YO
UR PRINTER IS NOT SET TO GRA
PHICS MODE***
950 OPEN #1:"RS232.CR.DA=8.B
A=4800"
960 PRINT #1:CHR$(27);"A";CH
R$(3)
970 FOR R=5 TO 20 :: PRINT #
1:CHR$(10);CHR$(27);"K";CHR$
(49);CHR$(0);CHR$(7);:: FOR
Q=5 TO 20 :: CALL GCHAR(R,Q,
E):: IF E=101 THEN 990
980 PRINT #1:CHR$(4);CHR$(4)
;CHR$(7);:: GOTO 1000
990 PRINT #1:CHR$(7);CHR$(7)
;CHR$(7);
1000 NEXT Q :: IF R=9 THEN P
RINT #1:" NAME : ";NA$ ELSE
IF R=16 THEN PRINT #1:" LINE
 :";LI
1010 NEXT R :: PRINT #1:CHR$
(10);CHR$(27);"K";CHR$(49);C
HR$(0);
1020 FOR R=1 TO 49 :: PRINT
#1:CHR$(4);:: NEXT R
1030 PRINT #1:CHR$(27);"0";C
HR$(10);"HEX : ";M$;CHR$(13)
1040 CLOSE #1 :: RETURN
1050 CALL DELSPRITE(#1):: CA
LL CLEAR :: CALL CHARSET ::
CALL SCREEN(8):: RUN "DSK1.L
OAD"
```